Reference architecture and ontology framework for digital twin construction

Jonas Schlenger, Kacper Pluta, Alwyn Mathew, Timson Yeung, Rafael Sacks, André Borrmann

Automation in Construction 174 (2025) 106111

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Reference architecture and ontology framework for digital twin construction

Jonas Schlenger [a,*], Kacper Pluta [c,d], Alwyn Mathew [e], Timson Yeung [b], Rafael Sacks [b], André Borrmann [a]

[a] Chair of Computational Modeling and Simulation, School of Engineering and Design, Technical University of Munich, Arcisstraße 21, 80333, Munich, Germany
[b] Faculty of Civil and Environmental Engineering, Technion - Israel Institute of Technology, 32000, Haifa, Israel
[c] Université Gustave Eiffel, CNRS, LIGM, F-77454, Marne-la-Vallée, France
[d] Inria, Université Côte d'Azur, F-06902, Valbonne, France
[e] University of Cambridge, CB3 0FA, Cambridge, United Kingdom

ARTICLE INFO

Keywords:
Digital Twin Construction (DTC)
Ontology
Reference architecture
Building construction
Data management
Linked building data (LBD)
Data integration
Interoperability
Digital twin (DT)

ABSTRACT

The application of digital twins in building construction faces challenges due to limited guidance on the necessary data management layers. This paper addresses this gap by investigating how the reference architecture for Digital Twin Construction (DTC) should be structured to manage planning information, raw monitoring data, and derived knowledge, as well as its data schema to compare project plans with status. By defining platform requirements and using Design Science Research Methodology, a solution was implemented and validated using a case study based on the ConSLAM dataset. A plugin-based DTC reference architecture employing multiple RDF graphs linked to specialized databases and the DTC Ontology as the internal data schema are introduced. This architecture guides construction companies in data-driven decision-making during execution. It establishes a foundation for managing digital twins and fosters the development of domain-specific services that benefit from clear data structures, supporting a holistic digital twin adaptable to project-specific needs.

1. Introduction

While industries like manufacturing and aerospace engineering have developed and adopted systems based on the Digital Twin (DT) concept, the architecture, engineering, construction, and operation industry still faces challenges in implementing digital twin systems and processes for a building's lifecycle [1,2]. Such a digital twin system, consisting of a physical building project, its digital counterpart, techniques to update the digital twin based on monitoring of the physical building and of the construction process, and mechanisms to deliver information from the digital twin to the production process, promises significant improvements when applied in the construction industry. This includes advanced construction management, proactive instead of reactive planning, closed-loop cycles for continuous improvement of the construction execution, among others [3]. More specifically, the digital twin, as a continuously updated knowledge base, can make construction management more time-efficient and less prone to errors. Traditionally, decisions are often based on visual inspections and intuition, whereas the digital twin can offer up-to-date, accurate information, enabling data-driven and reliable decision-making. Consequently, this approach can lead to improvements such as reducing delays, preventing resource mismanagement, and enhancing overall construction efficiency.

Project Intent Information (PII), which includes product information provided in Building Information Modeling (BIM) models and process information that defines production plans, construction schedules, and resource assignments, are fundamental starting points for DTs of the following lifecycle phases [1]. In the operational phase, Internet of Things (IoT)-based DTs that enhance the building's energy efficiency, comfort of its inhabitants, facility management, and others have been developed [4–6]. The implementation of a DT of the construction execution, providing a holistic view of the construction site, however, remains very challenging [3,7]. While many researchers focus on specific use cases, a holistic system is required that hosts such use case-driven services and serves as a basis for DTs of construction execution [3]. Here, the term service refers to a small, independent software tool with a clearly defined interface [8,9].

Sacks et al. [3] were the first to provide a conceptual basis for a DT of the construction execution phase, denoted as Digital Twin Construction (DTC). They describe DTC as a DT with a holistic view of the construction site to provide situational awareness to construction personnel to improve the efficiency of the construction process. Furthermore, DTC applies concepts of lean construction to continuously

* Corresponding author.
E-mail address: jonas.schlenger@tum.de (J. Schlenger).

https://doi.org/10.1016/j.autcon.2025.106111
Received 3 July 2024; Received in revised form 21 February 2025; Accepted 28 February 2025
Available online 15 March 2025

compare the current status of the construction site, Project Status Information (PSI), with PII, to identify deviations [3].

However, to date, the DTC paradigm remains mostly conceptual. Technical developments are required to demonstrate how DTC can be implemented in a software system. As a contribution, this paper is dedicated to the data storage and management of such a DT for the construction phase. The large volume of raw data collected on the construction site, together with the derived information and knowledge, requires tailored data management solutions. In addition, the information from the planning phase needs to be managed in conjunction with the information on the current status of the construction project. Identifying potential progress deviations through a comparison of intent and status requires clear, aligned data structures. This comparison between intent and status is one of the digital twin's most crucial features. It allows construction progress and potential deviations from the plan to be quantified. Such productivity-related information offers significant support to decision-makers by enabling timely identification of performance issues and facilitating comparisons and learning across multiple projects. Therefore, this aspect requires careful consideration.

Existing research efforts have focused on elaborating a reference architecture for DTs. A reference architecture outlines the components of a software platform along with their essential characteristics and interrelations, contrasting with a data schema, which describes the structure of the data managed and processed by these components. At the same time, a reference architecture remains neutral regarding the applied technology to adapt to company or project-specific boundary conditions. It serves as a blueprint that kickstarts the development of platform instances by providing a tested and established foundation for implementation [4,10]. Existing implementations of reference architectures found in academia, however, fail to address how data, the information derived from the data, and knowledge, can be properly managed within the DT [4–6,11,12]. The reference architecture proposed in this paper offers a structured data management solution to address the aforementioned challenges. Additionally, it enhances scalability, reliability, and interoperability. Unlike existing approaches, it also achieves a more robust integration of geometric information, which has previously been addressed only superficially.

Regarding DT data structures, existing data schemata [13,14] are designed to represent information about the situation on the construction site but lack mechanisms for a direct comparison between PII and PSI.

This paper addresses this data integration-related research gap by providing answers to the following Research Questions (RQs):

> **RQ1**: How does a reference architecture for DTC need to be structured to manage data, information, and knowledge simultaneously?
> **RQ2**: Which existing data schemata can serve as the baseline for DTC?
> **RQ3**: How should a data schema be structured to enable a definitive comparison of project intent and status?
> **RQ4**: How can the reference architecture and data schema combined provide precise data structures for DTC?

In this study, these questions are approached from the viewpoint of building construction projects, considering linear infrastructure projects out-of-scope. The proposed solution has been developed and tested within the framework of the EU Horizon 2020 project BIM2TWIN [15].

The remainder of this study is structured as follows: Section 2 introduces the DT concept and discusses existing reference architectures and data schemata for DTs. Section 3 presents technical design decisions regarding data management platforms for DTs. The proposed reference architecture for DTC is explained in Section 4, followed by the proposed data schema in Section 5. The latter comprises a set of existing ontologies in combination with the newly defined DTC Ontology. The case study is then presented in Section 6 to evaluate the suitability of the developed data structures. Section 7 discusses the limitations. Finally, Section 8 concludes the paper.

2. Literature review

This section introduces the digital twin paradigm and discusses the shortcomings of existing reference architectures and data schemata for digital twins, leading to the definition of the most pressing research gaps.

2.1. Digital twin paradigm

Initially, the DT concept emerged in the aerospace industry, where NASA applied it to digitally replicate spacecraft and predict their behavior. Grieves coined the term DT, focusing on product lifecycle management [2]. He outlined its three essential components: the physical product, its virtual representation, and a data connection to synchronize them [2]. Most digital twins initially rely on a static digital model without real-time synchronization. Once the physical object of interest is created, connections are established to enable the digital twin. In the construction sector, this happens at the start of the execution phase when the physical building begins to take shape, allowing the digital twin to be fully implemented. This is also where the DTC paradigm becomes relevant. Sacks et al. [3] define it as a holistic DT of the construction site. Beyond this, they propose applying lean construction principles like Plan-Do-Check-Act cycles to enable continuous productivity improvements. Most importantly, during the *check* phase, the project intent is compared with the current status, conducted at both the product level, comparing as-designed to as-built, and at the process level, comparing as-planned to as-performed. Any identified deviations are valuable insights for the construction personnel and contribute to their situational awareness.

This DTC paradigm forms the conceptual basis for the reference architecture and ontology framework research laid out in this paper. The paradigm as a whole can be described by the simplified workflow diagram shown in Fig. 1. Drawing an analogy to the Plan-Do-Check-Act cycle, *plan* is formalized in the PII, *do* corresponds to the building process, *check* is performed through monitoring, processing, and comparison, and finally, *act* is represented through feedback that influences the building process.

While in other industries, both communication directions of the DT, physical-to-virtual and virtual-to-physical, are fully automated, in the DTC context, the virtual-to-physical connection is far from being automated. Although technology continuously advances, human involvement remains crucial in decision-making, ensuring accountability as well as the execution of physical construction tasks [16]. The initial stage of DTC functions as an informative DT, gathering information from the physical site to provide decision makers with aggregated information and knowledge, guiding their decision-making process.

According to the classification by Kritzinger et al. [17], the DT described here is classified as a *digital shadow* because it solely automates the physical-to-virtual connection, and not the counter-direction. Nevertheless, the term DT is used for the remainder of this paper for simplification.

2.2. State-of-the-art reference architectures

On the one hand, high-level platform architectures focus on generic modules or layers with very little guidance regarding implementation. They are usually not specific to a particular domain due to their generality. In the context of DTC, many suitable high-level architectures exist designed for generic DTs, cyber–physical systems, or IoT platforms. On the other hand, low-level architectures are often domain or use-case-specific. Some provide example implementations with a selection of possible technologies and give detailed technical guidance [5,6,11,18].

Starting with high-level platform architectures, Mostafa et al. [11] developed a six-layer architecture for digital twin systems. The architecture comprises the physical layer, ingestion, persistence, inference, services, and consumption (see Fig. 2). It conceptualizes the data flow,

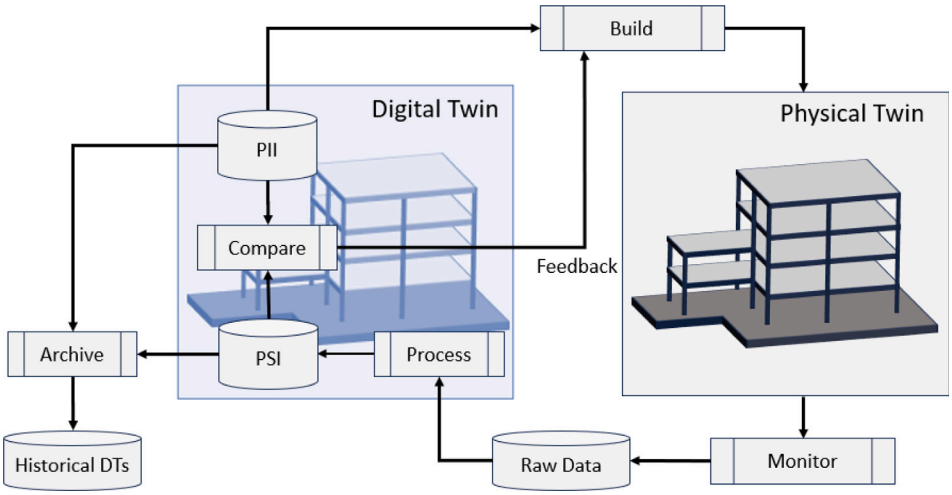

Fig. 1. Simplified Digital Twin Construction (DTC) workflow based on Sacks et al. [3].

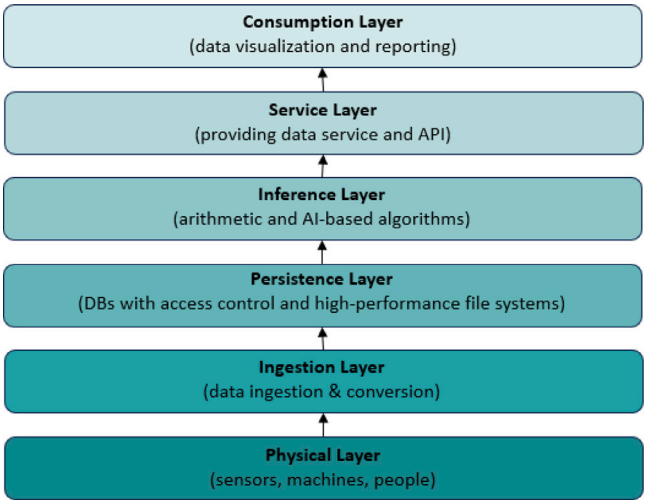

Fig. 2. Six-layer DT architecture by Mostafa et al. [11].

starting with collecting data from the physical building, which turns into valuable insights through conversion, storage, and interpretation. Lee et al. [12] proposed another layered architecture focused on cyber–physical systems. Although the terminology differs from Mostafa et al. [11], both agree on the need to convert raw data into information and knowledge. While Mostafa et al. [11] have the support of human decision-making in mind, Lee et al. [12] aim at a self-sufficient learning system. In addition to these two architectures, many similar approaches exist [19,20]. Although they differ in several aspects, all architectures acknowledge the importance of data management aspects of DTs due to their data-driven nature.

When looking at low-level platform architectures, Ramonell et al. [6] demonstrate their DT with varying building maintenance tasks. Their platform architecture is primarily based on a graph database, reliant on the Industry Foundation Classes (IFC), and connected to a Mainflux database to store all types of raw data. While this system can store data and information to some extent, the authors do not address how the knowledge gained by digital twin services is managed in this platform. Lu et al. [5] developed a platform architecture, illustrated through a case study involving a DT for the operational phase of buildings and cities. They successfully integrated data from various sources, including sensors, BIM models, and asset management software, simply by employing matching tables to link corresponding objects across databases. While effective for cases with clear one-to-one correspondences between databases, more advanced association methods are necessary for databases with varying granularities and complex dependencies. Chamari et al. [4] suggested a service-oriented architecture for asset management, utilizing Resource Description Framework (RDF) graphs and established ontologies to store metadata, primarily concerning buildings and connected sensors. Focusing on dashboards that merge building information with sensor data, they do not discuss integrating gained knowledge with existing data and information. Furthermore, Chamari et al. [4] adopted a distributed and modular service approach, which introduces challenges such as latency issues, potentially inconsistent versions of the digital twin existing simultaneously, and reduced control over schema compliance. The platform architecture approaches discussed here are summarized in Table 1.

Overall, the primary deficiency of existing platform architectures is managing the knowledge gained from digital twin services. While they elaborate on converting data into information and knowledge, feeding back the knowledge into the platform is not adequately discussed. The challenge lies in the gained knowledge that goes beyond updating, e.g., simple status values. Additionally, a DT of a construction site needs to be able to deal with a set of constantly growing raw data sources. To fully leverage this data, it needs to be easily discoverable, even if not all sources are initially known.

2.3. State-of-the-art data schemata for representing building data

Establishing a DTC platform requires a system architecture to define platform components and an internal data schema to govern data storage. Advanced data schemas, compared to flat, unstructured data, avoid redundancies and inconsistencies in project data and ensure interoperability between project stakeholders. In the architecture, engineering, construction, and operation industry, various data modeling paradigms are employed [22]. Monolithic schemas such as IFC aim for broad coverage, but bring complexity. In contrast, semantic web approaches suggest concise ontologies, promoting dataset comparability through ontology reuse, combining flexibility and rigor [22,23].

This section presents the most relevant data schemata in the context of DTC and assesses their ability to represent product and process-related aspects of DTC.

IFC is an internationally established standard maintained by buildingSMART International [13] for storing and exchanging construction-related content. It covers various aspects across all building lifecycle phases. However, its complexity, with several hundred classes, makes the use of IFC challenging [24]. In the context of construction processes, nested `IfcTasks` are used to represent individual processes of

Table 1
Summary of state-of-the-art DT architectures.

Ref.	Key features	Limitations
[11]	Six-layer architecture DT system to support decision-making Open-source implementation	Focused on indoor manufacturing
[12]	Five-layer architecture Dedicated to cyber–physical systems Goal: self-learning system	Focus on highly automated systems High-level description
[21]	5-Dimension DT including physical and virtual entity, connection, data, and services	Little information on data management aspects Very generic
[20]	Six-layer architecture Digital twin for IoT-based systems	Linking between structured and unstructured data does not become clear
[6]	DT for building maintenance Graph database based on IFC	Does not cover integration of derived knowledge
[5]	DT for the operational phase Integration of sensor data, BIM, and data from asset management tools	Matching tables cover only simple data interrelations Use of proprietary software
[4]	Service-oriented architecture DT for asset management BIM and sensor data	Does not cover integration of derived knowledge

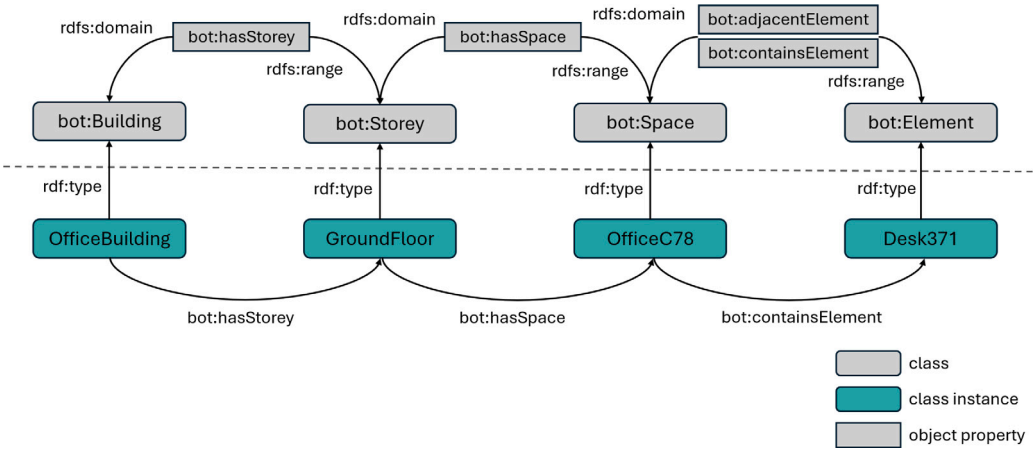

Fig. 3. Main BOT classes with several example instances [28].

a construction schedule and their hierarchical dependencies [13]. This process representation does not distinguish between various granularity levels, since both detailed construction steps and complete phases are modeled using `IfcTask`, posing challenges for automated interpretation. Moreover, IFC lacks a complete strategy for representing PII and PSI. For instance, an `IfcTask` contains attributes like `scheduleStart` and `actualStart` [13]. Since not all planned processes are executed as expected, a process could require additional tasks, resulting in multiple as-performed processes corresponding to a single as-planned process. Representing such a scenario is not supported by IFC.

IFC, originally defined in EXPRESS, has also been translated into Web Ontology Language (OWL) to enable its application in the Semantic Web context [25]. To divide ifcOWL into smaller modules, the Building Element Ontology (BEO) developed by Pauwels [26] encapsulates the building element part of ifcOWL in a small, easy-to-reuse ontology. Unlike IFC, the Building Topology Ontology (BOT) by Rasmussen et al. [27] is a minimal ontology, exclusively focusing on the building zones, elements, and their topological relationships. The main classes and some of their relations are visualized in Fig. 3. Since BOT is generic and can be used with other ontologies, it is considered suitable for DTC.

The Digital Construction Ontologies (DiCon) are a data schema dedicated to the management and execution of construction projects. Although they also use only one type of process class, called activity, they allow differentiation through three relationships: `hasLocationPhase`, `hasObjectPhase`, and `hasProcessPhase`. These define different types of process decompositions [14]. As a downside, the process class cannot represent different process precedence sequences, such as finish-to-start and start-to-start, which are essential in construction schedules. With the context class, various types of contexts can be defined, such as planning variants and simulation scenarios. While it does allow the separation of PII and PSI, it does not consider context-specific differences. Some classes or attributes should only be used for PII or PSI, but not both.

Finally, the Semantic Sensor Network and Sensor, Observation, Sample, and Actuator (SOSA/SSN) ontologies describe sensor systems with their observations and data-capturing techniques. [29]. Researchers have pointed out limitations in the storage of large amounts of time series data in graph-based databases [30]. However, the possibility of referencing external sources stored outside the graph is very suitable for DTC.

2.4. Research gap and contribution

In summary, the following research gaps and remaining challenges in implementing data management-related components of a DT of building construction have been identified:

(1) Existing platform architectures do not adequately address the integration of gained knowledge with data and information. The full potential of the DT can only be reached if services can profit from the knowledge gained by other services (addressing RQ1).
(2) In the DT environment, the sources of raw monitoring data can increase over time. Therefore, discoverability of available data is essential for digital twin services.
(3) Currently available data schemas lack explicit definitions for project intent and status. Nonetheless, comparing them is crucial to assess the project progress. To address RQ2 and RQ3, a novel data schema is necessary, leveraging existing standards where feasible.
(4) Existing data schemata do not represent varying process granularities and complex process preconditions needed for flexible project management.

In combination, the platform architecture and the data schema should provide clear data structures that cope with heterogeneous data, information, and knowledge to provide a complete data management solution that addresses RQ4.

3. Platform design considerations

In addition to the requirements arising from the identified research gap, additional aspects need to be taken into account when creating a platform for DTC.

3.1. Types of building data

Designing suitable data management structures demands a thorough understanding of the different types of building data that are relevant during construction execution. The focus lies on data directly related to the building production and construction process, including planning and monitoring information. Documents specifying contractual details or the organizational structure of the project parties are considered out-of-scope.

In the first step, all information can be divided into PII and PSI. PII describes the information related to the design and construction planning, representing the project team's intention, while PSI describes the actual situation on the construction site [3]. Both PII and PSI are continuously updated during the project: PSI in response to changes in status, presenting the past, and PII in response to management decisions to change product designs or production plans.

In the second step, information is roughly grouped by granularity and level of abstraction. The data, information, and knowledge concept by Ackoff [31] is applied for this purpose. *Data* are quantitative or qualitative values representing atomic properties of objects, processes or their environment. *Data* refers to raw data produced by sensors, including images, point clouds, time series data, etc. Typically, data is highly storage-intensive due to its large volume and often features a simplistic structure akin to relational tables. *Information* is created by processing, aggregating, and interpreting raw data. In essence, a single piece of information can be derived from multiple raw data points. For example, one might deduce the time that a piece of construction machinery spends on site by examining its global positioning system data. Other examples of information are the date a building element is completed and the time span for which a construction process is scheduled. Finally, *knowledge* is created by aggregating and interpreting information. It can be understood as learnings gained from long-term observations, pattern recognition, and comparative analysis, which can directly support the decision-making process. Comparisons of status information to intent information generate value judgements that are an aspect of knowledge about a project. In the context of this work, aspects of DTC knowledge are represented as Key Performance Indicators (KPIs). In construction, KPIs are commonly divided into the categories of quality, execution excellence of products and processes, and safety. These can include KPIs such as average product cycle times, number of defects per work package, and others [32]. Moving from data to knowledge reduces the volume due to their differences in granularity. More importantly, through interpretation and abstraction, meaningful insights become available that are required for decision-making.

3.2. Digital twin platform for data management

Hosting a DT for the construction phase requires a digital platform for data management and the coordination of software services. The literature distinguishes between monolithic and distributed platforms. Monolithic platforms operate as a single software unit hosted on one server, which ensures consistency between individual components. However, this also leads to a lack of flexibility and extensibility. On the other hand, distributed platforms are composed of independent, loosely coupled services with clearly defined interfaces. Architectures in this category include service-oriented, microservices, and event-driven architectures. While they have advantages in reusability and flexibility, they require sophisticated communication between the modules and introduce latency into the systems [4,8,9]. Given the diverse nature of information types that a digital twin of the execution phase must handle, such as alphanumeric properties, geometric information, and time series data, and the need to integrate functionalities for various use cases, a distributed platform is considered a more suitable choice due to its flexibility. However, there are also benefits to the simplicity of monolithic systems. This resulted in choosing a combined approach with a monolithic core module responsible for the key DT functionalities that is extended through flexible services, which are selected based on the use cases to be implemented. This design aligns closely with microkernel architectures, which feature a core module that supports extension through plug-in-like components [33].

Given such a platform architecture, web Application Programming Interfaces (APIs) are essential for communication between the central platform and the digital twin services. Web APIs are based on the Hypertext Transfer Protocol (HTTP) and offer a programmatic interface for software applications. Unlike Hypertext Markup Language (HTML), which delivers human-consumable content, Web APIs provide automated access for software tools. These APIs operate on the request–response principle, where a client sends a request to the server, which processes it and sends back a response. Various HTTP methods, such as GET, POST, PUT, and DELETE, are available to perform different actions on specified resources [34].

Although some researchers and practitioners implement services with individual data management [4], a centralized data management is considered more suitable for DTC. Centralized data management offers significant advantages by providing complete control over schema compliance, ensuring consistent data policies, and implementing robust security measures. It also effectively manages data redundancy, which can lead to inefficiencies in distributed systems. A single source of truth prevents conflicting versions of the DT, which is crucial due to the large number of construction stakeholders. Here, all services interact with the central data management module to continuously update the DT [35].

3.3. Types of database models

Databases form the fundamental basis for data management. For many years, relational databases dominated the market. They perform exceptionally well in handling tabular data, providing implicit relationships between the tables by primary and foreign keys [36]. Within the last decade, data storage requirements have changed. The amount of data to be stored, and its variety, make scalability and flexibility new data management requirements [37].

These changes have led to the development of so-called NoSQL databases, which do not rely on Structured Query Language (SQL) [38]. They are characterized by fast data access and their ability to handle semi- and unstructured data. Because of their simple structure, data

can be split and distributed across several servers, providing horizontal scalability [36,37].

Graph databases are one type of NoSQL database. They focus on highly connected data, representing relationships explicitly. One of the strengths of graph databases is their highly performant graph traversals, which depend on the size of the traversed graph section rather than the complete graph size. This is advantageous for large datasets [37].

There are two established approaches to graph databases. Labeled Property Graphs (LPGs) allow attaching attributes to nodes and relationships. Moreover, they usually follow a schema-less philosophy. This means it is possible to formulate an instance graph without defining a data schema. Various graph database providers have developed their own query languages. However, no universal standard has been established yet [37,39]. Compared to LPGs, RDF graphs natively cannot assign attributes to relationships [37,39]. Furthermore, RDF graphs are schema-based. So-called ontologies define specific terminology to give instance graphs a clear structure and meaning. Following semantic web principles [40], ontologies are published online to be reused and combined to form complex data schemata [39]. Finally, RDF graphs can be queried and validated using well-established standards such as SPARQL [41] and the Shapes Constraint Language (SHACL), which are considered standards within the W3C community [41,42].

For the purpose of a Digital Building Twin Platform (DBTP), it is argued that RDF graphs are the most suitable choice for the overarching database system. First, construction projects are interdisciplinary, involving many stakeholders and constantly changing requirements, which require a flexible database that can adapt to the evolving needs. Relational databases, however, require intensive effort to apply schema changes. At the same time, objects and processes on a construction site are characterized by complex relationships among them. The ability of RDF graphs to assign unambiguous semantic meaning to relationships is very beneficial for representing these relationships. RDF graphs are considered more suitable for storing highly-complex DT data due to their capacity to enforce strict schema compliance and the possibility to built upon existing ontologies. With many parties and software tools involved, an unambiguous schema is necessary to provide full interoperability between them and avoid data duplication and misinterpretation. Furthermore, the data validation and query languages available for RDF, such as SHACL and SPARQL, give RDF a distinct practical advantage.

4. Reference architecture for digital twin construction

This section introduces the developed reference architecture for DTC. Taking into account the research gaps identified in Section 2.4 and the platform design considerations in Section 3, the architecture focuses on the data management layer of the DT platform. In high-level platform architectures, this layer is also referred to as the persistence layer [11], virtual space layer [20], or cyber level [12].

Fig. 4 gives a complete overview of the reference architecture. It comprises the following platform modules: data management, data conversion and injection, data validation, web API, digital twin services, user authentication, and the dashboard. Note that despite the figure depicting a high-level architecture, it includes modules that belong to other platform layers to provide a complete view of a platform for DTC. These modules are, however, only described briefly.

4.1. Data management module

The data management module (in the center of Fig. 4) is divided into three layers: data, information, and knowledge. Their content is fundamentally different from each other and is, therefore, stored separately. Considering the large variety of data introduced in Section 3.1, none of the existing databases can provide optimal conditions for all data types. For this reason, using a combination of databases for different parts of the DT's content is necessary.

An RDF graph is used as the overarching system and the initial access point to any content stored in the DT. It semantically connects the content stored in various databases. Furthermore, it allows connections to be formed across the data, information, and knowledge layers whenever necessary. The downside of dealing with various query languages and access modalities of the individual databases is mitigated through a common API layer that hides this complexity from the DT services and dashboard users.

4.1.1. Data layer

The raw data captured for a DT of the construction execution has various structures and file formats. Each type of data is managed in a database whose underlying model fits the data structure best. This means, e.g., that sensor data is stored in a time series database, point clouds and images are stored in an object storage. Even though it is possible to, e.g., store sensor data in a graph database, this is not recommended because of performance issues, as pointed out by Zhang and Beetz [30].

A catalog graph is proposed as an overarching approach for managing the data layer. This graph-based catalog saves metadata about measuring devices, individual measurements, and provides links to the database where the data is stored. In this way, it is possible to query the catalog graph to discover available raw data and their storage locations. This approach is inspired by previous research on catalog systems for urban digital twins [43] and automotive sensor data [44]. Based on their findings, a graph-based catalog system is also considered suitable for DTC.

The web API of the DTC platform simplifies access to raw data sources. It allows querying the catalog and the connected databases simultaneously so that the digital twin services do not need to be familiar with the query languages of the individual databases. Whenever new monitoring data is stored in one of the databases, the catalog graph needs to be updated accordingly (see right-hand side of Fig. 4).

4.1.2. Information layer

For the information layer, the RDF graph is used to store information about building elements, construction processes, and other construction-related entities. These objects have complex relationships, which can be described very well in the graph format. Entities in the information layer are usually changed infrequently, such as a building element whose status may be changed only a few times within several weeks. This is another characteristic that fits graph databases well. Rapidly-changing entities, such as the position of an equipment or construction worker, are part of the data layer and handled as described in Section 4.1.1.

As shown by Pauwels et al. [45], storing detailed geometric information in a graph dramatically increases its size, and the information cannot be interpreted efficiently. Therefore, it is preferred to store detailed geometry in a dedicated object storage (see the information layer in Fig. 4). Object storage is characterized by its flat architecture, suitable for large amounts of mutually independent files, thus enhancing scalability. Additionally, accessing files through a web API interface streamlines data management processes [46]. Similarly to the data layer, a node in the information graph can be directly linked to the location where its corresponding geometry file is stored within the object storage. The format in which the geometry files are stored in the database can be selected according to the use case and domain of interest. For instance, the Graphics Library Transmission Format (glTF) might be preferred for web applications, while the Wavefront Object File Format (OBJ) could be chosen if the geometry's texture is relevant [47,48].

PII, including as-designed models as well as as-planned schedules, which are usually managed by construction companies, must be translated into the platform-internal data schema to initiate the information graph. This conversion is handled by the translation and injection module. Only then can the information be interpreted correctly by

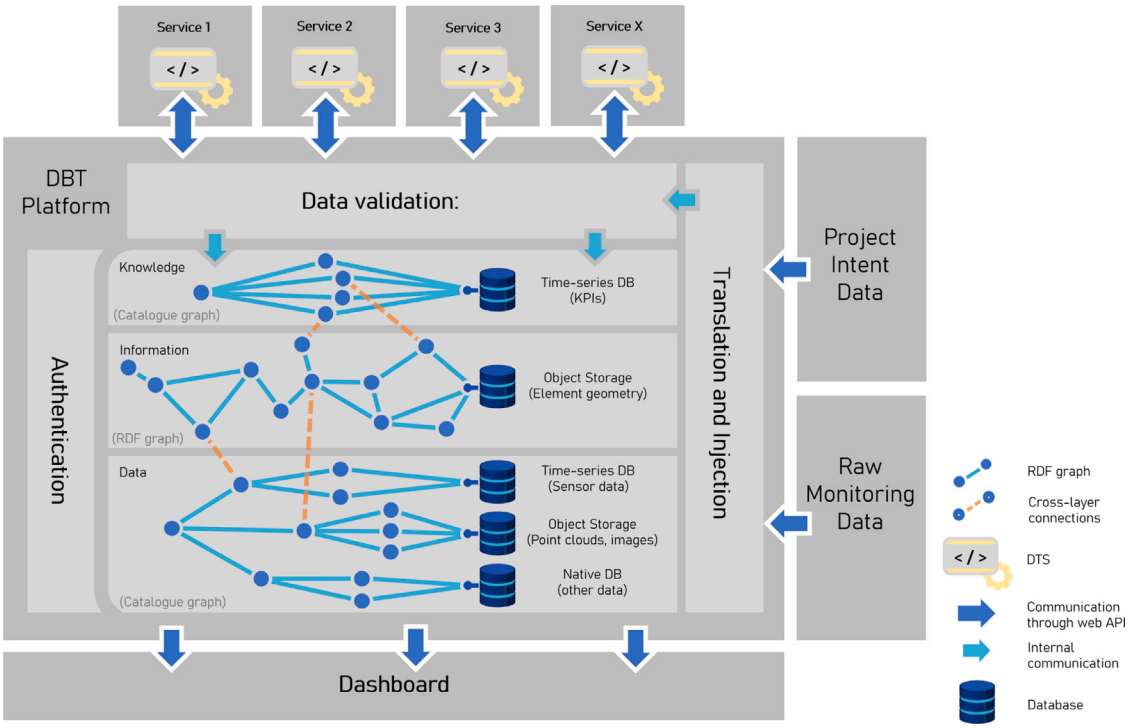

Fig. 4. Reference architecture for DTC.

the digital twin services and compared across projects. Once the PII is injected into the platform, the digital twin services can add information about the project status incrementally. Every time the PII is changed, the information graph must be updated accordingly.

4.1.3. Knowledge layer

The knowledge layer is structured similarly to the data layer. It considers KPIs as the primary type of knowledge. However, other forms of knowledge can also be incorporated. These can be for example observations of patterns, like the impact of high temperatures on the on-site productivity, or other insights derived from the information layer. In this paper, the emphasis is placed on KPIs, as they are already widely used and understood by construction experts, the end users of the proposed platform. A catalog graph is employed to store semantic information about the calculated KPIs, while the actual KPI values are stored in a time series database (see the knowledge layer in Fig. 4). Since KPIs are numeric values that are calculated frequently and specific to a particular time frame, time series databases are the most suitable option. The responsibility for calculating domain-specific KPI values lies with the services, as these calculations are not performed by the platform itself. Again, the web API simplifies queries to the knowledge layer, accessing the catalog graph and the corresponding time series database simultaneously.

4.2. Other DBTP components

Cross-layer connections

In most cases, connections between graph nodes occur within the same layer. However, cross-layer connections (as shown by the dotted lines in Fig. 4) can also be established, providing additional semantic information to better describe how specific nodes are related. For instance, a Global Positioning System sensor can be linked to the equipment it is installed on by connecting the sensor node in the data layer with the equipment node in the information layer. Similarly, construction site images (data layer) can be linked to the building elements (information layer) they depict, enabling advanced search capabilities when filtering for specific images, following the approach by Collins et al. [49]. The decision to enrich graphs in this way should be based on the specific characteristics of each use case and the requirements of the digital twin services. Technically, these layers can be implemented as named graphs that reside within the same database as the data, information, and knowledge graphs. Cross-layer connections are optional, as some digital twins can be adequately represented without them.

Services

The digital twin services are third-party services that analyze specific content about the DT to gain additional insights into the current status of the construction project or to process data to information, and information to knowledge. They use the web API to make data requests and add new information to the database. Digital twin services should be selected based on the primary challenges of the construction project. They can cover aspects such as progress monitoring, quality monitoring, equipment optimization, safety planning, and many others [50].

Validation

Due to the digital twin services that are deployed outside the DTC platform, the system must ensure that any content that is added by these services complies with the platform's data schema. Only then can the information be interpreted by others and related to other entities of the DT. The data validation is a critical module to ensure clear data structures. The validation can be directly integrated with the platform's API: every time a service tries to add content to the platform, it is checked first for compliance. The content is only added to the platform if it is compliant.

Dashboard

The dashboard of the DTC platform (see Figs. 5 and 6) uses the platform's web API to request information about the construction project and display it for managers, construction workers, and other site personnel. Depending on the type of dashboard, it can be read-only or used to update the DT based on some corrections or new decisions.

The dashboard should primarily offer functionalities to visualize information from the information and knowledge layers. In the information layer, key entities for the end user include processes and

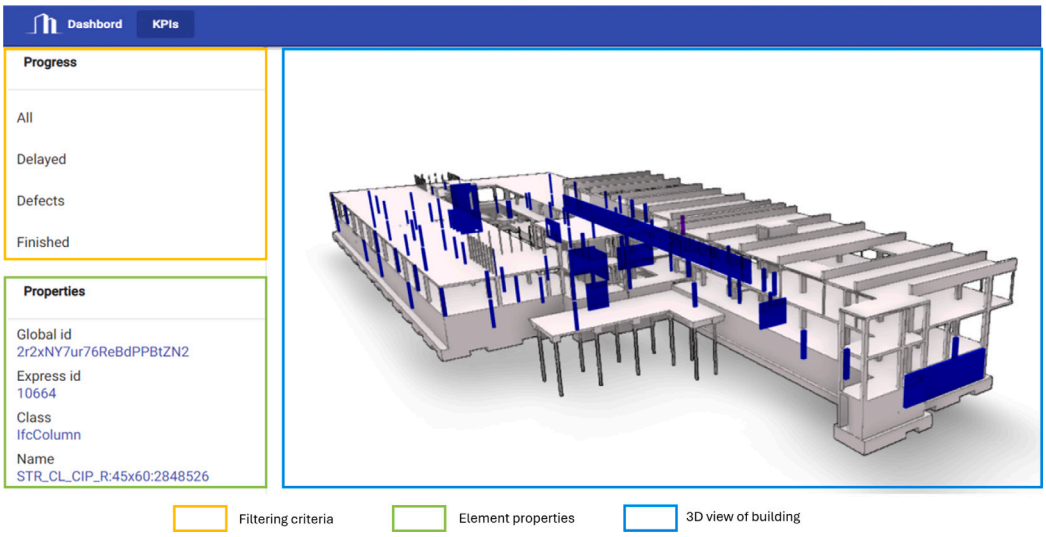

Fig. 5. Screenshot of the 3D viewer of the BIM2TWIN Dashboard [51].

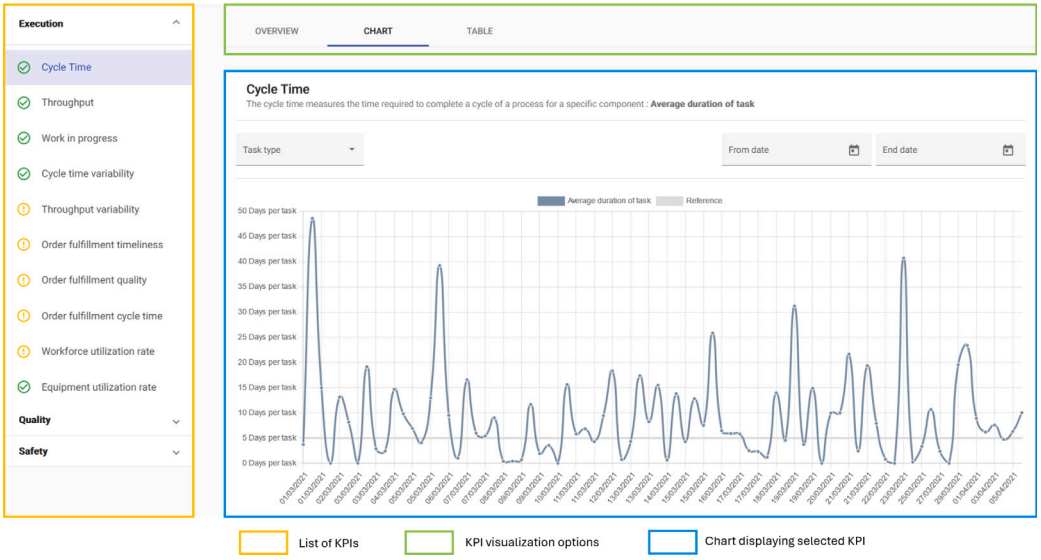

Fig. 6. Screenshot of the KPI view of the BIM2TWIN Dashboard [51].

building elements. This could be represented through tools such as a Gantt chart to display as-planned versus as-performed information and a 3D viewer to show as-designed and as-built building elements (see the light blue box in Fig. 5). Alternatively, a 4D viewer could combine these aspects, showing building geometry based on a specific point in time. These viewers can incorporate features typically found in BIM and project management tools, such as selecting building elements to view their properties (see the green box in Fig. 5), or applying color-coding to highlight statuses such as defects, delays, or other relevant factors (see the orange box in Fig. 5). Visualization of the knowledge layer is also essential, as it provides a summary of the construction site's overall status. In the case study, KPIs are the only type of knowledge considered. Displaying the progression of various KPIs over time, as illustrated in Fig. 6, can offer valuable insights to the end user. Visualizing data-related aspects is not considered critical for the end user. Still, e.g., displaying whether data collection is occurring as scheduled based on the predefined capture intervals can serve as an effective alert system in the event of sensor or network failures.

Authentication

The authentication module manages user profiles and access rights. Based on the project's organizational structure, access to specific database sections can be granted to designated individuals or groups, with permissions manually assigned by a central administrator. Access control is centrally managed through the platform's API, ensuring consistent enforcement of permissions throughout the system. Established frameworks and open protocols such as OpenID Connect and OAuth provide standardized solutions for authentication and authorization, enabling secure access control and user identity management across various components and services [52]. Given their widespread adoption and established implementation practices, these solutions are not further discussed in this research.

4.3. Comparison with state-of-the-art architectures

This section provides a detailed comparison to better illustrate how the proposed reference architecture stands apart from existing works. The approaches discussed in Section 2.2 are evaluated against the proposed architecture across several key characteristics: scope, data management structures, architectural pattern, communication technology, and geometry handling. These characteristics are summarized in Table 2. All presented approaches rely on well-established technologies. Consequently, their individual strengths and limitations stem from the

combination, arrangement, and interaction of these technologies rather than technological novelty. The following paragraphs delve into critical aspects in greater detail to underscore the specific improvements the proposed approach offers, particularly for digital twin construction.

The proposed approach significantly improves scalability by establishing a clear separation between data, information, and knowledge. By managing only the most essential elements within the respective data, information, and knowledge graphs, each component can be scaled more efficiently. Techniques like sharding can be applied to databases with exceptionally high data volumes, distributing data across multiple instances. For document and time-series databases, which often handle the largest data volumes, this division can be implemented seamlessly without compromising data integrity across instances.

While many existing approaches rely on distributed architectures, the proposed approach intentionally adopts a centralized architecture. This choice provides significant advantages, particularly in maintaining strict control over schema compliance. In the proposed design, schema compliance is enforced through a validation module directly integrated into the REST API, an aspect rarely addressed by other architectures. This centralized structure also prevents the coexistence of conflicting data versions, a common issue in microservice architectures where individual services manage separate parts of a digital twin's data. Additionally, it minimizes various types of latency that are typically associated with distributed systems.

Schema compliance is also a critical factor in enhancing the platform's interoperability. Additionally, the use of semantic web technologies with clearly defined ontologies establishes a strong foundation for interoperability within the proposed architecture. While some approaches also utilize semantic web technologies, the proposed method offers significant improvements in handling geometry, which enhances data integrity. Existing approaches often address geometry integration superficially. Many rely exclusively on IFC-based geometry. Typically, these systems link semantic objects to their geometry files only by ID, which limits the ability to integrate geometric information effectively in data queries. Furthermore, most current solutions do not address direct integration of as-built geometries from the field, often requiring geometry to be provided in IFC format. This introduces unnecessary complexity, as as-built geometry typically originates from laser scans and other measurement techniques that use geometry formats different from IFC. The proposed approach incorporates approximate geometric data directly into the information graph, enabling geometry-based queries. Furthermore, using the PLY format provides a more straightforward solution, as it is easier to generate and manipulate. Finally, the overarching graph databases facilitate deep data integration and discoverability, accommodating complex dependencies more effectively than limited mapping tables.

5. DTC ontology framework

In addition to the reference architecture, a data schema is proposed that describes how different types of content should be structured. As pointed out in Section 3.1, data, information, and knowledge require individual storage solutions. Therefore, they need their individual data schemata. In the RDF context, data schemata are defined by OWL ontologies, potentially combining multiple ontologies. While existing ontologies can cover some aspects of DTC, others still need to be defined for a holistic representation of construction sites. The main shortcomings of existing ontologies are summarized in Section 2.4. In this section, the set of ontologies to be used for data, information, and knowledge are introduced. For the information part, the newly developed Digital Twin Construction Ontology is presented, filling the identified research gap.

5.1. Ontologies for the representation of information

5.1.1. Digital Twin Construction Ontology

The Digital Twin Construction (DTC) Ontology is a new process-oriented ontology. The main contents of this ontology, based on the identified research gap, are the following:

- Construction processes at various levels of granularity
- Process sequences and preconditions to facilitate dynamic rescheduling
- Explicit representation of project intent and status

In the frame of the BIM2TWIN project, several workshops were held with construction professionals and domain experts to elaborate the core concepts of DTC from the point of view of various domains and use cases [50,54]. As a starting point, identified classes, relations, and attributes were captured in Unified Modeling Language (UML). In the second step, the UML diagrams were translated manually to an OWL ontology using the ontology modeling software Protégé [55]. This work builds upon and significantly extends the first schema version, originally presented by Schlenger et al. [56].

The DTC Ontology intends to define the core concepts of a DT of the construction project and should be extended through domain and use-case-specific ontologies. A critical design decision is how to represent project intent and status. The following factors were essential in determining a suitable representation:

(1) Although simple one-to-one correspondence between as-planned process instances and as-performed instances is common, there are also scenarios where required rework or unexpected site conditions result in several as-performed processes that correspond to a single as-planned process. Many-to-one relationships occur frequently, for example, when work on a zone is interrupted while the crew is withdrawn to some other location to perform other work. Here, one as-planned activity may need to be related with multiple as-performed activities. These cases cannot be represented with a single process instance using different attributes for as-planned and as-performed start and end times. The same applies to as-designed and as-built elements, which are not necessarily in a one-to-one relation.

(2) Related to this issue, a clear separation of intent and status would facilitate the representation of related intent and status instances at differing granularity levels. Commonly, construction schedules contain only rough planning, while very detailed information regarding the construction status is required to aggregate as-performed tasks into composite or cumulative as-built activities that are directly comparable with the planned schedule.

(3) Furthermore, defining classes for a specific context (intent, status, or both) allows enhanced data validation on the schema level. In this way, project intent and status can have separate validation rules. As an example, one can check that the defect class is only used for PSI, since defects are not part of project planning. In the case of user-defined contexts, such validation rules would be dependent on user-defined strings. As a result, all validation rules specific to PSI or PII become project or company-dependent and are not generic.

(4) Finally, the separation of intent and status has the simple advantage of clarity. Using different classes allows unambiguous addressing without requiring implementation of case-specific behavior.

For these reasons, the DTC Ontology represents the project intent and status as two distinct purpose-specific hierarchies. Where existing ontologies use two sets of attributes for the same class or user-defined contexts to differentiate between intent and status, the DTC Ontology defines separate classes specific to the project intent and the status. There are, for example, two primary classes for resources: `AsPlannedResource` for representing the project intent and `AsPerformedResource` for the project status. Some classes can be used in both contexts.

Although classes are mostly separate and distinct, the aim is to connect PII and PSI. The relationship `intentStatusRelation` was

Table 2
Comparison of existing reference architectures with the proposed approach.

Ref.	Domain	Data structures	Architecture pattern	Communication technology	Geometry handling
[12]	Smart factories Industry 4.0 Cyber–physical systems	Centralized data management	Not specified	Real-time requirements	Not specified
[11]	AI-driven systems Smart factories Manufacturing systems	Combination of structured and unstructured data Data lake as key data management component	Not specified	REST API Publish/Subscribe	Not specified
[20]	Generic digital twins IoT integration Factories, products, etc.	Combination of structured and unstructured data Both equally important	Not specified	Not specified	2D and 3D models No management guidance
[6]	Digital twins for building and infra-structure maintenance	Knowledge graph-based Ontology-based IoT platform integration	Micro-service architecture Docker	REST API Event-based system Message broker	IFC-based Conversion to GLB Linking via ID
[5]	Operation and maintenance of buildings and cities	Data from commercial software transferred to NoSQL database Use of mapping tables	Centralized, layered architecture	REST API	IFC-based
[53]	Operation and maintenance of buildings	Central RDF database Ontology-based NoSQL DBs for sensor data, etc.	Micro-service architecture Docker	REST API (GraphQL) Event-based system Message broker	IFC-based Conversion to XKT
This work	Digital twins of the construction phase	Separation of data, information, and knowledge Central RDF database Connection to other DBs	Centralized Microkernel architecture	REST API	Integration into RDF database PLY files for detailed geometry

designed for this purpose. Whenever possible, an instance of the project intent should have at least one counterpart in the project status and beyond that, allowing one-to-many relationships.

The terms defined in the DTC Ontology are arranged into five groups, facilitating a structured presentation for better comprehension and clarity.

Construction processes and preconditions

Construction processes are the central part of the DTC Ontology. They are specified in three granularity levels. The `WorkPackage` forms the topmost level. It contains details regarding the applied construction method and can be understood as an aggregation of lower-level processes. The `Activity` forms the second level. Every work package comprises several activities. Each activity describes a single construction step to build a group of objects, like placing formwork or pouring concrete. Activities are further decomposed into tasks. Each `Task` represents an activity related to a singular building element. Work package, activity and task are classes of the project intent. The corresponding project status classes are `Construction`, `Operation` and `Action`. The `ConstructionSchedule` class allows grouping all processes belonging to one construction schedule. Multiple schedule instances can be created for schedule alternatives, simulation scenarios, or other cases. A `Precondition` can be related to any process to describe the requirements that must be fulfilled before this process can be started. Explicit representation of these dependencies is essential for dynamic rescheduling based on the current status of the construction site. There are dedicated subclasses for preconditions related to a process depending on another process, a resource, a zone, an external factor, and information. Furthermore, an enumeration is defined for `ProcessPreconditions` to differ between process sequences like finish-start, start-finish, and so on. All precondition classes shall only be used for the project intent. An overview of the classes on the project intent side is given in Fig. 7.

Building structure

The classes defined by the well-established BOT ontology are reused to describe the building structure. BOT organizes building elements hierarchically according to the topological structure of the building, excluding geometric descriptions [27]. Since BOT does not differentiate as-designed elements from as-built elements, the Boolean attribute `isAsDesigned` is introduced to tell these elements apart. Furthermore, BEO is used to differentiate various building elements, such as walls, columns, and slabs [26]. Fig. 8 gives an overview of the classes related to the building structure.

Resources

According to the DTC Ontology, the resources required for an element's construction process include labor (work crews and individual construction workers), equipment such as heavy machinery and handheld tools, materials, and temporary equipment, including formwork and guardrails (see Fig. 9). The resource classes are intended for modeling the actual resources on the construction site, while the `ResourceAssignment` assigns a specific quantity of a resource or of a resource's capacity within a defined timeframe to a construction process. For this reason, one resource, like an excavator, can be assigned to various processes by defining several resource assignments with varying time intervals. All resource classes exist in the project status and intent, with the prefix as-planned or as-performed. As-planned resources are usually planned only roughly, while as-performed resources can be tracked in close detail, e.g., regarding their location, activity, or inactivity.

Working zones

A `LocationBreakdownStructure` formalizes how the construction site is divided into specific work areas [57]. This breakdown can vary between different processes, e.g., concreting works are performed in zones that are quite different from zones suitable for indoor works. For this reason, multiple location breakdown structures are used on the same construction site. The individual areas of a location breakdown structure are defined as `WorkingZones`. A zone might be equivalent to a particular building, story, or space, but that accordance is coincidental rather than conceptual. Unlike the classes `bot:building`, `bot:storey`, and `bot:space`, the working zones hold geometric information about their location and extent. Fig. 10 shows the definition of working zones in a UML diagram.

Defects

Lastly, defects are understood as situations where the condition of a building element deviates from the planned condition, in terms

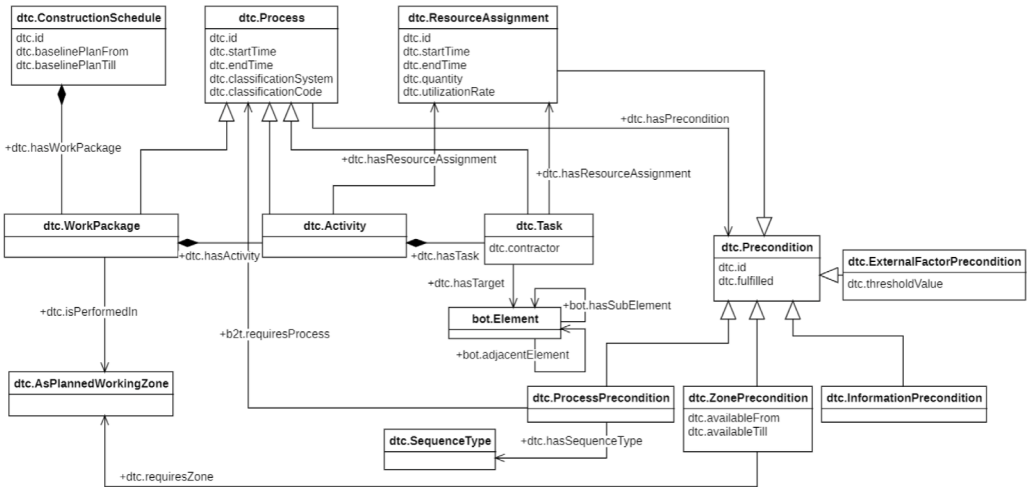

Fig. 7. UML diagram of process-related classes on the project intent side.

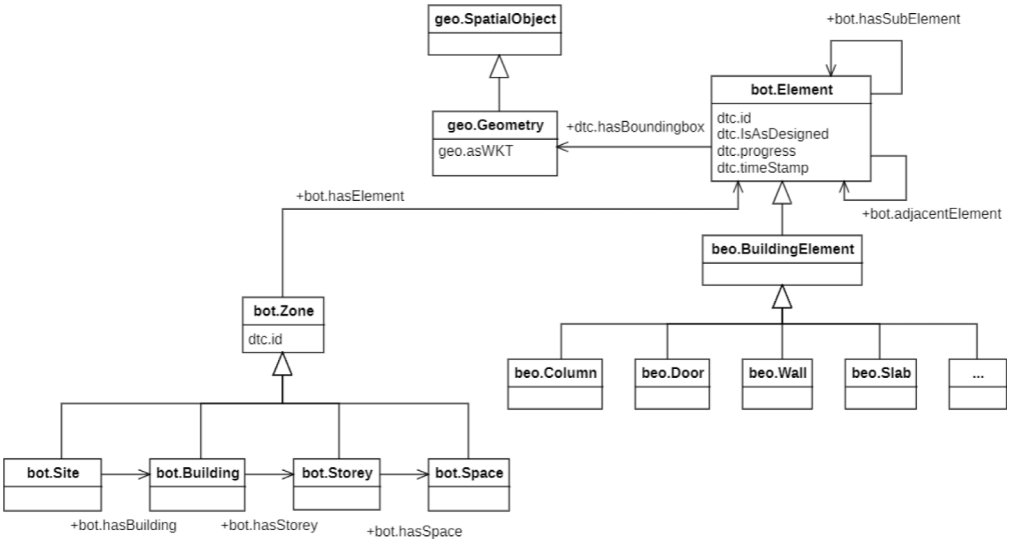

Fig. 8. UML diagram of the classes related to the building structure.

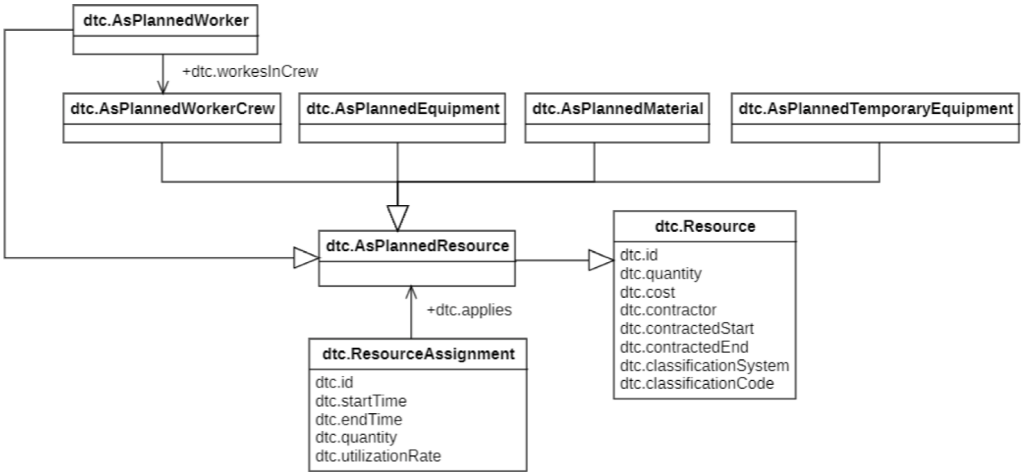

Fig. 9. UML diagram of the resource-related classes on the project intent side.

of quality, quantity, location, appearance, function, and more. This includes `VolumetricDefects` for deviation of size or shape, `PositionDefects` for misplaced elements, and `SurfaceDefects` for superficial damages, e.g., resulting from erroneous execution (see Fig. 11). Defects occur only in the project status information, as they are not planned in advance.

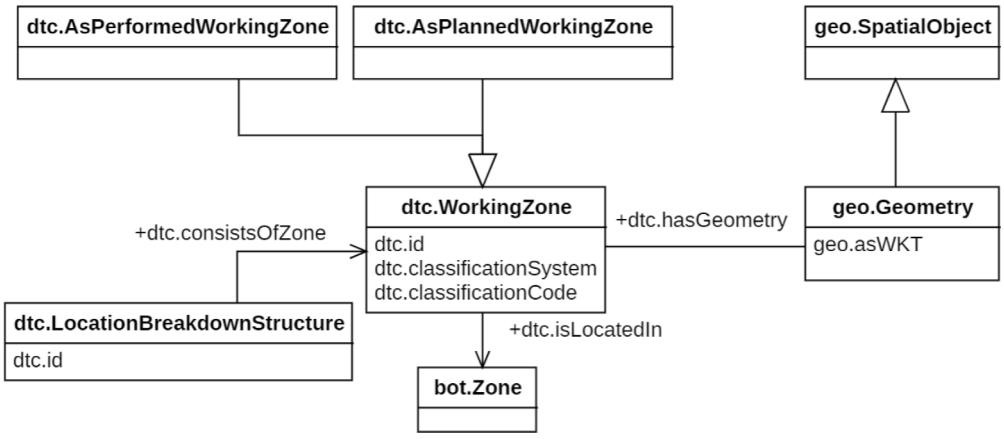

Fig. 10. UML diagram of the classes related to the working zones.

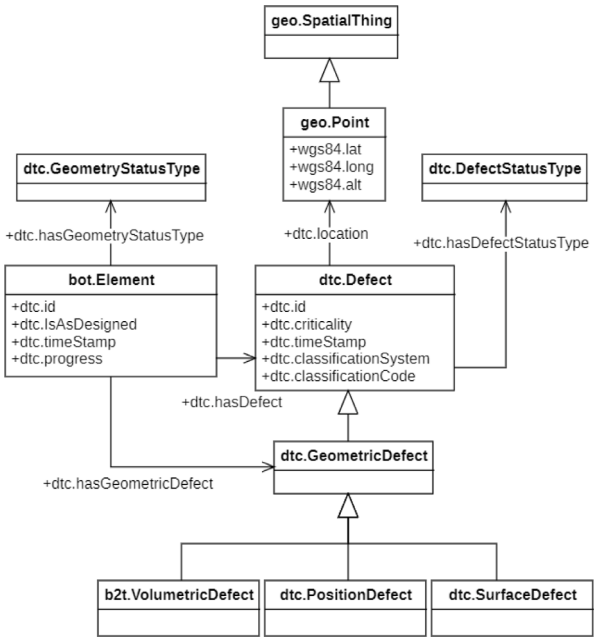

Fig. 11. UML diagram of the defect-related classes.

The classes of the five groups introduced here are also interrelated. For details, the reader is referred to the online documentation of the ontology available at https://dtc-ontology.cms.ed.tum.de/ontology/. This website provides detailed documentation following the World Wide Web Consortium best practice recipes for publishing RDF vocabularies [58].

5.1.2. Representing geometry in RDF

Up to this point, Section 5 focused exclusively on semantic information about the DT of the construction site. In addition to semantics, geometric information is essential for some use case. Use cases related to progress and quality monitoring of building elements, for example, require detailed geometry. Moreover, the exact extent of a working zone can be relevant to determining which building section a particular construction worker is working on, to name another example. Wagner et al. [59] have compiled various state-of-the-art approaches to integrate geometric information into RDF graphs.

In the context of DTC, selecting an approach for representing geometry must address requirements such as minimizing the size of the central RDF graph for efficient graph queries and ensuring flexibility in geometry format to accommodate diverse use cases belonging to various domains. Taking into account these requirements, a mixed approach between approaches 1 and 4, according to the numbering of Wagner et al. [59], was selected. Detailed geometric information is kept outside the RDF graph in non-RDF files, referenced in the graph by their file path (approach 4). Additionally, the bounding box information is directly stored in the graph to enable simple spatial queries, e.g., with GeoSPARQL.

It is assumed that many digital twin services stemming from different domains need to access geometric information. Some of these services might be generic and not specific to the construction industry. For this reason, a very versatile but construction-specific representation like IFC is not deemed suitable. In contrast, triangular geometry meshes are straightforward to interpret and process, domain-agnostic, and thus regarded as a suitable representation. For the non-RDF geometry files that are created for every element individually, the Polygon File Format (PLY) was selected [60]. This open geometry format describes geometries based on individual vertices and faces. It has a simple structure, but still allows for the integration of user-defined properties. The Well-Known-Text (WKT) representation integrated into the GeoSPARQL ontology was selected for the RDF representation of bounding boxes based on the recommendations of Pauwels et al. [45]. Spatial query languages such as GeoSPARQL and BimSPARQL support WKT [61]. Finally, for creating a reference from the RDF graph to the location of the PLY files, the File Ontology for Geometry formats (FOG) [62] is used. Fig. 12 displays the representation of a building element in the RDF graph according to the proposed approach.

5.1.3. Combined information representation

In combination, the following set of ontologies is proposed to represent the information layer of the DT. As the basis, the newly developed DTC Ontology is used, which focuses on processes and their inputs and outputs. DTC Ontology relies on BOT and BEO concerning the building structure and its elements. To cover geometric aspects of building elements and working zones, GeoSPARQL, World Geodetic System 1984 (WGS84), and FOG are applied.

This set of ontologies serves as the recommended baseline and was utilized in the case study presented in Section 6. However, leveraging semantic web techniques provides the flexibility to adjust and expand this base schema. Depending on the chosen use cases, additional ontologies can be integrated to represent more specialized information within the RDF graphs, representing data, information, and knowledge. Crucially, any new ontology introduced must be aligned with at least one of the other used ontologies. This alignment can be achieved by defining equivalent classes across different ontologies, establishing new relationships between them, or creating super-/subclass dependencies. For example, using both BOT and BEO, it was defined

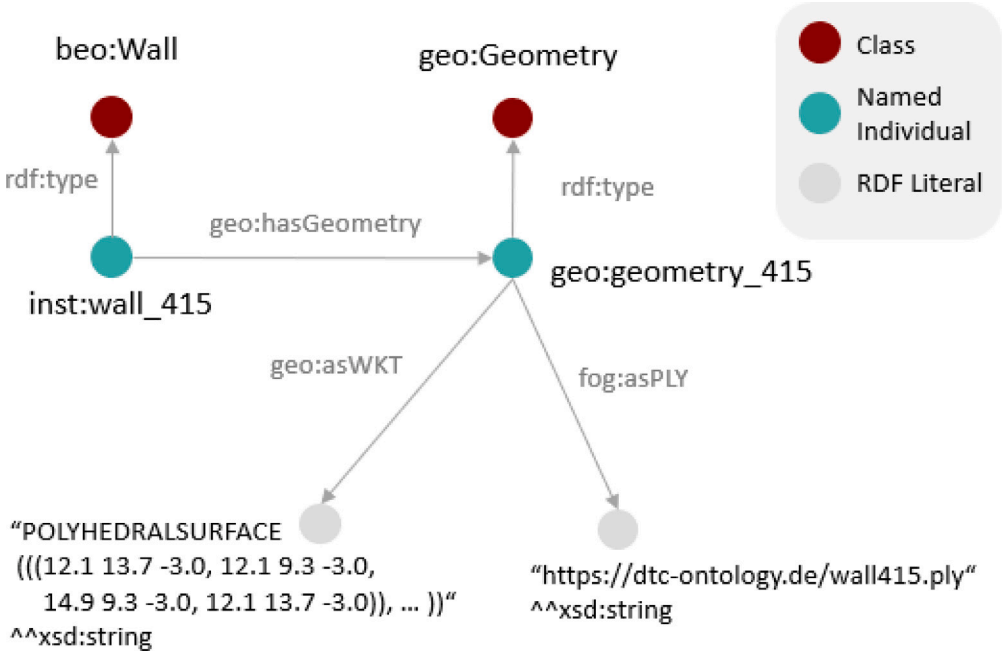

Fig. 12. Example for implemented geometry representation in RDF.

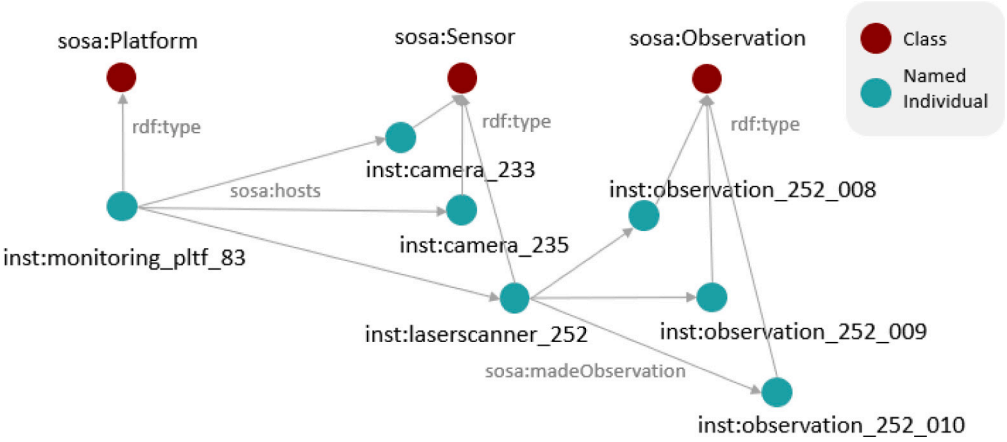

Fig. 13. Representation of sensor systems with SOSA/SSN.

that in the context of a digital twin for a construction site, the class `beo:BuildingElement` is treated as a subclass of `bot:Element`, aligning the two ontologies. This approach allows the set of ontologies to be tailored to each construction project, structuring the RDF graphs for data, information, and knowledge.

5.2. Ontologies for the representation of data

The data layer is represented through an RDF graph containing metadata about the raw monitoring data and forming connections to the databases where it is stored. To describe this RDF graph, the SOSA/SSN ontologies are used [29]. Fig. 13 shows a simple example graph of a monitoring system with two cameras and a laser scanner, including several observations. This forms the basis for the data catalog.

There are several options for describing the sensor measurements (`sosa:Observation`). For simple measurements that result in, e.g., a numeric value or a string, the result can be directly stored in the RDF graph with the relationship `sosa:hasSimpleResult`, which is only recommended for small amounts of data. Option 2 covers sensor data stored in external files. Here, the Data Catalog Vocabulary (DCAT) is used. The class `dcat:Dataset` and `dcat:Distribution` provides means to describe datasets and the location and format of their storage [63]. In the DTC scenario, `dcat:Dataset` is seen as a subclass of `sosa:Result`. Fig. 14 presents a small example graph.

A third option describes data stored in external databases. This option is similar to option 2, but uses `dcat:DataService` instead of `dcat:Distribution`. It provides attributes for describing API endpoints. These descriptions help the DBTP to establish connections to external databases and access them on request.

5.3. Ontologies for the representation of knowledge

The knowledge layer is structured similarly to the data layer. The KPIs are mostly numeric values that are calculated regularly. This structure fits well for storing them in a time series database. The different types of KPIs are defined in RDF using the saref4city ontology, including the `s4city:KeyPerformanceIndicator` class [64]. The time series database where the corresponding KPI values are stored is linked by describing the database endpoint with `dcat:DataService`. This connection follows the same structure as explained in Section 5.2.

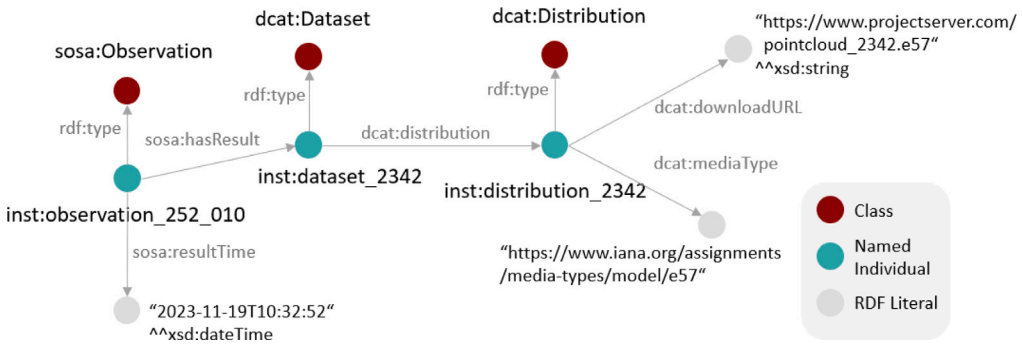

Fig. 14. Option 2: RDF representation of observations in externally stored files.

6. Case study

The publicly available ConSLAM dataset from [65] was used as a case study to validate the proposed platform architecture and data schema. This dataset consists of laser scan data collected during the construction of a shopping mall in London. For the case study, a dedicated platform instance was set up, involving the implementation of individual platform components and converting PII from the planning phase into the proposed schema to initialize the building graph. A progress monitoring service was developed to demonstrate how third-party services can be integrated into the DBTP. This point cloud processing service updates the PSI in the DBTP. In this case, the PSI is the information about the building elements identified in the point cloud data, specifying the date of first detection. The service then calculates a schedule-related KPI, which represents the only knowledge type considered in this case study and is thus the sole content stored in the platform's knowledge layer.

It is important to note that discussions with construction companies revealed that construction processes are typically planned for groups of building elements rather than for individual elements. For example, a single process may be defined for constructing all columns on a floor without specifying the order in which each column is built. While it is technically feasible to identify the progress of individual columns based on point cloud data, this level of detail is of little practical interest to construction companies. Therefore, progress information is aggregated to match the level of detail used in the construction schedule. Additionally, building elements are classified as completed or not existing, which is sufficient to test the proposed data structures.

This validation focuses on the interaction between the service and the platform, rather than the accuracy of the service's results. In other scenarios, the point cloud-based progress monitoring service could be replaced by an image-based service or another technology. The DBTP's role is to collect insights from various services and transparently present these results, including any potential inaccuracies, to the end user. Ensuring accuracy depends on selecting appropriate services at the project's outset, and end users must consider the provided information alongside possible deviations when making managerial decisions. While progress monitoring is a critical component of a DT for construction, a complete DT system would integrate multiple services, each addressing different aspects of the site or project. Therefore, this case study aims to demonstrate the platform's feasibility rather than its completeness.

6.1. General project information

The ConSLAM dataset, collected by Trzeciak et al. [65], was compiled during the reconstruction of Whiteley's shopping mall in London. This building comprises a six- to nine-story structure housing retail, residential, and recreational areas (see Fig. 15). Utilizing a handheld laser scanner, five scans were conducted between March and August 2022 [65].

Table 3
Point clouds.

Scan	Date	Based on original scans
1	2022-03-01	Sequence 1, first half
2	2022-03-15	Sequence 1, second half
3	2022-03-29	Sequence 2, first half
4	2022-04-12	Sequence 2, second half
5	2022-04-26	Sequence 1, 2, and 3

Based on the progress identified in the point clouds and schedules of similar construction projects that were monitored within the frame of the BIM2TWIN project, a synthetic construction schedule was created for the shopping mall. The University of Cambridge research team also provided the building's IFC file. Together, the schedule and IFC file constitute the PII used in this case study. Since the laser scans of the ConSLAM dataset cover only completed floors, all structural elements, such as columns and ceilings, were already constructed. Some scans were cut into several sections to simulate the earlier processes. Table 3 summarizes how the original scans were used to create several sub-scans based on the scanning sequences performed by Trzeciak et al. [65]. These scans constitute the data stored in the data layer. Overall, this still results in using the whole dataset but distributes the progress over more discrete steps. Although these artificial steps do not provide a fully realistic progress monitoring scenario, they still provide a realistic scenario from a data management perspective, which is the primary purpose of this case study.

6.2. Platform setup

The DBTP was implemented following the reference architecture presented in Section 4. The implementation was done in C# using the ASP.Net Core framework due to personal preference. This handles the coordination of the individual platform components and provides the web API for communication with external services (see Fig. 16). A similar implementation can also be achieved in other programming languages.

Concerning the instantiated databases, GraphDB[1] was used to store all RDF graphs, MinIO[2] was used for object storage like point clouds and geometry files, and InfluxDB[3] was used to store KPIs. To enhance the platform's API with data validation functionalities, the dotNetRDF[4] library was applied. It is based on validation rules formulated in SHACL. These so-called SHACL shapes were automatically derived from the ontology files that were part of the internal data schema, checking the domain and range of relationships and attributes. The generated

[1] https://graphdb.ontotext.com/
[2] https://min.io/
[3] https://www.influxdata.com/products/influxdb-cloud/
[4] https://dotnetrdf.org/

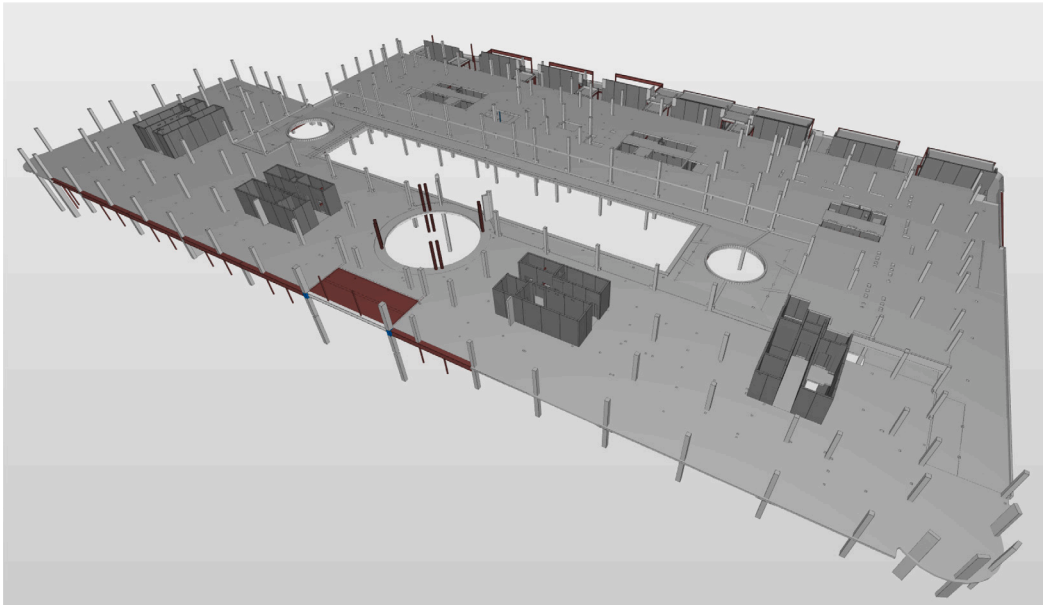

Fig. 15. BIM of Whiteley's shopping mall in London [65].

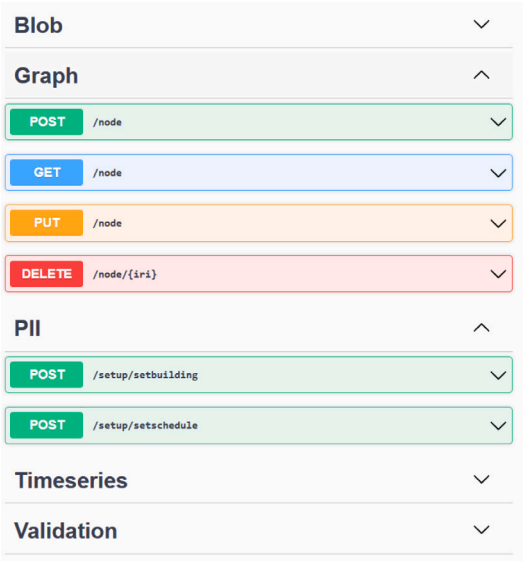

Fig. 16. Swagger API displaying a subset of the available API calls.

validation rules related to the DTC Ontology are also published at https://dtc-ontology.cms.ed.tum.de/ontology/ in Section 5: SHACL shapes.

For the conversion and injection of PII, a set of existing software tools and libraries was used together with newly implemented components. Initially, the IFC file of the ConSLAM project was manually checked for modeling errors and then converted to RDF with the IFCtoLBD converter [66]. This converter follows the topological structure of the buildings defined in BOT. Since the converter does not consider any geometric information, the geometry was added in a second step. With the help of the C# libraries xBIM[5] and tinyply,[6] the geometric information was read from the IFC file, converted to PLY files for the individual building components, and then attached to the topological graph as described in Section 5.1.2. Afterwards, the schedule provided as an Excel file was converted to the RDF representation using the DTC Ontology, and linked to the corresponding building element nodes created in the first step. Since DTC is a new ontology, the schedule conversion had to be implemented from scratch. Depending on the planning software and methodology applied by the construction company, the structure of the schedule file can vary significantly. This means that the implemented schedule conversion might need project-specific adaptations. Due to the lack of a standardized schedule description [67], this cannot be avoided in the scope of this paper.

Given that the only raw data available in this case study were the five point clouds captured by Trzeciak et al. [65], it was chosen to add them manually to the data catalog graph and the MinIO database through the platform's API. For more data-intensive monitoring projects, an automated data integration pipeline is recommended.

Fig. 17 gives an overview of the platform components implemented for the case study and the use of existing software components. For further details on the implementation, the reader is referred to the GitHub repository, which can be accessed at https://github.com/jschlenger/DBTP. The user authentication module and the dashboard were not implemented as part of the case study, since they were not considered relevant for evaluating the DTC Ontology and platform architecture from a data management perspective.

6.3. Digital twin service for point cloud-base progress monitoring

The progress monitoring service is fully integrated with the DBTP via the web API. This integration includes fetching, sending, and updating data from and to the DBTP. Before any processing can be executed, however, the point clouds need to pass through a few pre-processing steps.

Pre-processing

The processing in this step ensures the appropriate data quality, as well as their spatial alignment with respect to the BIM. The point cloud data are first down-sampled to make the operations tractable. In some cases, the data will also be cropped by hand to remove pieces unrelated to the building. Finally, an essential pre-processing step is global registration with the BIM model. Depending on the quality of the point clouds and the degree and number of local discrepancies between the point cloud data and the BIM, different methods may be used. These range from rough alignment by hand and refinement via

[5] https://docs.xbim.net/
[6] https://github.com/yk35/tinyply.net

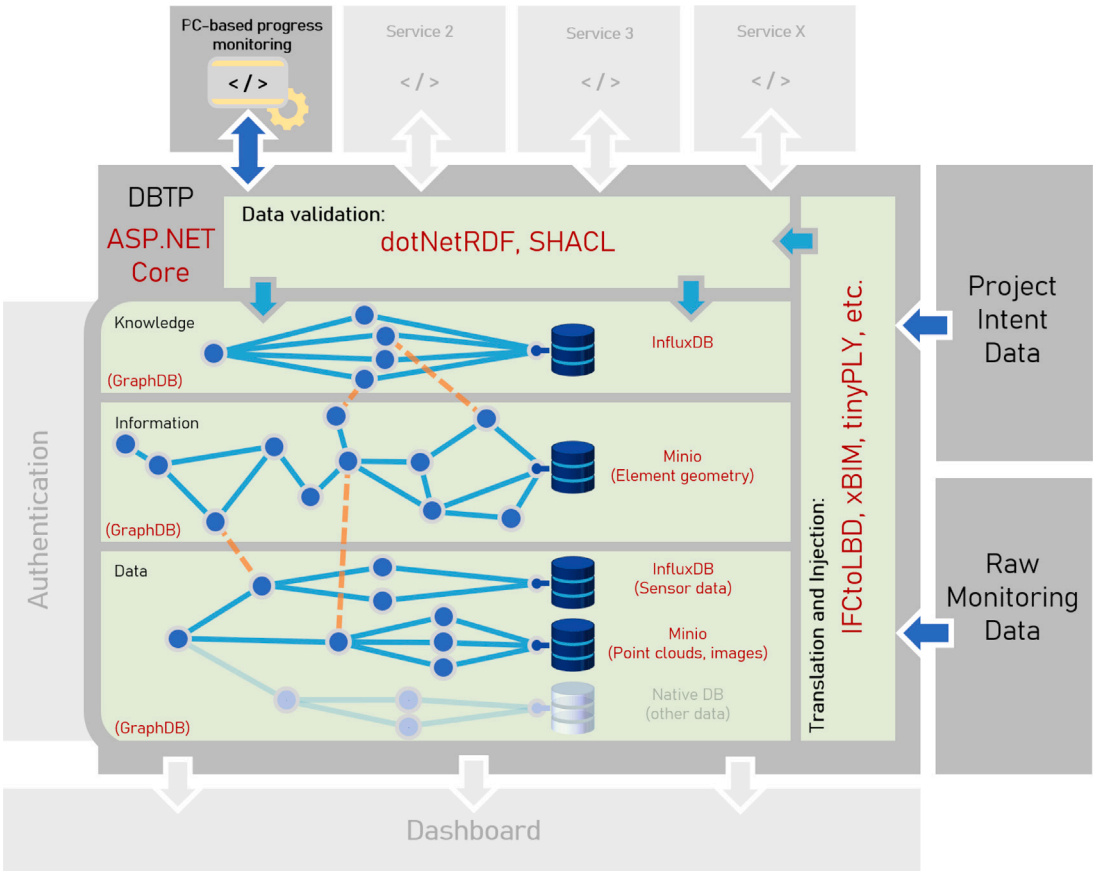

Fig. 17. Technologies applied to implement the proposed reference architecture. Components that were not implemented as part of the case study are grayed out.

Iterative Closest Point to methods more adapted to the building-related data, such as the one proposed by Monasse et al. [68].

For this case study, the BIM model and point cloud data were first manually aligned and then fine-tuned using the Iterative Closest Point method. This process required iterative manual adjustments followed by the Iterative Closest Point method, continuing until the root mean square error (see $RMSE_d$ in Trzeciak et al. [65], Definition 1 therein) was below one centimeter. Given the small number of point cloud sets involved, more extensive methods were not explored, as the total time for an experienced person did not exceed one hour.

Object detection

After pre-processing, the data are ready to be analyzed further. In the present case, the progress monitoring service fetches as-designed elements from the DBTP together with their geometry files to identify the presence of these elements in the point cloud data. Each as-designed element is used as a query element for the object detector, which verifies whether the given element can be identified in the point cloud data. More precisely, this process consists of decomposing the as-designed, query element's geometry into geometric primitives such as planes. In the second step, these primitives are searched in the point cloud data within a Region-of-Interest, which is defined by an enlarged bounding box of the as-designed element in question. The enlargement is necessary because the global registration performed during the pre-processing cannot resolve local displacements caused by inaccuracies introduced in the construction process. The object detection algorithm is based on a metric that evaluates the detection quality and decides whether the output represents a meaningful subset of points. It is worth mentioning that in the case of the point cloud data without normal vectors, the detection algorithm can reconstruct mostly globally oriented normal vector fields of the detected subset of the point cloud data, which is essential for the next processing step. Additional details regarding the detection method employed fall outside the scope of this study and are not essential to the primary objectives being addressed.

Mesh reconstruction

Due to their structure, point clouds are cumbersome for later storage and processing or direct usage for digital twins. Indeed, the preferred type of geometry representation for such an application is reconstructed surfaces represented in this case by polygonal meshes, preferably watertight meshes. In contrast to dense 3D point clouds, polygonal meshes do not require extensive storage space and are relatively easier to visualize. In the described pipeline, we use the kinetic algorithm proposed by Bauchet and Lafarge [69]. The algorithm is robust to noise and missing elements and well-suited for building elements commonly found in architectural BIM models.

Progress update

Once the geometry of the detected as-built element has been reconstructed from the point cloud data, the DBTP is updated with the newly identified information. This includes creating a new as-built node in the information graph, linking it with the respective as-designed element node, and uploading a PLY file containing the as-built geometry. It needs to be noted here that since the as-designed elements were used as query elements for the object detector, there is a straightforward association between these elements and their as-built counterparts. As soon as all the as-designed nodes in question are processed and the DBTP is updated accordingly, the as-performed nodes are generated based on the measurement date of the point cloud data.

Progress monitoring with ConSLAM

The computation of progress on the dataset proved to be challenging, mainly due to the imperfections in the data – which are expected from scans in a construction environment – some of the elements could not be detected correctly, which prevented marking corresponding

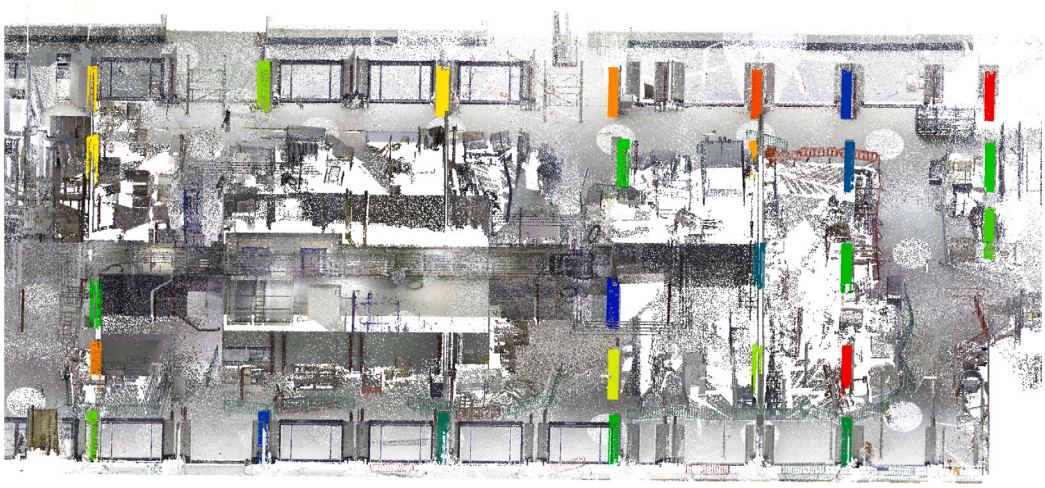

Fig. 18. Visualization of a part of the ConSLAM dataset with detected columns highlighted in different colors, chosen arbitrarily for improved clarity.

work packages as finished solely based on the results of the detections. To overcome this issue, inter-work package dependencies were traced, and rules based on the completeness of other connected work packages were introduced. All this together allowed to compute a progress-related KPI, discussed in detail in the following section. Fig. 18 illustrates detection results on the ConSLAM dataset, where the detected column instances are highlighted with different colors for a clear visual effect.

6.4. KPI calculation

Based on the as-built and as-performed information added by the progress monitoring service, KPIs can be calculated to summarize the current status of the construction site and provide construction managers with a quick overview of their projects. Usually, the KPI calculation is based on comparing the PII with the PSI, identifying the deviations between the two. Otherwise, KPIs not directly related to the PII can be interpreted by observing how they change over time or comparing them with similar construction projects.

In the ConSLAM case study, one KPI related to progress monitoring was defined, representing the only type of knowledge considered here. This ratio of days of delay to total number of days per work package KPI is calculated for every work package individually by dividing the number of days of delay by the total number of days since the start of the as-planned process. Here, delay is understood as the number of days a work package was finished after the as-planned end. Due to a lack of information in the case study data, it is assumed that the actual start date of the work package corresponds to the as-planned start date. This gives a general overview of schedule adherence, allowing construction managers to prioritize resources and investigate the root causes of delays, such as resource constraints, weather conditions, or subcontractor issues.

For the calculation of the KPI, information regarding the processes, elements and their status was requested from the DBTP through its web API. Based on this information, the KPI was calculated for every work package individually by comparing the as-planned with as-performed information. Finally, the resulting KPI values were sent back to the platform via the web API. This results in the KPI being stored in the InfluxDB time series database. The whole process is triggered every time updates regarding the as-performed part of the digital twin are received. In the case study, this corresponds to every time the progress monitoring service processes a new laser scan.

Fig. 19 displays the KPI values calculated for some of the work packages. These were calculated for five dates, corresponding to the dates when laser scans were captured, and PSI was added to the graph by the progress monitoring service. Between the five scan dates, the KPI values are linearly interpolated as a simplification. Values non-equal to zero means that the work package is delayed. The larger the value is, the greater the delay. For the work packages that were identified as finished, the KPI value remained constant since the process was terminated, and no further delay can be experienced. Although the temporal resolution of the scans is relatively low, general tendencies can be observed and the suitability of the proposed data structures can be confirmed.

6.5. Data availability

The data, information, and knowledge graphs created for the case study are published on GitHub at https://github.com/jschlenger/DBTP together with the KPI values exported from InfluxDB and the geometry files stored on MinIO. A similar setup using the same data structures was implemented on three BIM2TWIN pilot sites. Due to non-disclosure agreements, their data cannot be published. The main learnings from the ConSLAM case study and BIM2TWIN pilot site are presented in the following discussion.

7. Discussion

Structuring the DT into distinct data, information, and knowledge layers, with an RDF database as the overarching system and other connected databases, was identified as a versatile solution for effective data management, well-suited for DTC. This architecture provides a generic basis applicable to any construction project and addresses RQ1. Existing ontologies such as BOT, BEO, GeoSPARQL, SOSA/SSN, and DCAT were relied upon to organize the content in these three layers. However, no existing ontology adequately represented project intent and status, considering more than one-to-one relationships between them. The DTC Ontology was developed and published following W3C best practices at https://dtc-ontology.cms.ed.tum.de/ontology/ to fill this gap. Depending on the construction project and use cases, the proposed ontologies can be replaced and extended with others to adapt to project needs, answering RQ2. Compared to other ontologies, the DTC Ontology defines specific classes for project intent and status, separating the two. Linking intent and status while accounting for unequal numbers of instances on both sides enables comparisons even in complex scenarios. This principle drives the calculation of deviations, which is commonly used to determine KPI values. This provides an answer to RQ3. Addressing RQ4, the reference architecture and the ontology framework, in combination, provide a well-defined structure for organizing diverse content related to the execution of construction

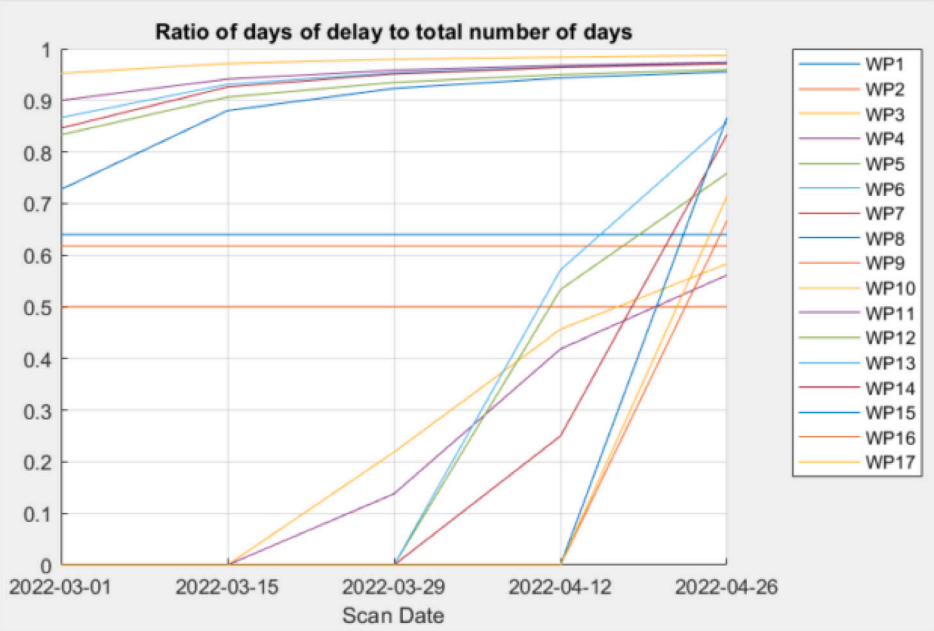

Fig. 19. KPI values calculated for the ConSLAM dataset based on the progress identified by the progress monitoring service.

projects. The validation module ensures that all data inserted follows this structure, ensuring interoperability between digital twin services. Nevertheless, several limitations require further investigation.

Since the proposed data management approach is based on comparing PII and PSI, it relies on the availability of detailed planning information. This information must be available digitally using open formats like IFC wherever possible. Moreover, the quality of this information strongly influences the quality of the knowledge gained through the DT. This concerns not only the quality but also the level of detail of the PII. With an increased level of detail, e.g., of the schedule and the building design, more meaningful insights can be created by the DT. The same holds true for the raw data captured on-site. Its quality and level of detail must be sufficient for the intended use cases, as it significantly impacts the final results.

Additionally, even during the construction execution, last-minute changes to the building design and the schedule are common [18]. Some of these changes are not documented in the initial planning documents and are, therefore, difficult to identify. Changes applied to, e.g., the IFC model, need to be reflected appropriately in the graph of the information layer. However, it is not trivial to identify changes between consecutive IFC files. Furthermore, the range of possible design changes is extensive, making an automated updating of the information graph while maintaining semantically correct node connections a complex task. Existing approaches have investigated supporting design collaboration through graph-based transformations and version control of IFC files [70]. More research is required to elaborate on how this concept can be transferred to a graph-based DT.

Another aspect that this research has not explored is the scalability of the proposed solution. Beyond the scope of a single project, the potential benefits lie in enabling comparability across multiple construction projects, fostering iterative learning from one project to the next. Although the proposed ontology framework establishes a common language for describing various construction sites, aligning information from multiple projects with differing levels of granularity remains a significant challenge. Moreover, distinguishing between project-specific, company-specific, and generic influencing factors will be critical for facilitating cross-project learning.

In the described case study, predominantly real-world data was used to demonstrate the platform's ability to handle such data effectively. However, the study was conducted after the completion of the selected construction project. Future research should focus on applying the proposed concept during the actual execution phase of ongoing construction projects. This would enable construction managers to utilize the proposed platform to support their decision-making and provide direct feedback on its practicality and effectiveness. Such studies would be indispensable to assess the added value that the approach can deliver. However, quantitatively measuring the improvements achieved remains a significant challenge. The inherent uniqueness of each construction project makes it difficult to establish direct comparisons, as it is not possible to manage a single project using two different sets of decisions to compare their impact on the final outcomes.

From a technical perspective, relying on web APIs for communication between platform components and integrated digital twin services introduces limitations in terms of real-time communication capabilities. Due to latency and other performance issues, web APIs are not suitable for high-frequency data exchange. For such use cases, communication technologies that follow a push pattern, such as WebSockets, are more appropriate than those that rely on polling for information [71]. However, since most digital twin use cases do not require real-time data exchange, except for scenarios such as tracking construction workers or equipment, providing live alerts for dangerous situations, or real-time control of robots on construction sites, web APIs remain the preferred choice [54]. Nevertheless, it is worth exploring how WebSocket-based communication could be integrated into the reference architecture to support services that do require real-time interaction.

Another limitation related to the reference architecture and the choice of technologies lies in the separation between the central platform and the plugin-like digital twin services. While this separation is crucial for flexible extension and project-specific customization, it also results in important information being excluded from the central platform. The digital twin services function as black boxes: they request data from the platform, perform analyses, and send back insights, but the underlying technical details, such as the methods applied, accuracy, and potential sources of error, are hidden from the platform. These details are crucial for end users to accurately interpret the results. More research is needed to develop concepts that enrich the data with metadata, including aspects such as accuracy, provenance, and related factors. Automating the generation of certain metadata based on the user account executing the API request could be part of the solution.

Finally, this paper has proposed a centralized approach for implementing the DBTP. With certain modifications, however, the proposed architecture could also be adapted for decentralized implementation. Although decentralization introduces challenges related to implementation complexity, data synchronization, and latency, it more closely aligns with the collaborative nature of construction projects, where numerous stakeholders are involved and companies may be reluctant to store data in external databases. For example, the solid server concept [72] uses so-called pods to store personal or company-specific information in a controlled environment. Every pod has dedicated storage units for RDF data and non-RDF data. This approach has similarities with the overarching RDF graph and the connected non-RDF databases proposed. While every project partner is in complete control of their data and its respective access rights, combining the content of the pods describes the construction project holistically. The pods can communicate and share their data through a common web API. In this way, the data, information, and knowledge layers can be implemented in a distributed fashion.

8. Conclusion

This paper has presented a comprehensive platform architecture designed specifically for DTC, with a primary focus on efficient data management throughout the construction phase of building construction projects. The architecture is structured around the fundamental layers of data, information, and knowledge, allowing seamless storage and retrieval of diverse types of construction-related content. This encompasses a data catalog for discoverability, enhanced comparison between project intent and status, and storage of KPIs which resulted from the analysis of data and information. The underlying technology are RDF graphs used as an overarching storage system in combination with specialized databases to ensure versatility and efficiency in data handling.

Additionally, an ontology framework was proposed as the platform's internal data schema, comprising both existing ontologies and the DTC Ontology tailored to concepts specific to DTs of the construction execution. The DTC ontology is a process-oriented ontology that fosters the comparison of project intent and status by explicit representation of both aspects. The ontologies used for the three data management layers can be complemented or replaced with other ontologies to account for project-specific needs. Validation functionalities are built into the platform, ensuring adherence to the internal data model, promoting data integrity and reliability and emphasizing the importance of clear data structures. These are particularly relevant in complex projects involving multiple stakeholders.

By adopting open standards and semantic web principles, like IFC, RDF, SHACL, and SPARQL, the platform offers flexibility and scalability, allowing for adaptation to project-specific requirements. The implemented case study serves as a practical guide for implementation, demonstrating the efficacy of the proposed architecture in a real-world scenario and providing examples for the usage of the ontology framework. Ultimately, the proposed platform serves as a foundational tool for realizing the holistic vision of DTC, providing a starting point for its practical implementation and further advancement.

While this paper has concentrated on a single construction project conducted after its completion, future work will aim to deploy the proposed DTC platform across a wider range of construction projects during ongoing construction activities. The knowledge gained from several construction projects can be combined through the common data structure to form a knowledge base of past projects. It should be investigated how the insights from this knowledge base can be applied to enhance the planning and execution of future projects, promoting an iterative learning process. Testing the platform on diverse construction projects will also provide a clearer understanding of its performance limits and scalability. Additionally, quantitative experiments should be conducted to gain deeper insights into the performance characteristics of the reference architecture and enable better comparability with other approaches. Since platform performance depends not only on the reference architecture but also heavily on the specific technologies used in implementation, benchmark tests must be carefully designed to assess the reference architecture independently of the chosen technologies. One approach could be to implement the proposed reference architecture using a different set of technologies, or to use the technologies employed in this paper to implement alternative reference architectures. Comparing various implementations on well-defined use cases with a standardized data set could provide a more detailed analysis of performance and help distinguish between technology-dependent and architecture-dependent characteristics.

CRediT authorship contribution statement

Jonas Schlenger: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kacper Pluta:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Data curation. **Alwyn Mathew:** Writing – original draft, Visualization, Validation, Software, Data curation. **Timson Yeung:** Writing – review & editing, Validation, Methodology, Conceptualization. **Rafael Sacks:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization. **André Borrmann:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jonas Schlenger reports financial support and travel were provided by European Commission. Kacper Pluta reports financial support and travel were provided by European Commission. Timson Yeung reports financial support and travel were provided by European Commission. Rafael Sacks reports financial support and travel were provided by European Commission. Andre Borrmann reports financial support and travel were provided by European Commission. Alwyn Mathew reports financial support and travel were provided by European Commission. Jonas Schlenger reports financial support was provided by German Research Foundation. Andre Borrmann reports financial support was provided by German Research Foundation. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The research described in this paper has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 958398, "BIM2TWIN: Optimal Construction Management & Production Control". Furthermore, the authors gratefully acknowledge the financial support of the Deutsche Forschungsgemeinschaft (DFG) in the frame of the programs SPP 2187 "Adaptive modularized constructions made in a flux" (Project Number 423969184) and Transregio 277 "Additive Manufacturing in Construction—The Challenge of Large Scale" (Project Number 414265976).

Data availability

The produced data and code are shared on GitHub. The link to the corresponding repository is provided in-text.

References

[1] V.V. Tuhaise, J.H.M. Tah, F.H. Abanda, Technologies for digital twin applications in construction, Autom. Constr. 152 (2023) http://dx.doi.org/10.1016/j.autcon.2023.104931.

[2] M. Grieves, J. Vickers, Digital Twin: Mitigation Unpredictable, Undesirable Emergent Behavior in Complex Systems, Springer International Publishing AG Switzerland, 2017, pp. 85–113, http://dx.doi.org/10.1007/978-3-319-38756-7.

[3] R. Sacks, I. Brilakis, E. Pikas, H.S. Xie, M. Girolami, Construction with digital twin information systems, Data- Centric Eng. E 14 (2020) 1–26, http://dx.doi.org/10.1017/dce.2020.16.

[4] L. Chamari, E. Petrova, P. Pauwels, A web-based approach to BMS, BIM and IoT integration: a case study, in: REHVA 14th HVAC World Congress, 2022, pp. 1–8, http://dx.doi.org/10.34641/clima.2022.228.

[5] Q. Lu, A.K. Parlikad, P. Woodall, G.D. Ranasinghe, X. Xie, Z. Liang, E. Konstantinou, J. Heaton, J. Schooling, Developing a digital twin at building and city levels: Case study of west cambridge campus, J. Manag. Eng. 36 (2020) 05020004, http://dx.doi.org/10.1061/(asce)me.1943-5479.0000763.

[6] C. Ramonell, R. Chacón, H. Posada, Knowledge graph-based data integration system for digital twins of built assets, Autom. Constr. 156 (2023) http://dx.doi.org/10.1016/j.autcon.2023.105109.

[7] C. Boje, A. Guerriero, S. Kubicki, Y. Rezgui, Towards a semantic construction digital twin: directions for future research, Autom. Constr. 114 (2020) http://dx.doi.org/10.1016/j.autcon.2020.103179.

[8] M.P. Papazoglou, W.-J. Van Den Heuvel, Service oriented architectures: approaches, technologies and research issues, VLDB J. 16 (2007) 389–415, http://dx.doi.org/10.1007/s00778-007-0044-3.

[9] S. Li, H. Zhang, Z. Jia, C. Zhong, C. Zhang, Z. Shan, J. Shen, M.A. Babar, Understanding and addressing quality attributes of microservices architecture: A Systematic literature review, Inf. Softw. Technol. 131 (2021) http://dx.doi.org/10.1016/j.infsof.2020.106449.

[10] H. Boyes, T. Watson, Digital twins: An analysis framework and open issues, Comput. Ind. 143 (2022) http://dx.doi.org/10.1016/j.compind.2022.103763.

[11] F. Mostafa, L. Tao, W. Yu, An effective architecture of digital twin system to support human decision making and AI-driven autonomy, Concurr. Comput.: Pr. Exp. 33 (2021) 1–15, http://dx.doi.org/10.1002/cpe.6111.

[12] J. Lee, B. Bagheri, H.A. Kao, A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manuf. Lett. 3 (2015) 18–23, http://dx.doi.org/10.1016/j.mfglet.2014.12.001.

[13] buildingSMART International, IFC schema specifications, 2021, https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/.

[14] Y. Zheng, S. Törmä, O. Seppänen, A shared ontology suite for digital construction workflow, Autom. Constr. 132 (2021) 103930, http://dx.doi.org/10.1016/j.autcon.2021.103930.

[15] BIM2TWIN, BIM2TWIN Website, 2023, https://bim2twin.eu.

[16] A.A. Akanmu, C.J. Anumba, O.O. Ogunseiju, Towards next generation cyber–physical systems and digital twins for construction, J. Inf. Technol. Constr. 26 (2021) 505–525, http://dx.doi.org/10.36680/j.itcon.2021.027.

[17] W. Kritzinger, M. Karner, G. Traar, J. Henjes, W. Sihn, Digital twin in manufacturing: A categorical literature review and classification, in: IFAC-PapersOnLine, vol. 51, Elsevier B.V., 2018, pp. 1016–1022, http://dx.doi.org/10.1016/j.ifacol.2018.08.474.

[18] S. Lee, F. Peña-Mora, M. Park, Quality and change management model for large scale concurrent design and construction projects, J. Constr. Eng. Manag. 131 (2005) 890–902, http://dx.doi.org/10.1061/(ASCE)0733-9364(2005)131:8(890).

[19] A.J. Redelinghuys, A.H. Basson, K. Kruger, A six-layer architecture for the digital twin: a manufacturing case study implementation, J. Intell. Manuf. 31 (2020) 1383–1402, http://dx.doi.org/10.1007/s10845-019-01516-6.

[20] A.R. Al-Ali, R. Gupta, T.Z. Batool, T. Landolsi, F. Aloul, A.A. Nabulsi, Digital twin conceptual model within the context of internet of things, Futur. Internet 12 (2020) 1–15, http://dx.doi.org/10.3390/fi12100163.

[21] F. Tao, M. Zhang, A. Nee, Five-Dimension Digital Twin Modeling and Its Key Technologies, Elsevier, 2019, pp. 63–81, http://dx.doi.org/10.1016/b978-0-12-817630-6.00003-5.

[22] A. Borrmann, J. Schlenger, N. Bus, R. Sacks, AEC digital twin data - why structure matters, in: 19th International Conference in Civil & Building Engineering, 2022, pp. 651–669, http://dx.doi.org/10.1007/978-3-031-32515-1.

[23] P. Pauwels, S. Zhang, Y.C. Lee, Semantic web technologies in AEC industry: A literature overview, Autom. Constr. 73 (2017) 145–165, http://dx.doi.org/10.1016/j.autcon.2016.10.003.

[24] L. van Berlo, T. Krijnen, H. Tauscher, A. van Kranenburg, P. Paasiala, Future of the industry foundation classes: towards IFC 5, in: 38th International Conference of CIB W78, 2021, pp. 123–137, https://itc.scix.net/paper/w78-2021-paper-013.

[25] W. Terkaj, P. Pauwels, A method to generate a modular ifcOWL ontology, in: CEUR Workshop Proceedings, Vol. 2050, 2017, pp. 1–12, https://api.semanticscholar.org/CorpusID:38847370.

[26] P. Pauwels, Building element ontology, 2018, https://pi.pauwel.be/voc/buildingelement/index-en.html.

[27] M.H. Rasmussen, M. Lefrançois, G.F. Schneider, P. Pauwels, BOT: The building topology ontology of the W3C linked building data group, Semant. Web 12 (2020) 143–161, http://dx.doi.org/10.3233/SW-200385.

[28] M.H. Rasmussen, P. Pauwels, M. Lefrançois, G.F. Schneider, C.A. Hviid, J. Karlshøj, Recent changes in the building topology ontology, in: 5th Linked Data in Architecture and Construction Workshop, 2017, pp. 1–7, http://dx.doi.org/10.13140/RG.2.2.32365.28647.

[29] A. Haller, K. Janowicz, S. Cox, M. Lefrançois, K. Taylor, D.L. Phuoc, J. Lieberman, R. García-Castro, R. Atkinson, C. Stadler, The SOSA/SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, actuation, and sampling, Semant. Web 10 (2019) 9–32, https://api.semanticscholar.org/CorpusID:204767879.

[30] Y. Zhang, J. Beetz, Semantic sensor data federation in dynamic knowledge graphs using RDF-star, in: International Workshop on Computing in Civil Engineering 2023, 2023, pp. 1–10, https://www.ucl.ac.uk/bartlett/construction/sites/bartlett_construction/files/semantic_sensor_data_federation_in_dynamic_knowledge_graphs_using.pdf.

[31] R. Ackoff, From data to wisdom - Presidential address to ISGSR, 1988, J. Appl. Syst. Anal. 16 (1989) 3–9, https://www.scirp.org/(S(351jmbntvnsjt1aadkposzje))/reference/ReferencesPapers.aspx?ReferenceID=713373.

[32] T. Pitkäranta, O. Alhava, K. Koivu, D1.3 - KPIs for evaluating the construction process behaviour, 2021, https://bim2twin.eu/wp-content/uploads/2022/10/Attachment_D1.3.pdf.

[33] E. Büttner, S. Richter, Microkernel an Architecture Pattern, vol. 2, Universitätsverlag Potsdam, 2005, pp. 18–27, http://dx.doi.org/10.1109/RELDIS.2001.970772.

[34] G. Block, P. Cibraro, P. Félix, H. Dierking, D. Miller, Designing Evolvable Web APIs with ASP.NET, O'Reilly Media, ISBN: 978-14-493-3771-1, 2014.

[35] R.T. Fielding, Architectural styles and the design of network-based software architectures, (Ph.D. thesis), University of California, Irvine, 2000, https://ics.uci.edu/fielding/pubs/dissertation/fielding_dissertation.pdf.

[36] A. Nayak, A. Poriya, D. Poojary, Type of NOSQL databases and its comparison with relational databases, Int. J. Appl. Inf. Syst. (IJAIS) 5 (2013) 16–19.

[37] I. Robinson, J. Webber, E. Eifrem, Graph Databases - New Opportunities for Connected Data, second ed., O'Reilly Media, 2015, http://dx.doi.org/10.1016/b978-0-12-407192-6.00003-0.

[38] K. Grolinger, W.A. Higashino, A. Tiwari, M.A.M. Capretz, Data management in cloud environments: NoSQL and NewSQL data stores, J. Cloud Comput.: Adv. Syst. Appl. 2 (2013) 1–24, http://dx.doi.org/10.1186/2192-113X-2-22.

[39] R. Angles, H. Thakkar, D. Tomaszuk, RDF and property graphs interoperability: Status and issues, in: CEUR Workshop Proceedings, vol. 2369, 2019, pp. 1–11, https://api.semanticscholar.org/CorpusID:174798876.

[40] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (2001) 34–43, https://www.lassila.org/publications/2001/SciAm.pdf.

[41] S. Harris, A. Seaborne, SPARQL 1.1 query language, 2013, https://www.w3.org/TR/sparql11-query/.

[42] H. Knublauch, D. Kontokostas, Shapes constraint language (SHACL), 2017, https://www.w3.org/TR/shacl/.

[43] M. Knezevic, A. Donaubauer, M. Moshrefzadeh, T.H. Kolbe, Managing urban digital twins with an extended catalog service, in: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 10, Copernicus Publications, 2022, pp. 119–126, http://dx.doi.org/10.5194/isprs-annals-X-4-W3-2022-119-2022.

[44] H. Dibowski, S. Schmid, Y. Svetashova, C. Henson, T. Tran, Using semantic technologies to manage a data lake: Data catalog, provenance and access control, in: 13th International Workshop on Scalable Semantic Web Knowledge Base Systems, 2020, pp. 65–80, http://www.ssws-ws.org/SSWS2020/SSWS2020_paper5.pdf.

[45] P. Pauwels, T. Krijnen, W. Terkaj, J. Beetz, Enhancing the ifcowl ontology with an alternative representation for geometric data, Autom. Constr. 80 (2017) 77–94, http://dx.doi.org/10.1016/j.autcon.2017.03.001.

[46] W. Penberthy, S. Roberts, Pro.NET on Amazon Web Services - Guidance and Best Practices for Building and Deployment, APress Media, 2023, http://dx.doi.org/10.1007/978-1-4842-8907-5_1.

[47] T. Davies, glTF and construction – part 1: Secrets of the cloud, 2018, https://constructingdata.wordpress.com/2018/09/07/gltf-and-construction-part-1-secrets-of-the-cloud/.

[48] J.D. Murray, W. vanRyper, Encyclopedia of Graphics File Formats, second ed., O'Reilly Media, ISBN: 978-1-565-92161-0, 1996.

[49] F. Collins, F. Pfitzner, J. Schlenger, Scalable construction monitoring for an as-performed progress documentation across time, in: 33. Forum Bauinformatik, 2022, pp. 70–79, http://dx.doi.org/10.14459/2022md1686600.

[50] J. Torres, R.S. Mateos, N. Lasarte, H.G. Pinto, F. Noiray, O. Alhava, M. Tual, S. Velasquez, D1.1 - as-is practices analysis and end-user requirements, 2022, https://bim2twin.eu/wp-content/uploads/2023/01/D1.1-BIM2TWIN.pdf.

[51] N. Bus, B. Fies, BIM2TWIN Dashboard, 2024, https://bim2twin.cstb.fr/.

[52] D. Fett, R. Küsters, G. Schmitz, The web SSO standard openid connect: In-depth formal security analysis and security guidelines, in: 2017 IEEE 30th Computer Security Foundations Symposium, CSF, 2017, pp. 189–202, http://dx.doi.org/10.1109/CSF.2017.20.

[53] L. Chamari, E. Petrova, P. Pauwels, An end-to-end implementation of a service-oriented architecture for data-driven smart buildings, IEEE Access (2023) http://dx.doi.org/10.1109/ACCESS.2023.3325767.

[54] A. Mediavilla, R.S. Mateos, J. Torres, D1.2 - definition of the digital workflows for the construction process, 2022, https://bim2twin.eu/wp-content/uploads/2023/01/D1.2-BIM2TWIN.pdf.
[55] N.F. Noy, M. Crubézy, R.W. Fergerson, H. Knublauch, S.W. Tu, J. Vendetti, M.A. Musen, Protégé-2000: An open-source ontology-development and knowledge-acquisition environment AMIA 2003 open source expo, in: AMIA 2003 Symposium, 2003, p. 953, http://protege.stanford.edu.
[56] J. Schlenger, T. Yeung, S. Vilgertshofer, J. Martinez, R. Sacks, A. Borrmann, A comprehensive data schema for digital twin construction, in: The 29th EG-ICE International Workshop on Intelligent Computing in Engineering, 2022, pp. 34–44, http://dx.doi.org/10.7146/aul.455.c194.
[57] R. Kenley, T. Harfield, Location breakdown structure (LBS): a solution for construction project management data redundancy, in: International Conference on Construction in a Changing World, 2014, pp. 1–11, https://api.semanticscholar.org/CorpusID:107678245.
[58] D. Berrueta, J. Phipps, Best practice recipes for publishing RDF vocabularies, 2008, https://www.w3.org/TR/swbp-vocab-pub/.
[59] A. Wagner, M. Bonduel, P. Pauwels, U. Rüppel, Representing construction-related geometry in a semantic web context: A review of approaches, Autom. Constr. 115 (2020) http://dx.doi.org/10.1016/j.autcon.2020.103130.
[60] G. Turk, The PLY polygon file format, 1994, http://gamma.cs.unc.edu/POWERPLANT/papers/ply.pdf.
[61] R. Battle, D. Kolas, GeoSPARQL: Enabling a geospatial semantic web, Semant. Web J. 3 (2011) 355–370, https://www.semantic-web-journal.net/sites/default/files/swj176_0.pdf.
[62] M. Bonduel, A. Wagner, P. Pauwels, FOG: File ontology for geometry formats, 2020, https://mathib.github.io/fog-ontology/.
[63] R. Albertoni, D. Browning, S. Cox, A. Beltran, A. Perego, P. Winstanley, Data catalog vocabulary (DCAT) - version 3, 2023, https://www.w3.org/TR/vocab-dcat-3/.
[64] L. Daniele, F. den Hartog, J. Roes, ETSI TS 103 410-4 - V1.1.2 - SmartM2M; Extension to SAREF; Part 4: Smart Cities Domain, 2019, https://www.etsi.org/deliver/etsi_ts/103400_103499/10341004/01.01.02_60/ts_10341004v010102p.pdf.
[65] M. Trzeciak, K. Pluta, Y. Fathy, L. Alcalde, S. Chee, I. Brilakis, P. Alliez, ConSLAM: construction dataset for SLAM, J. Comput. Civ. Eng. 37 (2023) http://dx.doi.org/10.1061/JCCEE5.CPENG-5212.
[66] M. Bonduel, J. Oraskari, P. Pauwels, M. Vergauwen, R. Klein, The IFC to linked building data converter - current status, in: 6th Linked Data in Architecture and Construction Workshop, vol. 2159, 2018, pp. 34–43, https://api.semanticscholar.org/CorpusID:53505961.
[67] J. Tulke, Kollaborative terminplanung auf basis von bauwerksinformationsmodellen, Inform. Archit. Und Bauwes. 4 (2010) https://d-nb.info/1116284456/34.
[68] P. Monasse, R. Djahel, B. Vallet, Registration for urban modeling based on linear and planar features, in: 2023 11th European Workshop on Visual Information Processing, EUVIP, IEEE, 2023, pp. 1–8, http://dx.doi.org/10.1109/EUVIP58404.2023.10323059.
[69] J.-P. Bauchet, F. Lafarge, Kinetic shape reconstruction, ACM Trans. Graph. 39 (2020) 1–14, http://dx.doi.org/10.1145/3376918.
[70] S. Esser, S. Vilgertshofer, A. Borrmann, Version control for asynchronous BIM collaboration: Model merging through graph analysis and transformation, Autom. Constr. 155 (2023) http://dx.doi.org/10.1016/j.autcon.2023.105063.
[71] V. Wang, F. Salim, P. Moskovits, The WebSocket Protocol, A Press, Berkeley, CA, 2013, pp. 33–60, http://dx.doi.org/10.1007/978-1-4302-4741-8_3.
[72] A.V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, T. Berners-Lee, Solid: A Platform for Decentralized Social Applications Based on Linked Data, Tech. Rep., MIT CSAIL & Qatar Computing Research Institute, 2016, https://api.semanticscholar.org/CorpusID:49564404.