This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

Theme Article: Digitalization of Smart Ecosystems

Smart Ecosystems and Digital Twins: an architectural perspective and a FIWARE-based solution

Franca Rocco di Torrepadula, Alessandra Somma, Alessandra De Benedictis, Nicola Mazzocca,
Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, 80125, IT

Abstract—Smart ecosystems today span several sectors, including smart manufacturing, energy management, smart cities, smart healthcare, precision farming, and others. Digital Twins (DTs) are emerging as a powerful technology that can act as the digital backbone of a smart ecosystem, providing the data, insights, and control capabilities needed for real-time optimization and collaboration among involved entities. In this paper, we focus on the architectural integration of DTs within smart ecosystems, addressing interoperability challenges and aligning with identified DT requirements. We introduce DT-enabled Ecosystem (DTE) architecture structured in: i) Logical View, outlining key DTE entities and relations; ii) Technological View, mapping entities and requirements onto FIWARE components; iii) Development View, providing low-level description in a smart mobility case study. This multi-view approach facilitates the deployment and scalability of DTs in diverse smart ecosystem scenarios.

Smart ecosystems connect physical devices and digital tools to improve processes across various sectors (e.g., smart cities) [1]. Their evolutionary dynamics and reliance on heterogeneous data sources challenge their technical sustainability, i.e., their ability to maintain the quality of service over a prolonged time.

In recent years, the **Digital Twin** (DT) paradigm has become a cornerstone of innovation, gaining traction in a wide range of contexts, including smart ecosystems. Digital Twins act as virtual replicas of physical assets, continuously updated with real-time data and augmented with advanced control and predictive functionalities [2], [3]. Consequently, DTs serve as a valuable source of data for smart ecosystems while actively contributing to their operation. They enable real-time information exchange with other physical systems and various DTs—both those representing different aspects of the same physical system and DTs from different subsystems [1].

Notwithstanding the growing interest in DTs recently shown by academia and industry, their engineering remains an intricate endeavor. This complexity arises from the absence of a standardized definition and a unified, domain-independent architecture. Despite the numerous proposals for DT architectures (e.g., [4], [5]), these are still often presented in a one-dimensional manner, leading to predominantly domain-specific solutions [6]. Moreover, the interdependence between software components and heterogeneous devices introduces issues related to data sharing and interoperability, not addressed by current solutions. These challenges impede the application of DTs to smart ecosystems and even raise questions about the necessity of doing so.

At current state, there are a few popular platforms for the implementation of smart ecosystems and DTs, including Microsoft Azure[1], Amazon Web Service (AWS) Twin Maker[2], Eclipse Ditto[3] and FIWARE[4]. Although Azure and AWS offer comprehensive suites of services for building and deploying applications and benefit from

---

F. Rocco di Torrepadula and A. Somma equally contributed to the methodology, research, writing and illustration. A. De Benedictis contributed to the methodology and writing. N. Mazzocca supervised the research.

XXXX-XXX © 2024 IEEE
Digital Object Identifier 10.1109/XXX.0000.0000000

[1] https://azure.microsoft.com/

[2] https://aws.amazon.com/it/iot-twinmaker/

[3] https://eclipse.dev/ditto/

[4] https://www.fiware.org/

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

strong communities and support, they are subject to vendor lock-in and may be very expensive for large deployments. If a lightweight and easy-to-use platform for smaller-scale implementations is needed, Eclipse Ditto may be a good option [7]. For large-scale deployments, FIWARE stands out as a formidable open-source tool to cope with data management, interoperability, and scalability, thanks to the numerous software components offered to help collect, process, and visualize data, and their adherence to open standards that ensures seamless integration among different systems and devices.

In light of the above considerations, this paper aims to facilitate the integration of DT technology within smart ecosystems by defining the architecture of a **DT-enabled Ecosystem** (**DTE**). The proposed DTE architecture is structured across three *Architectural Views*, inspired by Krutchen's View Model [8]: the Logical View, the Development View, and an additional Technological View that bridges high-level concepts with low-level architectural details. More specifically, this work:

- Analyzes existing DT and DTE literature to derive a comprehensive set of *DT requirements*.
- Introduces the *DTE Logical View* outlining the main entities involved and their interrelationships, ensuring alignment with the identified DT requirements.
- Presents the *DTE Technological View*, mapping DT requirements and DTE entities' responsibilities to appropriate FIWARE components to facilitate practical implementation.
- Defines the *DTE Development View* providing a detailed low-level architectural description applied to a real-world urban mobility case study.

DT Architectures

DT architectures have advanced in recent years, with studies exploring design approaches [6]. Early research, such as [9], focused on defining design patterns for DT-based systems. Building on this, [10] proposed reusable patterns for creating adaptive, autonomous DTs based on microservices, while [2] outlined a roadmap for developing autonomous DTs in digital factories. A complete snapshot of DT architectures highlighted the prevalent use of layered and service-oriented patterns [6].

Agent-based approaches to DT design have also been explored. For example, [4] presented a multi-agent DT system framework, while [1] discussed a one-dimensional architectural view for collaborative ecosystems. In application-specific research, [3] proposed a DT architecture for healthcare, while [11] adopted FIWARE technology to develop a platform-dependent DT architecture for urban environments. Efforts toward DT standardization are noteworthy: *i)* [12] examined the alignment of manufacturing DT architectures with ISO 23247 standard[5], stressing the need for standardized reference architectures. [5] proposed a domain-driven design of DT architectures. Interested readers can find a list of additional recommended papers in our GitHub repository [6].

DT Requirements

Based on our DT and DTE literature analysis (e.g., [1], [2], [7], [9], [10]), we identified a set of DT requirements describing what a DT must do to achieve its functionalities. Table 1 includes an ID, name, and brief description of each requirement. It also outlines how each requirement maps to the Logical View's entities and to FIWARE components, further discussed in the following two sections.

A DT must accurately represent its PT (`R1`). This requires *(i)* bidirectional data exchange to establish the closed-loop communication between the PT and the DT (`R4`), and *(ii)* bidirectional mirroring to ensure that the states of the PT and DT remain continuously aligned (`R2`).

The DT must store the states and event history (`R5`) in a context-aware manner (`R3`), retaining only information relevant to DT operational context. Multiple DTs can be grouped into a composite entity (`R6`) to form the *Digital Twin Aggregate*, useful for complex real-world systems (e.g., entire cities) that cannot be modeled as monolithic DTs.

DTs should be capable of predicting future behaviors of the PT, which aids in proactive planning and issue anticipation (`R7`), and adapting to issues affecting both the PT and the DT (`R8`). Interoperability must be addressed on three levels: i) system interoperability (`R9`), to enable communication, within the same DT, among different physical systems via suitable interfaces; ii) data interoperability (`R10`), to facilitate data exchange across various systems and formats; iii) platform interoperability (`R11`), to allow the extension of DTs with value-added services provided by or built in collaboration with third-party entities.

---

[5] https://www.iso.org/standard/75066.html

[6] https://github.com/alessandrasomma28/UrbanMobilityDigitalTwin/tree/main/FurtherReadings

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

TABLE 1. Mapping of DT Requirements to Logical View entities and FIWARE GEs.

ID	Name	Description	Responsible Entity	FIWARE support	FIWARE GE Component
R1	*Representativeness*	DT must mimic the status and features of the PT.	`DTModel`, `DTModelManager`	×	×
R2	*Bidirectional Mirroring*	Every status change or event in the PT is reflected in the DT, and modifications in the DT are mirrored in the PT.	`DTModelManager`, `Monitoring`, `Control`, `DTFeedback`, `Broker`, `DigitalAdaptor`, `PhysicalAdaptor`	P	Context Broker, IoT Agent
R3	*Contextualization*	DT must manage the PT state and event history considering only PT information relevant to the current context.	`DTModelManager`, `Broker`, `DataModel`	P	Context Broker, Smart Data Models
R4	*Bidirectional Communication*	DT gets data from the PT, and the PT receives DT information after processing.	`Broker`, `PhysicalAdaptor`, `DigitalAdaptor`	✓	Context Broker, IoT Agent, Kurento RT
R5	*Memorization*	DT must store all PT relevant data with context and maintain the PT current state.	`DTDataStorage`, `DataStorage`, `Broker`	P	Context Broker, Cygnus, Draco, STH Comet, QuantumLeap
R6	*Composability*	DT must support grouping multiple DTs into a composed entity, providing views on both the aggregated and individual DTs.	`DTModelManger`, `DTService`, `Broker`	P	Context Broker, Kong
R7	*Predictability*	DT must simulate PT behavior and interaction to determine the outcomes in a likely future.	`DTService`, `Prediction`	P	Perseo, Wirecloud
R8	*Adaptability*	DT must provide facilities to address and manage damages or issues affecting both the PT and itself.	`DTService`, `Prediction`, `Optimization`, `Control`, `DTFeedback`, `DigitalAdaptor`	P	IoT Agent
R9	*System Interoperability*	DT system must enable communication and interaction among different PTs within the same DT.	`Broker`	P	Context Broker
R10	*Data Interoperability*	DT must ensure data exchange across various systems and data formats.	`Broker`, `DataModel`	✓	Context Broker, Smart Data Models
R11	*Platform Interoperability*	DT system must support the integration of third-party value-added services.	`Broker`	✓	Context Broker

DTE Architecture: Logical View

The *DTE Logical View* provides a structural perspective of the DT-enabled Ecosystem. This View models DTE entities and their relations in the *UML class diagram* shown in Figure 1.

In the Logical View, DTE is divided into three subsystems: i) the *Physical Twin Subsystem* (PTS), encompassing the real-world physical assets, ii) the *Digital Twin Subsystem* (DTS), forming the digital core of the DTE; iii) the *Connector Subsystem* (CTS), facilitating data exchange between the DT and its PT, and among multiple DTs.

The PTS includes the actual `Physical System` and one or more `DataProvider` and `DataReceiver`. The DTS includes the DT ecosystem key entities to satisfy DT requirements listed in Table 1:

1) `DTModel` is the virtual representation of the physical system, modeled according to the *Composite pattern* [13] as it may be built as a composition of multiple models. A `Model` can be a `StructuralModel` (e.g., geometrical representation) or a `BehavioralModel`, describing PT dynamic aspects. The explicit multiplicities highlight that at least one behavioral model is *mandatory* in a DT, enabling the *simulation* execution to mimic PT states (`R1`).
2) `DTService` represents the services offered by the DTE, leveraging and combining the outcomes of the `DTModel`. As regards the `Operation` entity, being a smart system, the DT can perform four basic operations, namely `Monitoring` and `Control` (needed for requirement `R2`), and `Optimization` and `Prediction` (for `R7` and `R8`).
3) `DTModelManager` handles the available DT models, managing their creation and exposing their methods to one or more services, according to the *Façade pattern* [13] (requirement `R1, R2` and `R6`).
4) `DTDataStorage` is the repository for preserving (`R5`) DT-related data, e.g., historical data, domain-expert knowledge, outcomes from services or models execution, etc. Further details about the repository are tied to the kind of data to store, and hence to the investigated domain.
5) `DTFeedback`, provides targeted feedback to the physical system's `DataReceiver`, either as executable actions or alerts to be further handled by

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

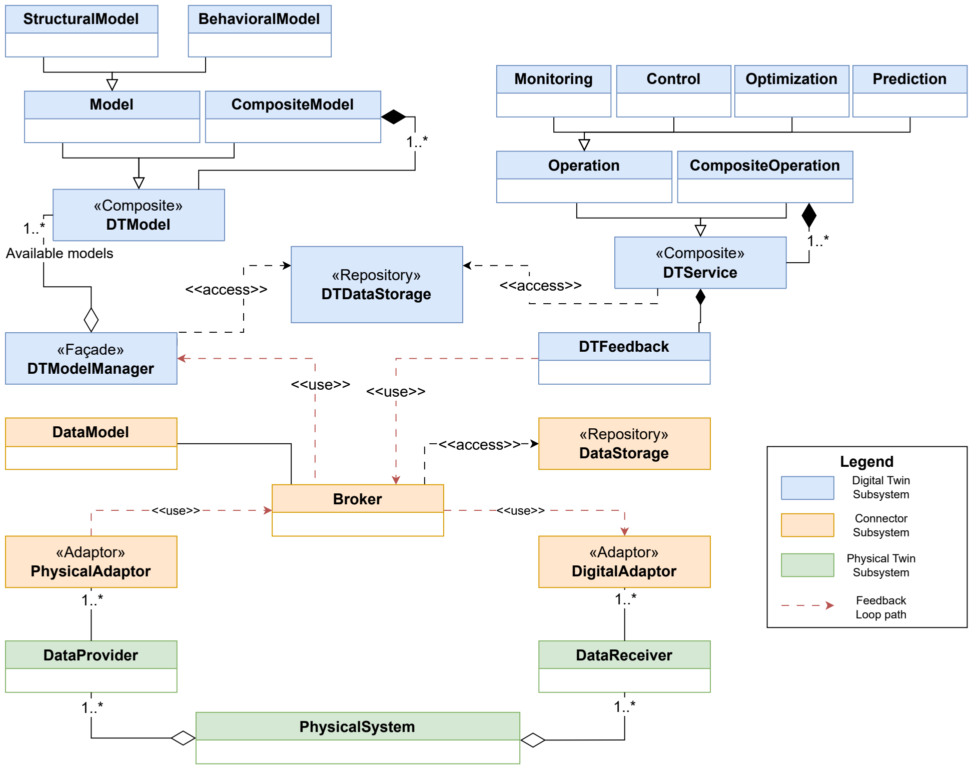

FIGURE 1. UML Class Diagram of DTE Logical View.

human operators [9] (needed for `R2`, and `R8`).

To realize the seamless bidirectional communication characterizing a proper DT, crucial for `R2` and `R4`, the `Connector Subsystem` has been included. Here, according to the *Adapter pattern* [13], the `PhysicalAdaptor` acts as a wrapper meant to standardize different, and even unforeseen, data sources into the common interface required by the `Broker`. Similarly, the `DigitalAdaptor` homogenizes the interfaces of different data receivers.

The `Broker` entity has been devised to cope with data interoperability (`R10`) and data sharing issues typical of a DTE [14], offering a standardized interface and associated information/`DataModel` (entity, attribute, metadata) for data sharing. Entities' data and metadata are stored in another repository included in the connector itself, namely `DataStorage`. As `DTDataStorage`, also `DataStorage` contributes to requirement `R5`, but it is meant to store the data exchanged through the `Broker`. Therefore, it aligns with the specific `DataModel` used. Conversely, `DTDataStorage` stores all data generated by `DTModels`, not necessarily conform to the `DataModel`. Let us note that the Connector Subsystem enables both the cooperation and interaction among different models and services, in the case of multiple virtual replicas of the same physical system, and the communication among DTs of different systems (requirements `R9` and `R11`).

The red loop in Fig. 1 highlights the bidirectional communication enabled by the `Connector Subsystem`. Particularly, data collected and homogenized by the `PhysicalAdaptor` can be accessed through the `Broker` by the `DTModelManager`. The results of the models are exploited to realize different services and feedback, which are provided by the `DTFeedback` to the `DigitalAdaptor`, again through the `Broker`. This enables `DataReceivers` to trigger novel actions on the physical entities.

DTE Architecture: Technological View powered by FIWARE

The alignment between DTE entities and the FIWARE GEs plays an important role in translating the Logical into the Development View. Table 1 illustrates the extent to which FIWARE components assist in implementing DTE entities fulfilling DT requirements. The symbols ×, P, ✓indicate whether FIWARE provides no, partial, or

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

complete technological support, respectively.

As shown in Table 1, FIWARE GEs do not fully meet the *Representativeness* (`R1`). This requirement demands modeling and simulation tools to execute domain-specific behavioral models, which go beyond FIWARE's capabilities. Partial support arises when FIWARE components cover only some functionalities required for specific DTE entities. For instance, the *Bidirectional Reflection* (`R2`) necessitates a continuous data flow between Digital and Physical Twins that can be facilitated by FIWARE Context Broker (CB), implementing the `Broker` class, and the IoT Agents (IoTA), realizing the `DigitalAdaptor` and `PhysicalAdaptor` functionalities.

The FIWARE CB, a mandatory element in any "*Powered by FIWARE*" ecosystem, is a publish-subscribe broker managing context information through the Next Generation Service Interfaces (NGSI) protocol. The four different types of brokers (Orion[7], Orion-LD[8], Scorpio[9] and Stellio[10]) vary on the NGSI version they implement. IoT Agents enable devices to communicate with the CB using their native protocols (e.g., HTTP or MQTT) and receive information from the broker. This bidirectional data exchange fully supports the *Bidirectional Synchronization* (`R4`).

Memorization (`R5`) and *Contextualization* (`R3`) involve storing state and event data relevant to the current context. Although the FIWARE CB handles only the current state, historical data can be stored using FIWARE GEs such as QuantumLeap[11], Cygnus, and others that interface with time-series databases. To ensure data interoperability (`R10`), the FIWARE CB and Smart Data Models (SDMs) standardize how data are structured in various contexts, facilitating compatibility and data sharing across systems.

For *Predictability* (`R7`) and *Adaptability* (`R8`), FIWARE offers partial support. IoT Agents aid in the implementation of adaptors, while components such as Perseo[12] and WireCloud[13] enable event processing and visualization. Regarding *Composability* (`R6`), FIWARE facilitates the grouping of multiple DTs through its Context Broker and Kong[14] offering extended API management capabilities and achieving interoperability at system (`R9`) and platform (`R11`) levels.

DTE Architecture: Development View of an Urban Mobility DT

The proposed DTE logical architecture has been instantiated in a small-scale *urban mobility* case study, considering the Intelligent Transportation System (ITS) of a major Italian city [15]. Implementation details can be found in our GitHub repository[15]. The ITS includes buses equipped with Automatic Passenger Counting (APC) systems and GPS devices to collect data on passenger flow and location. These data are processed and integrated into the DT behavioral model, which is executed in the *Simulator of Urban MObility*[16] (SUMO) to monitor the physical system. FIWARE GEs handle data to ensure interoperability. Additionally, data are utilized by Machine Learning (ML) models to forecast future behaviors. These insights are then applied to DT models to simulate alternative scenarios and optimize the real-world bus network, enhancing service quality and efficiency through a continuous feedback loop. As the data and infrastructure belong to a private company, we will limit the discussion to general information relevant to our narrative without delving into specifics.

Figure 2 depicts the UML Component&Connector diagram of the Development View, outlining DTE components, responsible for implementing entities of the Logical View. The *Physical Twin Subsystem* is represented by the bus vehicles equipped with APC and GPS devices. Since we do not have access to the actual running infrastructure, to simulate the real-time data collection from the PT, we realized a Python script (indicated as `simulated physical devices` component in the diagram) that mimics data transmission and reception, starting from a real dataset collected on the field.

The *Connector Subsystem* is mainly made of FIWARE tools and is structured as follows:

- `FIWARE Orion-LD` is the Context Broker exposing `Context Op.` API accessed by `DataModel Generator` and `FIWARE Mintaka`.
- `DataModel Generator` is responsible for context modeling leveraging on Smart Cities SDMs, using the Python library named ***ngsildclient***[17]. More in detail, our scenario consists of {*bus*, *bustrip*, *busroute*, *busstop*, *APC*, *GPS*} entities modeled by inheriting and extending SDMs, e.g., "UrbanMobility".

---

[7] https://fiware-orion.readthedocs.io/
[8] https://github.com/FIWARE/context.Orion-LD
[9] https://scorpio.readthedocs.io/en/latest/
[10] https://stellio.readthedocs.io/en/latest/
[11] https://quantumleap.readthedocs.io/
[12] https://fiware-perseo-fe.readthedocs.io/
[13] https://wirecloud.readthedocs.io/
[14] https://github.com/FIWARE/kong-plugins-fiware
[15] https://github.com/alessandrasomma28/UrbanMobilityDigitalTwin
[16] https://eclipse.dev/sumo/
[17] https://ngsildclient.readthedocs.io/

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

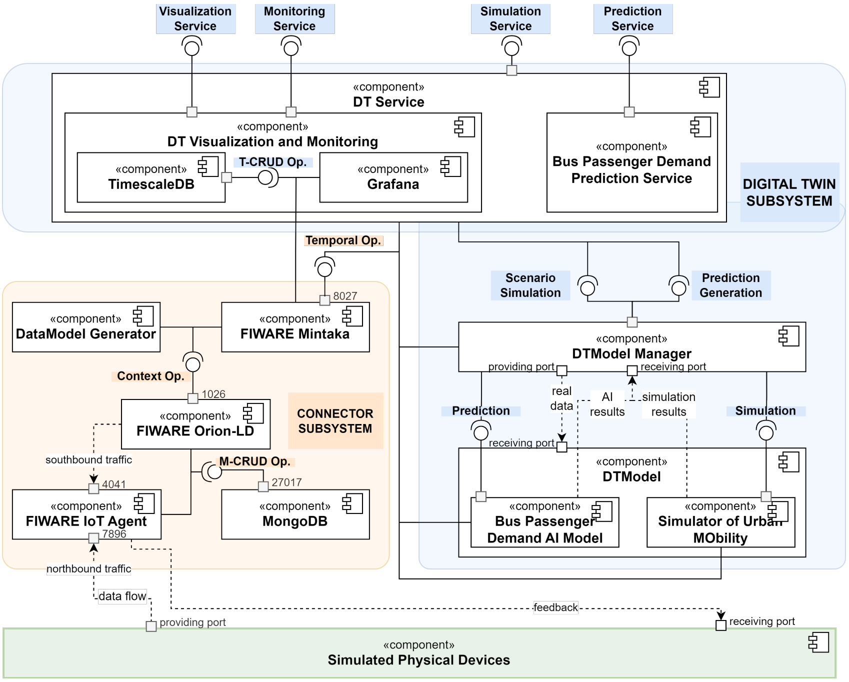

FIGURE 2. UML Component&Connector of DTE Development View.

- `MongoDB` is used by the CB to store context data and by the IoT Agent to hold device information (e.g., device authentication keys).
- `FIWARE Mintaka` exposes the `Temporal Op` interface and is responsible for persisting historic context in `TimescaleDB` through `T-CRUD Op` API.
- `FIWARE IoT Agent` JSON is the NGSI - Linked Data (LD) adapter receiving and adapting both data flow from the physical devices (*northbound* traffic) and commands coming from the CB (*southbound* traffic). Hence, in our case study, it realizes both `PhysicalAdaptor` and `DigitalAdaptor`.

The core of the architecture is the *Digital Twin Subsystem*, where the `DTModel` is composed of two behavioral models. On one hand, the open source microscopic and multi-modal traffic `Simulator of Urban MObility` simulates different urban scenarios considering both private and public traffic. Starting from origin-destination matrices, public transit scheduling data (in the GTFS format) and OpenStreetMap networks, we simulate weekday and weekend scenarios, allowing the evaluation of novel policies for traffic and/or public transport management. Moreover, we simulate anomaly situations such as events, disruptions, or strikes, to understand the impact of such situations on traffic and passenger flow, and the effectiveness of different policies.

On the other hand, `Bus Passenger Demand AI Model` is a three-layer Long Short Term Memory (LSTM) network realized with Tensorflow. It enables the prediction of passenger demand for future timestamps, based on historical or simulated data. Both DT models receive the PT northbound traffic by the *providing port* of `DTModel Manager` and send their outcomes on its *receiving port*. Additionally, they can retrieve or provide data directly by invoking `Mintaka` API.

The `DTModel Manager` exposes `Scenario Simulation` and `Prediction Generation` functionalities to the `DT Service`, for triggering the respective components. This chain enables the `Simulation Service` and bus passenger demand `Prediction Service`. The advantage of DT is that predictions can be further used to simulate alternative scenarios and make decisions, e.g., bus rerouting. Please note that, in our case study, the DTFeedback is included in `DT Service` and is the *bus re-scheduling* suggestion to human operators, sent on PT *receiving port*, via the `IoT Agent`. The other two DT services are the `Visualization` of PT and DT data and the `Monitoring` of PT states through its virtual

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

replica. They are implemented through `Grafana`[18], open source analytics and interactive visualization multi-platform. `DT Service` outcomes can be directly stored in TimescaleDB or accessed through `Mintaka`.

Discussion

We demonstrated how the DTE Development View for a small urban mobility scenario effectively implements the DTE entities from the Logical View. Despite the simplicity of our case study, the DT requirements are met through the integration of the SUMO simulator for behavioral modeling and FIWARE GEs for data management.

The multi-view architecture's strength lies in its flexibility and domain-agnostic design. With its modular structure, the DTE can efficiently be extended to integrate other aspects of urban scenarios, e.g., Weather DTs, Building DTs. Moreover, the Logical View can be applied in various domains with minimal changes. For instance, transitioning from urban mobility to other sectors would require only minor adjustments, such as adapting the DT model, and choosing a suitable simulator and FIWARE Smart Data Models.

FIWARE GEs ensure robust data and system interoperability, with APIs for seamless interaction between Context Broker and other services, such as Apache[19]. Moreover, the FIWARE Orion-LD Broker supports scaling with growing data volumes, while the IoT Agent facilitates adaptation to new data sources without custom adaptors.

Though theoretically capable of integrating diverse devices and data sources, our case study involves a limited number of entities. As the system scales, challenges in communication, storage, or processing may arise, particularly with increased demands on behavioral models. Additionally, heavy reliance on FIWARE introduces a dependency where future updates could impact performance and stability. To mitigate this, we use a modular approach to FIWARE components, allowing easier updates and reducing the risk of obsolescence.

Conclusion and Future Work

In this paper, we addressed the problem of what a DT-enabled ecosystem is and how it can be implemented, by focusing on the different Architectural Views and by providing a technological mapping with FIWARE.

Future work will focus on refining the general proposal by improving the multi-view architecture and validating its versatility across various domains. We plan to apply the DTE Logical View to other sectors (e.g., environmental monitoring) to showcase its domain-agnostic nature. For the urban mobility case study, we will perform large-scale scalability tests, optimize data transmission and storage, and explore distributed or cloud-based solutions. Finally, we will address security and privacy concerns using FIWARE tools like Keyrock[20] in both the Logical and Development Views.

Acknowledgments

This work has been partially supported by the *Hitachi Rail* company, which provided us valuable domain expertise, and by the Spoke 9 *Digital Society & Smart Cities* of ICSC - Centro Nazionale di Ricerca in High-Performance-Computing, Big Data and Quantum Computing, funded by European Union - NextGenerationEU (PNRR-HPC, CUP: E63C22000980007).

References

1. P. Kuruppuarachchi, S. Rea, and A. McGibney, "An architecture for composite digital twin enabling collaborative digital ecosystems," in *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 980–985, 2022.
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. A. De Benedictis, N. Mazzocca, A. Somma, and C. Strigaro, "Digital twins in healthcare: An architectural proposal and its application in a social distancing case study," *IEEE Journal of Biomedical and Health Informatics*, vol. 27, no. 10, pp. 5143–5154, 2023.
4. H. Marah and M. Challenger, *An Architecture for Intelligent Agent-Based Digital Twin for Cyber-Physical Systems*, pp. 65–99. Singapore: Springer Nature Singapore, 2023.
5. A. Macías, E. Navarro, C. E. Cuesta, and U. Zdun, "Architecting digital twins using a domain-driven design-based approach*," in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, pp. 153–163, 2023.

---

[18] https://grafana.com/

[19] https://fiware-cosmos.readthedocs.io

[20] https://fiware-idm.readthedocs.io

This article has been accepted for publication in IEEE Software. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/MS.2024.3518752

6. E. Ferko, A. Bucaioni, and M. Behnam, “Architecting digital twins,” *IEEE Access*, vol. 10, pp. 50335–50350, 2022.
7. D. Lehner, J. Pfeiffer, E.-F. Tinsel, M. M. Strljic, S. Sint, M. Vierhauser, A. Wortmann, and M. Wimmer, “Digital twin platforms: Requirements, capabilities, and future prospects,” *IEEE Software*, vol. 39, no. 2, pp. 53–61, 2022.
8. P. Kruchten, “The 4+1 view model of architecture,” *IEEE Software*, vol. 12, pp. 45–50, 11 1995.
9. B. Tekinerdogan and C. Verdouw, “Systems architecture design pattern catalog for developing digital twins,” *Sensors*, vol. 20, no. 18, 2020.
10. P. Bellavista, N. Bicocchi, M. Fogli, C. Giannelli, M. Mamei, and M. Picone, “Requirements and design patterns for adaptive, autonomous, and context-aware digital twins in industry 4.0 digital factories,” *Computers in Industry*, vol. 149, p. 103918, 2023.
11. J. Conde, J. Muñoz, A. Alonso, S. Lòpez-Pernas, and J. Salvachua, “Modeling digital twin data and architecture: A building guide with fiware as enabling technology,” *IEEE Internet Computing*, vol. PP, pp. 1–1, 02 2021.
12. E. Ferko, A. Bucaioni, P. Pelliccione, and M. Behnam, “Standardisation in digital twin architectures in manufacturing,” in *2023 IEEE 20th International Conference on Software Architecture (ICSA)*, pp. 70–81, 2023.
13. E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.
14. A. Alonso, A. Pozo, J. M. Cantera, F. De la Vega, and J. J. Hierro, “Industrial data space architecture implementation using fiware,” *Sensors*, vol. 18, no. 7, 2018.
15. F. Rocco di Torrepadula, S. Di Martino, N. Mazzocca, and P. Sannino, “A reference architecture for data-driven intelligent public transportation systems,” *IEEE Open Journal of Intelligent Transportation Systems*, 2024.

Franca Rocco di Torrepadula is a PhD student in Information Technology and Electrical Engineering at the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II, 80125 Naples, Italy. Her research activities include the definition and application of deep learning based approach for intelligent transportation systems, with the aim of enhancing public transports, focusing also on privacy and energy aspects. Rocco di Torrepadula received her Masters’ Degree in Computer Engineering in 2021 from the University of Naples Federico II. Contact her at franca.roccoditorrepadula@unina.it

Alessandra Somma is a PhD student in Information Technology and Electrical Engineering at the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II, 80125 Naples, Italy. Her research activities include Digital Twins, their architectural and security issues, their application in IoT/IIoT, healthcare and railway contexts and their usage for the enhancement of Cyber-Physical Systems resilience. Somma received her Masters’ Degree in Computer Engineering in 2021 from the University of Naples Federico II. Contact her at alessandra.somma@unina.it

Alessandra De Benedictis is an Assistant Professor at the Department of Electrical Engineering and Information Technology of the University of Naples Federico II, 80125 Naples, Italy. Her research interests include secure development methodologies, security assessment, and Digital Twins architectures, applications and services. De Benedictis received her PhD in Computer and Automation Engineering in 2013 from the University of Naples Federico II. Contact her at alessandra.debenedictis@unina.it.

Nicola Mazzocca is a Full Professor of Computer Systems at the Department of Electrical Engineering and Information Technologies of the University of Naples Federico II, 80125 Naples, Italy. His research interests include distributed systems architectures, high-performance computing, and safety-critical applications. Mazzocca received his PhD in Electronic and Computer Engineering in 1991 from the University of Naples Federico II. Contact him at nicola.mazzocca@unina.it.