Computers in Industry 152 (2023) 104007

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

OpenTwins: An open-source framework for the development of next-gen compositional digital twins

Julia Robles *, Cristian Martín, Manuel Díaz

ITIS Software, University of Malaga, Arquitecto Francisco Peñalosa, 18, 29071 Málaga, Spain

ARTICLE INFO

Dataset link: https://github.com/ertis-research/OpenTwins/

Keywords:
Digital twin composition
Open-source digital twin framework
Kafka-ML
3D visualizations
Industry 4.0

ABSTRACT

Although digital twins have recently emerged as a clear alternative for reliable asset representations, most of the solutions and tools available for the development of digital twins are tailored to specific environments. Furthermore, achieving complex digital twins often requires the orchestration of technologies and paradigms such as machine learning, the Internet of Things, and 3D visualization, which are rarely seamlessly aligned in open-source solutions. In this paper, we present an open-source framework for the development of compositional digital twins, i.e., advanced digital twins that link individual entities or subsystems to create a higher degree digital twin, allowing knowledge sharing and data relationships. In this open framework, digital twins can be easily developed and orchestrated with 3D-connected visualizations, IoT data streams, and real-time machine-learning predictions. To demonstrate the feasibility of the framework, a use case in the Petrochemical Industry 4.0 has been developed.

1. Introduction

Industry 4.0 has revolutionized production and technological capabilities in a wide range of sectors. The manufacturing industry stands out as a compelling example, showcasing substantial progress in enhancing productivity and fostering sustainable growth (Alsaadi, 2022). This advance is particularly evident in the automotive sector, where these technologies are commonly employed (Papulová et al., 2022). Furthermore, Industry 4.0 has made significant contributions to risk mitigation in the health and safety sector (Arana-Landín et al., 2023a), achieved remarkable efficiency gains in the energy sector (Arana-Landín et al., 2023b), and demonstrated its utility in supply chain management (Marinagi et al., 2023). Actually, in modern times, the assets surrounding us (e.g., Vehicle to Everything-V2X) are increasingly connected, sharing information with each other and with the environment to optimize their performance, prevent dangerous situations, and improve safety. We can highlight several emerging paradigms that have gone hand in hand with the industrial digitalization: the Internet of Things (IoT) (Díaz et al., 2016), which has enabled the real-time monitoring and actuation of multiple physical phenomena; artificial intelligence (AI) and machine learning (ML) (De Silva et al., 2020), which have paved the way to the modeling of the behavior of systems and processes (in some cases unknown) through data-driven approaches; cloud computing (Bello et al., 2021), which has made these computational needs possible; mixed reality (both augmented reality and virtual reality) (Yin et al., 2023), which has enabled more effective and accurate representations of real worlds assets; and simulation of digital twins that can synchronize with the status and future operating of the assets and simulate future outcomes (e.g., a machine failure). It is therefore not paradoxical that, at the same time as the surrounding sources of information (IoT) and the computing power (cloud computing) have increased, analytical techniques have also evolved to improve knowledge of the environment (AI/ML), then the assets are displayed in a 3D and realistic form (mixed reality), and last but not least, the assets and even the learning techniques (physics-informed machine learning (Karniadakis et al., 2021)) are better modeled through simulation.

A digital twin can be defined as a digital accurate and trustworthy representation of a physical asset provided through continuous monitoring, prediction, and optimization for decision-making (Rasheed et al., 2020). For instance, consider a train wheel bearing, which tends to have high maintenance and production costs (Márquez et al., 2020). A digital twin that predicts when bearings need to be repaired and/or replaced, as well as their estimated service life, could highly optimize operating costs and allow for better planning by railway companies. Although digital twins share similarities with cyber–physical systems, such as the integration of assets into the digital world, they go a step further (Nazarenko and Camarinha-Matos, 2020), providing accurate replicas of assets that behave as they would in the physical world.

Digital twins exploit the paradigms that have driven Industry 4.0 to monitor the environment (IoT) and seek continuous optimization/

* Corresponding author.
E-mail addresses: juliarobles@uma.es (J. Robles), cristian@uma.es (C. Martín), mdiaz@uma.es (M. Díaz).

https://doi.org/10.1016/j.compind.2023.104007
Received 3 May 2023; Received in revised form 30 June 2023; Accepted 8 August 2023
Available online 22 August 2023

prediction (simulation, AI/ML) on physical assets. In many processes, in addition to data-driven models, physically-based models provide accurate representations of asset behaviors through mathematical equations. However, these models are often difficult to obtain, so they are complemented by data-driven models such as AI/ML. Digital twins are, therefore, a combination of technologies, models, and paradigms integrated into a common interface for asset control and monitoring. In this context, the visualization (preferably in 3D) of assets also plays a key role, as one of the main functionalities of having reliable digital twins is the simulation of the assets in unknown and extreme situations in order to evaluate their behavior.

Today's digital twins are mostly designed for isolated, bounded solutions such as wind turbine gearbox bearing modeling (Mehlan et al., 2022). However, in the physical world, assets may comprise and require the interconnection of multiple and heterogeneous components, even from different manufacturers. An excellent illustration of this can be seen in automotive line production, where the integration of data from robotic hands, CNC machines, and the overall factory environment is crucial (Mendi, 2022). Moreover, another example is the digital twin of a spacecraft network system. It encompasses both physical components, such as servers and network terminals, and intangible elements, including process protocols and network traffic models (Zhao et al., 2022). In most cases, simulation systems have been used to model these complex behaviors, but these systems lack the flexibility and possibilities offered by digital twins, such as the capabilities for monitoring and prediction.

This orchestration of technologies and paradigms presented by digital twins requires frameworks for their continuous development and integration. Over the last few years, numerous frameworks have emerged that enable the development of digital twins. Architecture and design for digital twin frameworks are identified as areas where further work is needed (Boyes and Watson, 2022). A notable open-source example is Eclipse Ditto,[1] one of the most widely used solutions for multi-domain digital twins. Whereas Eclipse Ditto allows for the virtual abstraction of asset communications as well as fine-grained access control management, substantial extra integration efforts are required to achieve effective digital twins. Of course, we are referring to seamless harmonization with AI/ML techniques, integration with 3D rendering engines, sensor failure detection, and integrated visualization of the ecosystem, among others.

Moreover, digital twins often address individual assets, but these may be part of a global system (e.g., the bearings and the train), where orchestration of digital twins or digital twin composition may be necessary. The approach of compositional digital twins provides a wide range of benefits in terms of interaction, reusability, and distribution. This innovative concept empowers the creation of flexible and modular digital twins, making development and maintenance more straightforward. Furthermore, it enables the parallelization and distribution of digital twins across various devices, resulting in enhanced performance. In addition, the use of compositional digital twins facilitates detailed and comprehensive analysis at both individual and collective levels, leading to a deeper understanding of the system and significantly improving decision-making capabilities. Currently, there is a need for research on the integration and composition of digital twins for the digitalization of complex systems (Human et al., 2023).

In this paper, we present an open-source architecture and framework known as OpenTwins for the continuous development and integration of compositional digital twins, i.e., advanced digital twins that encapsulate complex systems and are composed of a collection of digital twins from individual entities or subsystems. This composition establishes data relationships among them, enabling knowledge sharing and linking their information to form a higher degree digital twin. This framework provides a modular and unified environment where users can define digital twins adapted to their needs. This solution has considered the main needs when developing digital twins (namely IoT monitoring, AI/ML, and 3D visualization), providing a unified interface for their monitoring, management, and continuous optimization. The framework has been designed on a scalable platform that allows for fault tolerance and high availability and opens the door to future extensions. The main contributions to this article are as follows:

1. An open-source framework and interface for the design, development, and continuous integration of effective digital twin composition.
2. Seamless orchestration with AI/ML techniques and data streams for continuous optimization and prediction, such as sensor failure detection.
3. 3D rendering engine integration for the design and visualization of 3D-powered digital twins.
4. Finally, this paper presents the validation of the framework in a manufacturing use case in the petrochemical industry.

The rest of the article is organized as follows. In Section 2, related work is discussed. Section 3 presents the digital twin framework architecture and its components. An evaluation of the framework is performed in Section 5. Lastly, our conclusions and future work are presented in Section 6.

2. Related work

The idea of digital twins was proposed by Professor Michael Grieves of the University of Michigan in 2003 (Grieves, 2016), although at the time it was described simply as a virtual representation capable of being equivalent to a physical product. Despite not having a great initial impact, Grieves was improving this concept until, in 2016, together with John Vickers, the term digital twin was established (Grieves and Vickers, 2017). On the other hand, between 2011 and 2016, NASA (National Aeronautics and Space Administration) put this idea into practice, becoming the first organization to deal with digital twins. Its research was a breakthrough in the field, as it defined the potential of digital twins (Glaessgen and Stargel, 2012) and the main challenges involved (Tuegel, 2012). However, digital twins did not gain recognition until 2016, when the research and advisory firm Gartner included them in their list of the top 10 strategic technology trends (CeArley et al., 2016) for two consecutive years. This marked the beginning of an exponential increase in the number of publications about digital twins.

2.1. Digital twin frameworks

The idea of digital twins has grown considerably, acquiring a major impact on research in recent years. Large companies such as Siemens, Microsoft, and Dassault have developed their own solutions for the construction and use of digital twins. Similarly, work is also carried out in the open-source field. One clear example of a digital twin solution is the large European project IoTwins,[2] which provides a big data platform for the delivery of digital twins in manufacturing. The resulting digital twins from IoTwins address open challenges in a variety of sectors, ranging from a football stadium to wind turbine predictive maintenance. The solutions developed are tailored to a given IoT-edge-cloud infrastructure use case, providing a wide range of services for the definition of digital twins. A modular digital twin framework (Rolle et al., 2021) enables the monitoring, assessment, and 3D visualization of assets. This framework was validated in a real manufacturing scenario, but as demonstrated in the evaluation, scalability can be a challenge in this architecture. A digital twin development guide has been built around the well-known FIWARE ecosystem (Conde et al., 2021). The solution describes a guide to how digital twins can be

[1] https://www.eclipse.org/ditto/

[2] https://www.iotwins.eu/

defined in FIWARE, but a framework adapted for digital twins has not been developed and validated.

Shah et al. (2021) consider the inclusion of open tools for the realization of a digital twin framework. However, this framework lacks flexibility since it is only adapted to computational fluid dynamics systems. Pang et al. (2021) develop a framework that combines both digital twin and digital thread technology, providing improved data management and, consequently, improving the performance and productivity of operational processes. However, this framework does not support the definition and management of a digital twin, requiring the use of an external tool for this purpose.

On the other hand, DTOP-Cristallo (Bonney et al., 2021) is a prototype of an operational platform for digital twins related to the UK DigiTwin project.[3] It is modular, open source, system-independent, and fully written with Flask Python. At the moment, it only focuses on the analysis and simulation aspects of digital twins, providing a set of tools that, with respect to given data, perform the operations requested by the user through a web interface. Kamath et al. (2020) suggest an architecture for the design and analysis of digital twins based on IoT-friendly open-source projects. It includes the most basic aspects of a digital twin: real-time IoT sensor data collection and storage, twin definition and management, and real-time analysis and visualization.

2.2. Digital twin composition

The composition of digital twins is also an important aspect to address in this area. A common scheme for creating a complex digital twin has not been established, but certain ideas have been proposed. Dai et al. (2021), in the context of the machining process, propose an ontology-based modeling method for manufactured parts. Liu et al. (2021) propose a multi-scale knowledge modeling method for digital twins focusing on product quality, particularly at the macro and micro levels. Jia et al. (2022) present a digital twin modeling method capable of dealing with multiscale, multi-scenario, or multidimensional digital twins. These complex digital twins are divided into simple digital twins. This method favors the scalability and reusability of components, although these divisions may reduce the accuracy of the digital twin.

Borth et al. (2019) study the issues that can arise when composing digital twins. They emphasize the impact of the architecture, recommending that it should be modular and have reflection mechanisms at sensitive points. In addition, they point out the importance of defining dedicated processes and operations to ensure the proper performance of the infrastructure. To this end, they advise using twin maintenance processes with modular computational models and combining automated local upgrades with expert-driven global upgrades. The snitch digital twin concept is presented in Calvo-Bascones et al. (2023). Snitch's digital twin consists of a behavior model based on digital twin data composition for the detection of anomalies. Although this approach is intended for anomaly detection in multi-agent systems, the use of basic techniques like quantiles and slope with the composition of digital twins is a feasible solution for this problem. Reiche et al. (2021) propose a structured network of digital twins controlled by a single point of truth. This point corresponds to the digital twin of the system (DTS), which is responsible for managing its subordinate twin network. This structure improves data access, allows the linking of digital twins from different sources, and promotes uniform and centralized communication, although the computational effort for twin representation will increase as the network expands.

Atkinson and Kühne (2021) discuss the advantages of multilevel modeling of digital twins. The orthogonal classification architecture (OCA) of the multilevel model overcomes the problem of accidental complexity that arises with the traditional two-tier cascading approach. This architecture favors scalability by allowing the creation of an unlimited number of domain classification levels. Each level will have its own features and constraints, which facilitate the creation of twins and ensure compliance with architectural principles. Our architecture also shares some of the components identified in Human et al. (2023) for the definition of digital twins, but our framework goes further by considering aspects necessary for the development of today's digital twins, such as integration with machine learning streaming and 3D visualization.

3. Open-source architecture for the design and development of 3D iot-AI-powered compositional digital twins

A microservice architecture has been designed with the aim of increasing the modularity of the platform by making functionality independent and the system scalable and reusable, allowing modules to be added, replaced, and connected without affecting the whole system.

Fig. 1 shows the current architecture of the platform. The main core of the platform is the Eclipse Ditto framework, which has a set of essential functionalities that act as a base for the development of digital twins. A microservice architecture has been built around Eclipse Ditto, composed of open-source tools that mostly belong to external projects. Certain services have also been developed specifically for the platform, mainly with the aim of connecting certain tools or adding functionalities that were not already covered. The different modules are connected mainly by means of an API or a protocol designed to handle real-time data, such as Apache Kafka, AMQP, or MQTT. As observed in the existing literature, there are other frameworks that share some of the functionalities proposed in this architecture, such as the work in Kamath et al. (2020), where monitoring and visualization of digital twins are proposed with open-source components. However, to the best of our knowledge, there is a lack of frameworks considering open source components to support the end-to-end digital twin design, i.e., asset monitoring, integration with AI/ML systems for learning/prediction with streaming data, user-friend visualization with 3D capabilities, and, as discussed, the digital twin composition. We have therefore considered open-source solutions and developed the necessary components to achieve this functionality in a unified framework, OpenTwins.

On the other hand, to facilitate the management, portability, and execution of the platform, a container-based structure has been used. Each service is packaged in a Docker container, and all the containers are managed through Kubernetes, a container orchestrator.

Eclipse Ditto provides an entity for the twin definition, access control, state storage, and connection support with IoT protocols, which can modify the state of the twins and share their events externally. The union of this framework with all the services shown in Fig. 1 provides the platform with the functionalities required in most digital twin platforms: reception of data from IoT devices, definition and update of twin composition, real-time storage of the states through time series, and user-friendly data visualization. In addition, two functionalities have been added that constitute a real advance in terms of open-source platforms for digital twins: interactive 3D visualization of the real-time and historical state of the twin and real-time data stream inference by machine learning.

Regarding these functionalities, the architecture can be divided into blocks related to the compositional and essential functionality of the platform, the prediction of data with Machine Learning, and the 3D visualization of the twin.

In GitHub,[4] the description of the system, the installation and connection manual of the architecture, the necessary documentation for its use, and the redirections to all the services and plugins developed (which, in addition to the code produced, also have their own installation manual and documentation of use) can be found.

[3] https://digitwin.ac.uk/

[4] https://github.com/ertis-research/OpenTwins/

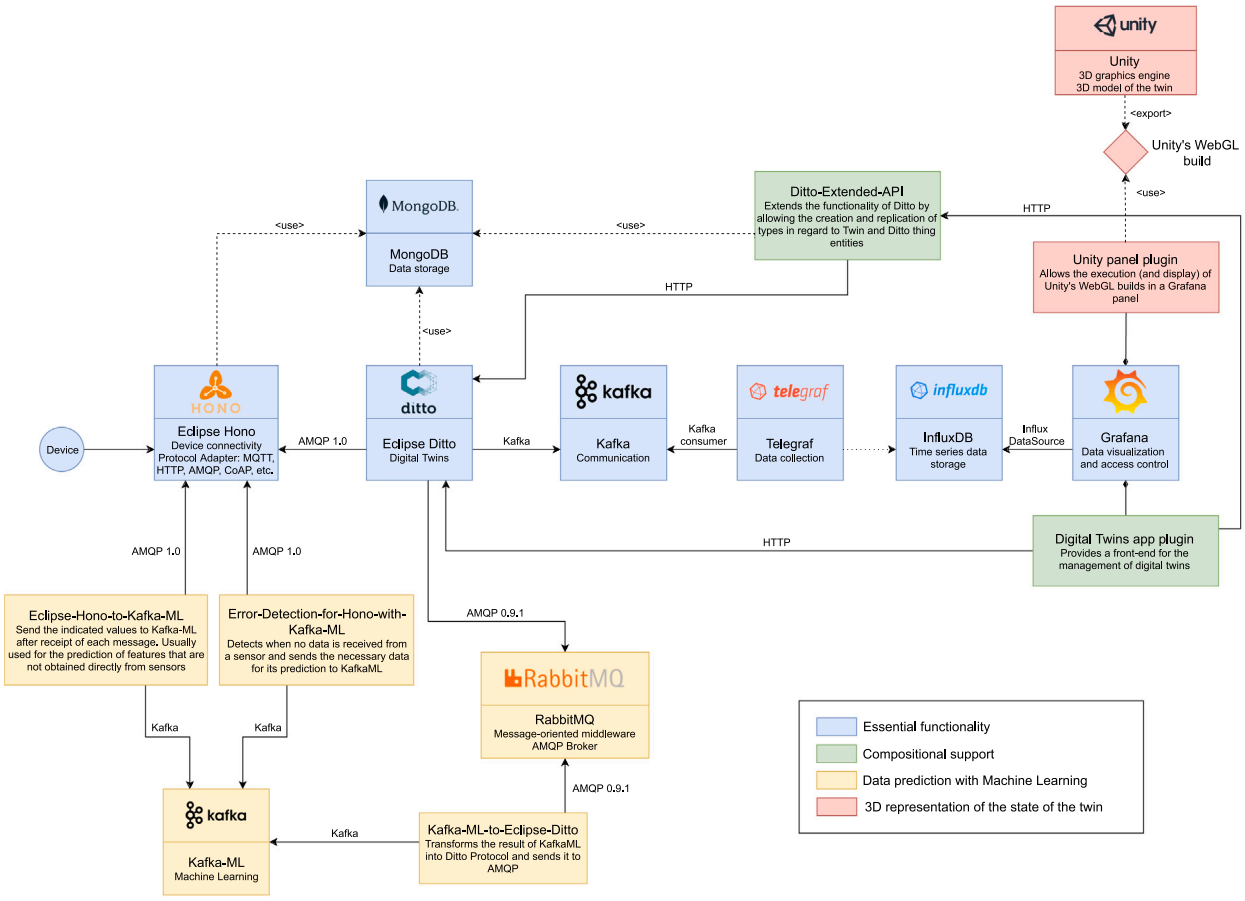


Fig. 1. An overview of the 3D-IoT-AI-powered digital twin architecture. In blue are the essential and compositional functions, in red the 3D representation, and in yellow the data prediction with ML. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Basic functionality

The blue components in the architecture (Fig. 1) represent the part of the architecture corresponding to this compositional and basic functionality. This part is mainly composed of open-source projects, but also includes two elements that have been developed to complete the desired functionality.

The objective of this block was to obtain a platform on which the digital twin of any element and its composition could be defined. For this purpose, the following functionalities can be considered basic for any digital twin:

1. Digital twin scheme definition.
2. Connection with IoT devices and collection of their information.
3. Storage of digital twin data in real-time series.
4. User-friendly visualization of data.

The main element of architecture is Eclipse Ditto, an open-source framework for building digital twins. Eclipse Ditto does not provide any system to obtain the information sent by the devices, so Eclipse Hono[5] was considered for this purpose. Eclipse Hono is a platform that provides several interfaces for connecting many IoT devices, unifying them into a single AMQP 1.0 endpoint where the information received can be read and commands can be sent to trigger actions on any IoT device. It can receive information via common IoT protocols, such as MQTT or AMQP, and custom adapters. This is the recommended tool to work with Eclipse Ditto, and, thanks to the Eclipse cloud2edge package,[6] the integration of the two is very convenient.

Another feature that Eclipse Ditto lacks is the storage of the twin state at different time instants. To solve this, InfluxDB,[7] has been chosen as a time-series database. This is a well-known database with a large community and is ideal for processing sensor data. To collect the data, we have considered Telegraf[8] a plugin-driven server that provides support for numerous data sources and with which we can configure the data ingestion into the database. As there is no possibility for Telegraf to collect the data directly from Eclipse Ditto because neither technology implements a broker for the protocols it supports, we need an intermediate element that allows its connection. Apache Kafka,[9] one of the best-known streaming and processing platforms for real-time data, is a suitable alternative since Telegraf has a Kafka-consumer[10] plugin available, and Eclipse Ditto offers the option to publish events in a Kafka topic.

Finally, Grafana[11] has been chosen to act as the front-end, i.e., the user interface for end-users. This technology provides support for metrics visualization from the most popular databases, including InfluxDB. It allows making queries in the language defined by the chosen data source and presenting the results in different types of interactive panels. These panels are part of dashboards, which can be modified to the

[5] https://www.eclipse.org/hono/

[6] https://www.eclipse.org/packages/packages/cloud2edge/

[7] https://www.influxdata.com/products/influxdb-overview/

[8] https://www.influxdata.com/time-series-platform/telegraf/

[9] https://kafka.apache.org/

[10] https://www.influxdata.com/integration/kafka-telegraf-integration/

[11] https://grafana.com/

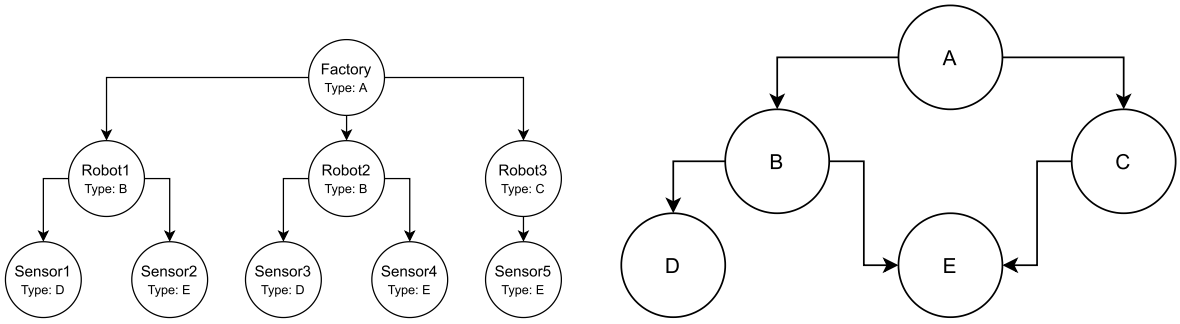


Fig. 2. Example composition of digital twins and their types.

user's liking. It also includes an access control system through roles. Another of its strong points, and one of the most important for the project, is that it allows the creation of personalized panels and the inclusion of any type of functionality by means of plugins, providing the libraries and documentation necessary for this.

3.2. Compositionality support

Eclipse Ditto offers the Ditto Thing entity, which always belongs to a namespace and is basically composed of an identifier and a series of attributes and features. The attributes correspond to the static part of the entity, whereas the features correspond to the dynamic part. In this way, the framework provides full freedom to define how models should be designed and how the tool should be used. After studying different options, such as considering a Ditto Thing as a complete twin and including each sensor as a feature of it, a design decision was made to assign a Ditto Thing to a single entity or sensor and to create parent–child hierarchies between these entities. This prevents the creation of oversized Ditto Thing entities, which could decrease the platform's performance, and facilitates the management, reuse, and communication of digital twins. In addition, it is also convenient to include the creation and management of twin types. This streamlines the tedious task of creating multiple twins that, while corresponding to different physical devices, have exactly the same features. For instance, in the case of a system comprising 10 components with identical features, it becomes feasible to define a single type of digital twin for that particular component and create 10 instances of twins based on it. Fig. 1 represents in green the part related to these functionalities.

A digital twin can be considered both an entity that receives information from a single device and one that is composed of other entities, which can also be understood as twins. A twin could then be represented as a tree, where each leaf is the representation of a single sensor. Thus, a factory that has three robots, and each one has particular sensors, could be contemplated as shown in Fig. 2(a).

Regarding twin types in the factory example, creating a type can facilitate the generation of additional robots of the same model, as is the case for robots 1 and 2. Additionally, the sensors that make up the robot type should be defined as their own types. For example, even though sensor 2 (and therefore sensor 4) may belong to a different robot type, they can still have the same model as sensor 5. Thus, the twin types form a cycle-free graph, which could be represented as shown in Fig. 2(b).

These guidelines have been implemented in a service called *Ditto-Extended-API*[12], which can be considered a layer above Eclipse Ditto and which provides an API that replaces the one offered by this technology. In addition to verifying that the specified constraints are satisfied, this service adds all the respective functionality to the twin types, allowing both their management and the creation of twins from them. It has been developed with the Node.js framework, an open-source framework that functions at runtime and delivers excellent performance in developing server-side tools, so it is a suitable option for implementing this service.

Moreover, an application plugin[13] has been developed for Grafana that adds a graphical interface for managing twins and types using the Ditto-Extended-API service, thus unifying all the functionalities of the platform in the same application. ReactJS is used as the implementation language, an open-source and component-based JavaScript framework whose main purpose is to create user interfaces for single-page applications. On the other hand, Grafana provides a series of libraries that allow improved integration with data sources and greater visual consistency with the rest of the tool.

3.3. Prediction model integration with Kafka-ML

This milestone aims to achieve the integration of the platform with machine learning algorithms. This might be useful for digital twins to predict their next state or a situation of failure, as, for instance, the values that a sensor should return in case no real data is received from it, either because it has been switched off or because it has had some kind of failure.

The part of the architecture that is in charge of achieving this objective corresponds to the yellow components in Fig. 1. In this part, the main component is Kafka-ML (Martín et al., 2022), which will be in charge of machine learning life cycle management and complement this architecture. In addition, in order to integrate it with Eclipse Hono and Eclipse Ditto and fulfill the required functionality, three specific services have been developed: *Eclipse-Hono-to-Kafka-ML*, *Error-Detection-for-Hono-with-Kafka-ML*, and *Kafka-ML-to-Eclipse-Ditto*.

Kafka-ML is an open-source framework[14] that manages the life cycle of ML/AI applications in production environments through continuous data streams. Unlike traditional frameworks that work on datasets or static files, Kafka-ML allows both training and inference with continuous data streams, enabling users to have fine control of the ingestion data in popular ML frameworks such as TensorFlow and PyTorch.

One of the main purposes in the area of machine learning focused on digital twins is the prediction of the future states of the twin, which may be of considerable utility if control or improvement of the element that the twin represents is sought. It also aims to predict certain features or values of the twin that cannot be measured or obtained directly or in any accurate way. As mentioned above, these models will be deployed in Kafka-ML, so for their connection with Eclipse Hono, a new specific *Eclipse-Hono-to-Kafka-ML*[15] service has been developed.

[12] https://github.com/ertis-research/extended-api-for-Eclipse-Ditto/

[13] https://github.com/ertis-research/digital-twins-plugin-for-Grafana/

[14] https://github.com/ertis-research/kafka-ml/

[15] https://github.com/ertis-research/eclipse-hono-to-kafka-ml/

This tool constantly reads from the Eclipse Hono endpoints that contain devices whose data must be sent to these deployed Kafka-ML models. When a message arrives, the service processes the data to comply with the required format and automatically sends it to the respective Kafka-ML-trained model.

Another of the strong points of this part is its ability to detect when a sensor is not sending its data and to act in consequence. To this end, service *Error-Detection-for-Hono-with-Kafka-ML*[16] has been created, which basically reads the information received by the specified Hono endpoints and checks that the devices that are part of it are sending their data in accordance with their periodicity. In case one of them does not send the data when it is due, the service will send the last values received from that sensor to a Kafka-ML-trained model with historical data. Kafka-ML will predict the next state of the system to avoid a service interruption due to the sensor failure until the sensor is available again. To identify instances of data deficiencies, one timer is set for each device to be controlled based on the time interval between the two most recent messages received. In the event of the timer's expiration, Kafka-ML is activated, and if a new message arrives, the timer is reset.

On the other direction, the data generated by Kafka-ML has to be consumed by Eclipse Ditto. Initially, the idea was to take advantage of the payload mapping functionality provided by Ditto to Kafka-ML by creating a source connection to each of the Kafka-ML output topics where models send the predictions and mapping the information received so that it could be supported by Eclipse Ditto. This was not viable since, at the time of the development of this platform, Eclipse Ditto had not implemented the connection with Apache Kafka acting as a data source. That is why it was decided to build an intermediate service[17] that would read the information from Kafka, map it to Eclipse Ditto Protocol, and publish it to a message broker that Eclipse Ditto could connect to, in this case, RabbitMQ,[18] which uses AMQP 0.9.1.

These services are thread-based and are managed through an API. A MongoDB database is used to maintain the persistence of the information provided by users, which will indicate the output topics or devices that need to be controlled. The Flask framework, whose programming language is Python, has been used, as it provides great facilities for creating APIs and, unlike Node.js, it allows multithreading.

To sum up, the integration of the digital twin platform with Kafka-ML allows the provision of intelligence to digital twins through deep/machine learning models and streaming data, and even to model the data disruption behavior of different sensors with the generation of simulated data.

3.4. 3D representation of the state of the twin

An important aspect of digital twin platforms is the representation of the data. It is common to find 3D representations of digital twins that considerably improve the visualization and comprehension of their information and state. Achieving the integration of this 3D representation is the main objective of this milestone.

The architecture that has been designed for 3D visualization corresponds to the red components in Fig. 1 and basically consists of the creation of a panel plugin for Grafana that allows the display of a 3D model developed with Unity[19] with which it will be possible to interact in both directions.

Unity is a software that mainly focuses on video game development, although it can also be used in other contexts. It is one of the most popular graphics engines with the largest community. Although it is not an open-source tool, it can be used free of charge for personal use or for low-budget projects. This technology allows assigning a certain behavior to 3D objects through the use of scripts and interacting both with the user and with other elements in the environment. Unity allows the project to be built in several formats, including a specific one for web rendering called WebGL. A panel plugin[20] has been developed for Grafana to load models in this format. This type of plugin allows the creation of a custom panel that usually represents or uses data series, which will depend on the query and the data source introduced by the user in the panel configuration.

The plugin panel also enables bidirectional interaction with both Grafana and the user. This means that any actions taken by the user within Grafana will be reflected in the corresponding 3D model, and conversely, any manipulations performed on the representation may impact the information displayed in other panels of the dashboard.

In order to support these functionalities, the plugin's implementation relies on the React Unity WebGL library, which facilitates the integration of Unity compilations exported to WebGL format into any React-based application and enables bidirectional communication between them.

4. Use case: Virtual analyzer in petrochemical industry

The OpenTwins platform has been validated through a Petrochemical Industry 4.0 use case. The objective of the use case is to define a virtual analyzer that is able to predict the freezing point of one of CEPSA end products (lubricant) based on the operating conditions and the properties of the feedstock. This process is carried out in the San Roque (Spain) Energy Park of CEPSA, one of the largest refineries in Spain. The freezing point is an important parameter that, due to its characteristics, has to be measured offline in a laboratory. Based on the monitoring of different operation conditions, such as the filter operational conditions, the aim is to predict in Kafka-ML the state of the freezing point in real-time for better control of the process. This continuous prediction, together with the status of the monitored sensors, has been modeled in a digital twin within our framework. Digital twins have also been studied before in the petrochemical industry. For instance, a digital twin for production control purposes of a catalytic cracking unit in the petrochemical industry is proposed (Min et al., 2019). In this work, we go further by considering the modeling, prediction, and 3D visualization of a process in this industry.

The company has provided us with real-time (through the MQTT protocol) and historical data (to train ML models) from the necessary sensors. These sensors will be considered twins in their own right, and together they will compose the main twin being sought. From them, the different digital twin types have been identified by grouping the sensors that are identical in operation and description, and a Ditto Thing scheme has been defined for each type and twin. For sending real-time data, a tenant was created in Eclipse Hono and credentialed devices have been added for each of the available sensors. At this point, the twins should be receiving the data correctly, and their state over time will be stored in InfluxDB, so Grafana dashboards can be created according to requirements.

The freezing point predictive model that has been developed as the target of the use case has been deployed in Kafka-ML. For its data input, a script periodically makes a call to Eclipse Ditto to collect the current state of the twins that represent the sensors required by the model. The output data of this model is collected by the Kafka-ML-to-Eclipse-Ditto service in order to update the freezing point feature contained in the digital twin.

For the 3D representation of the twin's state, a model was created in Unity that contains enough elements to represent each of the sensors that are part of the machine. In this model, each asset has been renamed

[16] https://github.com/ertis-research/error-detection-for-eclipse-hono-with-kafka-ml/

[17] https://github.com/ertis-research/kafka-ml-to-eclipse-ditto/

[18] https://www.rabbitmq.com/

[19] https://unity.com/

[20] https://github.com/ertis-research/unity-plugin-for-grafana/

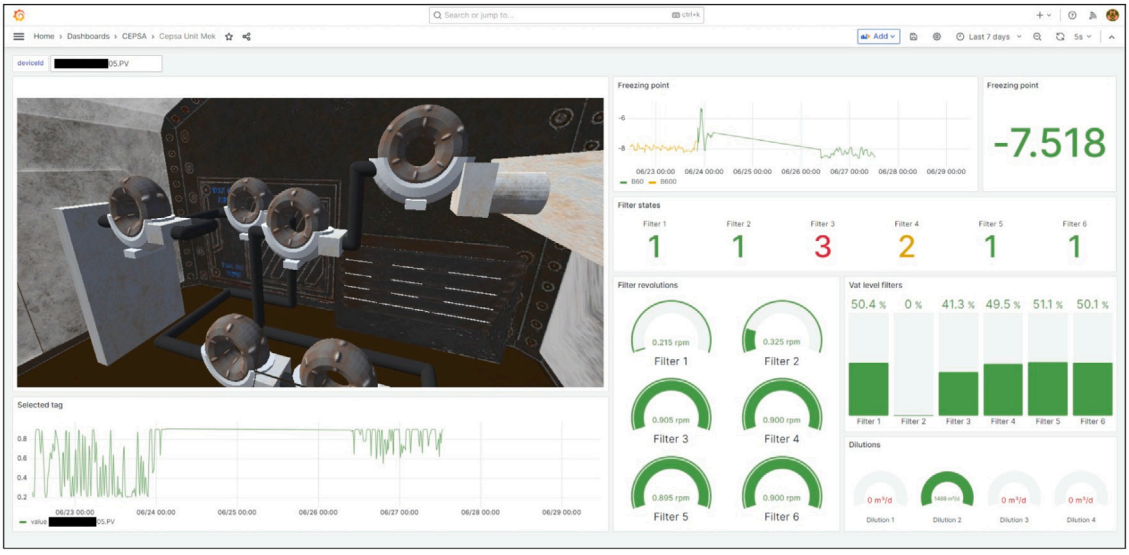


Fig. 3. Final result of the digital twin for the use case.

with the ID of the sensor it represents, and the necessary code has been implemented for the movement of the camera and the selection of assets by clicking on them, as well as the script necessary for the model integration in Grafana. Its WebGL export has been added to the Grafana public folder, and the Unity panel has been included in one of the boards.

Fig. 3 shows the final result of the digital twin-developed 3D representation for the described use case. As a result, an easily adaptable and extendable twin of the industrial process has been obtained, with an eye-pleasing representation of its real-time status using different types of graphics as well as a 3D model that, on receiving the data from the sensors, provides the possibility of displaying data of interest on the machine, such as its real movement, and allowing any type of interaction with the user. Likewise, the platform allows easy querying of the current state of the twin via the Eclipse Ditto API and its state over time, using any of the query options provided by InfluxDB. Furthermore, in terms of predictive concerns, machine learning models are easily integrated with the twins, with the resulting value being considered as any other feature of the twin.

The framework has successfully demonstrated its application in the petrochemical industry, serving as a practical use case. However, it should be noted that this solution is designed to be adaptable across various domains. This versatility is achieved through the flexible nature of its individual components.

When deploying a digital twin in a different domain, it becomes necessary to define the attributes and features of both the main twin and its sub-twins through the platform interface. Furthermore, if real-time data needs to be collected, the corresponding devices can be defined and monitored through Eclipse Hono, using the specific protocol employed by the IoT devices. Similarly, other models can be defined in Kafka-ML and integrated into OpenTwins to achieve the desired predictive behavior. As for the three-dimensional representation, a new 3D model of the asset has to be designed and compiled with Unity. However, once compiled, it can be effortlessly incorporated into the platform using the enabled plugin. In summary, creating a digital twin for a specific use case only requires establishing the corresponding schematics, models, and connections, without the need for any modifications to the architecture components.

5. Evaluation

The system, with a special interest in the freezing point prediction component, is currently being gradually implemented in the factory. Operators have provided generally positive feedback, indicating that the use of the digital twin is assisting them in enhancing process control, suggesting a promising start.

On the other hand, given the wide range of components used in the framework in question, it was decided to conduct a comprehensive evaluation with the primary objective of guaranteeing the optimal performance of the entire system. This evaluation aims to ensure that all elements involved function efficiently and maintain an acceptable level of coherence and consistency in terms of quality.

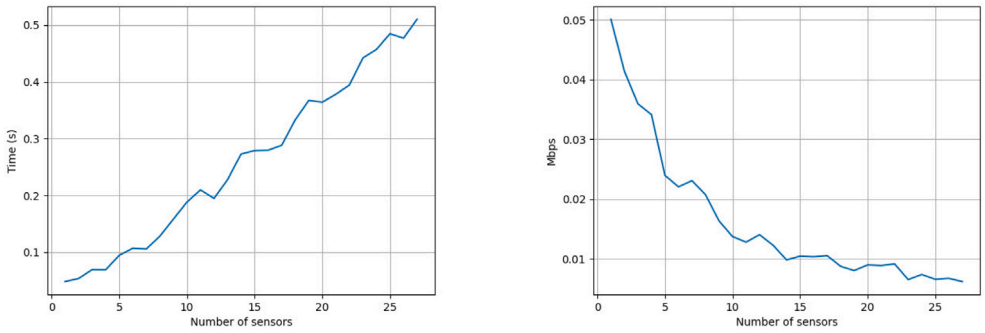
Therefore, a series of tests will be carried out to validate the performance and scalability through a latency analysis, as well as the availability of the platform. These tests will use the construction outlined in the previous section, taking into account the different flows of which the architecture is composed. Specifically, the tests will focus on the essential functionality (Test 1) and machine learning (Test 2) flows, as the 3D visualization relies on a single component. Moreover, evaluating the essential functionality accurately will require considering scenarios where multiple sensors receive simultaneous updates or where a single sensor receives multiple updates concurrently. Furthermore, it is important to consider whether the platform is capable of recovering quickly in the event of outages and to assess whether this would lead to data loss (Test 3).

Test 1 will probe the latency and throughput of the digital twin's core functionalities with respect to the number of connected sensors and the number of clients/connections used. Meanwhile, Test 2 will calculate the same properties but will focus on the predictive flow of the platform considering the number of simultaneous clients. Finally, Test 3 will check the fault tolerance of the platform by obtaining the average recovery time of the services and determining whether the failure of those services results in data loss.

5.1. Experimental setup

Hardware configuration. All the experiments were performed on a five-node Kubernetes cluster in our private cloud infrastructure in VMware vCloud. Each node has 4 virtual CPUs in 2 sockets and 16 GB of RAM. The client that sent the information and from where the results were measured was a PC with 64 GB of RAM, 1 CPU, and 10 cores.

Software configuration. Each one of the five nodes runs Kubernetes v1.19.3 and Docker 19.03.13 on top of Ubuntu 16.04.7 LTS. A Kubernetes master was deployed in one node, whereas the remaining four are Kubernetes workers. The PC with the client runs the Ubuntu server.



(a) Latency (b) Throughput

Fig. 4. Latency and throughput of test 1 for different number of sensors.

5.2. Test 1 - Essential functionality flow

The test will evaluate the latency and throughput from the moment data are sent to Eclipse Hono via MQTT to the moment they are stored in InfluxDB within the dataflow used in the Petrochemical Industry use case described. Here we have two test cases: different numbers of sensors receiving data simultaneously, and different numbers of clients/connections sending data simultaneously. The size of the data sent has not been taken into account because all the messages share the same format, so their size hardly varies.

5.2.1. Related to the number of sensors

For this test, historical data has been sent in the format specified by the company, creating an MQTT connection for each sensor and increasing the number of sensors to which data is sent simultaneously using threads. The time at which each input is stored in InfluxDB is then queried, and times are compared to obtain latency and throughput. The result of each test by sensor number is the average of ten repetitions of the test. Since the company provided us with data from 27 of their sensors, this is the maximum number of sensors to be called simultaneously in this test. Fig. 4 shows the results obtained.

It is clear that, as the number of affected sensors increases, latency increases almost linearly and throughput decreases exponentially. This result is in accordance with normality, and it should also be taken into account that in a real-life scenario, not all the sensors of the platform will receive the data simultaneously or with the same frequency, so we can determine that the platform performs well as the number of sensors (here twins) affected increases.

5.2.2. Related to the number of clients

This case is similar to the previous one, with the difference that only data updates will be sent to a single sensor by a different number of simulated clients using threads. The values sent in each message are always unique, as each client increments by 0.01 a global value, starting at 0. Likewise, the result of each test per number of clients is the average of ten repetitions of this test. Fig. 5 shows the results, which have also been limited to 27 clients to facilitate the understanding of the graph and the comparison with the one explained above.

As can be seen, the latency exceeds one second of delay after 20 simultaneous clients. The throughput, on the other hand, maintains a fair decrease. These results do not represent a problem since the most common stage is that a twin or device receives data from a single data source, except for simulated or predicted features, where the number of clients could usually increase by one or two. This behavior is also normal as sending all messages to a single twin can overload it. Therefore, we can conclude that the system reacts correctly to a coherent increase in the number of clients.

5.3. Test 2 - Machine learning prediction flow

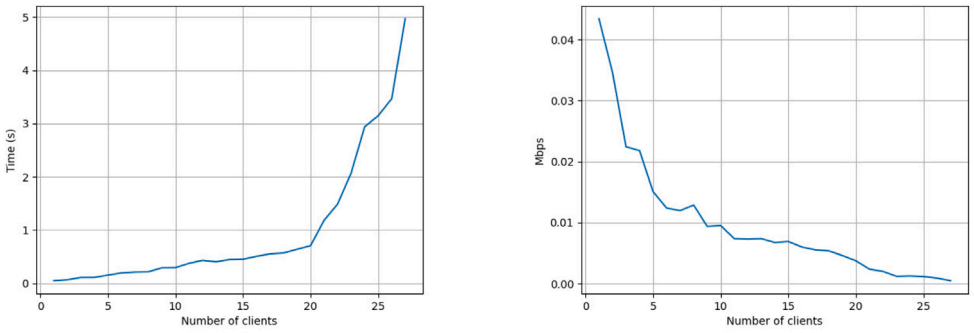
In this test, the latency and throughput will be calculated for the machine learning integration part of the architecture. Since there is only one prediction model running in the use case for the freezing point prediction, the test will always affect the same Eclipse Ditto twin, and what we will vary is the number of clients sending input data to the model. To relate each client to the outcome of the model, we have used data inputs with known results, avoiding their repetition during each of the tests. After the execution of the test with a certain number of clients, InfluxDB is consulted for the time in which each piece of data has been stored, which will allow later comparison. The result of each test is an average of 10 executions. Fig. 6 shows the results achieved in this test.

The results are very similar to those shown in the first test with respect to the number of clients. Latency grows until it exceeds one second delay with 17 simultaneous clients. Similarly, throughput decreases substantially. These are acceptable results compared to the real-time data flow, as the difference between them is minor, and it must be taken into account that a per-client prediction is made during the process. Moreover, in this case, as in the previous one, it is usual for a single client to initiate the flow, so the limitation of simultaneous clients would not be a problem. Therefore, it can be concluded that the platform exhibits appropriate responsiveness when confronted with an increased volume of concurrent clients during the machine learning-based prediction flow.

5.4. Test 3 - Error tolerance

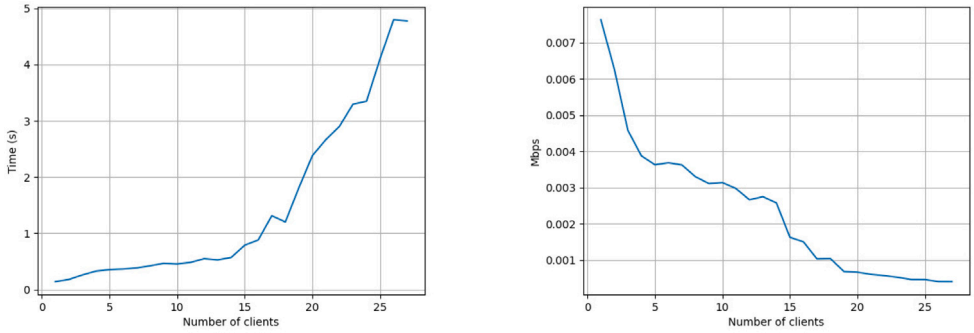
This last test is based on checking the platform's tolerance to errors. To do this, a script has been created that sends a message to Eclipse Hono via MQTT every half a second. While this is running, the pod corresponding to the service to be tested is manually deleted in Kubernetes. If a message cannot be sent, it will be resent as many times as necessary, maintaining the initial time of the first sending. When the end of the test is indicated, the data received will be extracted from InfluxDB for comparison. The maximum time difference of the test shall be considered the recovery time for that pod. Each test result is the average of five test runs. In Fig. 7, we can see the results. The selected pods follow the sequence from sending data to Eclipse Hono until the moment the corresponding twin is updated in Eclipse Ditto and is shown in the graph in that order. The combination of two or more pods has not been included, as the result coincides with the maximum recovery time between them.

As a result of this test, it can be observed that the vast majority of the services exhibit significantly reduced recovery times, which are fully acceptable for this kind of system. The exception is the Eclipse Hono MQTT adapter, which has a higher recovery time, probably owing to its dependence on the creation, configuration, and connection of an



(a) Latency (b) Throughput

Fig. 5. Latency and throughput of test 1 for different numbers of clients.



(a) Latency (b) Throughput

Fig. 6. Latency and throughput of test 2.

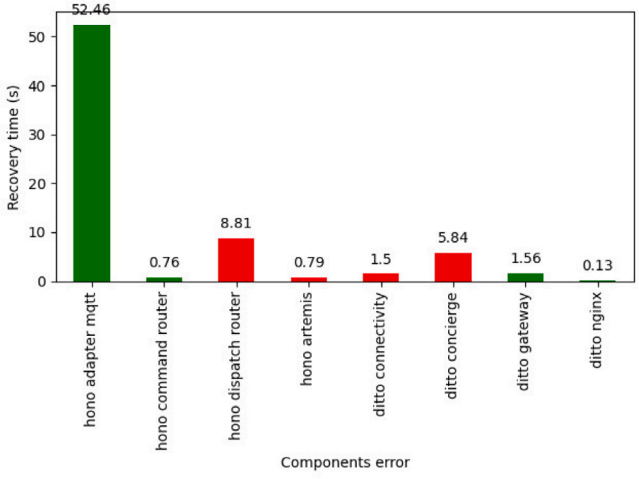


Fig. 7. Recovery time for each service. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

MQTT broker. On the other hand, by looking at the raw data obtained in the test, we can determine which services involve a loss of data during recovery. Those that do involve some data loss are marked in red in Fig. 7, while those that do not are shown in green. This is not a big problem, because of the short recovery time of the services, although its resolution will be studied for future work. Based on the above, we can conclude that the platform has good behavior with respect to errors, as it is capable of recovering from them without causing a great loss of data or requiring human intervention.

6. Conclusions and future work

Digital twins are emerging as a valuable resource in order to have a better understanding of and anticipate possible situations that assets may face in the physical world. Although a multitude of platforms have been defined in the literature for the development of digital twins, they are mainly focused on specific vertical contexts, and a significant evolution is needed to achieve effective and compositional digital twins. The provision of preferably-open platforms for the development of digital twins and their composition is one of the current needs in this area. In this paper, we propose an open-source platform for the development of compositional digital twins that can be adapted to multiple contexts. We refer to digital twins that can be seamlessly integrated with predictive models through the open platform Kafka-ML, interactive digital twins provided thanks to Unity's support for 3D representations, and a unified interface for real-time visualization and management of IoT monitoring data for simple and complex models. To demonstrate the feasibility of the platform, we have defined a digital twin of an industrial process in the petrochemical industry. This process monitors and predicts the freezing point of lubricant generation. Through the digital twin, plant users can visualize the status of the plant and its components (e.g., filters) directly on the 3D representation, as well as the freezing point prediction in real time.

As future work for the platform, we envisage supporting the FMI (Functional Mock-up Interface) standard in order to be able to simulate complex processes that follow this standard and integrate them with the 3D representations of the platform. Furthermore, we intend to generate hybrid twins that can take advantage of the low latency opportunities offered by edge/fog systems and demonstrate the viability of the platform in other contexts. We are currently working in the agricultural sector on the development of digital twin solutions for irrigation pivots and soil, with the overarching goal of creating a complex digital twin of the crops.

CRediT authorship contribution statement

Julia Robles: Software, Conceptualization, First manuscript draft and review. **Cristian Martín:** Supervision, Conceptualization, Manuscript draft and review. **Manuel Díaz:** Supervision, Conceptualization, Manuscript review, Funding.

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

Source code is open source[21] but data used for the use case is confidential.

Acknowledgments

This work is funded by the Spanish projects TSI-063000-2021-116 ('5G+TACTILE_2: Digital vertical twins for B5G/6G networks'), TED2021-130167B-C33 ('SIERRA: Application of Digital Twins to more sustainable irrigated farms'), PID2022-141705OB-C21 ('DiTaS: A framework for agnostic compositional and cognitive digital twin services'), and MIG-20221022 ('GEDERA: Intelligent Flexible Energy Demand Management in Coupled Hybrid Networks'). Funding for open access charge: Universidad de Malaga/CBUA.

References

Alsaadi, N., 2022. Modeling and analysis of industry 4.0 adoption challenges in the manufacturing industry. Processes 10 (10), 2150.

Arana-Landín, G., Laskurain-Iturbe, I., Iturrate, M., Landeta-Manzano, B., 2023a. Assessing the influence of industry 4.0 technologies on occupational health and safety. Heliyon 9 (3).

Arana-Landín, G., Uriarte-Gallastegi, N., Landeta-Manzano, B., Laskurain-Iturbe, I., 2023b. The contribution of lean management—Industry 4.0 technologies to improving energy efficiency. Energies 16 (5), 2124.

Atkinson, C., Kühne, T., 2021. Taming the complexity of digital twins. IEEE Softw. 39 (2), 27–32.

Bello, S.A., Oyedele, L.O., Akinade, O.O., Bilal, M., Delgado, J.M.D., Akanbi, L.A., Ajayi, A.O., Owolabi, H.A., 2021. Cloud computing in construction industry: Use cases, benefits and challenges. Autom. Constr. 122, 103441.

Bonney, M.S., de Angelis, M., Wagg, D., Dal Borgo, M., 2021. Digital twin operational platform for connectivity and accessibility using flask python. In: International ACM/IEEE Conference on Model-Driven Engineering Languages and Systems (MODELS), OCT 10-15. pp. 239–243.

Borth, M., Verriet, J., Muller, G., 2019. 2019 14th Annual Conference System of Systems Engineering (SoSE). IEEE, pp. 164–169.

Boyes, H., Watson, T., 2022. Digital twins: An analysis framework and open issues. Comput. Ind. 143, 103763.

Calvo-Bascones, P., Voisin, A., Do, P., Sanz-Bobi, M.A., 2023. A collaborative network of digital twins for anomaly detection applications of complex systems. Snitch Digital Twin concept. Comput. Ind. 144, 103767.

CeArley, D., Burke, B., Searle, S., et al., 2016. Gartner's top 10 strategic technology trends for 2017.

Conde, J., Munoz-Arcentales, A., Alonso, A., Lopez-Pernas, S., Salvachua, J., 2021. Modeling digital twin data and architecture: A building guide with FIWARE as enabling technology. IEEE Internet Comput..

Dai, S., Zhao, G., Yu, Y., Zheng, P., Bao, Q., Wang, W., 2021. Ontology-based information modeling method for digital twin creation of as-fabricated machining parts. Robot. Comput.-Integr. Manuf. 72, 102173.

De Silva, D., Sierla, S., Alahakoon, D., Osipov, E., Yu, X., Vyatkin, V., 2020. Toward intelligent industrial informatics: A review of current developments and future directions of artificial intelligence in industrial applications. IEEE Ind. Electron. Mag. 14 (2), 57–72.

Díaz, M., Martín, C., Rubio, B., 2016. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. J. Netw. Comput. Appl. 67, 99–117.

Glaessgen, E., Stargel, D., 2012. The digital twin paradigm for future NASA and U.S. air force vehicles.

Grieves, M.W., 2016. Origins of the digital twin concept.

Grieves, M.W., Vickers, J.H., 2017. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems.

Human, C., Basson, A., Kruger, K., 2023. A design framework for a system of digital twins and services. Comput. Ind. 144, 103796.

Jia, W., Wang, W., Zhang, Z., 2022. From simple digital twin to complex digital twin Part I: A novel modeling method for multi-scale and multi-scenario digital twin. Adv. Eng. Inform. 53.

Kamath, V., Morgan, J., Ali, M.I., 2020. Industrial IoT and Digital Twins for a Smart Factory : An open source toolkit for application design and benchmarking. In: 2020 Global Internet of Things Summit (GIoTS). pp. 1–6.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3 (6), 422–440.

Liu, S., Lu, Y., Li, J., Song, D., Sun, X., Bao, J., 2021. Multi-scale evolution mechanism and knowledge construction of a digital twin mimic model. Robot. Comput.-Integr. Manuf. 71, 102123.

Marinagi, C., Reklitis, P., Trivellas, P., Sakas, D., 2023. The impact of industry 4.0 technologies on key performance indicators for a resilient supply chain 4.0. Sustainability 15 (6), 5185.

Márquez, A.C., de la Fuente Carmona, A., Marcos, J.A., Navarro, J., 2020. Designing cbm plans, based on predictive analytics and big data tools, for train wheel bearings. Comput. Ind. 122, 103292.

Martín, C., Langendoerfer, P., Zarrin, P.S., Díaz, M., Rubio, B., 2022. Kafka-ML: connecting the data stream with ML/AI frameworks. Future Gener. Comput. Syst. 126, 15–33.

Mehlan, F.C., Nejad, A.R., Gao, Z., 2022. Digital twin based virtual sensor for online fatigue damage monitoring in offshore wind turbine drivetrains. J. Offshore Mech. Arct. Eng. 144 (6), 060901.

Mendi, A.F., 2022. A digital twin case study on automotive production line. Sensors 22 (18), 6963.

Min, Q., Lu, Y., Liu, Z., Su, C., Wang, B., 2019. Machine learning based digital twin framework for production optimization in petrochemical industry. Int. J. Inf. Manage. 49, 502–519.

Nazarenko, A.A., Camarinha-Matos, L.M., 2020. The role of digital twins in collaborative cyber-physical systems. In: Doctoral Conference on Computing, Electrical and Industrial Systems. Springer, pp. 191–205.

Pang, T., Restrepo, J., Cheng, C.-T., Yasin, A., Lim, H., Miletic, M., 2021. Developing a digital twin and digital thread framework for an 'industry 4.0' shipyard. Appl. Sci. 11, 1097.

Papulová, Z., Gažová, A., Šufliarskỳ, L., 2022. Implementation of automation technologies of industry 4.0 in automotive manufacturing companies. Procedia Comput. Sci. 200, 1488–1497.

Rasheed, A., San, O., Kvamsdal, T., 2020. Digital twin: Values, challenges and enablers from a modeling perspective. IEEE Access 8, 21980–22012.

Reiche, L.-T., Gundlach, C.S., Mewes, G.F., Fay, A., 2021. The digital twin of a system: A structure for networks of digital twins. In: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation(ETFA), SEP 07-10, 2021.

Rolle, R.P., Martucci, V.d.O., Godoy, E.P., 2021. Modular framework for digital twins: Development and performance analysis. J. Control Autom. Electr. Syst. 32 (6), 1485–1497.

Shah, K., Prabhakar, T., Sarweshkumar, C., Abhishek, S., et al., 2021. Construction of a digital twin framework using free and open-source software programs. IEEE Internet Comput..

Tuegel, E., 2012. The airframe digital twin: Some challenges to realization. In: Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference.

Yin, Y., Zheng, P., Li, C., Wang, L., 2023. A state-of-the-art survey on Augmented Reality-assisted Digital Twin for futuristic human-centric industry transformation. Robot. Comput.-Integr. Manuf. 81, 102515.

Zhao, Z., Wang, F., Gao, Y., Li, T., Ying, L., Liang, W., Li, Y., Dong, Z., 2022. Design of a digital twin for spacecraft network system. In: 2022 IEEE 5th International Conference on Electronics and Communication Engineering. ICECE, IEEE, pp. 46–50.

[21] https://github.com/ertis-research/OpenTwins/