Towards Development Platforms for Digital Twins: A Model-Driven Low-Code Approach

Judith Michael[1(✉)] and Andreas Wortmann[2]

[1] Software Engineering, RWTH Aachen University, Aachen, Germany
michael@se-rwth.de
[2] Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Stuttgart, Germany
andreas.wortmann@isw.uni-stuttgart.de

Abstract. Digital Twins in smart manufacturing must be highly adaptable for different challenges, environments, and system states. In practice, there is a need for enabling the configuration of Digital Twins by domain experts. Low-code approaches seem to be a meaningful solution for configuration purposes but often lack extension options. We propose a model-driven low-code approach for the configuration and reconfiguration of Digital Twins using language plugins. This approach uses model-driven software engineering and software language engineering methods to derive a configurable digital twin implementation. Moreover, we discuss some remaining challenges such as interoperability, language modularity, evolution, integration of assistive services, collaborative development, and web-based debugging.

Keywords: Digital Twin · Low-code platform · Model-driven software engineering · Software language engineering

1 Motivation

Digital Twins provide means to monitor and control Cyber-Physical Systems (CPSs) in various domains, such as smart manufacturing [29], biology [19], or autonomous driving [8]. They serve different purposes, such as analysis [23], control [31], or behavior prediction [21]. They promise tremendous potential to reduce cost and time and improve our understanding of the twinned systems.

Where human operators and their expertise are involved, such Digital Twins must provide access to data about the CPS in human-readable form via Graphical User Interfaces (GUIs), also called Digital Twin cockpits [10], allow for interaction and provide situational support via assistive services [22,30]. Cyber-Physical Production Systems (CPPSs) are long-living complex systems that

Published by Springer Nature Switzerland AG 2021
A. Dolgui et al. (Eds.): APMS 2021, IFIP AICT 630, pp. 333–341, 2021.
https://doi.org/10.1007/978-3-030-85874-2_35

operate in different environments and experience vastly different usage histories. Consequently, Digital Twins in production must be highly adaptable for different challenges, environments, and system states. Thus, they have to allow for configuration at the commissioning of the Digital Twin and throughout their lifetime. This configuration needs to be carried out by domain experts, which typically lack formal software engineering training.

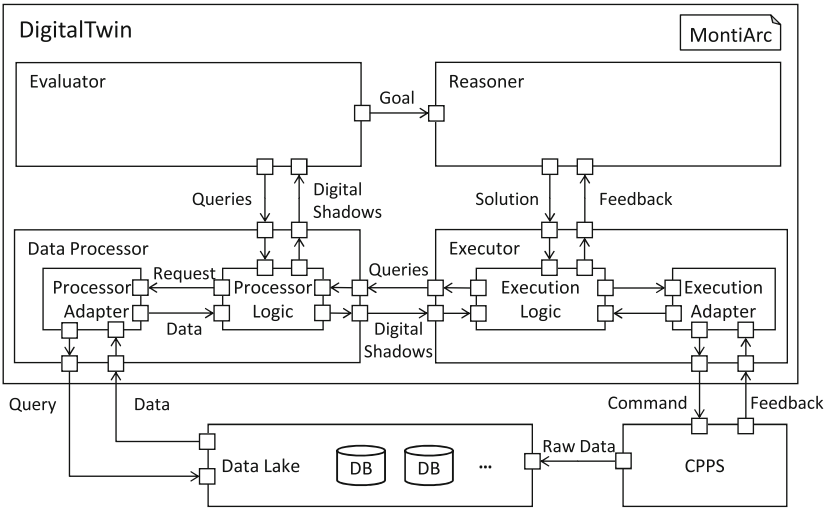

Fig. 1. Digital Twin architecture model

Low-code platforms [7,27] seem to be a promising way to enable domain experts to create, configure, and operate complex systems for a specific purpose in a particular domain. The combination of Model-Driven Software Engineering (MDSE) and low-code approaches have a high potential to facilitate the fast configuration, instantiation, and operation of Digital Twins. We envision a model-driven low-code approach for configuring Digital Twins and discuss challenges for future Low-Code Development Platforms (LCDP) that is based on our experience in MDSE for Digital Twins [3,10,20] and enterprise information systems [1,13] as well as in Software Language Engineering (SLE) [6,17]. In the following, we present preliminaries in Sect. 2, before we discuss our vision in Sect. 3, and discuss further future challenges in Sect. 4.

2 Background

Our vision rests on the model-driven architecture of Digital Twins and its integration with the MontiGem generator framework for Enterprise Information Systems as a visualization interface.

2.1 Digital Twins

Digital Twins are software systems comprising data, models, and services to interact with a CPPS for a specific purpose [3,10,20]. We devised a model-driven architecture of self-adaptive Digital Twins [4] that integrates various application-specific models to tailor the Digital Twin to specific challenges leveraging the MontiArc [5] architecture description language. The architecture (Fig. 1) realizes the MAPE-K loop [2] of self-adaptive systems through components that query a data lake (component `DataProcessor`), analyze resulting data (`Evaluator`), plan next actions (`Reasoner`), and execute these (`Executor`). To this end, it operates in the context of (1) a class diagram domain model and digital shadow [28] models, (2) leverages application-specific event models to trigger (re)actions of the Digital Twin, (3) uses various models (*e.g.,* Statecharts, case-based reasoning models, design of experiment models) to prescribe the behavior of CPPS and Digital Twin, and (4) connection models to communicate commands to the CPPS. In the following, we will leverage this architecture to integrate further (low-code) modeling languages towards truly domain-specific modeling of Digital Twins.

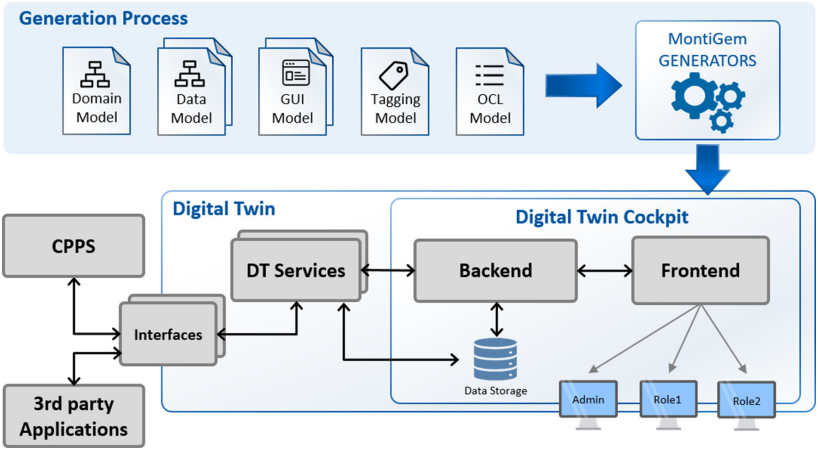

Fig. 2. Generating Digital Twin cockpits with MontiGem

2.2 MontiGem

Within [10], we have shown how to use the generator framework MontiGem [1] to create interactive Digital Twin cockpits. Figure 2 presents the main generation process. MontiGem can handle models in different Domain-Specific Languages (DSLs) as input. Required is only the domain model represented using Class Diagrams (CDs) [13]. Optional models include data models (views on the data structure), GUI models representing the graphical interfaces, tagging models for the addition of rights and roles, and Object Constraint Language (OCL) to define restrictions on the data model and data input validation in the GUI.

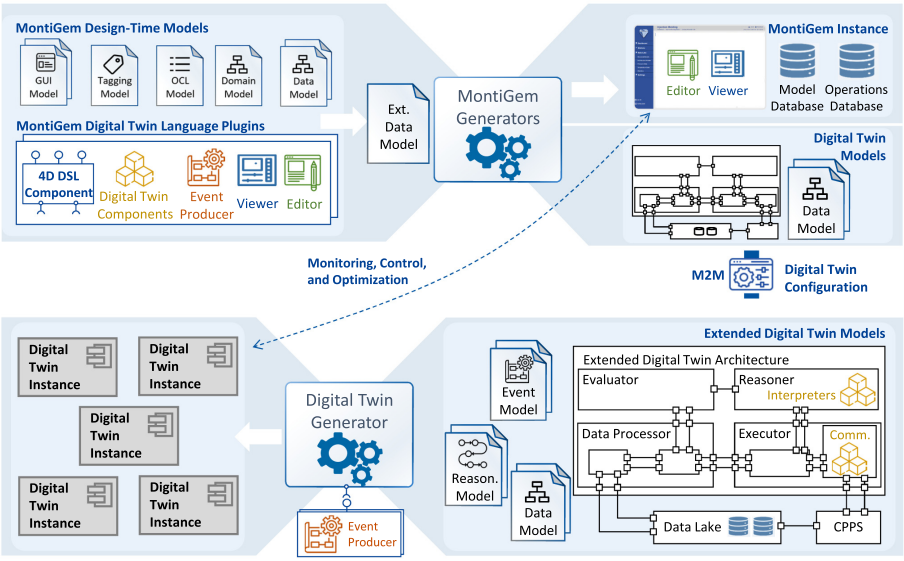

Fig. 3. A model-driven toolchain for the creation of LCDPs and Digital Twins

The generation result is an interactive Digital Twin cockpit, which can be integrated into a Digital Twin architecture, as shown in [10]. The Digital Twin cockpit and further components of the Digital Twin, such as reasoning or execution services, could be connected directly via the backend or by sharing data in a shared data storage. The proposed generation process allows for extensions and continuous re-generation in agile engineering processes. The generated code can be extended by hand-written classes, which use the extension functionality of object-oriented programming languages [16]. This method ensures that the additions remain during re-generation.

3 Model-Driven Synthesis of Digital Twin LCDPs

We envision combining our model-driven Digital Twin architecture with the MontiGem code generation framework to enable the pervasive model-driven development of low-code platforms for the creation, configuration, and operation of Digital Twins in production, as illustrated in Fig. 3.

MDSE of a LCDP for Digital Twins

Development of a LCDP for Digital Twins begins with selecting supported modeling languages through reasoning language plugins and communication language plugins. Each plugin comprises:

– A 4D DSL component [6] that encapsulates realizations of abstract and concrete syntax, well-formedness rules, and semantics in form of MontiCore [16]

grammars, Java context conditions, and FreeMarker-based code generators behind explicit interfaces of provided and required extensions.
- An interpreter for models of the component's DSL (reasoning language plugins) or a communicator translating reasoner input to communication objects for, *e.g.,* MQTT, OPC UA, or ROS (communication language plugins) as a (possibly hierarchically composed) MontiArc component model.
- An event producer creating Digital Twin event models from models of selected DSLs to invoke the interpreter of the Digital Twins to be created.
- A web-based editor and a web-based viewer for models of these DSLs that is compatible with the (graphical) language server protocol [26].

Given the models required for generating a MontiGem application and the language plugins and a model-to-model (M2M) transformation, the data model used as input for the MontiGem generator is extended with classes to capture the Abstract Syntax Tree (AST) of the available language plugins' DSLs. Afterward, the MontiGem generators process the MontiGem input models, the extended data model and the language plugins, generates a MontiGem application, integrates the editors into its user interface, creates databases for operation data and for models, and persists metadata about the available DSLs.

Model-Driven Creation of Digital Twins

With the low-code development platform in place, users can create Digital Twins conforming to our reference architecture using the available DSLs through a configuration assistant that employs information about these DSLs and their available models, *e.g.,* initially devised for other Digital Twins. To this end, they select (1) evaluation DSLs, *e.g.,* goal languages or BPMN for assistive services, (2) reasoning DSLs, *e.g.,* case-based reasoning [25] or PDDL [32], and (3) communication DSLs, *e.g.,* for connection to CPPS via OPC UA [24] and (4) assign an ID and further configuration information, *e.g.,* additional event models.

Based on the selection of DSLs, another M2M transformation integrates the interpreter components and communicator components of the corresponding language plugins into the reasoner and the executor of the Digital Twin's MontiArc architecture, respectively. Another M2M transformation produces new event models for the Digital Twins to react on model updates invoked by the connected MontiGem LCDP. The Digital Twin generator then takes the resulting extended Digital Twin architecture model, the handcrafted and generated event models, and its data models as input to produce executable Digital Twin instances. To this effect, it leverages the event producer components provided by the language plugins to derive events from the reasoning models that invoke their corresponding reasoner accordingly.

As the MontiGem LCDP and the Digital Twins operate on the same databases, the Digital Twin writes operation data, and the LCDP writes model changes, additional communication infrastructure is not necessary. In contrast, the CPPS is connected using a model of a communication language plugin.

4 Outlook

Our vision of a model-driven low-code approach for configuring Digital Twins facilitates researchers and practitioners in manufacturing in creating LCDPs integrated with Digital Twins and domain-specific modeling languages for the particular production challenges at hand. Extending, evolving, maintaining, and using the envisioned method in practices is subject to further challenges out of which selected are outlined below.

LCDP and Digital Twin Interoperability. Research and industry have produced various platforms (*e.g.,* Microsoft Azure, Amazon Greengrass, or Siemens MindSphere) and modeling techniques (*e.g.,* Eclipse Ditto, Hono, and Vorto, Microsoft Digital Twin Description Language) to model Digital Twins. These platforms are walled gardens that lock the users to a specific vendor without systematic means for interoperability. As production needs to integrate systems and solutions from various OEMs and suppliers, interoperability of Digital Twins and their platforms is essential. The LCDP in Fig. 3 allows to specify communication DSLs for different communication standards and generates needed application interfaces. Approaches such as [20] already showed how to generate such interfaces from models.

Modeling Language Modularity and Evolution. To support addressing evolving challenges with a LCDP for Digital Twins, the employed modeling languages and language infrastructure must allow for modularity within the language. This includes the combination of building blocks and the establishment of language hierarchies [18], which, for instance, is important if Digital Twins for different application areas in smart manufacturing request different depths of, *e.g.,* event or reasoning models.

Integration of Assistive Services. Where production is not fully automated, including human operators in the loop is crucial. To mitigate the increasing amount and detail of production information, Digital Twins should assist operators in making the best possible decisions [22]. Consequently, functionalities such as analyzing a current action, identifying next actions and suggesting their execution [15] should be integrated into the toolchain. By now, there is still research missing, *e.g.,* on which modeling languages could be incorporated into these processes, what aspects have to be modeled to provide meaningful, automated support, or to provide variety in supporting devices.

Collaborative Development. Digital Twins of production systems will address multiple concerns of the twinned system and its, *e.g.,* strategic, context. Consequently, multiple stakeholders, such as shop-floor experts and managers, might interact with the Digital Twin collaboratively. Textual modeling techniques generally support this. Within the last years, a variety of collaborative modeling tools evolved as browser- or cloud-based solutions [11,12,14]. However, within

the application domains of Digital Twins, areas such as support for graphical modeling, identifying and resolving modeling conflicts, as well as considering roles, rights, and corresponding views for successful collaborative intra- and inter-organizational modeling remain to be investigated.

Web-Based Debugging. For experimentation, configuration, and (virtual) commissioning, it is crucial to predict the Digital Twin's behavior on the twinned system. To this end, being able to debug, trace, and replay the Digital Twin's behavior is necessary. While there are various means to provide debugging, *etc.*., for modeling languages (*e.g.,* GEMOC Studio [9]), none of these provide generic or generative web-based interfaces.

5 Conclusion

The model-driven software engineering of low-code development platforms for Digital Twins in production promises powerful means to create and operate highly-specific platforms together with integrated Digital Twins for a more efficient configuration and operation of production systems. Our approach to combining MDSE and SLE to engineering such platforms signposts a possible realization of this vision.

References

1. Adam, K., Michael, J., Netz, L., Rumpe, B., Varga, S.: Enterprise information systems in academia and practice: lessons learned from a MBSE project. In: 40 Years EMISA: Digital Ecosystems of the Future: Methodology, Techniques and Applications (EMISA 2019). LNI, vol. P-304, pp. 59–66. Gesellschaft für Informatik e.V. (2020)
2. Arcaini, P., Riccobene, E., Scandurra, P.: Modeling and analyzing MAPE-K feedback loops for self-adaptation. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-managing Systems, pp. 13–23. IEEE (2015)
3. Bibow, P., et al.: Model-driven development of a digital twin for injection molding. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) CAiSE 2020. LNCS, vol. 12127, pp. 85–100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49435-3_6
4. Bolender, T., Bürvenich, G., Dalibor, M., Rumpe, B., Wortmann, A.: Self-adaptive manufacturing with digital twins. In: 2021 International Symposium on Software Engineering for Adaptive and Self-managing Systems (SEAMS), Los Alamitos, CA, USA, pp. 156–166. IEEE Computer Society, May 2021
5. Butting, A., Haber, A., Hermerschmidt, L., Kautz, O., Rumpe, B., Wortmann, A.: Systematic language extension mechanisms for the MontiArc architecture description language. In: Anjorin, A., Espinoza, H. (eds.) ECMFA 2017. LNCS, vol. 10376, pp. 53–70. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61482-3_4
6. Butting, A., Pfeiffer, J., Rumpe, B., Wortmann, A.: A compositional framework for systematic modeling language reuse. In: 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 35–46. ACM (2020)

7. Cabot, J.: Positioning of the low-code movement within the field of model-driven engineering. In: Guerra, E., Iovino, L. (eds.) MODELS 2020: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18–23 October 2020, Companion Proceedings, pp. 76:1–76:3. ACM (2020). https://doi.org/10.1145/3417990.3420210
8. Chen, X., Kang, E., Shiraishi, S., Preciado, V.M., Jiang, Z.: Digital behavioral twins for safe connected cars. In: 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 144–153 (2018)
9. Combemale, B., Barais, O., Wortmann, A.: Language engineering with the GEMOC studio. In: 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), pp. 189–191. IEEE (2017)
10. Dalibor, M., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Towards a model-driven architecture for interactive digital twin cockpits. In: Dobbie, G., Frank, U., Kappel, G., Liddle, S.W., Mayr, H.C. (eds.) ER 2020. LNCS, vol. 12400, pp. 377–387. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62522-1_28
11. Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: Collaborative repositories in model-driven engineering [software technology]. IEEE Softw. **32**(3), 28–34 (2015). https://doi.org/10.1109/MS.2015.61
12. Franzago, M., Di Ruscio, D., Malavolta, I., Muccini, H.: Collaborative model-driven software engineering: a classification framework and a research map. IEEE Trans. Softw. Eng. **44**(12), 1146–1175 (2018). https://doi.org/10.1109/TSE.2017.2755039
13. Gerasimov, A., Michael, J., Netz, L., Rumpe, B., Varga, S.: Continuous transition from model-driven prototype to full-size real-world enterprise information systems. In: 25th Americas Conference on Information Systems (AMCIS 2020). AIS Electronic Library (AISeL), Association for Information Systems (AIS) (2020)
14. Gray, J., Rumpe, B.: The evolution of model editors: browser- and cloud-based solutions. Softw. Syst. Model. **15**(2), 303–305 (2016). https://doi.org/10.1007/s10270-016-0524-2
15. Hölldobler, K., Michael, J., Ringert, J.O., Rumpe, B., Wortmann, A.: Innovations in model-based software and systems engineering. J. Object Technol. **18**(1), 1–60 (2019)
16. Hölldobler, K., Rumpe, B.: MontiCore 5 Language Workbench Edition 2017. Aachener Informatik-Berichte, Software Engineering, Band 32, Shaker Verlag, December 2017
17. Hölldobler, K., Rumpe, B., Wortmann, A.: Software language engineering in the large: towards composing and deriving languages. Comput. Lang. Syst. Struct. **54**, 386–405 (2018)
18. Johanson, A.N., Hasselbring, W.: Hierarchical combination of internal and external domain-specific languages for scientific computing. In: Zdun, U. (ed.) European Conference on Software Architecture Workshops (ECSAW 2014). pp. 1–8. ACM Press, New York (2014). https://doi.org/10.1145/2642803.2642820
19. Joordens, M., Jamshidi, M.: On the development of robot fish swarms in virtual reality with digital twins. In: 2018 13th Annual Conference on System of Systems Engineering (SoSE), pp. 411–416. IEEE (2018)
20. Kirchhof, J.C., Michael, J., Rumpe, B., Varga, S., Wortmann, A.: Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In: 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 90–101. ACM (2020)
21. Knapp, G., Mukherjee, T., Zuback, J., Wei, H., Palmer, T., De, A., DebRoy, T.: Building blocks for a digital twin of additive manufacturing. Acta Materialia **135**, 390–399 (2017)

22. Michael, J., Rumpe, B., Varga, S.: Human behavior, goals and model-driven software engineering for assistive systems. In: Koschmider, A., Michael, J., Thalheim, B. (eds.) Enterprise Modeling and Information Systems Architectures (EMSIA 2020), vol. 2628, pp. 11–18. CEUR Workshop Proceedings, June 2020
23. Pargmann, H., Euhausen, D., Faber, R.: Intelligent big data processing for wind farm monitoring and analysis based on cloud-technologies and digital twins: a quantitative approach. In: 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), pp. 233–237. IEEE (2018)
24. Pauker, F., Frühwirth, T., Kittl, B., Kastner, W.: A systematic approach to OPC UA information model design. Procedia CIRP **57**, 321–326 (2016)
25. Recio-García, J.A., González-Calero, P.A., Díaz-Agudo, B.: jcolibri2: a framework for building case-based reasoning systems. Sci. Comput. Program. **79**, 126–145 (2014)
26. Rodriguez-Echeverria, R., Izquierdo, J.L.C., Wimmer, M., Cabot, J.: Towards a language server protocol infrastructure for graphical modeling. In: 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, pp. 370–380 (2018)
27. Sahay, A., Indamutsa, A., Ruscio, D.D., Pierantonio, A.: Supporting the understanding and comparison of low-code development platforms. In: 46th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2020, Portoroz, Slovenia, 26–28 August 2020, pp. 171–178. IEEE (2020). https://doi.org/10.1109/SEAA51224.2020.00036
28. Schuh, G., et al.: Effizientere Produktion mit Digitalen Schatten. ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb **115**special), 105–107 (2020)
29. Um, J., Popper, J., Ruskowski, M.: Modular augmented reality platform for smart operator in production environment. In: 2018 IEEE Industrial Cyber-Physical Systems (ICPS), pp. 720–725. IEEE (2018)
30. Vathoopan, M., Johny, M., Zoitl, A., Knoll, A.: Modular fault ascription and corrective maintenance using a digital twin. IFAC-PapersOnLine **51**(11), 1041–1046 (2018)
31. Verner, I., Cuperman, D., Fang, A., Reitman, M., Romm, T., Balikin, G.: Robot online learning through digital twin experiments: a weightlifting project. In: Auer, M.E., Zutin, D.G. (eds.) Online Engineering & Internet of Things. LNNS, vol. 22, pp. 307–314. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-64352-6_29
32. Wally, B., et al.: Production planning with IEC 62264 and PDDL. In: 17th International Conference on Industrial Informatics (INDIN), vol. 1, pp. 492–499. IEEE (2019)