







A Distributed Event-Orchestrated Digital Twin Architecture for Optimizing Energy-Intensive Industries

Nicolò Bertozzi¹^a, Anna Geraci¹^b, Letizia Bergamasco^{1,2}^c, Enrico Ferrera¹^d,
Edoardo Pristeri¹^e and Claudio Pastrone¹^f

¹Fondazione Links, Via Pier Carlo Boggio 61, 10138 Turin, Italy

²Department of Control and Computer Engineering, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy
{name.surname}@linksfoundation.com

Keywords: Distributed Microservices Architecture, Digital Twin, Event-Based Orchestration, Interoperability, Energy Optimization, Industry, Computing Continuum.

Abstract: This paper presents a novel distributed architecture designed to spawn digital twin solutions to improve energy efficiency in energy-intensive industrial scenarios. By executing user-defined workflows, our platform enables the implementation of real-time monitoring, forecasting, and simulation microservices to enhance decision-making strategies for optimizing industrial processes. Leveraging a stateless centralized orchestration mechanism built around an Apache Kafka-based backbone, the platform ensures scalability, fault tolerance, and efficient handling of heterogeneous data. Key features include intuitive workflow configuration, asynchronous communication for streamlined workflow execution, and API-driven scheduling for dynamic, event-based task management. This platform will be deployed and validated in several energy-intensive industrial scenarios, supporting the management of energy systems of different plants, to prove its effectiveness across a wide range of energy management challenges.


1 INTRODUCTION


The rising global demand for smarter, more sustainable industrial practices has propelled the adoption of Digital Twin (DT) technologies. These systems, which create virtual replicas of physical processes or assets, hold immense potential for optimizing operations, reducing resource wastage, and addressing the pressing challenges of climate change. Within energy-intensive industries—responsible for nearly 27% of total energy consumption in EU countries (Odyssee-Mure Project, 2023)—DTs offer transformative solutions by enabling real-time energy monitoring, predictive analytics, and dynamic process optimization (Zhang et al., 2021; Henriksen et al., 2022). Implementing DT technologies is increasingly necessary to enable the seamless execution of Artificial Intelligence-driven analytics and simulation


modules, which are critical to optimizing industrial processes and decision-making. Achieving this requires the ability to integrate heterogeneous systems, execute complex workflows efficiently, and maintain scalability under dynamic industrial demands. Moreover, as industrial ecosystems become more complex, DT platforms must effectively orchestrate distributed components, ensuring efficient coordination, consistency, and fault tolerance.


Several challenges must be addressed for DT architectures to reach their potential. These include data integration and interoperability, ensuring seamless flow across diverse systems; human-computer interaction and usability, providing intuitive interfaces for non-expert users; scalability, managing growing workflows and events without performance degradation; and orchestration, enabling smart communication and coordination among modular components. Overcoming these hurdles requires innovative architectures tailored to the complexity and dynamic demands of modern industrial environments.


In response to these challenges, this paper introduces a distributed event-orchestrated architecture for digital twins, designed to streamline deployment and enhance operational efficiency. This novel architec-


^a <https://orcid.org/0000-0003-2049-9876>

^b <https://orcid.org/0009-0006-2251-8347>

^c <https://orcid.org/0000-0001-6462-1699>

^d <https://orcid.org/0000-0002-4671-3861>

^e <https://orcid.org/0000-0003-4538-3044>

^f <https://orcid.org/0000-0003-0471-8434>

ture represents a transformative advancement in industrial digital twin solutions by seamlessly integrating Internet of Things (IoT) sensors, Artificial Intelligence (AI) algorithms, and simulation tools. It is designed to enable effortless customization for diverse use cases while addressing the key limitations of existing platforms. The architecture's key features include: i) an intuitive workflow configuration that simplifies the management of complex interactions between modules; ii) a stateless microservices design and asynchronous workflow execution, ensuring efficiency in distributed environments; and iii) API-driven scheduling capabilities, allowing workflows to be triggered dynamically by time-based or data-driven events. These features collectively empower organizations to leverage DT functionality with greater adaptability, efficiency, and ease of use. This platform has been developed within the FLEXIndustries project, and will be deployed in several industrial energy management settings, for use cases such as energy demand forecasting and energy consumption optimization.

This paper is organized as follows: Section 2 reviews related digital twin approaches; Section 3 describes the presented digital twin architecture; Section 4 details its orchestration mechanisms; Section 5 presents the use cases in which the presented platform is deployed; and Section 6 highlights the conclusion and future work.

2 RELATED WORK

Several studies have demonstrated the effectiveness of DT architectures in sectors such as manufacturing (Javaid et al., 2023; Alberti et al., 2024), energy (Yu et al., 2022), and smart cities (Farsi et al., 2020), particularly in optimizing resource use and enhancing operational efficiency. One notable example of a digital twin framework is the REPLICIA system (Rossini et al., 2020), developed as part of the RECLAIM European project. REPLICIA offers a scalable and flexible architecture for realizing intelligent digital twins, leveraging an event-based communication model and technologies like Apache Kafka. However, the framework's distributed orchestration model, where each module has to manage its own communication logic, introduces limitations. Specifically, the absence of centralized orchestration can lead to complex, interdependent workflows, where mismanagement of message publishing and subscribing phases results in communication inefficiencies. Furthermore, REPLICIA requires manual configuration and hard-coding of new workflows, making the

process cumbersome and time-consuming.

In contrast to REPLICIA's distributed orchestration model, in this paper we introduce a centralized approach, which seeks to prevent inefficiencies by consolidating control in a central orchestration component. This simplification aims to streamline workflow management, reduce configuration overhead, and improve system scalability. While centralized orchestration offers significant advantages in terms of control and coordination, it can introduce risks such as bottlenecks, particularly in large-scale distributed environments. To mitigate such risks, we implement a stateless architecture, which contrasts with REPLICIA's stateful approach that stores all the intermediate variables and results locally. As highlighted in (Rossini et al., 2020), the stateful nature of REPLICIA can lead to delays in module availability due to the persistence of data. Instead, our stateless design eliminates reliance on local data storage, enabling more flexible and efficient handling of concurrent requests.

From the literature on orchestration, a clear need emerges for centralized and asynchronous orchestration mechanisms. Broadly, orchestration can be categorized into two main types: service orchestration and data orchestration. As discussed in (Čilić et al., 2023; Nguyen et al., 2020), service orchestration is commonly employed to manage workload demands. In these studies, tools like Kubernetes (Kubernetes, 2020) are highlighted as central solutions in this domain. In cloud-native environments, Kubernetes' Horizontal Pod Autoscaling (HPA) plays a crucial role by dynamically adjusting the number of application pods based on real-time traffic patterns, ensuring that the infrastructure scales up or down as needed to maintain optimal performance. On the other hand, data orchestration addresses the challenges of managing large-scale data flows. For instance, solutions like Apache Kafka (Sax, 2018) have been explored for enabling real-time processing of massive data streams, ensuring smooth data flow and low-latency handling (Escribà-Gelonch et al., 2024; Li et al., 2023). These works mention that Kafka offers several features making it a good fit for companies' requirements, including scalability, data grouping and partitioning, low latency, and the ability to handle a large number of diverse consumers.

Asynchronism and data orchestration are straightforward individually, but integrating them is challenging. A decision framework (Megargel et al., 2021) helps architects choose choreography, orchestration, or a hybrid approach based on requirements like coupling and visibility. For energy-intensive industries, a hybrid approach—combining asynchronous commu-

nication with centralized orchestration—best handles high data throughput and workflow control.

The DT extends beyond mere data orchestration, computation, communication, and control. In addition to these properties, Digital Twins must address security domains to ensure authorized access and restrict operation. In (Harper et al., 2019), the authors emphasize secure access as a core point of Digital Twin architectures. Clients interacting with Digital Twins should authenticate using established security best practices, which could be facilitated through federated identity systems. Furthermore, recent discussions in (Onwubiko et al., 2023; Hemdan et al., 2023; Salim et al., 2022) explore a blockchain-based solution for DT, which represents a promising technology to enhance DT security in the future.

In (Jwo et al., 2022), the term “Digital Twin” is renamed as “Data Twin”. Indeed, as we see in (He et al., 2019; Emmert-Streib and Yli-Harja, 2022; Ma et al., 2023), the quality of data is crucial in digital twins. Effective data management ensures that the DT accurately reflects the real world and supports informed decision-making. The data sources can vary significantly based on the type of data and sensor installations. As discussed in (Sun et al., 2020; Bousdekis and Mentzas, 2021), the interoperability layer of the platform must be flexible and robust.

Regarding task orchestration, traditional systems like Celery (Celery, 2024), Apache Airflow (Apache Software Foundation, 2024), and Luigi (Spotify, 2024) are widely adopted for managing workflows, each excelling in specific areas. Celery is ideal for high-throughput environments with its asynchronous task execution; Luigi provides a reliable framework for batch data processing; and Apache Airflow offers robust tools for complex workflows with comprehensive monitoring and dependency management. However, Celery’s stateful architecture limits scalability and fault tolerance; Luigi lacks support for distributed execution and dynamic scheduling; and Airflow’s complex configuration for dynamic or event-driven workflows may be challenging for new users.

Our framework builds upon the analysis of these strengths and weaknesses of popular orchestration platforms, to create an efficient and robust data orchestration mechanism for our scenarios. By adopting a stateless, microservices-based architecture, it eliminates the complexities associated with state management, enabling modular and scalable components that can be deployed independently across distributed environments. This design simplifies fault tolerance and ensures seamless scalability, even under dynamic workloads. Intuitive APIs provide dynamic scheduling capabilities, allowing workflows to be triggered

by events, time schedules, or data changes. Additionally, our platform supports advanced workflow management, such as real-time monitoring and seamless handling of complex task dependencies, all through a user-friendly configuration process. By addressing these trade-offs, our solution delivers unparalleled flexibility, scalability, and adaptability.

3 ARCHITECTURE

In this Section, we present an architecture developed for enabling digital twins of energivorous industries which enhance performance, particularly in terms of latency and communication efficiency. In this context, we define an Agent as an AI-driven or simulation module that receives data, performs computations, and generates results. Results from individual Agents may be also combined in a new user-defined workflow, to create more sophisticated outcomes. A workflow defines a series of interconnected tasks designed to achieve a specific outcome. When a workflow is in execution, we call it a pipeline. Therefore, multiple pipelines can be associated to the same workflow, with or without the same input data. The workflow file is a Python script that contains a set of platform-specific functions accessible to the user through a dedicated Python package, in order to simplify the definition of each task.

A crucial aspect of DT platforms is the orchestration of components, which coordinates interactions among platform elements to transform inputs into visible outputs. In a typical DT architecture, the input might be sensor data, while the output could be a real-time representation of a physical component’s state, such as the energy flow in an industrial plant. This process encompasses the automatic ingestion of heterogeneous data, secure storage in a persistent system, and efficient distribution to all orchestrated services. To enhance the collaboration between simulation-based and AI-based Agents, the Multi-Agent System (MAS) module, leveraging centralized orchestration functionalities, handles all these communications by publishing the corresponding control messages inside the central bus.

One of the distinctive features of our platform is its intuitive workflow configuration, allowing users to effortlessly create and modify workflows using the platform’s predefined services. This design offers exceptional flexibility by enabling workflows setup without requiring extensive coding, thereby lowering technical barriers for users. Moreover, leveraging an asynchronous communication, our platform allows independent workflows to proceed without interfer-

ence, enabling workflows to continue running while AI computations are underway. This concurrent execution is supported by a scalable, stateless design where each component executes tasks independently, treating every request as self-contained. This architectural approach enables seamless cloning of components to handle high loads, ensuring that the platform can dynamically scale to meet complex, high-demand requirements. Finally, our solution provides an efficient workflow scheduling via APIs, supporting both data-triggered and time-triggered events.

This framework provides four key functionalities: Data Gathering and Management, Data Orchestration, Data Processing and Platform Integration, as illustrated in the architecture diagram of Figure 1. The Data Gathering and Management functionality aims to manage the data flows from external sources to platform components. In particular, it is in charge of data gathering, storage, querying and distribution. The Data Orchestration functionality is designed to configure, deploy, monitor, and manage the execution of all the platform pipelines. The main component involved is the MAS, that acts as an orchestrator, efficiently distributing and executing Agents based on the data they require from the database. The Data Processing functionality is implemented by AI models and simulation services, such as energy cost forecasting. It enables AI-driven decision making by exposing the results in a platform front-end that hosts Graphical User Interfaces (GUIs) and data visualization tools. The Platform Integration functionality manages the whole infrastructure by integrating and deploying automatically the last stable versions of all the architecture modules, while ensuring security with customized authentication tools. In addition, this macro block is responsible for automatically triggering the retraining of the AI services, in order to avoid the risk of concept and data drifting.

The following paragraphs describe the role of each module depicted in Figure 1, highlighting the particular features they add to the platform.

3.1 Data Interoperability and Storage

The Data Interoperability and Storage (DIS) component facilitates the integration and management of data from a multitude of sources and protocols. The primary objective of this component is to establish a unified interface for the collection, storage, and retrieval of data from heterogeneous data sources. The compliance of this component with the Data Handler, and therefore with the DT architecture, is guaranteed by the exploitation of application communication protocols such as HTTP and MQTT, which charac-

terize the northbound of the supported interoperability frameworks, like EdgeX (Edge Foundry, 2020), Apache NiFi (Apache NiFi, 2006), and Eclipse Kura (Eclipse Kura, 2014).

3.2 Communication Backbone

The Communication Backbone (CB) facilitates the interaction between different Agents and services within the platform. It ensures efficient routing of data and requests between Agents responsible for different tasks, according to the pipelines requested by users. The success of this system hinges on an abstracted asynchronous communication proxied layer, which allows each service to receive the necessary task information only when requested by the MAS and, consequently, by the user. This decoupling ensures that Agents are not responsible for communication or orchestration, thereby maintaining the platform's flexibility and adaptability to changes or improvements. The models used to format the messages exchanged within the platform depend on the specific communication. In particular, three data models are actually adopted: one for the Agents, one for the Data Handler and one for the Multi-Agent System.

3.3 Agents

The orchestrated Agents represent the operational components of the platform. They can be either simulation-based or AI-based, and they perform user-requested computations, each handling a specific part of the process. The MAS coordinates the information flow between these Agents to provide an aggregated result to the user. In this way, the DT platform ensures that these components cooperate seamlessly to achieve a specific outcome. Each Agent operates with an abstracted communication layer, receiving only their specific tasks without full platform visibility. The physical communication between the Agents and the Metadata and Caching Storage (MCS) allows the formers to have access to the data they need to perform their activities, avoiding to exchange large messages on Kafka.

3.4 Multi-Agent System

The Multi-Agent System (MAS) serves as the platform's data orchestrator, initiating the Agents necessary for user-requested pipelines. It manages asynchronous operations, enabling the platform to handle new user inputs while executing past workflows. Connected to the orchestrated Agents and the Data Handler, the orchestrator distributes computational loads

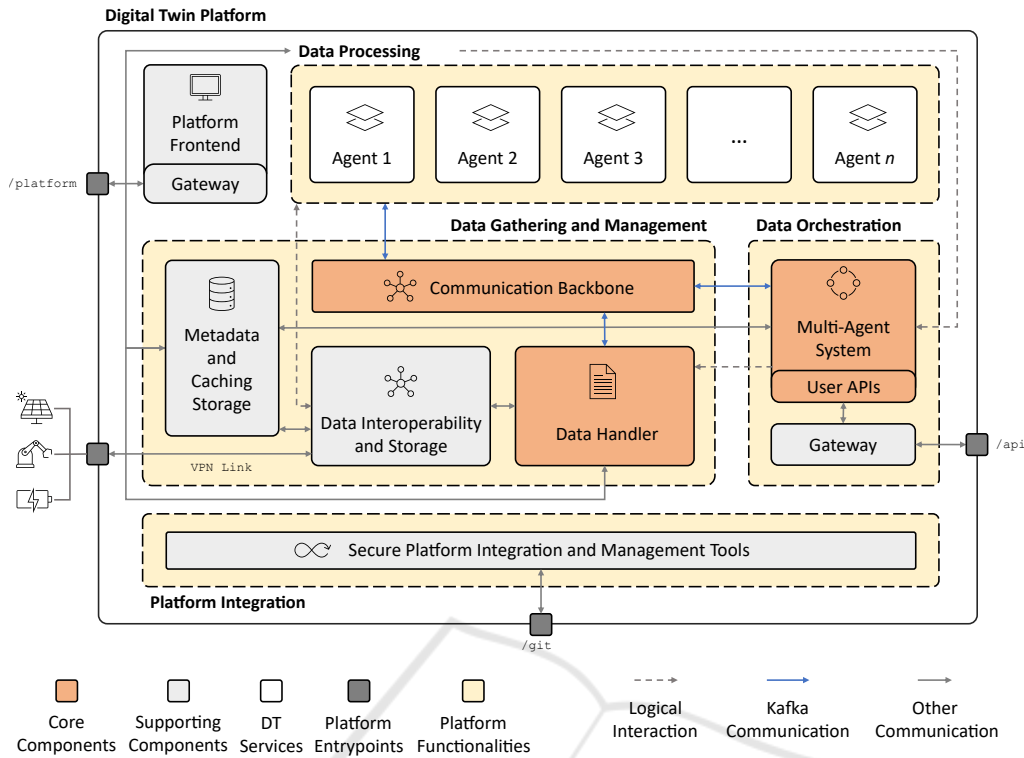


Figure 1: Digital Twin Platform Architecture.

and ensures tasks have the necessary data to proceed. This configuration enables efficient pipeline management and task execution within the platform.

The MAS can be configured through REST APIs, exposed by the User APIs Service. This module allows the user to schedule periodic workflow execution, to start a new DT pipeline, to list all the pipelines that are currently running, to get the result of the concluded ones, and to define new workflows.

3.5 Data Handler

The Data Handler (DH) oversees all data requests originating from the Agents, providing an asynchronous interface between the DIS and other core components. Hence, it handles the data requests intermediated by the MAS to the DIS. The unified data model adopted by the DIS is then wrapped within the DH messages, which are then transmitted over the CB using the platform's data models. The DH ensures that core components can comprehend exchanged events by establishing a separation between the DIS format and the data tasks format. Once the DIS has completed a request, the DH publishes the data to the MCS for Agent access and notifies it to the orchestrator.

3.6 Platform Frontend

The Frontend service provides access the platform dashboard. The GUI allows the user to schedule workflows, view a list of submitted tasks, and obtain the results of completed tasks. Finally, it is possible to retrieve the real-time data stream from the multiple data sources connected through the DIS.

3.7 Security Platform Integration and Management Tools

This module encompasses a suite of tools designed to facilitate monitoring of platform activity and to guarantee the security of interactions between internal and external actors. Health checks and metrics regarding the CB are exploited to keep track of the load capacity of the core modules, in the view of cloning some of them in case of overload. On the other side, Secure Sockets Layer (SSL) over Transmission Control Protocol (TCP) is exploited for protecting the APIs, while an Identity and Access Management (IAM) framework is responsible for the authentication and authorization of users and microservices.

4 ORCHESTRATION MECHANISM

The orchestration mechanism presented in this paper is based on a completely stateless centralized management system and employs asynchronous operations to guarantee a complete service decoupling within the platform. This centralized management system organizes the sequence of steps required to complete a pipeline, thereby reducing the computational load of each Agent. For clarity, the orchestration mechanism follows a centralized paradigm in order to concentrate the data management responsibility on a single component. However, it relies on a cluster of Kafka brokers and on stateless components, which ensure scalability and fault tolerance. The stateless components replication and the number of Kafka brokers is administered by the Kubernetes HPA. The asynchronous operations, enabled by Apache Kafka, treat simulation and AI requests as events. The components only respond to inputs from the MAS, thereby simplifying their operation. The statelessness of the system prevents the unavailability of modules by ensuring that Agents produce outputs based solely on inputs. The MAS initiates new requests once the required data is available, thereby maintaining Agent efficiency.

The distributed paradigm ensures that multiple modules evaluate tasks, thus allowing for the duplication of overloaded services. The integration of the orchestration mechanism into Apache Kafka enables the system to handle stateless, asynchronous requests, thereby maximizing the benefits of distributed architectures. Furthermore, this feature enables the DT to operate across the computing continuum. In this way, the DT can be seen as a unified entity within a more complex system comprising multiple instances of the same platform deployed across different locations, including far-edge, edge, fog, or cloud.

Figure 2 shows the sequence of steps leading the platform to complete a pipeline requested by the user. The backend APIs can be exploited as a high level interface that allows users to run new pipelines. As a consequence, the backend checks and, if necessary, retrieves from the MCS the last version of the workflow that has to be executed. At this stage, the MAS is finally activated through an *Orchestrator Task* message, sent on Kafka. This message activates the orchestrator, that has to start creating the tasks that compose the workflow. The most important fields of this message are the issuer, which is the identifier of the user that has created the pipeline, and the workflow file. In the example of Figure 2, the MAS firstly requests some data by publishing the *Data Task* messages addressed to the DH on Kafka,

as wrappers for the historical queries that *Agent 1* and *Agent 2* need. The *Data Task* mainly contains the query of the data that are useful for the pipeline execution. Once the DIS component completes these activities, the MAS can share the historical data provided by the DIS as input for the orchestrated Agents. This is achieved using the dedicated *Agent Task* message, which includes a reference composed of two fields: the issuer and the agent name. Depending on the values of these fields, the corresponding Agent will be triggered while others will not. Additionally, this message contains the MCS data location. Finally, the MAS can complete the pipeline by returning the overall outcome to the user, using the *Orchestrator Task* message. Since each of the above message types is used for both requests and responses, a field has been provided within them to differentiate communication beginnings from ends.

5 APPLICATION SCENARIOS

The microservice architecture presented in this work will be deployed in different industrial scenarios, in the context of the FLEXIndustries project. The project is a European Commission-funded initiative aimed at transforming energy-intensive industries through the dual focus on green energy optimization and digital twin technology for enhanced operational efficiency. The project involves companies from different production sectors, e.g., automotive, pharmaceuticals, and manufacturing. For all these diverse scenarios, our DT platform offers a fully reconfigurable framework. The practical implementation of each scenario is based on the definition of the interactions between the orchestrated Agents and the functionality they provide to the platform. In this way, it is easy to build vertical customization by simply adding the logic that determines the execution order and the input data required to the set of Agents involved in the realization of the scenario. To be compliant with our DT notations, each of these scenarios could be associated to a different workflow. The following paragraphs present examples of the covered use cases, focused on managing energy systems of the companies' plants.

Efficiency Monitoring. Monitoring process parameters with the ultimate goal of optimizing the energy efficiency of specific equipment and reducing overall energy consumption.

Renewable Energy Surplus Management. Forecasting and management of internal energy flows to support decisions about energy sales or storage.

Optimal Energy Source Selection. Selection of the

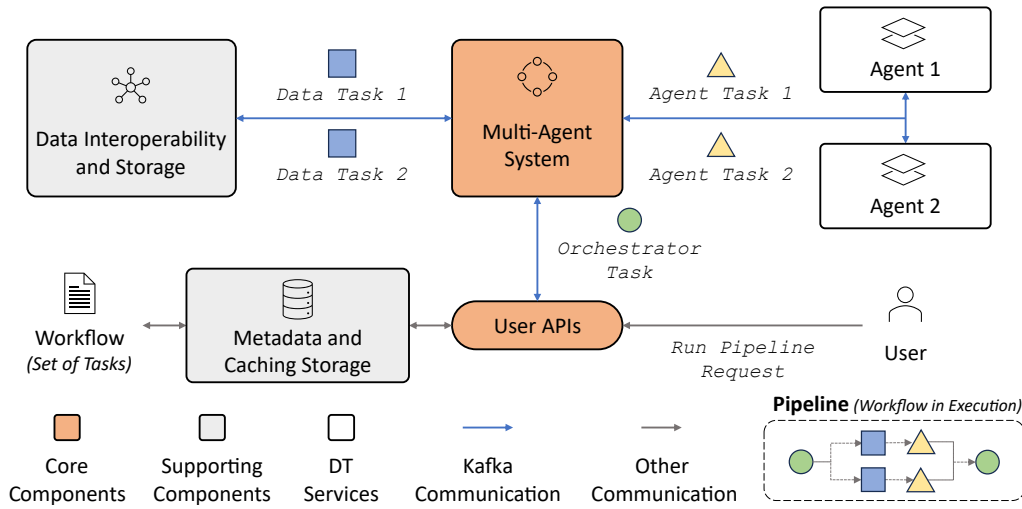


Figure 2: Steps involved in the execution of a pipeline within the Digital Twin Platform.

optimal energy source in a specific moment of the day for a given process.

Precise Energy Estimation. Accurate estimation of the energy consumption for companies that purchase electricity in fixed quantities, to avoid penalties for overuse or waste from underuse.

Optimized Scheduling. Generation of optimized schedules for machine operations, in situations where specific machines may need to be temporarily stopped due to high energy consumption.

Energy Monitoring. Visual representation of a company's plant, including all components and augmented information about energy consumption in each process and machine.

A key feature of the platform is its ability to reuse multiple Agents for various applications, enabling users to create diverse workflows with ease. For example, Agents providing energy price forecasting and energy demand forecasting can be utilized across multiple workflows serving different objectives. The platform has been designed and developed based on the requirements of energy-intensive industries within the context of the project. However, these requirements are not overly restrictive, allowing the platform to adapt and perform effectively across diverse industry contexts and scenarios, both within and beyond the energy sector.

6 CONCLUSION AND FUTURE WORK

In this paper we have presented a distributed Digital Twin platform designed to seamlessly integrate data from diverse sources, enabling real-time moni-

toring, predictive maintenance, and operational optimization across the industrial ecosystem. This platform's distinctive architecture and design make it a highly adaptable solution for modern energy management needs, equipped to serve multiple industries with specific yet flexible workflows.

A central feature of our DT platform is its user-configurable workflow framework. The platform abstracts the complexity of low-level operations, enabling users to derive actionable outputs by specifying high-level workflows definition, data inputs, and algorithms. Beyond flexibility, leveraging an asynchronous communication and a stateless component design, the platform offers exceptional scalability and the ability to execute multiple workflows concurrently. These features allow the platform to handle high computational loads, making it an ideal solution for data-intensive, industrial environments where computational power and responsiveness are crucial. Finally, our platform offers smart scheduling for periodic workflows, which can be triggered by data events or time intervals.

The presented DT architecture will undergo empirical testing across various industrial application scenarios, in the context of the FLEXIndustries project, ensuring the platform's robustness, scalability, and effectiveness in real world scenarios.

Future work will include the definition of a met-language for workflow configuration and updates, thus completely removing the need for programming language expertise and further improving the platform's usability. This will enable the decoupling the deployment of the platform and the workflow configuration, with the latter not requiring specialized ICT background.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Horizon Europe Programme under grant agreement n. 101058453.

REFERENCES

- Alberti, E., Alvarez-Napagao, S., Anaya, V., Barroso, M., Barrué, C., Beecks, C., Bergamasco, L., Chala, S. A., Gimenez-Abalos, V., Graß, A., et al. (2024). Ai lifecycle zero-touch orchestration within the edge-to-cloud continuum for industry 5.0. *Systems*, 12(2):48.
- Apache NiFi (2006). Apache nifi: Data processing and distributing system. Available at: <https://nifi.apache.org> (Accessed: 4 December 2024).
- Apache Software Foundation (2024). Apache airflow. Accessed: 2024-11-20.
- Bousdekis, A. and Mentzas, G. (2021). Digital twins for intelligent manufacturing: A survey of applications, architecture, and challenges. *Computers in Industry*, 128:103440.
- Celery (2024). Optimizing — celery 5.4.0 documentation. Accessed: 2024-11-20.
- Eclipse Kura (2014). Eclipse kura: Open-source iot edge framework. Available at: <https://eclipse.dev/kura/> (Accessed: 4 December 2024).
- Edge Foundry (2020). Edgex: Open-source edge platform. Available at: <https://www.odyssee-mure.eu/> (Accessed: 4 December 2024).
- Emmert-Streib, F. and Yli-Harja, O. (2022). What is a digital twin? experimental design for a data-centric machine learning perspective in health. *International Journal of Molecular Sciences*, 23(21):13149.
- Escribà-Gelonch, M., Liang, S., van Schalkwyk, P., Fisk, I., Long, N. V. D., and Hessel, V. (2024). Digital twins in agriculture: Orchestration and applications. *Journal of Agricultural and Food Chemistry*.
- Farsi, M., Daneshkhah, A., Hosseinian-Far, A., Jahankhani, H., et al. (2020). Digital twin technologies and smart cities.
- Harper, K. E., Malakuti, S., and Ganz, C. (2019). Digital twin architecture and standards.
- He, R., Chen, G., Dong, C., Sun, S., and Shen, X. (2019). Data-driven digital twin technology for optimized control in process systems. *ISA Transactions*, 95:221–234.
- Hemdan, E. E. D., El-Shafai, W., and Sayed, A. (2023). Integrating digital twins with iot-based blockchain: concept, architecture, challenges, and future scope. *Wireless Personal Communications*, 131(3):2193–2216.
- Henriksen, H. J. et al. (2022). A new digital twin for climate change adaptation, water management, and disaster risk reduction (hip digital twin). *Water*, 15(1):25.
- Javaid, M., Haleem, A., and Suman, R. (2023). Digital twin applications toward industry 4.0: A review. *Cognitive Robotics*, 3:71–92.
- Jwo, J. S., Lee, C. H., and Lin, C. S. (2022). Data twin-driven cyber-physical factory for smart manufacturing. *Sensors*, 22(8):2821.
- Kubernetes (2020). Kubernetes. <https://www.kubernetes.io>. Accessed: 23 June 2020.
- Li, X., Zhang, Y., Liu, Y., Li, P., Hu, H., Wang, L., and Zhang, C. (2023). A high-throughput big-data orchestration and processing system for the high energy photon source. *Journal of Synchrotron Radiation*, 30(6).
- Ma, Y., Zhu, X., Lu, J., Yang, P., and Sun, J. (2023). Construction of data-driven performance digital twin for a real-world gas turbine anomaly detection considering uncertainty. *Sensors*, 23(15):6660.
- Megargel, A., Poskitt, C. M., and Shankaraman, V. (2021). Microservices orchestration vs. choreography: A decision framework. In *2021 IEEE 25th International Enterprise Distributed Object Computing Conference (EDOC)*, pages 134–141.
- Nguyen, T. T., Yeom, Y. J., Kim, T., Park, D. H., and Kim, S. (2020). Horizontal pod autoscaling in kubernetes for elastic container orchestration. *Sensors*, 20(16):4621.
- Odyssee-Mure Project (2023). About the odyssee-mure project. Available at: <https://www.odyssee-mure.eu/> (Accessed: 21 May 2024).
- Onwubiko, A., Singh, R., Awan, S., Pervez, Z., and Ramzan, N. (2023). Enabling trust and security in digital twin management: A blockchain-based approach with ethereum and ipfs. *Sensors*, 23(14):6641.
- Rossini, R. et al. (2020). Replica: A solution for next generation iot and digital twin based fault diagnosis and predictive maintenance. *SAM IoT*, 2739:55–62.
- Salim, M. M., Comivi, A. K., Nurbek, T., Park, H., and Park, J. H. (2022). A blockchain-enabled secure digital twin framework for early botnet detection in iiot environment. *Sensors*, 22(16):6133.
- Sax, M. J. (2018). Apache kafka. In Kleppmann, M. and Lange, C., editors, *Encyclopedia of Big Data Technologies*. Springer, Cham.
- Spotify (2024). Luigi - design and limitations. Accessed: 2024-11-20.
- Sun, S., Li, W., Zhang, Z., Feng, J., and Zhang, C. (2020). Data twin driven intelligent manufacturing: A comprehensive review. *Journal of Manufacturing Systems*, 56:547–558.
- Yu, W., Patros, P., Young, B., Klinac, E., and Walmsley, T. G. (2022). Energy digital twin technology for industrial energy management: Classification, challenges and future. *Renewable and Sustainable Energy Reviews*, 161:112407.
- Zhang, X. et al. (2021). Digital twin for accelerating sustainability in positive energy district: A review of simulation tools and applications. *Frontiers in Sustainable Cities*, 3:663269.
- Čilić, I., Krivić, P., Podnar Žarko, I., and Kušek, M. (2023). Performance evaluation of container orchestration tools in edge computing environments. *Sensors*, 23(8):4008.