ARCHIVIO ISTITUZIONALE DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Measuring Digital Twin Entanglement in Industrial Internet of Things

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Bellavista, P., Bicocchi, N., Fogli, M., Giannelli, C., Mamei, M., Picone, M. (2023). Measuring Digital Twin Entanglement in Industrial Internet of Things. New York : IEEE [10.1109/ICC45041.2023.10278787].

Availability:
This version is available at: https://hdl.handle.net/11585/953979 since: 2024-01-27

Published:
DOI: http://doi.org/10.1109/ICC45041.2023.10278787

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

10 October 2026

This is the final peer-reviewed accepted manuscript of:

P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei and M. Picone, "Measuring Digital Twin Entanglement in Industrial Internet of Things," ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 5897-5903

The final published version is available online at: https://doi.org/10.1109/ICC45041.2023.10278787

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Measuring Digital Twin Entanglement in Industrial Internet of Things

Paolo Bellavista*, Nicola Bicocchi†, Mattia Fogli‡, Carlo Giannelli‡, Marco Mamei†, Marco Picone†
* *University of Bologna, Italy*; † *University of Modena and Reggio Emilia, Italy*; ‡ *University of Ferrara, Italy*
paolo.bellavista@unibo.it, {nicola.bicocchi, marco.mamei, marco.picone}@unimore.it, {mattia.fogli, carlo.giannelli}@unife.it

***Abstract*—Digital Twins (DTs) have recently emerged as a valuable approach for modeling, monitoring, and controlling physical objects in Industrial Internet of Things applications. Measuring the quality of entanglement between the digital and physical counterparts plays a crucial role in the adoption of DTs. In this paper, we propose a concise yet expressive metric for representing the quality of entanglement, namely Overall Digital Twin Entanglement (ODTE), based on two key factors: *timeliness* and *completeness*. Furthermore, the paper presents the development of our industrial testbed implemented on top of Kubernetes, where we show practical applications of the proposed ODTE metric by highlighting and discussing its benefits in realistic use cases.**

***Index Terms*—Digital Twins, Entanglement, Industrial IoT**

I. Introduction

Under the Industry 4.0 innovation umbrella, Digital Twins (DTs) are finding a place in the realm of Industrial Internet of Things (IIoT) applications [1], [2]. DTs have been defined as digital entities connected to Physical Objects (POs) and serving as their indistinguishable counterparts. While some works focus on DTs for designing and simulating POs and do not require a 'live' connection between DTs and POs at run-time (referred to as digital models in [3]), here we focus on the application domain where DTs mediate the interactions between applications and POs at run-time.

Because of their intrinsic relationship, DTs require regular synchronization with their corresponding POs. Recent works [4], [5], [6] characterize a small set of foundational properties for DTs, among which *reflection* and *entanglement* are focused on the requirement of keeping the DT and the PO synchronized over time. More specifically, the *reflection* property (also denoted as *shadowing* or *mirroring*) defines the digitalization procedure allowing to clone and keep synchronized the behavior and the states of the DT with the associated PO. On the one hand, physical changes or events faced by the PO should be reflected in the DT. On the other hand, variations occurring to the DT should be forwarded to the physical entity to trigger specific changes. In this context, the definition of *entanglement* as the representation of the linkage and the instantaneous exchange of information between DTs and POs highlights its strong relationship with the *reflection* property.

For guaranteeing adequate levels of entanglement and ensuring that DTs have a consistent representation of their associated POs and vice versa, it is of paramount importance to define a metric for measuring it. The standard set of performance indicators for measuring network link performance (required to keep DTs and POs entangled) is usually indicated with the general term of QoS. Traditionally, QoS focuses on network characteristics such as latency, jitter, and packet loss, all of which, even if relevant, are not able to capture application-specific nuances. For instance, an increase of one second in network latency could be irrelevant for an application requiring updates every minute while dramatic for another one monitoring near real-time phenomena and requiring at least ten updates per second. To avoid such issues, alternative measures have been developed, e.g., Quality of Experience (QoE) [7] or Quality of Information (QoI) [8], with the goal of evaluating the performance of applications instead of the network links they rely upon. However, existing QoE definitions focus on evaluating application quality from the lens of their users (e.g., video-conferencing) and are not suited for unsupervised use cases (i.e., applications where human feedback is unavailable). An analysis of the existing literature led Fizza et al. [7] to conclude that measuring QoE of applications where human involvement or feedback is not readily available can be approximated by observing four key features of collected data: *timeliness* (i.e., how fresh the collected data are for actually making decisions), *completeness* (i.e., the ratio of the amount of collected data to the total amount of required data), *accuracy* (i.e., the precision of the collected data), and *usefulness* (i.e., how useful the collected data are for the application).

Based on these considerations, we propose an original metric named *Overall Digital Twin Entanglement (ODTE)* capable of capturing in a concise yet expressive way the quality of entanglement between a DT and its connected PO. It addresses timeliness and completeness while disregarding accuracy and usefulness, which cannot be defined in a general manner but rather should be defined at the application level. ODTE provides, by definition, an easy-to-understand indicator normalized between 0 and 1. In this way, it synthesizes whether the state changes happening in both the DT and the PO are effectively communicated to the counterpart, allowing anyone (or anything) to monitor the communication process without any a priori, application-specific knowledge.

The remainder of the paper is organized as follows. Section II presents related work in the field. Section III outlines a reference scenario and theoretically defines the ODTE metric. Section IV details the in-the-field experiments that we de-

ployed within a real Kubernetes environment to quantitatively measure the proposed metric for two different applications and to highlight the benefits of the proposed metric. Section V draws final remarks.

II. Related Work

Several attempts have been made to define QoE-like metrics capable of enriching traditional QoS indicators with a synthetic representation of application-dependent qualities.

In the field of IoT systems, various attempts have been made to measure application QoE via both subjective and objective means. Concerning the subjective family, recent works [9], [10], [11], [12] propose QoE metrics in different contexts. However, these approaches do not assess how the application QoE relates to the individual components of IoT applications and models it through human evaluations. These approaches are not suitable for evaluating the quality of DT entanglement since it is a product of a machine-to-machine process. Concerning the objective family, in [13], authors aimed to ensure QoE through existing QoS metrics. They proposed a regression model between QoE and QoS indicators (after extracting their principal components). Their results show that, in case of the absence of human feedback, QoE can be derived from QoS parameters. In [14], authors proposed a QoE model for a communication app. They identified five key factors impacting QoE (i.e., integrality, retainability, availability, usability, and instantaneousness) and measured them. The final QoE value is a composition of these five measures, normalized between 0 and 1. Although these two proposals share several aspects with ours (i.e., QoE key factors are identified, QoE measure put in relation with lower-level QoS metrics, output values normalized on a fixed range), they are not tuned for capturing features of the DT entanglement.

In the field of manufacturing, it is widely used the *Overall Equipment Effectiveness (OEE)* metric—a QoE-like measure to evaluate production capabilities synthetically. It is defined as the product of three key factors: *availability* (i.e., the ratio of actually worked time to the total planned working time), *performance* (i.e., the ratio of the number of completed tasks to the number of tasks that could be hypothetically completed in the same period), and *quality* (i.e., the ratio of the number of tasks completed correctly to the total number of completed tasks). Its output is normalized between 0 and 1. Despite OEE shares many commonalities with ODTE, it lacks two key aspects. Firstly, it has been specifically designed for industrial rather than IoT environments. Secondly, it does not consider the *timeliness* aspect (i.e., how fresh the collected data are for actually making decisions), which is paramount for time-dependent applications.

Finally, in the field of software engineering, [15] proposes a metric for quantifying operational coupling in embedded control systems. The quantification of the metric is performed by considering the topology of connections, their multiplicity, replication, frequency, and accuracy of properties of the relationship. Then, individual couplings are combined into an overall coupling, where domain-specific heuristics and technology constraints are used to determine the weighting. Since it could be possible to exploit DTs as control systems, this approach could be applied to DTs as well. However, it is very complex and largely takes into account structural aspects of the software systems, which are not relevant for measuring the entanglement.

III. Measuring Digital Twin Entanglement

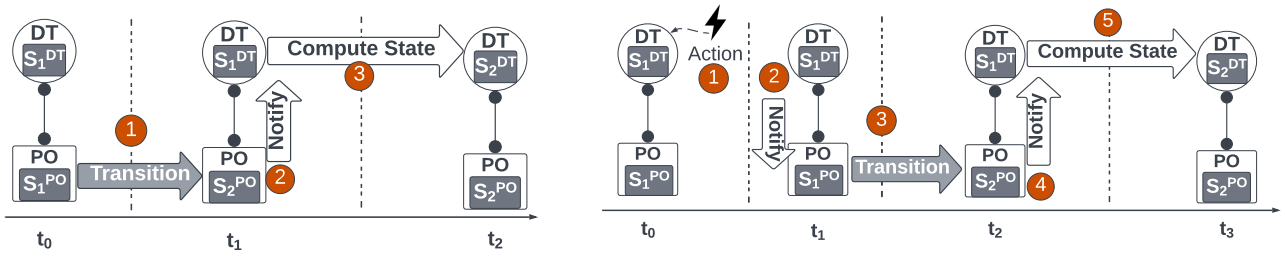
This section explores the concept of entanglement and how it is key for guaranteeing the reflection property between DTs and POs. Starting from these considerations, the ODTE metric is introduced and theoretically defined. From a general perspective, the interactions between DTs and POs can unfold in two key ways: *(a)* a state change in the PO has to be communicated to the DT; *(b)* a request to the DT from an external service (e.g., an IIoT application) has to be communicated to the PO and then a state change confirmation sent back to the DT and the external service.

Fig. 1a schematically depicts the synchronization flow required to keep aligned the DT and PO states (denoted as S_i^{PO} and S_i^{DT}) when the PO detects a state change. At the beginning S^{PO} and S^{DT} are aligned at version 1 (t_0). When a new physical event (e.g., a change in the environment) occurs, it triggers a variation of the physical state (changed to S_2^{PO}) and generates a state update toward the DT. At this point, there is a misalignment between the two counterparts since the physical variation has not yet been reflected on the DT (t_1). Only when the DT receives the state update and computes its new state S_2^{DT} the two counterparts are properly synchronized (t_2). In this first scenario, the entanglement is unidirectional and directly reflects the time shift between the state of the physical entity and its digitalized replica.

Fig. 1b, instead, represents a scenario where an action is performed on the DT (e.g., from an IIoT application) and has to be propagated to the PO. It is worth noting that an action issued on the DT—aiming at modifying the state of the PO—should be intended as another form of state synchronization. When the DT receives the action, it notifies the PO about the request and waits for its state transition (from S_1^{PO} to S_2^{PO}). Then, once the state change on the PO is confirmed, the state of the DT is updated as well (from S_1^{DT} to S_2^{DT}). In this second scenario, the entanglement is even more relevant since a bidirectional exchange of information is required.

A. Overall Digital Twin Entanglement (ODTE)

The main goal of the ODTE metric is to measure in a concise yet expressive way the interactions involving state updates between DTs and POs. Similarly to OEE, it is conceived as a multiplication of factors resulting in a number between 0 and 1. The factors involved—*timeliness* and *completeness*—have been suggested by Fizza et al. [7] for measuring the QoE of applications where data features can be captured while human feedback is unavailable. While we represent *timeliness* (T) as a single factor, *completeness* is represented with two sub-factors: *reliability* (R), i.e., the ratio of the received state updates to



(a) Unidirectional.

(b) Bidirectional.

Fig. 1: Unidirectional and bidirectional synchronization processes between the PO and DT.

the expected ones, and *availability* (A), i.e., the expected uptime of the PO from the perspective of the DT. Accordingly, ODTE is defined as:

$$ODTE = T \times R \times A \tag{1}$$

To quantify the timeliness of a state update, the DT needs to track the rate of incoming status updates over time, the elapsed time between when the PO produces a given update and when the DT receives it, and how long the DT takes to change its state (based on the received update). A suitable way to model this phenomenon is by making use of histograms. The DT may use a histogram to sample observations about the timeliness of the received updates. In this case, an observation o_i may be defined as follows:

$$o_i^{uni} = t_i^{DT} - t_i^{PO} + t_i^{exec} \tag{2}$$

where

- t_i^{DT} is the time at which the DT received the ith update;
- t_i^{PO} is the time at which the PO had produced the ith update;
- t_i^{exec} is the time the DT took to change state as a result of the ith update.

It is worth noting that o_i^{uni} only works for unidirectional entanglement. In the case of bidirectional entanglement, instead, an observation o_i may be modeled as follows:

$$o_i^{bi} = t_i^{PO'} - t_i^{DT'} + o_i^{uni} \tag{3}$$

where

- $t_i^{PO'}$ is the time at which the PO received the command from the DT;
- $t_i^{DT'}$ is the time at which the DT had issued the command.

We can now express the timeliness T as a quantile over a time window:

$$T(\varphi, t, O) \tag{4}$$

where

- $0 \leq \varphi \leq 1$ is the quantile;
- t is a time window (e.g., last 5 minutes);
- O is the set of observations about the received updates.

For example, $T(0.99, now - 5m, O) = 0.100$ means that 99% of the observations had timeliness of at most 100 ms over the last 5 minutes. For computing a normalized metric such as ODTE, it is useful to express the timeliness as a percentage instead of in seconds. Thus, (4) may also be defined as:

$$T'(T_d, t, O) \tag{5}$$

where T_d is the desired timeliness.

Equation (5) expresses the timeliness as a percentage and encapsulates any application-specific detail within the DT itself. It is reasonable, in fact, to assume that a DT is aware of the desired timeliness (T_d) of its physical counterpart. For example, if T_d is set to 200 ms (i.e., anything lower than 200 ms fulfills the requirement), $T'(200ms, 5m, O) = 0.999$ means that 99.9% of the updates had the desired timeliness. By doing so, anyone (or anything) monitoring the DT can understand if the timeliness of state updates respects the entanglement requirements without any a priori, application-specific knowledge.

Timeliness itself does not account for those updates which are never received by the counterpart that, instead, are taken into account by the completeness factor. As stated above, we split the contribution of the completeness factor into two sub-factors, namely R and A. Firstly, R measures the reliability of an entity expressed as the ratio of the received state updates to the expected ones within a specified time frame. Formally:

$$R(t, O) = \frac{u_{measured}(t, O)}{u_{expected}(t)} \tag{6}$$

where

- $u_{measured}(t, O)$ is the per-second average rate of the received updates based on the set of observations O over the time window t;
- $u_{expected}(t)$ is the *minimum* per-second average rate of the expected updates over the time window t. If $u_{measured}(t, O) > u_{expected}(t)$, then $R(t, O) = 1$.

For example, $R(now - 5m) = 0.5$ indicates that the DT received half of the expected updates within the last 5 minutes. Secondly, A measures the availability of the PO over a specified time frame. For example, $A(now - 5m) = 0.5$ means that the PO was active only half of the expected time

over the last 5 minutes. Putting the three components together, the ODTE is defined as:

$$ODTE = T'(T_d, t, O) \times R(t, O) \times A(t) \quad (7)$$

From an operational viewpoint, the DT should be responsible for quantifying its own ODTE to provide either human operators or IIoT applications with a representation of its entanglement. It would also be possible to compute the ODTE outside the DT, e.g., by third parties services querying a time-series database containing T, R, and A.

IV. Experimental Evaluation and Discussion

A notable use case of the Industry 4.0 revolution is the manufacturing sector. Typically, manufacturing firms are logically structured on three primary levels, i.e., shop floor, plant, and enterprise. Specifically, the shop floor level is where industrial automation takes place. This level consists of industrial machines, IIoT devices, human-machine interfaces, and programmable logic controllers. Then, the plant level is about the management of manufacturing processes whose primary component is the Manufacturing Execution System (MES). Lastly, the enterprise level is about decision-making to run business operations. The Enterprise Resource Planning (ERP) system helps managers make such decisions.

The Purdue model is arguably the most common network implementation of such a logical structure. A pillar of the Purdue model is the concept of network segmentation. In particular, the Purdue model recommends a hierarchical approach that splits the industrial network into five layers, with the first three layers related to Operational Technology (OT). The shop floor components crafting goods belong to layer 0/1. This layer relies on a time-sensitive network connecting industrial machines and programmable logic controllers. Then, layer 2 hosts devices that control the crafting processes (e.g., human-machine interfaces), whereas layer 3 includes those components that manage the whole manufacturing process (e.g., the MES). Lastly, layers 4 and 5 are dedicated to Information Technology (IT), such as web servers, email servers, databases, and the ERP system.

From a network perspective, each layer of the Purdue model is supposed to provide different performances. As a rule of thumb, the lower, the better. For example, since the goal of layer 0/1 is to meet safety-critical requirements, the network is expected to provide high reliability and low latency (e.g., within 10 ms). However, layer 0/1 does not provide as abundant computing resources as the upper layers. In this regard, layer 2 tends to provide worse network performance (e.g., within 25 ms) than layer 0/1 but potentially more computing resources and less stringent security requirements. As a result, it is paramount to quantify the ODTE under a spectrum of plausible network conditions to decide upon the optimal deployment of DTs within the target industrial environment.

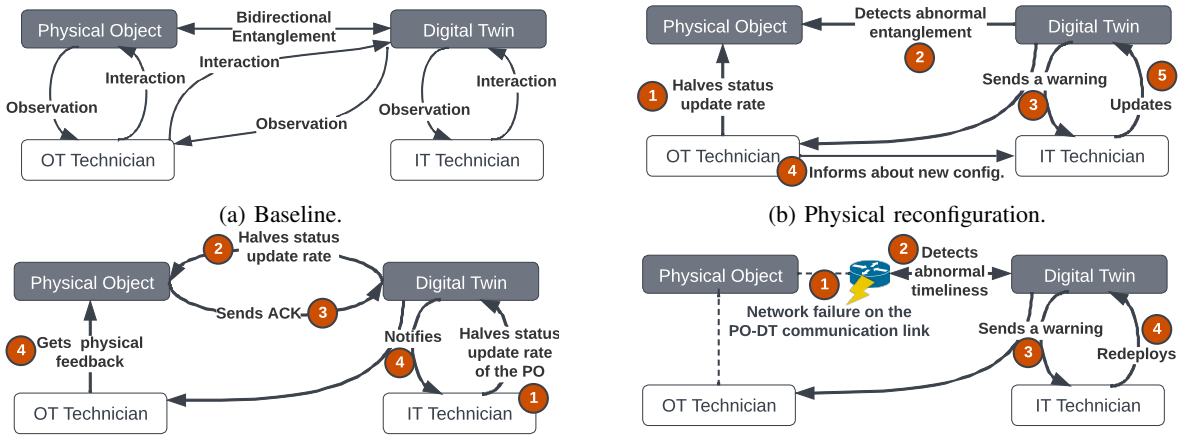
A. Four Illustrative Scenarios for Entanglement

We elaborated on four illustrative scenarios pointing out the main factors that may affect entanglement. Such scenarios illustrate the interactions between IT and OT in IIoT environments and how those interactions affect the entanglement. Specifically, each scenario involves an OT technician working on a PO and an IT technician working on a DT. For instance, the former might be a shop floor operator operating on a production line, while the latter might be a software architect deploying a DT as a microservice through a container orchestration system.

1) Baseline: The baseline scenario describes the interactions between IT and OT in a DT-based industrial environment (see Fig. 2a). Under the baseline scenario, we assume that those interactions do not disrupt the current entanglement characterizing the communication relationship between the PO and the DT. The OT technician interacts with the PO (e.g., a production line) to craft goods and observes the PO status to oversee what is going on. Additionally, the OT technician may request a simulation to the DT and, based on the simulation results, decide on the subsequent commands to send to the PO. The IT technician, instead, only interacts with the DT. Such interactions may relate, for example, to the deployment of the DT. It is worth remarking that different layers of the Purdue model provide different network performances, thus influencing the entanglement. Therefore, the IT technician should plan the DT deployment carefully and re-plan it dynamically according to the network conditions. As the number of DTs grows, so does the complexity of making effective decisions about their deployment. Thus, quantifying the entanglement in a concise yet expressive way becomes even more critical.

2) Physical Reconfiguration: The physical reconfiguration scenario sketches the case where an action of the OT technician on the PO disrupts the entanglement (see Fig. 2b). For instance, the OT technician may change the configuration of the PO, which may result in a different status update rate, say halving the status updates per second. In turn, the DT detects an abnormal entanglement because it still expects double the status updates it is actually receiving. As soon as the DT detects that the entanglement got disrupted, it notifies the OT/IT technicians. At this time, the IT technician can only infer that something is not going as expected. Therefore, the OT technician, whose initial interaction caused the misalignment between the PO and the DT, should notify the IT technician about the change they made to the PO. Then, the IT technician can update the configuration of the DT accordingly, thus bringing the system back to a steady-state phase.

3) Digital Reconfiguration: The digital reconfiguration scenario sketches the case where an action of the IT technician on the DT disrupts the entanglement (see Fig. 2c). At first glance, this scenario might seem symmetrical to the physical reconfiguration one, but it is not. In particular, the IT technician uses the DT to change the configuration of the PO. For example, the IT technician may halve the status update rate of the PO through the DT. As soon as the DT receives the instructions issued by the IT technician, it sets the PO accordingly. At this time, the DT must wait until the PO reports a status update reflecting a status change meeting the request(s) of the DT. Then, the DT can notify the OT/IT technicians back. Note



(a) Baseline.

(b) Physical reconfiguration.

(c) Digital reconfiguration.

(d) Anomaly detection (network failure).

Fig. 2: Illustrative scenarios showing the main interaction patterns between the involved cyber-physical industrial entities.

that the OT technician might have already noticed that the PO changed status because of physical feedback from the PO, e.g., a robot part of the production line where the OT technician is operating changed position.

4) Anomaly Detection: Under the physical and digital reconfiguration scenarios, an intentional action triggered the course of action affecting the entanglement. In contrast, the anomaly detection scenario is about things that could go wrong unpredictably (see Fig. 2d). In particular, this scenario takes into account anomalies striking either the PO (e.g., crash of the production line), the environment (e.g., poor network connectivity between the PO and the DT), or the DT (e.g., hardware fault of the server hosting the DT). Let us assume an outburst of latency upon the communication link that connects the PO and the DT. The DT can detect such an anomaly by looking at the timeliness of the received status updates from the PO. If we instead assume a crash of the PO, the DT can detect that something is not as expected because of a drop in the status update rate. Note that the OT technician may also detect the crash of the PO through physical feedback, e.g., the production line stops working. The recovery phase is started by the actor that detects the anomaly first. In the former case, the DT would start the recovery phase by notifying the IT technician, who might decide, e.g., to redeploy the DT somewhere else or fix the network. In the latter case, the OT technician would start the recovery phase, e.g., by fixing the PO. The outcome of the recovery phase is to bring the system back to a steady-state phase.

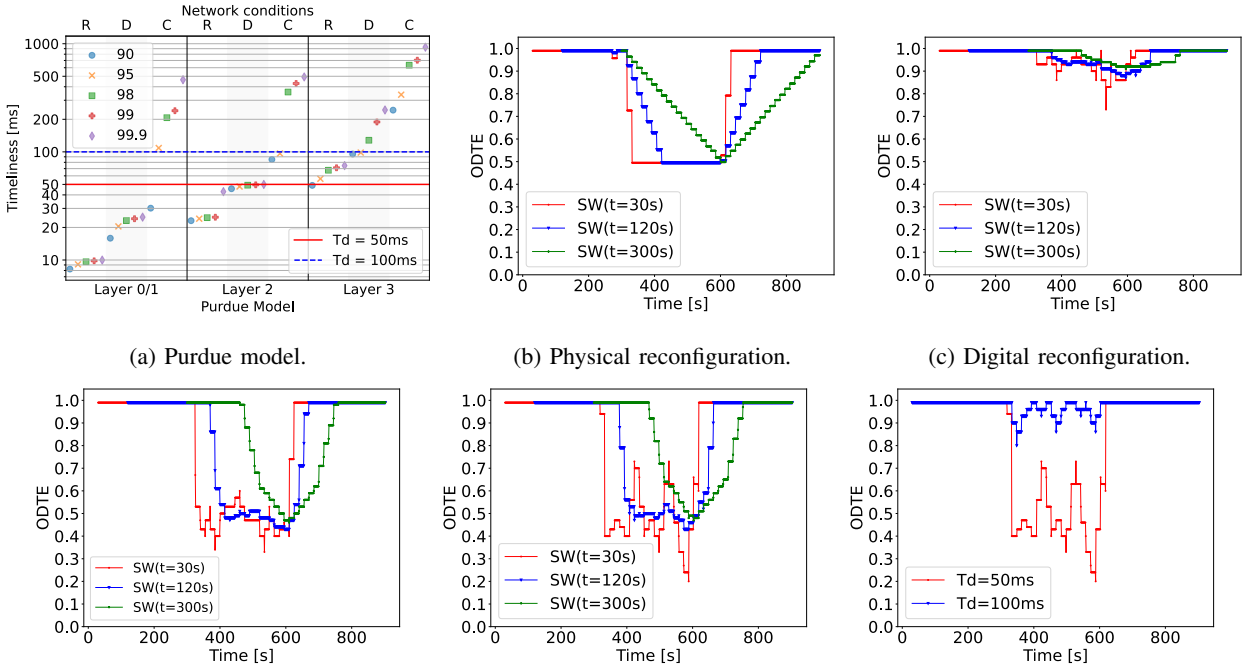
B. Testbed and Experiments

An emerging trend in industrial environments is to adopt cloud-oriented technologies, such as Virtual Machines (VMs) and containers [16]. In particular, a microservice approach makes software development, deployment, and management more effortless. In fact, microservices are also gaining momentum in Industry 4.0 [17], [18] and represent a valuable option for managing the lifecycle of DTs. Accordingly, we relied on Kubernetes to build a representative testbed in line with current industry trends. Specifically, Kubernetes—the de facto industry standard container-orchestration system—is a platform for automating the deployment, scaling, and management of containerized applications. On top of Kubernetes, we deployed Prometheus and Chaos Mesh. The former was used to scrape metrics from DTs (deployed as containers through Kubernetes), store such metrics in a time-series database, and query the database to extrapolate aggregate insights. The latter is a cloud-native chaos engineering platform for Kubernetes that allows injecting a broad spectrum of faults into a target. Through Chaos Mesh, we reproduced a broad spectrum of network conditions under which we tested the effectiveness of the proposed ODTE metric in quantifying the entanglement.

The testbed consisted of a Kubernetes cluster of four nodes, each within its own VM. A single node acted as the master (i.e., the node running the control plane) while the others joined as workers (i.e., regular cluster nodes). Each VM was equipped with 2 vCPU and 2 GB of RAM, each one based on Ubuntu. The testbed was automatically configured in a reproducible manner using Ansible, a well-known configuration management tool that configures a set of target nodes over SSH through a control node. In this way, we made up the Kubernetes cluster (i.e., Kubernetes along with the ancillary software required by Kubernetes to run successfully, such as cri-o and Flannel) as well as deployed Prometheus and Chaos Mesh. The project we developed to configure our testbed is publicly available on GitHub[1].

Through Kubernetes, we deployed a DT, a PO, and a message broker as containerized applications. The DT was implemented by creating a Java DT engine, which supports built-in modularity and a microkernel-oriented structure. Such an implementation relies on a shared multi-thread engine that effectively implements the DT behavior while defining its shadowing procedures, data processing, and interactions with external entities through dedicated interfaces. The PO, which mimics the behavior of an IIoT device, was also implemented

[1]https://github.com/fglmtt/kubemake/



(a) Purdue model. (b) Physical reconfiguration. (c) Digital reconfiguration.

(d) Anomaly detection: Latency. (e) Anomaly detection: Latency & Loss. (f) Comparison.

Fig. 3: Performance of the ODTE metric taking into account different relevant configurations and experimental scenarios.

in Java and, as the DT, exposes proper interfaces to make its behavior configurable at run-time. The message broker was Mosquitto, supporting the MQTT protocol. The PO sent status updates to the DT as MQTT messages, i.e., publishing on a specific topic to which the DT was subscribed. Analogously, the DT issued commands to the PO publishing on another topic to which the PO was subscribed.

To validate the ODTE metric, we first focused on the timeliness factor in the context of an industrial environment based on the Purdue model. In this regard, Table I details the network conditions we injected while performing the experiments. More specifically, we set the regular (R) network conditions potentially affecting each layer of the Purdue model. For example, a DT deployed at layer 0/1 under regular network conditions experienced a one-way latency of $2.5\ ms \pm 2.5\ ms$ (with a correlation between consecutive packets of 25%) and no packet loss. We also defined plausible deteriorated (D) and critical (C) network conditions for each layer. These experiments were conducted over a time window of 5 minutes.

TABLE I: Experiments based on the Purdue model layers

Purdue Model	Network conditions	Latency ± Jitter	Loss
Layer 0/1	Regular (R)	$2.5\ ms \pm 2.5\ ms$	-
	Deteriorated (D)	$5\ ms \pm 5\ ms$	5 %
	Critical (C)	$12.5\ ms \pm 12.5\ ms$	15 %
Layer 2	R	$12.5\ ms \pm 7.5\ ms$	-
	D	$25\ ms \pm 15\ ms$	5 %
	C	$50\ ms \pm 30\ ms$	15 %
Layer 3	R	$35\ ms \pm 15\ ms$	-
	D	$70\ ms \pm 30\ ms$	5 %
	C	$175\ ms \pm 75\ ms$	15 %

We then explored the responsiveness of the ODTE metric under the illustrative scenarios described in Section IV-A. The physical reconfiguration scenario was emulated by halving the status update rate sent by the PO to the DT, i.e., from 1 to 0.5 status updates per second. Then, we instantiated the digital reconfiguration scenario by forcing the DT to calculate 17.5K prime numbers while performing a state transition. Lastly, we produced two instances of the anomaly detection scenario to investigate the responsiveness of the ODTE metric to latency (i.e., $50\ ms \pm 50\ ms$) and the combined effect of latency (as before) and packet loss (i.e., 10%). We performed these experiments in three phases (5 minutes each) over a time window of 15 minutes overall. The first phase resembled the baseline scenario, the second put into action a given scenario, and the third consisted of rolling back what had been injected to reproduce the scenario (thus bringing the system back to the baseline).

C. Results

Fig. 3a shows the results we collected from the experiments focusing on timeliness. The results are expressed in percentiles, i.e., 90th, 95th, 98th, 99th, and 99.9th, and computed based on the metrics Prometheus scraped over a 5-minute window. Note that the left y-axis depicts the timeliness expressed in ms on a logarithmic scale (base 10). The bottom x-axis divides the figure into three vertical macro-sections, each representing a layer of the Purdue model. The top x-axis further divides those macro-sections based on plausible network conditions, i.e., regular, deteriorated, and critical, that might affect each layer of the Purdue model (see Table I). For

example, the yellow cross on the second column means that 95% of the status updates received by the DT had timeliness of at most 20 ms over the observed 5-minute window. The solid red horizontal line distinguishes the layers of the Purdue model that fit a DT with desired timeliness of 50 ms between those that do not. If the target is the 90th percentile, then layer 0/1 represents a suitable option under any network condition. Layer 2 is also a suitable option but only up to the deteriorated network conditions (the 90th percentile almost doubled the desired timeliness while critical network conditions occurred). The dashed blue horizontal line, instead, refers to a DT whose desired timeliness is 100 ms. If we still assume that the target is the 90th percentile, then both layers 0/1 and 2 are suitable deployment options under any network condition.

Fig. 3b, 3c, 3d, and 3e show the responsiveness of the ODTE metric over a 15-minute time window concerning the experiments instantiating the (a) physical reconfiguration scenario, (b) digital reconfiguration scenario, (c) the anomaly detection scenario where the anomaly was an outburst of latency, and (d) where two anomalies were in place simultaneously, i.e., latency and loss. Such figures depict three lines, i.e., red, blue, and green, each plotting the ODTE computed on a different sliding window (see the abbreviation SW in the legend), i.e., 30 s, 2 min, and 5 min, respectively. On the one hand, a shorter sliding window makes the ODTE metric more responsive (the red line reacts faster than the others to the scenario). On the other hand, a shorter sliding window makes the ODTE metric more sensitive to noise. The wide fluctuations depicted in Fig. 3e (see the red line) make evident the impact of a shorter sliding window on the ODTE metric. However, this does not mean that longer sliding windows are always better than shorter ones. The sliding window width choice should reflect the target DT sensitivity to short-lived variations of the entanglement over time. Finally, Fig. 3f shows the ODTE (sliding window of 30 s) concerning two DTs whose desired timeliness was 50 ms (red line) and 100 ms (blue line), and both were performing under the same anomaly detection scenario with latency and loss as described above. Note that it is straightforward to understand if the DT is experiencing a "good" or "bad" entanglement. Also, the application-specific knowledge, i.e., the desired timeliness, is embedded within the ODTE metric. Finally, it is worth pointing out that the DT with 100 ms of desired timeliness was much less influenced (blue line) by the scenario than the one whose desired timeliness was 50 ms (red line).

V. Conclusions and Future Work

In this work, we have presented ODTE—an innovative metric for quantifying the quality of entanglement between DTs and POs. The metric is based on factors, such as timeliness and completeness, for computing QoE without relying on subjective evaluations. More specifically, while timeliness has been represented with a single term, completeness has been split into two terms, namely reliability and availability. ODTE provides technicians with a concise yet expressive value normalized between 0 and 1, not requiring any application-specific knowledge to be correctly understood. Experimental results show that ODTE is responsive at quantifying the quality of entanglement under IIoT scenarios of practical interest. In future work, we plan to generalize the ODTE concept and develop additional metrics that account for other aspects of DTs, such as their interaction with external services and emergent properties of ensembles or hierarchies of DTs.

References

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020.

[2] K. Hribernik, G. Cabri, F. Mandreoli, and G. Mentzas, "Autonomous, context-aware, adaptive digital twins—state of the art and roadmap," *Computers in Industry*, vol. 133, p. 103508, 2021.

[3] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[4] 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.

[5] R. Minerva and N. Crespi, "Digital twins: Properties, software frameworks, and application scenarios," *IT Professional*, vol. 23, no. 1, pp. 51–55, 2021.

[6] A. Ricci, A. Croatti, S. Mariani, S. Montagna, and M. Picone, "Web of digital twins," *ACM Trans. Internet Technol.*, dec 2021, just Accepted.

[7] K. Fizza, A. Banerjee, K. Mitra, P. P. Jayaraman, R. Ranjan, P. Patel, and D. Georgakopoulos, "Qoe in iot: a vision, survey and future directions," *Discover Internet of Things*, vol. 1, no. 1, pp. 1–14, 2021.

[8] V. Sachidananda, A. Khelil, and N. Suri, "Quality of information in wireless sensor networks: A survey," *ICIQ (to appear)*, 2010.

[9] D.-H. Shin, "Conceptualizing and measuring quality of experience of the internet of things: Exploring how quality is perceived by users," *Information & Management*, vol. 54, no. 8, pp. 998–1011, 2017.

[10] Y. Ikeda, S. Kouno, A. Shiozu, and K. Noritake, "A framework of scalable qoe modeling for application explosion in the internet of things," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 425–429.

[11] M. Suryanegara, D. A. Prasetyo, F. Andriyanto, and N. Hayati, "A 5-step framework for measuring the quality of experience (qoe) of internet of things (iot) services," *IEEE Access*, vol. 7, pp. 175 779–175 792, 2019.

[12] W. Robitza, A. Ahmad, P. A. Kara, L. Atzori, M. G. Martini, A. Raake, and L. Sun, "Challenges of future multimedia qoe monitoring for internet service providers," *Multimedia Tools and Applications*, vol. 76, no. 21, pp. 22 243–22 266, 2017.

[13] L. Li, M. Rong, and G. Zhang, "An internet of things qoe evaluation method based on multiple linear regression analysis," in *2015 10th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2015, pp. 925–928.

[14] I. de la Torre Díez, S. G. Alonso, E. M. Cruz, and M. A. Franco, "Measuring qoe of a teleconsultation app in mental health using a pentagram model," *Journal of medical systems*, vol. 43, no. 7, pp. 1–5, 2019.

[15] D. Chen and M. Törngren, "A metrics system for quantifying operational coupling in embedded computer control systems," in *Proceedings of the 4th ACM international conference on Embedded software*, 2004, pp. 184–192.

[16] P. Bellavista, M. Fogli, C. Giannelli, and C. Stefanelli, "Application-aware network traffic management in mec-integrated industrial environments," *Future Internet*, vol. 15, no. 2, p. 42, 2023.

[17] M. Azarmipour, H. Elfaham, C. Gries, T. Kleinert, and U. Epple, "A service-based architecture for the interaction of control and mes systems in industry 4.0 environment," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2020, pp. 217–222.

[18] F. Siqueira and J. G. Davis, "Service computing for industry 4.0: State of the art, challenges, and research opportunities," *ACM Comput. Surv.*, vol. 54, no. 9, oct 2021.