

Technische Universität München

TUM School of Engineering and Design

Lehrstuhl für Computergestützte Modellierung und Simulation

# **Building Integration and Graph Analytics: A Digital Twin Approach Using Labelled Property Graphs**

Master's Thesis

for Master of Science Study Program Information Technologies for the Built Environment

Autor: Koray Inal

Matriculation Number: 3765990

Examiner: Prof. Dr.-Ing. André Borrmann

Supervisor(s): Fabian Pfitzner, Sebastian Esser

Begin Date: 23 April 2024

Deadline Date: 13 November 2024

## Table of Contents

1	Abstract	5
2	Introduction	6
2.1	Need for a Comprehensive Graph-Based Model	6
2.2	In-direct Interconnectivity between Project Components	7
2.3	Graph Theory	10
2.4	Neo4j and Cypher	12
2.5	Strengthening Use Cases	14
2.6	Problem Statement and Research Question	15
3	Related Works	16
3.1	Graph Representation of Building Information Models	16
3.2	Principles of Creating Graph Database Based on IFC Model	19
3.3	Details on converting IFC to Graph	21
3.4	Representation of Multiple Models in Graph Database	26
3.5	Graph Enrichment Strategies	28
3.6	Project Evaluation from BIM Management Perspective	30
3.7	Building Integration Graphs and Artificial Intelligence	33
3.8	Previous Research and Identified Research Gap	34
4	Methodology	37
4.1	Scope and Structure	37
4.2	Interdisciplinary Conjunction Graph Structure	38
4.3	Data Extraction Tools and Methods	43
4.4	IFC Meta Data Graph	45
4.5	BIM Management Meta Data Graph	47
4.6	IFC Object Graphs	49
4.7	Semantic Graph Enrichment	52
4.8	Graph Database Workflow	54
4.9	Graph Neural Network and Graph Based Machine Learning	56
5.	Case Study	59

---

5.1.	IFC Meta Graph Generation .....	60
5.2.	IFC Object Graphs Generation .....	65
5.3.	Key Characteristics of the Parsing Script.....	68
5.4.	Batch Cypher Import.....	69
5.5.	Interdisciplinary Conjunction Graph .....	69
5.6.	Analysis .....	70
5.7.	Semantic Enrichment.....	74
5.7.1.	By Amount of Components per square meter.....	75
5.7.2.	By Amount of Components coming from different Disciplines .....	77
5.7.3.	By the Elements that stem from IfcFlowTerminal .....	79
5.7.4.	Coordination Focus Points.....	81
5.8.	Graph Edge Detection with GraphSAGE .....	86
<b>6.</b>	<b>Results and Discussion</b> .....	<b>92</b>
6.1.	Implementation .....	92
6.2.	Potential for Generalizability .....	92
6.3.	Limitations.....	92
6.4.	Research Gap Discussion .....	93
<b>7.</b>	<b>Conclusion</b> .....	<b>94</b>

## Abbreviations

BIM	Building Information Modelling
EIR	Employers Information Requirements
BEP	BIM Execution Plan
LOIN	Level of Information Need
LPG	Labelled Property Graphs
MEP	Mechanical Electrical Plumbing
HVAC	Heating Ventilation Air Conditioning
CDE	Common Data Environment
ICG	Inter-discipline Conjunction Graph
GNN	Graph Neural Network
IMG	IFC Meta Graph
IMO	IFC Object Graph
GUID	Global Unique Identifier
RDF	Resource Description Framework
AEC	Architecture, Engineering and Construction
XML	Extensible Markup Language
URI	Uniform Resource Identifier
CSV	Comma Separated Values
API	Application Programming Interface

## 1 Abstract

Integrating diverse Industry Foundation Classes (IFC) models from various disciplines into a single comprehensive graph database presents formidable challenges. Each discipline contributes unique data structures and semantics, necessitating the harmonization of architectural, structural, and Heating, Ventilation, and Air Conditioning (HVAC) information into a unified framework. Additionally, merging geometry while considering the intricate topology and adjacencies defined within the IFC models further complicates the process, requiring a sophisticated approach to capture the spatial relationships inherent in building structures.

Many data retrieval queries pose challenges when using existing software, primarily because they typically work with single IFC models. The intricate hierarchical arrangement of the IFC schema hinders straightforward manual extraction of building information, demanding a profound comprehension of the IFC object model. Existing BIM query languages have limitations, notably in terms of the extensive understanding of the IFC object model and data mapping mechanisms required by users (Tauscher, 2013)

To address these challenges, a comprehensive graph database emerges as a vital solution. By representing building information as nodes and edges within a graph, it becomes possible to capture the rich network of connections between architectural elements, structural components, and HVAC systems. This approach not only facilitates the integration of diverse data sources but also enables a holistic understanding of building structures, paving the way for advanced applications such as digital twins and informed decision-making in building management. (Ismail A. N., 2017)

## Keywords

BIM, Building Information Graphs, Graph Analytics, Artificial Intelligence

## 2 Introduction

### 2.1 Need for a Comprehensive Graph-Based Model

The use of semantic web technologies in the Architecture, Engineering, and Construction (AEC) domain has emerged from the need to integrate information from different disciplines, all of which have interconnected relationships. This has led to the development of more advanced data models, such as labeled property graphs, to unify disparate sources of data into a cohesive structure. In this context, the pursuit of creating a comprehensive digital twin represents a significant advancement in Building Information Modeling (BIM) integration and graph database analytics. [6]

The endeavor to create a comprehensive digital twin utilizing labeled property graphs represents a multifaceted pursuit with significant implications for Building Information Modeling (BIM) integration and graph database analytics. By constructing a digital twin, the aim is to delve deeper into the intricate interplay between architectural, structural, and HVAC systems, thereby enhancing our understanding of building dynamics. Information represented in multiple BIM models of each discipline will be combined and connected within a comprehensive graph data model, revealing complex relationships between model elements from different discipline models and defining and analyzing these connections.

This initiative underscores the growing importance of digital twins as powerful tools for simulating, analyzing, and optimizing building performance across various domains. The study aims to propose an integration framework for melding BIM and graph theory, merging these two modeling paradigms to formulate a topology model for building projects. This entails extracting information on project elements and their interconnections from databases housing BIM data. Previous works by Nguyen, Oloufab, & Nassar (2005) and Paul & Borrmann (2009) have demonstrated the capability to automatically extract relationship information from BIM. (Ismail A. &, 2018)

The resulting graph-based model offers several advantages:

1. Unified representation of diverse project aspects (e.g., requirements, design, and construction planning) within a singular model.
2. Facilitation of data adjustments and updates through the application of graph transformation rules.
3. Exploration of project topology via graph-theoretic algorithms.

Utilizing a graph-based model elevates the representation to a higher abstraction level, transitioning from object-level depiction to a holistic portrayal of the entire project. The model can integrate different project facets—typically confined within separate models—such as the building program, the design, and the project plan. Each IFC object is construed as a node in the graph, and the seamless integration of disparate aspects within the graph-based model stems from defining relationships between elements of various facets as straightforward links between nodes. These relationships can also be derived from BIM databases, where they are denoted as `IfcRelationship` objects, subsequently represented as links in the graph.

For system representations like building projects, graph-based models prove apt, as they accommodate the addition of new element types and relationships over time. Graph Transformation (GT) rules facilitate the incorporation of new data types, dictating how the graph is structured and evolves. This approach extends BIM's scope beyond component design to encompass diverse aspects like user requirements, resource allocation, and maintenance activities. The application of graph-theoretic algorithms within the model permits the analysis of building project topology and issues involving numerous elements concurrently, thereby enabling efficient analysis and management of building projects. (Austern, 2024)

## **2.2 In-direct Interconnectivity between Project Components**

In construction projects, complex interconnections between various components are often not immediately apparent. These hidden relationships can span across different disciplines and phases of the project, making them challenging to identify. Understanding these connections is crucial, as they can significantly impact the overall project outcome. For example, a change in one part of the project might unexpectedly influence another, leading to issues that only become evident later in the process.

Addressing these unseen interdependencies early is essential for effective project management and achieving all project objectives.

Furthermore, not every aspect of a project is clearly defined from the outset, particularly those involving coordination between different disciplines. This challenge is discussed by Whelton and Ballard in *Wicked Problems in Project Definition* (2002). The complexity of these issues underscores the need for a thorough understanding of hidden interconnections to better manage and mitigate potential impacts throughout the project lifecycle.

This concept is later mentioned in the work of Isaac and Navon in *Modeling building projects as a basis for change control* (2009) as in following statement *“Modifications in the design and planning of building projects are made through a complex and iterative process, which can span an extended period. The full impact of these changes on the project often becomes apparent only after this process is completed.”* (Isaac & Navon, 2009)



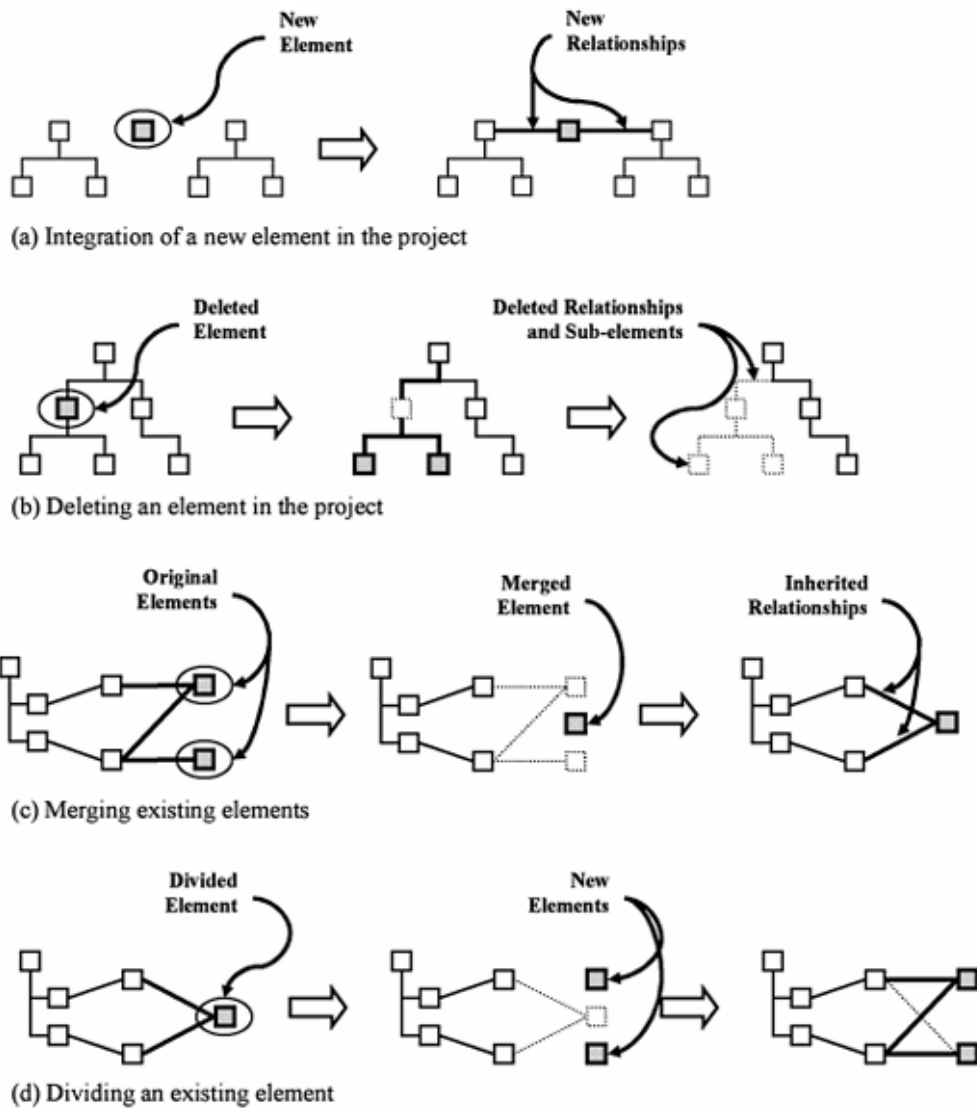


Figure 1 Impact of Changes within Project Isaac, Navon(2009)

In the illustration above, the possible outcomes of such changes are depicted. A graph structure is particularly well-suited for this context due to its clear, easily understandable representation of relationships. Once the connections between nodes are evaluated, the layered and indirect impacts of changes can be tracked with precision.

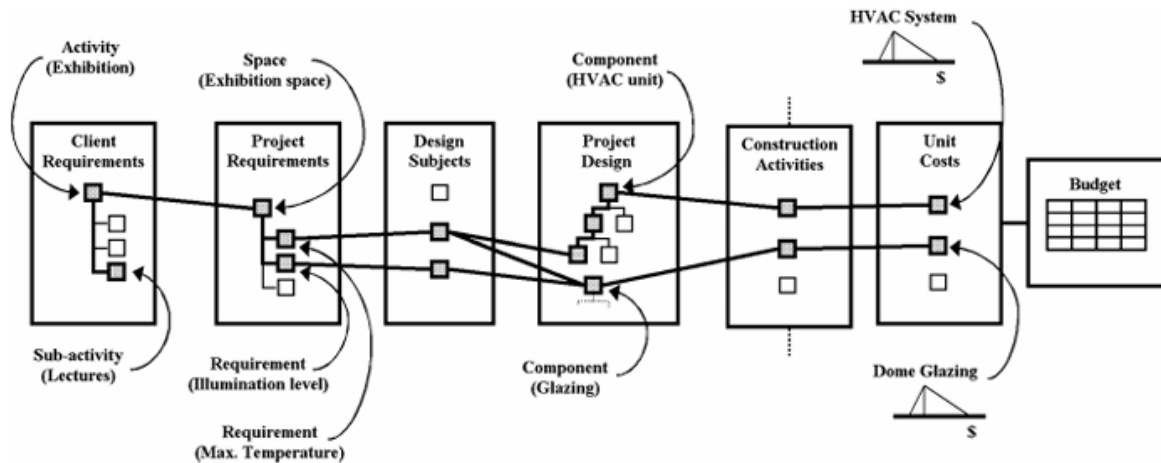


Figure 2 Identification of the impact of the additional user activity on the project. S. Isaac, R. Navon / *Automation in Construction* 18 (2009)

This paper addresses the issue from a consequence point of view, stating that unknown or later found out connections, requirements or prerequisites between project components cause delays or cost increases.

This figure illustrates the layered, in-directional nature of the construction project and its possible outcomes. As mentioned before, because the research paper aims to find out cost consequent relationships, project components were further connected to additional nodes representing activities, costs and eventually budget to find out what action would result in such a cost change.

## 2.3 Graph Theory

Graph theory is a branch of mathematics focused on the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph is made up of nodes (also called vertices) connected by edges (also called links or arcs). Graphs are used in various fields, from computer science and engineering to biology and social sciences, for modeling networks, relationships, and

### Key Concepts in Graph Theory

1. Vertices and Edges: Vertices (nodes) represent objects, and edges represent connections or relationships between them. In BIM, vertices could represent building elements, and edges could represent spatial or functional relationships.
2. Directed and Undirected Graphs: Directed graphs have edges with a one-way relationship, while undirected graphs have two-way relationships.

3. **Weighted Graphs:** Edges in weighted graphs carry a numerical value (weight) representing connection strength, such as distance in transportation networks.
4. **Connectivity:** A graph is connected if there is a path between any two vertices, important for applications like infrastructure planning.
5. **Subgraphs:** A subgraph is a smaller portion of a graph, useful for breaking down complex networks.
6. **Paths and Cycles:** A path is a sequence of connected vertices, and a cycle is a path where the first and last vertices are the same, helpful in detecting feedback loops.

#### Applications of Graph Theory

1. **Computer Science:** Used in algorithms for searching, sorting, and optimization, and widely applied in AI and machine learning.
2. **Social Network Analysis:** Models relationships (nodes and edges) in social networks to analyze influence and information flow.
3. **Biology:** Models biological networks like protein interactions and ecological systems.
4. **Engineering and Infrastructure:** Models transportation and communication networks, optimizing routes and resource management.

#### Graph Theory in BIM and IFC Models

In the context of BIM, graph theory is used to model the complex relationships between different building elements. Nodes represent building objects (e.g., walls, doors, spaces), and edges represent the relationships (e.g., spatial adjacency, functional dependencies) between these objects. Using graph databases like Neo4j, these models allow for advanced querying and analysis of building data, enabling features such as clash detection, route planning, and data validation.

Graph-based BIM representations enable interdisciplinary conjunction, where models from various disciplines (e.g., architecture, structural engineering, mechanical engineering) can be combined, queried, and analyzed as part of a unified project network. This process enhances data interoperability, as the graph can accommodate the rich, connected nature of building elements while allowing for flexible updates and

changes during the project's lifecycle. (Diestel, 2017) (Harary, 1969) (Gross & Yellen, 2018)

## 2.4 Neo4j and Cypher

Neo4j is a widely used graph database that follows the Labeled Property Graph (LPG) model. It is optimized for handling large-scale, highly interconnected data. Neo4j provides its own query language called Cypher, which allows users to perform efficient graph traversal and analysis. It has found applications in many industries, including social networks, fraud detection, and especially in building information modeling (BIM) and interdisciplinary project analysis.

### Labeled Property Graph (LPG) Model

A Labeled Property Graph is a graph where:

- Nodes (or vertices) represent entities or objects.
- Edges (or relationships) represent the connections between entities.
- Labels can be attached to nodes, identifying their types or categories.
- Properties (key-value pairs) can be stored on both nodes and edges to describe attributes and metadata.

### Cypher Query Language

Cypher is Neo4j's declarative query language, designed to simplify graph querying. It allows users to match patterns, perform traversals, filter data, and manipulate nodes and relationships. Its syntax is intuitive and focuses on graph pattern matching.

For example, a query in Cypher might look like:

```
1. MATCH (person:Person)-[:WORKS_AT]->(company:Company)
2. WHERE person.name = 'Alice'
3. RETURN company.name
4.
```

This query finds companies where people named "Alice" work. Cypher's declarative nature makes it highly readable and efficient for graph queries

## LPG vs RDF: Why Labeled Property Graphs Are Better for Certain Applications

While RDF is commonly used for graph-based data representation in the Semantic Web, LPG (Labeled Property Graphs) often provide more practical solutions, particularly for use cases like BIM.

### 1. Schema Flexibility:

- LPG: Nodes can have multiple labels and properties, and relationships can also carry properties, making it easier to model complex, evolving data.
- RDF: Relies on subject-predicate-object triples and ontologies, which make representing complex relationships harder and often inefficient.

### 2. Performance:

- LPG: Optimized for deep node traversal and large-scale, interconnected data, ideal for tasks like BIM.
- RDF: Queries, written in SPARQL, can be slower, particularly with complex or large datasets due to the sheer number of triples needed.

### 3. Readability and Ease of Querying:

- LPG: Cypher offers intuitive pattern matching for complex queries.
- RDF: SPARQL queries are often more verbose and harder to work with for complex structures.

```
1. MATCH (person:Person)-[:WORKS_AT]->(company:Company)
2. WHERE person.name = 'Alice'
3. RETURN company.name
4.
```

### 4. Real-World Modeling:

- LPG: Easily models complex, nested relationships in domains like construction.
- RDF: Often too rigid for evolving, domain-specific needs, requiring workarounds like reification to represent complex relationships.

### 5. Support for Multi-Model Data:

- LPG: Naturally integrates different models (e.g., architectural, structural) into a unified graph.

- RDF: Struggles with practical multi-disciplinary model integration due to its triple-based structure.

#### 6. Workarounds in RDF:

- RDF's rigid structure often requires complex workarounds, like reification, which adds unnecessary complexity in managing and querying data.

Overall, while RDF has its strengths, LPG's flexibility, efficiency, and simplicity make it a better fit for complex, interdisciplinary projects like BIM, providing a clearer and more scalable way to represent construction data. (neo4j, n.d.) (Borrmann & Scholz, Combining graph-based analysis and knowledge-based reasoning for managing infrastructure operation models, 2016)

## 2.5 Strengthening Use Cases

Previous research in this area consistently starts from defining BIM processes and common concepts related to BIM models and their intended purposes. One of the fundamental components of a BIM project is the Use Case concept. A BIM Use Case refers to a specific scenario or purpose for which a BIM model is created. These scenarios can range from clash detection and quantity takeoffs to performance analysis, facility management, and many more. Essentially, each BIM model is developed with the aim of fulfilling one or more specific Use Cases that reflect the project's requirements and goals.

The fulfillment of known Use Cases can serve as an important evaluation criterion for IFC models. By aligning the attributes, relationships, and structure of IFC models with their corresponding Use Cases, one can assess whether the model contains the necessary information and structure to achieve its intended outcomes. For instance, if a model is designed to facilitate energy analysis, it should include relevant data on material properties, geometry, and energy-related attributes such as insulation or window U-values.

Moreover, Use Case-based evaluation can be integrated into quality control workflows, enabling model checkers and project managers to validate whether the IFC models meet the project's specific needs. This aligns with model-checking tools like

Solibri or Navisworks, which are used to automate the validation of Use Case compliance through rule-based systems.

The connection to the Interdisciplinary Conjunction Graph (ICG) further strengthens this approach, as the ICG facilitates the seamless integration of multiple discipline-specific models into a single framework, where each Use Case can be tracked across different models and domains. The ICG can be enriched with Use Case-related attributes, allowing for the systematic tracking of how each discipline-specific model contributes to fulfilling overall project Use Cases.

In conclusion, incorporating Use Case fulfillment as an evaluation criterion not only enhances the precision of model assessments but also ensures that the models are geared toward practical project objectives. This strengthens the model's alignment with real-world needs, providing a structured method for verifying that all required Use Cases are addressed adequately within the BIM models. This approach paves the way for more comprehensive project evaluations, leveraging graph-based tools like the ICG to bridge the gap between multiple disciplines and facilitate more informed decision-making throughout the project lifecycle. [17]

## 2.6 Problem Statement and Research Question

Considering the limitations of the IFC structure and the multi-model nature of construction projects, the need for a central, comprehensive database has become apparent. This type of graph database, referred to as an Inter-Discipline Conjunction graph, is especially crucial for analyzing and determining project requirements and characteristics that span multiple disciplines. (Sebastian Esser, 2024)

The first research question is: *How can we create an Inter-Discipline Conjunction graph, identify and utilize relationships to enrich it, and determine whether these relationships can be generated automatically?*

The second research question is: *After creating the Inter-Discipline Conjunction Graph, how can we identify project characteristics that involve multiple disciplines, as discussed in the previous chapter? Can projects be evaluated utilizing graph comparison and pattern analysis techniques that are unique for graph databases?*

### 3 Related Works

To achieve the objectives outlined in the previous section, a comprehensive review of existing research in Building Information Modeling (BIM) and graph databases has been conducted. This study's foundation lies in understanding how graph databases, particularly within BIM, can enhance data management, analysis, and interdisciplinary integration by defining necessary objects within spaces, revealing underlying indirect connections and dependencies within projects.

Graph databases, due to their flexibility and efficiency in handling highly interconnected data, provide a powerful solution for addressing the complex relationships inherent in BIM models. Scholz and Borrmann's (2016) study, *Combining Graph-Based Analysis and Knowledge-Based Reasoning for Managing Infrastructure Operation Models*, illustrates how graph databases can enhance project quality by managing complex infrastructure models with interconnected data elements. Through the use of graph-based data representation, the researchers identified dependencies and connectivity that are often overlooked in traditional database structures, such as the interactions between structural and MEP systems. Their findings indicate that graph databases facilitate advanced query-based analysis, leading to improvements in project qualities like design coherence and cross-disciplinary coordination, ultimately boosting efficiency and reducing conflicts. (Scholz & Borrmann, 2016)

Scholz and Borrmann (2016) provide a foundational example of how graph-based systems can enhance interdisciplinary data management in BIM projects. By leveraging advancements in labeled property graphs and semantic web technologies, their study highlights how graph databases effectively manage complex, interconnected data across different project domains. These graph structures enable the integration of BIM models from various disciplines, making it possible to capture intricate relationships between architectural, structural, and MEP components within a unified data model. Building on such research, this section aims to highlight key methodologies and contributions that inform the present study's approach to managing interdisciplinary BIM data and improving cross-domain coordination.

#### 3.1 Graph Representation of Building Information Models



Based on prior research by Napps, Zahedi, König, and Petzold (2019), the graph representation of building information models has been extensively examined. In their study, *Visualization and Graph-Based Storage of Customised Changes in Early Design Phases*, the Industry Foundation Classes (IFC) data structure is identified as a core component in representing Building Information Modeling (BIM) data within graph databases. This work emphasizes the potential of graph-based systems to effectively manage and visualize BIM data, particularly during the early design stages. (Napps P. , Zahedi, König, & Petzold, 2019)

The graph representation of IFC entities captures the complexity of the building model by maintaining the interrelationships and attributes of different components. For example, in the labelled property graph (LPG) approach, each node represents an IFC entity, such as `IfcBuilding` or `IfcWall`, with properties stored within the nodes. Edges represent the interactions or relationships between these entities, such as adjacency or containment, and can also carry attributes. This method allows for detailed querying and comparison of different design variants, aiding architects in the early design phases by providing insights into various design options and their implications. (Napps, Zahedi, König, & Petzold, 2022)

Another relevant study is *The Semantic Link Between Domain-Based BIM Models* by Teclaw, Rasmussen, Labonnette, Oraskiri, and Hjelseth (2023). This research examines how semantic web technologies, particularly the use of ontologies and RDF-based formats, can improve interoperability and machine readability in Building Information Modeling (BIM). (Teclaw, Rasmussen, Labonnette, & Hjelseth, 2023)

The research focuses on improving interoperability by converting IFC models into RDF graphs using tools like IFC-LBD and utilizing ontologies such as BOT (Building Topology Ontology) and FSO (Flow System Ontology), illustrated in Figure 10. By creating unique URIs for each element, the methodology enables linking of equivalent elements across different domain models, effectively minimizing redundant data while preserving essential relationships between building components.

*Figure 10 IFC2LBD Tool Landing Page*

The IFCtoLPG tool is utilized to generate .ttl files, which are then further refined. This tool automatically converts IFC files into .ttl files, incorporating the ifcOWL ontology. As a result, the output retains the rich set of properties and relationships defined within the ifcOWL framework.

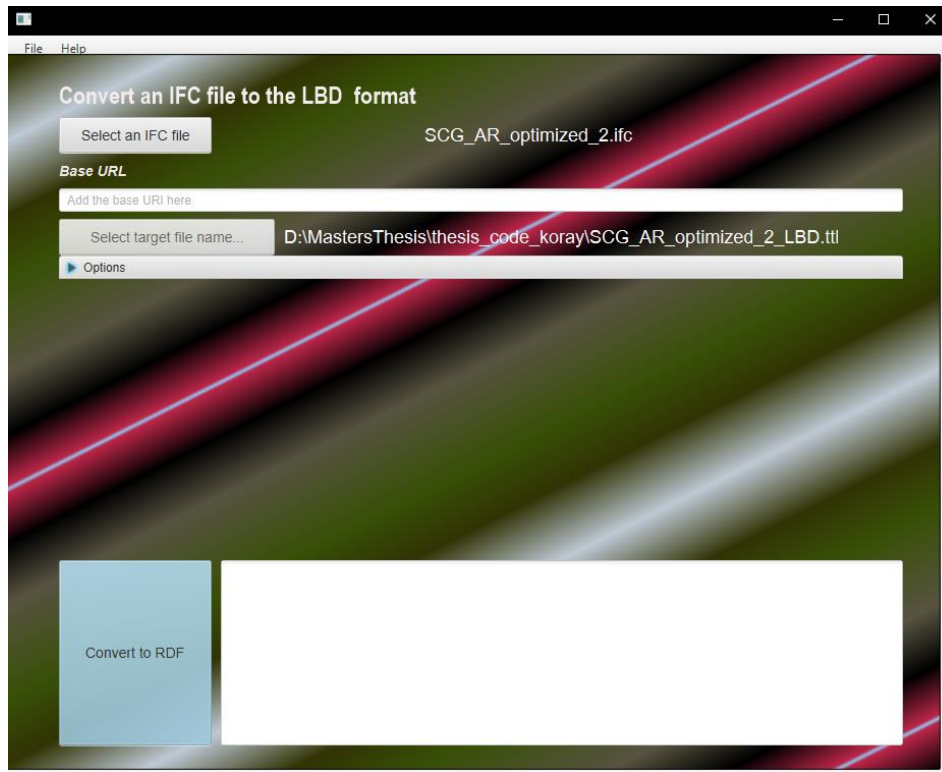


Figure 3 IFC2LBD Tool Landing Page

To enable the transfer of comprehensive IFC data, including all its properties and relationships, ifcOWL has been selected. ifcOWL is an ontology specifically designed for representing IFC data in graph databases. The IFCtoRDF converter is used to extract a .ttl file from the IFC model, ensuring that the rich and complex data embedded in the IFC is accurately represented and can be effectively leveraged in later stages of the workflow.

A case study involving architectural, plumbing, and ventilation models illustrates how this methodology improves query efficiency in SPARQL, showing a 30-element increase in graph data through additional semantic links. The approach simplifies complex queries and can be applied to real-world applications, providing an enhanced foundation for Digital Twin development by facilitating interdisciplinary data exchange.

This method addresses existing limitations by enabling a more efficient and flexible comparison of BIM elements, ultimately improving collaboration among stakeholders and contributing to the development of intelligent, machine-readable

models in construction. Future work will expand this framework to include interdisciplinary MEP elements, advancing the integration of BIM with Digital Twin technologies. (Teclaw, Rasmussen, Labonnette, & Hjelseth, 2023)

### 3.2 Principles of Creating Graph Database Based on IFC Model

To establish the desired graph database based on IFC models, the **IfcWebServer** and the work of Ismail, Strug, and Ślusarczyk in *Building Knowledge Extraction from BIM/IFC Data for Analysis in Graph Databases* (2018) provide foundational methods. Their study addresses the complexity of modern buildings, emphasizing the need for effective data extraction from IFC files to store in graph databases for efficient querying and knowledge extraction. (Ismail A. &, 2018)

Similarly, Alharbi and Hughes (2021) underscore the potential of graph databases for advanced BIM model analysis, demonstrating how IFC data can support interdisciplinary coordination across architectural, structural, and MEP domains. By transforming BIM data into labeled property graphs, both studies illustrate workflows that encode spatial and non-spatial data, such as room dimensions and door types, into graph structures. These structures facilitate the analysis of accessibility, safety, and other critical project aspects, enabling practical applications through streamlined querying processes. (Ismail, Strug, & Ślusarczyk, 2018)

Furthermore, *Building Knowledge Extraction from BIM/IFC Data for Analysis in Graph Databases* (2018) delves deeper into the IFC structure and indicates key concepts about the schema that are essential for the graph database. The authors describe the hierarchical structure of IFC models, where each entity (e.g., walls, doors, spaces) contains a set of attributes and relationships. These relationships can represent how different components of a building are connected or how certain physical attributes like door size or wall thickness are defined. The paper emphasizes the versatility of IFC, which allows it to describe a wide range of building components and their relationships. Key IFC entities include `IfcSpace`, representing functional areas like rooms, and `IfcWall`, describing the structural elements. The section concludes by highlighting the role of these entities in the larger BIM structure and their importance in extracting building knowledge.

**IFC WebServer**

[Upload](#) | [My models](#) | [MVDs](#) | [Scripts](#) | [Extensions](#) | [Plugins](#) | [IfcDoc](#) | [Help](#) | [Find](#) | [Contact us](#) | (korayinal) [logout](#)

**Upload new IFC model**

Choose a file (\*.ifc) from your computer and click the upload button

No file chosen

---

- **Don't use spaces or special characters**  
 (@ # / ( ) ? ' ' ; ; B ü ö ä ! | > ) in file name  
 - Don't upload files with asian letters  
 - File extensions are case-sensitive, don't upload \*.IFC or \*.DAE files

**Accepted formats:**

- \*.ifc (STEP format)

switch GUI: ([simple-interface](#), [BIM Reporter](#))

IFC schema:

Model views:

You didn't upload any IFC model

Run script

ifc   [All Classes](#) | [Non-Abstract Classes](#)

SpatialElements(6)	Building elements(40)	Relations(49)	All(653)	Non abstract classes(556)
SpatialStructureElements	BuildingElements	Relationships	2DCompositeCurve	2DCompositeCurve
Building	Beam	RelAggregates	ActionRequest	ActionRequest
BuildingStorey	BeamType	RelAssigns	Actor	Actor
Site	BuildingElementComponent	RelAssignsTasks	ActorRole	ActorRole
Space	BuildingElementPart	RelAssignsToActor	ActuatorType	ActuatorType
SpaceType	BuildingElementProxy	RelAssignsToControl	Address	AirTerminalBoxType
SpatialStructureElementType	BuildingElementProxyType	RelAssignsToGroup	AirTerminalBoxType	AirTerminalType

Selected classes:

User-defined report([Help](#))

Advanced filter([examples](#))  
 type a filter expressions:  
 o.class == IFCWINDOW and o.area >= 2

Output format:  
 HTML  XML  IFC  CSV  JSON

Project website: <https://github.com/ifcwebserver> | [Blog](#) | [BIMs sources](#)

Figure 4 IFC Webserver

The IFC Webserver platform allows users to upload their models and select specific spatial elements, building components, relationships, as well as both abstract and non-abstract IFC classes to extract from the models for conversion into a desired format. In *Building Knowledge Extraction from BIM/IFC Data for Analysis in Graph*

*Databases* (2018), the workflow is executed using CSV files for data extraction and conversion.

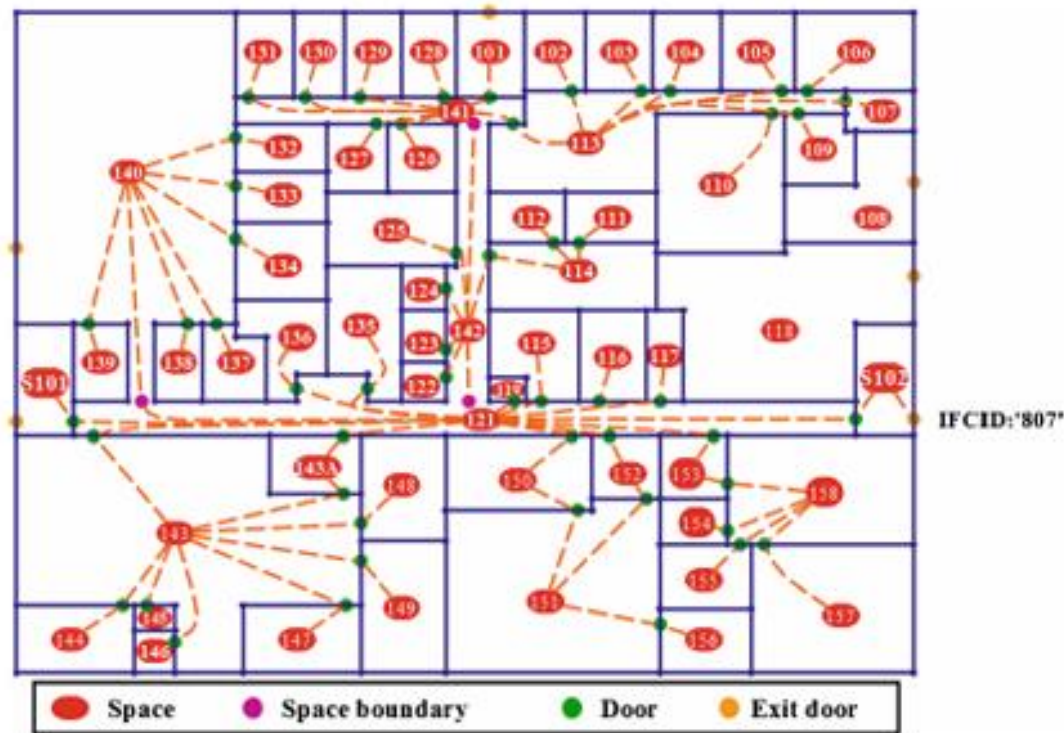


Figure 5 Emergency Routes drawn on the floor plan (Ismail, Slusarczyk 2018)

In Figure above, we can observe the rooms, their corresponding doors, the spaces they connect to, and the pathways leading to exits. The relevant nodes are classified as space boundaries, spaces, doors, and exit doors. This data is incorporated into the graph database and can later be queried for further analysis.

### 3.3 Details on converting IFC to Graph

Another valuable resource on this topic is the further research by Ali Ismail, titled *Application of Graph Databases and Graph Theory Concepts for Advanced Analysis of BIM Models Based on IFC Standards* (July 2017). This study builds on foundational BIM data extraction methods, focusing on leveraging graph databases and graph theory to enhance BIM model analysis. Ismail's work demonstrates how graph-based structures can effectively support complex queries and data integration, particularly across interdisciplinary BIM domains, improving data accessibility and analysis

efficiency. (Ismail, Scherer, & Nahar, Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC Standard, 2017)

The focus of the paper is to preserve the richness of the IFC structure while transferring it into a graph format with minimal information loss. To accomplish this, the IFC schema is carefully examined to determine the most effective way to represent its elements in a graph structure. The approach begins by distinguishing between instance data and metadata, referred to in the paper as the IFC Meta Graph and IFC Object Graph, respectively. This method ensures that the complexity and detail of the IFC structure are maintained in the graph model. Similar approaches are discussed in other research within the graph database domain, reinforcing the importance of separating metadata from instance data for effective data management and querying.

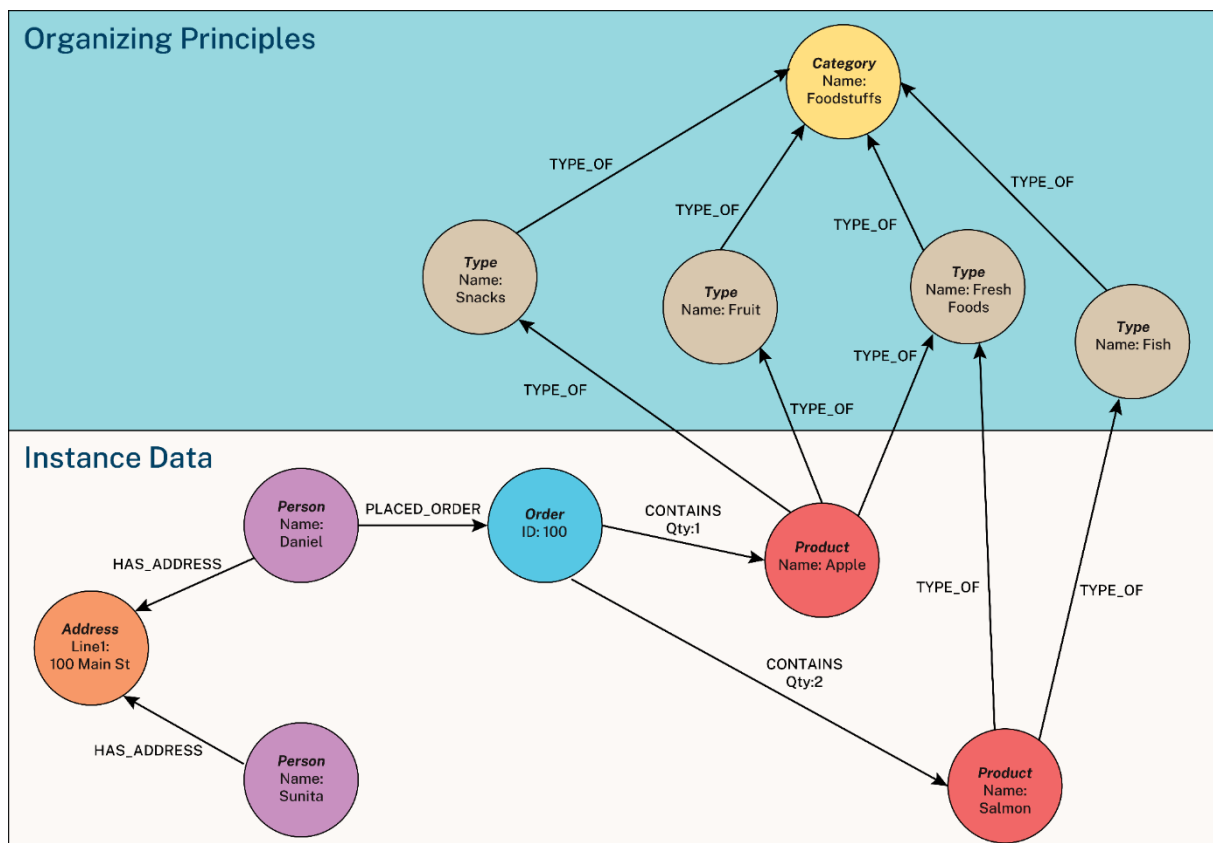
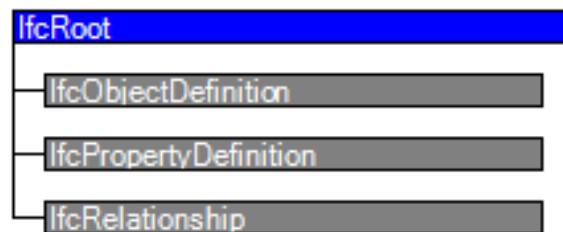


Figure 6 Meta Data and Instance Data Diagram (neo4j)

Figure 10 represents both the instance data and the organizing principles, making it easier to comprehend the structure of the data. A clear separation between the metadata and the instance data is evident, with these two sets of information being connected via relationships. These relationships are essential in maintaining the integrity of the data, ensuring that metadata and instance data are consistently linked.

By structuring the graph in this way, the rich, interwoven relationships between different components of the building model can be effectively captured, enhancing both the accuracy and depth of the analysis. The use of relationships to bind instance data to metadata also allows for more complex queries and insights, as it preserves the context and interdependencies within the data. This design ensures that both high-level organizational principles and detailed instance-level data are integrated seamlessly within the knowledge graph. (Stegeman, 2022)

The implementation of this approach in IFC Context goes as follows. Each IFC model is built from IFC entities arranged in a hierarchical structure. Each entity contains a fixed set of attributes, as well as additional properties if needed. The primary identifiers for these entities are the IFC attributes, which are standardized by buildingSMART. The IFC schema includes three main entity types that form the top level of the `IfcRoot` entity hierarchy.



*Figure 7 Fundamental entity types derived from IfcRoot*

- `IfcPropertyDefinition`: outlines the attributes that can be assigned to objects, enabling the sharing of key information across different instances. It can also detail the specific presence of an object within the project if it is associated with a single instance.
- `IfcObjectDefinition`: encompasses all handled objects or processes, including physical items and products such as roofs, windows, and slabs, which are tangible and visible.
- `IfcRelationship`: encompasses the connections between objects. It enables the assignment of relationship-specific properties directly to the relationship entities, avoiding duplication of these details in the object attributes. The abstract class `IfcRelationship` and its subtypes manage the connections among objects, allowing various properties to be linked to each relationship.

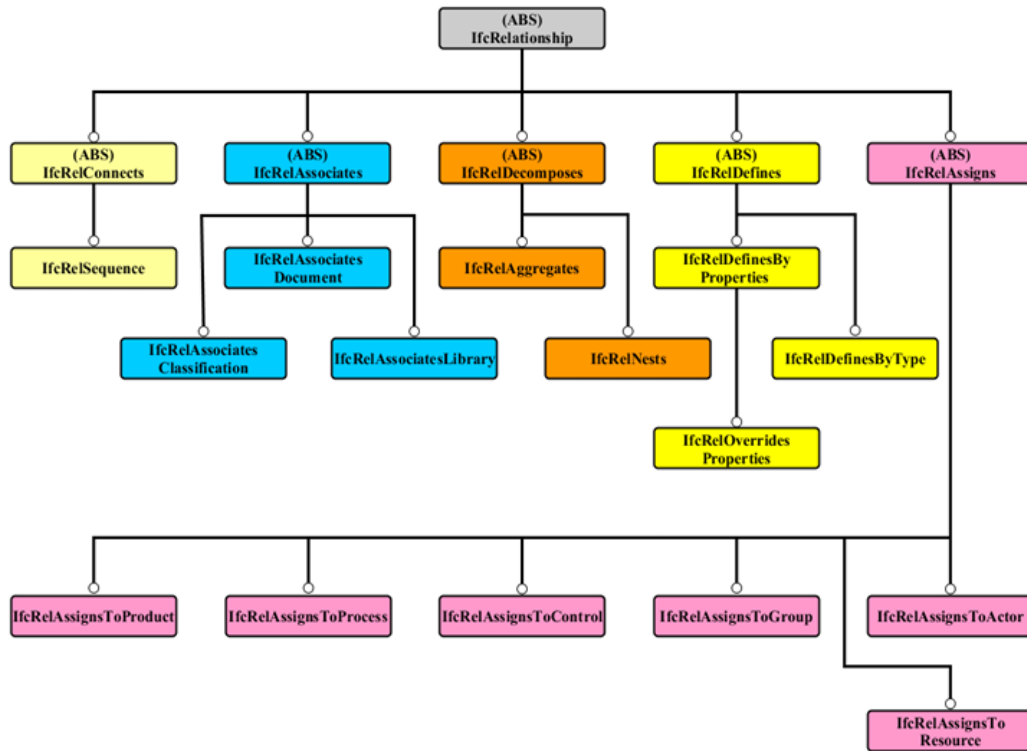


Figure 8 IFC Abstract objectified relationships

Figure 15 illustrates the IFC relationships available within the schema and their hierarchical structure. These relationships are a critical component of the IFC framework and must be carefully mapped into the graph structure to ensure accurate representation and functionality.

IFC classes contain direct attributes that can either define a relationship to another object or represent basic data types, such as integers, strings, logical values, or Booleans. In IFC models, a distinction is made between entity attributes, which are directly associated with the object, and attributes that indicate relationships to other objects. The latter is the more commonly used method to extend an object's properties. For instance, as illustrated in Figure 3, an IFC class may include simple data attributes linked to referenced objects, as with `IfcRoot`, or relationship attributes, as seen in `IfcProduct`, or both types of attributes together, as in `IfcObject`.

Moreover, objects can inherit attributes from their supertype classes, allowing property pairs to be assigned to almost any object type, thereby broadening the range of their attributes.



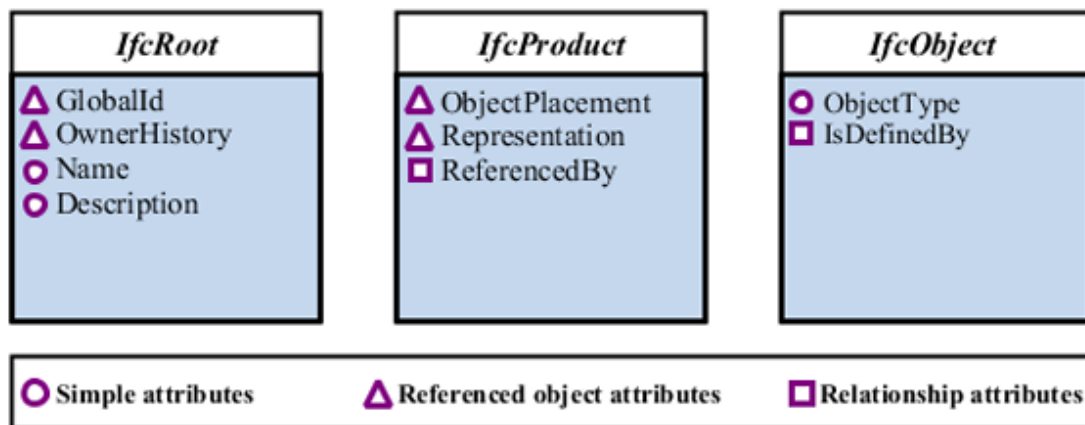


Figure 9 IFC Entity Attributes and Relationship Attributes

The inheritance system within the IFC Schema is fundamental for understanding project information. It plays a key role in structuring data for graph creation and subsequent querying. These referenced attributes, inherited from supertype classes, are not only vital for defining object relationships but can also be used for more specific classifications, such as material-based categorizations. This systematic approach allows for a more organized and detailed analysis of the project data within the graph structure.

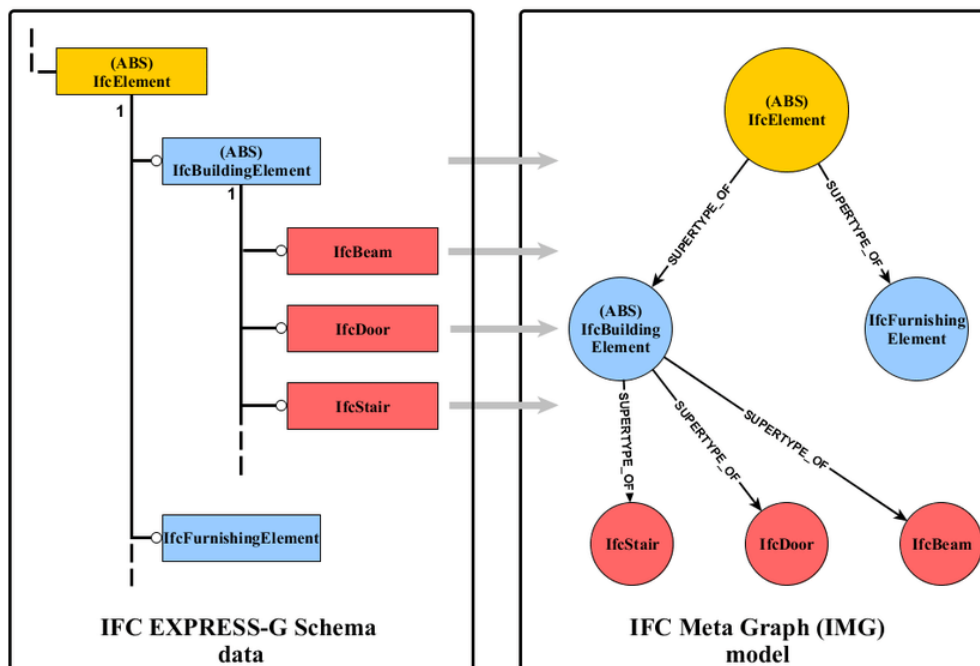


Figure 10 Translation of IFC Schema to IMG (Ismail, Scherer, Nahar 2017)

### 3.4 Representation of Multiple Models in Graph Database

In the research by Wang and Crete, Graph Representation of BIM Models (2023), the structure of an entire construction project is examined. Unlike working with a single model—representing the IFC structure in graph format—representing an entire project with multiple models from different disciplines requires additional processes. The paper states that sub-graphs can be created for each discipline, adhering to principles outlined in the first section. Although this paper primarily examines the use of CDEs, these aspects of the research are still relevant to this topic.

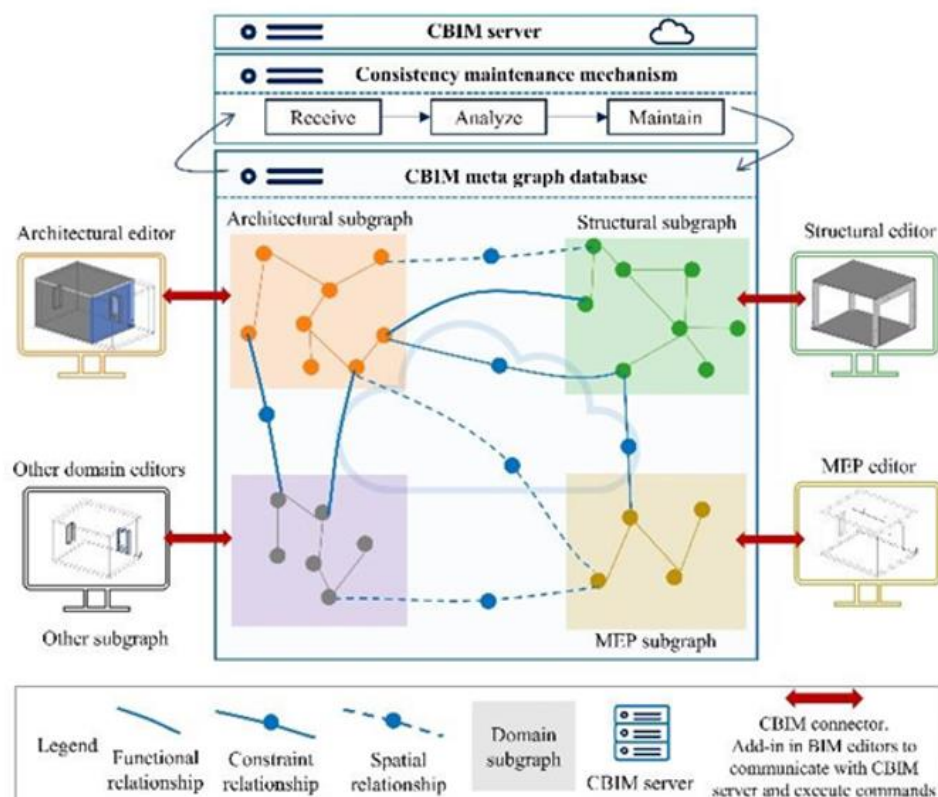


Figure 11 A Diagram representing the sub-graphs from the research paper Wang,Crete(2023)

Figure 18 illustrates that a sub-graph needs to be created for each discipline, and these sub-graphs should then be connected using specific methods. This approach, which combines sub-graphs from different disciplines, is also discussed in several other research papers, including Enhancing Design Coordination Across Disciplines Through Incremental Model Updates and Inter-discipline Conjunction Graphs (Esser, Bormann 2024). These are referred to as Inter-discipline Conjunction Graphs. (Esser & Bormann, 2024)

Next step in constructing the ICG, connecting these domain sub-graphs to each other via additional relationships. This process is called Semantic Graph Enrichment.

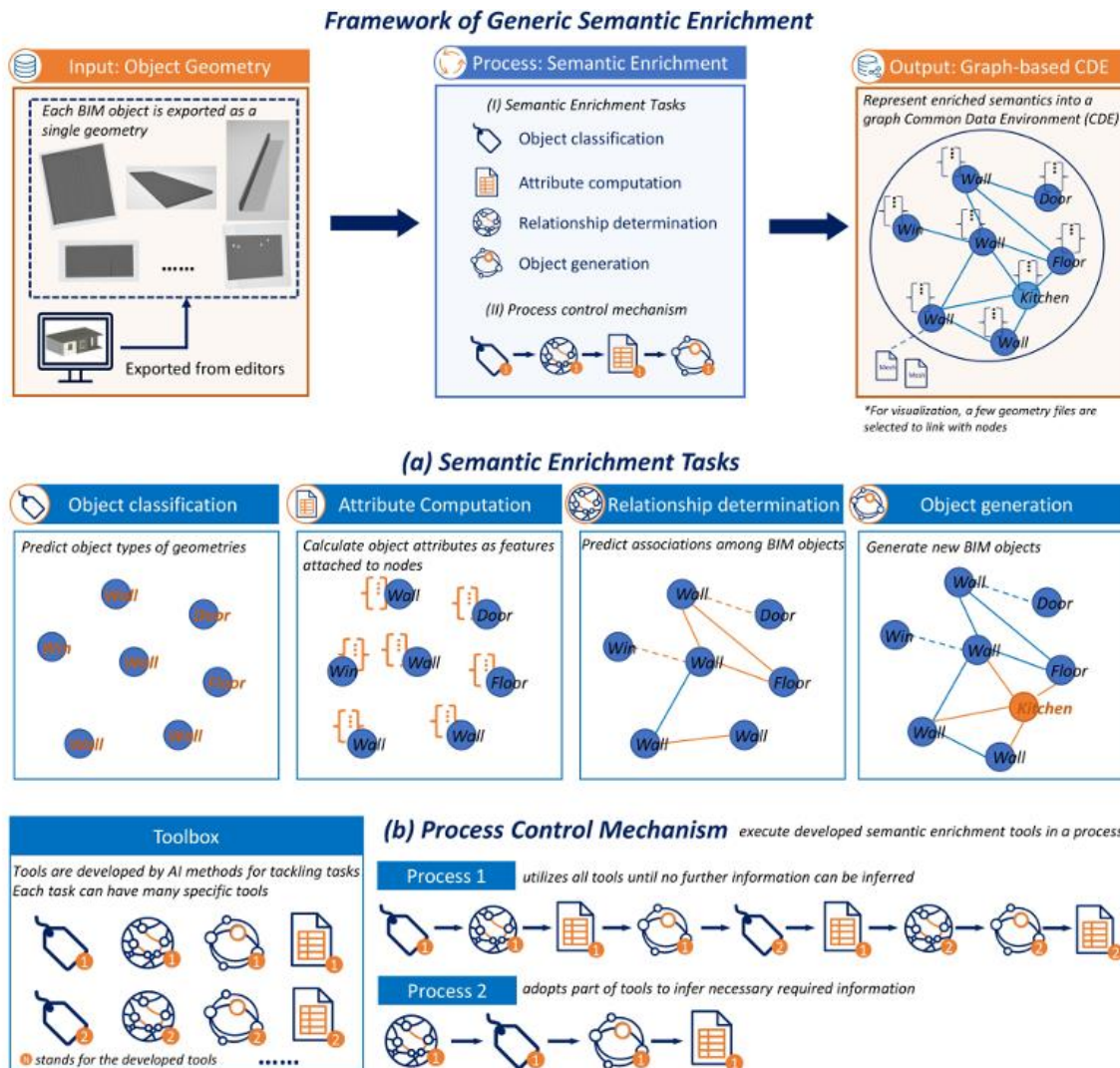


Figure 12 Framework of Semantic Enrichment Wang, Crete(2023)

The research outlines several steps for this process, beginning with classification, attribute computation, relationship determination, and object generation. This approach can also be applied to achieve the goals of our Inter-Discipline Conjunction Graph (ICG). The key consideration is to clearly define the goals and process the graph within this scope. Therefore, sufficient domain knowledge—in this case, construction project knowledge—must be present to guide the technical approach effectively. The figure above explains this concept and the steps of it thoroughly. Figure 19 illustrates the framework of the Semantic Enrichment processes.

### 3.5 Graph Enrichment Strategies

Various research papers have been reviewed to explore strategies for enriching graph structures. Due to the flexible and easily extendable nature of graph databases, inter-discipline connection graphs can be expanded to serve evolving project needs. By defining specific objectives, information can be gathered from different sources, and relevant node relationships can be established. The primary goal of this approach is to detect key project characteristics, which can then be utilized to make more informed decisions during the project lifecycle.

An example of this approach can be found in the work of Borrmann and Esser in “*Enhancing Design Coordination Across Disciplines Through Incremental Model Updates and Inter-discipline Conjunction Graphs*” (2024). In their research, two distinct models—one architectural and the other a furniture model containing rooms for facility management purposes—were analyzed. This study focused on improving design coordination between disciplines by examining how these models interact and connect through the use of inter-discipline conjunction graphs.

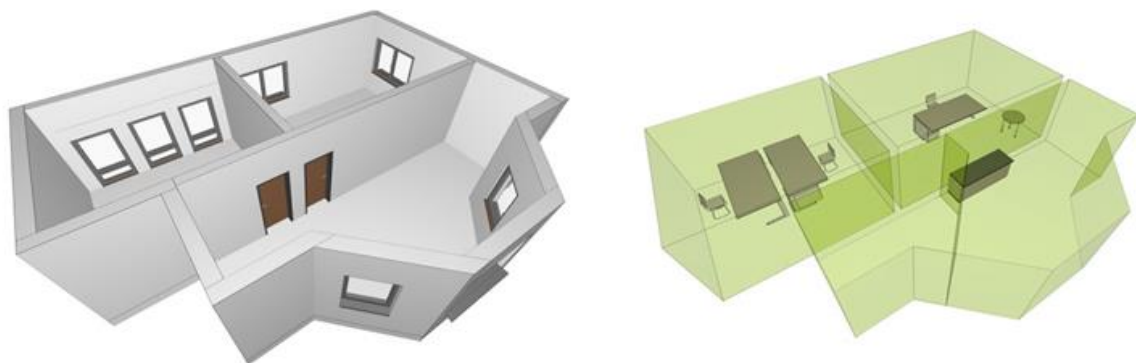


Figure 13 Architectural Model and Room Model with Furniture (Esser, Borrmann 2024)

The main questions addressed when connecting these models were: "What is the containing space for each furniture item?" and "What architectural components are adjacent to each room?" To answer these, the Solibri Model Checker tool was employed. Specifically, the Information Take-Off function within Solibri was utilized to extract and analyze the necessary data from both models. The results obtained from the Solibri Model Checker were then integrated into the Inter-discipline Conjunction Graph (ICG), with defined relationships linking the relevant elements. This helped establish the connections between the architectural components, furniture items, and their corresponding spaces, enhancing the overall coordination between the two

disciplines. (Borrmann & Esser, Enhancing design coordination across disciplines through incremental model updates and Inter-discipline Conjunction Graphs, 2024)

Figure 21 Illustrates utilizing additional software such as Solibri to generate project information which later on will be utilized to create more relationships inside the Graph database.

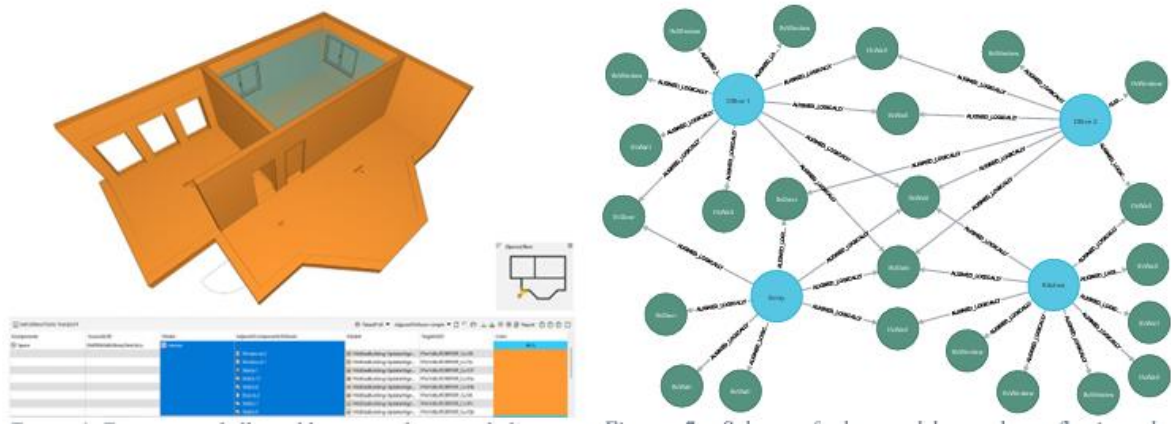


Figure 14 Solibri Information Take Off - ICG Implementation (Borrmann, Esser 2024)

Another valuable resource in this sense is the work from Kayhani, McCabe, Sankaran in “BIM-based construction quality assessment using Graph Neural Networks” (2023) (Kayhani, McCabe, & Sankaran, 2023)

## BIM2Graph

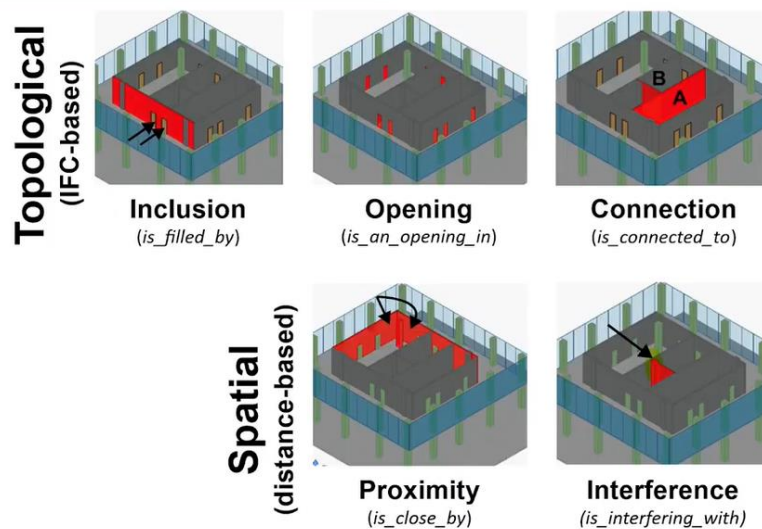


Figure 15 Detected Relationships (Kayhani, McCabe, Sankharan 2023)

In the research, the intended relationships were categorized into two types: Topological and Spatial. Topological relationships are derived from the IFC Schema, representing how different elements are connected within the model. Spatial relationships, on the other hand, are based on the *geometrical positioning* of elements in relation to one another, reflecting their physical layout and proximity in the model. In Figure 22 a classification system based on the properties of the relationships which are mentioned in the paper.

### 3.6 Project Evaluation from BIM Management Perspective

In collaboration with AEC3, a leading BIM Management firm, we have developed a rigorous approach to project evaluation centered around the use of model check rules. These rules are directly tied to the use cases defined by the German state, ensuring that our models can fulfil the intended objectives. (BIM Deutschland, n.d.) (AEC3, n.d.)



Liste der standardisierten Anwendungsfallbezeichnungen

000 Grundsätzliches	+
010 Bestandsfassung und -modellierung	+
020 Bedarfsplanung	+
030 Planungsvarianten bzw. Erstellung haushaltsbegründender Unterlagen*	+
040 Visualisierung	+
050 Koordination der Fachgewerke	+
060 Planungsfortschrittskontrolle und Qualitätsprüfung	+
070 Bemessung und Nachweisführung	+
080 Ableitung von Plan unterlagen	+
090 Genehmigungsprozess	+
100 Mengen- und Kostenermittlung	+
110 Leistungsverzeichnis, Ausschreibung, Vergabe	+

Figure 16 List of Use Cases

Our approach begins by clearly defining what we aim to achieve with the models. Once the objectives are established, we proceed to outline the necessary requirements to meet these goals, which may include both information requirements and coordination requirements. By aligning our model check rules with these predefined

use cases, we ensure that the models not only meet the technical specifications but also support the broader project goals.

This method of evaluation allows us to maintain a high standard of quality and efficiency in our BIM processes, ensuring that each model is thoroughly assessed against the relevant criteria before advancing to the next stage of the project.

### AwF / LOIN Verbindung

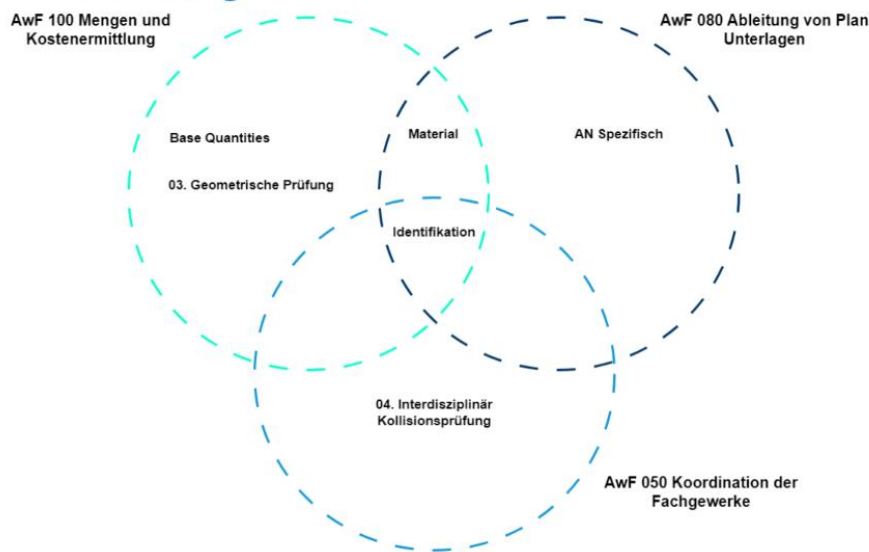


Figure 17 Use Case LOIN Connection Diagram

This diagram illustrates the potential requirements for specific use cases and their interrelations or overlaps. For example, in the diagram, 'AwF100' represents Quantities and Cost Calculations, 'AwF080' corresponds to the Derivation of Drawings, and 'AwF050' denotes the Coordination of Disciplines. At the intersection of these three use cases, we find the 'identification' criteria. Although this diagram provides a simplified representation of the concept, the core idea is to achieve the fulfillment of use cases by meeting these distinct criteria.

Konsistenzprüfung	BIM-M / BIM-GK	IFC Modell bereitgestellt	AWF 000
		IFC Version korrekt	AWF 000
		IFC Projektstruktur korrekt	AWF 000
		natives Modell bereitgestellt	AWF 000
		Namenskonvention eingehalten	AWF 000
		lokale Koordinaten korrekt	AWF 060
		Koordinationskörper vorhanden	AWF 060
		Koordinationskörper lagerichtig	AWF 060
Alphanumerische Prüfung	BIM-GK (BIM-M stichproben)	Grundstücksattributierung	AWF 050, 060
		Gebäudeattributierung	AWF 050, 060
		Geschossattributierung	AWF 050, 060
		Attribute Identifikation	AWF 050, 060
		Attribute Mengen und Maße	AWF 080, 100
		Attribute Baukonstruktion	AWF 070, 080, 100
		Attribute technische Anlagen	AWF 080, 100
		Attribute Kosten	AWF 100
		Attribute Material	AWF 070, 080, 100
		Attribute Raum	AWF 080, 100
		IFC Klassifikationen korrekt	AWF 000
Geometrische Prüfung	BIM-GK, BIM-K	Geschossweise modelliert	AWF 050
		korrekte Geschosshöhen	AWF 050
		Detaillierungsgrad adäquat	AWF 060
		Modellinhalt korrekt	AWF 060
		Kollisionen intradisziplinär	AWF 050
		Duplikate	AWF 050
		Kollisionen interdisziplinär	AWF 050
		Freiräume	AWF 050
		Übereinstimmung (TWP, ARC)	AWF 050

Figure 18 List of Model Check Criteria from AEC3

We have the model check criteria table from AEC3, which displays each model check criterion alongside its corresponding use case. These criteria are organized into logical groups, making it easier to understand their application. The checks range from simple attribute validations to more complex processes like duplicate checks or collision tests, ensuring that each aspect of the model aligns with the intended use cases.

In addition to general evaluation criteria, we also have specific project requirements to ensure that the project is both feasible and practical. For example, it is essential to verify that doors in fire safety walls possess the appropriate fire safety properties. Another example involves assessing the HVAC elements within a specific type of room, such as those labeled with the 'Office Space' parameter. In this context, project requirements dictate that Office Spaces must include a minimum of two sprinkler components and at least one heating component longer than 1.5 meters.

This level of in-depth analysis is technically more challenging than simple attribute checks and requires a more structured approach to maintain integrity, especially when evaluating multiple projects.



### 3.7 Building Integration Graphs and Artificial Intelligence

Further improvements naturally revolve around the advancement of Artificial Intelligence. The application of AI in the built environment varies widely. Established methods, such as computer vision and natural language processing, are already being used in various scenarios, such as construction tracking through automated detection. However, the integration of BIM data into these processes remains largely unexplored. The work by Rafael Sacks, Zijian Wang, and Huaquan Ying addresses this challenge and discusses the application of AI in the context of BIM. (Sacks, Wang, & Ying, 2024)




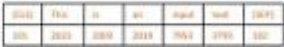







Domain	Machine-readable data	Learning-suitable representation*	Example techniques
Computer Vision	 Image	 Matrix	Convolutional neural network (CNN)
Natural Language Processing	 Text	 Tokens	Transformers
Audio Processing	 Audio	 Spectrogram	Hidden Markov Models (HMMs)
Networks	 Networks	 Graphs	Graph Neural Networks (GNNs)
Building Information Modelling	 BIM models in files or database		

Figure 19 Data for AI (Sacks, Wang, Ying 2024)

Figure 23 illustrates the common data sources and their suitable representation for further AI related processes.

The paper addresses two fundamental questions related to BIM data representation for machine learning:

- 1) What formats are most suitable for representing BIM data?
- 2) What learning techniques are best suited for BIM applications?

The authors argue that, to leverage the full richness of BIM models for AI, it's essential to design data formats that fit BIM-specific needs, rather than tailoring existing machine learning algorithms to pre-defined formats, which often results in incomplete information. They propose using graph-based representations of BIM data, which preserve the complexity of building objects, their relationships, and attributes, as a solution.

The paper introduces the use of knowledge graphs for storing BIM data and property graphs for computation. Knowledge graphs integrate multi-modal BIM information (e.g., object relationships, geometry), while property graphs allow for specific tasks like classification and feature extraction. The authors also conducted an experiment in BIM object classification, showing how graph-based features (like topology) improve the performance of both machine learning and graph neural network (GNN) algorithms.

Key findings from their experiment indicate that combining graph-based and geometry features yields more accurate results in object classification, validating the strength of graph representations in preserving BIM semantics. Despite these benefits, the approach also faces challenges, such as information loss during graph construction and the need for more generic solutions to represent complex BIM data in machine-readable formats.

The study suggests that more research is needed to explore how embedding techniques can transform BIM data into high-dimensional vectors for learning and proposes that self-supervised learning could further enhance AI applications for BIM.

The document emphasizes that new data representations and learning techniques are crucial for fully utilizing BIM information for AI, potentially leading to general AI systems for BIM, much like those in natural language processing.

### **3.8 Previous Research and Identified Research Gap**

Previous research and limitations within existing structures, such as the IFC schema, suggest that a more holistic approach to Building Information Modeling (BIM) can be achieved using graph databases. Given the multi-disciplinary nature of construction projects, graph databases provide a natural solution for addressing the complexity of inter-domain relationships and dependencies.

However, further research into the application of graph databases in the construction industry remains fragmented. While various studies have applied graph-based approaches to tackle different challenges, they often do so with similar techniques, and the potential for this approach—particularly in the evaluation of planning quality—has yet to be fully explored. The rich and multi-layered structure of graph databases, when derived from IFC models and enhanced through semantic enrichment techniques, can reveal critical project information that is difficult, if not impossible, to obtain using conventional methods.

At present, there is a gap in the research regarding the analysis of planning quality through this lens. Bridging this gap by combining the BIM management perspective with real-world experience from industry professionals could lead to more accurate and effective project evaluations. Such an approach would allow stakeholders to extract actionable insights from the interconnected data, thereby improving decision-making processes and enhancing project outcomes.

While previous research and existing structures, such as the IFC schema, suggest that Building Information Modeling (BIM) could benefit from graph databases for more holistic data management, there are notable gaps in applying this technology to analyze specific relationships and dependencies in multi-disciplinary projects. Graph databases are uniquely suited to addressing the complex, interwoven data of BIM environments, offering a way to model and navigate relationships across domains like architecture, structural engineering, and mechanical systems (Zhu et al., 2019). This capability is particularly beneficial in interdisciplinary areas such as HVAC and architectural spaces, where effective coordination can substantially impact project quality by improving energy efficiency, system integration, and overall functionality. (Borrmann & Scholz, combining graph-based analysis and knowledge-based reasoning for managing infrastructure operation models, 2016)

Despite the advantages of graph-based approaches, research in this area remains fragmented, often applying similar techniques without fully exploring their potential for quality assessment in project planning. *Application of graph databases and graph theory concepts for advanced analyzing of BIM models based on IFC standard.* (Alharbi & Hughes, 2021) have leveraged labeled property graphs (LPGs) and semantic enrichment techniques to uncover complex relationships in construction models, yet there has been limited focus on how these methods could specifically

enhance HVAC coordination within architectural spaces. Addressing this gap would provide stakeholders with a clearer view of HVAC elements' spatial and functional roles, supporting better alignment with project objectives and building performance standards.

Integrating insights from industry professionals, a BIM management perspective, and real-world data through graph databases could yield a more actionable framework for evaluating project planning quality. This approach would enable stakeholders to uncover indirect relationships—such as dependencies between HVAC systems and architectural layouts—and analyze them in a way that enhances decision-making, reduces errors, and ultimately leads to more robust project outcomes (Ismail, Strug, & Ślusarczyk, 2018)

## 4 Methodology

Based on previous research, a methodology has been developed. The aim here is to combine the best aspects of the separate studies that have been mentioned in previous chapters.

### 4.1 Scope and Structure

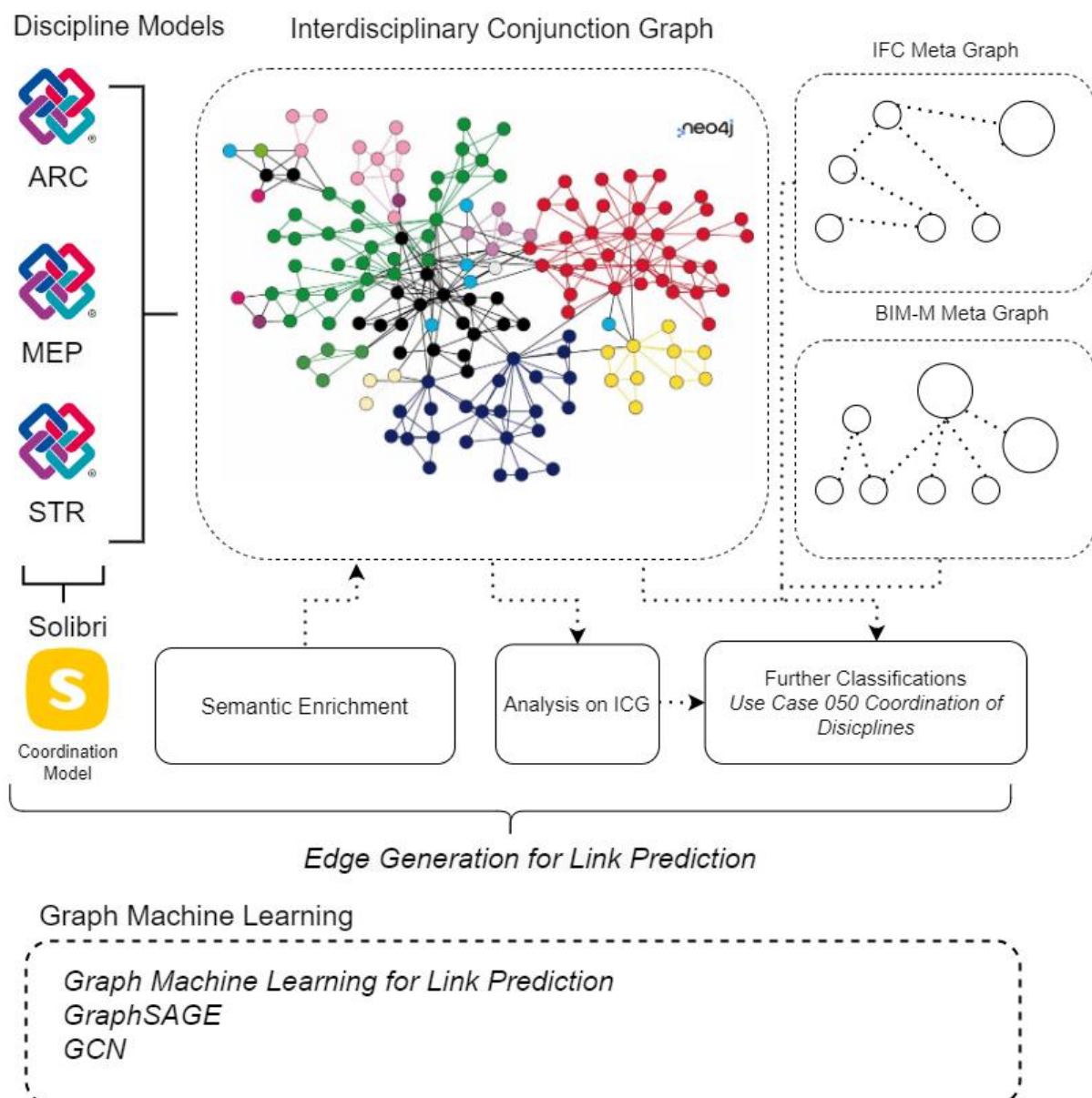


Figure 20 Scope of the Project

In the figure above, the project scope is outlined as a multi-step process. It begins with gathering data from IFC models using parser scripts and Solibri. This data is then used to construct the Interdisciplinary Conjunction Graph (ICG), which will be enriched with Meta Information Graphs and additional outputs from Solibri. The enriched graph will undergo project evaluation analysis, focusing on Use Case 050 (Coordination of Disciplines) by identifying critical spaces and their HVAC relationships. This analysis forms the basis for creating edges in the graph, which will be input into the machine learning algorithm GraphSage to automate edge detection for comprehensive project evaluation.

## 4.2 Interdisciplinary Conjunction Graph Structure

The foundation of this study is the **Interdisciplinary Conjunction Graph (ICG)**, which plays a pivotal role in integrating multiple domains of a construction project. The construction and characteristics of the ICG are shaped by the specific analyses required, which in turn are influenced by industry feedback and the technical necessities inherent to construction projects. These necessities stem from the multi-model nature of construction projects, the discipline-specific models involved, and the IFC-based data representation used across the industry.

A key component of the ICG is the inclusion of critical properties at the **node level**, ensuring that each element has the relevant attributes necessary for performing various analyses. These properties include, but are not limited to, IfcClass, Name, Quantity Information, and GUID (Globally Unique Identifier). These characteristics ensure that the graph can support a wide range of analyses and evaluations across different project domains.

The ICG's structure, designed to accommodate interdisciplinary collaboration, enables a comprehensive view of the construction project and facilitates insights that traditional data structures may overlook. A diagram of the graph's structure provides a

visual overview of how these elements and their properties are interconnected, supporting the intended analyses. Figure 25 illustrates the structure of the ICG.

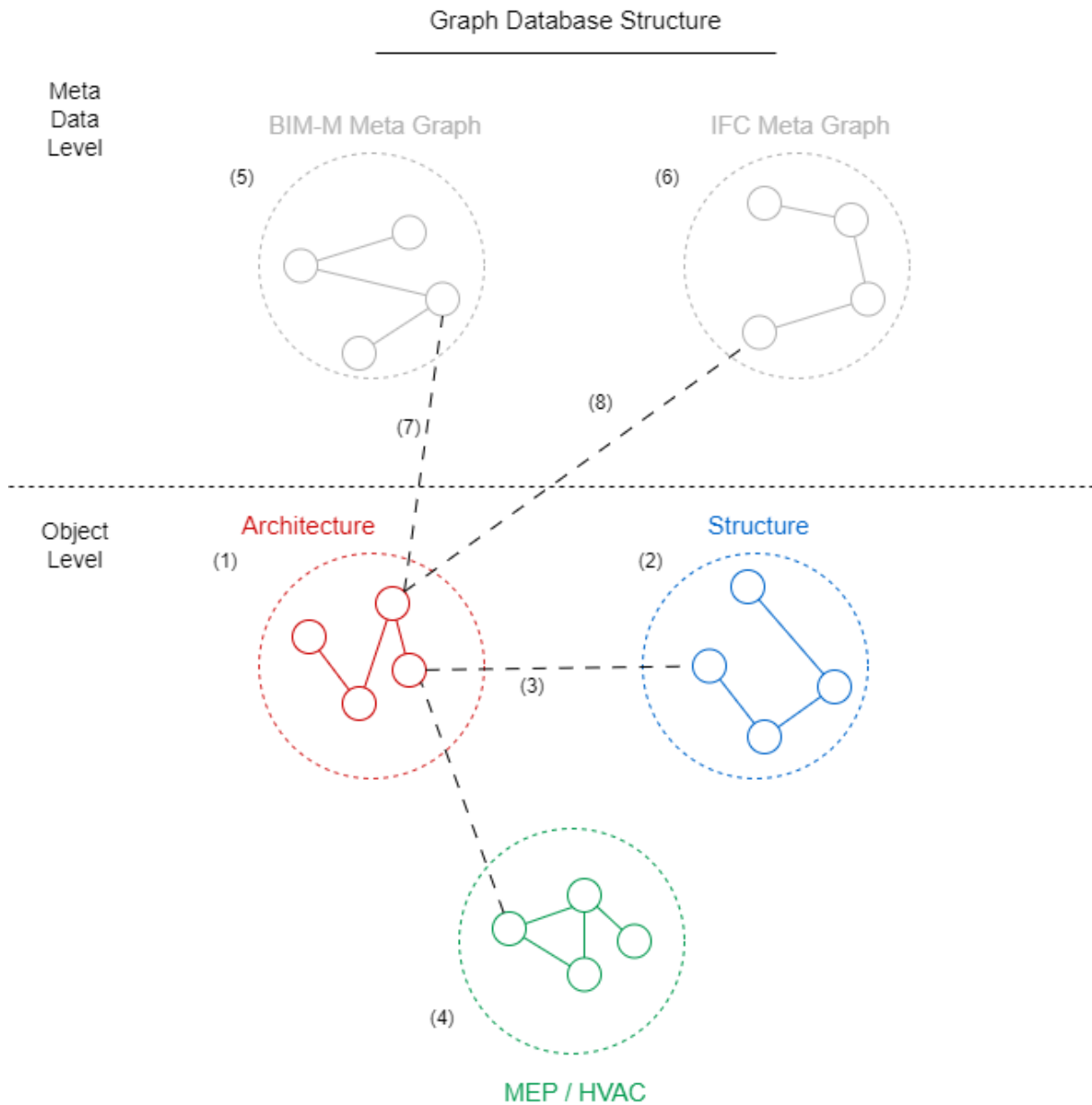


Figure 21 Interdisciplinary Conjunction Graph Structure

1. Architecture Subgraph
2. Structure Subgraph
3. Inter-Domain Relationship
4. MEP Subgraph
5. BIM-M Meta Graph Cluster
6. IFC Meta Graph Cluster

7. BIM-M Object Relationship
8. IFC Schema Relationship

As shown in the figure above, the structure of the Interdisciplinary Conjunction Graph (ICG) is divided into two main sections: the meta-data level and the object level. This separation is essential due to the distinct roles these layers play within the graph.

The meta-data level functions as an organizational framework that references the object level information back to the original data schemas. This layer leverages schema characteristics such as subtype-supertype relationships, ensuring that the relationships and structural rules defined in the schema are applied consistently to the object level. The meta-data level is responsible for maintaining the structural integrity of the graph and ensures that the relationships between classes are accurately reflected.

Meanwhile, the object level deals with the actual instances of the building elements, such as walls, doors, or spaces. These elements are mapped and organized based on the relationships defined in the meta-data level, allowing for a rich and accurate representation of the building's components.

The separation of these levels enables a more flexible and structured approach to managing the complex interrelations in BIM models, while also ensuring that the instance data adheres to the schema's organizational rules. The meta-data level effectively provides a roadmap for interpreting and querying the object level, allowing for comprehensive and efficient analysis.

The Meta-Data Level of the Interdisciplinary Conjunction Graph (ICG) includes two key subgraphs: the BIM-M Graph and the IFC Schema Meta Graph.

The BIM-M Graph is built upon industry knowledge and insights, primarily gathered through collaboration with AEC3, a BIM management firm. This graph encompasses crucial project evaluation criteria, BIM Use Cases, and other relevant data that contribute to the overall assessment and coordination of BIM models. By structuring this information within the BIM-M Graph, the model becomes a central repository for evaluating the performance of the project according to industry standards and specific project goals.



The second subgraph in the Meta-Data Level is the IFC Schema Meta Graph, which is a structured representation of the IFC class hierarchy. This subgraph captures the relationships and organization of the IFC schema, mapping the class structure and attributes as defined by buildingSMART. By integrating the IFC schema at the meta-data level, this graph ensures that the instance-level data (i.e., actual building elements and components) can be properly referenced and aligned with the overarching IFC structure. This schema enables deeper semantic understanding and cross-referencing of model data, facilitating the evaluation of the model's compliance with project goals and industry standards.

Together, these two subgraphs provide a foundation for the ICG, enabling the connection of detailed, instance-level information with broader project evaluation frameworks and established BIM Use Cases, creating a more holistic and structured approach to BIM data management and analysis.

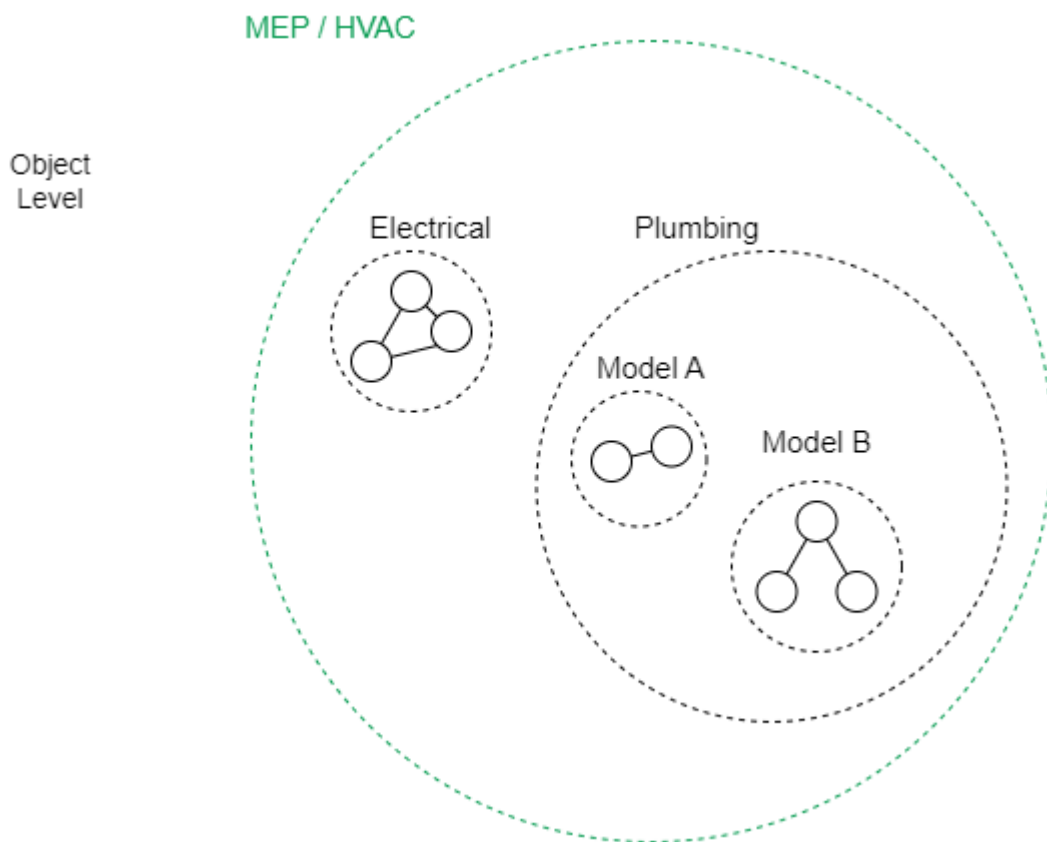
At the object level, we have the discipline sub-graphs, which are generated from the corresponding discipline-specific IFC models. These sub-graphs represent the objects, elements, and relationships pertinent to each discipline within the construction project. Each discipline sub-graph, such as those for architecture, structure, or MEP (Mechanical, Electrical, Plumbing), encapsulates the unique data and elements relevant to that specific field.

In many cases, the discipline sub-graphs contain further sub-discipline sub-graphs, particularly in complex domains like MEP. For example, within the MEP sub-graph, there may be distinct Electrical and Plumbing sub-graphs, each representing the objects, elements, and systems specific to those sub-disciplines. Additionally, within these sub-disciplines, there could be further segmentation into separate models or systems, such as lighting systems within the Electrical sub-graph or drainage systems within the Plumbing sub-graph.

This hierarchical structure ensures that all components of the project, regardless of their complexity or domain, are represented in a clear, organized manner. The breakdown into sub-graphs not only facilitates more effective analysis and coordination between disciplines but also supports the multi-model nature of modern construction projects. By maintaining this structure, it becomes easier to manage and query data

across different disciplines and sub-disciplines, while preserving the relationships and connections necessary for interdisciplinary collaboration.

The figure above illustrates this concept, where the relative sub-graphs and their relationships with each other are clearly visible, showcasing their hierarchical structure. This organization is particularly crucial in real-world projects, which often involve numerous stakeholders and multiple models across various disciplines.



*Figure 22 MEP Sub-graph with its sub-domains*

Lastly, the binding component of the various meta and object sub-graphs is referred to in the research as *\*Semantic Graph Enrichment\**. This concept focuses on extending or defining relationships that link these components together. In a graph database, this process is represented by the creation of additional edges that connect nodes across the sub-graphs.

The primary classification of this enrichment can be described in two categories:

1. Meta-to-Object Enrichment:

This involves creating relationships between the metadata (such as the IFC schema or BIM-M Graph) and the specific objects within the discipline sub-graphs. These connections ensure that the object-level elements, such as walls, rooms, and MEP components, can be traced back to their respective definitions in the metadata level, allowing for better validation and evaluation based on the intended use cases and standards.

## 2. Object-to-Object Enrichment:

This deals with relationships between various objects at the discipline sub-graph level. For instance, connecting elements like walls and doors based on their physical adjacency or functional relationships (e.g., MEP components to architectural spaces). This type of enrichment helps in forming a more interconnected representation of the project, capturing critical interdependencies and interactions between different components.

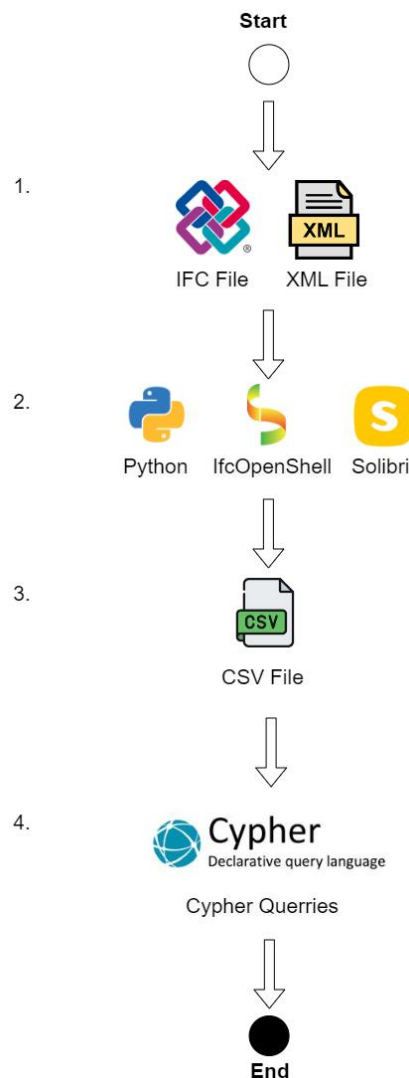
This concept of Semantic Graph Enrichment serves as a vital mechanism for making sense of complex project data, providing a way to query and analyze the information more efficiently. Through these enriched relationships, stakeholders can extract deeper insights from the data, leading to improved project management, coordination, and decision-making.

Due to the flexible nature of graph database structures, enrichment processes, meta-graphs, and object-graphs can be extended without any significant constraints. This allows for the seamless incorporation of additional evaluation criteria and sub-domain knowledge into the existing graph model. New sub-graphs can be introduced and later linked to the broader structure, providing a dynamic and scalable framework.

For example, if fire safety regulations were a necessary component of the project, this domain knowledge could be converted into a meta-graph. This fire safety meta-graph could then be referenced within the existing object structure, enabling stakeholders to analyze how specific instances in the project impact or comply with fire safety requirements. This flexibility makes graph databases an invaluable tool for accommodating complex and evolving project needs, ensuring that various domains, such as safety, structural integrity, and regulations, can all be integrated and evaluated cohesively.

### 4.3 Data Extraction Tools and Methods

The tools and methods are defined through the workflows and necessary input materials. A flow diagram has been developed to point out the graph generation and possibly semantic graph enrichment processes.



*Figure 23 Generation Workflow Diagram*

This diagram illustrates the possible workflow for the generation step in the study. For the analysis section, a distinct workflow might be utilized. The generation workflow is roughly divided into four key steps:

1. **Input Step:** The process begins with either IFC models or raw data such as XML files, which will be used to form the meta-graph. These input files provide the necessary structure and data for the subsequent steps.

2. **Interaction Tools and Methods:** In this phase, tools are needed to interact with the input data. Several options are available:

- Model Viewer/Checker: This tool is particularly useful for visualizing IFC models, running tests, and conducting information take-offs. In this study, Solibri Model Viewer is chosen due to its extensive functionality and rich toolset.

- Python and IfcOpenShell: Python, known for its easy-to-read syntax and powerful capabilities, is another essential tool. When combined with libraries like IfcOpenShell (an open-source library), Python can parse IFC files, allowing for data extraction and interaction with IFC models.

3. Middle Step (Intermediary Format): In some cases, an intermediary format is required for further processing. For example, when working with Solibri, the output might be in CSV format. This file would need to be adjusted and formatted appropriately to proceed to the next stages.

4. Query Generation for Graph Database: The final step involves generating queries for the graph database using Cypher. This step is crucial for analyzing the extracted data in the graph database structure, allowing efficient querying and analysis of the relationships between elements.

Each of these steps contributes to the seamless transformation of building data into a graph structure, enabling advanced analysis and project insights.

#### 4.4 IFC Meta Data Graph

To generate the IFC Meta Graph (IMG) for BIM data analysis, the process begins with parsing the IFC XML representation through a Python script. This script reads the XML structure and identifies the necessary classes, attributes, and relationships, following the buildingSMART standards. The script creates entity relationships, focusing on entities such as `IfcElement` and their associated properties.

Once relationships are mapped, Cypher queries are written to generate nodes and edges for the graph database. These queries define each class as nodes and connect them with their appropriate edges based on parsed relationships. A second Python script then executes these Cypher queries, automating the import into the graph database and establishing a structured meta graph. This structured setup in a graph database enables detailed querying, facilitating advanced project analysis and cross-disciplinary coordination. Figure below illustrates the described workflow.

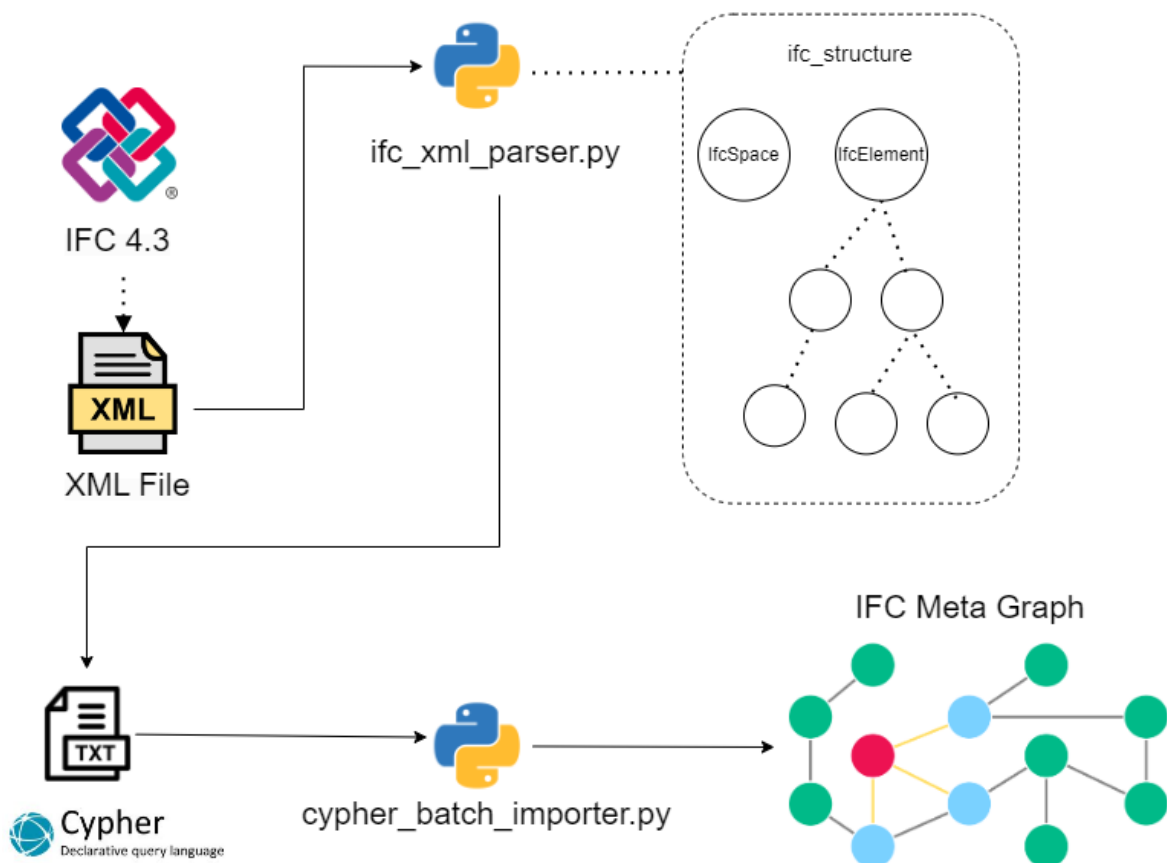


Figure 24 IFC Object Graph Workflow

By automating the node and relationship generation, this workflow not only saves time but also ensures precision in representing the IFC schema in a graph-based format. This approach is also flexible for possible changes in the future, because it can be used for all IFC schemas with XML files.

#### 4.5 BIM Management Meta Data Graph

The BIM Management Meta Data graph, developed with insights from AEC3 Deutschland GmbH, focuses on defining project evaluation criteria and connecting object graphs through fulfillment or containment relationships. These connections ensure that the data flows seamlessly between the meta and object levels, facilitating accurate evaluations of project components. The use cases, as defined by BIM Deutschland, serve as the core element in this structure, guiding the design and integration of the graph's relationships. Main Use Cases that fit into this context can be 050 Coordination of Disciplines.



Figure 25 BIM-M Meta Graph Model Check Criteria Node

Evaluation criteria ensure the models can efficiently support the defined project purposes. Expanding this framework allows for further integration of specific industry knowledge and requirements, making it adaptable to varying project needs. This framework is illustrated in Figure 29.

The BIM-M Meta graph is structured around a central node representing the Model Check Criteria, which serves as the focal point for the evaluation process. From this central node, the relationship "consists\_of" extends to various evaluation criteria, linking them directly to the model's core objectives. These evaluation criteria form the foundation for assessing the quality and integrity of the BIM models. Additionally, the



Figure 26 Use Cases Graph Implementation

"fulfills" relationship ties the central node to a Use Case Parent node, reflecting the connection between the defined criteria and the specific use cases that the BIM models are designed to address. This structure allows for a clear and organized method of linking model quality evaluations to real-world use cases, ensuring that the BIM models meet the necessary requirements for project goals and use-case applications. Used Node labels and relationship labels can be seen in the figure below.

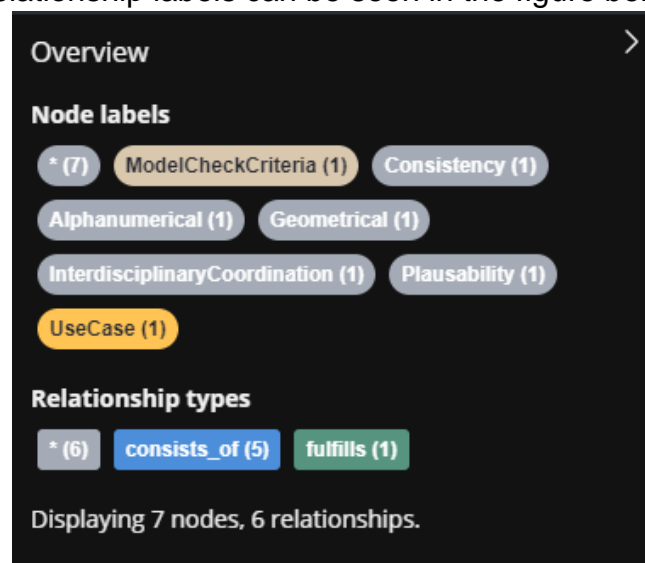


Figure 27 Node Labels and Relationship Labels that are connected to the Model Check Criteria



The BIM Use Cases are also incorporated into the BIM-M Meta graph, allowing the connection of instances from the object graphs to each corresponding use case. This linkage provides a framework to evaluate how specific elements within the object graphs impact the fulfillment of particular use cases. By modeling the use cases within the meta-graph, the structure enables the identification and analysis of how individual components influence project objectives. Figure 31 illustrates this mapping. Figure 32 illustrates how the BIM-M Graph is structured entirely.

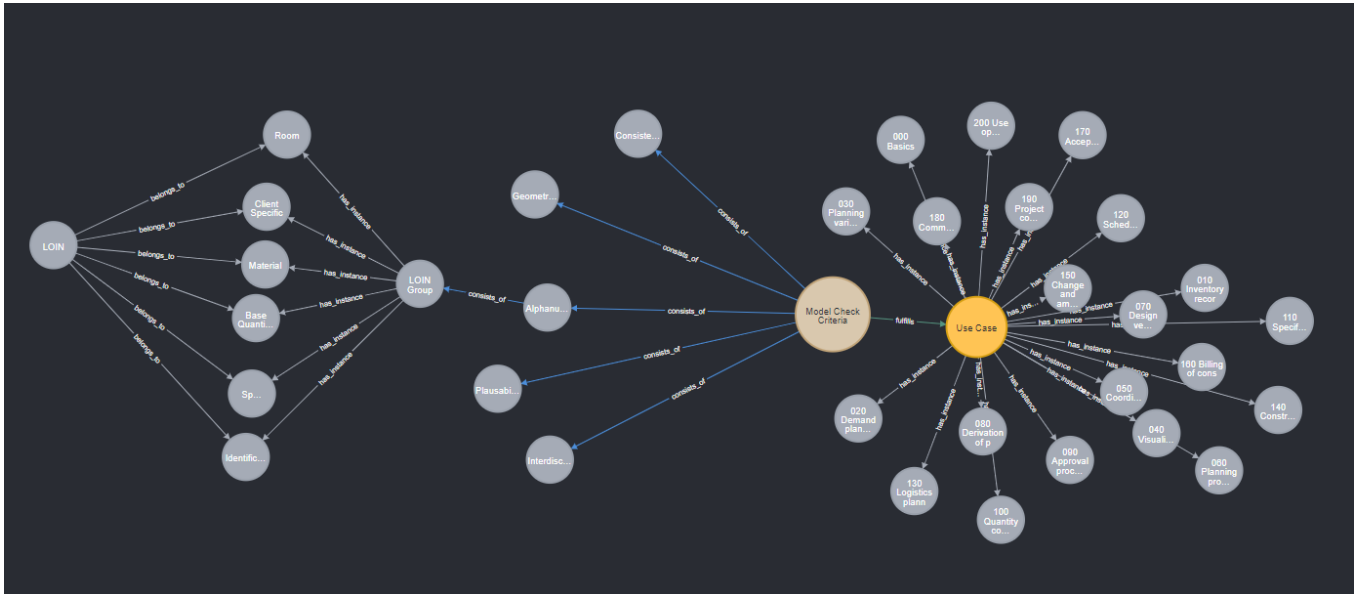


Figure 28 BIM-M Meta Graph

#### 4.6 IFC Object Graphs

For the object graphs derived from the IFC models, the Ruby script from ifcwebserver was used as a basis. Although the script effectively handles the conversion of individual IFC models into the desired formats, several adjustments were necessary to meet the goals of the Interdisciplinary Conjunction Graph (ICG). These modifications primarily involved further parameterization of the created nodes and relationships to ensure a specific organizational structure. This enhanced structuring allows the graph to capture the intricate relationships between various discipline-specific models, facilitating better coordination and analysis across different domains. Additionally, it helps tailor the graph data model to the unique requirements of the project's evaluation and coordination processes.

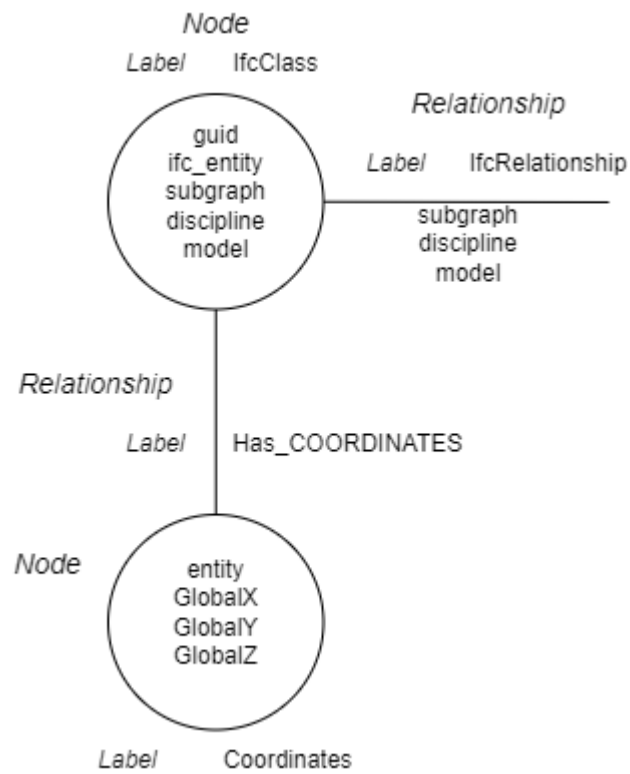


Figure 29 Node and Relationship Structure with main attributes

The figure above explains the desired entity, relationship and coordinate node structure. This structure will not only help to make it easier for certain queries to function but also make it easier for the AI related processes to function.

The figure below shows the developed workflow to generate and import the IFC Object Graphs (IOG) to the ICG. A parsing script is developed to parse the IFC Model utilizing IfcOpenShell. This script brings the data from the IFC Model to the desired structure. Cypher queries are generated within this structure. These queries are outputted into a TXT file and then another Python script reads these TXT files and imports the Cypher commands into the generated Interdisciplinary Conjunction Graph in neo4j.

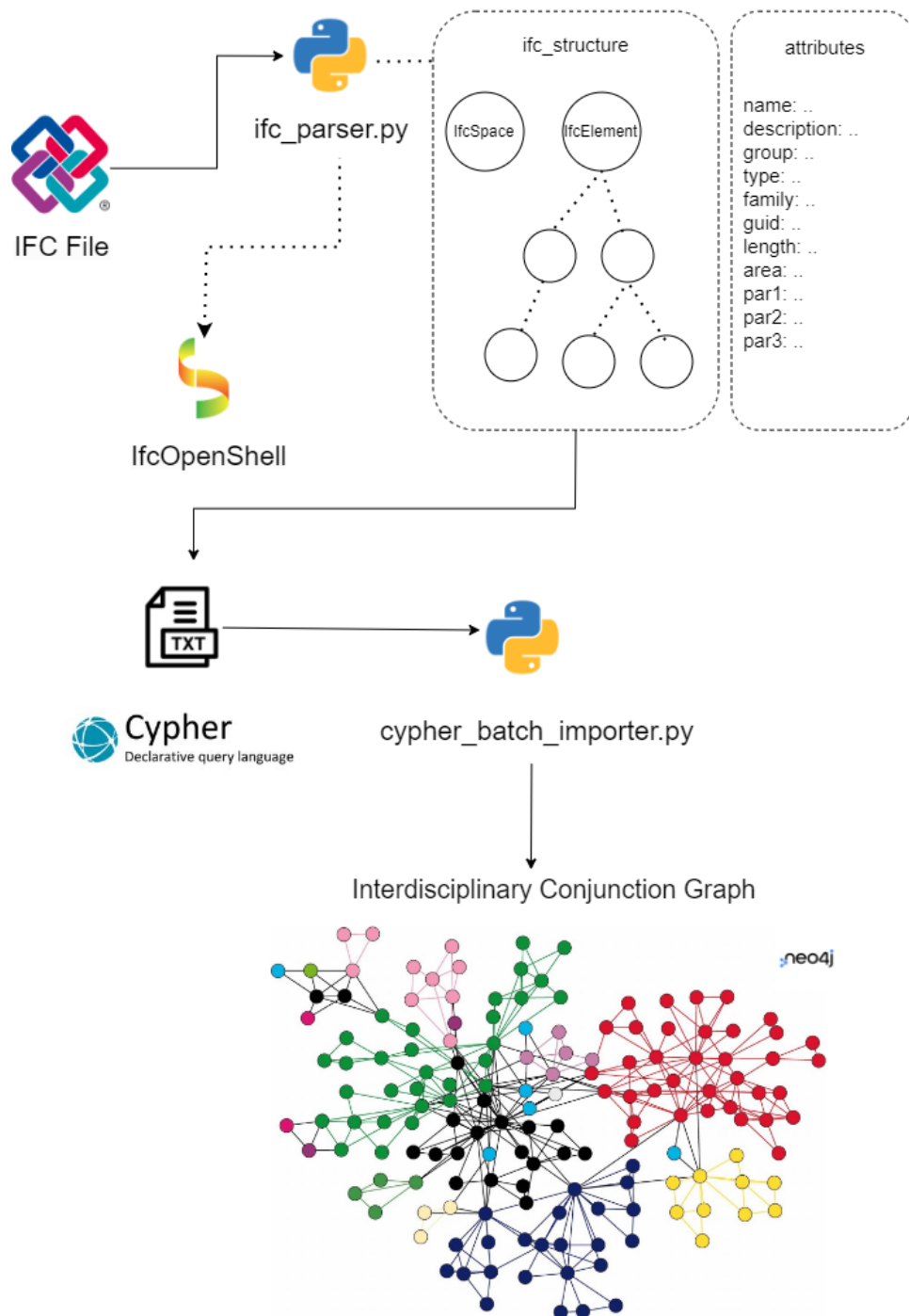


Figure 30 IFC Object Graph Generation Workflow

## 4.7 Semantic Graph Enrichment

Semantic Enrichment workflow varies depending on the goal of the enrichment process. The defined Enrichment processes are listed as the following:

1. **IFC Meta Graph → IFC Object Graphs**
2. **IFC Object Graphs → IFC Object Graphs**
  - **Room\_Contains**
    - Architecture Subgraph → MEP Subgraph
    - Architecture Subgraph → Structural Subgraph

### 1. IFC Meta Graph → IFC Object Graphs

This process is executed by linking the IFC class of instances from the IFC Object Graph (IOG) to the IFC Meta Graph. A direct Cypher query can be employed for this, with the class labels from the IFC Meta Graph and the `ifcClass` attributes from the IOG serving as primary indicators. This establishes the foundation for connecting high-level schema data to instance-level objects.

### 3. Room\_Contains (Architecture Subgraph → MEP Subgraph)

This relationship is created using Solibri Collision Detection tests. By leveraging the GUID of the rooms, objects that collide with the rooms can be linked back to the room itself. This is particularly important for defining spatial containment between architectural and MEP components, ensuring accurate representation of system interactions.

### 4. Room\_Contains (Architecture Subgraph → Structural Subgraph)

Similar to the MEP Subgraph enrichment, this relationship also utilizes Solibri Collision Detection. The room GUID is used to link structural objects that intersect or are contained within specific rooms, facilitating cross-disciplinary coordination between structural and architectural components.

In the figure below, the 'ROOM\_CONTAINS' relationship is illustrated. The models are loaded into a Solibri file, generating the Coordination Model. A Crash test will be executed to get the items inside the IfcSpace elements. These will be outputted into a bcfzip file. A parser for IfcSpace is written to get the GUIDs of IfcSpace elements from Architecture Model. A BCF parser will process the viewpoint xml files from the bcfzip file and run it against the list of GUIDs from the IfcSpace parser file to match the Rooms and other objects. The results will be processed into Cypher queries in a text file. Then this text file will be read using another importer Python script to enhance the Interdisciplinary Conjunction Graph.

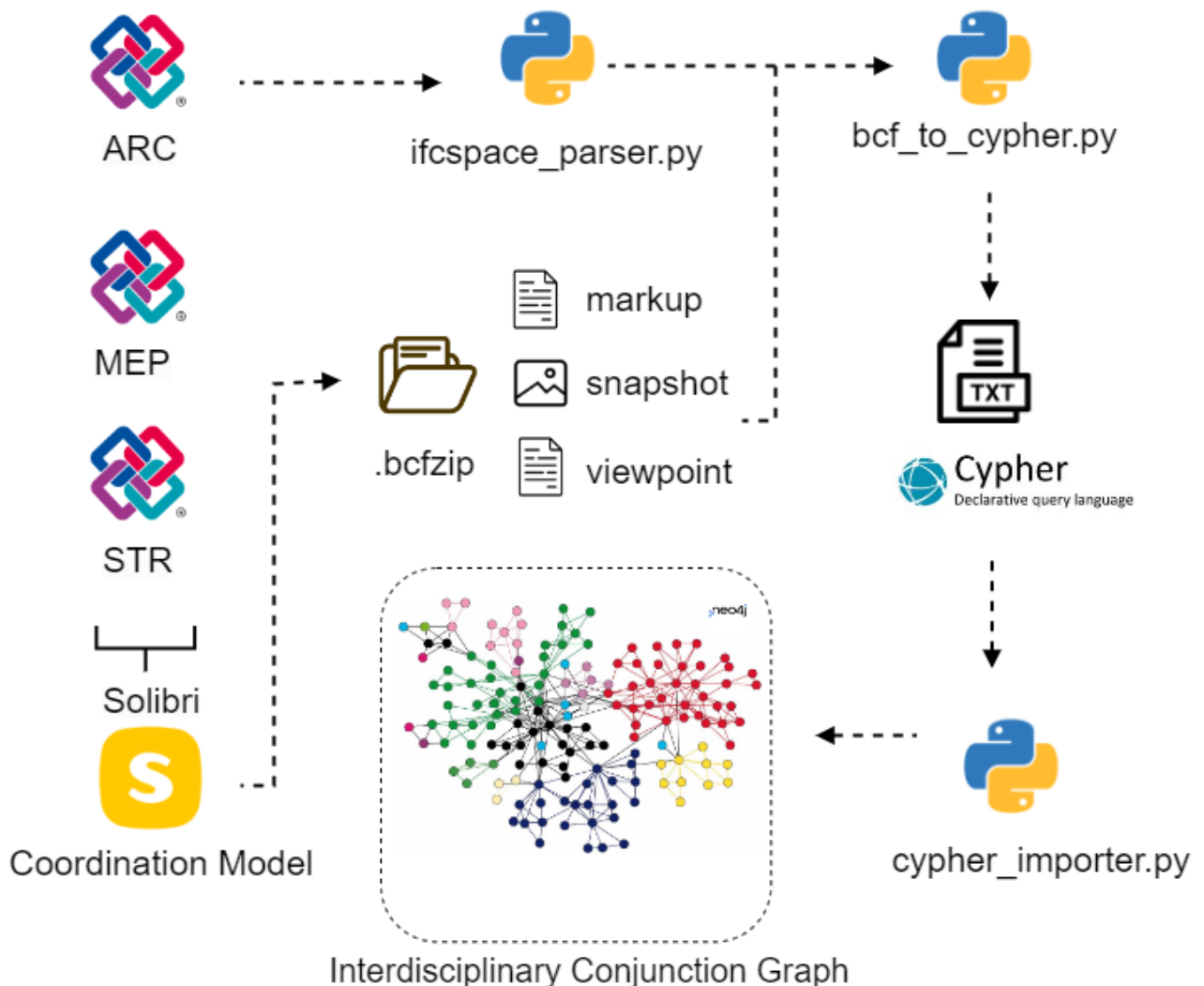


Figure 31 ROOM\_CONTAINS Semantic Enrichment Workflow

## 4.8 Graph Database Workflow

In preparation for applying Graph Machine Learning techniques, the structure of the knowledge graph is carefully designed to support link prediction algorithms. The relationships within the graph are organized to enable predictions that enhance coordination in the BIM process. By structuring the graph with this aim, link predictions can effectively identify and strengthen critical connections, ultimately improving project coordination and alignment with BIM objectives.

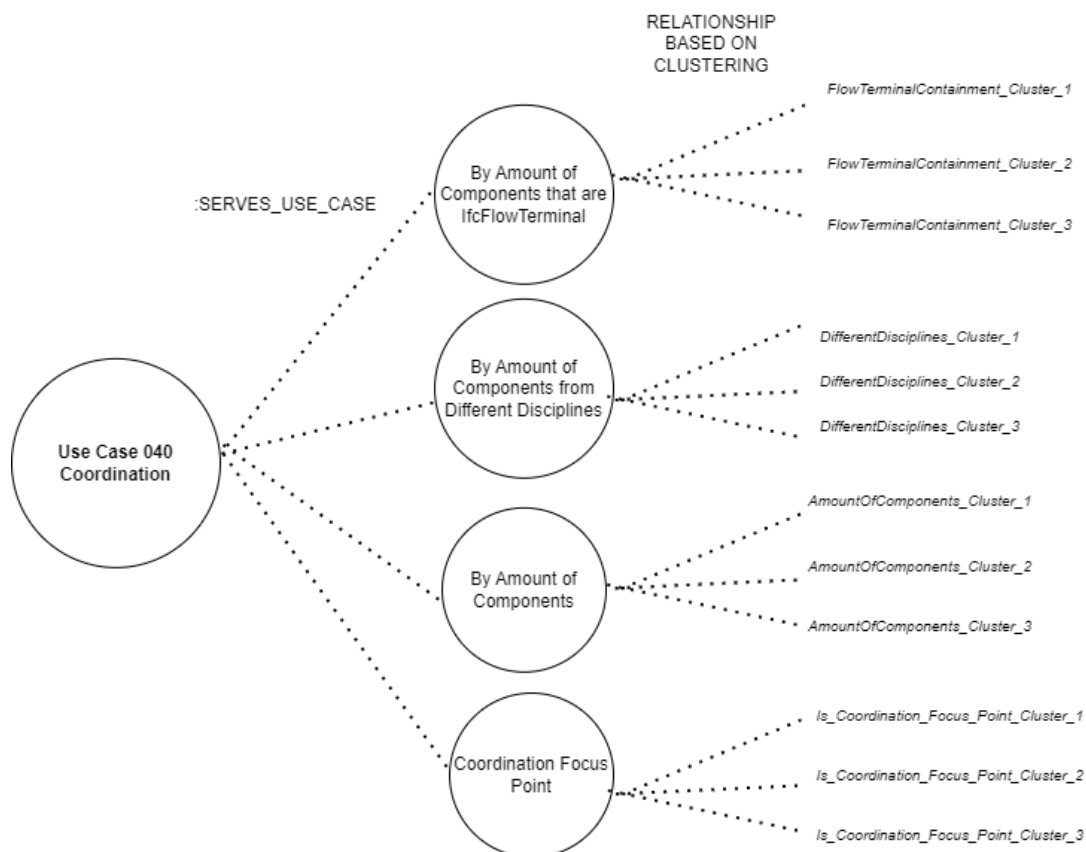


Figure 32 Use Case 040 Coordination Enhancement Schema

As illustrated in the figure above, Use Case 040 (Coordination) from the BIM-M Meta Graph, as defined by BIM Deutschland, is linked to four distinct nodes representing key criteria for evaluating discipline coordination. By leveraging “ROOM\_CONTAINS” relationships associated with IfcSpace components in the graph, these criteria are effectively established to guide project evaluation and improve interdisciplinary alignment.

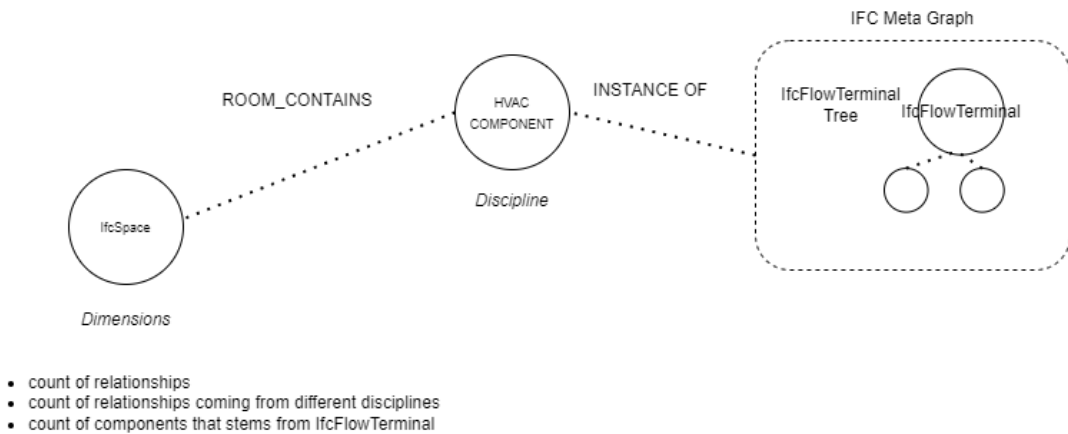


Figure 33 Schema Enhancement based on ROOM\_CONTAINS

A Python script, executed in a Jupyter Notebook, will be used to analyze the graph according to specified criteria. Using Python’s Scikit-learn library, the results will be clustered based on three main criteria:

- Count of relationships
- Count of relationships coming from different disciplines
- Count of components that stem from IfcFlowTerminal

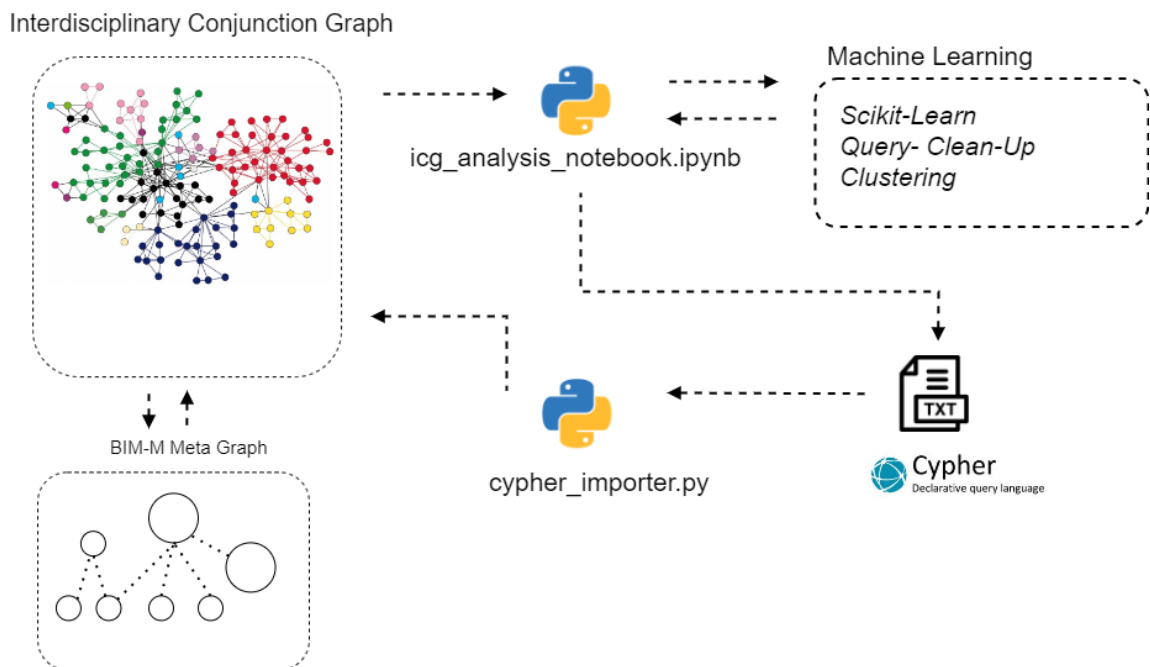


Figure 34 ICG Analysis Workflow

These criteria, along with room sizes, are integrated into a scoring system expressed as counts per square meter. Using this system, a final metric, “Coordination Focus Point,” is developed, where a score of 1 indicates the highest intensity of coordination needs. IfcSpace components are grouped into three levels based on this metric, identifying spaces that require intensive coordination. This classification aims to inform and improve project decisions by highlighting areas where focused coordination is essential. The figure above illustrates the process.

## 4.9 Graph Neural Network and Graph Based Machine Learning

### 1. Exporting Data from Neo4j to CSV

To begin, we export the relevant graph data stored in the Neo4j database into CSV files for further analysis. This involves querying the graph to retrieve two key components:

- **Node Data:** We export the attributes of the nodes in the graph. These nodes represent building components such as spaces, walls, and doors. For each node, we extract essential properties like its ID (GUID), discipline (such as architecture or MEP), NetFloorArea, and spatial coordinates (GlobalX, GlobalY, GlobalZ). These attributes are stored in a CSV file for further processing.
- **Edge Data:** Next, we export the edges connecting the nodes, which represent the relationships between the components. For each edge, we extract the source and target nodes, the relationship type (such as "contained in", "hosts"), and additional attributes like subgraph information. These edges are stored in another CSV file for training the model.

The exported CSVs provide structured data that can be easily loaded into **pandas** DataFrames for further processing.

### Graph Representation and Node Feature Preparation

Once the data is exported, we transform it into a graph representation. Each row in the node CSV represents a building component, and each row in the edge CSV represents the relationship between two components. We prepare the node features by extracting and normalizing the relevant attributes, such as the NetFloorArea, discipline, and spatial coordinates. These features will serve as input to the GraphSAGE model.



## 2. GraphSAGE for Edge Prediction

To predict the relationships between IfcSpace nodes and specialist trades coordination nodes, we apply GraphSAGE (Graph Sample and Aggregation). GraphSAGE is a type of graph neural network (GNN) that aggregates information from neighboring nodes to generate embeddings. These node embeddings are then used to predict edge types.

- **Node Embeddings:** GraphSAGE learns node representations by aggregating the features of neighboring nodes. These embeddings are used to capture the spatial and functional relationships between building components.
- **Edge-Level Prediction:** Instead of predicting node labels, GraphSAGE is used to predict the relationship between two nodes (i.e., the edge), which is the core task. The model aggregates the source and target node features and uses them to classify the relationship between the two connected nodes.

## 3. Feature and Label Setting for Training

The features for training are derived from the node attributes of the building components represented in the graph. Each node corresponds to a building element, such as IfcSpace, IfcWall, or IfcDoor, and its features include geometric and semantic properties like NetFloorArea, discipline, Clean\_name, and spatial coordinates (GlobalX, GlobalY, GlobalZ). These features are collected from the IFC model and preprocessed to form a unified feature vector for each node. For edge-level prediction the labels are defined based on the type of relationship between connected nodes. Specifically, the edges represent the interactions between building components, and each edge is labeled with a binary class that indicates the presence or absence of a specific type of relationship (e.g., whether an edge represents a "040 Coordination of Specialist Trades" relationship). These labels are used as the target for the GraphSAGE model, where the goal is to predict the relationship type for each edge.

## 4. Negative Sampling in GraphSAGE

GraphSAGE uses a negative sampling technique during training to improve the model's ability to distinguish between true and false relationships. In the context of edge prediction, the model learns to classify whether an edge between two nodes represents a meaningful relationship or not. Positive samples (i.e., edges representing true relationships) are directly taken from the graph, where each edge connects two nodes with an actual relationship. Negative samples, on the other hand, are generated by randomly sampling pairs of nodes that do not share an edge. These pairs are treated as non-edges and are assigned a label of 0, indicating the absence of a relationship. The model is trained to distinguish between these positive and negative samples, ensuring that it learns to correctly predict true relationships while avoiding false connections. Negative sampling helps to balance the training process, preventing the model from becoming biased toward predicting positive relationships, which can be especially important when the graph contains a high number of negative or missing edges.

## 5. Model Training and Evaluation

We train the GraphSAGE model using the node features and edge indices. During training, we optimize the model with Binary Cross-Entropy Loss, given that we are predicting whether a given edge corresponds to a specific relationship. We use mini-batching to manage memory usage efficiently and avoid overloading the system with large graphs.

## 5. Case Study

For the case study, a coordination model with several IFC models has been made available. This model contains an office building with following models. These models are also loaded into a Solibri file as can be seen in the figure below.

Model Name	Discipline
SCG_AR_optimized_2-withPositionCorrection	Architecture
SCG_ST	Structural
SCG_AA	Landscape
SCG_DL	Ventilation
SCG_E0	Electrical Appliances
SCG_EL	Electric
SCG_HK	Heating/Cooling
SCG_LB	Laboratory Equipment
SCG_LU	Ventilation
SCG_RB	Structural
SCG_SA	Plumbing
SCG_SB	Steel Construction
SCG_SP	Sprinkler

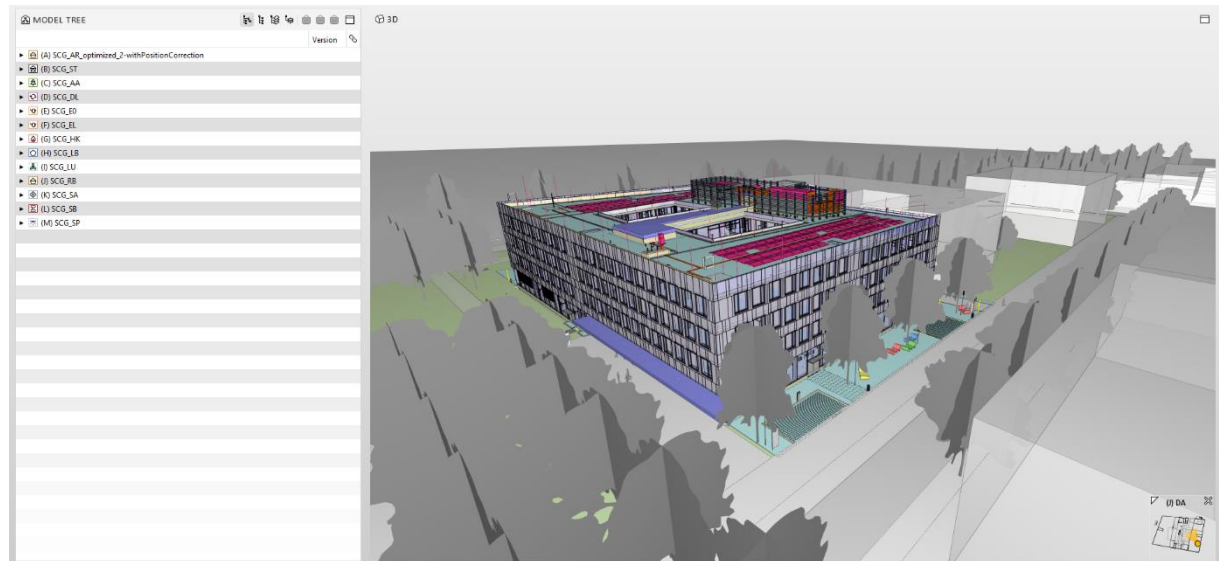


Figure 35 Screenshot from the Coordination Model

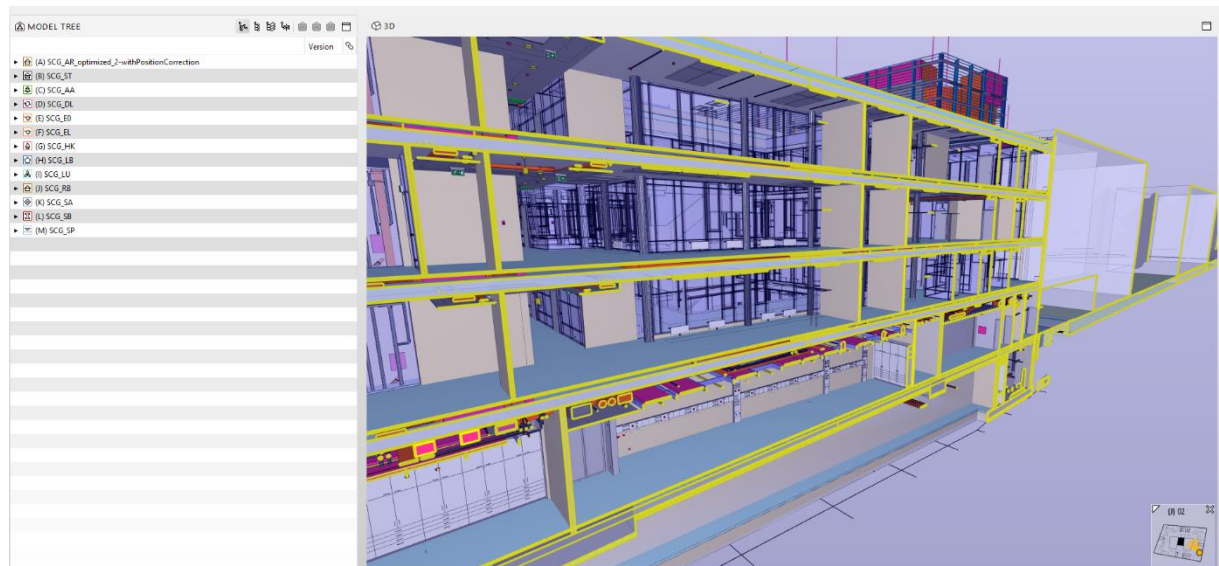


Figure 36 Section from the Coordination Model

## 5.1. IFC Meta Graph Generation

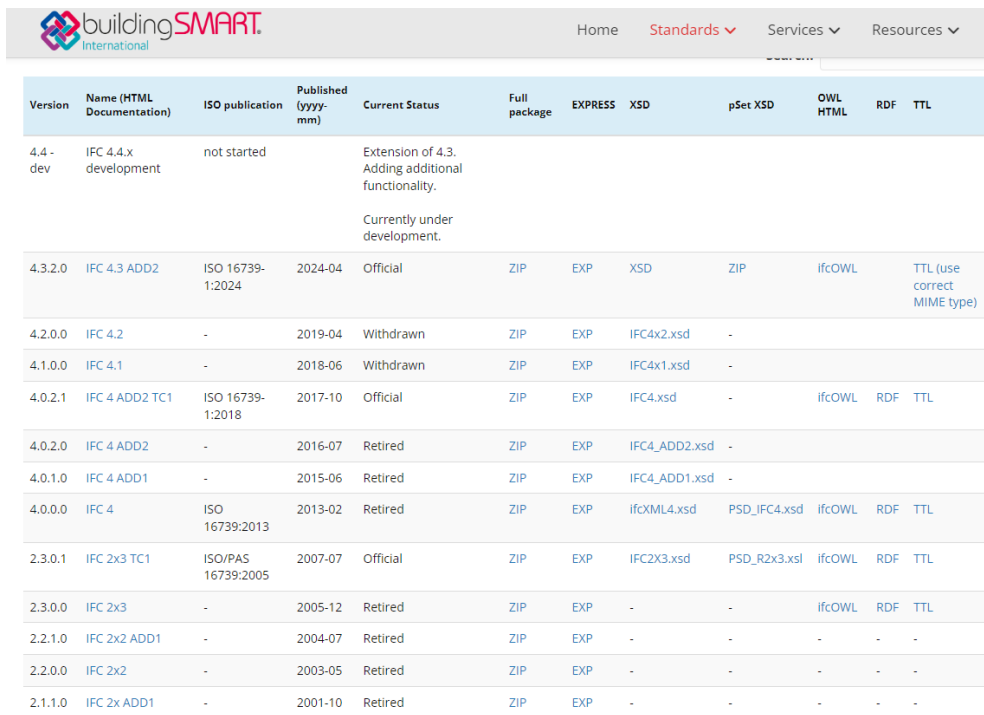
The creation of the IFC Meta Graph (IMG) adheres to the standards established by buildingSMART, the organization responsible for the development and maintenance of the IFC standard. buildingSMART provides detailed documentation of the IFC class schema and the corresponding relationships, publishing these schemas for each IFC version in multiple formats on their official website.

For the development of the IFC Meta Graph, IFC 4 ADD2 TC1 was selected. This version represents the official ISO-approved standard and is currently one of the most widely adopted versions in the AEC industry. The choice of this version aligns with current industry practices, as it offers improved support for various disciplines and is

optimized for the latest developments in BIM workflows. Additionally, the models that will be used in the case study are also based on IFC 4, further supporting the decision to use this version for the meta graph creation.

By using IFC 4 ADD2 TC1, the meta graph ensures compatibility with widely accepted standards while capturing the comprehensive relationships and attributes inherent in modern building information models. This will facilitate a seamless analysis and enable the use of the latest tools and methodologies in the case study.

The production of the IFC Meta Graph follows the standards defined by buildingSmart. Resources published by buildingSmart indicate the Class schema and relevant relationships. BuildingSmart also publishes the schemas of each version in various formats on their website. For the meta graph creation version IFC 4 ADD2 TC1 has been chosen. This version is the official ISO approved standard and it's being widely used in the industry at the moment. The models will be used in the case study are also in IFC4 which supports this decision. (buildingSmart, n.d.)



Version	Name (HTML Documentation)	ISO publication	Published (yyyy-mm)	Current Status	Full package	EXPRESS	XSD	pSet XSD	OWL HTML	RDF	TTL
4.4 - dev	IFC 4.4.x development	not started		Extension of 4.3. Adding additional functionality. Currently under development.							
4.3.2.0	IFC 4.3 ADD2	ISO 16739-1:2024	2024-04	Official	ZIP	EXP	XSD	ZIP	ifcOWL		TTL (use correct MIME type)
4.2.0.0	IFC 4.2	-	2019-04	Withdrawn	ZIP	EXP	IFC4x2.xsd	-			
4.1.0.0	IFC 4.1	-	2018-06	Withdrawn	ZIP	EXP	IFC4x1.xsd	-			
4.0.2.1	IFC 4 ADD2 TC1	ISO 16739-1:2018	2017-10	Official	ZIP	EXP	IFC4.xsd	-	ifcOWL	RDF	TTL
4.0.2.0	IFC 4 ADD2	-	2016-07	Retired	ZIP	EXP	IFC4_ADD2.xsd	-			
4.0.1.0	IFC 4 ADD1	-	2015-06	Retired	ZIP	EXP	IFC4_ADD1.xsd	-			
4.0.0.0	IFC 4	ISO 16739:2013	2013-02	Retired	ZIP	EXP	ifcXML4.xsd	PSD_IFC4.xsd	ifcOWL	RDF	TTL
2.3.0.1	IFC 2x3 TC1	ISO/PAS 16739:2005	2007-07	Official	ZIP	EXP	IFC2X3.xsd	PSD_R2x3.xsl	ifcOWL	RDF	TTL
2.3.0.0	IFC 2x3	-	2005-12	Retired	ZIP	EXP	-	-	ifcOWL	RDF	TTL
2.2.1.0	IFC 2x2 ADD1	-	2004-07	Retired	ZIP	EXP	-	-	-	-	-
2.2.0.0	IFC 2x2	-	2003-05	Retired	ZIP	EXP	-	-	-	-	-
2.1.1.0	IFC 2x ADD1	-	2001-10	Retired	ZIP	EXP	-	-	-	-	-

Figure 37 IFC Schema Listings from buildingSmart

An automated workflow has been established to streamline the creation of the IFC Meta Graph. This workflow utilizes a custom Python script designed to fetch the XSD (XML Schema Definition)

file published by buildingSMART. The XSD file serves as a structured representation of the IFC schema.

Example XML Snippet from the IFC4 Schema.

```
1.      <xs:element          name="IfcWall"          type="ifc:IfcWall"
substitutionGroup="ifc:IfcBuildingElement" nillable="true"/>
2.      <xs:complexType name="IfcWall">
3.          <xs:complexContent>
4.              <xs:extension base="ifc:IfcBuildingElement">
5.                  <xs:attribute name="PredefinedType"
type="ifc:IfcWallTypeEnum" use="optional"/>
6.              </xs:extension>
7.          </xs:complexContent>
8.      </xs:complexType>
```

Once the XSD file is fetched, the Python script processes it by iterating over its contents to create nodes that correspond to the various IFC classes. In addition to generating these class nodes, the script defines and assigns the "supertype\_of" relationship, which reflects the inheritance structure present within the schema. This relationship is crucial for accurately modeling the hierarchy and connections between different IFC classes, ensuring that the graph retains the rich structural information embedded in the IFC schema. Additionally, another logic has been added to the script to handle the abstract classes and filter only the instantiable classes.

Python Script to generate Cypher Queries for IMG (see Appendix)

Another challenge encountered in this workflow is the excessive complexity of the IFC schema, leading to an emerging need for its simplification. Parsing the XML file without applying any form of filtering results in unmanageable and overwhelming data outputs. Therefore, refocusing on project evaluation and specific project goals becomes essential at this stage. Given that HVAC components and interdisciplinary coordination are the focus of this research, the `IfcElement` class and its sub-nodes have been selected as the primary class tree. Additionally, `IfcRoot`, `IfcBuildingStorey`, and `IfcSpatialStructure` are included and linked to the `IfcElement` tree to capture the relevant relationships within the object graphs. This targeted selection allows the workflow to focus on the essential components and relationships while filtering out

unnecessary complexity, making it more manageable and aligned with the project's objectives.

The figure illustrates the class hierarchy until the IfcElement tree within the Meta Graph. The Figure 29 explains the entity inheritance schema within the IFC Structure. To capture the relevant relationships of the IfcElement Tree, this schema is carefully examined. The interpretation of the hierarchy tree can be seen on Figure 30. In this Figure we are seeing the graph made from the Ifc Element tree and its corresponding relationships with its subclasses.

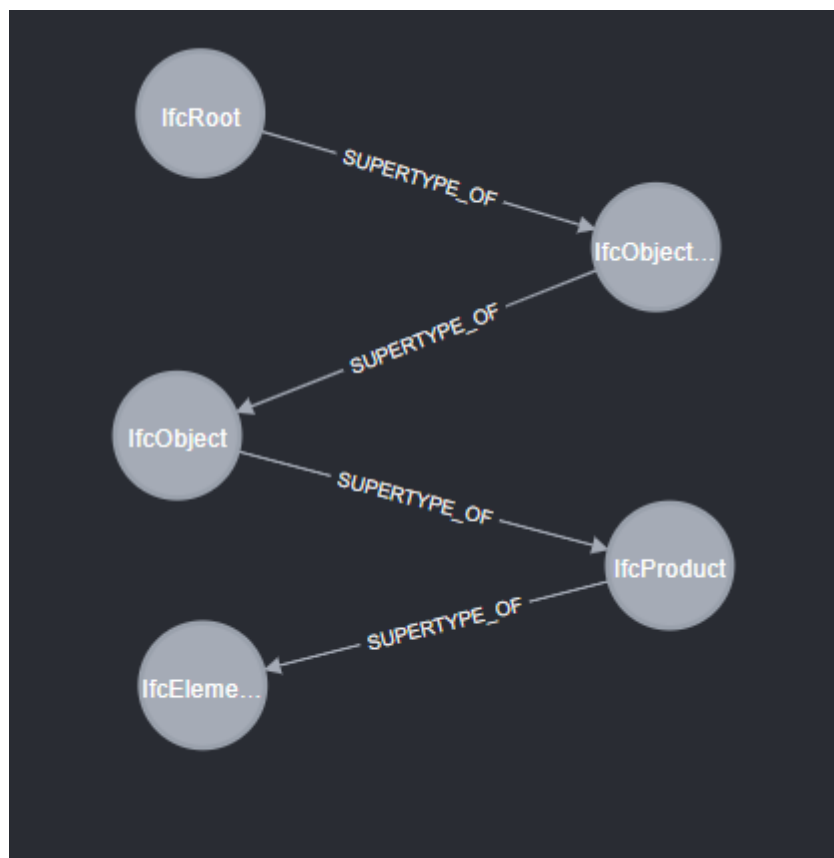


Figure 38 IFC Hierarchy in Graph Structure

### 5.4.3.18.2 Entity inheritance [↗](#)

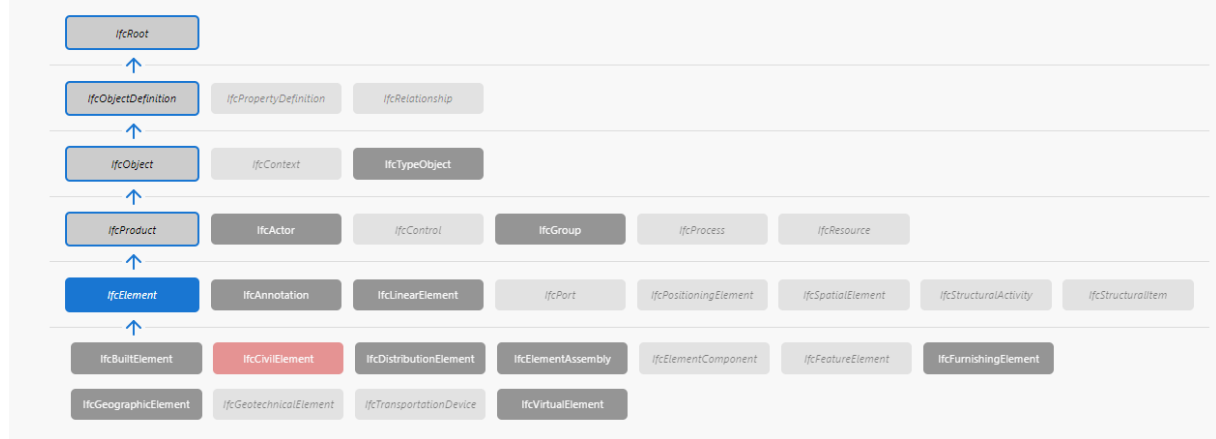


Figure 39 Class Structure of IfcElement Tree

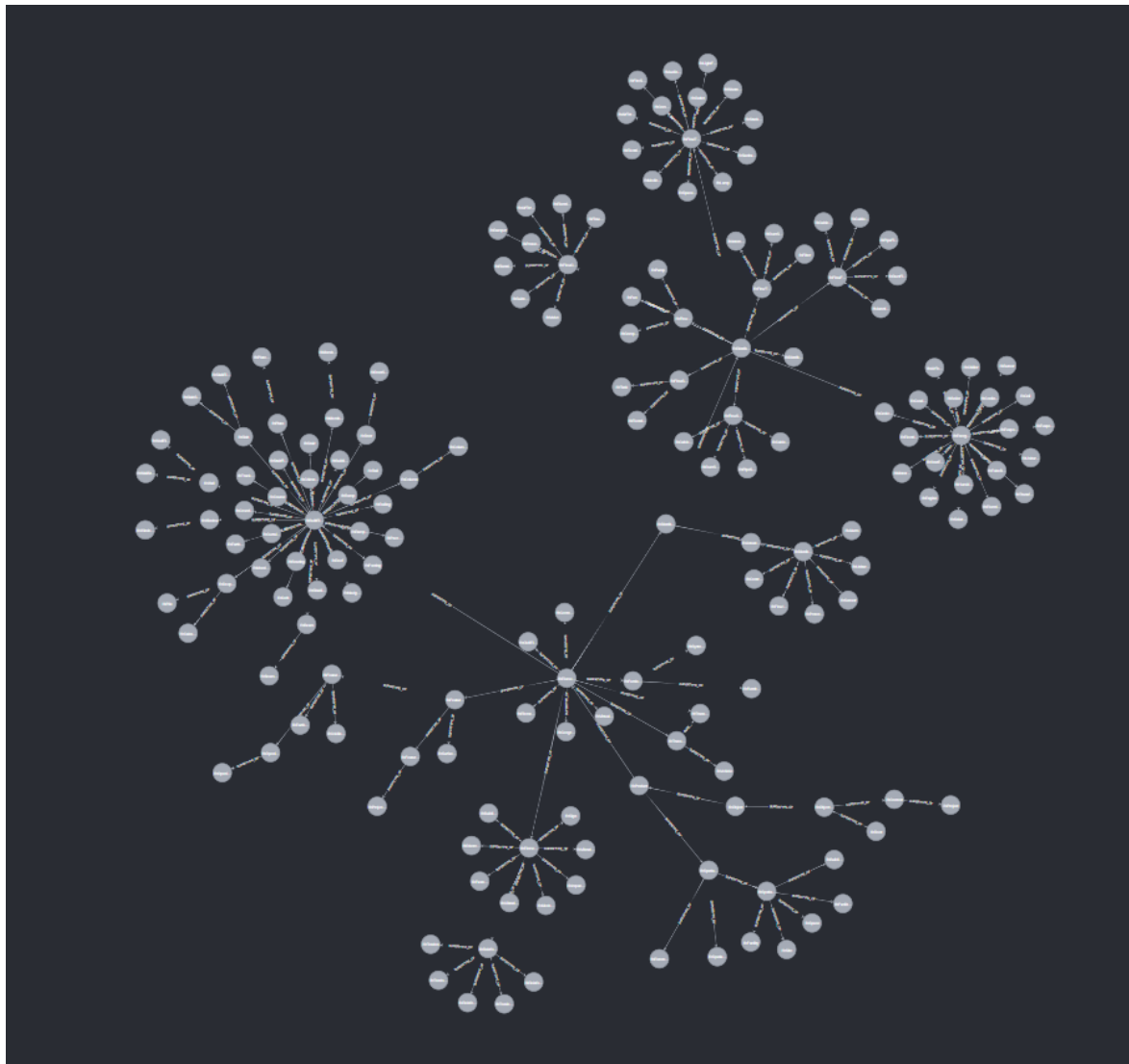


Figure 40 Sub-Class Structure of IfcElement



## 5.2. IFC Object Graphs Generation

The object graphs for each model has been generated through the developed methodology.

```
1. CREATE (elem:IfcWall {
2.     id: "unique-id",
3.     guid: "global-id",
4.     ifc_entity: "IfcWall",
4.     name: "Wall Name",
5.     subgraph: "Architecture Subgraph",
6.     discipline: "Architecture",
7.     properties: {"Height": "2.5", "Width": "0.3", "FireRating": "A1"}
8. });
9.
```

The desired outcome of the script for IFC Entities inside the model.

The ruby script for the transformation of IFC Files to neo4j compatible formats by Ali Ismail. <https://github.com/ifcwebserver/IFC-to-Neo4j> (Ismail A. , n.d.)

A python script has been developed to meet these needs. As input, arrays can be used to assign these properties to the generated nodes and relationships. This is also essential for batch imports.

For the Python Script see Appendix.

```
1. # example Array
2. variable_name = ["Model Name", "Subgraph Name", "Discipline"]
3.
```

Due to the massive amount of information coming from IFC files, the script from ifcwebserver needs to be re-adjusted considering the project goals.

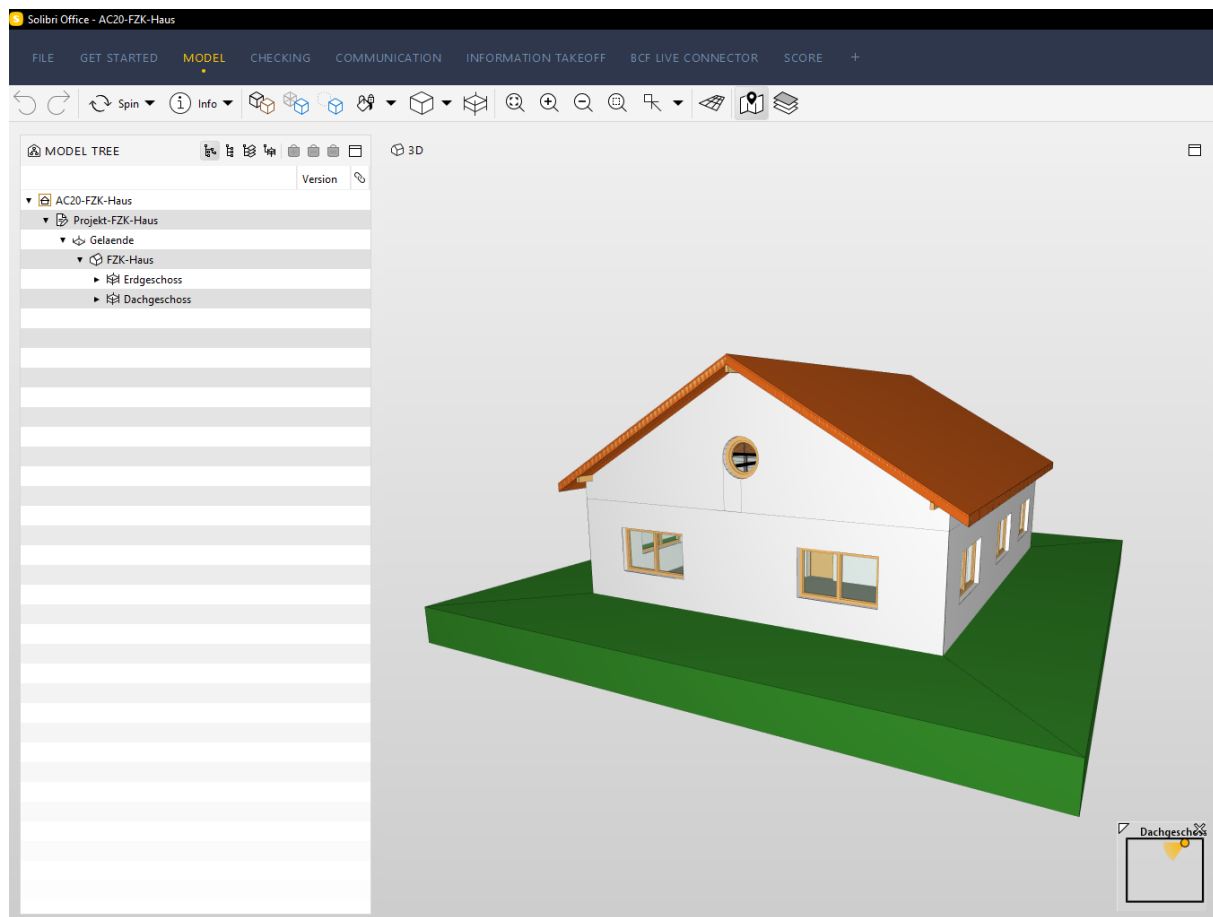


Figure 41 AC3-FZK-Haus.ifc - Test Project

For tests purposes an example project has been taken from Karlsruhe Institute of Technology. AC3-FZK-Haus.ifc is a simple house project in IFC format. (Technology) The script is tested on the FZK House project. These are the results without optimization.

```

D:\MastersThesis\thesis_code_koray\vcg_cypher\AC20-FZK-Haus_import_commands.txt - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
AC20-FZK-Haus_import_commands.txt
11587 MATCH (a), (b) WHERE a.id = "f3833a1d-2c2e-4551-9eb0-87cdae8917dd" AND b.id = "ec098d46-3ddf-4234-be87-d02bef7cb233" CREATE (a)-[:IsDefinedByProperties]->(b);
11588 MATCH (a), (b) WHERE a.id = "90cd8ddf-338a-4e95-90dc-e05822fa3e61" AND b.id = "a3a18921-e5df-4837-86b4-ea29630896cb" CREATE (a)-[:VoidsElements]->(b);
11589 MATCH (a), (b) WHERE a.id = "78128715-7e8c-423f-afd0-67c8165f9b3c" AND b.id = "a3a18921-e5df-4837-86b4-ea29630896cb" CREATE (a)-[:VoidsElements]->(b);
11590 MATCH (a), (b) WHERE a.id = "d2d9482f-e9a8-4350-bc90-bdf4c4efef493" AND b.id = "99b04ece-9598-45b4-ae9f-fec3f7fba51a" CREATE (a)-[:IsDefinedByProperties]->(b);
11591 MATCH (a), (b) WHERE a.id = "d25e6020-1bac-47fd-a726-897dc2453f87" AND b.id = "d9f809a9-1da4-4c89-a04b-81c5fbb773c6" CREATE (a)-[:IsDefinedByProperties]->(b);
11592 MATCH (a), (b) WHERE a.id = "471a9428-6f71-47c2-8958-c8c222486e41" AND b.id = "39a9945d-038f-4d2d-b12b-665aa6dea0b5" CREATE (a)-[:IsDefinedByProperties]->(b);
11593 MATCH (a), (b) WHERE a.id = "779ba921-3ce3-49f5-bb82-d63cb27d33d2" AND b.id = "113d7947-3d6e-4d18-b661-327ab5b148ca" CREATE (a)-[:IsDefinedByProperties]->(b);
11594 MATCH (a), (b) WHERE a.id = "23c9f85a-c9d6-4ce4-b54b-0442e7ce7856" AND b.id = "385ca85c-bda3-49e2-880b-7918545b9e1b" CREATE (a)-[:IsDefinedByProperties]->(b);
11595 MATCH (a), (b) WHERE a.id = "80f728a9-843b-425b-9152-698bcf9de228" AND b.id = "16c4e24c-dcee-45f2-8b01-f90c82d11c12" CREATE (a)-[:IsDefinedByProperties]->(b);
11596 MATCH (a), (b) WHERE a.id = "f8e1e171-44b9-49f8-ab8d-012a2c0371a5" AND b.id = "506ec90d-6c36-461f-97d4-7d5b0181fc09" CREATE (a)-[:IsDefinedByProperties]->(b);
11597 MATCH (a), (b) WHERE a.id = "70d9a35f-443b-440e-bc59-afdc5282de4d" AND b.id = "c32f066b-6692-4d7a-b7a3-4f4713b3f242" CREATE (a)-[:IsDefinedByProperties]->(b);
11598 MATCH (a), (b) WHERE a.id = "9b344849-b46f-4dd7-816d-8e0017bfb11d" AND b.id = "8bc4d9f6-e089-4d69-bdd4-70cc74e2b37a" CREATE (a)-[:IsDefinedByProperties]->(b);
11599 MATCH (a), (b) WHERE a.id = "779ba921-3ce3-49f5-bb82-d63cb27d33d2" AND b.id = "12341c8b-7408-48b0-95bf-13cca13386bc" CREATE (a)-[:IsDefinedByProperties]->(b);
11600 MATCH (a), (b) WHERE a.id = "23c9f85a-c9d6-4ce4-b54b-0442e7ce7856" AND b.id = "180310d0-46d0-4196-8f4c-258b4d489c09" CREATE (a)-[:IsDefinedByProperties]->(b);
11601 MATCH (a), (b) WHERE a.id = "70d9a35f-443b-440e-bc59-afdc5282de4d" AND b.id = "dc326456-502a-4df3-b6fd-77b78c497291" CREATE (a)-[:IsDefinedByProperties]->(b);
11602 MATCH (a), (b) WHERE a.id = "69c4d7d8-a79d-4619-be9a-4ec80bc94202" AND b.id = "a56cee458-9fbb-4f38-b620-571548e4fc3f" CREATE (a)-[:IsDefinedByProperties]->(b);
11603 MATCH (a), (b) WHERE a.id = "2614d88d-6959-494e-bb7b-bea7bee9877a" AND b.id = "a76dda1c-ba9d-4ef6-9b8f-cf3a00acac1e" CREATE (a)-[:IsDefinedByProperties]->(b);
11604 MATCH (a), (b) WHERE a.id = "31d117b4-6e6b-462d-aa18-849da006726b" AND b.id = "1905dd90-cab9-48ca-878b-26aee33d8035" CREATE (a)-[:IsDefinedByProperties]->(b);
11605 MATCH (a), (b) WHERE a.id = "90cd8ddf-338a-4e95-90dc-e05822fa3e61" AND b.id = "bd752794-1005-4fc9-8772-4bd3f562b54" CREATE (a)-[:BoundedBy]->(b);
11606 MATCH (a), (b) WHERE a.id = "7951b97b-9d4f-4c0a-80db-72312f9df72f" AND b.id = "ccf498e3-2141-47b3-82c8-1035cdfdeb8c" CREATE (a)-[:IsDefinedByProperties]->(b);
11607 MATCH (a), (b) WHERE a.id = "90cd8ddf-338a-4e95-90dc-e05822fa3e61" AND b.id = "69d4580a-e7dc-4859-b01c-704f30eec354" CREATE (a)-[:BoundedBy]->(b);
11608 MATCH (a), (b) WHERE a.id = "5010dfc7-e228-48b5-b4f4-bbc9e7df2950" AND b.id = "69d4580a-e7dc-4859-b01c-704f30eec354" CREATE (a)-[:BoundedBy]->(b);
11609 MATCH (a), (b) WHERE a.id = "779ba921-3ce3-49f5-bb82-d63cb27d33d2" AND b.id = "3298feb-54b1-4388-acdd-b488c0541d11" CREATE (a)-[:Decomposes]->(b);
11610 MATCH (a), (b) WHERE a.id = "20ba6c00-2871-4d01-980c-1b5d4ad4383b" AND b.id = "e1cd1bb4-74a4-4f6a-85ee-18e087a94df7" CREATE (a)-[:IsDefinedByType]->(b);
11611 MATCH (a), (b) WHERE a.id = "58daf0e-c42a-4c99-9b30-b1a6fa6f54dc" AND b.id = "e1cd1bb4-74a4-4f6a-85ee-18e087a94df7" CREATE (a)-[:IsDefinedByType]->(b);
11612 MATCH (a), (b) WHERE a.id = "6217f57d-bdd5-4225-a305-592cfff4e2e43" AND b.id = "0da473a7-37cd-4fb7-aa64-bd8c77065705" CREATE (a)-[:IsDefinedByProperties]->(b);
11613 MATCH (a), (b) WHERE a.id = "f4a9642d-05d5-4fee-91d5-fbae3e7bd8d3" AND b.id = "db532c79-424d-4c1b-8afc-4ee44d130111" CREATE (a)-[:HasAssociations]->(b);
11614

```

Figure 42 Script Output without optimization

As can be observed from the output, a small example project generates over 11,000 lines of code, making it impractical for use in real-life scenarios. The sheer volume complicates management and understanding, hindering effective collaboration and increasing the risk of errors. To achieve more control over the parsing processes and achieve simpler results, own parser/converter script has been developed.

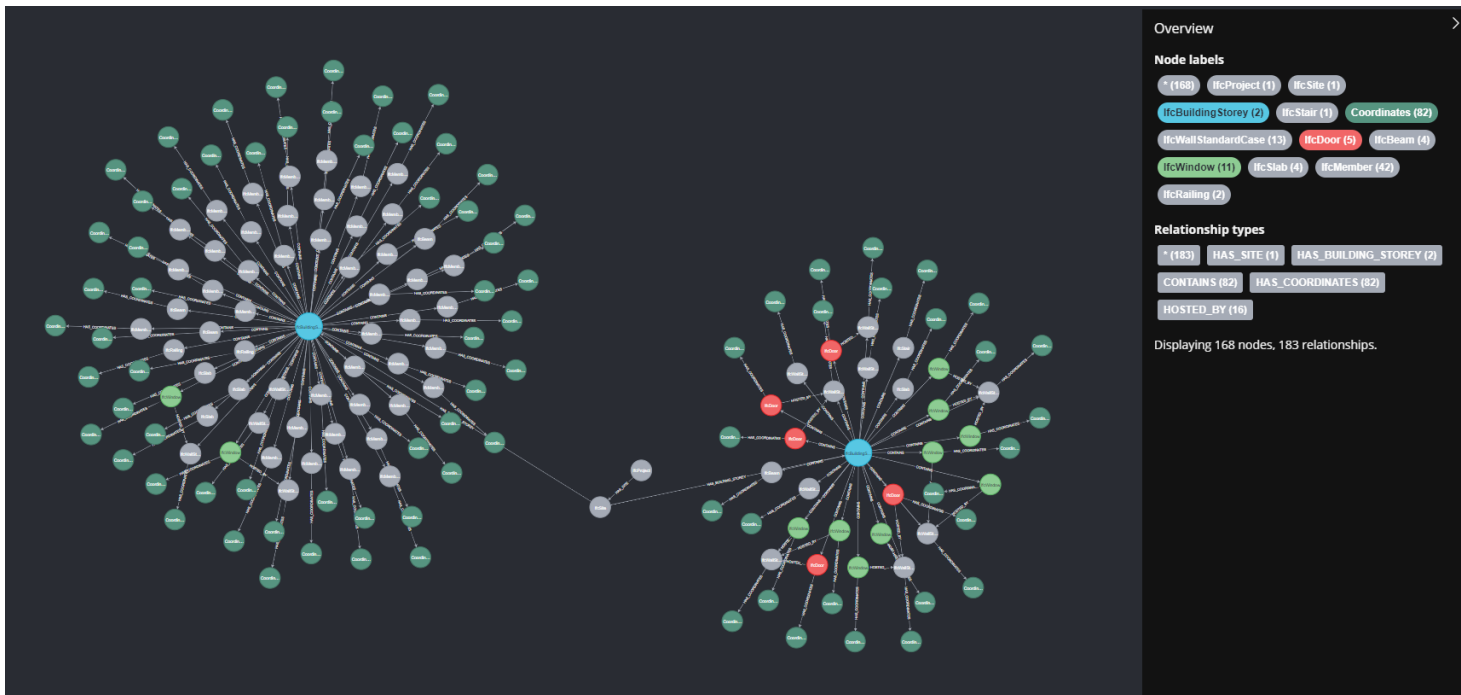


Figure 44 Fzk-Haus Graph Representation with the optimized Python Script

As observed in the figure above, the optimized script produces a cleaner output. Further enhancements can be achieved through enrichment techniques or by directly modifying the script to accommodate additional processes. This approach fully utilizes the benefits of labeled property graphs (LPG), allowing attributes and other IFC-related

information to be directly mapped to the elements, resulting in a significantly simpler structure. This can be seen on the figure below.

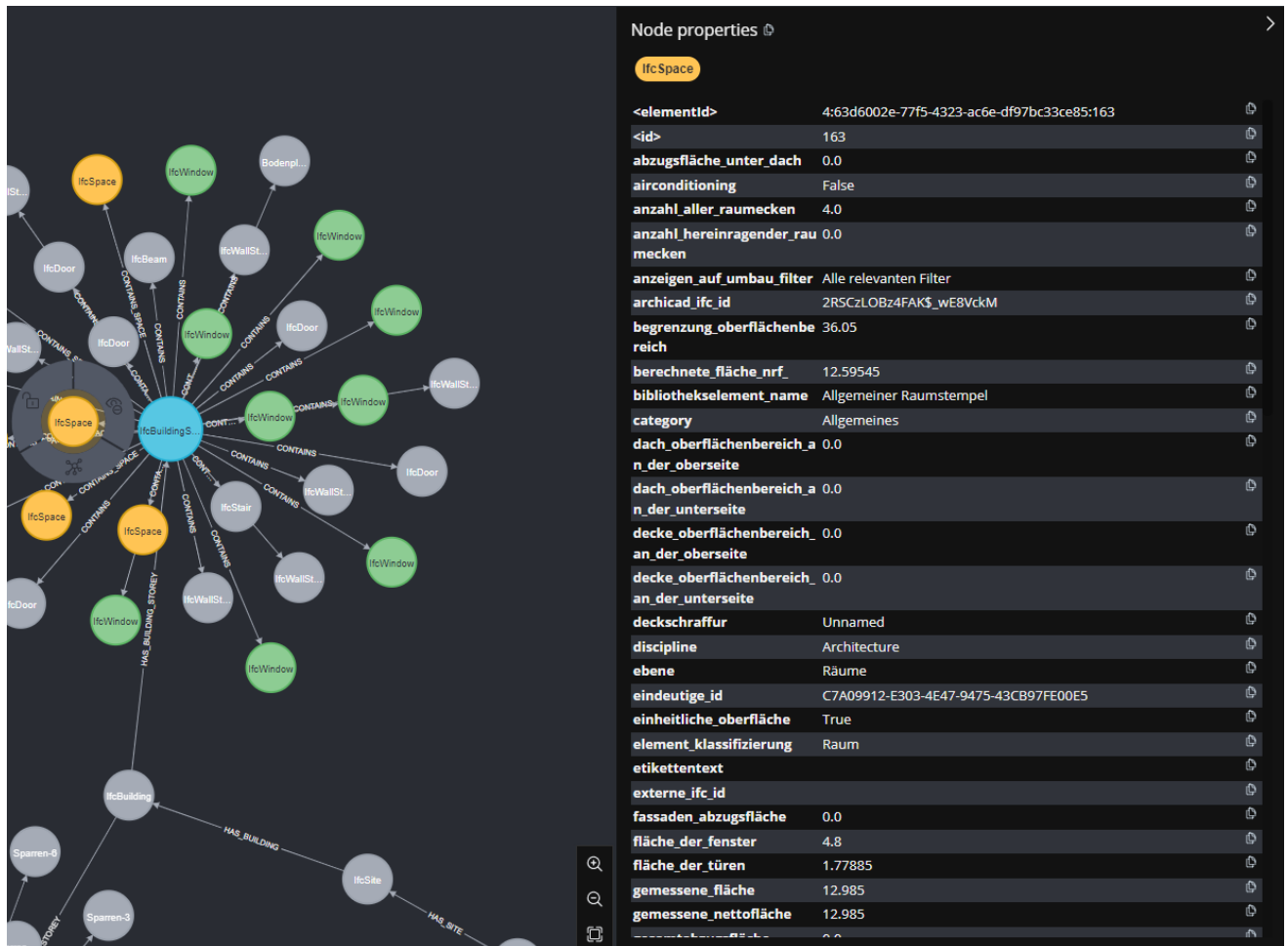


Figure 43 Node Properties

### 5.3. Key Characteristics of the Parsing Script

First Node Creation: Nodes are created according to the defined standard. The script iterates through the model, searching for elements under IfcElement. For each element, a corresponding node is generated.

1. Modular Construction of the Script: The script is designed in a modular fashion, with key functions as follows:
  - extract\_properties: Extracts the properties of the targeted object, flattens them, and returns a dictionary to be appended to the Cypher script. This function utilizes the IsDefinedBy and RelatingPropertyDefinition features of the IFC schema.

- `check_value`: Checks if the value contains valid characters for Cypher syntax. This function is called while extracting properties.
  - `escape_value`: If a value contains problematic characters for Cypher syntax, this function is called to replace those characters.
  - `extract_coordinates`: Extracts the coordinates of the given element. While iterating through the entities inside the model, this function generates a separate node called "Coordinates" with relevant x, y, z coordinate information, and links this node to the element using a `HAS_COORDINATES` relationship. To achieve this, the script leverages the `ObjectPlacement`, `RelativePlacement`, and `IfcAxis2Placement` features of the IFC schema.
2. `process_ifc_model`: This is the main function that iterates through the elements inside the model. It generates nodes for each entity with relevant information and links them to their spatial container using the `HOSTED_BY` relationship. For doors and windows, the script uses the `IfcRelFillsElement` and `IfcRelVoidsElement` features from the IFC schema to capture and represent the hosting walls of these elements.

The script is also can be found in Appendix A under `ifc_to_cypher.py`

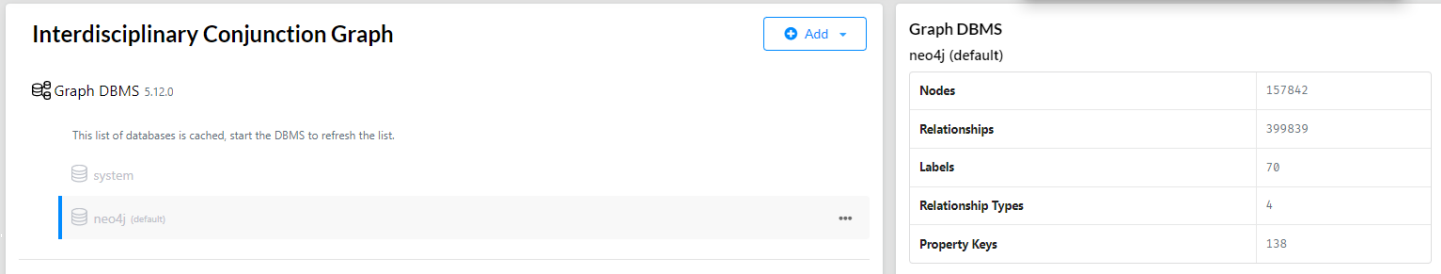
#### 5.4. Batch Cypher Import

A Python script was employed to batch import Cypher commands from `.txt` files into the graph database. The script not only automates the import process but also includes a validation function that checks the integrity of each Cypher query before execution. If any issues are detected, the validation function raises an error, preventing faulty queries from being executed. This ensures a smooth and reliable data import process, maintaining the consistency and accuracy of the graph database.

For the script see the file `cypher_batch_importer.py` in Appendix A.

#### 5.5. Interdisciplinary Conjunction Graph

The project models were processed using the parser script tool described in the methodology section and then imported into the Neo4j database through a batch importer script.



The screenshot displays the 'Interdisciplinary Conjunction Graph' interface. On the left, there is a section for 'Graph DBMS 5.12.0' with a message: 'This list of databases is cached, start the DBMS to refresh the list.' Below this, a list of databases is shown, including 'system' and 'neo4j (default)'. On the right, a table titled 'Graph DBMS neo4j (default)' provides summary statistics:

Category	Count
Nodes	157842
Relationships	399839
Labels	70
Relationship Types	4
Property Keys	138

## 5.6. Analysis

To analyze the current state of the **Inter-Discipline Conjunction Graph (ICG)** and the project as a whole, a series of queries and visualizations were employed. This analysis is crucial for understanding the impact and significance of the various models involved in the project and how they interact with one another. The queries were executed using **Cypher**, the native query language for Neo4j, through a Python script embedded in a Jupyter Notebook.

The use of Jupyter Notebook provides an organized and efficient environment to execute and track both queries and visualizations in one place. This setup allows for a dynamic workflow, where query results can be directly visualized and assessed, making it easier to analyze complex relationships within the graph and adjust the approach based on real-time feedback. By integrating Cypher queries into this workflow, a deeper understanding of the data can be achieved, highlighting the interconnectedness and dependencies of various elements across different disciplines.

An Example: A cypher query to see the count of nodes according to their ifc\_entity parameter.

```
1. MATCH (entity)
2. WHERE entity.ifc_entity IS NOT NULL
3. RETURN entity.ifc_entity AS entityType, count(entity) AS entityCount
4. ORDER BY entityType
5.
```

	entity Type	entityCount
1	"IfcActuator"	110
2	"IfcAirTerminal"	3104
3	"IfcAlarm"	592
4	"IfcAudioVisualAppliance"	98
5	"IfcBeam"	202
6	"IfcBuildingElementProxy"	3644
7		

Figure 44 Query Results

As can be seen from the example, the results are not easy to comprehend directly from the neo4j browser, therefore a visual in this sense would be greatly helpful. Figure below illustrates the count of nodes grouped by IfcEntity well.

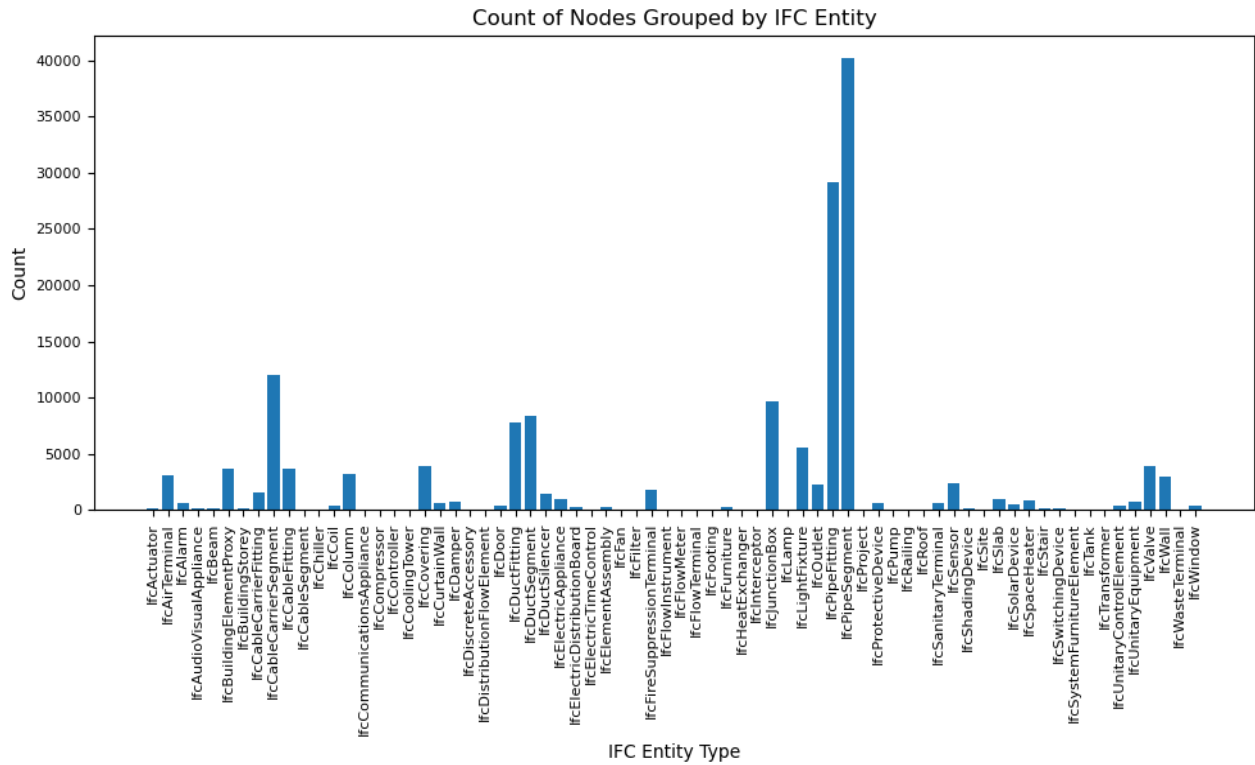


Figure 45 Count of Nodes by IFC Entity Bar Chart

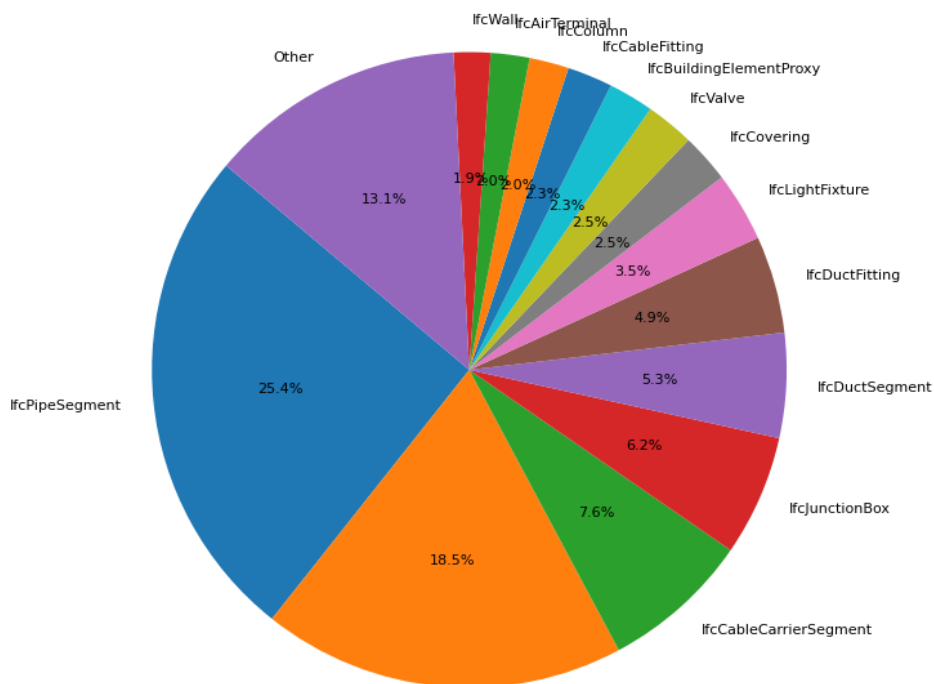


Figure 46 Distribution of Nodes by IFC Entity



Here we have the count of nodes by discipline. Distribution of node count and percentages can be seen in the figures below.

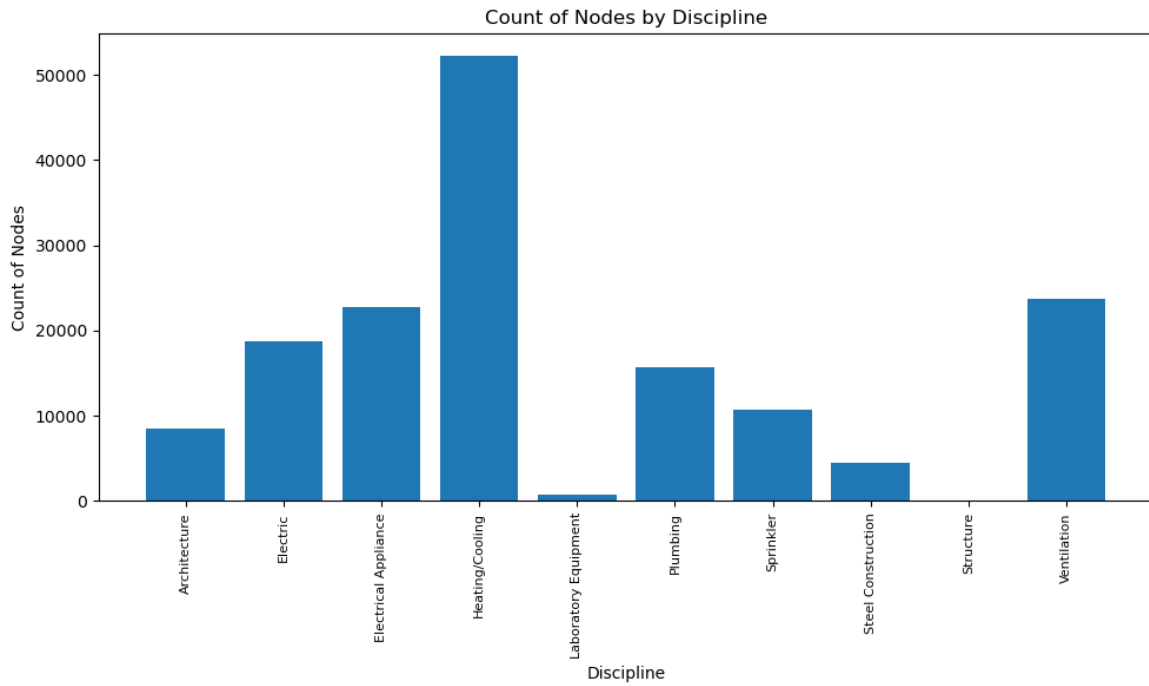


Figure 47 Count of Nodes by Discipline Bar Chart

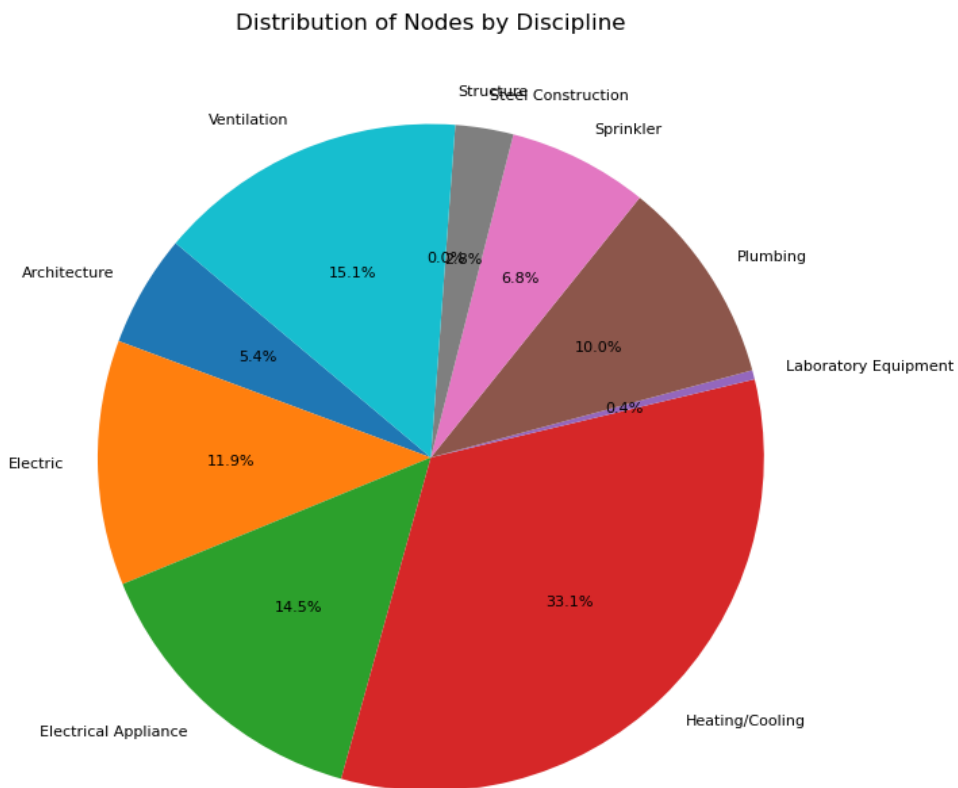


Figure 48 Distribution of Count of Nodes by Discipline

## 5.7. Semantic Enrichment

After successfully implementing the object graphs, further processing with BIM-M Meta Graph is implemented. First, for the defined criteria and child nodes are mapped into the ICG. The figure below illustrates the mapped project evaluation criteria.

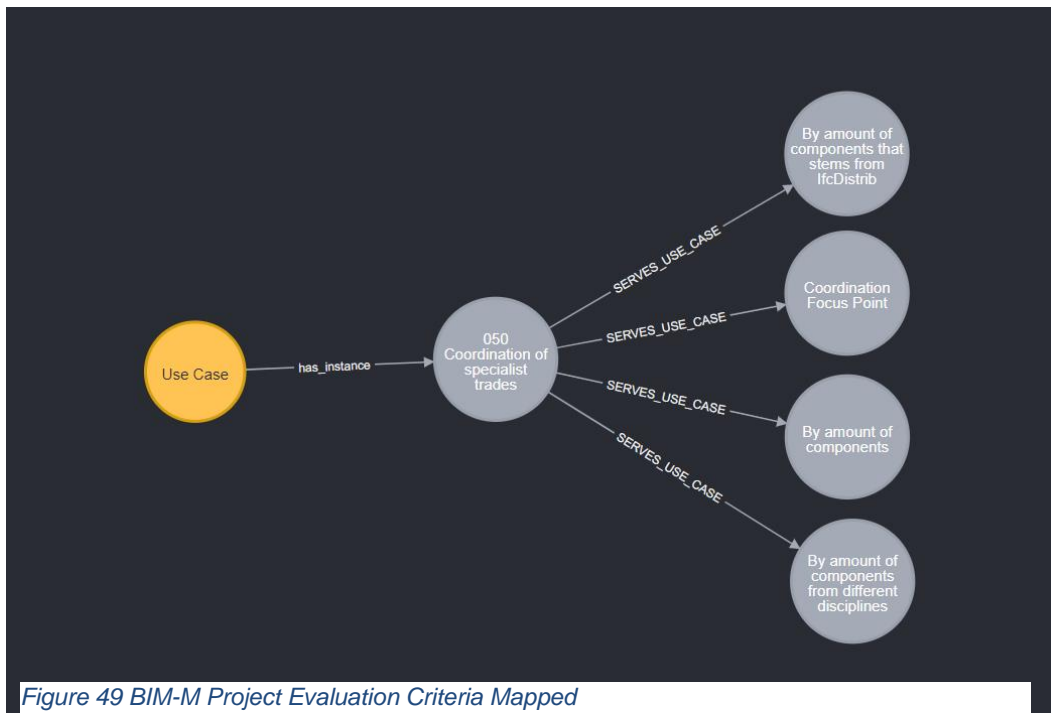


Figure 49 BIM-M Project Evaluation Criteria Mapped

As can be seen from the figure the ICG has been examined in the following 4 criteria. These nodes are listed below and the linking the IfcSpace nodes workflow and results are illustrated below.

1. By amount of components from per sqm
2. By amount of components coming from different disciplines
3. By amount of components that stem from IfcDistributionFlowElement

And using these 3 metrics last node for a summary as Coordination Focus Point.

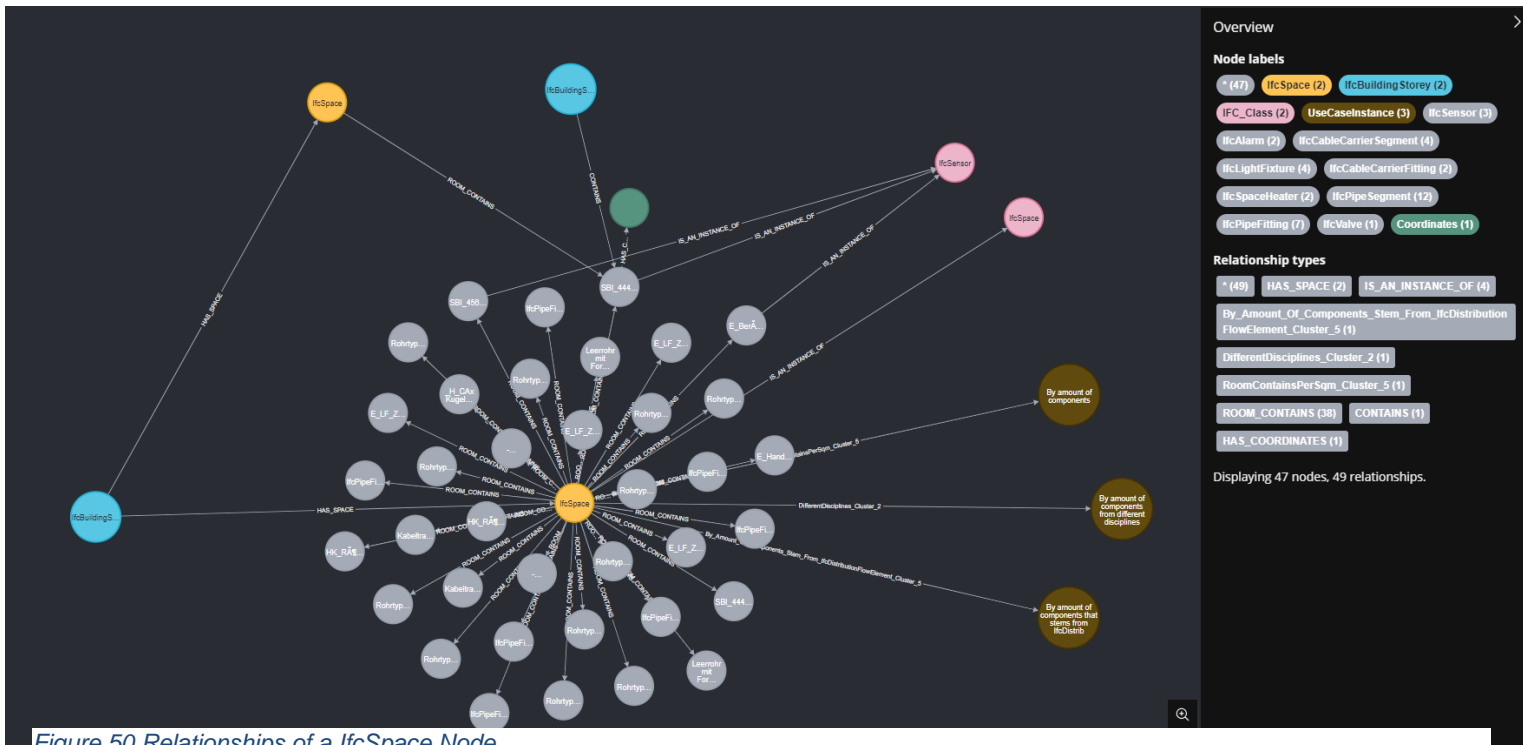


Figure 50 Relationships of a IfcSpace Node

The figure above shows the relationships of a IfcSpace Node. Here the utilized ROOM\_CONTAINS relationships and the relative HVAC components can be seen. Pink Nodes represent the Nodes from the IFC Meta Graph, Brown Nodes represent the created child nodes for project evaluation criteria.

### 5.7.1. By Amount of Components per square meter

This metric is calculated using the Net Floor Area attribute of IfcSpace nodes and their count of ROOM\_CONTAINS relationships. Since each ROOM\_CONTAINS are linked to the relevant HVAC Component node only once, the count of relationships is equal to the count of components.

After querying this information, the results are clustered using Python's Scikit-learn library clustering algorithm. Results are expressed in the charts below.

IfcSpace ROOM\_CONTAINS Counts - Sampled by Clusters

IfcSpaceGUID	IfcSpaceName	IfcSpaceCleanName	GrossFloorArea	RoomContainsCount	RoomContainsPerSqm	Cluster
1\$ku6QJ2v52QBT4B5DNP2N	03109_S	Schacht	0.49	94	191.83673469387756	1
2cgdGge6PEJeAijkLPzXvJ	02109_S	Schacht	0.49	64	130.6122448979592	2
2awkYtQMPFh8vurLKeE50A	01109_S	Schacht	0.49	57	116.3265306122449	2
1\$ku6QJ2v52QBT4B5DNPC5	03043_S	Schacht	0.89	102	114.6067415730337	2
1ku6QJ2v52QBT4B5DNP3	03181_S	Schacht	1.02	96	94.11764705882352	2
1\$ku6QJ2v52QBT4B5DNP3A	03241_S	Schacht	1.42	111	78.16901408450704	3
2cgdGge6PEJeAijkLPzXvV	02043_S	Schacht	0.89	67	75.28089887640449	3
2awkYtQMPFh8vurLKeE50y	01043_S	Schacht	0.89	64	71.91011235955057	3
3TYI_Xecj06e_eHP16H8YV	00043_S	Schacht	0.61	42	68.85245901639344	3
3c1Y6GebXC\$fflKld80N2V	03296	Schacht	0.66	42	63.63636363636363	3
0jr4jfovP3thl51tNbaP99	01296	Schacht	0.66	28	42.42424242424242	4
1noEDQhCP5ig0R2mKwR	01011_S	Schacht	2.35	96	40.85106382978723	4
2cgdGge6PEJeAijkLPzXvC	02296	Schacht	0.66	25	37.878787878787875	4
02pjiQdRbDTQmiyx1UP3ay	00181_S	Schacht	0.61	23	37.704918032786885	4
1\$ku6QJ2v52QBT4B5DNPC_	03011_S	Schacht	2.35	81	34.46808510638298	4
1H1EjcfOLEJe70tc4qb3uG	00243_S	Schacht	0.38	5	13.157894736842104	5
3TYI_Xecj06e_eHP16H8Zv	00124_S	Schacht	3.65	48	13.15068493150685	5
3TYI_Xecj06e_eHP16H8YH	00111_S	Schacht	0.87	11	12.64367816091954	5
254IH0ug1EqYyyU9o\$cgXM	-1247	Druckluft	31.34	351	11.199744735162732	5
2TQjW1nNnBgw4_YD9QTuVf	00051	IT Raum	10.28	115	11.186770428015565	5

Figure 52 Table of Room\_Contains per sqm Clusters

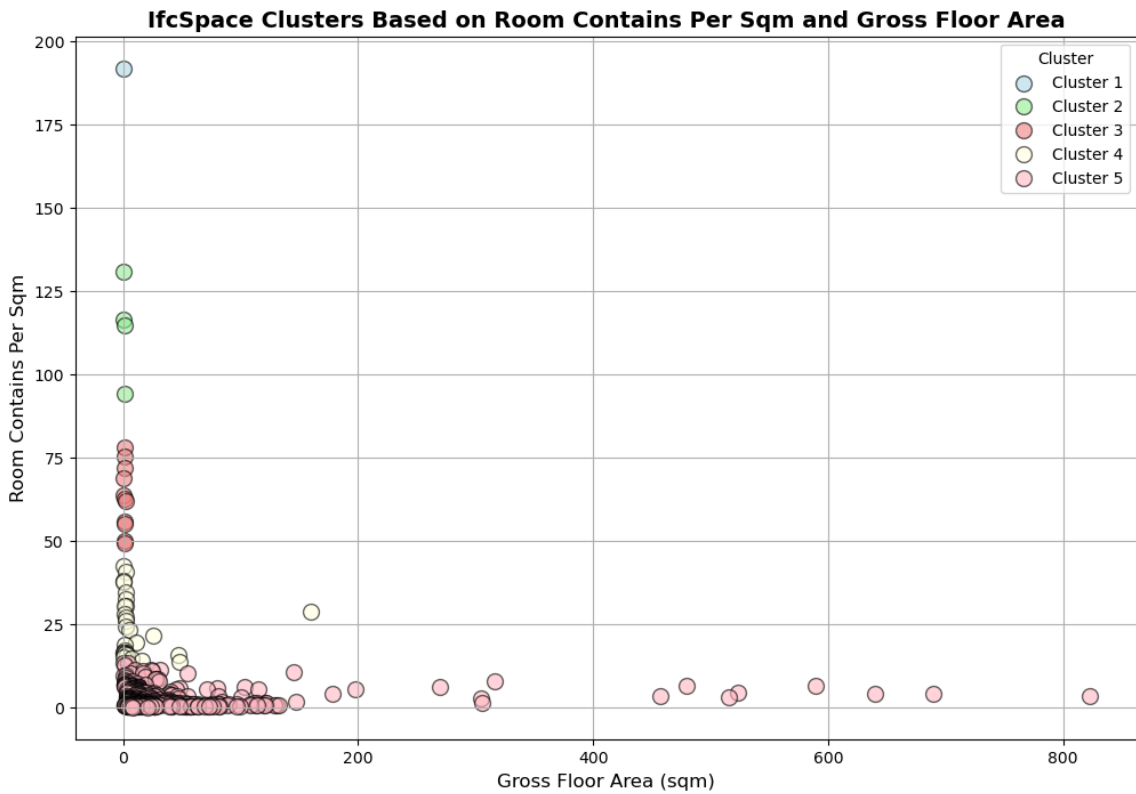


Figure 51 Scatter Plot of Room Contains Per Sqm Clusters

From the clusters, it can be realized that according to this metric Shaft Spaces has the most amount of HVAC components per sqm. It can also be seen that first cluster is significantly denser than the rest of the components.

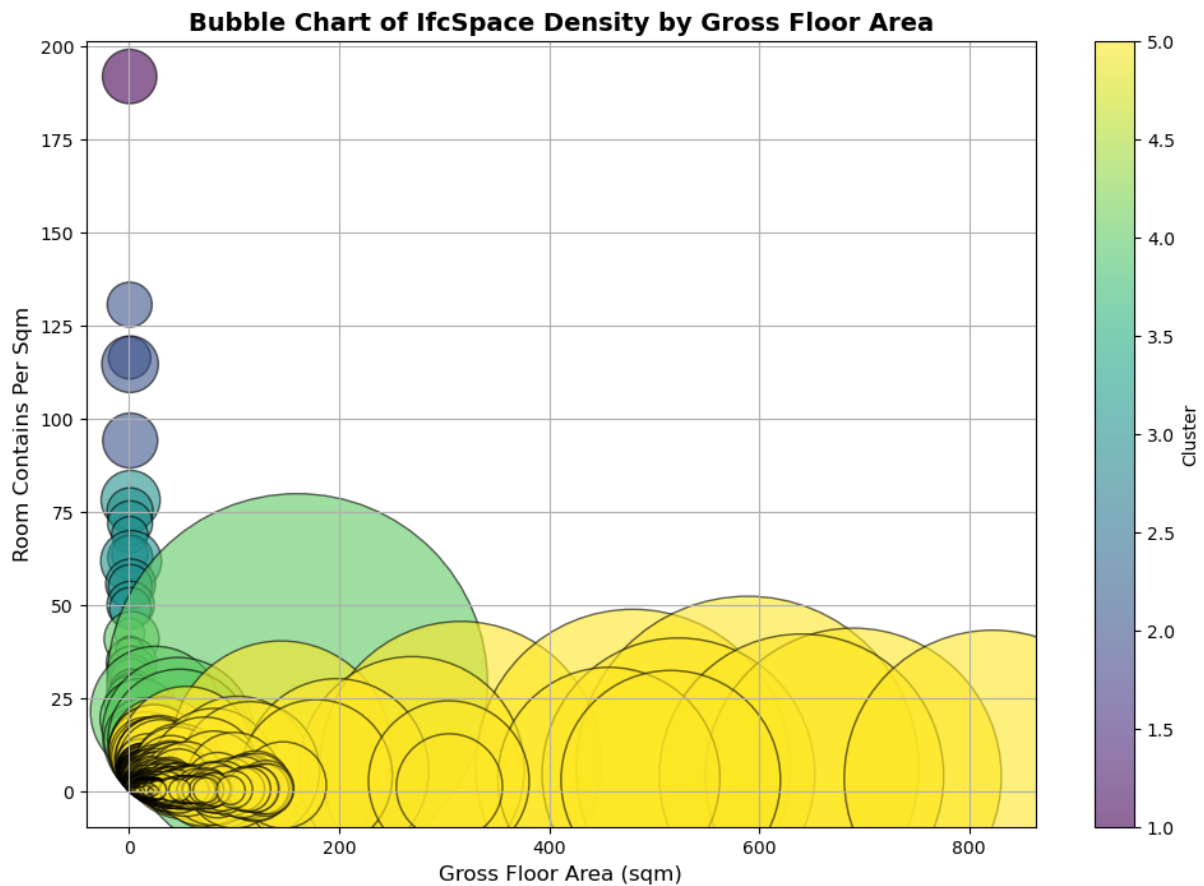


Figure 53 Bubble Chart of IfcSpace Density by Area

In the figure above the relation between the sizes of rooms and component density can be observed. Bigger rooms drop below the threshold of 50 components per sqm and they are significantly less dependent on other disciplines in comparison to clusters 2 and 1.

### 5.7.2. By Amount of Components coming from different Disciplines

Second metric that I've created is count of components coming from different disciplines. For this again the results of the related Cypher query were fed in to the Scikit-Learn's Clustering Algorithm. The table below shows 5 examples of IfcSpace's from each cluster.

ifcSpaceGUID	ifcSpaceName	ifcSpaceCleanName	GrossFloorArea	RoomContainsCount	UniqueDisciplines	Cluster
3H_5mnG51DzBvQGVI9Ptrx	00127a	Elektro-Mechanische Werkstatt 00.06	54.46	561	6	1
3H_5mnG51DzBvQGVI9Ptrx	00161	Messräume / IOT-CPS/DPR 00.04	197.55	1081	6	1
3Uq1NPUHv8NO2BUb\$qeFuZ	27	FLACHE EG 6	822.41	2802	6	1
3Uq1NPUHv8NO2BUb\$qeFuW	28	FLACHE EG 7	45.05	247	6	1
3H_5mnG51DzBvQGVI9Ptr8	00165	Messr. / INN-WLN 00.05b	19.14	179	6	1
1FqHRfQLP4TfRQyuyAhE0j	-1109_X	Treppenhaus Nord	15.77	37	4	2
0TxTjaQB8E85GamRY2Als	-1099	Sprinkler	47.49	272	4	2
2cgdGge6PEJeAijkLPzxxv0	02104_S	Schacht	2.24	21	4	2
2cgdGge6PEJeAijkLPzxxv8	02125_X	Verkehrsfli-NW-TUM	128.23	80	4	2
1vkhI\$URD7r98uZn9vDb6R	00067	Vortragssaal 1 + 2	146.57	239	4	2
28Mbt1jJL8ggwQNPCEXyix	00289_X	Treppenraum	96.83	26	3	3
OwoOH6MmvAsQ6hADx55ktL	00101a	Forum / Coworking	84.23	136	3	3
2xdLMDmpzE1QGfQ7b34Xk0	00241_S	Schacht	1.39	26	3	3
2cgdGge6PEJeAijkLPzxxuq	02244b	Meeting Virtual	41.75	40	3	3
2cgdGge6PEJeAijkLPzxxur	02187_X	Open Space	63.48	23	3	3
2awkYTQMPFh8vurLKeE50G	01259_S	Schacht	3.04	13	2	4
1Wl6B50vj68BahidM_zWgm	03238_X	Treppenraum Süd	20.49	16	2	4
1Wl6B50vj68BahidM_zWgu	03180_X	Treppenraum West	20.49	16	2	4
1\$ku6QJ2v52QBT4B5DNP2N	03109_S	Schacht	0.49	94	2	4
0ZENxBlz56xR8DsqfwrEjm	-1308	Technik Server	89.15	65	2	4
0Kg3dMgD5B9eYTMptkIC6i	01290_X	Verkehrsfläche-Zugang Süd	25.78	16	1	5
0Kg3dMgD5B9eYTMptkIC1E	01025	P-Box-SO	4.1	8	1	5
0Kg3dMgD5B9eYTMptkIC1D	01026	P-Box-SO	4.1	7	1	5
0Kg3dMgD5B9eYTMptkIC0J	01161	TT-NW	8.26	2	1	5
0Kg3dMgD5B9eYTMptkICFc	01027_X	Stauraum-SO	8.24	1	1	5

Figure 55 Table of Examples from Clusters of IfcSpaces by RoomContains Count coming from Unique Disciplines

In this table it can be observed that room sizes plays an important role but there also exceptional Rooms such as again Shafts, Technical Rooms and Staircases which contains components from different unique disciplines.

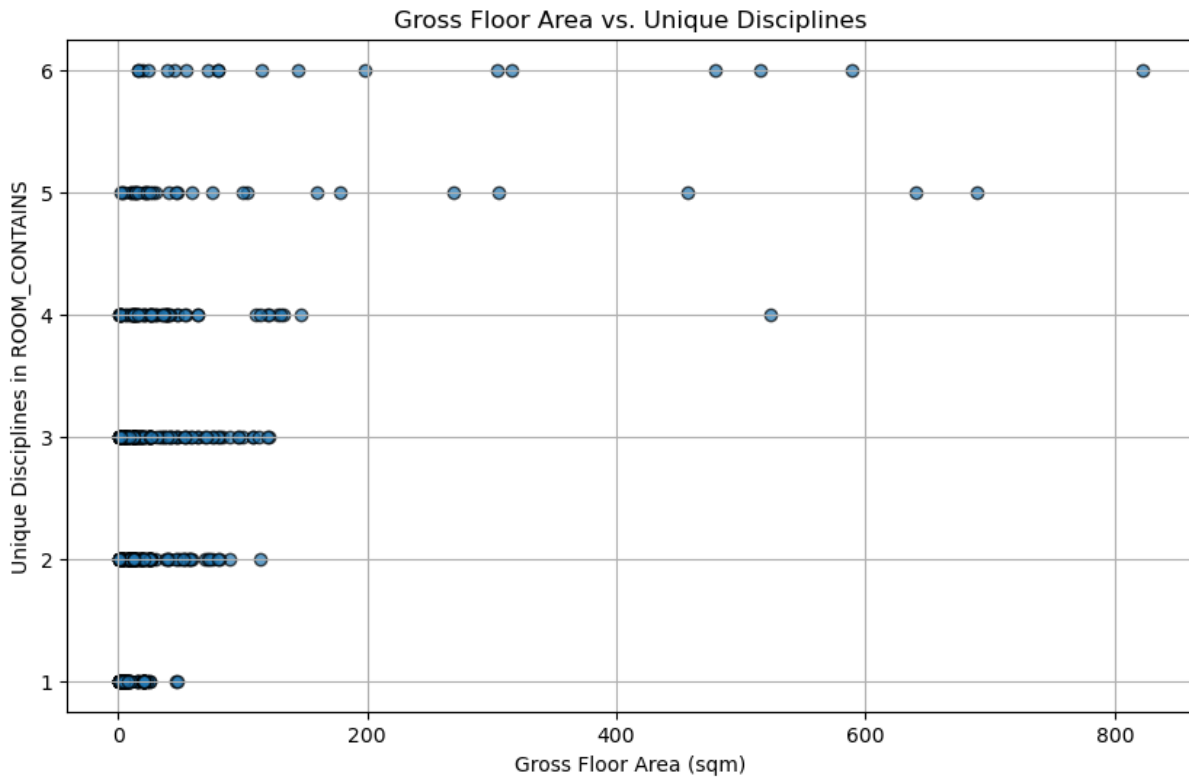


Figure 54 Scatter Plot of Unique Disciplines in ROOM\_Contains against Floor Area

The table above shows the relationship between Floor area and the Unique Discipline counts. It can be seen that the bigger rooms have significantly more components coming from different disciplines in comparison to other rooms.

### 5.7.3. By the Elements that stem from IfcFlowTerminal

First step of this implementation is to match the components in IFC Object Graphs to IFC Meta Graph. The Query below creates the IS\_AN\_INSTANCE\_OF relationship between the IFC Meta Graph and the nodes from IFC Object Graphs.

```

1. // Match elements with an ifc_entity and corresponding IFC_Class nodes
2. MATCH (element)
3. WHERE element.ifc_entity IS NOT NULL
4. WITH element, element.ifc_entity AS entity_name
5. MATCH (class:IFC_Class {name: entity_name, subgraph: "IFC Meta Graph"})
6. // Create IS_AN_INSTANCE_OF relationship only if it doesn't already exist
7. MERGE (element)-[rel:IS_AN_INSTANCE_OF {subgraph: "IFC Meta Graph"}]-
>(class)
8. RETURN COUNT(rel) AS created_relationships
9.

```

After creating these relationships, following queries were executed to achieve the desired outcome. First query finds the child nodes of IfcFlowTerminal from the IFC Meta Graph.

```

1. // Step 1: Find all subclasses (child nodes) of IfcFlowTerminal in the
hierarchy
2. MATCH (terminalClass:IFC_Class {name: "IfcFlowTerminal"})-
[:SUPERTYPE_OF*]->(subClass)
3. RETURN DISTINCT subClass.name AS SubclassOfIfcFlowTerminal
4.

```

Second step matches the HVAC components that match these nodes that are found from the first query.

```

1. // Step 2: Match all components that are instances of IfcFlowTerminal or
its subclasses
2. MATCH (terminalClass:IFC_Class {name: "IfcFlowTerminal"})-
[:SUPERTYPE_OF*]->(subClass)

```

3. WITH DISTINCT subClass
4. MATCH (component)-[:IS\_AN\_INSTANCE\_OF]->(subClass)
5. RETURN component.guid AS ComponentGUID, component.name AS ComponentName, subClass.name AS ClassType
- 6.

Again, the results are divided into clusters using Python's Scikit-learn library. In the figures below the results are illustrated by several charts.

ifcSpaceGUID	ifcSpaceCleanName	ComponentCount	NetFloorArea	Cluster
0ZENxBIz56xR8DsqfwrEJC	Technik_Heizg./Kälte	4508	160.15	1
3Uq1NPUHv8NO2BUb\$qeFuj	FLACHE EG 9	3472	589.02	1
0Ztmi7twHFLOkofgEwtzjr	FLACHE EG 1	2951	479.81	2
2aDfjdiUHDwOfg5tXri\$In	Fläche 1	2659	689.73	2
3Uq1NPUHv8NO2BUb\$qeFuZ	FLACHE EG 6	2624	822.41	2
2GTxNg6nD9rQZzaciJGm9C	Fläche 4	2423	640.33	2
3Uq1NPUHv8NO2BUb\$qeFuq	FLACHE EG 11	2347	316.01	2
2p4W9J7xvAle15vnhpKVhf	Fläche 2	1489	456.9	3
3Uq1NPUHv8NO2BUb\$qeFut	FLACHE EG 10	1442	144.81	3
3H_5mnG51DzBvQGVI9PsA_	Technikum T 00.01	1379	515.84	3
3H_5mnG51DzBvQGVI9Ptro	Messräume / IOT-CPS/DPR 00.04	1007	197.55	3
1vkhI\$URD7r98uZn9vDb6O	TUM Technikum 00.02a	753	304.5	4
35mHXF961CmezE3pBjZ4Ph	Technik Lüftung	722	178.19	4
20cFuSwZX0xBUPUu_MA36H	WC H	709	46.57	4
254IHoug1EqQyyU9o\$cgXb	Flur	634	103.56	4
0ZENxBIz56xR8DsqfwrEJE	Technik Sani	610	48.0	4
0TxTJaQBb8E85GamRY2Als	Sprinkler	265	47.49	5
2Y0UEX4X1Alhyizv\$nkBQK	TUM Technikum 00.02b	263	80.86	5
0hkCv0zkL9bhyFm8VZoeNd	UXP-Labor	245	24.25	5
2JL5cu_xj7Yx8DLyZwH8Ilg	Messr. / INN-WLN+SES 00.05a	241	23.32	5
0WBZ9gipX0wvWGuNdXal_D	FLACHE EG 5	231	29.08	5

Figure 56 Table of Clusters from Spaces with IfcFlowTerminal stemming Elements



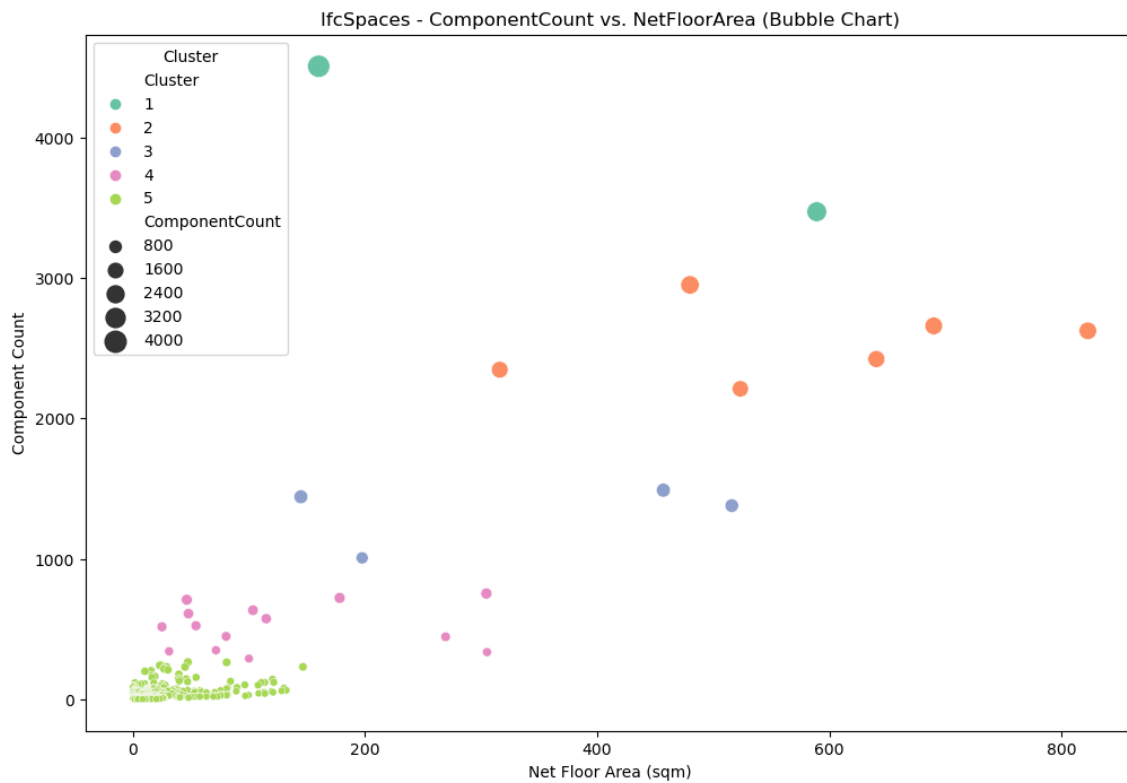


Figure 57 Bubble Chart of Component Counts with Size

The figure above illustrates the cluster distribution and room sizes with bubble sizes according to their component counts. Here it can be seen that majority of the rooms falls in to the fifth cluster with small floor area and below 1000 components.

#### 5.7.4. Coordination Focus Points

Finally, an additional metric is derived from the three previously defined metrics by calculating their average score. This average score is then used in a secondary clustering process to identify IfcSpace nodes that are particularly dependent on coordination with other disciplines. This final clustering aims to pinpoint spaces with high interdisciplinary dependency, further enhancing insights for targeted project coordination.

	IfcGUID	Clean_Name	sqm	RoomContainsDensity Cluster	DifferentDiscipline Cluster	FlowElement Cluster
0	009GnbXLL059b8Tyw17JtN	Kaffeebar	39.73	5	2	5
1	02pjiQdRbDTQmiyx1UP3ao	Schacht	2.20	4	2	5
2	02pjiQdRbDTQmiyx1UP3ay	Schacht	0.61	4	3	5
3	05XIidsJB94QAglZ5Mj89Y3	Schacht	1.17	4	4	5
4	0F6hPQ8en5rQ6QKaZdFQ7e	ELT	40.95	5	2	5
...	...	...	...	...	...	...
437	3ji0wNwSPF08oZ\$q\$8T3tA	WC-D	18.21	5	3	5
438	3ji0wNwSPF08oZ\$q\$8T3tC	WC-H	19.60	5	3	5
439	3ji0wNwSPF08oZ\$q\$8T3tF	Beh.-WC	5.13	5	3	5
440	3rFg0sMZz1H8zaR_imQ0\$	Dieselpump Sprinkl	26.55	5	2	5
441	3tU6_Vv1X0COsTfNK2MhMN	Stauraum-NO	4.72	5	5	5

Figure 58 Table of Clusters of IfcSpace Nodes

The figure above shows the cluster groups of the IfcSpace Nodes in the ICG. Final coordination node metric is calculated based on these cluster group numbers.

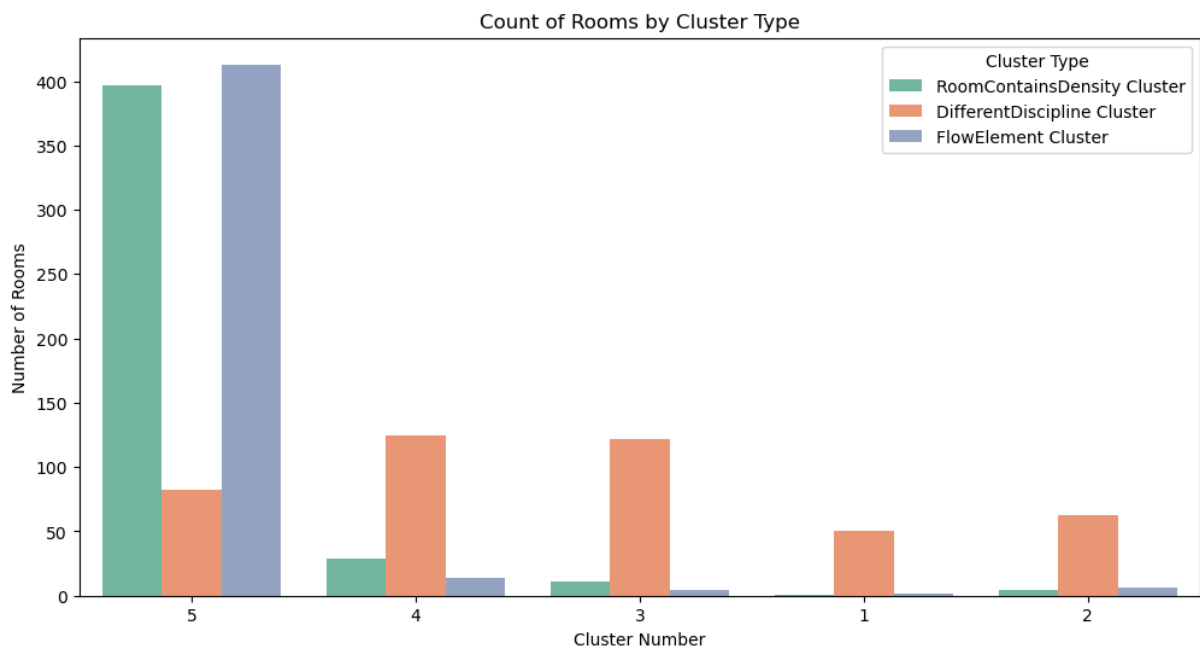


Figure 59 Bar Chart of Count of Rooms by Cluster Type

The figure above shows the distribution of rooms by their cluster type. This chart also displays the number of rooms in each cluster group and their relation to other clusters.

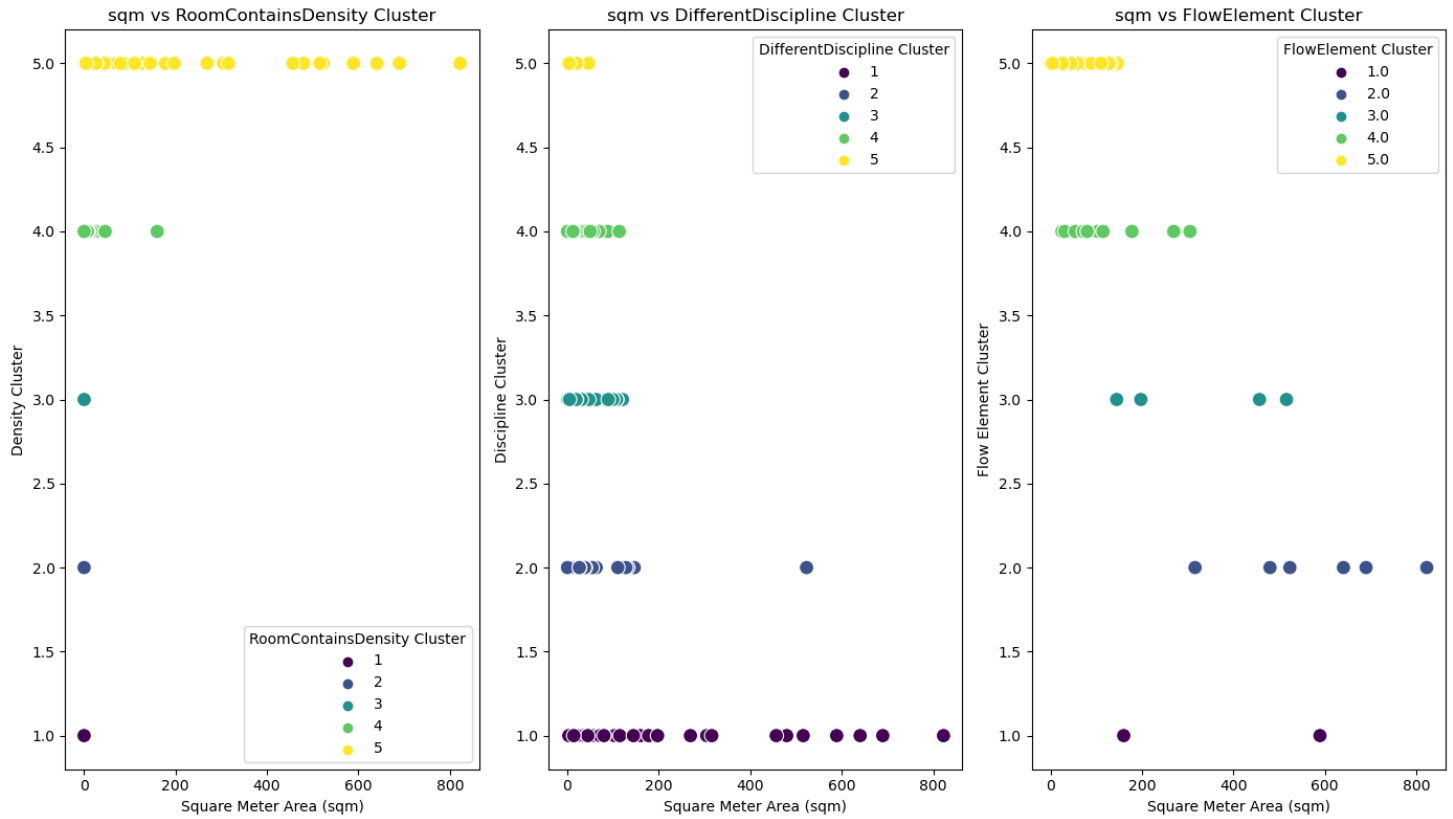


Figure 60 Scatter Plot Charts of Cluster Groups

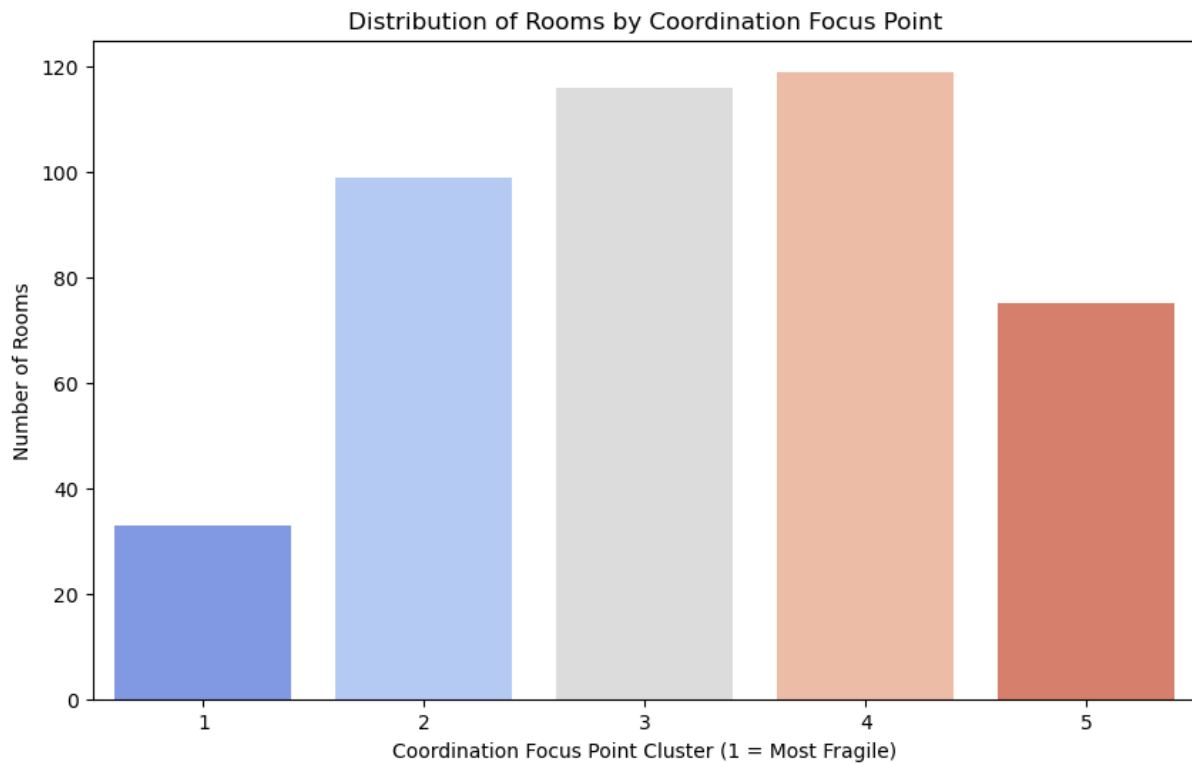
The Scatter Plot charts of Cluster groups illustrates the distribution of rooms by their net floor area.

IfcGUID	Clean Name	sqm	RoomContainsDensity Cluster	DifferentDiscipline Cluster	FlowElement Cluster	Coordination Focus Point
0ZENxBIz56xR8DsqrwEJC	Technik Heizg./Kälte	160.15	4	1	1.0	1
0ZENxBIz56xR8DsqrwEJE	Technik Sani	48.0	4	1	4.0	1
0Ztmi7twHFLokofgEwtzjr	FLACHE EG 1	479.81	5	1	2.0	1
0bxo4NI3PFRQEUw41Gfs7Z	WC D	25.17	4	1	4.0	1
0exv2FP6f5fxTOlgXI9PtO	SIBEL	1.12	3	2	5.0	1
009GnbXLL059b8Tyw17jtN	Kaffeebar	39.73	5	2	5.0	2
02pjiQdRbDTQmiyx1UP3ao	Schacht	2.2	4	2	5.0	2
02pjiQdRbDTQmiyx1UP3ay	Schacht	0.61	4	3	5.0	2
0F6hPQ8en5rQ6QKaZdFQ7e	ELT	40.95	5	2	5.0	2
0Gw8H9dUHEyOfR9xbEKIY0	Verkehrsfäche-SW	121.44	5	2	5.0	2
05XidsjB94QAglZ5Mj89Y3	Schacht	1.17	4	4	5.0	3
0GnNqVsxLBRQ05ov9k3YDY	Verkehrsfäche-Zugang Nord	76.92	5	3	5.0	3
0GnNqVsxLBRQ05ov9k3YER	Verkehrsfäche-Zugang Süd	37.11	5	3	5.0	3
0jr4JfovP3thl51tNbaP99	Schacht	0.66	4	4	5.0	3
0Kg3dMgD5B9eYTMptkIByL	Verkehrsfäche-SW	119.64	5	3	5.0	3
0Kg3dMgD5B9eYTMptkIBzQ	EZ/TT-NW	17.1	5	4	5.0	4
0Kg3dMgD5B9eYTMptkIBzS	TT-NO	19.7	5	4	5.0	4
0Kg3dMgD5B9eYTMptkIBzX	MF-SO	26.5	5	4	5.0	4
0Kg3dMgD5B9eYTMptkIBzY	I-Buro-SO	73.02	5	4	5.0	4
0Kg3dMgD5B9eYTMptkIBzZ	I-Buro-NO	75.71	5	4	5.0	4
0GnNqVsxLBRQ05ov9k3YEQ	Verkehrsfäche-Zugang Süd	25.78	5	5	5.0	5
0Kg3dMgD5B9eYTMptkIC0J	TT-NW	8.26	5	5	5.0	5
0Kg3dMgD5B9eYTMptkIC1D	P-Box-SO	4.1	5	5	5.0	5
0Kg3dMgD5B9eYTMptkIC1E	P-Box-SO	4.1	5	5	5.0	5
0Kg3dMgD5B9eYTMptkIC69	Copy-SO	6.74	5	5	5.0	5

Figure 61 Table of Example IfcSpaces from Clustering Results

The table above presents examples from the clustered groups, revealing intriguing results. Various rooms are grouped in unexpected ways, with the first group representing the rooms most dependent on coordination, based on the defined metrics.

This grouping highlights areas requiring intensive interdisciplinary alignment, providing valuable insights for targeted project management.



*Figure 62 Distribution of Rooms by Coordination Focus Point Groups*

The bar chart above shows the distribution of rooms across different Coordination Focus Point groups. Notably, the most fragile group, which requires the highest level of coordination, is comparatively smaller than the other clusters. This distribution underscores the unique needs of this group, emphasizing the importance of targeted coordination efforts in these specific rooms.

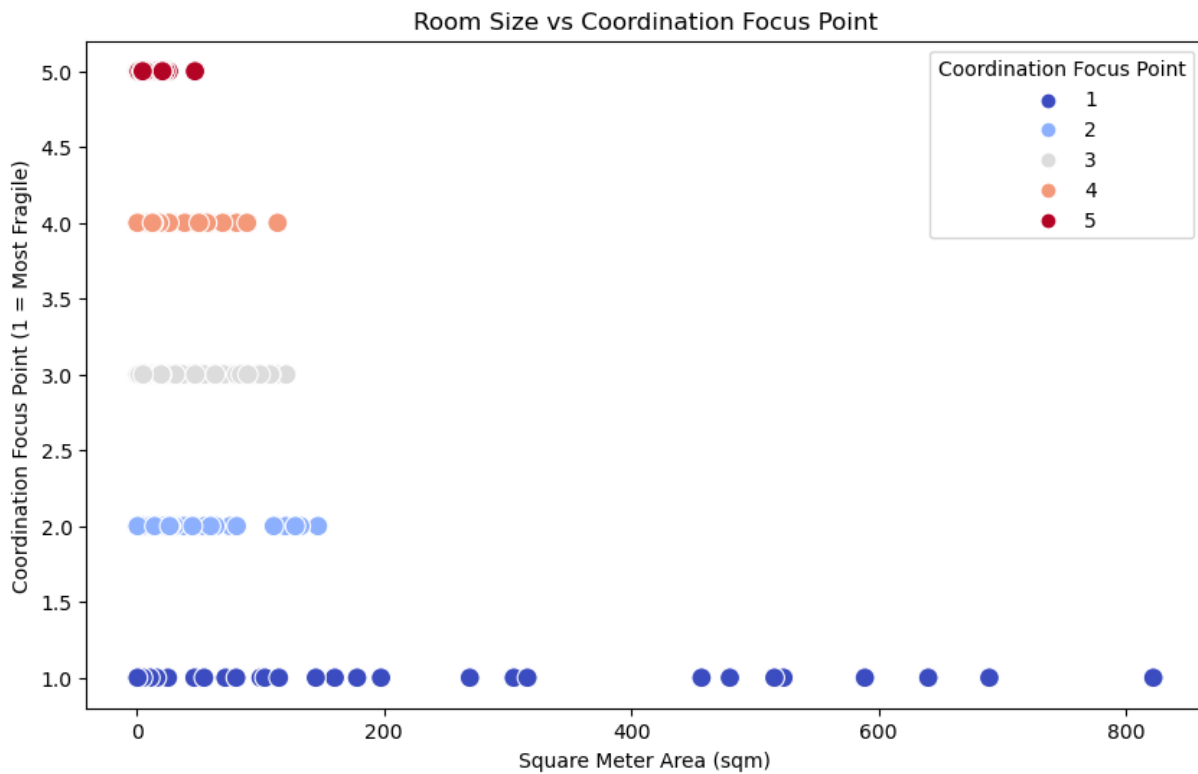


Figure 63 Scatter Plot of Room Floor Areas grouped by Coordination Focus Points

The scatter plot above displays rooms by size, grouped according to their Coordination Focus Point. It reveals a trend where larger rooms tend to have higher coordination dependencies. This insight can be valuable for project decision-making, as it highlights areas where larger spaces may require more intensive coordination efforts across disciplines.

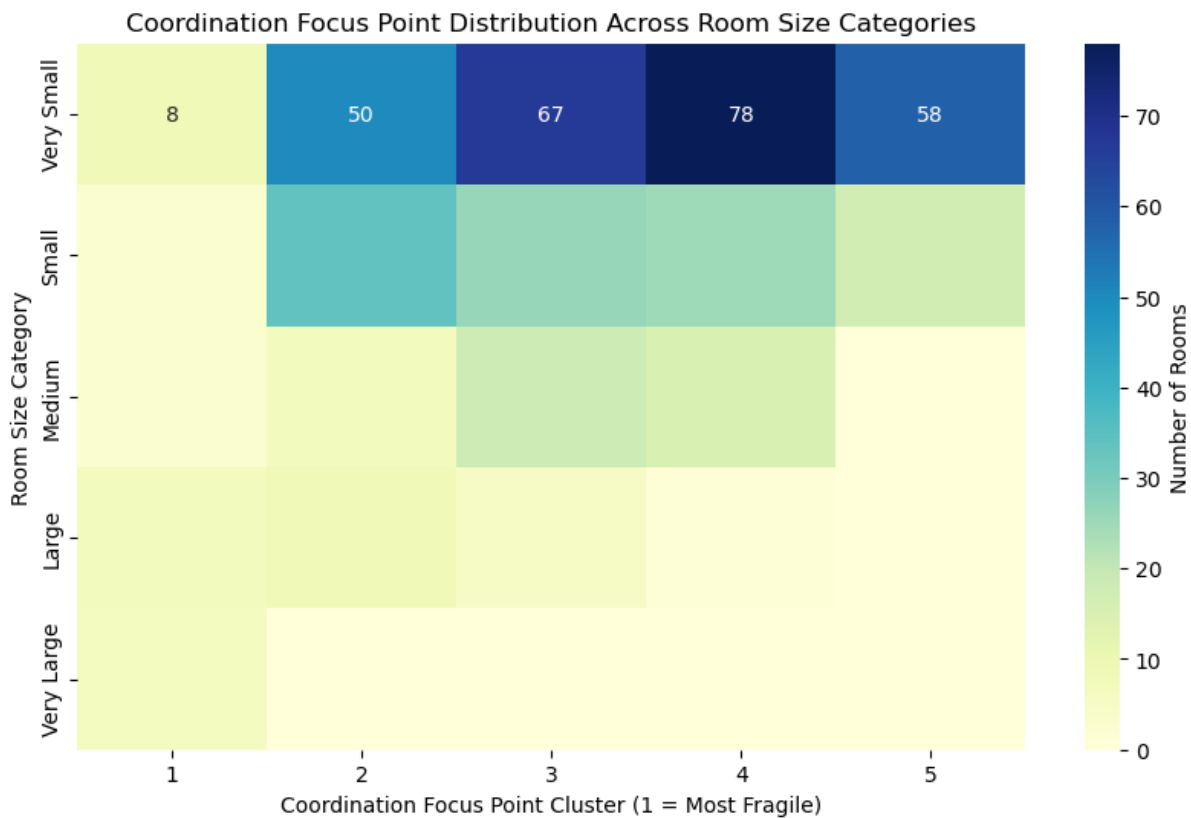


Figure 64 Heat Map Chart of Coordination Focus Point Distribution Across Room Sizes

The heat map above shows the distribution of Coordination Focus Point groups across different room size categories. It clearly indicates that the majority of rooms fall into the "very small" size category and belong to the fourth group, which has a low dependency on coordination with other disciplines. This distribution highlights that smaller rooms generally require less intensive interdisciplinary coordination.

## 5.8. Graph Edge Detection with GraphSAGE

CSV files were exported using prepared scripts within a Jupyter Notebook, as shown in the queries displayed in the figures below. These exported attributes are set to serve as features in the machine learning process, providing a structured dataset that captures the key project metrics needed for accurate and efficient model training.

```
# Define the query to export node data with elementId as graphId
cypher_query_export_nodes = """
CALL apoc.export.csv.query(
  "MATCH (n)
  RETURN n.elementId AS graph_id,
         n.guid AS id,
         n.NetFloorArea AS NetFloorArea,
         n.discipline AS discipline,
         n.Clean_name AS Clean_name,
         n.GlobalX AS GlobalX,
         n.GlobalY AS GlobalY,
         n.GlobalZ AS GlobalZ,
         n.ifc_entity AS ifc_entity",
  "nodes_with_graphId.csv",
  {}
)
"""

# Execute the query using the Neo4j driver
with driver.session() as session:
    session.run(cypher_query_export_nodes)
    print("Node data with graphId (elementId) exported to 'nodes_with_graphId.csv'")
```

✓ 2.3s

Node data with graphId (elementId) exported to 'nodes\_with\_graphId.csv'

Figure 65 Query to export nodes

```
# Define the query to export edge data with elementId as graphId
cypher_query_export_edges = """
CALL apoc.export.csv.query(
  "MATCH (n)-[r]->(m)
  RETURN r.elementId AS relationship_id,
         n.elementId AS source_graph_id,
         m.elementId AS target_graph_id,
         r.type AS relationship_type",
  "edges_with_graphId.csv",
  {}
)
"""

# Execute the query using the Neo4j driver
with driver.session() as session:
    session.run(cypher_query_export_edges)
    print("Edge data with graphId (elementId) exported to 'edges_with_graphId.csv'")
```

✓ 0.4s

Edge data with graphId (elementId) exported to 'edges\_with\_graphId.csv'

Figure 66 Query to export edges

```

4s # Create a mapping from graphId to unique indices
graphId_to_index = {graph_id: idx for idx, graph_id in enumerate(nodes_df['graph_id'].unique())}

# Map source and target `graphId` to indices in the edges DataFrame
edges_df['source_idx'] = edges_df['source_graph_id'].apply(lambda x: graphId_to_index.get(x, -1))
edges_df['target_idx'] = edges_df['target_graph_id'].apply(lambda x: graphId_to_index.get(x, -1))

# Remove invalid edges where source_idx or target_idx is -1 (missing graphId mappings)
edges_df = edges_df[(edges_df['source_idx'] != -1) & (edges_df['target_idx'] != -1)]

# Create the edge_index tensor (source and target indices)
import torch
edge_index = torch.tensor(edges_df[['source_idx', 'target_idx']].values.T, dtype=torch.long)

print(f"Edge index tensor shape: {edge_index.shape}")

Edge index tensor shape: torch.Size([2, 364261])

```

Figure 68 Edge Tensor Process

```

[5] from sklearn.preprocessing import LabelEncoder

# Encode categorical variables
label_encoder_discipline = LabelEncoder()
nodes_df['discipline_encoded'] = label_encoder_discipline.fit_transform(nodes_df['discipline'].fillna('Unknown'))

label_encoder_clean_name = LabelEncoder()
nodes_df['clean_name_encoded'] = label_encoder_clean_name.fit_transform(nodes_df['Clean_name'].fillna('Unknown'))

label_encoder_ifc_entity = LabelEncoder()
nodes_df['ifc_entity_encoded'] = label_encoder_ifc_entity.fit_transform(nodes_df['ifc_entity'].fillna('Unknown'))

# Normalize numeric features (e.g., 'GlobalX', 'GlobalY', 'GlobalZ')
nodes_df['GlobalX'] = (nodes_df['GlobalX'] - nodes_df['GlobalX'].mean()) / nodes_df['GlobalX'].std()
nodes_df['GlobalY'] = (nodes_df['GlobalY'] - nodes_df['GlobalY'].mean()) / nodes_df['GlobalY'].std()
nodes_df['GlobalZ'] = (nodes_df['GlobalZ'] - nodes_df['GlobalZ'].mean()) / nodes_df['GlobalZ'].std()

# Create the feature tensor (include both original and encoded features)
node_features = nodes_df[['NetFloorArea', 'discipline_encoded', 'Clean_name_encoded', 'GlobalX', 'GlobalY', 'GlobalZ', 'ifc_entity_encoded']]

# Handle missing values in node features if they exist
node_features = node_features.fillna(0)

# Convert node features to torch tensor
x = torch.tensor(node_features.values, dtype=torch.float)
print(f"Node features tensor shape: {x.shape}")

Node features tensor shape: torch.Size([316210, 7])

```

Figure 67 Node Tensor Shape

As illustrated in the figures above, the CSV files were transformed into tensors using the sklearn-preprocessing package. This intermediate step is essential for utilizing GraphSAGE, as it converts the data into a format suitable for graph-based machine learning, enabling efficient feature propagation and improved model performance.



```
[ ] import torch
import torch.nn as nn
from torch_geometric.nn import SAGEConv

# Define the GraphSAGE model for edge-level prediction
class GraphSAGEEdgePrediction(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(GraphSAGEEdgePrediction, self).__init__()
        self.conv1 = SAGEConv(in_channels, 128) # First GraphSAGE layer
        self.conv2 = SAGEConv(128, out_channels) # Second GraphSAGE layer
        self.fc = nn.Linear(out_channels * 2, 1) # Output layer (edge classification)

    def forward(self, x, edge_index):
        # Apply first GraphSAGE layer
        x = self.conv1(x, edge_index)
        x = torch.relu(x) # Apply ReLU activation

        # Apply second GraphSAGE layer
        x = self.conv2(x, edge_index)

        # Combine source and target node features (aggregation for edge prediction)
        source = x[edge_index[0]] # Get source node features
        target = x[edge_index[1]] # Get target node features

        # Concatenate the features of the source and target nodes
        edge_features = torch.cat([source, target], dim=-1)

        # Apply fully connected layer for binary classification
        edge_pred = self.fc(edge_features)
        return edge_pred
```

Figure 69 Model used in the Process

In the figure above, the model used in this process is illustrated. A GraphSAGE model with 2 convolutional layers is implemented, as shown in the code snippet above, which sets up the model for the training process. This configuration allows for effective feature aggregation from neighboring nodes, enhancing the model's capacity to learn from the graph structure and capture complex relationships within the data.

```

# Initialize model, loss function, and optimizer
model = GraphSAGEEdgePrediction(in_channels=x.shape[1], out_channels=64)
optimizer = optim.Adam(model.parameters(), lr=0.001) # Use a smaller learning rate
loss_fn = nn.BCEWithLogitsLoss() # Binary cross entropy loss (for edge classification)

# Early stopping parameters
patience = 10 # Number of epochs to wait for improvement
best_loss = float('inf')
epochs_without_improvement = 0

# Define the number of epochs
num_epochs = 2

# Training loop with early stopping
for epoch in range(num_epochs):
    model.train()
    optimizer.zero_grad()

    # Forward pass: compute node predictions
    out = model([data.x, data.edge_index])

    # Compute the loss
    loss = loss_fn(out.squeeze(), data.y)

    # Backward pass and optimization
    loss.backward()
    optimizer.step()

    # Check for improvement
    if loss.item() < best_loss:
        best_loss = loss.item()
        epochs_without_improvement = 0 # Reset counter
    else:
        epochs_without_improvement += 1

    # Print the loss every 10 epochs
    if epoch % 10 == 0:
        print(f"Epoch {epoch+1}/{num_epochs}, Loss: {loss.item():.4f}")

    # Early stopping: stop if no improvement for 'patience' epochs
    if epochs_without_improvement >= patience:
        print("Early stopping triggered. Training stopped.")
        break

```

Epoch 1/2, Loss: 13.7825

Figure 70 Code Snippet for the training process

The figure above presents the code used for the training process. The current model structure demonstrates the ability to predict edges in the training graph; however, signs of overfitting are evident. To address this and enhance model optimization, further experimentation is necessary. This could involve tuning

---

hyperparameters, implementing regularization techniques, or adjusting the model architecture to improve its generalization to new data.

## 6. Results and Discussion

### 6.1. Implementation

The clustering analysis provided a practical approach to identifying coordination dependencies within various room types. Using the Coordination Focus Point metric, rooms were evaluated based on relationship count, discipline diversity, and the presence of IfcFlowTerminal components, adjusted for room size. This structured scoring enabled the categorization of rooms by interdisciplinary needs, identifying those requiring intensive coordination. Furthermore, visualizations such as bar charts, scatter plots, and heat maps offered additional insights, highlighting the areas where coordination efforts are best allocated, especially in larger, more complex spaces. The use of mini-batching and class weighting addressed memory efficiency and class imbalance, ensuring an effective, scalable analysis.

### 6.2. Potential for Generalizability

The approach shows strong potential for broader application through integration with different knowledge meta graphs. This flexibility validates the **Interdisciplinary Conjunction Graph** (ICG) framework as adaptable to a range of project evaluation needs, enhancing insights and supporting evolving requirements in BIM projects. By integrating various meta-information sources, the ICG framework can dynamically respond to a variety of interdisciplinary needs, offering a versatile tool for ongoing project assessment across domains.

### 6.3. Limitations

1. Preprocessing: Due to the complexity of the dataset, preprocessing steps were extensive. Ensuring accurate parsing and formatting of data for graph

representation posed challenges, particularly when addressing diverse data types across disciplines.

2. **Data Collection:** Gathering high-quality, standardized BIM data across multiple domains is challenging. Inconsistent data formats and incomplete records required additional processing to prepare for graph analysis.
3. **Need for Further Experimentation:** While the model provided insightful results, additional experimentation is necessary to refine clustering and enhance prediction accuracy. Exploring alternative models or tuning existing parameters could lead to improved generalizability and performance.

#### **6.4. Research Gap Discussion**

This study addresses the previously identified research gap by applying graph-based methodologies specifically to analyze the dependencies between HVAC systems and architectural spaces. Through the development and implementation of the Interdisciplinary Conjunction Graph (ICG), the benefits of this approach are evident. By mapping and enriching relationships across disciplines, the ICG framework provides stakeholders with a comprehensive view of the spatial and functional roles of HVAC elements in relation to architectural layouts. This approach enables clearer insights into how interdisciplinary interactions influence overall project quality and alignment with performance standards.

The structured analysis facilitated by the ICG framework offers tangible advantages, such as enhancing coordination, improving decision-making, and reducing project errors. By revealing indirect relationships, such as dependencies between HVAC and architectural components, this methodology bridges the gap in traditional BIM evaluation techniques, offering an actionable framework for project planning quality assessment. The benefits demonstrated here underscore the potential of graph databases to yield robust, data-driven insights in complex BIM environments, validating this approach as a powerful tool for industry-wide application.

## 7. Conclusion

This thesis presented a structured approach for utilizing graph databases to enhance interdisciplinary coordination within BIM projects. By implementing the Interdisciplinary Conjunction Graph (ICG) framework, this study enabled the analysis of dependencies between HVAC components and architectural spaces, addressing previously identified gaps in BIM-related quality assessments. The methodology was applied to a case study, where clustering analysis and graph-based evaluations provided actionable insights into project coordination needs, particularly in identifying critical spaces that require focused interdisciplinary alignment.

A key contribution of this research is the integration of labeled property graphs (LPGs) with BIM data to facilitate comprehensive, real-time project evaluations. The ICG framework demonstrates how graph structures, enriched with meta-information, can improve coordination by revealing indirect relationships, which traditional BIM methods often overlook. This approach supports project stakeholders in making data-driven decisions, enhancing project quality and alignment with performance standards, while highlighting areas for optimization across multiple disciplines.

The potential for scalability and adaptability within this graph-based framework suggests that it could be applied to a broad spectrum of construction projects, expanding its use in different stages of the project lifecycle. However, some limitations emerged, particularly around data preprocessing, data integration from diverse sources, and the need for further experimentation to refine model accuracy. While these challenges point to areas for future research, the findings of this thesis provide a strong foundation for advancing BIM methodologies through graph-based data integration. This study highlights the effectiveness of graph databases in supporting complex project evaluations and demonstrates the promising role of graph-based approaches in the future of BIM and construction project management.

## References

- AEC3. (kein Datum). Von <https://www.aec3.de/> abgerufen
- Austern, G. B. (February 2024). *Incorporating Context into BIM Derived Data - Leveraging Graph Neural Networks for Building Element Classification*. Von [https://www.researchgate.net/publication/378274239\\_Incorporating\\_Context\\_into\\_BIM-Derived\\_Data-Leveraging\\_Graph\\_Neural\\_Networks\\_for\\_Building\\_Element\\_Classification](https://www.researchgate.net/publication/378274239_Incorporating_Context_into_BIM-Derived_Data-Leveraging_Graph_Neural_Networks_for_Building_Element_Classification)
- BIM Deutschland*. (kein Datum). Von <https://www.bimdeutschland.de/bim-deutschland/liste-der-standardisierten-anwendungsfallbezeichnungen>
- Boeykens, S., & Neuckermans, H. (September 2008). *Representational Limitations and Improvements in Building Information Modeling*. Von [https://www.researchgate.net/publication/30868070\\_Representational\\_Limitations\\_and\\_Improvements\\_in\\_Building\\_Information\\_Modeling](https://www.researchgate.net/publication/30868070_Representational_Limitations_and_Improvements_in_Building_Information_Modeling)
- Borrmann, A. &. (2021). *Building Information Modeling*. Springer.
- Borrmann, A., & Esser, S. (2024). Enhancing design coordination across disciplines through incremental model updates and Inter-discipline Conjunction Graphs. Quebec.
- Borrmann, A., & Scholz, M. (2016). Combining graph-based analysis and knowledge-based reasoning for managing infrastructure operation models. *Advanced Engineering Informatics*, 3(30), 352-367.
- buildingSmart. (kein Datum). *IFC Schema Specifications*. Von <https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/> abgerufen
- Diestel, R. (2017). *Graph Theory*. Springer.
- Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2008). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Architects, Engineers, Contractors and Fabricators*. Hoboken, NJ, USA: John Wiley & Sons.

- Esser, S., & Bormann, A. (2024). Enhancing design coordination across disciplines through incremental model updates and Inter-discipline Conjunction Graphs. *ICCBE*. Montreal.
- Gross, J., & Yellen, J. (2018). *Graph Theory and Its Applications*. Hall/CRC.
- Harary, F. (1969). *Graph Theory*. Addison-Wesley.
- Isaac, S., & Navon, R. (2009). Modeling building projects as a basis for change control. *Automation in Construction*.
- Ismail, A. (kein Datum). Von <https://github.com/ifcwebserver/IFC-to-Neo4j> abgerufen
- Ismail, A. &. (May 2018). *Building Knowledge Extraction from BIM/IFC Data for Analysis in Graph Databases*. Von [https://www.researchgate.net/publication/320391974\\_Analyzing\\_Building\\_Information\\_Using\\_Graph\\_Theory](https://www.researchgate.net/publication/320391974_Analyzing_Building_Information_Using_Graph_Theory)
- Ismail, A. N. (2017). *Application of graph databases and graph theory concepts for advanced analyzing of BIM models based on IFC standard*. Von [https://www.researchgate.net/publication/318600860\\_Application\\_of\\_graph\\_databases\\_and\\_graph\\_theory\\_concepts\\_for\\_advanced\\_analysing\\_of\\_BIM\\_models\\_based\\_on\\_IFC\\_standard](https://www.researchgate.net/publication/318600860_Application_of_graph_databases_and_graph_theory_concepts_for_advanced_analysing_of_BIM_models_based_on_IFC_standard)
- Ismail, A., Scherer, R., & Nahar, A. (2017). Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC Standard. Nottingham.
- Kayhani, N., McCabe, B., & Sankaran, B. (2023). BIM-based construction quality assessment using Graph Neural Networks.
- M. Bonduel, J. O. (2018). The IFC to linked building data converter - Current status. *CEUR Workshop proceedings*.
- Napps, D., Zahedi, A., König, M., & Petzold, F. (2022). Visualisation and graph-based storage of customised changes in early design phases. *39th International Symposium on Automation and Robotics in Construction*. Bogota.
- neo4j. (2024). *Neosemantics*. Von <https://neo4j.com/labs/neosemantics/>
- neo4j. (kein Datum). *neo4j Documentation*. Von <https://neo4j.com/docs>



- Pauwels, P., Zhang, S., & Lee, Y.-C. (2017). Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*, 73(0926-5805), 145-165.
- Sacks, R., Wang, Z., & Ying, H. (2024). Two Fundamental Questions Concerning BIM Data Representation for Machine Learning. *CIB W78*. Marrakesh.
- Sebastian Esser, A. B. (August 2024). *Enhancing design coordination across disciplines through incremental model updates and Inter-discipline Conjunction Graphs*. Von [https://www.researchgate.net/publication/381739603\\_Enhancing\\_design\\_coordination\\_across\\_disciplines\\_through\\_incremental\\_model\\_updates\\_and\\_Inter-discipline\\_Conjunction\\_Graphs](https://www.researchgate.net/publication/381739603_Enhancing_design_coordination_across_disciplines_through_incremental_model_updates_and_Inter-discipline_Conjunction_Graphs) abgerufen
- Solibri. (kein Datum). *www.solibri.com*. (Nemetschek)
- Stegeman, J. (22. July 2022). *www.neo4j.com/blog*. Von [https://neo4j.com/blog/what-is-knowledge-graph/?utm\\_source=LinkedIn&utm\\_medium=OrganicSocial&utm\\_campaign=KG--&utm\\_ID=&utm\\_term=&utm\\_content=-Blog--&utm\\_creative\\_format=&utm\\_marketing\\_tactic=&utm\\_parent\\_camp=&utm\\_partner=&utm\\_persona=](https://neo4j.com/blog/what-is-knowledge-graph/?utm_source=LinkedIn&utm_medium=OrganicSocial&utm_campaign=KG--&utm_ID=&utm_term=&utm_content=-Blog--&utm_creative_format=&utm_marketing_tactic=&utm_parent_camp=&utm_partner=&utm_persona=)
- Tauscher, E. &. (August 2013). *Analyzing Building Information Using Graph Theory*. Von [researchgate: https://www.researchgate.net/publication/296602879\\_Generic\\_BIM\\_queries\\_based\\_on\\_the\\_IFC\\_object\\_model\\_using\\_graph\\_theory](https://www.researchgate.net/publication/296602879_Generic_BIM_queries_based_on_the_IFC_object_model_using_graph_theory)
- Technology, K. I. (kein Datum). <https://www.ifcwiki.org/index.php?title=File:AC20-FZK-Haus.ifc>. Von <https://www.ifcwiki.org/index.php?title=File:AC20-FZK-Haus.ifc>
- Teclaw, W., Rasmussen, M., Labonnette, N., & Hjelseth, E. (2023). The semantic link between domain-based BIM models. *Conference: 11th Linked Data in Architecture and Construction Workshop*. Matera, Italy.
- Whelton, M., & Ballard, G. (2002). Wicked Problems in Project Definition. *Proceedings of the International Group for Lean Construction 10th Annual Conference*. Brazil.

---

Yalcinkaya, M., & Singh, V. (2015). Patterns and trends in Building Information Modeling (BIM) research: A Latent Semantic Analysis. *Automation in Construction*, 59(0926-5805), 68-80.

## Table of Figures

Figure 5 Impact of Changes within Project Isaac,Navon(2009) .....	9
Figure 6 Identification of the impact of the additional user activity on the project. S. Isaac, R. Navon / Automation in Construction 18 (2009).....	10
Figure 7 IFC2LBD Tool Landing Page.....	18
Figure 8 IFC Webserver .....	20
Figure 9 Emergency Routes drawn on the floor plan (Ismail, Slusarczyk 2018).....	21
Figure 10 Meta Data and Instance Data Diagram (neo4j) .....	22
Figure 11 Fundamental entity types derived from IfcRoot .....	23
Figure 12 IFC Abstract objectified relationships .....	24
Figure 13 IFC Entity Attributes and Relationship Attributes .....	25
Figure 14 Translation of IFC Schema to IMG (Ismail, Scherer, Nahar 2017) .....	25
Figure 15 A Diagram representing the sub-graphs from the research paper Wang,Crete(2023).....	26
Figure 16 Framework of Semantic Enrichment Wang,Crete(2023) .....	27
Figure 17 Architectural Model and Room Model with Furniture (Esser, Borrmann 2024) .....	28
Figure 18 Solibri Information Take Off - ICG Implementation (Borrmann, Esser 2024) .....	29
Figure 19 Detected Relationships (Kayhani, McCabe, Sankharan 2023).....	29
Figure 20 List of Use Cases .....	30
Figure 21 Use Case LOIN Connection Diagram.....	31
Figure 22 List of Model Check Criteria from AEC3 .....	32
Figure 23 Data for AI (Sacks, Wang, Ying 2024).....	33
Figure 24 Scope of the Project .....	37
Figure 25 Interdisciplinary Conjunction Graph Structure .....	39
Figure 26 MEP Sub-graph with its sub-domains.....	42
Figure 27 Generation Workflow Diagram.....	44
Figure 28 IFC Object Graph Workflow.....	46

Figure 29 BIM-M Meta Graph Model Check Criteria Node .....	47
Figure 30 Use Cases Graph Implementation.....	48
Figure 31 Node Labels and Relationship Labels that are connected to the Model Check Criteria.....	48
Figure 32 BIM-M Meta Graph .....	49
Figure 33 Node and Relationship Structure with main attributes .....	50
Figure 34 IFC Object Graph Generation Workflow .....	51
Figure 35 ROOM_CONTAINS Semantic Enrichment Workflow .....	53
Figure 36 Use Case 040 Coordination Enhancement Schema .....	54
Figure 37 Schema Enhancement based on ROOM_CONTAINS .....	55
Figure 38 ICG Analysis Workflow .....	55
Figure 39 Screenshot from the Coordination Model .....	60
Figure 40 Section from the Coordination Model .....	60
Figure 41 IFC Schema Listings from buildingSmart .....	61
Figure 42 IFC Hierarchy in Graph Structure .....	63
Figure 43 Class Structure of IfcElement Tree.....	64
Figure 44 Sub-Class Structure of IfcElement.....	64
Figure 45 AC3-FZK-Haus.ifc - Test Project .....	66
Figure 46 Script Output without optimization .....	66
Figure 47 Node Properties.....	68
Figure 48 Query Results.....	71
Figure 49 Count of Nodes by IFC Entity Bar Chart.....	72
Figure 50 Distribution of Nodes by IFC Entity.....	72
Figure 51 Count of Nodes by Discipline Bar Chart .....	73
Figure 52 Distribution of Count of Nodes by Discipline.....	73
Figure 53 BIM-M Project Evaluation Criteria Mapped.....	74
Figure 54 Relationships of a IfcSpace Node.....	75
Figure 55 Scatter Plot of Room Contains Per Sqm Clusters .....	76
Figure 56 Table of Room_Contains per sqm Clusters.....	76
Figure 57 Bubble Chart of IfcSpace Density by Area .....	77

Figure 58 Scatter Plot of Unique Disciplines in ROOM\_Contains against Floor Area 78

Figure 59 Table of Examples from Clusters of IfcSpaces by RoomContains Count coming from Unique Disciplines ..... 78

Figure 60 Table of Clusters from Spaces with IfcFlowTerminal stemming Elements 80

Figure 61 Bubble Chart of Component Counts with Size..... 81

Figure 62 Table of Clusters of IfcSpace Nodes ..... 82

Figure 63 Bar Chart of Count of Rooms by Cluster Type ..... 82

Figure 64 Scatter Plot Charts of Cluster Groups ..... 83

Figure 65 Table of Example IfcSpaces from Clustering Results ..... 83

Figure 66 Distribution of Rooms by Coordination Focus Point Groups..... 84

Figure 67 Scatter Plot of Room Floor Areas grouped by Coordination Focus Points 85

Figure 68 Heat Map Chart of Coordination Focus Point Distribution Across Room Sizes ..... 86

Figure 69 Query to export nodes ..... 87

Figure 70 Query to export edges ..... 87

Figure 71 Node Tensor Shape ..... 88

Figure 72 Edge Tensor Process ..... 88

Figure 73 Model used in the Process ..... 89

Figure 74 Code Snippet for the training process ..... 90

## Appendix A

### Source Code

#### 1. BIM Management Meta Graph

- **bim-m\_meta\_graph.txt**

Cypher commands for the BIM Management meta graph.

#### 2. IFC Meta Graph

- **ifc\_meta\_graph.py**

Python script that generates Cypher commands from the XMD file of IFC Schema.

- **ifc\_create\_nodes.txt**
- **ifc\_create\_relationships.txt**

Generated txt files that contain Cypher queries that generate the nodes and relationships.

- **ifc\_meta\_graph\_exceptions.txt**

Another txt file with cypher queries that handles exceptions.

#### 3. IFC Object Graphs

##### ifc to cypher.py

Python script to generate Cypher commands from IFC Files.

#### 4. BCF to Cypher

##### bcf to cypher.py

Python script to generate Cypher commands from BCF Files.

#### 5. Cypher Importer

##### cypher\_batch\_importer.py

Python script to import .txt format files into Neo4j Database.

#### 6. Graph Analysis Notebook

##### lca\_analysis.ipynb

Jupyter Notebook file to analyze the Graph database on Neo4j.

## Statement of Independent Work

Hiermit erkläre ich, dass ich die vorliegende Bachelor-Thesis selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ich versichere außerdem, dass die vorliegende Arbeit noch nicht einem anderen Prüfungsverfahren zugrunde gelegen hat.

München, 10.11 2024

Koray Inal

---

Vorname Nachname

Koray Inal

Christoph-Probst Straße 12 / 332

80805 München

koray.inal@tum.de