DigitMan
Occupant-based DIGITal predictive MANteinance to improve the built environment

Grant number: 2022JMRX2A

WP 1 – Conceptual framework and database definition

T.1.2 – Database structuring

DELIVERABLE ID	D1.2
Deliverable Title	Integrated structuring of a common database
Date	26/07/2024
Last revision date	23/07/2024
Revision	003
Main partner	UNIBO
Additional partners	UNIVPM, POLIMI
Authors of the contribution	Marco D'Orazio, Riccardo Gulli, Giorgia Predari, Guido Romano, Graziano Salvalai, Roberto Villa
Deliverable type	Report
Number of pages	13

Abstract

This deliverable presents the integrated approach for structuring the DIGITMAN project's common database. It establishes the foundational data structures supporting the project's objectives. The document first outlines the system architecture and the methodologies chosen for creating and managing the database. Then, it provides detailed reports to document the classification systems adopted for organically mapping maintenance activities, functional types of spaces, and equipment devices within the database. These classification systems align coherently with the different datasets from the partner institutions. Finally, the document reports the property sets and quantities used to assign data to the DIGITMAN database elements. They are based on a shared notation aligned with the partner's datasets and international standards and schemas.

Keywords

Data Integration, Database Management System, Classification System, Semantic Mapping, Property Attribution

Approvals

Role	Name	Partner	Date
Coordinator	Marco D'Orazio	UNIVPM	26/07/2024
Task leader	Riccardo Gulli	UNIBO	26/07/2024

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA
UNIVERSITÀ POLITECNICA DELLE MARCHE
Finanziato dall'Unione europea NextGenerationEU
ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA
Ministero dell'Università e della Ricerca
POLITECNICO MILANO 1863

Revision versions

Revision	Date	Short summary of modifications	Partner
001	02/05/2024	Document creation and drafting	UNIBO
002	01/06/2024	Review	UNIBO
003	23/07/2024	Text formatting and grammar checking	UNIBO

Summary

Introduction

Enabling digital building management is a notoriously complex task, primarily due to data diversity, heterogeneity, and fragmentation across various applications, data sources, and stakeholders in the building management field [1–3]. It follows that the development of digital decision support systems (DDSS) for the management of the built heritage must be done on the basis of highly fragmented and heterogeneous data that originates from a wide diversity of actors and source systems with different intended uses. Moreover, the acquired data may not necessarily align with the requirements, standards, or protocols of DDSS applications. On the one hand, DDSSs should be able to support various functionalities, each with unique information requirements that must be accommodated. On the other hand, DDSSs should ensure adequate data standardization in a unified, flexible, and accessible manner.

Various researchers have proposed 'multi-tier' architectures to support heterogeneous application environments with the aim of making data accessible, aggregated, usable, and exploitable, streamlining the development processes for DDSSs. These research efforts included releasing cyber-physical systems (CPSs), IoT, smart building, and DT platforms [4, 5].

Multi-tier architecture, or 'onion' architecture, is a framework for organizing software components into distinct layers, each representing a different level of functionality [6]. These layers are logical divisions, meaning they categorize components based on their roles and responsibilities, regardless of where they are physically located in the system. Each layer is self-contained, focusing on a specific aspect of the software's functionality. Components within a single layer work together to perform tasks related to that layer's function but remain independent of components in other layers. Communication in this architecture is structured and controlled. A layer can only interact directly with its immediate neighbor, the one just above (upstream) or just below it (downstream) in the architecture. This ensures that each layer has a clear and defined role, reducing complexity and enhancing the system's maintainability. In light of this, multi-tier architecture design has several advantages. Primarily, it facilitates the independent implementation of each layer as a cohesive unit, eliminating the need for extensive knowledge about the other layers. This independence is a significant benefit, as it allows for flexibility and ease in the development and maintenance of each layer. Secondly, this design promotes modularity in each layer through a loosely coupled approach. This modularity enables the replacement or alteration of layers with alternative implementations that provide the same fundamental services. Such a feature is advantageous for adapting to changing requirements or technologies without overhauling the entire system. Furthermore, the software components within a given layer are not only pivotal for the functionality of that specific layer but can also contribute to constructing and providing services for other layers [4].

This deliverable describes the proposed system architecture for developing the DIGITMAN application. The system architecture model guides the project's development from a technical and operational perspective, adhering to the multi-tier architecture structure and capturing its advantages. Additionally, this description methodically outlines the process of structuring the shared database. Alongside the model description, the text includes DIGITMAN's reference vocabulary, which encompasses a classification system and mapping attributes, properties, and quantities attributable to the project's information.

1 System architecture

This section presents the system architecture model adopted by DIGITMAN. This model, depicted in Figure 1, serves as a guide for defining the project's DDSS from a technical standpoint. It is composed of six layers, described in the following paragraphs, which are:

- the 'data acquisition' layer,
- the 'data transmission' layer,
- the 'data storage' layer,
- the 'data extraction and processing' layer,
- the 'data integration' layer,
- the 'data visualization' layer.

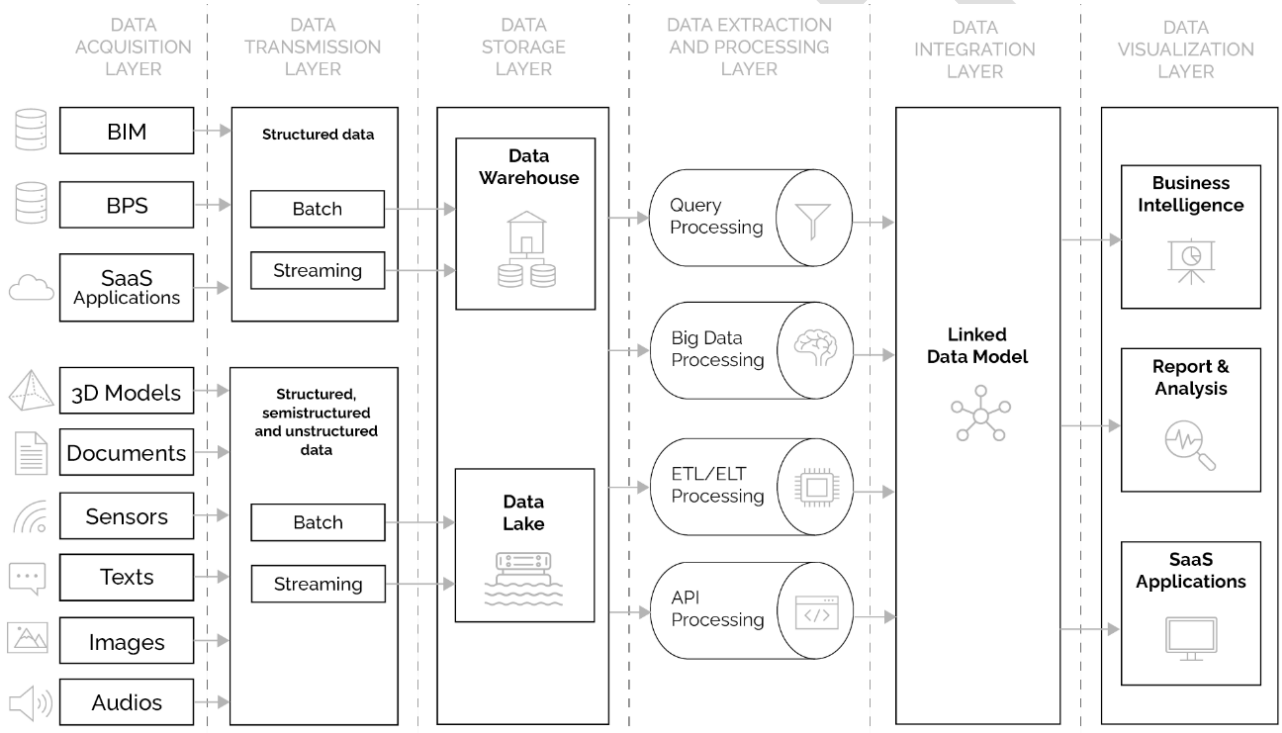

Figure 1. System architecture model for built asset management applications.

1.1 Data acquisition layer

The data acquisition layer is the initial layer in the multi-layered DDSS architecture. It allows the collection of data related to the built asset. It determines the quantity, quality, and type of data to be collected and ingested into the DDSS, as well as the methods for capturing this data. In other words, this layer catches all data types inherent in the lifecycle of a building - or building portfolio - to be acquired from the many data sources available to achieve the project's purposes.

The data collected in the data acquisition layer is divided into structured and unstructured:

- **Structured data** come primarily from models, whether these are to be created from scratch or, more rarely, already available to the administration. These include Building Information Models (BIMs) and Building Performance Simulation (BPS) models, which are essential for hierarchically structuring the DDSS information about the physical and spatial elements that make up buildings. Structured data also lists all those data that, directly or indirectly, come from SaaS

(Software-as-a-Service) applications or databases already used by the administration and can contribute to enriching the DDSS information system.

- **Unstructured data**, on the other hand, concerns all those data that are not ready to be directly readable by a computer in a systematic way and that, therefore, need processing and aggregation to be transformed into valuable information. This data has no value without being aggregated and merged with other data. Unstructured data comes from static data sources, such as non-semantic three-dimensional models (e.g., point clouds) and documents (e.g., technical building reports, contracts, but also floor plans and drawings, in formats such as PDF or DWG), or dynamic data sources, such as sensors, IoT devices, meters, but also text logs (e.g., user feedback logs), image and video devices, and other similar raw data sources.

In DIGITMAN, the primary data considered in the data acquisition layer are:

- BIM models of UNIVPM and POLIMI's buildings. These are developed thanks to the TBIM approach provided in Deliverable 1.1;
- BPS models of UNIVPM and POLIMI's buildings. These models will be developed in WP3 and WP4 for energy and safety-related calculations;
- Data from the SaaS applications of UNIVPM and POLIMI. These include data about:
    - The composition of the real estate portfolios of the two universities, with the identification of buildings and spaces and the association of basic properties such as the location of the buildings, their function, and the floor plan dimensions of the spaces.
    - The maintenance activities carried out in the two portfolios, including both raw data (text messages/emails sent by management operators to report maintenance issues) and structured data (logs of maintenance activities, emergencies, and timing, extractable in tabular format);
- Raw unstructured data from the sensors installed in POLIMI's pilot building.

1.2 Data transmission layer

The data transmission layer is the second layer in the DDSS architecture. It enables data transfer and communication, typically involving data exchange between different systems or actors. This layer handles the diverse array of structured, semi-structured, and unstructured data gathered in the data acquisition layer – each with its own speed, frequency, and transmission volume requirements – and transmits it to the higher layers of the DDSS.

Transiting models and documents from one system to another, whether via human or machine operations, is a critical process. To maintain the highest level of reliability, the digital models must not only meet the predefined information requirements of the DDSS but also be verifiable and traceable. Moreover, data transmission methods vary greatly depending on the type of data.

- Models, unlike other types of data sources, are transmitted to the system only occasionally - either at the time of their creation or when modifications are made. For example, a BIM model would be updated in the system following any physical changes to the asset it represents. Integrating data versioning and historicization tools is essential in such cases to ensure that the data is traceable back to its originator and that its modification history is preserved and accessible.
- Documents requiring entry into the DDSS undergo a similar process. They also necessitate robust validation, tracking, and versioning systems to ensure authenticity and relevance before being integrated into the DDSS. This approach guarantees that all information, whether in the form of models or documents, is accurate, up-to-date, and securely managed within the system. In contrast, SaaS applications may send data to the DDSS more frequently, with the frequency depending on the specific nature and function of the SaaS. For instance, a SaaS system dedicated to planning and scheduling building occupancy would likely update the DDSS whenever there is a change in its data. This type of data, often referred to as 'batch' data, can initially be transmitted using an export/import approach. However, to streamline and automate the management

process, more sophisticated methods of communication are advisable -for example, the development of connectors between the SaaS and the DDSS, facilitated by specific APIs.

- Finally, a separate discussion should be made for the more dynamic and potentially real-time data sources. This aspect focuses on more technical issues that relate primarily to IoT communication technologies. This kind of connectivity comes in many wireless forms, such as Bluetooth, Zigbee, WiFi, LoRaWAN, or Cellular 5G [4, 7]. Since various factors jointly affect data transmission performance, a trade-off is necessary between these compound factors to achieve optimal choice. In addition, it is crucial to consider the reliability of different systems as well as their security, both from an IT perspective and for user privacy.

The following data transmission methods are used to structure the shared database of DIGITMAN, as outlined in the deliverables related to WP1 (D1.1, D1.2, D1.3). During the activities of WP5, further investigations will be carried out to understand if and how it is possible to automate further and standardize these methods to facilitate a more organic utilization of the data.

- BIM and BPS models are stored locally and transmitted to the database using some connectors developed in Python. These connectors allow data integration from these models into graph structures (as illustrated later and anticipated in D1.1).
- The data related to the portfolio composition have been extracted once from the Facility Management applications used by UNIVPM and POLIMI through manual operations in tabular format (csv/Excel). These are then processed by other specially developed Python connectors, which map the data description in the tables according to the alignment and classification system proposed by DIGITMAN.
- Maintenance reporting data are periodically extracted from the maintenance management applications used by UNIVPM and POLIMI management entities through manual operations in tabular format (csv/Excel). These are then processed by other specially developed Python connectors, which map the data description in the tables according to the alignment and classification system proposed by DIGITMAN.
- Sensor data are periodically extracted in tabular form (CSV/TXT) from various online platforms provided by service providers, as detailed in D1.1. These data are processed using specially developed Python scripts to clean and normalize them, adhering to the classification method proposed by DIGITMAN for Sensor Quantities, as further explained in Section 4.

1.3 Data storage layer

The data storage layer is the third layer in the DDSS architecture. It allows for data to be stored in digital repositories structured to allow easy data extraction.

Depending on whether the data is structured or unstructured, this repository can have two different forms: data warehouse (DW) or data lake (DL). A DW is a data repository that holds structured, filtered, and processed data for a specific purpose, whereas a DL is a large pool of data with no clear purpose. Data warehouses, in detail, hold massive volumes of data acquired from various sources, often using preset schemas.

- A DW is typically a purpose-built relational database -e.g., a MySQL database- that runs on specialized hardware or in the cloud. DWs have long been used to store enterprise data and power business intelligence and analytics applications [8]. In a certain sense, a BIM model can also be intended as a DW, being it, fundamentally, a relational database.
- DLs, instead, have arisen as large data repositories that hold raw data and offer a plethora of functionality via metadata descriptions. Although the DL is a type of enterprise data storage, it lacks the analytical features often associated with data warehouses. They are repositories that store raw data in their original formats and provide a standard access interface. In some instances, data may travel downstream from DL to DW to be processed, packaged, and ready for consumption [8].

Although some features and use cases overlap between data warehouses and data lakes, there are fundamental distinctions in the data management philosophies, design characteristics, and optimal use conditions for each technology. However, a new paradigm combining DW and DL's benefits has emerged: the data lakehouse [9]. It is a new open data management architecture that combines the flexibility, cost-effectiveness, and scalability of DL with DW's data management and transaction capabilities to perform business intelligence and machine learning activities on all data. Such a component represents a crucial element within the system, as it acts as a comprehensive centralized data repository that is highly flexible. It can collect, integrate, and store data from various sources, including the following: spatial, geometric, topological, and technical information; real-time data from IoT devices; technical and regulatory documentation; data from external service platforms; aggregated data and KPIs; and data from records and change modifications.

Data storage in DIGITMAN is conducted in different types of databases (graph databases, document databases, and relational databases) according to different purposes (Figure 2). These types of databases reflect three distinct approaches to organizing and presenting data.

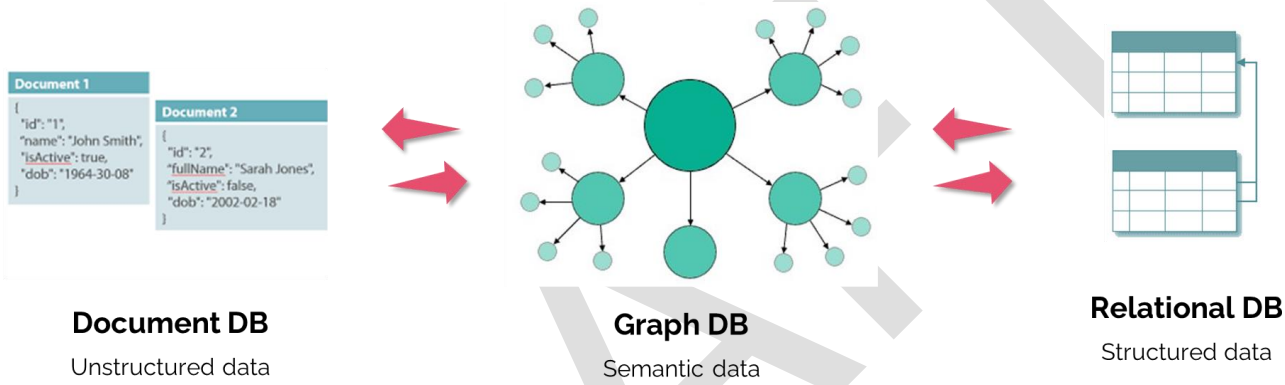

Figure 2. Data storage approach proposed by DIGITMAN.

Graph DBs group data into nodes and edges. Nodes represent entities; edges indicate relationships between entities. The properties and labels of the nodes and edges, which specify the attributes of the entities and connections, define the relationships between them. Relational DBs are data models that organize data into related tables. Each table consists of a set of rows that each represent an instance of the data and a set of columns that each indicate a different attribute or characteristic of the data. Primary and foreign keys, which link relevant data across tables, define the relationships between the tables. Document DBs (also known as document-oriented databases) are databases that store information in documents. Document databases are considered to be non-relational (or NoSQL) databases. Document databases use flexible documents instead of storing data in fixed rows and columns.

Graph DB's structural variance influences the way data is structured and retrieved with respect to more traditional data structures [10]. For instance, within relational models data is organized into tables, and queries commonly involve employing SQL to select and combine data from diverse tables. In contrast, graph models arrange data into nodes and edges, and queries are typically executed using a query language designed explicitly for the graph database. An additional contrast between the two models pertains to their capabilities. Relational models are particularly adept at handling structured data that features clearly defined relationships among entities. Conversely, graph models are adept at accommodating unstructured and semi-structured data characterized by intricate and dynamic relationships among entities. Graph models also possess the ability to depict intricate relationships and interdependencies, an aspect that poses challenges within relational models. Furthermore, graph structures are usually well-suited for web-based applications and can be easily aligned with ontologies where elements and connections are mapped with nodes and edges.

DIGITMAN combines these technologies by using:

- Graph DBs to map the static elements that compose the buildings (i.e., spatial elements, interfaces, and equipment) and their semantic relationships.
- Relational DBs to store data about maintenance requests. The use of the relational model is because the available maintenance data are already stored in relational database management systems in the university services.
- Document DBs to store dynamic sensor data smoothly and straightforwardly.

Technically, for the purpose of WP1's data acquisition activities:

- NetworkX [11] (a Python library for graph analysis) and JSON-LD (JavaScript Object Notation for Linked Data) are used for managing graph databases and storing semantic data;
- JSON (JavaScript Object Notation) is used to manage the documents that will be uploaded to document-based DBs for storing sensor observations;
- SQLite3 will be temporarily used to store the tabular data of maintenance requests. Specifically, these data are stored in the ".db" format used for relational databases.
- BIM and BPS data are stored in the formats commonly used in the field, such as Industry Foundation Classes (IFC) [12], Autodesk Revit, and Topologic JSON [13] for BIM, and EnergyPlus Input Data Format (IDF) for building energy models.

In WP5, these storage systems will be scaled up on the web, and cloud service providers will furnish more solid solutions. Contacts with suppliers have already been initiated and will be finalized in the coming months.

1.4 Data extraction and processing layer

The data extraction and processing layer is the fourth layer in the DDSS architecture. This layer extracts data from the data storage layer and processes it for use in the data integration layer. It is an intermediate layer used to extract data from the digital repositories, use it for performing KPI-based calculations, and organize it according to the data structure defined in Deliverable D1.1.

On the one hand, this level includes extraction tools such as query and ETL (Extraction, Transformation, Load) algorithms. Here, using APIs allows for communication between different data archiving systems. For example, queries based on APIs are used for data extraction from BIM or BPS models or from documents containing sensor observations. On the other hand, this level establishes a data encoding and formatting system. This is based on three cornerstones: a) the system ontology (presented in D1.1), b) the system data format (presented in D1.1), and c) an information vocabulary common to the DDSS (presented in this deliverable). Clarifying these aspects permits transforming data into valuable information with semantic meaning, allowing efficient and interoperable data exchange and integration. Last, big data algorithms are included in this layer to process the substantial volumes of data extracted from data lakes. This processing is aimed at condensing data into more concise datasets, thereby reducing the overall data volume and facilitating efficient data integration in subsequent levels.

The data extraction and processing layer will be performed in WP2, WP3, and WP4.

1.5 Data integration layer

The fifth layer of the DDSS system architecture model is the data integration layer. Its goal is to integrate all data resources and digital models in a unified data environment. It is the location where data are effectively managed and linked before being displayed on the user interfaces.

All the information passing from this layer is treated hierarchically and semantically (according to the principles already described in D1.1) to allow efficient data storage, integration, query, and analysis, all requirements that guarantee good performance of the DDSS. For this reason, in this layer, data and

information are framed within flexible and modular data structures, such as knowledge graphs and liked data structures.

Preliminary tests have been conducted to implement the data integration layer. However, more in-depth explorations will be deferred to WP5 in the future.

1.6 Data visualization layer

The data visualization layer, also named "service layer", is the top layer of the DDSS architecture. It interprets knowledge from the bottom layers and allows users to engage with the DDSS through user-friendly services. Its implementation coincides with the front-end development of applications and microservices. Such microservices extract data and information from the integration layer to make it usable for users to use specific information. They can encompass a variety of interface types, including reporting and analysis systems, business intelligence dashboards, and SaaS applications.

The functionalization of data through services is based on "service encapsulation". The advantage of encapsulation into services is that such services can be designed differently, even though they refer to the same data, based on the capabilities and needs of their end users. In this sense, the data visualization layer makes it possible to orient the DDSS toward its usage needs (e.g., maintenance, safety, and operational needs).

Data visualization will be performed in WP5.

2 Classification system

The method for delivering DIGITMAN's classification system has been illustrated in Deliverable 1.1. Table 1 provides references to the classification system tables adopted in the project.

Table 1: Classification system adopted by DIGITMAN.

Deliverable	Content	Type	Link
1.2.2.1 Maintenance Activities	*Classification table of maintenance activity types aligning UNIVPM-POLIMI dataset with OmniClass.*	*Table*	*DIGITMAN_ClassificationSystem_ActivityTypes.xlsx*
1.2.2.2 Occupancy Types	*Classification table of space functions aligning UNIVPM-POLIMI dataset with OmniClass.*	*Table*	*DIGITMAN_ClassificationSystem_OccupancyTypes.xlsx*

3 Property sets and properties

In addition to the classification and ontological systems, all properties attributable to objects within DIGITMAN's database have been mapped. The mapped properties are those strictly necessary for conducting the What-If/How-To investigations. Table 2 provides references to property mapping documents.

Table 2: Property Sets within the DIGITMAN project.

Deliverable	Content	Type	Link
1.2.3.1 Activity PSets and Properties	*Summary table of properties and property sets to describe maintenance activities.*	*Table*	*DIGITMAN_PropertySets_Activity.xlsx*
1.2.3.2 Topology PSets and Properties	*Summary table of properties and property sets to describe spaces.*	*Table*	*DIGITMAN_PropertySets_Topology.xlsx*
1.2.3.3 Interface PSets and Properties	*Summary table of properties and property sets to describe interface elements.*	*Table*	*DIGITMAN_PropertySets_Interface.xlsx*
1.2.3.4 Equipment PSets and Properties	*Summary table of properties and property sets to describe equipment elements.*	*Table*	*DIGITMAN_PropertySets_Equipment.xlsx*

4 Quantities

Moreover, the quantities (or observable properties) associated with dynamic simulation and sensor data acquisition have been mapped. References are reported in Table 3.

Table 3. Observable or quantities for dynamic data.

Deliverable	Content	Type	Link
1.2.4.1 Sensor quantities	*Summary table of quantities measured by sensors.*	*Table*	*DIGITMAN_Quantities_SensorQty.xlsx*
1.2.4.2 Simulation quantities	*Summary table of quantities measured by simulations.*	*Table*	*DIGITMAN_Quantities_SimulationQty.xlsx*

5 References

1. Merino, J., Xie, X., Moretti, N., Chang, J.Y., Parlikad, A.K.: Data integration for digital twins in the built environment. Presented at the 2022 European Conference on Computing in Construction July 24 (2022). https://doi.org/10.35490/EC3.2022.172.

2. Xie, X., Moretti, N., Merino, J., Chang, J., Pauwles, P., Parlikad, A.K.: Enabling building digital twin: ontology-based information management framework for multi-source data integration. Presented at the CIB World Building Conference 2022 , Melbourne (2022).

3. Chamari, L., Petrova, E., Pauwels, P.: A web-based approach to BMS, BIM and IoT integration. CLIMA 2022 conference. 2022: CLIMA 2022 The 14th REHVA HVAC World Congress (2022). https://doi.org/10.34641/CLIMA.2022.228.

4. Lu, Q.: Digital twins in the built environment: fundamentals, principles and applications. ICE Publishing, London (2022).

5. Zhao, J., Feng, H., Chen, Q., Garcia de Soto, B.: Developing a conceptual framework for the application of digital twin technologies to revamp building operation and maintenance processes. Journal of Building Engineering. 49, 104028 (2022). https://doi.org/10.1016/j.jobe.2022.104028.

6. Terrazas, G., Ferry, N., Ratchev, S.: A cloud-based framework for shop floor big data management and elastic computing analytics. Computers in Industry. 109, 204–214 (2019). https://doi.org/10.1016/j.compind.2019.03.005.

7. Alioto, M., Shahghasemi, M.: The Internet of Things on Its Edge: Trends Toward Its Tipping Point. IEEE Consumer Electron. Mag. 7, 77–87 (2018). https://doi.org/10.1109/MCE.2017.2755218.

8. Nambiar, A., Mundra, D.: An Overview of Data Warehouse and Data Lake in Modern Enterprise Data Management. BDCC. 6, 132 (2022). https://doi.org/10.3390/bdcc6040132.

9. Inmon, B., Levins, M., Srivastava, R.: Building the data lakehouse. Technics Publications, Sedona (2021).

10. Brath, R.: Graph analysis and visualization: discovering business opportunity in linked data. Wiley, Indianapolis, IN (2015).

11. NetworkX - NetworkX Documentation, https://networkx.org/. [Accessed: 23/07/2024]

12. buildingSMART International: Industry Foundation Classes (IFC), https://technical.buildingsmart.org/standards/ifc. [Accessed: 23/07/2024]

13. topologicpy, https://pypi.org/project/topologicpy/. [Accessed: 23/07/2024]