Preliminary Patterns for Artificial Intelligence-based Digital Twins

SHAUKAT ALI, Simula Research Laboratory, Oslo, Norway

Digital twins have shown promising results in many domains, such as healthcare and communication. Such digital twins have been designed to support various activities, such as anomaly detection and predicting failures during the design and operation of their corresponding physical twins. We have identified four preliminary patterns for creating digital twins based on developing digital twins with artificial intelligence (AI) techniques in different industrial and real-world contexts for software systems. Based on our experience, we report these patterns in this paper.

Keywords: Digital Twin, Machine Learning, Modeling and Simulation, Cyber-Physical Systems, Software Systems

1 INTRODUCTION

Digital twins (*"virtual replicas"*) have found applications in diverse domains for various system types [6]. However, this paper focuses on digital twins (DTs) of software systems such as cyber-physical systems, the Internet of Things, and other large-scale software. We proposed a high-level pattern for DTs in [11] for such systems (see Figure 1). A DT is a virtual representation of a system (i.e., *Physical Twin (PT)*) [6]. A DT receives continuously live data from the PT and keeps itself synchronized with the most recent PT state. Depending on its maturity level and security/safety considerations, a DT may control PT and take necessary actions. In our pattern, a DT comprises two components, i.e., the model and the capability. The digital twin model represents the virtual representation of the PT, possibly abstracting and simulating the PT's behavior. For example, it could be a simulation model created in Matlab/Simulink. The digital twin capability represents the functionality to be performed by the digital twin with support from the model. An example of capability is predicting failures before they occur in the PT. Such capability could be implemented with, for example, a machine learning prediction model. The capability relies on the model to perform its functionality more precisely.

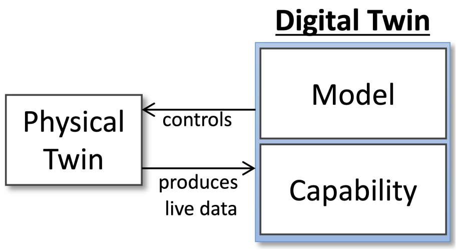

Fig. 1. Generic Pattern for Digital Twins

We realized the high-level DT pattern from [11] to build DTs with artificial intelligence (AI) techniques for different system types including water treatment and other plants (e.g., gas pipeline) [7, 8], autonomous driving systems [1], industrial elevators [9], train control and management system [10], medical devices [5], and cancer registration and support system [2]. Based on such experience, we identified four patterns for creating AI-based digital twins presented in this paper from Section 2 to Section 5.

2 PATTERN 1: CLASSICAL-AI DIGITAL TWIN

Context. Such DTs are built with data collected from the operation of a system's current or previous version(s). Such DT is common when: (1) the DT is built for an operational PT; (2) the DT wasn't considered during the design and development of the PT; (3) PT (e.g., medical devices in healthcare [5]) are purchased from a third party, and they are black-boxes with limited or no access to the internal functionality (e.g., source code).

Problem. How do we build a DT for a software system for which only historical and live data are available? In this case, other detailed knowledge about the system behavior, including design and development artifacts, is unavailable. Even if such knowledge is available, it is not detailed enough to construct a digital twin.

Forces. 1) *Need for Sufficient Amount of Data:* Sufficient amount of past data are needed to build a reasonable quality digital twin model and capability; 2) *High-Quality Data:* The quality of the past data must be good enough to be used for building a digital twin model and capability; 3) *Synchronization between DT and PT:* The data from PT to DT shall be transferred to DT with an acceptable delay to perform its intended capability; 4) *Interoperability between classical and AI models:* Classical and AI model need to work together to support the intended capability, which otherwise, could affect the capability, e.g., delayed predictions to a point where such predictions are not useful anymore.

Solution. This pattern is shown in Figure 2. *First*, an initial DT model (referred to as *Classical Model* in Figure 2) is built from past data collected from a system's operation, e.g., as an automaton, which is then continuously improved with real-time data from the operational system. *Second*, based on the continuous live data, the initial DT model is continuously fine-tuned to reflect the most recent PT state. On top of the DT model, the DT capability (referred to as *AI Model* in Figure 2) is built by employing AI techniques (e.g., neural networks) on the past data. This AI model and the DT model can perform various capabilities, e.g., anomaly detection using the most up-to-date state of the system reflected in the DT model and live data coming from the PT. In addition, the *Classical Model* and *AI Model* work together. The *Classical Model* supports the *AI Model* in performing the intended capability, whereas *AI Model* helps in improving the *Classical Model* further.

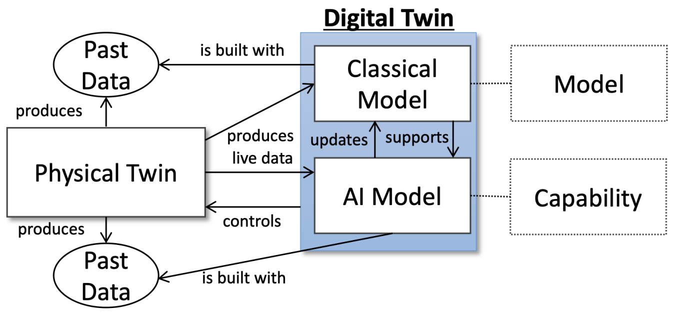

Fig. 2. Classical-AI Digital Twin Pattern

Implementation. Various AI algorithms are relevant depending on the type of system for which we are building digital twins and the type of capability needed. Examples include learning methods such as timed automata [7, 8], deterministic automata (e.g., with the L* algorithm), and stochastic automata as Markov chains [4] to build the DT model. Depending on the task, the DT capability can be built with various machine learning algorithms. For example, various variations of convolutional neural networks can be employed to identify objects on the road in the environment of autonomous driving systems. On the other hand, long short-term memory neural networks can be employed for predictions based on time series data, e.g., for predicting possible anomalies during the operation of a system with its digital twin.

Known Uses. DTs have been built for water treatment and other plants [7, 8], elevator systems [9], and train control and management systems [10]. For instance, the DT model was built as a timed automated [3], whereas Generative Adversarial Networks were applied to build DT-based anomaly detection capability in [7, 8].

Consequences. Such digital twins can be built with data, significantly reducing the cost of building digital twins manually. Moreover, building such digital twins requires less domain expertise since existing machine learning algorithms can be employed. On the negative side, the data may not reflect the detailed knowledge about the internal behavior of the PT and, therefore, might not help in making more accurate capabilities.

3 PATTERN 2: FULLY AI DIGITAL TWIN

Context. Such DTs have the same context as Pattern 1.

Problem. How to build a DT for a software system for which past and live data are available, but the system cannot be precisely represented as a classical model?

Forces. The first three forces related to data discussed in Pattern 1 are also relevant in this pattern. In addition, a force related to whether a combined AI model can accurately represent both the model and capability exist.

Solution. A DT and its capability are built as one AI model (see Figure 3), e.g., with various AI models depending on the context (e.g., see [2, 5]) from the past data. Later on, based on the live data, the AI model is continuously fine-tuned and improved. The AI model also performs its intended capability based on the live data.

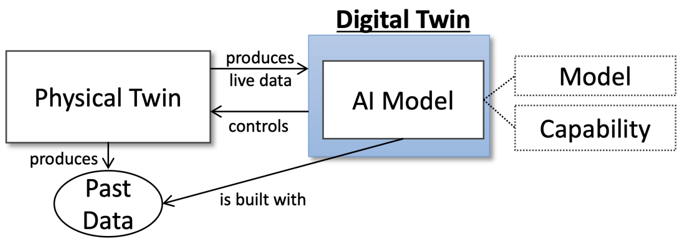

Fig. 3. Fully AI Digital Twin Pattern

Implementation. The implementation details related to the AI model in Pattern 1 also apply to this pattern.

Known Uses. DTs have been built for medical devices [5] and cancer registry system [2]. In both cases, variations of neural networks were built as the DT cores for predicting medical device responses and cancer medical rules validation, respectively.

Consequences. These DTs can be built solely on data, requiring less manual effort. However, the AI model of the DT needs to be kept updated to represent the most up-to-date version of the system due to evolution, which can be expensive.

4 PATTERN 3: MANUAL MODEL-AI DIGITAL TWINS

Context. These DTs are built with AI-based approaches and manually created models by domain experts. Such DTs are common in contexts where more detailed knowledge about the systems is available (e.g., design models) in addition to data. This allows for creating more detailed DTs since, with design models, one can capture a more detailed system view that is often missing in data collected, e.g., from sensors and actuators (e.g., relationships among different system components).

Problem. How to build a DT for a software system for which more detailed knowledge about the software system is available in addition to data?

Forces. The three forces related to data discussed in Pattern 1 also apply in this context. In addition, there is another force related to interoperability between the classical model and the AI model, as described in Pattern 1, which also applies here; however, with a difference that the classical model in Pattern 1 was learned from past data, whereas in this pattern, such model is manually created by a domain expert. To differentiate these two models, we will call the model in this pattern as *Manual Model*. An additional force exists related to the manual model's quality, which could have errors due to a misunderstanding of a domain expert about PT's behavior, leading to inaccurate PT behavior captured in its DT. In addition, the manual model may not be detailed enough to represent the PT precisely.

Solution. This pattern is shown in Figure 4. *First*, manual models are created, representing the digital twin model (referred to as *Manual Model* in Figure 4). Such a model is further improved with live data (see Figure 4). In addition, the DT capability is built as an *AI Model* based on past data. The *AI Model* and *Manual Model* work together to provide the intended capability. Similar to Pattern 1, *Manual Model* assists the AI Model in performing more precise capability. In contrast, the *AI Model* further helps to improve the *Manual Model* as more knowledge about the PT becomes available based on new data.

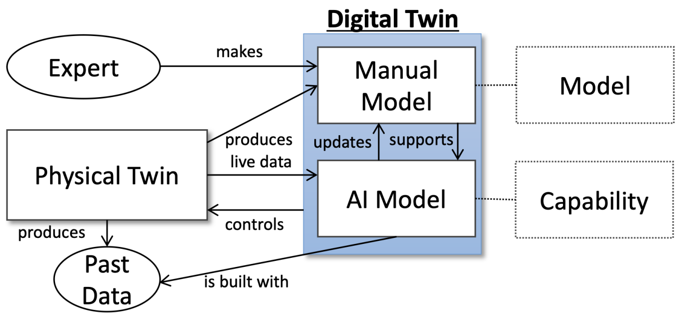

Fig. 4. Manual Model-AI Digital Twin Pattern

Implementation. For manual modeling of digital twins, many solutions exist, such as Open Modelica, Matlab/Simulink, and many tools supported with many formalisms in INTO-CPS association [1]. Concerning AI algorithms, the implementations discussed in Pattern 2 are still applicable in this case.

Known Uses. This pattern was implemented for autonomous driving systems in [1], where initial models were created in Modelica and later improved with data.

Consequences. Such digital twins are often much more detailed and can offer advanced analyses than the ones fully built with machine learning algorithms. However, building such digital twins costs extra since manual modeling is labor-intensive. In addition, manual modeling could have quality issues due to model faults.

[1]https://into-cps.org/

5 PATTERN 4: HYBRID-AI DIGITAL TWINS

Context. Such DTs are typically first built manually during the design and development of their underlying physical twin, and initially, no data is available. Later, additional DT is built when the data becomes available. As a result, two DTs for the same PT are available, one manually built by domain experts, whereas the second is built automatically with AI techniques from data. The key difference between this pattern and the previous three patterns is that we build two separate DTs, which are later integrated to form an integrated DT. In contrast, only one DT was built in the previous three patterns.
Problem. How to build and integrate two DTs of the same software system built from two perspectives: a manual modeling-based DT and an AI-based DT?
Forces. This pattern shares the same forces as Pattern 1, related to data, and Pattern 3, related to manual modeling. An additional force exists due to the differences between the manual modeling-based DT and the AI model-based DT, which may make their integration difficult and inefficient for the intended capability.
Solution. This pattern in shown in Figure 5. First, a DT is created based on manual modeling. Second, when the data becomes available, an AI-based DT is created using relevant AI techniques. Later, both DTs are integrated to form a more comprehensive DT (shown as *Integrated Digital Twin* in Figure 5).

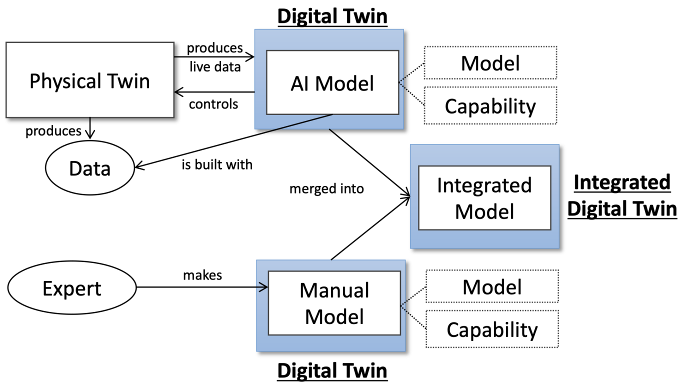

Fig. 5. Hybrid-AI Digital Twin Pattern

Implementation. In this case, the implementation details discussed for Pattern 3 also apply to manual modeling. In addition, the implementation details for AI-based DT from Pattern 2 also apply to this pattern.
Known Uses. This pattern was applied to build DTs of medicine dispensing machines [5], where model-based DT was created as executable UML state machines in parallel to neural network-based DT from data.
Consequences. Creating two DTs incurs additional costs. In addition, their integration might not be possible or become inefficient for the capability for which they are designed. The evolution of these DTs becomes challenging since one must maintain the two DTs and later integrate them. Integrating the two DTs could also be challenging and incur extra costs.

6 CONCLUSIONS AND FUTURE WORKS

Based on building artificial intelligence (AI)-based digital twins for industrial and real-world software systems, we presented four high-level patterns for creating digital twins. We described their context, presented the patterns, and provided their known cases. In the future, we plan to describe detailed patterns from various aspects, such as communication between digital twins and physical twins, AI models, and digital twin evolution.

7 ACKNOWLEDGMENTS

The Co-Evolver project (Project# 286898), funded by the Research Council of Norway, supports this work. We want to thank Anja Bertels, Y. C. Cheng, Martin Gutsche, Kiro Harada, Jiwon Kim, Cesare Pautasso, and Joseph W. Yoder for their feedback on the paper.

REFERENCES

[1] Qiwei Chen, Tiexin Wang, Chengjie Lu, Tao Yue, and Shaukat Ali. 2022. Enhancing the Realism of Autonomous Driving Simulation with Real-Time Co-Simulation. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (MODELS '22)*. Association for Computing Machinery, New York, NY, USA, 659–667. https://doi.org/10.1145/3550356.3561558

[2] Chengjie Lu, Qinghua Xu, Tao Yue, Shaukat Ali, Thomas Schwitalla, and Jan Nygård. 2023. EvoCLINICAL: Evolving Cyber-Cyber Digital Twin with Active Transfer Learning for Automated Cancer Registry System. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1973–1984. https://doi.org/10.1145/3611643.3613897

[3] Alexander Maier. 2014. Online Passive Learning of Timed Automata for Cyber-Physical Production Systems. (2014).

[4] Edi Muškardin, Bernhard K. Aichernig, Ingo Pill, Andrea Pferscher, and Martin Tappler. 2021. AALpy: An Active Automata Learning Library. In *Automated Technology for Verification and Analysis*, Zhe Hou and Vijay Ganesh (Eds.). Springer International Publishing, Cham, 67–73.

[5] Hassan Sartaj, Shaukat Ali, Tao Yue, and Julie Marie Gjøby. 2023. HITA: An Architecture for System-level Testing of Healthcare IoT Applications. arXiv:cs.SE/2309.04223

[6] Andreas Wortmann. 2023. Digital Twins Definitions. Retrieved December 30, 2023 from https://awortmann.github.io/research/digital_twin_definitions/

[7] Qinghua Xu, Shaukat Ali, and Tao Yue. 2021. Digital Twin-based Anomaly Detection in Cyber-physical Systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. 205–216. https://doi.org/10.1109/ICST49551.2021.00031

[8] Qinghua Xu, Shaukat Ali, and Tao Yue. 2023. Digital Twin-Based Anomaly Detection with Curriculum Learning in Cyber-Physical Systems. *ACM Trans. Softw. Eng. Methodol.* 32, 5, Article 113 (jul 2023), 32 pages. https://doi.org/10.1145/3582571

[9] Qinghua Xu, Shaukat Ali, Tao Yue, and Maite Arratibel. 2022. Uncertainty-Aware Transfer Learning to Evolve Digital Twins for Industrial Elevators. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1257–1268. https://doi.org/10.1145/3540250.3558957

[10] Qinghua Xu, Shaukat Ali, Tao Yue, Zaimovic Nedim, and Inderjeet Singh. 2023. KDDT: Knowledge Distillation-Empowered Digital Twin for Anomaly Detection. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2023)*. Association for Computing Machinery, New York, NY, USA, 1867–1878. https://doi.org/10.1145/3611643.3613879

[11] Tao Yue, Paolo Arcaini, and Shaukat Ali. 2021. Understanding Digital Twins for Cyber-Physical Systems: A Conceptual Model. In *Leveraging Applications of Formal Methods, Verification and Validation: Tools and Trends*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 54–71. https://doi.org/10.1007/978-3-030-83723-5_5