2023 European Conference on Computing in Construction
40th International CIB W78 Conference
Heraklion, Crete, Greece
July 10-12, 2023

A PROPOSED SYSTEMS-CENTRIC ONTOLOGY FOR A GRAPH-BASED DIGITAL TWIN

Akhileswar Yanamala[1], Ashit Harode[1], and Walid Thabet[1] PhD, CM-BIM
[1]Virginia Tech, Blacksburg, VA, USA

Abstract

This research utilizes the Neo4j platform to create a graph database to represent the graphical components of a facility model and their spatial relationships. A graph-based digital twin is proposed for a 2-story facility by embedding systems-centric static data into the graph and linking it to the building's Navisworks BIM using a python script. The outputs of this study include a new ontology for creating graph-based digital twins, implementation of the graph using Neo4j, and a python script to link the graph to the model. Linking dynamic data to the graph model is explored and discussed. This approach can improve the representation and understanding of facility systems and their interrelationships.

Introduction

To improve facility management productivity, increase proactive maintenance decision-making, and reduce cost it is necessary to have access to operation and maintenance (O&M) data and information (Chen et al., 2020, Becerik-Gerber et al., 2012). This objective can be supported by using Building Information Modelling (BIM) to act as a central repository of information. But for BIM to be an effective tool to support facility management and operations it needs to be configured to provide critical O&M information along with as-built details of the facility and assets within it. Such BIMs are also known as FM-Capable BIMs or FM-enabled BIMs (Becerik-Gerber et al., 2012, Ensafi et al., 2022). Sacks et al., 2018, Sadeghi et al. (2018), and Yang and Ergan (2017), provided examples where data embedded in the handover BIMs focused only on space and asset management. They showed that the utilization of FM-Capable BIM to support facility management and maintenance is slow and is in its infancy.

Facility managers operate with a systems-driven point of view of their facility (Sacks et al., 2018). This means that they visually try to identify building systems to determine components' relationships and dependencies which would allow for faster and more informed decision-making during facility emergencies (Ensafi et al., 2022). Ensafi et al. (2022) defined and developed requirements and procedures for configuring system-centric as-built models. The research work defined four characteristics of a handover BIM model that can effectively support facility management and maintenance: (i) data-centric model, (ii) complete and accurate model graphics, (iii) systems-centric model, and (iv) ability to link the model to the owner's facility management system. As part of their validation, an FM-Capable system-centric BIM model was developed within Navisworks along with systems-centric data standards. System-centric viewpoints were created using the inbuilt "Find Items" functionality of Navisworks. The model was tested by identifying building systems components related to two emergency scenarios. Though the research by Ensafi et al. (2022) is relevant and effective in developing a Systems-Centric FM-Capable BIM, there are areas of improvement in its application. This is mainly due to how systems-centric data is stored and accessed in Navisworks, the software used for validating their study and a preferred software for the construction industry.

The area of improvement related to the storage of systems-centric data when utilizing tools such as Navisworks for creating systems-centric FM-Capable BIM remains unexplored. The systems-centric search of Navisworks model elements using the "Find Item" functionality works similarly to a search query in a relational database (tabular database). One of the limitations of relational databases is that, complicated relationships between individual objects become expensive to calculate and represent requiring a large number of joins between databases (Batra and Tyagi, 2012). Given that building assets are related to each other through systems, sub-systems, and locations, using only relational database-based queries to isolate elements would result in sub-par results, leading to sub-par utilization of FM-Capable BIM in facility maintenance and management.

To overcome this limitation the authors in this paper propose the use of a graph database to store system-centric data for FM-Capable BIM to perform searches required to improve proactive maintenance decision-making and to respond to a maintenance emergency. Graph databases provide equal importance to the relationship between the stored objects and the objects themselves (Batra and Tyagi, 2012).

Literature Review

Resource Description Framework (RDF) and Labeled Property Graphs (LPG) are two popular graph models. RDF is a World Wide Web Consortium standard model with ontologies and vocabularies openly available. Data in RDF is linked using a subject-predicate-object structure where the subject is a node, the predicate is an edge/relationship, and the object can be another node or a literal value (Baken, 2020). The nodes and edges in an RDF graph are named using an HTTP Unique Resource Identifier (URI). This HTTP URI allows the created RDF graphs to be seamlessly accessed by other stakeholders on the project. Therefore, RDF allows for the integration of many different types of data from multiple stakeholders, improving the interoperability of the created graph database.

On the other hand, LPG, natively used by the Neo4j platform used in this current research, has nodes and edges in its internal structure. The main difference between the LPG and RDF is that nodes and edges in LPG are capable of carrying properties within themselves in a key-value pair. Whereas, in RDF properties of a node can be described using additional nodes or literal values and relationships. This makes LPG graphs more compact in structure than the RDF (Baken, 2020). (Baken, 2020) after conducting qualitative and quantitative analysis of both mode graphs concluded that for real time operation LPG performed better than RDF.

Since the objective of this paper is to create a graph-based Digital Twin, the authors adopted a graph model that could easily represent the nodes as assets/spaces with systems-centric properties and can assist in the integration and navigation of real-time operational data. LPG allowed for embedding properties in nodes, making nodes a better representation of assets and allowed for a faster traversal of the network to support real-time data. For this paper the authors explored the LPG based graph model created using Neo4j.

The authors acknowledge the superiority of RDF based graph model in creating an interoperable graph database when compared to LPG due to its open-source ontology and use of HTTP URI. But since the objective of this research was not to test the interoperability of the created graph database but rather to test the useability of the graph database to accurately represent the building system and support real-time data integration LPG based Neo4j was chosen in this research over RDF.

The basic structure of graph databases is defined by nodes that represent the entity, edges that represent the relationship between the nodes, and properties. In the graph ecosystem, node and edge elements, their properties, and the relations between the elements can be defined as an ontology (McComb, 2019). Dermeval et al. (2016) have investigated multiple literatures and found empirical evidence in the field of RE (Requirements Engineering) for the benefits of using an ontology, focused on reducing ambiguity, inconsistency, and incompleteness. Stanciu (2021) along with RealEstateCore Consortium, developed a digital twins definition Language-based ontology to support the creation of digital twins of smart buildings in the real estate industry using graphs. Their ontology comprised of four main sets of interfaces including “Asset”, “LogicalDevice”, “Capability”, and “Space”, six additional base interfaces including “Agent”, “Building Component”, “Collection”, “Document”, “Event”, and “Role”, and nine relationships including “isPartOf/hasPart”, “hasCapability”, “includedIn”, “locatedIn”, “hosts”, “serves”, “feeds”, “hasBuildingComponent”, and “owns”. Gnecco et al. (2023) defined their own ontology to create a graph-based digital twin generated from a Revit model using Dynamo. The work investigated capturing real-time data from sensors installed in an academic lab space as well as from devices attached to occupants in that space. A Neo4J graph was used to represent the relations between the different entities such as people, environmental and wearable sensors, the supporting equipment used, and the facility. The ontology used four relationship types, including: IsPartOf, IsMontoredBy, TakesPartin, and IsLinkedTo.

Even though graph databases do not replace relational databases, they are especially suitable for storing data that contain many related data (Fernandes and Bernardino, 2018). The use of graph databases has been increasing in areas such as Semantic Web and Social Network Analysis (Fernandes and Bernardino, 2018).

The use of graph databases has also been explored in the construction industry. Hor et al. (2018) utilized graph databases as an efficient way of representing and visualizing real-world data. They proposed an architectural design that focuses on the integration of BIM-GIS data with Resource Description Framework graph databases with querying and filtering capabilities. This integration was validated by applying the developed graph database to an intelligent urban mobility web application on a game engine platform. Similarly, Malinverni et al. (2020) explored the use of a graph database to integrate both BIM and Geographic information System (GIS) data into a single graph. Other than just the storage of data, graph database has also been used to automate clash correction sequences based on a clash dependency network. Hu et al. (2020) proposed a graph-based network theory to improve the clash correction sequence. The paper utilized the graph database to analyze clash dependencies to predict the most optimum sequence for clash resolution and reduce the number of elements required to move to generate a clash-free model. Use of graph databases has also been explored in the creation of digital twins for the construction industry. Abdelrahman et al. (2022) utilized a graph database to combine spatial data extracted from BIM, the indoor location of the occupants, and thermal comfort feedback from the occupants. The database is then used as input for a classification machine learning algorithm to predict occupant thermal preference with a 14%-28% accuracy improvement over conventional thermal preference prediction input variables.

Durão et al. (2018) concluded that more applied research needs to be conducted in the implementation of digital twin. The authors in this paper support such notion and, by utilizing graph database to represent building systems, are proposing a graph-based digital twin to enhance building systems visualization to improve facility operations and provide better response to maintenance emergencies. The proposed research addresses three main questions: (1) How can a graph database be implemented to allow facility managers perform better systems-centric searches to improve proactive maintenance decision-making and better respond to emergencies? (2) What is the required graph ontology to facilitate the creation of a systems-centric graph database? and (3) How can the proposed graph database be utilized to implement a graph-based digital twin to support facility maintenance?

The following sections describe the case study used for implementation, and the research methods and steps proposed for developing the graph ontology and the graph-based digital twin. An example maintenance scenario search queries are presented to test the graph model. Finally, the paper concludes with summarizing the results and proposed future work.

Case Study Overview

The case study used in this research is an elementary and middle school two-story building, with a total gross area of 122,525 square feet. The building has spaces for classrooms, labs, studios, administrative offices, cafeteria, kitchen, and a gymnasium with some ancillary spaces. Using the 2D plans and the Revit model, the mechanical system was analyzed, and the building was divided into four (4) thermal mechanical zones based on the different configurations of the mechanical system components used to cool, heat, and ventilate the different spaces. All classrooms, labs, studios, and main administrative offices are designated as Zone 1. This zone employs four Dedicated Outdoor Air System (DOAS) units located on the roof and are the primary system providing 100% fresh outside air. A Variable Refrigerant Flow (VRF) system with several condenser units on the roof and several terminal units in the various spaces of Zone-1 are used as a secondary system allowing for localized temperature conditioning and control. The kitchen, cafeteria, and gymnasium areas are served by three separate packaged Rooftop Air Handling Units (RTUs) and comprise Zones 2, 3 and 4 respectively. The RTUs provide fresh intake air into the building that is mixed with a percentage of return air for energy savings.

The work presented in this paper will focus on Zone 4 which comprises the Gymnasium along with additional spaces including locker rooms and bathrooms for boys and girls, an office space, a storage room, and a small corridor. A packaged Rooftop Unit (RTU-1) provides supply air to the Gymnasium and other spaces. A percentage of air from the gymnasium space is returned and mixed with the outside air for energy savings. The other percentage is exhausted using two gravity roof ventilators (GRH-1 and GRH-2) located in the gymnasium ceiling. The gravity roof ventilators GRH-1 & GRH-2 have no moving parts and provide ventilation for the gymnasium space by removing heat (summer) and moisture (winter). Air from the toilets and locker rooms in Zone 4 is 100% exhausted using an exhaust fan (EF-6) located on the roof of the 2nd floor and directly above the gymnasium toilets. There are transfer ducts connecting the office and the storage room with the adjoining corridor to allow free air flow between these spaces providing for natural ventilation.

Figure 2 shows detailed 3D Revit model of the mechanical system for Zone 4 that will be represented using our graph-based digital twin. All equipment, terminal units (grilles and diffusers), duct and duct fittings 3D components were assigned a unique ID. Unique IDs for equipment (e.g. RTU-1 for roof top units) were extracted from the 2D design plans. Ducts, duct fittings and terminal units are assigned a unique ID defined by the authors based on the mechanical subsystem that the component belongs to. For example, duct components belonging to the supply air (SA) system are provided a sequential number with a SA designation (Duct-SA01 through Duct-SA27), whereas components belonging to the return air (RA) and exhaust air (EA) systems are numbered as Duct-RA01 to Duct-RA19 and Duct-EA01 to Duct-EA28 respectively. Duct fittings are numbered similarly to ducts. For example, Duct Fitt-SA01 through Duct Fitt-SA-34 were used for all duct fittings belonging to the supply air (SA) system. Grilles are numbered as GR-SA01 to GR-SA12, GR-RA01 to GR-RA02, and GR-EA01 to GR-EA02 for supply air, return air, and exhaust air systems respectively. Supply diffusers, return diffusers and exhaust diffusers are also numbered as Diff-SA01 to Diff-SA06, Diff-RA01 to Diff-RA04, and Diff-EA01 to Diff-EA06 respectively.

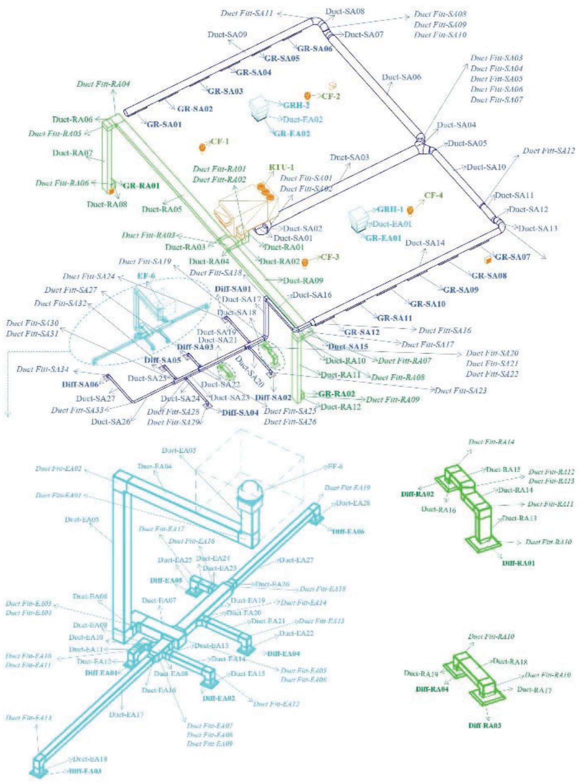
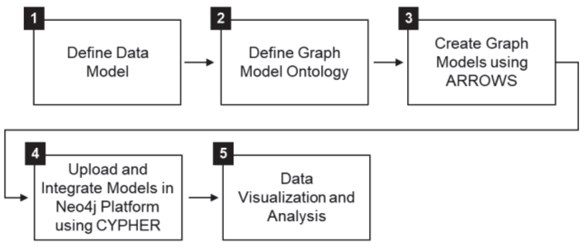


Figure 2: 3D Revit model of Zone 4 mechanical system showing components with their unique ID.

Research Methods

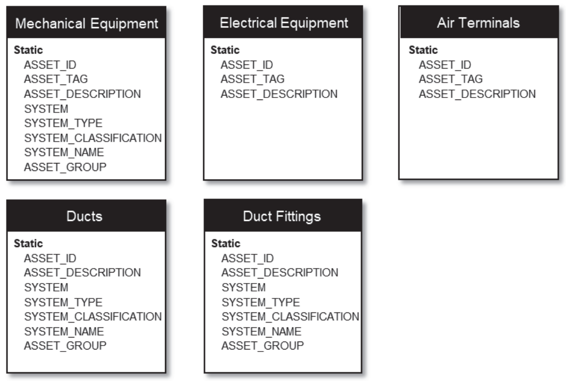


1
Define Data Model
2
Define Graph Model Ontology
3
Create Graph Models using ARROWS
4
Upload and Integrate Models in Neo4j Platform using CYPHER
5
Data Visualization and Analysis

Figure 3: Methodology steps.

Figure 3 shows a 5-step methodology that includes (1) a systems-centric data model and (2) a proposed graph ontology (3-4) to develop the graph-based digital model and (5) perform the systems-centric analysis using the case study.

In *Step 1*, the data model described in Figure 4 is used to define the different data parameters for typical components (or instances) of the mechanical system. Parameters could be static providing record data about the instance, or dynamic (telemetry) providing real-time performance data about the instance. The data model does not describe any relationships between instances of the graph.



Mechanical Equipment
Static
ASSET_ID
ASSET_TAG
ASSET_DESCRIPTION
SYSTEM
SYSTEM_TYPE
SYSTEM_CLASSIFICATION
SYSTEM_NAME
ASSET_GROUP

Electrical Equipment
Static
ASSET_ID
ASSET_TAG
ASSET_DESCRIPTION

Air Terminals
Static
ASSET_ID
ASSET_TAG
ASSET_DESCRIPTION

Ducts
Static
ASSET_ID
ASSET_DESCRIPTION
SYSTEM
SYSTEM_TYPE
SYSTEM_CLASSIFICATION
SYSTEM_NAME
ASSET_GROUP

Duct Fittings
Static
ASSET_ID
ASSET_DESCRIPTION
SYSTEM
SYSTEM_TYPE
SYSTEM_CLASSIFICATION
SYSTEM_NAME
ASSET_GROUP

Figure 4: Data model for Zone 4 mechanical system.

Step 2 defines a graph model ontology structure using nodes (vertices) and relationships (edges) to describe the different instances of Zone 4 mechanical system, the various relationship types, and their data model. The proposed ontology shown in figure 5 is developed to support a systems-centric view of the facility through a graph database and consists of a main set of interfaces and a number of relationship types. Instances of these interfaces are defined as nodes using a unique label. Nodes are assigned properties based on the data model defined in Figure 4. Two interfaces are currently defined:

1. Asset: An integral component of the facility that is not part of its structure. This includes mechanical equipment, air terminals (grilles and diffusers), electrical equipment (electric panels and circuits), ducts, and duct fittings.
2. Space: A continuous area or expanse. Three types of spaces are proposed including rooms, floors, and zones.

Nodes are interconnected using relationships that maybe assigned properties. We define five relationship types:

1. LOCATION_SERVED: Defines the coverage or service provided by a given asset (including equipment, ducts, duct fittings and air terminals) to one or more spaces. For example, a Roof Top Unit (RTU) may provide heating, cooling, and ventilation service to a number of rooms, floors or zones in a building.
2. EQUIPMENT_SERVED: Defines the support provided or impact of one equipment on another. For example, a fan may provide support to the Roof Top Unit (RTU).
3. LOCATION_CODE: Defines the space in which an asset (equipment and air terminals only) is physically located or installed. For example, a fan maybe located in a specific room within a floor and a zone.
4. CONNECTED_TO: Identifies the asset that another asset is connected to. For example, a duct component is connected to another duct component or a terminal unit (e.g. diffuser).
5. EMBEDDED_IN: Identifies the asset that hosts another asset.

The arrow direction for the LOCATION_SERVED, and LOCATION_CODE relationships is always from the specific asset node to its respective space node. For the CONNECTED_TO relationship type between different assets including, equipment, ducts/duct fittings and air terminals, the direction of the arrows follows the direction of airflow for the supply, return or exhaust air systems. The arrow direction orientation for other CONNECTED_TO relationships such as equipment and panels is from equipment to circuits and panels. For EQUIPMENT_SERVED relationships, the arrow direction is from the equipment providing service (e.g. fan) to the equipment being served (e.g. RTU or AHU).

In *Step 3*, the complete graph database for Zone-4 mechanical system was generated using the Arrows application (Arrows, 2020), a JavaScript-based tool from Neo4j labs. The application allows to graphically create graph nodes and edges and attach data such as node labels,

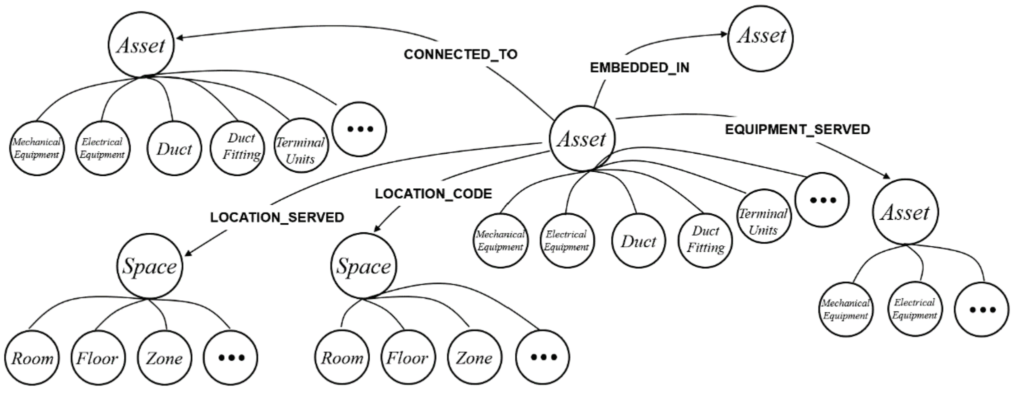


Figure 5: Graph Model Ontology.

edge types, and node/edge properties. Arrows allows to export graph models into different formats including Cypher, JSON, PNG, SVG and GraphQL. Five separate component graph databases were first created: Gym-SA, Gym-RA_EA, OtherSpaces-SA, OtherSpaces-RA_EA and Electrical. This allowed to focus on the specific scope of each component graph to ensure accuracy and eliminate errors.

Figure 6 depicts a segment of the Gym-SA graph database. The graph shows the RTU-1 equipment node and its properties, and several defined relationships or edges with other nodes, including Room, Floor, Zone, Ducts, Duct Fittings, and Electrical Equipment, also with their own properties.

The RTU-1 (Node: Mechanical Equipment) has two identification properties and six systems-centric properties defined. It is connected to DUCT-SA01 (Node: Ducts) through a CONNECTED_TO relationship type. DUCT-SA01 has one identification property and four systems-centric properties defined. The RTU-1 location is defined using two LOCATION_CODE relationships pointing at two space node types (Node: Floor and Node: Zone) with a single property for each node: FLOOR_NO = Roof and ZONE_NO = Zone-4. The space that the RTU-1 serves (i.e. provides service to) is defined using a LOCATION_SERVED relationship type and is associated with one space node type (Node: Room) with a single property: ROOM_NO = Gymnasium. To identify the electrical panel and circuit number that controls the electric supply to the roof top unit, the RTU-1 is connected to the U-LMPB-MEMA_1 circuit (Node: Electrical Equipment) using a CONNECTED_TO relationship with an assigned property denoting the circuit number. The circuit node is connected to the U-LMPB-MEMA electrical panel (Node: Electrical Equipment) also with a CONNECTED_TO relationship type. The panel location is defined using a LOCATION_CODE relationship and is associated with one space node type (Node: Room) with a single property: FLOOR_NO = ELEC 203.

Arrows automatically generates a Cypher query for each graph. In *Step 4*, the Arrows Cypher query for each of the five component graph databases was imported and merged in the Neo4j platform to create the complete graph model. Nodes that are common (repetitive) are combined to avoid creation of duplicate nodes and eliminate broken links in the final complete graph. Different common nodes exist between the five graphs and are identified manually and tabled. For example, the Mechanical System node with ASSET_ID = RTU-1 exists in four of the five Arrows component graphs created: Gym-SA, Gym-RA_EA, OtherSpaces-SA and Electrical. To track repetitive nodes and eliminate duplication, graphs were imported in the following order: Gym-SA, Gym-RA_EA, OtherSpaces-SA, OtherSpaces-RA_EA, and Electrical.

Specific modifications to the Cypher query for each component graph to eliminate the common repetitive nodes and combine the graphs is illustrated by the example shown in Figure 7.

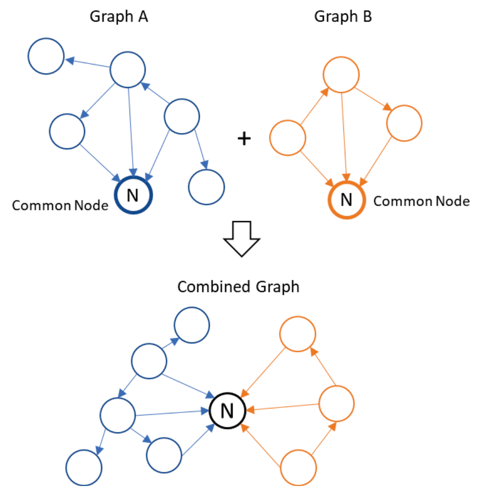


Figure 7: Example to show how the five component graphs are merged in Neo4j.

The figure shows two example graphs "Graph A" and "Graph B" with a common node "N". The two graphs can

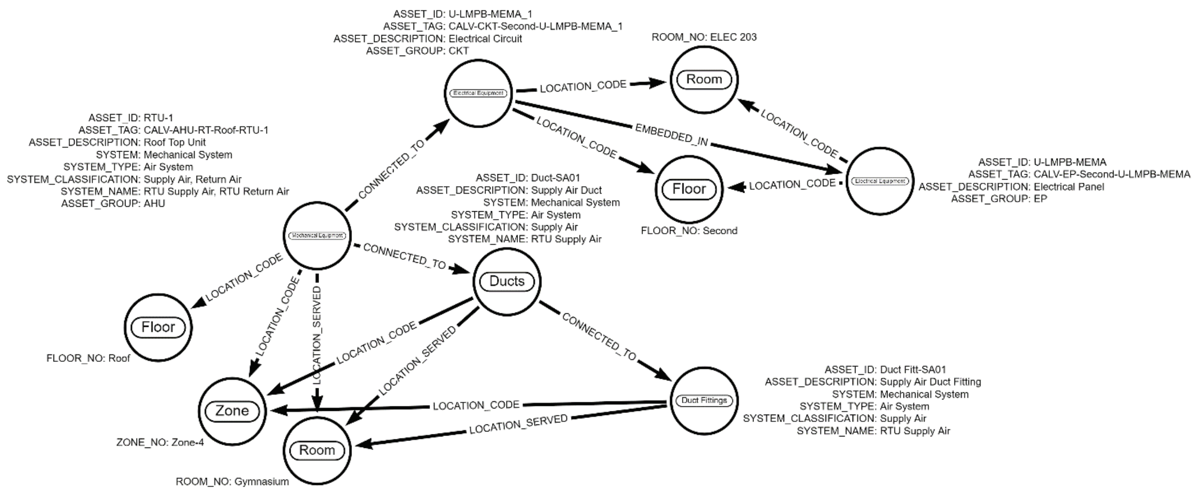


Figure 6: A segment of the Gym-SA graph database created using the Arrows application.

be combined to create a single graph database in Neo4j. using the following process:

1. The first Cypher query for "Graph A" imported from Arrows is executed in Neo4j to create the graph.
2. The second Cypher query of "Graph B" also imported from Arrows is first modified in Neo4j before it is executed:
    a. At the beginning of the Cypher query of "Graph B", a 'Merge' query is added to combine the common nodes "N" in "Graph B" and "Graph A" into a single node. The format for this additional query is:
    MERGE n*xx*: N{NodeProperty: "Property Value"}
    b. The Cypher query of "Graph B" is modified for the common node to eliminate the creation of a duplicate. For example, the node creation command n*xx*: N{NodeProperty: "Property Value"} would be changed to only 'n*xx*', since all the details of that node are already defined in the 'Merge' query added earlier.
    c. The modified Cypher query of "Graph B" is then executed to create a combined graph database of "Graph A" and "Graph B" without any duplicate nodes.

In *Step 5,* the combined Neo4j graph database customized with systems-centric properties is queried to filter and identify specific components of Zone 4 mechanical system necessary to be visualized during performing a standard maintenance task, or to address a maintenance emergency. Facility staff can execute pre-defined Cypher queries or create new ones to isolate and display nodes and edges of mechanical components associated with the maintenance task or are contributing to or impacted by the emergency. Using the Cypher queries, sub-graph databases can be isolated that contains the required asset and space nodes along with their relationship. The Asset_IDs of the identified assets in these sub-graphs are then extracted and exported to a csv file format using a Cypher query. An example Cypher query to export Asset IDs of nodes with same labels to a csv file format is shown in table 1.

Table 1: Cypher query to export graph as CSV

Cypher Query
RETURN a.ASSET_ID as Asset ID CALL apoc.export.csv.query(query, "file name.csv", {})

In parallel, using the Navisworks case study a search set is created to select a building system element in the model using the search rule "Category = ASSET PROPERTIES", "Property = ASSET_ID", "Condition = contains", and "Value = Duct-SA01". This search set is exported out of Navisworks as an XML file. This step is done to keep the XML schema intact when modifying the XML and importing it back to Navisworks in the next step. Once the excel file from neo4j and XML from Navisworks are obtained, Python code is used to modify the XML file to include additional search rules that have the "Value" field equal to the ASSET_IDs present in the excel file. The pseudo-code for this python code is provided in table 2. Once the XML file is modified it is saved and imported back into the Navisworks case study model.

Table 2: Pseudo-code to modify an XML file

```
Pseudo-code to modify an XML file
1:  df ← Import Excel File
2:  tree ← Import XML File
3:  a ← Find in tree child name "conditions"
4:  Loop for each value to ASSET_ID in df
5:          i ← ASSET_ID
6:            c ← Find in tree child name "condition"
with attribute "flags" value as "10"
7:          dupe ← Copy c
8:          Set dupe attribute "flags" value as "74"
9:          Set dupe's third child value as i
10:         Append dupe to a
11: Save modified XML
```

The following section describes an example maintenance scenario and the required Cypher query to identify the mechanical system components contributing to or impacted by the scenario.

Systems-Centric Analysis of Zone-4 Mechanical System using the Graph Database

To test the graph database, a system-centric query was performed to identify related nodes and relationships associated with a facility maintenance scenario. The objective of the query was to search and identify the assets and spaces needed to be visualized to perform the maintenance task.

Scenario: A facility manager has to switch off the RTU-1 unit (Mechanical Equipment) for regular maintenance. The circuit number (Electrical Equipment) that controls the RTU-1 unit and the panel number (Electrical Equipment) in which the circuit resides need to be identified. The facility manager also needs to determine the floor and the room where the electrical panel and RTU-1 are located. The Cypher query created in Neo4j to perform the search on graph database is shown in table 3.

Table 3: Cypher query to perform search for proposed scenario

Cypher Query:
Match (a: 'Mechanical Equipment' {ASSET_ID: 'RTU-1'}) – [:LOCATION_CODE] -> (b) Match (a) – [:CONNECTED_TO] -> (c: 'Electrical Equipment') Match (c)-[:EMBEDDED_IN]->(d: 'Electrical Equipment') Match (d)-[:LOCATION_CODE]->(e:Room) Return a, b, c, d, e

The cypher query uses 4 "Match" commands and requests 5 nodes and 4 relationships as output.

An alternative Cypher query to perform the same search and return the same results using a single "Match" command is shown in table 4.

Table 4: Alternative Cypher query to perform search

Alternate Cypher Query:
Match (a)<- [:LOCATION_CODE] - (b: 'Mechanical Equipment'{ASSET_ID: 'RTU-1'}) – [:CONNECTED_TO] -> (c: 'Electrical Equipment') - [:EMBEDDED_IN]->(d: 'Electrical Equipment')– [:LOCATION_CODE]-> (e:Room) Return a, b, c, d, e

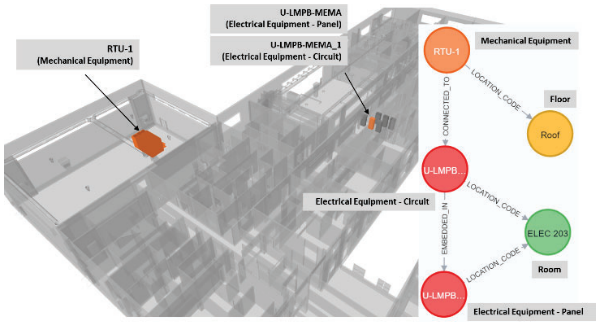


Figure 8: Highlighted components in the Navisworks model based on Neo4j scenario query.

The Python code is executed, and the XML file is modified. The assets corresponding to the neo4j sub-graph are isolated in the Navisworks model as shown in figure 8.

Any Cypher search query can be saved under a search phrase and description using Bloom within Neo4j. Bloom is a data visualization tool from Neo4j to explore and interact freely with the Neo4j graph database (Bloom, 2018). By saving different search queries for different maintenance scenarios using meaningful save phrases, faculty managers can use Bloom to execute any query to quickly identify components of their graph database without the need for coding. The need of Cypher coding by facility managers to query the graph database can be eliminated and pre-defined search phrases with predefined Cypher queries can be used every time the same information needs to be identified.

Dynamic Data

The digital twin developed by authors is a static-based twin. Generally speaking, a digital twin is a virtual representation of a physical object, system, or process that allows for real-time analysis, monitoring, and optimization. While digital twins are often associated with dynamic live data, it is important to note that this is not always the case. In fact, there is no standard definition of what a digital twin actually is. By providing a virtual model of a physical object or system, digital twins can help to optimize performance, reduce costs, and improve safety. Dynamic ive data is not in the scope of this paper, as the developed digital twin is a representation of a building assets only focusing on systems-centric static data to achieve the objective of the research. However, dynamic data can also be linked to the developed graph database from an external storage source. While this may not be real-time data, it is often close to real-time and allows for a more accurate understanding of how a physical object or system is performing.

Conclusion

The graph-based digital twin was tested to perform a systems-centric search to identify impacted building system components during a typical maintenance scenario. The twin graph database was queried to identify the circuit and electrical panel numbers associated with the roof top unit (RTU-1) and to determine their location in the building. The query returned the required information necessary to complete the maintenance task.

Facility Management (FM) staff can also utilize these searches to accurately determine impacts across the facility and develop strategies to respond rapidly and in real-time to any emergency by visualizing how systems interact with one another and with building spaces and their occupants.

The graph ontology developed is utilized as a blueprint for creating the graph database of the selected case study maintaining the consistency of the type data associated with it. The added advantage of the developed graph ontology is that it is dynamic and can be expanded in the future to define more nodes (interfaces) and relationships. Sensors and their relationships with existing assets and spaces can be defined to expand the developed ontology to include dynamic live data. The graph database developed is also flexible to make any changes or add extra components with ease, similar to the process of merging individual component graphs in this research. This makes it easier if the need arises to expand the ontology and to update the existing graph database accordingly.

Future research will also aim at a more streamlined/automated approach to link the graph database with the 3D model. This will involve creating an integrated platform, possibly a web application using .NET framework. The framework will host the 3D model using Autodesk Platform Services and connect it directly with the Neo4j graph database, so that users can query the graph directly from the web application. The proposed web application will provide users with an integrated solution to query and view assets and their data in a single solution.

The authors concur with Durão et al., (Durão et al., 2018) that more digital twin implementation research is needed as it would benefit the AEC industry in accelerating the evolution of digital twins. Various entities have reported and defined a multi-level scale of a digital twin maturity implementation. For example, Autodesk (n.d.) has defined five levels of maturity for a digital twin implementation: (1) Descriptive Twin, (2) Informative Twin, (3) Predictive Twin, (4) Comprehensive Twin, and (5) Autonomous Twin. Based on this definition, the authors have implemented a Descriptive graph-based digital twin. The Descriptive twin is limited to collecting and visualizing the data through the graph model and the associated query searches (What happened?). The work presented here will be expanded in the future to integrate real-time data collected from sensors. The maturity level of the digital twin implementation will also be expanded to an Informative level to allow to generate insights through aggregating and analyzing the data (Why did it happen?). An Informative digital twin will need to be supported by expanding the proposed ontology with new labels (e.g. to allow for sensor representation), and new relationship types.

References

Abdelrahman, M. M., Chong, A. & Miller, C. 2022. Personal Thermal Comfort Models Using Digital Twins: Preference Prediction With Bim-Extracted Spatial–Temporal Proximity Data From Build2vec. Building And Environment, 207, 108532.

Arrows. 2020. Arrows [Online]. Neo4j Labs. Available: Https://Arrows.App/ [Accessed December 17, 2023].

Autodesk. N.D. Digital Twins In Construction, Engineering & Architecture [Online]. Available: Https://Www.Autodesk.Com/Solutions/Digital-Twin/Architecture-Engineering-Construction [Accessed December, 17 2022].

Baken, N. Linked Data For Smart Homes: Comparing Rdf And Labeled Property Graphs. Ldac2020—8th Linked Data In Architecture And Construction Workshop, 2020. 23-36.

Batra, S. & Tyagi, C. 2012. Comparative Analysis Of Relational And Graph Databases. International Journal Of Soft Computing And Engineering (Ijsce), 2, 509-512.

Becerik-Gerber, B., Jazizadeh, F., Li, N. & Calis, G. 2012. Application Areas And Data Requirements For Bim-Enabled Facilities Management. Journal Of Construction Engineering And Management, 138, 431-442.

Bloom, N. J. 2018. Bloom [Online]. Neo4j. Available: Https://Neo4j.Com/Product/Bloom/ [Accessed December 17, 2022].

Chen, W., Das, M., Chen, K. & Cheng, J. C. Ontology-Based Data Integration And Sharing For Facility Maintenance Management. Construction Research Congress 2020: Computer Applications, 2020. American Society Of Civil Engineers Reston, Va, 1353-1362.

Dermeval, D., Vilela, J., Bittencourt, I. I., Castro, J., Isotani, S., Brito, P. & Silva, A. 2016. Applications Of Ontologies In Requirements Engineering: A Systematic Review Of The Literature. Requirements Engineering, 21, 405-437.

Durão, L. F., Haag, S., Anderl, R., Schützer, K. & Zancul, E. Digital Twin Requirements In The Context Of Industry 4.0. Ifip International Conference On Product Lifecycle Management, 2018. Springer, 204-214.

Ensafi, M., Harode, A. & Thabet, W. 2022. Developing Systems-Centric As-Built Bims To Support Facility Emergency Management: A Case Study Approach. Automation In Construction, 133, 104003.

Fernandes, D. & Bernardino, J. 2018. Graph Databases Comparison: Allegrograph, Arangodb, Infinitegraph, Neo4j, And Orientdb.

Gnecco, V. M., Vittori, F. & Pisello, A. L. 2023. Digital Twins For Decoding Human-Building Interaction In Multi-Domain Test-Rooms For Environmental Comfort And Energy Saving Via Graph Representation. Energy And Buildings, 279, 112652.

Hor, A., Gunho, S., Claudio, P., Jadidi, M. & Afnan, A. 2018. A Semantic Graph Database For Bim-Gis Integrated Information Model For An Intelligent Urban Mobility Web Application. Isprs Annals Of Photogrammetry, Remote Sensing & Spatial Information Sciences, 4.

Hu, Y., Castro-Lacouture, D., Eastman, C. M. & Navathe, S. B. 2020. Automatic Clash Correction Sequence Optimization Using A Clash Dependency Network. Automation In Construction, 115, 103205.

Malinverni, E. S., Naticchia, B., Lerma Garcia, J. L., Gorreja, A., Lopez Uriarte, J. & Di Stefano, F. 2020. A Semantic Graph Database For The Interoperability Of 3d Gis Data. Applied Geomatics, 1-14.

Mccomb, D. 2019. Semantic Ontology: The Basics [Online]. Available: Https://Www.Semanticarts.Com/Semantic-Ontology-The-Basics/ [Accessed December 11, 2022].

Sacks, R., Eastman, C., Lee, G. & Teicholz, P. 2018. Bim Handbook: A Guide To Building Information Modeling For Owners, Designers, Engineers, Contractors, And Facility Managers, John Wiley & Sons.

Sadeghi, M., Mehany, M. & Strong, K. Integrating Building Information Models And Building Operation Information Exchange Systems In A Decision Support Framework For Facilities Management. Construction Research Congress 2018, 2018. 770-779.

Stanciu, A. 2021. Realestatecore, A Smart Building Ontology For Digital Twins, Is Now Available [Online]. Internet Of Things Blog By Microsoft: Microsoft. Available: Https://Techcommunity.Microsoft.Com/T5/Internet-Of-Things-Blog/Realestatecore-A-Smart-Building-Ontology-For-Digital-Twins-Is/Ba-P/1914794 [Accessed December 12, 2022].

Yang, X. & Ergan, S. 2017. Bim For Fm: Information Requirements To Support Hvac-Related Corrective Maintenance. Journal Of Architectural Engineering, 23, 04017023.