# Knowledge Graphs for Multidisciplinary Co-Design: Introducing RDF to BHoM

Diellza Elshani[1], Alessio Lombardi[2], Al Fisher[2], Steffen Staab[3,4], Daniel Hernández[3] and Thomas Wortmann[1]

[1] *Chair for Computing in Architecture (CA), Institute for Computational Design and Con-struction (ICD), Faculty of Architecture and Urban Planning, University of Stuttgart, Keplerstraße 11, Stuttgart, 70174, Germany*
[2] *Buro Happold, 17 Newman St, London, W1T 1PD, United Kingdom*
[3] *Department for Analytic Computing (AC), Institute for Parallel and Distributed System (IPVS), University of Stuttgart, Universitätsstraße 38, Stuttgart, 70174, Germany*
[4] *Electronics and Computer Science, University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom*

**Abstract**

Architecture, engineering, and construction (AEC) projects require multidisciplinary solutions; therefore, one physical asset results in several disciplinary representations. Interoperability problems between different software often hinder disciplinary data integration, which prevents the recognition of violated design constraints until it is too late. The open-source platform Building Habitat object Model (BHoM) integrates many AEC concepts across existing languages and platforms, allowing multiple discrete disciplinary representations of buildings. The Linked Building Data Community Group of the World Wide Web Consortium uses Semantic Web standards to store and share data. This paper explores and presents methods to recast BHoM's object model as a knowledge graph using Semantic Web standards. It analyzes BHoM in an ontological context by elaborating on its approach to object-oriented programming and comparing it to the Semantic Web standards.

Furthermore, it presents BHoM RDF prototypes as a potential solution to the interoperability problem of multidisciplinary design in AEC. It exemplifies BHoM RDF prototypes on a small demonstrator project. When combined with Semantic Web standards, the BHoM framework can increase the use of knowledge graphs in the AEC industry, improving data interoperability and assisting design decisions through reasoning.

**Keywords**

Interoperability, Knowledge representation, Linked building data, Graph data modelling, Ontology engineering, Data mapping, Open source.

## 1. Introduction

The building industry is fragmented [1], and requires multidisciplinary solutions [2]. To develop disciplinary solutions, architecture, engineering, and construction (AEC) professionals create discipline-specific information. Therefore, one physical asset results in several distinct disciplinary representations [3]. An integrative approach, such as 'Co-Design' [4], considers design and analysis

LDAC 2022: 10th Linked Data in Architecture and Construction Workshop, May 29, 2022, Hersonissos, Greece
EMAIL: diellza.elshani@icd.uni-stuttgart.de (D. Elshani); alessio.lombardi@BuroHappold.com (A. Lombardi); al.fisher@BuroHappold.com (A. Fisher); steffen.staab@ipvs.uni-stuttgart.de (S. Staab); daniel.hernandez@ipvs.uni-stuttgart.de (D. Hernández); thomas.wortmann@icd.uni-stuttgart.de (T. Wortmann)
ORCID: 0000-0003-2902-341X (D. Elshani); 0000-0002-0780-4154 (S. Staab); 0000-0002-7896-0875 (D. Hernández); 0000-0002-5604-1624 (T. Wortmann)


methods simultaneously, encompassing building systems, materials, manufacturing, and construction processes. Therefore multidisciplinary co-design must integrate data from the early stages of design; furthermore, AEC professionals should be able to exchange data about building elements rather than files [5]. In current practice, interoperability problems between different software often hinder data integration. These interoperability issues may prevent the recognition of violated design constraints until it is too late, i.e., until construction has already started.

To tackle interoperability issues the Industry Foundation Classes (IFC) data schema was created [6]. The IFC data schema represents building elements, including semantic information such as attributes, relationships, and abstract concepts or processes in a monolithic approach. However, the fragmented nature of the AEC industry contrasts with the monolithic approach of IFC [1]. A common data schema causes restrictions on how buildings can be represented and thus explored [7].
Contrary to IFC, the open source platform Building Habitat object Model (BHoM) [8] does not rely on a single "neutral format" but rather provides adapters to map data among many AEC concepts across existing languages and platforms, and it allows multiple discrete disciplinary representations of buildings and building elements; with an open and extendable data dictionary.

The Linked Building Data Community Group (LBD-CG) [9] of the World Wide Web Consortium (W3C) has focused on using Semantic Web standards as an open, decentralized alternative to the existing centralized and file-based BIM[2] approaches of storing and sharing data [10]. Semantic Web technology remains one of the most promising approaches to share data and achieve interoperability between heterogeneous systems [5]. The Semantic Web uses knowledge graphs, a graph-structured database model, to integrate and link data. Besides the advantages of linking data, formal Semantic Web languages have rich expressivity supported by reasoning tools, which allow making claims, known as axioms, about these data [11]. In AEC, this inferred knowledge would allow recognizing design constraints from involved disciplines such as structural or fabrication from early design stages.

IFC is available in Semantic Web standards represented as a knowledge graph. But the support for AEC knowledge graphs and linked data in tool development remains low [12]. The reason for this low support might be the monolithic approach and hierarchical format of IFC in representing data. Elshani, et al., [3] discuss centralized data exchange over a shared data schema as unsuitable for co-design and argues for using federated data interoperability instead. Furthermore, they suggest using BHoM to construct disciplinary representations of buildings and building elements as a framework that applies federated interoperability. In a federated interoperability paradigm, each discipline uses its corresponding tool to represent discipline-specific data, but they share a core ontology that all disciplines can relate to and refine [3]. BHoM has a shared mapping process through identifiers, but lacks a shared ontology which would allow linking BHoM objects. Linked data, where each discipline manages local data, improves data interoperability; thus, it provides a good foundation for data exchange [13].

Therefore, this paper compares BHoM and Semantic Web standards and presents prototypes that introduce Semantic Web standards to BHoM, applying linked data principles alongside BHoM as a potential solution to the interoperability problem of multidisciplinary design in AEC. Linked BHoM object models would not only allow constructing many multiple discrete disciplinary representations of building elements but also carry all the advantages of the Semantic Web, including reasoning ability by augmenting the meaning of the pre-defined knowledge.

In this paper, we elaborate on BHoM in an ontological context by explaining its object-oriented programing (OOP) approach and comparing it to Semantic Web standards. Afterwards, we present mapping methods to convert existing BHoM object models to the standard database model for data interchange on the web, the Resource Description Framework (RDF). We discuss and present technical implementations of BHoM RDF prototypes. We also restrict BHoM data with RDF Schema (RDFS) and the Web Ontology Language (OWL) vocabulary. Finally, we present a small demonstrator of our approach, discuss its advantages and limitations. The presented building demonstrator is a building described in BHoM objects converted to an OWL graph, using the developed method to convert the BHoM data model to Semantic Web standards.

[2] Building Information Modeling files using the IFC data schema.

## 2. Background

BHoM is an open source collaborative computational framework that integrates many concepts across existing AEC languages and platforms [8] [14]. BHoM consists of (1) object models (oM), (2) engines that operate on data, (3) adapters that map and translate data and (4) user interfaces to manipulate data. Currently, BHoM has over 1.200 object models with an extendable data dictionary and adapters to over 30 different software. It also allows using customized object models. While OOP classes implement or inherit implementations of methods, RDF does not have methods. But BHoM's interpretation of OOP is closer to an "ontological" approach than the "classical" OOP paradigm [15]. In the following sections, we first discuss BHoM's approach to OOP and afterwards compare BHoM with the technological standards set by the W3C and the LBD-CG.

### 2.1. BHoM 's Interpretation of OOP

At its core, BHoM's object model is a set of C# a programming language types (classes, interfaces, enums, etc.) and methods. But BHoM takes an approach that often departs from conventional OOP principles by: (1) Separating functionality from types and (2) focusing on ontologically meaningful interfaces.

First and foremost, BHoM imposes that all functionality is separated from the types. In other words, BHoM types have attributes but no methods. All functionality applicable to the oM types is isolated, and it is primarily grouped in C# projects called *Engines*. Similarly to oM projects, these groups target a specific domain, use specific namespaces and are suffixed with "*Engine*", e.g. methods for Structural Engineering are placed in the *Structure_Engine* under namespaces starting with *BH.Engine.Structure*. This separation makes BHoM similar to a virtual object database like those used to convert objects to relational databases[3].

Further, the development of BHoM types strictly follows the composition over inheritance principle. There are numerous interfaces and interface implementations, with each interface holding not only a contractual agreement with implementing types but also an ontological meaning.

### 2.2. Comparison of BHoM and Semantic Web standards

Whereas BHoM does not apply the standards set by the W3C, LBD community group, and developed through the Linked Data for Architecture and Construction (LDAC) workshops, conversion is still possible since they differ mainly in their technical implementations. The following subsections compare BHoM to Semantic Web standards regarding their identifiers, data models, data schemas, data exchange formats, and querying methods (see Table 1).

**Identifiers**. A noticeable difference between the two approaches is the way they identify things. Whereas in the Semantic Web, all nodes and edges (except for literals, e.g. strings, dates, etc.) are identified using a dereferenceable Uniform Resource Identifier (URI) [12], BHoM uses Globally Unique IDentifier (GUID) to identify the object models. Linked Data of the Semantic Web relies on HTTP URLs as identifiers of things [16]. But BHoM conventions also establish that every oM type and every piece of functionality is separated into different files. Because each file is committed to a public Git repo, each oM type is associated with a Unique Resource Identifier (URI) in the form of an HTTP address.

**Database Model.** While the Semantic Web uses a graph-based data model [17], BHoM relies on an object-oriented data model. RDF is a standard graph-based data model for data interchange on the internet [10] and consists only of 'triples': subject, predicate and object [12], whereas BHoM does not include explicit classes that describe the relationships between objects.

[3] Database Object-Relational Mappers (ORMs)

**Data Schema.** The Semantic Web relies on two languages to state the schema and terminological knowledge of an RDF graph: RDF Schema (RDFS) [18] and the Web Ontology Language (OWL) [19]. The former is a minimal language that describes classes and relationships. The latter extends RDFS with features that can be mapped into an expressive description logic. A further approach to describe RDF graphs is SHACL, which constrains data to follow the schema. Compared to relational data models, where an upfront defined schema is followed at each step, modeling data as a graph offers more flexibility to integrate new data sources [11]. On the other hand, BHoM object models are classified in specific namespaces, e.g., *Architecture_oM*, *Structure_oM*, *Acoustic_oM*, etc., allowing disciplinary representation of buildings and building elements. BHoM oM schema relies on the C# classes modeling. For example, a representation of a room in the Architectural BHoM namespace is:

```
namespace BH.oM.Architecture.Elements
{
   public class Room : BHoMObject, IRegion, IElement2D
   {
      public virtual ICurve Perimeter { get; set; } = null;
      public virtual Point Location { get; set; } = null;
   }
}
```

The namespace *BH.oM.Architecture.Elements* locates the Room type in the architectural domain, while the suffix *Elements* is a term conventionally used throughout BHoM to categorise things in a domain of reference. The Room type owns two properties, Perimeter and Location, which are in turn other oM types. Each type and property (e.g. Perimeter) is usually accompanied by description attributes (e.g. ICurve) that help convey their semantic value.

**Querying.** While the Semantic Web uses SPARQL to navigate data, BHoM querying is generally implemented through framework-specific functions hosted in the *BHoM_Engine* Query classes. In addition, certain *BHoM_Adapters* are able to convert oM models to other queryable formats, for example, the *Mongo_Adapter*. An advantage of graph query languages is their compactness for certain queries [20]. Furthermore, graph querying also allows retrieving and manipulating data from combined graphs. Specifically, SPARQL allows the full use of relational operators, including path expressions.

**Table 1**
BHoM and Semantic Web standards comparison

| | Identifiers | Database Model | Data Schema | Data Exchange Format | Querying |
|---|---|---|---|---|---|
| **Semantic Web** | URI | RDF | RDFS, OWL, SHACL | TTL, N-Triples, JSON-LD, RDF/XML | SPARQL |
| **BHoM** | GUID | Object-Oriented | BHoM Namespaces and BHoM classes | JSON | BHoM_Engine Query and MongoDB |

## 2.3. Summary

Considering BHoM's rich semantics and its approach to OOP in separating functionality from the objects, it is possible to state that the BHoM oM is effectively akin to an ontology: because each oM class owns semantic data, we can have a mapping to an ontological class. Not only each BHoM class can be mapped to an ontological class, but it is also possible to derive many ontological relationships from the BHoM classes (e.g. from inheritance or interface classes).

Currently, BHoM does not implement any class to represent the relationship between different oM classes. For example, a column can be described in the physical domain via the oM type *BH.oM.Physical.Elements.Column* and it can have an alternative representation in the structural engineering domain through the oM type *BH.oM.Structure.Elements.Bar*. Concretely, a typical design workflow may involve conversion or creation of a structural model for finite element analysis from a BIM model, with the logic and relationships between these distinct multidiscipline elements being encoded inside the computational process/script or indeed lost within any manual steps taken. However, there is at present no class that can capture these possible relationships between instances of oM Columns and oM Bars, say, to enable logical persistence beyond the script itself.

The RDF simple entailment semantics follow the open-world assumption, and ontologies using the RDFS and OWL languages follow the open-world assumption. On the contrary, SPARQL has operators that do not follow the open-world assumption, such as operators minus and optional. Under the open-world assumption, the RDF/OWL ontology admits more interpretations than the BHoM model under the closed-world assumption. All inferences under the open-world assumption are also valid under the closed-world assumption. Hence, RDF/OWL inferences do not hinder interoperability. BHoM is written in an OOP language for convenience reasons, but it does not adhere strictly to OOP design patterns. BHoM is designed to be extensible and to avoid limitations in representational potency and interoperability. This paper shows how BHoM models can be translated to RDF/OWL ontologies, and strictly speaking, the resulting ontology is not equivalent to the BHoM model. Inference in BHoM models can be complemented with RDF/OWL features.

By leveraging BHoM's particular approach to OOP and bridging the gap in the definition of relations between oM types, it is possible to convert BHoM concepts into Semantic Web vocabularies such as RDF, RDFS and OWL. BHoM ontology can be enriched with other relations, allowing inferential reasoning to be applied to oM models.

## 3. Methods

As argued in the previous section, BHoM's rich semantics and BHoM's approach to OOP in separating functionality from objects make it possible to convert BHoM object models to an OWL ontology. In order to perform such a conversion, we considered the following sub-problems: (1) Terminological translation and generation, which represents the ontology of BHoM objects; and (2) Assertional translation and generation - which will contain object instances, specifically actual BHoM data, in an RDF data model. This paper focuses on the first sub-problem. It proposes a set of mappings (see Table 2) between BHoM framework types, type properties and inter-type relationships to ontological concepts, such as classes, object properties and data properties as well as datatypes.

Mapping BHoM to RDF requires a specific approach because BHoM doesn't have oM types that represent relationships between different oM types. Therefore, before mapping existing BHoM concepts to RDF, RDFS, and OWL, a new oM interface IRelation is introduced and a set of classes implementing it which serve as owl:ObjectProperties. Currently, three classes are introduced that implement IRelation: hasProperty, IsAListOf and RequiresProperty. Together with information extracted from oM types, these newly introduced classes obtain a the foundation to representing the code and its semantic knowledge in a graph-based database.

Table 2 presents several mapping from BHoM oMs to Semantic Web standards, including mappings for BHoM components such as identifiers, object classes, inheritance and interface implementations, class properties, as well as datatypes.

BHoM classes and interfaces (such as BHoM_Room or BHoM_ICurve) are mapped to a class in OWL. Both class inheritance relations and interface implementation relations are represented as rdfs:subClassOf. BHoM interfaces are ubiquitous and are often used to represent concepts in a taxonomic manner. For example, the interface IElement1D represents all types that have a linear (1 dimensional) representation, e.g.: columns, beams, pipes. This taxonomic information that interfaces hold makes the interface implementation relationship between concepts an ideal candidate for the rdfs:subClassOf.

A BHoM class may define a property for its instances. The values that the property can take is defined in the BHoM class. For example, the code "public virtual ICurve Perimeter" in the definition of class Room states that the property Room#Perimeter takes values from objects that implement the interface ICurve. The values of the property can be mapped to RDF terms, which can be instances of classes or literals.

A technical implementation of the conversion, including the presented mapping concepts, is shown in the results section.

**Table 2**
BHoM to RDF mapping concepts

| BHoM component | Semantic Web equivalent | Comment |
|---|---|---|
| Concept identifier | URI | The original GitHub oM class URI. |
| GUID -Instance identifier | URI | URI built on a domain name, concatenated with the GUID of BHoM instances. |
| Object class<br>e.g. BH.oM.Physical.Elements Column | an owl:Class,<br>e.g.bhom:Physical.Elements Column | An exception to this rule are BHoM Enum types. |
| Class inheritance relation | rdfs:subClassOf | |
| Interface (eg. IElement1D) | bhom:IElement1D a owl:Class | In BHoM interfaces are ubiquitous and are often used to represent concepts in a taxonomic manner. |
| Interface implementation relation | rdfs:subClassOf | |
| Class property<br>(eg. Perimeter of Room, or OrientationAngle of Bar) | Type owl:Class,<br>or rdfs:Literal | If the property is a class with further properties it becomes an owl:Class, else rdfs:Literal |
| Property relation<br>-not implement in BHoM | bhom:hasProperty<br>Type owl:ObjectProperty<br>or type owl:DatatypeProperty | A new property named bhom:hasProperty is introduced |
| Datatypes | xsd rdfs:Datatype | System.String are represented with xsd:string; BHoM types that do not fall in any other category are treated as Literals. |
| Lists | Type rdf:Seq | To simplify serialization rdf:Seq is proposed instead of rdf:list. |
| IEnumerables and HashSets | Type rdf:Bag | Unordered collections, since they do not possess ordering property. |
| Dictionaries | Type rdf:Bag of owl:classes | Dictionaries are unordered collections of Tuples. A named graph is used to represent a set of tuples (Key/Value). |
| Enums | Type rdf:Alt | rdf:Alt doesn't have the ordering property C# Enums have, but such quality only matters from a programming perspective. rdf:Alt also allows for a default value, which is mapped to the first value of the enum in BHoM |

## 4. Results

The previous chapter introduces mapping methods of BHoM's object model to the RDF model. The technical implementation of BHoM RDF is currently at a prototype stage whose main aim is to demonstrate its feasibility. Its development is open-sourced in the GitHub repository RDF_prototypes (https://github.com/BHoM/RDF_Prototypes). The repository consists of a BHoM RDF_Toolkit, a Visual Studio C# solution built on the BHoM framework. The Toolkit parses the existing code base using a mixture of C# Reflection, document parsing and processing, to build a graph of the BHoM types and their code-base relationships (inheritance, interface implementation, property exposure, etc.) using RDF, RDFS and OWL vocabulary. A close up view of BHoM oM, BHoM_Object base described in BHoM (left) and its generated graph (right) is presented in Figure 1. Whereas every *BHoMObject* implements the *IBHoMObject* interface, *IBHoMObject* implements the *IObject* interface. Therefore in OWL convention, a *BHoMObject* becomes a subclass of *IBHoMObject*, and *IBHoMObject* is a subclass of *IObject*. Properties such as *GUID*, *Name*, *Fragments*, *Tags* and *Dictionary* belong to the interface *IBHoMObject*; therefore, they belong to the *IBHoMObject* class in its OWL representation.

```
namespace BH.oM.Base
{
   public class BHoMObject : IBHoMObject
   {

      public virtual Guid BHoM_Guid { get; set; } = Guid.NewGuid();

      public virtual string Name { get; set; } = "";

      public virtual FragmentSet Fragments { get; set; } = new FragmentSet();

      public virtual HashSet<string> Tags { get; set; } = new HashSet<string>();

      public virtual Dictionary<string, object> CustomData { get; set; } = new Dictionary<string, object>();
   }
}
```

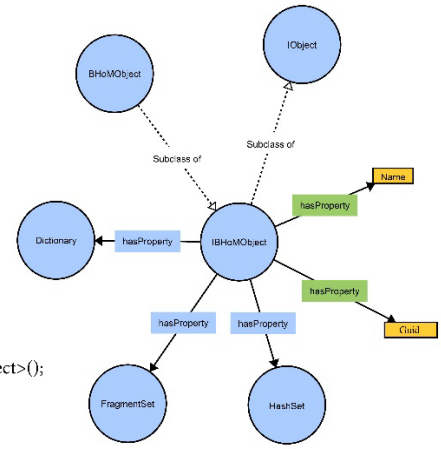


**Figure 1**: BHoM_Object base (left) and its corresponding graph (right).

The RDF_Toolkit implements a conversion to the JSON-LD structured format for every existing BHoM object, resulting in over 1200 classes. We focus on the generations of the following ontologies of the BHoM framework:

1. Domain-specific ontologies, or namespace ontologies. These are ontologies that illustrate the relations between classes defined in the domain-specific BHoM namespaces, e.g. *BH.oM.Architecture*, or *BH.oM.Structure* (structural engineering) (see Fig. 2).
2. Cross-domain ontologies, which describe inner and interface relations between namespace ontologies.
3. Concept-specific ontologies, or object ontologies, which focus on specific BHoM classes. E.g. a cluster of building elements that correspond to a specific building system.

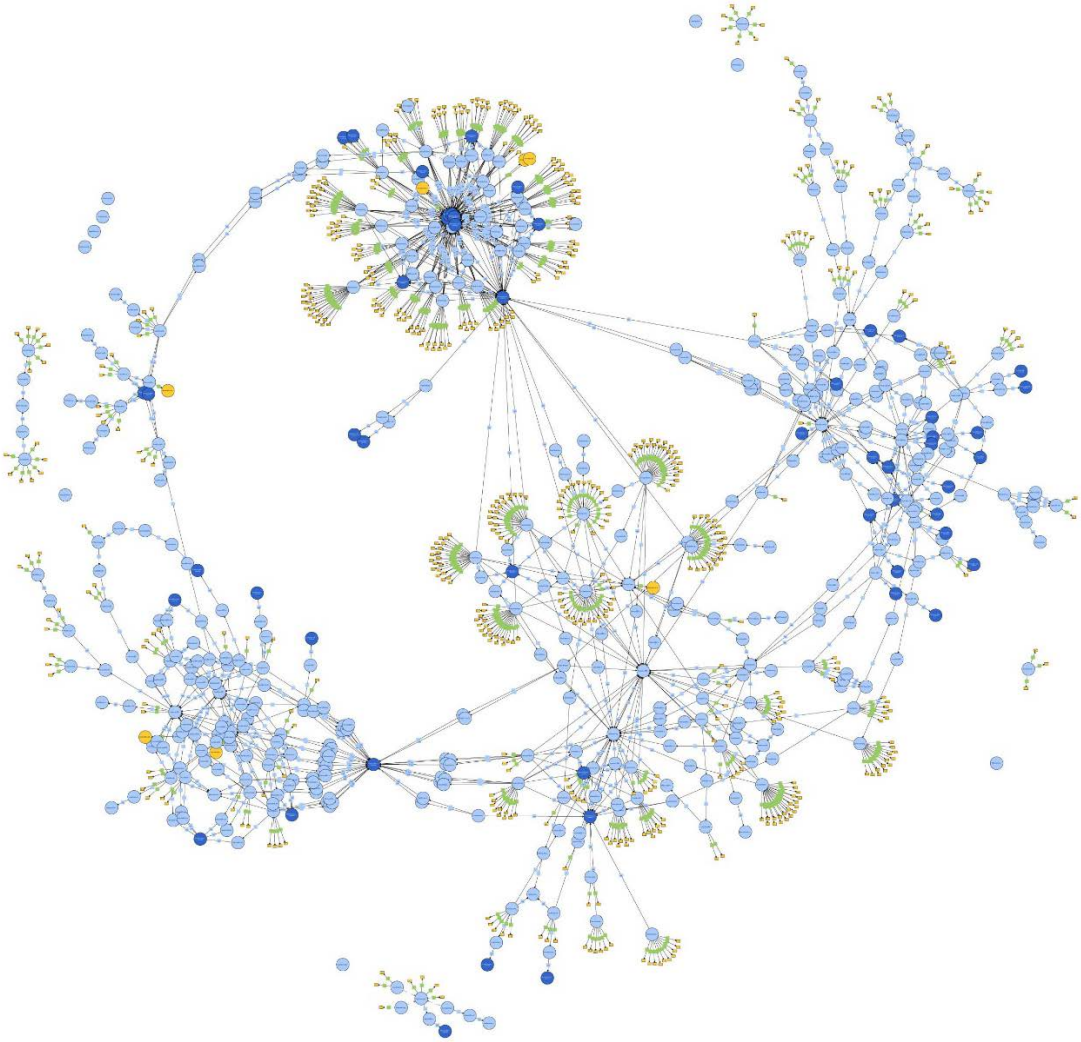

**Figure 2**: Graph of the ontology for the BH.oM.Structure (structural engineering) namespace.

## 4.1. Building Demonstrator

Following the concept-specific ontologies generation, this section presents a demonstrator building described in BHoM objects and in its generated graph database. The building consists of five BHoM oMs: room, wall, column, floor, and ceiling. Additionally, the graph includes all oM that help to build the aforementioned building elements (e.g. Curve, Point, and Name as input oMs to construct a Room). To search for classes that build oMs graphs, the tree recursion is set to 100. This parameter works to iteratively define how many levels on the graph nodes are expanded. This large number helps to reveal all atomic data entities. The resulting building graph is visualized in the WebVOWL (see Fig. 3). The demonstrator building graph resulted in 231 triples, 47 unique Classes, including main concepts such as Room or Column, and 145 Properties, mainly HasProperty with different domains and range. As example from the exported graph in *.ttl format, an owl:class (Room) and an owl:ObjectProperty (HasProperty) are presented below:

```
#  ---------------------------- Class  1-------------------------
<https://github.com/BHoM/BHoM/blob/main/Architecture_oM/Elements/Room.cs> rdf:type owl:Class;
        rdfs:subClassOf <https://github.com/BHoM/BHoM/blob/main/BHoM/BHoMObject.cs>
        rdfs:label "Room (BH.oM.Architecture.Elements)"@en .
```

```
# ---------------------------- Property 4-------------------------
<https://github.com/BHoM/RDF_Prototypes/blob/main/RDF_oM/HasProperty.cs>  rdf:type owl:ObjectProperty ;
        rdfs:label "HasProperty"@en;
        rdfs:domain <https://github.com/BHoM/BHoM/blob/main/Architecture_oM/Elements/Room.cs>;
        rdfs:range <https://github.com/BHoM/BHoM/blob/main/Geometry_oM/Curve/Polyline.cs> .
```

Since the domain and range of the properties differ, the graph resulted in a large number of properties with the same label (mainly HasProperty). To avoid confusion between different owl:ObjectProperties naming of the relations needs to be reevaluated. However, the representation of BHoM building data in RDF triples demonstrates that any BHoM object model can be mapped to a graph-based database model using Semantic Web standards. Such workflows with selected building elements allows to describe specific building systems and form re-usable building system ontologies from BHoM.
In the scope of this paper, methods to populate the graph with actual data from Grasshopper 3D are not investigated. However, data population from different editors (such as Protégé) is possible, demonstrating the feasibility of a complete BHoM RDF knowledge graph. Extending this framework to generate object instances from design platforms through the BHoM interfaces would provide a solid foundation for describing complex building models' in knowledge graphs across many AEC software in a straightforward way.

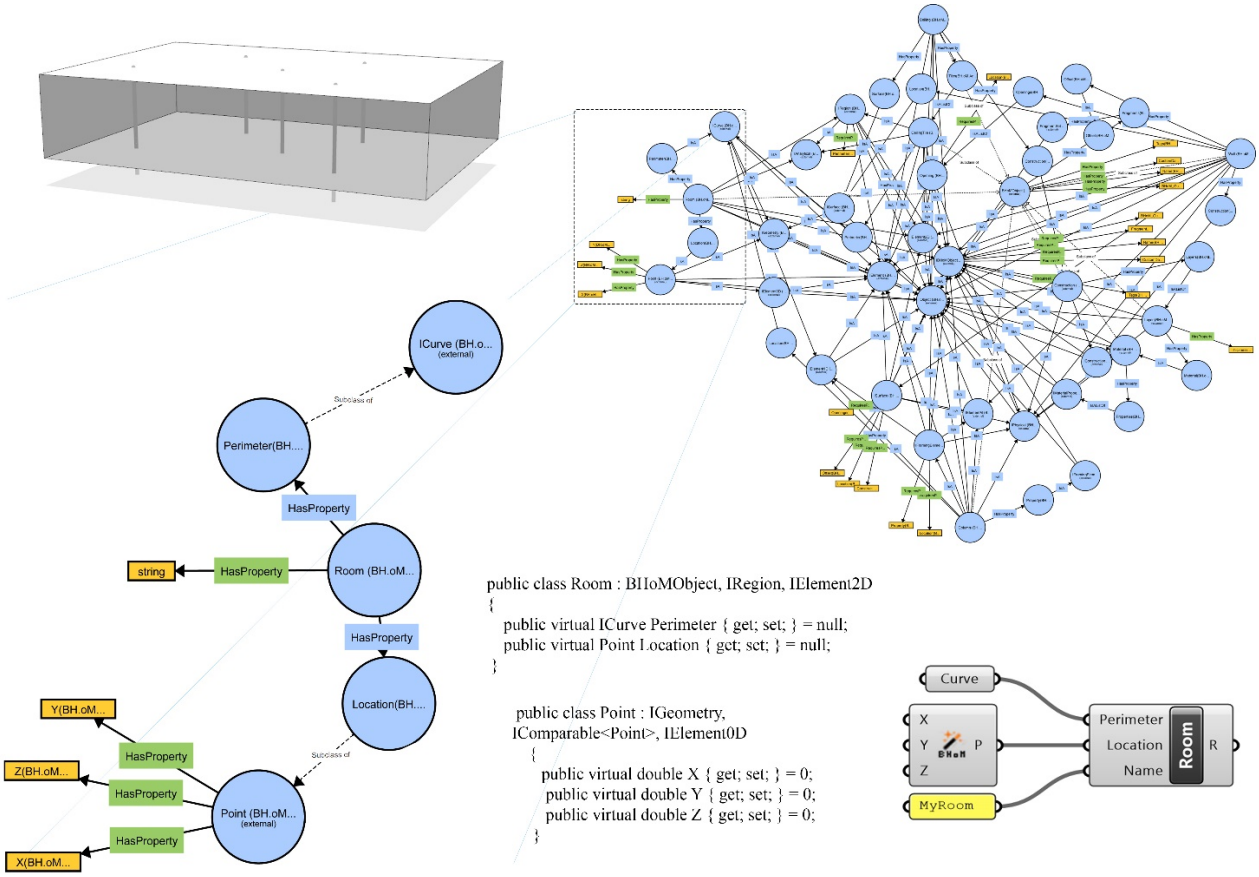


**Figure 3**: BHoM RDF building demonstrator, with a zoom on the Room class.

## 5. Conclusion

This paper presents BHoM's RDF prototypes as a potential solution to the interoperability problem of multidisciplinary design in AEC. The method section presents a mapping approach of the BHoM oM to Semantic Web standards. The results section demonstrates a procedural generation of ontologies from BHoM oM types, resulting in customized ontologies. The RDF prototype results show that further refinement of the ontology classes, properties and data types is needed. Future work will focus on refining BHoM's terminological translation that RDF_Toolkit generates, e.g. refining relations, developing alignment modules between domain-specific ontologies, modularising concept-specific ontologies based on building systems, etc.

Further work will also consider BHoM's assertional translation, which follows with the questions how we can scale this project to a framework that allows building ontologies within Grasshopper 3D. Such work makes it feasible to represent any oM model in a knowledge graph. BHoM knowledge graphs will serve to validate assumptions across distributed multidisciplinary design models. *RDF_prototypes* Toolkit should include features that allow the procedural generation of ontologies from any supported *BHoM_UI* and for any given set of BHoM oMs. Users should be able to define inter-type relations to describe ontological connections between different types and their properties.

Further work also will consider alignment modules to existing AEC ontologies such as Building Topology Ontology (BOT) and ifcOWL.

We expect that these efforts will ultimately contribute to an increased use of knowledge graphs in the AEC industry, which would address interoperability problems without file conversions and improve decision-making via inferential reasoning.

## 6. Acknowledgements

Partially supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2120/1 – 390831618 and COFFEE - STA 572_15-2.

## 7. References

[1] J. Werbrouck, P. Pauwels, J. Beetz, and E. Mannens, *Data Patterns for the Organisation of Federated Linked Building Data*. 2021.

[2] J. R. Haymaker and C. K. M. Fischer, "A methodology to plan, communicate and control multidisciplinary design processes," 2005.

[3] D. Elshani, T. Wortmann, and S. Staab, "Towards Better Co-Design with Disciplinary Ontologies: Review and Evaluation of Data Interoperability in the AEC Industry.," 2022.

[4] J. Knippers, C. Kropp, A. Menges, O. Sawodny, and D. Weiskopf, "Integrative computational design and construction: Rethinking architecture digitally," *Civil Engineering Design*, vol. 3, no. 4, pp. 123–135, Sep. 2021, doi: 10.1002/cend.202100027.

[5] J. F. Tchouanguem Djuedja, F. H. Abanda, B. Kamsu-Foguem, P. Pauwels, C. Magniont, and M. H. Karray, "An integrated Linked Building Data system: AEC industry case," *Advances in Engineering Software*, vol. 152, p. 102930, Feb. 2021, doi: 10.1016/j.advengsoft.2020.102930.

[6] buildingSMART, "Industry Foundation Classes. bSI Standards," n.d. http://www.buildingsmart.com

[7] B. Toth, P. Janssen, R. Stouffs, A. Chaszar, and S. Boeykens, "Custom digital workflows: a new framework for design analysis integration," *International Journal of Architectural Computing*, vol. 10, pp. 481–500, Dec. 2012.

[8] "BHoM. The Buildings and Habitats object Model," *https://bhom.xyz/*, n.d.

[9] P. Pauwels, Dr. K. Mcglinn, S. Törmä, and J. Beetz, "Linked Data," in *Building Information Modeling: Technology Foundations and Industry Practice*, 2018, pp. 181–197. doi: 10.1007/978-3-319-92862-3_10.

[10] P. Poinet, D. Stefanescu, and E. Papadonikolaki, “Collaborative Workflows and Version Control Through Open-Source and Distributed Common Data Environment,” in *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering*, vol. 98, E. Toledo Santos and S. Scheer, Eds. Cham: Springer International Publishing, 2021, pp. 228–247. doi: 10.1007/978-3-030-51295-8_18.
[11] A. Hogan *et al.*, “Knowledge Graphs,” *Synthesis Lectures on Data, Semantics, and Knowledge*, vol. 12, no. 2, pp. 1–257, Nov. 2021, doi: 10.2200/S01125ED1V01Y202109DSK022.
[12] P. Pauwels, A. Costin, and M. H. Rasmussen, “Knowledge Graphs and Linked Data for the Built Environment,” in *Industry 4.0 for the Built Environment*, vol. 20, M. Bolpagni, R. Gavina, and D. Ribeiro, Eds. Cham: Springer International Publishing, 2022, pp. 157–183. doi: 10.1007/978-3-030-82430-3_7.
[13] S. Törmä and N. V. Hoang, “DRUMBEAT platform—a web of building data implementation with backlinking,” in *eWork and eBusiness in Architecture, Engineering and Construction*, 1st Edition., 2016, p. 9.
[14] P. Poinet and A. Fisher, “Computational Extensibility and Mass Participation in Design,” 2020, pp. 40–47. doi: 10.2307/j.ctv13xprf6.9.
[15] A. Kay, “A Few Thoughts About ‘OOP,’” n.d. https://d1b10bmlvqabco.cloudfront.net/attach/i4g9dayv6xc4gn/gsmlulvh9ga/i61imjtf6ryn/kayoop.pdf
[16] T. Berners-Lee, “Note on Linked Data,” 2006. https://www.w3.org/DesignIssues/LinkedData.html
[17] L. Ehrlinger and W. Wöß, “Towards a Definition of Knowledge Graphs,” Sep. 2016.
[18] D. Brickley and R. V. Guha, “RDF Schema 1.1.” 2014. [Online]. Available: https://www.w3.org/TR/2014/REC-rdf-schema-20140225/
[19] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, “OWL 2 Web Ontology Language Primer (Second Edition),” *W3C Recommendation*, 2021. https://www.w3.org/TR/owl2-primer/
[20] L. Aw, L. Livermore, and I. Kaplan, *A Semantic Graph Query Language A Semantic Graph Query Language 1*. 2006. doi: 10.13140/2.1.3429.0568.