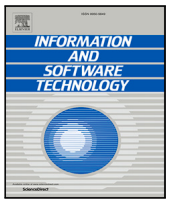
Information and Software Technology 198 (2026) 108227

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Interaction patterns between Artificial Intelligence and Digital Twins in the industrial domain

Matteo Martinelli [a,*], Marco Lippi [b], Marco Picone [a], Stefano Mariani [a]

[a] DISMI, University of Modena and Reggio Emilia, via Amendola 2 - 42122 Reggio Emilia (RE), Italy
[b] DINFO, University of Florence, Via di S. Marta, 3 - 50139 Firenze (FI), Italy

ARTICLE INFO

Dataset link: https://github.com/lian-yue0515/ACE-LIO

Keywords:
Digital Twins
Smart manufacturing
Cyber-physical systems
Artificial Intelligence

ABSTRACT

Context: The adoption of Artificial Intelligence (AI) in industrial production systems has raised significant expectations for increased efficiency and innovation. Nevertheless, challenges such as the distributed nature of industrial operations, the heterogeneity of physical devices, and the complexity of real-world processes continue to hinder AI integration. Digital Twins (DTs) have emerged as a promising abstraction to decouple physical complexity from digital representations, facilitating more effective system management.

Objective: This work investigates how AI can be systematically integrated with DTs in industrial contexts. The goal is to identify and characterize a set of interaction patterns that leverage the complementary strengths of AI and DTs to enhance industrial intelligence and performance.

Methods: Drawing on a structured view of how responsibilities can be shared between AI technologies and DT-enabled shop floors, the paper defines four interaction patterns—AI Observing DTs, AI Advising DTs, AI Controlling DTs, and AI Embedded in DT. Each pattern is analyzed in terms of its roles, data and control flows, and typical application scenarios, and is illustrated on a DT-enabled physical micro-factory that reproduces realistic production conditions.

Results: The four patterns show how different placements and responsibilities of AI components with respect to DT layers impact modularity, reuse of AI models, maintainability, and integration with legacy industrial systems. The micro-factory illustration highlights how the patterns can support practical use cases, including root-cause analysis of performance degradation, machine-level health monitoring, and AI-based production scheduling.

Conclusion: Structuring AI–DT integration around interaction patterns provides a concrete way to bridge the gap between conceptual opportunities and operational industrial systems. The proposed patterns offer a reusable design vocabulary for positioning AI with respect to DT layers in cyber–physical production systems, and for reasoning about the architectural trade-offs of alternative integration strategies.

1. Introduction

Modern industrial production systems face growing pressure to manage complexity, variability, and uncertainty across heterogeneous assets, legacy software, and distributed plants [1,2]. Companies must continuously adapt layouts, equipment, and processes to volatile market demands, while at the same time optimizing resource usage, guaranteeing delivery times, and promptly reacting to failures or unforeseen events without disrupting operations [3,4]. Within this context, the industrial domain is increasingly looking for technologies that can reshape how the physical shop floor is observed, controlled, and made intelligent.

Among these technologies, Artificial Intelligence (AI) and Digital Twins (DTs) have emerged as key enablers for Cyber-Physical Production Systems (CPPSs) [5]. AI techniques promise advanced analytics, predictive capabilities, and decision support across logistics, quality, scheduling, ergonomics, and maintenance tasks [6–8], but they often remain confined to siloed solutions tailored to specific problems [5]. DTs offer a structured digital representation of physical assets and processes [9,10], acting as an abstraction and orchestration layer that collects, contextualizes, and organizes shop-floor data while mediating interactions back to the physical world [11].

Recent research has thus started integrating DTs and AI with the aim of "making physical systems intelligent" by first digitalizing them

* Corresponding author.
E-mail address: martinellim@unimore.it (M. Martinelli).

https://doi.org/10.1016/j.infsof.2026.108227
Received 16 September 2025; Received in revised form 5 June 2026; Accepted 6 June 2026
Available online 13 June 2026

and then attaching AI modules that exploit the resulting data [12–14]. This trend is particularly evident in industrial applications, where DTs are used as a gateway for AI-driven monitoring, prediction, and control across different levels of the CPPS stack. Yet, existing works often propose their own ad-hoc approach for coupling AI techniques and DT components, focusing on specific use cases or sectors, and rarely questioning whether the chosen integration strategy is conceptually sound and reusable [5,6]. As a consequence, the literature provides valuable examples of AI-enhanced DT solutions, but it lacks a proper understanding of how AI and DTs should interact in a systematic and scenario-aware manner, especially in presence of heterogeneous and retrofitted legacy systems.

This work aims to address this gap by introducing four interaction patterns between AI technologies and DT-enabled shop floors in the industrial domain. These patterns capture how responsibilities can be distributed between DTs and AI along the spectrum from pure observation to direct control, clarifying where intelligence is located, how it consumes DT-managed data, and how it affects the physical system. Each pattern is defined in terms of its main roles, data and control flows, and typical application scenarios, highlighting strengths and limitations with respect to modularity, separation of concerns, ease of AI management, and compatibility with existing industrial software ecosystems.

To ground these conceptual patterns into a realistic environment, we provide an experimental illustration based on a physical micro-factory that reproduces key characteristics of real-world shop floors, such as heterogeneous equipment, multiple stations, and non-trivial material flows. The micro-factory is equipped with a DT layer that digitalizes machines and processes, while different AI modules are instantiated to realize concrete scenarios like root-cause analysis, predictive maintenance, and scheduling, each configured according to one of the proposed patterns. In this way, the micro-factory serves as a "validation by instantiation" of the patterns, showing how they can be effectively applied, combined, and reconfigured in practice rather than remaining pure abstractions. Table 4 provides a comparative overview of the four AI–DT interaction patterns, summarizing their main responsibilities, typical data and control flows, and representative industrial use cases.

Finally, the paper discusses the contribution of these four patterns in light of existing DT and AI literature, and of a broader responsibility map that positions DTs and AI across key industrial digitalization tasks. We argue that the patterns provide a conceptual lens to reason about AI–DT integration choices, to classify existing and future solutions, and to support architects in selecting appropriate configurations for specific industrial scenarios, including those involving complex legacy systems. In particular, the discussion reflects on how the patterns relate to existing DT architectures, how they can help isolate or mitigate constraints imposed by legacy Manufacturing Execution System (MES), Enterprise Resource Planning (ERP), and Process Control System (PCS), and how they could scale beyond the micro-factory to larger, multi-site, and cross-sector settings. In line with this red thread, the remainder of the article (i) reviews DT and AI integration works, and motivates the need for an integrated view of interaction patterns; (ii) presents the four patterns in detail; (iii) illustrates their instantiation in the micro-factory, and (iv) discusses their value, limitations, and implications for future research and industrial practice.

Building on these motivations and objectives, this work addresses the following Research Questions (RQs):

- **RQ1.** How can the mutual positioning of AI and Digital Twins in industrial CPSs — with respect to the locus of AI, the directionality of influence, and the scope of the operational context — be structured into a set of reusable interaction patterns?
- **RQ2.** How do the resulting AI–DT interaction patterns, characterized by different combinations of locus, influence directionality, and context scope, impact architectural properties such as modularity, maintainability, reuse and integration of legacy systems in the industrial CPSs domain?

2. Related works

The concept of DT has been implemented following different principles, but always sharing the core idea of transposing the physical context into the digital one — and possibly the other way round. The idea of DT was introduced between 1999 and 2002 by Michael Grieves within Product Lifecycle Management (PLM) [15], then iteratively refined [16,17]. From its conceptualization, several attempts to engineer DTs have led to the definition of different architectural patterns, each with specific goals [18–20]. Nonetheless, these works focus only on internal DT architectures and their CPPS relations, not on AI integration architecture.

AI techniques, in fact, besides being used in the industrial domain for many years [21,22], lately have been frequently adopted together with DTs [23–25], mostly *inside* DTs to improve data pre-processing or to enhance DT functionalities with some form of intelligence (usually, prediction capabilities). Despite general awareness of the potential for mixing the concept of AI and DT (interest in this regard is exposed in [13,26–28]), few articles partially analyze the possible interaction patterns between AI and DT from a software architecture perspective, and they do so from very specific standpoints. Authors in [29] propose the concept of a *micro-architecture* to describe the possible ways in which DTs and autonomous software agents (that are a specific form of AI) can be combined according to the design principles therein introduced as well. However, the paper is focused on the agent abstraction to encapsulate AI, not on AI techniques in general, and does not consider the inherent complexities and peculiarities of industrial deployments as the main source for requirements elicitation. An alternative perspective is offered by [25], where authors explore AI technologies applied *inside* DTs in the scientific literature. The findings highlight that AI is applied mostly for optimization, classification, and regression at a research level, focusing mostly on historical data gathered by DTs and lacking effective feedback loops between the physical and the digital domain. Despite the valuable contribution, interaction patterns other from having an AI model living within a DT are neglected. The interest in mixing AI and DTs technologies is also reported in [30], where AI-based Digital Twins for adaptive simulations in industrial design and manufacturing are presented, modeled and discussed, with a focus on low-latency, real-time responses. Similarly, in [31] the mutual benefit of combining DT applications with modern Generative-AI (Gen-AI) solutions is discussed, still considering an AI *living inside the DT*. Another loosely related work is [5], where authors focus mostly on how AI is used for data acquisition and validation, first and foremost, and then for data analytics and model building. In our view, data acquisition and validation should rather be mostly a DT responsibility, as motivated in Section 3.

The interest in classifying potential interaction patterns between an AI positioned inside a DTs is supported by contributions considering this scenario [25,31], but the involvement of external AIs interacting with a layer of multiple DTs can be interesting and useful as well [29], especially in the industrial domain. MES and ERP, as well as activities schedulers, are practical examples of applications integrating an incremental amount of intelligent components and modules [32–34]. To achieve their task most effectively, such software is interested in both the state of single physical system entities, information collected and organized by DTs themselves, as well as in the layout, architectural state, and higher-level information of the shop floor. However, most of these works primarily address the integration of AI into industrial software components, without considering how the relevant information is gathered from the physical world, thus leaving out the role of DTs entirely.

More in general, several forms of interaction can arise between AI models and industrial DTs, both involving AI living *inside* a single DT, and a single or group of DTs interacting with an AI *external* to them. An alternative point of view endorsing interaction patterns different from only having an AI internal to a DT, comes from the cyber-security

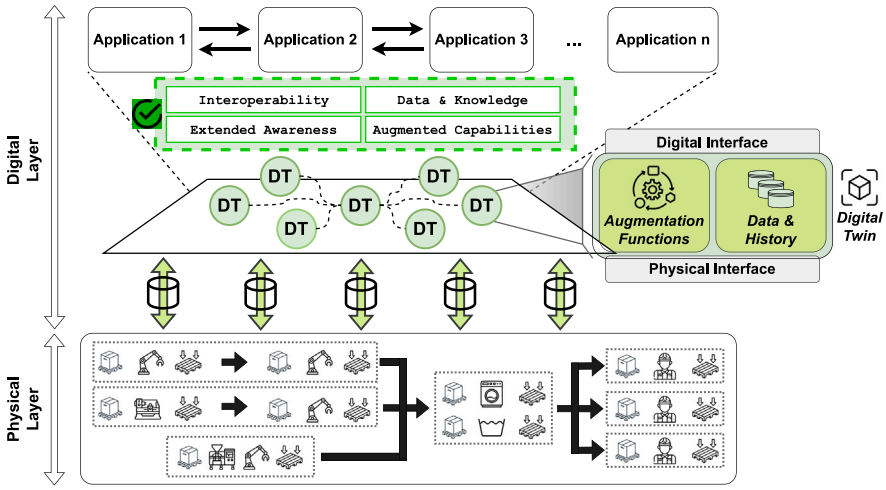


Fig. 1. Digital Twins as a mediating layer between the physical shop floor and industrial applications, providing digital interfaces, data management, interoperability, augmentation functions, and extended physical awareness to higher-level services.

field [35,36]. These studies explore AI-based security tools observing a set of DTs, as in the case of intrusion detection systems, to improve the security of future CPSs.

Before moving on to the interaction patterns we foster for a conceptually and practically sound grounded integration of AI with DTs in CPPSs, the next section further discusses the motivations behind such an integration. This section also serves to clarify the different ways in which AI and DTs can interact, which will then be implemented by the patterns proposed in Section 4.

3. Motivation

In this section, we analyze the impact of AI in production management, highlighting contributions and drawbacks, and remarking on the need for an additional step for the effective integration of AI in the considered context. Then, we add DTs to the picture, to clarify how they can complement AI and be completed in turn.

3.1. The challenges of using AI in industry

In the attempt of improving physical systems with AI, several applications have already been proposed in the industrial context, ranging from fuel and oil consumption forecasting (e.g., in the maritime sector [14]), through end time prediction of manufacturing jobs to tighten logistic operations (e.g., see [6] and citations therein), to the construction of production processes representation with data mining techniques [37]. The vast majority of these works are limited to the *physical* and *specialized* nature of modern manufacturing systems, which lack a proper and generalized representation in the digital domain. The strong specialization of AI solutions makes it difficult to reuse established approaches across different physical domains, even if they share similar issues [5].

Challenge 1: AI specialization. Specialized AI solutions are the tip of the iceberg in the problem of bringing AI into industrial systems, fostering the so-called "Industrial AI" (IAI) [38]. Attempts to apply AI to industrial scenarios are already a matter of fact, but its widespread adoption has not yet been reached. Many limitations contribute to hindering IAI diffusion.

Challenge 2: data quality. A second, pervasive challenge concerns data quality. Industrial environments often involve noisy signals, missing values, and inconsistent logging, so that AI models depend not only on large volumes of data but also on rigorous curation and cleaning [39–41]. Recent work on data-centric AI highlights that model performance and reliability are tightly constrained by data quality, and that even moderate noise or label errors can lead to degraded accuracy and systematic biases in downstream decisions [40,41]. For industrial AI, this means that limitations of AI models cannot be separated from limitations of the data pipelines that feed them, and that poor data can easily turn AI-based decision support into a source of inefficiencies rather than a remedy [5].

Challenge 3: physical heterogeneity. Industrial systems typically integrate diverse equipment selected for performance and cost-effectiveness, making it unlikely that heterogeneity will decrease over time [38,42]. Even among devices of the same category, data formats and access methods can differ significantly, hampering interoperability and complicating the consistent application of AI models.

In this setting, DTs become essential. As dynamic virtual counterparts of physical systems and their components, they decouple real-world complexity (there including heterogeneity) and hyper-specialization from its virtual representation, allowing incoming data to be processed through internal models and according to the data quality standards desired. This enables both the pre-processing of data into actionable insights for industrial software and the embedding of intelligence within the DT itself via AI models. The synergy between AI and DT proves especially valuable in cyber–physical production systems, where DTs help bridge the gap between AI requirements and the specific constraints of industrial environments.

3.2. Reconciling physical and digital worlds with Digital Twins

In fact, DTs function as an abstraction layer for the underlying physical systems, organizing and structuring the continuous flow of information from real-world assets while providing tools to efficiently process and prepare upcoming data for AI-driven tasks. Their capabilities — including interacting directly with physical assets, enabling seamless collaboration among physical entities by removing data silos, and enhancing the physical systems' functionalities through internal models — represent a significant advancement towards greater *data quality* and *homogeneity* within industrial systems.

Because of the DTs' role depicted so far, they shall act as an intermediate *layer* between the physical world and the digital applications that need to observe, analyze, and control it, rather than as an additional service placed on top of existing systems [5]. In this configuration, DTs expose to higher-level software a coherent, contextualized, and manipulable view of shop-floor entities, relieving domain applications from the burden of directly accessing heterogeneous devices, field buses, and control systems. Fig. 1 illustrates the concept of a DT-Layer mediating between the physical domain and multiple applications,

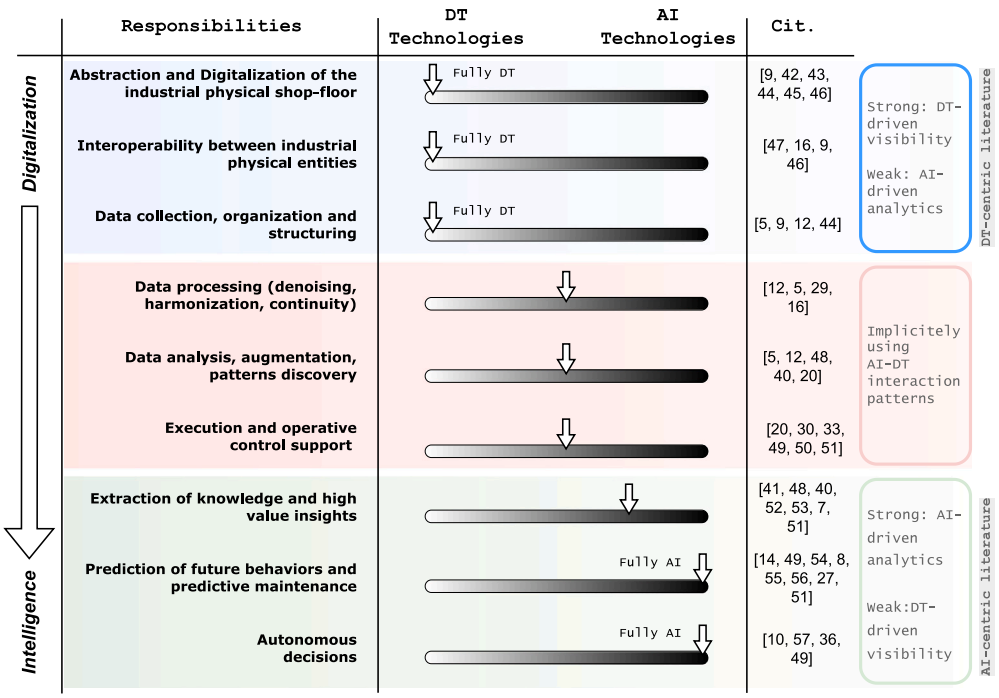


Fig. 2. Layered allocation of industrial shop-floor responsibilities between Digital Twin and AI technologies, from digitalization tasks to intelligence-oriented functions.

providing interfaces, data and history, interoperability, augmentation functions and extended physical awareness to upper level applications, thus making the underlying shop floor digitally accessible. In this regard, DTs support a clear division of responsibilities with AI. DTs are in charge of the *abstraction*, *manipulation*, and *organization* of data coming from physical entities, ensuring that the digital system remains scalable as the underlying physical counterpart evolves; AI models, in turn, are responsible for consistently extracting knowledge and high-value information from DT-prepared data, emulating human intelligence to better accomplish the tasks for which they are designed [5]. In particular, the data handling process undertaken by DTs can be explicitly shaped around the requirements of AI models, thereby minimizing or even eliminating the drawbacks of industrial data being intermittent, heterogeneous, and noisy [31].

While DTs are primarily responsible for data abstraction and preprocessing, and AI model act only on data, DTs operations themselves can be enhanced through the selective integration of AI techniques after having satisfied the structured physical-data-ingestion problem. For instance, AI models can support data cleaning, anomaly detection, interpolation of missing values, and semantic alignment across heterogeneous data sources [31]. By embedding lightweight or specialized AI modules within the DT-Layer, the pre-processing pipeline becomes more adaptive and intelligent, especially in scenarios with high data variability or uncertainty, ensuring that the quality of data reaching higher-level AI applications is maximized and further strengthening the DT–AI collaboration.

In return, AI systems' advanced capabilities can improve DT tasks for industrial operations, offering tools for advanced analytics, inference, and behavior fueled by data received and organized by DT components. The resulting system is expected to bridge the gap between existing MES, ERP, and PCS systems, finally supporting industrial knowledge workers, system visibility, and advanced optimization.

3.3. Motivation summary

The widespread adoption of Artificial Intelligence in industrial production has long been expected, but its deployment still struggles with structural limitations of current shop-floor systems [5,22,25]. Industrial environments are heterogeneous, noisy, distributed, and frequently reconfigured, which makes the direct application of AI to raw physical data hard to sustain in practice [1,3,5]. Three main challenges emerge: (i) the strong *specialization* of AI solutions to specific physical deployments, which hampers their transfer and reuse; (ii) pervasive *data-quality issues* due to noise, missing values, and inconsistent logging; and (iii) marked *physical heterogeneity* of devices, interfaces, and protocols, which obstructs uniform and scalable access to shop-floor information.

Digital Twins are increasingly recognized as the key technology to address these challenges and to provide a robust entry point for AI in Cyber-Physical Production Systems. As summarized in Fig. 2, DTs are responsible for *abstraction and digitalization* of the shop floor, *interoperability* among industrial entities, and *data collection, organization, and structuring* [23,32]. In essence, DTs explicitly *model* the physical contexts to which they are connected, continuously receiving, harmonizing, and organizing shop-floor data so as to expose a coherent, high-quality digital representation of the underlying assets. AI technologies, instead, operate only on data — irrespective of their physical origin — and exploit this DT-mediated visibility to perform advanced analytics, prediction, and decision-making [17]. This separation of concerns lets DTs cope with physical complexity and heterogeneity, while AI focuses on extracting value from the prepared data.

The literature-inspired distribution of responsibilities in Fig. 2 shows this transition from *visibility* to *intelligence* along the vertical axis. Activities devoted to *abstraction and digitalization*, *interoperability*, and *data collection, organization, and structuring* are predominantly assigned to DT technologies, which mediate between the physical layer and higher-level software while improving data consistency and quality [9,12,43]. DT-level pre-processing can itself be enhanced through embedded AI modules devoted to data cleaning, anomaly detection, and completion of missing information, further mitigating data-quality issues before data reach higher-level AI services [31,41]. Responsibilities such as *data processing*, *data analysis and pattern discovery*, and *execution and operative control support* can be implemented either by DT internal models or through dedicated AI applications that consume DT-exposed

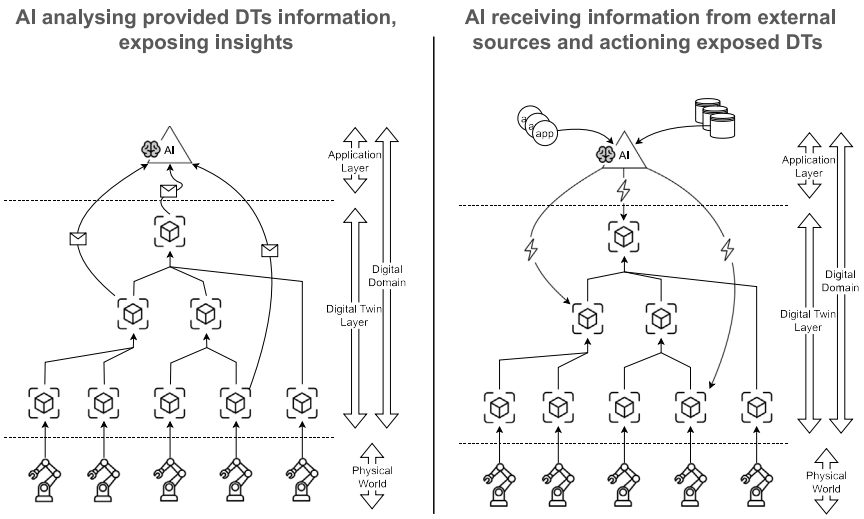


Fig. 3. High-level architectural configurations of AI–DT interactions in an industrial environment. Both scenarios involve the interaction of a DT layer with external AIs.

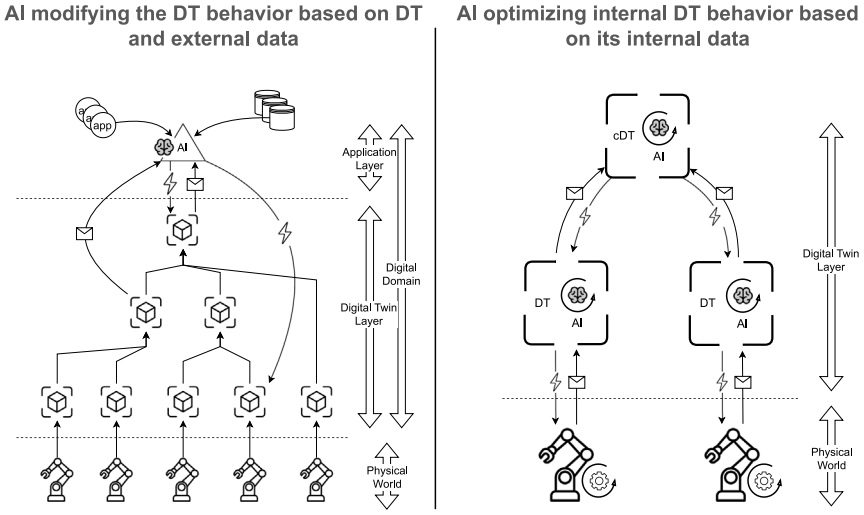


Fig. 4. Advanced AI–DT interaction configurations, emphasizing feedback loops and AI embedding within DTs, complementing the high-level view of Fig. 3.

information [5,29,34,44]. When moving towards *prediction of future behaviors and predictive maintenance* and *autonomous decisions*, AI becomes essential — for instance via Machine Learning models or Agent-based systems — while still relying on DTs for timely, contextualized data about the physical system [5,8,45,46].

This synergy directly addresses the three motivating challenges. DTs decouple AI solutions from the details of individual sensors and controllers, alleviating their physical *specialization* and supporting reuse across differently deployed but similarly modeled contexts. DT-driven abstraction and pre-processing, possibly aided by local AI modules, improve *data quality* before it is consumed by higher-level models [5, 31].

4. Proposed interaction patterns

The depicted division of responsibilities between AI and DTs in the industrial domain opens up a positive spiral of interactions capable of impacting the future industry. Such interaction patterns help technical stakeholders design, manage, and improve the digitalized factory, providing a useful project and communication tool.

Considering a DT as a *self-contained entity representing its physical counterpart*, four interaction patterns can be defined. This definition is based on the intersection between (i) the possible "digital locations" where an AI model can be placed with respect to the DT layer and (ii) the ways in which the AI model consequently interacts with the DT (and, possibly, with the associated physical context). To summarize these alternatives at a high level, Figs. 3 and 4 provide an overview of the possible interaction configurations in an industrial environment, which serves as a conceptual map for the patterns detailed in the following sections.

Fig. 3 offers a first-level classification of AI–DT interaction configurations, distinguishing cases where AI mainly consumes DT-managed information (left) from cases where AI actively drives or orchestrates DT-enabled applications based on information sources different from the DT layer (right). This figure abstracts from individual DT instances and AI models, focusing instead on the architectural positioning of AI with respect to the DT layer and the application layer.

Fig. 4 zooms in on more advanced configurations, highlighting when an external AI directly shapes DT behavior based on mixed data sources or when the AI is embedded within individual DTs. Compared to Fig. 3, it shifts the focus from mere positioning to the intensity and direction of the feedback loops between AI, the DT layer, and the physical world.

The match between the mutual positioning of AI and DTs and their consequent interaction model defines which information is exposed to

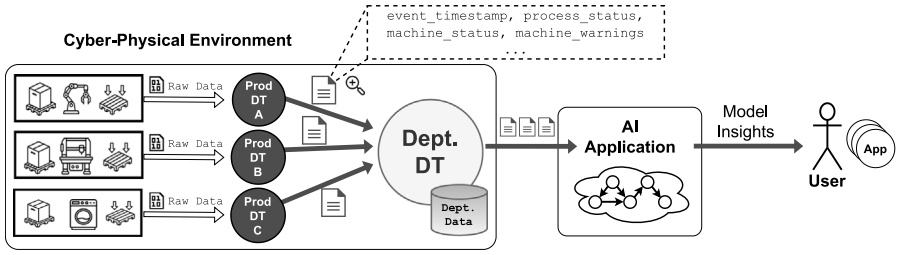


Fig. 5. Flow of data and insights in the *Observer pattern*, where the Departmental Digital Twin stream shop floor events to an AI application that generates model-based insights for end users.

the AI component, and how the AI model can affect the DT given such information. While the DT is strictly entangled with its physical twin, which lives in a confined context, the AI model can instead draw information from different sources, and the combination of these sources with DT interactions gives rise to the proposed patterns.

These four patterns, responding to RQ1, are summarized in Figs. 3 and 4 and are described in the following sections, considering a CPPS context twinned by a DT layer. For each pattern, its characteristics, areas of application, and related peculiarities are discussed.

4.1. AI Observing DTs: the Observer pattern

The first interaction pattern between DT and AI is the *AI Observing DTs pattern*. This pattern allows an AI model (or intelligent application) to *observe* the whole or a sub-part of the physical context by means of the DT layer and to compute advanced information to make predictions accordingly.

Consider the following problem: the need to build a prediction mechanism based on the state of multiple parts of the industrial shop floor. Since parts involved in the analysis belong to different contexts, the AI system must be able to gain data from all of these contexts without being relegated to a particular one. Placing the AI model inside one or more DTs is illogical, as each DT maps only a very narrow or strictly specified part of the physical world, limiting the observing capability of the AI model and raising useless communication and synchronization problems between each model replica living inside each DT. Consequently, to keep the system simple and the boundaries well defined, the AI model must live *outside* the DT layer, gaining data exposed by the DTs of interest and to let it properly *observe* all the sub-parts it is interested in. A scheme of the AI Observing DTs pattern is reported in Fig. 5.

In this scenario, the AI application in charge of observing DTs has its own business logic, knows the type of data it needs to operate, and exploits information from DTs to effectively carry out its analysis tasks. On the other hand, the AI application is not aware of the internal structure, models, or behavior of each DT, but this knowledge is not required to achieve its goals: its purpose is to collect the data exposed by each DT to maintain a broad view of the relevant areas. In this configuration, AI components consume information exposed by the DT layer to observe the physical system without directly closing the loop with actuation.

This setting is closely related to what is often referred to in the literature as a Digital Shadow [47] (DS), where a physical asset is mirrored by a purely observational digital representation and data flow is essentially unidirectional from the physical world to the digital one. However, there is a key difference with respect to a traditional DS. In the Observer pattern, the digital element is a full-fledged DT that already performs its own internal computation and abstraction over the physical system; the AI component then observes the information exposed by this DT layer in a unidirectional way, without providing feedback to the DT itself. In other words, the DT is capable of a bi-directional communication with the physical asset, while the external AI acts only as a consumer of DT-level insights.

Considering hierarchical and composable ecosystems of DTs [43], the level of hierarchy with which AI can interact is not limited beforehand and depends substantially on the type of modeling that each scenario requires. Assuming that in the modeling phase no limitations are given, the AI application observing DTs of interest can retrieve data from a single high-level DT, from multiple high, middle, or low-level DTs, or with a hybrid approach, as reported for example in Fig. 3, in the left box.

The AI Observing DTs pattern well fits in all the industrial contexts where it is necessary to draw and offer to external applications or stakeholders a future state or some complex analysis of the system or of its subpart. A good example is shown in [48]. In this work, the responsibility of the DT layer is to properly collect and prepare data from the physical system after some kind of disruption; after that, the data coming from the physical environment is analyzed by an external AI devoted to finding cause-and-effect relations among variables exposed by DTs. The result is then offered externally to the DT layer to domain experts, with the goal of supporting them and cutting down the root cause analysis time of the considered disruption.

Other possible applications involve KPI prediction of machines, departments, or even of the whole factory, information that can be very useful to be shared with domain experts. As an example, it is possible to consider the computation of the whole production capacity in the near future. This metric is crucial for different industrial stakeholders, spanning from the commercial pillar to the industrial strategic boards. Indeed, it describes whether the current assets are going to be saturated, and consequently, whether the system needs to be updated or expanded, guiding strategic decisions. For the commercial area, instead, knowing when and how the system will reach its maximum capacity is useful to properly drive commercial relationships with customers, taking data-driven negotiation decisions.

The same kind of reasoning can be done for other industrial KPIs, as those involved in safety, logistics, or energy consumption, but is not limited to it. This pattern can also be useful for advanced use cases, such as system layout classification, optimal positioning of the Customer Order Decoupling Point [49,50], or to suggest improvements to be applied to the system to obtain enhanced performance at a lower cost.

4.2. AI Advising DTs: the Advisor pattern

The second interaction pattern between DT and AI is the *AI Advising DTs pattern*. Here, an AI model or intelligent application *advises* the physical system or its parts via the DT layer using information gathered externally.

Since the DT layer is responsible for collecting and processing data from the physical system, accurately representing it in the digital domain, and potentially modifying the state of the physical system or its own internal state through feedback actions, it is expected to interact with applications operating within the digital domain. These upper-level applications may be interested in monitoring the physical context via the DT layer or in providing useful information for the CPS that originates from other applications external to the DT layer. This

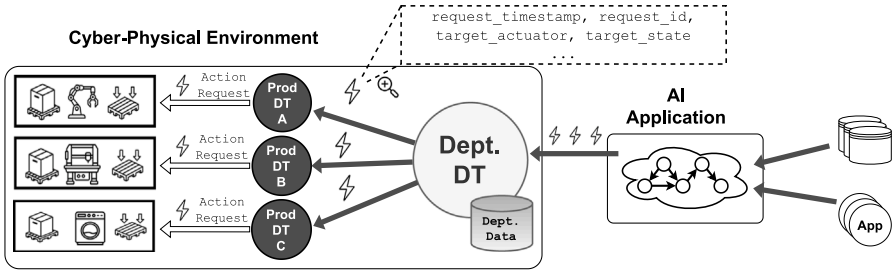


Fig. 6. Flow of action requests and recommendations in the *Advisor pattern*, where the Departmental Digital Twin mediate control commands from the shop floor and rely on an AI application to suggest suitable actuator states.

scenario is the reverse of the AI Observing DTs pattern (Section 4.1), where AI reasons on data from external sources and passes results to the DT layer via *actions*.

The organization of the AI Advising DTs pattern can consequently be addressed by considering the best positioning of the AI application. The AI module, in this case, must find a place where it has the ease to gather needed data from external sources, and then distribute the reasoning result to one or more DTs part of the DT layer. The similarity with the AI Observing DTs pattern helps in understanding that placing the AI component inside each DT of interest would be inefficient: it would replicate the same model across multiple DTs, multiply communication channels with external data sources, and complicate maintenance, since every model update would need to be propagated to several replicas. Moreover, the advising AI is not primarily interested in the internal state of individual DTs as a data source, so embedding it into DTs would unnecessarily overload their structure with additional data-management responsibilities and risk slowing down day-to-day operations. All the considered drawbacks highlight that a different structure must be taken into account.

The opposite approach, similar to the AI Observing DTs, consists of putting the AI application externally to the DT layer, letting it interact with DTs of interest when needed. This setup allows the AI component to manage internally all business processes regarding the data collection phase from external sources of interest, without slowing down or disrupting the operations of each DT. Then, the AI application acting as the DTs' advisor passes the reasoning results to the DT of interest through their capability of accepting action requests that will be then managed by internal models of each DT, according to its purpose. In Fig. 6, the AI Advising DTs pattern is reported. In this way, the responsibilities of each piece of software are clearly divided, DTs do not risk being disrupted, and each update involving the model in charge of reasoning based on given external data sources happens only once, if needed. As in the AI Observing DTs pattern, positioning the AI application advising DTs externally to DTs of interest is also justified by the fact that it is not focused on the internal state of each DT as a source of data: its target is only to pass the results of its computations to the correct DT, when they are done.

This pattern is particularly valuable in industry to share information or modify production based on external data. An intelligent advisor assessing outside data and advising the production system is a turning point, enabling applications once limited to niche sectors. For example, machines can modulate operation based on predicted energy costs—a task otherwise complex. Currently, only energy producers can predict power demand due to their strict balancing needs. The AI Advising DTs pattern extends this capability to any DT-enabled shop floor, allowing production adjustment based on power cost forecasts.

Another scenario where the AI Advising DTs pattern can show its potential involves advanced communication between industry entities part of the same supply chain. It is quite common for industrial partners to share key information about their production to improve the overall supply chain performance. Under this perspective, suppliers' data represent the external source of information that, after the processing operated by an AI advising DTs system, can lead to some exchange or adaptation of the associated DT-enabled production system. This can be the case for quality-related data: in this context, classifying or predicting the quality detail of a given set of raw material can be very useful to adapt internal transformation operations, obtaining a system capable of fine-tuning its activities and potentially consistently improving its quality output over time.

4.3. AI Controlling DTs: the Controller pattern

The third pattern of interaction between DT and AI is the *AI Controlling DTs pattern*. This pattern allows an AI model (or intelligent application) to gather data from different sources *both external to the DT layer as well as from the DT layer itself* and to pass the result of the consequent reasoning to the DTs of interest.

This pattern stems from the previous two and can be considered as their composition. Indeed, the AI Observing DTs has been defined as a model that computes starting from data exposed by the DT layer, while the AI Advising DTs pattern does the same but considering data sources external to the DT layer, possibly giving feedback to it accordingly to its computations. Problems considering this pattern arise from the right positioning of the involved AI application, as in the previous two cases. Indeed, positioning the AI in charge of controlling DTs inside each DT of interest would assign DTs responsibilities they were not designed for, replicate the same control model across multiple DTs, multiply communication channels with external data sources, and complicate model updates, since changes would have to be propagated to several replicas across the DT layer.

Placing instead the AI aimed at controlling DTs externally to the DT layer limits depicted problems and poses the intelligent application in a context that better suits its purposes. The consequent division of responsibilities is clear, resulting in a better maintainability of the AI module and in a minimization of the number of connections with external data sources needed to be carried out by the intelligent controller. Finally, involved DTs are not burdened by the AI module controlling DTs' operations, only experimenting with the positive contribution of the interaction of the considered external AI model. The scheme of the AI Controlling DTs is given in Fig. 7. The AI Controlling DTs pattern resulting architecture is therefore the fusion of the *AI Observing* and *AI Advising DTs patterns*, letting the controller gather data from both the DT architecture and from data sources external to the DT layer, to then pass computation results to the DT layer itself.

The AI Controlling DTs pattern can be involved in all the context where an intelligent system, external to the DT architecture, computes some decision about how the physical system should behave, based on the DT layer state and external data sources, and then feeds back this decision to the DT layer to obtain the desired effect on its physical counterparts. The industrial software systems devoted to such activities are industrial operations schedulers. Industrial schedulers are responsible for planning industrial production following its actual state and eventual additional information. Additional information can span information coming from *the system itself*, as predicted disruptions of

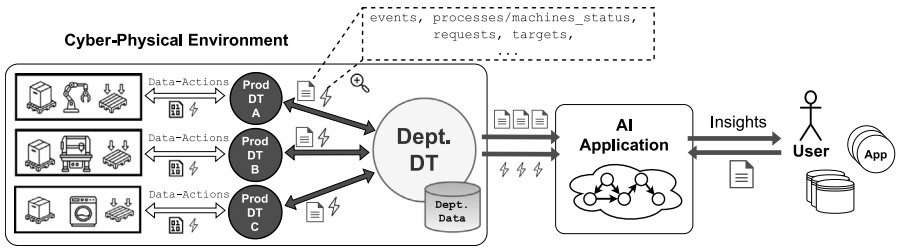


Fig. 7. Closed-loop coordination in the *Controller pattern*, where departmental Digital Twins and the AI application jointly manage data and actions to drive autonomous control over shop-floor processes and machines.

the shop floor, or information coming *externally from the system*, as predicted market demand for the product to process. The selected information can then be processed by the AI application, which in this case works as a smart scheduler coordinating the DTs: its output corresponds to the planned activities distributed among the physical system by means of the DT layer.

The application of this pattern is not necessarily limited to the scheduling of production activities: in fact, it can also be used to schedule maintenance activities [44,45] (also according to production activities) or to explore different operation planning and control systems, such as those involving *agents* [51]. In this regard, a detailed analysis of similarities and differences between DTs and Agents or Multi-Agents Systems is proposed in [52], suggesting specific interaction patterns, that can also be explored in the industrial context.

The research area of production scheduling and operations control presents several works proposing advanced optimization approaches [34,53] or leveraging the use of AI models [54], also with the goal of reaching real-time optimizations [46]. Nevertheless, what is really missing is the *closed-loop* between the information gathered from the physical system (i.e., the *physical system visibility*, enabled by DTs) and the *action feedback* towards the physical system itself. The interaction pattern between the DT layer and the AI Controlling DTs can fill this gap, allowing several improvements in industrial activities.

4.4. AI Embedded in DT: the Embedded pattern

The last interaction pattern between DT and AI discussed is the *AI Embedded in DT* pattern. Unlike others, its goal is to *add intelligence* directly into a DT, and consequently its physical counterpart, enabling advanced behaviors.

The definition of this pattern can be easily derived considering the following problem: the need to forecast future states of a physical entity starting from the digital representation provided by its DT. Considering the DT as the digital representation of a physical asset is straightforward, as representing the physical asset is the specific responsibility of its associated DT. Consequently, the DT has a complete vision over its physical counterpart, or, in other words, the DT operates, by design, in the context of its physical counterpart. In such a setup, the possible positioning of the AI model in charge of computing the future states of the given DT can be both placed inside and outside of the DT, as in the patterns previously discussed. If placed outside of the DT, the AI model can access a limited vision of the DT and its associated context, as it can only view the information *exposed* from the DT. All the other sets of information received and processed by the DT are, instead, hidden from the AI model, which — although it could potentially generate certain predictions — will have very limited forecasting capabilities. Moreover, as the considered scenario involves an AI module that must provide some prediction useful for the DT of interest, it is expected that the two entities will spend a non-negligible level of resources to exchange data and feedback, resulting in a communication overhead.

An approach opposite to the one depicted so far places the AI component *inside* of the DT of interest, justifying the name of *AI Embedded in DT pattern*. This approach differs from the one previously described as the AI model can better fulfill its responsibilities, because it can now access a richer context, that is, the whole stream of data received from the physical counterpart, and all the possible results of the data manipulation operations carried out by the DT. A scheme of the AI Embedded in DT is reported in Fig. 8. Under the communication operations lens, having the AI model embedded inside the DT eliminates the need for the two entities to communicate, as the AI model is itself part of the DT's internal structure. The approach well fits the expected structure of a DT, as it is internally characterized by one or more sets of *models* that define the behavior of the DT. Each model of the DT can have different characteristics, *augmenting* [9,55] the computation of its state, which is then exposed to entities outside of itself. In a holistic approach, no limitations are given to *how* a model must be structured, leaving a free place also for AI approaches. Moreover, the AI Embedded in DT pattern also guarantees prompt feedback of the AI computational result towards the physical domain, if needed. Indeed, living inside the DT makes it possible to access *actions* the DT is capable of towards the physical counterpart.

Going beyond the strict definition and description of the AI Embedded in DT pattern, it is possible to consider it as a key enabler for the adoption of IAI in future industries. As discussed in Section 3.2, a positive interaction arises between AI and DTs, with DTs that can perform the task of capturing data from their physical context, and AI models that can contribute to cleaning the resulting data, supporting DTs in organizing a high-quality high-volume set of data relative to the associated physical counterpart. This scenario can be achieved through the use of the AI Embedded in DT pattern, through the usage of a model specialized in improving the quality of the data received from the physical domain. In this way, the gathered data are cleaned from the noise typical of the physical domain, enabling, in conjunction with the high volume of information that the physical world generates and that the DT is capable of intercepting, an effective application of IAI at all levels. The system resulting from the application of the AI Embedded in DT pattern is beneficial for the AI Observing, Advising, and Controlling DTs patterns, for each embedded AI living inside of the given DT, and consequently, for the whole DT layer as well as the upper-level applications living in the digital domain, underlying how the AI Embedded in DT pattern can be a turning point for a higher adoption of IAI models in the industry of the future.

In the industrial context, the AI Embedded in DT pattern is useful to augment DT's capabilities and introduce intelligence for adopted physical equipment. This pattern fits well for scenarios where it is necessary to infer the actual health of a machine through its DT, or to predict when it is going to disrupt, and to manage the related maintenance activity accordingly. The scientific literature is full of predictive maintenance approaches, ranging from actual prediction methods [56], prescriptive tools [8], and integrated frameworks [45]. Nevertheless, the AI Embedded in DT approach ensures a proper scalability of the system, clearly dividing responsibilities between the part of the system devoted to computing the future state of a physical asset (i.e., the DT) and the component responsible to reschedule the overall production activities accordingly (i.e. the external scheduler eventually following the Controller pattern).

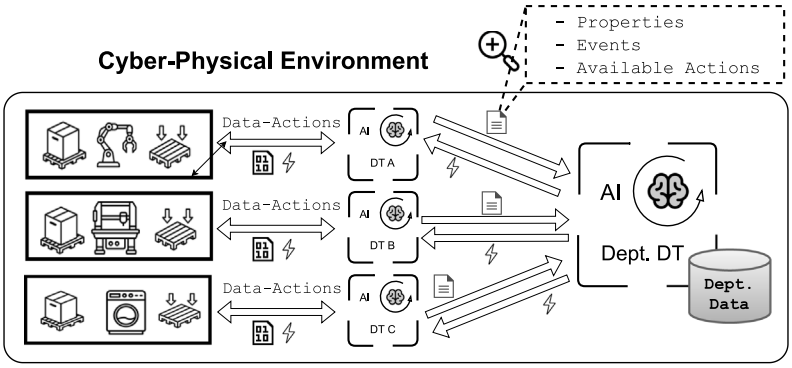


Fig. 8. Decentralized decision making in the *Embedded pattern*, where AI capabilities are co-located with production Digital Twins to manage local data–action loops while coordinating with the Departmental Digital Twin.

Another use case that could be done with the AI Embedded in DT pattern is the one depicted in [6]. In this work, the authors build a DT layer responsible for representing the physical world and, on top of that, the Proactive Material Handling (PMH) system. The PMH system employs an AI model, and its goal is to predict when an actual production task carried out by a physical asset (represented by its DT) will end. In a production system, this information can be used to anticipate the logistic resource allocation and to promptly move the transformed material from the actual production node. The effect is the elimination of the production node stop due to the output buffer block and a consequent smart coordination system between the production and the logistics department. While [6] aligns with the AI Controlling DTs pattern, embedding the AI in the DT for tracking start/end times is more convenient. Computing predictions directly within the production node DT avoids exposing, gathering, and organizing data externally. This approach also allows easier customization of prediction models within each DT due to production system heterogeneity. Once computed, the embedded AI can easily trigger coordination with the DT of the logistics department, due to its integrated context. Such DT then processes the request, schedules picking, and assigns tasks to the appropriate entity.

5. Experimental illustration

To illustrate the emerging interaction patterns between AI and DTs in the industrial domain, a use case is presented. It concerns a CPPS built around a physical micro-factory, i.e., a miniaturized production system by Fischertechnik,[1] and its digital counterpart within the DT layer of the digital domain. Despite its small scale, this setup captures key industrial challenges thanks to its architectural complexity, making it a benchmark for research [57–60].

The configuration includes nine machines operated by three heterogeneous controllers, grouped into three functional departments. The micro-factory architecture, shown in Fig. 9, highlights these departments: the Indexed Department on the left, the Multi-Process Department on the right, and the Robot Vacuum Gripper at the center. Although the physical layout is fixed, limiting certain operational variations, the hardware diversity and restricted data access at the controller level ensure that the scenario reproduces the essential complexities of real-world industrial environments.

The DT architecture of the micro-factory, shown in Figs. 10 and 11, follows the principles described in [10]. The first layer hosts machine DTs representing individual physical components; these aggregate into department DTs, which in turn compose the overall factory DT. Each DT instance runs on a dedicated server and receives data from PLCs.

[1] Fischertechnik research equipment (url: https://www.fischertechnik.de/en/products/industry-and-universities), schematized in Figs. 9, 10, and 11.

To objectively evaluate the proposed interaction patterns, a design-level assessment focusing on the impact of each pattern on system evolution and maintenance in the Micro-Factory case study is introduced. The dimension evaluated are: (D-1) modularity and separation of concerns, evaluating how clearly separated are the elements constituting the pattern, that are involved DTs, AI models and physical assets; (D-2) reusability of AI component, highlighting the degree of reusability of the AI component when a DT is added to the CPS; (D-3) AI component maintainability and replaceability, reporting the intervention effort needed at a CPS level to replace or update the AI model; (D-4) legacy integration and digitalization, describing the effort needed using the proposed patterns to digitalize a legacy physical system.

In this perspective, the experimental illustration provides concrete evidence that the proposed interaction patterns can be instantiated on a realistic CPS, supporting their practical applicability beyond the conceptual level.

5.1. AI Observing DTs for root-cause analysis of degraded performance

In industrial systems, performance tracking forms the foundation for evaluating operations. It typically relies on established KPIs, with *Overall Equipment Effectiveness* (OEE) [61] being the most common. OEE focuses on the performance of individual production nodes; aggregating nodes within a department yields the *Weighted OE*E for that area.

Tracking OEE metrics enables identification of under performing areas and supports improvement projects or *root-cause analysis* in case of system failures. The is almost always carried out by high-skilled domain experts, with investigations that take weeks to months. An intelligent CPPS capable of autonomously calculating performance metrics and analyzing data can significantly reduce this manual effort [48]. OEE is defined as Availability × Performance × Quality (A × P × Q) [61]. In the Fischertechnik implementation, A and Q are assumed to be 100%, as no shift schedule or quality checks are applied; hence, OEE depends solely on machine performance:

$$P = \frac{N \times CT}{RT}$$

where N is the total number of processed products [pc], CT the ideal cycle time [s/pc], and RT the actual run time [s].

Each machine DT computes the cycle time using available sensors: those with presence sensors detect product arrival directly, while others (e.g., in the Multi-Process Department) use a virtual presence token generated by their DT. State transitions between idle and working trigger timestamps and performance updates. Performance degradation can spread through such tightly coupled systems, complicating root-cause identification—especially in distributed environments. To address this, a causality learning system must identify the machine responsible for slowdowns by selecting the proper AI–DT interaction pattern for OEE monitoring.

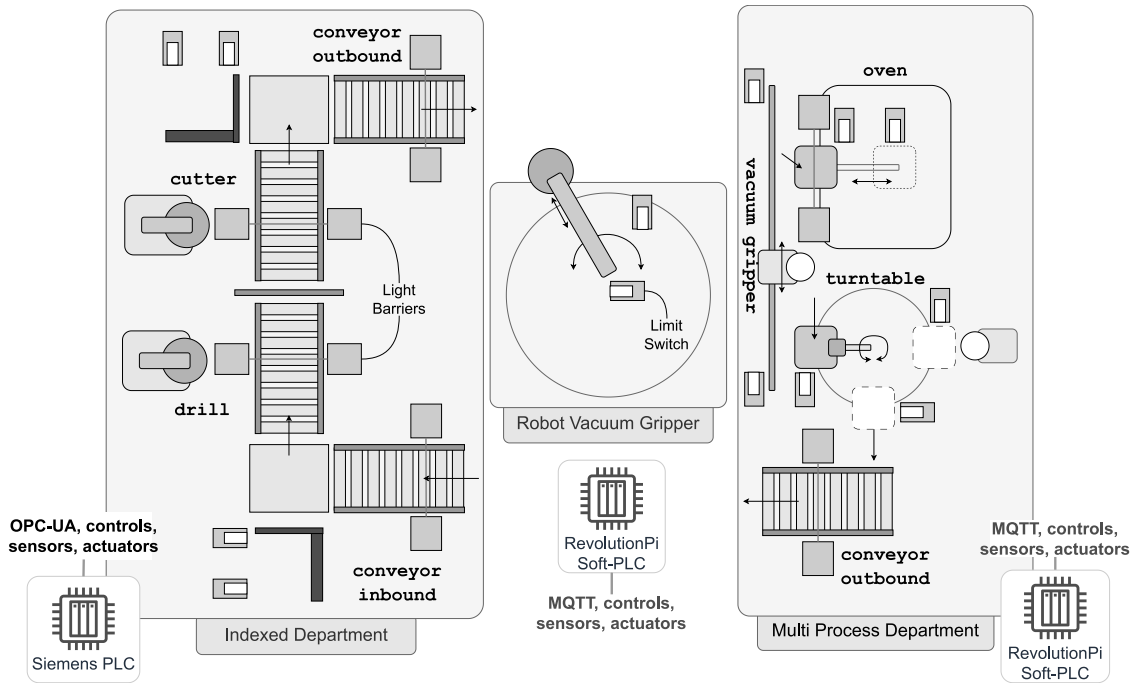


Fig. 9. Representation of the micro-factory cyber–physical production system, detailing main components of its layout, the PLC-based control, and high-level sensing, and actuation infrastructure.

Following the necessity to implement a causality learning system capable of unveiling the machine causing the slowdown of the whole system, selecting an appropriate interaction pattern between the target AI model and the available pool of DTs monitoring physical OEE metrics is a crucial task. The main requirements guiding this choice are threefold. (i) First, the AI model must deliver its results to domain experts working at the upper-level application layer, where analyses are visualized within a dashboard. In this setup, the information flow is unidirectional — from the physical system to the digital layer — without feedback towards actuators or control logics. (ii) Second, the causality model requires access to a broader operational context, since an observed degradation may stem from a single malfunction or a combination of small inefficiencies distributed across different areas. (iii) Third, the AI model does not require full access to every DT variable; its role is to provide an initial, high-level analysis supporting human-led root cause identification, thus reducing the effort of manual investigation. Given these conditions, the most appropriate architectural configuration corresponds to the *AI Observing DTs* pattern.

In the micro-factory setup, this pattern materializes as the AI module collecting data exposed by machine and department DTs, processing them to infer likely sources of performance degradation, and exposing the results to higher-level applications for visualization and decision support. When two or more machines' performance decreases at a critical level, the external intelligent application triggers its operations, starting to collect information from related DTs. Notably, since DTs' responsibility is to receive, organize, and homogenize data, the external application does not have the responsibility of cleaning and reordering data: the dirty work has already been done by DTs internally, adapting the operations with respect to the physical counterpart. Then, the intelligent application discovers the causal relationships between machines' performance level, exposing to system experts the one that is responsible for the overall system slowdown. Table 1 summarizes the design-level assessment of the AI Observing DTs pattern along the four dimensions (D-1)–(D-4). The resulting system is clearly modular, since the DTs that capture machines' OEE and related data are cleanly separated from the external AI model that processes this data and exposes the results to users. The AI component is largely reusable: adding a new machine and its DT does not disrupt the Observer AI nor require major adaptations. The Observer AI can also be updated or replaced without any impact on the DT layer. When applied to legacy systems, DTs encapsulate PLCs and physical assets, simplifying interaction with the Observer AI at the cost of an initial DT wrapping of existing equipment.

Table 1
Design-level assessment dimensions for the AI Controlling DTs pattern in the micro-factory experimental illustration.

Dimension	Description
(D-1) Modularity & separation	**Highly clear** – AI and DT are clearly separated; the AI only reads the exposed DT state, while the DT encapsulates heterogeneous devices and low-level data.
(D-2) Reusability of AI component	**Highly reusable** – Analytics can be reused when adding new DTs with compatible schemas.
(D-3) AI maintainability/ replaceability	**Highly maintainable** – Models can be updated or swapped without modifying the DT structure or controllers.
(D-4) Legacy integration & digitalization	**Medium/High integrable** – DT hides legacy PLC/MES, facilitating interaction with external AI models, but requires prior DT wrapping of assets.

5.2. AI Embedded in DT for machine health monitoring

Predictive maintenance models have become state-of-the-art tools in today's shop floors, and their widespread use witnesses the benefit of actively monitoring key aspects of production systems. The proposed AI-DT patterns of interaction can reshape the way these models are implemented into the system, promoting advanced integration and avoiding information silos and resulting limitations.

In the predictive maintenance scenario, the objective is to implement AI models capable of estimating or classifying the health status of individual production nodes based on key operational parameters.

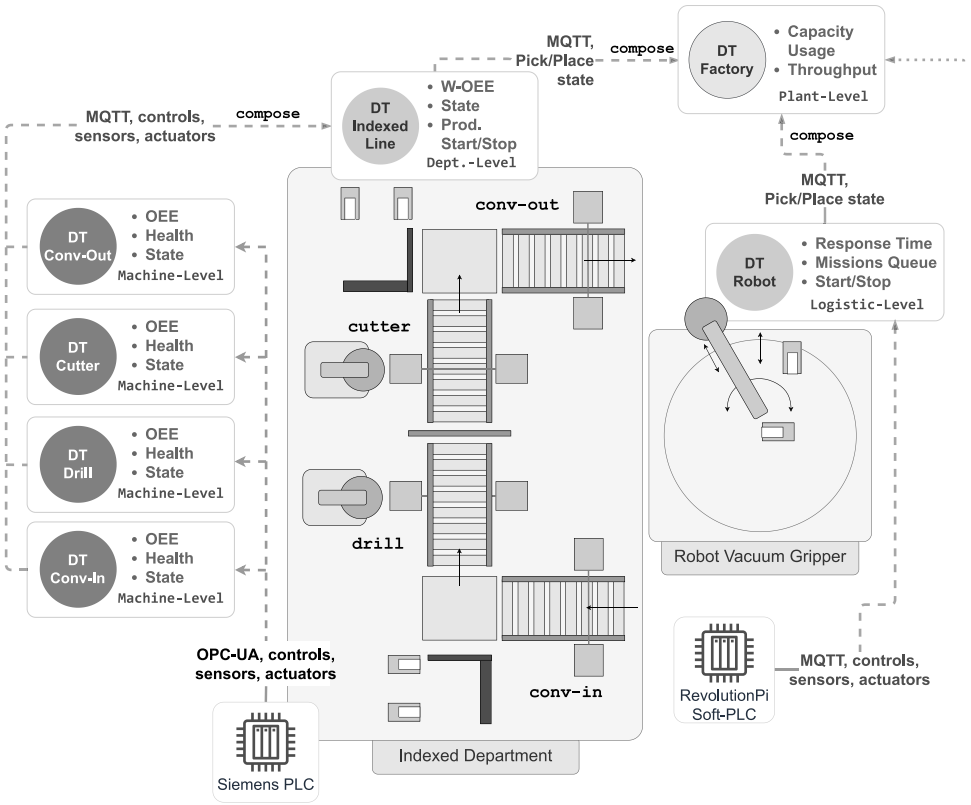


Fig. 10. Representation of the left-side physical layout of the micro-factory and its hierarchy of Digital Twins, linking machine-, department-, and plant-level DTs to PLC-based sensing, control, and robot handling.

As each industrial setting employs heterogeneous equipment, it is often necessary to develop specialized AI models tailored to the specific data and operating conditions of each machine. Given this requirement, each AI model must operate within the local context of its corresponding production node, relying on the data that node generates. The model analyzes the incoming signals, updates the digital representation of the node, and augments its DT state with diagnostic or prognostic information. When anomalies or degradations are detected, the AI may trigger coordination actions towards the physical layer using DTs as proxies of target physical entities—for example, pausing the operation of machines interacting with the faulty unit or initiating maintenance procedures within the affected department.

These characteristics clearly indicate that the appropriate architectural configuration is the AI Embedded in DT pattern. In our micro-factory, each predictive-maintenance model runs inside the DT of its corresponding machine, continuously updating the DT state with health indicators and, when necessary, triggering local actions such as pausing operations or initiating maintenance procedures.

Taking as an example the output conveyor of the last machine in the Indexed Line Department, it has been equipped with an additional wireless vibrational sensor responsible for collecting vibration data from the attached machine. The associated DT has then been used to support the whole working pipeline, from data collection to model analysis. Indeed: (i) the associated DT has been used as a collector to ingest data, organize and save it as a history; (ii) then, the data has been extracted to perform the AI model training; (iii) finally, the resulting model has been integrated directly into the DT for active operational usage.

Since the collected information is referred to as vibrational data, the target of the training was to set a threshold that, when consistently exceeded, triggered a change in the state of the DT representing the machine. Notably, the DT also comes in handy in this type of activity, offering internally to the model the capability of turning its state to "unhealthy", consequently stopping the associated physical counterpart with an action request. Nevertheless, the system can be further expanded thanks to the presence of the DT, enabling the possibility to notify other applications or maintenance operators, acting as a cyber–physical coordination system. The resulting architectural assessment for the AI Embedded in DT pattern are reported in Table 2, using the same four design-level dimensions (D-1)–(D-4) adopted for the previous pattern.

Table 2
Design-level assessment dimensions for the AI Embedded in DT pattern in the micro-factory experimental illustration.

Dimension	Description
(D-1) Modularity & separation	**Highly clear** – Intelligence lives inside the DT, but is clearly divided from the latter; the DT can, indeed, live also without the DT component.
(D-2) Reusability of AI component	**Medium/Highly reusable** – Embedded logic is asset-class-specific, reusable mainly via DT templates and cloning.
(D-3) AI maintainability/ replaceability	**Medium/Highly maintainable** – Model updates may require redeploying or reconfiguring the DT, but changes remain localized to that asset.
(D-4) Legacy integration & digitalization	**Medium integrable** – DT still encapsulates legacy interfaces, now enriched with embedded AI, but customization is per-asset-class.

Under the evaluation perspective, the components enabling this feature are clearly identified and separated: the DT mirrors, collects, and processes data from the physical sources (here, the conveyor PLC and the additional sensor), while the embedded AI module extracts knowledge from this data. The AI module shows a medium level of

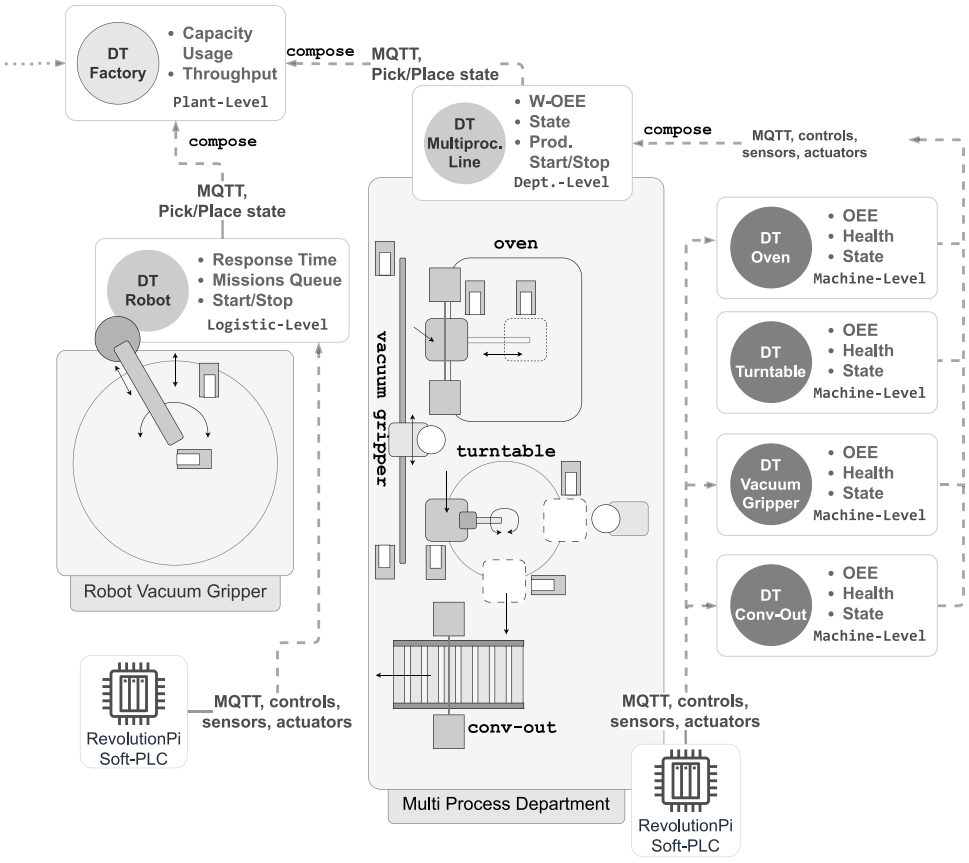


Fig. 11. Representation of the right-side physical layout of the micro-factory and its associated Digital Twins, highlighting multi-process equipment, robot handling, and machine-, department-, and plant-level DT hierarchy.

reusability, mainly constrained by its asset-class-specific nature. Updating the AI model requires, indeed, redeploying the associated DT, but the resulting changes remain localized to that asset. In legacy systems, inserting intelligence is strongly supported by DTs encapsulating existing interfaces, although customization is per asset class and thus implies a medium effort in complex legacy environments.

5.3. AI Controlling DTs for scheduling based on the health of machines

Performance degradation and maintenance activities generally require scheduling related activities. Scheduling is a very important pillar in day-to-day industrial operations, and its pivotal role is supported by a very rich scientific literature. However, the effectiveness that a scheduler can have on an industrial system is often limited by the lack of a *closed-loop* between scheduled activities and their real-time monitoring at the shop-floor level, or in other words, between the digital planning and its physical execution. AI-DT patterns have a positive impact in this regard, supporting on one hand the missing link closing the feedback loop and, on the other hand, providing a clear architectural approach to implement an effective smart operation scheduler.

In the production scheduling scenario, the objective is to coordinate manufacturing operations efficiently across all production units while accounting for both internal and external constraints. To do so, an AI-based scheduler must operate from a global system perspective, continuously observing changes across the shop floor. The information it processes ranges from production queues, work-in-progress status, and completed tasks to planned maintenance activities and unexpected disruptions. Importantly, such information does not depend on the detailed internal state of each machine but on their aggregated operational status.

The decision space of the scheduler also extends beyond the physical system, incorporating external contextual data such as incoming customer orders, expected product demand, and material or resource availability. Combining these heterogeneous inputs, the AI computes optimized schedules that are then fed back to the digital and physical layers, assigning or rescheduling tasks accordingly. After each feedback cycle, data are collected again and used to refine subsequent scheduling decisions.

Given these requirements, we adopt the AI Controlling DTs pattern. In the micro-factory, this takes the form of a centralized AI scheduler that consumes DT-level information and upper-level planning data to compute production plans, which are then propagated back to the DT layer for execution on the physical shop floor.

This pattern clearly defines the control loop between DTs and physical assets while maintaining modular separation between local intelligence (e.g., machine-level predictive maintenance) and global intelligence (e.g., scheduling). In the micro-factory implementation illustrated in Fig. 12, each DT embeds its own local AI model for health monitoring (as detailed in Section 5.2), whereas the external AI-based scheduler operates atop the DT layer, collecting state data and external demand information to plan and dynamically adapt production activities. Table 3 summarizes the design-level assessment of the AI Controlling DTs pattern in the micro-factory case.

The resulting *intelligent micro-factory*, also depicted in Fig. 13, has multiple smart capabilities. Starting from the top part, it can integrate any kind of smart scheduler [46,53,62] capable of organizing activities optimizing key parameters, considering also failure rates, average repairing times, and related information exposed by the underlying DTs [53,63]. Second, it can easily interact with any production component of interest, thanks to the abstraction brought by DTs and associated broken silos. Third, underlying machine DTs can interact

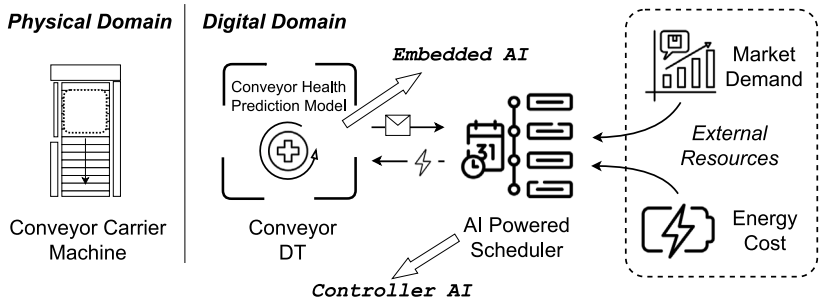


Fig. 12. Use case combining *Embedded* and *Controller patterns*, where a conveyor Digital Twin embeds a health-prediction model and collaborates with an AI scheduler that optimizes operations based on external demand, resources, and energy costs.

Table 3
Design-level assessment dimensions for the AI Controlling DTs pattern in the micro-factory experimental illustration.

Dimension	Description
(D-1) Modularity & separation	**Highly clear** – AI and DT clearly separated, with the AI powered by DT and external data sources and elaborating the feedback towards DT, and the DT mediates the feedback towards actuators.
(D-2) Reusability of AI component	**Medium/Highly reusable** – Controller logic reusable when a new DT is added, assuming that the new DT can handle internally and properly route action requests, as the one related to activity schedules - coming from the controller.
(D-3) AI maintainability/ replaceability	**Medium/Highly maintainable** – Controller models can be updated and swapped mostly without modifying related DTs; nevertheless, there may arise scenarios where the DT internal model handling actions requests must be aligned to the new Controller.
(D-4) Legacy integration & digitalization	**Medium/High integrable** – DT hides legacy PLC/MES, facilitating interaction with external AI models, but requires prior DT wrapping of assets.

with the scheduler, notifying it of any existing or upcoming disruption, allowing the scheduler to react accordingly, organizing a new production plan, and coordinating with maintenance operators. The positive integration and interaction between the two software components is, on one side, promoted thanks to the abstracted DT layer, and on the other, the chosen interaction patterns promote a clear definition of responsibilities.

The modularity of the Controller pattern emerges from the clear separation between the AI scheduler and the DT layer coupled with physical components. The AI module offers a medium-to-high level of reusability, since CPS updates rarely require changes to the scheduler itself; however, DTs must be able to handle action requests from the scheduler, which entails a limited adaptation effort. AI maintainability is likewise medium-to-high, as the Controller can be updated or replaced with minimal impact on the DT layer, except in cases where DT internal models must be aligned with new control policies. From a legacy perspective, DTs act as a unifying abstraction over heterogeneous PLCs and controllers, so integrating the Controller pattern mainly requires modeling the relevant DTs and their control endpoints rather than replacing existing shop-floor systems.

5.4. Experimental summary

The experimental illustration on the micro-factory shows that the Observer, Controller and Embedded patterns can be instantiated in realistic CPS scenarios while preserving the intended distribution of responsibilities along the cyber–physical continuum. The design-level assessment, responding to RQ2, highlights that all patterns achieve a high degree of modularity and separation between AI and DTs, with differences emerging mainly on AI reusability, maintainability and the effort required for legacy digitalization.

Table 4
Summary of the design-level assessment of the four AI–DT interaction patterns in the micro-factory case along the dimensions (D-1) modularity and separation, (D-2) reusability of the AI component, (D-3) AI maintainability/replaceability, and (D-4) legacy integration and digitalization. H stays for High, M/H for Medium-to-High; M for Medium.

Pattern	(D-1)	(D-2)	(D-3)	(D-4)
AI Observing DTs	H	H	H	M/H
AI Advising DTs	H	M/H	M/H	M/H
AI Controlling DTs	H	M/H	M/H	M/H
AI Embedded in DT	H	M/H	M/H	M

The remaining pattern, i.e., AI Advising DTs, occupies an intermediate position between Observing and Controlling: the separation between the AI model and the DT is still high, with a reusability level that ranges from medium to high. Indeed, the logic of the Advisor is reusable when a new DT is added, assuming that the new DT can handle internally and properly route actions request coming from the advisor. Similarly, Advisory models can be updated and swapped mostly without modifying the related DTs; nevertheless, scenarios may arise where the DT internal model handling action requests must be aligned to the new Advisor. When the AI Advising DTs pattern has to be applied to legacy systems, the integrability level spans from medium to high. As in the Observer and Controller pattern, indeed, the Advisors take advantage from the DT hiding the technological details of the legacy physical system.

6. Discussion

The four AI–DT interaction patterns proposed in this work — AI Observing DTs, AI Advising DTs, AI Controlling DTs, and AI Embedded in DTs — constitute the main conceptual contribution of the paper, as they provide a systematic vocabulary to describe how AI components and DTs distribute sensing, decision-making, and actuation responsibilities in industrial CPPSs. These patterns synthesize recurring configurations that are only implicitly discussed in current literature on AI-enabled DTs and intelligent manufacturing [5,25,38], and they make explicit the architectural decisions that often remain hidden behind case-specific implementations. With respect to RQ1, the study explicitly specifies (i) the locus of AI (inside or outside DTs), (ii) the directionality of influence (observation only vs. bidirectional control), and (iii) the scope of context (local vs. system-wide).

From the perspective of existing frameworks, the patterns refine and operationalize the roles typically assigned to AI and DTs in layered architectures and DT life-cycles. Prior work commonly focuses either on internal DT design and synchronization patterns, or on catalogs

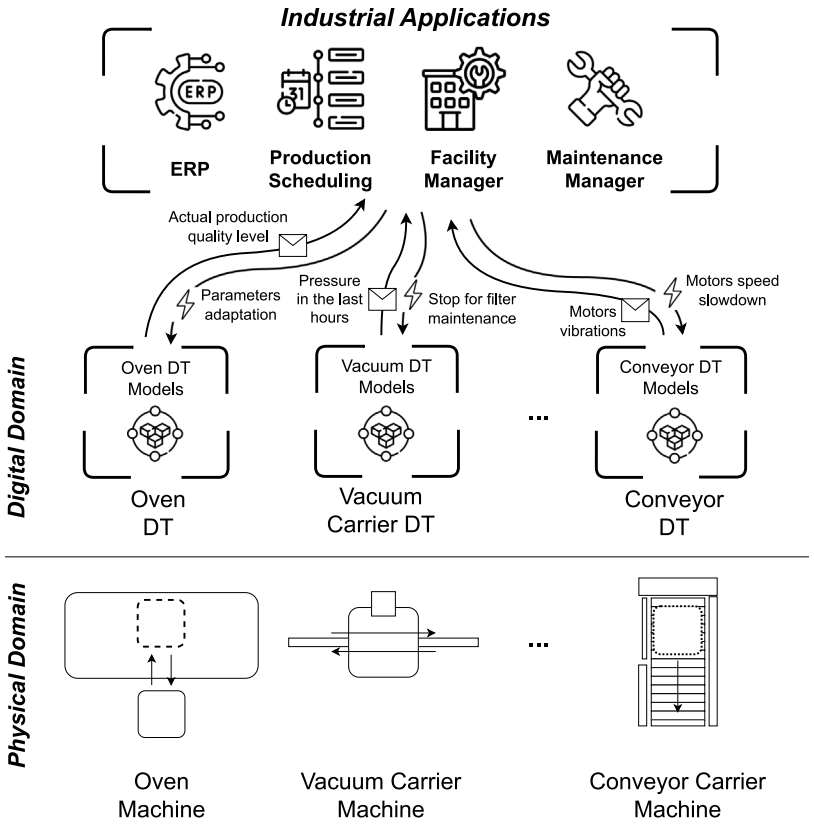


Fig. 13. Overall CPPS architecture of the micro-factory, showing how Digital Twin models for key machines interact with industrial applications to exchange production insights, maintenance alerts and other useful information across digital and physical domains.

of DT architectures [17–20,26], without specifying how intelligence should be positioned with respect to twins or how data and control should flow between them and AI components in a reusable way. In parallel, recent surveys on AI-in-DTs [5,22,25] and agent–DT micro-architectures [29,52] recognize AI embedded in the twin or co-located with agents, but they stop short of providing a unified, scenario-oriented taxonomy of AI–DT interaction modes that can guide concrete industrial deployments. By contrast, the four patterns proposed here directly map these usage scenarios to architectural responsibilities, making them easier to compare, reproduce, and extend.

This mapping is reflected in the responsibility distribution depicted in Fig. 2. In AI Observing DTs, sensing and state curation remain within DTs, while AI performs non-intrusive analysis and exposes insights to upper-level applications, leaving actuation outside the loop in a shadow-like setting. In AI Advising DTs, the data flow is overturned, with AI analyzing upper-level applications data and forwarding actions requests accordingly to DTs. In AI Controlling DTs, analysis and decision-making are centralized in an AI controller that consumes both aggregated DT information and upper-level applications data flows and issues commands back to the DT architecture, matching controller-style agent frameworks that coordinate multiple twins [52]. Finally, in AI Embedded in DTs, analysis and local decision support are pulled inside the DT boundary, aligning with micro-architectures in which DTs encapsulate both state and embedded intelligence for machine-level autonomy and decision making. In particular, the last three patterns introduce different forms of feedback from AI to the DT layer and, through it, to the physical system, which aligns more closely with what the literature often denotes as true DTs [47], where the digital representation both reflects and influences its physical counterpart.

Beyond architectural separation of concerns, the patterns also impact on how data quality is managed along the AI–DT pipeline. In particular, positioning DTs as an intermediate layer between the physical shop floor and AI services naturally supports the enforcement of data-quality policies — such as outlier detection, imputation, and consistency checks — before data are consumed by learning or inference components [40,41]. This aligns with recent views that emphasize data-centric AI and the symbiotic relationship between ML and data cleaning, where ML can both benefit from and contribute to automated error detection and repair routines [31,41].

Conceptually, the framework can act as a lens to classify and compare AI–DT solutions in future research. Many recent works propose AI-driven DT applications for anomaly detection, predictive maintenance, scheduling, embodied control, or what-if simulation, but their architectural assumptions are often buried in implementation details. By tagging each solution with one of the four patterns (or a combination thereof), researchers can systematically analyze trade-offs in terms of latency, explainability, fault isolation, and ease of deployment. For industrial practitioners, the patterns provide a structured way to select architectures when designing new CPPS projects: for example, choosing AI Embedded in DTs when local autonomy and tight coupling with machine-level sensing are key, or adopting AI Controlling DTs when global coordination and cross-asset optimization dominate. The same lens can support other tasks, such as educating stakeholders about AI–DT roles, designing governance and safety mechanisms for autonomous operations, or defining migration road maps from manual to AI-assisted decision processes.

A further implication concerns integration with heterogeneous legacy systems — MES, ERP, PCS, SCADA, and proprietary controllers — which are ubiquitous in industrial environments. The DT layer presented in Fig. 1 already hints at DTs acting as an intermediate facade between legacy infrastructures and AI-enabled applications: DTs absorb protocol diversity, reconcile data models, and expose a uniform interface that decouples AI design from low-level integration issues.

From the stand point of any of proposed interaction pattern, the legacy system is nothing more than a DT to interact with. Then, the DT takes care of all the needed technology details to promptly translate any of the interactions had with an AI module (internal or external) with a proper information to share with its associated physical twin.

Regarding the scalability of the patterns, there is, in principle, no inherent limit to the number of AI modules that can interact with DTs, nor to the number of DTs an AI module can handle. A set of DTs may simultaneously interact with multiple external AI components following the Observing, Advising, and Controlling patterns, while also hosting embedded AI within their internal logic. This flexibility stems from treating AI and DT technologies as separate modules that communicate through well-defined interfaces. Practical scalability limits, however, depend on how DTs and AI components are engineered. In scenarios based on monolithic DTs (like the approach taken by Eclipse Ditto[2] for centralized open-source DTs), scalability is strongly constrained by the capabilities of the hosting hardware. In contrast, microservice-based DT architectures [11] can distribute functionality across the network, alleviating these hardware bottlenecks. Similar considerations apply to AI components, emphasizing that the proposed patterns remain compatible with different internal architectural choices and support modular deployment strategies. A stricter scalability boundary may arise for the AI Embedded in DT pattern. Even when DTs follow microservice principles, a DT instance running on a single host may hit physical limits if the embedded AI model requires substantial computational resources. In principle, this issue can be mitigated by further distributing the internal elements of a DT across the network, at the price of increased latency and additional technological trade-offs. From an operational perspective, further limitations may appear when multiple AI modules with feedback capabilities (AI Advising and Controlling DTs) act on the same DT context. Conflicting feedback may lead to inconsistent behavior, with potentially disruptive effects on the overall CPPS. To address this, several strategies are possible, ranging from the introduction of a single source of truth for shared AI decisions to local arbitration or feedback evaluation mechanisms integrated within each DT. A more in-depth investigation of coordination and conflict-resolution strategies in multi-AI scenarios is left for future work.

Additional aspects remain only partially validated in this work. The patterns have been demonstrated in a Micro-Factory context with a limited number of machines, and quantitative evaluations of performance — latency, throughput, robustness to communication failures, resilience to model drift, or safety guarantees under faulty predictions — are still open issues. Moreover, the interaction between high-capacity generative models and real-time industrial constraints is emerging and requires further investigation in terms of verification, monitoring, and human-in-the-loop mechanisms.

Overall, the patterns should be seen as a starting point rather than a closed taxonomy. Future work may refine them, identify sub-patterns tailored to specific sectors (e.g., process industries vs. discrete manufacturing), and couple them with quantitative design guidelines—for example, mapping workload characteristics and non-functional requirements (safety, latency, explainability) to pattern selection. In parallel, systematic benchmarking across larger, heterogeneous industrial testbeds will be necessary to assess how the patterns behave under realistic constraints, and to what extent they support long-term evolution of AI-enabled DT layers in the presence of legacy systems and changing production demands.

7. Conclusions

This paper investigated how AI and Digital Twins can be systematically combined in industrial CPSs by addressing two research questions (RQ1–RQ2) on their mutual positioning and architectural impact, and by introducing four interaction patterns between AI technologies and DT-enabled shop floors. Within this context, in fact, there is a growing need for AI models capable of continuously processing large amounts of data that are inherently heterogeneous, noisy, and dynamic. This is the setting where the impact of DTs can be crucial in order to fill the gap between the physical and the digital world.

Starting from a deep analysis of the integration challenges between AI and DTs, we propose four interaction patterns that take into account the different ways in which these two technologies can be combined in industrial applications, thus responding to RQ1. Three patterns refer to settings where there is an external AI system that interacts with a single DT or a layer of DTs: we name such patterns (i) the *AI Observing DTs*, (ii) the *AI Advising DTs*, and (iii) the *AI Controlling DTs*, according to the different ways in which the information flows across AI and DTs. Then, a last pattern considers an AI model living *inside* a DT: we name such pattern the *AI Embedded in DT*.

In order to guide the reader through the possible applications of each of these patterns in a real-world industrial scenario, we also described an experimental evaluation based on a physical micro-factory research equipment based on a heterogeneous set of controllers, on top of which a layer of DTs has been implemented. Our study covers applications of root cause analysis, intelligent production schedulers, and predictive maintenance models, showing how the proposed patterns are capable of capturing the multi-faceted nature of interactions between AI and DTs in a CPPS. The experimental evaluation is then completed by assessing the modularity, reuse, maintainability an legacy integration capability of each pattern—thus responding to RQ2.

In future works, we plan to move our analysis one step forward, so as to quantitatively validate the performance of the proposed interaction patterns on real industrial cases. We are also investigating the possibility to develop modeling tools that will support system design according to the proposed patterns. From the perspective of explainable AI, we are considering the idea of plugging neuro-symbolic approaches within the proposed patterns, especially in those domains where background knowledge is given in some form of structured information, such as rules or constraints. Finally, we will study the evolution of patterns along the temporal domain, by allowing composite patterns to dynamically change throughout the industrial life cycle.

CRediT authorship contribution statement

Matteo Martinelli: Writing – review & editing, Writing – original draft, Methodology, Investigation, Conceptualization. **Marco Lippi:** Writing – review & editing, Validation, Investigation, Conceptualization. **Marco Picone:** Writing – review & editing, Methodology, Conceptualization. **Stefano Mariani:** Writing – review & editing, Validation, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code and datasets supporting the findings of this study are publicly available at: https://github.com/lian-yue0515/ACE-LIO.

[2] Eclipse Ditto: https://eclipse.dev/ditto/.

References

[1] W. Elmaraghy, H. Elmaraghy, T. Tomiyama, L. Monostori, Complexity in engineering design and manufacturing, CIRP Ann 61 (2012) 793–814, http://dx.doi.org/10.1016/J.CIRP.2012.05.001.

[2] K. Park, G.E.O. Kremer, Assessment of static complexity in design and manufacturing of a product family and its impact on manufacturing performance, Int. J. Prod. Econ. 169 (2015) 215–232, http://dx.doi.org/10.1016/J.IJPE.2015.07.036.

[3] V. Modrak, Z. Soltysova, Exploring the complexity levels of discrete manufacturing processes, IFAC-PapersOnLine 52 (2019) 1444–1449, http://dx.doi.org/10.1016/J.IFACOL.2019.11.402.

[4] S.N. Samy, T. Algeddawy, H. Elmaraghy, A granularity model for balancing the structural complexity of manufacturing systems equipment and layout, J. Manuf. Syst. 36 (2015) 7–19, http://dx.doi.org/10.1016/j.jmsy.2015.02.009.

[5] Z. Jan, F. Ahamed, W. Mayer, N. Patel, G. Grossmann, M. Stumptner, A. Kuusk, Artificial intelligence for industry 4.0: Systematic review of applications, challenges, and opportunities, Expert Syst. Appl. 216 (2023) 119456, http://dx.doi.org/10.1016/J.ESWA.2022.119456.

[6] W. Wang, Y. Zhang, R.Y. Zhong, A proactive material handling method for CPS enabled shop-floor, Robot. Comput.-Integr. Manuf. 61 (2020) 101849, http://dx.doi.org/10.1016/j.rcim.2019.101849.

[7] F. Lolli, A.M. Coruzzolo, C. Forgione, P.G.T. Neto, Ergonomic risk reduction: A height-adjustable mesh truck for picking activities evaluated with a depth camera, Ergon. Des. 77 (2023) http://dx.doi.org/10.54941/AHFE1003371.

[8] A. Padovano, F. Longo, L. Nicoletti, L. Gazzaneo, A. Chiurco, S. Talarico, A prescriptive maintenance system for intelligent production planning and control in a smart cyber-physical production line, Procedia CIRP 104 (2021) 1819–1824, http://dx.doi.org/10.1016/J.PROCIR.2021.11.307.

[9] R. Minerva, G.M. Lee, N. Crespi, Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models, Proc. IEEE 108 (2020) 1785–1824, http://dx.doi.org/10.1109/JPROC.2020.2998530.

[10] M. Martinelli, J. Zhang, A.-K. Splettstoßer, M. Picone, M. Lippi, A. Wortmann, Hierarchical digital twin ecosystem for industrial manufacturing scenarios, in: 2024 50th Euromicro Conference on Software Engineering and Advanced Applications, SEAA, IEEE, 2024, pp. 56–63.

[11] M. Martinelli, A. Barbone, R. Morandi, M. Picone, S. Burattini, A. Ricci, Modular engineering of industrial digital twins: The WLDT approach, in: 2025 21st International Conference on Distributed Computing in Smart Systems and the Internet of Things, DCOSS-IoT, 2025, pp. 451–458, http://dx.doi.org/10.1109/DCOSS-IoT65416.2025.00077.

[12] A. Barbone, N. Bicocchi, M. Martinelli, R. Morandi, M. Picone, On-device AI and digital twins: A synergistic approach to intelligent cyber-physical systems, Future Gener. Comput. Syst. 175 (108068) (2026) 108068.

[13] A. Aghazadeh Ardebili, M. Zappatore, A.I.H.A. Ramadan, A. Longo, A. Ficarella, Digital Twins of smart energy systems: a systematic literature review on enablers, design, management and computational challenges, Energy Inform. 7 (1) (2024).

[14] D. Kaklis, I. Varlamis, G. Giannakopoulos, T.J. Varelas, C.D. Spyropoulos, Enabling digital twins in the maritime sector through the lens of AI and industry 4.0, Int. J. Inf. Manag. Data Insights 3 (2) (2023) 100178, http://dx.doi.org/10.1016/j.jjimei.2023.100178, URL https://www.sciencedirect.com/science/article/pii/S2667096823000253.

[15] M. Grieves, Origins of the digital twin concept, 2016, http://dx.doi.org/10.13140/RG.2.2.26367.61609.

[16] R. Eramo, F. Bordeleau, B. Combemale, M. van den Brand, M. Wimmer, A. Wortmann, Conceptualizing digital twins, IEEE Softw. 39 (2022) 39–46, http://dx.doi.org/10.1109/MS.2021.3130755.

[17] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, M. Picone, Requirements and design patterns for adaptive, autonomous, and context-aware digital twins in industry 4.0 digital factories, Comput. Ind. 149 (103918) (2023) 103918.

[18] B. Tekinerdogan, C. Verdouw, Systems architecture design pattern catalog for developing digital twins, Sensors (Basel) 20 (18) (2020) 5103.

[19] W. Sulaiman Khail, P. Schnizer, Patterns in digital twin development, in: Proceedings of the 29th European Conference on Pattern Languages of Programs, People, and Practices, ACM, New York, NY, USA, 2024, pp. 1–8.

[20] J. Leng, Z. Chen, W. Sha, Z. Lin, J. Lin, Q. Liu, Digital twins-based flexible operating of open architecture production line for individualized manufacturing, Adv. Eng. Inform. 53 (101676) (2022) 101676.

[21] W.W. Weiss, R.S. Balch, B.A. Stubbs, How artificial intelligence methods can forecast oil production, in: Proceedings - SPE Symposium on Improved Oil Recovery, OnePetro, 2002, pp. 212–227, http://dx.doi.org/10.2118/75143-MS.

[22] J. Lee, H. Davari, J. Singh, V. Pandhare, Industrial Artificial Intelligence for industry 4.0-based manufacturing systems, Manuf. Lett. 18 (2018) 20–23, http://dx.doi.org/10.1016/j.mfglet.2018.09.002, URL https://www.sciencedirect.com/science/article/pii/S2213846318301081.

[23] M. Javaid, A. Haleem, R. Suman, Digital twin applications toward industry 4.0: A review, Cogn. Robot. 3 (2023) 71–92, http://dx.doi.org/10.1016/j.cogr.2023.04.003, URL https://www.sciencedirect.com/science/article/pii/S2667241323000137.

[24] L. Ragazzini, E. Negri, M. Macchi, A digital twin-based predictive strategy for workload control, IFAC-PapersOnLine 54 (2021) 743–748, http://dx.doi.org/10.1016/J.IFACOL.2021.08.183.

[25] T. Kreuzer, P. Papapetrou, J. Zdravkovic, Artificial intelligence in digital twins—A systematic literature review, Data Knowl. Eng. 151 (102304) (2024) 102304.

[26] W. Alghamdi, Synchronization patterns for digital twin systems, J. Appl. Data Sci. 5 (3) (2024) 1026–1037.

[27] C. Semeraro, M. Lezoche, H. Panetto, M. Dassisti, Pattern-based digital twin for optimizing manufacturing systems: A real industrial-case application, IFAC-PapersOnLine 54 (1) (2021) 307–312.

[28] B. Picano, M. Becattini, L. Carnevali, E. Vicario, Democratized learning enabling multi-level digital twin model integration, in: 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation, ETFA, 2024, pp. 1–8, http://dx.doi.org/10.1109/ETFA61755.2024.10710988.

[29] S. Burattini, S. Mariani, S. Montagna, M. Picone, A. Ricci, Distributing intelligent functionalities in the Internet of Things with agents and Digital Twins, Internet Things 31 (2025) 101560, http://dx.doi.org/10.1016/J.IOT.2025.101560.

[30] I. Rojek, D. Mikołajewski, E. Dostatni, J. Cybulski, M. Kozielski, Personalization of AI-based digital twins to optimize adaptation in industrial design and manufacturing—review, Appl. Sci. (Basel) 15 (15) (2025) 8525.

[31] C. Savaglio, V. Barbuto, F. Mangione, G. Fortino, Generative digital twins: A novel approach in the IoT edge-cloud continuum, IEEE Internet Things Mag. 8 (1) (2025) 42–48, http://dx.doi.org/10.1109/IOTM.001.2400035.

[32] A. Shojaeinasab, T. Charter, M. Jalayer, M. Khadivi, O. Ogunfowora, N. Raiyani, M. Yaghoubi, H. Najjaran, Intelligent manufacturing execution systems: A systematic review, J. Manuf. Syst. 62 (2022) 503–522.

[33] M.J.C. Samonte, H.B. Chu, L.A.T. Ogaya, Artificial intelligence integration and administration into pharmaceutical manufacturing execution systems: Assessing potentials and challenges for process optimization, in: 2024 14th International Conference on Software Technology and Engineering, ICSTE, 2024, pp. 184–190, http://dx.doi.org/10.1109/ICSTE63875.2024.00040.

[34] Z. Jiang, S. Yuan, J. Ma, Q. Wang, The evolution of production scheduling from Industry 3.0 through Industry 4.0, Int. J. Prod. Res. 60 (11) (2022) 3534–3554.

[35] S. Krishnaveni, T.M. Chen, M. Sathiyanarayanan, B. Amutha, CyberDefender: an integrated intelligent defense framework for digital-twin-based industrial cyber-physical systems, Clust. Comput. 27 (6) (2024) 7273–7306.

[36] S. Krishnaveni, S. Sivamohan, B. Jothi, T.M. Chen, M. Sathiyanarayanan, TwinSec-IDS: An enhanced intrusion detection system in SDN-Digital-Twin-based industrial cyber-physical systems, Concurr. Comput. 37 (3) (2025).

[37] J. Friederich, D.P. Francis, S. Lazarova-Molnar, N. Mohamed, A framework for data-driven digital twins for smart manufacturing, Comput. Ind. 136 (2022) http://dx.doi.org/10.1016/J.COMPIND.2021.103586.

[38] R.S. Peres, X. Jia, J. Lee, K. Sun, A.W. Colombo, J. Barata, Industrial artificial intelligence in industry 4.0 - systematic review, challenges and outlook, IEEE Access 8 (2020) 220121–220139, http://dx.doi.org/10.1109/ACCESS.2020.3042874.

[39] X. Chu, I.F. Ilyas, S. Krishnan, J. Wang, Data cleaning, in: Proceedings of the 2016 International Conference on Management of Data, ACM, New York, NY, USA, 2016, pp. 2201–2222.

[40] P.-O. Côté, A. Nikanjam, N. Ahmed, D. Humeniuk, F. Khomh, Data cleaning and machine learning: a systematic literature review, Autom. Softw. Eng. 31 (2) (2024).

[41] I.F. Ilyas, T. Rekatsinas, Machine learning and data cleaning: Which serves the other? ACM J. Data Inf. Qual. 14 (3) (2022) 1–11.

[42] A. Seetharaman, N. Patwa, A.S. Saravanan, A. Sharma, Customer expectation from Industrial Internet of Things (IIOT), J. Manuf. Technol. Manag. 30 (2019) 1161–1178, URL https://api.semanticscholar.org/CorpusID:197432277.

[43] J. Michael, J. Pfeiffer, B. Rumpe, A. Wortmann, Integration challenges for digital twin systems-of-systems, in: Proceedings - 10th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems, SESoS 2022, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 9–12, http://dx.doi.org/10.1145/3528229.3529384, URL https://dl.acm.org/doi/10.1145/3528229.3529384.

[44] J. Dalzochio, R. Kunst, E. Pignaton, A. Binotto, S. Sanyal, J. Favilla, J. Barbosa, Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges, Comput. Ind. 123 (2020) 103298, http://dx.doi.org/10.1016/J.COMPIND.2020.103298.

[45] A. Encapera, A. Gosavi, S.L. Murray, Total productive maintenance of make-to-stock production-inventory systems via artificial-intelligence-based iSMART, Int. J. Syst. Sci.: Oper. Logist. 8 (2021) 154–166, http://dx.doi.org/10.1080/23302674.2019.1707906.

[46] Y. Li, S. Carabelli, E. Fadda, D. Manerba, R. Tadei, O. Terzo, Machine learning and optimization for production rescheduling in Industry 4.0, Int. J. Adv. Manuf. Technol. 110 (2020) 2445–2463, http://dx.doi.org/10.1007/S00170-020-05850-5.

[47] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital Twin in manufacturing: A categorical literature review and classification, IFAC-PapersOnLine 51 (11) (2018) 1016–1022, http://dx.doi.org/10.1016/j.ifacol.2018.08.474, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018. URL https://www.sciencedirect.com/science/article/pii/S2405896318316021.

[48] M. Lippi, M. Martinelli, M. Picone, F. Zambonelli, Enabling causality learning in smart factories with hierarchical digital twins, Comput. Ind. 148 (2023) 103892, http://dx.doi.org/10.1016/J.COMPIND.2023.103892.

[49] M. Hallgren, J. Olhager, Differentiating manufacturing focus, Int. J. Prod. Res. 44 (2006) 3863–3878, http://dx.doi.org/10.1080/00207540600702290, URL https://www.tandfonline.com/action/journalInformation?journalCode=tprs20.

[50] J. Olhager, Strategic positioning of the order penetration point, Int. J. Prod. Econ. 85 (2003) 319–329, http://dx.doi.org/10.1016/S0925-5273(03)00119-1.

[51] Q. Liu, C. Chen, S. Chen, Key technology of intelligentized welding manufacturing and systems based on the internet of things and multi-agent, J. Manuf. Mater. Process. 6 (2022) http://dx.doi.org/10.3390/JMMP6060135.

[52] S. Mariani, M. Picone, A. Ricci, About digital twins, agents, and multiagent systems: A cross-fertilisation journey, in: Autonomous Agents and Multiagent Systems. Best and Visionary Papers - Workshops, in: Lecture Notes in Computer Science, vol. 13441, Springer, 2022, pp. 114–129, http://dx.doi.org/10.1007/978-3-031-20179-0_8.

[53] M. Geurtsen, J.B.H.C. Didden, J. Adan, Z. Atan, I. Adan, Production, maintenance and resource scheduling: A review, Eur. J. Oper. Res. 305 (2) (2023) 501–529.

[54] Z. He, K. Wang, H. Li, H. Song, Z. Lin, K. Gao, A. Sadollah, Improved Q-learning algorithm for solving permutation flow shop scheduling problems, IET Collab. Intell. Manuf. 4 (2022) 35–44, http://dx.doi.org/10.1049/CIM2.12042.

[55] A. Ricci, A. Croatti, S. Mariani, S. Montagna, M. Picone, Web of digital twins, ACM Trans. Internet Technol. 22 (2022) 1–30, http://dx.doi.org/10.1145/3507909.

[56] T.P. Carvalho, F.A. Soares, R. Vita, R. da P. Francisco, J.P. Basto, S.G. Alcalá, A systematic literature review of machine learning methods applied to predictive maintenance, Comput. Ind. Eng. 137 (2019) 106024, http://dx.doi.org/10.1016/J.CIE.2019.106024.

[57] P. Klein, L. Malburg, R. Bergmann, FTOnto: A domain ontology for a fischertechnik simulation production factory by reusing existing ontologies, LWDA 2454 (2019) 253.

[58] R. Sala, F. Pirola, G. Pezzotta, On the development of the digital shadow of the fischertechnik training factory industry 4.0: An educational perspective, Procedia Comput. Sci. 217 (2023) 640–649.

[59] S. Schneidereit, A.M. Yarahmadi, T. Schneidereit, M. Breuß, M. Gebauer, YOLO-based object detection in industry 4.0 fischertechnik model environment, in: Proceedings of SAI Intelligent Systems Conference, Springer, 2023, pp. 1–20.

[60] A. Hellwig, J. Michael, J. Pfeiffer, B. Rumpe, H. Thillmann, A. Wortmann, Digital twins in manufacturing and the use of models, 2025.

[61] S. Nakajima, Introduction to TPM: Total Productive Maintenance, Productivity Press, 1995, pp. 0–129.

[62] Y.F. Wang, Adaptive job shop scheduling strategy based on weighted Q-learning algorithm, J. Intell. Manuf. 31 (2020) 417–432, http://dx.doi.org/10.1007/S10845-018-1454-3.

[63] X. Gu, X. Jin, J. Ni, Prediction of passive maintenance opportunity windows on bottleneck machines in complex manufacturing systems, J. Manuf. Sci. Eng. Trans. ASME 137 (2015) http://dx.doi.org/10.1115/1.4029906/376298.