*Proceedings of the 2025 Winter Simulation Conference*
*E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.*

# FROM DIGITAL TWINS TO TWINNING SYSTEMS

Giovanni Lugaresi[1,2] and Hans Vangheluwe[3,4]

[1]Dept. of Mechanical Eng., KU Leuven, Leuven, BELGIUM
[2]Flanders Make at KU Leuven, Leuven, BELGIUM
[3]Dept. of Computer Science, University of Antwerp, Antwerp, BELGIUM
[4]Flanders Make at the University of Antwerp, Antwerp, BELGIUM

## ABSTRACT

Digital twins are rapidly emerging as a disruptive innovation in industry, offering a dynamic integration of physical systems with their virtual counterparts through data-driven and real-time synchronization. In this work, we explore the foundational principles, architectures, and life cycle of digital twins, with a particular emphasis on their use in simulation-based decision-making. We dissect the core components of a digital twin and examine how these elements interact across a system's life cycle. The notion of a twinning system is introduced as a unifying framework for sensing, simulation and digital models/shadows/twins, with applications in diverse fields. We outline the most significant choices that stakeholders must make. A case-study is based on a lab-scale manufacturing system and illustrates how DTs are constructed, validated, and used for scenario analysis and autonomous decision-making. We also discuss the challenges associated with synchronization, model validity, and the development of internal services for operational use.

## 1 INTRODUCTION

Over the past decade, Digital Twins (DTs) have gained significant momentum as a foundational technology for integrating physical systems with their digital representations. Originally coined in the context of product life cycle management (PLM) (Grieves and Vickers 2017), the term *digital twin* now broadly refers to replica of a physical entity that is continually updated with data from the physical system. The bi-directional synchronization enables monitoring, forecasting, control, and optimization throughout the system's life cycle (Tao et al. 2019; Jones et al. 2020). DT emerged together with and was fueled by advances in Industry 4.0 technologies, including the internet-of-things, cyber-physical systems, cloud computing, and artificial intelligence. These technological shifts have made it increasingly feasible to model, simulate, and interact with complex systems in real time, thus bringing the vision of DTs closer to industrial reality (Fuller et al. 2020).

### 1.1 Industrial Context

The adoption of DTs is deeply embedded in ongoing transformations in manufacturing, energy systems, transportation, healthcare, and infrastructure management. Stakeholders find new ways to improve decision-making, reduce downtime, enable predictive maintenance, and optimize resource utilization. For instance, in advanced manufacturing, DTs are increasingly used to manage smart factories (Kritzinger et al. 2018), while in energy systems, DTs are used to balance supply and demand, integrate renewables, and simulate grid dynamics under uncertainty (Jafari et al. 2023). The global DT market was valued at approximately USD 8.6 billion in 2022 and has been projected to grow to USD 137.7 billion by 2030, with a compound annual growth rate (CAGR) of 42.6% (Fortune Business Insights. 2023). Industry surveys confirm this trend: only about 5% of companies report that DTs are not part of their digital transformation strategy, while 86% identify them as a critical component (Dertien and Macmahon 2022). In parallel with this

Table 1: Number of publications resulting from the query *"digital twin*"* on the Scopus database.

| **Year** | 2016 | 2017 | 2018 | 2019 | 2020 | 2021 | 2022 | 2023 | 2024 | 2025 (May) |
|---|---|---|---|---|---|---|---|---|---|---|
| **Publications** | 29 | 127 | 369 | 1054 | 1841 | 3202 | 5244 | 7499 | 9897 | 2965 |

industrial momentum, DTs are becoming the subject of formal standardization, notably through the ISO 23247 framework for Digital Twin manufacturing systems (ISO 23247:2021. 2021).

Academic interest has surged as well. A search for the term *"digital twin*"* on the Scopus database yielded 32,264 documents (conducted on 2025-05-02). As shown in Table 1, the number of publications over the last decade reflects both the rapid growth of research activity and the expanding diversity of DT applications across domains.

### 1.2 From Modeling and Simulation to (Digital) Twins

Simulation models have been essential tools in engineering and systems science. However, traditional models are often static, scenario-specific, and loosely coupled with real-time data. In contrast, a Digital Twin introduces the need for continual updates, tightly integrated with sensor and process data, and embedded within a feedback loop that allows it to influence and be influenced by the physical system (Negri et al. 2017). This evolution from static simulation to dynamic twinning is not just technical – it requires rethinking model fidelity, validity, synchronization mechanisms, and the role of models (as used in various forms of analysis such as simulation) as active agents in operational contexts. DTs thus represent a convergence of multiple threads: data acquisition, multi-physics simulation, artificial intelligence and/or machine learning analytics, and human-machine interaction. One of the key innovations lies in the way simulation is embedded in ongoing system management, supporting not only design-time analysis but also runtime decision-making (Rasheed et al. 2020).

### 1.3 Notable Proofs-of-Concept

Several proof-of-concept implementations have demonstrated the feasibility and value of DT. For example, GE and Siemens have developed digital twins for turbine engines to monitor wear and predict failures (Fuller et al. 2020). In the automotive sector, Tesla's virtual representations of vehicles facilitate over-the-air updates and predictive maintenance (Patil et al. 2024). In academic and lab settings, testbeds such as smart factories and cyber-physical labs have illustrated how DTs can support scenario testing, anomaly detection, and autonomous control (Schleich et al. 2017).

### 1.4 Scope and Structure of This Work

The remainder of this tutorial is structured to progressively introduce and deepen the reader's understanding of Digital Twinning from foundational concepts to implementation practices. Section 2 outlines core definitions and historical classifications. Section 3 introduces the twinning paradigm, presenting a high-level view of DTs as systems of synchronized virtual and physical counterparts. It outlines the goals of DTs and proposes conceptual architectures (sections 3.1 and 3.2), with a particular focus on Twinning Experiments (TEs) as key units of analysis and control. Section 3.3 discusses the use of appropriate modeling formalisms, such as DEVS, Petri Nets, and differential equations, as well as workflows for orchestrating experiments. Section 3.4 covers the deployment stage, detailing the tools, protocols, and runtime systems involved in realizing DTs, with attention to standardization efforts. Section 4 reflects on model validity. All uses of models, including the trustworthy operation of DTs, are critically dependent on the fact that these models are only used within their validity range. Section 5 presents an illustrative use case, highlighting how the concepts and architectures described are applied in practice. Finally, section 6 offers concluding reflections on the current state and future directions of digital twinning.

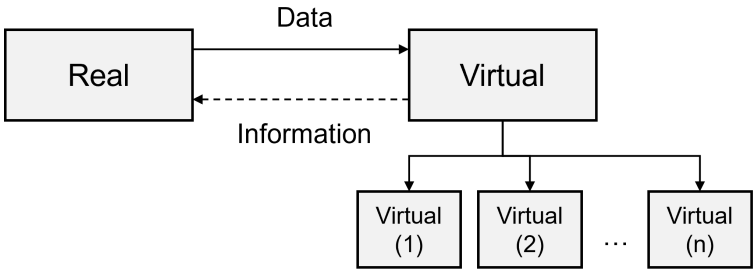


Figure 1: A simple conceptual architecture of a digital twin (adapted from Grieves and Vickers, 2017).

## 2 FOUNDATIONAL CONCEPTS

The concepts behind twinning can already be seen in ancient times in the use of sand tables, rudimentary terrain models on which positions of assets are continually updated to monitor and simulate complex battleground situations, based on which strategic and tactical decisions would be made and implemented in the field (Kirschenbaum 2023). At NASA, the idea of a "digital twin" was born in the 1960s as a "living model" of the Apollo missions. In response to Apollo 13's oxygen tank explosion and subsequent damage to the main engine, NASA employed multiple simulators to evaluate the failure and extended a physical model of the vehicle to include digital components (Allen 2021). In the early 1990s, Gelernter introduced the concept of Mirror Worlds—digital models that reflect aspects of the real world and are continually updated via real-time data streams (Gelernter 1993). Building on this vision, the concept of feeding live data into simulation models was further developed under the notion of Symbiotic or Online Simulation (Davis 1998; Fujimoto et al. 2021). A symbiotic simulation system is characterized by a two-way interaction between a simulation and a physical system, where sensor data from the physical environment is used to update a model that in turn is used to perform predictive analyses or what-if scenario experiments to improve the physical system's behavior (Aydt et al. 2008).

The term Digital Twin was introduced by Michael Grieves in the context of Product Lifecycle Management (PLM). The definition originally proposed by Michael Grieves is *"A set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level. At its optimum, any information that could be obtained from inspecting a physically manufactured product can be obtained from its digital twin"* (Grieves and Vickers 2017). This definition reflects the initial conception of the DT as a means to support product-related decision-making across the entire product life cycle—from beginning-of-life to end-of-life—within a PLM framework aimed at closing the loop between design, production, use, maintenance, and disposal. The initial DT conceptual architecture by Grieves is illustrated in Figure 1. From the outset, the concept emphasized the coexistence of a physical system and its digital counterpart, with information structures that evolve alongside the system across its life cycle, to keep them consistent. The physical and virtual domains are connected via data flows that enable mutual influence. Initially described as the conceptual ideal for PLM, the concept was later renamed to the information mirroring model before finally being named the Digital Twin (Grieves and Vickers 2017).

Numerous alternative definitions have been proposed in recent literature, each highlighting particular aspects of DTs–such as the integration and interconnection of their components, the nature of the digital counterpart, as well as their predictive and descriptive capabilities. Barricelli et al. (2019) conducted a comprehensive review of 75 papers, categorizing them based on how DTs are defined across sectors such as manufacturing, aviation, and healthcare. A cross-domain systematic mapping study by Dalibor et al. (2022) retrieved 1471 papers and selected 356 of them for review. This resulted in a comprehensive feature model, with a software engineering focus on twinning. As a followup of this study, a catalog of 112 definitions (on 3 May 2025) of the term "Digital Twin" is kept at https://awortmann.github.io/research/digital-twin-definitions/. The ICOSE Systems Engineering paper (Michael et al. ) defines Digital Twins

from the point of view of Model Driven Engineering. Domain-specific definitions tailored to their contexts can also be found in Negri et al. (2017) for manufacturing, Xiong and Wang (2022) for aviation, and Croatti et al. (2020) for healthcare.

### 2.1 Digital Models, Digital Shadows, and Digital Twins

Kritzinger et al. (2018) proposed a classification framework that differentiates between various types of digital representations based on the level and direction of automated data exchange between a physical entity and its digital counterpart. Within this framework, a Digital Model (DM) is described as "a digital representation of an existing or planned physical object that does not involve any form of automated data exchange between the physical object and the digital object." The term Digital Shadow (DS) has gained traction in recent literature to denote digital models that are continually aligned with their physical counterparts (Kritzinger et al. 2018). In this case, the digital representation mirrors the current state of the physical entity with sufficient precision so that querying the DS yields the same information concerning Properties of Interest (PoIs) as if it were the physical entity. Digital shadows can support a variety of functions, including state visualization, monitoring, anomaly detection and predictive alerts. In contrast, a Digital Twin (DT) is characterized by "fully integrated bidirectional data flows between the physical and digital objects," enabling continual synchronization and interaction.

Matta and Lugaresi (2024) outlined the fundamental attributes of DTs. In this view, DTs function first as *digital* platforms that collect, process, and analyze data and information. Second, they are capable of *describing* the current state or projecting future states of their physical counterparts by hosting computational and simulation models. Third, as twins, they must maintain a certain level of *synchronization* with the physical world to ensure consistency between the virtual and real entities.

### 2.2 What is missing?

As digital twins take on a more prominent role in the design and operation of complex systems, it becomes essential to re-examine their conceptual foundations, to clarify the terminology used, and to investigate the common ground on which their main applications are based. The novel ideas from Grieves and the subsequent discussions on digital twins are often focused on either problem-specific modeling or on technology choices. As a consequence, conceptual design, choice of modeling formalism, deployment, operation and evolution of DTs is often ad-hoc. In what follows, we use the term *twinning paradigm* as introduced in (Paredis and Vangheluwe 2024) as an umbrella term to capture all aspects of the development, deployment, operation, inter-operation and evolution of twins. In analogy with the object-oriented programming or the agile paradigm, the twinning paradigm comprises workflows and architectures that look at a problem in terms of an Actual Object (AO), which is an interface to the System under Study (SuS) and a Twin Object (TO), which is the virtual counterpart (*e.g.,* a simulation model) of the SuS. To bypass the endless discussion about terminology, the twinning paradigm spans the entire spectrum between Modeling and Simulation, and Internet of Things; thus subsuming Digital Models, Digital Shadows, Digital Twins, Physical Models, and Physical Twins, and combinations thereof.

The twinning paradigm is used as a guideline to build twinning systems. It suggests choices to be made by a twinning system engineer, from conceptual to deployment stages. These stages are elaborated on in the following section.

## 3 TWINNING SYSTEMS

A Twinning System connects a System under Study (SuS) with a model of that SuS in some appropriate formalism (and its simulator/executor) (Grieves 2014) and keeps them *continually* synchronized. In such a Twinning system, we call the Actual Object (AO) a conceptual, abstracted view on and interface to the SuS and its environment. Similarly, a component holding a conceptual representation of the SuS and its environment is called a Twin Object (TO). SuS can be physical (*e.g.,* a machine) *or* digital (*e.g.,* software).

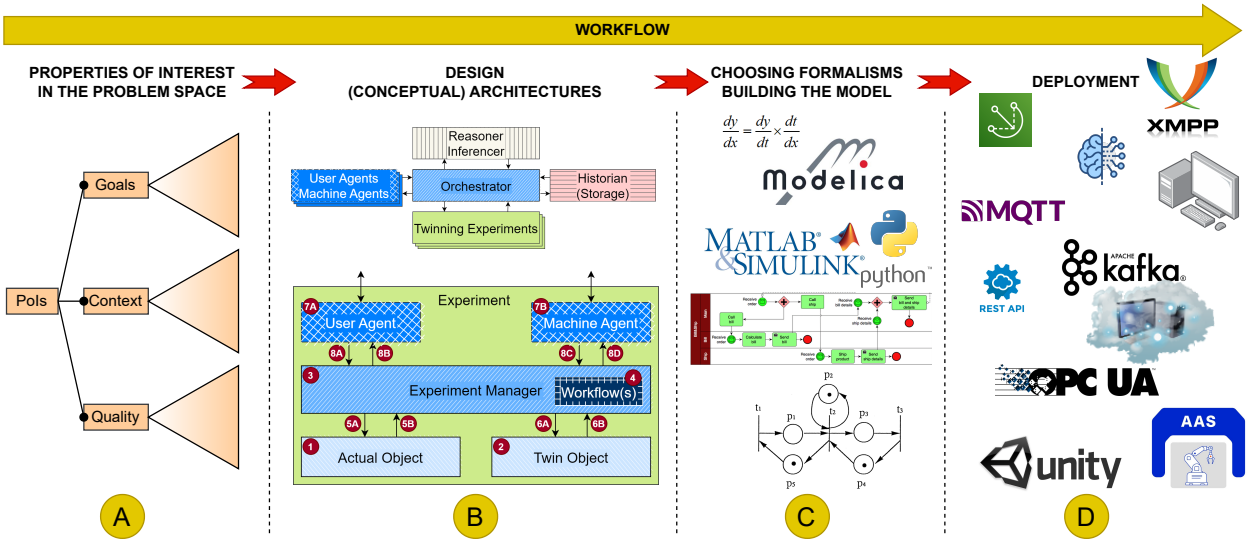


Figure 2: Generic workflow of a Twinning System.

Similarly, the model can be digital *or* physical. The former is generally considered a *Digital Twin* and the latter an *Analog Twin* (Paredis, Gomes, and Vangheluwe 2021), but many different variations and combinations exist, including the case where either SuS or Twin, or both, are conceptual (Guizzardi 2007; Proper and Guizzardi 2024).

To break down the creation of Twinning Systems into manageable stages, we introduce a high-level workflow, shown in Figure 2. It is based on the IIRA Architecture Framework (Industry IoT Consortium 2022) and the ISO/IEC 12207 standard (Singh 1996). This workflow is comprised of four inter-dependent stages:

- *Stage A - Define the goals*: identify specific objectives (*e.g.,* safety monitoring, predictive maintenance, optimization) using a feature tree to clarify requirements;
- *Stage B - Design conceptual architectures*: map goals to modular experiment architectures and define interaction and orchestration between the components at a high level;
- *Stage C - Select the modeling formalisms*: choose the appropriate modeling languages (such as differential equations, the Discrete-EVent system Specification (DEVS), or Petri nets) and analysis techniques (such as simulation or model checking) based on requirements and architecture;
- *Stage D - Deploy the system*: implement the architecture on hardware/software platforms using suitable technologies and communication protocols.

These four stages are outlined in the following sub-sections.

### 3.1 Goal Setting: Making Choices in the Problem Space

This first stage concerns the identification of requirements. It is meant to answer *why* a Twinning System must be constructed. One may classify these reasons to aid in selection. Kang and Lee (2013) distinguish *goals* (or purposes), *usage contexts* and *quality assurance*. There are many possible reasons for constructing a Twinning system as found in the literature (Dalibor et al. 2022; Van der Valk et al. 2020; Jones et al. 2020; Wanasinghe et al. 2020; Minerva et al. 2020).

Figure 3 shows a non-exhaustive collection of possible goals, in the form of a feature model. Connections with filled/open circles denote mandatory/optional features. The full arrows identify dependencies and the dashed arrows denote optional dependencies. Every path through the feature tree, corresponding to a sequence of choices made during requirements analysis of a Twinning system, leads to a specific goal. For

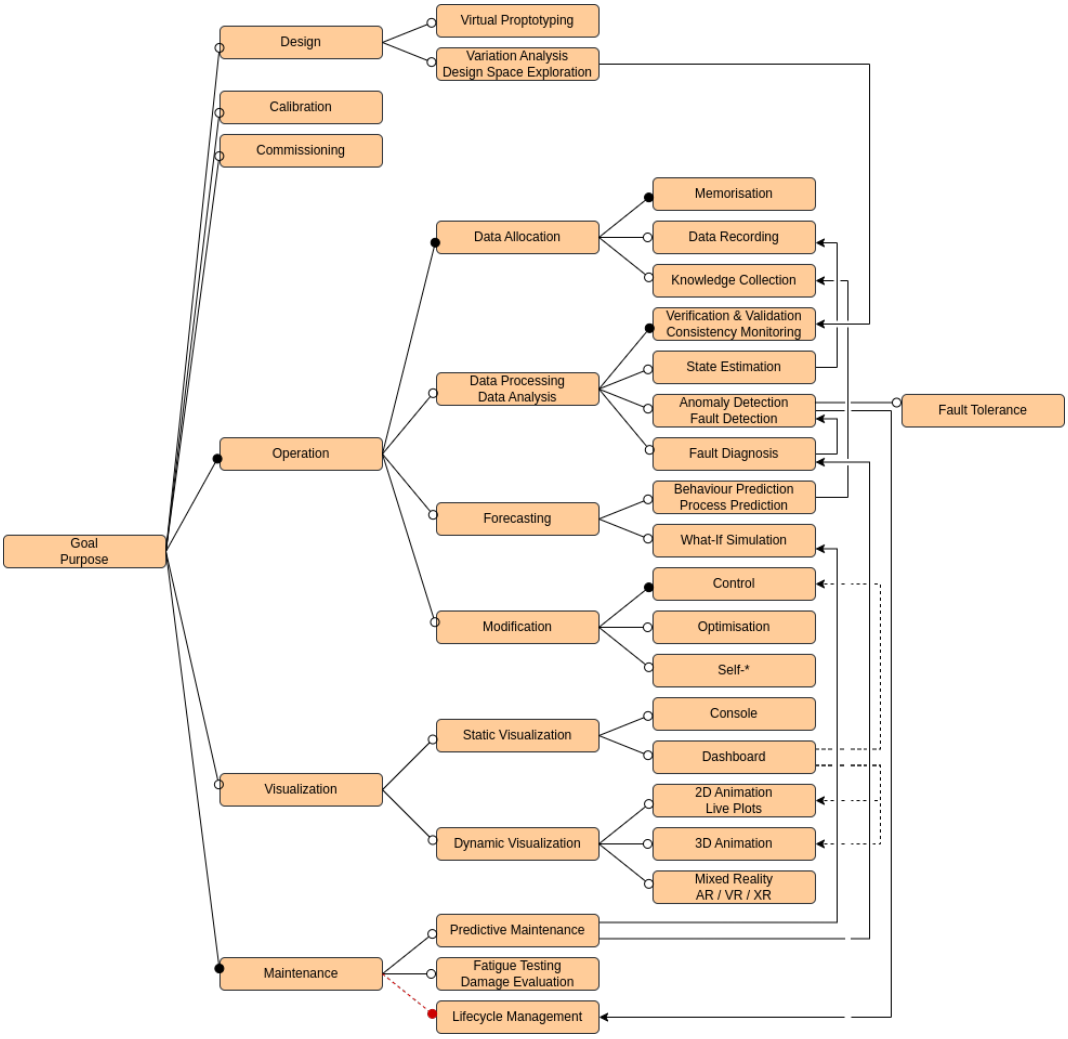


Figure 3: A Feature Model describing possible goals of the Twinning System (from Paredis, 2025).

example, Operation leads to Data Processing/Analysis, which leads to Anomaly Detection. Once this goal has been selected, the Properties of Interest (PoIs) over which in this case Anomaly Detection is required still need to be selected. In a vehicle, a PoI may for example be velocity: when actual velocity deviates too much from expected velocity (*i.e.,* as predicted by a twin), an anomaly is reported.

The correct selection of the goals and properties of interest drives the choices in downstream stages of the twinning system. For example, in Figure 4, a simple case involving a resistor shows the different modeling choices: if the property of interest is geometry, a CAD model would suffice, while if the property is the constitutive relation between the resistance $R$ of the resistor, the voltage drop $V$ over and the current $i$ through it, a mathematical model $V = Ri$ is to be chosen. A more elaborate breakdown of goals, usage contexts and quality assurance can be found in (Paredis and Vangheluwe 2024).

### 3.2 (Conceptual) Architecture Design

Once choices have been made in stage A, individual system components required to (functionally) ensure the valid behavior of the Twinning System need to first be identified and combined, at a conceptual level. Stage B answers the question *what* is required for a twin to be constructed. To explain this conceptual architecture, we introduce Twinning *experiments*. An experiment is an intentional set of (possibly hierarchically composed)

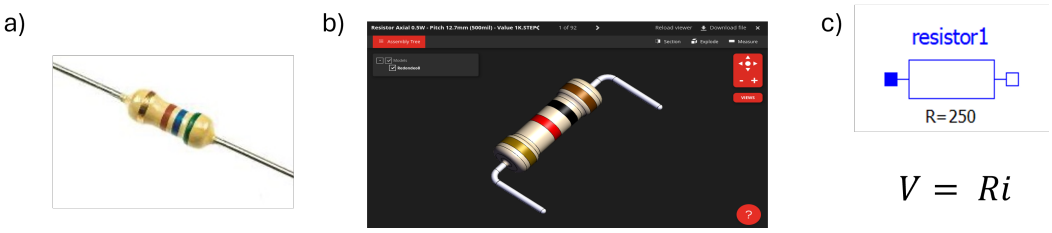


Figure 4: Example of modeling choices driven by the Properties of Interest – a) a real-world resistor, b) the CAD model of a resistor, c) the electrical diagram of a resistor and an equation relating current, resistance and voltage.

activities, carried out on a specific SuS in order to accomplish a specific set of goals. Each experiment should have a description, setup and workflow, such that it is (at least) *repeatable* (Plesser 2018).

The diagram at the bottom of stage B in Figure 2 shows a conceptual reference architecture, following the IIRA's *functional viewpoint*. The figure is annotated with numbered variation points, also known as presence conditions, to include or remove components and/or connections, based on the choices made in stage A. As such, the figure, also known as a 150% model, represents an entire family of possible conceptual architectures.

The *Actual Object* (AO) ① is an *abstraction* of a specific *view* on the SuS, including the environment in which it operates. The *Twin Object* (TO) ② is a component that is continually kept synchronized with the SuS, with respect to the chosen goals and PoIs. This component may for example be a trained neural network, the execution of code, or a (real-time)simulator for a model in an appropriate formalism. The content of the TO in terms of formalism(s) and model(s) is chosen in stage C. ③ is a so-called Experiment Manager (EM –which can also be seen as an "*Orchestrator*" in the case of black-box components) which enacts a *workflow* ④, a model of *how* the experiment is to be executed. Because the experiment is created for a specific set of requirements, the requirement's logic is contained in ③. For instance, if we only want to visualize the current state, the collection of this state is done by the EM. If instead our goal is Anomaly Detection, the EM needs to compute the distance between the behavior observed by the Actual Object and the behavior computed by the Twin Object –typically over a moving time window– and produce a notification when this distance exceeds a given threshold. ⑤ and ⑥ denote the communication between the EM and the AO or TO, respectively.

The Orchestrator can communicate with a User Agent (7A) or Machine Agent (7B), access points for a user or another system to obtain information about/from or send information to this experiment. At the top of stage B in Figure 2, the top-level architecture for the orchestration of multiple twin experiments is shown. A top-level orchestrator receives input (from top-level User Agent(s) and/or Machine Agent(s)) and spawns new Twinning Experiments. Note that these experiments can be short-running, long-running, or never-ending.

Data gathered from the experiments is stored in the "Historian". This is a blackboard (append-only, to support versioning/traceability) data lake that contains all historical data of all Twinning experiments. Reasoning can be done by querying the Historian. This in turn may spawn new Twinning Experiments. Twinning experiments may call upon the Historian and Reasoner, via the Orchestrator. If the experiment that must be spawned was already carried out in the past, the orchestrator might (driven by input from the "Reasoner" component) collect the answers from the Historian instead (Mittal, Eslampanah, Lima, Vangheluwe, and Blouin 2023). This is shown as an example on the timeline in Figure 5. Some questions obtained from the User/Machine Agent spawn "Reasoning" processes, others launch Twinning Experiments. "Experiment 1" is a short experiment and "Experiment 2" is a never-ending experiment that continually shares information with the orchestrator.

Note that many combinations of real-time and as-fast-as possible simulation are possible: anomaly detection for example compares data obtained through the AO from a SuS with data obtained from the

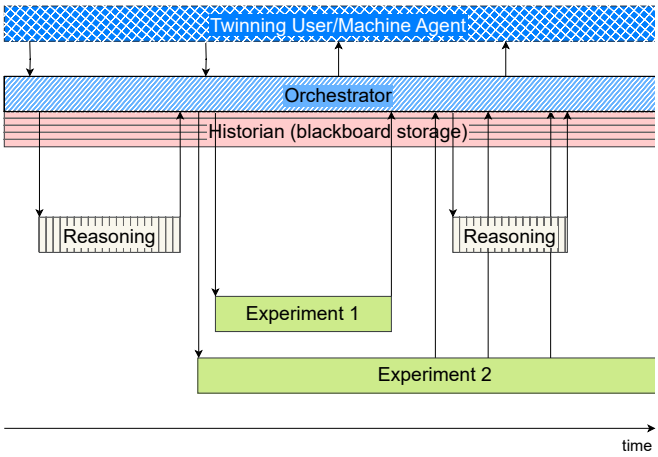


Figure 5: An example trace of Twinning Experiments (from Paredis, 2025).

TO (in the form of a real-time simulation experiment of a model of the SuS) and signals an anomaly when the distance between the Properties of Interest (PoIs) obtained through both AO and TO exceeds a certain threshold. To then determine the cause of this anomaly, many virtual experiments (*i.e.,* as-fast-as possible simulations) may be started, each with a different fault injected into the original model. The virtual experiment that best matches the observed anomalous behavior is then considered to be the cause of the anomaly. This may in turn serve as the basis for a design-space exploration in search of a corrective action.

### 3.3 The Choice of Modeling Formalisms and Models

In this stage, appropriate modeling languages are chosen. Appropriateness pertains to the goals and PoIs of the SuS selected in stage A. How to do this is the expertise of the modeling and simulation community. Although each model is expressed using a specific formalism, no single formalism is universally optimal. Every formalism offers unique strengths and limitations, and selecting the most suitable ones for each aspect of a system is crucial for effective modeling. Multi-Paradigm Modeling (MPM) advocates modeling each component and behavior of a SuS explicitly using the most appropriate formalism(s), viewpoint(s), level(s) of abstraction and tool(s) for the task at hand (Mosterman and Vangheluwe 2004). To support this approach, Mustafiz et al. (2012) introduced a framework for model-based systems engineering that combines a Process Model (PM) with a Formalism Transformation Graph (FTG), serving as a structured map of artifact types (meta-models), transformation activities (contracts), and their interconnections.

The selection of an appropriate model depends largely on the intended purpose and the service it is meant to support. Matta and Lugaresi (2024) suggest to distinguish two main types of virtual representations: (1) model-based and (2) model-free. Depending on the application context, a DT can leverage either or both approaches.

Model-based DTs use models that embed structural and behavioral knowledge of the physical system. These models can include physics-based formulations, Discrete Event Simulation (DES), and analytical models. For instance, in a production system, a reliability model may support predictive maintenance scheduling, while a material flow model may be used when the PoI is the production output under different conditions. These models are particularly valuable for systems governed by clear physical laws or systematic processes, such as mechanical assemblies or discrete-part manufacturing systems. Due to their complexity, however, model-based twins can be computationally intensive, require significant input data, and show longer processing times. As a compromise, meta-modeling techniques such as surrogate modeling are often employed to approximate complex models with simplified versions (*e.g.,* linear or non-linear regression, Kriging, kernel-based models, or neural networks).

Model-free DTs rely mostly on observed historical data rather than on explicit system models. They use data-driven algorithms to extract insights from the data generated by the physical system (i.e., through machine learning). These insights can help identify patterns, detect anomalies, or predict outcomes without a formal model of the system's inner workings. For example, in a manufacturing system, machine learning can be used to identify which parameters most strongly influence throughput or yield, and provide suggestions to improve system performance. Such choices are common in highly variable or complex systems, such as semiconductor manufacturing, where formal modeling may be infeasible.

### 3.4 Deployment Stage

The conceptual architecture from stage B needs to be realized. In this step, frameworks, languages, middleware, communication protocols etc. need to be selected. Common choices are MQTT, DDS, OPC UA, and ROS 2. How to do this is the expertise of the systems engineering and deployment community. The particular choices made here lead to the (software) architectures often seen in the Digital Twin literature. Deployment is essential yet it is also the stage with the most complex and varied set of decisions and its study remains underdeveloped (Hassan and Aggarwal 2023). Jeong et al. (2022) proposed a five-layer model for the deployment stage, offering a step-by-step methodology that emphasizes the implementation process.

Choosing specific tools, frameworks, modeling languages, and technologies often depends on multiple factors, such as the developers' available resources, technical expertise, and economic context. Given a well-defined system architecture and clear objectives, deployment can be approached as a form of simulation-based design exploration. Various technologies may be simulated to identify optimal solutions, provided their models are available. When simulation models exist for a given technology, integration through co-simulation, embedding, or model transformation can ensure representational fidelity. However, if no such models are available, deploying that technology in the simulation requires close interaction between the simulated and physical domains. For example, communication protocols often come with precise behavioral specifications, which makes them easier to replicate in simulations using the same logic. However, building such detailed models is time-consuming and typically focuses on specific components rather than system-level behavior. A common simplification is to represent protocols through fixed delays, but a more accurate approach involves sampling from realistic delay distributions, ideally based on empirical measurements from the actual technology.

## 4 A DISCUSSION ON MODEL VALIDITY

Data collected from the physical system can be analyzed to extract deeper insights into its state. One of the most critical forms of analysis is validation, which ensures that the physical and digital representations are consistent and accurately reflect the same system with respect to the properties of interest (Lugaresi et al. 2023). Once models are calibrated prior to use, they must be validated so ensure that when used in a Digital Twin, they remain valid within the entire operational domain of the system. Validation concerns ensuring that a Digital Twin (DT) remains current and accurately reflects its physical counterpart, particularly in terms of its ability to replicate system behavior –such as performance outputs and parameter trends. Validation focuses primarily on two aspects: the logical structure of the DT and its response to input behaviors (Lugaresi et al. 2023). Online validation plays a crucial role in maintaining synchronization, ensuring that the DT can adapt to unexpected changes in the physical system (Lugaresi et al. 2022; Hua et al. 2022). This continual alignment helps identify significant deviations that cannot be attributed to random variation. In model-based engineering, the validity range of a model is always assumed to encompass the operating range of a system. If this is not the case, any decisions based on the model are meaningless. If an anomaly is detected by a twinning system for example, this could be indicative of a malfunction in the SuS, but it could also mean that the model is used outside its validity range. When it is detected that a model is used outside its validity range, a different model needs to be used. This detection and selection

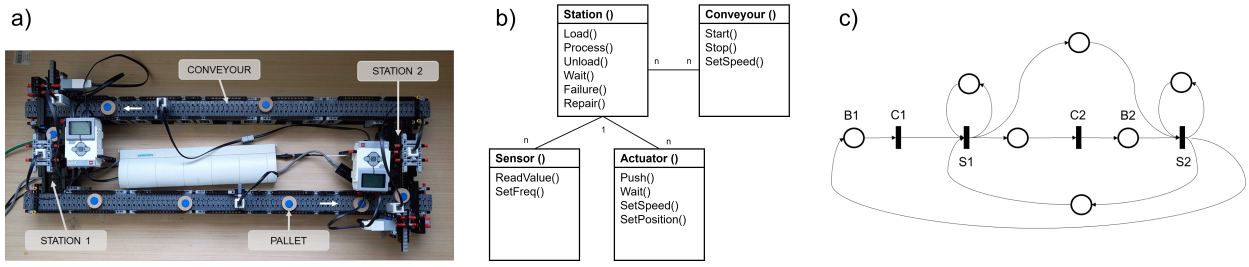


Figure 6: Use Case – a) two-station closed-loop lab-scale system, b) class diagram model of the system, c) Petri Net model (figure from Matta and Lugaresi, 2024).

of a new model is difficult, especially if the operation of the system is time-constrained (Biglari and Denil 2022). A validity frame (Denil et al. 2017; Van Acker et al. 2024) may be used to capture the range of operation within which a model is valid. A validity frame is the possibly infinite set of experiments for which the Properties of Interest obtained through real-world experimentation are close enough (given an appropriate distance metric and a distance threshold) to the same Properties of Interest obtained through virtual experimentation. A distinction is made between this theoretical set, a so-called *abstract validity frame* and a *concrete validity frame*, a finite collection of experiments that yielded close enough PoIs. A *concrete invalidity frame* collects all the experiments that did not yield close enough PoIs.

## 5 USE CASE

A simple production system is proposed as an illustrative example. The use case proposed in Matta and Lugaresi (2024) is revisited from the point of view of the twinning system.

The physical setup consists of a closed-loop two-station, lab-scale manufacturing system, depicted in Figure 6a. The line is assembled using LEGO Mindstorms components (Lugaresi et al. 2021). Each station is consists of an EV3 intelligent brick, three optical sensors, a motor, and a part entry system. The EV3 serves as the station's controller, interfacing with both sensors and actuators via a python script (Bacelar dos Santos and Chalissery Lona 2023). The line operates under the assumption of an unlimited supply of parts and negligible loading/unloading times. Each station processes one pallet at a time, holding it for a duration corresponding to the physical processing operation. In the event of a failure, the pallet remains in the station for an extended period until repair is completed. Pallets are transferred between stations by conveyors, which also act as buffers where pallets may queue if the next station is occupied. Indeed, a station can only release a pallet if there is available space in the downstream buffer. Both stations operate under a blocking after service policy. The processing times at the two stations follow triangular distributions with parameters $(3,5,8)$ and $(2,3,5)$ seconds, respectively. Each buffer accommodates up to 8 pallets, with 12 pallets circulating in the system overall. This configuration effectively emulates real-world manufacturing dynamics, including stochastic behavior, blocking, and potential deadlocks. In the following, it is detailed how each step of the twinning system workflow in Figure 2 has been instantiated for this specific use case.

### 5.1 Stage A - Goal Setting

With reference to Figure 3, this use case concentrates on several operational goals: (1) *State Estimation*, and the consequent Data Recording, allowing for a collection of system states to highlight trends and spot significant deviations from the desired states; (2) *Forecasting* via what-if scenario analyses, in particular following disruptive events that require prompt corrective decision making, and the consequent (3) *Production Control*, where the system behavior is steered. A set of properties of interest for this system is selected, consistent with the system-level production control granularity. In particular, the *material flows* (work-in-progress) are traced, and the *production throughput* is recorded (*e.g.,* pallets per hour).

### 5.2 Stage B - (Conceptual) Architecture and Experiments Setting

In this use case, a realistic production situation is taken as reference: a production plan is already in place, and at a certain point, the condition of Station 2 begins to degrade. Its processing time slows down, now following a triangular distribution with parameters $(9, 14, 11)$. The Digital Twin then analyzes the situation and explores possible countermeasures to address the issue. The experiment manager uses the virtual entity to perform a what-if analysis. For this simple case, the analysis is conducted on two alternatives: (1) *do nothing*, keep producing at a slower pace and repair the station at the end of the shift; (2) *react*, stop the plant to allow repairing activities and then continue with the production pace before the slow-down. With reference to Figure 5, it is possible to identify two main experiments in this use case. Experiment 1 is a finite horizon simulation where the system performance in the remaining part of the production shift is predicted. Experiment 2 is a long term simulation where the main goal is monitoring the alignment (i.e., validation) between the physical and digital systems (Lugaresi et al. 2023).

### 5.3 Stage C - Choice of Formalisms and Model Building

The choice of formalisms is done with reference to the goals and PoIs identified in stage A.

For *state estimation*, an accurate descriptive model of the system possible states and behavior is needed. The class diagram in Figure 6b depicts the physical entity model at a type level, developed using the latest available knowledge about the system. This model uses UML class diagrams to represent key components and associations between them as well as their multiplicities: (1) the stations, namely finite production resources, (2) the conveyors, (3) the sensors, and (4) the actuators.

To allow *forecasting*, in this case a discrete event model of the system is created. The model is built using the Petri net formalism, where transitions represent station processing activities and places correspond to pallet locations, *e.g.,* in stations or on conveyors, as shown in Figure 6c. A discrete event simulation (DES) software can be used to implement the logic described by the Petri Net. It is worth noticing that the simulation model does not have to mirror the physical model exactly; abstractions and simplifications are often introduced during modeling. For instance, transport between stations is approximated using a dummy station with a fixed processing time. As a virtual entity, the simulation model may support a larger set of goals and useful services, such as forecasting end-of-shift performance under current conditions and diagnostic assessments.

### 5.4 Stage D - Deployment

In the deployment stage, the major decisions pertain to technologies and tools that implement the chosen architecture and the related models.

As identified in section 5.1, the goal of state estimation requires data recording, and the proper tools must be selected. In this case, data is stored in a real-time database built in *InfluxDB*, a tool that enables both data recording and real-time monitoring. For instance, the identifier of the last processed pallet by a station can be found via a query to the database component. For convenience, data resulting from analyses and predictions such as throughput and system time are stored in the same database . Hence, this tool is appropriate for the second goal, *forecasting*.

All the identified goals necessitate the realization of connections between different system components. In this case, the MQTT messaging protocol is used to collect data and control the system. Each message has a specific structure, based on its aim. For instance, a data-collecting message indicating an activity starting in a station is written as the dictionary $\{\text{“activity"} : s, \text{“id"} : id, \text{“ts"} : time, \text{“tag"} : \text{“s"}\}$, where $s$ is a variable indicating the station number, $id$ indicates the identifier of the pallet, $time$ the event time-stamp in UNIX format, and $tag$ is a string indicating the activity performed in the station. Other specific message structures are introduced to encode control signals to the physical actuators, thus enabling online prescriptions (Lugaresi et al. 2021).

Finally, the twinning system is deployed, and online scenario evaluation is carried out within a controlled laboratory setting. In this case, the DES model performing the what-if scenario experiments is implemented in *Simpy*, Two separate simulation experiments are run to assess which alternative yields the highest production output for the remainder of the shift. The predicted outcomes indicate an average of $165 \pm 3$ parts for the first scenario, and $209 \pm 3$ parts for the second. As a result, the second scenario is chosen, and the corresponding prescription is forwarded for execution.

## 6 CONCLUDING REFLECTIONS

This work has provided an overview of the (digital) twinning systems, with the aim to familiarize readers with the evolving nature and potential of this paradigm. However, the main limitations of this work should be acknowledged. This paper has not covered several important concepts that merit deeper exploration, such as cyber-security an dealing with change in for example requirements, or technology. Also, the perspectives in this paper are biased by the authors' expertise and experience. Ongoing research is essential to overcome the various challenges that currently hinder the effective deployment and scalability of DTs. Among these are architectural challenges: a standardized, practical DT architecture is still lacking, one that supports both reusable components (*e.g.,* data exchange interfaces) and application-specific services. To date, numerous DT architectures have been proposed, but these are often tailored to specific domains and use cases. Consequently, DT development is highly context-dependent, with decisions about architecture, communication protocols, and data management tightly linked to the application requirements. Additionally, integration challenges emerge once DTs are operational, particularly in maintaining seamless coordination between physical and digital entities. Model validity presents another critical issue. Several DT implementations rely on offline model validation methods, which often assume static conditions and involve extensive experimentation. DT applications are often developed for isolated components rather than for entire systems (of systems). This piecemeal approach can limit the broader benefits of DTs, especially in complex environments such as supply chains, where sensing and control must extend beyond individual assets. Last but not least, a more rigorous theoretical foundation for DTs is still missing. Formal definitions of the functional mappings between physical and digital systems are needed—clarifying their scope, mathematical structure, and domain-specific applicability. Advancing this foundational understanding will be key to unlocking the full potential of twinning across sectors.

## ACKNOWLEDGMENTS

This work is partially based on the Ph.D. Thesis of Randy Paredis (Paredis 2025) and the previous related tutorials (Matta and Lugaresi 2023; Matta and Lugaresi 2024) .

## REFERENCES

Allen, B. 2021, November. "ASME Digital Twin Summit keynote: Digital Twins and Living Models at NASA". Technical Report 20210023699, Langley Research Center Hampton, Virginia, United States.

Aydt, H., S. J. Turner, W. Cai, and M. Y. H. Low. 2008. "Symbiotic Simulation Systems: an Extended Definition Motivated by Symbiosis in Biology". In *2008 22nd Workshop on Principles of Advanced and Distributed Simulation*, 109–116. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Barricelli, B. R., E. Casiraghi, and D. Fogli. 2019. "A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications". *IEEE Access* 7:167653–167671 https://doi.org/10.1109/ACCESS.2019.2953499.

Biglari, R., and J. Denil. 2022. "Model Validity and Tolerance Quantification for Real-time Adaptive Approximation". In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 668–676. New York, NY, USA: Association for Computing Machinery https://doi.org/10.1145/3550356.3561604.

Croatti, A., M. Gabellini, S. Montagna, and A. Ricci. 2020. "On the Integration of Agents and Digital Twins in Healthcare". *Journal of Medical Systems* 44:1–8.

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

Davis, W. J. 1998. "On-line Simulation: Need and Evolving Research Requirements". In *Handbook of Simulation*, edited by J. Banks, 465–516. New York, United States: John Wiley and Sons, Inc.

Denil, J., S. Klikovits, P. J. Mosterman, A. Vallecillo, and H. Vangheluwe. 2017. "The Experiment Model and Validity Frame in M&S". In *Proceedings of the Symposium on Theory of Modeling & Simulation*, TMS/DEVS '17. San Diego, CA, USA: Society for Computer Simulation International.

Dertien, S., and C. Macmahon. 2022. "State of Digital Twin 2022". Parametric Technology Corporation (PTC). https://www.ptc.com/en/industry-insights/digital-twin, accessed 30th September, 2024.

Bacelar dos Santos and Chalissery Lona 2023. "DtwinPy GitHub Repository". https://github.com/DtwinPy-Team, accessed May 3rd 2025.

Fortune Business Insights. 2023. "Digital Twin Market Report". https://www.fortunebusinessinsights.com/digital-twin-market-106246, accessed 30th September, 2024.

Fujimoto, R., W. Lunceford Jr, E. H. Page, and A. Uhrmacher. 2021. "Grand Challenges for Modelling and Simulation (Dagstuhl Seminar 02351)". Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Fuller, A., Z. Fan, C. Day, and C. Barlow. 2020. "Digital Twin: Enabling Technologies, Challenges and Open Research". *IEEE access* 8:108952–108971.

Gelernter, D. 1993. *Mirror Worlds: or the Day Software Puts the Universe in a Shoebox... How it Will Happen and What it Will Mean*. Oxford, United Kingdom: Oxford University Press.

Grieves, M. 2014. "Digital Twin: Manufacturing Excellence Through Virtual Factory Replication". *White paper* 1(2014):1–7.

Grieves, M., and J. Vickers. 2017. "Digital twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems". In *Transdisciplinary perspectives on complex systems*, 85–113. Springer.

Guizzardi, G. 2007. "On Ontology, Ontologies, Conceptualizations, Modeling Languages, and (Meta)Models". In *Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006*, 18–39. NLD: IOS Press.

Hassan, A. A., and G. Aggarwal. 2023. "Sustainable Manufacturing: Digital Twinning for a Mechanical Assembly Production Line". In *2023 IEEE Smart World Congress (SWC)*, 758–763. IEEE.

Hua, E. Y., S. Lazarova-Molnar, and D. P. Francis. 2022. "Validation of Digital Twins: Challenges and Opportunities". In *2022 Winter Simulation Conference (WSC)*, 2900–2911 https://doi.org/10.1109/WSC57314.2022.10015420.

Industry IoT Consortium 2022. "The Industrial Internet Reference Architecture". Accessed: April 5th 2024.

ISO 23247:2021. 2021. "Automation Systems and Integration — Digital Twin Framework for Manufacturing — Part 1: Overview and General Principles". Standard, International Organization for Standardization, Geneva, Switzerland.

Jafari, M., A. Kavousi-Fard, T. Chen, and M. Karimi. 2023. "A Review on Digital Twin Technology in Smart Grid, Transportation System and Smart City: Challenges and Future". *IEEE Access* 11:17471–17484.

Jeong, D.-Y., M.-S. Baek, T.-B. Lim, Y.-W. Kim, S.-H. Kim, Y.-T. Lee, *et al*. 2022. "Digital Twin: Technology Evolution Stages and Implementation Layers with Technology Elements". *Ieee Access* 10:52609–52620.

Jones, D., C. Snider, A. Nassehi, J. Yon, and B. Hicks. 2020. "Characterising the Digital Twin: A Systematic Literature Review". *CIRP Journal of Manufacturing Science and Technology* 29:36–52 https://doi.org/https://doi.org/10.1016/j.cirpj.2020.02.002.

Kang, K. C., and H. Lee. 2013. "Variability Modeling". In *Systems and software variability management*, 25–42. Springer.

Kirschenbaum, M. 2023. "Granular Worlds: Situating the Sand Table in Media History". *Critical Inquiry* 50(1):137–163 https://doi.org/10.1086/726299.

Kritzinger, W., M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. "Digital Twin in Manufacturing: A Categorical Literature Review and Classification". *IFAC-PapersOnLine* 51(11):1016–1022.

Lugaresi, G., V. V. Alba, and A. Matta. 2021. "Lab-scale Models of Manufacturing Systems for Testing Real-time Simulation and Production Control Technologies". *Journal of Manufacturing Systems* 58:93–108.

Lugaresi, G., S. Gangemi, G. Gazzoni, and A. Matta. 2022. "Online Validation of Simulation-Based Digital Twins Exploiting Time Series Analysis". In *2022 Winter Simulation Conference (WSC)*, 2912–2923 https://doi.org/10.1109/WSC57314.2022.10015346.

Lugaresi, G., S. Gangemi, G. Gazzoni, and A. Matta. 2023. "Online Validation of Digital Twins for Manufacturing Systems". *Computers in Industry* 150:103942.

Matta, A., and G. Lugaresi. 2023. "Digital Twins: Features, Models, and Services". In *2023 Winter Simulation Conference (WSC)*, 46–60. IEEE.

Matta, A., and G. Lugaresi. 2024. "An Introduction to Digital Twins". In *2024 Winter Simulation Conference (WSC)*, 1281–1295. IEEE.

Michael, J., L. Cleophas, S. Zschaler, T. Clark, B. Combemale, T. Godfrey, *et al*. "Model-Driven Engineering for Digital Twins: Opportunities and Challenges". *Systems Engineering* https://doi.org/https://doi.org/10.1002/sys.21815.

Minerva, R., G. M. Lee, and N. Crespi. 2020. "Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models". *Proceedings of the IEEE* 108(10):1785–1824 https://doi.org/10.1109/JPROC.2020.2998530.

Mittal, R., R. Eslampanah, L. Lima, H. Vangheluwe, and D. Blouin. 2023. "Towards an Ontological Framework for Validity Frames". In *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 801–805. IEEE.

Mosterman, P. J., and H. Vangheluwe. 2004. "Computer Automated Multi-paradigm Modeling: An Introduction". *Simulation* 80(9):433–450.

Mustafiz, S., J. Denil, L. Lúcio, and H. Vangheluwe. 2012. "The FTG+ PM Framework for Multi-Paradigm Modelling: An Automotive Case Study". In *Proceedings of the 6th International Workshop on Multi-paradigm Modeling*, 13–18.

Negri, E., L. Fumagalli, and M. Macchi. 2017. "A Review of the Roles of Digital Twin in CPS-based Production Systems". *Procedia Manufacturing* 11:939–948.

Paredis, R. 2025. *A multi-paradigm modelling foundation for twinning within the context of systems engineering*. Ph. D. thesis, University of Antwerp.

Paredis, R., C. Gomes, and H. Vangheluwe. 2021. "Towards a Family of Digital Model / Shadow / Twin Workflows and Architectures". In *Proceedings of the 2nd International Conference on Innovative Intelligent Industrial Production and Logistics (IN4PL 2021)*, 174–182: SCITEPRESS – Science and Technology Publications, Lda.

Paredis, R., and H. Vangheluwe. 2024. "Modelling and Simulation-Based Evaluation of Twinning Architectures and Their Deployment". *Proceedings of the 14th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*:170–182 https://doi.org/10.5220/0012865300003758.

Patil, S., S. Mishra, N. Gobi, I. Alam, R. Jain *et al*. 2024. "Fortifying Connected Vehicles Based Cybersecurity Measures for Secure Over-the-Air Software Updates.". *Journal of Cybersecurity & Information Management* 14(1).

Plesser, H. E. 2018. "Reproducibility Vs. Replicability: A Brief History Of A Confused Terminology". *Frontiers in neuroinformatics* 11:76.

Proper, H. A., and G. Guizzardi. 2024, September. "Understanding the Variety of Domain Models: Views, Programs, Animations, and Other Models". *SN Comput. Sci.* 5(7) https://doi.org/10.1007/s42979-024-03163-y.

Rasheed, A., O. San, and T. Kvamsdal. 2020. "Digital Twin: Values, Challenges and Enablers from a Modeling Perspective". *IEEE access* 8:21980–22012.

Schleich, B., N. Anwer, L. Mathieu, and S. Wartzack. 2017. "Shaping the Digital Twin for Design and Production Engineering". *CIRP Annals* 66(1):141–144.

Singh, R. 1996. "International Standard ISO/IEC 12207 Software Life Cycle Processes". *Software Process Improvement and Practice* 2(1):35–50.

Tao, F., F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, *et al*. 2019. "Digital Twin-driven Product Design Framework". *International Journal of Production Research* 57(12):3935–3953.

Van Acker, B., P. D. Meulenaere, H. Vangheluwe, and J. Denil. 2024. "Validity Frame–enabled Model-Based Engineering Processes". *Simulation* 100(2):185–226 https://doi.org/10.1177/00375497231205035.

Van der Valk, H., H. Haße, F. Möller, M. Arbter, J.-L. Henning, and B. Otto. 2020. "A Taxonomy of Digital Twins". In *AMCIS*, 1–10. Salt Lake City, USA.

Wanasinghe, T. R., L. Wroblewski, B. K. Petersen, R. G. Gosine, L. A. James, O. De Silva, *et al*. 2020. "Digital Twin for the Oil and Gas industry: Overview, Research Trends, Opportunities, and Challenges". *IEEE Access* 8:104175–104197.

Xiong, M., and H. Wang. 2022. "Digital Twin Applications in Aviation Industry: A Review". *The International Journal of Advanced Manufacturing Technology* 121(9-10):5677–5692.

## AUTHOR BIOGRAPHIES

**GIOVANNI LUGARESI** is Assistant Professor at the Department of Mechanical Engineering of KU Leuven (Belgium). He works on production planning and control, process mining, and robust optimization. His email is giovanni.lugaresi@kuleuven.be.

**HANS VANGHELUWE** is a Professor at the University of Antwerp (Belgium) where he heads the Modeling, Simulation and Design Lab (MSDL). MSDL, one of the labs in the Antwerp Systems and Software Modelling (AnSyMo) research group, is part of Flanders Make, the Strategic Research Centre for the Flemish manufacturing industry. His research is on multi-paradigm modeling of complex, cyber-physical systems using Twinning. His e-mail address is Hans.Vangheluwe@uantwerpen.be.