applied sciences

MDPI

Article

A Composable Architectural Model for Digital Twin Computing Applications

Saverio Ieva [1,2,†], Davide Loconte [1,†], Andrea Pazienza [3,†], Matteo Colombo [3,†], Federico Marzo [3,†], Giuseppe Loseto [2,4,†], Floriano Scioscia [1,2,*,†] and Michele Ruta [1,2,†]

[1] Department of Electrical and Information Engineering, Polytechnic University of Bari, Via E. Orabona 4, I-70125 Bari, Italy; saverio.ieva@poliba.it (S.I.); davide.loconte@poliba.it (D.L.); michele.ruta@poliba.it (M.R.)
[2] donkeyPower S.r.l., Via E. Orabona 4, I-70125 Bari, Italy; loseto@lum.it
[3] NTT DATA Italia S.p.A., Via G. Amendola 146, I-70126 Bari, Italy; andrea.pazienza@nttdata.com (A.P.); matteo.colombo@nttdata.com (M.C.); federico.marzo@nttdata.com (F.M.)
[4] Department of Engineering, LUM "Giuseppe Degennaro" University, S.S. 100 km 18, I-70010 Casamassima, Italy
* Correspondence: floriano.scioscia@poliba.it; Tel.: +39-080-5963054
† These authors contributed equally to this work.

Abstract

Digital Twins (DTs) are increasingly deployed in Industry 4.0 to enable real-time monitoring, analysis, and control, yet the transition from isolated DT instances to plant-wide ecosystems across cloud and edge infrastructures introduces fragmentation and coordination challenges among heterogeneous assets, data sources, and services. This paper addresses this gap by proposing a cloud-native Digital Twin Computing Layer (DTCL) that provides a unified control and orchestration plane for composing and operating DT applications in Smart Manufacturing. The DTCL is designed as a three-tier architecture comprising a developer-facing user interface, a Deploy Engine for automated deployment and lifecycle management, and a Service Catalog of reusable, independently deployable microservices. Standardized interaction is supported through semantic DT models and API- and message-based communication mechanisms. A governance workflow, based on service discovery and validation, is introduced to support non-redundant integration and controlled evolution of services. The approach is demonstrated through a Smart Manufacturing predictive maintenance case study and further extended with a Smart Mobility scenario for urban public transport planning, highlighting the flexibility of the DTCL across different application domains. Overall, the DTCL supports modular composition, interoperability, and lifecycle governance across heterogeneous Digital Twin applications, providing a scalable foundation for both industrial and urban data-driven scenarios.

Keywords: digital twin platforms; digital twin computing layer; composable architecture; Industry 4.0; smart manufacturing; smart mobility

Check for updates

Academic Editor: Arkadiusz Gola

Received: 24 March 2026
Revised: 22 April 2026
Accepted: 27 April 2026
Published: 5 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Industry 4.0 manufacturing systems are increasingly integrating distributed cyber-physical components and DT to enable real-time monitoring, analysis, and control of operations. As factories scale up deployments from isolated Digital Twins of single machines to plant-wide DT ecosystems spanning cloud and edge devices, the complexity of managing these distributed assets grows significantly. In such large-scale, heterogeneous environments, effectively orchestrating each component becomes a major challenge [1].

Recent studies have explored how AI and user-interface technologies can enhance Digital Twin systems, improving decision-making and operator support on the shop floor. Approaches combining generative AI, such as LLMs, and knowledge-graph reasoning have shown benefits in terms of information accessibility and context-aware guidance [2,3]. Similar approaches are emerging in industrial DT implementations that integrate AI and Industry 4.0 technologies for anomaly detection and process optimization [4,5]. At the infrastructure level, cloud-native and cloud–edge architectures based on microservices and serverless execution are increasingly adopted to distribute intelligence across industrial IoT systems [6]. Despite these advancements, current Digital Twin implementations in manufacturing still lack a unified management layer capable of coordinating distributed components and diverse functionalities. Solutions that are currently in use are frequently compartmentalized, as they are designed to address specific use cases (e.g., predictive maintenance, logistics optimization) without offering a general mechanism for integrating heterogeneous data sources, analytics modules, simulation models, or advanced interaction components. There is a clear industrial need for an overarching DTCL that can orchestrate multiple DT and associated services.

This paper proposes cloud-native DTCL architecture specifically designed for Smart Manufacturing environments. The DTCL provides a unified control layer that orchestrates multiple heterogeneous Digital Twin services, overcoming the fragmentation that characterizes current solutions. Building on microservice and serverless design principles, the DTCL operates seamlessly across the C2E continuum [7], supporting the distributed deployment of DT components where they are most effective. Its execution framework supports uniform data exchange based on standardized Digital Twin models and encourages the composition and reuse of DT services across heterogeneous environments. The DTCL acts as a flexible and modular control plane, simplifying the integration of data analytics, simulation engines, and interaction mechanisms into industrial workflows, while providing the scalability, interoperability, and maintainability required for next-generation Industry 4.0 systems. In addition to industrial scenarios, the proposed architecture is also evaluated in a Smart Mobility context, demonstrating its applicability beyond manufacturing environments.

The main contributions of this paper can be summarized as follows:

- A cloud-native DTCL for Smart Manufacturing. This paper introduces the DTCL as a dedicated architectural layer that orchestrates heterogeneous DT and associated services across the cloud-to-edge continuum, addressing fragmentation and coordination limitations of existing Industry 4.0 Digital Twin solutions.
- A composable architectural model based on reusable Digital Twin services. The proposed architecture adopts a composable paradigm in which Digital Twin functionalities are encapsulated as reusable, independently deployable microservices. This enables flexible composition, incremental system evolution, and systematic reuse across heterogeneous Smart Manufacturing scenarios.
- A unified orchestration and lifecycle management mechanism across the cloud-to-edge continuum. The DTCL provides automated deployment, versioning, and lifecycle management of Digital Twin services through a cloud-native Deploy Engine, ensuring scalability, consistency, and reproducibility across distributed infrastructures.
- A governance framework for service validation and reuse in Digital Twin ecosystems. The architecture integrates validation and cataloging mechanisms that support non-redundant integration, controlled evolution, and coherent management of reusable Digital Twin services within large-scale manufacturing platforms.
- Validation through a predictive maintenance use case in Industry 4.0 manufacturing. The applicability of the proposed architecture is demonstrated through a predictive maintenance scenario, showing how the DTCL supports the end-to-end develop-

ment lifecycle of data-driven Digital Twin applications, from service composition to deployment and operation.
- Extension and validation through a Smart Mobility use case for urban public transport planning. Complementing the industrial scenario, the proposed architecture is validated in a Smart Mobility context, demonstrating how the DTCL supports data ingestion, scenario simulation, routing validation, and KPI-driven decision-making in large-scale, dynamic urban environments.

The remainder of this article is organized as follows. Section 2 reviews related work and the technological background of Digital Twins in manufacturing. Section 3 presents the proposed DTCL architecture and its core components. Section 4 describes the implementation of the prototype and discusses its applicability through case studies in both Smart Manufacturing and Smart Mobility. Finally, Section 5 discusses the results and outlines future research directions.

2. Background

2.1. Digital Twin Concepts and Evolution

A DT-based system requires three essential components: a *Physical Twin*, its *Digital Twin*, and a bidirectional data flow connecting the two domains [8]. Physical assets generate data that feed the virtual space, where information is processed for modeling, testing, and optimization before being returned to the physical system through a continuous feedback loop. The global relevance of DT is evidenced by the exponential rise in scientific publications and the rapid market growth reported in recent years, with adoption accelerating further during the COVID-19 period [9]. This trend is supported by advances in computing power, communication networks, IoT sensing, AI/ML, and scalable cloud infrastructures. As DT adoption grows across sectors, it becomes increasingly important to understand the value they provide in each application domain [1,10].

The DT paradigm is structured around three fundamental pillars: modeling, visualization, and simulation [8,11]. These elements are illustrated in Figure 1, which shows how the Digital Twin processes data from the physical world and exposes the resulting information, insights, and predictions through dedicated capabilities. Modeling constitutes the foundation of the virtual representation. In this phase, the physical entity is described using semantic standards such as DTDL (Digital Twins Definition Language: https://azure.github.io/opendigitaltwins-dtdl/DTDL/v3/DTDL.v3.html, accessed on 2 February 2026), OWL [12], or AAS [13]. The resulting digital model captures properties, relationships, and structural or behavioral characteristics, and is continuously updated with data from sensors, IoT devices, monitoring systems, and other sources. DT can represent individual components, machines, production lines, or even entire plants and energy networks requiring the exploitation of SoS, where multiple Digital Twin instances cooperate and share information to optimize system-wide behavior. Visualization allows stakeholders to interact with the DT through 2D or 3D representations. Depending on the use case, this may take the form of dashboards, engineering information models, or immersive VR/AR environments. Such visual interfaces enhance situational awareness, facilitate diagnosis, and support human–machine collaboration. Finally, simulation enables the evaluation of system performance under different operating conditions. Through physics-based, data-driven, or hybrid models, simulation supports both predictive and prescriptive analyses. When coupled with AI and ML techniques, the DT can autonomously explore alternative scenarios, forecast system evolution, and provide real-time decision support.

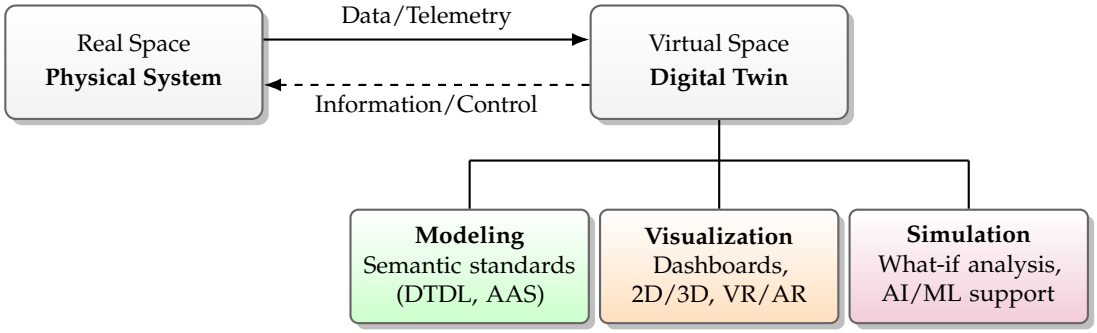


Figure 1. Conceptual view of the DT paradigm.

Common DT architectures follow a hierarchical multilayer structure [1], where responsibilities are progressively abstracted from the shop floor to higher-level services. As illustrated in Figure 2, the Physical Layer (L0) governs the interaction with the real world, handling sensor and actuator data from machines, robots, and industrial assets. Above it, the Data Layer (L1) manages bidirectional information flows, aggregating raw telemetry and supporting data-processing pipelines, storage, and basic analytics. The Integration Layer (L2) provides the middleware and communication mechanisms that interconnect heterogeneous systems, ensuring interoperability, protocol translation, and cybersecurity enforcement across the infrastructure. Moving upward, the Service Layer (L3) hosts higher-level functionalities such as simulation services, diagnostics, prognostics, and advanced analytics. At the top of the stack, the Decision Layer (L4) exposes supervisory intelligence for optimization, fault prediction, and strategic decision support, integrating the outputs of the underlying layers to enable informed operational and managerial actions.

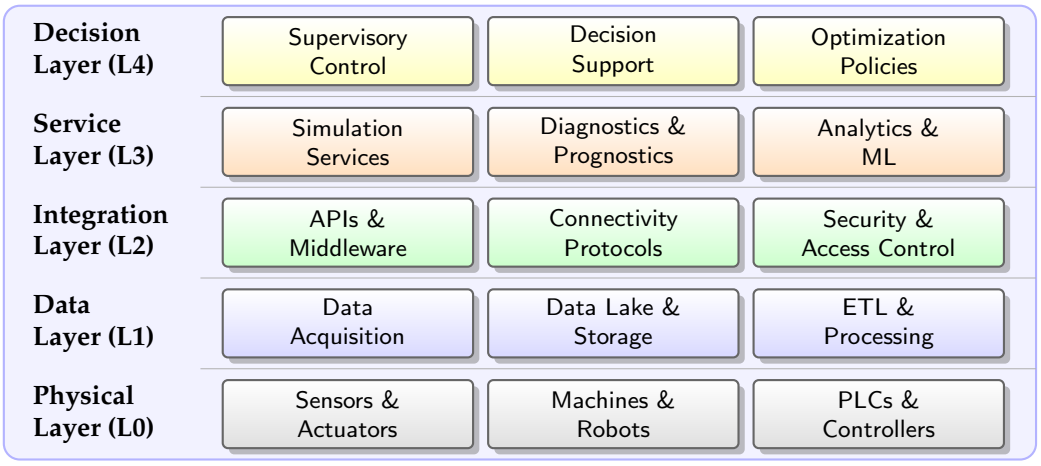


Figure 2. Five-layer DT architecture with functional components in each layer.

As DT leverages advancements in IoT, AI, and cloud technologies to maintain real-time alignment between physical and digital entities [14], DTP serve as new-generation information systems that support the design, integration, monitoring, and lifecycle management of digital representations of physical assets, enabling *data-driven decision-making* through capabilities such as modeling and simulation, interoperability, collaboration, near real-time monitoring, advanced analytics, optimization, scalability, and context-aware visualization. Through these features, DTP allow organizations to evolve from isolated DT toward interconnected ecosystems, unlocking holistic optimization in Smart Manufacturing.

The design of a DT-based system requires a rigorous scientific and technical approach. A DT platform should address specific operational challenges rather than serving solely as a monitoring tool, and its development should follow an incremental, composable strategy

that progressively expands the system's scope. Given the heterogeneity of industrial environments, DT solutions must adapt to diverse devices, data sources, and standards while ensuring reliability, scalability, security, and efficiency. In this context, fragmentation is one of the emerging risks for state-of-the-art DT solutions, which can affect each DT layer, as in what follows.

- L0: protocol incompatibility. Sensors, controllers, and IoT devices often adopt heterogeneous communication and control protocols, such as proprietary fieldbus solutions, industrial Ethernet variants, or platform-specific interfaces. This makes the Integration Layer L2 more expensive and fragile, often requiring middleware, gateways, protocol converters, or custom engineering. As a result, performance degrades and real-time synchronization becomes harder to guarantee.
- L1: data silos and lack of semantic interoperability. Relevant data for a DT may be distributed across independently managed repositories, with differences in structure, ownership, and update frequency. This complicates unified analysis at L1, increasing the need for expensive ETL, data cleaning and preprocessing. If not properly faced, data fragmentation can reduce the usefulness of insights obtained from analytics. Beyond format and encoding compatibility, ensuring a common interpretation of information is also essential for data exchange, integration and processing. Preventing ambiguities in naming conventions, data models, units, and contextual assumptions is especially necessary in collaborative and cross-domain environments, where DT must combine engineering, operational, and business information into one coherent model. Knowledge representation languages endowed with formal and explicit semantics are increasingly adopted for this purpose.
- L2: lack of unified API. Adequate domain knowledge is essential to avoid ad hoc implementations and cost optimization, particularly by reducing time-to-market without highly capital-intensive solutions. Adopting a common Integration Layer, supported by API and microservice-based architectures, facilitates interoperability, reuse of services, and extensibility across heterogeneous platforms.
- L3: service lifecycle management limitations. As discussed more in depth in Section 2.2, many platforms support deployment and monitoring, but provide limited capabilities for managing services consistently across design, implementation, update, reuse, and retirement phases. This becomes a problem when services must evolve over time as assets, models, or user requirements change. Without strong lifecycle management, updates may break compatibility, dependencies may become difficult to trace, and services may not remain aligned with the current state of the physical system.
- L4: fragmented governance of intelligence. A DT platform may successfully connect devices, aggregate data, and model physical processes, yet still fail to operate as a coherent and useful system if decision support and optimization policies remain uncoordinated and disconnected from end users' goals. Human-in-the-loop approaches are necessary to mitigate this risk, grounded in user-centered design [15], interactive machine learning and symbiotic AI [16] paradigms.

The next section further discusses these issues in the state of the art. The proposed DTCL model aims to mitigate or overcome such fragmentation risks. In this perspective, standards-compliance is a primary requirement. Several international organizations promote standardization and collaboration within the DT domain. The DTC (Digital Twin Consortium: https://www.digitaltwinconsortium.org/, accessed on 2 February 2026) unites industry, academia, and governmental bodies to define shared practices, interoperability frameworks, and educational resources. Several standards contribute to the technical specification and governance of DT systems. IEC 62443 addresses cybersecurity in industrial control environments, ensuring resilient and secure DT implementations [17], while

ISO 8000 [18] defines data-quality requirements essential for reliable analytics [19]. Interoperability is supported through ISO 10303 [20] (STEP), which standardizes the exchange of product data across platforms [21]. Complementary standards further strengthen semantic and architectural foundations: IEEE 1872-2015 [22] provides ontologies for robotics and automation, ISO/IEC 30182:2017 [23] outlines guidelines for DT creation and lifecycle management, and IEC 61499 [24] supports modular, distributed control architectures. Finally, ISO/IEC 27001 [25] offers an overarching framework for information security management, crucial for protecting DT infrastructures and data-processing layers [26].

2.2. Related Work

Data modeling plays a central role in the development of DT, serving as the foundation for creating accurate virtual representations of physical systems. As shown in Table 1, existing approaches can be broadly grouped into three major categories, (a) data-driven, (b) physics-based, and (c) Physics-Informed Machine Learning, each contributing complementary strengths to the construction of high-fidelity virtual replicas. Data-driven approaches rely on heterogeneous data sources, including sensors, IoT devices, historical repositories, and contextual information [27]. Real-time streams ensure synchronization with the physical system, while historical and contextual data support trend analysis, predictive modeling, and situational awareness [28]. Common techniques include statistical analysis, machine learning, deep learning, feature engineering, simulation-based optimization, and streaming analytics [29]. Recent studies investigate few-shot learning strategies to address data scarcity and improve cross-domain generalization in fault diagnosis scenarios [30]. Complementary approaches based on domain generalization further aim to learn invariant representations that remain robust under varying operating conditions and unseen domains [31].

In parallel, physics-based modeling remains fundamental for high-fidelity DTs. These methods embed physical laws—e.g., conservation principles, thermodynamics, electromagnetism, or fluid dynamics—into mathematical formulations that capture system behavior [14]. Solving these formulations allows predicting responses under varying conditions, identifying vulnerabilities, and optimising design. Challenges arise in modeling nonlinear, multi-scale, and chaotic systems, which require approximation or model-order reduction techniques to balance fidelity and computational cost [32].

More recently, Physics-Informed Machine Learning (PIML) has emerged as a hybrid paradigm that augments learning-based models with physical priors. Techniques such as Physics-Informed Neural Networks (PINNs) [33], symbolic regression [34], hybrid FEM–ML [35], variational inference with physical constraints [36], and data assimilation demonstrate improved interpretability, generalization, and robustness, particularly when data availability is limited. Beyond numerical modeling, semantic modeling is gaining importance for enabling interoperability and reasoning across heterogeneous DT systems. Linked Data techniques allow distributed integration of information sources, while KG enriched with ontologies provide structured representations for classes, relations, and instances. Technologies such as RDF [37], SPARQL [38,39], and ontology languages like OWL with OWL 2 profiles [12,40] support semantic query, reasoning, and model consistency. For DT-specific modeling, the DTDL [41] offers a JSONLD-based schema for describing interfaces, telemetry, properties, and relationships, enabling composable and interoperable DT graphs.

While modeling- and semantics-oriented approaches constitute the computational core of DT, recent research has increasingly shifted towards architectural frameworks capable of orchestrating complex CPS and IIoT ecosystems in Smart Manufacturing and Industry 4.0. These include layered reference models, service-oriented DT platforms, cloud–edge con-

tinuum architectures, and composable microservice-based environments, which structure how data, models, and services are integrated, deployed, and governed—from shop-floor devices up to enterprise systems. The following sections review these recent Digital Twin architectures and outline the limitations they present, which the proposed DTCL aims to overcome. A first group of contributions focuses on reference models and high-level conceptual frameworks for smart factories and Digital Twins. The work in [42] introduces a Digital Twin-driven Smart Manufacturing paradigm based on a multi-dimensional reference model that links physical resources, data, models, services, and lifecycle activities within an integrated SM environment. A complementary perspective is offered in [43], which proposes a data-driven DT framework for Smart Manufacturing systems, emphasizing data acquisition, automated model generation, and simulation-based synchronization with the physical factory. Conceptual modeling efforts further extend this view to IoT-oriented environments, where layered DT architectures are defined from the Physical Layer up to cloud-based virtual space, analytics, and application layers, with edge devices collecting data and cloud services hosting the Digital Twin [28]. Survey-based analyses classify DT applications and highlight challenges related to semantic heterogeneity, data governance, and architectural fragmentation across industrial domains [1]. More recent reviews synthesize the key technologies underpinning DT-driven Smart Manufacturing—including sensing infrastructures, data-processing pipelines, model synchronization mechanisms, and the role of scalable cloud–edge platforms in supporting Industry 4.0 deployments [44]. Recent surveys analyze DT in IoT ecosystems, highlighting challenges related to interoperability, standardization, and the integration of heterogeneous data sources and systems [45]. Architectural studies further emphasize the layered nature of DT and the complexity arising from the interaction of data, models, services, and communication infrastructures in distributed environments [46]. Systematic analyzes also point to the growing role of IoT, AI, and cloud–edge computing paradigms, while identifying persistent challenges in scalability, interoperability, and system integration across large-scale DT deployments [47].

Table 1. Comparison of modeling approaches for Digital Twins.

Approach	Strengths	Limitations	Applications
Data-driven	Direct learning from data; adaptable; effective for real-time insights.	Needs large high-quality datasets; limited interpretability.	Predictive maintenance, anomaly detection [27,29].
Physics-based	High fidelity; interpretable; grounded in physical laws.	Computationally demanding; difficult when physics modeling is incomplete.	Simulation, stress analysis, failure prediction [14,32].
PIML	Merges physics knowledge with ML; improved generalization; robust with scarce data.	Complex design; requires domain expertise; training may be unstable.	PINNs, symbolic regression, hybrid FEM–ML [33–36].

A second group of contributions targets shop-floor and factory-level architectures that explicitly address distributed computation and real-time coordination. The framework in [48] proposes a DT-enabled GiMS for fixed-position assembly islands, combining unified digital representations at object, product, and system levels with cloud-based services and real-time synchronization mechanisms to support complex assembly operations. Building on cloud–edge integration, the reference architecture in [49] introduces a CFE collaboration model for DT-enabled smart factories, distributing computation across hierarchical layers to improve responsiveness, reduce data-transmission pressure, and support near real-time decision-making. Although these solutions advance the use of Digital Twins at the shop-

floor level and address latency-sensitive computation through hierarchical or cloud-backed deployments, they remain limited in terms of semantic interoperability, systematic module reuse, and automated validation workflows across heterogeneous DT. As such, they do not yet provide a unified, composable execution layer capable of governing distributed DT services along the full C2E continuum.

A third line of work investigates cross-domain surveys and system-level analyses of DT. Existing surveys provide broad perspectives on DT applications, architectures, and enabling technologies across multiple domains, highlighting challenges related to interoperability, lifecycle management, and the integration of heterogeneous data sources [1,50]. Recent studies further emphasize the complexity of DT ecosystems, including the interplay between data, models, communication infrastructures, and emerging paradigms such as IoT, AI, and cloud–edge computing. However, despite their breadth, these works remain largely high-level and do not systematically analyze architectural capabilities such as service orchestration, semantic capability discovery, and lifecycle governance across different platforms and implementations.

In parallel with research-oriented frameworks, several industrial COTS platforms have emerged to support the implementation of DT in real-world scenarios. Representative examples include Azure Digital Twins (https://learn.microsoft.com/en-us/azure/digital-twins/, accessed on 2 February 2026) and AWS IoT TwinMaker (https://docs.aws.amazon.com/iot-twinmaker/, accessed on 2 February 2026), which provide cloud-native environments for modeling digital assets and integrating heterogeneous data sources, as well as Siemens Insights Hub (https://www.siemens.com/en-gb/products/insights-hub/, accessed on 2 February 2026) and ThingWorx (https://www.ptc.com/en/products/thingworx, accessed on 2 February 2026), which offer industrial platforms combining data analytics, application enablement, and service integration capabilities. Additional solutions such as IBM Maximo (https://www.ibm.com/products/maximo, accessed on 2 February 2026) and Bosch IoT Suite (https://bosch-iot-suite.com/, accessed on 2 February 2026) focus on asset lifecycle management and large-scale IoT integration, while platforms like GE Vernova Asset Performance Management (APM)/Predix (https://www.gevernova.com/software/products/asset-performance-management, accessed on 2 February 2026) and Hitachi Lumada (https://www.hitachi.com/products/it/lumada/, accessed on 2 February 2026) provide industrial analytics ecosystems targeting predictive maintenance and operational optimization. Furthermore, technologies such as Eclipse Hono (https://www.eclipse.org/hono/, accessed on 2 February 2026) emphasize scalable device connectivity and data ingestion within IoT infrastructures, supporting the lower layers of DT implementations. While these platforms provide mature infrastructures for asset modeling, data integration, and real-time monitoring, they typically offer limited native support for semantic capability discovery, reusable service composition, and lifecycle-aware orchestration of distributed DT services.

Table 2 compares architectural approaches and reference models, focusing on features that are particularly relevant for the proposed DTCL rather than quantitative performance metrics, and including aforementioned industrial COTS platforms as well as research-oriented DT frameworks. This choice reflects the objective of the proposed DTCL, which is not to optimize task-specific performance (e.g., accuracy or latency), but to improve flexibility, composability, and lifecycle governance of DT applications.

The comparison is based on a qualitative analysis of publicly available documentation, reference architectures, and recent survey studies, focusing on the presence of architectural capabilities rather than quantitative performance evaluation. The reported features indicate whether a capability is natively supported by each platform or framework.

It is important to note that several industrial platforms provide some of these capabilities through external services or integrations, rather than as part of a unified, semantically enabled execution layer for DT services.

Table 2. Comparison of representative industrial COTS platforms and research DT frameworks. The comparison focuses on architectural and qualitative aspects rather than quantitative performance metrics (✓: supported, ✗: not supported).

Platform/Framework	Cloud -Native	Service Orchestration	Semantic Modeling	Capability Discovery	Lifecycle & Governance	DT Composability	Cloud —Edge
Industrial COTS platforms							
Azure Digital Twins	✓	✗	✓	✗	✓	✗	✓
AWS IoT TwinMaker	✓	✗	✓	✗	✓	✗	✓
Siemens Insights Hub	✓	✗	✓	✗	✓	✗	✓
ThingWorx	✓	✓	✓	✗	✓	✓	✓
IBM Maximo/Watson IoT	✓	✗	✗	✗	✓	✗	✓
GE Vernova APM/Predix	✓	✗	✗	✗	✓	✗	✓
Bosch IoT Suite/IoT Things	✓	✗	✓	✗	✓	✗	✓
Hitachi Lumada	✓	✗	✗	✗	✓	✗	✓
Eclipse Hono	✓	✗	✗	✗	✗	✗	✗
Research frameworks and reference models							
DT-driven SM Reference Model [42]	✗	✗	✗	✗	✓	✗	✗
Data-driven DT Framework [43]	✗	✗	✗	✗	✓	✗	✗
IoT-oriented DT Conceptual Model [28]	✓	✗	✗	✗	✓	✗	✓
DT-enabled GiMS [48]	✗	✗	✗	✗	✓	✗	✗
Cloud–Fog–Edge DT Smart Factory [49]	✓	✗	✗	✗	✓	✓	✓
DT-enabled SM Technologies Survey [44]	✓	✓	✗	✗	✓	✗	✓
Proposed DTCL	✓	✓	✓	✓	✓	✓	✓

The analysis highlights that, while industrial platforms provide mature solutions for data integration and asset modeling, and research frameworks offer conceptual foundations, none of the surveyed solutions jointly provide (i) a composable, microservice-based DT execution layer, (ii) explicit semantic modeling and capability discovery, and (iii) systematic lifecycle, validation, and governance support across the cloud-to-edge continuum. The proposed DTCL addresses this gap by offering a modular, semantically enabled, cloud-native control layer that orchestrates reusable DT services across heterogeneous SM scenarios.

3. Digital Twin Computing Layer Architecture

3.1. Architectural Overview

The DTCL constitutes the core architectural framework enabling the creation, orchestration, and deployment of DT applications. Its architecture follows a modular, composable, and cloud-native paradigm, ensuring scalability, interoperability, and domain-agnostic applicability across industrial, urban, and environmental domains. The DTCL adopts a three-tier logical model, comprising the user interface, Deploy Engine, and module catalog, supported by a distributed physical implementation that leverages containerization, multi-cloud infrastructures, and API-centric communication mechanisms. The design adheres to the Composable Digital Twin Architecture (CDTA) paradigm, allowing DT applications to be built from reusable and configurable services that interact dynamically through standardized interfaces. As illustrated in Figure 3, the proposed DTCL architecture is organized into three hierarchical sections (Digital Twin Developer Service, Digital Twin Computing Layer, and Service Catalog), each fulfilling a distinct role within the overall ecosystem. This structure enables scalability, interoperability, and seamless integration, in line with modern principles of composable and cloud-native system design. Each section includes the following DTCL components:

- User Interface (UI): The front-end environment for configuring, visualizing, and composing DT applications. Through an interactive dashboard, users can create projects, select modules from the Service Catalog, interconnect components, and deploy complete applications.
- Deploy Engine: The orchestration backbone responsible for distributing, versioning, and managing the lifecycle of DT instances. It automates provisioning, testing, and scaling processes using CI/CD pipelines and container orchestration tools (e.g., Kubernetes, Docker).
- Packaged Business Capabilities (PBCs) Service Catalog: A repository of standardized, self-contained microservices that represent reusable DT functionalities. Each module is containerized, configurable through YAML/JSON descriptors, and composable with other modules via REST, gRPC, or GraphQL APIs.

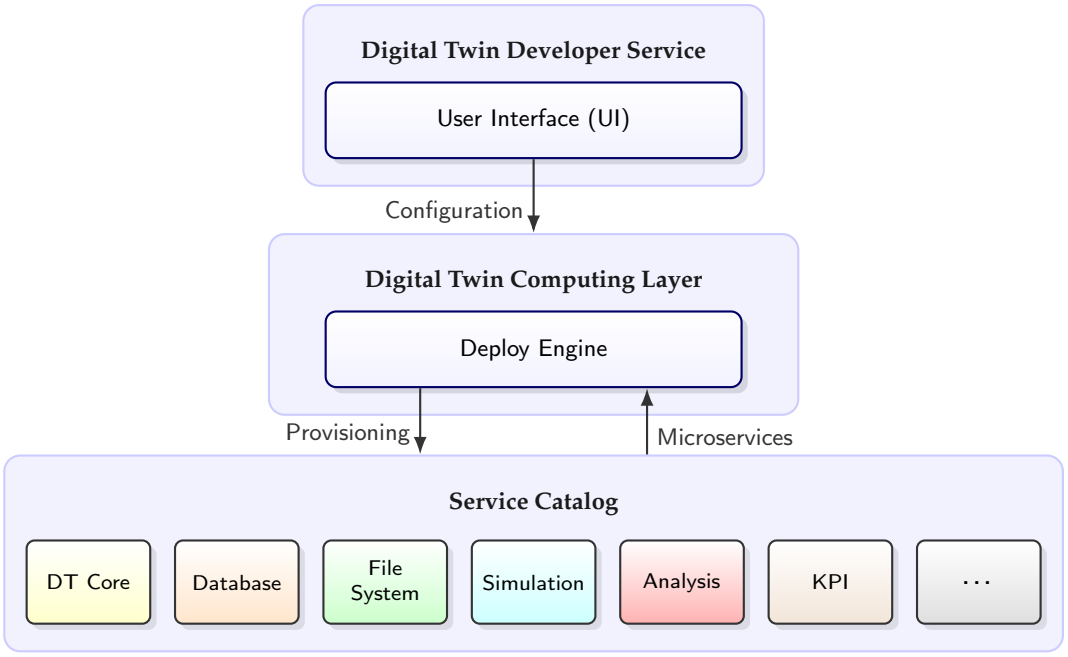


Figure 3. DTCL overview.

At the top of the architecture, the Digital Twin Developer Service provides the UI, serving as the primary interaction point between users and the underlying infrastructure. Through this interface, developers and engineers can design, configure, and deploy DT applications without deep technical knowledge of back-end operations. The UI abstracts system complexity by offering an intuitive environment for project management, service configuration, and visualization of simulation results. Acting as a design and orchestration hub, it allows users to access components from the Service Catalog, logically assemble them, and establish data connections between modules. Once the configuration is complete, the deployment request is passed to the Deploy Engine. It manages execution and resource allocation and coordinates all activities required to build and operate a DT solution, e.g., integration, configuration, and lifecycle of selected services, ensuring consistent operation across modules. The Deploy Engine leverages containerization and cloud-native techniques to provide dynamic scalability, optimal performance, and automated version control. By enforcing standardized configuration rules and service dependencies, it minimizes human error and facilitates reproducibility. Conceptually, it acts as an execution orchestrator managing interactions among modular entities through APIs and message-passing mechanisms also ensuring that data from physical devices, simulations, and analytics flows seamlessly across modules. At the foundation lies the Service Catalog, which hosts a set

of modular components that can be dynamically combined to construct DT applications. Each component is implemented as an independent microservice, interoperable through standardized interfaces. The catalog includes key modules such as:

1. DT Core: The fundamental engine for creating, defining, and synchronizing DT entities using standardized modeling languages like DTDL.
2. Database: Manages static configuration data and dynamic time-series data from sensors or simulations.
3. File System: Handles large data objects, including 3D models and simulation outputs.
4. Simulation: Executes predictive and physics-based models to replicate real-world behavior.
5. Analysis: Performs analytical and machine learning operations for insight extraction and process optimization.
6. KPI Module: Computes and monitors performance indicators derived from DT data.

This modular design ensures flexibility by allowing developers to select only the components needed for specific use cases. Each service can be deployed independently, promoting reusability and reducing redundancy. Moreover, the architecture supports continuous evolution by enabling the seamless integration of new modules and updates without impacting existing deployments. Overall, the architecture follows a top-down integration model, where users operate at the abstraction level of the Developer Service, while the Deploy Engine and Service Catalog manage orchestration, computation, and resources at lower levels. The bidirectional data flow allows deployment commands to propagate downward and processed results or analytics to return upward for visualization and decision support. The architecture is further detailed in terms of its deployment model and execution across distributed environments in Section 3.3.

3.2. Core Components and Enabling Technologies

The DTCL represents a concrete realization of the composable and data-driven paradigms that increasingly characterize research and industrial innovation in Smart Manufacturing and Industry 4.0. Designed as a core subsystem of a larger digital platform, the DTCL provides the technological and methodological framework for constructing and orchestrating DT applications from modular, interoperable, and reconfigurable components. Its conceptual foundations align closely with the themes outlined in the data-driven Digital Twin for Smart Manufacturing and Industry 4.0 Special Issue: both emphasize data architectures, predictive modeling, and system interoperability as key enablers of intelligent, adaptive, and sustainable industrial ecosystems. The DTCL adopts the principles of the CDTA, translating them into a software infrastructure where each functionality is encapsulated in independent modules, e.g., data ingestion, analytics, visualization, or system control. Instead of developing each application as a static and monolithic system, the composable paradigm enables users to build complex DTs by assembling pre-existing services. This approach reduces redundancy, accelerates development cycles, and promotes a continuous exchange of data and models among heterogeneous domains. The architecture of the DTCL is organized around three essential elements:

- reusable components, providing the functional building blocks of the system;
- a Search Engine, facilitating the discovery and composition of such components;
- a Service Validator, which guarantees coherence, non-redundancy, and compliance of new modules within the platform.

Together, these elements create an ecosystem that supports the creation, validation, and reuse of DT services across sectors. The resulting environment directly supports the special issue's focus on data-driven architectures and predictive analytics, as it transforms raw industrial data into actionable knowledge through modular and scalable computational structures.

3.2.1. Reusable Components

Within the DTCL, reusable components form the backbone of a composable and scalable environment for developing DT applications. Each component is conceived as an autonomous and interoperable unit that can be combined with others to build complex systems. This modular approach supports flexibility, accelerates development, and reduces redundancy, addressing one of the central challenges in DT research, moving from domain-specific prototypes to adaptable, cross-sector solutions. Reusable components are designed according to a set of guiding principles that ensure robustness and interoperability:

- Reusability: components are domain-agnostic and based on standardized interfaces and APIs, allowing integration into heterogeneous DT architectures.
- Configurability: each module can be tailored through adjustable parameters such as communication protocols, data sampling rates, or algorithmic settings, without modifying the underlying code.
- Extensibility: explicit extension points enable the introduction of new functionalities without affecting system stability.
- Information hiding: internal logic remains encapsulated, exposing only what is necessary for secure interaction.
- Separation of concerns: each component performs a specific function, simplifying maintenance and promoting specialization.
- Loose coupling: communication between modules relies on asynchronous or API-based exchanges, minimizing dependencies and facilitating distributed deployment.

These principles make the DTCL a living ecosystem that can evolve with minimal disruption. Components are grouped into six functional categories reflecting the full Digital Twin lifecycle:

- Data Services—acquisition and transformation of physical and virtual data;
- Integration Services—communication among Digital Twins and external systems;
- Intelligence Services—artificial intelligence and machine learning for prediction and optimization;
- User Experience Services—visualization and interactive interfaces in 2D, 3D, or immersive formats;
- Management Services—orchestration, monitoring, and lifecycle control;
- Trustworthiness Services—data security, privacy, and reliability.

These components define a DTCL platform that promotes reuse and standardization while remaining adaptable to new industrial domains. Their modular design fosters a sustainable development model where innovation is achieved through composition rather than reconstruction. In this sense, the DTCL exemplifies how composable architectures can bridge the gap between isolated DTs and interoperable ecosystems, paving the way for data-driven, intelligent, and resilient industrial applications.

3.2.2. Search Engine

The Search Engine represents a fundamental component of the DTCL, designed to support the semantic discovery, classification, and composition of reusable services within the platform. Its primary objective is to simplify navigation across a heterogeneous and evolving Service Catalog, while ensuring that service selection and integration remain coherent with the architectural principles of the DTCL and the CDTA pattern.

Rather than relying solely on textual search, the DTCL Search Engine adopts a semantically enriched classification model based on a structured system of tags. This system is directly aligned with the Digital Twin capabilities taxonomy defined by the Digital Twin Consortium, which provides a shared conceptual framework for categorizing DT function-

alities. In particular, services are classified according to six high-level macro-categories: Data Services, integration, intelligence, User Experience, management, and trustworthiness. Each macro-category represents a tag family that groups together services with similar functional responsibilities. Within each family, more granular tags are used to describe specific capabilities, allowing a fine-grained semantic characterization of each service while preserving a common vocabulary across the platform.

Every reusable component registered in the DTCL Service Catalog is associated with a semantic service profile, which includes:

- a set of macro-category tags describing the functional capabilities of the service;
- a textual description outlining the service purpose and scope;
- metadata related to exposed interfaces and supported communication patterns;
- information about dependencies on other services or infrastructural components;
- references to configuration parameters and deployment constraints.

This structured metadata enables the Search Engine to act as a semantic index rather than a simple lookup tool. By grounding service descriptions in a shared taxonomy, the Search Engine ensures consistency in classification and facilitates interoperability across heterogeneous applications and domains. To clarify the semantic approach, we first outline the underlying ontology design and then show how it supports service discovery and composition through a concrete example.

From an ontology design perspective, a reusable DT component is described through a semantic profile combining structured metadata, tag-based annotations, and ontology-aligned representations. In this model, the central concept is `dtcl:Service`, which represents a reusable runtime service within the DTCL. Each service is linked to a Digital Twin interface through `dtcl:bindsTo`, exposes telemetry streams through `dtcl:exposesTelemetry`, references structural properties through `dtcl:hasProperty`, and is associated with one or more functional capabilities through `dtcl:hasCapability`. Services are further annotated with tags via `dct:subject`, while tags are related to capabilities through `skos:related`.

The semantic layer is aligned with existing standards and ontologies. In particular, DTDL [41] is used to describe Digital Twin interfaces, telemetry, and properties; AAS [51] is used to represent industrial asset models through concepts such as `aas:Submodel`; domain ontologies such as SEAS [52] are used to represent systems and observable processes; GeoNames provides location-related semantics; and standard LD [53] vocabularies for RDF [37] and OWL [54] support annotation and classification, such as RDFS [55], SKOS [56] and Dublin Core [57]. In this way, the ontology schema combines DTCL-specific concepts with reusable external vocabularies without enforcing a monolithic representation.

The ontology-oriented schema underlying this model is shown in Figure 4 and defines the semantic structure used for service description. It highlights how a service is semantically connected to its interface, telemetry, properties, asset-level alignment, capability, and tag-based annotations. This schema provides the structural basis on which the Search Engine operates.

A simple example clarifies how this representation supports service discovery. Consider a request for a predictive maintenance service performing real-time anomaly detection on vibration telemetry acquired from an industrial motor. Such a request can be represented as a structured semantic request grounded in the same core ontology used for service annotation. In particular, the query is expressed through the same semantic dimensions used in the catalog, namely tag-based annotations (`dct:subject`), functional capabilities (`dtcl:hasCapability`), and exposed telemetry (`dtcl:exposesTelemetry`), and may be further refined by query-specific constraints such as preferred deployment conditions.

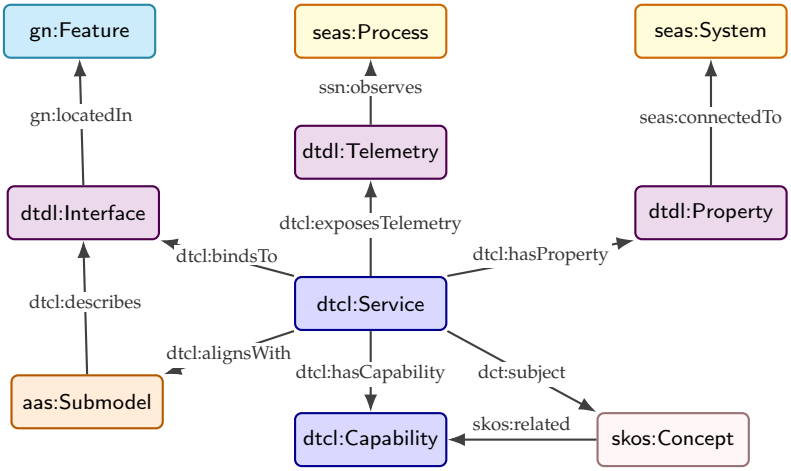


Figure 4. Simplified ontology-oriented semantic model for reusable services in the DTCL. Colors identify semantic layers and vocabularies: DTDL concepts (violet), AAS concepts (orange), domain ontologies such as SEAS (yellow), contextual vocabularies such as GeoNames (cyan), DTCL-specific concepts (blue), and tag taxonomy elements based on SKOS (pink).

Digital Twin interfaces and telemetry types referenced in the JSON-LD examples correspond to DTDL models.

Within this framework, service discovery is driven by semantic matching, which combines tag-based filtering with ontology-aware semantic similarity. The request in Listing 1 first filters candidate services through shared annotations such as `dct:subject`, `dtcl:hasCapability`, and `dtcl:exposesTelemetry` within the semantic matching phase.

Listing 1. Example JSON-LD request for service discovery.

```
{
  ''@context'': {
    ''dtcl'': ''https://example.org/dtcl#'',
    ''dtdl'': ''https://example.org/dtdl#'',
    ''dct'': ''http://purl.org/dc/terms/''
  },
  ''@id'': ''dtcl:Query01'',
  ''@type'': ''dtcl:ServiceQuery'',
  ''dct:subject'': ''dtcl:PredictiveMaintenanceTag'',
  ''dtcl:hasCapability'': ''dtcl:AnomalyDetection'',
  ''dtcl:exposesTelemetry'': ''dtdl:VibrationTelemetry'',
  ''dtcl:preferredDeployment'': ''dtcl:Edge''
}
```

In addition, service discovery is guided by semantic descriptors derived from DTDL/AAS models, which provide structural and interface-level information for candidate services. The Search Engine then produces a set of candidate services, which are subsequently ranked and passed to the discovery and compatibility-check phases. In the examples in Listing 2 and Listing 3, respectively, `dtcl:AnomalyDetectionService01` is ranked as the most relevant result, while `dtcl:BatchMaintenanceAnalytics02` is only partially compatible.

Listing 2. Example JSON-LD description of a matching service.

```
{
  ''@context'': {
    ''dtcl'': ''https://example.org/dtcl#'',
    ''dtdl'': ''https://example.org/dtdl#'',
    ''aas'': ''https://example.org/aas#'',
    ''dct'': ''http://purl.org/dc/terms/'',
```

```
    ‘‘ssn’’: ‘‘http://www.w3.org/ns/ssn/’’,
    ‘‘seas’’: ‘‘https://w3id.org/seas/’’
  },
  ‘‘@id’’: ‘‘dtcl:AnomalyDetectionService01’’,
  ‘‘@type’’: ‘‘dtcl:Service’’,
  ‘‘dtcl:bindsTo’’: ‘‘dtdl:MotorInterface’’,
  ‘‘dtcl:exposesTelemetry’’: ‘‘dtdl:VibrationTelemetry’’,
  ‘‘dtcl:hasProperty’’: ‘‘dtdl:HealthState’’,
  ‘‘dtcl:hasCapability’’: ‘‘dtcl:AnomalyDetection’’,
  ‘‘dtcl:alignsWith’’: ‘‘aas:MotorSubmodel’’,
  ‘‘dct:subject’’: ‘‘dtcl:PredictiveMaintenanceTag’’,
  ‘‘ssn:observes’’: ‘‘seas:Process’’,
  ‘‘seas:connectedTo’’: ‘‘seas:System’’
}
```

Listing 3. Example JSON-LD description of a partially matching service.

```
{
  ‘‘@context’’: {
    ‘‘dtcl’’: ‘‘https://example.org/dtcl#’’,
    ‘‘dct’’: ‘‘http://purl.org/dc/terms/’’
  },
  ‘‘@id’’: ‘‘dtcl:BatchMaintenanceAnalytics02’’,
  ‘‘@type’’: ‘‘dtcl:Service’’,
  ‘‘dtcl:hasCapability’’: ‘‘dtcl:MaintenanceAnalytics’’,
  ‘‘dct:subject’’: ‘‘dtcl:PredictiveMaintenanceTag’’
}
```

The same semantic representation also supports service composition. Once candidate services have been validated through the compatibility-check phase, they can be composed into higher-level workflows when their interfaces, telemetry streams, properties, and capabilities are semantically and functionally compatible.

Telemetry data are associated with observable processes through `ssn:observes`, while properties are connected to system-level entities through relations such as `seas:connectedTo`.

The operational logic of this process is illustrated in Figure 5. Starting from the DT service description, semantic annotations and tags are used to support filtering and semantic matching. The resulting candidate services are then passed to the discovery and compatibility-check phases, which determine whether the selected services can be consistently composed.

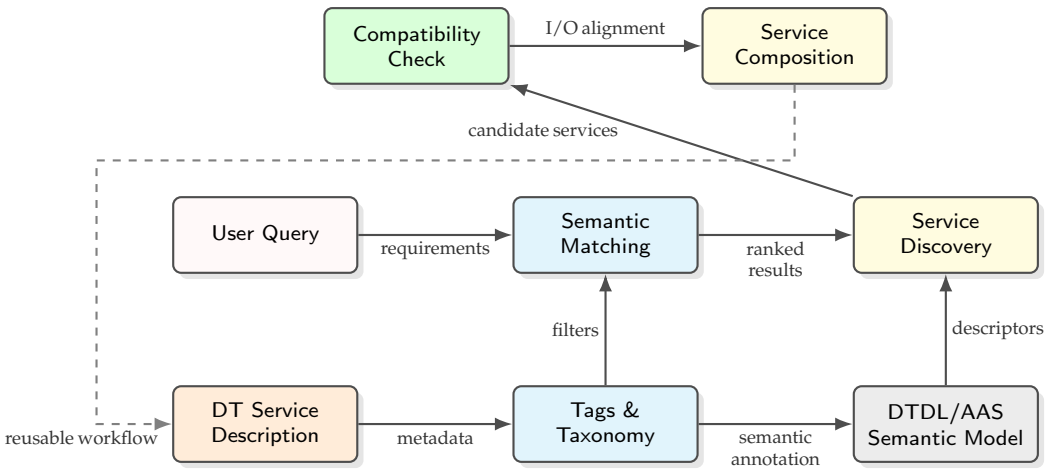


Figure 5. Semantic workflow for service description, discovery, and composition in the proposed DTCL.

The Search Engine implements a hybrid retrieval strategy that combines full-text search with semantic filtering based on tags. Users may query the catalog using free-text keywords, select one or more macro-categories, or combine textual queries with tag-based filters. This flexibility allows users to explore the catalog at different levels of abstraction, from high-level capability discovery to precise identification of services matching specific functional requirements.

Search results are ranked according to relevance criteria that take into account both textual similarity and semantic alignment with the selected tags. This approach improves the precision of search outcomes and reduces the effort required to identify suitable components within large and heterogeneous catalogs.

Beyond discovery, the semantic structure of the Search Engine directly supports architectural linking and service composition within the DTCL. By explicitly encoding service capabilities and dependencies, the Search Engine enables:

- identification of complementary services across different macro-categories;
- verification of compatibility among services based on declared interfaces and interaction patterns;
- awareness of required supporting components during application assembly.

In this way, the Search Engine contributes to reducing architectural inconsistencies and promotes valid compositions aligned with DTCL design principles. The information it provides can be leveraged by higher-level orchestration mechanisms—such as the Deploy Engine—to automate deployment workflows and ensure that selected services can be correctly interconnected.

By transforming the Service Catalog into a semantically organized repository of capabilities, the Search Engine plays a central role in enabling reuse, preventing redundancy, and supporting the controlled evolution of the DTCL. It acts as a bridge between human-driven design activities and machine-supported orchestration processes, ensuring that service composition remains transparent, scalable, and consistent over time.

Overall, the Search Engine constitutes a key enabler of the composable paradigm within the DTCL, providing the semantic foundation necessary to move from isolated reusable components toward coherent, interoperable, and evolvable Digital Twin ecosystems.

3.2.3. Service Validator

The Service Validator represents the quality and governance layer of the DTCL. Its main purpose is to assess and approve new services before they are added to the platform, ensuring that each contribution is technically sound, non-redundant, and consistent with the DTCL architecture. In doing so, it complements the Search Engine, which it uses to identify potential overlaps or similar components already present in the catalog. When a developer submits a new module, the Validator automatically compares its description, functionality, and metadata against existing entries. If similarities are detected, a list of comparable services is returned, guiding the contributor toward refinement or integration. If the component is unique, the system confirms its eligibility and produces a standardized documentation template summarizing key information such as scope, main functions, architecture overview, and associated tags. The validation process is handled through a dedicated interface that supports both automation and expert review. Automated checks guarantee efficiency, while human oversight ensures that new services align with the technical and conceptual vision of the DTCL. By enforcing documentation standards and preventing redundancy, the Service Validator transforms the DTCL catalog into a curated and evolving ecosystem. It promotes the reuse of existing solutions, encourages incremental innovation, and ensures that each addition strengthens the platform's coherence and interoperability across industrial domains.

3.3. Deployment in the Cloud–Edge Continuum

Building upon the architectural components described above, the DTCL adopts a composable deployment model that governs how application functionalities are instantiated and executed across distributed environments.

Within this model, PBCs represent the fundamental deployment units. Each PBC encapsulates a cohesive set of microservices and APIs implementing a specific functional capability, enabling Digital Twin applications to be constructed as compositions of interoperable and independently deployable modules. This abstraction allows complex systems to be decomposed into modular building blocks, supporting incremental evolution, reconfiguration, and reuse without affecting overall system stability.

The deployment and lifecycle management of these components are orchestrated through a cloud-native control plane, which leverages Kubernetes (Kubernetes: https://kubernetes.io/, accessed on 2 February 2026) as the execution substrate and Crossplane (Crossplane: https://www.crossplane.io/, accessed on 2 February 2026) as an abstraction layer for infrastructure management. This combination enables a unified Infrastructure-as-Code (IaC) approach, where both infrastructure and application resources are defined declaratively and managed consistently across heterogeneous environments.

A defining characteristic of the DTCL is the adoption of policy-driven deployment strategies across the cloud–edge continuum. Rather than statically assigning components to specific environments, deployment decisions are dynamically determined based on operational constraints such as latency requirements, data locality, computational intensity, and resilience considerations.

In greater detail, the DTCL adopts a policy-driven placement strategy across the cloud–edge continuum characterized by:

- Edge-first allocation: latency-sensitive services such as real-time inference, anomaly detection, and control logic are deployed on edge nodes using a first-fit strategy based on proximity to data sources and available resources.
- Cloud fallback: when edge resources are insufficient or saturated, services are redirected to cloud infrastructure, ensuring continuity of execution at the cost of higher latency.
- Cloud allocation: computationally intensive but latency-tolerant services such as large-scale simulation, batch analytics, or model training are directly assigned to cloud resources, leveraging elastic compute capacity.

From an operational perspective, this strategy defines a two-tier decision framework combining locality-aware placement for real-time services with resource-driven offloading for high-computation workloads. It balances conceptual simplicity with adherence to practical deployment patterns in cloud–edge systems, providing a baseline for more advanced optimization mechanisms to be envisioned in future work.

This policy-driven approach seeks an optimal placement of application components either in centralized cloud infrastructures or in distributed edge nodes, ensuring efficient execution in scenarios characterized by geographically dispersed data sources and real-time processing needs. To support this dynamic behavior, the control plane continuously reconciles the desired and actual system states, automatically handling configuration drifts, failures, and scaling operations. This reconciliation mechanism enables a self-adaptive infrastructure capable of maintaining consistency and operational stability across hybrid and multi-cloud deployments.

By combining composable application design with policy-based infrastructure orchestration, the DTCL provides a unified framework that bridges application logic and distributed execution environments. This integration enables the realization of scalable,

resilient, and interoperable Digital Twin ecosystems, capable of operating seamlessly across the cloud–edge continuum.

4. Toward Practical Applications

4.1. Proposed Implementation

This section describes the concrete implementation of the proposed DTCL architecture, detailing the technologies, components, and interaction mechanisms enabling its operation in distributed environments.

The DTCL is implemented as a cloud-native, microservice-based platform designed to support the deployment and orchestration of composable Digital Twin applications across heterogeneous infrastructures. The architecture is centered around a control layer, referred to as the Head Quarter (HQ), which coordinates application lifecycle management, infrastructure provisioning, and runtime orchestration.

Application components are deployed as containerised microservices within Kubernetes clusters, enabling scalability, portability, and fault isolation. The HQ cluster manages back-end services responsible for orchestrating application components and coordinating their execution.

From a functional perspective, the system is structured into multiple layers. The presentation layer consists of a web-based interface that provides users with capabilities for infrastructure management, service discovery, application composition, and system monitoring. This interface interacts with back-end services through RESTful APIs, ensuring a decoupled architecture and enabling independent evolution of front-end and back-end components.

The application layer is composed of specialized microservices organized according to their functional roles, including orchestration services, identity and access management modules, notification services, and licensing components. A central Catalog Manager is responsible for managing the lifecycle of application components, maintaining consistency between configuration definitions and deployed resources, and enabling dynamic composition of Digital Twin applications.

Inter-service communication follows a hybrid interaction model that combines synchronous API-based communication with asynchronous event-driven messaging. An event streaming backbone based on Apache Kafka (Kafka: https://kafka.apache.org/, accessed on 2 February 2026) enables loose coupling between services, supporting scalability, resilience, and distributed processing. This approach allows system components to evolve independently while maintaining consistent data exchange mechanisms. In practice, these services are organized as interoperable modules exposing REST endpoints and event streams, enabling the construction of end-to-end application pipelines integrating data ingestion, processing, orchestration, and analytics components.

Data management is implemented through domain-specific persistence layers, typically based on relational databases, which store application data, service metadata, and configuration information. The separation of data domains ensures maintainability and facilitates the independent evolution of system components.

Infrastructure and application configurations are managed through versioned configuration files, enabling automated provisioning and consistent deployment.

Finally, the platform integrates an observability layer providing monitoring, logging, and distributed tracing capabilities. These features enable continuous assessment of system performance, facilitate anomaly detection, and support troubleshooting in complex distributed environments. The implementation described in this section is concretely instantiated in the Smart Mobility case study presented in the following subsection.

4.2. Case Studies

To demonstrate the practical applicability and generality of the proposed DTCL, two complementary case studies are presented, targeting distinct domains and operational contexts. The objective of this section is to validate the DTCL not only as a conceptual architectural framework, but as a deployable and extensible solution capable of supporting real-world Digital Twin applications.

The first case study focuses on Smart Manufacturing, a domain characterized by tightly coupled cyber-physical systems, high-frequency telemetry, and stringent reliability requirements. This scenario highlights the capability of the DTCL to enable predictive maintenance through the integration of heterogeneous industrial assets and advanced analytics services. The second case study addresses Smart Mobility, representing a large-scale, dynamic, and data-intensive environment where multiple agents, real-time data streams, and scenario-based simulations coexist. This use case demonstrates how the DTCL can be instantiated in urban contexts to support decision-making processes through KPI-driven evaluation of alternative mobility configurations.

Together, these case studies provide complementary evidence of the DTCL's flexibility, scalability, and composability across domains with significantly different requirements, thereby validating its role as a general-purpose architectural layer for next-generation Digital Twin systems.

4.2.1. Smart Manufacturing Case Study

The DTCL provides a structured architectural foundation for the implementation of predictive maintenance in Smart Manufacturing environments. Its primary objective is to enable a seamless integration between Operational Technology (OT) and Information Technology (IT) through a modular, service-oriented, and composable architecture capable of transforming heterogeneous industrial data into actionable intelligence. By design, the DTCL addresses the limitations of monolithic Digital Twin solutions, which often lack scalability, flexibility, and long-term evolvability.

Modern production systems are inherently complex, comprising heterogeneous assets such as robotic arms, conveyors, PLC-controlled machines, and embedded sensors that generate high-frequency telemetry. Traditional maintenance strategies, either corrective or schedule-based, are insufficient in such contexts, as they fail to capture the dynamic interdependencies between operational conditions, asset degradation, and production variability. The DTCL enables the construction of a continuously synchronized digital representation of the production line, supporting predictive diagnostics, performance forecasting, and data-driven decision-making.

From an architectural standpoint, the DTCL adopts a layered and composable structure. As illustrated in Figure 6, the Smart Manufacturing case study is obtained by selecting and orchestrating a subset of reusable services from the DTCL component catalog. This configuration separates foundational Digital Twin services, communication interfaces, and higher-level intelligence components, while preserving the same architectural principles of the overall platform.

At the foundational level (Layer 1), the DT Core and the persistent data store maintain the authoritative digital state of physical assets. Each machine is modeled using the DTDL, which defines its structure, telemetry, and relationships within the production system. This semantic abstraction ensures consistency across simulation, analytics, and visualization services.

Layer 2 provides standardized integration and communication mechanisms through REST APIs, MQTT brokers, and GraphQL interfaces. These services decouple data producers from consumers, enabling interoperability across heterogeneous OT environments and

facilitating integration with external systems. Real-time telemetry about temperature, vibration, torque, and energy consumption is ingested and stored as time-series data, forming the basis for both monitoring procedures and historical analysis.

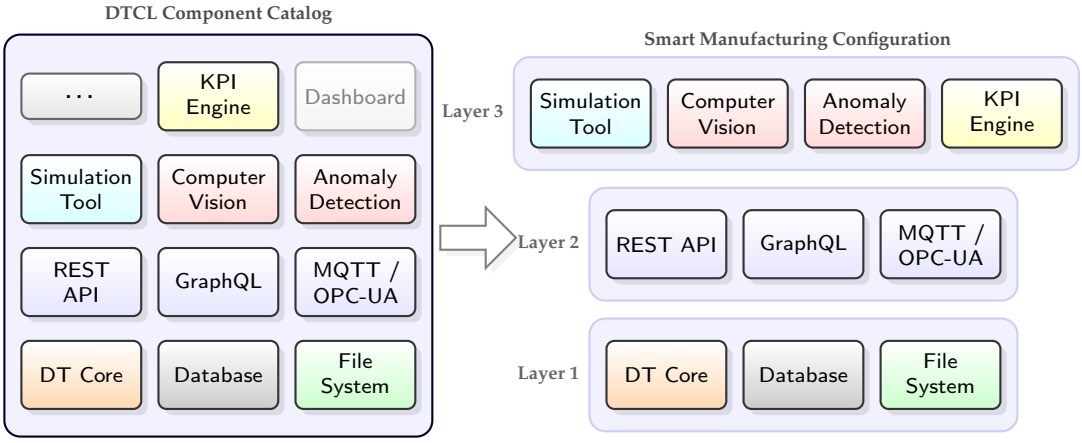


Figure 6. Selection and orchestration of DTCL components for the Smart Manufacturing case study. The figure highlights how the manufacturing scenario is obtained by selecting and composing a subset of reusable services from the DTCL component catalog.

At the upper level (Layer 3), simulation and intelligence services operate on top of the DT Core. Physics-based simulation tools and data-driven models are employed to reproduce equipment behavior under varying operational scenarios, supporting what-if analyses and degradation forecasting. In parallel, analytical services leverage machine learning techniques for anomaly detection and predictive maintenance, including the estimation of Remaining Useful Life (RUL) and Mean Time Between Failures (MTBF). The modular nature of these services allows them to be independently deployed, updated, or replaced without impacting the overall system.

The figure makes explicit that the manufacturing application is not implemented as a monolithic stack, but as a domain-specific composition of reusable DTCL components, selected according to the requirements of predictive maintenance in a microfactory environment.

The practical applicability of the DTCL architecture is reflected in industrial deployments that adopt a similar compositional paradigm. This composability enables incremental evolution of predictive maintenance applications and supports reuse across different assets, production lines, or industrial domains. A representative example is the Digital Twin–Enabled Microfactory developed by NTT DATA and presented within the Digital Twin Consortium technology showcase (Digital Twin-Enabled Microfactory for Smart Manufacturing: https://www.digitaltwinconsortium.org/initiatives/technology-showcase/micro-factory/, accessed on 2 February 2026). In this case, a physical microfactory equipped with industrial robots, PLCs, and IoT sensors is mirrored by a service-based Digital Twin architecture closely aligned with the layered DTCL model. Telemetry is acquired through OPC-UA and MQTT gateways, persisted in a time-series database, and managed by a central Digital Twin core. Simulation services and computer vision modules are integrated as independent components to support production-line fault detection and predictive maintenance, while communication interfaces remain decoupled from core logic.

To provide a more concrete understanding of how the DTCL operates in a real industrial context, the behavior of the Digital Twin–Enabled Microfactory can be described through a representative operational scenario.

As shown in Figure 7, the DTCL supports a closed-loop operational logic in which physical telemetry is continuously transformed into predictive indicators and maintenance decisions through the coordinated interaction of modular services.

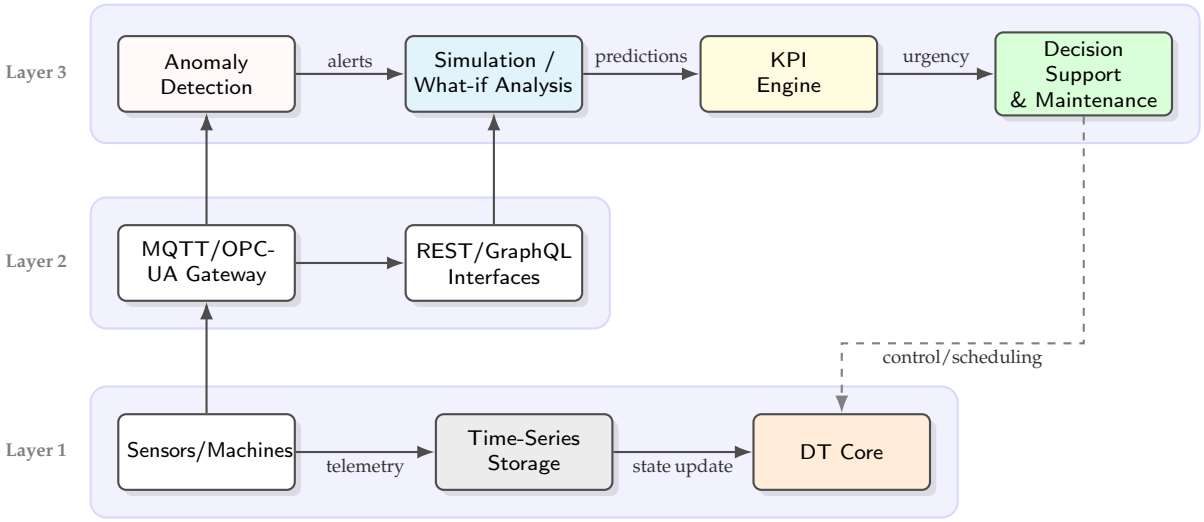


Figure 7. Operational workflow of the DTCL in the Smart Manufacturing case study.

Consider a production line composed of robotic arms and conveyor systems assembling mechanical components. During normal operation, each physical asset continuously streams telemetry data (e.g., vibration, temperature, torque, and energy consumption) through OPC-UA and MQTT gateways. These data are ingested in real time by the Integration Layer (Layer 2) and forwarded to the DT Core (Layer 1), where they are stored as time-series data and used to update the digital representation of each asset.

At this stage, the DT Core maintains a synchronized state between the physical and virtual systems, enabling a consistent and continuously updated model of the production line. This state becomes the input for higher-level services operating in Layer 3.

When an anomalous pattern begins to emerge (e.g., an increase in vibration on a robotic joint), the telemetry stream triggers the activation of analytical services. A machine learning-based anomaly detection module processes the incoming data and identifies deviations from normal operating conditions. This module interacts with the DT Core to retrieve historical data and contextual information, improving the robustness of the prediction.

In parallel, a simulation service is invoked to evaluate the potential evolution of the detected anomaly under different operating conditions. By leveraging physics-based or hybrid models, the system performs what-if analyses to estimate the degradation trajectory of the component and compute predictive indicators such as RUL.

The outputs of analytical and simulation services are then aggregated and passed to the KPI module, which computes performance indicators such as failure probability, expected downtime, and maintenance urgency. These results are propagated upward through the DTCL and made available to visualization and decision-support interfaces.

At this point, the DTCL enables a closed-loop decision process. Based on the computed KPIs, the system can either notify human operators or trigger automated actions through integration services, such as scheduling maintenance interventions or adjusting operating parameters to mitigate the risk of failure.

From a system perspective, this workflow highlights how the DTCL orchestrates the interaction between data ingestion, Digital Twin synchronization, analytics, simulation, and decision-making services. Each component operates as an independent microservice, yet contributes to a coherent end-to-end pipeline, demonstrating the effectiveness of the composable architecture in managing complex industrial processes.

The microfactory deployment demonstrates how simulation tools, analytics services, and visualization components can be composed and orchestrated without altering the underlying architectural structure. Observed outcomes about early fault detection, reduced

downtime, and adaptability to multiple industrial scenarios provide concrete evidence that the DTCL architecture can be instantiated effectively in real manufacturing contexts. Overall, the DTCL establishes a robust and extensible architectural foundation for predictive maintenance in Industry 4.0. Its layered, service-oriented, and composable design enables continuous synchronization between physical and digital systems, seamless integration with enterprise platforms, and long-term evolution of analytical capabilities. These characteristics position the DTCL as a practical enabler of adaptive, resilient, and data-driven manufacturing systems, setting the stage for the validation process discussed in the following section.

4.2.2. Smart Mobility Case Study

While the Smart Manufacturing scenario demonstrates the applicability of the DTCL in industrial environments, the architecture is inherently domain-agnostic and can be instantiated across different application domains. To validate its flexibility and generality, it has also been implemented in a Smart Mobility scenario focused on urban public transport planning.

The developed prototype enables decision-makers to simulate and evaluate alternative mobility configurations through a Digital Twin representation of the transportation network. The system integrates heterogeneous data sources, including GTFS datasets and geospatial information derived from OSM (OSM: https://www.openstreetmap.org/, accessed on 2 February 2026), enabling the reconstruction of routes, stops, and service schedules within a consistent virtual environment.

From an architectural perspective, the Smart Mobility prototype follows the composable principles defined by the DTCL. The system is structured as a set of loosely coupled microservices, each encapsulating a specific functional capability. The back-end layer is implemented using FastAPI (FastAPI: https://fastapi.tiangolo.com/, accessed on 2 February 2026) services, responsible for data ingestion, scenario management, and KPI computation, while a PostgreSQL (PostgreSQL: https://www.postgresql.org/, accessed on 2 February 2026)/PostGIS (PostGIS: https://postgis.net/, accessed on 2 February 2026) database is used for persistent storage of both static configuration data and dynamic spatial information.

The composable nature of the DTCL becomes even more evident in the Smart Mobility scenario. As shown in Figure 8, the same platform logic is preserved, but the selected and orchestrated components differ from the manufacturing configuration in order to satisfy the requirements of urban public transport planning.

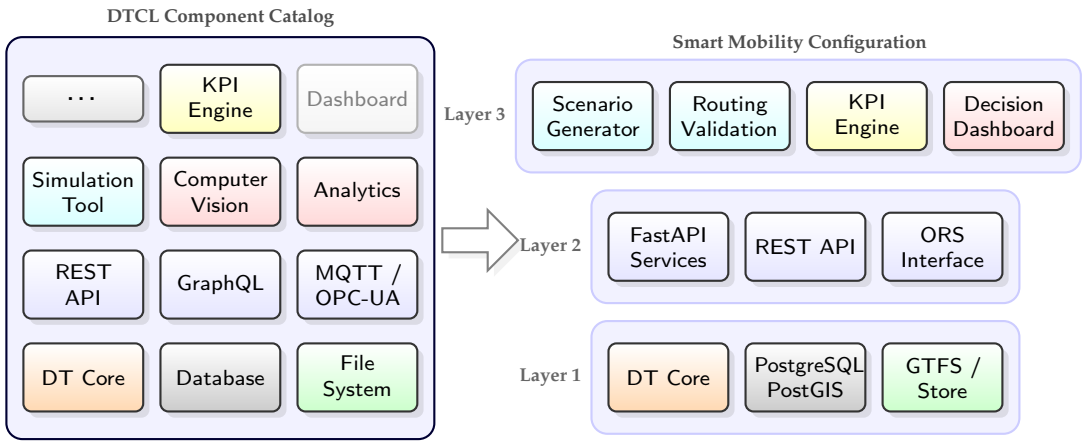


Figure 8. Selection and orchestration of DTCL components for the Smart Mobility case study. The figure shows how a different subset of reusable DTCL services is configured to support urban public transport planning, scenario simulation, routing validation, and KPI-driven decision support.

Routing and network analysis functionalities are integrated through external services such as ORS (ORS: https://openrouteservice.org/, accessed on 2 February 2026), enabling the validation of route feasibility and the computation of travel times and accessibility metrics. These components are deployed as containerized services within the DTCL environment, ensuring portability and scalability. The Digital Twin model enables the definition of a baseline scenario derived from the original GTFS dataset, which is then modified through the introduction of scenario deltas representing changes in routes, frequencies, or service configurations. Each modified scenario is processed through a simulation pipeline that updates routing information, validates network consistency, and computes a set of KPI.

The evaluation framework is KPI-driven and supports quantitative comparison between baseline and simulated scenarios. The considered indicators include accessibility metrics (e.g., service coverage and reachability), operational performance (e.g., travel time and waiting time), and sustainability-related measures (e.g., estimated CO_2 emissions). This approach enables data-driven assessment of mobility interventions and supports informed decision-making. Overall, this case study demonstrates how the DTCL architecture can be concretely instantiated in a real-world scenario beyond industrial contexts, highlighting its flexibility, composability, and applicability to data-intensive, distributed, and simulation-driven domains such as Smart Cities and urban mobility.

4.3. Framework Validation

The validation process described in this section is primarily grounded in the Smart Manufacturing case study, which provides the most detailed reference scenario for assessing service onboarding, compliance, and operational integration within the DTCL. The validation is structured around three complementary dimensions:

1. Functional integration, ensuring that newly developed predictive maintenance services, such as machine learning models for RUL estimation or anomaly detection, can be seamlessly integrated within the DTCL ecosystem.
2. Technical compliance, verifying adherence to DTCL architectural principles, including composability, interoperability, and cloud-native deployment constraints.
3. Operational performance, assessing scalability, reliability, and predictive accuracy when the system is applied to high-frequency telemetry streams generated by real manufacturing assets.

The adopted methodology is based on an iterative and evidence-based approach, combining automated verification mechanisms with expert-driven evaluation performed by platform owners and domain specialists in industrial maintenance. This hybrid strategy ensures both technical rigor and alignment with operational manufacturing objectives. The approach is consistent with the Composable Business and PBC paradigm, in which enterprise systems are decomposed into modular, reusable services. When instantiated within the DTCL, each predictive maintenance capability can be independently deployed, evolved, or replaced without affecting the overall system stability, thereby supporting agility, resilience, and long-term evolvability. The validation workflow follows the structured service onboarding and acceptance process implemented by the Service Validator subsystem of the DTCL. The process unfolds across five main phases, each mapped to the predictive maintenance use case.

1. Preliminary Analysis and Search Phase. The process begins by querying the DTCL Service Catalog to determine whether an existing service already fulfills the required predictive maintenance functionality (e.g., vibration-based anomaly detection or RUL prediction). If no suitable service is identified, a new module is proposed. In the case study, this corresponds to the introduction of a machine learning service designed to process time-series telemetry from sensors embedded in robotic arms or PLC-controlled machines.

The DTCL Search Engine supports this phase by identifying functionally similar services and preventing redundancy.

2. *Service Definition and Documentation*. The proposed predictive maintenance service is formally described using a standardized template that specifies its functional scope, expected inputs (e.g., temperature, vibration, torque), outputs (e.g., health indicators, RUL estimates), architectural dependencies, and interaction patterns with the DT Core and data store. This documentation ensures architectural consistency and provides the basis for both automated and manual validation activities.

3. *Automated Compliance Validation*. The DTCL Service Validator automatically verifies the service against platform-level technical constraints, including containerization requirements, supported API protocols (REST, GraphQL), messaging patterns (MQTT), and configuration schemas (YAML/JSON). In the predictive maintenance scenario, this step ensures that the analytical service can be deployed alongside existing simulation and data-ingestion services without violating interoperability, scalability, or security guidelines.

4. *Expert Review and Technical Validation*. Following automated validation, a panel of DTCL owners and Smart Manufacturing domain experts evaluates the service's alignment with predictive maintenance objectives and its integration with the layered DTCL architecture. Functional tests are executed in a sandbox environment to verify compatibility with the Deploy Engine, DT Core, Database, and Simulation services. For the case study, this phase confirms that the analytical service correctly retrieves historical and real-time telemetry, executes inference pipelines, and produces predictive indicators in a format consumable by higher-level decision-support or visualization components.

5. *Deployment and Continuous Monitoring*. Once approved, the validated service is published in the Service Catalog and exposed through the Developer Service UI. During operational testing, real telemetry from industrial sensors is ingested through MQTT gateways, while simulation data generated by the DT Core is processed in parallel. System behavior is continuously monitored, and performance is evaluated against predefined Key Performance Indicators (KPIs), including inference latency, scalability under load, and predictive accuracy.

The validation process employs both quantitative and qualitative metrics. Quantitative metrics include prediction accuracy of failure-detection and RUL models, computational efficiency of simulation and inference services, end-to-end data latency, and system availability. Qualitative metrics comprise ease of service integration, modular reusability across different assets or production lines, and compliance with interoperability and composability requirements. Applied to the predictive maintenance case study, the validation demonstrates that new analytical and simulation services can be integrated into the DTCL without disrupting existing components. This confirms the architecture's composability and adaptability, while the Deploy Engine ensures consistent orchestration across services. The automated validation workflow further minimizes redundancy and reinforces architectural robustness. Overall, the practical exploitation of the predictive maintenance scenario validates the DTCL as a scalable, reliable, and evolvable architectural framework for industrial Digital Twin applications. The modular validation process—from discovery and compliance checking to deployment and monitoring—demonstrates that the DTCL can accommodate emerging analytical techniques and domain-specific functionalities without requiring architectural reconfiguration. These results confirm that the DTCL provides a sustainable and interoperable foundation for next-generation Industry 4.0 ecosystems, enabling continuous innovation through the structured integration of advanced Digital Twin services.

The Smart Mobility case study presented in Section 4.2 further complements this validation by demonstrating the applicability of the DTCL in an additional data-driven and simulation-intensive domain.

4.4. Benefits and Open Challenges

Building on the DTCL architecture introduced in Section 3 and the prototype-based assessment discussed in Section 4.2 and validated Section 4.3, this section reflects on the technical impact of adopting a composable, cloud-native control layer for DT-based applications in Smart Manufacturing and Smart Mobility case studies. The observed outcomes confirm that the proposed approach mitigates recurring limitations reported in the literature, notably fragmentation, limited interoperability, and weak lifecycle management, while also surfacing open challenges that become more prominent as the scale and heterogeneity of deployments increase [49].

While several of these benefits emerge most clearly in the predictive maintenance scenario, similar advantages are also observed in the Smart Mobility case study, where modular services support scenario simulation, routing analysis, and KPI-based evaluation without modifying the underlying architectural structure.

From a benefits perspective, the most evident impact concerns composability and architectural evolvability. The case study shows that capabilities such as data ingestion, storage, simulation, and AI/ML-driven analytics can be treated as independently deployable building blocks rather than tightly coupled parts of a monolithic application. In practical terms, this separation reduces coupling between OT-facing telemetry acquisition and higher-level intelligence services, enabling iterative replacement or upgrade of analytics components without requiring changes to the DT Core logic. This is particularly relevant for predictive maintenance, where models and feature pipelines evolve frequently, and where the cost of system downtime (including re-deployment) can otherwise dominate the benefits of advanced analytics.

A second impact area is interoperability-by-design. By centering integration around standardized service interfaces and shared semantic representations (e.g., DTDL and AAS), the DTCL provides a consistent contract between heterogeneous services and assets. While the broader DT literature recognizes interoperability as a persistent barrier to scaling from isolated DT to ecosystems [1], the results in Section 4.3 suggest that enforcing uniform interaction patterns and model-based descriptions substantially lowers integration effort when composing new pipelines. Moreover, the layered physical architecture adopted in the case study clarifies responsibilities between foundational services (authoritative state and persistence), integration services (API gateways and messaging), and intelligence services (simulation and analytics), thereby improving maintainability and facilitating cross-team development.

A third benefit concerns lifecycle governance and controlled evolution. The validation workflow discussed in Section 4.3 operationalises a structured onboarding process for new services, combining discovery, documentation, automated compliance checks, expert review, and monitored deployment. This directly targets the risk of uncontrolled service proliferation in large DT platforms, where duplicated capabilities and inconsistent interface conventions can quickly erode reusability and increase operational cost. In this respect, the DTCL acts not only as an orchestration layer, but also as a governance mechanism that systematically aligns new contributions with platform-wide constraints and conventions.

Despite these benefits, the evaluation also highlights open challenges. A first challenge is scalability of orchestration and governance under ecosystem growth. While microservice-based decomposition supports horizontal scaling, the overhead of service discovery, validation, dependency management, and version compatibility may increase non-linearly

as the catalog expands. The governance mechanisms that improve robustness at small-to-medium scale may become a bottleneck in large deployments unless complemented by richer automation, regression testing, and policy-driven acceptance criteria.

A second challenge concerns cloud–edge placement trade-offs along the C2E continuum. Prior architectures distribute computation across cloud/edge to improve responsiveness and reduce data-transfer pressure, but they also introduce constraints in resource availability, observability, and failure handling [49]. In the DTCL context, deciding where to deploy latency-sensitive inference, where to execute simulation workloads, and how to manage state consistency across tiers remains a non-trivial optimisation problem. Addressing this requires deployment policies that are aware of workload profiles, network conditions, and safety constraints, and that can adapt at runtime without undermining reproducibility.

A third challenge is semantic alignment at scale. Although a shared modelling approach can reduce ambiguity within a controlled prototype, real industrial settings often include legacy schemas, vendor-specific models, and inconsistent naming conventions. Extending semantic interoperability beyond a single project may therefore require explicit alignment strategies, versioned model governance, and mechanisms to detect and resolve semantic conflicts, echoing issues highlighted in IoT-oriented and survey-based studies [1,28].

5. Conclusions and Future Work

This paper has presented a composable and cloud-native DTCL designed to support the development, orchestration, and lifecycle management of DT-based applications in Smart Manufacturing. Motivated by the limitations of current industrial DT deployments, often characterized by fragmentation, limited interoperability, and application-specific architectures, the proposed DTCL has introduced a unified control layer that enables the systematic composition and operation of heterogeneous Digital Twin services across the C2E continuum. The main contribution of this work lies in the definition of a modular architectural model that treats DT functionalities as reusable, independently deployable services rather than tightly coupled components. By combining semantic modelling, standardized interfaces, and cloud-native orchestration mechanisms, the DTCL provides a coherent framework for integrating data ingestion, simulation, analytics, and visualization within a single governed ecosystem. The architecture is explicitly designed to support incremental system evolution, allowing new services to be introduced, validated, and deployed without disrupting existing applications.

The applicability of the proposed approach has been demonstrated through a predictive maintenance case study in a Smart Manufacturing context and further extended through a Smart Mobility scenario for public transport planning. The combined evaluation across industrial and urban domains highlights the flexibility, composability, and general applicability of the DTCL beyond domain-specific implementations. The proposed architecture demonstrates how operational telemetry from heterogeneous systems can be integrated with simulation and AI/ML-based analytics to support fault detection and performance forecasting. The validation process has further confirmed that the DTCL effectively supports the end-to-end lifecycle of DT services, from discovery and compliance checking to deployment and continuous monitoring. These results indicate that the DTCL can serve as a practical enabler for scalable and maintainable industrial digital-twin ecosystems.

At the same time, the study has highlighted several directions for future work. From a technical perspective, further research is needed to address scalability and orchestration efficiency as the number of DT and services increases. Optimizing service placement and execution across cloud and edge resources remains a key challenge, particularly for latency-sensitive applications. In addition, while semantic modeling provides a strong foundation

for interoperability, extending semantic alignment mechanisms to large-scale, multi-vendor environments requires more advanced model governance and conflict-resolution strategies. Future developments will also focus on strengthening automation within the governance workflow, reducing validation overhead while preserving architectural consistency and trustworthiness. Integrating adaptive deployment policies, enhanced observability, and data-quality assessment mechanisms represents another promising direction to improve robustness under real industrial conditions. Finally, evaluating the DTCL across additional use cases beyond the Smart Manufacturing and Smart Mobility scenarios presented in this paper, such as production optimization, energy management, or human–machine collaboration, will further assess its generality and impact beyond predictive maintenance.

Author Contributions: Conceptualization, S.I., A.P. and G.L.; methodology, A.P., F.S. and M.R.; software, M.C. and F.M.; validation, A.P. and M.C.; formal analysis, S.I., G.L. and F.S.; investigation, S.I. and A.P.; resources, A.P., M.C. and F.M.; data curation, A.P. and M.C.; writing—original draft preparation, S.I., A.P., D.L. and G.L.; writing—review and editing, F.S. and M.R.; visualization, D.L. and A.P.; supervision, S.I., A.P. and G.L.; project administration, A.P. and M.R.; funding acquisition, F.S. and M.R. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by *NXT Digital Platform* project (grant number M1ERDW2), co-funded by NTT DATA Italia S.p.A. and the European Regional Development Fund for Apulia Region 2014/2020 Operating Program.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

Conflicts of Interest: A.P., M.C. and F.M. are employed by NTT DATA Italia S.p.A.; S.I., G.L., F.S. and M.R. are partners in donkeyPower S.r.l. 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.

References

1. da Silva Mendonça, R.; de Oliveira Lins, S.; Valente de Bessa, I.; de Carvalho Ayres, F.A., Jr.; Paiva de Medeiros, R.L.; Ferreira de Lucena, V., Jr. Digital Twin Applications: A Survey of Recent Advances and Challenges. *Processes* **2022**, *10*, 744. [CrossRef]
2. Ieva, S.; Bilenchi, I.; Gramegna, F.; Pinto, A.; Scioscia, F.; Ruta, M.; Loseto, G. Enhancing Last-Mile Logistics: AI-Driven Fleet Optimization, Mixed Reality, and Large Language Model Assistants for Warehouse Operations. *Sensors* **2025**, *25*, 2696. [CrossRef]
3. Ieva, S.; Loconte, D.; Loseto, G.; Ruta, M.; Scioscia, F.; Marche, D.; Notarnicola, M. A Retrieval-Augmented Generation Approach for Data-Driven Energy Infrastructure Digital Twins. *Smart Cities* **2024**, *7*, 3095–3120. [CrossRef]
4. Tancredi, G.P.; Vignali, G.; Bottani, E. Integration of Digital Twin, Machine-Learning and Industry 4.0 Tools for Anomaly Detection: An Application to a Food Plant. *Sensors* **2022**, *22*, 4143. [CrossRef] [PubMed]
5. González-Herbón, R.; González-Mateos, G.; Rodríguez-Ossorio, J.R.; Domínguez, M.; Alonso, S.; Fuertes, J.J. An Approach to Develop Digital Twins in Industry. *Sensors* **2024**, *24*, 998. [CrossRef]
6. Loconte, D.; Ieva, S.; Gramegna, F.; Bilenchi, I.; Fasciano, C.; Pinto, A.; Loseto, G.; Scioscia, F.; Ruta, M.; Di Sciascio, E. Serverless Microservice Architecture for Cloud-Edge Intelligence in Sensor Networks. *IEEE Sens. J.* **2025**, *25*, 7875–7885. [CrossRef]
7. Maia, A.; Boutouchent, A.; Kardjadja, Y.; Gherari, M.; Soyak, E.G.; Saqib, M.; Boussekar, K.; Cilbir, I.; Habibi, S.; Ali, S.O.; et al. A survey on integrated computing, caching, and communication in the cloud-to-edge continuum. *Comput. Commun.* **2024**, *219*, 128–152. [CrossRef]
8. Grieves, M.W. Digital Twins: Past, Present, and Future. In *The Digital Twin*; Springer International Publishing: Cham, Switzerland, 2023; pp. 97–121. [CrossRef]
9. Singh, M.; Fuenmayor, E.; Hinchy, E.P.; Qiao, Y.; Murray, N.; Devine, D. Digital Twin: Origin to Future. *Appl. Syst. Innov.* **2021**, *4*, 36. [CrossRef]

10. Barricelli, B.R.; Casiraghi, E.; Fogli, D. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. *IEEE Access* **2019**, *7*, 167653–167671. [CrossRef]
11. Boschert, S.; Rosen, R. Digital Twin—The Simulation Aspect. In *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*; Springer International Publishing: Cham, Switzerland, 2016; pp. 59–74. [CrossRef]
12. W3C Working Group. OWL 2 Web Ontology Language: Primer. 2012. Available online: http://www.w3.org/TR/2012/REC-owl2-primer-20121211/ (accessed on 2 February 2026).
13. Rahal, J.R.; Schwarz, A.; Sahelices, B.; Weis, R.; Anton, S.D. The asset administration shell as enabler for predictive maintenance: A review. *J. Intell. Manuf.* **2025**, *36*, 19–33. [CrossRef]
14. Liu, X.; Jiang, D.; Tao, B.; Xiang, F.; Jiang, G.; Sun, Y.; Kong, J.; Li, G. A systematic review of digital twin about physical entities, virtual models, twin data, and applications. *Adv. Eng. Inform.* **2023**, *55*, 101876. [CrossRef]
15. Preuss, K.; Schulte, S.N.; Rzazonka, L.; Befort, L.; Fresemann, C.; Stark, R.; Russwinkel, N. Towards A Human-Centered Digital Twin. In *Proceedings of the 16th CIRP Conference on Intelligent Computation in Manufacturing Engineering*; Elsevier: Amsterdam, The Netherlands, 2023; Volume 118, pp. 324–329.
16. Desolda, G.; Esposito, A.; Lanzilotti, R.; Piccinno, A.; Costabile, M.F. From human-centered to symbiotic artificial intelligence: A focus on medical applications. *Multimed. Tools Appl.* **2025**, *84*, 32109–32150. [CrossRef]
17. Shaaban, A.M.; Chlup, S.; El-Araby, N.; Schmittner, C. Towards Optimized Security Attributes for IoT Devices in Smart Agriculture Based on the IEC 62443 Security Standard. *Appl. Sci.* **2022**, *12*, 5653. [CrossRef]
18. *ISO 8000-1:2022*; Data Quality. International Organization for Standardization: Geneva, Switzerland, 2022. Available online: https://www.iso.org/obp/ui/#iso:std:iso:8000:-1:ed-1:v1:en (accessed on 2 February 2026).
19. Perez-Castillo, R.; Carretero, A.G.; Caballero, I.; Rodriguez, M.; Piattini, M.; Mate, A.; Kim, S.; Lee, D. DAQUA-MASS: An ISO 8000-61 Based Data Quality Management Methodology for Sensor Data. *Sensors* **2018**, *18*, 3105. [CrossRef]
20. *ISO 10303-1:2024*; Industrial Automation Systems and Integration—Product Data Representation and Exchange. International Organization for Standardization: Geneva, Switzerland, 2024. Available online: https://www.iso.org/obp/ui/#iso:std:iso:10303:-1:ed-3:v1:en (accessed on 2 February 2026).
21. Pratt, M.J. Introduction to ISO 10303—The STEP standard for product data exchange. *J. Comput. Info. Sci. Eng.* **2001**, *1*, 102–103. [CrossRef]
22. *IEEE 1872-2015*; IEEE Standard Ontologies for Robotics and Automation. IEEE Standards Association: Piscataway, NJ, USA, 2015. Available online: https://standards.ieee.org/ieee/1872/5354/ (accessed on 2 February 2026).
23. *ISO/IEC 30182:2017*; Smart City Concept Model—Guidance for Establishing a Model for Data Interoperability. International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland, 2017. Available online: https://www.iso.org/standard/53302.html (accessed on 2 February 2026).
24. *IEC 61499-1:2012*; Function Blocks—Part 1: Architecture. International Electrotechnical Commission: Geneva, Switzerland, 2012. Available online: https://webstore.iec.ch/en/publication/5506 (accessed on 2 February 2026).
25. *ISO/IEC 27001:2022*; Information Security, Cybersecurity and Privacy Protection—Information Security Management Systems—Requirements. International Organization for Standardization/International Electrotechnical Commission: Geneva, Switzerland, 2022. Available online: https://www.iso.org/standard/27001 (accessed on 2 February 2026).
26. Humphreys, E. *Implementing the ISO/IEC 27001 Information Security Management System Standard*, 1st ed.; Artech House, Inc.: Norwood, MA, USA, 2007.
27. He, R.; Chen, G.; Dong, C.; Sun, S.; Shen, X. Data-driven digital twin technology for optimized control in process systems. *ISA Trans.* **2019**, *95*, 221–234. [CrossRef] [PubMed]
28. Al-Ali, A.R.; Gupta, R.; Zaman Batool, T.; Landolsi, T.; Aloul, F.; Al Nabulsi, A. Digital Twin Conceptual Model within the Context of Internet of Things. *Future Internet* **2020**, *12*, 163. [CrossRef]
29. Bordeleau, F.; Combemale, B.; Eramo, R.; van den Brand, M.; Wimmer, M. Towards Model-Driven Digital Twin Engineering: Current Opportunities and Future Challenges. In *Proceedings of the Systems Modelling and Management*; Babur, Ö., Denil, J., Vogel-Heuser, B., Eds.; Springer: Cham, Switzerland, 2020; pp. 43–54.
30. Zhao, H.; Liu, C.; Dang, X.; Xu, J.; Deng, W. Few-Shot Cross-Domain Fault Diagnosis of Transportation Motor Bearings Using MAML-GA. *IEEE Trans. Transp. Electrif.* **2026**, *12*, 1165–1174. [CrossRef]
31. Li, J.; Deng, W.; Ding, J.; Zhao, H. IBN-MixStyle Network with Dynamic Weighted Invariant Risk Minimization for Domain-Generalized Bearing Fault Diagnosis. *IEEE Trans. Consum. Electron.* **2025**, *71*, 9929–9939. [CrossRef]
32. Tao, F.; Zhang, H.; Liu, A.; Nee, A.Y.C. Digital Twin in Industry: State-of-the-Art. *IEEE Trans. Ind. Inform.* **2019**, *15*, 2405–2415. [CrossRef]
33. Lawal, Z.K.; Yassin, H.; Lai, D.T.C.; Che Idris, A. Physics-Informed Neural Network (PINN) Evolution and Beyond: A Systematic Literature Review and Bibliometric Analysis. *Big Data Cogn. Comput.* **2022**, *6*, 140. [CrossRef]
34. Udrescu, S.M.; Tegmark, M. AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.* **2020**, *6*, eaay2631. [CrossRef]

35. Capuano, G.; Rimoli, J.J. Smart finite elements: A novel machine learning application. *Comput. Methods Appl. Mech. Eng.* **2019**, *345*, 363–381. [CrossRef]
36. Bacsa, K.; Lai, Z.; Liu, W.; Todd, M.; Chatzi, E. Symplectic encoders for physics-constrained variational dynamics inference. *Sci. Rep.* **2023**, *13*, 2643. [CrossRef] [PubMed]
37. W3C Working Group. RDF 1.1 Primer. 2014. Available online: https://www.w3.org/TR/rdf11-primer/ (accessed on 2 February 2026).
38. Beckett, D. RDF 1.1 N-Triples. 2014. Available online: https://www.w3.org/TR/2014/REC-n-triples-20140225/ (accessed on 2 February 2026).
39. Beckett, D.; Berners-Lee, T.; Prud'hommeaux, E.; Carothers, G. RDF 1.1 Turtle. 2014. Available online: https://www.w3.org/TR/turtle/ (accessed on 2 February 2026).
40. W3C Working Group. OWL 2 Web Ontology Language: Profiles. 2012. Available online: https://www.w3.org/TR/2012/REC-owl2-profiles-20121211/ (accessed on 2 February 2026).
41. Douceur, J. Digital Twins Definition Language (DTDL) Version 3. 2025. Available online: https://github.com/Azure/opendigitaltwins-dtdl/blob/master/DTDL/v3/DTDL.v3.md (accessed on 15 November 2025).
42. Lu, Y.; Liu, C.; Wang, K.I.K.; Huang, H.; Xu, X. Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robot.-Comput.-Integr. Manuf.* **2020**, *61*, 101837. [CrossRef]
43. Friederich, J.; Francis, D.P.; Lazarova-Molnar, S.; Mohamed, N. A framework for data-driven digital twins of smart manufacturing systems. *Comput. Ind.* **2022**, *136*, 103586. [CrossRef]
44. Jiang, X.; Liu, D.; Sun, Y.; Li, G. Digital Twin-Driven Smart Manufacturing: A Review of Key Technologies and Applications. *Electronics* **2023**, *12*, 345. [CrossRef]
45. Hakiri, A.; Gokhale, A.; Yahia, S.B.; Mellouli, N. A comprehensive survey on digital twin for future networks and emerging Internet of Things industry. *Comput. Netw.* **2024**, *244*, 110350. [CrossRef]
46. Jeremiah, S.R.; El Azzaoui, A.; Xiong, N.N.; Park, J.H. A comprehensive survey of digital twins: Applications, technologies and security challenges. *J. Syst. Archit.* **2024**, *151*, 103120. [CrossRef]
47. Sacoto-Cabrera, E.J.; Perez-Torres, A.; Tello-Oquendo, L.; Cerrada, M. IoT, AI, and Digital Twins in Smart Cities: A Systematic Review for a Thematic Mapping and Research Agenda. *Smart Cities* **2025**, *8*, 175. [CrossRef]
48. Guo, D.; Zhong, R.Y.; Lin, P.; Lyu, Z.; Rong, Y.; Huang, G.Q. Digital twin-enabled Graduation Intelligent Manufacturing System for fixed-position assembly islands. *Robot.-Comput.-Integr. Manuf.* **2020**, *63*, 101917. [CrossRef]
49. Li, Z.; Mei, X.; Sun, Z.; Xu, J.; Zhang, J.; Zhang, D.; Zhu, J. A reference framework for the digital twin smart factory based on cloud-fog-edge computing collaboration. *J. Intell. Manuf.* **2025**, *36*, 3625–3645. [CrossRef]
50. Wu, H.; Ji, P.; Ma, H.; Xing, L. A Comprehensive Review of Digital Twin from the Perspective of Total Process: Data, Models, Networks and Applications. *Sensors* **2023**, *23*, 8306. [CrossRef]
51. Industrial Digital Twin Association (IDTA). Details of the Asset Administration Shell—Part 1: The Exchange Information Model. 2023. Available online: https://industrialdigitaltwin.org (accessed on 2 April 2026).
52. Lefrançois, M. Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Patterns. In *SEMANTiCS 2017 Workshops Proceedings: SIS-IoT*; CEUR-WS: Aachen, Germany, 2017. Available online: https://ceur-ws.org/Vol-2063/sisiot-paper2.pdf (accessed on 2 February 2026).
53. Bizer, C.; Heath, T.; Berners-Lee, T. Linked Data—The Story So Far. *Int. J. Semant. Web Inf. Syst.* **2009**, *5*, 115–143. [CrossRef]
54. W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview (Second Edition). 2012. Available online: https://www.w3.org/TR/owl2-overview/ (accessed on 2 February 2026).
55. Brickley, D.; Guha, R. RDF Schema 1.1. 2014. Available online: https://www.w3.org/TR/rdf-schema/ (accessed on 2 February 2026).
56. Miles, A.; Bechhofer, S. SKOS Simple Knowledge Organization System—Reference. 2009. Available online: https://www.w3.org/TR/2009/REC-skos-reference-20090818/ (accessed on 2 February 2026).
57. Dublin Core Metadata Initiative. DCMI Metadata Terms. 2020. Available online: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/ (accessed on 2 February 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.