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

Diellza Elshani¹, Alessio Lombardi², Al Fisher², Steffen Staab³,⁴ Daniel Hernández³ and Thomas Wortmann¹

¹) Chair for Computing in Architecture (CA), Institute for Computational Design and Construction (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, Universitätstraße 38, Stuttgart, 70174, Germany
⁴) Electronics and Computer Science, University of Southampton, University Road, Southampton, SO17 1BJ, United Kingdom
Outline

- **Introduction**
- **Background**
  - BHoM’s Interpretation of OOP,
  - Comparison of BHoM and Semantic Web standards;
- **Methods**
- **Results**
- **Conclusion**
Introduction

• The building industry requires multidisciplinary solutions

• Data dictionaries should be editable and extendable

• Federated Interoperability

• Semantic Web standards as an open, decentralized alternative to the existing centralized and file-based BIM approaches of storing and sharing data
Background: BHoM’s Interpretation of OOP

BHoM takes an approach that often departs from conventional OOP principles by:

1. Separating functionality from types
2. Focusing on ontologically meaningful interfaces.
Background: BHoM’s Interpretation of OOP

1) 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.

2) 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.
Background: BHoM ‘s Interpretation of OOP

1) 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.

2) 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.
### Background: Comparison of BHoM and Semantic Web standards

<table>
<thead>
<tr>
<th></th>
<th>Identifiers</th>
<th>Database Model</th>
<th>Data Schema</th>
<th>Data Exchange Format</th>
<th>Querying</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Semantic Web</strong></td>
<td>URI</td>
<td>RDF</td>
<td>RDFS, OWL, SHACL</td>
<td>TTL, N-Triples, JSON-LD, RDF/XML</td>
<td>SPARQL</td>
</tr>
<tr>
<td><strong>BHoM</strong></td>
<td>GUID</td>
<td>Object-Oriented</td>
<td>BHoM Namespaces and BHoM classes</td>
<td>JSON</td>
<td>BHoM_Engine Query and MongoDB</td>
</tr>
</tbody>
</table>
Method

Object-Oriented Database Model

Graph-Based Database Model

BHoM
#open source

https://github.com/BHoM/RDF_Prototypes
Method

All BHoM Objects become owl:Classes

Inheritance between classes results with new owl:classes as subclass of the initial object. Therefore they become rdfs:subClassOf the (super)class.

Interface implementations results with new owl:classes as subclass of the initial object. Therefore they become rdfs:subClassOf the (super)class.

Properties of the class become either owl:Classes, and get connected with new owl:ObjectProperties, or become rdfs:Literal and get connected with owl:DatatypeProperties
## Method

<table>
<thead>
<tr>
<th>BHoM component</th>
<th>Semantic Web equivalent</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept identifier</td>
<td>URI</td>
<td>The original BHoM class URI.</td>
</tr>
<tr>
<td>GUID - Instance identifier</td>
<td>URI</td>
<td>URI built on a domain name, concatenated with the GUID of BHoM instances.</td>
</tr>
<tr>
<td>Object class</td>
<td>an owl:Class, e.g. bhm:PhysicalElements Column</td>
<td>An exception to this rule are BHoM Enum types.</td>
</tr>
<tr>
<td>Class inheritance relation</td>
<td>rdf:s:subClassOf</td>
<td>In BHoM Interfaces are ubiquitous and are often used to represent concepts in a taxonomic manner.</td>
</tr>
<tr>
<td>Interface (e.g. IEIdentifier)</td>
<td>bhm:IEIdentifier a owl:Class</td>
<td></td>
</tr>
<tr>
<td>Interface implementation relation</td>
<td>rdf:s:subClassOf</td>
<td></td>
</tr>
<tr>
<td>Class property</td>
<td>Type owl:Class, or rdfs:Literal</td>
<td>If the property is a class with further properties it becomes an owl:Class, else rdfs:Literal</td>
</tr>
<tr>
<td>Property relation</td>
<td>bhm:hasProperty</td>
<td>A new property named bhm:hasProperty is introduced</td>
</tr>
<tr>
<td>not implement in BHoM</td>
<td>Type owl:ObjectProperty or owl:DatatypeProperty</td>
<td></td>
</tr>
<tr>
<td>Datatypes</td>
<td>xsd:rdfs:Datatype</td>
<td>System.String are represented with xsd:string; BHoM types that do not fall in any other category are treated as Literals.</td>
</tr>
<tr>
<td>Lists</td>
<td>Type rdf:Seq</td>
<td>To simplify serialization rdf:Seq is proposed instead of rdf:List.</td>
</tr>
<tr>
<td>IEnumerables and HashSets</td>
<td>Type rdf:Bag</td>
<td>Unordered collections, since they do not possess ordering property.</td>
</tr>
<tr>
<td>Dictionaries</td>
<td>Type rdf:Bag of owl:classes</td>
<td>Dictionaries are unordered collections of Types. A named graph is used to represent a set of tuples (Key:Value). rdf:Alt doesn't have the ordering property of 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.</td>
</tr>
<tr>
<td>Enums</td>
<td>Type rdf:Alt</td>
<td></td>
</tr>
</tbody>
</table>

#open source

[https://github.com/BHoM/RDF_Prototypes](https://github.com/BHoM/RDF_Prototypes)
namespace BHoM.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>();
    }
}
BHoM RDF Timber Column
Results: namespace ontology
Institute for Computational Design and Construction

Results: Building Demonstrator
Results: Building Demonstrator

Refine the ontology

Refine mapping of datatypes

Integrate this conversion in design tools.
Conclusion and future work

- BHoM’s assertional translation ABox
- BHoM Engine methods translation to RBox ontological rules

Knowledge Graph

**Tbox: terminology component**
- introduces the terminology, i.e., the vocabulary of an application domain; defines the classes and relations, ontology

**Abox: assertion component**
- contains assertions about individuals in the Tbox vocabulary

**RBox: rules**
- contains rules about data
BHoM uses primitive data types to represent geometry
Allows adding semantic information to objects
Come join us in Stuttgart!

We are looking for a Computer Science Ph.D. Student to work in Knowledge Graphs and Semantic Web.

BE better bim: KNOWLEDGE GRAPHS FOR CO-DESIGNING BUILDINGS.

28 May 2022

- artificial intelligence, - advanced computation, - computational design | - online workshop |

English

Description:

Required Skills: Intermediate Grasshopper, not required but preferred: basic understanding of C#, RDF and description logics

Required Software: Rhino, Grasshopper, Slack

Maximum number of participating students: 40

https://digitalfutures.international/workshop/better-bim-knowledge-graphs-for-co-designing-buildings/

Online

27 June – 01 July

Workshop leaders:

- Žilinskas Rudiškas: research associate at the chair for Computing in Architecture (CAI) Institute for Computational Design and Construction (IC), University of Stuttgart

- Prof. Dr. Stefano Sandri: Dean of the School of Architecture, University of Stuttgart

- Prof. Dr. Thomas Wirthmann: Chair for Computing in Architecture (CAI) Institute for Computational Design and Construction (IC), University of Stuttgart

- Prof. Dr. Markus Hoeppli: Head of the Institute for Informatics and Distributed Systems (IDS), University of Stuttgart

- Prof. Dr. Stefano Sandri: Chair for Computing in Architecture (CAI) Institute for Computational Design and Construction (IC), University of Stuttgart