A Framework for Generic Semantic Enrichment of BIM Models

Article *in* Journal of Computing in Civil Engineering · October 2023

DOI: 10.1061/JCCEE5.CPENG-5487

5 authors, including:

Zijian Wang
Technion - Israel Institute of Technology
14 PUBLICATIONS **174** CITATIONS
SEE PROFILE

Rafael Sacks
Technion - Israel Institute of Technology
260 PUBLICATIONS **15,071** CITATIONS
SEE PROFILE

Boyuan Ouyang
Technion - Israel Institute of Technology
7 PUBLICATIONS **27** CITATIONS
SEE PROFILE

Huaquan Ying
Technion - Israel Institute of Technology
22 PUBLICATIONS **136** CITATIONS
SEE PROFILE

A Framework for Generic Semantic Enrichment of BIM models

Zijian Wang*[1], Rafael Sacks[1], Boyuan Ouyang[1], Huaquan Ying[1], and André Borrmann[2]

[1]VClab, Faculty of Civil and Environmental Engineering, Technion Israel Institute of Technology, Haifa, Israel. Corresponding author email: zijian.wang@campus.technion.ac.il

[2]Chair of Computational Modeling and Simulation, Technical University of Munich, Munich, Germany.

This paper has been published in the Journal of Computing in Civil Engineering.

DOI: https://doi.org/10.1061/JCCEE5.CPENG-5487.

Citation: Wang, Z., Sacks, R., Ouyang, B., Ying, H. and Borrmann, A., 2023. A Framework for Generic Semantic Enrichment of BIM Models. Journal of Computing in Civil Engineering, 38(1), pp. 1–18.

ABSTRACT

The design intent and many meaningful semantics in Building Information Modeling (BIM) models are often implicit, and some explicit semantics are lost during model exchanges. These missing information can be artificially supplemented through a process called semantic enrichment. However, previous research on semantic enrichment has primarily focused on specific tasks, leading to a limited scope of predictions, and lacks comprehensive, seamless approaches. In this study, we aim to infer BIM semantics from fundamental model data, pure object geometries, and organize the predicted results into a graph-based Common Data Environment (CDE) to support intelligent applications. Consequently, we propose a framework of generic semantic enrichment which includes four fundamental tasks in the context of graphs and a process control mechanism to execute a set of tools in a proper sequence. To validate its feasibility, we selected a real-world apartment model and developed six tools to generate the graph-based CDE from its object geometries. Additionally, applications were implemented on the graph-based CDE, such as 1) enriching

a pure geometry model from SketchUp to a BIM model in Revit, and 2) easing interoperability problems by reconstructing ArchiCAD models in Revit or downgrading Revit models to earlier versions. The experiment's success demonstrates the feasibility of constructing BIM graphs from object geometries through semantic enrichment to enable intelligent applications. This study establishes a theoretical foundation for graph semantic enrichment and opens a door to further explore intelligent applications on BIM graphs.

1 INTRODUCTION

The Architecture, Engineering, and Construction (AEC) industry is characterized by its fragmented nature, involving numerous stakeholders with specialized skills who work together throughout the lifecycle of a project (Borrmann et al. 2018). BIM technologies assist participants to design, store and deliver models virtually and digitally (Sacks et al. 2018). Each expert has specific requirements and preferences for BIM authoring tools, and prefers to keep their own models rather than work on an all-encompassing one. Therefore, intense collaborations and communication are necessary throughout the lifecycle of BIM projects. Ideally, modeling data can be delivered across disciplines and software vendors without information loss and maintain accessibility.

In reality, BIM vendors develop their proprietary schemas which are designed for data exchange inside one company's ecosystem and are usually confidential and inaccessible to others. This leads to the interoperability problem across BIM platforms. One common solution to the issue is to map modeling information from native schemas to an open schema, i.e. the Industry Foundation Classes (IFC). However, part of the modeling information is lost during the compilation process (Pazlar and Turk 2008) and mapping errors are common (Venugopal et al. 2015).

Semantic enrichment (SE) aims to interpret implicit building semantics and supplement them back to models, so they can be reused for multiple purposes with minimal rework (Belsky et al. 2016; Bloch 2022). Previous studies focused primarily on methods that can be deployed in semantic enrichment tasks, such as using expert systems, machine learning, and deep learning, to achieve BIM object classification (Wu et al. 2022; Yu et al. 2022). These studies illustrated the feasibility of using different techniques for SE, but they usually focus on a specific type of task rather than

providing comprehensive required BIM semantics. Additionally, the enrichment process typically terminates after executing the developed method, so that reuse of the enriched results is not tested.

In fact, exact object geometries which contain objects' dimension and location information could be regarded as the fundamental data for any BIM objects as long as the Level of Development (LOD) is beyond 200. Basically, almost all BIM design tools consistently support export of solid geometries into accessible and generic file formats (Sacks et al. 2018).

Additionally, researchers have suggested using a graph data structure to store building information, as a graph is a flexible way to store any information and graphs have the advantageous capability to represent relationships among BIM objects explicitly. The Linked Building Data (LBD) community has contributed many efforts to standardize the terminologies and formats of BIM graph representation by constructing open ontologies and leveraging techniques from the semantic web domain (Pauwels et al. 2022). On this basis, BIM graphs could have uniform terminologies and the graphs are accessible to others.

At the same time, researchers have recognized that certain types of BIM data, such as time-series data and object geometries, are not well-suited for representation in a graph format (Pauwels et al. 2022). Therefore, Ouyang et al. (2022) proposed a core-extension layer graph data structure where BIM objects and relationships are represented in a core graph and other data are stored in corresponding suitable formats in the extension layer but linked back to the core graph. This is similar to the concept of CDE proposed by ISO 19650 (2018), although the original concept of CDE was designed at the file-based level, and used to support coordination across disciplines to facilitate collaboration. In this study, we adopt the concept of a common data environment, but redefine it as the data container that can host the object-level graph and other types of data, naming it 'graph-based CDE'.

Recognizing that object geometries are the foundational data in BIM models, we aim to enrich BIM semantics step by step, starting from pure geometries and continuing until all potential information is recovered or the requirements are met. Therefore, in this study, we investigate 1) the use of semantic enrichment to supplement information from pure object geometries and

organize the results into a graph-based CDE, and 2) the utilization of the graph-based CDE to enable intelligent applications.

To achieve these goals, we propose a framework of generic semantic enrichment in Section 4. We then construct a series of SE tools in Section 5, and select a case study to generate its graph-based CDE for implementing applications (Section 6). In Section 7, we discuss the strengths and limitations of this study and the potential for extending it to across-domain coordination. Literature review, research questions and objectives, and conclusions are presented in Sections 2, 3 and 8, respectively.

2 BACKGROUND

2.1 Semantic enrichment of BIM

Modern BIM tools provide a three-dimensional environment to design models with geometry, alphanumeric data, and semantics. However, these tools still struggle with interpreting information that is not explicitly represented in the model (Bloch 2022). When BIM models of existing facilities are created from point cloud capturing technologies the results are purely geometric models (Mafipour et al. 2022). Insufficient semantics can impact data exchange quality and hinder or prevent the use of a range of building analysis and performance simulation applications. Therefore, semantic enrichment of BIM models has been presented to infer implicit semantics automatically (Belsky et al. 2016), and facilitate their use in a target application or procedure (Bloch 2022).

An early study developed a rule-based algorithm for classifying bridge components (Sacks et al. 2017), demonstrating the feasibility of semantically enriching BIM models while opening the door for the use of artificial intelligence (AI) techniques for BIM object classification. Rule-based approaches, which consist of machine-readable rules that encapsulate expert knowledge, have the advantage of being transparent to users, but they lack the scalability to extend to other scenarios. Consequently, data-driven methods based on statistical theories have been applied in recent years. For example, Koo et al. 2019; Koo et al. 2021 used a machine learning algorithm, Support Vector Machine (SVM), and two 3D deep learning models, Multi-view Convolutional Neural Networks

(MVCNN) and PointNet, to classify IFC objects. Researchers have also converted IFC objects into points and compiled the points into graphs to launch Graph Neural Networks (GNN) for classifying the types of objects (Collins et al. 2021). Another study compiled BIM apartment models into graphs and applied GNN algorithms for room type classification, illustrating that GNN can leverage relationship features in the graph to improve prediction capabilities (Wang et al. 2021; Wang et al. 2022b). The success of all these studies has broadened the toolbox for semantic enrichment.

In addition to object classification, Bloch and Sacks (2020) identified three other enrichment tasks: creation, association, and calculation. A rule-based method, topology-based inter-domain object correspondence (TIOC) algorithm, was introduced by Ouyang et al. (2022) to establish topological and correspondence relationships across BIM disciplinary graphs. This work demonstrated the feasibility of accomplishing association tasks and expanding the scope of semantic enrichment to multi-discipline contexts. Apart from that, there are few studies on developing tools to address the generation and calculation tasks.

Furthermore, existing SE studies concentrate on developing different techniques for specific tasks, and the enriched results are limited to a single type of building semantics, such as object types. However, a BIM model contains more information than just object types, encompassing object attributes, relationships, design intent, and more. Additionally, previous studies presented inference results to users but rarely integrated them back into the BIM model, which prevents reuse of the predictions in downstream procedures (Bloch 2022). There is thus also a need for a method to represent both existing and generated semantics explicitly in a way that enables downstream applications.

2.2 Interoperability

The concept of BIM relies on the consistent use of a comprehensive building model as a foundation for all data exchange operations, eliminating the need for manual data entry and reducing the risk of errors (Borrmann et al. 2018). Interoperability, or the ability to exchange data between applications (Sacks et al. 2018), is essential for improving collaborative workflows, and this is recognized as a critical factor for successful design projects (Won et al. 2013).

However, the AEC industry still faces interoperability challenges. First, BIM information is complex in the sense that it encompasses object geometries and various semantics. The industry involves diverse experts with different knowledge and experience, leading to multiple representations of a building project and the management of heterogeneous models (ISO 19650 2018). In addition, no one organization in the AEC industry has the economic clout or knowledge to define effective interoperability for the whole industry (Sacks et al. 2018). Last, commercial software vendors often create their own ecosystems using proprietary schemas, making it difficult to exchange information across platforms and disciplines (Varoudis and Patlakas 2014).

BuildingSMART, an international organization, has dedicated years to developing the IFC as an open, vendor-neutral data exchange format (BuildingSmart 2019). Due to its open schemas and adoption by software, models can be compiled from proprietary schemas to IFC files and shared with other participants (Borrmann et al. 2018). In this regard, the IFC approach offers a practical solution for addressing interoperability issues, and has became the prevalent method today (Rasmussen et al. 2021). However, it is challenging to define a holistic schema that meets the entire industry's requirements (Kim et al. 2012), and translation errors may occur when mapping models from proprietary schemas to IFC (Venugopal et al. 2015).

Instead of manually sharing BIM files, researchers have explored uploading BIM models to a central cloud repository (Hietanen 2002). An early BIM cloud platform, BIMserver from Beetz et al. (2010), can store IFC data in the cloud, enabling functions like model checking, merging, and versioning. There are also commercial platforms, such as BIM Collaborate from Autodesk Inc. (2022), which provide a cloud repository for users to upload and access BIM models, but these models are still stored in files. Compared to the conventional approach, cloud BIM platforms distinguish themselves by using cloud servers to store model files instead of relying on local hard drives. It remains a file-based approach for collaboration. There is also a trend towards exploring object-level representations of BIM models to enable intelligent applications, like cross-domain consistency maintenance and version control (Beetz et al. 2010; Pauwels et al. 2011; Törmä 2013; Sacks et al. 2022; Esser et al. 2022). This approach explores the possibility of communicating data

at the object level instead of the file level with the goal of enhancing interoperability.

Another approach to facilitate data exchange is to use an Application Programming Interface (API) to retrieve required information from the target software. Examples include Revit's Open API and Bentley's MDL. Many interfaces are designed for software within a single company's ecosystem or through business agreements between two or more companies (Sacks et al. 2018). As such, interfaces relying on direct links can be considered as extensions of the software, while interoperability issues persists when the exchange goes beyond a single vendor's ecosystem.

3 RESEARCH GAPS, OBJECTIVES, QUESTIONS AND METHODOLOGY

Much of the research to date on semantic enrichment of BIM models focuses on the feasibility and performance of AI tools for SE. Several gaps remain to be addressed. Existing SE studies accomplished tasks for object classification and association, but tools for creating objects and calculating attribute values are still lacking. There is a need to develop diverse SE tools to predict the comprehensive semantics of BIM models. Additionally, while previous studies showed local successes, a broad SE solution to solve the interoperability problem remains absent.

To address these gaps, this study explores use of fundamental BIM model data, i.e. pure object geometries, as the starting point for developing a diverse set of SE tools to enrich building semantics. The predicted results are then organized as a graph-based CDE to facilitate operation of analysis and simulations on BIM graphs. Therefore, we aim to 1) construct a framework for enriching semantics from object geometries; 2) develop a series of SE tools to accomplish all SE tasks under the framework; 3) implement intelligent applications that leverage the enriched semantics to address BIM challenges.

Consequently, we will answer these research questions: Is it feasible to construct a graph-based CDE of BIM models from pure geometries using semantic enrichment? Can we construct intelligent applications on the graph-based CDE for solving BIM problems?

This research follows the design science methodology (vom Brocke et al. 2020). We initially identified problems from the industry as well as from the literature, and defined the objectives of the study. Next, we created a framework of generic semantic enrichment. Then we developed

experiments based on a case study to demonstrate the solution path for the identified problems. Limitations and feedback from the experimental results help us improve the theory and the tools. This process is repeated until the experiment results can provide answers to the predefined research questions.

4 FRAMEWORK OF GENERIC SEMANTIC ENRICHMENT

Generic semantic enrichment aims to generate BIM graphs from object geometries to enable advanced BIM applications. The framework of generic semantic enrichment comprises the fundamental enrichment tasks and the process control mechanism, as illustrated in Fig. 1. There are four types of fundamental semantic enrichment tasks under the context of graphs (Section 4.1). Tools are constructed to accomplish enrichment tasks, and executed in a sequence (Section 4.2). The enriched semantic results are structured into a graph-based CDE with a core graph layer containing building objects and relationships, and a separate extension layer storing geometries (Section 4.3). The graph-based CDE can be used for various applications, such as enriching geometry design, easing interoperability issues and linking multi-disciplinary graphs for maintaining model consistency.

4.1 Semantic enrichment tasks and tools

We have defined four fundamental SE tasks: object classification, attribute computation, relationship determination and object generation. These are derived from Bloch and Sacks (2020) but tailored to fit the graph context (Fig. 1 (a)). Object classification utilizes methods to predict the types of objects. Attribute computation aims to calculate object properties, such as dimensions. Once computed, these attributes can be inserted back into object nodes.

Relationship determination involves calculating new associations among BIM objects, making their relationships explicit. Relationships can describe the topological facts of two objects, such as adjacency. Additionally, there are functional and constraint relationships between BIM objects (Sacks et al. 2022). For instance, *host* could be a functional relationship between a door and a wall, and a *minimum distance* constraint exists between a pipe (which belongs to the Mechanical, Electrical, and Plumbing (MEP) domain) and an architectural wall where the pipe is installed on the surface of the wall with a small gap. Explicitly representing functional and constraint relationships

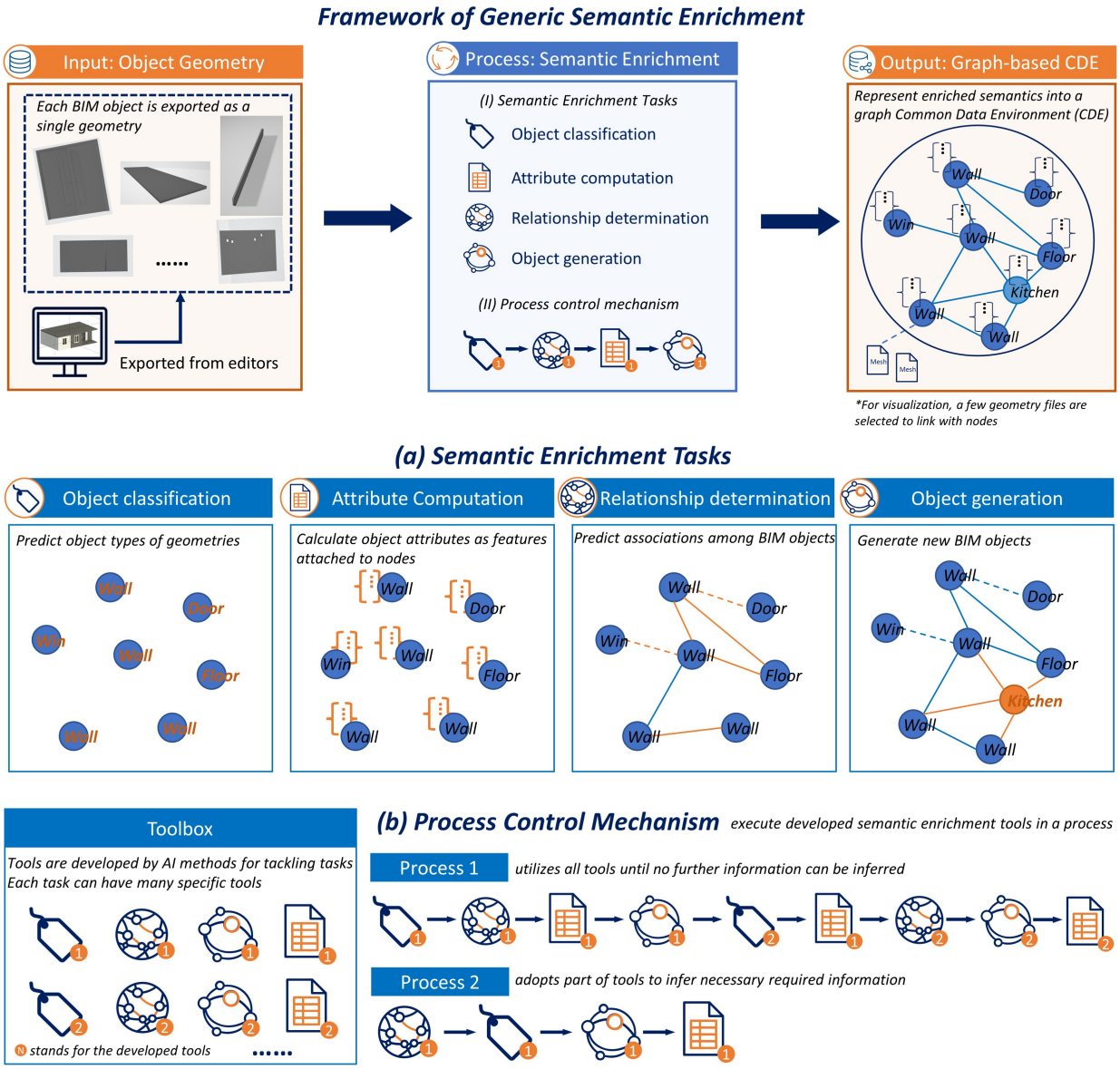

Fig. 1. Framework of generic semantic enrichment ((a) Semantic Enrichment Tasks, (b) Process Control Mechanism).

has been shown to facilitate advanced applications, such as cross-domain change propagation (Wang et al. 2022a).

Object generation is the task of creating nodes for new objects, often requiring also new relationships to associate the new nodes with existing nodes. The newly generated nodes can be physical or virtual BIM elements and can represent decomposition or aggregations of elements. For instance, when entering a more detailed design, one architectural wall could be divided to

generate two finishing layers and a core layer. Conversely, two structural columns and a beam could be aggregated to form a single structural frame object.

Tools are developed methods to accomplish tasks. For instance, numerous studies employ AI methods, such as rule-based algorithms, machine learning, and deep learning, to create SE tools for achieving BIM object classification tasks. Despite these tools being constructed using different methods, they all aim to accomplish the same task of BIM object classification.

4.2 Process control mechanism

Process control is widely adopted in the manufacturing industry to monitor and adjust production processes to achieve desired outputs and to optimize the overall performance of the system (Stavropoulos et al. 2013; Chryssolouris et al. 1992). The process of enriching a BIM model is similar to making a product in manufacturing, in which the inputs - object geometries - are fed to a series of SE tools to generate an enriched graph-based CDE, considered as the outputs.

In the context of generic semantic enrichment, the process control mechanism executes a variety of developed SE tools in a sequence to infer implicit semantics. The 'toolbox' on the left-hand side of Fig. 1 (b) contains tools intended for different tasks and scenarios. Certain tools use pure geometry as input, while others may require the output from other tools as input. Therefore, the sequence of tool execution is influenced by the nature of the available tools, the input requirements for each tool, and any dependencies between tools. The enrichment process is programmed to terminate when no additional information can be inferred, ensuring all possible data are generated. Additionally, one could program the process to terminate if and when some predefined set of data requirements is satisfied. Both conditions are shown on the right-hand side of Fig. 1 (b). Note that in cases where the tools cannot yield all the information required by target applications or users - for example, where data is insufficient for inference, or where the confidence level of a prediction falls below a preset threshold - the system must refer to the user to provide any missing information.

The mechanism should be evaluated for correctness, efficiency, and robustness. Efficiency can be evaluated by the required computational resources of tools in general, and the total processing

time of the process. Moreover, there is a risk of failing downstream tools if an upstream tool generates incorrect outputs. In the worst case, erroneous input could impact all subsequent tools, reducing the system's overall robustness. Ensuring the accuracy of tools whose outputs are fed to others is crucial. One potential path to improvement is to combine the results of several tools. For example, rule-based, machine learning and deep learning methods are different tools designed for the same object classification task, while the final results could be determined by assessing the levels of confidence among the three predictions.

4.3 Interoperable graph-based CDE

The enriched output from each step is organized into a graph-based CDE following the data structure designed by Ouyang et al. (2022). The data repository consists of a core layer and an extension layer, where the core layer contains the BIM graph representing the building objects and their relationships, and the extension layer stores geometries that are virtually linked to the corresponding object nodes in the core layer. In the graph, nodes represent objects, edges represent relationships, and object attributes are inserted into the object nodes. To ensure transparency and readability, ontologies from the LBD community are adopted to standardize the names, and the graph is compiled following Resource Description Framework (RDF) rules. Geometries in the extension layer are also stored in an open format, ensuring interoperability of the entire CDE. The graph format and structure of the CDE are designed to be generic and accessible rather than being tailored to a specific purpose.

4.4 Why generic?

The term *generic* has multiple meanings in this pipeline. Firstly, it refers to the use of accessible and interoperable data as the only input to the pipeline. We utilize exact geometries that are essential in all 3D models and can be exported from design software without loss of information.

Secondly, it encompasses the four fundamental types of graph semantic enrichment tasks. These tasks have been summarised from various situations of enriching BIM graphs and can guide users in developing tools accordingly.

Thirdly, it relates to the transparent and open representation of output data. All semantics are

organized into a graph-based CDE with a well-defined data structure. The terminology utilized in the graph conforms to open ontologies and the compilation follows RDF rules, allowing receivers to access and interpret the enriched data canonically.

Lastly, the enriched data has diverse potential applications. We do not limit the purposes or provide guidelines to the use of the enriched data. Instead, the enriched data is a flexible and versatile resource that researchers and developers can utilize in a variety of ways to cater to their specific needs.

5 EXPERIMENTAL IMPLEMENTATION

To validate the feasibility of generic semantic enrichment, we have chosen the initial design of an apartment as the scenario. This is a common situation in which architects often exchange models with colleagues using various software and platforms. This scenario is not excessively complex either, making it suitable for assessing the feasibility of the concept from an implementation perspective. Specifically, we selected a real-world apartment project, Apartment 2 (Apt2) at LOD 200, to demonstrate the development of semantic enrichment tools and the enrichment process. The Apt2 model is shown in Fig. 2.

We have developed five specific tools and combined them in a six-step process for enriching the apartment as shown in Fig. 3. Each tool has different input and output and is constructed using AI methods or geometry computation. The first tool, object classification, predicts object types from the geometries in Experiment (Exp) 1. Another tool generates *hosting/hosted* and *adjacent* relationships among objects using geometries in Exp 2. Two tools for object generation have been developed: one for essential building topology objects and another for space objects (Exp 3). The last tool computes object attributes based on geometries and object types (Exp 4).

For implementation, we primarily utilized Python programming language to develop rule-based algorithms incorporated with scikit-learn (Pedregosa et al. 2011) for machine learning and Deep Graph Library (DGL) (Wang 2019) for graph deep learning. To handle geometry computation, we selected the Trimesh library (Dawson-Haggerty et al. 2019). For compiling and retrieving RDF graphs, we utilized the RDFLib library (Boettiger 2018). Revit was selected as the design software,

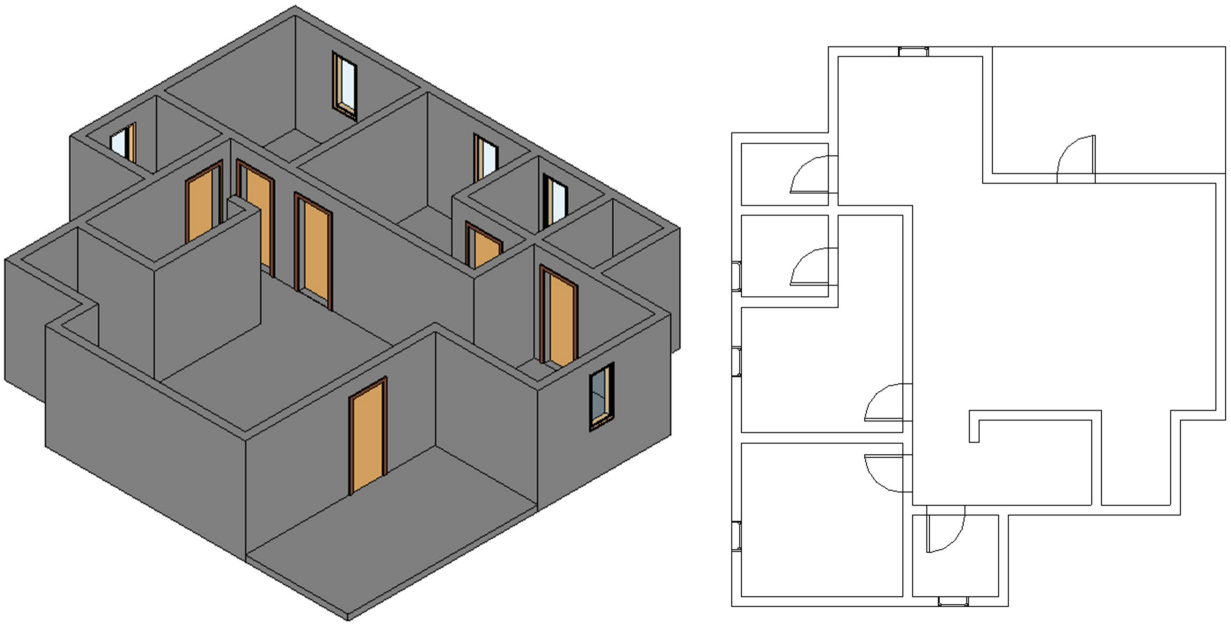

Fig. 2. Diagram of Apt2.

Process	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
Task	Object classification	Attribute computation	Relationship determination	Object generation	Object generation	Attribute computation
Input	Geometry	Geometry, object types	Geometry	Geometry, object attributes	Geometry, object attributes	Geometry, object types
Output	Object types: *wall, slab, window, door*	Physical object dimension attributes	Relationships: *adjacent, hosting/hostedby*	Generate *level, building, site* objects and *containing* links	Generate *space*, classify *space type* and link with bounding objects	Space object dimension attributes
Method	Machine learning	Geometry computation	Rule-based inferencing	Rule-based inferencing	Rule-based inferencing + graph neural network	Geometry computation
Tool development	Exp 1*	Exp 4	Exp 2	Exp 3.1	Exp 3.2	Exp 4

*Exp is an abbreviation for Experiment.

Fig. 3. Semantic enrichment tools and process for this experiment.

and we used Dynamo, a visual programming tool within Revit, to access and modify information in the Revit database. Finally, we employed GraphDB to visualize graphs (Güting 1994). All experiments were performed on a personal computer with an Intel 6-core i7 CPU (2.70 GHz) and 32.0 GB RAM.

5.1 Enrichment process design

The enrichment process is detailed in Fig. 3. Object classification and relationship determination, two tools that only require geometry information, are executed at the early stage. Afterward, building spatial structural objects such as the *site*, *building*, and *levels* are created using geometry and physical object attributes. Additionally, space objects are formed and linked to the bounding physical objects. The attribute computation tool is executed twice, first for establishing the attributes of physical objects needed to generate the spatial objects, then for calculating the attributes of newly formed space objects. The enrichment process stops when no more semantics can be inferred. It is important to note that there can be numerous combinations of tools in creating the enrichment sequence, and the presented one is an example. The sequence of the attribute computation and the relationship determination tools could be switched, among other possible variations.

5.2 Exp 1: Object type classification

BIM object classification is not a novel challenge, and various methods have been explored by researchers to address this problem. Wu et al. (2022) found that machine learning methods that leverage geometry, location, and metadata features can obtain accurate performance, with a precision of approximately 99% on five classes of objects that were tested. Regardless of correctness, machine learning is often more computationally efficient than deep learning. Machine learning also demonstrates better scalability than rule-based algorithms that rely on hand-crafted rule sets. Based on these evaluations of correctness, efficiency and scalability, we decided to use machine learning techniques for classifying apartment objects.

Apartment initial design generally comprises floors, walls, doors, and windows. We explored existing open datasets and found that they did not satisfy our requirements in terms of scope, format, and quality. IFCNetCore, a benchmark consisting of 7,930 objects from 20 classes (Emunds et al. 2022), does not include the window class in its open core version. ArchiShapesNet, another open dataset, contains all four required objects types (Yu et al. 2022), but it is composed of 2D rendered images instead of original 3D geometry files. Additionally, BIMGEOM (Collins et al. 2021) includes the required objects, but quality issues emerged upon examination. For instance, some

objects are incorrectly labeled as "windows", and some geometries are not watertight.

Therefore, we decided to construct our own dataset. BIM models of 32 apartment units were collected from projects in Israel which included both architectural and MEP models. A Dynamo program was executed to extract the geometry of each object from Revit's database and save its mesh as a Polygon File Format (PLY) file (Wang et al. 2023a). The dataset includes eight classes of building elements distributed equally across architectural and MEP families. In total, after removing duplicates, the dataset contains 3,024 instances, as illustrated in Fig. 4.

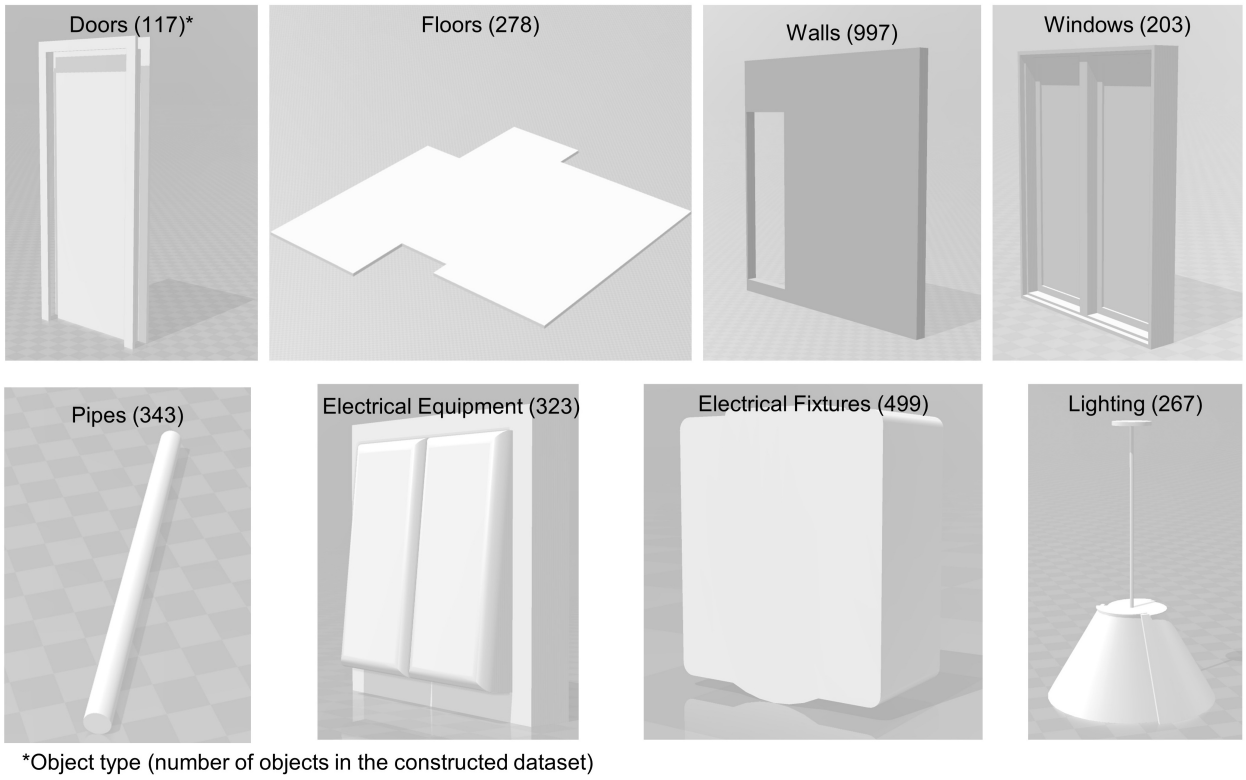

Fig. 4. Examples of BIM objects in the constructed dataset.

In this experiment, we extracted 19 features from the solid objects based on their dimensions, location, and mesh characteristics for machine learning algorithms, as summarized in TABLE 1. To extract mesh features, we considered the properties of vertices, edges, faces, and facets. For location features, we normalized the centroid of each object to be the origin. Bold features in TABLE 1 are used by the decision tree in TABLE 2.

Feature engineering is applied to each object geometry to generate object features and labels in

TABLE 1. Designed features for machine learning algorithms.

No.	Feature	Category
1	Area	Dimension
2	**Volume**	
3	Delta x of bounding box	
4	Delta y of bounding box	
5	**Delta z of bounding box**	
6	**Num of faces**	Mesh
7	**Max area of one face**	
8	**Face ratio (max/min)**	
9	**Num of adjacent faces**	
10	**Num of facet (faces forms facets)**	
11	**Num of unique edges**	
12	Length of unique edges	
13	**Num of vertices**	
14	X_max	Location
15	X_min	
16	Y_max	
17	Y_min	
18	Z_max	
19	**Z_min**	

a tabular form. Data are randomly divided into training and validation sets, with 70% of the data for training, 10% for validation, and the remaining 20% for testing. Eight machine learning algorithms were employed, and their respective performance levels are summarized in TABLE 2. The results show that the best-performing algorithms are the decision tree and random forest, achieving 99% accuracy in the test set. We also employed the F1 score as a metric to evaluate the predictive capability across classes. The results demonstrate that both the decision tree and the random forest yield F1 scores of 0.99, which is close to the ideal 1. This reflects that both algorithms have a balanced predictive ability on this dataset.

Moreover, we used the Apt2 data which were not involved in the training and testing process as the blind data to further test the performance of the algorithms. Results show that the decision tree and the random forest can predict all objects correctly in Apt2 with 100% accuracy.

In addition, we exported the decision tree features to gain insight into the algorithm's function.

TABLE 2. Results of object type classification.

No.	Machine learning algorithm	Validation accuracy	Test accuracy	F1
1	Logistic regression	23.92%	24.02%	0.25
2	Support vector machine (SVM)	46.40%	46.85%	0.35
3	K-nearest neighbors (KNN)	94.13%	88.89%	0.89
4	Gaussian naive bayes	83.14%	81.78%	0.80
5	Perceptron	57.79%	58.56%	0.47
6	Linear support vector classification	70.71%	69.77%	0.60
7	**Decision tree**	100.00%	99.80%	0.99
8	**Random forest**	100.00%	99.90%	0.99

The features utilized by the decision tree are marked in bold in TABLE 1. It shows that 10 out of the 19 features were adopted for the classification process. The nine unused features could be utilized by the algorithms in cases where the classification scope needs to be expanded. This leaves a margin that could maintain reasonable performance when meeting more difficult situations. In summary, in this study, both the decision tree and the random forest algorithms proved capable of classifying the required objects.

5.3 Exp 2: Relationship determination

Relationship classes analysis

In order to instantiate relationships among apartment objects, it is essential to identify the potential relationship classes beforehand. Therefore, we conducted an analysis of object-object relationships from four widely-used ontologies in the Linked Building Data (LBD) community: ifcOWL, Building Topology Ontology (BOT), Building Product Ontology (BPO), and Brick. ifcOWL provides a semantic web-compliant representation of the IFC schema (Pauwels and Terkaj 2016). BOT aims to represent the core topological concepts of a building (Rasmussen et al. 2020), while BPO describes assembly structures and component interconnections (Wagner and Rüppel 2019). Lastly, Brick is designed to standardize the representation of asset information in buildings (Balaji et al. 2016).

Fig. 5 presents a comparison between the different ontologies' definitions of five semantic relationships that we consider ubiquitous within a building model. In this study, *Adjacency* is

Relationship	Explanation	Name in the ontology				Adopted in this study
		ifcOWL (Pauwels and Terkaj)	BOT (Rasmussen et al.)	BPO (Wagner and Rüppel)	Brike (Balaji et al.)	
Adjacency	Objects physically touch each other	ifcRelConnectsElement ifcRelConnectsPathElements	-	isConnectTo isConnectFrom isConnectWith	-	cbim: adjacentTo
Host	An object functionally hosts another	ifcRelVoidsElement	hasSubElement*	-	-	cbim:hosting cbim:hostedby
Space contain	Objects belong to a level or space	ifcRelContainedInSpatialStructure	containsElement	-		bot:containsElement
Aggregation	An object is aggregated by components	ifcRelAggregates	hasSubElement*	isPartOf consistOf	hasPart isPartOf	Not involved
Space bouding	Objects bound space	ifcRelSpaceBoundary	adjacentElement	-	-	bot:adjacentElement

A hyphen, -, indicates that similar classes do not exist in the ontology
* bot:hasSubElement has two meanings, either one object hosting another one or the aggregation of component objects

Fig. 5. Relationship classes summarized from ontologies.

defined as a topological fact that signifies the physical touch of two objects, such as a wall that is in contact with another wall. This relationship can be generated by analyzing the geometries of the objects.

In addition to topology, there are also functional relationships that contain semantics, such as *host*. As experts, we understand that a window should fit into a wall opening, in which case the wall becomes the *host* member of the window. However, determining such a relationship using topological rules alone is difficult, as the bounding box of a wall may not physically contain the entire window if the window sill is wider than the thickness of the wall. Another functional relationship is aggregation, which indicates that an object is composed of components. For instance, a facade may aggregate four curtain wall sections.

There are also two types of relationships between physical and virtual objects: *Space contain* and *Space bounding*. The former describes the relationship between a physical object and a level or space object to which it belongs. The latter refers to the relationships between a space and its bounding physical objects.

This study focuses on instantiating four types of relationships: *Adjacency*, *Host*, *Space contain*, and *Space bounding*. As there are no existing ontology classes for representing physical touch among objects without direction, we adopted *CBIM:adjacentTo* from Cloud-based BIM (CBIM) ontology (Ouyang 2023). Similarly, existing classes such as *ifcRelVoidsElement* and *bot:hasSubElement* do not directly convey the hosting functional relationship; therefore, we adopted *cbim:hosting* and *cbim:hostedby* designed with directions (Ouyang 2023). Addition-

ally, the belonging relationships between physical objects and spaces were represented using *bot:containsElement*, and the relationship between bounding objects and spaces was expressed using *bot:adjacentElement*. Although the scenario in this study did not include aggregation, rule-based algorithms can be designed to handle such relationships.

Relationship instantiation is distributed in Exp 2 and Exp 3. We presented a rule-based method to predict the *Adjacency* and *Host* relationships in Algorithm 1 (Section 5.3). Since objects such as levels and spaces do not exist at the beginning, we generated objects along with relationships linked with existing nodes in Exp 3 (Section 5.4).

Relationship instantiation

Algorithm 1 leverages the characteristics of bounding boxes and exact geometries to generate *Adjacency* and *Host* relationships among physical building objects. Two functions are developed to compute specific features inside the algorithm. The first function is TOPOLOGYXYZ, which calculates the topological relationships of two objects when projected onto the X, Y, and Z axes respectively. Its output is a list with three elements, such as ['ContainedIn', 'ContainedIn', 'ContainedIn'], indicating that the projection of Obj_1 is inside the projection of Obj_2 in X, Y, and Z axes. The CBIM ontology defines nine possible topological instances in each axis (refer to Fig. 9 of Sacks et al. (2022)). The second function, BBXOVERLAP, computes the ratio between the overlapped volume of two objects' bounding boxes over the volume of the smaller object's bounding box. This feature could be useful in determining *Host* relationships.

The front part of the algorithm is designed to filter out those object pairs that cannot exhibit *Adjacency* or *Host* relationships. In theory, both adjacent and host relationships require objects to be in touch with each other but not overlapping. Based on this idea, the algorithm first excludes object pairs that are not in contact. To do this, it computes the clear space distances between the bounding boxes of each pair of objects in the X, Y, and Z directions and sums the values. If the sum d_{bbx} is greater than zero, the two objects cannot be in contact (see Lines 15 to 17 in Algorithm 1). The goal of this step is to filter out impossible object pairs to speed up the process, thus efficiency is a key factor. We use bounding boxes rather than mesh geometry, as bounding boxes contain fewer

Algorithm 1 *Adjacency* and *Host* relationship generation

Input: Object geometry list: L_{geo};
Predefined geometry volume overlapping tolerance ϵ_{vol};
Predefined geometry distance tolerance ϵ_{dis};
Predefined bounding box overlapping tolerance ϵ_{bbx}
Output: Object relationship list L_{rel}

function TopologyXYZ(Obj_1, Obj_2)
    # Project objects into one axis, calculate the topology
    for *axis* in [x, y, z] **do**
        $rel \leftarrow$ TopologyInAxis(Obj_1, Obj_2, *axis*) *#rel would be "Contain", "ContainedIn", "ExactOverlap", etc.*
        Rel_{xyz} append rel
    end for
    return Rel_{xyz}
end function
function BBXOverlap(Obj_1, Obj_2)
    $V_{overlap} \leftarrow$ OverlapBBXVol(Obj_1, Obj_2)
    $V_s \leftarrow$ SmallerBBXVol(Obj_1, Obj_2)
    $R_{overlap} \leftarrow V_{overlap}$ / V_s
    return $R_{overlap}$
end function
for Obj_1 and Obj_2 in L_{geo}, Obj_1! = Obj_2 **do**
    $d_{bbx} \leftarrow | \Delta X | + | \Delta Y | + | \Delta Z |$
    if d_{bbx} == 0 **then** *# Use bounding box to speed up*
        $d_{geo} \leftarrow$ GeometryDistance(Obj_1, Obj_2)
        if $d_{geo} \leq \epsilon_{dis}$ **then**
            $V_{overlap} \leftarrow$ GeometryOverlapVol(Obj_1, Obj_2)
            if $V_{overlap} \leq \epsilon_{vol}$ **then**
                $Rel_{xyz} \leftarrow$ TOPOLOGYXYZ(Obj_1, Obj_2)
                $R_{overlap} \leftarrow$ BBXOVERLAP(Obj_1, Obj_2)
                if Rel_{xyz} == ['Cn'*, 'Cn', 'Cn'] **then**
                    $res \leftarrow$ [Obj_1, "hosting", Obj_2]
                else if Rel_{xyz} == ['CdIn'*, 'CdIn', 'CdIn'] **then**
                    $res \leftarrow$ [Obj_1, "hosted", Obj_2]
                else if Rel_{xyz} == ['Cn', 'Cn', 'CdIn'] and $R_{overlap} \geq \epsilon_{bbx}$ **then**
                    $res \leftarrow$ [Obj_1, "hosting", Obj_2]
                else if Rel_{xyz} == ['CdIn', 'CdIn', 'Cn'] and $R_{overlap} \geq \epsilon_{bbx}$ **then**
                    $res \leftarrow$ [Obj_1, "hosted", Obj_2]
                else
                    $res \leftarrow$ [Obj_1, "adjacent", Obj_2]
                end if
                L_{rel} append res
            end if
        end if
    end if
end for
* 'Cn' is 'Contain';'CdIn' means 'ContainedIn'

points for computation. Moreover, calculating the distance in one dimension, $d(p, q) = |p - q|$, is computationally faster than in three dimensions, $d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2}$. Only filtering for objects that may be in contact, the algorithm calculates the exact geometry distance of the objects and proceeds if the exact geometry distance is close to zero (Lines 18 to 19). Next, the algorithm calculates the overlapped volume of the object geometries and continues only if the overlapped volume is close to zero, indicating that the objects are not overlapping (Lines 20 to 21). By applying these checks, the algorithm filters out any irrelevant object pairs, leaving only those that are touching but not overlapping.

The latter part of the algorithm assigns relationships by evaluating the two features, the bounding box topological relationships and the overlapped volume ratio ($R_{overlap}$) (Lines 22 to 23). If all three topological instances are "Contain" or "ContainedIn," "hosting" or "hosted" relationships are directly assigned to the pair of objects (Lines 24 to 27). In situations where there are two "Contain" or "ContainedIn" instances, the algorithm considers the $R_{overlap}$ value to assign "hosting" or "hosted" relationships (Lines 28 to 31). The bounding box overlapping tolerance ϵ_{bbx} is set to 0.7 after calculating pairs of hosting relationship objects. All remaining pairs are assigned the "adjacency" relationship.

Results

We manually labeled the relationships between objects in the design model as the ground truth data to evaluate the predicted relationships quantitatively. Specifically, we examined each object in the apartment model and assigned possible relationships to object pairs, either *Adjacency* or *Host*. Their IDs and the labeled relationship were recorded as the ground truth. The output from Algorithm 1 is a list that contains pairs of object IDs and the predicted relationships.

Correctness criteria were adopted to evaluate the performance of the algorithm. Accuracy, defined as the ratio of correct predictions to total predictions, provides a direct view of the model's ability. However, it does not consider the impact of different classes, necessitating the use of Precision, Recall, and F1 scores for a more comprehensive evaluation. Precision measures a model's capacity to accurately identify relevant objects by calculating the ratio of true positives

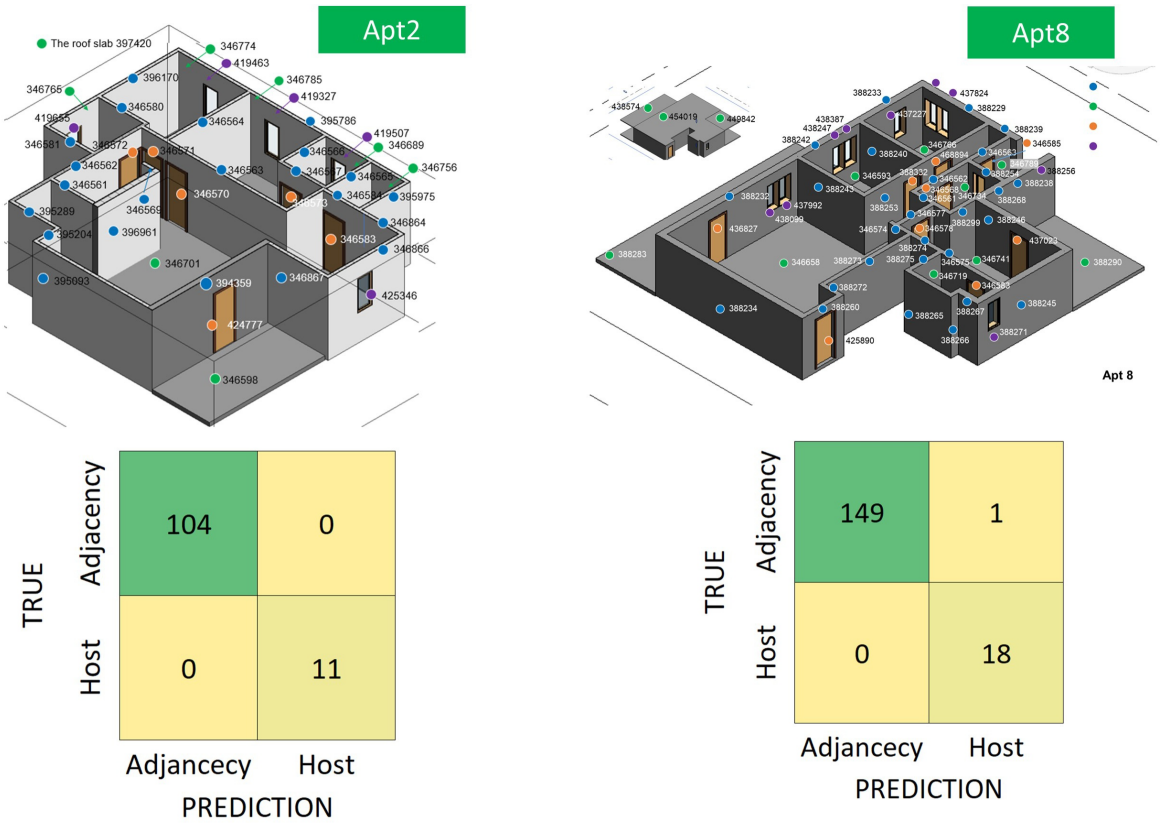

Fig. 6. Results of Algorithm 1 ((a) Results of algorithm development, (b) Results of test).

over all positives. In contrast, Recall is the ratio of true positives over all relevant instances. The F1 score, integrating both Precision and Recall, is regarded as a comprehensive metric that reflects the model's predictive ability across varying classes.

The results of Algorithm 1 are presented in Fig. 6. It shows that all correctness criteria on Apt 2 reach a score of 1, indicating that all predicted relationships are correct and there are no missing or extra predictions. Since Apt2 was used to fine-tune the algorithm throughout its development, the

results may not reflect the actual performance of the algorithm. Therefore, we introduced another model, Apt8, as blind data that were not involved in algorithm development, to test the algorithm. The accuracy for Apt8 reaches 0.99, and its Precision, Recall and F1 scores are equal to or greater than 0.97. The confusion metrics reveal one error, where an *Adjacency* instance is incorrectly labeled as *Host*. In sum, the results from both Apt2 and Apt8 demonstrate that the algorithm has the balanced ability to correctly predict both *Adjacency* and *Host* relationships between building objects.

Moreover, we adopted the processing time as the criterion for evaluating efficiency, which includes the time that the algorithm is executing but excludes the time spent reading and writing data. The processing time in total is around 3.5 minutes for Apt2, which means that it takes 1.8 seconds on average to generate a relationship instance. Similarly, it took around 1.8 seconds to generate one relationship on average in Apt8.

The algorithm has two primary advantages. First, it takes object geometries as the only input, making it generic to various scenarios. Second, it has the ability to predict the *Adjacency* and *Host* relationships correctly. However, researchers need to adjust three predefined tolerance values manually to achieve better performance, and these are a limitation of the procedure.

5.4 Exp 3: Object generation

Exp 3.1: Level, building, site generation

Building spatial structure typically includes sites, buildings, levels (stories), and other objects within those levels. While using a design editor, users are not obligated to create spatial structure objects, as they are often well prepared by the software or included in the templates. However, for generic semantic enrichment, the input is physical object geometries, making it necessary to recover the essential building spatial structure.

In this study, we devised a set of rules for recovering spatial structure objects from levels to the building and to the site. First, we merged floors with the same maximum value in the Z-axis (Z_{max}), and recomputed their areas. Then, we grouped floors by their Z_{max} values with a height threshold set at 1.5 meters. It indicates that whenever the distance of any floors' Z_{max} values is larger than the

threshold, the two floors were separated into different groups. Also, each group represents a level containing the clustered floors. We created an equivalent number of level objects to the number of groups. Furthermore, within each group, we assigned the Z_{max} value from the largest floor as the elevation of that level. Physical and virtual objects on the level are connected to the *level* object (Ying et al. 2019), through the 'bot:containsElement' relationship.

Note that we made the assumption that there is only one building on the site, which allowed us to generate a building object and a site object directly. All levels were then linked to the *building*, and the *building* was connected to the *site* with the 'bot:containsElement' relationship.

Exp 3.2: Space generation and classification

Spaces play a crucial role in building semantics used for simulation, quantity take-off, and so on (Ying and Lee 2021). Typically, users manually create and label spaces. Certain software, like Revit, provides a function to automatically separate a space; however, it still requires users to label or input the space type manually. Previous studies explored space type prediction by using rule-based and deep learning techniques (Bloch and Sacks 2018; Wang et al. 2022b), with the prerequisite that spaces had been created.

We designed a two-step process to 1) create the missing space objects and 2) classify the space types as shown in Fig. 7. A rule-based algorithm is constructed to leverage floors and walls to generate space objects. A GNN model follows to predict the space types. Space nodes are generated in the enriched graph, and space geometries are created and stored as PLY files. The predicted space types are inserted into the space nodes as attributes.

In Algorithm 2, we developed a space generation method by leveraging the fact that internal rooms within a building should be closed by walls and floors. First, we identify walls within the same level. The walls are then joined and projected onto the XY plane, forming a 2D contour map of all walls within the storey. Following that, the algorithm identifies every closed polygon within the contour map by searching for sets of connected paths forming closed loops that indicate the footprints of room spaces. Walls that enclose the footprints are recorded as the bounding objects to the spaces. Next, we generate a temporary space geometry by extruding the footprint polygons

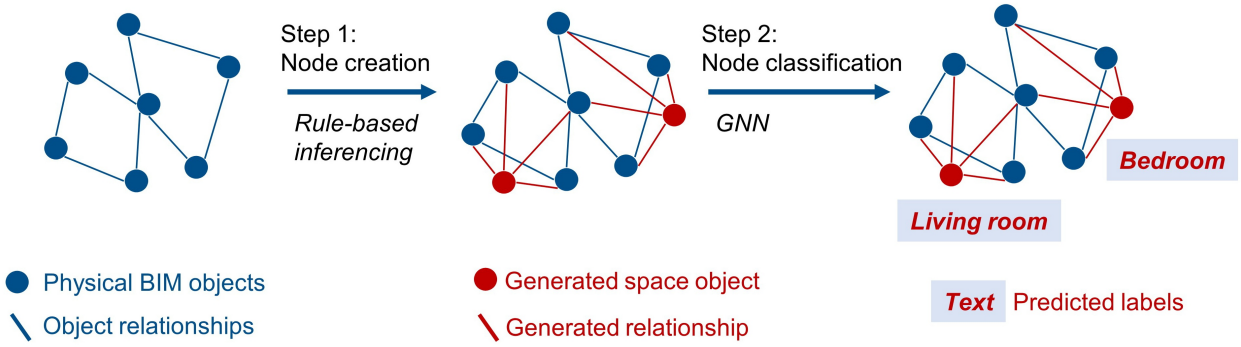

Fig. 7. Pipeline of space object generation and space type classification.

Algorithm 2 Space generation

Input: Object geometry list: L_{geo} ;
Object types and attribute table: D;
Predefined geometry distance tolerance: ϵ_d
Output: Space object geometry list: L_{space};
Object relationship list L_{rel}

for walls on the same level **do**:
  $G_{walls} \leftarrow$ Concatenate(L_{obj}, walls) *# The union of walls at the same level*
  $P_{wall} \leftarrow$ Project(G_{walls}, XY) *# Project into the XY plane*
  $L_{close} \leftarrow$ PolygonsClosed(P_{wall}) *# Find closed polygons and save in a list*
  for p_{close} in L_{close} **do**
    $L_{boundwall} \leftarrow$ ConsistObject(p_{close}) *# Return walls that consist the closed polygon*
    $h_{temp} \leftarrow FrequentHeight(L_{boundwall})$
    $G_{temp} \leftarrow$ CreateSpaceFromPolygon(p_{close}, h_{temp}) *# Create a temporary space by using an estimated height for finding out bounding floors*
    for G_{floor} in L_{geo} **do**
      $d \leftarrow$ GeometryDistance(G_{temp}, G_{floor})
      if $d < \epsilon_{dis}$ **then** *# Find bounding floors*
        $L_{boundfloor}$ append $floor$
      end if
    end for
    $L_{boud} \leftarrow L_{boundwall} + L_{boundfloor}$
    $G_{space} \leftarrow$ CreateSpaceFromBoundGeometry(L_{boud}) *# Concatenate bounding walls and floors to form a closed internal space as the space object*
    L_{space} append G_{space}
    L_{rel} append SpaceRel(G_{space}, L_{boud})
  end for
end for

in the Z direction of increasing elevation with a height that most walls share. This temporary geometry is used to identify the space bounding floors by calculating the geometry distance to find the nearest lower and upper floor objects. After identifying the bounding walls and floors, we combine all bounding objects to create an internal room as the generated space. We also save its exact geometry and register the relationships between each space and all its neighboring objects.

In the second step, we predict the types of spaces generated by a GNN-based room type classification method (Wang et al. 2022b). To accomplish this, we extract all space nodes from the enriched graph to form a property graph for GNN. Each node in the property graph represents an unclassified space object, while the edges in the graph represent the various relationships between the spaces. We apply feature engineering to compute the characteristics from relationships, which are then turned into node and edge feature metrics. Next, we feed the property graph into a customized GNN model, SAGE-E (Wang et al. 2022b), for node type classification. The predicted results are space types such as kitchens, bedrooms, balconies, toilets and so on. These labels are inserted into the graph as space node attributes.

The results indicate that all desired space objects in an apartment unit were successfully created in step 1, and the GNN algorithm managed to classify five out of seven spaces correctly in step 2. It is worth noting that three relationships are required in the GNN model to compute features during the process of property graph generation: wall connection, door connection, and virtual wall connection. While wall and door connections can be determined automatically by examining existing objects, virtual wall connections still require human labeling, making this step semi-automated in this study.

5.5 Exp 4: Attribute computation

We identified three main categories of object attributes: location-related, dimension-related, and semantic-related. The Location attributes of an object describe its position information, such as its height offset from a specific level. The buildingSMART Data Dictionary (bSDD) for defining BIM objects and their attributes rarely includes location-related attributes for common physical building objects (buildingSMART 2022). The IFCtoLBD converter also abandons all

location-related attributes during compilation (Bonduel et al. 2018). This is because that location-related attributes can be evaluated from exact geometry and are not frequently used. Thus, we recommend retrieving object geometries and computing location-related attributes when specific location information is needed.

The dimension attributes describe the size and shape of an object. In this study, we developed a tool to compute dimension-related attributes, including height, area, volume, width, thickness, perimeter, length, slope, and width. All of the listed attributes can be measured from the exact geometry and assigned with object types.

Semantic attributes relate to the object's function or purpose. The object type from Exp 1 is a kind of semantic attribute. In additional, hosting and adjacent attributes from Exp 2, and space bounding from Exp 3 are also semantic attributes but they are represented as relationships in the graph rather than node attributes.

5.6 Enrichment results

The output of each enrichment tool was organized into a graph-based CDE, and the enriched graph is presented in Fig. 8, where nodes represent building objects named after their types and unique IDs, and edges represent the relationships with directions. Attributes, including computed features and geometry file paths, were attached to corresponding nodes but were hidden in the GraphDB visualization for better readability.

The enriched graph provides a representation of the building's spatial structure, from the site to the objects contained in each storey, as illustrated in Part 1 of Fig. 8. It also explicitly shows the inferred relationships between building objects. For example, Part 2 depicts the bidirectional *adjacentTo* relationships among two walls and the directed *hosting/hosted* relationships between a door and a wall. Furthermore, space objects are connected to their bounding walls and floors with the *adjacentElement* relationship, and linked to their corresponding external geometry files as displayed in Part 3. Part 4 depicts the attributes attached to an arbitrary wall node, which include dimension-related properties as well as a retrievable file directory from which users can access its exact geometry conveniently.

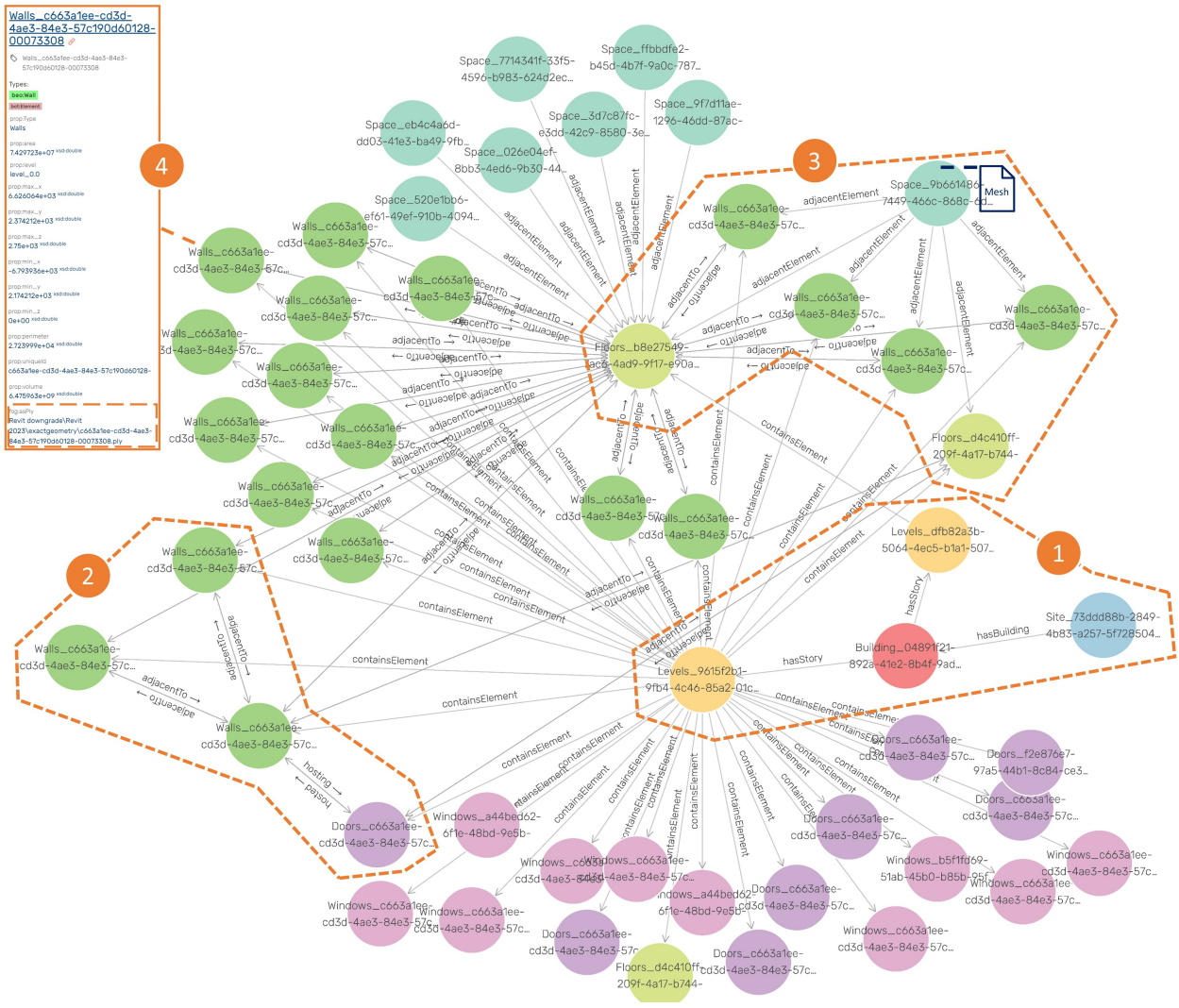

Fig. 8. Enriched graph of Apt2. (1. Building spatial structure: site, building, story. 2. Generated *adjacentTo* and *hosting/hosted* relationships. 3. A generated *Space* node and relationships linked to its bounding objects. 4. Part of the attributes of a wall and its geometry file address. For readability, some relationships are hidden.)

5.7 Evaluation and comparison

This study initially evaluates the number of nodes and edges within a graph (TABLE 3). Additionally, we provide a more detailed content comparison for the selected graphs, specifically focusing on the presence of certain types of data (Fig. 9).

Three Apt2 graphs were obtained for comparison. The first graph was generated by the generic semantic enrichment approach presented in this study. The second graph was constructed using

the CBIM-Revit parser, which retrieves all accessible information from the Revit database using Dynamo APIs and organizes it into an LBD graph CDE (Wang et al. 2023a). It keeps all accessible raw information from the Revit environment directly, which can partially showcase the data richness in the design software. The third graph was created by the IFCtoLBD parser after exporting the Apt2 model from Revit as an IFC file.

We first compared the data quantity to understand the richness of data in each graph as listed in TABLE 3. The first three criteria evaluate RDF graph features using the number of triples, nodes, and edges. The results show that the enriched graph has higher values for the three criteria than the other two graphs. Since many of the object attributes are ignored during the compilation process, the IFCtoLBD graph contains fewer attributes. Additionally, the exported data from Revit contain little edge-related information.

The last three criteria are related to property graphs which are computed by converting the RDF graphs into property graphs. A dense graph in mathematics means the number of edges is close to the maximal edge numbers, in which case each node is linked to all other nodes. At the opposite extreme, a graph with few edges has a density close to zero and is referred to as a sparse graph. Here, the graph density of the enriched graph is three times higher than the other two, indicating that the nodes in the enriched graph are more tightly connected. The enriched graph also demonstrates a higher value regarding the average node degree, indicating that the nodes it contains possess more links on average compared to nodes from the other two graphs.

TABLE 3. Data quantity comparison among BIM graphs.

Criterion	Enriched graph	Revit-Dynamo graph	IFCtoLBD graph	Note
Num of triples	**1369**	1173	536	The IFCtoLBD graph has fewer attributes
Num of nodes	**51**	44	45	The IFCtoLBD graph has one more Building node than the Revit-Dynamo graph
Num of edges	**417**	52	101	Original Revit exported data contain few relationships
Graph density	**0.1631**	0.05759	0.0556	Value 0 represents a graph without edges and 1 a complete graph
Average node degree	**8.1569**	2.4186	2.4444	
Avg. node attributes	21	**25**	11	The Revit-Dynamo graph contains full attributes, most of which are location related.

With regards to object attributes, most are omitted by the IFCtoLBD graph which excludes area, host ID and so on. Most of the listed attributes exist in the exported IFC file while they are lost during the conversion to the IFCtoLBD graph. This might be due to the incomplete development

			Enriched graph	Revit-Dynamo graph	IFCtoLBD graph
1. Object	Physical	Wall	22	22	22
		Floor	8	8	8
		Door	6	6	6
		Window	5	5	5
	Virtual	Site	1	1	1
		Building	1	×	1
		Level	2	2	2
		Space	6	△	△
2. Relationship	Essential	Type	92	×	46
		BOT:hasbuilding	1	×	1
		BOT:hasStory	2	×	2
		BOT:containsElement	41	41*	41
	Semantic	CBIM:adjacentTo	208	×	×
		CBIM:hosting	11	11*	11**
		CBIM:hostedby	11	11*	11**
		BOT:adjacentElement	51	×	×
3. Attribute	Dimension	Area	✓	✓	×
		Volume	✓	✓	×
		Height	✓	✓	✓
		Width	✓	✓	✓
		Perimeter	✓	✓	×
		Thickness	✓	✓	×
		Slope	✓	✓	×
		Length	✓	✓	×
	Other	Category	✓	✓	✓
		Type	✓	✓	✓
		Host ID	✓	✓	×
		Structural usage	×	✓	×
		Room bounding***	✓	×	×
		Base is attached	✓	✓	×
		Top is attached	✓	✓	×
4. Object geometry			Link to geometry files	Link to geometry files	None

* In Revit graphs, "hosting/hosted" and "containsElement" relationships are referred by attributes.
** IFCtoLBD graphs use "bot:hasSubElement" instead of "cbim:hosting/hosted".
*** Room bounding here refers to the objects that bound a space.
Legend: Values are the number of this data. ✓ means the existence of this data. × stands for the lack of this data.
△ means this information does not exist by default, but it presents in the graph if users manually generate it in BIM editors.

Fig. 9. Data content comparison among BIM graphs.

of the converter (the authors planned to explain the attribute mapping process in a future study – Bonduel et al. (2018)). Lastly, object geometries are preserved in both the enriched and Revit graphs where their file paths are stored as object attributes. However, the geometry information is omitted deliberately during the compilation process of the IFCtoLBD graph to reduce graph sizes (Pauwels and Roxin 2017; Bonduel et al. 2018).

From the data content perspective, the enriched graph stands out for having the most listed data types, encompassing both physical and virtual objects, relationships, and attributes. The Revit-Dynamo graph contains rich attributes but lacks meaningful object relationships. The IFCtoLBD graph, while it preserves the complete building spatial structure objects, falls short in terms of preserving attributes and geometries. Furthermore, both the Revit and IFCtoLBD graphs lack semantics, like the space nodes, and the adjacent/bounding relationships, as these data are inferred by specific enrichment tools rather than generated by design software.

6 VALIDATION EXAMPLES

This study explores two applications using semantic enrichment. The first is generating BIM models from pure geometry models (Section 6.1), and the second addresses interoperability issues across BIM platforms and software (Section 6.2). These two examples focus on a single discipline, architectural design. Sharing and coordinating models across disciplines involves tackling not only semantic and interoperability issues but also coordination challenges, which are discussed in Section 7.2.

6.1 Example 1: Enriching pure geometry to BIM models

In the initial design phase, architects often complete the conceptual design in 3D geometry design software like SketchUp. These software tools provide a flexible environment for creative work but typically contain only geometry information. As the project progresses, participants require more design semantics instead of just pure geometries. Consequently, in practice, participants must manually reconstruct a BIM model based on the pure geometries in a BIM modeling software. This process can be time-consuming and cumbersome.

We propose a SE approach to reconstruct a BIM model using geometries exported from 3D design software, as illustrated in Fig. 10. In theory, this approach is applicable to any geometry design software, as long as it supports the export of object geometries, and to any BIM software that provides APIs for creating BIM objects. For demonstration purposes, we used SketchUp 2023 for geometry design and Revit 2022 as the BIM software.

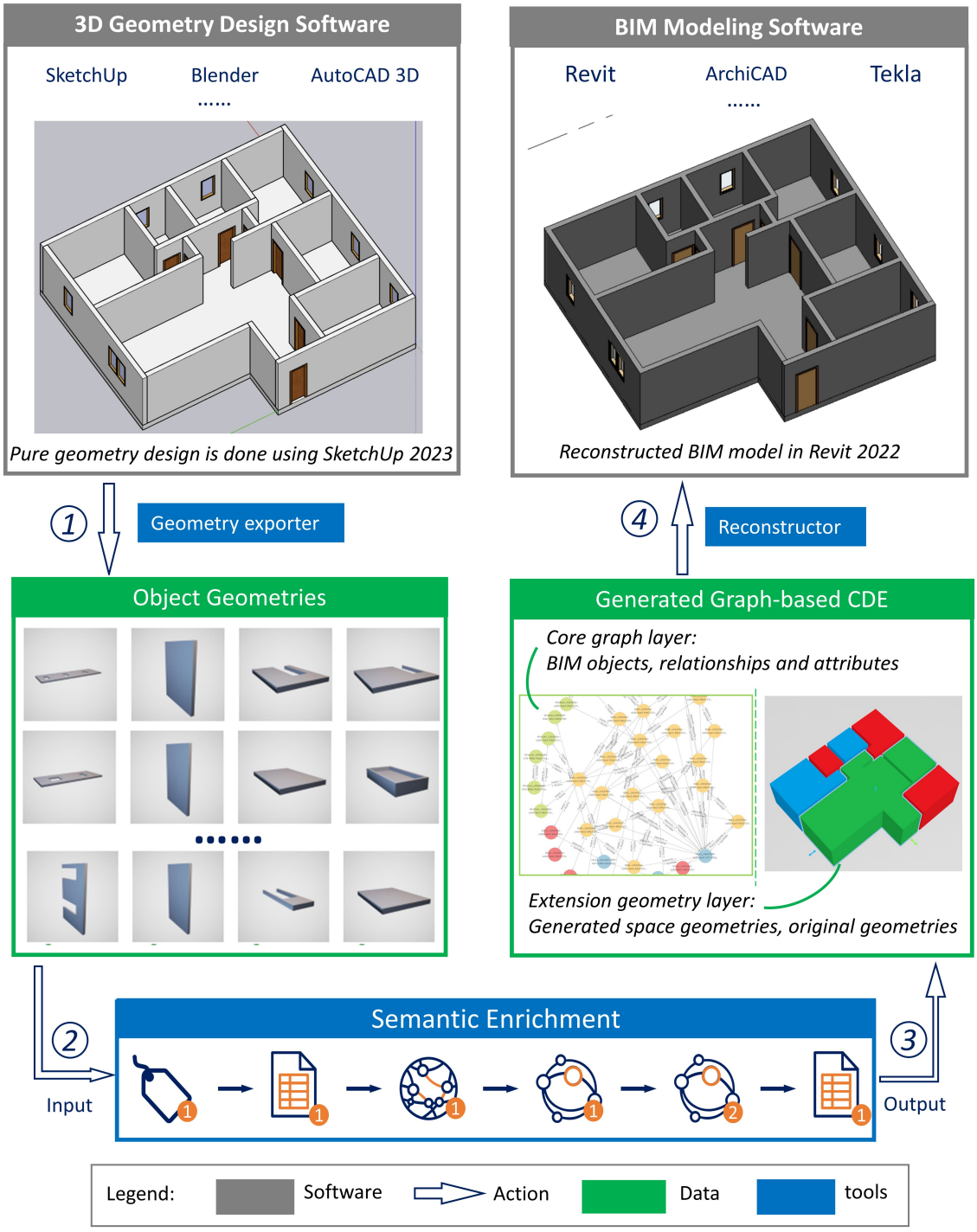

Fig. 10. Process of enriching pure geometry into BIM models.

An apartment model was designed in SketchUp 2023. Using the default export function, every solid object was exported as a geometry file. These object geometries were then fed into the SE process. We adopted all developed SE tools to enrich possible building information and organized the results into a graph-based CDE, as shown in the right green box of Fig. 10. The database consisted of two parts: a graph layer for representing building objects, relationships, and attributes, and an extension layer for storing geometries independently.

Following this, a Reconstructor was used to rebuild BIM objects in Revit 2022 using the enriched semantics. The Reconstructor, developed with Dynamo and Python, retrieves the required object information from the graph-based CDE and inputs it into Dynamo functions for generating objects (Wang et al. 2023b). For example, creating a window using Dynamo requires the hosted object, the location, and the window type. The Reconstructor searches for the needed information from the CDE to satisfy the function for generating a new window object inside Revit 2022. During this reconstruction process, all types of generated semantics, including object types, relationships, and attributes, are utilized.

Compared to the original design, the BIM model in Revit 2022 contains not only geometries but also object semantics. Furthermore, the reconstructed model includes additional virtual objects generated by SE tools.

6.2 Example 2: Facilitating interoperability

Designers often encounter situations where they are unable to access models from different software applications. Even within the same vendor environment, there can be a chance that an earlier version of the software cannot read models created by a newer version of the same tool. A novel approach proposed in this study is to utilize the generated semantics and software APIs to address interoperability issues, as illustrated in Fig. 11.

The proposed approach consists of three steps. First, object geometries are exported from one BIM software. Next, semantic enrichment tools are executed sequentially to enrich the models, which are organized as a graph-based CDE. Third, a Reconstructor program runs to query essential information from the graph-based CDE and reconstruct models in another BIM software.

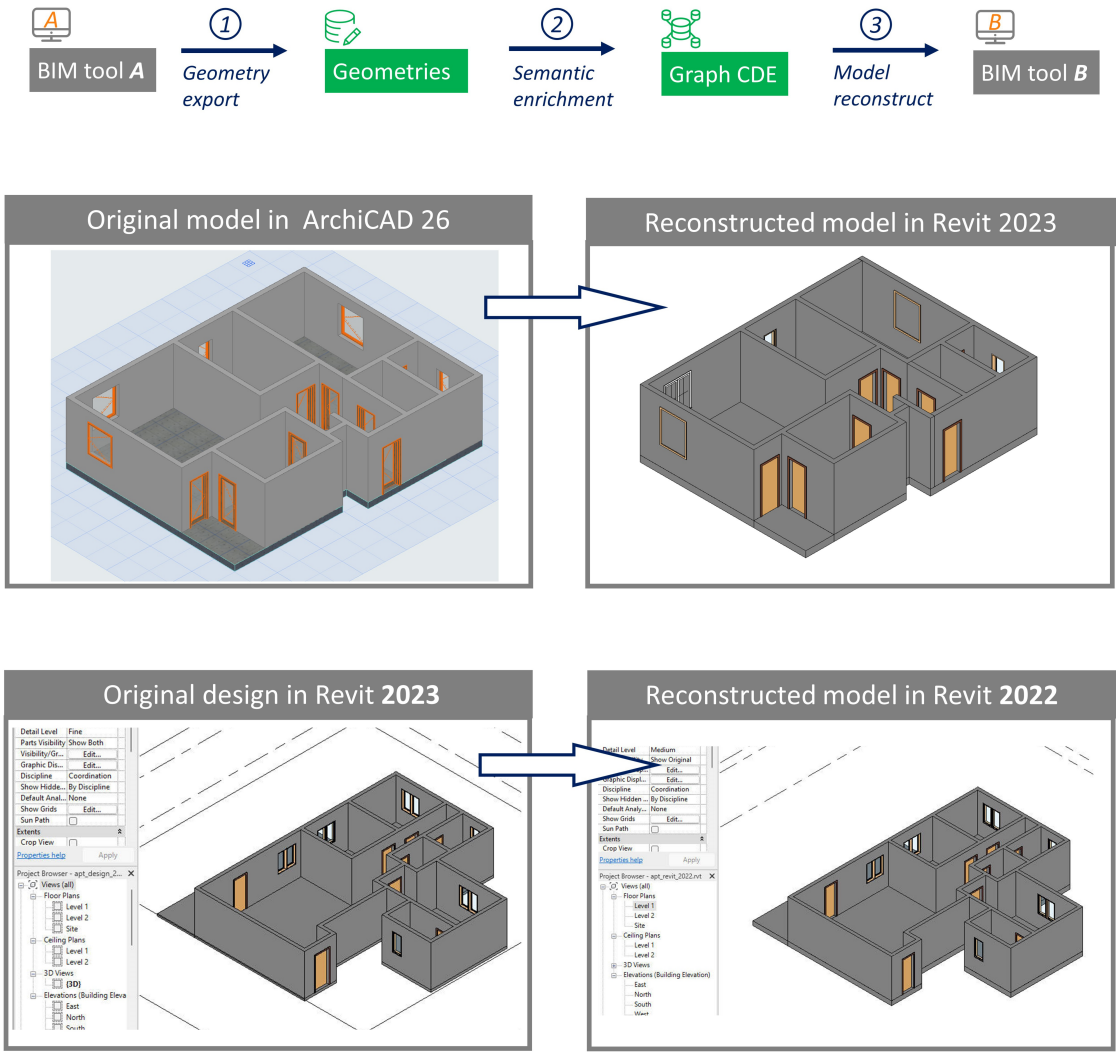

Fig. 11. A novel approach for facilitating interoperability issues by semantic enrichment.

To test this approach, we selected an apartment unit from an example project released by Graphisoft (2023). It is a typical two-bedroom apartment designed using ArchiCAD 26 software. Object geometries were exported from ArchiCAD 26 and fed into the constructed SE tools. The Dynamo Reconstructor used the necessary information to rebuild a model in Revit 2023, as illustrated in Fig. 11.

Similarly, we tested the approach for downgrading Revit models to earlier versions. An architectural apartment model was designed using Revit 2023, and its object geometries were exported as PLY files. Through the enrichment process, a model was reconstructed in an earlier-version,

Revit 2022.

TABLE 4. Object quantity comparison between the original model in Revit 2023 and the reconstructed model in Revit 2022.

Object type	Object quantity	
	Revit 2023	Revit 2022
Site	1	1
Building	1	1
Story	2	2
Door	8	8
Slab	3	3
Wall	19	19
Window	10	10
Space	0	8

To evaluate the results, we exported the object quantities from the two Revit models and compared them in TABLE 4. The table shows that the reconstructed model in Revit 2022 has the same number for each object type, except that it has eight additional space objects. Furthermore, each object pair in the two Revit models was compared manually, examining their locations, dimensions, and object types. Our findings revealed that the reconstructed model aligns accurately with the original design. It is important to note that our approach generates new objects in another software instead of merely duplicating information. As a result, the unique or global IDs from the two models do not match.

Compared to the file-based approach, the SE approach for interoperability offers more comprehensive semantics. For example, 3D space geometries, which are difficult to generate using existing BIM software, can be automatically constructed by SE tools, facilitating downstream simulations. Furthermore, the SE approach shifts data storage from a file-based level to an object-based level, enabling more advanced applications such as version control (Esser et al. 2022).

7 DISCUSSION

7.1 Strengths and weaknesses

This study exhibits several strengths in the proposed concept, tools, and applications. First, the framework defines four graph semantic enrichment tasks and a process control mechanism for operating the tools developed for those tasks in a flexible flow sequence.

The second strength lies in the five practical tools that were developed. They achieve the goals of classifying apartment objects, determining host and adjacent relationships, generating implicit objects and computing object attributes with efficient and accurate performance.

Third, the application on the generated graph-based CDE showcases an innovative approach to enrich designs modeled with geometry alone, and ease the interoperability problem. The approach does not rely on vendor proprietary schemas and it requires only object geometries, which can be exported from design software with accessible formats.

However, this study also faces certain limitations with respect to its tools and application scenarios. First, the developed rule-based algorithms incorporate several threshold values that need to be tuned by users to optimize performance. Second, the implemented tools have been designed specifically for apartment scenarios. This requires efforts from the community and commercial companies to develop practical tools for wider use case scenarios.

Moreover, this experiment is designed for LOD 200 models. When moving to a higher LOD model, the situation could be more complex and it requires further research. There should be a more detailed representation of BIM objects. For instance, layers within walls should be defined when rising from LOD 200 to LOD 300. This might require the introduction of wall layer nodes in the graph that aggregate into a wall node. Furthermore, specific attributes need to be inferred, such as the material and the thermal transmittance. This poses another feasibility problem, i.e., whether we can use object geometries alone to infer semantic attributes like materials. To the best of our knowledge, it is possible to adopt machine learning to predict materials, as there are potential patterns (e.g., beam cross-section shapes) that can be used as features for learning.

7.2 Across-domain coordination

The exchange of models across disciplines presents a more complex challenge than a typical interoperability problem. Due to the federated nature of the industry, each discipline holds its own specific model and does not take on design models from other disciplines directly. Engineers, for example, prefer to reconstruct structural models based on architectural designs. Successful collaboration across domains should deliver data accurately and interoperably and coordinate objects that represent different meanings based on disciplines. While the successful application of generic semantic enrichment can address the first requirement of data accuracy and transparency, there is still a need for a coordination method.

The Cloud-based BIM (CBIM) concept aims to link multidisciplinary BIM graphs with spatial, functional, and constraint relationships and implement intelligent applications on the linked CBIM meta graph to enhance across-domain design coordination (Sacks et al. 2022). One successful application of the CBIM concept is enabling change propagation across architectural and structural models. This involves placing designs in one architectural software and one structural editor independently and connecting them to a cloud server. Whenever a model is modified, the cloud server can analyze the change and propagate either geometry reference or correction commands to the affected discipline to assist across-domain design coordination (Wang et al. 2022a). Generally, CBIM provides an object-level solution to enhance collaboration across disciplines.

One possible direction to solve the across-disciplinary coordination problem is to combine the synergy of generic semantic enrichment and the CBIM concept. Generic semantic enrichment offers an interoperable, accurate, and rich graph-based data repository that uses only object geometries regardless of disciplines and software schemas. CBIM can then be implemented to instantiate relationships to link these domain-specific graphs together, forming a linked meta graph. Coordination applications can be placed on top of the linked graphs to achieve information exchange and change propagation. Together, they construct a smooth and close collaboration future.

8 CONCLUSIONS

This study explores inferring BIM semantics from object geometries and compiling the enriched

data to BIM graphs for simulations and analyses. The main findings and contributions are as follows:

Accomplished four SE tasks by developing tools and datasets. We developed five tools to achieve four types of SE tasks. For relationship determination, we proposed a novel geometry computation algorithm to instantiate *host* and *adjacency* relationships. For object generation, we devised two rule-based methods to generate spatial structure objects and space objects. In terms of object classification, we constructed a dataset and eight machine learning algorithms for common apartment object classification. Finally, we developed a method for attribute computation.

A framework of generic semantic enrichment. We defined four basic types of semantic enrichment tasks within the BIM graph context and proposed the process control mechanism for executing a set tools in a proper sequence.

Intelligent applications using generic semantic enrichment and graph representation of BIM. The first application involves enriching pure object geometries from SketchUp into a BIM model in Revit. The second application addresses the interoperability problem by bridging different BIM platforms and software, such as reconstructing ArchiCAD models in Revit.

Moreover, we evaluated BIM graphs from the perspectives of data feasibility and quantity to measure the data richness of BIM graphs. We also discussed the synergy of generic semantic enrichment and the CBIM concept for enabling a wider range of across-domain intelligent applications.

In response to the presented research questions, we have demonstrated the feasibility of reconstructing a graph-based CDE from pure object geometries to enable intelligent applications, in the context of residential apartments at LOD 200. Extending to other scenarios may involve a similar process and requires further testing and validation. Moreover, this study establishes the theory basis for graph semantic enrichment, and provides another direction for solving BIM problems at the object level. We believe that this study can inspire readers to develop diverse semantic enrichment tools and deploy more smart applications on the BIM graph.

9 DATA AVAILABILITY STATEMENT

Some data related to Exp 1 Object type classification are available from the corresponding

author upon reasonable request.

10 ACKNOWLEDGMENTS

The work is part of the Cloud-based Building Information Modelling (CBIM) project, a European Training Network. The CBIM project receives funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant with agreement No 860555.

REFERENCES

Autodesk Inc. (2022). "BIM Coordination & Collaboration | Autodesk BIM Collaborate, <https://construction.autodesk.com/products/autodesk-bim-collaborate>.

Balaji, B., Bhattacharya, A., Fierro, G., Gao, J., Gluck, J., Hong, D., Johansen, A., Koh, J., Ploennigs, J., Agarwal, Y., Berges, M., Culler, D., Gupta, R., Kjærgaard, M. B., Srivastava, M., and Whitehouse, K. (2016). "Brick." *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, New York, NY, USA, ACM, 41–50, <https://dl.acm.org/doi/10.1145/2993422.2993577> (nov).

Beetz, J., van Berlo, L., de Laat, R., and van den Helm, P. (2010). "Bimserver. org–an open source ifc model server." *Proceedings of the CIP W78 conference*, 8.

Belsky, M., Sacks, R., and Brilakis, I. (2016). "Semantic Enrichment for Building Information Modeling." *Computer-Aided Civil and Infrastructure Engineering*, 31(4), 261–274.

Bloch, T. (2022). "Connecting research on semantic enrichment of BIM-review of approaches, methods and possible applications." *Journal of Information Technology in Construction (ITcon)*, 27.

Bloch, T. and Sacks, R. (2018). "Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models." *Automation in Construction*, 91(July 2017), 256–272.

Bloch, T. and Sacks, R. (2020). "Clustering Information Types for Semantic Enrichment of Building Information Models to Support Automated Code Compliance Checking." *Journal of Computing in Civil Engineering*, 34(6), 04020040.

Boettiger, C. (2018). *rdflib: A high level wrapper around the redland package for common rdf applications*, <https://doi.org/10.5281/zenodo.1098478>.

Bonduel, M., Oraskari, J., Pauwels, P., Vergauwen, M., and Klein, R. (2018). "The ifc to linked building data converter-current status." *Proceedings of the 6th Linked Data in Architecture and Construction Workshop (LDAC 2018)*, Vol. 2159, CEUR Workshop Proceedings, 34–43.

Borrmann, A., König, M., Koch, C., and Beetz, J. (2018). *Building information modeling: Why? what? how?* Springer.

BuildingSmart (2019). "IFC Schema, <https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/> (apr).

buildingSMART (2022). "Buildingsmart data dictionary." *buildingSMART Data Dictionary*, <https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/> (Aug).

Chryssolouris, G., Domroese, M., and Beaulieu, P. (1992). "Sensor Synthesis for Control of Manufacturing Processes." *Journal of Engineering for Industry*, 114(2), 158–174.

Collins, F., Braun, A., Ringsquandl, M., Hall, D., and Borrmann, A. (2021). "Assessing ifc classes with means of geometric deep learning on different graph encodings." *Proc. of the 2021 European Conference on Computing in Construction*.

Dawson-Haggerty et al. (2019). "trimesh, <https://trimsh.org/>.

Emunds, C., Pauen, N., Richter, V., Frisch, J., and van Treeck, C. (2022). "SpaRSE-BIM: Classification of IFC-based geometry via sparse convolutional neural networks." *Advanced Engineering Informatics*, 53, 101641.

Esser, S., Vilgertshofer, S., and Borrmann, A. (2022). "Graph-based version control for asynchronous BIM collaboration." *Advanced Engineering Informatics*, 53, 101664.

Graphisoft (2023). "Archicad sample projects, <https://community.graphisoft.com/t5/Let-s-get-started/Archicad-Sample-Projects/ta-p/304186>.

Güting, R. H. (1994). "Graphdb: Modeling and querying graphs in databases." *VLDB*, Vol. 94, Citeseer, 12–15.

Hietanen, J. (2002). "BLIS review: IMSvr, <http://ve.cic.vtt.fi/IMSvr/IMSvr.wsdl>.

ISO 19650 (2018). *Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling* , <https://www.iso.org/standard/68078.html>.

Kim, I., Kim, J., and Seo, J. (2012). "Development of an ifc-based idf converter for supporting energy performance assessment in the early design phase." *Journal of Asian Architecture and Building Engineering*, 11(2), 313–320.

Koo, B., Jung, R., Yu, Y., and Kim, I. (2021). "A geometric deep learning approach for checking element-to-entity mappings in infrastructure building information models." *Journal of Computational Design and Engineering*, 8(1), 239–250.

Koo, B., La, S., Cho, N. W., and Yu, Y. (2019). "Using support vector machines to classify building elements for checking the semantic integrity of building information models." *Automation in Construction*, 98, 183–194.

Mafipour, M., Vilgertshofer, S., and Borrmann, A. (2022). "Digital twinning of bridges from point cloud data by deep learning and parametric models." *ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022*, CRC Press, 543–550.

Ouyang, B. (2023). "Foundations of Collaborative Multi-discipline Cloud Building Information Modeling." M.S. thesis, Technion Israel Institute of Technology, Technion City, Haifa, Israel.

Ouyang, B., Wang, Z., and Sacks, R. (2022). "Semantic Enrichment of Object Associations Across Federated BIM Semantic Graphs in a Common Data Environment." *EUROPEAN CONFERENCE ON PRODUCT AND PROCESS MODELING 2022*, TRONDHEIM, NORWAY, 1–8.

Pauwels, P., Costin, A., and Rasmussen, M. H. (2022). "Knowledge Graphs and Linked Data for the Built Environment." Springer, Cham, 157–183.

Pauwels, P., De Meyer, R., and Van Campenhout, J. (2011). "Interoperability for the design and construction industry through semantic web technology." *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6725 LNCS, 143–158.

Pauwels, P. and Roxin, A. (2017). "Simplebim: From full ifcowl graphs to simplified building

graphs." *eWork and eBusiness in Architecture, Engineering and Construction*, CRC Press, 11–18.

Pauwels, P. and Terkaj, W. (2016). "EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology." *Automation in Construction*, 63, 100–133.

Pazlar, T. and Turk, Ž. (2008). "Interoperability in practice: geometric data exchance using the ifc standard." *Journal of Information Technology in Construction (ITcon)*, 13(24), 362–380.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). "Scikit-learn: Machine learning in python." *the Journal of machine Learning research*, 12, 2825–2830.

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., and Pauwels, P. (2020). "BOT: The building topology ontology of the W3C linked building data group." *Semantic Web*, 12(1), 143–161.

Rasmussen, M. H., Lefrançois, M., Schneider, G. F., and Pauwels, P. (2021). "Bot: The building topology ontology of the w3c linked building data group." *Semantic Web*, (Preprint), 1–19.

Sacks, R., Eastman, C., Lee, G., and Teicholz, P. (2018). *BIM handbook: A guide to building information modeling for owners, designers, engineers, contractors, and facility managers*. John Wiley & Sons.

Sacks, R., Ma, L., Yosef, R., Borrmann, A., Daum, S., and Kattel, U. (2017). "Semantic Enrichment for Building Information Modeling: Procedure for Compiling Inference Rules and Operators for Complex Geometry." *Journal of Computing in Civil Engineering*, 31(6), 04017062.

Sacks, R., Wang, Z., Ouyang, B., Utkucu, D., and Chen, S. (2022). "Toward artificially intelligent cloud-based building information modelling for collaborative multidisciplinary design." *Advanced Engineering Informatics*, 53(June), 101711.

Stavropoulos, P., Chantzis, D., Doukas, C., Papacharalampopoulos, A., and Chryssolouris, G. (2013). "Monitoring and control of manufacturing processes: A review." *Procedia CIRP*, 8, 421–425.

Törmä, S. (2013). "Semantic linking of building information models." *Proceedings - 2013 IEEE 7th International Conference on Semantic Computing, ICSC 2013*, 412–419.

Varoudis, T. and Patlakas, P. (2014). "A model for a distributed building information system." *Proceedings of the 32nd International Conference on Education and Research in Computer Aided Architectural Design in Europe*, 505–513.

Venugopal, M., Eastman, C. M., and Teizer, J. (2015). "An ontology-based analysis of the industry foundation class schema for building information model exchanges." *Advanced Engineering Informatics*, 29(4), 940–957.

vom Brocke, J., Hevner, A., and Maedche, A. (2020). "Introduction to design science research." *Design science research. Cases*, 1–13.

Wagner, A. and Rüppel, U. (2019). "BPO: The building product ontology for assembled products." *CEUR Workshop Proceedings*, 2389, 106–119.

Wang, M. Y. (2019). "Deep graph library: Towards efficient and scalable deep learning on graphs." *ICLR workshop on representation learning on graphs and manifolds*.

Wang, Z., Ouyang, B., and Sacks, R. (2022a). "Cbim: Graph-based inter-domain consistency maintenance for bim models." *Available at SSRN 4259999*.

Wang, Z., Ouyang, B., and Sacks, R. (2023a). "Cbim: object-level cloud collaboration platform for supporting across-domain asynchronous design, <https://arxiv.org/abs/2303.03854>.

Wang, Z., Sacks, R., and Yeung, T. (2022b). "Exploring graph neural networks for semantic enrichment: Room type classification." *Automation in Construction*, 134, 104039.

Wang, Z., Yeung, T., Sacks, R., and Su, Z. (2021). "Room type classification for semantic enrichment of building information modeling using graph neural networks." *Proc. of the Conference CIB W78*, Vol. 2021, 11–15.

Wang, Z., Ying, H., Sacks, R., and Borrmann, A. (2023b). "Cbim: A graph-based approach to enhance interoperability using semantic enrichment.

Won, J., Lee, G., Dossick, C., and Messner, J. (2013). "Where to focus for successful adoption of building information modeling within organization." *Journal of construction engineering and management*, 139(11), 04013014.

Wu, J., Akanbi, T., and Zhang, J. (2022). "Constructing invariant signatures for aec objects to

support bim-based analysis automation through object classification." *Journal of Computing in Civil Engineering*, 36(4), 04022008.

Ying, H. and Lee, S. (2021). "Generating second-level space boundaries from large-scale ifc-compliant building information models using multiple geometry representations." *Automation in Construction*, 126, 103659.

Ying, H., Zhou, H., Lu, Q., Lee, S., and Hong, Y. (2019). "Semantic enrichment of as-is bims for building energy simulation." *Advances in Informatics and Computing in Civil and Construction Engineering: Proceedings of the 35th CIB W78 2018 Conference: IT in Design, Construction, and Management*, Springer, 733–740.

Yu, Y., Ha, D., Lee, K., Choi, J., and Koo, B. (2022). "ArchShapesNet: a novel dataset for benchmarking architectural BIM element classification algorithms." *Journal of Computational Design and Engineering*.