Systems Engineering

WILEY

COMMUNICATION OPEN ACCESS

Model-Driven Engineering for Digital Twins: Opportunities and Challenges

Judith Michael[1] | Loek Cleophas[2,3] | Steffen Zschaler[4] | Tony Clark[5] | Benoit Combemale[6] | Thomas Godfrey[4] | Djamel Eddine Khelladi[7] | Vinay Kulkarni[8] | Daniel Lehner[9] | Bernhard Rumpe[1] | Manuel Wimmer[9] | Andreas Wortmann[10] | Shaukat Ali[11] | Balbir Barn[12] | Ion Barosan[2] | Nelly Bencomo[13] | Francis Bordeleau[14] | Georg Grossmann[15] | Gabor Karsai[16] | Oliver Kopp[10] | Bernhard Mitschang[10] | Paula Muñoz Ariza[17] | Alfonso Pierantonio[18] | Fiona A. C. Polack[19] | Matthias Riebisch[20] | Holger Schlingloff[21] | Markus Stumptner[15] | Antonio Vallecillo[17] | Mark van den Brand[2] | Hans Vangheluwe[22,23]

[1]RWTH Aachen University, Aachen, Germany | [2]TU Eindhoven, Eindhoven, the Netherlands | [3]Stellenbosch University, Stellenbosch, South Africa | [4]King's College London, London, UK | [5]Aston University, Birmingham, UK | [6]Université de Rennes, Rennes, France | [7]CNRS, Université de Rennes, Rennes, France | [8]TATA Consulting, Pune, India | [9]JKU Linz, Linz, Austria | [10]Universität Stuttgart, Stuttgart, Germany | [11]Simula Research Laboratory, Oslo, Norway | [12]Middlesex University, London, UK | [13]Durham University, Durham, UK | [14]ETS, Montréal, Canada | [15]University of South Australia, Adelaide, Australia | [16]Vanderbilt University, Nashville, Tennessee, USA | [17]University of Málaga, Málaga, Spain | [18]University of L'Aquila, L'Aquila, Italy | [19]University of Hull, Hull, UK | [20]Universität Hamburg, Hamburg, Germany | [21]HU Berlin, Fraunhofer FOKUS, Berlin, Germany | [22]University of Antwerp, Antwerp, Belgium | [23]Flanders Make, Lommel/Leuven/Kortrijk, Belgium

Correspondence: Judith Michael (michael@se-rwth.de)

Received: 15 March 2024 | **Revised:** 3 February 2025 | **Accepted:** 20 March 2025

Funding: This work was partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) and the Agence Nationale De La Recherche (ANR)—France—Model-Based DevOps—505496753 and ANR-22-CE92-0068, and the Key Digital Technologies (KDT) Joint Undertaking through the European Union's Horizon Europe project MATISSE, grant agreement No. 101140216.

Keywords: cyber-physical systems | digital twin | model-driven engineering | systems engineering

ABSTRACT

Digital twins are increasingly used across a wide range of industries. Modeling is a key to digital twin development—both when considering the models which a digital twin maintains of its real-world complement ("models *in* digital twin") and when considering models *of* the digital twin as a complex (software) system itself. Thus, systematic development and maintenance of these models is a key factor in effective and efficient digital twin development, maintenance, and use. We argue that model-driven engineering (MDE), a field with almost three decades of research, will be essential for improving the efficiency and reliability of future digital twin development. To do so, we present an overview of the digital twin life cycle, identifying the different types of models that should be used and re-used at different life cycle stages (including systems engineering models of the actual system, domain-specific simulation models, models of data processing pipelines, etc.). We highlight some approaches in MDE that can help create and manage these models and present a roadmap for research towards MDE of digital twins.

1 | Introduction

Digital twins are among the key drivers of advanced manufacturing [1]. Recent surveys [2, 3] have identified a wide range of application areas and technological approaches to building digital twins. Digital twins are software systems that provide services on top of virtual representations of actual systems. They typically enhance actual systems into cyber-physical systems by providing

mechanisms for data collection, (partial) modeling, planning, and decision-making, and changing the state and behavior of the actual system. Thus, digital twins are complex software systems, raising questions about how they can be engineered both efficiently and effectively.

In research and practice, there is no single definition of a digital twin although all leading definitions have common features, such as sending and receiving data from the actual system or providing a virtual representation [4]. To navigate the design options for twins, it is important for digital twin engineers to have conceptual models [5] and reference models that capture such twins' essential features and variation points and to collaborate with data scientists, as well as domain, system, and software engineers. To date, there has been little attention paid in research and practice to digital twin development methods. We argue that models from software and systems engineering can be used to define methods that are used to guide development teams and could be used as the basis of digital twin development platforms [6].

Digital twins manage models of the actual system and offer services including, but not limited to *analysis*, *diagnosis*, *prediction*, *fault detection*, and *planning*. Digital twins could cover different capabilities [7]: (1) data fusion from sensors to detect properties; (2) consolidation of properties to populate models; (3) the use of model snapshots to provide dashboards and detect situations; (4) projection of situations to speculate about future states (simulation); (5) prediction of future situations from historical information (learning); (6) adaptation that provides decision-support (AI); up to (7) partial and total adaptive control. Each type of service will place requirements on the representation and processing of the system models.

The field of Model-Driven Engineering (MDE) has been argued to support the efficient and effective development, deployment, and maintenance of software systems in general [8]. MDE could be used throughout the digital twin life cycle to *specify* and *design* a twin in order to provide a (possibly executable) prototype. These models might be used to automatically generate parts (or even all) of the twin, including configuring its connections to the actual system [6, 9]. An alternative might be that the twin uses models at run-time to execute the models directly in order to improve its adaptability or its ability to provide feedback.

This article reviews the opportunities for MDE to support the field of digital twins. It is organized as follows: Section 2 shows key aspects of digital twins and describes the state-of-the-art for their contexts and life cycle; Section 3 sketches challenges around models and modeling in digital twins; Section 4 provides an overview of the key aspects of MDE relevant to the digital twin life cycle; finally, Section 5 performs a gap analysis leading to a research roadmap.

2 | Digital Twins

To understand where digital twin engineering can be supported by the field of MDE, we first describe what parts constitute digital twins, explain their life cycle, and describe relevant digital twin contexts.

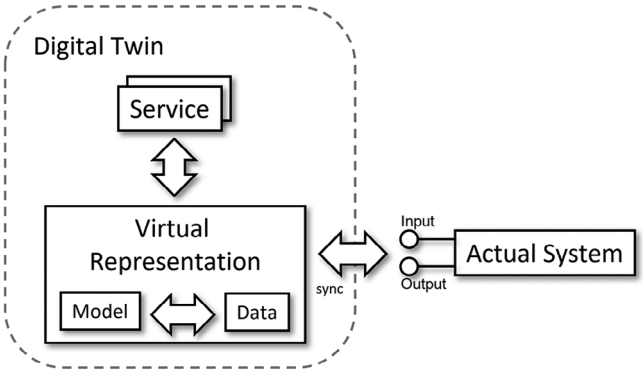

FIGURE 1 | Digital twin schema derived from Tao et al. [11, 12].

2.1 | What is a Digital Twin

There are many different definitions of "digital twins" in the literature [2] and recent standards such as [10], focusing on different aspects. For the purposes of this paper, we use the following conceptualization based on the 5D model from Tao et al. [11]. This conceptualization differentiates three key components (see Figure 1):

1. The *actual system*, which is a system or object in the real world. Note that our understanding includes a wide range of systems and objects, including socio-technical or biological systems. The actual system offers interfaces for *output* (that is, sensing/extracting data about the state of the actual system) and *input* (that is, controlling the state of the actual system either directly or by supporting decision making processes of human controllers).
2. A *virtual representation* of the *digital twin*, which includes *models* and connected *data* digitally representing the configuration and state of the actual system.
3. A set of *services* provided on top of the information captured in the digital twin, which includes, for example, services for the synchronization of selected properties with the actual system at a defined synchronization rate, for reasoning about information from the virtual representation, and visualization and reporting information to digital twin users [4].

A digital twin is a software system, that includes the virtual representation of the actual system with models and data, and the services needed to fulfill the purpose a digital twin serves.

To realize the software system representing a digital twin and its services, industry and research suggest different reference architectures, for example, by the Digital Twin Consortium [13], the standard series ISO-23247 for manufacturing [14, 15], Reference Architecture Model Industry 4.0 (RAMI 4.0) [16], semantic Asset Administration Shells [17], or more conceptual architectures such as the 3D model [18], the 5D model [11, 12], or the work by Newrzella et al. [19], Kritzinger et al. [20], Josifovska et al. [21], Boyes and Watson [22], or Eramo et al. [5]. These architectures

provide an overview of different functionalities a digital twin could cover. However, each concrete, domain-specific digital twin's software architecture has to be tailored for the purposes it aims to cover.

2.2 | Digital Twin Contexts

Digital twins can be designed for different categories of actual systems. The actual system and its environment constitute the *context* of the digital twin. There are fundamental properties that characterize the context: whether the actual system is an engineered or a nonengineered system, whether it is a controlled or noncontrolled system and whether it contains inanimate or animate components or human beings. Based on these properties, we can distinguish three main categories of actual systems which may form the context:

1. **Engineered systems** are systems whose design and realization have been carefully planned. We differentiate three types of such systems, based on whether they have been developed with digital control in mind:
   a. **Cyber-physical (including embedded) systems** are technically engineered products or groups of products comprising physical components and computational control devices. An example could be a robot, a modern car, a wind turbine, an air plane, a production line, or a power plant. Crucially, often they already have digital components that provide an interface for a digital twin to interact with the actual system.
   b. **Technical systems without software**. This category contains artefacts such as tables, steam engines, classical bicycles, hydraulic presses, windmills, bridges, buildings, and so forth. These systems are engineered, but do not have a software component (yet). To allow the creation of digital twins, we need to enhance the system with digital interfaces for data collection and for detecting changing states and behaviors.
   c. **Software** systems sit somewhere halfway between the previous two. Software is naturally digital, making the creation of a digital twin of the software easier in principle. However, the information captured digitally is not typically the information one wants to reflect in the digital twin, necessitating the creation of additional interfaces (e.g., for the monitoring of build processes).
2. **Biological/natural systems** which are not designed by human engineers and can exist without human intervention, such as biological organisms and ecosystems, for example, trees, forests, plantations, animal populations, or weather phenomena. Here, creating digital interfaces is much more challenging and often means a transformation from pure biological to bio-technical systems.
3. **Socio-technical systems and processes, including organizations**, that is, systems that involve human behavior and interaction between humans and technical systems. An example could be a hospital, the flow of patients through a healthcare system, or the organizational structure of a company. They might already provide digital interfaces for some aspects, for example, technical devices, but also require to add new ones, for example, to detect human behavior.

These different categories each come with their own requirements as well as broader aims, such as supporting sustainability, privacy, or maintainability.

2.3 | Digital Twin Life Cycle

The life cycle of a digital twin is deeply connected to the life cycle of the actual system it is twinning. The life cycle of the digital twin and its actual system does not have to run in lockstep. For example, consider digital twins used for simulation-based prediction during system design: the life cycle of such a digital twin will be fully completed alongside the design phase of the actual system and the digital twin may no longer be used once the actual system has been constructed and is in operation. Figure 2 shows the life cycle of some examples for such actual systems and some examples for digital twins.

For an *engineered object*, the life cycle includes the design, construction, operation, and maintenance until the end of life. A digital twin of a product line can accompany the whole life cycle starting with, for example, engineering models created during model-based systems engineering [23]; simulation and forecasting services; simulation data in the design; continuous updating of data and models with the reality during construction, monitoring, and forecasting during operation until their demolition or refurbishment during the end-of-life phase. The same phases occur for a concrete production machine or one concrete product produced in such a production machine. However, in relation to the life cycle of the product line, their life cycles start later.

The life cycle of *biological systems* such as humans, plants, or animals includes the (optional) plan, growth, life, and end-of-life phase. If we take a biological-system experiment as an example—for example, to create a lab-grown ear—a digital twin already accompanies the planning of the experiment and uses services to check the plans with regulators, observe the ear during its growing phase, and regulate the environment of the ear to provide better growth. After transfer to a human, the ear might be further observed and stimulated to determine whether this experiment is successful or not. The same life cycle occurs for other biological processes such as 3D bio-printing or in vitro fertilization. Furthermore, a digital twin could accompany humans or organs of humans such as a digital twin for the brain [24], a human heart [25], or lungs. For sensing and actuating a biological system synchronized with its digital twin, digital twin developers might consider the whole bio-technical system including relevant technological solutions as the actual system.

The life cycle of *software systems* is similar to the one of engineered objects. In waterfall-like development processes, developers start with the analysis and design of a system, implement, generate, operate, re-engineer, stop, or replace the system in the end-of-life phase. In agile processes, the first three phases are continuously repeated for each software product increment. During the whole life cycle, a digital twin could accompany such a software system. If we create a software product line (SPL)—used for developing families of similar software products—we could have a digital twin for the SPL reference design and implementations as well as each concrete realization of the SPL in an individual software product.

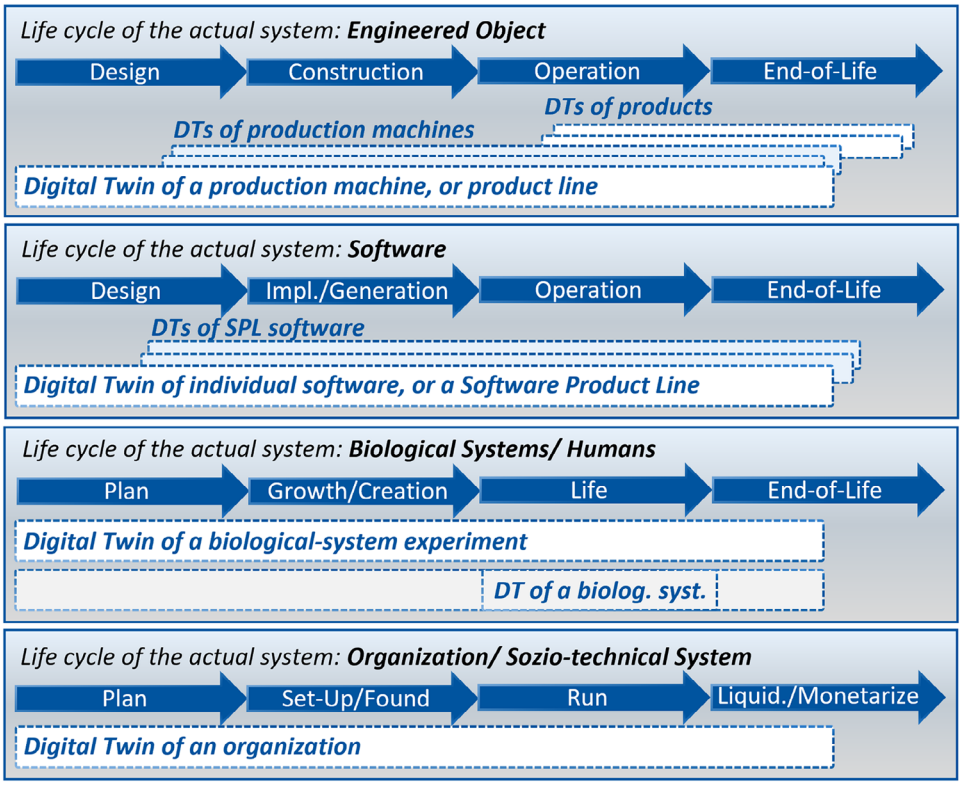

FIGURE 2 | Digital twins for different types of actual systems and their life cycle.

The life cycle of socio-technical systems such as an *organization* can be summarized as planning, set-up or founding an organization, running it, and liquidating or monetizing it. If a structured planning phase exists, for example, in start-up incubators, if founders start by elaborating a business plan, or if a crowdfunding campaign should support the funding of a company, a corresponding digital twin could already exist from the beginning. If a company already exists, the digital twin might only exist during the operation of a company, for example, to monitor business processes or key performance indicators.

Changes in the actual system and its environment may lead to re-adjustments and changes in the digital twin, such as adding new services, handling new types of models, or changing data structures and visualization needs. These changes lead to the need for continuous DevOps [26] cycles during the lifetime of the actual system and its digital twin.

3 | Challenges Around Models and Modeling in Digital Twins

The previous section presented typical elements of digital twins and their life cycle, highlighting some of the challenges in developing and maintaining digital twins. Importantly, digital twins include *models* (in the form of digital representations that can be used to derive insights) of some actual system. This section examines these models in detail: what are the different types of models involved in a digital twin's life cycle and how can making these models explicit help us engineer and maintain digital twins more efficiently and effectively.

Models are more than data: They capture structure and provide context for the interpretation of data. Digital twins include data gathered from the actual system and models that support the interpretation and analysis of these data. Some models are derived from data—either a posteriori from historical data or (updating) dynamically. Other models are provided as a priori descriptions/prescriptions [27] independent of data, though they may be parameterized with data from the actual system. Where information cannot be easily measured or computed, one can use approximation models, also known as surrogate models. These aim to learn functional relationships between data inputs and outputs statistically, but can sometimes lead to a lack of explainability. Models are on the knowledge and information planes of the well-known data–information–knowledge hierarchy [28, 29].

In our discussion, we make a fundamental distinction between two types of models as follows:

1. *Models* in *digital twins* are models of the actual system that are maintained inside the digital twin. They capture information about the actual system and enable analysis, simulation, inspection, and so forth. These models can be descriptive, predictive, or prescriptive (see [5]).
2. *Models* of *digital twins*, on the other hand, consider the digital twin *system* as a complex software-intensive system and capture information about the structure, behavior, and state of the digital twin system.

The following subsections discuss each of these two types in turn.

3.1 | Models *in* Digital Twins

These are the models we often think of first when considering a digital twin—after all, models of an actual system are an integral part of every digital twin. Considering them as explicit artefacts, rather than as an implicit model that perhaps *emerges* from data in a database together with a particular analysis algorithm (say, a deep-learning-based classifier), allows us to think about their structure, purpose, and life cycle more deeply.

3.1.1 | Kinds of Models

Digital twins include different kinds of models. In particular, modeling experts differentiate structural and behavioral models as follows:

1. *Structural models*, which capture the logical structure of the actual system—that is, its subsystems, components, and actors. Examples are, for example, structural models of railwork networks, citizen energy communities, or production plants. Structural models can be expressed in general purpose modeling languages—for example, Systems Modeling Language (SysML) block definition diagrams—as well as domain-specific modeling languages (DSMLs), for example, domain-specific architectural component-and-connector diagrams.
2. *Behavioral models*, which capture key processes and interactions in the actual system and between the actual system and its environment. A typical modeling language for such processes in the business domain is the Business Process Model and Notation (BPMN) standard. In the systems engineering domain, state machines are a familiar modeling language, but more domain-specific models also exist.

Orthogonal to these two kinds, modeling experts also structure models based on properties of the system engineers aim to develop models for engineered or natural systems. These models could have structural and behavioral aspects.

1. *Models for engineered systems* capture detailed mechanisms, the tangible structure, and behavior of these systems. Examples include CAD models, building information models (BIM) in the construction domain, geographic information system (GIS) models (for example of the geographical layout of a railway network), or SysML models describing the structure and behavior of a production machine.
2. *Models for natural systems* aim to describe the structure and behavior of these existing systems. Examples include mathematical models (e.g., ODE-style descriptions of chemical reaction networks), models describing the topology and geographical structure of the world, or weather models as simulations of the future state of the atmosphere through time.

Models in digital twins can be captured in general-purpose modeling languages but also in highly DSMLs [30, 31]. This broad variety of languages raises challenges around integrating models captured in different modeling languages, created by experts in different organizational units, or focused on different aspects of the actual system, ensuring their consistency and coherent evolution over time.

Systems of interest for representation through a digital twin are often systems of systems [32]. Different subsystems in such a system of systems are controlled by different organizations or actors, making the idea of a single, centrally managed digital twin of the complete system unrealistic. Instead, for a practical realization of digital twins, there is a need for federating digital twins, reflecting the compositional construction of the system being twinned and making the digital twin itself a system of systems [33]. In such a situation, each of the "sub-twins" will have its own ways of referring to key elements of the system being represented. Centrally aligning these model concepts to a commonly agreed ontology is usually difficult because of the nature of distributed organizational control in the actual system. Instead, digital twin engineers need to be able to map representations onto each other. Capturing the decisions we make in such a mapping explicitly is important for quality assurance purposes.

3.1.2 | Purpose

Every model is constructed for a purpose [34]. The purpose determines what aspects of reality we include in the model and what we abstract away. This abstraction, in turn, determines what the model can be used for and where its limitations lie. For example, a simulation model created for analyzing the flow of streams of passengers through an airport may not have enough information to analyze individual passenger decision making.

This focus on a purpose and different abstractions is interesting in the context of digital twins, where the lifetime can be long. The questions digital twin users ask of a digital twin—that is, the digital twin use cases—are likely to change, leading to a slow change in the purpose of the digital twin over time. The services in the digital twin need to evolve with this change. As a result, the models underlying these services may no longer be sufficiently accurate or include all the information required and may need to change, too. This need for change creates challenges around maintainability and consistency: large, monolithic models will be more difficult to evolve consistently, efficiently, and systematically [35]. Even the modeling tools and formalisms may change over time, given the potentially very long lifetimes of digital twins and their actual systems. Equally, evolving models will likely change the alignment between models and the underlying data. One solution attempt might be to collect as much data as possible and speculatively integrate as much of it as possible into models even if this information is not required for the current purpose. This solution attempt may provide flexibility when the purpose changes (assuming changes in purpose can be predicted reasonably well), but will make the use of the models more difficult for the initial purpose.

For digital twins to be used as decision-support systems (for example, in socio-technical digital twins [36]) or even more autonomously (as in the definition of digital twins proposed by Kritzinger et al. [20]), digital twin users must be able to

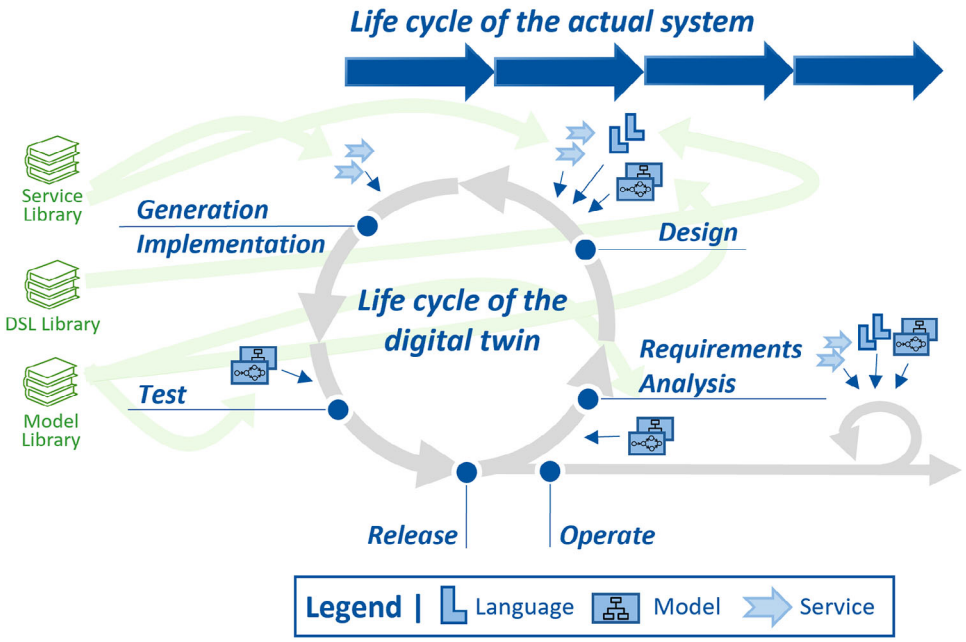

FIGURE 3 | Life cycle of a digital twin in detail.

trust them [37]. This means digital twin users must be able to understand how the digital twin has arrived at its decisions and recommendations [38] and that the models inside the digital twin are sufficiently accurate representations of the actual system for the recommendations and decisions to be appropriate. Simulation is a core service provided by digital twins and fundamental to its planning and decision-making capabilities. In the context of engineered systems, trust in models and simulations has been discussed under the labels of "experimental" or "validity" frame [39, 40]. These take the perspective that the essential validation required is whether a simulation can reproduce the input–output behavior of an actual system to a desired level of accuracy. In the context of complex systems (including biological and socio-technical systems), research suggests [41] that a richer set of information is required to ensure trust in the model, linking modeling decisions to the scientific literature, real-world experimentation, expert decision-making, and so forth. In particular, this approach suggests that we need to be able to trust the structure of the models and not just their input–output behavior. Establishing trust at this level requires making a structured *argument* [42, 43], similar to how one might construct a safety-assurance case. Where digital twins integrate artificial intelligence, in particular, machine learning, similar arguments are required to ensure trust [44].

3.2 | Models *of* Digital Twins

Here, the focus is on some of the modeling concerns relevant as part of the development of a digital twin as a complex software system. This development can be connected to the model-based systems engineering process [45], however, it is a process on its own requiring agile development methods.

Figure 3 takes a closer look at the continuum formed by the whole life cycle of a digital twin, as supported by DevOps principles, that is, the requirements analysis, design, generation and implementation, test, release, and operate phase. This continuum enables the seamless transition from one step to another in the life cycle. From a software engineering point of view, a high degree of re-use is to be aimed for; libraries of models, dDSMLs, and digital twin services foster such re-use (the green arrows in Figure 3 show the library re-use).

Digital twin engineers collect relevant requirements for the digital twin and create requirements models within the requirements analysis phase of a digital twin. This collection can include functional requirements of the digital twin to be built up to specific user needs and wishes. The requirements are then used to design the digital twin. In the design phase, we plan the first version of the digital twin and create design models such as for the system architecture or data structure, select which languages should be available in the digital twin during its runtime, and select which services from the service library to include in the digital twin. The production code and test code that uses the selected digital twin services and functionalities are derived from the design models by hand or using generative techniques as in Model-Driven Engineering (see Section 4). After successful tests, the digital twin can be released and start its operation. The operation of the digital twin is characterized by the continuous monitoring of the actual system and its context.

3.3 | Models Across the Digital Twin Life Cycle

As digital twins exist for long-living actual systems, for example, up to 20 years for injection molding machines or wind-turbine parks, and up to 50 years for bridges or buildings, it is clear that requirements for the actual system and the digital twin of it, and its context may change massively. This continuous need to react to changes requires an engineering approach with continuous

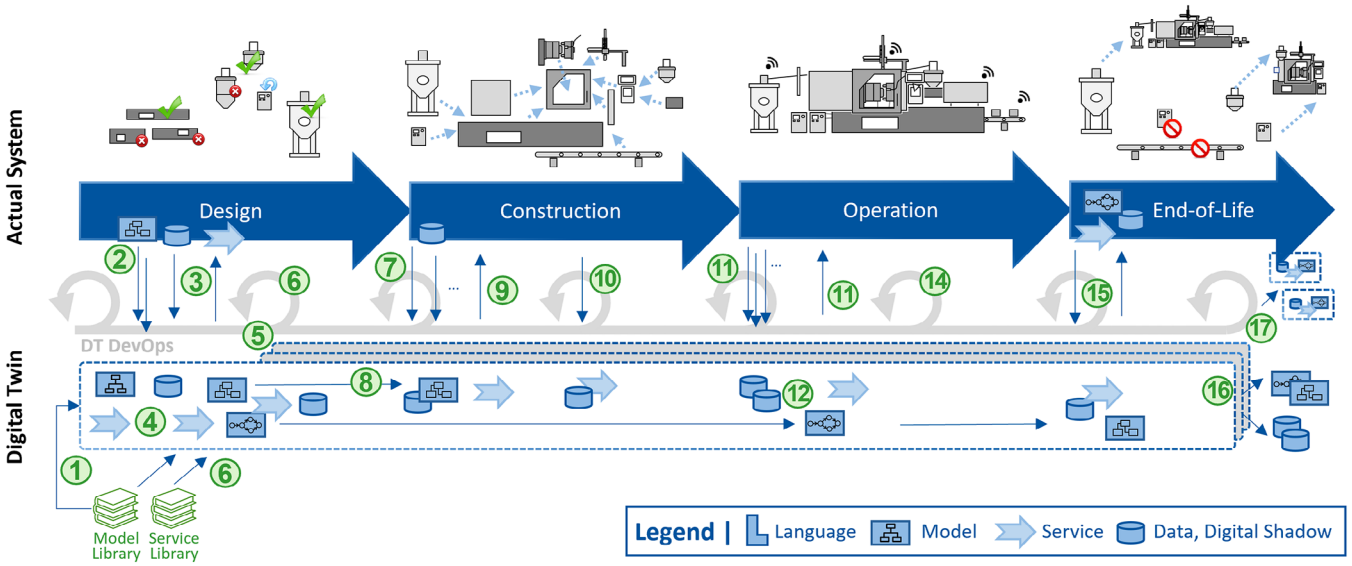

FIGURE 4 | Life cycle of the actual system accompanied by the life cycle of its digital twin.

DevOps cycles for the digital twin which influences both models in digital twins and models of digital twins.

Since these continuous DevOps cycles of the digital twin are triggered by changes in the system or its context, we are focusing on these changes in the ongoing relationship between the actual system and the digital twin. Figure 4 takes a closer look into the four main phases of the life cycle of an engineered actual system: design where the actual system is planned; construction where it is built; operation where it is used; and end-of-Life where it is refurbed, demolished, and recycled. In an ideal world, all of these phases are accompanied by a digital twin that continuously transforms from a design digital twin to a construction digital twin, or monitoring digital twin. This transformation requires continuous DevOps cycles for the digital twin (the DevOps cycle from Figure 3 is shown as continuous gray loops below the actual system in Figure 4).

Already at the beginning of the design phase of the actual system (1) in Figure 4, we have to engineer the digital twin by re-using models of the actual system and digital twin services as discussed in Figure 3, to have a digital twin already available when the design of the actual system starts.

Depending on the type of actual system represented by the digital twin, models of the system may already play a role independently of the existence of a digital twin. In particular, where the actual system is an engineered system (including, for example, production line systems), models are likely to exist that were used in *developing* the actual system. To ensure high fidelity of the digital twin as well as efficient development of the digital twin, it is helpful for digital twin engineers to be able to re-use these models directly inside the digital twin. However, re-use can be challenging, as it requires the models to be kept consistent with the actual system throughout its design and implementation. It also requires the definition of mechanisms for how elements in the model connect to elements in the actual system. This connection is a general requirement and may become easier where design models are re-used, as the relevant traceability information may already have been captured.

In the design phase ((2) in Figure 4), models about the actual system are created, for example, systems engineering models. These models are continuously updated in the digital twin as they evolve in reality over time. Moreover, (3) data about the engineered system is created in third-party applications—for example, sets of parameters in simulation and optimization software—and shared with the digital twin and communicated back in other stages of the design phase. The digital twin can handle data and models as it (4) includes services—for example, for data and parameter visualization, simulation, or model validation.

The engineering process of the digital twin has to take into account that the design process of the actual system often starts with the design of objects which are then realized not in lots of size 1 but where several actual systems are constructed and operated. Thus, multiple digital twins have to be realized, based on the same initial set of engineering models, data and services at the beginning, which are then adapted to the actual context during construction. When the planning of a product line moves on to a concrete engineered product, (5) we can transfer the planning data and models of a product line digital twin to the digital twins of concrete production machines. Those are then further detailed during their design phase and receive updated models, and data.

Changes in this design process or changing requirements might make it necessary to (6) start a new dev-cycle of the digital twin and, for example, update existing models of digital twins, or include additional models or services from the libraries.

Complex actual systems take some time to be actually constructed—from several months up to several years for production lines and factories. In this construction phase, we receive (7) data on the construction process and have to (8) update models if the planned ones deviate from reality. Based on

such changes, engineers might have to 9 update parameters in the cyber-part of an actual cyber-physical object.

As in the design phase, also in this phase changing management, changing laws, new key performance indicators, new services, and new data might require changing the digital twin and 10 starting new dev-cycles.

In the operation phase, the digital twin 11 monitors the actual system and receives data continuously. One can 12 compare planned runtime models with real runtime models and if deviations occur, the digital twin 13 sends execution commands to adapt the actual system. In additional development cycles 14, one can add new languages—for example, to add runtime models, or new services (e.g., for process discovery from data or process conformance checking).

In the end-of-life phase, the actual system and its parts can take different paths: The system can be deconstructed, parts can be recycled, re-used, or thrown away. This change in the actual system might require 15 to reconfigure our digital twin, for example, based on new data or the absence of expected data. Due to legal reasons 16, we might need to keep data and models (or the whole digital twin) longer than the actual system. Moreover, we might 17 re-use parts of the actual system and provide related data, models, and services for the digital twin of the other actual system.

To summarize some main challenges when developing and evolving digital twins: Besides the current practice of ad hoc development of digital twins, several problems are related to the underlying complexities of the actual system [46, 47]: the system complexity due to a large number of heterogeneous subsystems, time complexity related to the long lifespan of the actual system and changing realities, integration complexity due to heterogeneous artefacts, different stakeholders and heterogeneous views, as well as information complexity due to heterogeneous data and models. In addition, budget challenges might influence the development and maintenance of digital twins. Moreover, current digital twin practices face a gap in information and feedback flows between the different life cycle phases.

4 | Model-Driven Engineering

As we have seen, models play a key role in many aspects of digital twin development. This prominent role of models raises important questions about how these models should be created, maintained, manipulated, and used (see also Tao et al. [48, Sect. V.B]). Some of these challenges have already surfaced in the discussion above. For example, we have seen the importance of being able to re-use models across life cycle stages.

MDE [8] is a subfield of software engineering that focuses on using models as first-class artefacts in the software engineering process. As a result, the field has developed many techniques for the efficient development of modeling languages, effective manipulation and validation of models, and techniques for maintenance and optimization of models. These techniques will also be useful in the creation and use of digital twins. The next section discusses some of the ways in which MDE can help address digital twin challenges and outlines some open challenges that should be addressed between the MDE and digital twin communities.

To set the context for this discussion, we briefly set out the key concepts in MDE. For space reasons, this section cannot be an exhaustive introduction to the field. For more detail, Brambilla et al. [8] provide a good introduction.

- *Models* are the key ingredients of MDE. A model is an abstraction of parts of the real world for a purpose.
- In MDE, models need to be expressed in a *modeling language* that has been formalized (i.e., exists in a way that is unambiguously processable by a computer). Modeling languages can be general-purpose, like the Unified Modeling Language or SysML. For many projects, DSMLs, created specifically for the development of applications in a particular domain, can be more useful.
- Models can be transformed into other models or into text (e.g., program code, documentation, reports, etc.) by automated *model transformations*, often expressed in dedicated transformation languages.
- A large range of tools and mechanisms exist to *manage models*. These support model versioning, model comparison, model analysis, model validation, model composition and re-use, linking models, and many more.

5 | MDE of Digital Twins: A Research Roadmap

The previous sections described the different aspects and processes involved in MDE of digital twins. They covered the topics of development, execution, and analysis of a digital twin, and throughout, highlighted the concerns and considerations developers should have in developing a digital twin. In this section, we perform a gap analysis, discussing the remaining challenges facing digital twin development and potential avenues for future research topics. Figure 5 shows an overview of current challenges, existing solutions from the MDE Body of Knowledge and needed action items for each topic further discussed in the following subsections.

5.1 | Development Guidelines and Standardization for Digital Twins and their Engineering

Digital twin development is still largely "artisanal" and would benefit from the possibility of re-use and from standardization of development knowledge [49], such as ongoing in ISO/IEC JTC 1/SC 41 for IoT and DTs[1] or for cities and municipalities in DIN SPEC 91607—2024-11 [50]. Even though implementation differences exist between digital twins in different contexts and environments, the potential to standardize reference architectures or requirements capture for digital twins remains. In an MDE-based approach to digital twin development, such knowledge would be captured in DSMLs, models, and model transformations. Research challenges remain around specifying

Topic	Challenges	MDE BoK	Action Items
Guidelines and Standards	Ad-hoc development	Modeling languages, model transformations, reference models	Contextualization for digital twin domains with domain experts
Interoperability	Integration complexity due to heterogeneous artefacts, different stakeholders, heterogenous views	Interchange formats, model transformations, code generation, interoperability models	Formalizing structures currently used in specific domains for realizing bidirectional mappings between different digital twin implementation approaches
Multi-paradigm/ Multi-view	System complexity and heterogeneous domain knowledge and needs	Multi-paradigm modeling/multi-viewpoint modeling frameworks including concepts, methods, and tools	Efficient methods to define multiple views on digital twins with consistency ensurance between different views, abstraction levels, etc.
Evolution	Time complexity due to changing realities and long lifetimes	Model evolution and co-evolution approaches, change types, change impact, and repairs	Continous evolution support for running digital twins/actual systems, change categorizations for digital twins and consistency requirements
Continuous Engineering	Costs, system complexity and heterogeneous domain expert backgrounds, gap between design and operation	Model-based DevOps based on models@runtime techniques and tools	Abstraction levels for digital twin data and scalable models@runtime to deal with long-living systems, feedback methods from operation to design
Composition of Digital Twins	Information complexity due to heterobegenous data and models	Model/modeling language composition operators and theories	Development of language support for composing digital twins (both during design-time and runtime)

FIGURE 5 | Overview of the research roadmap: challenges, existing solutions from the MDE Body of Knowledge and needed action items.

the concrete DSMLs and transformations: For example, what are the concepts required in DSMLs that can capture the typical requirements, assumptions, and scope of digital twin systems? What are the most important nonfunctional properties of digital twin systems, including specific properties such as synchronization frequencies and fidelity? How can they be captured in models of digital twins and can such models be automatically transformed for analysis at digital twin design time or even to generate digital twin implementations that achieve the required nonfunctional properties by construction?

5.2 | Digital Twin Interoperability

Integration and data exchange between different components is a key technical challenge in digital twin development. Correct and timely data conversion when synchronizing the information in the digital twin with the status of the actual system is crucial for digital twin efficacy, particularly where virtual models and services within a digital twin are largely heterogeneous and use different data formats. Time-based qualities, security, and modularity are core digital twin qualities and are closely tied to design and data transfer concerns. An MDE approach could help address these challenges in two ways as follows: (i) using DSMLs, data formats could be standardized across components for more efficient information interchange; and (ii) where such alignment is not possible, the translations required could be implemented using model transformation technology, making them explicit and amenable to analysis and verification. Where digital twin components are provided externally as "black boxes," MDE offers opportunities for automating the generation of wrappers and glue-code to increase re-use efficiency. In the area of simulation—keeping in mind that many digital twins include a simulation capability—distributed co-simulation [51] and the high-level architecture [52] are established approaches for black-box composition. These approaches can be combined with the generation capabilities of MDE to achieve even greater flexibility across domains [53]. Challenges remain, of course. For example, only a limited understanding currently exists of what the structure of the information to be interchanged should be (though work such as the Asset Administration Shell [17] has made an MDE-based start on these questions). Equally, many data interchange transformations need to be bidirectional. The theory and tooling for bidirectional transformation is an active research area within MDE.

5.3 | Multiparadigm and Multiview Modeling for Digital Twins

Models in a digital twin often span multiple levels of abstraction. For example, a digital twin for agriculture may include a model concerning the qualities and properties of fields such as turnover rate and yield, but may also include a model on individual crops and their optimal growth conditions, and so forth. Integrating these different abstraction levels into a single digital twin is often far from trivial. Communication protocols between models at different abstraction levels, and aggregation of data collection from those models, need to be considered carefully. Often, a digital twin involves a diverse range of stakeholders during development, maintenance, and use. Different stakeholders may require different viewpoints on the digital twin (both in relation to models *in*, and models *of* the digital twin), including the type of data exposed to the user and the format/syntax of data presentation. In MDE, the areas of multiparadigm and multiview modeling [54–56] address challenges like the two above, making the theory and tools developed highly relevant for digital twin development. Significant open research challenges remain. For

example, research is needed on how to minimize development overheads such as facilitating stakeholder engagement, implementation time for different data views, and view integration. Bidirectional synchronization and transformations play a key role here [57].

5.4 | Model, Object, and Digital Twin Evolution

Section 2.3 discusses the relationship between the life cycle of the actual system and the life cycle of the digital twin. We highlighted that this relationship requires constant co-evolution between the digital twin and the actual system—beyond just bidirectional data updates, this also needs to support updates to the structure of any models kept in the digital twin [58]. For example, when new services and components are added to the actual system, these need to also be represented in the digital twin, requiring appropriate amendments to the structure of any models in the digital twin. MDE research has a significant body of work exploring aspects of co-evolution between models and the reality they represent or reactive updates from one model to connected models [59]. These ideas are also applicable in the digital twin context. However, new challenges arise related to the need for the digital twin to be updated automatically when changes in the actual system occur. For example, how and when a digital twin should be "disconnected from/reconnected to" the actual system and how the virtual representation and services of the digital twin can be updated appropriately and automatically. Digital twin engineers also have to consider how these changes may impact the consistency of models/meta-models and data during transition periods.

5.5 | Model-Based DevOps for DTs

Digital twins require models of the actual system. Creating these models initially can be challenging. For engineered actual systems in particular, models often already exist from the system design phase. However, these can currently be difficult to re-use when creating digital twins. In MDE, the idea of models@runtime [60, 61] provides tools and techniques for managing system (design) models at runtime of a software system. This idea has led to the extension of DevOps ideas to the space of run-time model-based DevOps [62, 63], including the smooth transition from design models to runtime models [64]. Digital twins need to manage very large models and data sets efficiently and effectively, and this management continues to be a research challenge for models@runtime approaches, too. Similarly, digital twin engineering still raises challenges around identifying appropriate abstraction and refinement mechanisms to synthesize the rich detail of actual data into processable model information and vice versa [65]. Moreover, feedback from the operation phase of the actual system and its digital twin back to the system design and its design models could improve engineered systems.

5.6 | Composition of Digital Twins

Actual systems are complex, and as discussed before, represent actual systems of systems. As a consequence, digital twins cannot be developed monolithically, but rather need to be considered as federations of subtwins [33]. This federation raises challenges about the safe and robust composition of different digital twins, their models, and simulations. MDE has a long tradition of exploring issues of modularity and composition/integration of models, transformations, and DSMLs, both in foundational terms and practical tools. All of these are applicable to digital twin development and can provide significant benefits. MDE is also able to capture typical digital twin composition operators in appropriate DSMLs, which would enable more efficient expression of new composition scenarios, including their dynamic manipulation and optimization. However, significant challenges remain [64], not least around managing information protection between different organizations in charge of different subdigital twins, and in relation to supporting dynamic federation (and disbandment) of digital twins. This challenge requires dynamically adjusting models and simulations, possibly partway through simulation runs, while making sure that real-time and historic data (which may be based on different twin federations) are managed robustly and reliably.

6 | Conclusions

To create, maintain, and evolve digital twins along with their actual system over a long lifespan is a significant engineering challenge. Modeling is at the core of digital twins [48]. In this paper, we have discussed different forms of modeling in and of digital twins and how MDE provides the basic techniques to support the digital twin life cycle in an agile manner. We have identified different ways of applying MDE to digital twin development and operation and have identified future research challenges for MDE-based digital twin development.

Acknowledgments

This paper is the result of discussions during Dagstuhl Seminar 22362 "Model-Driven Engineering of Digital Twins". We thank the team at Dagstuhl for supporting our discussions. This study is partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) and the Agence Nationale De La Recherche (ANR)—France—Model-Based DevOps—505496753 and ANR-22-CE92-0068. Website: https://mbdo.github.io and partially funded by the Key Digital Technologies (KDT) Joint Undertaking through the European Union's Horizon Europe project MATISSE, grant agreement No. 101140216.

Open access funding enabled and organized by Projekt DEAL.

Data Availability Statement

Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Endnotes

[1] https://www.iso.org/committee/6483279.html

References

1. W. Xian, K. Yu, F. Han, L. Fang, D. He, and Q. L. Han, "Advanced Manufacturing in Industry 5.0: A Survey of Key Enabling Technologies and Future Trends," *IEEE Transactions on Industrial Informatics* 20, no. 2 (2024): 1055–1068, https://doi.org/10.1109/TII.2023.3274224.

2. M. Dalibor, N. Jansen, B. Rumpe, et al., "A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins," *Journal of Systems and Software* 193 (2022): 111361, https://doi.org/10.1016/j.jss.2022.111361.

3. H. M. Muctadir, D. A. M. Negrin, R. Gunasekaran, L. Cleophas, M. van den Brand, and B. R. Haverkort, "Current Trends in Digital Twin Development, Maintenance, and Operation: An Interview Study," *Software and Systems Modeling* 23, no. 5 (Springer, 2024): 1275–1305, https://doi.org/10.1007/s10270-024-01167-z.

4. J. Zhang, C. Ellwein, M. Heithoff, J. Michael, and A. Wortmann, "Digital Twin and the Asset Administration Shell: An Analysis of 3 AASs Types and Their Feasibility for Digital Twin Engineering," *Journal Software and Systems Modeling (SoSyM)* (Springer, 2025), https://doi.org/10.1007/s10270-024-01255-0.

5. R. Eramo, F. Bordeleau, B. Combemale, M. Van Den Brand, M. Wimmer, and A. Wortmann, "Conceptualizing Digital Twins," *IEEE Software* 39, no. 2 (2021): 39–46, https://doi.org/10.1109/MS.2021.3130755.

6. M. Dalibor, M. Heithoff, J. Michael, et al., "Generating Customized Low-Code Development Platforms for Digital Twins," *Journal of Computer Languages (COLA)* 70 (2022): 101117, https://doi.org/10.1016/j.cola.2022.101117.

7. D. J. Wagg, K. Worden, R. J. Barthorpe, and P. Gardner, "Digital Twins: State-of-the-Art and Future Directions for Modeling and Simulation in Engineering Dynamics Applications," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering* 6, no. 3 (2020): 030901, https://doi.org/10.1115/1.4046739.

8. M. Brambilla, J. Cabot, M. Wimmer, and L. Baresi, *Model-Driven Software Engineering in Practice*, 2nd ed. (Morgan & Claypool, 2017).

9. J. C. Kirchhof, J. Michael, B. Rumpe, S. Varga, and A. Wortmann, "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* (ACM, 2020), 90–101, https://doi.org/10.1145/3365438.3410941.

10. *International Electrotechnical Commission, ISO/IEC 30173:2023 Digital Twin – Concepts and Terminology* (ISO/IEC, 2023), https://www.iso.org/standard/81442.html.

11. F. Tao, W. Liu, M. Zhang, et al., "Five-Dimension Digital Twin Model and Its Ten Applications," *Computer Integrated Manufacturing Systems* 25, no. 1 (2019): 1–18.

12. F. Tao, M. Zhang, and A. Nee, "Five-Dimension Digital Twin Modeling and Its Key Technologies," in *Digital Twin Driven Smart Manufacturing*, ed. F. Tao, M. Zhang, and A. Nee (Academic Press, 2019), 63–81, chap. 3.

13. D. McKee, "Platform Stack Architectural Framework: An Introductory Guide" (Digital Twin Consortium, 2023), https://www.digitaltwinconsortium.org/platform-stack-architectural-framformework-an-introductory-guide-form.

14. International Standardization Organization, "ISO/DIS 23247-1. *Automation Systems and Integration–Digital Twin Framework for Manufacturing–Part 1: Overview and General Principles* (ISO, 2020), https://www.iso.org/standard/75066.html.

15. E. Ferko, A. Bucaioni, P. Pelliccione, and M. Behnam, "Standardisation in Digital Twin Architectures in Manufacturing," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)* (IEEE, 2023), 70–81.

16. K. Schweichhart, *Reference Architectural Model Industrie 4.0 (RAMI 4.0) - An Introduction* (Publikationen der Plattform Industrie 4.0, 2016), https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html.

17. S. R. Bader and M. Maleshkova, "The Semantic Asset Administration Shell," in *Semantic Systems. The Power of AI and Knowledge Graphs: 15th International Conference, SEMANTiCS 2019* (Springer, 2019), 159–174.

18. M. Grieves and J. Vickers, *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems* (Springer, 2017), 85–113.

19. S. R. Newrzella, D. W. Franklin, and S. Haider, "Three-Dimension Digital Twin Reference Architecture Model for Functionality, Dependability, and Life Cycle Development Across Industries," *IEEE Access* no. 10 (IEEE, 2022): 95390–95410, https://doi.org/10.1109/ACCESS.2022.3202941.

20. W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in Manufacturing: A Categorical Literature Review and Classification," in *IFAC 51* (2018), 1016–1022.

21. K. Josifovska, E. Yigitbas, and G. Engels, "Reference Framework for Digital Twins Within Cyber-Physical Systems," in *2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)* (IEEE, 2019), 25–31.

22. H. Boyes and T. Watson, "Digital Twins: An Analysis Framework and Open Issues," *Computers in Industry* 143 (2022): 103763, https://doi.org/10.1016/j.compind.2022.103763.

23. S. Shoshany-Tavory, E. Peleg, A. Zonnenshain, and G. Yudilevitch, "Model-Based-Systems-Engineering for Conceptual Design: An Integrative Approach," *Systems Engineering* 26, no. 6 (2023): 783–799, https://doi.org/10.1002/sys.21688.

24. K. Amunts, M. Axer, S. Banerjee, et al., "The Coming Decade of Digital Brain Research - a Vision for Neuroscience at the Intersection of Technology and Computing," *Imaging Neuroscience* 2 (MIT Press, 2024): 1–35, https://doi.org/10.1162/imag_a_00137.

25. G. Coorey, G. A. Figtree, D. F. Fletcher, et al., "The Health Digital Twin to Tackle Cardiovascular Disease: A Review of an Emerging Interdisciplinary Field," *npj Digital Medicine* 5, no. 1 (Springer Nature, 2022): 126, https://doi.org/10.1038/s41746-022-00640-7.

26. G. Kim, J. Humble, P. Debois, J. Willis, and N. Forsgren, *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations* 2nd ed. (IT Revolution Press, 2022).

27. B. Combemale, J. Kienzle, G. Mussbacher, et al., "A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems," *IEEE Software* 38, no. 4 (2020): 71–84, https://doi.org/10.1109/MS.2020.2995125.

28. R. Ackoff, "From Data to Wisdom: Presidential Address to ISGSR, June 1988," *Journal of Applied Systems Analysis* 16 (1989): 3–9.

29. J. Rowley, "The Wisdom Hierarchy: Representations of the DIKW Hierarchy," *Journal of Information Science* 33, no. 2 (2007): 163–180, https://doi.org/10.1177/0165551506070706.

30. B. H. C. Cheng, B. Combemale, R. B. France, J. M. Jézéquel, and B. Rumpe, "On the Globalization of Domain Specific Languages," in *Globalizing Domain-Specific Languages* (Springer, 2015).

31. B. Combemale, R. France, J. M. Jézéquel, B. Rumpe, J. Steel, and D. Vojtisek, *Engineering Modeling Languages: Turning Domain Knowledge Into Tools*, Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series (CRC Press, 2016).

32. J. Michael, J. Pfeiffer, B. Rumpe, and A. Wortmann, "Integration Challenges for Digital Twin Systems-of-Systems," in *10th IEEE/ACM Int. Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems* (IEEE, 2022), https://doi.org/10.1145/3528229.3529384.

33. T. Olsson and J. Axelsson, "Systems-of-Systems and Digital Twins: A Survey and Analysis of the Current Knowledge," in *18th Annual System of Systems Engineering Conference (SoSe'23)* (2023), 1–6.

34. H. Stachowiak, *Allgemeine Modelltheorie* (Springer, 1973).

35. W. Torres, v. d. M. G. J. Brand, and A. Serebrenik, "A Systematic Literature Review of Cross-Domain Model Consistency Checking by Model Management Tools," *Software and Systems Modeling* 20, no. 3 (Springer, 2021): 897–916, https://doi.org/10.1007/S10270-020-00834-1.

36. B. S. Barn, "The Socio-Technical Digital Twin: On the Gap Between Social and Technical Feasibility," in *IEEE Conference on Business Informatics* (2022), https://doi.org/10.1109/CBI54897.2022.00009.

37. P. Laplante, "Trusting Digital Twins," *IEEE Computer* 55, no. 7 (2022): 73–77, https://doi.org/10.1109/MC.2022.3149448.

38. J. Michael, M. Schwammberger, and A. Wortmann, "Explaining Cyberphysical System Behavior With Digital Twins," *IEEE Software* 41, no. 1 (IEEE, 2024): 55–63, https://doi.org/10.1109/MS.2023.3319580.

39. B. P. Zeigler, A. Muzy, and E. Kofman, *Theory of Modeling and Simulation*, 3rd ed. (Elsevier, 2018).

40. J. Denil, S. Klikovits, P. J. Mosterman, A. Vallecillo, and H. Vangheluwe, "The Experiment Model and Validity Frame in M&S," in *Proceedings of the Symposium on Theory of Modeling & Simulation* (Society for Computer Simulation International, 2017), 1–12.

41. S. Stepney and F. A. C. Polack, *Engineering Simulations as Scientific Instruments: A Pattern Language* (Springer, 2018), https://doi.org/10.1007/978-3-030-01938-9.

42. S. Zschaler and F. A. C. Polack, "Trustworthy Agent-Based Simulation: The Case for Domain-Specific Modelling Languages," *Software and Systems Modeling* 22, no. 2 (Springer, 2023): 455–470, https://doi.org/10.1007/s10270-023-01082-9.

43. P. Wilsdorf, S. Zschaler, F. Haack, and A. M. Uhrmacher, "Potential and Challenges of Assurance Cases for Simulation Validation," in *2024 Winter Simulation Conference* (IEEE, 2024), https://doi.org/10.1109/WSC63780.2024.10838818.

44. S. Burton and B. Herd, "Addressing Uncertainty in the Safety Assurance of Machine-Learning," *Frontiers in Computer Science* 5 (2023): 1132580, https://doi.org/10.3389/fcomp.2023.1132580.

45. J. Bickford, D. L. Van Bossuyt, P. Beery, and A. Pollman, "Operationalizing Digital Twins Through Model-Based Systems Engineering Methods," *Systems Engineering* 23, no. 6 (2020): 724–750, https://doi.org/10.1002/sys.21559.

46. K. Feichtinger, K. Meixner, F. Rinker, et al., "Industry Voices on Software Engineering Challenges in Cyber-Physical Production Systems Engineering," in *IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)* (IEEE, 2022), https://doi.org/10.1109/ETFA52439.2022.9921568.

47. J. Michael, *Model-Driven Engineering of Digital Twins With Informative and Assistive Services*, Aachener Informatik-Berichte, Software Engineering (Shaker Verlag, 2025).

48. F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics* 15, no. 4 (2019): 2405–2415, https://doi.org/10.1109/TII.2018.2873186.

49. S. A. Niederer, M. S. Sacks, M. Girolami, and K. Willcox, "Scaling Digital Twins From the Artisanal to the Industrial," *Nature Computational Science* 1, no. 5 (2021): 313–320.

50. DIN, "DIN SPEC 91607:2024-11: Digital Twins for Cities and Municipalities," 2024, https://doi.org/10.31030/3575521.

51. S. Gil, E. Kamburjan, P. Talasila, and P. G. Larsen, "An Architecture for Coupled Digital Twins With Semantic Lifting," *Software and Systems Modeling* (Springer, 2024): 1–26, https://doi.org/10.1007/s10270-024-01221-w.

52. IEEE, "IEEE 1516-2010: IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules," 2010, https://standards.ieee.org/ieee/1516/3744/.

53. S. Zschaler, N. Mustafee, A. Harper, et al., "On Simulation Reuse in Healthcare Applications," *Journal on Simulation, Transactions of the SCS* (2025). Under review.

54. H. Vangheluwe, J. De Lara, and P. J. Mosterman, "An introduction to Multi-Paradigm Modelling and Simulation," AI, Simulation and Planning in High Autonomy Systems - AIS'2002 Conference (2002), 9–20.

55. M. Amrani, D. Blouin, R. Heinrich, A. Rensink, H. Vangheluwe, and A. Wortmann, "Multi-paradigm modelling for cyber–physical systems: a descriptive framework,", *Software and Systems Modeling* 20, no. 3 (Springer, 2021): 611–639, https://doi.org/10.1007/s10270-021-00876-z.

56. P. J. Mosterman, H. Vangheluwe, "Computer Automated Multi-Paradigm Modeling: An Introduction," Simulation: Transactions of the Society for Modeling and Simulation International 80, no. 9 (Sage Journals, 2004): 433–450, https://doi.org/10.1177/0037549704050532.

57. F. Bordeleau, B. Combemale, R. Eramo, M. Van den Brand, and M. Wimmer, "Towards Model-Driven Digital Twin Engineering: Current Opportunities and Future Challenges," in *Systems Modelling and Management - First International Conference, ICSMM 2020*, Communications in Computer and Information Science, vol. 1262 (Springer, 2020), 43–54.

58. I. David and D. Bork, "Towards a Taxonomy of Digital Twin Evolution for Technical Sustainability," in *International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (ACM/IEEE, 2023), https://doi.org/10.1109/MODELS-C59198.2023.00147.

59. C. C. Rațiu, W. K. G. Assunção, E. Herac, R. Haas, C. Lauwerys, and A. Egyed, "Using Reactive Links to Propagate Changes Across Engineering Models," *Software and Systems Modeling* (Springer, 2024), https://doi.org/10.1007/s10270-024-01186-w.

60. H. Giese, N. Bencomo, L. Pasquale, et al., "Living With Uncertainty in the Age of Runtime Models," in *Models@run.time - Foundations, Applications, and Roadmaps [Dagstuhl Seminar 11481, November 27 - December 2, 2011]*, LNCS, vol. 8378, (Springer, 2011), 47–100.

61. N. Bencomo, S. Götz, and H. Song, "Models@run.time: A Guided Tour of the State of the Art and Research Challenges," *Journal Software and Systems Modeling (SoSyM)* 18, no. 5 (Springer, 2019): 3049–3082, https://doi.org/10.1007/s10270-018-00712-x.

62. J. Hugues, A. Hristosov, J. J. Hudak, and J. Yankel, "TwinOps - DevOps Meets Model-Based Engineering and Digital Twins for the Engineering of CPS," in *23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Comp* (ACM, 2020), https://doi.org/10.1145/3417990.3421446.

63. B. Combemale and M. Wimmer, "Towards a Model-Based DevOps for Cyber-Physical Systems," in *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment* (Springer International Publishing, 2020), 84–94, https://doi.org/10.1007/978-3-030-39306-9_6.

64. B. Combemale, J. M. Jézéquel, Q. Perez, et al., "Model-Based DevOps: Foundations and Challenges," in *International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)* (ACM/IEEE, 2023), https://doi.org/10.1109/MODELS-C59198.2023.00076.

65. N. Bencomo, J. Cabot, M. Chechik, et al., "Abstraction Engineering," arXiv, 2024, https://arxiv.org/abs/2408.14074.