BHoM Abstract Model
Daniel Hernandez
March 23, 2022

This document proposes an abstract model for the Building Habitat object Model (BHoM) and describes a default mapping from BHoM to OWL using the proposed model.

1 BHoM Abstract Model

The BHoM data model is implemented in the programming language C# as a set of classes and interfaces. For simplicity, we do not distinguish between classes and interfaces. Hence, we assume a finite set $\mathbf{C}$, called the *set of classes*, which represents BHoM classes and interfaces. The BHoM data model includes standard C# types (e.g., numbers and strings) and classes that do not belong to BHoM. We therefore assume a finite set $\mathbf{T}$, called the *set of data types*, which represents standard C# types and non-BHoM classes and interfaces. Finally, we assume a finite set $\mathbf{P}$, which represents the properties of the BHoM classes and interfaces. We assume that these three sets $\mathbf{C}$, $\mathbf{T}$, and $\mathbf{P}$ are pairwise disjoint.

An essential aspect of the BHoM model is that its elements are described differently depending on the designer's discipline. The following examples illustrate this aspect showing a BHoM class having different attributes in structural and acoustic design.

Example 1. *In the structural design, a panel is a planar surface defined by a list of planar edges (over the external contour), a list of openings that are coplanar with the contour of the panel, a property definition (that describes the thickness and the material of the panel, and the angle. Panels are modeled using the BHoM class* Panel *with the attributes:*

1. ExternalEdges *of type* list(Edge)*,*
2. Openings *of type* list(Opening)*,*
3. Property *of type* ISurfaceProperty*,*
4. OrientationAngle *of type* double*.*

Elements Panel*,* Edge*,* Opening*, and* ISurfaceProperty *are classes*[1] *(i.e., belong to the set* $\mathbf{C}$*); element* double *is a data type (i.e., belong to set* $\mathbf{T}$*); and elements* ExternalEdges*,* Openings*,* Property*, and* OrientationAngle *are properties (i.e., belong to* $\mathbf{P}$*). The word* list *modifies the type of the property to indicate that the type is not an atomic value, but a list of elements of the specified type. For instance, the value for the property* ExternalEdges *has to be a list of elements of the type* Edge*.*

[1]The element ISurfaceProperty is actually a C# interface. As we already mentioned this distinction is not relevant for this abstract model.

Example 2. *In acoustic design, a panel is defined by a mesh, an identifier, and a dictionary that maps frecuencies to numbers. Panels are modeled using the BHoM class* Panel *with the attributes:*

1. Surface *of type* Mesh*,*
2. PanelID *of type* int*,*
3. R *of type* dict(Frequency, double)*.*

Elements Panel *and* Mesh *belong to set* **C***; elements* Frequency*,* int*, and* double *belong to set* **T***, and elements* Surface*,* PanelID*, and* R *belong to* **P***. The word* dict *is used to indicate that the type is not atomic, but a dictionary.*

Motivation. Examples 1 and 2 show that concept Panel has different properties in each discipline. Since designers of the building structure do not need information regarding the building acoustic (and vice versa), irrelevant information is hidden for them. However, designers from both disciplines need to interchange information to design and construct a building. In the current workflow, designers translate their designs from one discipline (e.g., structure) to another (e.g., acoustic). In the BHoM abstract model, a design consists of a set of objects of one discipline. In the translation from the structure to the acoustic discipline, an object representing a panel losses its structural attributes (e.g., ExternalEdges) and acquires new properties (e.g., R). Furthermore, the result is not the same object since it changes its identity (i.e., is copied with a different object ID), and a heuristic is needed to fill out the missing values. The whole design of the building is a set of disciplinary design models where there is no explicit relationship between objects representing the same entity in the building (e.g., the same panel) and no information about how the missing properties were filled. These limitations motivate the translation of BHoM models to OWL. The main advantage of this translation is to provide disciplinary and integrated views of the design. In our translation to OWL, we represent a disciplinary view as a subset of the RDF dataset whose scope is limited to the objects in the discipline, whereas the integrated view is the merge of the disciplinary views and the following metadata:

1. *Provenance relations among design objects.* There are multiple reasons to derive design objects. Examples are the translation above between disciplines and updates in the data in the same discipline. The W3C recommends the PROV-O ontology to annotate artifacts with their provenance.
2. *Flexible data scope.* In the current BHoM system, objects are grouped in disjoint sets where each set defines a building design. Objects in the structural design are not beyond the scope of structural design, and objects of acoustic design are not in the scope of structural design. In an RDF dataset integrating all building designs, elements can be organized more flexibly. For instance, we can see an update of a building design as the act of removing and adding elements. Removed elements are not in the

scope of the new design, but preserved elements are in both. Objects can be beyond the scope of design disciplines. For instance, a constraint C in the acoustic design may not be satisfied in the structural design, leading to an update in the structural design. Hence, the constraint is beyond the scope of acoustic design. Furthermore, constraint C is valid in the new structural design but not in the old one. This flexibility on the validity of a constraint illustrates the need to annotate scopes more flexibly. RDF allows annotating data by using multiple reification schemes and named graphs.

3. *Current and outdated designs.* It is desirable that designers focus on the current design and that the outdated designs be hidden from their working view. As we already mentioned, designs are updated to satisfy constraints. Hence, an outdated design can be reconsidered if a constraint is no longer be valid. To this end, the metadata can allow identifying the data that have to be in the current view and old data that may be reconsidered.

4. *The identity of objects.* Currently, a new panel object is created when a design is translated from the structural to the acoustic discipline. Similarly, a panel can be updated, changing its properties. We have to answer whether the old and the new panels are the same panel (i.e., with a single identity). If we do not consider both panels the same, we must add the corresponding metadata to indicate that the new panel is derived from the old panel.

1.1 Disciplinary schema

This subsection defines the BHoM abstract model restricted to a discipline. Intuitively, a disciplinary schema consists of class definitions, a relation of hierarchy among these classes, and properties belonging to class definitions. Each property is constrained by a type, which indicates the values objects can have for this property.

Definition 1. *Given a set of classes $C \subset \mathbf{C}$, the set of* BHoM types *over C, denoted* $\mathrm{types}(C)$*, is the minimal set defined recursively as follows. If $t \in C \cup \mathbf{T}$ then $t \in \mathrm{types}(C)$. If $t_1, t_2 \in \mathrm{types}(C)$ then $\mathrm{list}(t_1) \in \mathrm{types}(C)$ and $\mathrm{dict}(t_1, t_2) \in \mathrm{types}(C)$.*

Example 3. *Consider both definitions of class* Panel *in Examples 1 and 2, and the set $C = \{\mathsf{ISurfaceProperty}, \mathsf{Edge}\}$.* ISurfaceProperty *is a BHoM type over C because* $\mathsf{ISurfaceProperty} \in C$*,* double *is a BHoM type over C because* $\mathsf{double} \in \mathbf{T}$*,* $\mathrm{list}(\mathsf{Edge})$ *is a BHoM type over C because* $\mathsf{Edge} \in C$*, and* $\mathrm{dict}(\mathsf{Frecuency}, \mathsf{double})$ *is a BHoM type over C because* $\mathsf{Frecuency} \in \mathbf{T}$ *and* $\mathsf{double} \in \mathbf{T}$

Alternatively, we could have defined types over $\mathbf{C}$. Definition 1 restrict types over a subset C because we want to define types for a specific discipline. We next state (Definition 2) that the schema of discipline is restricted to a finite set of classes.

Definition 2. *A* BHoM disciplinary schema S *is a quin* $(C, P, \sqsubseteq, \text{dom}, \text{range})$ *where:*

1. *C and P are finite subsets of $\mathbf{C}$ and $\mathbf{P}$, respectively,*
2. *$\sqsubseteq$ is a partial order in $\mathbf{C}$,*
3. $\text{dom} : \mathbf{P} \to C$ *is a function called the* property domain*,*
4. $\text{range} : \mathbf{P} \to \text{Types}(C)$ *is a function called the* property range*.*

Given a disciplinary schema $S = (c, P, \sqsubseteq, \text{dom}, \text{range})$, a class $c \in C$ and a property $p \in P$, we said that property p is a property *of class c in the disciplinary schema S, denoted $c.p \in S$, if there exists a class $c' \in C$ such that $\text{dom}(p) = c'$ and $c \sqsubseteq c'$. We write* $\text{properties}(c, S)$ *to denote the set $\{p : c.p \in S\}$.*

Example 4. *Another example.*

In C# it is possible to define two classes with properties with the same name and different data types. This is precluded by Definition 2. Indeed, it is not difficult to see that if two different classes c_1 and c_2 have the same property p in a disciplinary schema S (i.e., $c_1.p \in S$ and $c_2.p \in S$), then there exists a class c such that $c_1 \sqsubseteq c$ and $c_2 \sqsubseteq c$ and $c.p \in S$.

We next define the databases that satisfy a BHoM disciplinary schema.

Definition 3. *A* database *over a disciplinary schema $S = (C, P, \sqsubseteq, \text{dom}, \text{range})$ is a triple $(O, \text{type}, \text{value})$ where:*

1. *O is a finite set disjoint with $\mathbf{C}$, $\mathbf{T}$, $\mathbf{P}$, called the* set of objects*,*
2. $\text{type} : O \to C \cup \mathbf{T}$ *is a function,*
3. value *is a partial function such that* $\text{dom}(\text{value}) \subseteq \{(o, p) : \text{type}(o).p \in S\}$, *and if* $\text{value}(o, p)$ *is defined then:*
   (a) $\text{type}(\text{value}(o, p)) \sqsubseteq \text{range}(p)$ *if* $\text{range}(p) \in \mathbf{C}$,
   (b) $\text{type}(\text{value}(o, p)) = \text{range}(p)$ *if* $\text{range}(p) \in \mathbf{T}$.

Example 5. *To do*

1.2 Multidisciplinary schema

So far, we have presented an abstract model for a disciplinary designs. This section presents the model for multidisciplinary designs. This section is in progress.

2 Default mapping from BHoM to OWL

This section presents the translation from a multidisciplinary schema. We assume the following values to be used as parameters for the translation:

- For each element $t \in \mathbf{T}$ we assume an URI t.uri. For example, if t is the C# int type then t.uri = `xsd:integer`.
- For each element x (e.g., design discipline, class, or property) we assume a name (denoted x.name) and a label (denoted x.name). For example, if x is the class RigidLink then S.name = `RigidLink` and S.label = `Rigid Link`. The RDF convention is that names of properties are written in lower cammel case, and names of disciplines and classes in upper cammel case.

Then the OWL ontology for a BHoM multidisciplinary schema is the result of the following translation tasks. Each of these tasks is consists of a description followed by the Turtle code returned. Elements enclosed in curly braces represent parameters of the returned code.

2.1 Terminological translation

1. We assume a base URI and the prefix `bhom` to be used on the definition of class and property URIs.

   @prefix bhom: <http://bhom.org/>

   ```
   @base <http://bhom.org/> .
   @prefix bhom:  <> .
   ```

2. For each disciplinary schema S, we define prefixes for classes, object properties, and datatype properties. By using different prefixes we avoid naming clashes between names of classes and properties of different disciplines. Recall that, for instance, the class name Panel is used in the structural and in acoustic disciplines.

   ```
   @prefix classes{S.name}:  <classes/{S.name}> .
   @prefix properties{S.name}:  <properties/{S.name}> .
   ```

Everything that is not a BHoMObject is a DatatypeProperty

3. We add some general information of the BHoM ontology. We define the class `bhom:element`, and the properties `bhom:hasObjectPropertyValue` and `bhom:hasDatatypePropertyValue` to represent the class for all BHoM elements, and the properties that range to objects and datatypes, respectively.

   = a property related to another BHoM Object

   ```
   bhom:Element rdf:type owl:Class ;
        rdfs:label "BHoM Element"@en .
   bhom:hasObjectPropertyValue rdf:type owl:ObjectProperty ;
        rdfs:label "BHoM Object Property" ;
        rdfs:domain bhom:Element ;
        rdfs:range bhom:Element .
   bhom:hasDatatypePropertyValue rdf:type owl:DatatypeProperty ;
        rdfs:label "BHoM Datatype Property" ;
        rdfs:domain bhom:Element .
   ```

4. BHoM classes have common data. For instance, all BHoM classes have a global unique identifier.

```
bhom:guid rdf:type owl:DatatypeProperty ;
     rdfs:label "Global Unique Identifier" ;
     rdfs:domain bhom:Class ;
     rdfs:range xsd:string .
```

5. We separate elements by disciplinary. For instance, in structural design, all classes are subclasses of `bhom:StructuralElement`, and all properties are subproperties of either `bhom:hasStructuralObjectPropertyValue` or `bhom:hasStructuralDatatypePropertyValue`. The rationale of this separation is to allow for reasoning about the discipline of elements.

```
bhom:{S.name}Element rdfs:subClassOf bhom:Element ;
     rdfs:label "{S.label} Element"@en .
bhom:has{S.name}ObjectPropertyValue
     rdfs:subPropertyOf bhom:hasObjectPropertyValue ;
     rdfs:label "{S.label} ObjectProperty"@en .
bhom:has{S.name}DatatypePropertyValue
     rdfs:subPropertyOf bhom:hasDatatypePropertyValue ;
     rdfs:label "{S.label} DatatypeProperty"@en .
```

6. For each disciplinary schema S with set of classes C and each class $c \in C$ we define a subclass of class `bhom:{S.name}Element`.

```
          acoustic    Panel                          acoustic
classes{S.name}:{c.name} rdfs:subClassOf bhom:{S.name}Element ;
     rdfs:label "{c.label}"@en .
```

7. For each disciplinary schema $S = (C, P, \sqsubseteq, \mathrm{dom}, \mathrm{range})$, class $c \in C$, and property $p \in P$ where $c.p \in S$ and $\mathrm{range}(p) \in \mathbf{T}$ define a subproperty of `bhom:has{S.name}DatatypePropertyValue`.

```
properties{S.name}:{p.name}
     rdfs:subPropertyOf bhom:has{S.name}DatatypePropertyValue ;
     rdfs:label "{p.label}"@en ;
     rdfs:domain classes{S.name}:{dom(p).name} ;
     rdfs:range {range(p).uri} .
```

8. For each disciplinary schema $S = (C, P, \sqsubseteq, \mathrm{dom}, \mathrm{range})$, class $c \in C$, and property $p \in P$ where $c.p \in S$ and $\mathrm{range}(p) \in C$ define a subproperty of `bhom:has{S.name}ObjectPropertyValue`.

```
properties{S.name}:{p.name}
     rdfs:subPropertyOf bhom:has{S.name}ObjectPropertyValue ;
     rdfs:label "{p.label}"@en ;
     rdfs:domain classes{S.name}:{dom(p).name} ;
     rdfs:range classes{S.name}:{range(p).name} .
```

9. We need rules for translating properties whose types have the form $\mathrm{list}(t)$ or $\mathrm{list}(t_1, t_2)$. These cases are not complicated, but require a little more work.

2.2 Assertional translation

So far, we have described how the classes and property definitions are translated from BHoM to OWL. This section shows how the assertional data is translated. This section is in progress.