TECHNISCHE UNIVERSITÄT DRESDEN

Institute of Construction Informatics, Faculty of Civil Engineering

Ontologies for Structural Engineering Models

Ontologien für Tragwerksmodelle

by

Prateek Agrawal

from

Ramnagar, India

A Thesis submitted to the Institute of Construction Informatics, Faculty of Civil Engineering, University of Technology Dresden in partial fulfilment of the requirements for the degree of

Master of Science

Responsible Professor: Prof. Dr.-Ing. habil. Karsten Menzel

Second Examiner: Prof. Dr.-Ing. Raimar Scherer

Advisor: Janakiram Karlapudi, MSc

Dresden, 26th February, 2021

(insert here the original sheet with your task)

(insert here the original sheet with your task)

Declaration of originality

I confirm that this assignment is my own work and that I have not sought or used the inadmissible help of third parties to produce this work. I have fully referenced and used inverted commas for all text directly quoted from a source. Any indirect quotations have been duly marked as such.

This work has not yet been submitted to another examination institution – neither in Germany nor outside Germany – neither in the same nor in a similar way and has not yet been published.

Dresden,

Place, Date

(Signature)

Acknowledgements

Foremost, I would like to express my sincere gratitude to Prof. Dr.-Ing. habil. Karsten Menzel and Prof. Dr.-Ing. Raimar Scherer for providing me this wonderful opportunity to work on my master thesis under the Institute of Construction Informatics, TU Dresden.

Furthermore, my profound gratitude to my supervisor Mr. Janakiram Karlapudi for his constant support, guidance, patience and immense knowledge. I am fortunate to have him as my supervisor.

Finally, I would like to thank my family and friends for their support and inspiration throughout my academic life.

Abstract

Introduction of digital technologies such as Computer-Aided Design (CAD), Building Information Modelling (BIM) in Architecture, Engineering, Construction and Operation (AECO) sector have not just enhanced the designing, analyzing and planning process for a structure but also improved data exchange and data management process by documenting all physical and functional information for a construction project. Efficient data sharing between diverse domains of AECO sectors is one major aim of BIM technologies. Usage of domain specific software and one-to-one interaction between diverse software for data transfer resulted in different data formats, several data interfaces and ineffective collaboration. In order to better data exchange and sharing process between different domains of Architectural, Engineering and Construction (AEC), Industry foundation Classes (IFC) schemas was developed but still interoperability issues such as data loss, misinterpretation of information etc. frequently occur during bidirectional data exchange using IFC specially between architectural domain and structural engineering domain (SED). Semantic Web technologies offer the possibility to integrate data from different data model and diverse domains using web and can be considered as an alternate option to solve interoperability issues.

In this research study, reasons for inadequate interoperability between SED and architectural domain is analyzed and validated by case study. Further, solutions proposed by researchers to achieve efficient data exchange using syntactic approach and semantic technologies were presented using case studies. Moreover, this research study uses semantic web technology to ensure semantic interoperability between architectural domain and SED which includes ontology development, query of heterogeneous information from architectural and structural analysis model and construction of query results into developed ontologies to check the applicability of ontologies. Completeness of proposed approach is validated through a case study.

Table of Contents

1 Introduction ... 1

1.1 Motivation ... 1

1.2 The Objective of the thesis: ... 2

1.3 Problem Statements ... 3

1.4 Organization of Thesis ... 3

2 State of the Art and researches ... 4

2.1 Building Information Modelling (BIM) ... 4

2.1.1 History of BIM in ACE industry ... 4

2.1.2 BIM in AEC Industry ... 5

2.2 Interoperability in AEC industry ... 7

2.2.1 Industry Foundation Classes (IFC) ... 8

2.2.2 IFC interoperability Framework ... 11

2.2.3 Interoperability challenges between Architectural and Structural domain .. 12

2.2.4 Case Study ... 16

2.3 Researches and proposed solutions ... 19

2.3.1 Case Study ... 20

2.3.2 Linked data and Ontologies approach for semantic interoperability ... 23

2.4 Ontology ... 24

2.4.1 Core components of Ontology ... 25

2.4.2 Ontology Description languages ... 27

2.4.3 Possible usage and need of Ontology in AEC industry ... 30

2.4.4 Case Study ... 32

3 Methodology for development and validation of Ontology ... 35

3.1 Assessment of IFC Structural Analysis domain ... 35

3.1.1 Zone and Building Site ... 36

3.1.2 Structural elements ... 38

3.1.3 Material and Property Definition ... 41

3.1.4 Load Definition ... 42

3.1.5 Load combination and Analysis Results ... 44

3.2 Development of Domain Ontologies ... 46

3.2.1 Tool and Ontology development process ... 47

3.2.2 Annotations and Namespaces 48
3.2.3 Building Element Ontology (BEO) 50
3.2.4 Building Load Ontology (BLO) 53
3.2.5 Building Element Representation Ontology (BERO) 57
4 Construction of knowledge base 64
4.1 Query of architectural information 65
4.1.1 IfcBuilding, IfcBuildingStorey and IfcSite information 65
4.1.2 Structural members and properties 67
4.2 Construction of query results into developed ontology 69
5 Conclusions and Future Work 72
5.1 Conclusions 72
5.2 Future Work 73
6 References 74
7 Appendices 77

LIST OF FIGURES

FIGURE 1: KEYWORDS WITHIN INTEROPERABILITY-RELATED ISSUES IN BIM [2] 1
FIGURE 2: BIM ENVIRONMENT [3] 4
FIGURE 3: DATA TRANSFER AMONG DIFFERENT BIM MODELS [4] 6
FIGURE 4: USAGE OF BIM DURING ENTIRE BUILDING LIFE CYCLE IN AEC INDUSTRY [5] 6
FIGURE 5: IFC BASED DATA INTEROPERABILITY BETWEEN VARIOUS SOFTWARE TOOLS [9] 7
FIGURE 6: DATA SCHEMA ARCHITECTURE WITH CONCEPTUAL LAYERS [12] 9
FIGURE 7: IFC DATA MODEL INCLUDING GRAPHICAL AND NON-GRAPHICAL DATA [13] 11
FIGURE 8: IFC INTEROPERABILITY LAYERS [7] 11
FIGURE 9: DATA INTEROPERABILITY BETWEEN DIFFERENT DOMAINS [9] 13
FIGURE 10: INFORMATION DELIVERY PROCESS BASED ON IDM AND MVD [17] 14
FIGURE 11: 3D MODEL OF STRUCTURE DESIGNED USING REVIT SOFTWARE [21] 16
FIGURE 12: LEFT: REVIT MODEL AND RIGHT: SCIA ENGINEER MODEL [21] 17
FIGURE 13: DISJOINT NODES IN SCIA ENGINEER MODEL [21] 18
FIGURE 14: GEOMETRIC INCONSISTENCY FINDINGS [21] 18
FIGURE 15: POSSIBLE WAYS TO GENERATE A STRUCTURE ANALYSIS MODEL [18] 19
FIGURE 16: DATA TRANSFORMATION WORKFLOW BY [4] 19
FIGURE 17: INFORMATION DELIVERY PROCESS BASED ON USER-DEFINED ERS [17] 20
FIGURE 18: ERS BASED STRUCTURAL ELEMENTS [17] 21
FIGURE 19: FLOW DIAGRAM OF THE EXCHANGE MODEL DELIVERY TOOL [17] 22
FIGURE 20: INFORMATION DELIVERY FROM STRUCTURE DOMAIN TO ARCHITECTURE DOMAIN [17] 23
FIGURE 21: TYPES OF ONTOLOGY [30] 25
FIGURE 22: CLASSES, CLASS HIERARCHY AND DISJOINT CLASSES IN PROTÉGÉ 26
FIGURE 23: INDIVIDUALS IN PROTÉGÉ 27
FIGURE 24: RDF TRIPLES 28
FIGURE 25: USE OF RDF SCHEMA VOCABULARY FOR DESCRIBING CLASSES AND PROPERTIES [33] 29
FIGURE 26: RDF SCHEMA IN RDF/XML FILE FORMAT 29
FIGURE 27: SPARQL QUERY [35] 30
FIGURE 28: THE TECHNICAL ROADMAP FOR BUILDING SUPPORT BY BUILDINGSMART [36] 31
FIGURE 29: RESEARCH METHODOLOGY PROCESS [39] 33
FIGURE 30: TRANSFORMATION OF IFC ENTITIES INTO RDF/OWL CLASSES [39] 33

FIGURE 31: RECALL RESULTS FOR DIFFERENT FEATURES OF CASE STUDY [39] 34
FIGURE 32: RESEARCH METHODOLOGY PROCESS 35
FIGURE 33: RELATIONSHIP BETWEEN ZONE AND SPACE [12] 36
FIGURE 34: SPATIAL STRUCTURE ELEMENT COMPOSITION [12] 37
FIGURE 35: REFERENCE IFC ENTITIES FOR STRUCTURAL MEMBERS [12] 38
FIGURE 36: IFC SCHEMAS FOR GEOMETRIC REPRESENTATION AND OBJECT PLACEMENT [12] 39
FIGURE 37: REPRESENTATION ITEM STYLE [12] 40
FIGURE 38: TOPOLOGY REPRESENTATION: EDGE AND VERTEX [12] 41
FIGURE 39: RELATIONSHIP BETWEEN OBJECT AND ITS PROPERTIES [12] 41
FIGURE 40: BEAM OCCURRENCE GRAPH IN IFC DATA [12] 42
FIGURE 41: RELATIONSHIP BETWEEN STRUCTURAL ACTIVITY AND STRUCTURAL ITEM [12] 43
FIGURE 42: STRUCTURAL LOAD GRAPH FOR STRUCTURAL CURVE MEMBER IN IFC DATA [12] 44
FIGURE 43: IFC SCHEMAS FOR LOAD COMBINATION AND STRUCTURAL ANALYSIS [12] 44
FIGURE 44: LOAD GROUP IN IFC DATA [12] 45
FIGURE 45: RELATIONSHIP BETWEEN LOAD GROUP, ANALYSIS MODEL AND RESULT GROUP [12] 46
FIGURE 46: INTERCONNECTION BETWEEN DEVELOPED DOMAIN ONTOLOGIES 49
FIGURE 47: BASIC STRUCTURE OF BUILDING ELEMENT ONTOLOGY 50
FIGURE 48: BUILDING ELEMENT AND RELATED OBJECT PROPERTIES OF BEO IN PROTÉGÉ 51
FIGURE 49: BUILDING ELEMENTS, BUILDING AND PROPERTY RELATIONSHIP GRAPH 52
FIGURE 50: REASONING AND INFERRED INFORMATION THROUGH REASONER 53
FIGURE 51: OVERVIEW OF BUILDING LOAD ONTOLOGY 54
FIGURE 52: LOAD CLASS HIERARCHY AND RELATIONSHIP WITH PROPERTY GRAPH 55
FIGURE 53: CLASS HIERARCHY AND DATA PROPERTIES OF BLO 56
FIGURE 54: OVERVIEW OF BUILDING ELEMENT REPRESENTATION ONTOLOGY 57
FIGURE 55: BERO SUPER CLASSES AND RELATIONSHIP GRAPH 58
FIGURE 56: CLASS HIERARCHY AND "CURVE" CLASS DESCRIPTION 59
FIGURE 57: DESCRIPTION OF CLASS "POINTS" 60
FIGURE 58: OVERVIEW OF STRUCTURAL ANALYSIS ONTOLOGY 61
FIGURE 59: GRAPHICAL REPRESENTATION OF CLASS HIERARCHY FOR "LOAD COEFFICIENT" CLASS AND "BUILDING ELEMENT SAFETY FACTOR" CLASS 62
FIGURE 60: CLASS HIERARCHY OF "STRUCTURAL ANALYSIS" CLASS AND "ANALYSIS RESULT" CLASS 63
FIGURE 61: LEFT: ARCHITECTURAL MODEL IN REVIT AND RIGHT: STRUCTURAL ANALYSIS MODEL IN RSTAB 64

Figure 62: Query of information for *IfcBuilding* entity using SPARQL....65
Figure 63: Graphical representation of query results....66
Figure 64: Query of information for IfcBuildingStorey and IfcSite entity using SPARQL66
Figure 65: SPARQL query for IfcStructuralCurveMember information....67
Figure 66: SPARQL query to obtain values of different section property parameters....68
Figure 67: Results for section properties....69
Figure 68: SPARQL query for construction of results in BEO ontology....69
Figure 69: graphical representation of result of "Construct" query....70
Figure 70: Instances of class beo:BuildingElements....71
Figure 71: Data values of instance IfcStructuralCurveMember_2865....71
Figure 72: Object property hierarchy of BEO....77
Figure 73: Graphical representation of class hierarchy of BERO....77
Figure 74: Object property hierarchy of BERO....78
Figure 75: Class hierarchy and object property hierarchy of BLO....78
Figure 76: Class hierarchy of SAO....79
Figure 77: Data properties of SAO....79

List of Tables

TABLE 1: IFC DATA FILE FORMATS 8

TABLE 2: DATA REQUIREMENT FOR ONTOLOGY CONSTRUCTION 36

TABLE 3: NAMESPACES USED FOR ONTOLOGY 49

TABLE 4: QUERY RESULTS FOR BUILDING, SITE AND BUILDING STOREY 67

TABLE 5: QUERY RESULTS FOR STRUCTURAL MEMBERS 68

List of Abbreviations and Symbols

BIM	Building Information Modelling
IFC	Industry Foundation Classes
SED	Structural Engineering Domain
AEC	Architectural, Engineering and Construction
STEP	Standard for the Exchange of Product Model
XML	Extensible Markup Language
MVD	Model View Definition
IDM	Information Delivery Manual
PM	Process Map
FP	Functional Part
ER	Exchange Requirement
BPMN	Business Process Modelling Notation
URI	Uniform Resource Identifier
RDF	Resource Description Framework
OWL	Web Ontology Language
SPARQL	Protocol and RDF Query Language
BEO	Building Element Ontology
BLO	Building Load Ontology
BERO	Building Element Representation Ontology
SAO	Structural Analysis Ontology

1 Introduction

1.1 Motivation

Due to many stakeholders and technologies, Architectural, Engineering, Construction and Operation (AECO) industry provides a stimulating environment for collaboration and sharing of project information. Engineers and experts in this industry use diverse software tools for their business tasks and then cooperate with each other for interoperability. However, due to complexity of data and size of projects, efficient and uninterrupted sharing of data is still questionable, which leads to loss of time and finance and this problem is dominant between Architectural and Structural Analysis domain. With the substantial progress of Building Information Modelling (BIM) technologies and development of open standards like Industry Foundation Class (IFC), problems of inefficient data sharing are tackled to some extend but still major issues like data loss, geometric misrepresentation etc. are commonly noticed due to extreme heterogeneity and complexity of data.

Hence, for improving the data and information exchange process in this multidisciplinary industry, investigation and development of alternative technologies, rather than complete dependency on BIM and IFC, are crucial and one such possible way is "Ontology based Sematic Web Technology". Concept of Semantic Web and Ontological methods are already being used in AEC industry for decades and offers the possibility of data combination from diverse data model and multiple domains using the web [1].

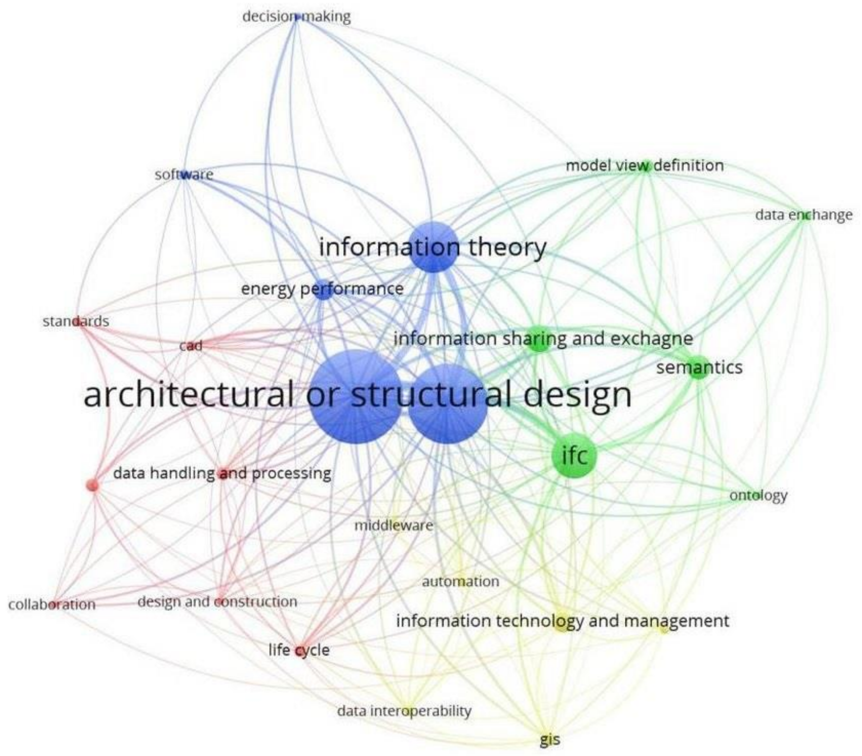


Figure 1: Keywords within interoperability-related issues in BIM [2]

1.2 The Objective of the thesis:

Fundamental objectives of this research study are mentioned below:

1) **Comprehensive analysis on interoperability in Structural Engineering Domain (SED)**

Although with the development of BIM technologies and open standards like IFC, issues related to interoperability has been reduced to some extent but still manual efforts are required for seamless data exchange. Complete analysis of problems related to interoperability is crucial and further research for alternative solutions for undertaking these problems are important.

2) **Detailed study of Open BIM IFC schema concerning data requirements**

IFC is an open data scheme developed by buildingSMART for efficient data exchange and data sharing between heterogeneous software application. Several prominent software tools related to Structural Engineering domain have import and export capabilities of IFC data model. Therefore, investigation and proper understanding of IFC schemas are important for development of a framework that can reduce interoperability issues in mentioned domain.

3) **Development and demonstration of Ontological model for denoting architectural and structural analysis data**

Ontologies are formal explicit specification of a shared conceptualization of a domain of interest and are based on descriptive logical languages like OWL, RDF schema. These languages provide semantics to information which can be shared, captured, stored, reused and linked using Semantic Web technologies. Further, using ontologies extra information can be inferred and queried out from building models. Hence, ontologies and other Semantic Web approaches open possibilities for a proper and efficient information system in AEC industry for data exchange and management. In this thesis, domain ontology for structural analysis domain was developed and used for denoting information from architectural domain and structural analysis domain.

1.3 Problem Statements

Structural analysis is one of the most crucial process in pre-construction phase of the building and provides important information for building construction. Since, structural analysis building model is created from architectural model, efficient and meaningful data transfer between these two domains are vital. Studies about data exchange problems between SED and architectural domain are necessary and further need to find possible solutions to tackle data exchange problems is significant. So, this research study focusses on following questions and complications:

1. The reasons for interoperability issues in SED.
2. Necessity for alternative solutions for reduction of data loss and misinterpretation during interoperability.
3. Why ontological models can be a solution for inefficient data transfer in SED?
4. What are the factors for development of a domain ontology?
5. Issues and steps for development of ontology model.

1.4 Organization of Thesis

This thesis is summarized in major five sections. The first section includes motivation and key objectives of thesis. The next section contains all the theoretical literature and concepts of the research which is divided into four parts and further confirmed by case studies. The third section corresponds to the methodology for development of ontology and accessing the required data for ontology development. Demonstration use case for applicability of the developed ontologies along with results and discussion of the data exchange demonstration being the fourth section of thesis. The last section mentions to drawn conclusion and recommended future work for improvement of interoperability framework and further use of developed ontologies.

2 State of the Art and researches

2.1 Building Information Modelling (BIM)

BIM is an intelligent model-based methodology that gives understanding to help you plan, design, build and manage buildings, infrastructures and provides intelligence for individual structural elements (columns, slabs, etc.) and, in addition to standard spatial relationships, provides system- and building-wide knowledge and information (system flows or building loads). The attributes of "intelligence" to objects include parametrically defined graphical and non-graphical details, giving the ability to represent geometric and functional relationships between building elements to designers, structural engineers, project managers, and developers. This data feeds an automated database that, in turn, covers all design documents and construction schedules for building project [3]. BIM technologies help in interconnection of different domains and act like a central agency for data transfer and work management between diverse discipline of AEC industry as shown in figure 2.

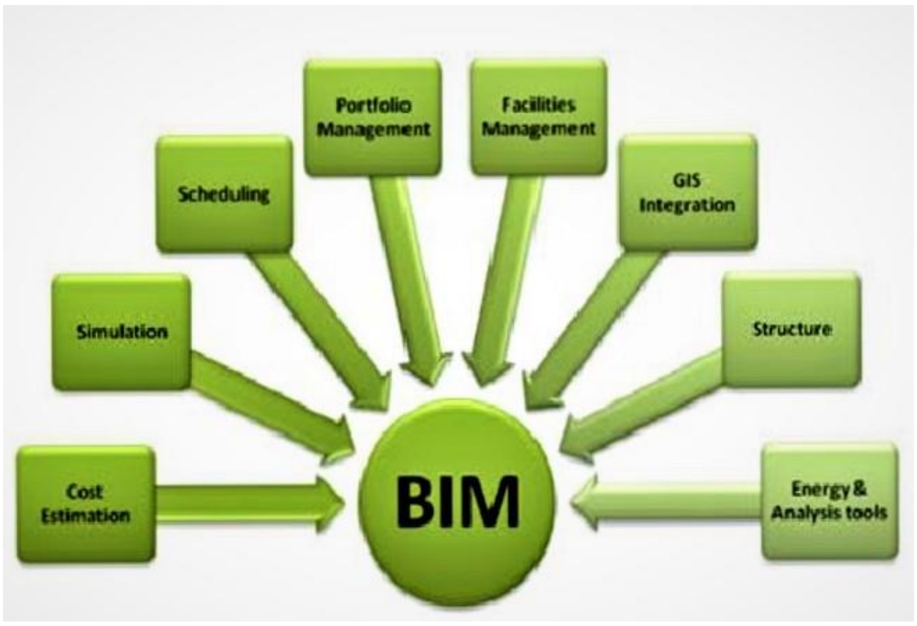


Figure 2: BIM environment [3]

According to American National BIM Standard (NBIMS), a definition of BIM is as follows: "BIM is defined as a digital representation of physical and functional characteristics of a facility. As such it serves as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its lifecycle from inception onward" [4].

2.1.1 History of BIM in ACE industry

Before 1980's building design documents were modelled by drawing lines on papers but with several economic changes and globalization of market concept of product modelling gained momentum. With introduction of computers, manual drafting of building design were replaced by computer aided drafting (CAD) which revolutionized the construction process. In the early 1980s, the use of CAD in the AEC industry accelerated and enabled for improved efficiency in the time taken for drafting designs.

The issue with CAD solutions was that details about building elements, materials and resources that the industry needed could not be fully optimized for planning and managing them. Therefore, using the concepts of CAD, BIM's conceptual progress started in 2002 via Autodesk, where information on building objects could be stored and handled for a project in databases [5]. Thus, concepts of digital BIM technologies were introduced in AEC industry after decades of research which transformed the way building systems are designed, constructed, managed and maintained.

Overall, BIM is a logical successor of traditional Computer aided design (CAD) along with a lot of advantages which can be summarized as:

- Using BIM, a building data model can be represented in multi-dimensional model data such as 3D (geometrical information), 4D (3D with time dimension and planning), 5D (4D with cost dimension) and so on.
- By using open standards for data exchange such as IFC (Industry Foundation Classes) through BIM platform, a better work efficiency and building quality can be achieved with less cost and resources because it provides an opportunity for cooperative work and better communication between different disciplines of project through seamless data transfer and improved information flow.
- BIM offers an environment for the purpose of virtual construction and visual analysis which provides the opportunity to discover and tackle potential problems in early stage of project and can help to save time and resources [4].

2.1.2 BIM in AEC Industry

BIM is being used in a variety of growing industries all over the world like Architecture, Engineering and Construction (AEC), building design, construction management, building energy analysis as well as in industrial and natural resources projects. Specially, in AEC industry, due to involvement of high finances, complex task and heterogeneous data formats BIM techniques enabled great advantages in designing, analyzing and documenting all physical and functional information of a building and construction project. Due to multiple disciplines, data sharing and exchange between different software tools is an inevitable need in this industry specially between architectural domain and structural domain. The architectural domain and structural domain require two types of building models: physical model for architectural design and discretized model for structural details and for further analysis. Starting with the architects, they need to effectively share the architectural model with the owners and structural engineers and track the engineering phase as structural design relies upon the effective sharing of information and data created from the architectural design. The structural design is a complex and dynamic process, in which the structural mechanical behavior under the impact of different loads should be determined and analyzed. As shown in Figure 3 structural model consists of two different models i.e., detail model and analysis model. Structural detail models mainly include structural geometrical shapes, section properties and material details extracted from architectural model whereas analysis models contain information like member size, spacing, layout, bar size etc. Further loading conditions, boundary conditions etc. properties are determined by structural engineer for analysis. With the adoption of BIM techniques these stages can be integrated which enables reduction in time, effort, cost and errors [4].

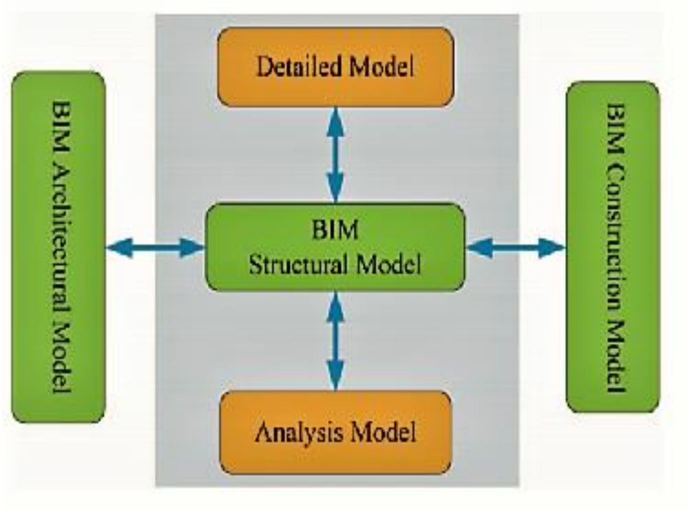
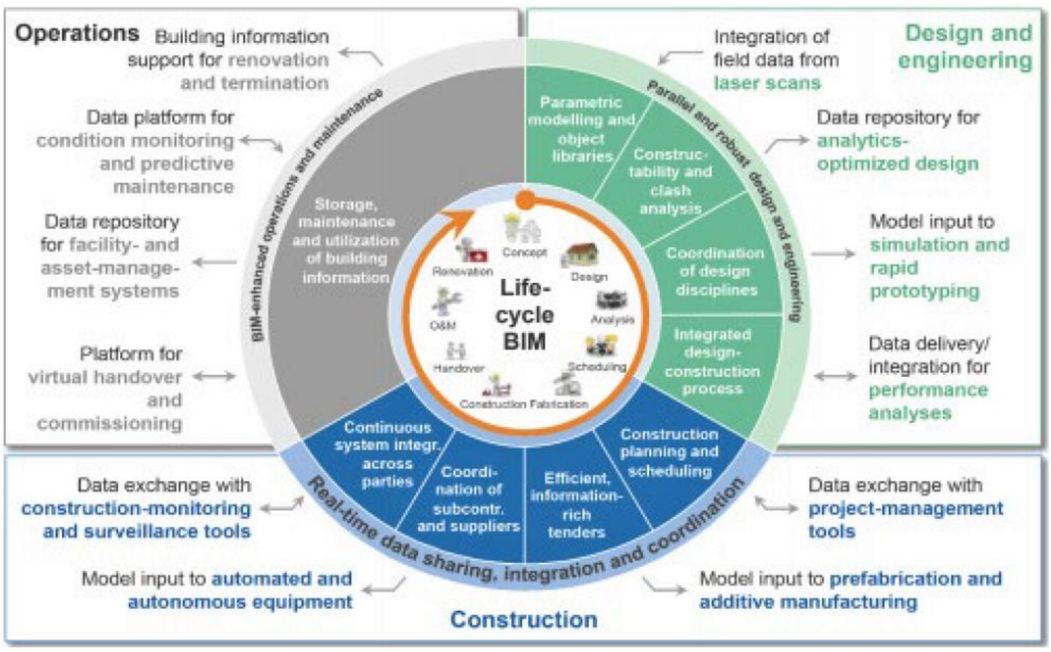


Figure 3: Data transfer among different BIM models [4]

BIM model contain a large range of information regarding the product and process [6] hence, BIM techniques and tools have transformed ACE industry by improving the quality and productivity of building projects. BIM's primary vision is to produce real, accurate, relevant and easily editable data and make it accessible to numerous project stakeholders and the key condition for fulfilling this aim is efficient interoperability. Figure 4 illustrates the collaborative nature of BIM technologies for different domains of AEC industry by providing a fundamental platform for different process of AEC industry.



Operations
Building information support for renovation and termination
Data platform for condition monitoring and predictive maintenance
Data repository for facility- and asset-management systems
Platform for virtual handover and commissioning
BIM-enhanced operations and maintenance
Storage, maintenance and utilization of building information
Design and engineering
Integration of field data from laser scans
Data repository for analytics-optimized design
Model input to simulation and rapid prototyping
Data delivery/ integration for performance analyses
Parallel and robust design and engineering
Parametric modelling and object libraries
Construc-tability and clash analysis
Coordination of design disciplines
Integrated design-construction process
Life-cycle BIM
Concept
Design
Analysis
Scheduling
Fabrication
Construction
Handover
O&M
Renovation
Continuous system integr. across parties
Coordi-nation of subcontr. and suppliers
Efficient, information-rich tenders
Construction planning and scheduling
Real-time data sharing, integration and coordination
Data exchange with construction-monitoring and surveillance tools
Data exchange with project-management tools
Model input to automated and autonomous equipment
Model input to prefabrication and additive manufacturing
Construction

Figure 4: Usage of BIM during entire building life cycle in AEC industry [5]

Since, data sharing is one of the key factors of collaboration throughout building life cycle specially between architectural and structural model, it is important to create a suitable and efficient environment which allows uninterrupted sharing of data. Various efforts from both academia and industry have been made to address inconsistency in data interoperability but still problems like missing objects, geometric distortion and further data losses are usually observed in these domains. For better data sharing open data exchange schema such as IFC have been developed, which allows users from different software tools to share data and files. As a result, BIM and IFC act as a central key point in building lifecycle involving all stakeholders by providing design, delivery and maintenance of building assets along with efficient managing and sharing of data.

2.2 Interoperability in AEC industry

According to European Interoperability Framework (EIF) "Interoperability means the ability of information and communication technology (ICT) systems and of business processes they support to exchange data and to enable the sharing of information and knowledge" [7]. AEC industry typically involves two separate domains: architectural domain and structural domain. As stated, architectural domain focuses on architectural design which mainly includes geometrical information of building elements, spacing arrangements etc. whereas structural domain focuses on structural design and mechanical behaviors of building elements along with loading conditions and necessary boundary conditions for structural analysis. Most of this information for structural domain is extracted from architectural model so, it is quite important to create a digital database that gives freedom to transfer complex proprietary data generated by heterogeneous software tools with ability to transfer true meaning of data. Due to complexity of generated data by diverse domains of AEC industry and misinterpretation of data among various domain experts, efficient data transfer is complex and challenging [8].

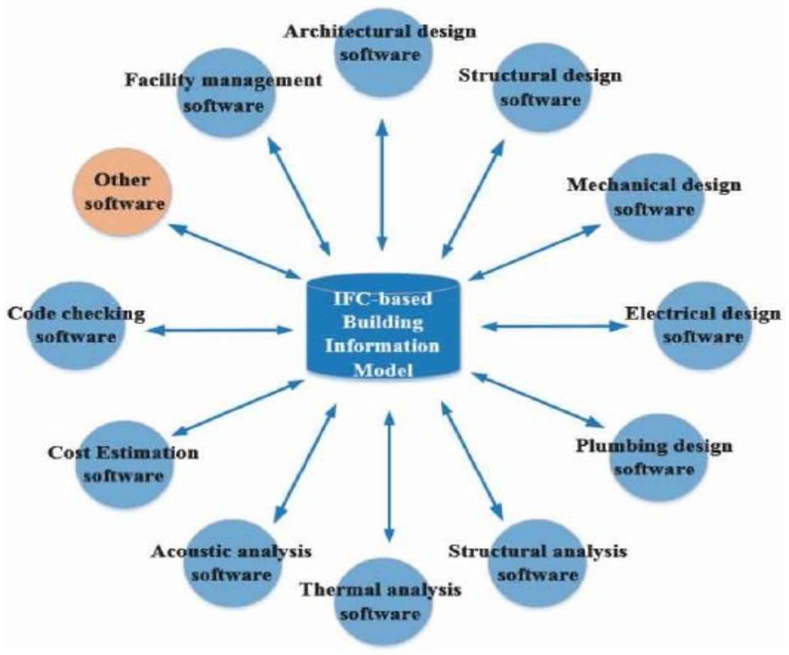

Architectural design software
Facility management software
Structural design software
Other software
Mechanical design software
Code checking software
IFC-based Building Information Model
Electrical design software
Cost Estimation software
Plumbing design software
Acoustic analysis software
Structural analysis software
Thermal analysis software

Figure 5: IFC based data interoperability between various software tools [9]

For efficient transfer of information between different heterogeneous tools in these domains certain open data exchange schema has been developed like IFC, Green Building Extensible Markup Language (gbXML) among which IFC is most widely used for data exchange between AEC software. IFC allows conversion of heterogeneous data format from diverse disciplines of AEC industry into a common file format (.ifc file) and enables real time data exchange as shown in Figure 5. This process enables the systematic and fast flow of data from one domain to other and helps in time management and reduction of extra human efforts.

2.2.1 Industry Foundation Classes (IFC)

IFC is open and neutral data format for openBIM with the aim of improving data exchange practices among different software tools developed and maintained by buildingSMART registered as ISO 16739. IFC data schema is defined in EXPRESS data definition language and XML Schema Definition (XSD) language and follows the Standard for the Exchange of Product Model Data (STEP) for physical file and Extensible Markup Language (XML) file format for exchanging data. The XML schema is automatically generated from IFC EXPRESS source via the "XML representation of EXPRESS schema and data", defined as ISO10303-28 ed. 2 to ensure the same data consistency and for conversion of data in both direction [10]. IFC has been developed to integrate building information throughout the life-cycle of the project, from planning through design and assessment, construction to post-occupancy, service and maintenance.

Table 1: IFC data file formats

File format	**Specification**
.ifc	It is standard format of IFC files based on STEP physical file format and should validate according to IFC EXPRESS specification.
.ifcXML	It is XML based representation of IFC data which is required by software for data exchange.
.ifcZIP	It is compressed IFC file with much smaller file size. It requires to have single .ifc or .ifcXML data file in the main directory of the zip archive.

"IFC schema is a standardized data model that codifies, in a logical way...

...the **identity** and **semantics** (name, machine-readable unique identifier, object type or function) ...

...the **characteristics** or **attributes** (such as material, color, and thermal properties) ...

...and **relationships** (including locations, connections, and ownership) ...

...of **objects** (like columns or slabs) ...

...**abstract concepts** (performance, costing) ...

...**processes** (installation, operations) ...

...and **people** (owners, designers, contractors, suppliers, etc.)" [11].

IFC released latest version in 2016 as IFC4 – Addendum 2 (IFC4 ADD2) consisting 766 entities. IFC has been classified in four conceptual layers as shown in Figure 6, where each schema is assigned to one conceptual layer.

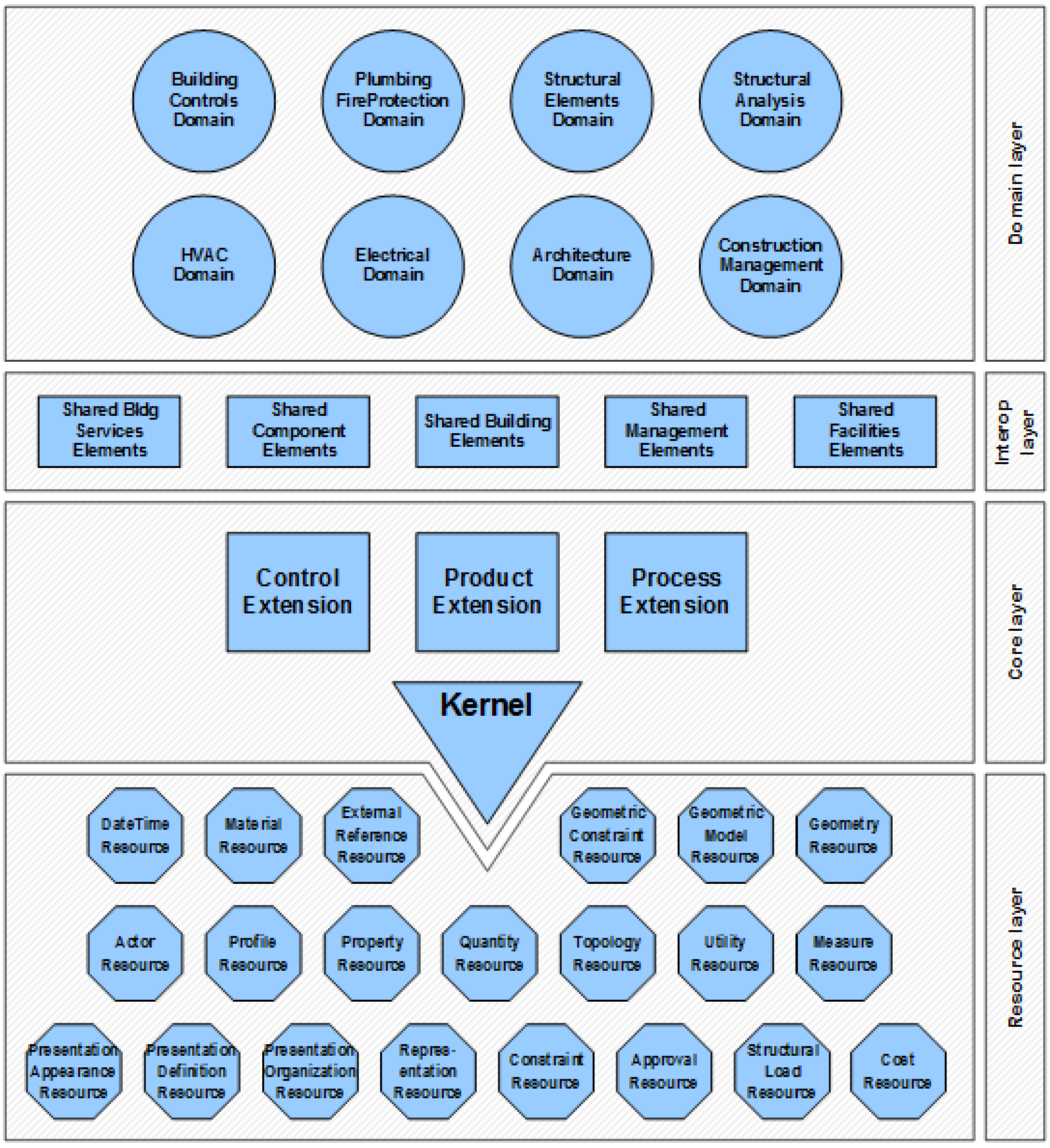


Figure 6: Data schema architecture with conceptual layers [12]

The resource layer is lowest layer in IFC data schema having entities related to geometry, material, placement and representation of objects etc. These entities are not assigned with global unique Identifier (GUID) and can be referred by classes in other layers. All entities defined above resource layer carry a unique ID and can be used independently. The core layer contains the most general entity definitions for kernel and core extension. The interoperability layer contains definitions which are typically utilized for exchange and sharing of construction information across different domains. These definitions are specific to general product, process or resource specialization. The highest layer in IFC data schema architecture is domain layer which includes schemas that are related with products, processes or resources specific to certain discipline. These definitions are typically utilized for intra-domain exchange and sharing of information [12].

IFC structure is hierarchical. *IfcRoot* is most abstract and root class for all the entity definitions so, all entities are sub types of *IfcRoot* and can be used independently apart from entities in resource layer. The three fundamental subtypes of *IfcRoot* are *IfcObjectDefinition*, *IfcPropertyDefinition* and *IfcRelationship*.

- ***IfcObjectDefinition***: "*IfcObjectDefinition* is the generalization of any semantically treated thing or process, either being a type or an occurrence. Object definitions can be named, using the inherited **Name** attribute, which should be a user recognizable label for the object occurrence" [12].
- ***IfcPropertyDefinition***: "*IfcPropertyDefinition* defines the generalization of all characteristics (i.e., a grouping of individual properties), that may be assigned to objects. Currently, subtypes of *IfcPropertyDefinition* include property set occurrences, property set templates, and property templates" [12].
- ***IfcRelationship***: "*IfcRelationship* is the abstract generalization of all objectified relationships in IFC. Objectified relationships are the preferred way to handle relationships among objects. This allows to keep relationship specific properties directly at the relationship and opens the possibility to later handle relationship specific behavior. There are two different types of relationships, 1-to-1 relationships and 1-to-many relationship. used within the subtypes of IfcRelationship" [12].

IFC data format contains graphical and non-graphical data including relations between object, properties, attributes, metadata etc. as shown in Figure 7. The properties are grouped in property sets and relationships describe the connection between individual elements and components and between spatial and non-spatial components. Therefore, the data in IFC includes all of the information included in BIM model. For example, a structural element such as a beam, contain information such as dimension, shape, location, mechanical behavior, material, layers which is important for structural engineer but along with that it also contains other information like date of installation, thermal insulation properties, fire resistance etc. which is irrelevant for structural analysis. Hence, to standardize information delivery method and to support IFC, buildingSMART developed some other open standards like Information Delivery Manual (IDM), Model View Definition (MVD), International Framework for Dictionaries (IFD) and BIM Collaboration Framework (BCF).

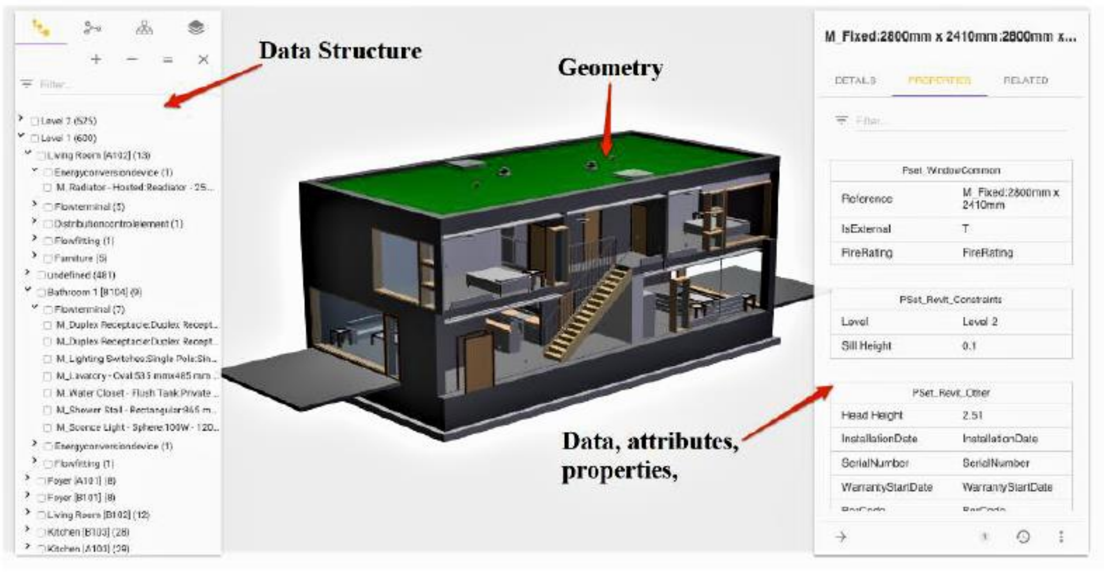


Figure 7: IFC data model including graphical and non-graphical data [13]

2.2.2 IFC interoperability Framework

IFC interoperability for AEC projects are majorly defined in five layers which can be presented as a pyramid shape as shown in **Figure 8**.

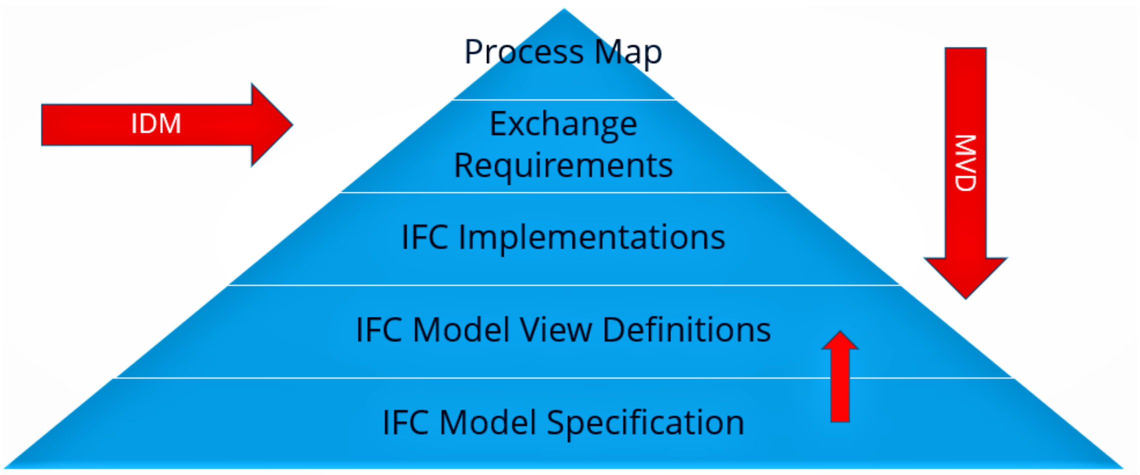


Figure 8: IFC interoperability layers [7]

- **Model View Definition (MVD):** A MVD frameworks relevant information for the efficient transmission of data among stakeholders in construction-related processes and provide guidelines to IFC data related to exchange requirements (ERs). It decreases and screen the needed information to be exchanged for a specific work process [14]. In general definition of MVD depends on the required functionality along with denoted BIM objects and attributes in process and interaction maps.
- **Information Delivery Manual (IDM):** IDM prescribes a database of information that must be included in the exchange model for a particular purpose in BIM through process maps, interaction maps and the linked Exchange Requirement Model (ERM). "Process maps describe the flow of activities within a particular

topic, the actors' roles and information required, created and consumed, while interaction maps define roles and transactions for a specific purpose or functionality. The ERM is the technical solution defining a set of information that needs to be exchanged to support a particular business requirement" [15].

The lower layers (MVD) supply the upper layers (IDM) with guidelines for information. By technological developments and implementation, the upper-level layers respond to the requests. The lower layers must be aware about the practicality of their requests, that it can be easily accepted and implemented by upper layers (Figure 8).

However, despite bulk of research involving technical improvements and testing of data exchange between architectural design and structural engineering domain, seamless transfer of digital models across these domains without data inconsistencies is still a major challenge in practice.

2.2.3 Interoperability challenges between Architectural and Structural domain

Construction sector is one of the most leading industry in world in terms of job creation and revenue production. In 2016 about 18 million jobs were created in this sector within European Union which resulted in contributing 9% of entire GDP [5]. Since structural analysis is an integral and important part of building construction projects and structural model is created from architectural model, it is important to address and solve interoperability issues between structural domain and architectural domain to increase productivity in AEC industry. Major reasons for inadequate interoperability in these two domains are:

Official IFC certification: Most of the software tools related to architectural domain and structural engineering domain use IFC neutral file format for bidirectional data sharing between each other and to authorise IFC import and export capabilities of software tools, buildingSMART adopts IFC 2×3 Coordination View 2.0 and IFC4 Reference View 1.2 as standard template [16]. This validation process includes import and export of standard object-level model like beam, column etc. and import and export of project model which mainly includes standard objects from prior step. Hence, certification process mainly aims at ability to exchange information via IFC instead of data exchange quality [9]. However, when the IFC data format is used for BIM interoperability by these certified software tools in practical projects, which contain much more complex model and information, interoperability issues such as data loss and misrepresentation commonly arises.

Difference in domain knowledge of software tools: Multidisciplinary data interoperability through IFC includes two mapping. First mapping takes place when model is exported from internal schema of software tool to IFC model and second mapping when IFC model is imported to internal schema of another tool as IFC schema is a medium for bidirectional data sharing between software tools related to different domains. Due to difference in internal data schemas of these software tools, all the information does not gets mapped correctly which causes several interoperability issues. This issue can be analysed by an example:

An Architecture designs a building data model called **Set A** using a software called **Software A.** This model is being exported as IFC format called **Set A'.** In this process, mapping will take place from internal data schema of Software A to IFC schema. Structural Engineer needs to import this model for further structural design and structural analysis using a software let's say, **Software B** with data model called **Set B** as illustrated in Figure 9(a)**.** The model gets imported from IFC to Software B. In this process another mapping takes place from IFC schema to internal data schema of Software B.

Preferably, all elements from Set A' should be mapped perfectly in Set B, so that $\forall a \in A' \rightarrow a \in B$ as in Figure 9(b) but this is generally far from reality as several interoperability issues concerning data loss are regularly encountered which shows,

some element $\boldsymbol{a_i}$ from Set A' belong to Set B that is $a_i \in ((A' \cap B) \subset B)$ (Figure 9(C))

some other element $\boldsymbol{a_j}$ from Set A' may not be included in Set B that is $a_j \in ((A' \wedge a_j) \notin B)$

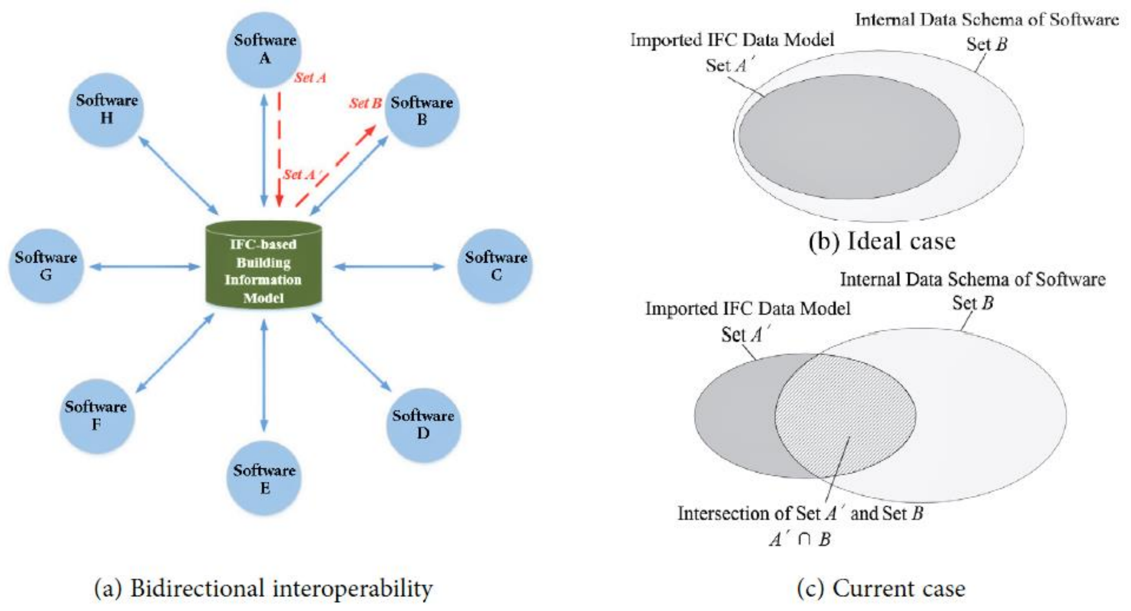


Figure 9: Data interoperability between different domains [9]

which concludes that heterogeneous software tools do not support all the information from different discipline. The data supported by a software tool is typically a subset of all construction project details. Imported model can be well interpreted, when a tool imports a model that relates to the same domain. Since, architectural domain and structural domain are quite vast and data in these domains contain complex and sensitive, graphical and non-graphical information, accurate internal mapping of information is merely possible.

For instance, there are some building elements e.g., a_m (m = 1,2,3,4,5) in Set A for which there is no one to one mapping schema in Set B. In this case, these all elements can be mapped to one particular schema in Set B e.g., b_0 via a specific function called g, that is $\exists g(a_m) = b_0$ (m = 1, 2, 3, 4,5) despite of fact that b_0 can be expressing different information that that being represented by a_m. While some software tools follow many-to-onemapping procedure, some of other software tools do not have any schemas to map this information. For example, *IfcBuildingElementProxy* is an IFC schema entity which is used to represent many IFC object entities [9].

Use of top-down and relational approach by IFC: IFC uses a "top-down" and relational approach, which yields in a relative complex data representation schema and a large data file size. Although, it has the ability to maintain semantic integrity automatically but it is very complex to program and be implemented in software. Whereas, gbXML adopts a "bottom-up" approach, which is flexible, open source, has a less layer of complexity and a relatively straight-forward data schema [7].

Implementation of IDM and MVD: As defined in section 2.2.2, IDM determines the data set that must be contained in exchange model whereas, MVD specify required information for exchanges of building model data among building project experts. IDM and MVD are complementary to each other and change in one of these require an update in another. Proper development and implementation of IDM, s and related MVD, s is necessary for data exchange in specific domain.

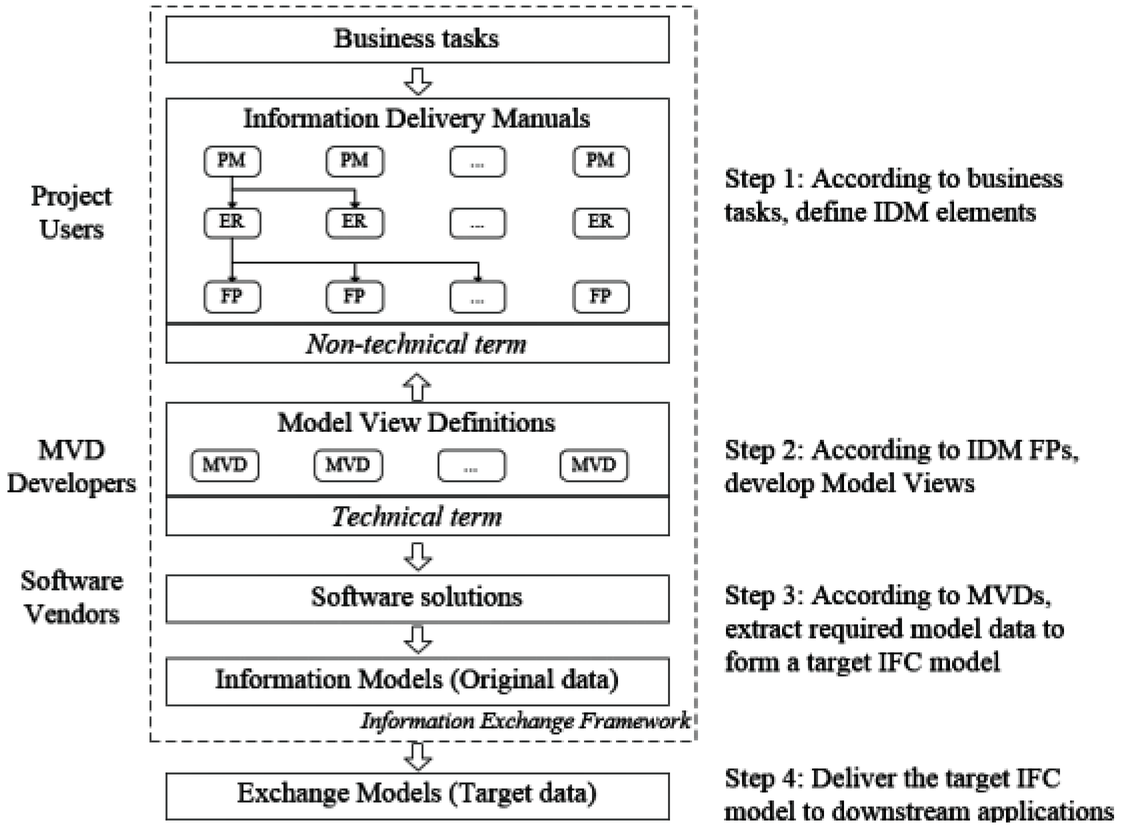

Business tasks
Information Delivery Manuals
PM
PM
...
PM
ER
ER
...
ER
FP
FP
...
FP
Non-technical term
Project Users
Step 1: According to business tasks, define IDM elements
Model View Definitions
MVD
MVD
...
MVD
Technical term
MVD Developers
Step 2: According to IDM FPs, develop Model Views
Software Vendors
Software solutions
Information Models (Original data)
Step 3: According to MVDs, extract required model data to form a target IFC model
Information Exchange Framework
Exchange Models (Target data)
Step 4: Deliver the target IFC model to downstream applications

Figure 10: Information delivery process based on IDM and MVD [17]

According to steps depicted in Figure 10, elements of IDMs are defined conferring to data requirement to represent a specific task or process for a business and MVDs are developed according to elements defined for IDM. There is no clear logical relation between the data sets in the IDM exchange requirements and those in the MVDs, and the mapping that interprets requirements of an IDM into ones of an MVD is exposed to numerous interpretations, without semantic and logical consistency [17].

Inter-domain relationship: For digitalization of data exchange workflow in AEC inter-domain relationship between domain-specific representations is important. Undefined workflows and lack of collaboration between software industry with in architectural and structural domain leads to misinterpretation of data and information of models. There are no standards defined by software tools on how to model the elements, rather it depends on person to person and it may be perceived in different ways by end users. Since correct interpretation of data, efficient data exchange between architectural design and structural analysis model is significant, need for standardization of inter-domain relationship is important. "The mapping of the business process with the inter-domain relations remains in the hand of the software developers and software industry, not the ACE industry or the end users [18].

Information gap in IFC schema: IFC is still work in progress. Although with each new version of IFC the information gap related to structural domain is reducing but still there are still some gaps which needs to be filled with new concepts, definitions, attributes or property sets. Several researches on IFC-based data exchange have been conducted which suggested improvement of IFC schema. Wan and Chen (2004) [19] suggested several improvements in IFC schema after complete assessment of IFC schemas for the requirement of SAP2000 and six other structural analysis tools (ETABS, pro 2003 etc.).The research was conducted using IFC 2×2 edition and changes were made in new editions of IFC versions. For example, under IfcMaterial entity only isotropic materials were allowed but in latest version IFC 4 another attribute has been added for anisotropic materials. These knowledge gaps result in data loss and require extra human effort during data exchange process.

IFC domain extension: "IFC domain extension requires users understanding in: what input information is needed in this domain and how to minimize potential conflicts between the extension and similar definitions that already existed" [7]. Several research projects required IFC domain extension for better data exchange. For example, an experiment was conducted by sack and his colleague to improve precast workflow and was successful by extending IFC support from architectural design to construction project [20]. IFC domain extension could help in BIM interoperability issue as technologies to fully integrate ACE are still in progress. So, methods on proper application of IFC domain extension to achieve full interoperability from architectural domain to structural domain is still questionable [7].

2.2.4 Case Study

Master Thesis by Rafeequl Islam

Reference Reason: Interoperability issues between architectural domain and structural domain using IFC data model

Project Name: Master thesis by Rafeequl Islam

Contributor: Institute of Construction Informatics, Technical University of Dresden, Germany

Used software tools: Autodesk Revit 2018 (Student version), FZK viewer-5.2_Build-992 and SCIA Engineer

Dimension of building model:10.6 m×5.47 m×8.6 m

Motivation and research methodology

Architectural domain and structural domain are collaborative in nature. Efficient data exchange with correct interpretation between these two domains is key factor for increasing productivity and reducing error in AEC industry. Hence, to investigate data interoperability efficiency in these domains, demonstration of data transfer between a BIM modelling tool and structural engineering software was conducted using IFC4 ADD2 data model.

A three-storey building model was designed using widely used tool in architectural domain i.e., Autodesk Revit 2018. The dimension of structural model was 10.6 m in length, 5.47 m in width with elevation of 8.6 m. Standard structural element such as concrete beams and column, steel beams, column and slabs were used to model the structure.

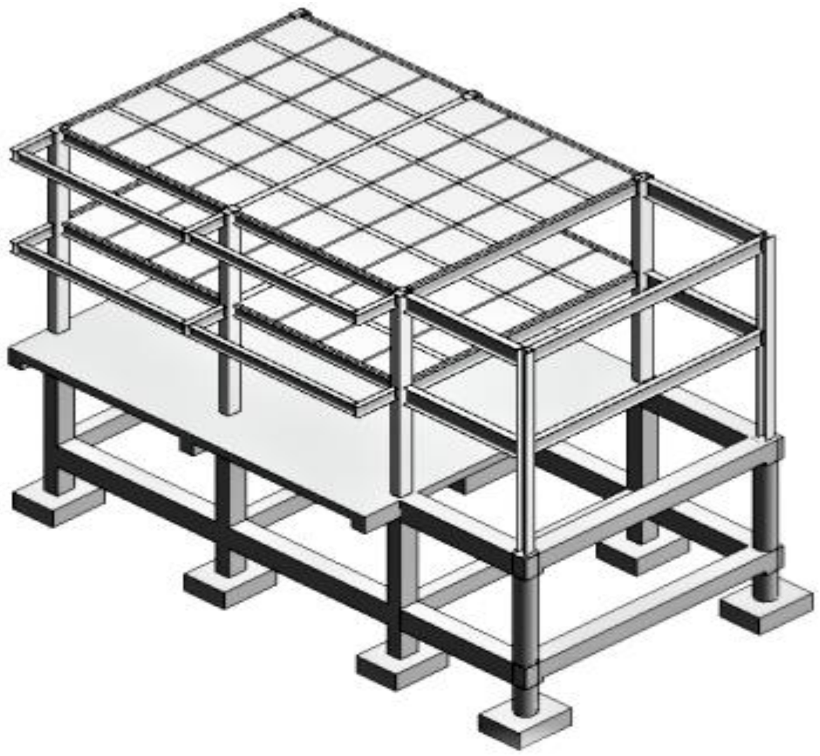


Figure 11: 3d model of structure designed using Revit software [21]

The file was exported as IFC file format from Revit software. Further, the model was checked by a model checker developed by Karlsruhe Institute of Technology, Germany (FZK viewer-5.2_Build-992) and later the IFC file was imported to SCIA Engineer software for structural design and analysis (Figure 12).

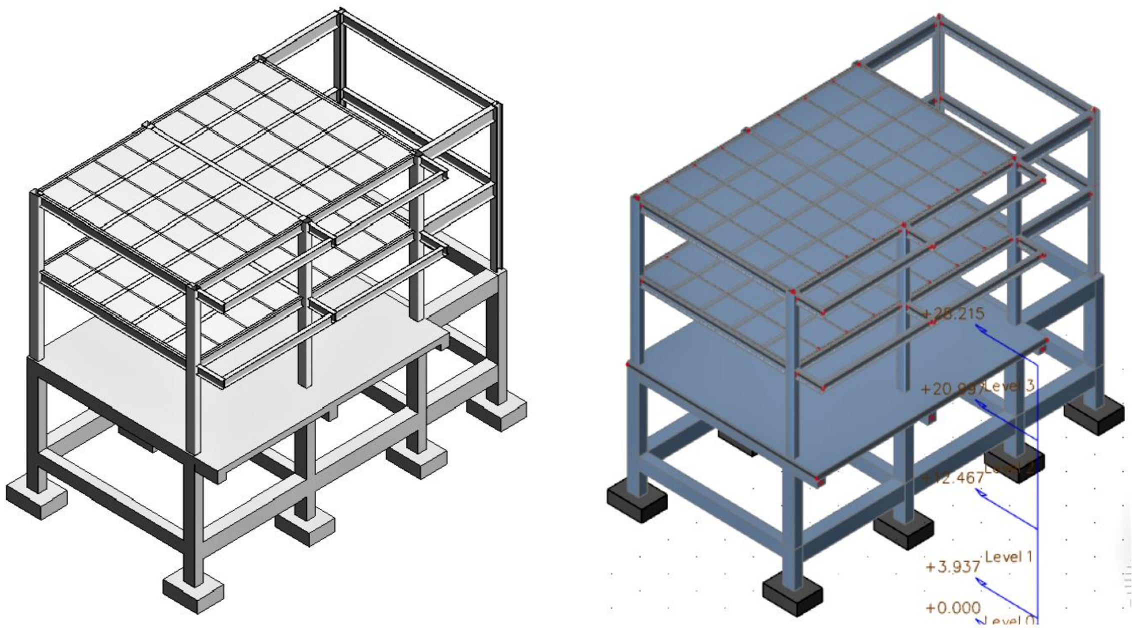


Figure 12: left: Revit model and right: SCIA Engineer model [21]

Important findings of data exchange demonstration:

1. **Data loss in mapping of information from IFC to structural analysis software:** IfcSite is an entity in IFC data schema which includes details about building site such as latitude, longitude etc. Site data is important for structural analysis to determine zones for various loading conditions and wind effect at various elevations. However, there is no mapping schema for such entity in internal data schema of SCIA Engineer software tool. As stated, due to lack of inter-domain relationship and difference in domain knowledge, such problems arise which cause data loss and require manual efforts.
2. **Disjoint nodes:** SCIA Engineer tool perceived centre line of some members as end-to-end distance of member after converting model in analytical model for structural analysis which resulted in disjoining of some connection nodes (Figure 13) causing instability of structure in vertical direction when structural analysis was performed.

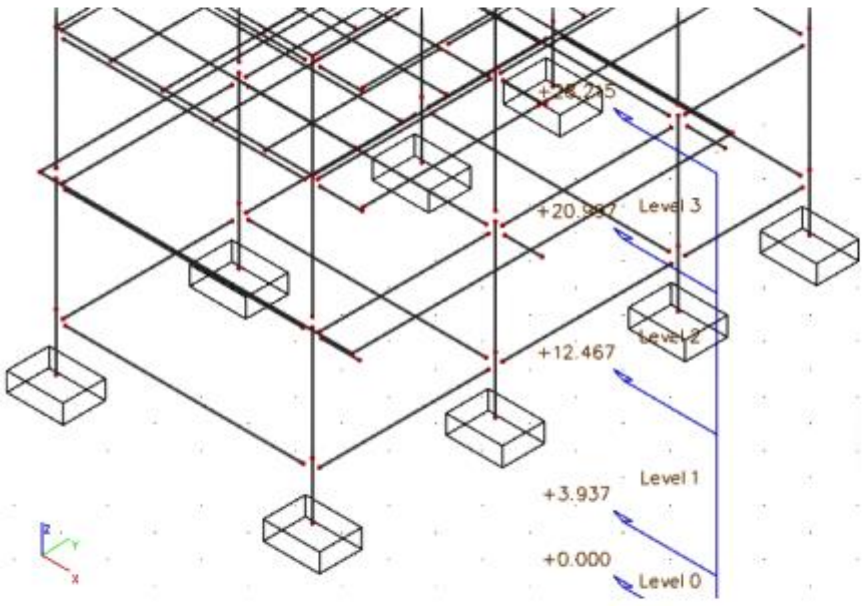
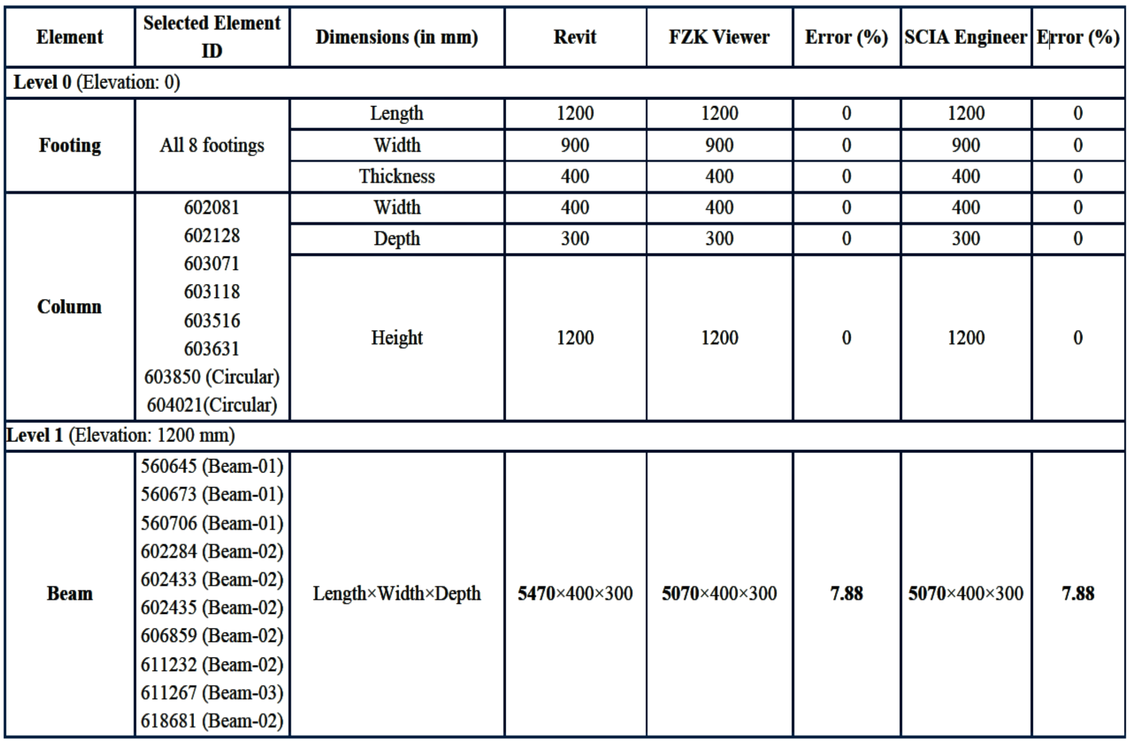


Figure 13: Disjoint nodes in SCIA Engineer model [21]

3. **Geometric misinterpretation:** Deviation in length and height of structural member at first level and second level was noticed in SCIA Engineer tool when compared to model in Revit software tool summarized in Figure 15. The reason for this misinterpretation was incorrect mapping of entity between software tools.



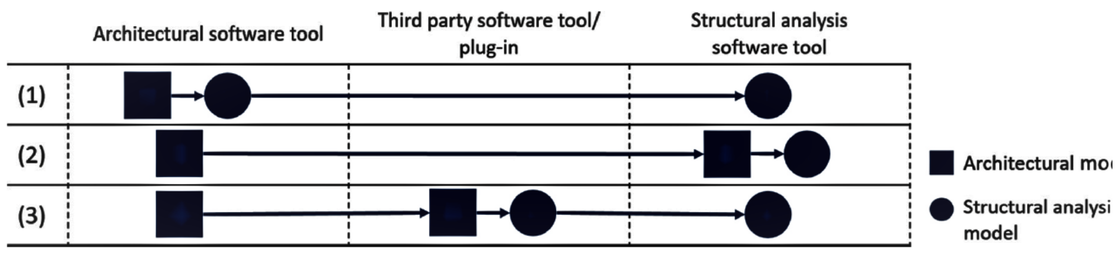
Element	Selected Element ID	Dimensions (in mm)	Revit	FZK Viewer	Error (%)	SCIA Engineer	Error (%)
Level 0 (Elevation: 0)							
Footing	All 8 footings	Length	1200	1200	0	1200	0
		Width	900	900	0	900	0
		Thickness	400	400	0	400	0
Column	602081 602128 603071 603118 603516 603631 603850 (Circular) 604021(Circular)	Width	400	400	0	400	0
		Depth	300	300	0	300	0
		Height	1200	1200	0	1200	0
Level 1 (Elevation: 1200 mm)							
Beam	560645 (Beam-01) 560673 (Beam-01) 560706 (Beam-01) 602284 (Beam-02) 602433 (Beam-02) 602435 (Beam-02) 606859 (Beam-02) 611232 (Beam-02) 611267 (Beam-03) 618681 (Beam-02)	Length×Width×Depth	**5470×400×300**	**5070×400×300**	**7.88**	**5070×400×300**	**7.88**

Figure 14: Geometric inconsistency findings [21]

2.3 Researches and proposed solutions

Several researches and studies have been conducted to improve the data exchange scenario between architectural BIM tools and structural analysis tools. Ramaji and Memari [22] recognised three probable ways, as shown in Figure 15, to generate structural analysis model from architectural model:

- "Structural analysis model is created in native software tool
- Structural analysis model can be created from a native model by a structural analysis tool
- An additional tool (or a plug-in) is used to generate the structural analysis model" [18]



Architectural software tool
Third party software tool/ plug-in
Structural analysis software tool
(1)
(2)
(3)
Architectural mo
Structural analysi model

Figure 15: Possible ways to generate a structure analysis model [18]

Most promising solution to achieve data interoperability is via introducing third party plug-in as software tools related to particular domain mostly focus on requirement of their own domain but to achieve full interoperability an integrated model is required. For example, Liu and Zhang [4] used a third party structural design tool (YJK) as software-specific integration tool to extract BIM structural model from IFC based architectural model for structural analysis (Figure 16).

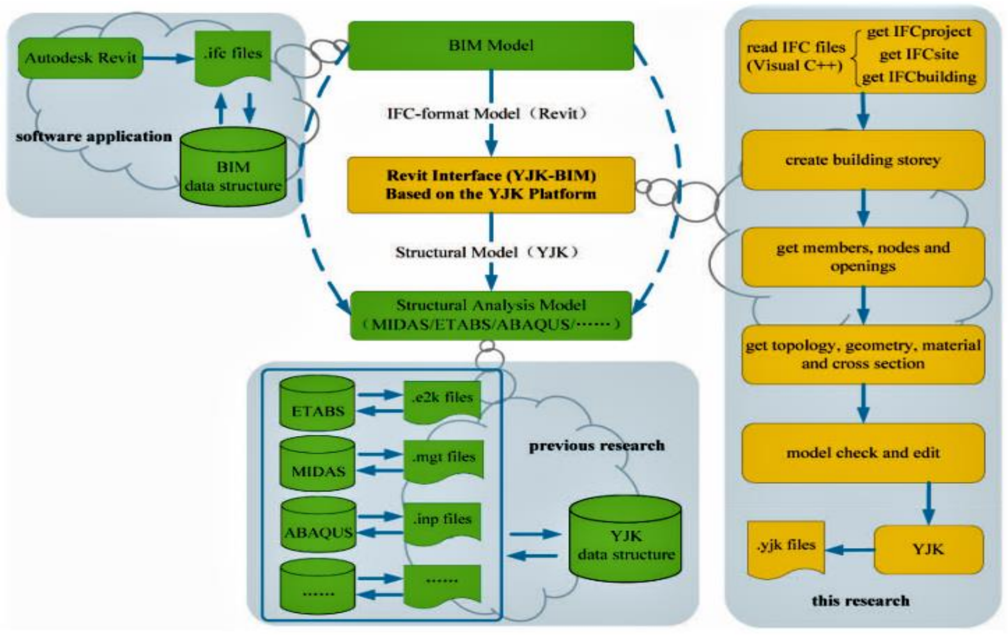


Figure 16: Data transformation workflow by [4]

Further other frameworks were suggested like Wan and Chen [19] followed server based data exchange which involved transferring IFC building data model to S2K file format. In the research IFC schemas for structural analysis were analysed and several recommendations were suggested. Wang and Cui [23]proposed transformation of IFC based structural model before importing the model in structural analysis tool like MIDAS.

2.3.1 Case Study

Reference Reason: Accuracy of data exchange for structural engineering

Project Title: Exchange Requirement-based delivery method of structural design information for collaborative design using IFC

Location: Shanghai, China

Contributor: Department of Civil Engineering, Shanghai Hao Tong University

Objective and methodology

To achieve efficient data sharing for structural engineering research was conducted on Exchange requirement (ER) based information delivery where user can define required objects and attributes based on ERs of building project. Further, objects and attributes could be mapped to assigned IFC data via proposed ER matrix. Mapping of IFC entities through ER matrix will using an IFC-based algorithm and target model could be generated through developed delivery tool.

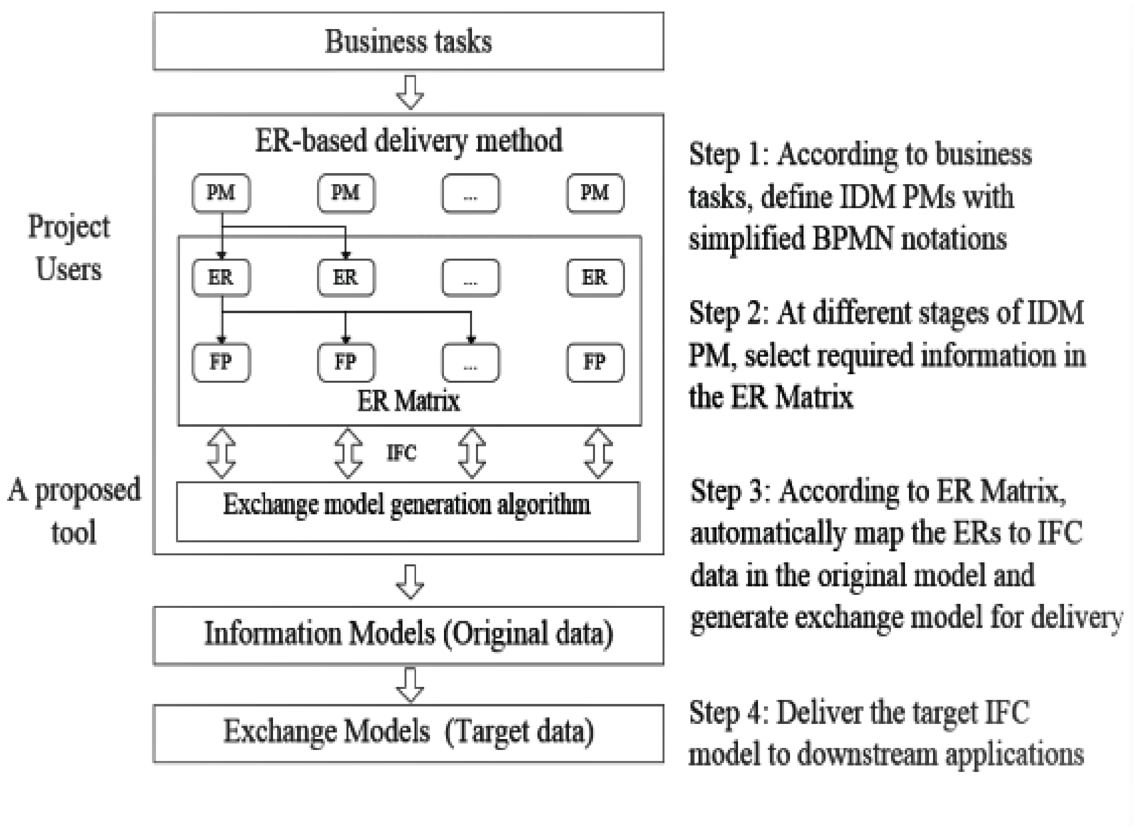

Business tasks
ER-based delivery method
PM
PM
...
PM
Project Users
ER
ER
...
ER
FP
FP
...
FP
ER Matrix
IFC
A proposed tool
Exchange model generation algorithm
Information Models (Original data)
Exchange Models (Target data)
Step 1: According to business tasks, define IDM PMs with simplified BPMN notations
Step 2: At different stages of IDM PM, select required information in the ER Matrix
Step 3: According to ER Matrix, automatically map the ERs to IFC data in the original model and generate exchange model for delivery
Step 4: Deliver the target IFC model to downstream applications

Figure 17: Information delivery process based on user-defined ERs [17]

Step 1: Defining the Process Map (PM) of structural design: "PM describes the relationship between activities, participants and ERs and uses Business Process Modelling Notations (BPMN) to define a process flow for a particular task, including activities, participants and exchange model" [17]. Based on 22 notations of simplified BPMN, PM of structural design was defined and was reviewed by several structural engineer experts.

Step 2: Developing ER Matrix for information delivery: ER vary at every stage of design and differs for every domain. So, exchange requirements were separated into three types: project, exchange and domain-specific elements. Project elements are constant throughout building project and information related to these elements need not to be delivered to other domains i.e., site and location. Exchange elements included elements on which information needs to be delivered with each design stage for example beam, column. Reinforcement details is example of domain-specific elements which belong to individual domain.

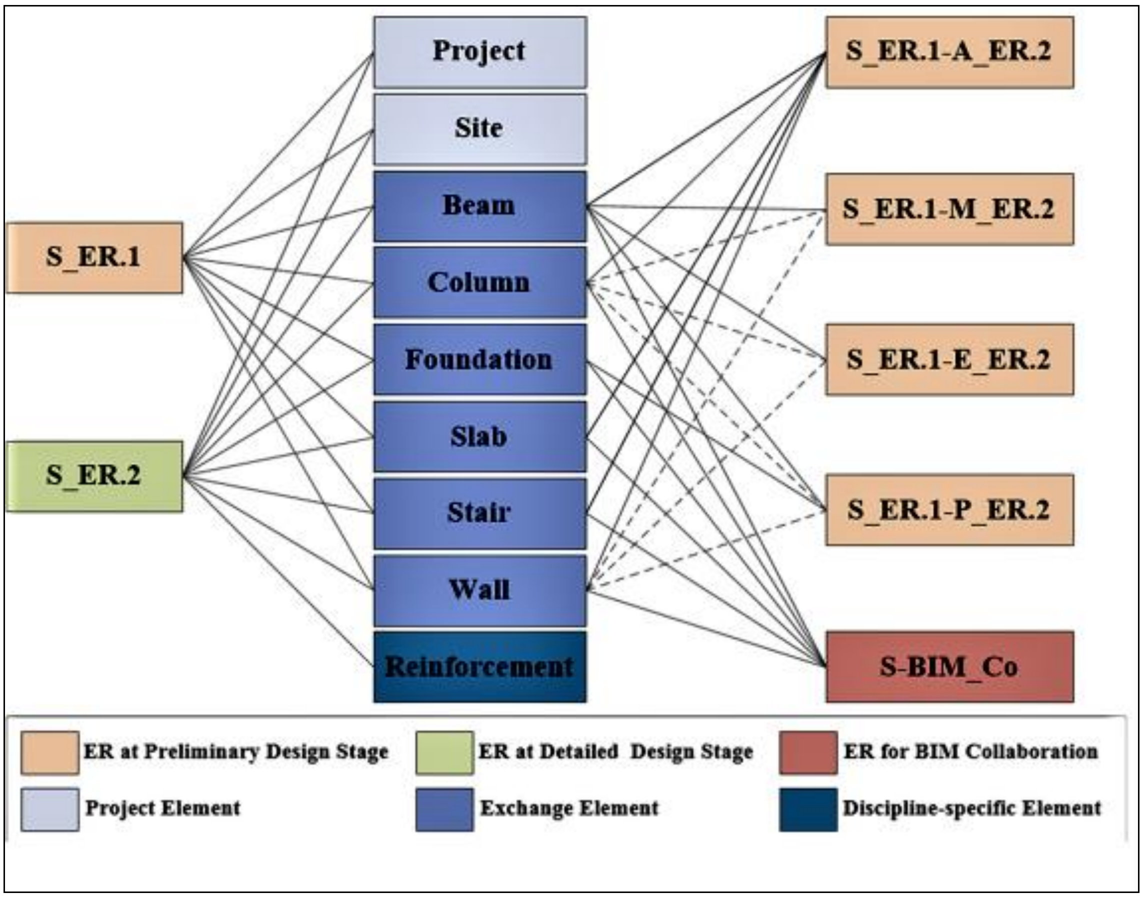
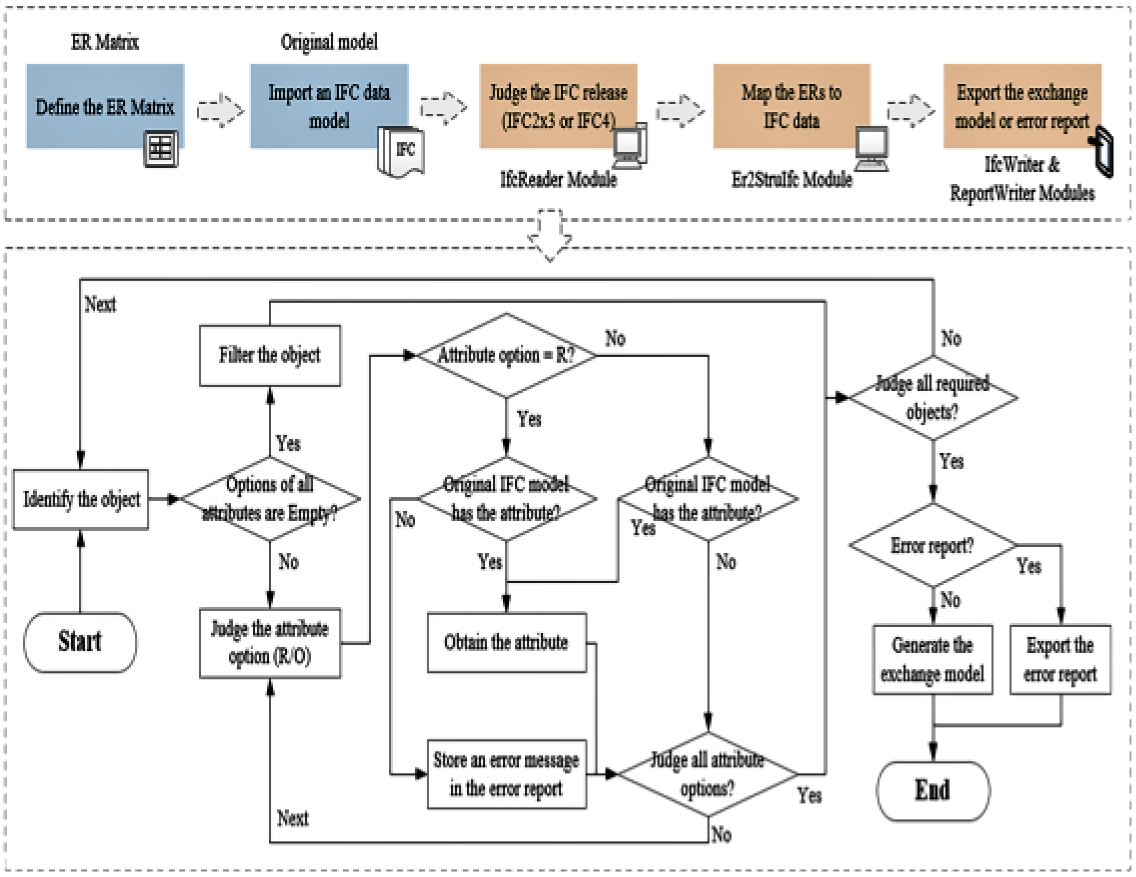


Figure 18: ERs based structural elements [17]

Designed ER matrix can be modified by users, depending on different project requirements and can be defined by using user interface (UI) and XML-based language.

Step 3: User-defined ERs mapping to IFC data of structural model: Using an IFC based algorithm mapping between ERs and IFC data was done for structural mapping. IFC 2×3 version was adopted as data format but IFC4 was also used for mapping through which required information could be automatically exported from IFC model.

Step 4: Development of the delivery tool for structural design information:



ER Matrix
Define the ER Matrix
Original model
Import an IFC data model
IFC
Judge the IFC release (IFC2x3 or IFC4)
IfcReader Module
Map the ERs to IFC data
Er2StruIfc Module
Export the exchange model or error report
IfcWriter & ReportWriter Modules
Next
Filter the object
Attribute option = R?
No
No
Judge all required objects?
Yes
Yes
Identify the object
Options of all attributes are Empty?
Original IFC model has the attribute?
Original IFC model has the attribute?
No
Yes
Yes
No
No
Yes
Error report?
Yes
No
Start
Judge the attribute option (R/O)
Obtain the attribute
Generate the exchange model
Export the error report
Store an error message in the error report
Judge all attribute options?
Yes
End
Next
No

Figure 19: Flow diagram of the exchange model delivery tool [17]

IFC-based Model delivery tool was developed using four modules which are *IfcReader*, *IfcWriter*, **Er2StruIfc** and **ReportWriter**. **Er2StruIfc** modules queries needed information from original IFC model and this information is interpreted by *IfcReader*. If the data cannot be gathered, error messages will be sent. Further, *IfcWriter* integrates the IFC data and generates an IFC model to deliver to software tools for analysis. If ERs is not fulfilled, **ReportWriter** module returns back to experts for adjustment of information.

Demonstration of data exchange using developed framework

Area of building – 47,293 m^2 (9-storey main building and 4-storey reading rooms)

Software used – ArchiCAD, Tekla Structures and MagiCAD

Important findings of data exchange demonstration

- **Model exchange at preliminary stage:** Structural design model was automatically extracted from architectural model at early stage with all the structural elements complied in ER Matrix.
- **Model exchange at second design stage:** Structural design model at later stage was modelled with detailed information and with special conditions and all the models were correctly interpreted and delivered to other discipline automatically.

- **Generation of integrated model:** model from different domains were integrated for BIM collaboration and 92.17% of file size reduction was noticed compared to structural model file size at stage 1.
- **Overall performance:** Performance of ER matrix was found to be completely satisfactory along with model delivery tool, fulfilling all the ERs. Overall, accuracy and efficiency of information was achieved through proposed software.

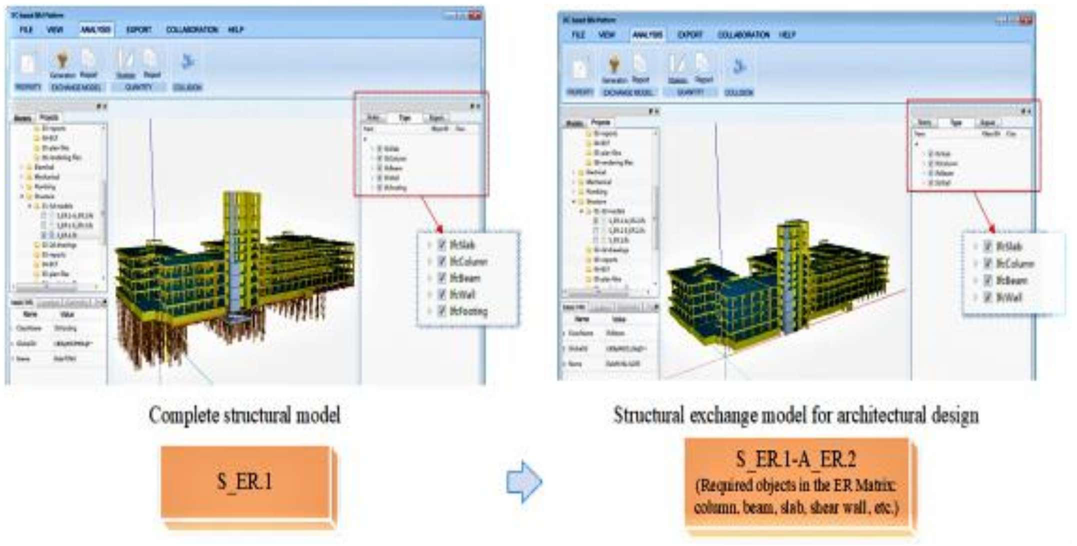

IfcSlab
IfcColumn
IfcBeam
IfcWall
IfcFooting
IfcSlab
IfcColumn
IfcBeam
IfcWall
Complete structural model
Structural exchange model for architectural design
S_ER.1
S_ER.1-A_ER.2
(Required objects in the ER Matrix: column, beam, slab, shear wall, etc.)

Figure 20: Information delivery from structure domain to architecture domain [17]

Several other frameworks were proposed and data transfer demonstrations were conducted by scholars but still complete success is not achieved in terms of data interoperability between architectural and structural domain. Lack of communication and relationship between domain specific software tools along with distortion of data during conversion of information from IFC schema are major concerns for achieving efficient data transfer. As a result, several researches have proposed to use Semantic Web (SW) technology for efficient interoperability through Linked data and Ontologies approach. True meaning of interoperability is data exchange along with true description and semantic web enables the description of information with its inherent semantics [24] and enables the system to interpret true meaning of semantic data for further use.

2.3.2 Linked data and Ontologies approach for semantic interoperability

Semantic Web is a web of linked structured data carrying semantic meaning and enables machines to make logical inferences which are not explicitly specified by individuals whereas, "Semantic interoperability is the ability to exchange information and use it, ensuring that the precise meaning of the information is understood by any other application that was not initially developed for this purpose. Semantic interoperability enables system to process (i.e., use it isolated or combined with their own information) in a meaningful way the information produced by other applications thus making it an important requirement for communication and productivity improvement " [25]. In other words, Semantic Web tools like Ontologies provide the chances to process web data either by humans or machines through rich semantic depiction of concepts in a well-defined domain. Further, data can be linked and shared across the web through a data infrastructure for reuse using Linked data (LD) technologies and can be identified using Uniform Resource Identifier (URIs). Semantic web technologies i.e., Resource Description Framework (RDF) and Protocol and RDF query Language (SPRQL) can be further used to build applications around that data.

Four principles of Linked data:

- Use URIs as name for things.
- Use HTTP URIs so that people look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
- Include links to other URIs so that they can discover more things [26].

Semantic Web offers several possibilities to address major issues in AEC domain like data interoperability, correct interpretation of data and others. Several Ontologies and frameworks have been already developed addressing these issues. Extended information about ontologies, ontology languages and ontology research studies in construction industry along with a case study is described in next part of the chapter.

2.4 Ontology

"Being *qua* being" i.e., the study of attributes that belongs to things because of their very nature is definition of ontology according to Greek philosopher Aristotle [27]. Basically, ontology is philosophical term which refers to study about nature of being and existence. In Computer Science, ontologies are a source to model the structure of a system or domain which includes definition of entities, properties and relationship between them. "Ontology's aim is to substitute the expert and to automate course of translation one profession's unknown expression into another to allow meaning transfer between software" [28]. Therefore, ontologies are developed to enable the modelling of knowledge about specific domain or application on the basis of defined vocabulary which helps in communication between either humans or computers. Vocabulary is loose form of ontology which describes about the concept, properties, relationships and restrictions in defining these concepts in a particular domain. It can be very complex or extremely simple, depending on the ontology and act as a backbone for development of any ontology. According to (Jasper,1999) "An ontology may take variety of forms, but necessarily it will include a vocabulary of terms and some specification of their meaning which includes definitions and an indication of how concepts are interrelated which collectively impose a structure on the domain and contain the possible interpretation of terms" [29]. To express ontologies several Semantic Web ontology languages like RDF, OWL etc. have been developed. Detailed information about these ontology languages have been given in next part of this chapter. Ontology can have different appearances and can be used for different purposes depending on the user. For example, ontologies are encoded in machine readable ontology language for data storage whereas, it can have graphical or formal visualization for knowledge representation.

Ontologies have been classified in three main categories (Figure 21) depending on the knowledge they capture:

- **Upper-level Ontology:** These ontologies are also known as foundational ontologies and are developed to describe the general concepts that can be shared between many domains such as space, time etc. These ontologies are generally used to model application specific ontologies.
- **Domain Ontology:** These ontologies represent specific concepts mostly representing knowledge related to a specific area of interest, for example, construction, management. Domain ontologies can cover any topic and are based on conceptual modelling.
- **Hybrid Ontology:** These ontologies are also called application ontologies and are combination of upper-level ontology and domain ontology. Depending on the specific domain application, it defines definitions that are generally associated with both related ontologies.

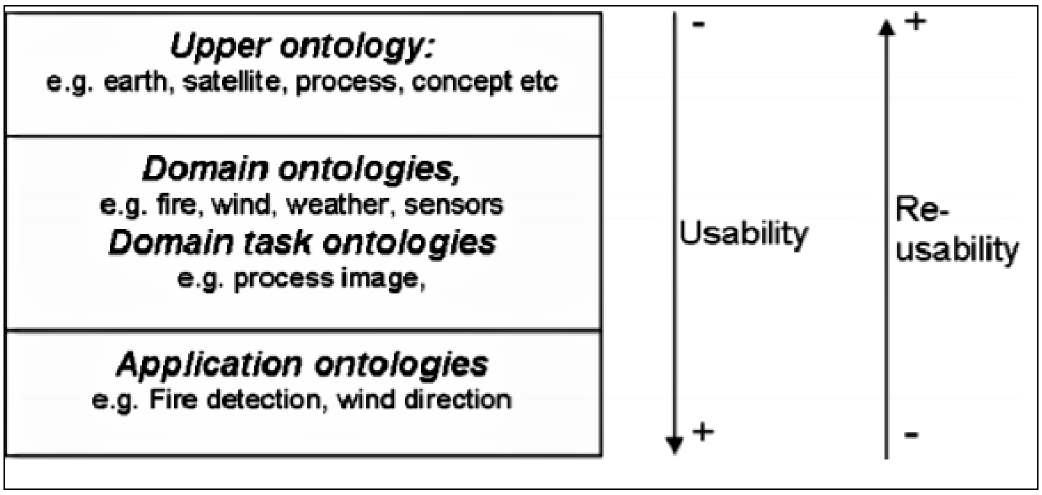


Figure 21: Types of Ontology [30]

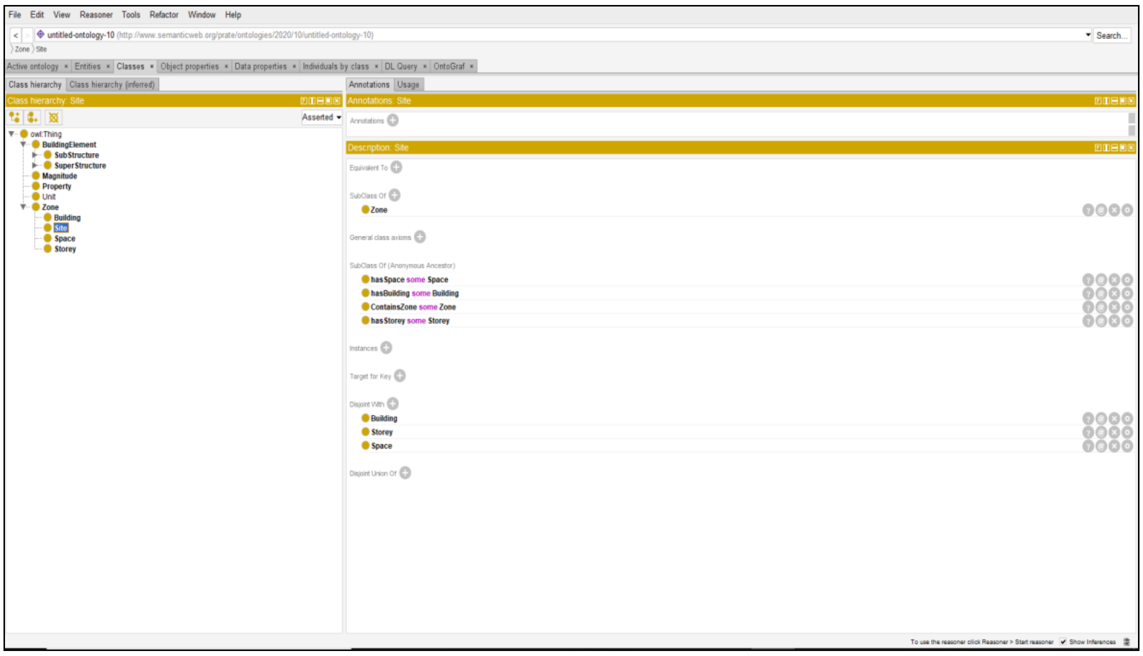
2.4.1 Core components of Ontology

Class or Concepts: Classes are terms that belong to a domain of interest and are most basic elements in ontology modelling. A class is a type of thing like object, place, person, concept, event etc. It can be physical thing like a structural member of a building or abstract like strength of that member.

Class hierarchy: Ontology is formed by a set of taxonomy relationship among classes. A class can include other class. The main class or parent class is called superclass whereas, included class or child class is called subclass. Classes and subclasses form a hierarchical taxonomy where relationship between these classes is clearly defined. Members of a subclass inherit all the characteristics of their superclass which means that every property or concept that a main class holds is also true for subclass. For example, superclass

“StructuralMember” can have a subclass “Beam” or “Column” which will contain all concepts of class “StructuralMember”.

Disjoint classes: In some cases, classes can have properties that are completely different from properties of some other classes which means that being member of certain class can specifically exclude from being member of some different class. This mismatched relationship between two classes is referred to disjointness of class. An individual cannot be member of both disjoint classes. For example, class “StructuralMember” and class “Load” are two classes and both these classes have no similar properties. Hence, mentioned two classes will be disjoint classes and for proper computer inferencing through computer this connection should be clearly defined in an ontology. Figure 22 shows a typical class hierarchy and defined disjoint classes in ontology editor tool Protégé.



```
untitled-ontology-10 (http://www.semanticweb.org/prate/ontologies/2020/10/untitled-ontology-10)
Zone > Site

Class hierarchy: Site
owl:Thing
  BuildingElement
    SubStructure
    SuperStructure
  Magnitude
  Property
  Unit
  Zone
    Building
    Site
    Space
    Storey

Annotations: Site
Annotations

Description: Site
Equivalent To
SubClass Of
  Zone
General class axioms
SubClass Of (Anonymous Ancestor)
  hasSpace some Space
  hasBuilding some Building
  ContainsZone some Zone
  hasStorey some Storey
Instances
Target for Key
Disjoint With
  Building
  Storey
  Space
Disjoint Union Of
```

Figure 22: Classes, class hierarchy and disjoint classes in Protégé

Object Properties: Relationships between classes are established through object properties and these relationships are important for representing knowledge. For instance, class “Building” can be connected to class “StructuralMember” through an object property “hasStructuralMember”. Like classes, object property class can also have subproperties and inherit characteristics from superclass.

Domain and Range: Domain and range restrict the classes that are interconnected via a given property. Suppose, class “Building” and class “StructuralMember” are connected through an object property “hasStructuralMember” so, domain of this statement will be subject (Building), range of this statement will be object (StructuralMember) and property will describe a directional relationship (domain →property →range) between two classes (Figure 24).

Instances or Individuals: Individuals are specific entities that belongs to a specific class or may belong to more than one class and are most basic component of an ontology. For example, “A” is an instance of class “Beam” so, it will contain all the property of the class and means that A is a beam. Some instances may not belong to any classes.

Data property and data types: The data property sets or returns the value of the data attribute of an element/individual. The domain of data property is a class whereas, range is a datatype. A datatype can be a numerical value, a string or literal or other. For instance, individual “A” which is an instance of class “Beam” can have a data property “hasLength” with a numerical value for length. As shown in Figure 23, BE1 is an instance of class BuildingElement and connected to instance P1 through object property hasProperty. Individual BE1 has attribute Length with value of 5600.

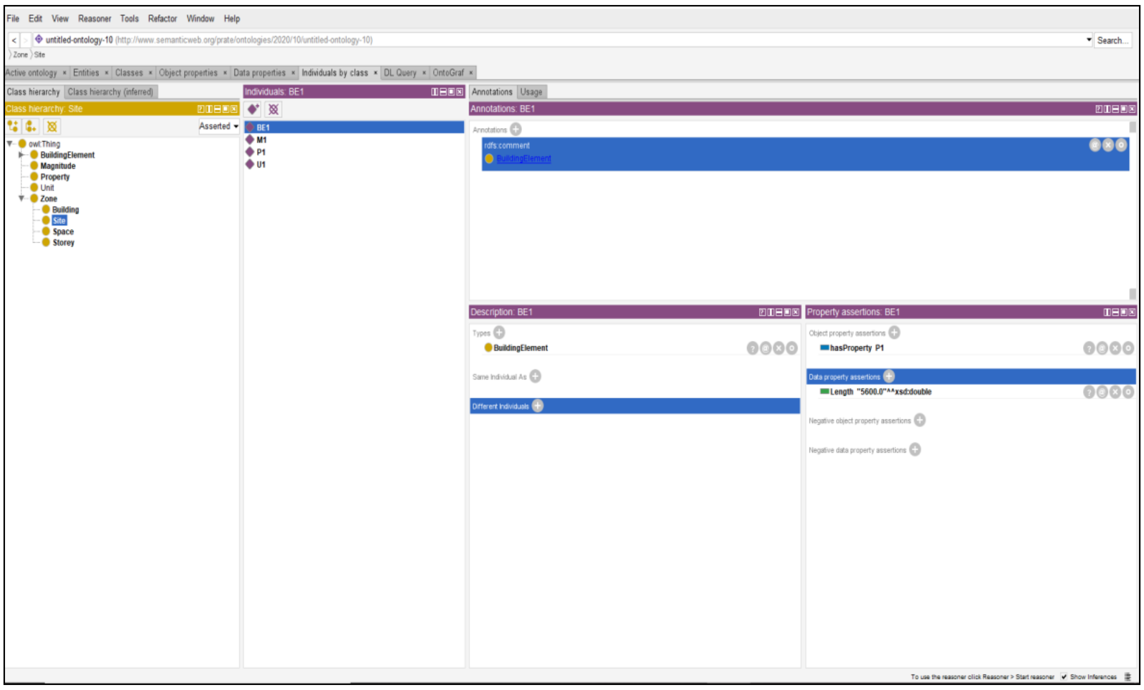

File Edit View Reasoner Tools Refactor Window Help
untitled-ontology-10 (http://www.semanticweb.org/prate/ontologies/2020/10/untitled-ontology-10)
Active ontology Entities Classes Object properties Data properties Individuals by class DL Query OntoGraf
Class hierarchy: Site
owl:Thing
BuildingElement
Magnitude
Property
Unit
Zone
Building
Site
Space
Storey
Individuals: BE1
BE1
M1
P1
U1
Annotations: BE1
rdfs:comment
BuildingElement
Description: BE1
Types
BuildingElement
Same Individual As
Different Individuals
Property assertions: BE1
Object property assertions
hasProperty P1
Data property assertions
Length "5600.0"^^xsd:double
Negative object property assertions
Negative data property assertions

Figure 23: Individuals in Protégé

2.4.2 Ontology Description languages

Ontology description languages have been developed to encode ontologies. These languages serve as knowledge representation artifacts for ontology and have well defined syntax. According to Tim Berner-Lee [31] an ontology description language must possess certain properties:

- It must have a reasonably compact syntax.
- It must have a well-defined semantics so that one can say precisely what is being represented.
- It must have a sufficient expressive power to represent human knowledge.
- It must have efficient, powerful and understandable reasoning mechanism.
- It must be usable to build large knowledge base.

Several ontology languages like F-Logic, Semantic Web Rule language (SWRL), Web Service Modelling Language (WSML), RDF, OWL, SPARQL etc. have been developed. Some most commonly used languages are:

Resource Description Framework (RDF): With the initiative of World Wide Web Consortium (W3C). RDF and RDF Schema (RDFS) language was developed in the context of Semantic Web Interest Group and serves as broadly acknowledged language for encoding ontologies. RDF is a general-purpose language for representing information on web. This framework is compact, readable, understandable and offers high expressiveness. It provides graph-based data model for structuring data as statements or triples about resources. A statement or triple (Figure 24) consists of three components: subject (resource), predicate (property) and object (value). For example, "Mathew has birth place Dresden" is a statement or triple, where Mathew is subject, Dresden is Object and has birth place is predicate which describes relationship between subject and object.

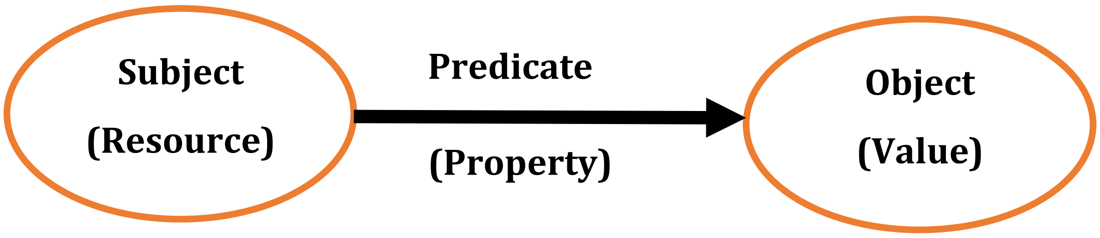


Figure 24: RDF triples

Subject and predicate of a RDF triples must have a URI whereas, object can be either be identified using URI or can be literals (raw text). These triples can have different format depending on serialization syntax such as RDF/XML, Terse RDF Triple Language (Turtle), N-Triples etc. The main features of language include:

- A simple and steady sentence structure.
- URI abbreviation using prefixes.
- Repetition of another object for same resource and property using a comma and repetition of another property for same subject using a semicolon.
- Variables must allow rules to be expressed [32].

Further RDF Schema (RDFS) was developed as an extension of RDF vocabulary as RDF restricts axiomatization to domain and range restrictions. RDF Schema has built in semantic meaning and provides facility to describe classes, subclasses and properties and specifies the domain and range of a property. RDF Schema system is similar to object-oriented programming languages like JAVA. In Java classes are defined on the basis of properties but in RDF schema systems properties are defined on the foundation of classes they interconnect. RDFS enable the users to understand the data through logical inferencing capabilities and uses RDF/XML syntax. Some of the major RDFS elements are: Resource, class, subClassOf, Property, subPropertyOf, domain, range, type, literals etc

(Figure25). Further information about these key elements is described in next part of this chapter.

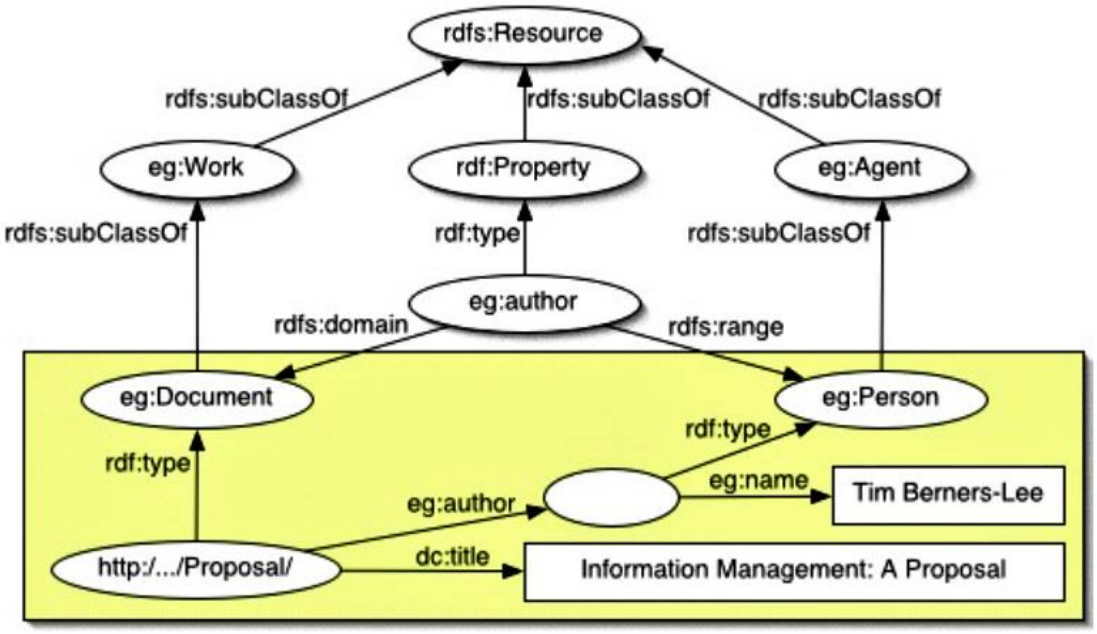
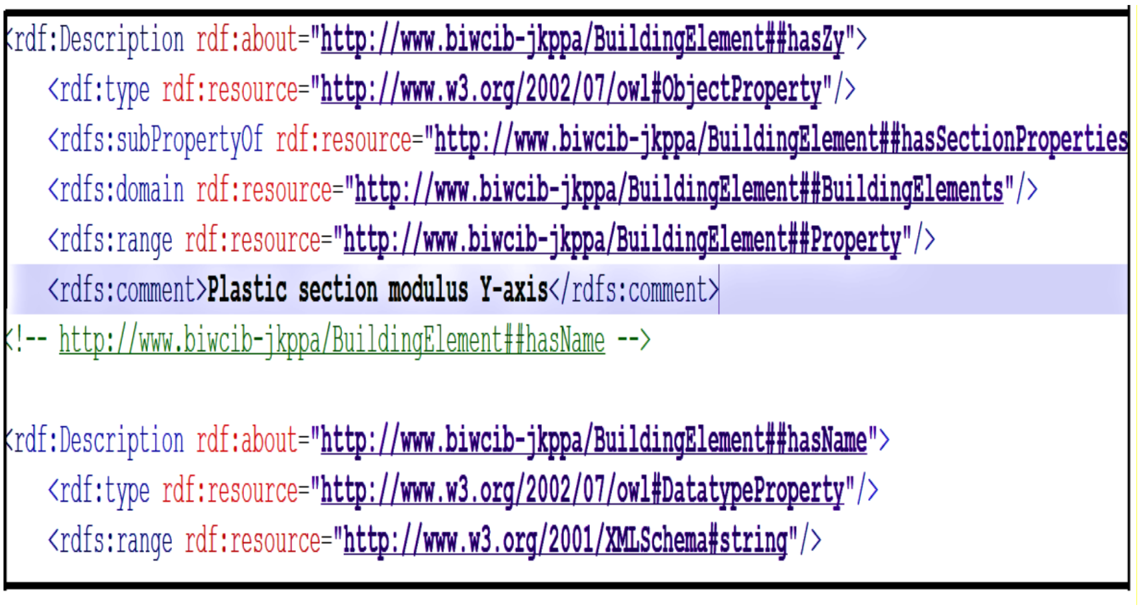


Figure 25: Use of RDF Schema vocabulary for describing classes and properties [33]

Figure 26 is an example of RDF schema in RDF/XML file format. The URI in first row is the description of an object property which is "hasZy". The data also depicts that the mentioned object property is sub property of "hasSectionProperties". Domain of this object property is "BuildingElements" and range is "Property". RDF schema provides meaning to classes and properties and provides a technique through which meaning to classes and properties can be indicated.

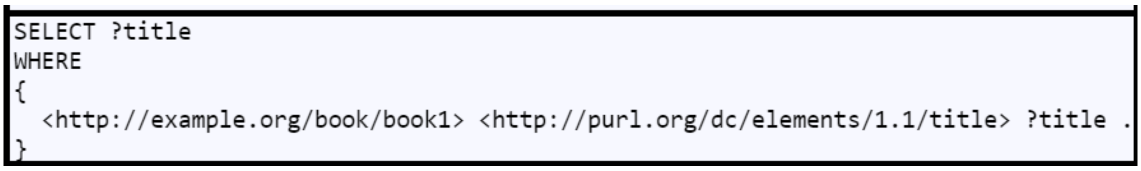


```
<rdf:Description rdf:about="http://www.biwcib-jkppa/BuildingElement##hasZy">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:subPropertyOf rdf:resource="http://www.biwcib-jkppa/BuildingElement##hasSectionProperties
    <rdfs:domain rdf:resource="http://www.biwcib-jkppa/BuildingElement##BuildingElements"/>
    <rdfs:range rdf:resource="http://www.biwcib-jkppa/BuildingElement##Property"/>
    <rdfs:comment>Plastic section modulus Y-axis</rdfs:comment>
<!-- http://www.biwcib-jkppa/BuildingElement##hasName -->

<rdf:Description rdf:about="http://www.biwcib-jkppa/BuildingElement##hasName">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

Figure 26: RDF Schema in RDF/XML file format

Web Ontology Language (OWL): OWL is currently most prominent language today to express ontologies for their use. RDF schema can define a simple class, property but using OWL complex classes can be constructed by means of logical expression. According to W3C “OWL is a computational logic-based Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relation between things and knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit” [34]. OWL is not a direct extension of RDF Schema but usually expressed in RDF/XML syntax and allows wider reasoning and inferencing of information using OWL reasoners. OWL contains strong syntax and a larger vocabulary compared to RDF Schema. Using OWL relation between different classes can be defined and at the same time it allows to specify particular properties (object property and data property including specific datatypes) to those classes. OWL has a universal class called Thing. All classes are subclasses of Thing and all individuals are instance of Thing. Major features of OWL include:

Protocol and RDF Query Language (SPARQL): SPARQL is a query language designed and recommended by W3C RDF Data Access Working Group. SPARQL can be used to retrieve data from RDF graph in form of result sets or RDF graph. It has no inference capacity rather query information from models. Any information stored in RDF format or other model that can be converted into RDF/XML format can be queried using SPARQL. Information is accessed through a basic triple pattern like RDF where some variables replace the elements which are being queried.



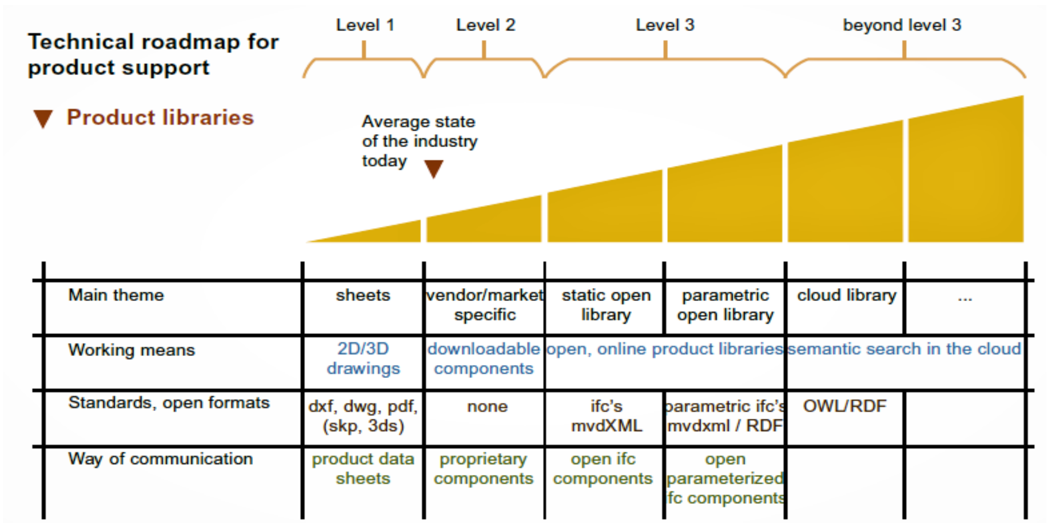
```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Figure 27: SPARQL query [35]

A simple SPARQL query is shown in Figure 27. This query is to find the title of a book from given graph data. Main two elements of this query: “the SELECT clause identifies the variables to appear in the query results, and the WHERE clause provides the basic graph pattern to match against the data graph” [35].

2.4.3 Possible usage and need of Ontology in AEC industry

Civil and construction sectors are one of the major economy sectors in every part of the world and provides a major contribution in overall revenue generation. Increment of productivity in these sectors depend on several technologies, out of which information technology is one of the important factors. The key role of information technology includes exchanging, storing, reproducing, retrieving and integrating of information along with its true semantics from various domains of industry. Implementation of BIM technologies in AEC industry has put a tremendous impact on data and information management but as stated before, there are still major challenges that need to be addressed to achieve full and efficient data management specially during the process of data exchange/transfer. Hence, as an alternative, use of Semantic Web technology like ontologies were investigated by several scholars to address interoperability issues. The use of Semantic Web in AEC industry is also supported by BuildingSMART and therefore, ifcOWL ontology was generated as a domain ontology for construction industry. Figure 28 illustrates the technical roadmap, which depicts need of comprehensive research and studies for complete implementation of Semantic Web technologies in AEC industry.


Technical roadmap for product support
Product libraries
Level 1
Level 2
Level 3
beyond level 3
Average state of the industry today

Main theme	sheets	vendor/market specific	static open library	parametric open library	cloud library	...
Working means	2D/3D drawings	downloadable components	open, online product libraries		semantic search in the cloud	
Standards, open formats	dxf, dwg, pdf, (skp, 3ds)	none	ifc's mvdXML	parametric ifc's mvdxml / RDF	OWL/RDF	
Way of communication	product data sheets	proprietary components	open ifc components	open parameterized ifc components		

Figure 28: The technical roadmap for building support by BuildingSMART [36]

Ontologies allow representation of information in structured graphs and integrate building information from different domains. This information contains true semantic meanings and can be further queried and verified using semantic technologies, which are crucial in AEC domains. Use of ontologies in AEC industry, offer solutions for several issues, which are:

Interoperability: Ontologies uses single data model (RDF) for data representation along with addition of description logics. Further, different information can be linked together in web like fashion. Based on these characteristics, Abdul-Ghafour and others [37] suggested an approach for improving interoperability of CAD information using a Common Feature Design ontology (CDFO), through which non-geometric data from different domains were combined, exchanged and reused. Karlapudi and others [38] developed Digital construction – Building material ontology (DICBM) for the representation of building material data and to enhance interoperability during collaborative workflow. Further, ontologies can be helpful in finding diverse identical partial elements when partial models (architectural model) are exchanged across domains using linksets. Linksets represent the relationship between partial models (interlinked RDF graph) and human efforts are required to manage links between different ontologies. Despite of needed human intervention, ontologies have opened possibilities to tackle some interoperability issue practically [36]. Further, if these RDF graphs possess a formal structure, then a mapping schema can be devised between specific pairs of schemas, which can be used for automatically inference of data using an inference engine.

Theoretically, many other promising frameworks have been proposed that leads to efficient interoperability across different domains of AEC industry but researches are still being conducted to validate these hypotheses.

Linking different domains: Interoperability is ability to load same information in multiple applications where, linking different domains means ability to integrate different content from multiple applications (GIS data, geometrical data) [36]. According to Berners-Lee "Ontology's unifying logical language will enable data from different domains to be progressively linked into a universal web". Use of Linked Data technology for combining diverse building data from heterogeneous domains have shown substantial possibilities for linking different domains knowledge. For instance, Karan and Irizarry [39] proposed a method for heterogeneous data integration between BIM tools and Geographic Information System (GIS) tools using ontology and other semantic web technology (SPARQL)

Logical inference and proofs: Ontologies are used to represent semantic meaning of knowledge using OWL, which is grounded in Description logic. Formal logic basis of Semantic Web languages allows inference of information using inference engines. Further, this basis also allows to automatically generate the proofs for what is inferred in a reasoning process [36] which can be used as a proof by semantic web applications for their results.

Use of Ontologies and other Semantic Web technologies have higher potential to become a next trend in AEC industry and offer high possibilities to solve major data related issues particularly interoperability.

2.4.4 Case Study

Reference Reason: Use of Ontology for integration and interoperability of information

Project Title: BIM and GIS Integration and Interoperability Based on Semantic Web Technology

Researcher: Ebrahim P. Karan, Javier Irizarry, John Haymaker

Contributor: American Society of Civil Engineers (ASCE)

Objective and methodology

Geographic Information System (GIS) provides important data related to construction site and are vital part of preconstruction planning while, BIM represents significant building components and plays important part for design and construction process. Data integration and efficient data exchange between these two heterogeneous domains are crucial in AEC industry. Data formats of these domains are quite different as GIS data are usually two dimensional while, BIM data represent mainly three-dimensional objects. Users need to have a complete knowledge of both domains in order to interpreted correct information as data exchange tools lack semantic meaning which causes misinterpretation and errors.

To overcome these problems, a framework was proposed as in Figure 29 for Semantic interoperability across BIM and GIS tools using ontology so that, meaningful exchange of information can be achieved.

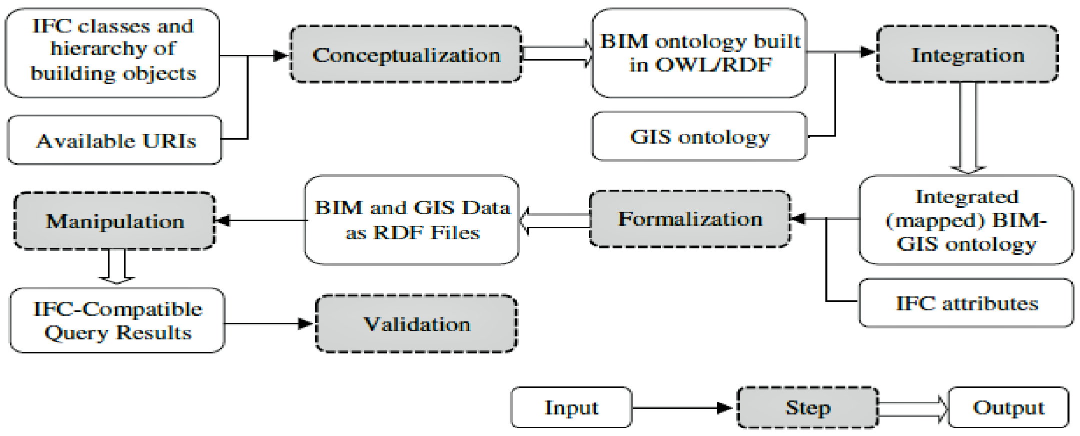
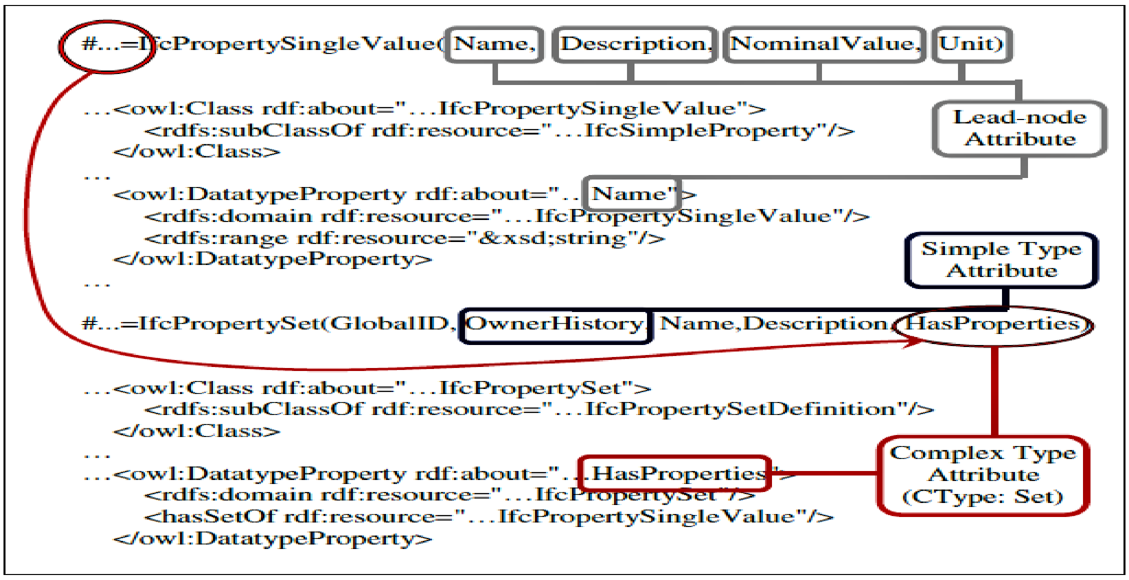

IFC classes and hierarchy of building objects
Available URIs
Conceptualization
BIM ontology built in OWL/RDF
GIS ontology
Integration
Integrated (mapped) BIM-GIS ontology
IFC attributes
Formalization
BIM and GIS Data as RDF Files
Manipulation
IFC-Compatible Query Results
Validation
Input
Step
Output

Figure 29: Research methodology process [39]

In first step, the IFC schemas for BIM were analysed for ontology construction. The ontology was constructed in terms of graph, where IFC entities, their properties and relationships between entities were established using OWL descriptive language. In this ontology (BIM ontology) natural IFC entities were defined at top of hierarchy as primitive classes and defined concepts were defined as subclasses of primitive classes.

In second step, ontology mapping was used to determine semantically corresponding entities among BIM and GIS ontology. GIS ontology was not developed but rather, already existed ontologies were used for mapping process. For mapping process Graph Matching for Ontologies (GMO) approach was adopted which uses RDF bipartite graph model, which gave an BIM-GIS mapped ontology. In next step, IFC and GML files were converted into RDF graphs. and then RDF query language was used to extract data from RDF/OWL file. Figure 30 shows the IFC entities and their matching RDF/OWL classes translation.



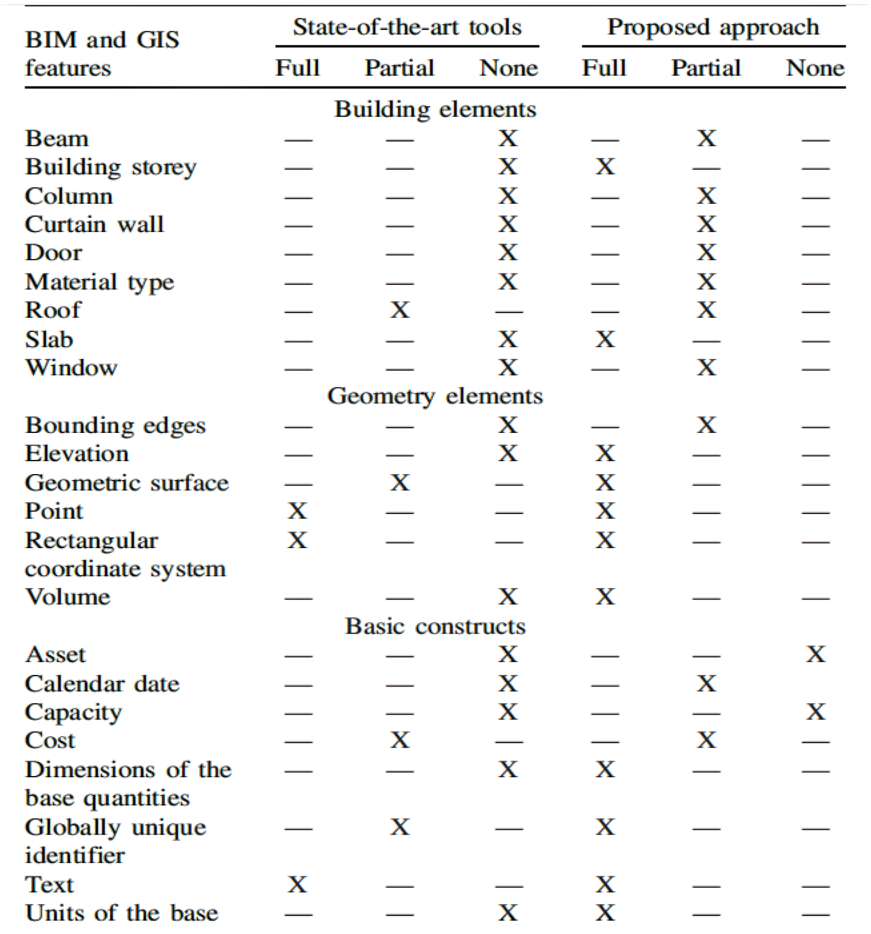
```
#...=IfcPropertySingleValue(Name, Description, NominalValue, Unit)
...<owl:Class rdf:about="...IfcPropertySingleValue">
      <rdfs:subClassOf rdf:resource="...IfcSimpleProperty"/>
   </owl:Class>
...
   <owl:DatatypeProperty rdf:about="...Name">
      <rdfs:domain rdf:resource="...IfcPropertySingleValue"/>
      <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>
...
#...=IfcPropertySet(GlobalID, OwnerHistory, Name,Description, HasProperties)
...<owl:Class rdf:about="...IfcPropertySet">
      <rdfs:subClassOf rdf:resource="...IfcPropertySetDefinition"/>
   </owl:Class>
...
...<owl:DatatypeProperty rdf:about="...HasProperties">
      <rdfs:domain rdf:resource="...IfcPropertySet"/>
      <hasSetOf rdf:resource="...IfcPropertySingleValue"/>
   </owl:DatatypeProperty>
```

Figure 30: Transformation of IFC entities into RDF/OWL classes [39]

Later, model of The School of Nursing at the University of West Georgia was used for validation of aforementioned framework. The model consists a three-storey building with area of 65,000 ft^2 and certain appropriate parameters were considered to check the efficiency of demonstration like description of curtain walls. Interoperability using standard method (use of IFC format) was also conducted for comparison.

Important findings of research study

1) Using standard method of interoperability only 24% of the features were semantically shared between BIM and GIS tools while, 42% of data were exchanged with true description using proposed framework.
2) Full recall rate (data exchange from BIM to GIS and back again) through proposed approach was 40% whereas, only 10% via standard method as correct interpretation of exchanged data is inefficient using BIM and GIS tools due to different data formats (Figure 31).
3) Due to lack of similarity of concepts in final system, mapped ontology was not able to fully capture the semantics of concepts.
4) It can be concluded by results that; ontologies can be used to integrate information across the different domains and have potential to extend interoperability efficiency.



BIM and GIS features	State-of-the-art tools			Proposed approach		
	Full	Partial	None	Full	Partial	None
			Building elements			
Beam	—	—	X	—	X	—
Building storey	—	—	X	X	—	—
Column	—	—	X	—	X	—
Curtain wall	—	—	X	—	X	—
Door	—	—	X	—	X	—
Material type	—	—	X	—	X	—
Roof	—	X	—	—	X	—
Slab	—	—	X	X	—	—
Window	—	—	X	—	X	—
			Geometry elements			
Bounding edges	—	—	X	—	X	—
Elevation	—	—	X	X	—	—
Geometric surface	—	X	—	X	—	—
Point	X	—	—	X	—	—
Rectangular coordinate system	X	—	—	X	—	—
Volume	—	—	X	X	—	—
			Basic constructs			
Asset	—	—	X	—	—	X
Calendar date	—	—	X	—	X	
Capacity	—	—	X	—	—	X
Cost	—	X	—	—	X	—
Dimensions of the base quantities	—	—	X	X	—	—
Globally unique identifier	—	X	—	X	—	—
Text	X	—	—	X	—	—
Units of the base	—	—	X	X	—	—

Figure 31: Recall results for different features of case study [39]

3 Methodology for development and validation of Ontology

This research study has mainly two objectives i.e., development of domain ontology for structure analysis domain and to check applicability and usability of ontology for data capture and data exchange from architectural model to structural analysis model. To develop ontology, first IFC schemas for structural analysis domain was analysed for information requirements and then four ontologies were developed using Protégé. Each ontology represents certain specifications related to structural analysis domain like building elements, loads etc. In next step, consistency of ontology was checked using reasoner. Further, to check the applicability of developed ontologies, a three-storey architectural model was exported to IFC file format and imported to RSTAB structural analysis software. Structural analysis was performed and again model was exported from RSTAB to IFC file format as .ifc file and then exported file was converted to RDF/XML file format. In last step using SPARQL query language for RDF, information was queried from the file and queried information was further constructed in developed ontology to check applicability of ontologies. Figure 32 represents methodology work flow.

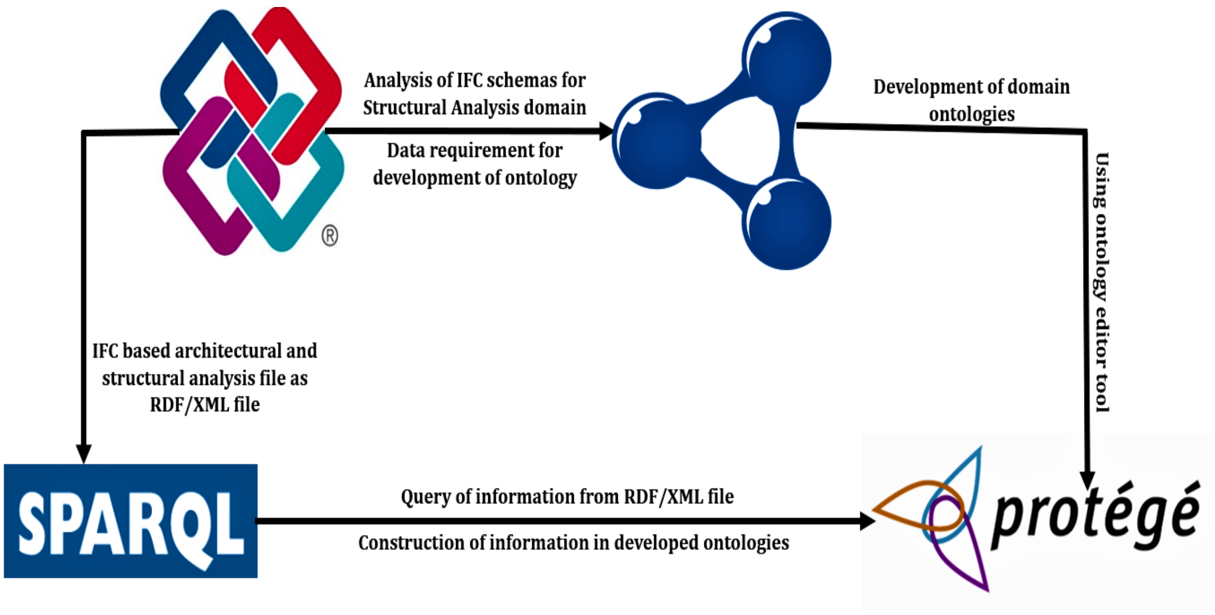

Analysis of IFC schemas for Structural Analysis domain
Data requirement for development of ontology
Development of domain ontologies
Using ontology editor tool
IFC based architectural and structural analysis file as RDF/XML file
SPARQL
Query of information from RDF/XML file
Construction of information in developed ontologies
protégé

Figure 32: Research methodology process

3.1 Assessment of IFC Structural Analysis domain

Ontology is modelled on the basis of well-defined concepts and vocabulary of domain in interest. Since, IFC data format is most widely used for data exchange across architectural domain and structural analysis domain so, IFC4 – Addendum 2 schemas for structural analysis was completely analysed to find the data requirement for development of domain ontologies. Data requirement for development of ontologies can be classified into four major categories namely: Building site and Building element data, material and property information, load information and structural analysis methods and results. The four different categories and corresponding information are listed in Table 2.

Table 2: Data requirement for ontology construction

Structural Element and Site	Object property and representation	Load assignment	Structural Analysis
Building site and storey data Structural Elements data Topology data	Member Properties data Member placement data Member representation	Load details data Load Values data Load Placement data	Load combination Load factor Load calculation Structural Analysis method Result and failure check

3.1.1 Zone and Building Site

IFC schema *IfcZone* is a subtype of *IfcGroup* (Figure 33). Zone is set of spaces or other zones and cannot have placement and geometric representation. *IfcZone* inherits attribute **Name** that generally provides short name or number of zones and attribute **LongName** for full name. Objectified relationship *IfcRelAssignToGroup* is used to group *IfcSpace* into *IfcZone* as shown through Figure 33.

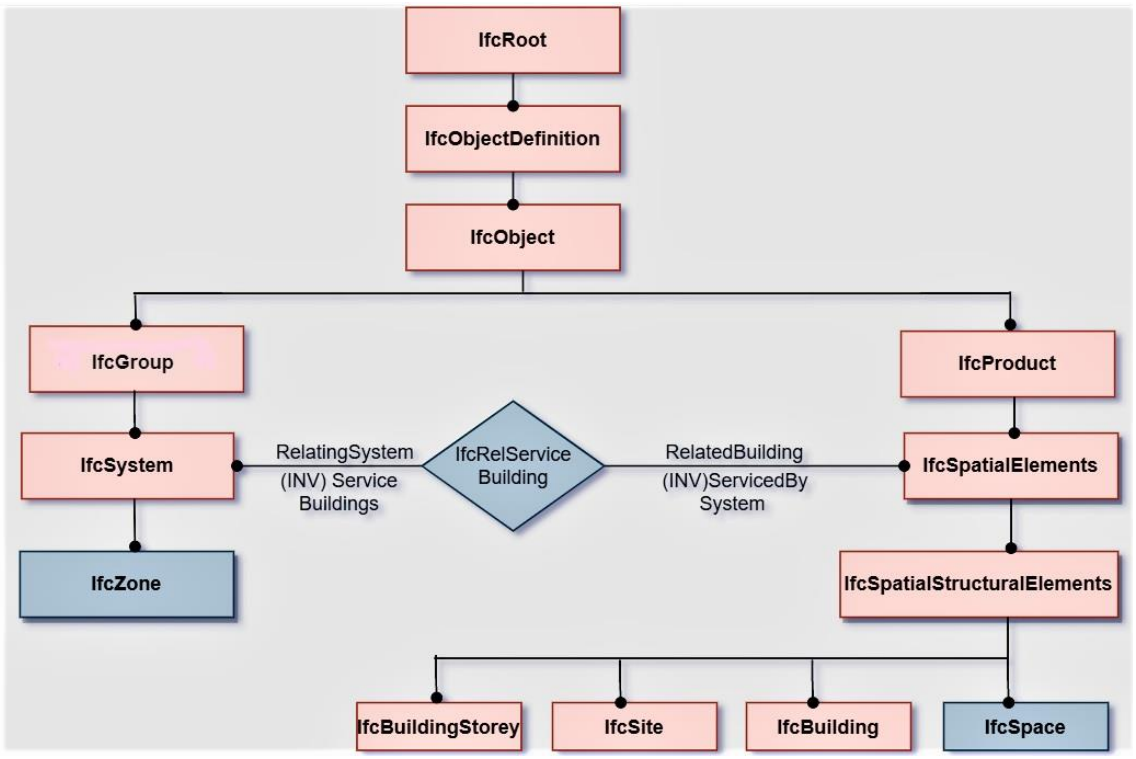


Figure 33: Relationship between Zone and Space [12]

IFC schema for space is *IfcSpace* which is subtype of *IfcSpatialStructureElement*. Spatial structure elements are used to define a spatial structure and are key elements for building project. *IfcSpatialStructureElement* consist four subsets namely: *IfcSpace* (space), *IfcBuilding* (building), *IfcSite* (site), *IfcBuildingStorey* (storey). Relationship *IfcRelAggregate* is used to establish relationship between different spatial structure elements (Figure 34).

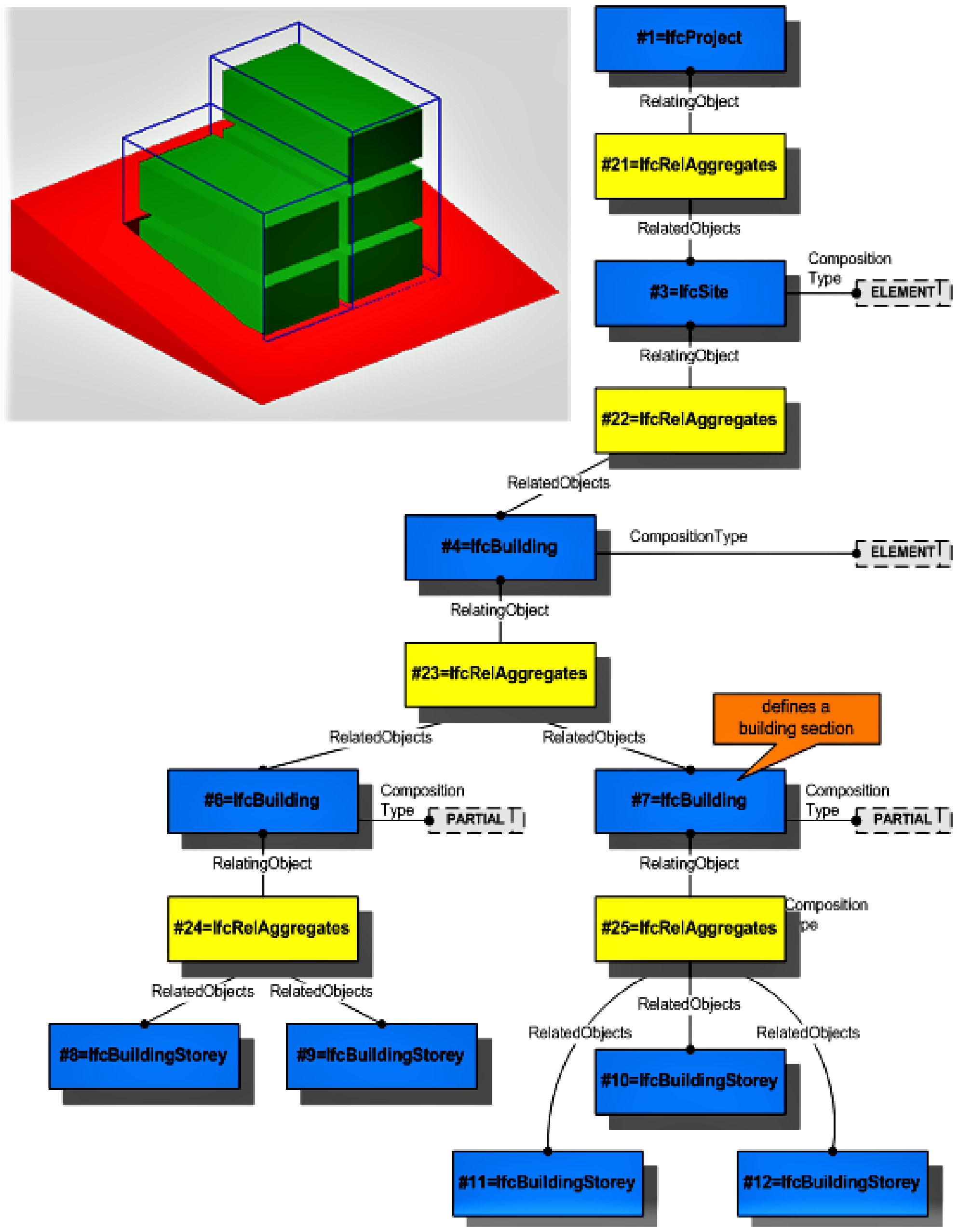


Figure 34: Spatial structure element composition [12]

Site describe about the area, place or land of any type of construction or building project. *IfcSite* includes latitude, longitude and elevation with respect to real world i.e., absolute placement details or coordinates related to other spatial structure elements i.e., local placement. Site details also include address and land title number to provide identification. *IfcBuilding* represents building which is a basically a structure. There can be a group of buildings in a project or a single building. *IfcBuilding* contain attributes to provide information about elevation of building, elevation of terrain and building address. Storey is represented by *IfcBuildingStorey* which is generally linked with building. Building storey is used to build spatial structure of a building. *IfcBuildingStorey* contains attribute **Elevation** through which elevation of storey is known. Elevation of each storey is given as local elevation relative to the height of associated building. Space is an area or volume bounded and is represented as *IfcSpace* in IFC data. Space is associated to building storey and are included in a storey.

3.1.2 Structural elements

All the major building structural elements such as beams, columns, load bearing walls, slabs etc. are represented under IFC entity *IfcStructuralMember* which is superclass for *IfcStructuralCurveMember* and *IfcStructuralSurfaceMember* (Figure 35). Instance of *IfcStructuralCurveMember* describe edge members i.e., beams, columns etc. which can be either straight or curved. Face members like slabs, walls etc. represented by *IfcStructuralSurfaceMember* entity.

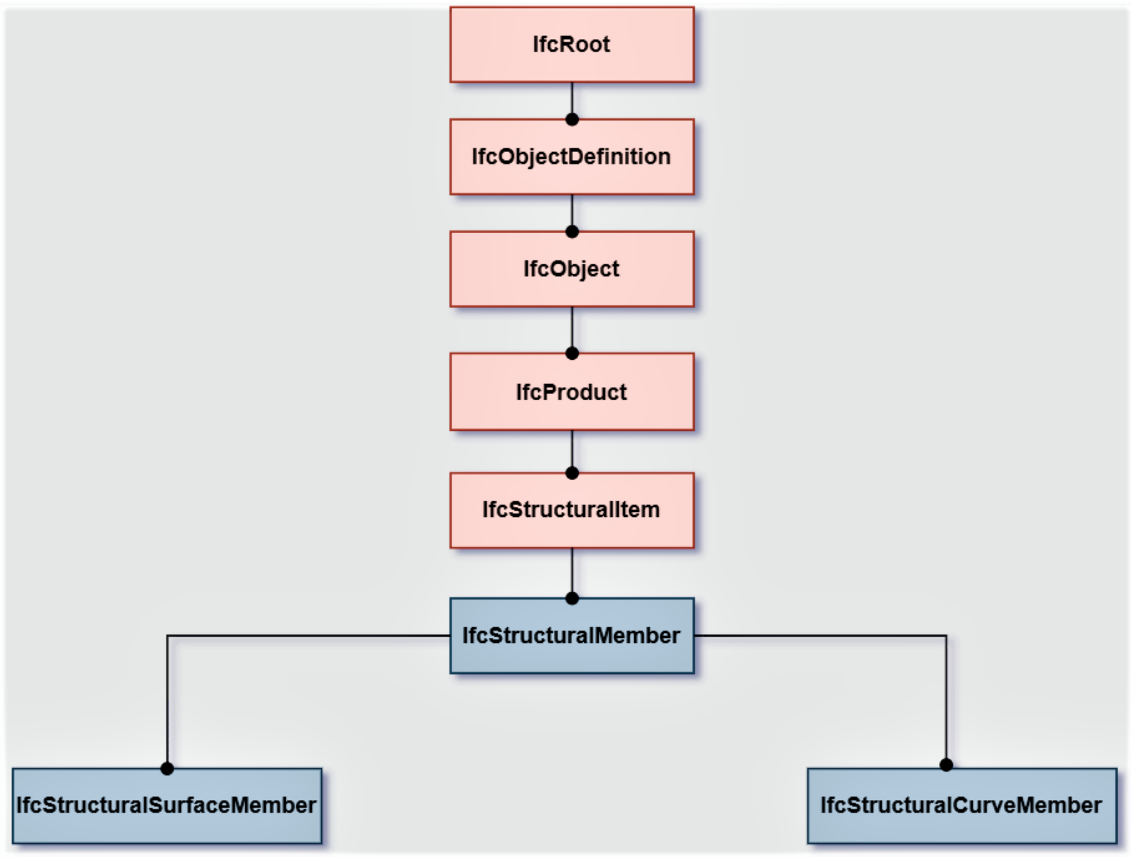


Figure 35: Reference IFC entities for structural members [12]

Axis attribute of *IfcStructuralCurveMember* gives direction ratio in two- or three-dimensional space. *IfcStructuralSurfaceMember* carries attribute **Thickness** to provide thickness of structural surface member.

Each structural member carries a global unique ID along with name and description. **ObjectPlacement** and **Representation** attributes are inherited from superclass *IfcProduct.* Entities related to placement and representation belongs to IFC resource layer and entities from these layers are used to provide additional information for IFC entities which belongs to upper layer or domain specific IFC schemas.

*IfcObjectPlacemen*t defines placement of an object as absolute placement (relative to world coordinate system) in space or relative placement (relative to placement location of other object) or constrained placement (relative to design grid axis) as depicted in Figure 36. Coordinate system for object placement can be either in two-dimension axis placement (*IfcAxis2Placement2D*) or in three-dimensional axis placement (*IfcAxis2Placement3D*). *IfcProductRepresentation* is related to representation (geometric, topology, mapped) of a product. Any object can have many or no geometric representations and all the geometric representation for a particular object should be defined within same coordinate system (Figure 36). *IfcRepresentation* defines the general concept of representing product properties and shape. It can be used to define geometric, topology and shape representation to products using representation item.

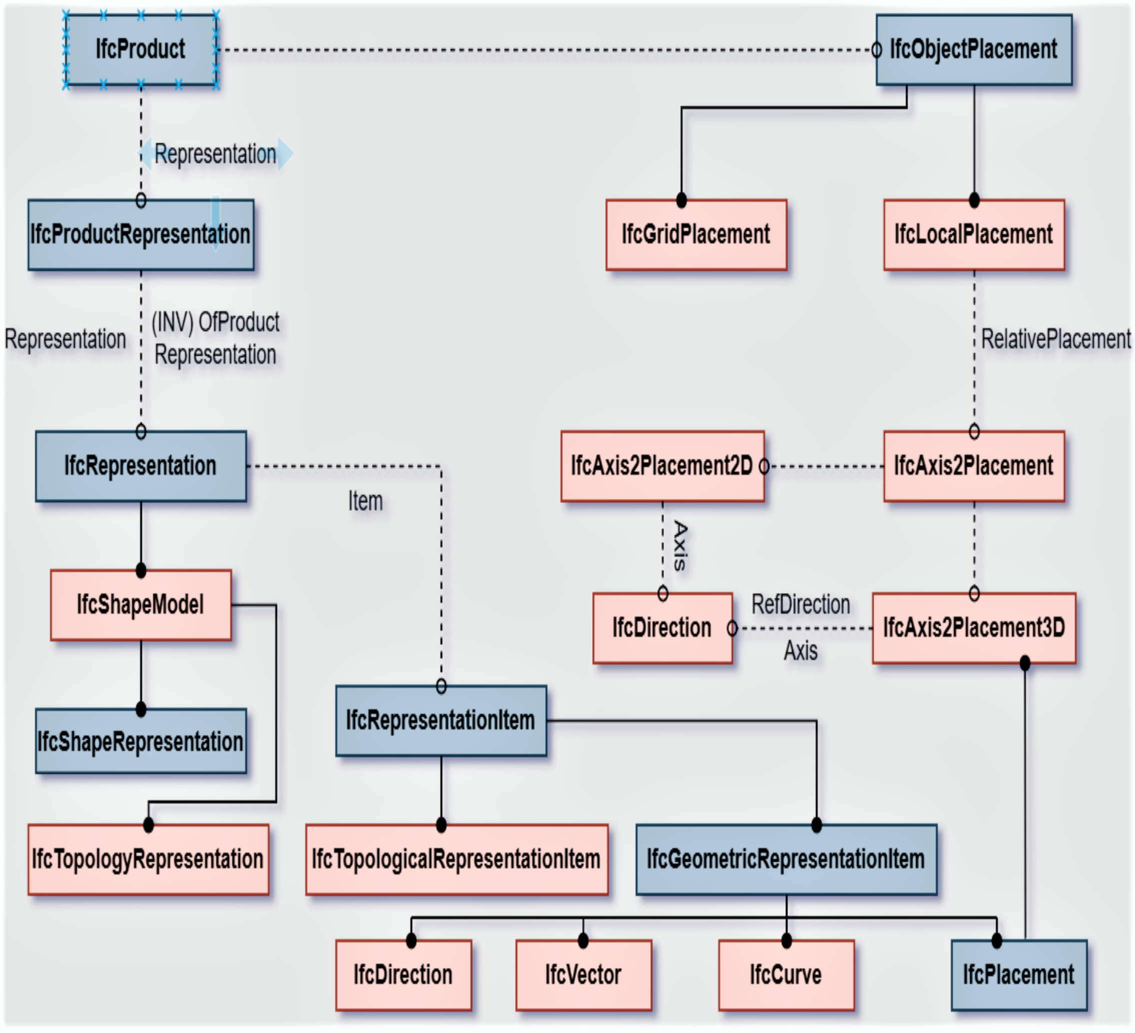


Figure 36: IFC schemas for Geometric Representation and Object Placement [12]

IfcShapeRepresentation defines the concept of geometry of a product or a product component. It includes inherited attributes like **RepresentationType** to define model (Curve, surface) used for shape representation and attribute **RepresentationIdentifier** depicts the kind of representation (Axis, body) captured by shape representation. *IfcObjectPlacement* has to be provided for each product that has shape representation. As illustrated in Figure 37, elements of *IfcRepresentationItem* are used to identify shape representation of any object defined under *IfcProduct* and are used within entity *IfcRepresentation*. *IfcRepresentationItem* are generally geometrical or topological representation items which participates directly for representation of an element or contributes to define other representation items. *IfcRepresentationItem* have four subclasses for different representation (geometric, shape, topology).

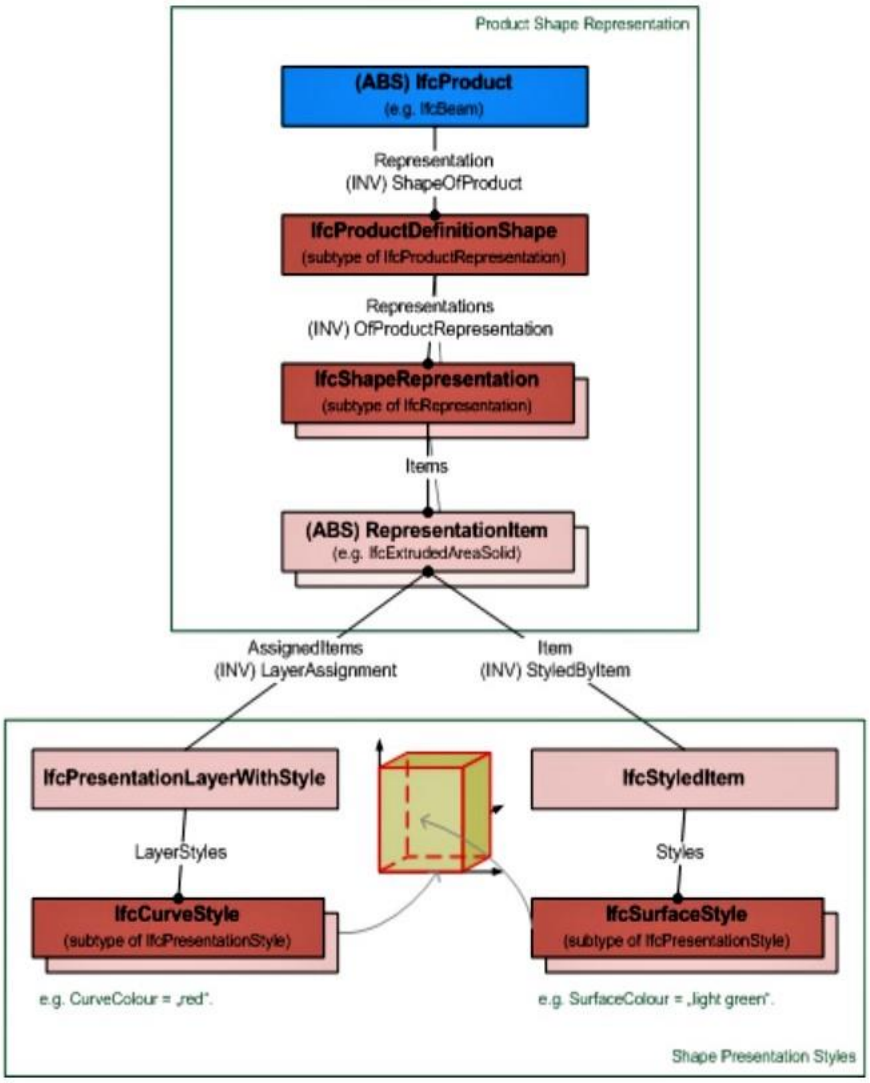

Product Shape Representation
(ABS) IfcProduct
(e.g. IfcBeam)
Representation
(INV) ShapeOfProduct
IfcProductDefinitionShape
(subtype of IfcProductRepresentation)
Representations
(INV) OfProductRepresentation
IfcShapeRepresentation
(subtype of IfcRepresentation)
Items
(ABS) RepresentationItem
(e.g. IfcExtrudedAreaSolid)
AssignedItems
(INV) LayerAssignment
Item
(INV) StyledByItem
IfcPresentationLayerWithStyle
IfcStyledItem
LayerStyles
Styles
IfcCurveStyle
(subtype of IfcPresentationStyle)
IfcSurfaceStyle
(subtype of IfcPresentationStyle)
e.g. CurveColour = „red".
e.g. SurfaceColour = „light green".
Shape Presentation Styles

Figure 37: Representation item style [12]

Topology representation of structural curve members and structural surface members are identified using elements of *IfcTopologicalRepresentationItem*. Edge members i.e., beam, columns consist edge information given be *IfcEdge*. This entity defines topological connection between two vertices (start vertex point and end vertex point). *IfcVertexPoint* is computed through 3D coordinates represented by *IfcCartesianPoint* (Figure 38). To represent Surface members *IfcFace*, *IfcLoop*, *IfcFaceBound* etc. entities are used.

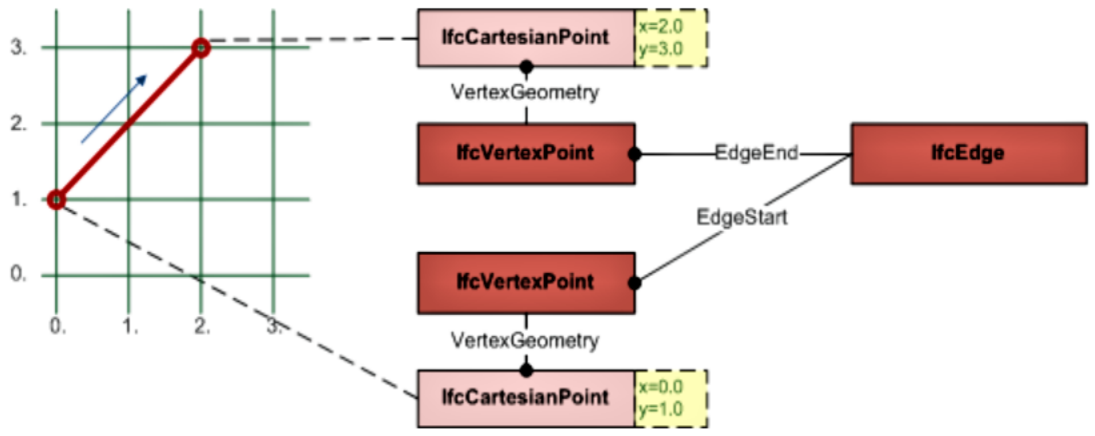


Figure 38: Topology representation: Edge and Vertex [12]

3.1.3 Material and Property Definition

Properties of structural members are inherited attribute from superclass *IfcObject*. Structural members within IFC data do not carry all the property definitions within itself and hence, the relationship between object and property definitions provides this extra information about members. Objectified relationship *IfcRelDefinesByType* is used to establish relationship between an object and object type (Figure 39), which is defined by a set of properties. All properties are accommodated in entity *IfcPropertySet*. Each property within *IfcProperty* is assigned with a name and further details.

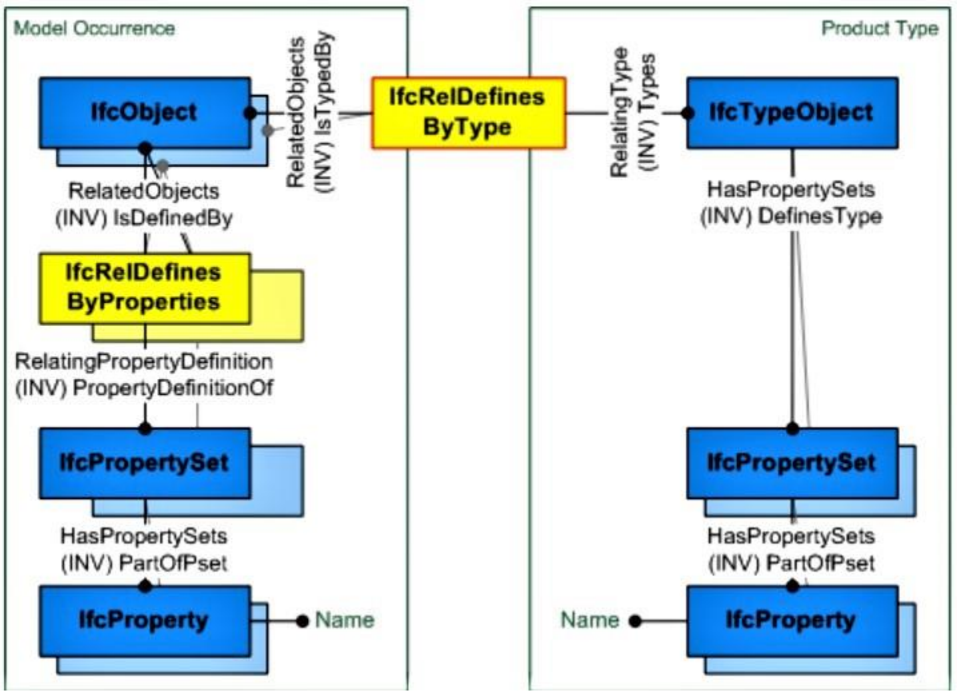


Figure 39: Relationship between object and its properties [12]

The material definition of structural members is given by entity *IfcMaterial*. The material parameters are linked to structural members using *IfcRelAssociatesMateria*l relationship.

IfcMaterialLayerSetUsage is used for layered elements (walls, slabs) to depict certain direction and offset from the axis reference curve and represent the positioning of layer set through entity *IfcMaterialLayerSet*. *IfcMaterialLayer* lies within *IfcMaterialLayerSet* and defines the relevant parameters like thickness of each layer.

Definition of standard profiles of structural curve members (beam, column) is given by *IfcProfileDef* and is used to define standard set of commonly used section profiles by member geometry. This entity is associated with *IfcMaterialProfileSet* and *IfcMaterialProfileSetUsage* to give section information of structural elements. Figure 40 illustrates the occurrence graph of a beam in IFC data. Alignment of the material profile set according to cardinal point is indicated by *IfcMaterialProfileSetUsage. IfcMaterialProfileSet* depicts the profile of single material.

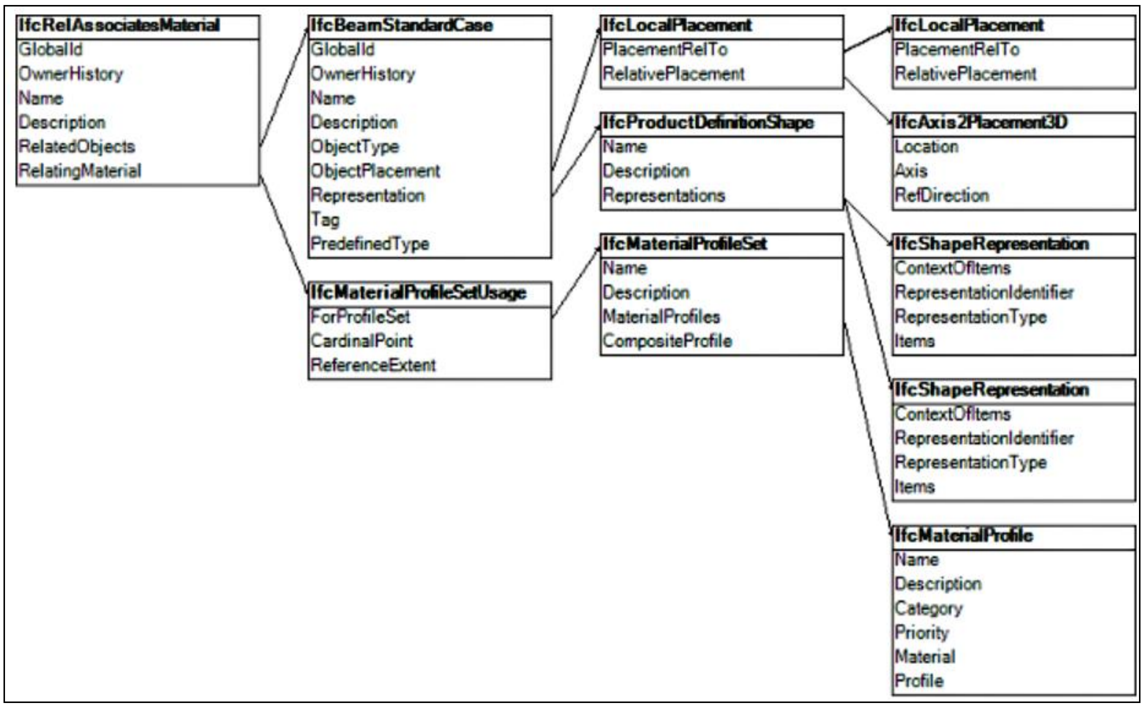


Figure 40: Beam occurrence graph in IFC data [12]

3.1.4 Load Definition

Loads assignment and information are crucial factor for structural analysis domain. Load is generally a weight, pressure or force that acts on a structural element and create instability. Proper calculation and analysis of reaction or instability generated by loads are important for building lifecycle. IFC entity *IfcStructuralActivity* combines definition of all actions (forces, displacements etc.) and reactions (deflections, support reactions etc.) that acts on a structural member (Figure 41). Relationship *IfcRelConnectsStructuralActivity* interrelates structural activity to structural member as shown in Figure 41. Subclasses of *IfcStructuralActivity* inherit attributes that defines a load and establish the relationship of loads with connections and structural members. For example, attribute **AppliedLoad** defines the load type, direction of load and values of applied load. **GlobalOrLocal** attribute describes weather the coordinate system of loading direction is relative to global coordinate system or local coordinate system.

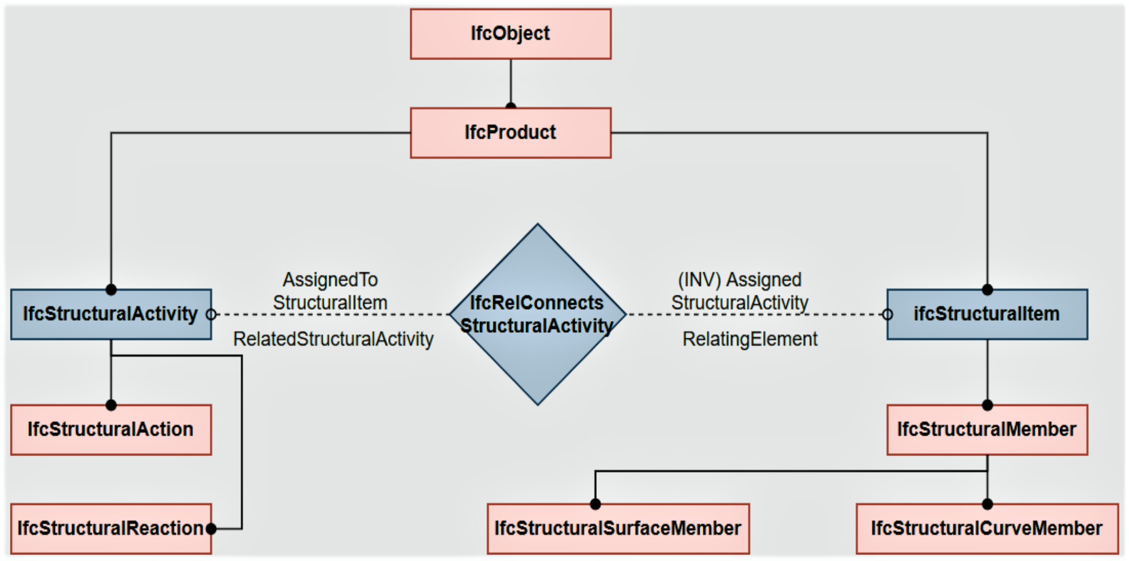


Figure 41: Relationship between Structural activity and Structural item [12]

Global coordinate system is established by **ObjectPlacement** attribute and is common for structural items and structural activities while, local coordinate system is defined by attribute **Representation** and is local to structural item. *IfcStructuralAction* and *ifcStructuralReaction* are subclasses of IfcStructuralActivity and defines structural action that acts upon a structural member and generated reaction from applied action respectively. Representation and placement of structural activities and connected structural items are identical if:

- Instances of *IfcStructuralPointAction* or *IfcStructuralPointReaction* connect with a point item.
- Instances of *IfcStructuralCurveAction* or *IfcStructuralCurveReaction* connect with a curve item.
- Instances of *IfcStructuralSurfaceAction* or *IfcStructuralSurfaceReaction* connect with a surface item and acts on the entire surface.

If the connection between instances of *IfcStructuralActivity* and *IfcStructuralItem* is not within same dimension then instances of structural activity will have placement and product representation.

IfcStructuralLoad is an IFC entity associated with *IfcStructuralActivity* which is supertype of all the loads defined for structural analysis. Attribute **Name** provides the name to the load defined through the identity. *IfcStructuralLoadorResult* is abstract class for simple loads through which all static loads like linear force, planer force, temperature load is defined. Another class *IfcSurfaceReinforcementArea* provides reinforcement area details (shear reinforcement, surface reinforcement) of structural member through attributes **ShearReinforcement** and **SurfaceReinforcement.** Figure 43 depicts the structural load graph of a curve member within IFC data.

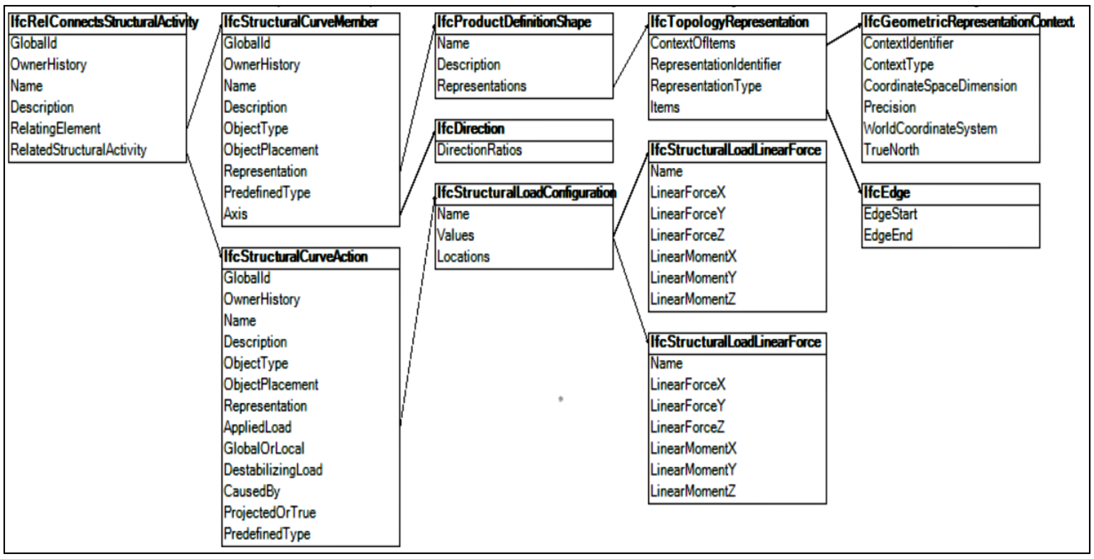


Figure 42: Structural load graph for structural curve member in IFC data [12]

3.1.5 Load combination and Analysis Results

In this section, IFC schemas related to load combination, analysis method, analysis results and reactions are described. *IfcGroup* class belongs to IFC core data schema is represents all arbitrary groups and uses *IfcRelAssignsToGroup* relationship to establish group collection. A group can be any collection of objects (products, processes etc.).

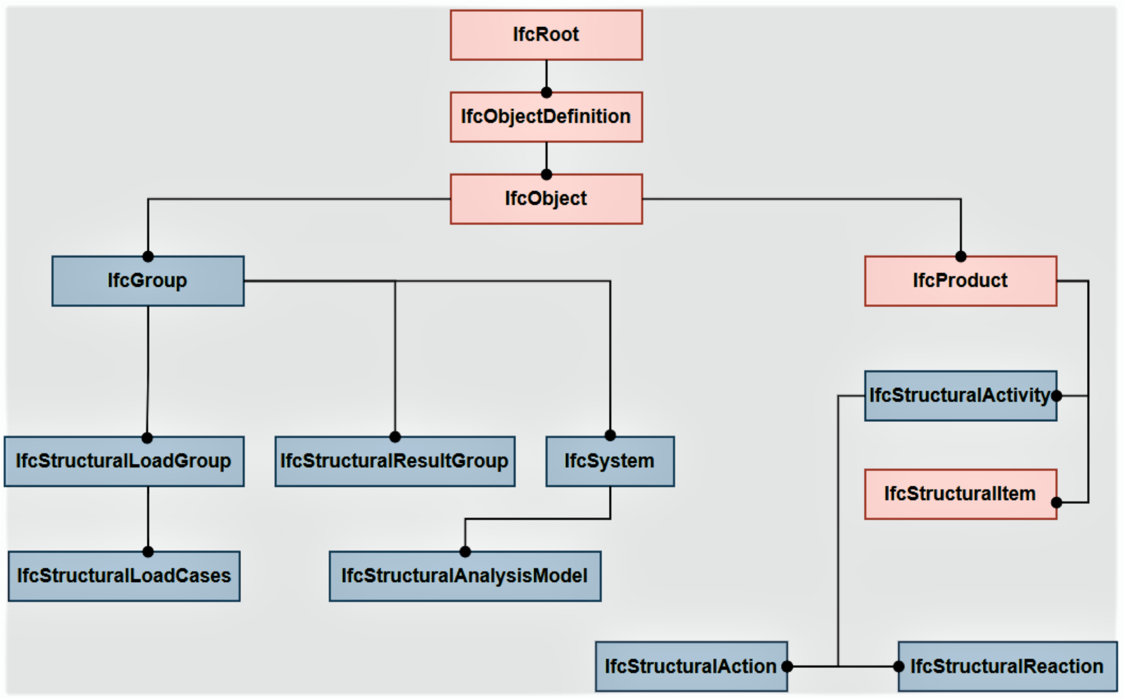


Figure 43: IFC schemas for load combination and structural analysis [12]

IfcStructuralLoadGroup provides the grouping mechanism for instances of *IfcStructuralAction* (including subclasses) through relationship *IfcRelAssignToGroup* (Figure 44). These groups can be defined as load groups, load cases or load combination.

Assignment relationship *IfcRelAssignsToGroupByFactor* is used to group load cases into load combinations by providing factor for load cases in load combination (Figure 44).

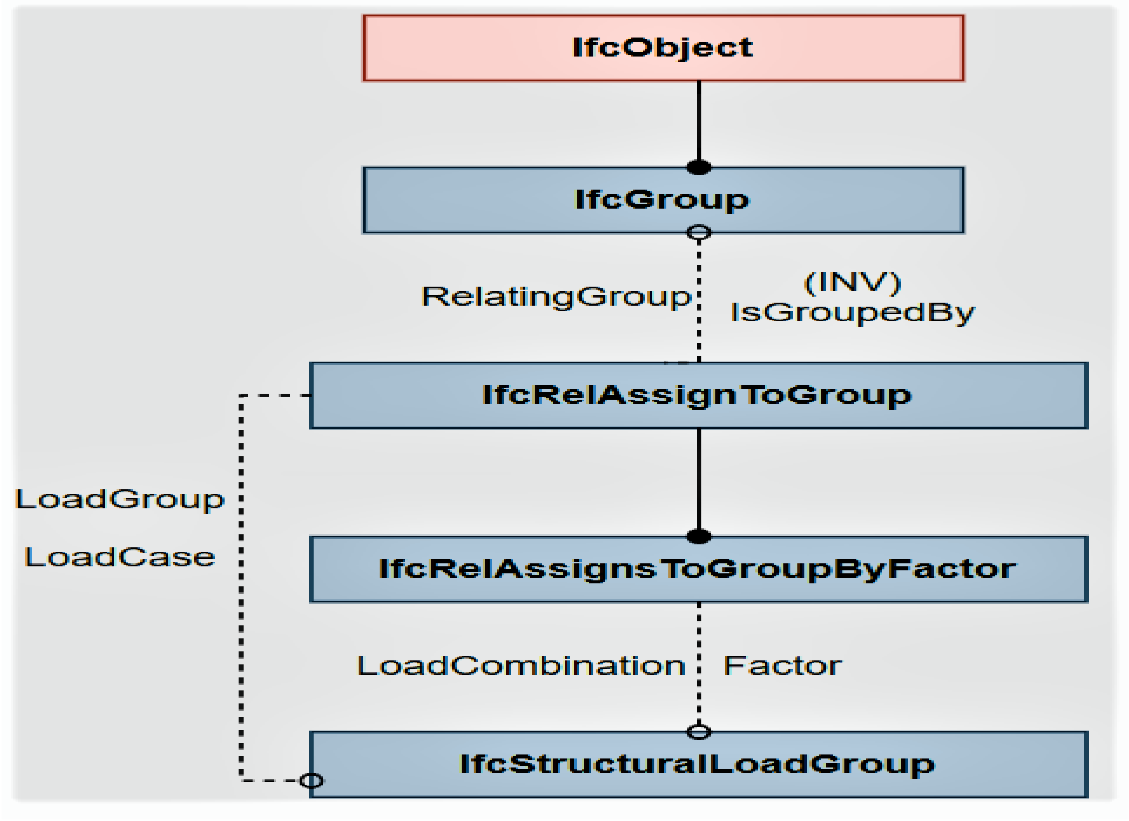


Figure 44: Load group in IFC data [12]

IfcLoadGroupTypeEnum entity is used to differentiate between load groups, load cases and load combinations, which are defined as follow:

- LOAD_GROUP: Container for loads which are instances of class *IfcStructuralAction* and are grouped together for specific purpose. Instances of these groups (LOAD_GROUP) are members of *IfcStructuralLoadGroup*.
- LOAD_CASE: Container for LOAD_GROUPs and instances of class *IfcStructuralAction*. Instances of these groups (LOAD_CASE) belong to class *IfcStructuralLoadCase* and contain loads that originate from same source.
- LOAD_COMBINATION: Instances of LOAD_CASEs are provided with a factor through *IfcRelAssignsToGroupByFactor* to group load cases into load combinations, which are contained in LOAD_COMBINATION. Instances of these groups only contains instances of class *IfcStructuralLoadCase*.

IfcStructuralAnalysisModel class is defined to gather all needed information to represent structural analysis model. It collects all the data related to structural member, load cases and combinations, member connections with all load results. Grouping of structural members and structural connection is performed by inherited inverse attribute **IsGroupedBy** and hierarchy between different analysis model is provided by another inherited inverse attribute **IsDecomposedBy** from *IfcObject***.** *IfcAnalysisModelTypeEnum* entity differentiates between different analysis model based on analysis model dimensionality (IN_PLANE_LOADING_2D, OUT_PLANE_LOADING_2D, LOADING_3D). Further user defined analysis model can also be identified using above entity. Information related to load cases and load combinations defined in *IfcStructuralLoadGroup* class are collected through attribute **LoadedBy** (Figure 45).

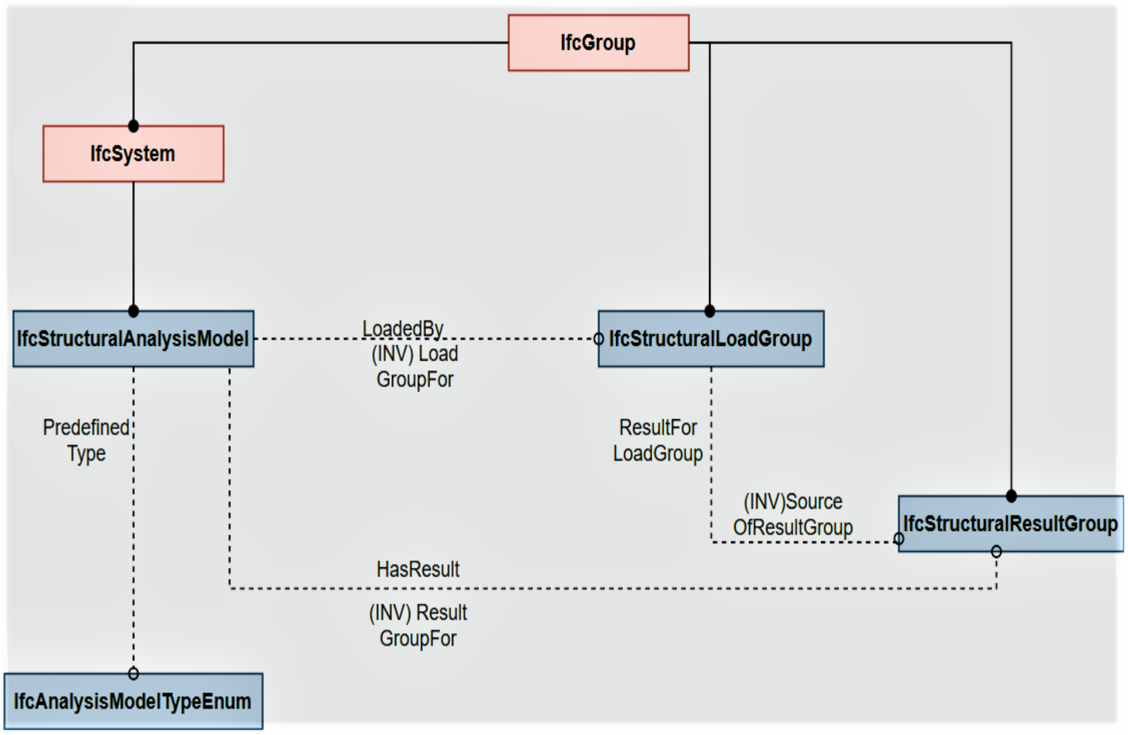


Figure 45: Relationship between load group, analysis model and result group [12]

Reactions or results (deflections, internal forces) resulted from applied loads and analyzed through analysis model are described in *IfcStructuralReaction* class. Analyzed result information are captured by instances of class *IfcStructuralResultGroup* through inverse attribute **ResultGroupFor** (Figure 45). Further results are grouped through class *IfcStructuralResultGroup* to capture the connection to the underlying basic load group. Grouping of results are carried out by inverse relationship *HasAssignmen*t and *IfcRelAssignToGroup* relationship inherited from class *IfcGroup*. Information of analysis theory is given by attribute **TheoryType** which is associated with class *IfcAnalysisTheoryTypeEnum*.

3.2 Development of Domain Ontologies

Four ontologies namely: Building Element Ontology (BEO), Building Element Representation Ontology (BERO), Building Load Ontology (BLO) and Structural Analysis Ontology (SAO) were developed to denote the information from architectural model and structural analysis model in order to achieve semantic interoperability between two domains.

3.2.1 Tool and Ontology development process

Protégé – a free and open-source ontology editor tool is used in this thesis for development of domain ontologies. Protégé is based on JAVA and supports latest OWL 2 Web Ontology Language and RDF specifications from the W3C. Protégé allows the user to create, visualize (Onto graph) and manipulate ontologies in various syntax (Turtle, RDF/XML) and enables to find many support and plugins to work with the platform. Further, use of reasoners like Hermit enables users to check consistency of ontology and helps in inference of additional information based on ontology knowledge.

Ontologies represents concepts of a domain of interest through class and class hierarchy (major and important concepts of domain), properties and sub properties (features, attributes and relationship between classes), class and property restrictions and individuals (instances representing classes). Hence, to develop an ontology determination of the scope and area of ontology is important. In simple terms, answer to question that, "what knowledge should the proposed ontology represent", must be clear. Further, concepts and knowledge related to interested domain must be acquired in order to use that concepts to define classes, properties and restrictions to the classes to develop ontology. Hence, important steps for ontology construction processes includes:

- **Scope and role of ontology:** Defining domain and scope of projected ontology is important in order to determine that what information is represented through ontology and for what purpose ontology can be used.
- **Reusing existing ontology:** If needed, existing ontologies can be identified, reused and referred using web. It is one of the applications of ontological approach that it can be linked on web, identified using URIs and then reused for any use. It reduces effort and time for development of new ontology.
- **Defining Class and class hierarchy:** Development of classes and class hierarchy is most important for ontology construction. Classes and class hierarchy are based on well-defined taxonomy. A parent class or superclass or main class is a major term or concept related to domain of ontology and child class, or subclass is a narrow term included in super class. Properties and attributes from superclass are inherited to its subclasses which specify that every instance of superclass is also an instance of related superclass. Class hierarchy can be developed by top-down approach, bottom-up approach or combination of both approaches depending on ontology domain and user.
- **Defining properties and attributes:** Using OWL, two types of properties (object property and data property) can be assigned to classes. Properties are used to establish relationship between classes or to assign a particular data value to a class or a group of classes. Data value can be integer, real, string double etc. Like classes, property hierarchy can also be designed including main property and subproperties.
- **Assigning domain and range to properties:** Domain and range is assigned to properties based on RDF triples. A class or resource to which a property is

attached is domain of property and value or RDF statement is range of property. Range of an object property is a class while, range of data property is a data type.

- **Defining restrictions:** Restrictions are provided to properties or classes to define in order to represent certain exact information through ontologies. A property can be provided with exact data type it can belong or data type that property cannot take. Moreover, classes and properties can be restricted to number of values that can be assigned to them through cardinality axioms. Similarly, axioms like negative object property assertion, negative data property assertion can be assigned to certain class which implies that instance belonging to that class cannot be defined using particular asserted properties.
- **Creating individuals:** Individuals are instances belonging to a certain class or can represent a group of class. Individuals are used to assign data value to classes.

3.2.2 Annotations and Namespaces

To capture and exchange the information from structural analysis model, ontologies for structural analysis domain were developed. Structural analysis domain contains a vast amount of data and representation of every concept in a single ontology is quite confusing and would create a lot of ambiguity. To overcome mentioned problem, four different ontologies were created which represents certain specific concepts of structural analysis domain and are interconnected to each other (Figure 46). Namespaces used in ontologies are given in Table 3. These ontologies are:

- **Building Element Ontology (BEO)**: BEO represents concepts of zone, building site, building storey etc. as in Building Topology Ontology (BOT) and general structural elements like beam, column, slabs etc with their properties.
- **Building Element Representation Ontology (BERO):** BERO contains concepts related to general geometrical and topological representation of building elements.
- **Building Load Ontology (BLO):** In this ontology, concepts related to different loads (dead load, environmental load etc.) acting on building and building structural members, type of loads (point load, distributed load etc.) and load properties are defined.
- **Structural Analysis Ontology (SAO):** SAO represents structural analysis model concepts. Concepts related to load coefficient and load combination system, structural analysis method, load calculation results etc. are defined in this ontology.

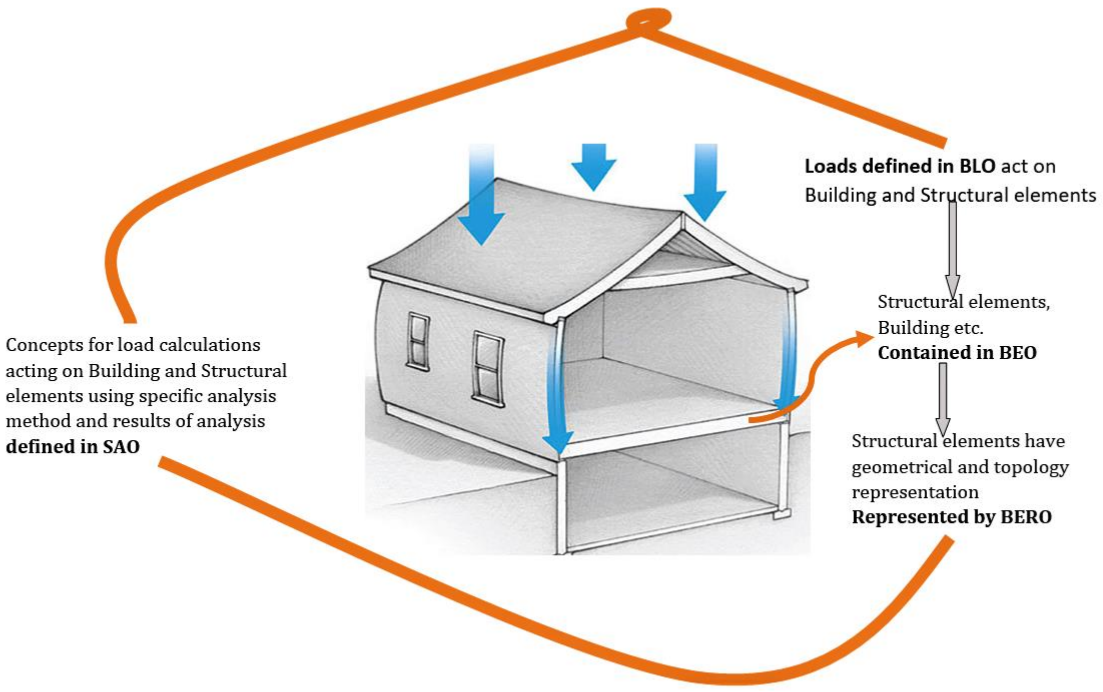

Loads defined in BLO act on Building and Structural elements
Structural elements, Building etc. Contained in BEO
Structural elements have geometrical and topology representation Represented by BERO
Concepts for load calculations acting on Building and Structural elements using specific analysis method and results of analysis defined in SAO

Figure 46: Interconnection between developed domain ontologies

Table 3: Namespaces used for ontology

beo	<http://www.biwcib-jkppa/BuildingElement#>
bero	<http://www.biwcib-jkppa/ElementRepresentation#>
blo	<http://www.biwcib-jkppa/Load#>
sao	<http://www.biwcib-jkppa/StructuralAnalysis#>
owl	<http://www.w3.org/2002/07/owl#>
rdf	<http://www.w3.org/1999/02/22-ref-syntax-ns#>
rdfs	<http://www.w3.org/2000/01/rdf-schema#>
xml	<http://www.w3.org/XML/1998/namespace>
xsd	<http://www.w3.org/2001/XMLSchema#>
terms	<http://purl.org/dc/terms/>

3.2.3 Building Element Ontology (BEO)

Structural element's information is one of the most basic and crucial data for structural analysis domain. Structural element data includes many vital information like geometric cross section information, section property information, strength and material information etc. Most of this information is extracted from architectural model for structural analysis so, correct and meaningful mapping of information from architectural model to structural analysis model during data exchange is important.

IFC model represent structural curve member (beam, column) and structural surface member (slab, wall) in different classes and their properties are specified by distinct attributes and entities but in BEO all structural members are defined in same class hierarchy. BEO represents information about common structural elements and their properties. Additionally, with reference of Building Topology Ontology (BOT) concepts of zone, space, building site, building and building storey is also described in this ontology. Diagrammatical structure of BEO is depicted in Figure 47.

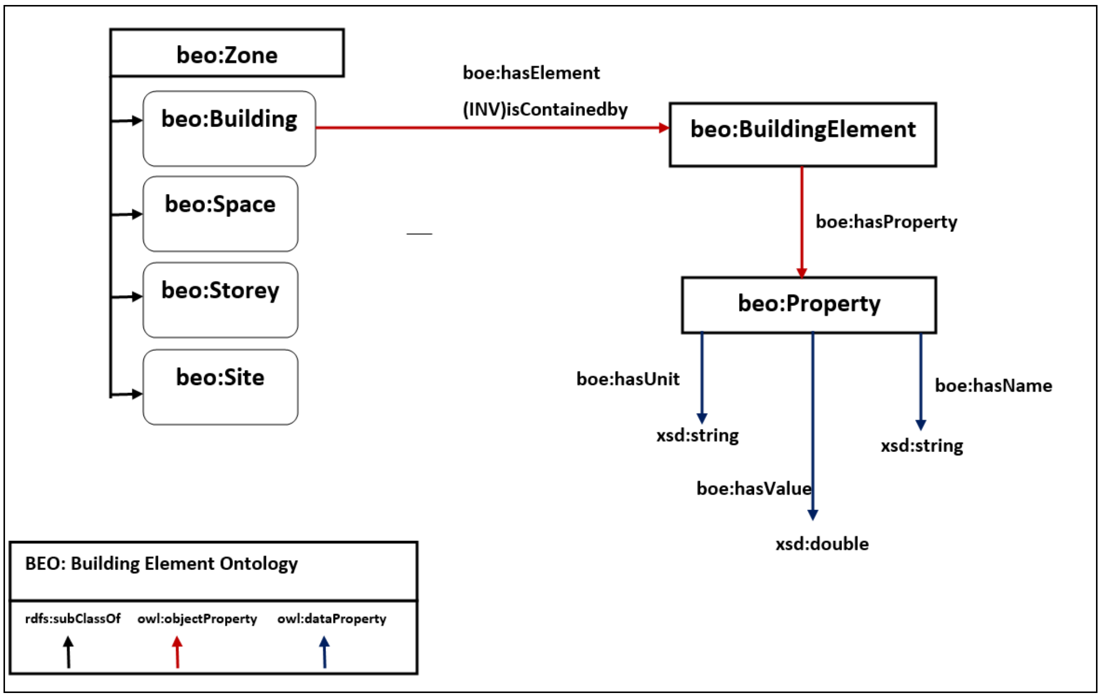
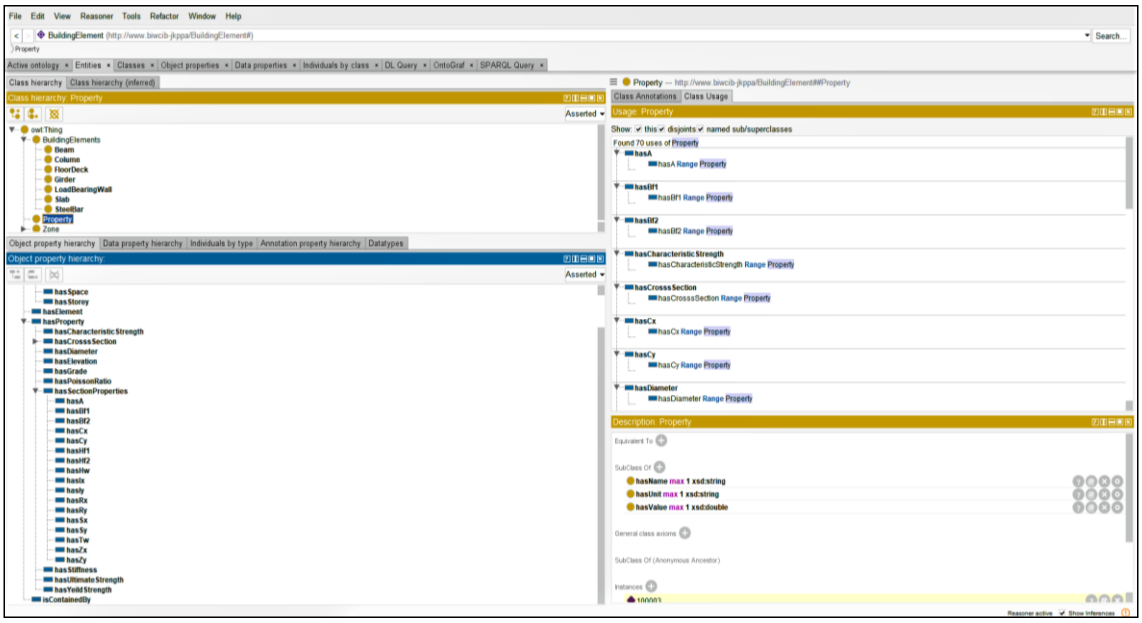


Figure 47: Basic structure of Building Element Ontology

Zone and sub-zones: Zone is part of the world which is located either physically or virtually, represented as beo:Zone in ontology. Superclass beo:Zone consist four subclasses namely: beo:Site, beo:Building, beo:Storey and beo:Space. Site is a location or area where building is located. Building is structure that is being built. Building may describe a single structure or a number of structures. Storey is level part of building and contained in building while, space is three-dimensional extent contained in storey. All subclasses of beo:Zone are disjoint classes with each other which denotes that any instance which belongs to one of these classes cannot represent other classes. Object property beo:containsZone is a symmetric and transitive property. Relationship between zone and its subclasses are represented as RDF triples below:

- **beo:Zone** *beo:containsZone* **beo:Zone** (symmetric and transitive property)
- **beo:Zone** *beo:hasBuilding* **beo:Building**
- **beo:Zone** *beo:hasSpace* **beo:Space**
- **beo:Zone** *beo:hasSite* **beo:Site**
- **beo:Zone** *beo:hasStorey* **beo:Storey**

Building elements and properties: Class beo:BuildingElements is superclass for all the specific structural elements (beam, column, slabs etc.) used during building construction. Relationship between class beo:Building and class beo:BuildingElements is defined by object property beo:hasElements. Object property beo:isContainedBy is inverse property of beo:hasElements (Figure 48). Each building elements have several properties like cross section, strength, stiffness, section properties etc. Object property beo:hasProperty connects class beo:BuildingElement and beo:Property. According to concept of hierarchy, each subclass of class beo:BuildingElement inherits that particular relationship.



```
BuildingElement (http://www.biwcib-jkppa/BuildingElement#)
〉Property

Class hierarchy: Property
owl:Thing
  BuildingElements
    Beam
    Column
    FloorDeck
    Girder
    LoadBearingWall
    Slab
    SteelBar
  Property
  Zone

Object property hierarchy:
    hasSpace
    hasStorey
  hasElement
  hasProperty
    hasCharacteristicStrength
    hasCrosssSection
    hasDiameter
    hasElevation
    hasGrade
    hasPoissonRatio
    hasSectionProperties
      hasA
      hasBf1
      hasBf2
      hasCx
      hasCy
      hasHf1
      hasHf2
      hasHw
      hasIx
      hasIy
      hasRx
      hasRy
      hasSx
      hasSy
      hasTw
      hasZx
      hasZy
    hasStiffness
    hasUltimateStrength
    hasYeildStrength
  isContainedBy

Property — http://www.biwcib-jkppa/BuildingElement#Property
Usage: Property
Show: ✓ this ✓ disjoints ✓ named sub/superclasses
Found 70 uses of Property
hasA
  hasA Range Property
hasBf1
  hasBf1 Range Property
hasBf2
  hasBf2 Range Property
hasCharacteristicStrength
  hasCharacteristicStrength Range Property
hasCrosssSection
  hasCrosssSection Range Property
hasCx
  hasCx Range Property
hasCy
  hasCy Range Property
hasDiameter
  hasDiameter Range Property

Description: Property
Equivalent To
SubClass Of
  hasName max 1 xsd:string
  hasUnit max 1 xsd:string
  hasValue max 1 xsd:double
General class axioms
SubClass Of (Anonymous Ancestor)
Instances
  100003
```

Figure 48: Building element and related object properties of BEO in Protégé

Class beo:Property is domain of three data properties beo:hasName (data type - xsd:string), beo:hasUnit (data type - xsd:string), beo:hasValue (data type - xsd:double). Each property of structural member can be identified using this relationship. For example, let us assume that structural member Slab (beo:Slab) has structural length (beo:hasLength) 2 meters as a property (beo:Property). So, through data property beo:hasName, property name "Length" can be identified. Similarly, data property beo:hasValue will assign "2" as length value and beo:hasUnit will identify meter as "Unit" of length (Figure 49).

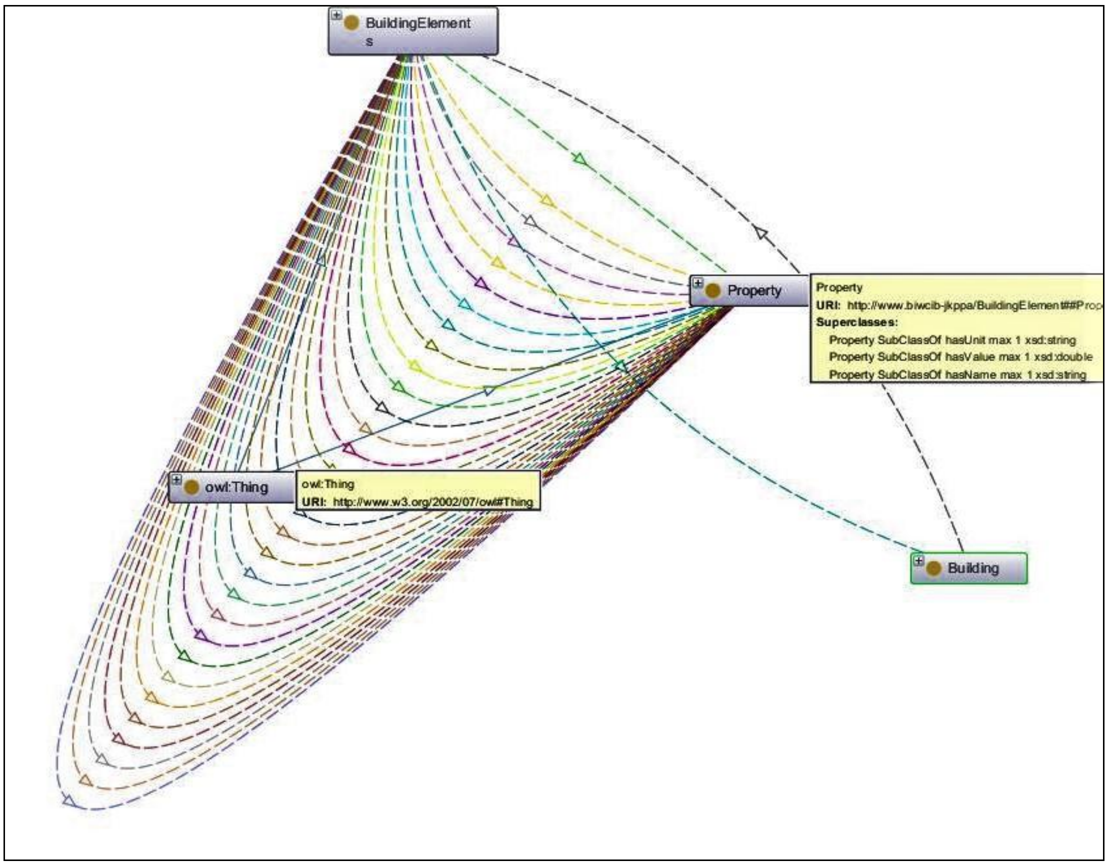

BuildingElements
Property
Property
URI: http://www.biwcib-jkppa/BuildingElement##Prop
Superclasses:
Property SubClassOf hasUnit max 1 xsd:string
Property SubClassOf hasValue max 1 xsd:double
Property SubClassOf hasName max 1 xsd:string
owl:Thing
owl:Thing
URI: http://www.w3.org/2002/07/owl#Thing
Building

Figure 49: Building elements, Building and Property relationship graph

Cardinality restrictions were provided to class beo:Peoperty:

hasName **max** 1 xsd:string: Maximum one (or zero) value can be assigned to class beo:Property through attribute (data property) beo:hasName. Data type "String" is specified for this data property. Certain structural member like walls do not contain some specific properties like section properties, in that case no value will be assigned to the class.

hasValue **max** 1 xsd:double: Maximum one (or Zero) value can be assigned to class beo:Property through attribute (data property) beo:hasValue. Data type "Double" is specified for this data property

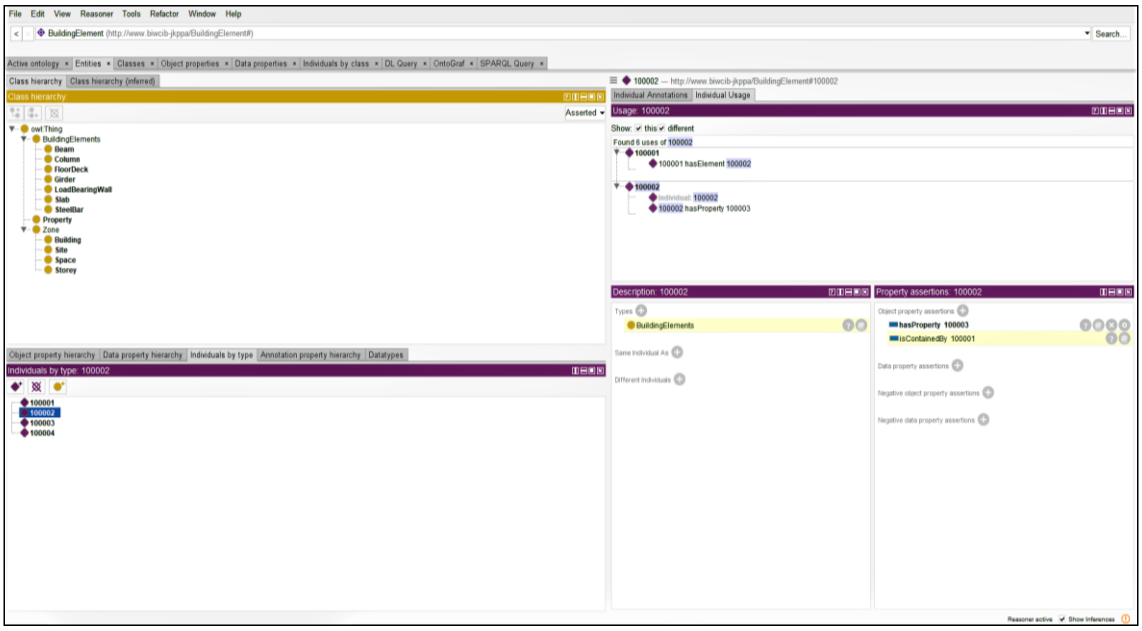
hasUnit **max** 1 xsd:string: Maximum one (or Zero) value can be assigned to class beo:Property through attribute (data property) beo:hasUnit. Data type "String" is specified for this data property

Reasoning and inference: Protégé allow reasoning and inference of new information which was not asserted before, through reasoner like Hermit. Reasoners are used to check consistency of ontology and provide new information on the basis of already asserted concepts in ontology. For example, four individuals were created namely 100001, 100002, 100003 and 100004 and certain details were provided for each individual like:

- Individual 100001 has object property **beo:hasElement** (Individual 100002)
- Individual 100002 has object property **beo:hasProperty** (Individual 100003)

- Individual 100003 has data property **beo:hasValue** (56.0) data type - xsd:double
- Individual 100004 has object property **beo:hasBuilding** (Individual 100001)

Reasoner inferred on the basis of provided knowledge and concept that individual 100001 is an instance of class beo:Building (Figure 50), individual 100002 belongs to class beo:BuildingElements, individual 100003 fits for class beo:Property and last individual is an instance of class beo:Zone.



Class hierarchy:

- owl:Thing
  - BuildingElements
    - Beam
    - Column
    - FloorDeck
    - Girder
    - LoadBearingWall
    - Slab
    - SteelBar
  - Property
  - Zone
    - Building
    - Site
    - Space
    - Storey

Individuals by type: 100002

- 100001
- 100002
- 100003
- 100004

100002 — http://www.biwcib-jkppa/BuildingElement#100002

Usage: 100002

Show: ✓ this ✓ different

Found 6 uses of 100002

- 100001
  - 100001 hasElement 100002
- 100002
  - Individual: 100002
  - 100002 hasProperty 100003

Description: 100002

Types
BuildingElements

Same Individual As

Different Individuals

Property assertions: 100002

Object property assertions
hasProperty 100003
isContainedBy 100001

Data property assertions

Negative object property assertions

Negative data property assertions

Figure 50: Reasoning and inferred information through reasoner

3.2.4 Building Load Ontology (BLO)

Loads and loading activities are applied on building structure and structural members and identification of loading conditions and loads are one of the most vital job in structural analysis domain. Quality, strength, material, design and placement of building elements highly depends on loading scenarios. Loading scenarios depends on many conditions like material of structural member, environmental conditions (wind zone, snow zone), use purpose of structure (residential, commercial) etc. Hence, structural analysis process mostly works around load calculation and calculation results. This information is crucial for various domains of AEC industry and efficient and semantic interoperability is important to achieve seamless data transfer of such information in collaborative work flow.

BLO defines the concepts of load, type of load, loading direction including values, location of loads. Further environmental loads and their load zone is also represented in BLO. Entities (beo:BuildingElement etc.) from Building Load Ontology is also used for construction of Building Load Ontology since, both ontologies are interconnected. Figure 51 illustrates the overview of classes and properties defined in BLO.

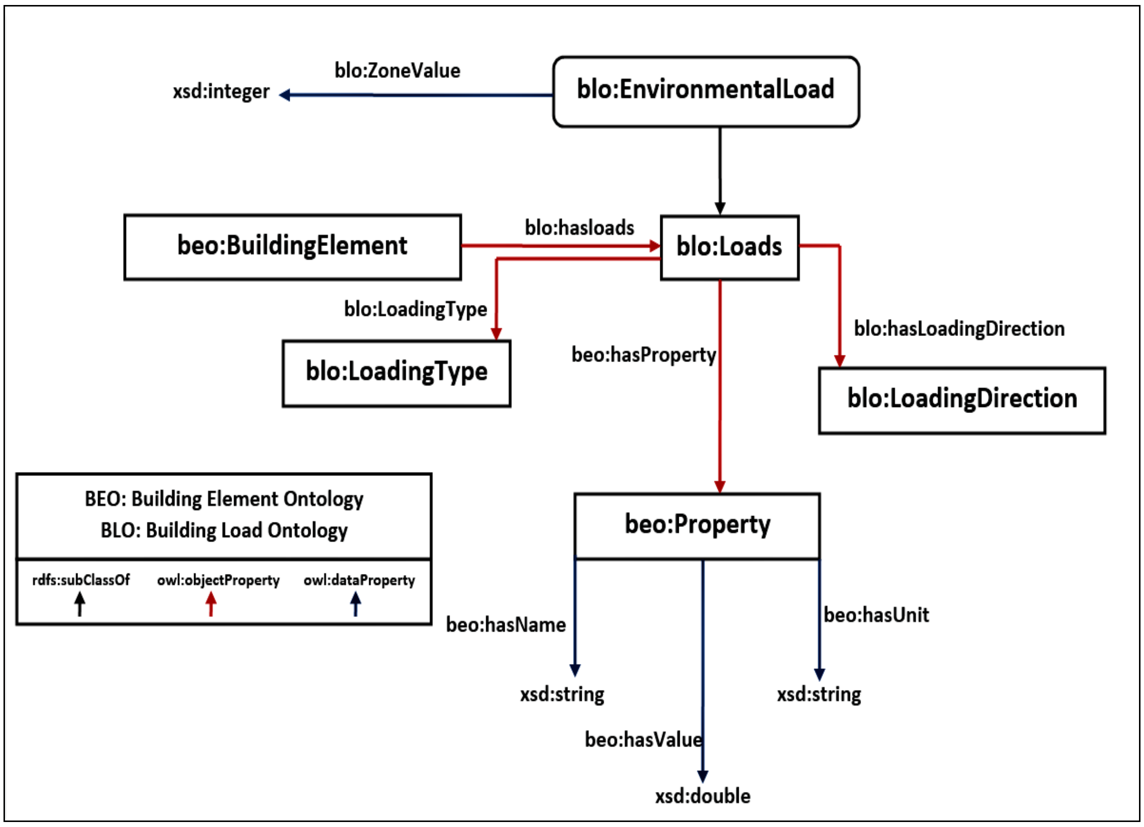


Figure 51: Overview of Building Load Ontology

Loads and properties: Superclass blo:Loads represents the all types of loads like dead load (blo:DeadLoad), environmental load (blo:EnvironmentalLoad), impact load (blo:ImpactLoad) and live load (blo:LiveLoad). Class blo:EnvironmentalLoad is parent class for blo:SnowLoad and class blo:WindLoad whereas, classes blo:MachineryLoad and blo:Vibrations are subclasses of blo:ImpactLoad. Object property blo:hasLoads connect class beo:BuildingElement and class blo:Loads. Every load has certain property and class beo:Property which is linked through object property beo:hasProperty. beo:hasName (data type - xsd:string), beo:hasUnit (data type - xsd:string), beo:hasValue (data type - xsd:double) are three data properties with their domain as beo:Property to identify the specific name, value and unit of load. Graphical representation of load hierarchy and relationship between loads and properties is provided in Figure 52.

Data property blo:hasCartesianLoacation provides location of the load in coordinate system and has data type xsd:string and data property blo:LocationReference (data type – xsd:string) informs weather provided coordinate system is global or local. Class blo:EnvironmentalLoad is explicitly linked to datatype blo:ZoneValue (data type – xsd:integer) for determination of zone (Figure 51) for wind load and snow load as each zone is associated with a special factor that contributes in load combination method during structural analysis process.

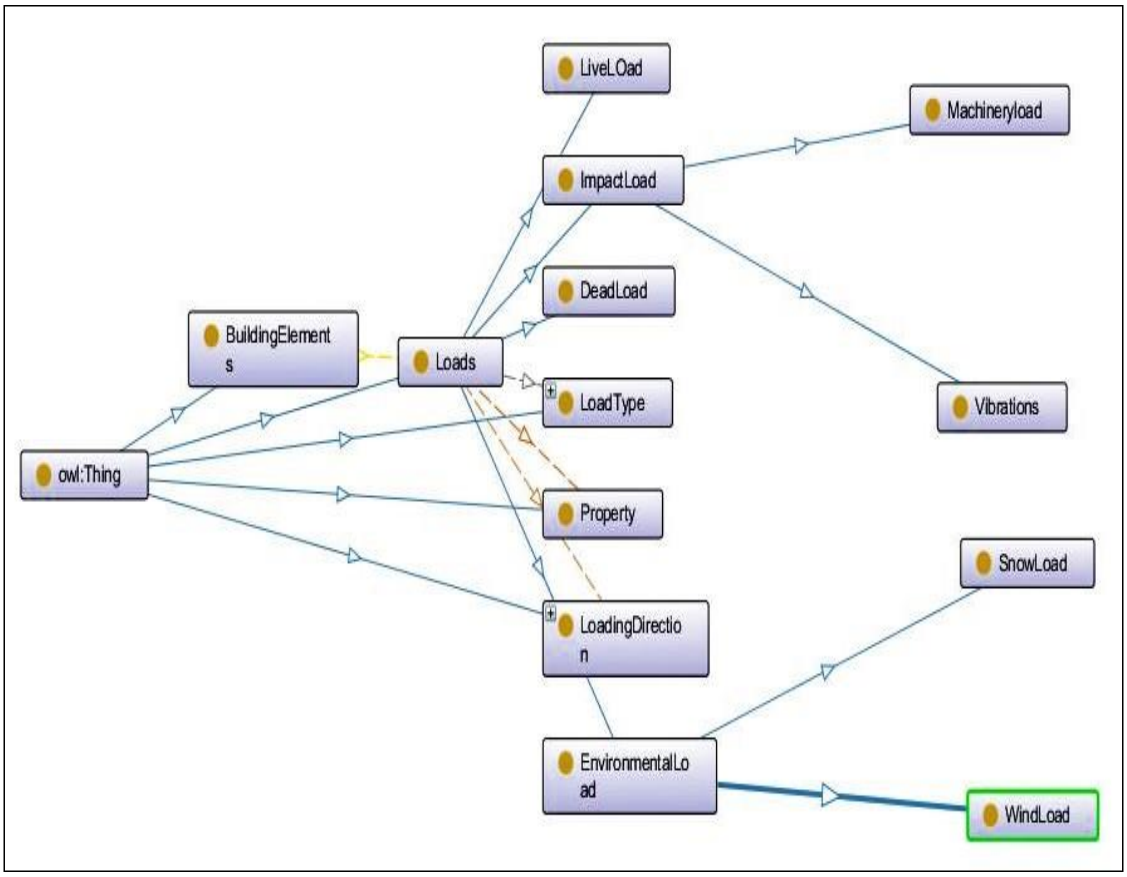


Figure 52: Load class hierarchy and relationship with Property graph

Load type and loading direction: Loads can act on a single point referred as point load or on a whole surface called as distributed load. Defining the type of load is essential as it effects the calculation of structural analysis to a great extent. Superclass blo:LoadType describes such loading type and consists of three subclasses which are: blo:DistributedLoad, blo:PointLoad and blo:UniformlyDistributedLoad. Load class and load type class are interrelated through object property blo:hasLoadType. Attribute (data property) blo:LoadingType specifies the exact loading type the through data type xsd:string. . Cardinality restriction was provided to class blo:LoadType that it can contain **max** one data value for loading type as each load can be categorized as one specific load type.

Certain structural members are used specifically as a compression or tension member and direction of loads on these members directly impacts the strength of structure. In order to specify direction of loads another Superclass blo:LoadingDirection is defined. Object property blo:hasLoadingDirection related class blo:Loads to class blo:LoadingDirection and data property blo:LoadingDirection (data type – xsd:string) provides specific direction to a load. Cardinality restriction was provided to class blo:LoadingDirection that it can contain **some** data value for loading direction as some loads can act in more than one direction at same time like snow load, live load etc. Figure 53 represents class hierarchy and data properties of BLO.

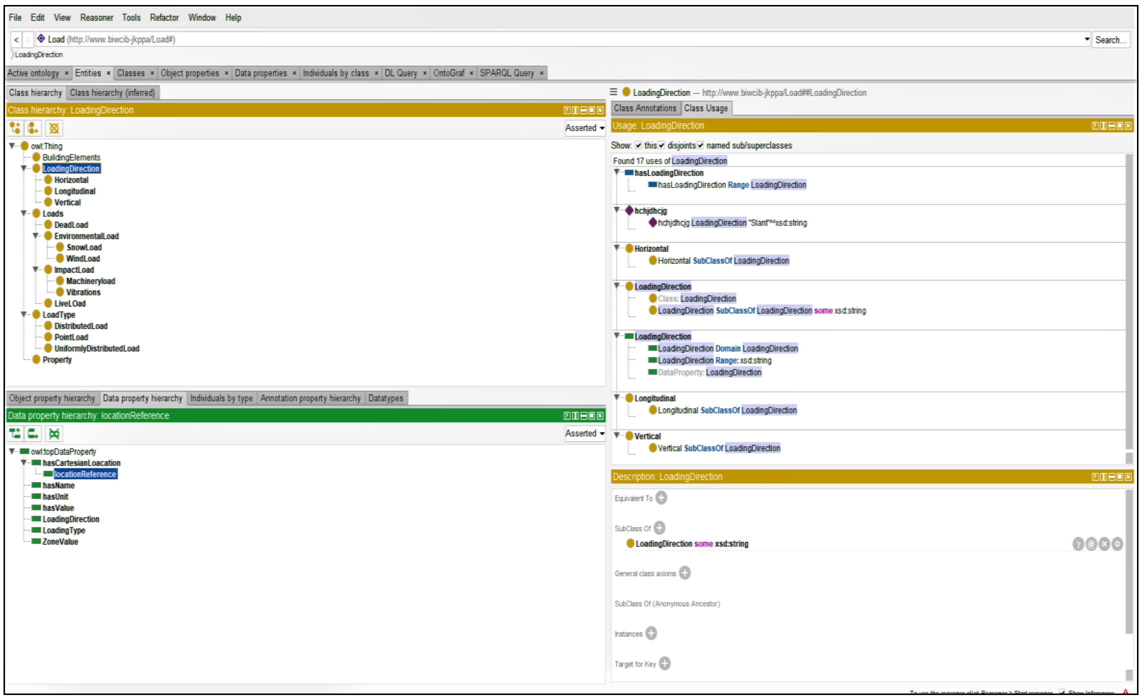

Class hierarchy: LoadingDirection
owl:Thing
BuildingElements
LoadingDirection
Horizontal
Longitudinal
Vertical
Loads
DeadLoad
EnvironmentalLoad
SnowLoad
WindLoad
ImpactLoad
Machineryload
Vibrations
LiveLoad
LoadType
DistributedLoad
PointLoad
UniformlyDistributedLoad
Property
Data property hierarchy: locationReference
owl:topDataProperty
hasCartesianLoacation
locationReference
hasName
hasUnit
hasValue
LoadingDirection
LoadingType
ZoneValue
Usage: LoadingDirection
Found 17 uses of LoadingDirection
hasLoadingDirection Range LoadingDirection
hchjdhcjg LoadingDirection "Slant"^^xsd:string
Horizontal SubClassOf LoadingDirection
LoadingDirection SubClassOf LoadingDirection some xsd:string
LoadingDirection Domain LoadingDirection
LoadingDirection Range: xsd:string
Longitudinal SubClassOf LoadingDirection
Vertical SubClassOf LoadingDirection
Description: LoadingDirection
SubClass Of
LoadingDirection some xsd:string

Figure 53: Class hierarchy and data properties of BLO

Reasoning and inference: Like BEO, reasoner was used for BLO to check the consistency of ontology and to perform reasoning. Five individuals were created namely: 0000L1, 0000L2, 0000L3, 0000L4 and 0000L5 and certain details were provided for each individual like:

- Individual 0000L1 has object property **blo:hasLoads** (Individual 100002)
- Individual 0000L2 has data property **blo:hasCartesianLoacation** (x=25, y=26, z=4) data type - xsd:string
- Individual 0000L3 has data property **blo:hasLoadingDirection** (horizontal) data type - xsd:string
- Individual 0000L4 has data property **blo:ZoneValue** (4) data type - xsd:integer
- Individual 0000L5 has data property **blo:LoadingType** (Point load) data type - xsd:string

Reasoner inferred on the basis of provided knowledge and concept that individual 0000L1 is an instance of class beo:BuildingElements, individual 0000L2 belongs to class blo:Loads, individual 0000L3 fits for class blo:LoadingDirection. Instance 0000L4 belongs to blo:EnvironmentalLoad while individual 0000L5 is an instance of blo:LoadType.

3.2.5 Building Element Representation Ontology (BERO)

Representation of building elements is an important and crucial process for structural analysis domain. Building elements can be geometrically represented or topologically represented and these representations data are essential in order to determine boundary conditions and to evaluate loading effect on structural members. Representations provide information related to basic structure, curve style, edge style, number of layers, placement direction and location etc. of a structural member. Most of this information is contained in architectural model, while some specifications like assignment of boundary conditions, structural member layouts etc are defined by structural engineers. Final, after assigning all these details the model is prepared for structural analysis hence, this process requires meaningful data exchange between several domains repeatedly as building models are continuously adjusted by each domain after corrections.

BERO represent the knowledge related to object placement and product representation (geometrically and topologically). These concepts are defined in IFC schemas explicitly for structural curve member and structural surface member but in representation ontology a combined approach is defined where members can be identified through their representation. BERO also includes concepts from Building Element Ontology (BEO) and Building Load Ontology (BLO) as all these ontologies are linked together and represent knowledge of structural analysis. Basic structure of BERO is depicted through Figure 54.

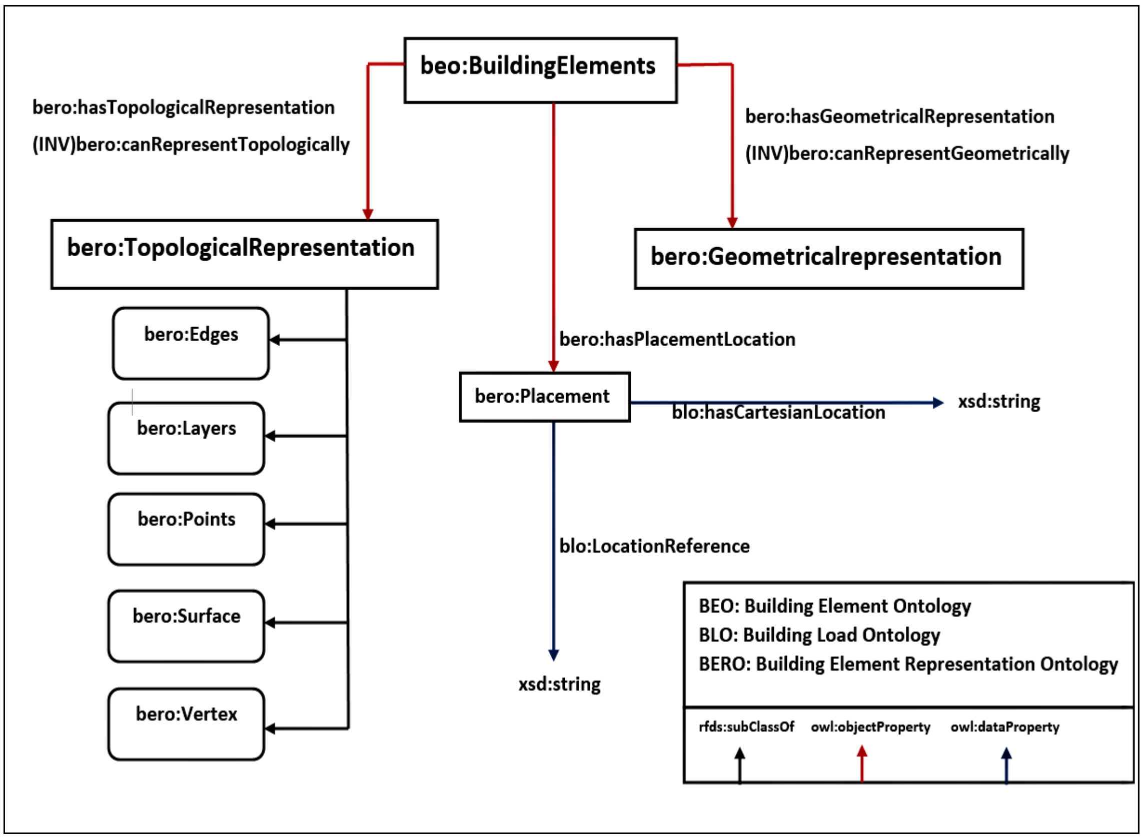

beo:BuildingElements
bero:hasTopologicalRepresentation
(INV)bero:canRepresentTopologically
bero:hasGeometricalRepresentation
(INV)bero:canRepresentGeometrically
bero:TopologicalRepresentation
bero:Geometricalrepresentation
bero:Edges
bero:Layers
bero:Points
bero:Surface
bero:Vertex
bero:hasPlacementLocation
bero:Placement
blo:hasCartesianLocation
xsd:string
blo:LocationReference
xsd:string
BEO: Building Element Ontology
BLO: Building Load Ontology
BERO: Building Element Representation Ontology
rfds:subClassOf
owl:objectProperty
owl:dataProperty

Figure 54: Overview of Building Element Representation Ontology

Building element and placement: Accurate placement and orientation of building elements in a structure increases the strength and life of a structure and provides proper alignment to the structure. Information regarding structural element placement is vital in order to analyse building performance and structural stability. IFC data defines the placement of structural members through entity *IfcObjectPlacement* and its attributes. In BERO relationship between class beo:BuildingElement and class bero:Placement is defined using object property bero:hasPlacementLocation. Attribute blo:hasCartesianLocation (data type: xsd:string) provides location of an object in coordinate system. Location of an object can be defined with respect to global coordinate system or relative to reference axis of structure which can be identified by another data property blo:LocationReference. Further attribute beo:Name provides identification to structural member for which location information is provided. Through restriction axiom it was specified that each structural member can have maximum one data value "Name" so that clear identification for every structural member can be provided. Interconnection between building element class and placement class is shown in Figure 55 in graphical form.

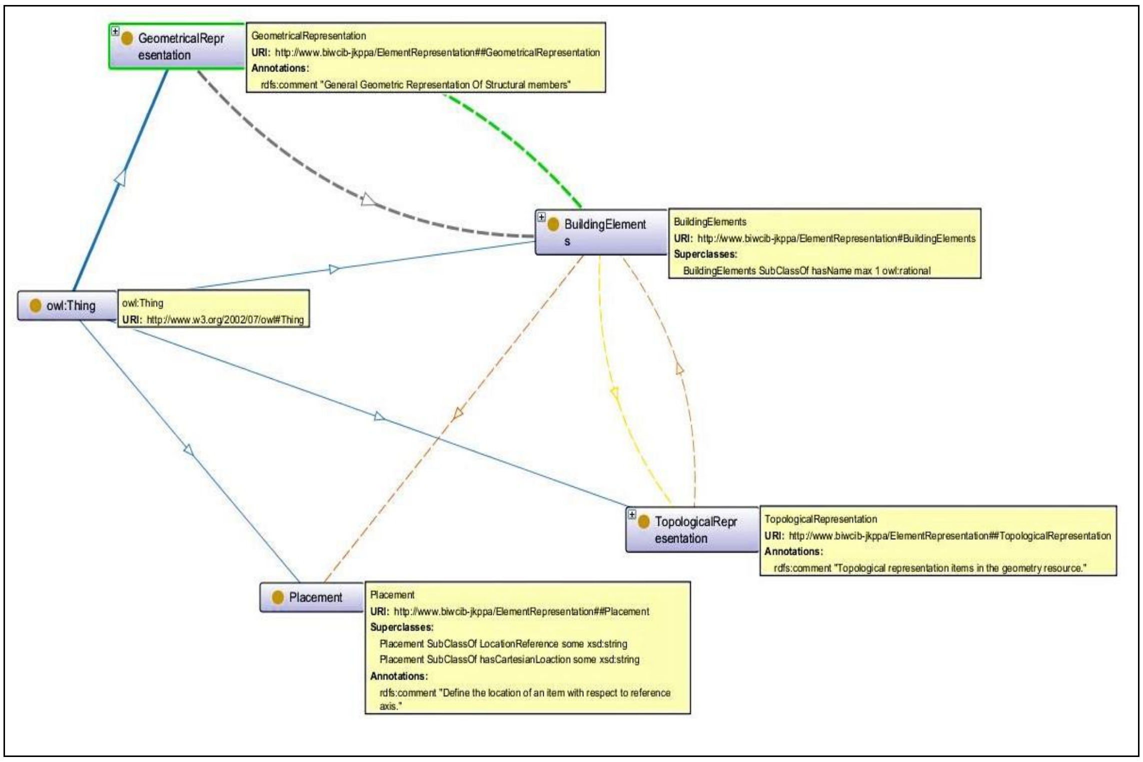

GeometricalRepresentation
GeometricalRepresentation
URI: http://www.biwcib-jkppa/ElementRepresentation##GeometricalRepresentation
Annotations:
rdfs:comment "General Geometric Representation Of Structural members"
BuildingElements
BuildingElements
URI: http://www.biwcib-jkppa/ElementRepresentation#BuildingElements
Superclasses:
BuildingElements SubClassOf hasName max 1 owl:rational
owl:Thing
owl:Thing
URI: http://www.w3.org/2002/07/owl#Thing
TopologicalRepresentation
TopologicalRepresentation
URI: http://www.biwcib-jkppa/ElementRepresentation##TopologicalRepresentation
Annotations:
rdfs:comment "Topological representation items in the geometry resource."
Placement
Placement
URI: http://www.biwcib-jkppa/ElementRepresentation##Placement
Superclasses:
Placement SubClassOf LocationReference some xsd:string
Placement SubClassOf hasCartesianLoaction some xsd:string
Annotations:
rdfs:comment "Define the location of an item with respect to reference axis."

Figure 55: BERO super classes and relationship graph

Geometrical representation: Geometrical information of structural member carries data related to geometry. Due to detailed and complicated data structure of geometrical information, inconsistency and misinterpretation is commonly noticed during data exchange. Geometrical data carries very less semantic information hence, special detailing is required to represent geometric information of structural members. IFC data schemas provide a list of schemas for geometric representation of objects and these schemas (*IfcCurve*, *IfcVector*, *IfcDirection* etc.) are subclasses of *IfcGeometricRepresentationItems*.

Object property bero:hasGeometricalRepresentation and inverse object property bero:canRepresentGeometrically establish the relationship between class beo:buildingElement and class bero:GeometricalRepresentation (Figure 54). Geometrical representation class defines the location of represented element or part of element through attribute blo:hasCartesianLocation and blo:LocationOfReference. Subclass bero:Vector describes several vectors that is associated with member to describe shape and orientation of member. Vector carries a direction through data property bero:hasDirectionRatio (data type: xsd:string) and magnitude of vector using another data property bero:hasDimension (data type - xsd:double). Direction ratio is ratio of component of an object in all three direction. Collection and representation of all vectors defines the orientation of an object in three-dimensional space. Another subclass bero:Curve represents an arc length greater than zero on the member. Curve is defined by length of curve and thickness of curve (Figure 56). BERO represents the curve on the structural member by three data properties bero:hasDimension for arc length, bero:hasThickness for arc thickness and beo:hasUnit to provide unit of both measurement (length and thickness). This was ensured by providing restrictions to class. It was specified that for identification of curve on a structural member minimum one arch length value and one thickness value must be provided by attributes.

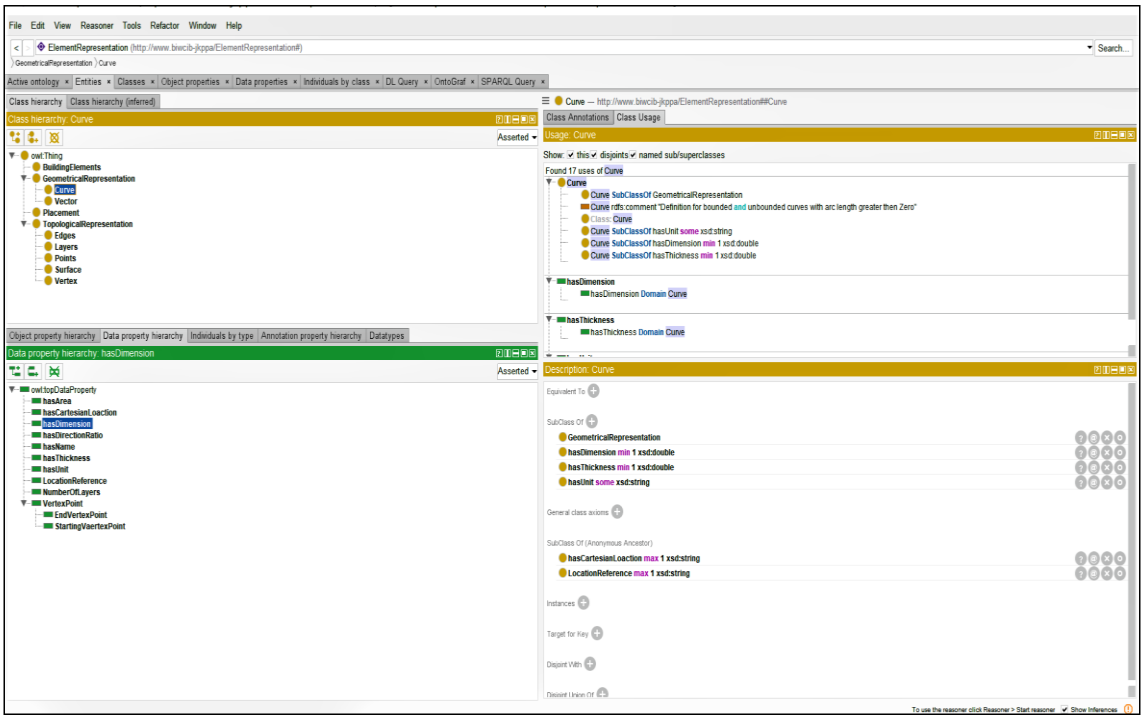
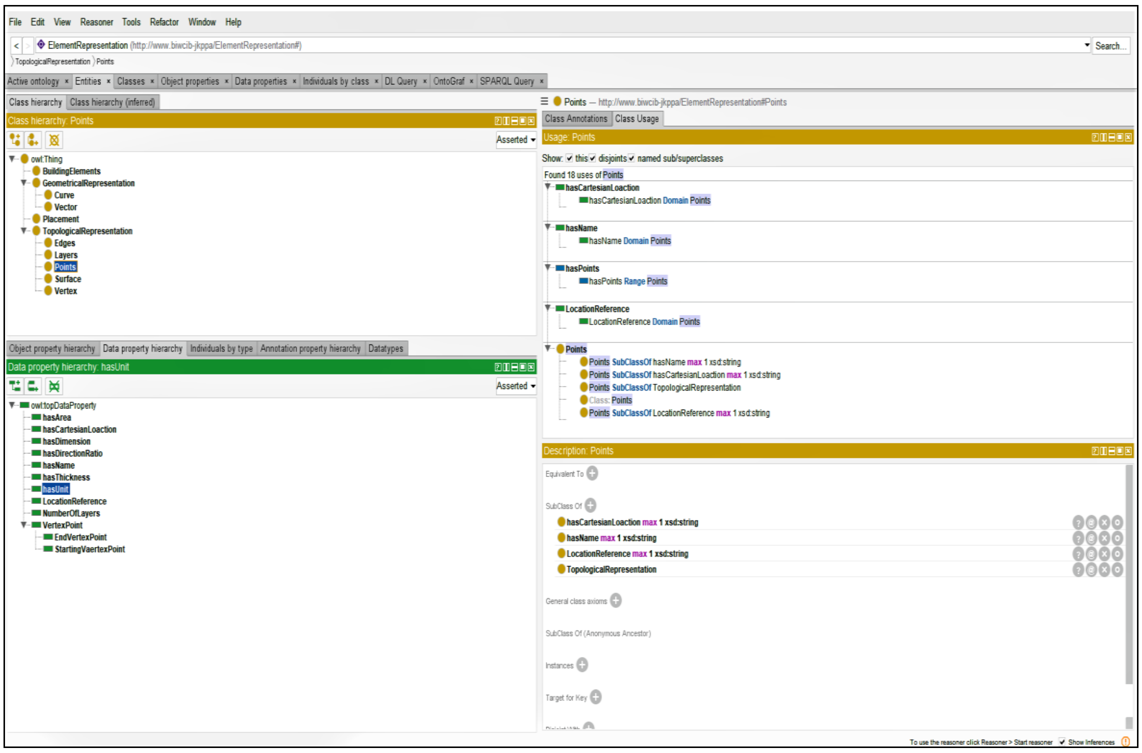

```
File  Edit  View  Reasoner  Tools  Refactor  Window  Help
ElementRepresentation (http://www.biwcib-jkppa/ElementRepresentation#)
GeometricalRepresentation > Curve
Active ontology | Entities | Classes | Object properties | Data properties | Individuals by class | DL Query | OntoGraf | SPARQL Query

Class hierarchy | Class hierarchy (inferred)
Class hierarchy: Curve                                   Asserted
owl:Thing
  BuildingElements
  GeometricalRepresentation
    Curve
    Vector
  Placement
  TopologicalRepresentation
    Edges
    Layers
    Points
    Surface
    Vertex

Object property hierarchy | Data property hierarchy | Individuals by type | Annotation property hierarchy | Datatypes
Data property hierarchy: hasDimension                    Asserted
owl:topDataProperty
  hasArea
  hasCartesianLoaction
  hasDimension
  hasDirectionRatio
  hasName
  hasThickness
  hasUnit
  LocationReference
  NumberOfLayers
  VertexPoint
    EndVertexPoint
    StartingVaertexPoint

Curve — http://www.biwcib-jkppa/ElementRepresentation##Curve
Class Annotations | Class Usage
Usage: Curve
Show: ✓ this ✓ disjoints ✓ named sub/superclasses
Found 17 uses of Curve
Curve
  Curve SubClassOf GeometricalRepresentation
  Curve rdfs:comment "Definition for bounded and unbounded curves with arc length greater then Zero"
  Class: Curve
  Curve SubClassOf hasUnit some xsd:string
  Curve SubClassOf hasDimension min 1 xsd:double
  Curve SubClassOf hasThickness min 1 xsd:double
hasDimension
  hasDimension Domain Curve
hasThickness
  hasThickness Domain Curve

Description: Curve
Equivalent To
SubClass Of
  GeometricalRepresentation
  hasDimension min 1 xsd:double
  hasThickness min 1 xsd:double
  hasUnit some xsd:string
General class axioms
SubClass Of (Anonymous Ancestor)
  hasCartesianLoaction max 1 xsd:string
  LocationReference max 1 xsd:string
Instances
Target for Key
Disjoint With
Disjoint Union Of

To use the reasoner click Reasoner > Start reasoner   ✓ Show Inferences
```

Figure 56: class hierarchy and "Curve" class description

Topological representation: Topological representation is representation of any object that provides the concepts in context to other representation like geometrical representation. *IfcTopologyRepresentation* is superclass for all the topology representation related entities. For curve members topology representation is provided using entity *IfcEdge*, *IfcVertex* etc. while for surface member IFC model uses entities like *IfcFace*, *IfcFaceBound* etc. Representation ontology describes the connection between class beo:BuildingElement and class bero:TopologicalRepresentation using object property bero:hasTopologicalRepresentation. This relationship is also represented by inverse object property bero:canRepresentTopologically (Figure 54). Topological representation

Class contains five subclasses which are bero:Edges, bero:Layers, bero:Points, bero:Surface, bero:Vertex. Surface class represent surface of a member like wall, slab. Surface of member is defined by area of surface provided by object property bero:hasArea and beo:hasUnit to provide unit of surface area. Surface contains a set of points and each points have a location in coordinate system with reference of surface. These point sets are used to provides set of vectors for describing orientation and alignment of members in geometrical representation. Data properties beo:hasName provides identification to each point, blo:hasCartesianLocation represent location of point on surface of member (Figure 57). Further, class bero:Layers define number of layers of a structural surface member via attribute bero:NumberOfLayers (data type – xsd:nonNegativeInteger).



```
ElementRepresentation (http://www.biwcib-jkppa/ElementRepresentation#)
TopologicalRepresentation 〉Points

Class hierarchy: Points
owl:Thing
    BuildingElements
    GeometricalRepresentation
        Curve
        Vector
    Placement
    TopologicalRepresentation
        Edges
        Layers
        Points
        Surface
        Vertex

Data property hierarchy: hasUnit
owl:topDataProperty
    hasArea
    hasCartesianLoaction
    hasDimension
    hasDirectionRatio
    hasName
    hasThickness
    hasUnit
    LocationReference
    NumberOfLayers
    VertexPoint
        EndVertexPoint
        StartingVaertexPoint

Points — http://www.biwcib-jkppa/ElementRepresentation#Points
Usage: Points
Show: ✓ this ✓ disjoints ✓ named sub/superclasses
Found 18 uses of Points
hasCartesianLoaction
    hasCartesianLoaction Domain Points
hasName
    hasName Domain Points
hasPoints
    hasPoints Range Points
LocationReference
    LocationReference Domain Points
Points
    Points SubClassOf hasName max 1 xsd:string
    Points SubClassOf hasCartesianLoaction max 1 xsd:string
    Points SubClassOf TopologicalRepresentation
    Class: Points
    Points SubClassOf LocationReference max 1 xsd:string

Description: Points
Equivalent To
SubClass Of
    hasCartesianLoaction max 1 xsd:string
    hasName max 1 xsd:string
    LocationReference max 1 xsd:string
    TopologicalRepresentation
General class axioms
SubClass Of (Anonymous Ancestor)
Instances
Target for Key

To use the reasoner click Reasoner > Start reasoner ✓ Show Inferences
```

Figure 57: description of class "Points"

Class bero:Edges defines the connection of two vertices of a curve member. Vertices should connect in default straight line pattern if no curve is provided. Edge class is domain of object property bero:hasVertexPoint , while range is class bero:Vertex. Vertex is defined by attribute bero:hasVertexPoint (data type – xsd:string) which provides location of vertex in cartesian location.

Like other ontologies, reasoner was used to check consistency of ontology and for reasoning of information by creating individuals for each class.

Structural Analysis Ontology (SAO): Load calculation and result of structural analysis process determines the stability and strength of a structure. Assignment of load combination and determination of different methods of structural analysis process is assigned by structural engineer and later this information is shared between different domains and stakeholders for proofing. Placement and orientation of structural members, material quality and model of structure is finalized on the basis of structural analysis results. Meaningful and uninterrupted data exchange of this information is necessary in order to reduce errors and human effort.

Analysis ontology (SAO) represent concepts related to formation of load combination, calculation of result using specific structural analysis method, imposing result on structure and structural member for failure check etc. An overview of SAO is shown in Figure 58.

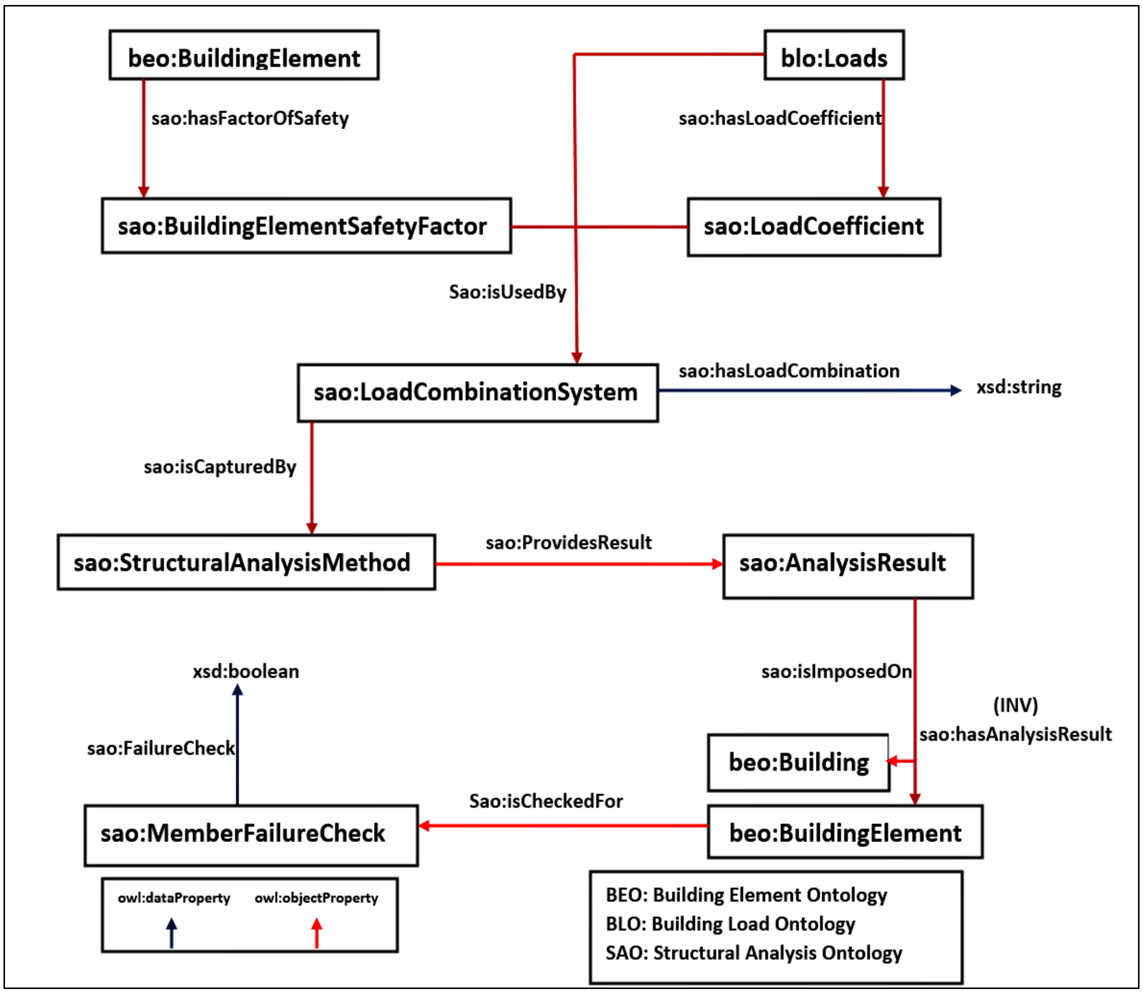


Figure 58: Overview of Structural Analysis Ontology

Load factor, coefficients and load combination: Load combination for structural analysis is created according to different needs, purposes and depends on many factors. There are different standard factors for load combination that depends on structural member material like steel member, concrete member etc. Then different load coefficients are determined which depends on type of loads (live load, dead load etc.). For environmental loads (snow load, wind load), standard load coefficients are taken depending on zone for load combinations. Further, load combinations also depend on type of structure (residential structure, offices). After, assigning all the factors and coefficients load combination are created for two basic design factors i.e., Serviceability limit state (SLS) and ultimate limit state (ULS).

IFC data model defines grouping of loads through entity *IfcStructuralLoadGroup*. Then loading factors are assigned to load cases by entity *IfcRelAssignsToGroupByFactor* to convert load cases into load combinations. SAO defines this concept using three relationships (Figure 58). Class blo:Loads provides type of loads and value of each load along with unit and class beo:BuildingElements represents concepts of structural member and their properties. Safety factor and load coefficient values are provided by subclasses of sao:BuildingElementSafetyFactor and class sao:Loadcoefficient respectively (Figure 59).

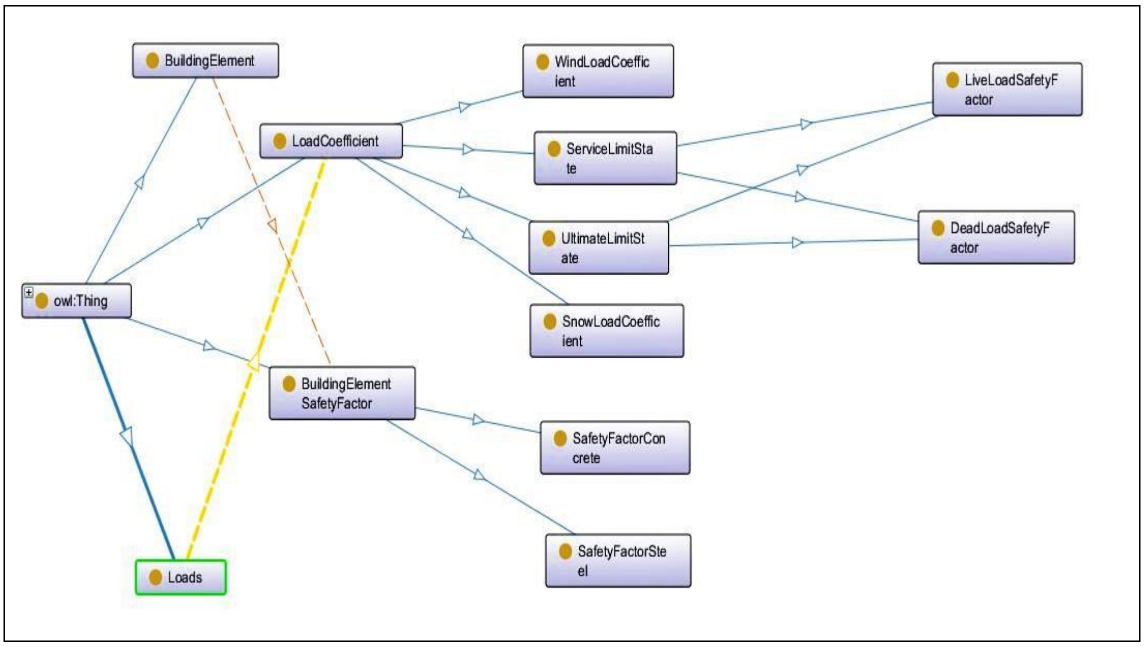


Figure 59: Graphical representation of class hierarchy for "Load coefficient" class and "Building element safety factor" class

Subclasses of Load coefficient class and Building element safety factor class contains attributes to provide load coefficient factor (data type – xsd:double) and safety factor (data type – xsd:double). These values are used to form load combinations using load combination systems. This concept is represented in SAO using object property sao:isUsedBy, which connects class blo:Loads, class sao:BuildingElementSafetyFactor and class sao:LoadCoefficient to class sao:LoadCombinationSystem. Load combination system has two classes sao:ServiceLoadCombination and sao:UltimateLoadCombination. Data property sao:hasLoadCombination (data type – xsd:string) represents load combinations information collected by SAO. There is no restrictions on number of load combinations information that can be provided by Load combination system class.

Analysis method and result: Ifc entity *IfcStructuralAnalysisModel* is used for load calculation by collecting all the load combinations from entity *IfcStructuralLoadGroup*. Calculated results are then grouped by *IfcStructuralResultGroup* entity. Analysis ontology represent this concept using object property sao:isCapturedBy, which connects class load combination system to class Structural analysis method (Figure 58). As shown in Figure 60, class sao:StructuralAnalysisMethod defines different methods of structural analysis like static linear analysis, buckling analysis etc. and provides analysis result to class sao:AnalysisResult. Relationship between class Structural analysis and class Analysis result is established through object property sao:providesResult.

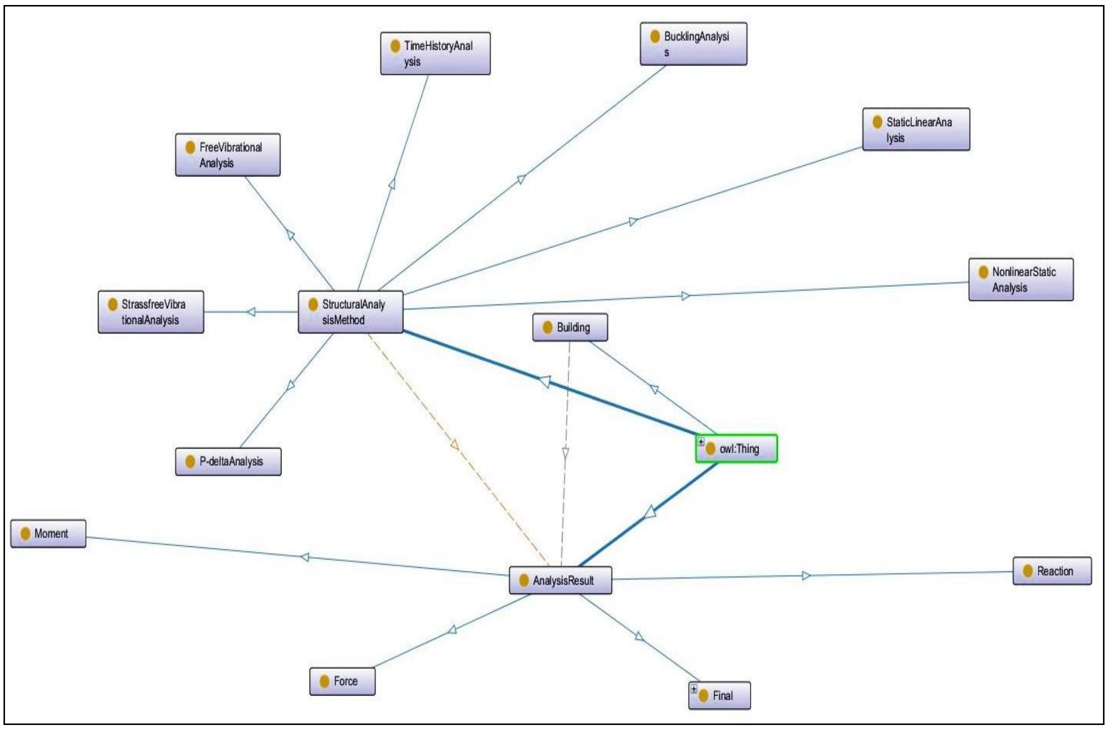


Figure 60: class hierarchy of "Structural analysis" class and "Analysis result" class

Subclasses of Analysis result class i.e., sao:Force, sao:Moment, sao:Reaction and sao:Final represent different type of reactions results provided by structural analysis method. Each result class have three attributes to provide value for results which are:

- beo:hasName (xsd:string) – to identify the type of reaction i.e., moment, force etc.
- beo:hasValue (xsd:double) – to provide value of reaction.
- beo:hasUnit (xsd:string) – to provide unit of reaction i.e., KN/mm^2, KN/m^2 etc.

Imposition and failure check: Analysis results are imposed on structure and structural member to check the effect of reactions. If imposed value is more than the resistance capacity of structural member then failure of member failure is considered and changes are made in building model. SAO represent knowledge of imposition of results on structural member and structure by object property sao:isImposedOn, which links Analysis results class to Building and Building element class. Result validation concept is described by class sao:MemberFailureCheck. This class contains attributes sao:hasResistancevalue which provides details of resistance value of a structural member and imposed value is defined by result class. Data property sao:FailureCheck (data type – xsd:boolean) depicts result as True or False.

For validation of constructed ontology for knowledge representation and data exchange a frame model was designed and structural analysis was performed using standard load cases according to Eurocode specifications.

4 Construction of knowledge base

To check the applicability and usability of developed ontologies for semantic interoperability and for denoting the data from architectural domain and structural analysis domain, a three-storey composite structure, modelled in Revit was taken [21]. The structure consists of slab, beam, column and foundation. The overall dimension of the structure is 10600 mm in length and 5470 mm in breadth (taken from centre to centre of exterior column) with overall height of 8600mm. Structure consist both concrete and steel members (Case Study, section - 2.2.4). The model was exported to IFC file format using Design Transfer View (DTV) as Model View Definition and then imported to RSTAB – frame analysis tool using IFC 2×3 coordination view.

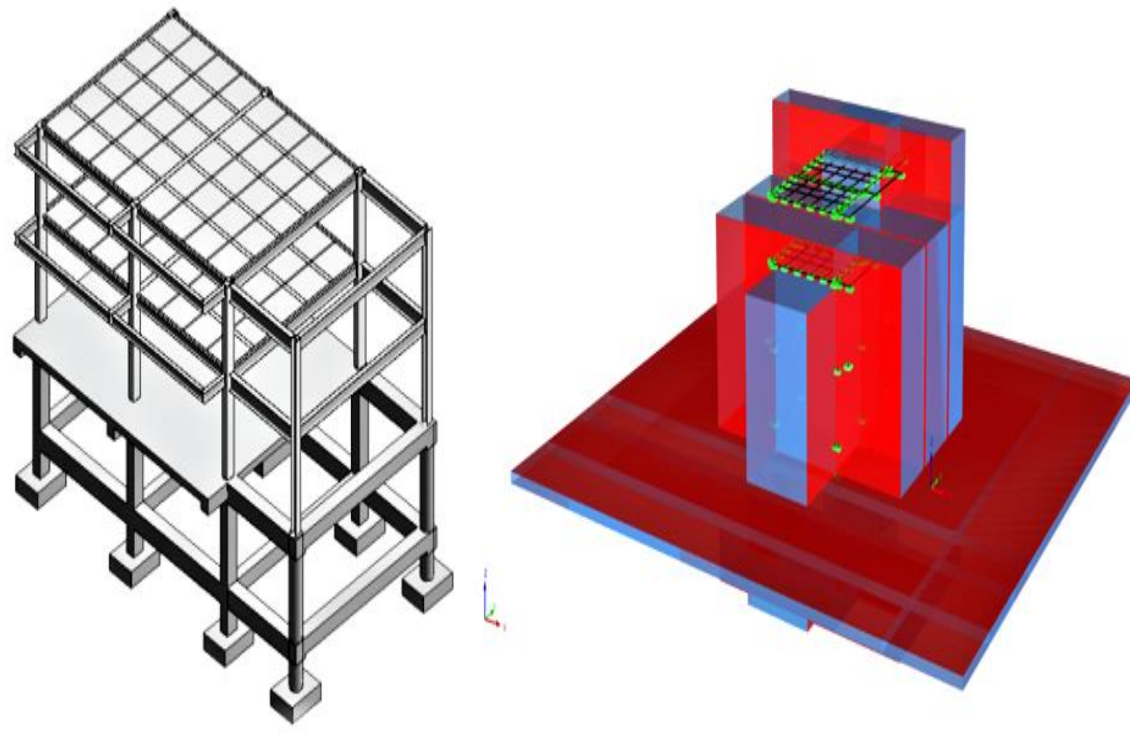


Figure 61: Left: architectural model in Revit and Right: structural analysis model in RSTAB

Geometric inconsistency and data loss was noticed for model imported to structural analysis tool. Several material data like poison's ratio, modulus of elasticity etc. was missing and then assigned manually. Moreover, structure did not had stability in vertical direction due to misinterpretation of geometric data. Fixed supports were assigned at several nodes to provide stability to the structure. Standard loads like dead load, imposed load, wind load etc. were assigned on several members and geometric linear analysis method was adopted for structural analysis.

Later, the structural analysis model was exported to IFC data model using IFC 2×3 coordination view as RSTAB does not support export of file for IFC 4 data schemas. Restored IFC file was converted to RDF file format using a JAVA based conversion tool called IFCtoRDF [40].

Protocol and RDF query language (SPARQL) was used to query information from RDF file which contain both architectural and structural analysis data using GraphDB tool. GraphDB is semantic graph database and enables users to query information from RDF based files using SPARQL. The results are presented in tables and can be downloaded in various syntax or can be visualized as graph. It is a tool for data integration and data relationship exploration.

Protocol and RDF query language (SPARQL) has certain important clauses like "Where", "Select", "Construct" that is used to query specific information and to substitute queried information into already existing templets to generate new statements.

Select: Select clause identifies the variables to appear in the query results.

Where: Where clause provides the basic graph pattern to match against the data graph

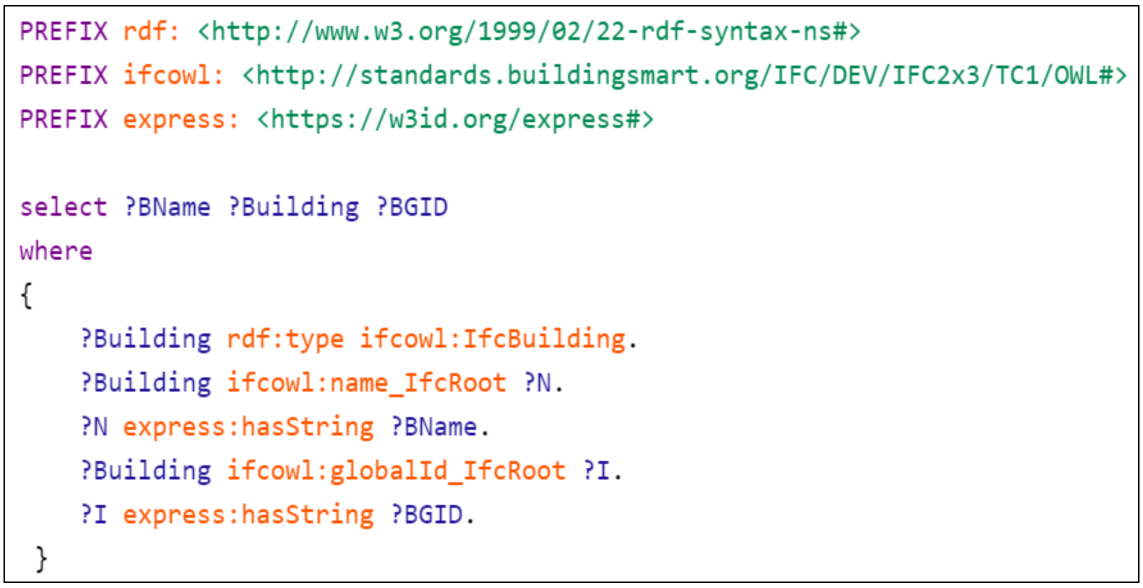
Construct: Construct clause is used to substitute the queried information given by clause "select" into a model or templet that already exist. The output graph from the Construct template is formed from just two of the solutions from graph pattern matching.

Using these SPARQL clause several information was queried and further constructed in already existing ontologies.

4.1 Query of architectural information

Ontologies were developed in order to achieve semantic interoperability between architectural domain and structural analysis domain. To check the applicability of developed ontologies architectural information like building data, site data, building element data was queried from RDF based IFC file which contains both architectural and structural analysis data.

4.1.1 IfcBuilding, IfcBuildingStorey and IfcSite information



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX express: <https://w3id.org/express#>

select ?BName ?Building ?BGID
where
{
    ?Building rdf:type ifcowl:IfcBuilding.
    ?Building ifcowl:name_IfcRoot ?N.
    ?N express:hasString ?BName.
    ?Building ifcowl:globalId_IfcRoot ?I.
    ?I express:hasString ?BGID.
 }
```

Figure 62: Query of information for *IfcBuilding* entity using SPARQL

Figure 62 depicts the SPARQL query for extracting information from *IfcBuilding* entity. IfcBuilding describes the building or structure for a construction project. Name of the building (BName) is provided by entity *name_IfcRoot* and global ID (BGID) is provided by *globalId_IfcRoot*. All queries are formed as RDF triples and required information is mentioned in "select" clause. Results are provided in tabular form or can be seen as graphs. Figure 63 represents the queried result from entity *IfcBuilding* in graphical form.

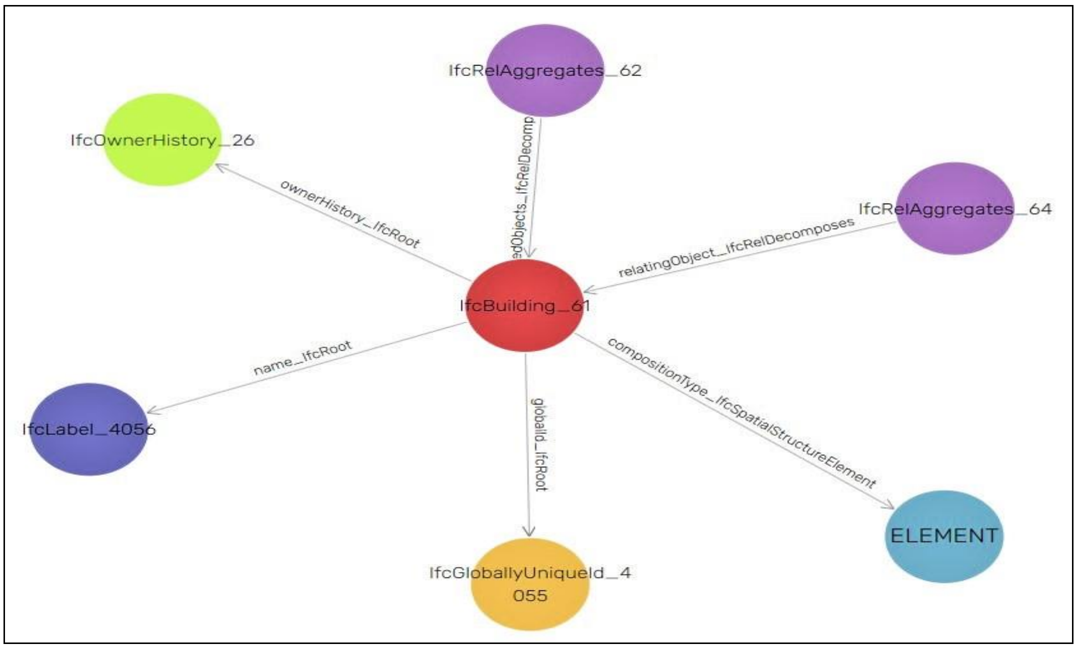
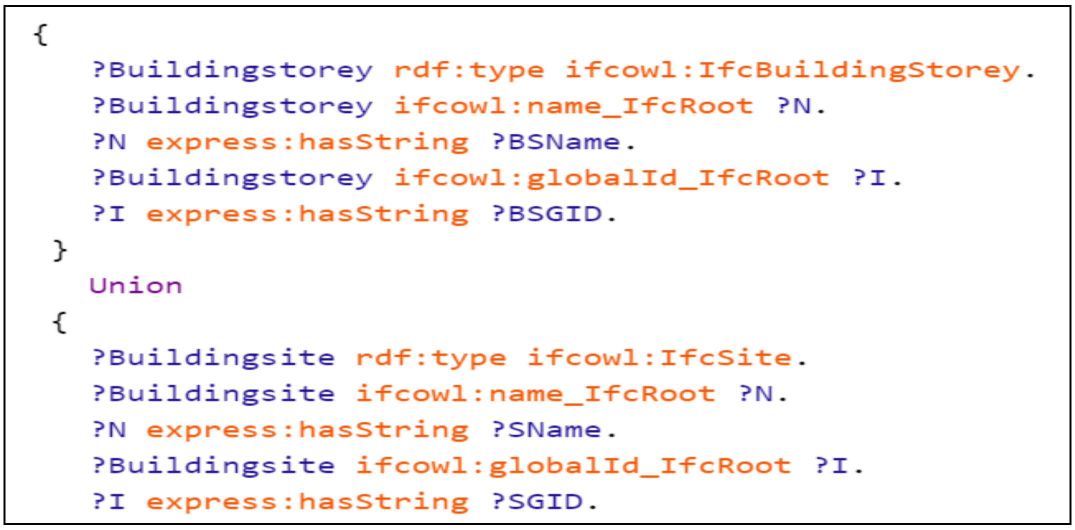


Figure 63: Graphical representation of query results

Similarly, site and building storey information was queried. Information of different objects (Building storey and site) can be queried simultaneously using "Union" as shown in Figure 64. Table 4 summarized the query results.



```
{
    ?Buildingstorey rdf:type ifcowl:IfcBuildingStorey.
    ?Buildingstorey ifcowl:name_IfcRoot ?N.
    ?N express:hasString ?BSName.
    ?Buildingstorey ifcowl:globalId_IfcRoot ?I.
    ?I express:hasString ?BSGID.
 }
    Union
 {
    ?Buildingsite rdf:type ifcowl:IfcSite.
    ?Buildingsite ifcowl:name_IfcRoot ?N.
    ?N express:hasString ?SName.
    ?Buildingsite ifcowl:globalId_IfcRoot ?I.
    ?I express:hasString ?SGID.
```

Figure 64: Query of information for IfcBuildingStorey and IfcSite entity using SPARQL

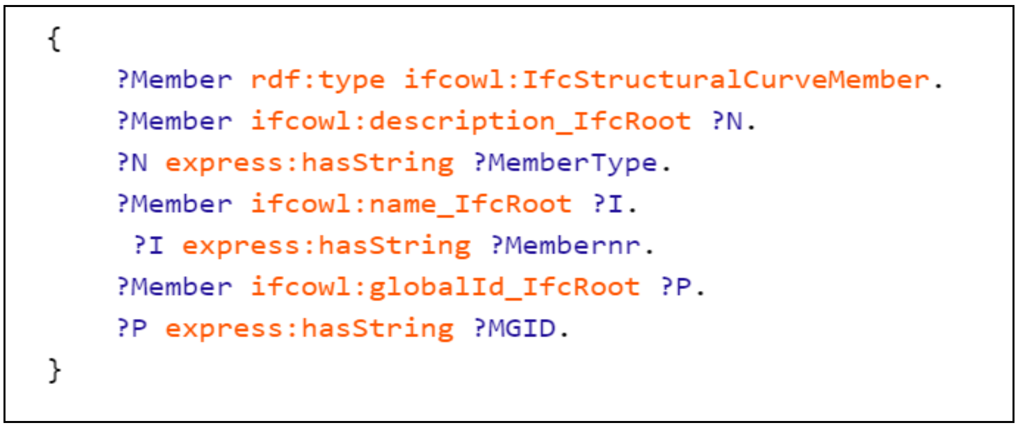
Table 4: query results for Building, Site and Building storey

IFC Entity	Global ID	Name	Individual
IfcBuilding	LwzJrb67GTmpYgXKPRO3Gu	Building	IfcBuilding_61
IfcBuildingStorey	2run6VGSsoOS7FUmI2x2l6	Building Storey	IfcBuildingStorey_63
IfcSite	TVb54j_VRr4uYUXlh_tdwf	Site	IfcSite_59

4.1.2 Structural members and properties

IFC structural analysis domain represent structural member such as beam, column through entity *IfcStructuralMember* while each structural element is represented explicitly by *IfcBuildingElemnt* entity such as *IfcBeam*, *IfcColumn* structured in IFC architectural domain. Hence, IFC based structural analysis model do not contain special class for each member type but rather represented by two general entity: *IfcStructuralCurveMember* and *IfcStructuralSurfaceMember* whereas, IFC based structural model contains special classes for each type of structural member. Structural model used for query of information consist slab, foundation, column and beam as structural member. Since RSTAB is a frame analysis tool, slab information was not mapped into RSTAB internal data schema when model was imported to RSTAB from IFC data model. As a result, only *IfcStructuralCurveMember* entity was found in final IFC data model which consist beam and foundation because analysis software represents beam and column as beam member.

As shown in Figure 65, information for member unique global ID (MGID), member type description (provided by architectural tool) and member number in structure (provided by analysis software) was queried using SPARQL query language.



```
{
    ?Member rdf:type ifcowl:IfcStructuralCurveMember.
    ?Member ifcowl:description_IfcRoot ?N.
    ?N express:hasString ?MemberType.
    ?Member ifcowl:name_IfcRoot ?I.
     ?I express:hasString ?Membernr.
    ?Member ifcowl:globalId_IfcRoot ?P.
    ?P express:hasString ?MGID.
}
```

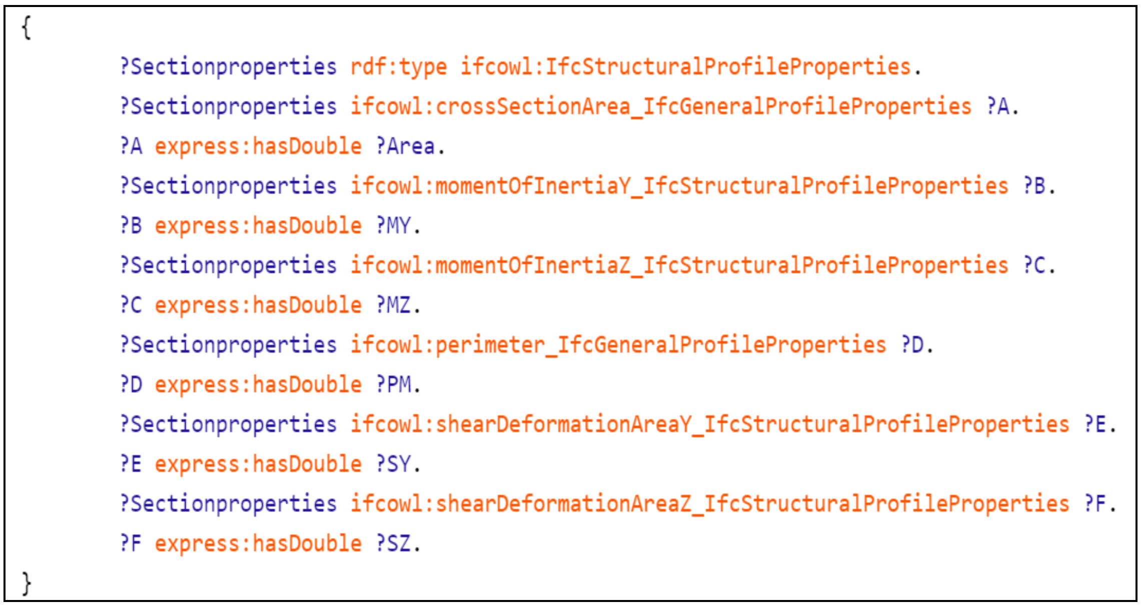
Figure 65: SPARQL query for IfcStructuralCurveMember information

Query result presented information of 83 structural members out of which 7 instances of IfcFooting and 76 instances of IfcBeam were found. Result for randomly selected four structural members are presented in Table 5.

Table 5: query results for structural members

IFC Entity	Global ID	Member Number	Individual
IfcFooting	"B96tcJd_ZBj6oxzAT07BNt"	1	IfcStructuralCurveMember_2751
IfcFooting	dukcvDtIDuyKlB_9J61oBR	5	IfcStructuralCurveMember_2827
IfcBeam	WUxr5aCUZYc2xTgRVkqPA1	18	IfcStructuralCurveMember_3074
IfcBeam	3DneFve0Y22sC5SdTq4Kgr	77	IfcStructuralCurveMember_4195

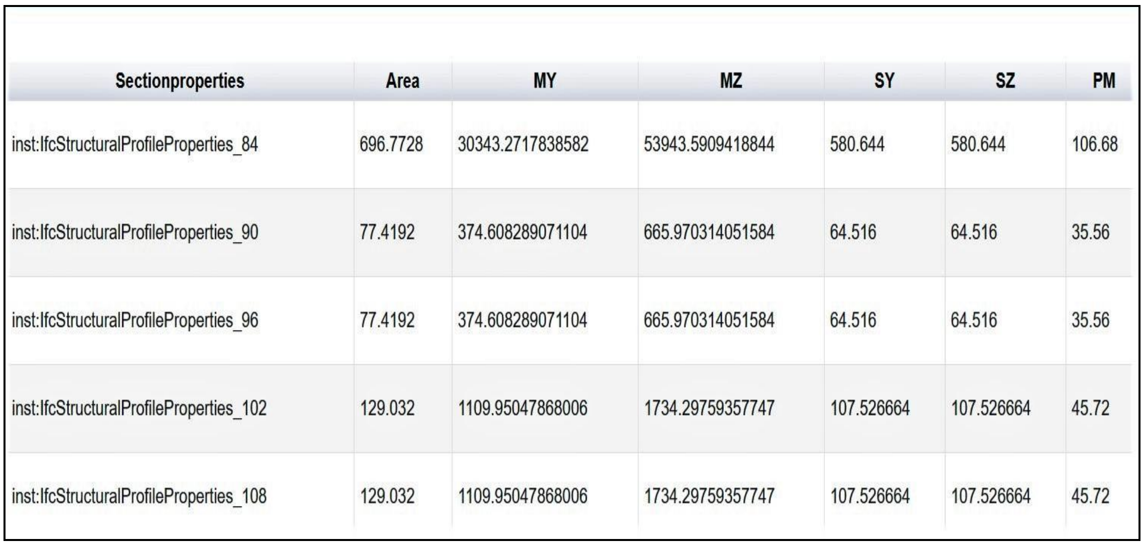
Structural members such as beam, column etc. are associated with section profile definitions and are used in designing of structural members. Section properties of structural curve members are listed in entity *IfcProfileDef* and *IfcProfileProperties.* Information were queried for different properties enlisted under section properties like moment of inertia, shear deformation area etc.



```
{
        ?Sectionproperties rdf:type ifcowl:IfcStructuralProfileProperties.
        ?Sectionproperties ifcowl:crossSectionArea_IfcGeneralProfileProperties ?A.
        ?A express:hasDouble ?Area.
        ?Sectionproperties ifcowl:momentOfInertiaY_IfcStructuralProfileProperties ?B.
        ?B express:hasDouble ?MY.
        ?Sectionproperties ifcowl:momentOfInertiaZ_IfcStructuralProfileProperties ?C.
        ?C express:hasDouble ?MZ.
        ?Sectionproperties ifcowl:perimeter_IfcGeneralProfileProperties ?D.
        ?D express:hasDouble ?PM.
        ?Sectionproperties ifcowl:shearDeformationAreaY_IfcStructuralProfileProperties ?E.
        ?E express:hasDouble ?SY.
        ?Sectionproperties ifcowl:shearDeformationAreaZ_IfcStructuralProfileProperties ?F.
        ?F express:hasDouble ?SZ.
}
```

Figure 66: SPARQL query to obtain values of different section property parameters

Figure 66 shows SPARQL query used to acquire values for section property parameters. **Area** indicates cross section area of structural member, **MY** will give value for moment of Inertia in y-direction whereas, **MZ** is for moment of Inertia in Z-direction. **SY** and **SZ** is short name assigned to shear deformation area in y-direction and in z-direction respectively and **PM** is measurement of perimeter of members. Result of the query gives information of 10 instances of *IfcStructuralProfileProperties* which indicates that 10 different types of material and cross-sections were taken for designing of structural members of modelled structure. Figure 66 shows the obtained section property results for five instances of *IfcStructuralProfileProperties*.

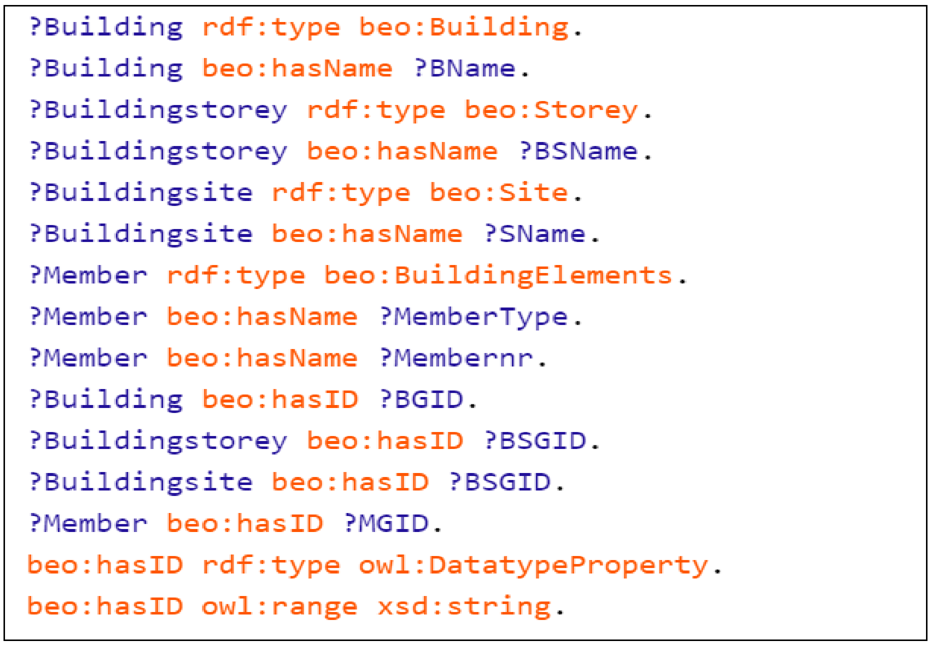


Sectionproperties	Area	MY	MZ	SY	SZ	PM
inst:IfcStructuralProfileProperties_84	696.7728	30343.2717838582	53943.5909418844	580.644	580.644	106.68
inst:IfcStructuralProfileProperties_90	77.4192	374.608289071104	665.970314051584	64.516	64.516	35.56
inst:IfcStructuralProfileProperties_96	77.4192	374.608289071104	665.970314051584	64.516	64.516	35.56
inst:IfcStructuralProfileProperties_102	129.032	1109.95047868006	1734.29759357747	107.526664	107.526664	45.72
inst:IfcStructuralProfileProperties_108	129.032	1109.95047868006	1734.29759357747	107.526664	107.526664	45.72

Figure 67: Results for section properties

4.2 Construction of query results into developed ontology

SPARQL allows substitution of results into already developed model or templet using “Construct” clause. Query results were constructed into Building Element Ontology (BEO). Figure 68 shows the query used to construct obtained results into BEO ontology.



```
?Building rdf:type beo:Building.
?Building beo:hasName ?BName.
?Buildingstorey rdf:type beo:Storey.
?Buildingstorey beo:hasName ?BSName.
?Buildingsite rdf:type beo:Site.
?Buildingsite beo:hasName ?SName.
?Member rdf:type beo:BuildingElements.
?Member beo:hasName ?MemberType.
?Member beo:hasName ?Membernr.
?Building beo:hasID ?BGID.
?Buildingstorey beo:hasID ?BSGID.
?Buildingsite beo:hasID ?BSGID.
?Member beo:hasID ?MGID.
beo:hasID rdf:type owl:DatatypeProperty.
beo:hasID owl:range xsd:string.
```

Figure 68: SPARQL query for construction of results in BEO ontology

Instance of building were substituted as instance of class beo:Building which has name (BName). BEO data property hasName provides name to instances of beo:Building.

Similarly, query result for building storey and building site were assigned to beo:Storey and beo:Site respectively. Class beo:BuildingElements contains all instances of *IfcStructuralCurveMember* (Member) along with attribute beo:hasName to provide name and member number to instances.

SPARQL also enables the user to construct new knowledge into templet. For example, BEO do not contain any data property to provide global Id value of instance. So, to represent value of global ID new data property beo:hasID is defined (Figure 68).

Result of "Construct" query contains set of triples with either subject or object as class of templet or model to which results are substituted. For example, IfcBuilding_61→type→Building (Figure 69) where Building is class beo:Building which belongs to BEO.

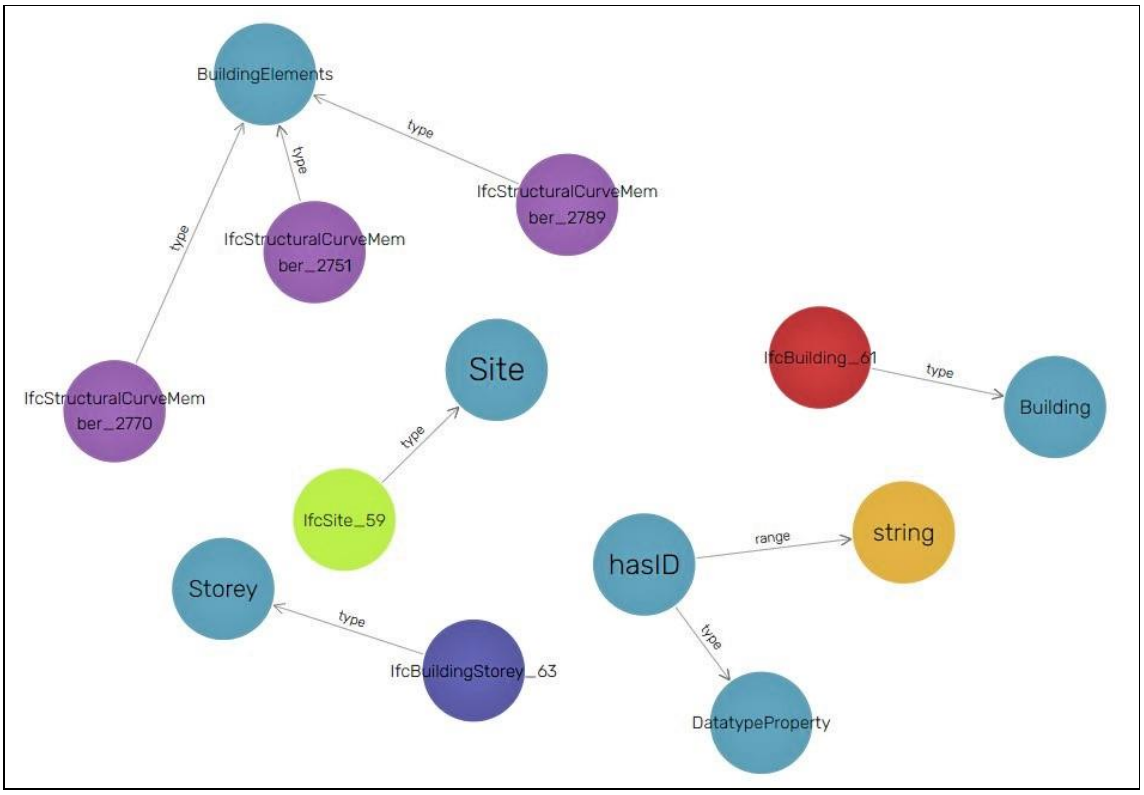
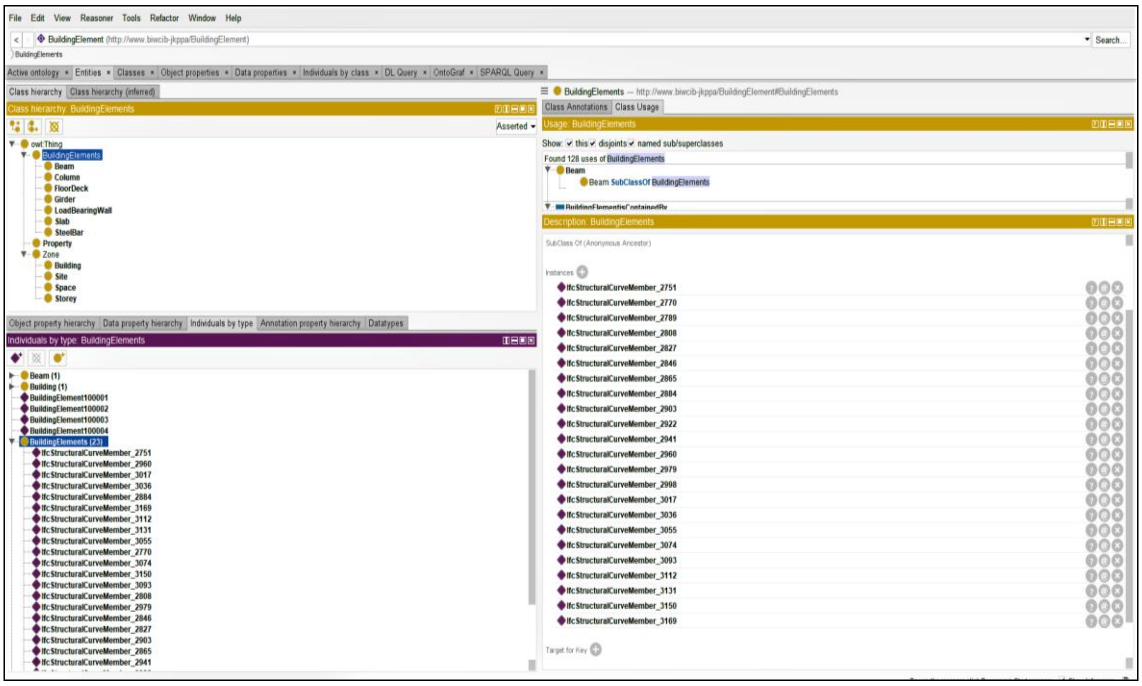


Figure 69: graphical representation of result of "Construct" query

Result was saved as another ontology (.ttl format) and then imported to Building Element Ontology. BEO was able to represent the instances of building, site, storey and building elements. Building elements query results gave information of 83 building elements but BEO was able to represent only 23 out of that (Figure 70). Represented instances carried name, member number and global ID correctly. Building members *IfcBeam*, *IfcFooting* were represented as member of class beo:BuildingElements and not class beo:Beam because structural analysis software do not categorized structural member as distinct beam or column type member but rather assigned them member numbers and IFC data model represented all structural members under common entity *IfcStructuralCurveMember*. Figure 70 illustrates 23 instances of class beo:BuildingElements substituted from IFC based structural analysis model. Architectural tools and structural analysis tools define certain data in different ways and therefore, most of the instances of *IfcBuildingElement* entity cannot be semantically transferred to ontology.


Class hierarchy: BuildingElements

- owl:Thing
  - BuildingElements
    - Beam
    - Column
    - FloorDeck
    - Girder
    - LoadBearingWall
    - Slab
    - SteelBar
  - Property
  - Zone
    - Building
    - Site
    - Space
    - Storey

Individuals by type: BuildingElements

- Beam (1)
- Building (1)
- BuildingElement100001
- BuildingElement100002
- BuildingElement100003
- BuildingElement100004
- BuildingElements (23)
  - IfcStructuralCurveMember_2751
  - IfcStructuralCurveMember_2960
  - IfcStructuralCurveMember_3017
  - IfcStructuralCurveMember_3036
  - IfcStructuralCurveMember_2884
  - IfcStructuralCurveMember_3169
  - IfcStructuralCurveMember_3112
  - IfcStructuralCurveMember_3131
  - IfcStructuralCurveMember_3055
  - IfcStructuralCurveMember_2770
  - IfcStructuralCurveMember_3074
  - IfcStructuralCurveMember_3150
  - IfcStructuralCurveMember_3093
  - IfcStructuralCurveMember_2808
  - IfcStructuralCurveMember_2979
  - IfcStructuralCurveMember_2846
  - IfcStructuralCurveMember_2827
  - IfcStructuralCurveMember_2903
  - IfcStructuralCurveMember_2865
  - IfcStructuralCurveMember_2941

BuildingElements — http://www.biwcib-jkppa/BuildingElement#BuildingElements

Usage: BuildingElements

Found 128 uses of BuildingElements

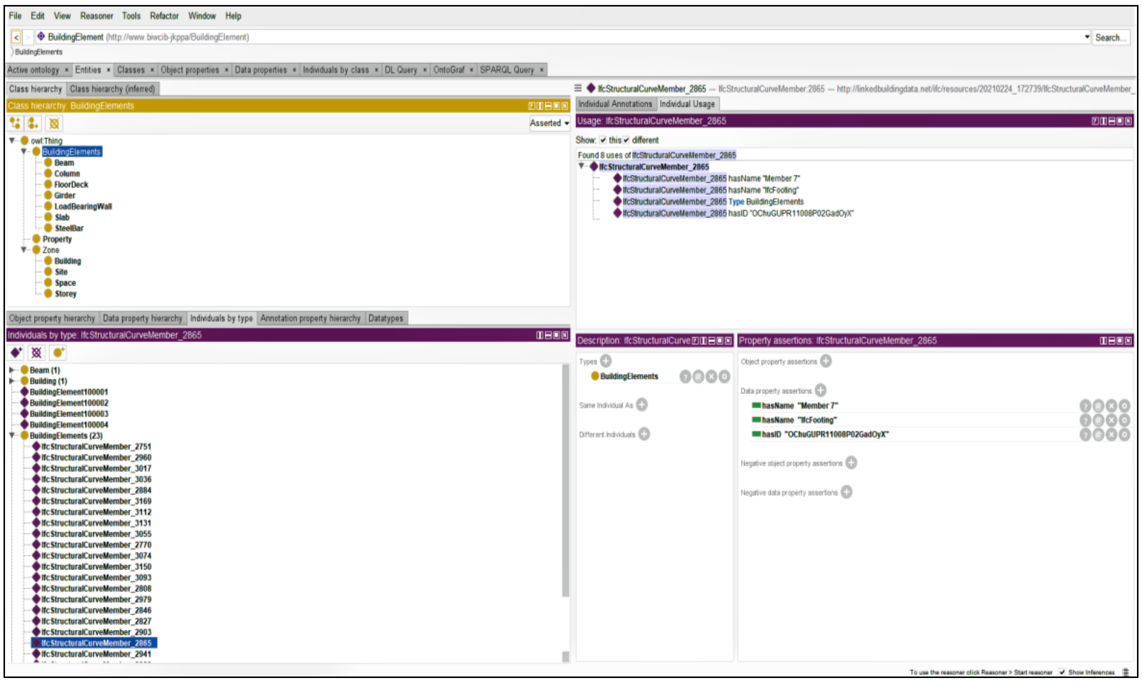
- Beam
  - Beam SubClassOf BuildingElements

Description: BuildingElements

Instances

- IfcStructuralCurveMember_2751
- IfcStructuralCurveMember_2770
- IfcStructuralCurveMember_2789
- IfcStructuralCurveMember_2808
- IfcStructuralCurveMember_2827
- IfcStructuralCurveMember_2846
- IfcStructuralCurveMember_2865
- IfcStructuralCurveMember_2884
- IfcStructuralCurveMember_2903
- IfcStructuralCurveMember_2922
- IfcStructuralCurveMember_2941
- IfcStructuralCurveMember_2960
- IfcStructuralCurveMember_2979
- IfcStructuralCurveMember_2998
- IfcStructuralCurveMember_3017
- IfcStructuralCurveMember_3036
- IfcStructuralCurveMember_3055
- IfcStructuralCurveMember_3074
- IfcStructuralCurveMember_3093
- IfcStructuralCurveMember_3112
- IfcStructuralCurveMember_3131
- IfcStructuralCurveMember_3150
- IfcStructuralCurveMember_3169

Figure 70: Instances of class beo:BuildingElements


IfcStructuralCurveMember_2865 — IfcStructuralCurveMember:2865 — http://linkedbuildingdata.net/ifc/resources/20210224_172739/IfcStructuralCurveMember_

Usage: IfcStructuralCurveMember_2865

Found 8 uses of IfcStructuralCurveMember_2865

- IfcStructuralCurveMember_2865
  - IfcStructuralCurveMember_2865 hasName "Member 7"
  - IfcStructuralCurveMember_2865 hasName "IfcFooting"
  - IfcStructuralCurveMember_2865 Type BuildingElements
  - IfcStructuralCurveMember_2865 hasID "OChuGUPR11008P02GadOyX"

Description: IfcStructuralCurve

Types: BuildingElements

Property assertions: IfcStructuralCurveMember_2865

Data property assertions

- hasName "Member 7"
- hasName "IfcFooting"
- hasID "OChuGUPR11008P02GadOyX"

Figure 71: Data values of instance IfcStructuralCurveMember_2865

5 Conclusions and Future Work

5.1 Conclusions

As per defined objectives of this thesis, the study has been conducted based on the researches and concepts available before and using those in current investigation. The main idea behind the development of ontologies and use of semantic web technologies for semantic interoperability was understood. Further, poor implementation of syntactic interoperability approaches has also influenced in development of ontologies and use of semantic web technologies. Further, three case studies on interoperability issues and use of ontologies as a solution to interoperability issues have also been inspiration for this research study.

1. As mentioned, the first objective of this this is "comprehensive analysis on interoperability in Structural Engineering Domain". This research study aims to propose a method using semantic web technologies to integrate data from architectural domain and structural analysis domain. In order to achieve that proper understanding of drawbacks of using standard methods is important. A complete investigation was carried out in order to understand interoperability issues and reasons behind those issues. Following conclusions were made based on carried investigation:

    - Domain specific software tools have different internal schemas and during bi-directional data exchange, incorrect mapping and misinterpretation of information issues arises which causes data loss, errors and requires human effort and time. To overcome these issues inter-domain interpretation standards should be developed or building should be modelled using a standard integrated building data model.

    - IFC data model is most widely used for data exchange between diverse domain specific software. IFC certification, which permits software tools to import and export model using IFC data model requires capability to import and export basic structural object and model. In practical use where complex models and information exchange is required, certified tools fail to achieve effective sharing of information. Introduction of domain-specific certification process and re-consideration of certification standards is required in order to better data exchange process.

    - Undefined workflow and lack of collaboration between software industry within structural engineering domain and architectural domain leads to undefined modelling standards and misinterpretation of information. Proper communication and establishment of standards by representatives of software developers can frame a well-defined workflow which will benefit everyone and will ease the process of data exchange and mapping process.

Thus, it can be concluded that despite of many efforts and development of technologies meaningful data transfer between architectural and structural engineering domain is still a big question.

2. Second main objective of the thesis is “development and demonstration of Ontological model for denoting structural analysis data and architectural data”. In order to achieve semantic interoperability between architectural domain and structural analysis domain ontologies for structural analysis domain were developed and practical applicability was checked using a case study. Following observations were made based on proposed approach:

    - Developed ontology was able to denote architectural data queried from IFC based structural analysis model which indicates the enhanced data exchange and integration between architectural domain and structural analysis domain from syntactic level to semantic level by providing semantics to data.
    - Based on results it can be concluded that semantic web technologies enable the users and experts to represent, share and integrate architectural and structural analysis data through ontologies.
    - Architectural tools and structural analysis tools define certain data in different ways and therefore, most of the instances of *IfcBuildingElement* entity cannot be semantically transferred to ontology and query of those information using SPARQL could be time consuming.

    The idea of using semantic web technologies for data exchange and data integration between architectural domain and structural engineering domain offers strong possibility of semantic data exchange, correct interpretation of geometric information, domain knowledge integration and reusability of data.

5.2 Future Work

1. Development of an integrated software tool that can capture architectural data as well as structural analysis data from developed ontologies.
2. Proposed methodology aims to integrate data from architectural domain and structural analysis domain however, semantic web technologies offer possibility to integrate data and process. Developed ontologies should be used for various case study that aim to integrate process which lead to improvement in exchange of information.
3. Data exchange demonstration for domain specific building models and complicated loading scenarios should be carried out.
4. Developed ontologies could be used for integration framework which aims to exchange and integrate data from two different processes like BIM and Facility Management (FM) through semantic web technology.

6 References

[1] C. Mirarchi *et al.*, “An approach for standardization of semantic models for building renovation processes,” vol. XLIII, pp. 69–76, 2020.

[2] R. Jin, B. Zhong, L. Ma, A. Hashemi, and L. Ding, “Integrating BIM with building performance analysis in project life-cycle,” *Autom. Constr.*, vol. 106, no. June, 2019.

[3] A. Boukara, A. Naamane, and M. France, “A Brief Introduction to Building Information Modeling (BIM) and its interoperability with TRNSYS,” *Renew. Energy Sustain. Dev.*, vol. 1, no. 1, pp. 126–130, 2015.

[4] Z. Q. Liu, F. Zhang, and J. Zhang, “The building information modeling and its use for data transformation in the structural design stage,” *J. Appl. Sci. Eng.*, vol. 19, no. 3, pp. 273–284, 2016.

[5] P. Varghese, “Influence and Adoption of BIM within the AEC Industry,” no. February, pp. 1–32, 2019.

[6] J. Karlapudi and S. Shetty, “A methodology to determine and classify data sharing requirements between openBIM models and energy simulation models,” *Jrds.Ir*, 2019.

[7] R. Ren, J. Zhang, and H. N. Dib, “BIM interoperability for structure analysis,” *Constr. Res. Congr. 2018 Constr. Inf. Technol. - Sel. Pap. from Constr. Res. Congr. 2018*, vol. 2018-April, no. March 2018, pp. 470–479, 2018.

[8] J. Karlapudi *et al.*, “Enhancement of BIM Data Representation in Product-Process Modelling for Building Renovation Enhancement of BIM Data Representation in Product-,” 2020.

[9] H. Lai and X. Deng, “Interoperability analysis of ifc-based data exchange between heterogeneous BIM software,” *J. Civ. Eng. Manag.*, vol. 24, no. 7, pp. 537–555, 2018.

[10] E. K. Amoah and T. V. Nguyen, “Optimizing the usage of Building Information Model (BIM) interoperability focusing on data not tools,” *Proc. 36th Int. Symp. Autom. Robot. Constr. ISARC 2019*, no. July, pp. 1081–1090, 2019.

[11] “buildingSmart, ‘Industry Foundation Classes - Introduction.’” [Online]. Available: https://technical.buildingsmart.org/standards/ifc/. [Accessed: 22-Jan-2021].

[12] “buildingSMART, ‘IFC 4 ADD2 Documentation.’” [Online]. Available: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2/HTML/. [Accessed: 22-Jan-2020].

[13] R. Honti and J. Erdélyi, “Possibilities of bim data exchange,” *Int. Multidiscip. Sci. GeoConference Surv. Geol. Min. Ecol. Manag. SGEM*, vol. 18, no. 2.2, pp. 923–930, 2018.

[14] J. Karlapudi and K. Menzel, “Analysis on automatic generation of BEPS model from BIM model,” *Jrds.Ir*, 2020.

[15] R. Volk, J. Stengel, and F. Schultmann, “Building Information Modeling (BIM) for existing buildings - Literature review and future needs,” *Autom. Constr.*, vol. 38, no. March, pp. 109–127, 2014.

[16] “buildingSmart, ‘IFC certification participants.’” [Online]. Available: https://technical.buildingsmart.org/services/certification/ifc-certification-participants/. [Accessed: 23-Jan-2021].

[17] H. Lai, C. Zhou, and X. Deng, “Exchange requirement-based delivery method of structural design information for collaborative design using industry foundation classes,” *J. Civ. Eng. Manag.*, vol. 25, no. 6, pp. 559–575, 2019.

[18] G. Sibenik and I. Kovacic, “Assessment of model-based data exchange between architectural design and structural analysis,” *J. Build. Eng.*, vol. 32, no. April, p. 101589, 2020.

[19] C. Wan, P. Chen, and R. L. K. Tiong, “Assessment of IFC for structural analysis domain,” *ITcon*, vol. 9, no. May, pp. 75–95, 2004.

[20] Y. S. Sacks, R., Kaner, I., Eastman, C. M., and Jeong, “‘The Rosewood experiment-building information modeling and interoperability for architectural precast facades.’ Automation in Construction,” pp. 19(4) p 419-432., 2010.

[21] R. Islam, “Investigation of data sharing in the Structural Engineering domain,” Dresden, Germany, 2020.

[22] A. M. M. I.J. Ramaji, “Interpretation of structural analytical models from the coordination view in building information models, Autom. ConStruct.,” 2018.

[23] X. Wang, Z. P. Cui, Q. L. Zhang, and H. Z. Yang, “Creating structural analysis model from IFC-based structural model,” *Adv. Mater. Res.*, vol. 712–715, pp. 901–904, 2013.

[24] P. Pauwels, R. De meyer, and J. Van Campenhout, “Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface,” *Interoperability Des. Constr. Ind. through Semant. web Technol.*, vol. 6725 LNCS, no. December, 2011.

[25] C. Lima, C. Silva, C. Duc, P. Sousa, and J. Pimentao, “Interoperability among Semantic Resources in Construction: Is it Feasible?,” 2005.

[26] “Principles of Linked data.” [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed: 04-Feb-2021].

[27] N. Guarino, D. Oberle, and S. Staab, “Handbook on Ontologies,” *Handb. Ontol.*, no. May, pp. 0–17, 2009.

[28] K. Michal, Š. Michal, and B. Zdeněk, “Interoperability through ontologies,” *IFAC Proc. Vol.*, vol. 11, no. PART 1, pp. 196–200, 2012.

[29] M. Uschold and R. Jasper, “A Framework for Understanding and Classifying Ontology Applications,” *Methods*, pp. 1–12.

[30] D. Moodley, “Ontology Driven Multi-Agent Systems: An Architecture for Sensor Web Applications,” ** Thesis*, no. December, pp. 1–228, 2009.

[31] T. Berners-Lee, “The Semantic Web as a language of logic.” [Online]. Available: https://www.w3.org/DesignIssues/Logic.html. [Accessed: 05-Feb-2021].

[32] T. Berners-Lee and D. Connolly, “A readable RDF syntax.” [Online]. Available: https://www.w3.org/TeamSubmission/n3/. [Accessed: 05-Feb-2021].

[33] “RDF Vocabulary Description Language 1.0: RDF Schema.” [Online]. Available: https://www.w3.org/TR/2002/WD-rdf-schema-20020430/. [Accessed: 05-Feb-2021].

[34] “OWL 2 Web Ontology Language Primer (Second Edition).” [Online]. Available: https://www.w3.org/TR/owl2-primer/. [Accessed: 05-Feb-2021].

[35] “SPARQL 1.1 Query Language.” [Online]. Available: https://www.w3.org/TR/2013/REC-sparql11-query-20130321/. [Accessed: 05-Feb-2021].

[36] P. Pauwels, S. Zhang, and Y. C. Lee, “Semantic web technologies in AEC industry: A literature overview,” *Autom. Constr.*, vol. 73, pp. 145–165, 2017.

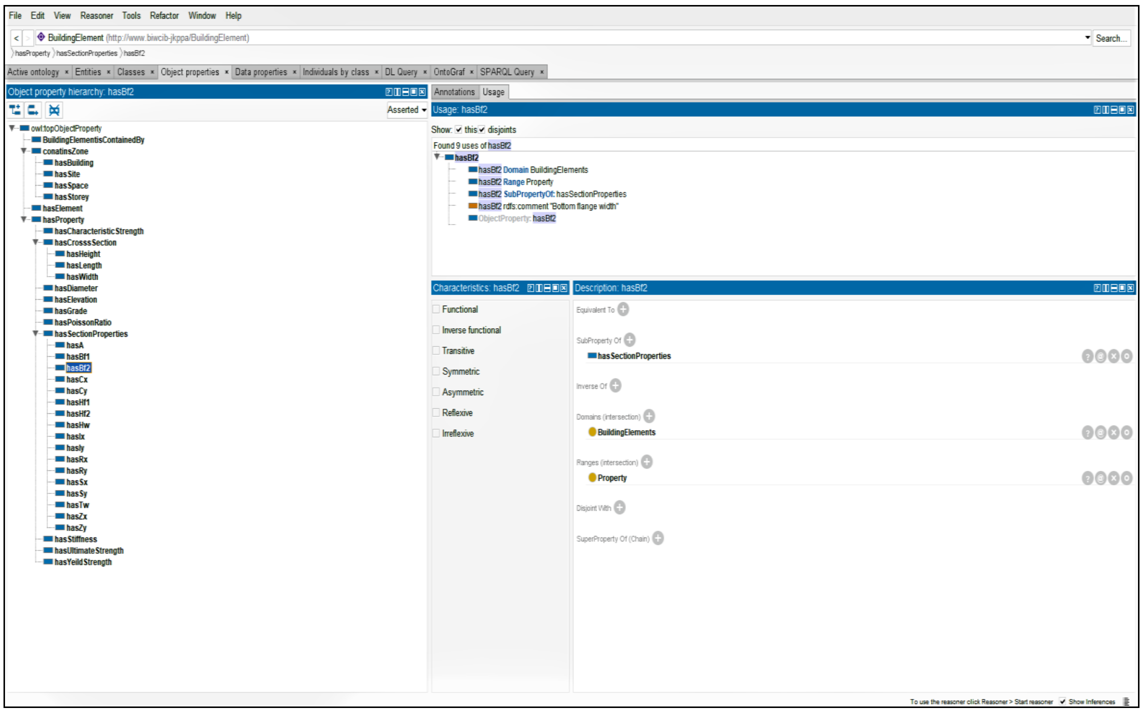
[37] S. Abdul-Ghafour, P. Ghodous, B. Shariat, and E. Perna, “A Common Design-Features Ontology for Product Data Semantics Interoperability,” *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. WI 2007*, pp. 427–430, 2007.

[38] V. Prathap, K. Janakiram, M. Karsten, and E. At., “A Semantic data model to represent building material data in AEC collaborative workflows A Semantic data model to represent building material data in AEC collaborative workflows,” pp. 0–10, 2020.

[39] E. P. Karan, J. Irizarry, and J. Haymaker, “BIM and GIS Integration and Interoperability Based on Semantic Web Technology,” *J. Comput. Civ. Eng.*, vol. 30, no. 3, p. 04015043, 2016.

[40] “IFCtoRDF converter.” .

7 Appendices



BuildingElement (http://www.biwcib-jkppa/BuildingElement)

hasProperty 〉hasSectionProperties 〉hasBf2

Object property hierarchy: hasBf2

```
owl:topObjectProperty
    BuildingElementisContainedBy
    conatinsZone
        hasBuilding
        hasSite
        hasSpace
        hasStorey
    hasElement
    hasProperty
        hasCharacteristicStrength
        hasCrosssSection
            hasHeight
            hasLength
            hasWidth
        hasDiameter
        hasElevation
        hasGrade
        hasPoissonRatio
        hasSectionProperties
            hasA
            hasBf1
            hasBf2
            hasCx
            hasCy
            hasHf1
            hasHf2
            hasHw
            hasIx
            hasIy
            hasRx
            hasRy
            hasSx
            hasSy
            hasTw
            hasZx
            hasZy
        hasStiffness
        hasUltimateStrength
        hasYeildStrength
```

Usage: hasBf2

Show: ✓ this ✓ disjoints

Found 9 uses of hasBf2

```
hasBf2
    hasBf2 Domain BuildingElements
    hasBf2 Range Property
    hasBf2 SubPropertyOf: hasSectionProperties
    hasBf2 rdfs:comment "Bottom flange width"
    ObjectProperty: hasBf2
```

Characteristics: hasBf2

- ☐ Functional
- ☐ Inverse functional
- ☐ Transitive
- ☐ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

Description: hasBf2

Equivalent To

SubProperty Of
hasSectionProperties

Inverse Of

Domains (intersection)
BuildingElements

Ranges (intersection)
Property

Disjoint With

SuperProperty Of (Chain)

To use the reasoner click Reasoner > Start reasoner ✓ Show Inferences

Figure 72: Object property hierarchy of BEO

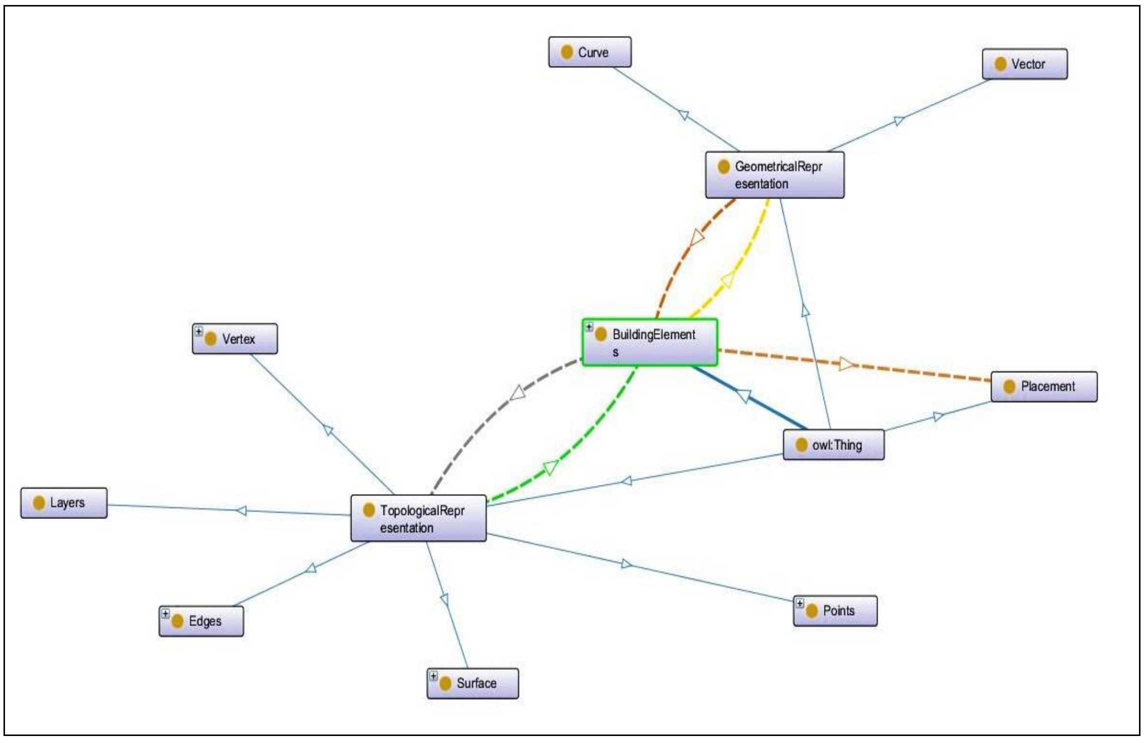
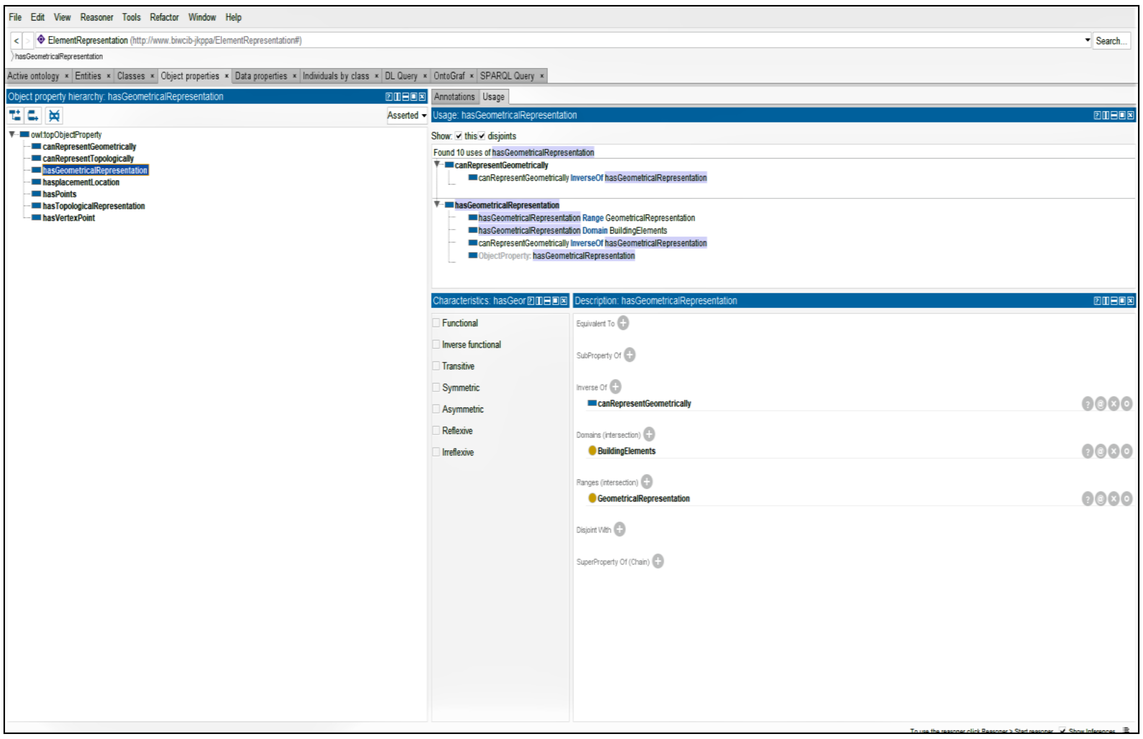
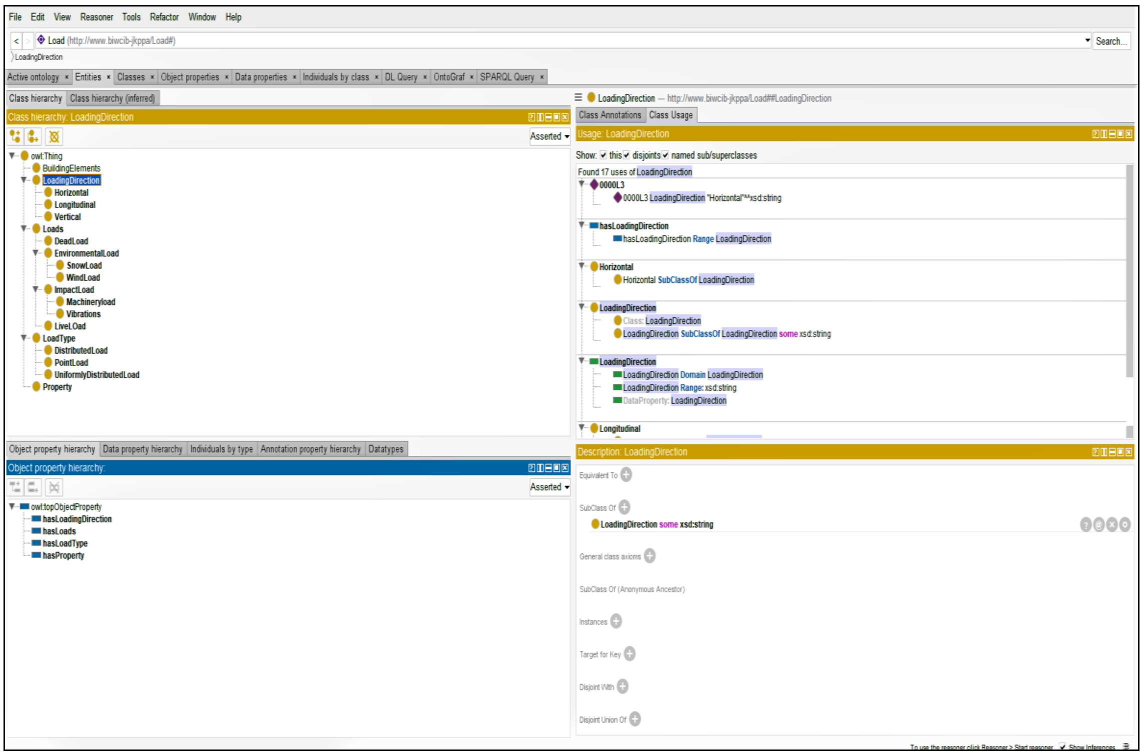


Figure 73: Graphical representation of class hierarchy of BERO



```
ElementRepresentation (http://www.biwcib-jkppa/ElementRepresentation#)
hasGeometricalRepresentation

Object property hierarchy: hasGeometricalRepresentation
owl:topObjectProperty
    canRepresentGeometrically
    canRepresentTopologically
    hasGeometricalRepresentation
    hasplacementLocation
    hasPoints
    hasTopologicalRepresentation
    hasVertexPoint

Usage: hasGeometricalRepresentation
Show: ✓ this ✓ disjoints
Found 10 uses of hasGeometricalRepresentation
canRepresentGeometrically
    canRepresentGeometrically InverseOf hasGeometricalRepresentation
hasGeometricalRepresentation
    hasGeometricalRepresentation Range GeometricalRepresentation
    hasGeometricalRepresentation Domain BuildingElements
    canRepresentGeometrically InverseOf hasGeometricalRepresentation
    ObjectProperty: hasGeometricalRepresentation

Characteristics: hasGeor
☐ Functional
☐ Inverse functional
☐ Transitive
☐ Symmetric
☐ Asymmetric
☐ Reflexive
☐ Irreflexive

Description: hasGeometricalRepresentation
Equivalent To
SubProperty Of
Inverse Of
    canRepresentGeometrically
Domains (intersection)
    BuildingElements
Ranges (intersection)
    GeometricalRepresentation
Disjoint With
SuperProperty Of (Chain)
```

Figure 74: Object property hierarchy of BERO



```
Load (http://www.biwcib-jkppa/Load#)
LoadingDirection

Class hierarchy: LoadingDirection
owl:Thing
    BuildingElements
    LoadingDirection
        Horizontal
        Longitudinal
        Vertical
    Loads
        DeadLoad
        EnvironmentalLoad
            SnowLoad
            WindLoad
        ImpactLoad
            Machineryload
            Vibrations
        LiveLOad
    LoadType
        DistributedLoad
        PointLoad
        UniformlyDistributedLoad
    Property

Object property hierarchy:
owl:topObjectProperty
    hasLoadingDirection
    hasLoads
    hasLoadType
    hasProperty

LoadingDirection — http://www.biwcib-jkppa/Load##LoadingDirection
Usage: LoadingDirection
Show: ✓ this ✓ disjoints ✓ named sub/superclasses
Found 17 uses of LoadingDirection
0000L3
    0000L3 LoadingDirection "Horizontal"^^xsd:string
hasLoadingDirection
    hasLoadingDirection Range LoadingDirection
Horizontal
    Horizontal SubClassOf LoadingDirection
LoadingDirection
    Class: LoadingDirection
    LoadingDirection SubClassOf LoadingDirection some xsd:string
LoadingDirection
    LoadingDirection Domain LoadingDirection
    LoadingDirection Range: xsd:string
    DataProperty: LoadingDirection
Longitudinal

Description: LoadingDirection
Equivalent To
SubClass Of
    LoadingDirection some xsd:string
General class axioms
SubClass Of (Anonymous Ancestor)
Instances
Target for Key
Disjoint With
Disjoint Union Of
```

Figure 75: Class hierarchy and object property hierarchy of BLO

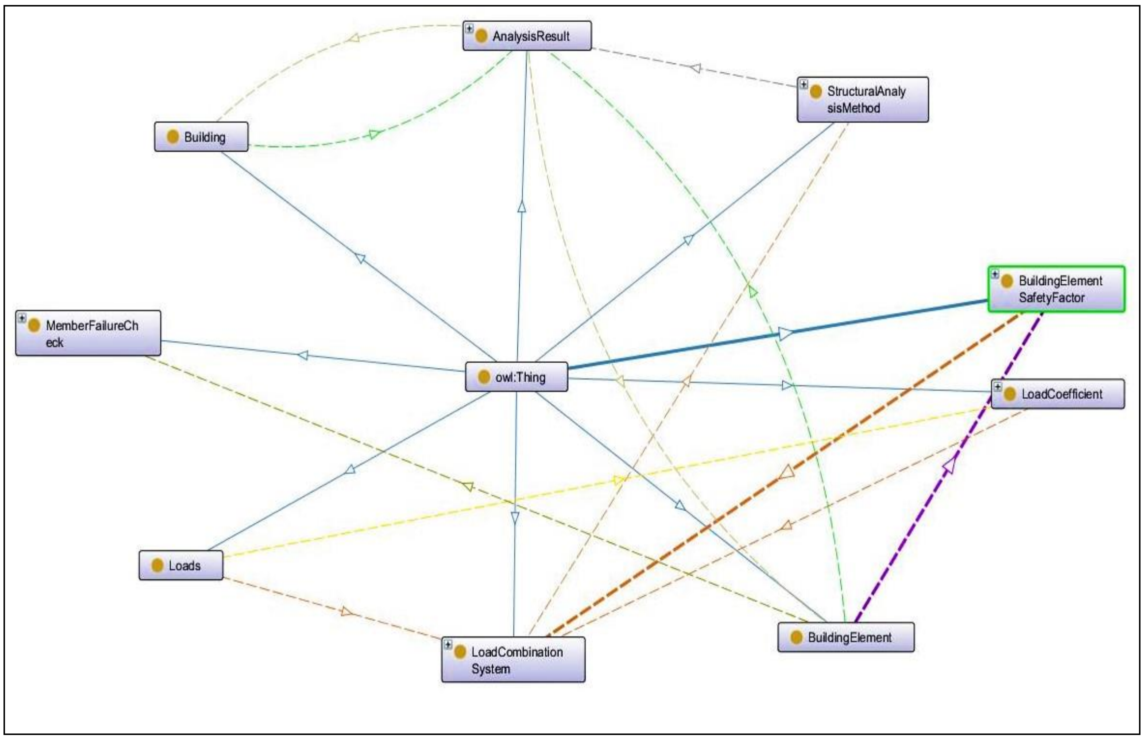
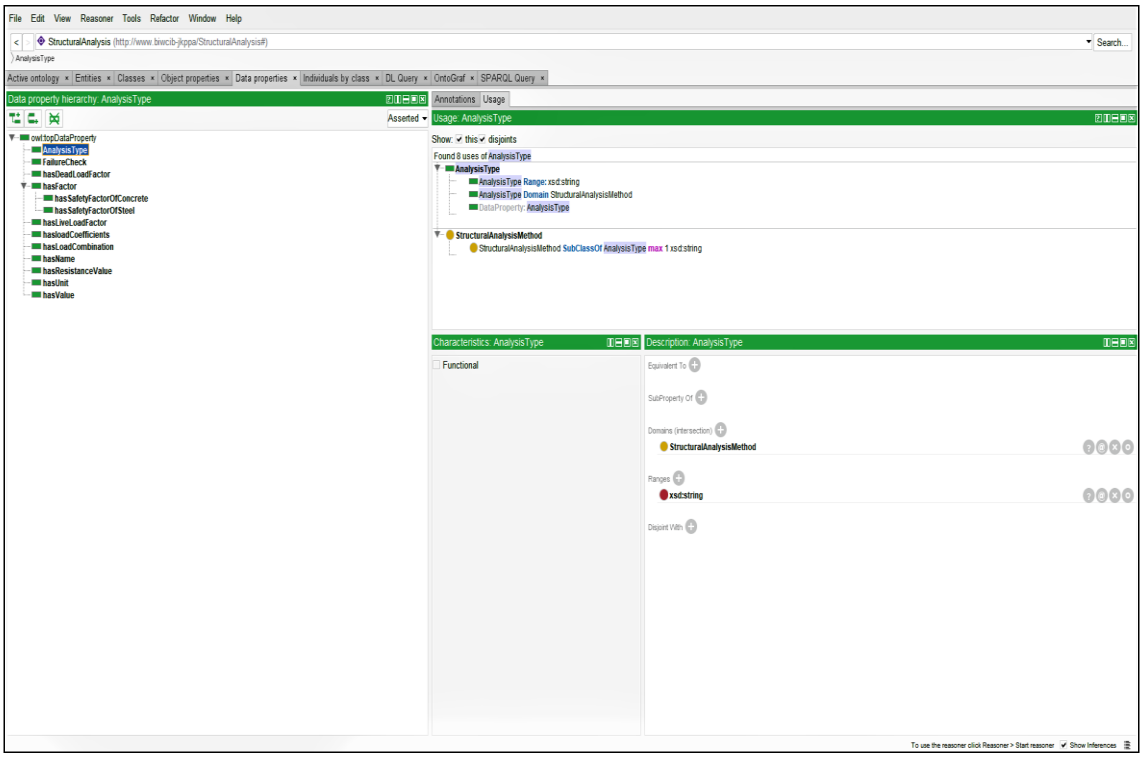


Figure 76: Class hierarchy of SAO


File Edit View Reasoner Tools Refactor Window Help
Search...
Active ontology × Entities × Classes × Object properties × Data properties × Individuals by class × DL Query × OntoGraf × SPARQL Query ×
Annotations Usage
Asserted

StructuralAnalysis (http://www.biwcib-jkppa/StructuralAnalysis#)
AnalysisType

Data property hierarchy: AnalysisType

- owl:topDataProperty
  - AnalysisType
  - FailureCheck
  - hasDeadLoadFactor
  - hasFactor
    - hasSafetyFactorOfConcrete
    - hasSafetyFactorOfSteel
  - hasLiveLoadFactor
  - hasloadCoefficients
  - hasLoadCombination
  - hasName
  - hasResistanceValue
  - hasUnit
  - hasValue

Usage: AnalysisType

Show: ✓ this ✓ disjoints

Found 8 uses of AnalysisType

- AnalysisType
  - AnalysisType Range: xsd:string
  - AnalysisType Domain StructuralAnalysisMethod
  - DataProperty: AnalysisType
- StructuralAnalysisMethod
  - StructuralAnalysisMethod SubClassOf AnalysisType max 1 xsd:string

Characteristics: AnalysisType

☐ Functional

Description: AnalysisType

Equivalent To

SubProperty Of

Domains (intersection)
- StructuralAnalysisMethod

Ranges
- xsd:string

Disjoint With

To use the reasoner click Reasoner > Start reasoner ✓ Show Inferences

Figure 77: Data properties of SAO