Cyber-Physical Interaction Engineering of Industrial Digital Twin Ecosystems

Cyber-Physical Interaction Engineering of Industrial Digital Twin Ecosystems

Mattia Fogli*, Gaia Mazziotta*, Samuele Burattini†, Marco Picone‡, Carlo Giannelli*, Cesare Stefanelli*
* *University of Ferrara, Italy,* † *University of Bologna, Italy,* ‡ *University of Modena and Reggio Emilia, Italy*
{mattia.fogli, gaia.mazziotta, carlo.giannelli, cesare.stefanelli}@unife.it, samuele.burattini@unibo.it, marco.picone@unimore.it

***Abstract*—Industrial digital twin (DT) ecosystems consist of potentially heterogeneous DTs interacting with their respective counterparts in the physical space and with digital entities in the virtual space. Such interactions may vary dynamically and unpredictably, making their engineering a complex and largely unexplored research field. To address this gap, this work revisits the service interaction patterns originally identified in the context of collaborative business processes. This includes a discussion on whether, and to what extent, such patterns are suitable for industrial DT ecosystems. The focus is on the issues of entanglement and composition in cyber-physical interactions. Next, the paper proposes a set of requirements to engineer cyber-physical interactions and engineers a DT implementation compliant with such requirements. Finally, the implemented DT is experimentally evaluated in an industrial scenario.**

***Index Terms*—cyber-physical interactions, digital twins, ecosystems, industrial internet of things, software engineering**

I. Introduction

The concept of digital twin (DT) was first introduced by Grieves in 2002 [1], although the term DT was only coined years later by Vickers [2]. As defined by Grieves [2], the DT model consists of a physical twin (PT) that exists or will exist in the physical space (PS), a virtual counterpart (i.e., the DT) that represents the PT in the virtual space (VS), and a connection between them. Since then, DTs have been proposed for various industrial applications, such as job scheduling [3], anomaly detection [4], and zero defect manufacturing [5].

Minerva et al. [6] identified a set of characterizing properties related to DTs. Some of these properties are foundational, meaning that without them, there is no real DT implementation. *Entanglement* is one such foundational property. Ideally, entanglement ensures that any change of interest in the PS is timely reflected in the VS, and vice versa. Whenever there is an undesired misalignment between the spaces, time is required for reconciliation [7].

Typically, the PS does not consist of a single PT to be represented in the VS. Rather, it consists of an ecosystem of PTs. Accordingly, the VS will likely form a digital ecosystem—the counterpart of the physical ecosystem [8]. Therefore, *a DT ecosystem is a system in which potentially heterogeneous DTs interact with their respective PTs in the PS and with digital entities in the VS*.

In an industrial DT ecosystem, some DTs may be entangled with a single PT, while others may be entangled with a set of logically correlated PTs. For example, a production line in an industrial environment, which is a set of logically correlated PTs (machines, conveyor belts, sensors, etc.), may be represented in the VS by a single DT. Minerva et al. [6] defined *composability* as the ability to group a set of PTs into a composed one. Larsen et al. [9] and Gill et al. [10] also investigated compositions of DTs. This work focuses on the concept of composability as defined by Minerva et al. [6].

Industrial DT ecosystems may have particularly complex interactions. These may be classified in *cyber-physical interactions* and *digital interactions*. The former, occurring between the VS and PS, describe how DTs interact with their respective PTs. The latter, merely occurring in the VS, describe how DTs interact with each other or with traditional services. As both the PS and VS tend to change over time, interactions must vary accordingly. This requires engineering DT interactions in a flexible, configurable, and extensible manner.

Interaction patterns have been extensively studied in the context of traditional services. Notably, Barros et al. [11] identified how services typically interact with each other to accomplish collaborative business processes. In a collaborative business process, parties interact with each other in a variety of ways, and the number of parties involved may reach dozens or even hundreds. An example of collaborative business process is the following. Consider a travel agency that allows contingent flight reservations in specific situations—such as urgent requests or high-demand routes. Customers provide an ordered list of preferred flight carriers, and the agency contacts these carriers sequentially, attempting to secure a reservation within a short timeframe. Once a reservation is successfully made with one of the carriers, the agency ceases further contact attempts and informs the customer.

A peculiarity of collaborative business processes is that all the interacting parties belong to the VS. In contrast, industrial DT ecosystems are also characterized by cyber-physical interactions. Although several patterns for DTs have already been proposed in the literature [12], [13], [14], [15], [16], interaction patterns for DT ecosystems remain largely unexplored. The contributions of this paper are as follows:

- The interaction patterns proposed by Barros et al. [11] are revisited in the context of DTs. This includes a discussion on whether, and to what extent, such patterns are suitable for industrial DTs ecosystems. The focus is on the issues of entanglement and composition in cyber-physical interactions.

- A set of requirements to engineer cyber-physical interactions is proposed. These requirements are flexibility, configurability, and extensibility.
- The engineering of a DT compliant with such requirements is showcased and experimentally evaluated in an industrial scenario.

The paper is organized as follows. Section II introduces the main architectural abstractions of a DT, discusses entanglement in this context, and provides the rationale behind this work. Section III revisits the interaction patterns proposed by Barros et al. [11], compares cyber-physical and digital interactions, and proposes a set of requirements to engineer cyber-physical interactions for industrial DT ecosystems. Section V engineers a DT implementation compliant with such requirements and experimentally evaluates it in an industrial scenario. Section VI gives an overview of related work. Section VII provides conclusive remarks.

II. Background and Motivation

A. Background

A DT can be described as the combination of three high-level components [6], [17]. First, the *physical interface* receives data (i.e., a virtual representation) from and sends actions (i.e., a physical actuation) to PTs. Second, the core of the DT encapsulates a *model*, which shapes the DT behavior. The model is used by the *shadowing function* (or mirroring function) to compute the digital state, i.e., the *DT state*. One or more *augmentation functions* could also rely on the model to augment the capabilities of the PTs in the VS. Third, the *digital interface* publishes digital states and supports the invocation of digital actions to modify the behavior of the PTs through the DT. Fig. 1 schematically depicts these architectural components, highlighting that DTs behave as bridges between the PS and the VS.

This bidirectional information flow ensures that the DT can accurately twin its physical counterpart(s), with the fidelity of this process determined by model granularity and reconciliation requirements. High-fidelity DTs provide precise and timely virtual representation of the physical counterpart(s)—a crucial requirement for a variety of applications, e.g., from predictive maintenance to operational optimization.

In the context of DTs, the concept of *entanglement* [6], [7] has been defined as the instantaneous exchange and computation of information between DTs and PTs. Entanglement ensures DT quality and fidelity. This concept is influenced by three primary factors: networking (the time required to transfer data between the spaces), computation (the time required to process the received data), and data quality (insufficient or low-quality data can prevent successful reconciliation). A proper characterization of cyber-physical interactions is fundamental for structured DT modeling. This applies to both simple cyber-physical interactions, where a DT represents a single PT, and complex ones, where a DT represents a set of logically correlated PTs.

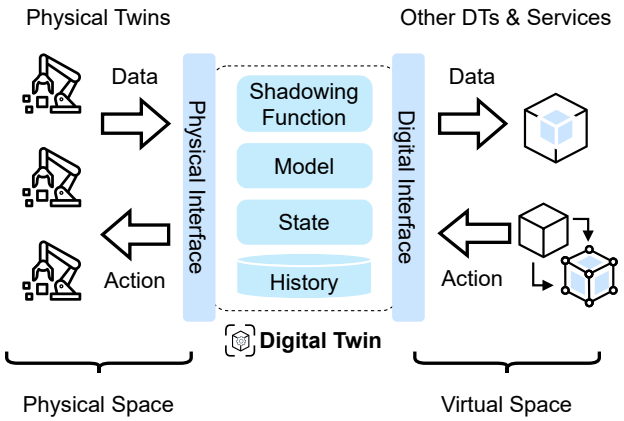

Fig. 1: Main architectural components of a DT.

B. Motivation

As previously mentioned, Barros et al. [11] identified a set of patterns that describe the service interactions that typically occur to accomplish collaborative business processes. Table I provides an overview of these patterns and Fig. 2 schematically depicts them. The patterns differ in terms of the number of parties involved in an interaction (bilateral vs. multilateral), the maximum number of exchanges that occur per interaction (single-transmission vs. multi-transmission), and whether the party that receives the response is the one who sent the request or not (round-trip vs. routed). Note that, as defined by the authors, bilateral interactions cover both one-way and two-way interactions, and single-transmission interactions refer to those that involve a maximum of two exchanges per interaction.

A collaborative business process may involve several services, or parties. These parties may interact with each other in various ways. For example, $\mathcal{P}_{3a}$ (see Table I and Fig. 2) is the *send/receive* pattern. This pattern describes a single-transmission, bilateral interaction where party X sends a request to party Y and, consequently, party X receives a response from party Y.

In the context of collaborative business processes, the identified patterns involve only parties that belong to the VS (digital interactions). In contrast, an industrial DT ecosystem is also characterized by cyber-physical interactions, which are foundational for the so-called *twinning process*—the process ensuring that changes in one space are reflected in the other. In fact, a DT cannot possibly build a virtual representation of its PT without first receiving enough useful data from the PS.

The way interactions are engineered can affect both the time required to transfer data between the spaces and the time required to process the received data. It is worth remarking that transfer and processing times are two of the factors influencing entanglement (the third is data). The same considerations are even more relevant in the case of composition, as there are multiple interactions concurrently occurring between the spaces. As these are not concerns of traditional services, there is a lack of discussion on whether, and to what extent, the

TABLE I: Interaction patterns

Type	Ref.	Name	Description
Single-transmission, bilateral	$\mathcal{P}_1$	Send	Party X sends a message to party Y
	$\mathcal{P}_2$	Receive	X receives a message from Y
	$\mathcal{P}_{3a}$	Send/receive	X sends a request to Y, which, in turn, replies to X
	$\mathcal{P}_{3b}$	Receive/send	Dual of $\mathcal{P}_{3a}$
Single-transmission, multilateral	$\mathcal{P}_4$	Racing incoming messages	X expects to receive one among a set of (different types of) messages
	$\mathcal{P}_5$	One-to-many send	X sends messages (same type, but content may differ) to $Y_1, \ldots, Y_n$
	$\mathcal{P}_6$	One-from-many receive	X receives logically correlated messages from $Y_1, \ldots, Y_n$ in a time frame
	$\mathcal{P}_{7a}$	One-to-many send/receive	X sends a request to $Y_1, \ldots, Y_n$. Requests may be identical or logically correlated. Responses are expected in a given time frame
	$\mathcal{P}_{7b}$	One-from-many receive/send	Dual of $\mathcal{P}_{7a}$
Multi-transmission	$\mathcal{P}_8$	Multi-responses	X sends a request to Y, which, in turn, replies until required
	$\mathcal{P}_9$	Contingent requests	X sends a request to Y_1. If X does not receive a response within a certain time frame, X sends a request to Y_2, then Y_3, and so on
	$\mathcal{P}_{10}$	Atomic multicast notification	X sends notifications to $Y_1, \ldots, Y_n$ such that $Z \subset Y$ are required to accept the notification within a certain time frame
Routing	$\mathcal{P}_{11}$	Request with referral	X sends a request to Y indicating that any response should be sent to $Z_1, \ldots, Z_n$ depending on certain conditions
	$\mathcal{P}_{12}$	Relayed request	X sends a request to Y, which, in turn, delegates to $Z_1, \ldots, Z_n$. $Z_1, \ldots, Z_n$ interact with party X, while Y observes the interactions
	$\mathcal{P}_{13}$	Dynamic routing	A request is routed to $X_1, \ldots, X_n$ based on certain conditions

patterns identified by Barros et al. [11] (see Table I and Fig. 2) are suitable to engineer cyber-physical interactions in industrial DT ecosystems. The following bridges this gap.

III. Cyber-Physical Interaction Engineering

This section is organized as follows. Section III-A models interactions for industrial DT ecosystems. Section III-B compares cyber-physical and digital interactions. Section III-C makes the case for engineering cyber-physical interactions in a flexible, configurable, and extensible manner.

A. Problem Modeling

In an industrial DT ecosystem, each $\{dt_1, dt_2, \ldots, dt_m\} \in \mathcal{DT}$ may be entangled with one or more PTs $\{pt_1, pt_2, \ldots, pt_n\} \in \mathcal{PT}$ of the PS ($\mathcal{PT} \subseteq \mathcal{PS}$). In the VS, there may also be other digital entities, such as traditional services $\{s_1, s_2, \ldots, s_k\} \in \mathcal{S}$. Therefore, $\mathcal{DT} \subseteq \mathcal{VS}$. Then, the set of interacting parties between the spaces can be formally defined as follows:

$$\forall pt \in \mathcal{PT}, \exists dt \in \mathcal{DT} : \{pt, dt\} \in \mathcal{I}_{\mathcal{PS}-\mathcal{VS}}, \tag{1}$$

where $\{pt, dt\}$ indicates that party pt and party dt interact with each other, and $\mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$ is the set of interacting pairs involved in cyber-physical interactions. The underlying assumption is that $\exists! pt \in \mathcal{PT}, s \in \mathcal{S} : \{pt, s\} \in \mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$, as DTs always mediate cyber-physical interactions and thus a PT directly interacting with a traditional service cannot exist.

For each interacting pair $\{pt, dt\}$, (pt, dt) represents the interaction by which dt builds the virtual representation of pt (regardless of which party actually initiated the interaction), and (dt, pt) represents the interaction by which dt acts on pt (this does not apply to DTs that do not act on PTs). If $dt \in \mathcal{DT}$ is the composition of multiple PTs, say $pt_1, pt_2, pt_3 \in \mathcal{PT}$, then $\{pt_1, dt\}, \{pt_2, dt\}, \{pt_3, dt\} \in \mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$.

The identified patterns (see Table I and Fig. 2) can be used to label interactions. Consider the previous example on composition. Then, $(pt_1, dt)_{\mathcal{P}_{3a}, \mathcal{P}_6}$, $(pt_2, dt)_{\mathcal{P}_6}$, $(pt_3, dt)_{\mathcal{P}_6}$, $(dt, pt_1)_{\mathcal{P}_5}$, $(dt, pt_2)_{\mathcal{P}_5}$, and $(dt, pt_3)_{\mathcal{P}_5}$ mean that party dt:

- Sends a state request to party pt_1 and expects a response. This interaction follows the *send/receive* pattern ($\mathcal{P}_{3a}$);
- Correlates state information from parties pt_1, pt_2, and pt_3. This interaction follows the *one-from-many receive* pattern ($\mathcal{P}_6$);
- Sends actions to pt_1, pt_2, and pt_3. This interaction follows the *one-to-many send* pattern ($\mathcal{P}_5$).

In this example, dt adopts different interaction patterns to get state information from its PTs. For example, pt_1 may provide a Hypertext Transfer Protocol (HTTP) endpoint from which to get state information, while pt_2 and pt_3 may publish states as Message Queuing Telemetry Transport (MQTT) messages.

The set of all possible interacting parties $\mathcal{I}$ is not limited to $\mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$. In fact, DTs may interact with each other or with traditional services. For example, an industrial application (i.e., a service) may run a what-if analysis to determine whether a reconfiguration of a production line improves operational efficiency or not. Therefore, $\mathcal{I} = \mathcal{I}_{\mathcal{PS}-\mathcal{VS}} \cup \mathcal{I}_{\mathcal{VS}}$, where $\mathcal{I}_{\mathcal{VS}}$ is the set of interacting parties in the VS.

B. Cyber-Physical Interactions vs. Digital Interactions

It is worth noting that cyber-physical interactions ($\mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$) are fundamentally different from digital interactions ($\mathcal{I}_{\mathcal{VS}}$).

First, any $\{pt, dt\} \in \mathcal{I}_{\mathcal{PS}-\mathcal{VS}}$ affects entanglement as the resulting interactions are done either to build a virtual representation in the VS (pt, dt) or to act on the PS (dt, pt). In this regard, there are some patterns that may be good fallback options to ensure entanglement under challenging circumstances. In particular:

- The *racing incoming messages* pattern ($\mathcal{P}_4$) applies when dt no longer receives the expected volume of data from the PS, but a good approximation of the digital state can still be computed using a subset of such data.

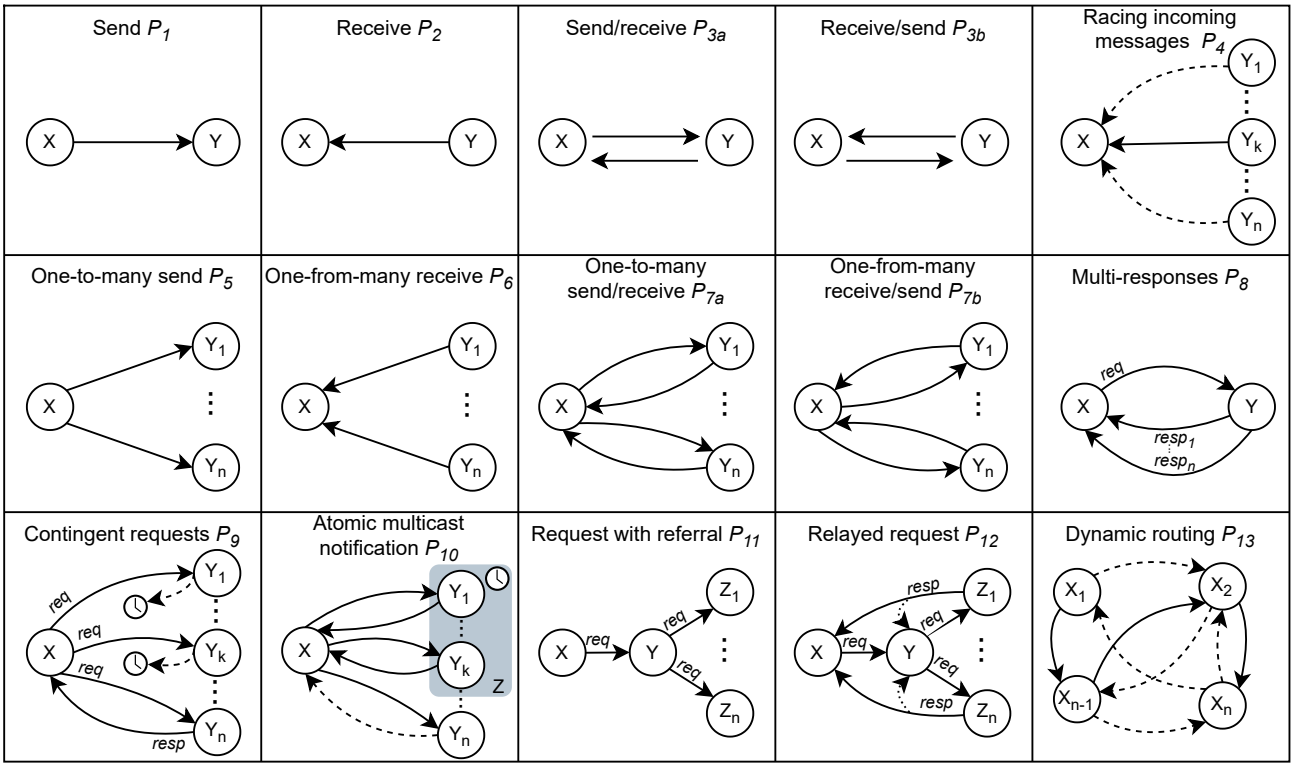

Fig. 2: Schematic representation of the interaction patterns.

- The *contingent requests* pattern ($\mathcal{P}_9$) applies when dt could retrieve some data (perform an action) from (on) different PTs but the first candidate does not satisfy the request timely.
- The *atomic multicast notification* pattern ($\mathcal{P}_{10}$) applies when dt requires to perform a task that consists of some actions that must be carried out in a given order and at a given time (that is, a choreography of actions).

These considerations do not apply to any $\{s, dt\} \in \mathcal{I}_{\mathcal{VS}}$, as there is no physical counterpart involved. However, digital interactions could still indirectly affect entanglement. For example, s can send a request to dt that requires an action on pt to be fulfilled.

Second, any $dt \in \mathcal{DT}$ performs composition if

$$|\{pt \in \mathcal{PT} : \{pt, dt\} \in \mathcal{I}_{\mathcal{PS}-\mathcal{VS}}\}| > 1. \quad (2)$$

The combination of multiple patterns may be required if the PTs expose heterogeneous interfaces, such as pull (HTTP) vs. push (MQTT) models as in the previous example on composition. It is worth remarking that composition adds complexity to the problem of entanglement. On the one hand, data coming from different PTs must be correlated to compute the digital state. This requires a correlation policy and a stop condition to not wait forever if a party fails to send a message. The stop condition must be selected based on the required entanglement. On the other hand, actions sent by the DT to the PTs must be properly orchestrated while acting on the PS. In this regard, the *one-to-many send* pattern ($\mathcal{P}_5$), *one-from-many receive* pattern ($\mathcal{P}_6$), *one-to-many send/receive* pattern ($\mathcal{P}_{7a}$), and *one-from-many receive/send* pattern ($\mathcal{P}_{7b}$) are the natural fit to engineer interactions that involve multiple physical counterparts (i.e., multiple parties). As described above, the *racing incoming messages* pattern ($\mathcal{P}_4$), *contingent requests* pattern ($\mathcal{P}_9$), and *atomic multicast notification* pattern ($\mathcal{P}_{10}$) may also help ensure entanglement under challenging circumstances.

Third, any $dt \in \mathcal{DT}$ must necessarily be aware of its physical counterpart(s). This does not mean that $|\mathcal{I}_{\mathcal{PS}-\mathcal{VS}}|$ may not vary over time. In fact, a new party (be it a $pt \in \mathcal{PT}$ or a $dt \in \mathcal{DT}$) may enter into a space or the existing cyber-physical interactions may change. The former is a new deployment, while the latter is a reconfiguration. In both cases, responsibility lies with those in control of the system, whether it is a technician who installs (or reconfigures) a PT or a system engineer who deploys (or reconfigures) a DT. Therefore, $|\mathcal{I}_{\mathcal{PS}-\mathcal{VS}}|$ may vary, but always predictably (known known). Note that this simplifies the implementation of some patterns, such as the *one-to-many send* pattern ($\mathcal{P}_5$), *one-from-many receive* pattern ($\mathcal{P}_6$), *one-to-many send/receive* pattern ($\mathcal{P}_{7a}$), *one-from-many receive/send* pattern ($\mathcal{P}_{7b}$), *atomic multicast notification* pattern ($\mathcal{P}_{10}$), and *dynamic routing* pattern ($\mathcal{P}_{13}$), for which the potential unknown number of participating parties is traditionally identified as an issue. In contrast, digital interactions are unpredictable. In fact, services are not always

under the control of those who maintain the industrial DT ecosystem. Therefore, $|\mathcal{I}_{\mathcal{VS}}|$ may vary unpredictably (known unknown).

Lastly, the *multi-responses* pattern ($\mathcal{P}_8$), *request with referral* pattern ($\mathcal{P}_{11}$), *relayed request* pattern ($\mathcal{P}_{12}$), and *dynamic routing* pattern ($\mathcal{P}_{13}$) do not represent typical cyber-physical interactions. Specifically, the *multi-responses* pattern ($\mathcal{P}_8$) describes an atypical interaction as data tend to flow periodically between the spaces—especially from the PS to the VS. Instead, the *request with referral* pattern ($\mathcal{P}_{11}$), *relayed request* pattern ($\mathcal{P}_{12}$), and *dynamic routing* pattern ($\mathcal{P}_{13}$) refer to routed interactions where the sender is not the receiver, making it particularly challenging to quantify entanglement for the sender. Therefore, the scope of these patterns is primarily limited to digital interactions. For example, the *multi-responses* pattern ($\mathcal{P}_8$) may be implemented by a service that wants to observe the behavior of a DT over a given period.

C. Engineering Requirements

Cyber-physical interactions are driven by how the PS is organized. Consider an industrial machine. If the machine is a single element, then the corresponding DT may implement (a combination of) the *send* pattern ($\mathcal{P}_1$), *receive* pattern ($\mathcal{P}_2$), *send/receive* pattern ($\mathcal{P}_{3a}$), or *receive/send* pattern ($\mathcal{P}_{3b}$). Instead, if the industrial machine is a group of logically correlated PTs, then the corresponding DT may have to employ composition. Since the lifetime of industrial machines tends to be in the order of years (if not even decades), the so-called *retrofitting* is a common practice. Retrofitting refers to the process of upgrading an existing system by adding newer hardware. As manufacturers usually forbid software upgrades for safety reasons, the newly installed hardware may not be fully integrated with the upgraded industrial machine. For example, the industrial machine may not provide state information about newly installed hardware, which, therefore, is to be queried independently. This would require a switch to a single-transmission, multilateral interaction, thus (a combination of) the *one-to-many send* pattern ($\mathcal{P}_5$), *one-from-many receive* pattern ($\mathcal{P}_6$), *one-to-many send/receive* pattern ($\mathcal{P}_{7a}$), and *one-from-many receive/send* pattern ($\mathcal{P}_{7b}$). In other words, any $dt \in \mathcal{DT}$ might begin its life cycle with a $\{pt_1\} \in \mathcal{PT}$—thus single-transmission, unilateral interactions—and end up with $\{pt_1, pt_2, \ldots, pt_n\} \in \mathcal{PT}$—thus single-transmission, multilateral interactions. This requires **flexibility**. To this purpose, we claim that a DT is flexible if it can support different types of interactions transparently.

As the number of physical counterparts grows, the type of interaction itself may not necessarily change but its configuration certainly does. Consider a $dt \in \mathcal{DT}$ with two PTs, say $\{pt_1, pt_2\} \in \mathcal{PT}$, that does *not* act on the PS, and whose interactions follow the *one-from-many receive* pattern ($\mathcal{P}_6$). Therefore, such interactions can be described as follows: $(pt_1, dt)_{\mathcal{P}_6}$ and $(pt_2, dt)_{\mathcal{P}_6}$. Then, suppose $pt_3 \in \mathcal{PT}$ enters into the system. Even with the assumption that the new interaction follows the same pattern, i.e., $(pt_3, dt)_{\mathcal{P}_6}$, dt would stop correlating as soon as the states of pt_1 and pt_2 come in if the stop condition is not configured accordingly. This requires **configurability**. To this purpose, we claim that a DT is configurable if it can support (re)configuration of the interactions in place dynamically.

Lastly, the number of physical counterparts may negatively affect entanglement. A countermeasure to increasing composition complexity may be the combination of multiple patterns. In the previous example, the data provided by pt_2 and pt_3 may be helpful but not necessary to produce a valid digital state. In this case, the interactions may be engineered as follows: $(pt_1, dt)_{\mathcal{P}_6,\mathcal{P}_4}$, $(pt_2, dt)_{\mathcal{P}_6}$, and $(pt_3, dt)_{\mathcal{P}_6}$, with the *racing incoming messages* pattern ($\mathcal{P}_4$) indicating that dt stops correlating as long as the state of pt_1 comes in. Note that this is neither a transition from one type of interaction to another, nor a (re)configuration of an existing interaction, but a brand-new type of interaction. This requires **extensibility**. To this purpose, we claim that a DT is extensible if it can accommodate new types of interactions seamlessly.

IV. Implementation

We engineered a DT that supports cyber-physical interactions in a flexible, configurable, and extensible manner. Fig. 3 shows the architecture of the implemented DT and the additional components (i.e., physical adapter, aggregation service, and digital adapter) to meet the engineering requirements described in Section III-C. Note that the designed architecture also aligns with the general DT architecture discussed in Section II (see Fig.1).

First, we developed a library that provides the required abstractions for the twinning process. This library includes classes to model physical and digital states, methods to select the aspects of interest of the physical counterpart(s) (i.e., the shadowing function), mechanisms to manage digital state augmentation, and a metric to measure entanglement. Using these abstractions, we developed the DT. Specifically, the DT exposes a set of representational state transfer (REST)/HTTP endpoints to retrieve previously computed digital states and trigger the twinning process when required. The DT relies on an external database to save the history.

Then, we developed physical and digital adapters. The physical adapter translates from MQTT—the protocol supported by the PTs—to HTTP—the protocol supported by the DT. Note that the physical adapter enables cyber-physical interactions between heterogeneous parties—namely, PTs publishing physical states as MQTT messages, and the DT exposing REST/HTTP endpoints. In terms of interaction patterns, the PTs implemented a push model—the *receive* pattern ($\mathcal{P}2$) from the perspective of the DT—while the DT followed a request-response model, corresponding to the *receive/send* pattern ($\mathcal{P}3b$) when retrieving physical states. The digital adapter provides the same service, but towards the VS. Also, we implemented an aggregation service operating as a proxy between the physical adapter and the DT. The aggregation service aggregates incoming physical states based on a correlation policy and triggers the twinning process with a single aggregated request. The correlation continues until a stop

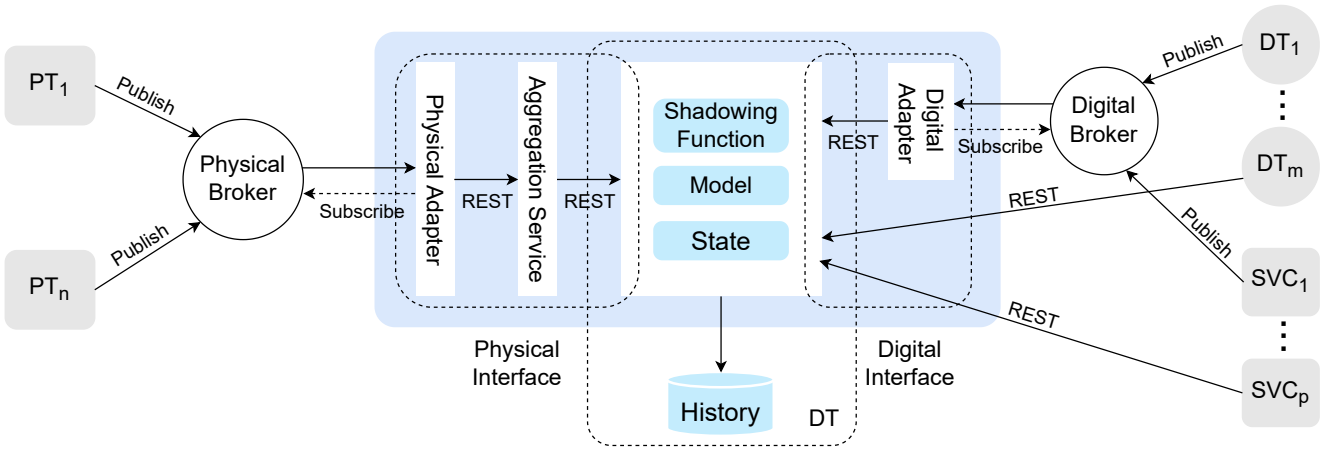

Fig. 3: Architecture overview of the implemented DT.

condition is met, which may be either a timeout or the receipt of (a subset of) the expected physical states.

We developed the DT, physical and digital adapters, and aggregation service in Python. The adapters and the aggregation service were deployed as sidecars of the DT. Sidecars are auxiliary containers that run in the same pod as the main application container. They are designed to extend or support the functionality of the main application container by offering additional services without requiring changes to the main application code. The benefits of this approach are as follows. First, different types of interactions can be supported transparently (flexibility), and new types of interactions can be accommodated seamlessly (extensibility) by either adding or combining sidecars. Second, the deployed sidecars can be (re)configured dynamically once injected (configurability). None of these operations requires any modification or downtime of the DT, whose implementation remains straightforward regardless of the heterogeneity of the spaces or the complexity of the interactions.

V. Evaluation

A. Setup

We provisioned the testbed in a private OpenStack environment using Terraform and configured it with Ansible. The testbed consisted of five instances running Ubuntu 22.04 LTS, where one instance (m1.small: 1 vCPU, 2 GiB of memory) hosted the scripts to run the experiments, while the remaining instances formed a cluster to deploy containerized digital entities, such as DTs. In the cluster, one instance (m1.medium: 2 vCPU, 4 GiB of memory) ran the control plane, and the three remaining instances (m1.large: 4 vCPU, 8 GiB of memory) served as worker nodes. The following software was installed on the cluster: CRI-O (container runtime), Kubernetes (orchestrator), Flannel (network plugin), and Istio (service mesh). Additionally, we deployed Prometheus to automatically scrape the metrics of interest and Mosquitto as physical and digital broker to relay MQTT traffic from the PS to the VS. Lastly, we released the implemented components with Helm—a package manager for Kubernetes.

B. Results

We used Locust to emulate the PTs. Specifically, we implemented a Locust script to generate MQTT messages containing physical states from a given number of PTs, $\{pt_1, pt_2, \ldots, pt_5\} \in \mathcal{PT}$, at a given rate (1 states/s). Accordingly, we configured the DT ($dt \in \mathcal{DT}$) to expect 1 physical state/s with a desired timeliness of 1 s. As a rule of thumb, the average time elapsed between two changes should be considered the upper limit for updating the counterpart(s) [6]. We emulated state computation by forcing the DT to find a given number of primes when the twinning process is triggered. The interested reader may refer to [7] for a detailed explanation of Overall Digital Twin Entanglement (ODTE)—the metric used for measuring entanglement.

Fig. 4 shows the CPU usage of the DT as the number of PTs grows, with each PT treated independently. In other words, the correlation policy was *not* to correlate any incoming physical states. Formally, $\forall pt \in \mathcal{PT}, (pt, dt)_{\mathcal{P}_2}$. This is a poor design choice because the CPU usage grows linearly with the number of PTs. The main drawback of such a design is the side effect of CPU saturation. As shown in Fig. 5, the ODTE drops well below 1—which would indicate perfect entanglement—when the *receive* pattern ($\mathcal{P}_2$) is employed by five PTs concurrently sending state updates ($5 \times \mathcal{P}_2$).

A countermeasure is to switch to a single-transmission, multilateral interaction pattern, such as the *one-from-many receive* pattern ($\mathcal{P}_6$). These patterns decouple the number of physical states from the number of times the twinning process is triggered. This switch was achieved by reconfiguring the aggregation service. In particular, the correlation policy was changed to correlate the incoming physical states together, with a stop condition of 0.5 s. Therefore, $\forall pt \in \mathcal{PT}, (pt, dt)_{\mathcal{P}_6}$. In other words, if $|\mathcal{PT}| = 5$, the aggregation service aggregates all the incoming physical states published by $pt_1, \ldots, pt_5$, waiting at most 0.5 s before triggering the

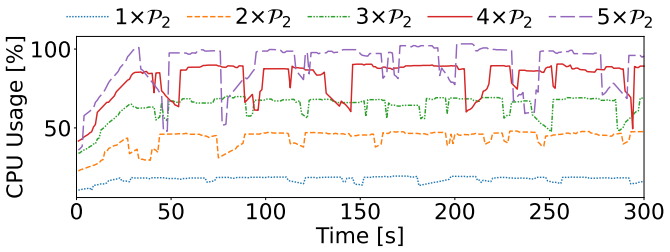

Fig. 4: CPU usage of the DT with $n \times \mathcal{P}_2, 1 \leq n \leq 5$.

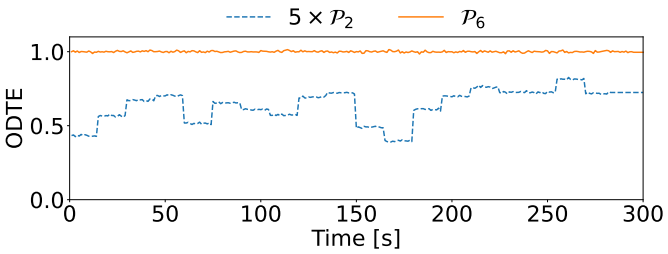

Fig. 5: ODTE comparison: $5 \times \mathcal{P}_2$ vs. $\mathcal{P}_6$ (1 from 5).

twinning process. The stop condition must be set lower than the desired timeliness (i.e., 1 s) to ensure that the physical states are sufficiently fresh when received by the DT. The exact margin depends on the time it takes for the physical states to reach the DT, as well as the time the DT requires to compute the state. Profiling is necessary to determine this value on a case-by-case basis.

Fig. 6 illustrates the impact on the ODTE of a single PT operating at a reduced physical state update rate (pt_5 was configured to publish 0.75 states/s). The observed degradation in ODTE performance occurred because $\mathcal{P}_6$ initiates the twinning process either upon receiving all expected states or when the stop condition is met. Since pt_5 published at a lower rate than $pt_1, \ldots, pt_4$, the aggregation service occasionally triggered the twinning process before sufficient data had been collected. As a result, the DT failed to produce a valid state due to incomplete information, thereby reducing the reliability factor of the ODTE metric.

If the DT can compute the digital state based on a subset of the physical states, a potential countermeasure is to combine the *one-from-many receive* pattern ($\mathcal{P}_6$) with the *racing incoming messages* pattern ($\mathcal{P}_4$). This was accomplished by reconfiguring the stop condition of the aggregation service to stop waiting and trigger the twinning process either upon receiving the physical states from $pt_1, \ldots, pt_4$ or when the predefined stop condition was met (using the same timeout value as before). Formally, this configuration is expressed as $(pt_1, dt)_{\mathcal{P}_4,\mathcal{P}6}, \ldots, (pt_4, dt)_{\mathcal{P}_4,\mathcal{P}6}, (pt_5, dt)_{\mathcal{P}_6}$. In this setup, the physical states from pt_5 were considered optional for computing the digital state. Note that cyber-physical interaction engineering is not merely a technical concern, but also requires domain-specific knowledge, such as the entanglement requirements of the DT. In this regard, an improper configuration of the *racing incoming messages* pattern may disrupt the entanglement, as the DT may no longer consistently receive the expected physical states.

In conclusion, the transition from n instances of the *receive* pattern ($\mathcal{P}_2$) to the *one-from-many receive* pattern ($\mathcal{P}_6$) was seamlessly handled by the aggregation service, transparently to the involved parties (flexibility). Furthermore, the aggregation service was dynamically reconfigured to support an increasing number of PTs (configurability), and it also accommodated the combination of the *one-from-many receive* pattern ($\mathcal{P}_6$) with the *racing incoming messages* pattern ($\mathcal{P}_4$) (extensibility).

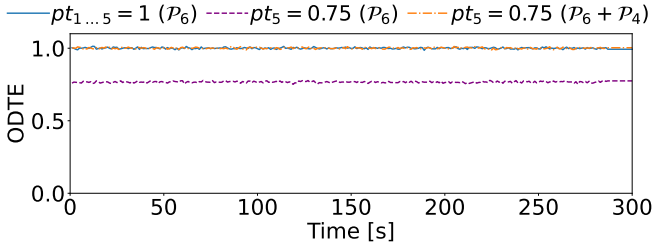

Fig. 6: ODTE degradation of $\mathcal{P}_6$ with a slower PT ($pt_5 = 0.75$) and possible countermeasure ($\mathcal{P}_6 + \mathcal{P}_4$).

VI. Related Work

Several works proposed patterns for DTs. The proposed patterns address recurring problems in communication (Section VI-A), architectural design (Section VI-B), as well as software and system engineering (Section VI-C) of DTs.

A. Communication Patterns

Alghamdi and Albassam [12] identified a catalog of synchronization patterns, where synchronization refers to the process that ensures the DT adequately reflects its physical counterpart(s). In this work, synchronization is conceived as unidirectional—from physical to digital. It is worth noting that these synchronization patterns do not necessarily describe the interactions that occur between the spaces or within the VS. Rather, such patterns describe the ways in which events logically flow from the PS to the VS, as synchronization is considered one-way only. Henneke et al. [18] analyzed the existing protocols used in the context of cyber-physical systems (CPSs) and identified common patterns, specifically, request-response, discovery, and publish-subscribe. They further categorized these patterns based on typical application scenarios, message flows, and protocol suitability, offering a framework for designing communication in CPS. However, the patterns identified by Alghamdi and Albassam [12] and Henneke et al. [18] lie at a higher level of abstraction than the ones identified by Barros et al. [11].

B. Architectural Patterns

Malakuti et al. [19] proposed a four-layer architecture pattern for DTs. Specifically, the layers are information providers, model providers, DT providers, and applications. The information providers may be applications, databases, devices, or platforms. The model-view-controller pattern is used to

manage different kinds of models and DTs. Applications interact with DTs through proper interfaces. Edrisi et al. [13] designed an architectural pattern specifically tailored for DTs of an organization [20]. This pattern consists of a business architecture that provides structural and behavioral models to be twinned, an information architecture that provides the virtual representation of the organization, and a synchronization component for alignment. Although these patterns describe the types of relationships among architectural components, they do not address the interactions that occur.

C. Software and System Engineering Patterns

Bellavista et al. [14] adapted well-known software engineering patterns to the realm of DTs. The objective of these patterns is to support the design of adaptive, autonomous, and context-aware DTs. The authors also elaborated on the requirements met by the discussed patterns, based on the properties identified by Minerva et al. [6]. Tekinerdogan et al. [15] recognized a set of patterns recurring throughout the life cycle of a DT. The stages include concept, development, production, utilization, support, and retirement. These patterns are primarily for system engineers. Lehner et al. [16] proposed a catalog of modeling patterns that extends structural DT models with behavioral models demonstrating how these behaviors can be integrated into existing platforms without requiring heavyweight modifications to platform code bases. Semeraro et al. [21] introduced a data-driven methodology that systematically identifies and formalizes "invariant modeling patterns" for DTs, enabling the creation of a reusable pattern library that simplifies their design. As with architectural patterns, however, software and system engineering patterns do not primarily deal with interactions.

VII. Conclusion

This work revisited the service interaction patterns originally proposed for collaborative business processes, with a focus on the extent to which such patterns apply to cyber-physical interactions in industrial DT ecosystems. Specifically, some of these patterns are appropriate for composition ($\mathcal{P}_5$, $\mathcal{P}_6$, $\mathcal{P}_{7a}$, and $\mathcal{P}_{7b}$), some are useful to guarantee entanglement under challenging circumstances ($\mathcal{P}_4$, $\mathcal{P}_9$, $\mathcal{P}_{10}$), and others do not really fit for purpose ($\mathcal{P}_8$, $\mathcal{P}_{11}$, $\mathcal{P}_{12}$, and $\mathcal{P}_{13}$). Furthermore, a DT compliant with the identified requirements for engineering cyber-physical interactions—namely, flexibility, configurability, and extensibility—was implemented and experimentally evaluated in an industrial scenario. The results demonstrate the practical implementation of selected patterns, highlight the impact of poorly engineered cyber-physical interactions on entanglement, and illustrate how multiple patterns can be combined to effectively address DT composition.

Acknowledgment

This work was partially supported by the Italian Ministry of Education and Research (MUR) PRIN 2022 PNRR DATRUST Project (Project ID: P20225KTR4, CUP: I53D23006060001) and by the European Union - NextGenerationEU, PNRR SERICS PE00000014 Spoke 8 Cascading Funding Z4I Project (CUP 33C22002810001).

References

[1] M. Grieves, "Completing the cycle: Using plm information in the sales and service functions [slides]," in *SME Management Forum*, 2002.

[2] M. W. Grieves, *Digital Twins: Past, Present, and Future*, 2023, vol. 1.

[3] Y. Fang, C. Peng, P. Lou, Z. Zhou, J. Hu, and J. Yan, "Digital-twin-based job shop scheduling toward smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 12, pp. 6425–6435, 2019.

[4] A. Castellani, S. Schmitt, and S. Squartini, "Real-world anomaly detection by using digital twin systems and weakly supervised learning," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4733–4742, 2021.

[5] Z. Lv, J. Guo, and H. Lv, "Safety poka yoke in zero-defect manufacturing based on digital twins," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1176–1184, 2023.

[6] R. Minerva, G. M. Lee, and N. Crespi, "Digital twin in the iot context: A survey on technical features, scenarios, and architectural models," *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020.

[7] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Odte: A metric for digital twin entanglement," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 2377–2390, 2024.

[8] W. Li, Y. Badr, and F. Biennier, "Digital ecosystems: challenges and prospects," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, ser. MEDES '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 117–122. [Online]. Available: https://doi-org.ezproxy.unibo.it/10.1145/2457276.2457297

[9] P. G. Larsen, P. Talasila, and J. Fitzgerald, "Towards the composition of digital twins," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14900 LNCS, p. 103 – 122, 2024.

[10] M. S. Gill, J. Zhang, A. Wortmann, and A. Fay, "Toward automating the composition of digital twins within system-of-systems," in *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024, pp. 1–4.

[11] A. Barros, M. Dumas, and A. H. M. Ter Hofstede, "Service interaction patterns," vol. 3649, 2005, p. 302 – 318.

[12] W. Alghamdi and E. Albassam, "Synchronization patterns for digital twin systems," *Journal of Applied Data Sciences*, vol. 5, no. 3, p. 1026 – 1037, 2024.

[13] F. Edrisi, D. Perez-Palacin, M. Caporuscio, M. Hallberg, A. Johannesson, C. Kopf, and J. Sigvardsson, "Ea blueprint: An architectural pattern for resilient digital twin of the organization," *Communications in Computer and Information Science*, vol. 1462, p. 120 – 131, 2021.

[14] P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, "Requirements and design patterns for adaptive, autonomous, and context-aware digital twins in industry 4.0 digital factories," *Computers in Industry*, vol. 149, p. 103918, 2023.

[15] B. Tekinerdogan and C. Verdouw, "Systems architecture design pattern catalog for developing digital twins," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5103

[16] D. Lehner, S. Sint, M. Eisenberg, and M. Wimmer, "A pattern catalog for augmenting digital twin models with behavior; [ein musterkatalog zur erweiterung von digitalen zwillingsmodellen um verhaltenssichten]," *At-Automatisierungstechnik*, vol. 71, no. 6, p. 423 – 443, 2023.

[17] A. Ricci, A. Croatti, S. Mariani, S. Montagna, and M. Picone, "Web of digital twins," *ACM Trans. Internet Technol.*, vol. 22, no. 4, nov 2022.

[18] D. Henneke, M. Elattar, and J. Jasperneite, "Communication patterns for cyber-physical systems," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, Sep. 2015, pp. 1–4.

[19] S. Malakuti, J. Schmitt, M. Platenius-Mohr, S. Grüner, R. Gitzel, and P. Bihani, "A four-layer architecture pattern for constructing and managing digital twins," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11681 LNCS, p. 231 – 246, 2019.

[20] U. V. Riss, H. Maus, S. Javaid, and C. Jilek, "Digital twins of an organization for enterprise modeling," *Lecture Notes in Business Information Processing*, vol. 400, p. 25 – 40, 2020.

[21] C. Semeraro, M. Lezoche, H. Panetto, and M. Dassisti, "Data-driven invariant modelling patterns for digital twin design," *Journal of Industrial Information Integration*, vol. 31, 2023.