Sustainable Digital Twin Engineering for the Internet of Production

Shan Fur, Malte Heithoff, Judith Michael, Lukas Netz, Jérôme Pfeiffer, Bernhard Rumpe, and Andreas Wortmann

1 Introduction

Digital twins (DTs) [1–3] have become more prevalent recently. They are used to support the design, operations, and analysis of complex systems in many domains, such as automotive [4], construction [5], medicine [6], or robotics [7], and comprise much information about the systems and processes of the twinned original system. They promise not only a better understanding of cyber-physical production systems (CPPSs) during their design time [5], but also more efficient operations of these systems [8]. For this purpose, (i) operational knowledge must be obtained from operational data, which serves to optimize the system under consideration and to develop subsequent system versions more efficiently and (ii) expert knowledge must

M. Heithoff · J. Michael (✉) · L. Netz · B. Rumpe
Software Engineering, RWTH Aachen University, Aachen, Germany
e-mail: michael@se-rwth.de

M. Heithoff
e-mail: heithoff@se-rwth.de

L. Netz
e-mail: netz@se-rwth.de

B. Rumpe
e-mail: rumpe@se-rwth.de

S. Fur · J. Pfeiffer · A. Wortmann
Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, Stuttgart, Germany
e-mail: shan.fur@isw.uni-stuttgart.de

J. Pfeiffer
e-mail: jerome.pfeiffer@isw.uni-stuttgart.de

A. Wortmann
e-mail: andreas.wortmann@isw.uni-stuttgart.de

E. Karaarslan et al. (eds.), *Digital Twin Driven Intelligent Systems and Emerging Metaverse*, https://doi.org/10.1007/978-981-99-0252-1_4

be made machine-processable for the operation of digital twins. Operational knowledge and expert knowledge [9] can be represented by models and, thus, automatically processed at runtime by the digital twin and the system. The operation of such digital twins therefore requires the use of software and system models at runtime, which demands that appropriate models can be formulated, analyzed, and used for interpretation or synthesis. For this purpose, the modeling languages in which these models are formulated must be explicit, machine-processable, and meaningfully integrated. Yet, most digital twins are designed and engineered ad-hoc, in a piecemeal fashion, which is costly, binds valuable engineering resources and hampers research as well as industrial application of digital twins.

Leveraging the abstraction and automation of model-driven development yields more efficient and sustainable engineering methods for digital twins. Based on interdisciplinary research conducted at the German "Internet of Production"[1] excellence cluster, we combine model-driven methods for the engineering of information systems, software architectures, and software language engineering to systematically and sustainably engineer digital twins. Within this chapter, we discuss challenges on the road to a systematic engineering of digital twins, present our model-driven approach for their engineering as well as possible implementations for different purposes. Our insights may guide researchers and practitioners to sustainable, planned, and efficient engineering and operations.

In the following, Sect. 2 introduces foundations before Sect. 3 discusses challenges in engineering digital twins. Section 4 introduces sustainability with and for digital twins, and Sect. 5 presents model-based approaches to consider sustainability with and for digital twins. Section 6 debates the related work before the last section concludes.

2 Background

2.1 Digital Twins in Production

A pillar of "Industry 4.0" [10, 11] is the digitization of participating CPPSs, processes, and stakeholders to facilitate design-space exploration, integration, verification and validation, monitoring, and the optimization of system behavior. Under the umbrella term "digital twin" [2, 3], research and industry in production have produced various approaches to modeling the digital representations of CPPSs for specific purposes. These approaches define digital twins as "digital equivalent to a physical product" [12], an "always current digital image of the production system" [13], "a mimic of a real-world asset displaying up to date information of what is currently happening" [14], "an integrated virtual model of a real-world system containing all of its physical

[1] Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy—EXC 2023 Internet of Production—390621612. Website: https://www.iop.rwth-aachen.de.

information and functional units" [15], or "a virtual representation of an asset used from early design through building and operation" [16]. While there are many more approaches to define digital twins, most of these assume the same main functionality inherent in the definitions above: to digitally represent a system, process, or person during their operation [17].

In the following, we understand digital twins as software systems comprising data, models, and services to interact with a CPPS for a specific purpose. Therefore, each digital twin is connected (twinned) with an original system (the CPPS), which it represents and interacts with. We also assume the distinction that a digital twin has automated data flows from and to the twinned original system [2] to (i) obtain data from it to represent its behavior and (ii) to send data to it to change its behavior. These purposes can include obtaining a better understanding of the system under development or operation [1], predicting its behavior [18, 19], such as its need for maintenance, or prescribing its behavior to optimize it [8, 20].

2.2 Model-Driven Engineering

The number of software systems is continuously increasing. Additionally, they are becoming more important and more complex [21, 22]. The increasing complexity of the software of such CPPSs demands better concepts, methods, and tools that enable overcompensating this growth in complexity and harnessing their potential. An important reason for the complexity of CPPS' software is the conceptual gap between the problem domain challenges and the solution domain peculiarities. Overcoming this gap with handcrafted solutions demands enormous efforts and gives rise to *accidental complexities* [21], which are addional challenges in the solution domain that the problem domains abstract away from. For instance, to make a robot pick an object (a problem domain task), one needs to program it taking care of memory management, persisting data, or network access (solution domain peculiarities). These accidental complexities increase software engineering risks, and it is paramount to reduce these.

Model-driven engineering [21, 23] captures software development methodologies that employ models to increase abstraction and reduce the conceptual gap. Therefore, from a model-centered perspective [24], researchers and practitioners utilize models as primary communication and development artifacts for various engineering activities, ranging from design, to documentation, requirements modeling, implementation, or deployment. To be machine-processable, these models conform to modeling languages [25], which can use more abstract terminology and concepts than programming languages. Using modeling languages tailored to a specific domain, so-called domain-specific languages (DSLs), enables problem domain experts to contribute to the development of complex systems directly and without needing to become solution domain experts. Models of such languages then can be automatically translated into software artifacts leveraging code generators that embody domain expertise (e.g., how to take memory management into account properly).

2.3 Digital Shadows for Production

Digital Shadows (DSs) [26, 27] capture the idea of collecting, aggregating, and abstracting manufacturing data for specific purposes sufficiently fast, such that decisions made on this data can impact processes and CPPSs having produced the data in a timely fashion. In contrast to digital twins, digital shadows, therefore, comprise only a reduced or abstracted subset of the available data (and possibly models) to represent a system with respect to a specific purpose. For this purpose, digital shadows need to comprise various kinds of data (such as measurement data, simulation data, and models) from different sources. We abstract and aggregate the data in domain-specific and application-specific ways and augment it with necessary metadata to fulfill the DSs' purpose. Through their specific abstraction and aggregation, digital shadows can be optimal data structures to ensure timely decision-making in CPPSs.

A result of the interdisciplinary research in the German "Internet of Production" cluster of excellence is a reference model that captures the main concepts of the digital shadow. It acts as a common foundation to standardize the architecture of the relevant data held in a digital shadow. The reference model is shown in Fig. 1 as a UML class diagram and shows the reference structure, consisting among others of the main container *DigitalShadow*, the *asset* it stands for, the *purpose* the shadow fulfills and it's contained *DataTraces*. As the authors of [26] propose a purpose-driven approach for the DS, the purpose specifies what the DS is supposed to do. An asset "is an item, thing, or entity that has potential or actual value to an organization" [28, 29] and can be of physical or virtual nature. Assets are, e.g., manufacturing machines, human workers or a software component. For more complex assets, the structure can be composed into subcomponents. *Sources* (which also can be an asset) produce data of interest for our system and are, e.g., measurements or sensor data. This data is captured as *DataPoints* gathered in *DataTraces* where data points are single data entries, e.g., a table column as a snapshot at a point in time. A data trace always originates from a source (e.g., machine) and can be enriched with *MetaData* holding additional information like the experimental machine setup relevant for this data. In addition, *Models* supply a deeper understanding of the system structure and behavior or provide calculation specifications on how to aggregate and abstract the gathered data. Using this reference model, we can formalize the contextualized data aggregation in a common manner that sets the foundation to model-driven generation of digital shadows.

3 Challenges of Digital Twins

The idea of a digital shadow is subject to ongoing research [30, 31] and in the past has rarely been properly distinguished from the idea of a digital twin. Following a popular definition of digital twins based on the data flows between the original and its digital representation [2], the distinction is clear: a digital shadow follows the

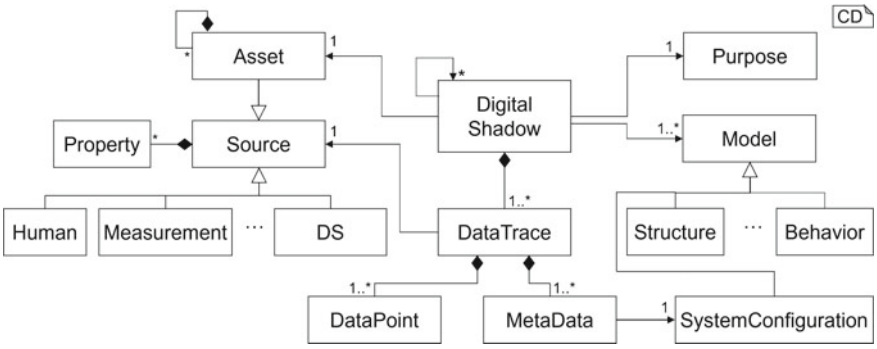

Fig. 1 Conceptual model describing digital shadow structures [26]

original system under a specific illumination (view), whereas the digital twin follows the original system as well but also may change the behavior of that system. To this end, a digital twin must be a complex software system that interacts with the original system: neither a simulation model nor some cloud-based data storage and analytics (e.g., with AWS or Microsoft Azure) alone can fulfill the requirements imposed by this definition.

This confusion about definitions is a prime reason for why research on the reuse of digital twin (parts) is not advancing at a fast pace. Despite knowing that reuse (e.g., in the form of inheritance, frameworks, libraries, or containers) is one of the prime drivers of efficient software engineering (essentially, digital twins are software), there currently is little research on reusing digital twin parts to systematically and efficiently engineer more complex digital twins from less complex ones. For instance, when composing a car chassis with a motor, the corresponding digital twins should be composed as well, or when integrating a sensor in a production system and the system into a factory, their twins should be integrated as well. Instead, digital twins are created in an ad-hoc manner.

In the literature, there are different life cycle stages of digital twins w.r.t. to the twinned system. A DT might represent the original system "as-designed", "as-manufactured", or "as-operated" [32]. While digital twins "as-designed" tend to be in place at design time of their corresponding original system [33], the latter two kinds of representations often aim at representing the runtime of the original system. There is little research on integrating these perspectives, which can entail that there are DTs of a system "as-designed" and "as-operated" that are developed independent from another and with little synergies between both. One important reason behind this is that this terminology hides a fundamental distinction: a digital twin of a system "as-designed" aims to represent the idealized type of that system, that is, its developers aspire to it as being a valid representation for all instances of that system, whereas a digital twin "as-manufactured" or "as-operated" needs to incorporate manufacturing tolerances and effects, as well as the wear and tear of the individual system instance. This gap demands further investigation.

In line with these challenges, there is also little research on systematically engineering digital twins together with the original system (greenfield) or after the original system has been deployed (brownfield). Both enable different functions in the corresponding digital twins. In a greenfield approach, the data required for the digital twin to perform its intended functions can be considered in the interface of the original system. However, in a brownfield approach, this might not be possible and hence, brownfield engineering of digital twins will be subject to restrictions regarding what can be observed from the original system.

4 Sustainability and Digital Twins

Sustainability is of increasing importance. It comprises social, economic, and environmental aspects [34]. Social sustainability is often related to respect for individuals, equal opportunities, diversity, and human rights. Economic sustainability is related to economic growth. Environmental sustainability has the aim to improve human welfare through the protection of natural capital. They can be seen either as distinct perspectives [35] or as a system influencing one another [36]. The further definitions of these aspects differ in the literature, but what seems to unite them is the critique of the economic status quo from different ecological and social perspectives [34]. The United Nations General Assembly has further detailed these areas and suggest 17 sustainable development goals (SDGs) with 169 associated targets [37] that should be achieved by the member states. This results in passing on the targets to companies, local authorities, and nudging of individuals.

Recent literature about sustainability and digital twins has a special focus on sustainability assessment or evaluation, e.g., for educational buildings [38], railway station buildings [5], or smart campuses [39]. Other work focuses on the improvement of sustainability performances of whole value chains due to the simulation and optimization services digital twins are able to provide, e.g., in production [40], the future development and assessment of intelligent manufacturing along a set of indicators [41] and life cycle sustainability assessment in the clothing industry [42].

To sum up these approaches, we can develop digital twins

- to support the assessment of sustainability targets due to their ability to monitor, calculate, and visualize key sustainability indicators defined by humans,
- for the simulation and forecasting of sustainability indicators if they use historic information together with forecasting algorithms, and
- which integrate digital twin services that
  - assist with responsible consumption and use in relation to created products,
  - enable simulation of different variants of digital twins before building the physical one to improve resource efficiency,
  - facilitate optimizing production processes toward waste reduction and energy saving allowing a responsible production, and
  - provide self-adaptability to improve resource efficiency.

Talking about the *sustainable development* of certain objects, none of these approaches consider that the digital twins themselves could be the objects that are sustainably developed. Model-driven engineering [43] is a promising approach to support the sustainable development of digital twins in different regards, especially when using DSLs. It leads to (1) an increased development speed and reduced development time; (2) better software quality, e.g., less bugs, because of well-defined domain-specific modeling languages, automated model checking, transformation, as well as test and test case generation, leading to reduced development time in the long run of a software system; (3) improved maintainability as cross cutting implementation aspects can be changed in one place which again reduces the development time; and (4) empowered domain experts by developing low-code platforms for the development of digital twins.

Considering human resources, these aspects are supporting sustainable development goals in the areas of resource efficiency in consumption and production and allow to reduce technological inequalities. Less development time leads to reduced energy needs for the engineering process of digital twins.

In the following section, we show some of these approaches and their impact for companies, products, and humans.

5 Approaches for Sustainably Developed Digital Twins

We have conceived, analyzed, discussed, and realized different model-driven implementations of digital twins and the process to derive large parts of their implementation from these models to reduce the effort and resources required in engineering digital twins and ultimately improve the economic sustainability of their development.

5.1 Model-Driven Engineering of Self-adaptive Digital Twins

We have realized our architecture for DTs using the component and connector architecture description language MontiArc [44]. In MontiArc, software systems are described using hierarchical components that are connected via typed directed ports. Ports describe incoming and outgoing messages of components. Components can either be decomposed, consisting of one or more subcomponents, or atomic, providing a behavior implementation on themselves.

Our digital twin architecture [1, 8] facilitates self-adaptive manufacturing by recognizing the behavior of the twinned system that diverges from the intended behavior over time. It then takes measurements to fix or mitigate this divergent behavior. We employ different modeling techniques to leverage domain expertise to improve the capabilities of digital twins for adapting to such situations.

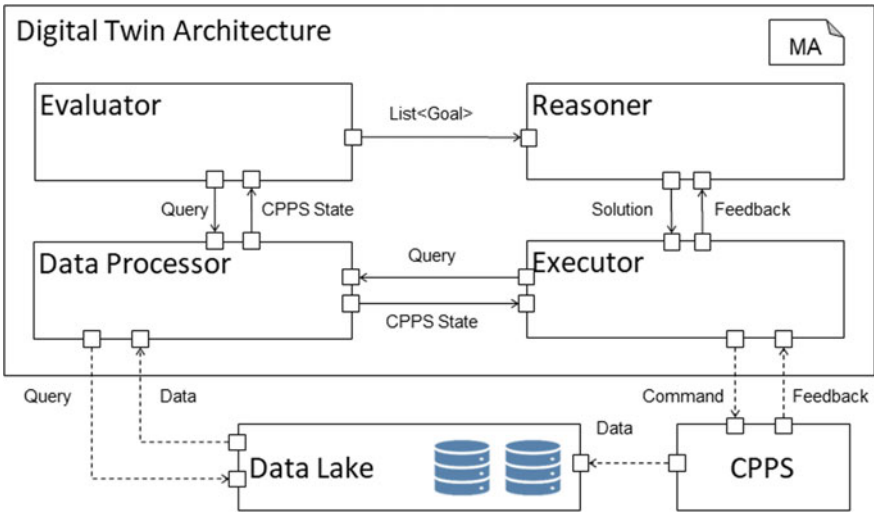

Fig. 2 Reference architecture for self-adaptive digital twins (based on [1])

The architecture for self-adaptive digital twins consists of four components each realizing a step in the self-adaptive loop (see Fig. 2):

1. *Data Processor*: It is connected to and collects relevant data from the *Data Lake*. The *Data Lake* encapsulates multiple databases storing data about the physical system and its environment.
2. *Evaluator*: It supervises the data collected by the data processor and triggers the reasoner if unintended or divergent behavior is recognized.
3. *Reasoner*: If triggered, based on data describing the intended behavior, it plans a solution to rectify the diverging behavior.
4. *Executor*: It is connected to the physical system, converts the solution of the reasoner into machine-executable commands, sends them to the physical system, and observes their execution.

To instruct and transfer domain knowledge into the digital twin architecture in a model-driven fashion, we leverage different domain-specific languages:

- *UML/P class diagrams* [45] describe elements of the domain, and their relations with each other.
- *Object-constraint language*[2] formulate constraints on the classes of the domain model.
- *Event condition action* [1] models enable domain experts to define events based on conditions over instances of the classes of the domain model. Actions define the digital twin's reaction to an occurring event. This action triggers the reasoner

[2] https://www.omg.org/spec/OCL/2.4/PDF.

of the digital twin to plan a solution for the detected event. Event models are interpreted during runtime.

- *Case-based reasoning* [8] is used for problem-solving. It reuses existing solutions from already encountered situations and computes solutions to occurring problematic situations on their basis. A case contains a condition based on the domain model, its solution, and the intended situation after applying the solution. Case similarity models define similarity metrics for cases. With that, the case-based reasoner knows based on which parameters, cases are more or less similar. Both, the case description models and the similarity models are interpreted during runtime. With more cases solved, the case base grows over time.
- *Communication specifications* enable the definition of data types that are accessible via a specific endpoint with a defined protocol. Currently, the architecture supports OPC-UA [46] and MQTT[3]as communication protocols in our communication specification models.
- Query definitions are specifiable via an *expression language*. It comprises the expressions such as maximum, minimum, or simple mathematical expressions. The query language is employed for collecting and aggregating data in digital shadows from sources such as the *Data Lake*.

For deriving models of the mentioned modeling languages, existing engineering models of the CPPS can be reused. To this end, CAD, kinematics, material flow, control models, and others can be employed to derive events and parts of the domain model. This lowers the initial effort in developing the digital twin. The automatically derived models can be enriched by domain experts with further details.

Regarding sustainability, our digital twin enables humans to define sustainability via the modeling of events and case-based reasoning goals, and the respective architecture components responsible for monitoring and calculating indicators based on these models enable the assessment of sustainability targets. Furthermore, our digital twin is self-adaptive to improve efficiency and save resources by optimizing the production process.

Sustainable development is achieved through increased development speed by employing modeling languages and models for different aspects of the digital twin. These modeling languages are domain-specific, and, thus, empower domain experts to configure the digital twin to their needs through the reuse and semantic reification of common concepts of the model-driven development of digital twins.

5.2 Generating Digital Twin Cockpits

Digital twins require an interface for inspection and control. We define a digital twin cockpit as follows:

"A digital twin cockpit is the user interaction part (UI/GUI) of a digital twin.

[3] http://mqtt.org.

It provides the graphical user interface for

1. visualizations of its data organized in digital shadows and models, and
2. the interaction with services of the digital twin, and thus
3. enabling humans to access, adapt, and add information and
4. monitor and partially control the physical system [47].”

Key aspects for a digital twin cockpit, such as the data structures and user interfaces, can be described with models. Using the MontiGem generator framework [48], developers produce a web application that can serve as a cockpit for a digital twin. It can be connected to the reference architecture for self-adapting digital twins [49]. The target application is a server-client architecture connected to a relational database (see Fig. 3). To realize the digital twin cockpit, a data structure and a UI generator are used. The first one generates the infrastructure that connects both the database and the client to the server. Based on the input domain model, it provides a multitude of interfaces to perform CRUD operations in the database as well as process client input in a systematic, authorized, and authenticated manner. The components created by the UI generator are fitted to the provided interfaces from the data-structure generator and thus provide implementation for detailed views on the current data available to the back end.

This approach supports reusability of components and models, due to its model-driven design, enabling sustainable methodologies for software development. With

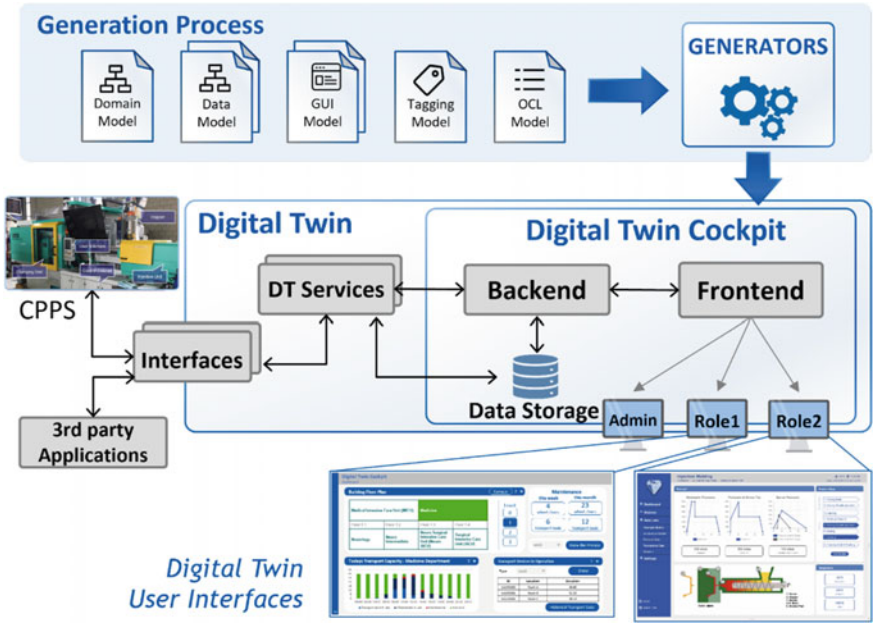

Fig. 3 Artifacts of the MontiGem generator framework used to generate a digital twin cockpit with multiple interfaces according to each role of the end user

this approach, the developer can rapidly implement digital twin cockpits that are easily adapted with custom logic and extended with further complex data structures.

5.3 *Process Prediction as a Digital Twin Service*

Using process mining techniques, we are able to analyze event data from physical systems and extract information related to processes, e.g., discover process models [50]. To systematically improve the operation of digital twins, we can use these techniques: Process discovery from runtime data of the physical object, conformance checking if the processes of the physical object are running as planned, and simulations or process prediction using process models where changes in processes in the long run can be foreseen [18].

We need to additionally provide functionality to handle explicated processes of the physical object and its context in the digital twin cockpit, which leads us to the concept of *process-aware digital twin cockpits*. A process-aware digital twin cockpit "is a digital twin cockpit that additionally provides functionality to handle explicated processes of the physical object and its' context" [47]. To integrate process prediction and conformance checking services allows us to analyze the processes of the CPPS based on real-time data during runtime of a self-adaptive DT.

We envision a model-driven DT architecture that uses models at runtime and incorporates process mining techniques (see Fig. 4) covering the following six steps:

1. **Generation of the DT**: Using domain knowledge of experts and from engineering models, e.g., AutomationML, CAD, Modelica, SysML, we can create a set of application-specific models such as class diagrams for the structure, or process models representing the machine processes or processes of related

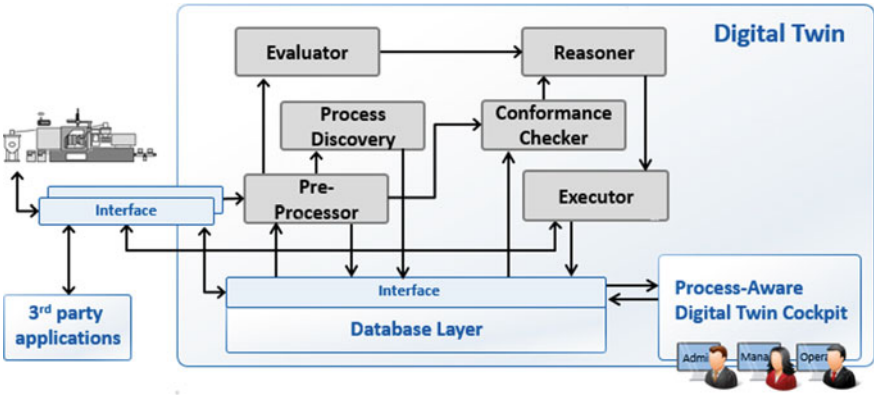

Fig. 4 Process discovery services as well as conformance checking services as parts of a digital twin visualized in process-aware digital twin cockpits

humans. Together with application-independent models, e.g., the DT architecture or the basic digital shadow or process model structure to handle models during runtime, these models are used as input for code generators to generate a DT (see Fig. 4).

2. **Configuration of the generated DT application**: We have to add relevant runtime models and specify digital shadow types. We need to know for which purpose the DS is needed, which data accessed by a fully qualified address with respective data points and metadata, should be chosen and how it should be aggregated.
3. **Initialization of DSs**: Using the DS types, the DT computes relevant DSs from the latest data and stores them for further processing.
4. **Process discovery**: By applying process discovery algorithms on data transformed into event logs, we can discover how processes run in the real environment. They can be stored and visualized in the DT cockpit.
5. **Runtime analysis**: In a next step, discovered processes can be compared to processes-to-be in conformance checkers which allows to identify problems and reconfiguration needs of the physical object. Moreover, they can be used for process prediction.
6. **User interaction**: At runtime of the DT, human users can view data and processes about and interact with the physical object and the provided services of the DT.

These functionalities allow us to create DTs supporting sustainability, e.g., to compare as-is-processes with to-be processes regarding their use of materials and production of waste.

5.4 Low-Code Platforms for Model-Driven Digital Twins

Digital twins are highly complex software systems that require a high amount of development resources over a long time period. A high number of the components of a digital twin are either systematic or can be categorized as “boilerplate code”, thus enabling a good application of generative approaches. Low-code development platforms (LCDP) aim to operate on similar principles as generators, by providing a large amount of implementation based on simple configurations and models.

The generator framework MontiGem [48] can be used to create a LCDP that enables the developer to configure and finally generate a digital twin [51, 52] (see Fig. 5). MontiGem is extended with LCDP language plugins and a model library covering multiple use cases. This enables the generation of a LCDP that provides the developer with user interfaces to configure the multiple models that ultimately define the digital twin. Once configured, the same framework can be used again to generate further digital twins for the same use case.

This approach combines the benefits of generators and LCDPs to rapidly produce multiple digital twins that can be used with similar configurations, e.g., on multiple

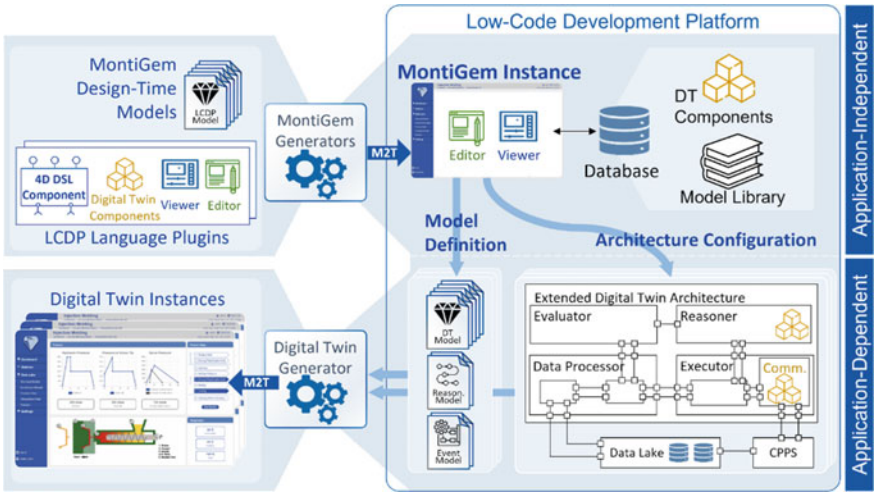

Fig. 5 Process in which MontiGem is used to generate an LCDP, which in turn is used to configure and control digital twins

machines on the same factory floor. This approach reduces development resources and, thus, supports the sustainable development of DTs.

5.5 Ontologies and Digital Twins

Ontologies [53] provide the terminology to express domain knowledge and make this locally gathered knowledge and its semantic concepts globally available. Ontologies, similar to UML class diagrams, describe structural parts of (production) data as well as their semantic interconnection. Reasoning and inference rules are used to derive new knowledge on a given data set. One of the driving purposes of ontologies is the ability to reuse domain knowledge: A detailed ontology, published by a group of researchers, can easily be repurposed by others allowing for combining ontologies to larger vocabularies tailored to a specific domain.

In our digital twin architecture, we can use ontologies to enrich the domain's structural part described by annotating additional information to classes and attributes in the class diagram. Commonly used concepts in production, such as sensors or material properties, are already defined in ontologies [54] and provide the development process of DTs with a formally defined vocabulary. This way, we can connect locally specified structure elements to globally valid concepts in a model-driven manner, reuse pre-existing components, and provide an easier comparison to other use cases.

5.6 Assistive Services Within Digital Twins

Digital twins could provide *intelligent assistive services* supporting human operators, e.g., they could assist operators in production processes in making the best possible decisions using human-behavior goals [55]. This is important, as operators have to handle an increasing amount of detailed information. Within not fully automated production steps, we can use assistive services within DTs to analyze a current task, identify next tasks, and suggest their (semi-automated) execution [56]. These intelligent assistive services [57] (1) provide support for human behavior tailored to specific situations, e.g., in case of adding resources in the production process or making repairments. (2) The support is based on stored and real-time monitored structural context and behavior data. Within DTs, especially ones which explicitly handle process information, large parts of the data are already available, e.g., process models for describing the physical object or relevant data and parameters. Missing aspects which have to be additionally modeled for realizing assistive services are support processes including user interaction. (3) The support is provided at the time the person needs it, e.g., when a human user has to operate with the CPPS, or when the person requests it, e.g., via interaction components of the digital twin. To realize such assistive services within the system architecture of DTs requires additional interaction components for user communication and assistive visualization and the internal handling of process models during runtime. Process models can be defined manually or one can use methods for the semantic annotation of user manuals [58] of production machines to automate the setup of assistive information.

To incorporate assistive services within digital twins support two different aspects of sustainability: Within the production process, assistive services can be used to provide users step-by-step support to improve key sustainability indicators. If products are delivered to customers together with a digital twin of the product, assistive services can be used to support customers with improving key sustainability indicators related to the use of the product.

5.7 Calibration and Adjustment of Simulation Models of Digital Twins

The simulation models used in virtual commissioning are suitable as a basis for digital twins of production systems, as they represent the behavior of these systems and are often already available from the development phase. These “as-designed” simulation models represent the respective production system in a generalized way [59, 60]. However, a digital twin requires simulation models that represent the real system; an “as-operated” model. If the “as-designed” models are used without any adaptations to the real system, there would be considerable discrepancies between the behavior of the general simulation models and that of the concrete production system. For this reason, they cannot be used directly within the digital twin. Therefore,

efforts are being made to synchronize the behavior of the simulation model and the real production system so that the model can become part of a real "twin" of the concrete production system instance.

So far, simulation models for virtual commissioning are only used in the development phase of production systems to test the control code [61]. With regard to the digital twin, there are some publications in which discrepancies between the simulation model and the real system are investigated and compensated for specific aspects of the digital twin [62–65]. For this, the simulation models must be adapted to the reality of the concrete CPPS instances, whose behavior increasingly diverges from the idealized assumptions of these simulation models due to tolerances in the structure, environmental influences or wear—"as-operated". However, the issue of how to convert the "as-designed" models into "as-operated" models is not addressed. The initial "as-designed" simulation models, thus, increasingly deviate from the reality of the actual system behavior, whereby they lose their predictive capabilities and analyses of this leading to erroneous conclusions. During commissioning, the general simulation models of the system must therefore be transformed into instance models of a concrete, physically constructed system in order to continue enabling useful analyses. Currently, however, changes through adaptation or wear and tear to the (often long-lived) CPPS are not represented in the digital twins or in updates to their simulation models, so that control and simulation of the CPPS assumes outdated assumptions encoded in the models and information relevant to the development of future versions of the represented CPPS is lost [66]. This is essentially a modeling challenge [66, 67].

To solve this modeling challenge, a sustainable methodology needs to be developed to (partially) automatically transform the general simulation models of CPPS into precise instance models during commissioning. For this purpose, it is investigated how different modeling concepts, methods, and tools of model-driven software development, such as software language engineering [56, 68, 69], model transformations [70] and self-adaptive digital twins [9, 8, 44], can be suitably adapted and combined in a domain-specific way.

6 Related Work

In the development process of production plants, it has become common for domain experts to design digital twins as simulation models with the purpose to develop and test the control code of the plant at an early stage. The real control hardware and communication medium (field bus system) are used for this as a hardware-in-the-loop simulation. It is advantageous if the simulation models are calculated in the real-time cycle of the communication periphery in order to be able to carry out the most meaningful tests possible. This process is commonly referred to as virtual commissioning. In most cases, it is sufficient to model simple kinematics and material flow to positively influence control code development. However, this does not apply to the use of simulation models in an operational simulation. In some cases, highly

accurate simulation models are required that cannot be calculated in real time. Co-simulation has established itself for these cases [71]. There are also other potentials of these simulation models that go beyond pure virtual commissioning, such as control optimization by means of macroscopic material flow models [72] or the support of control development by means of AI [73–75]. These aspects make these simulation models interesting as a possible component of digital twins.

Recently, various digital twin platforms have emerged [76]. They aim at supporting developers in creating digital twins and digital shadows and provide tooling support for different aspects of a DT. Numerous DT platforms already exist from various vendors, e.g., by IBM,[4] Oracle,[5] Siemens,[6] Amazon Web Services[7] [77], Microsoft Azure[8] [78], and Eclipse.[9] We already compared the latter three platforms in terms of functionality [79]. All three of the investigated DT platforms provide comprehensive infrastructure for defining data structures for the data exchanged between physical and digital twins. For this purpose, they even provide modeling languages. However, models of these languages are not interoperable and restricted to their platform. Also, they miss standardized interfaces to enable composition of digital twins, or command models that enable interaction with other value-adding services, such as simulations or prediction. Because the platforms only provide modeling techniques for structural modeling, compared to our solution behavior, i.e., the communication, evaluation, and reasoning, of the DT cannot be defined in a model-based way and requires hand-written code to be deployed to the used cloud platform. This requires skilled software engineers, whereas our approach does not require programming skills from domain experts. Microsoft's and Amazon's DT platforms are part of their cloud ecosystems providing the advantage that their DTs can use machine learning capabilities, data processing algorithms, and CI/CD functionalities provided by their ecosystem. However, our approach to digital twins can be easily deployed to these platforms, enabling users to experience the advantages of these platforms. Our approach could be extended with interfaces to integrate features of the platforms.

[4] https://digitaltwinexchange.ibm.com/.

[5] https://docs.oracle.com/en/cloud/paas/iot-cloud/iotgs/oracle-iot-digital-twin-implementation.html.

[6] https://siemens.mindsphere.io/content/dam/cloudcraze-mindsphere-assets/03-catalog-section/05-solution-packages/solution-packages/digitalize-and-transform/Siemens-MindSphere-Digitalize-and-Transform-sb-72224-A8.pdf.

[7] https://aws.amazon.com/de/greengrass/.

[8] https://docs.microsoft.com/en-us/azure/digital-twins/overview.

[9] https://github.com/eclipse/vorto.

7 Conclusion

In this chapter, we have presented a collection of methods for the sustainable model-driven development of digital twins. These methods are based on our understanding of digital twins as software systems that comprise data, models, and services enabling interaction with a CPPS for specific purposes. Leveraging models as primary development artifacts supports (a) the sustainable engineering of digital twins as well as (b) the engineering of sustainable digital twins, i.e., to leverage monitoring, controlling, and optimizing the behavior of CPPSs through their digital twins. The methods can be applied to digital twins in a variety of application domains and shall guide researchers and practitioners in the conception, engineering, and operations of digital twins.

References

1. Bibow, P., Dalibor, M., Hopmann, C., Mainz, B., Rumpe, B., Schmalzing, D., Schmitz, M., & Wortmann, A. (2020). Model-driven development of a digital twin for injection molding. In *International conference on advanced information systems engineering (CAiSE'20)*. Springer.
2. Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11)
3. Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., & Nee, A. Y. C. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems 58*.
4. Chen, X., Kang, E., Shiraishi, S., Preciado, V. M., & Jiang, Z. (2018). Digital behavioral twins for safe connected cars. In *ACM/IEEE international conference on model driven engineering languages and systems*.
5. Kaewunruen, S., Xu, N. (2018). Digital twin for sustainability evaluation of railway station buildings. *Frontiers in Built Environment*, *77*.
6. Lauzeral, N., Borzacchiello, D., Kugler, M., George, D., Rémond, Y., Hostettler, A., & Chinesta, F. (2019). A model order reduction approach to create patient-specific mechanical models of human liver in computational medicine applications. *Computer Methods and Programs in Biomedicine*, *170*.
7. Verner, I., Cuperman, D., Gamer, S., & Polishuk, A. (2019). Training robot manipulation skills through practice with digital twin of Baxter.
8. Bolender, T., Bürvenich, G., Dalibor, M., Rumpe, B., & Wortmann, A. (2021). Self-adaptive manufacturing with digital twins. In *2021 International symposium on software engineering for adaptive and self-managing systems (SEAMS)* (pp. 156–166). IEEE Computer Society.
9. Feichtinger, K., Meixner, K., Rinker, F., Koren, I., Eichelberger, H., Heinemann, T., Holtmann, J., Konersmann, M., Michael, J., Neumann, E.-M., Pfeiffer, J., Rabiser, R., Riebisch, M., & Schmid, K. (2022). Industry voices on software engineering challenges in cyber-physical production systems engineering. In *2022 IEEE 27th International conference on emerging technologies and factory automation (ETFA)*. IEEE.
10. Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, *6*.
11. Khan, A., Turowski, K. (2016). A survey of current challenges in manufacturing industry and preparation for industry 4.0. In *Proceedings of the first international scientific conference on intelligent information technologies for industry (IITI'16)*, pp. 15–26.
12. Avventuroso, G., Silvestri, M., & Pedrazzoli, P. (2017). A networked production system to implement virtual enterprise and product lifecycle information loops. *IFAC-PapersOnLine, 50*(1), 2017.

13. Biesinger, F., Meike, D., Kraß, B., & Weyrich, M. (2019). A digital twin for production planning based on cyber-physical systems: A case study for a cyber-physical system-based creation of a digital twin. *Procedia CIRP 79*.
14. Eyre, J. M., Dodd, T. J., Freeman, C., Lanyon-Hogg, R., Lockwood, A. J., & Scott, R. W. (2018) Demonstration of an industrial framework for an implementation of a process digital twin. In *ASME international mechanical engineering congress and exposition* 52019.
15. Park, K. T., Nam, Y. W., Lee, H. S., Im, S. J., Noh, S. D., Son, J. Y., & Kim, H. (2019). Design and implementation of a digital twin application for a connected micro smart factory. *International Journal of Computer Integrated Manufacturing*, *32*(6).
16. Sharma, P., Hamedifar, H., Brown, A., & Green, R. (2017). The dawn of the new age of the industrial Internet and how it can radically transform the offshore oil and gas industry. In *Offshore technology conference*. OnePetro.
17. Dalibor, M., Jansen, N., Rumpe, B., Schmalzing, D., Wachtmeister, L., Wimmer, M., & Wortmann, A. (2022). A cross-domain systematic mapping study on software engineering for Digital Twins. *Journal of Systems and Software, 193*, 111361. Elsevier.
18. Brockhoff, T., Heithoff, M., Koren, I., Michael, J., Pfeiffer, J., Rumpe, B., Uysal, M. S., Van Der Aalst, W. M., & Wortmann, A. (2021). Process prediction with digital twins. In *ACM/IEEE international conference on model driven engineering languages and systems companion (MODELS-C)*. IEEE.
19. Knapp, G. L., Mukherjee, T., Zuback, J. S., Wei, H. L., Palmer, T. A., De, A., & DebRoy, T. J. A. M. (2017). Building blocks for a digital twin of additive manufacturing. *Acta Materialia 135*.
20. Braun, S., Dalibor, M., Jansen, N., Jarke, M., Koren, I., Quix, C., Rumpe, B., Wimmer, M., & Wortmann, A. (2023). Engineering digital twins and digital shadows as key enablers for industry 4.0. In B. Vogel-Heuser & M. Wimmer (Eds.), *Digital transformation: core technologies and emerging topics from a computer science perspective* (pp. 3–31). Berlin, Heidelberg: Springer.
21. France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *Future of Software Engineering (FOSE'07)*. IEEE.
22. Kriebel, S., Markthaler, M., Salman, K. S., Greifenberg, T., Hillemacher, S., Rumpe, B., Schulze, C., Wortmann, A., Orth, P., & Richenhagen, J. (2018). Improving model-based testing in automotive software engineering. In: *IEEE/ACM international conference on software engineering: software engineering in practice track (ICSE-SEIP)*
23. Selic, B. (2003). The pragmatics of model-driven development. *IEEE Software*, *20*(5).
24. Mayr, H. C., Michael, J., Shekhovtsov, V. A., Ranasinghe, S., & Steinberger, C. (2018). A model centered perspective on software-intensive systems. In *Enterprise modeling and information systems architectures (EMISA'18)*, CEUR 2097.
25. Hölldobler, K., Rumpe, B., & Wortmann, A. (2018). Software language engineering in the large: Towards composing and deriving languages. *Computer Languages, Systems & Structures*, *54*.
26. Becker, F., Bibow, P., Dalibor, M., Gannouni, A., Hahn, V., Hopmann, C., Jarke, M., Koren, I., Kröger, M., Lipp, J., Maibaum, J., Michael, J., Rumpe, B., Sapel, P., Schäfer, N., Schmitz, G. J., Schuh, G., & Wortmann, A. (2021). A conceptual model for digital shadows in industry and its application. In: *Conceptual modeling*, ER'21. Springer.
27. Riesener, M., Schuh, G., Dölle, C., & Tönnes, C. (2019). The digital shadow as enabler for data analytics in product life cycle management. *Procedia CIRP 80*.
28. DIN ISO 55000:2017-05, Asset Management—Übersicht, Leitlinien und Begriffe.
29. Spec, D. I. N. (2016). 91345: Reference architecture model industrie 4.0 (rami4. 0). *DIN Std. DIN SPEC* 91.345.
30. Landherr, M., Schneider, U., & Bauernhansl, T. (2016). The application center industrie 4.0-industry-driven manufacturing, research and development. *Procedia Cirp, 57*, 26–31. Elsevier.
31. Heithoff, M., Michael, J., & Rumpe, B. (2022). Enhancing digital shadows with workflows. In *Modellierung 2022 satellite events* (pp. 142–146). Gesellschaft für Informatik e.V. https://doi.org/10.18420/modellierung2022ws-017
32. Ríos, J., Staudter, G., Weber, M., Anderl, R., & Bernard, J. (2020). Uncertainty of data and the digital twin: A review. *International Journal of Product Lifecycle Management, 12*(4), 329–358.

33. Michael, J., Nachmann, I., Netz, L., Rumpe, B., & Stüber, S. (2022). Generating digital twin cockpits for parameter management in the engineering of wind turbines. In *Modellierung 2022* (pp. 33–48). Gesellschaft für Informatik.
34. Purvis, B., Mao, Y., & Robinson, D. (2019). Three pillars of sustainability: In search of conceptual origins. *Sustainability Science, 14*, 681–695.
35. Brown, B. J., Hanson, M. E., Liverman, D. M., & Merideth, R. W. (1987). Global sustainability: Toward definition. *Environmental Management, 11*.
36. Macnaghten, P., & Jacobs, M. (1997). Public identification with sustainable development: Investigating cultural barriers to participation. *Global Environmental Change*, *7*(1).
37. UN: Transforming our world: The 2030 Agenda for sustainable development. Resolution adopted by the general assembly on 25 September 2015. https://sdgs.un.org/2030agenda
38. Tagliabue, L. C., Cecconi, F. R., Maltese, S., Rinaldi, S., Ciribini, A. L. C., & Flammini, A. (2021). Leveraging digital twin for sustainability assessment of an educational building. *Sustainability*.
39. Zaballos, A., Briones, A., Massa, A., Centelles, P., & Caballero, V. (2020). A smart campus' digital twin for sustainable comfort monitoring. *Sustainability*, *12*.
40. Barni, A., Fontana, A., Menato, S., Sorlini, M., & Canetta, L. (2018). Exploiting the digital twin in the assessment and optimization of sustainability performances. In *International Conference on Intelligent Systems (IS)*.
41. Li, L., et al. (2020). Sustainability assessment of intelligent manufacturing supported by digital twin. *IEEE Access 8*
42. Riedelsheimer, T., Dorfhuber, L., & Stark, R. (2020). User centered development of a digital twin concept with focus on sustainability in the clothing industry. *Procedia CIRP 90*.
43. Stahl, T., Völter, M., & Czarnecki, K. (2006). *Model-driven software development: Technology, engineering, management*. Wiley.
44. Ringert, J. O., Rumpe, B., Wortmann, A. (2014). Architecture and behavior modeling of cyber-physical systems with MontiArcAutomaton. In Aachener Informatik-Berichte, Software Engineering, Band 20. ISBN 978-3-8440-3120-1. Shaker Verlag.
45. Rumpe, B. (2017). *Agile modeling with UML: Code generation, testing, refactoring*. Springer.
46. Leitner, S. H., & Mahnke, W. (2006). OPC UA–service-oriented architecture for industrial applications. *ABB Corporate Research Center, 48*(61–66), 22.
47. Bano, D., Michael, J., Rumpe, B., Varga, S., & Weske, M. (2022). Process-aware digital twin cockpit synthesis from event logs. *Journal of Computer Languages (COLA)*, *70*.
48. Adam, K., Michael, J., Netz, L., Rumpe, B., & Varga, S. (2020). 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'19)*, LNI 304, GI.
49. Dalibor, M., Michael, J., Rumpe, B., Varga, S., & Wortmann, A. (2020). Towards a model-driven architecture for interactive digital twin cockpits. In *Conceptual modeling*. Springer. https://doi.org/10.1007/978-3-030-62522-1_28
50. van der Aalst, W. M. P. (2016). *Process mining*. Springer.
51. Dalibor, M., Heithoff, M., Michael, J., Netz, L., Pfeiffer, J., Rumpe, B., Varga, S., & Wortmann, A. (2022). Generating customized low-code development platforms for digital twins. *Journal of Computer Languages (COLA)*, *70*.
52. Michael, J., & Wortmann, A. (2021). Towards development platforms for digital twins: A model-driven low-code approach. In *IFIP advances in information and communication technology, advances in production management systems. Artificial intelligence for sustainable and resilient production systems*. Springer.
53. Knublauch, H., Oberle, D., Tetlow, P., Wallace, E., Pan, J. Z., & Uschold, M. (2006). A semantic web primer for object-oriented software developers. *W3c working group note, W3C*.
54. Eastman, R. D., Schlenoff, C. I., Balakirsky, S. B., & Hong, T. H. (2013). A sensor ontology literature review. NISTIR 7908.
55. Michael, J., Rumpe, B., & Varga, S. (2020). Human behavior, goals and model-driven software engineering for assistive systems. In *Enterprise modeling and information systems architectures (EMSIA'20)*, CEUR 2628.

56. Hölldobler, K., Michael, J., Ringert, J. O., Rumpe, B., & Wortmann, A. (2019). Innovations in model-based software and systems engineering. *The Journal of Object Technology*, *18*(1), AITO.
57. Michael, J. (2022). A vision towards generated assistive systems for supporting human interactions in production. In *Modellierung 2022 satellite events* (pp. 150–153). Gesellschaft für Informatik e.V.
58. Steinberger, C., & Michael, J. (2020). Using semantic markup to boost context awareness for assistive systems. In *Smart assisted living: toward an open smart-home infrastructure*. Springer.
59. Armendia, M., Cugnon, F., Berglind, L., Ozturk, E., Gil, G., & Selmi, J. (2019). Evaluation of machine tool digital twin for machining operations in industrial environment. *Procedia CIRP 82*.
60. Verein Deutscher Ingenieure e.V. u. Verband der Elektrotechnik Elektronik Informationstechnik e.V.: Virtuelle Inbetriebnahme—Einführung der virtuellen Inbetriebnahme in Unternehmen, VDI/VDE 3693 Blatt 2. Beuth Verlag, 2018.
61. Pritschow, G., Röck, S. (2004). "Hardware in the Loop" simulation of machine tools. *CIRP Annals 53*(1).
62. Kain, S., Dominka, S., Merz, M., & Schiller, F. (2009). Reuse of HiL simulation models in the operation phase of production plants. In *International Conference on Industrial Technology (ICIT'09)*. IEEE.
63. Talkhestani, B. A., Jazdi, N., Schlögl, W., & Weyrich, M. (2018). A concept in synchronization of virtual production system with real factory based on anchor-point method. *Procedia CIRP, 67*, 13–17.
64. Wei, Y., Hu, T., Zhou, T., Ye, Y., & Luo, W. (2021). Consistency retention method for CNC machine tool digital twin model. *Journal of Manufacturing Systems*.
65. Zipper, H., Diedrich, C. (2019). Synchronization of industrial plant and digital twin. In *International conference on emerging technologies and factory automation (ETFA)*. IEEE, pp. 1678–1681.
66. Bucchiarone, A. et al. (2021). What is the future of modeling? *IEEE Software*, *38*(2).
67. Wortmann, A., Barais, O., Combemale, B., & Wimmer, M. (2020). Modeling languages in industry 4.0: an extended systematic mapping study, *Software and System Modeling, 19*(1), 67–94.
68. Butting, A., & Wortmann, A. (2021). Language engineering for heterogeneous collaborative embedded systems. In *Model-based engineering of collaborative embedded systems* (pp. 239–253). Springer.
69. Gupta, R., Kranz, S., Regnat, N., Rumpe, B., & Wortmann, A. (2021). Towards a systematic engineering of industrial domain-specific languages. In *IEEE/ACM international workshop on software engineering research and industrial practice (SER&IP)*.
70. Kai, A., Hölldobler, K., Rumpe, B., & Wortmann, A. (2017). Modeling robotics software architectures with modular model transformations. *Journal of Software Engineering for Robotics (JOSER), 8*(1).
71. Scheifele, C., Verl, A., Riedel, O. (2019). Real-time co-simulation for the virtual commissioning of production systems. *Procedia CIRP*, *79*
72. Kienzlen, A., Weißen, J., Verl, A., Göttlich, S. (2020). Simulative Optimierung der Steuerungsparameter eines Materialflusslayouts mit Bandförderern. Forschung im Ingenieurwesen.
73. Jaensch, F., Csiszar, A., Kienzlen, A., & Verl, A. (2018). Reinforcement learning of material flow control logic using hardware-in-the-loop simulation. In *International conference on artificial intelligence for industries (AI4I)*.
74. Jaensch, F., Csiszar, A., Scheifele, C., & Verl, A. (2018). Digital twins of manufacturing systems as a base for machine learning. In *International conference on mechatronics and machine vision in practice (M2VIP)*.
75. Jaensch, F., Csiszar, A., Scheifele, C., & Verl, A. (2019). Reinforcement learning of a robot cell control logic using a software-in-the-loop simulation as environment. *In International conference on artificial intelligence for industries (AI4I)*.

76. Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., & Nee, A. Y. C. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, *58*.
77. Kurniawan, A. (2018). Learning AWS IoT: Effectively manage connected devices on the AWS cloud using services such as AWS Greengrass, AWS button, predictive analytics and machine learning. Packt Publishing Ltd.
78. Klein, S. (2017). *IoT solutions in Microsoft's azure IoT Suite.* Apress.
79. Lehner, D., Pfeiffer, J., Tinsel, E. F., Strljic, M. M., Sint, S., Vierhauser, M., Wortmann, A., & Wimmer, M. (2021). Digital twin platforms: Requirements, capabilities, and future prospects. *IEEE Software.*