Improving the GraphQL, JSON and RDF Representations of buildingSmart Data Dictionary

Vladimir Alexiev, Mihail Radkov, Nataliya Keberle

W3C Linked Building Data Community Group meeting, 15 May 2023

ontotext

Outline

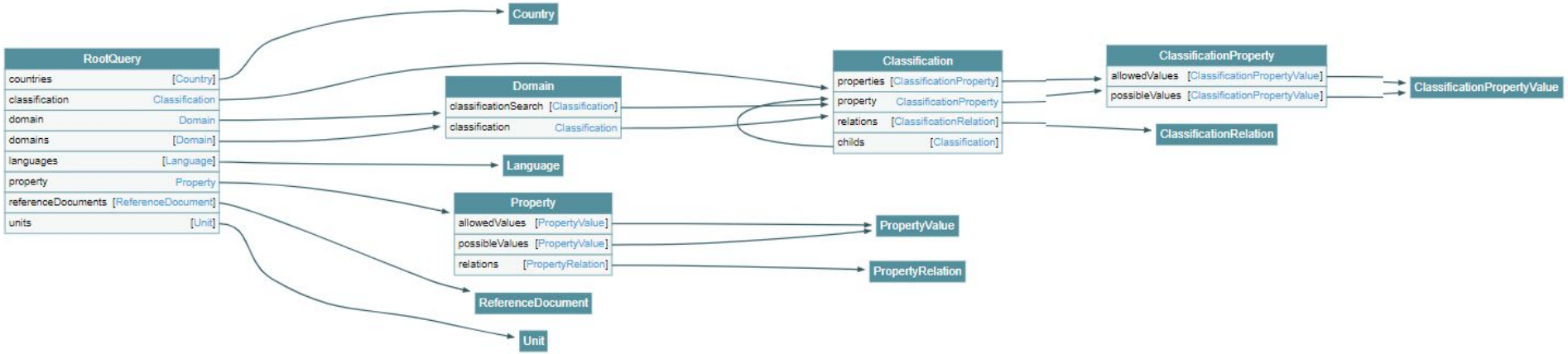
- Highlight the defects in the original GraphQL implementation of bSDD
- Overview the refactored solution proposed by Ontotext
- Overview data quality issues
- Overview the proposed improvements

BSDD GRAPHQL SCHEMA: VOYAGER

W3C Linked Building Data Community Group meeting, 15 May 2023

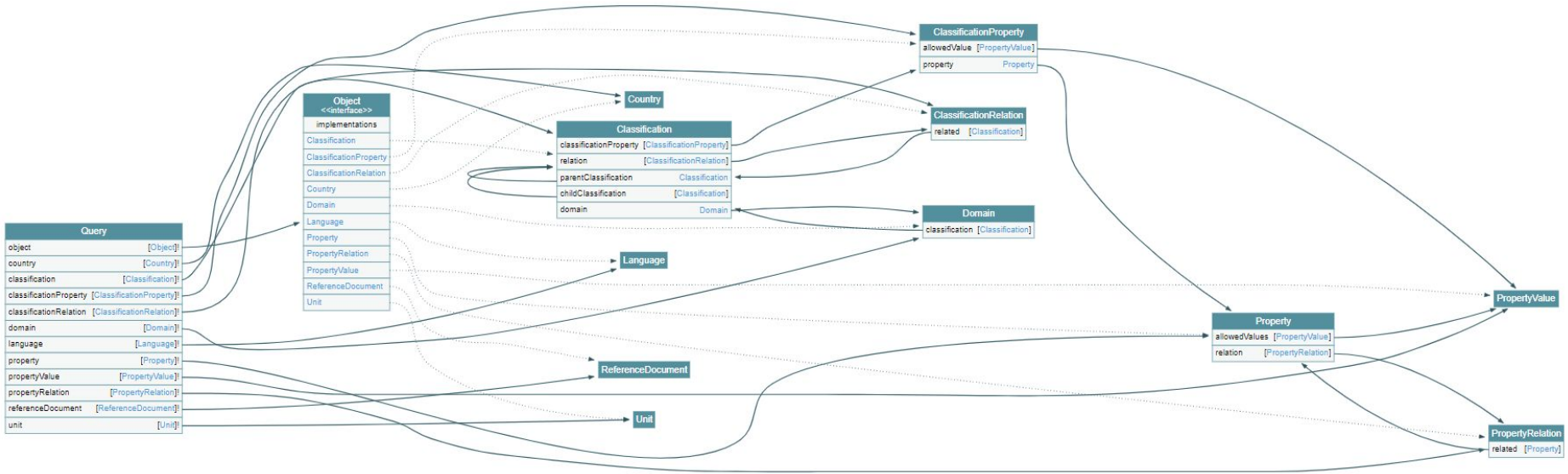
ontotext

Voyager: Original Schema



ontotext

Voyager: Refactored Schema



ontotext

Original GraphQL: Findings (1/3)

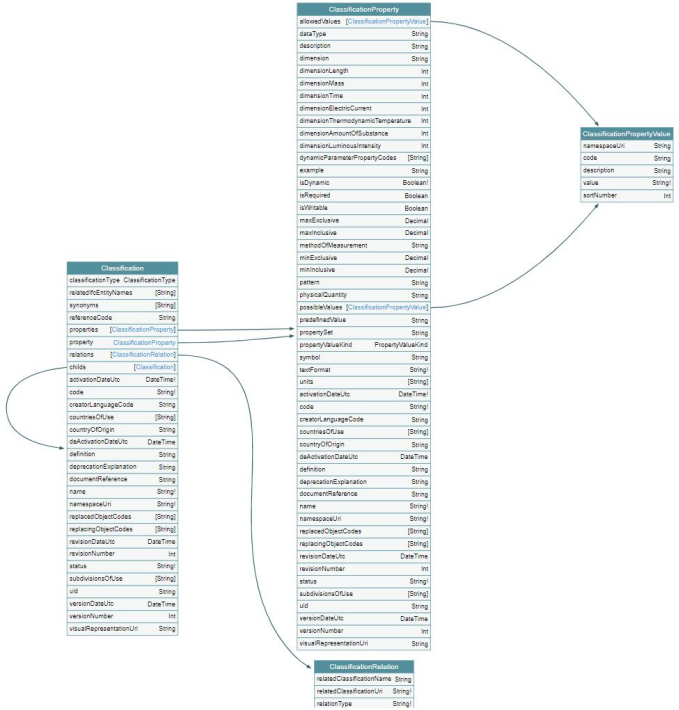
- Reference entities `ReferenceDocument, Country, Unit, Language` are disconnected from the rest of the schema
- Relation entities have only an incoming link but no outgoing link
- Many entities cannot be queried directly from the `Root`
- No backward arrows to get from a lower-level entity back to its “parent” entity
- A number of parallel arrows. GraphQL schema can use parameters to distinguish between the different uses

Original GraphQL: Findings (2/3)

At the high level of detail:

- `Property` and `ClassificationProperty` are very similar, but there's no inheritance/relation between them
- `PropertyValue` and `ClassificationPropertyValue` are exactly the same, so can be reduced to one entity

Original GraphQL: Findings (3/3)



Mixture of singular/plural in property names(*)

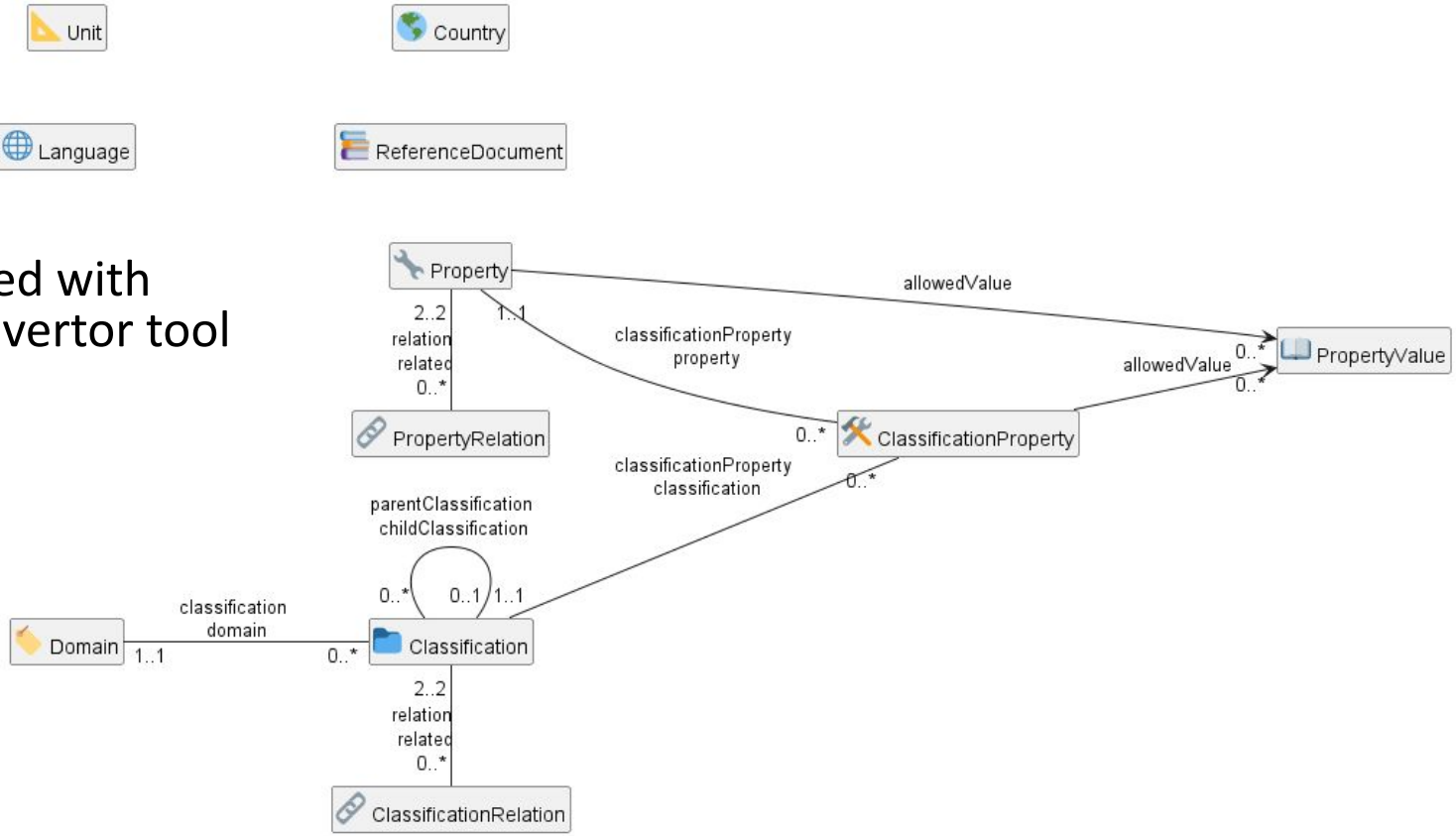
`property/properties`, `relations`, `synonyms`, `countriesOfUse`, `relatedIfcPropertyNames`, etc.

(*) - already discussing at forums.buildingsmart.org

ontotext

bSDD Refactored Schema: PlantUML

PlantUML is used with soml2puml convertor tool



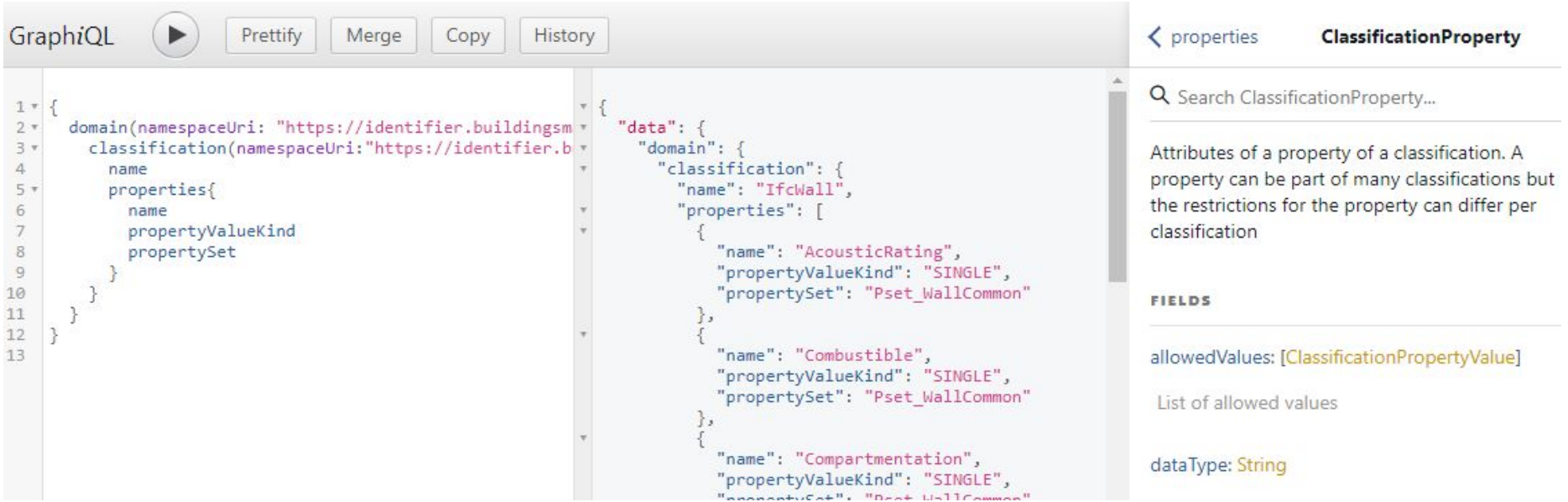
ontotext

Refactored GraphQL: Improvements

- All entities are queryable directly from the `Root`
- No parallel links, use GraphQL query parameters instead
- Pagination for bulk query results
- GraphQL syntax highlight, keyboard shortcuts, search in the query text, query response errors
- Each link is named the same as target entity
- Navigation between entities is bidirectional
- A single entity `PropertyValue` is used by both `Property` and `ClassificationProperty`
- Property names are in singular

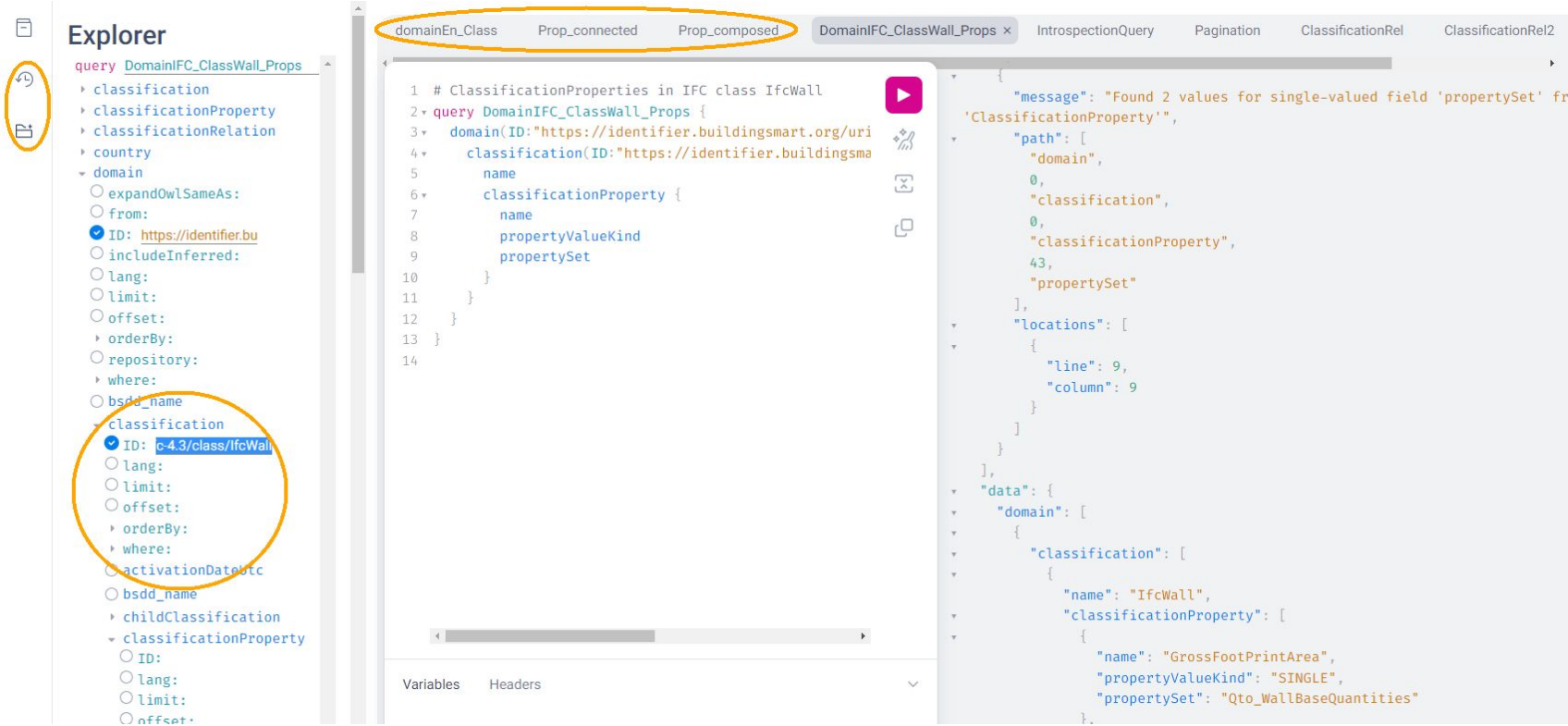
ontotext

GraphiQL: Original



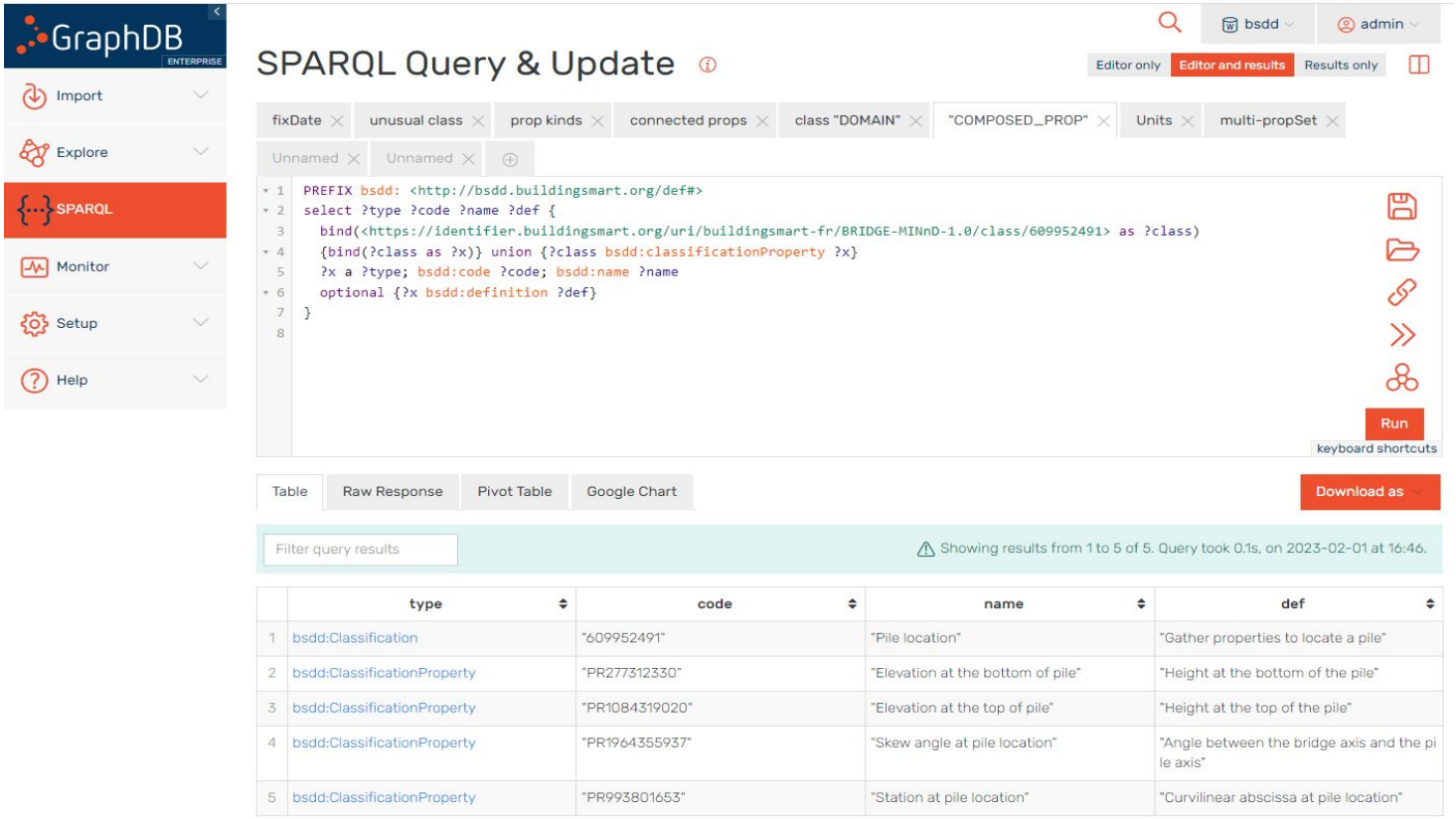
ontotext

GraphiQL: Refactored



ontotext

Refactored bSDD: SPARQL endpoint



ontotext

SUGGESTED IMPROVEMENTS

ontotext

Presentation

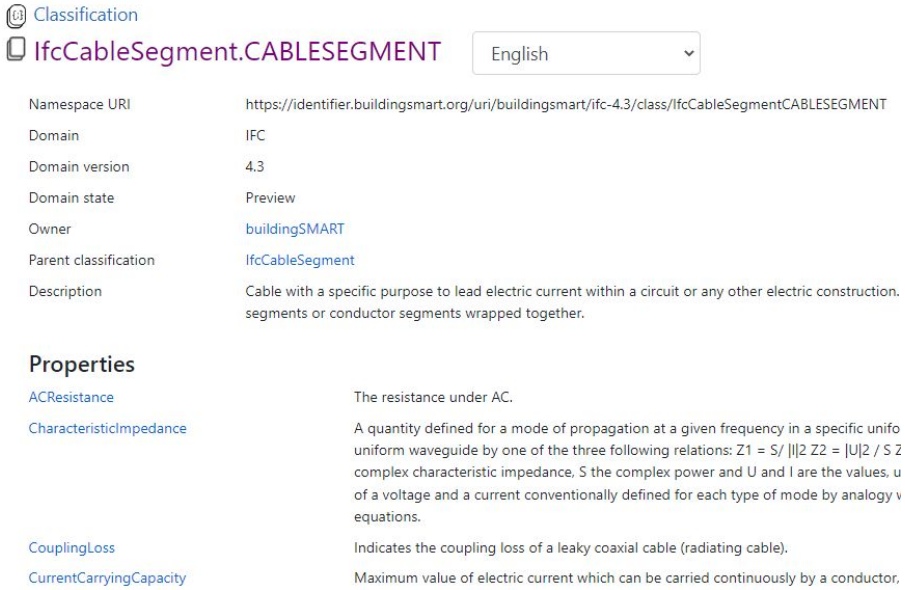
- Uniform identification for the search
- Equal data retrieved from different API
- Improve URL structure and consistency

ontotext

Uniform Identification

February 2023: **IfcCableSegment** in Web UI has URL:
https://search.bsdd.buildingsmart.org/Classification/Index/58453

May 2023: **IfcCableSegment** in Web UI has another URL:
https://search.bsdd.buildingsmart.org/Classification/Index/70992


Classification

IfcCableSegment.CABLESEGMENT

English

Namespace URI	https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.3/class/IfcCableSegmentCABLESEGMENT
Domain	IFC
Domain version	4.3
Domain state	Preview
Owner	buildingSMART
Parent classification	IfcCableSegment
Description	Cable with a specific purpose to lead electric current within a circuit or any other electric construction. segments or conductor segments wrapped together.

Properties

ACResistance	The resistance under AC.
CharacteristicImpedance	A quantity defined for a mode of propagation at a given frequency in a specific unifo uniform waveguide by one of the three following relations: Z1 = S/ |I|2 Z2 = |U|2 / S Z complex characteristic impedance, S the complex power and U and I are the values, u of a voltage and a current conventionally defined for each type of mode by analogy v equations.
CouplingLoss	Indicates the coupling loss of a leaky coaxial cable (radiating cable).
CurrentCarryingCapacity	Maximum value of electric current which can be carried continuously by a conductor,

ontotext

Uniform Identification

IfcCableSegment has also **URI assigned by a data provider**:

https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.3/class/IfcCableSegment CABLESEGMENT

CableSegment entity as displayed at the bSDD web site

Classification

IfcCableSegment.CABLESEGMENT

Namespace URI	https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.3/class/IfcCableSegmentCABLESEGMENT
Parent Namespace URI	https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.3/class/IfcCableSegment
Description	Cable with a specific purpose to lead electric current within a circuit or any other electric construction. conductor segments wrapped together.

Properties

Name	Data type
ACResistance	Real
CurrentCarryingCapacity	Real
DCResistance	Real
FunctionReliable	Boolean
HalogenProof	Boolean
HasProtectiveEarth	Boolean
InsulationVoltage	Real
MassPerLength	Real
MaximumBendingRadius	Real
MaximumCurrent	Real
MaximumOperatingTemperature	Real

ontotext

Uniform Identification

Non-unique identification violates FAIR Findability principle

F1: (Meta)data are assigned a globally unique and persistent identifier

ontotext

Equal Data Retrieved from Different API

We have compared three representations returned by the bSDD server:

- JSON from the GraphQL API
  - `https://test.bsdd.buildingsmart.org/graphiql/`
- JSON from the REST (entity) API
  - `curl https://identifier.buildingsmart.org/uri/buildingsmart/<domain>/class|prop/<name>` and
- RDF from the REST (entity) API
  - `curl -Haccept:text/turtle \ https://identifier.buildingsmart.org/uri/buildingsmart/<domain>/class|prop/<name>`

ontotext

Equal Data Retrieved from Different API

We selected entities of each class that have the **maximum number of filled fields**, and [compared the results returned by each API].

	GraphiQL UI	JSON API	RDF API	problems/comments
Classification	Sample GraphQL			property names are in CamelCase, whereas in GraphQL and JSON API they return in camelCase
activationDateUtc	present	present	present	
childs	present	absent	absent	
classificationType	present	absent	absent	
code	present	present	present	
countriesOfUse	present	present	absent	
countryOfOrigin	present	absent	absent	
creatorLanguageCode	present	absent	absent	
deActivationDateUtc	present	absent	absent	
definition	present	present	present	
deprecationExplanation	present	absent	absent	
documentReference	present	absent	absent	
domain	absent	absent	present	feild name differs in JSON vs RDF (it's better in RDF: refers to the target entity, not its URI)
domainNamespaceUri	absent	present	absent	
name	present	present	present	name="IfcWall.SOLIDWALL" include "." but there is no "." in namespaceUri and referenceCode
namespaceUri	present	present	absent	
parentClassificationReference	absent	present	absent	
properties	present	present	present	
property	present	present	present	
referenceCode	present	present	present	
relatedIfcEntityNames	present	absent	absent	
relations	present	present	present	
replacedObjectCodes	present	present	absent	
replacingObjectCodes	present	present	absent	
revisionDateUtc	present	absent	absent	some domains, eg ifc4.3, are missing this field

GraphiQL UI	https://test.bsdd.buildingsmart.org/graphiql/
JSON API	https://identifier.buildingsmart.org/uri/buildingsmart/{domain}/{class|prop}/{name}
RDF API	-Haccept:text/turtle https://identifier.buildingsmart.org/uri/buildingsmart/{domain}/{class|prop}/{name}

ontotext

Improve URL Structure and Consistency

- Almost all bSDD domain URLs now have the same structure: `https://identifier.buildingsmart.org/uri/<org>/<domain>-<version>`
- URIs can be more “hackable”, allowing users to navigate the hierarchy by pruning the URI: `https://identifier.buildingsmart.org/uri/<org>/<domain>/<version>`
- In some cases, the `<org>` is repeated in the `<domain>` part

D. Garijo and M. Poveda-Villalón, ``Best practices for implementing FAIR vocabularies and ontologies on the web,'', 2020

L. Dodds and I. Davis, ``Linked data patterns: A pattern catalogue for modelling, publishing, and consuming linked data. Linked data patterns,'' Sep. 06, 2022.

Improve URL Structure and Consistency

- In some cases, the `<org>` name doesn't quite mesh with the domain name, perhaps due to the way bSDD allocates `<org>` identifiers to bSDD contributors
  - `bim-de/DINSPEC91400`: the publisher of this spec is DIN (the German standards organization), not the `bim-de` initiative
  - `digibase/volkerwesselsbv`: bimregister.nl news from 2018 suggest that `digibase` is a new company/initaitive within Volker Wessel
  - `digibase/nen2699`: the publisher of this spec is NEN (the Netherlands standards organization), not the `digibase` company/initiative
  - `digibase/digibasebouwlagen`: perhaps the org name `digibase` should not be repeated as the prefix of the domain `bouwlagen` (building layers)

Explicate domain versions

`https://identifier.buildingsmart.org/uri/acca/ACCAtest-0.1`

can become

`https://identifier.buildingsmart.org/uri/acca/ACCAtest/0.1`

A new entity `DomainVersion` can provide linking all versions of a domain to its master `Domain` entity.

Improve URL Structure and Consistency

- Declare URLs to be `ID` and use a mandatory field `id`
  - Most GraphQL implementations call this field simply `id`, whereas bSDD uses `namespaceUri`
  - Many nodes do not have their own `namespaceUri` field, or it is not fully populated

Entity Classes vs classificationTypes

The key field `classificationType` specifies the kind of classification.

E.g., there is the classification with name **décret 2011-321 (23/03/2011)** from ATALANE/REX-OBJ-1.0 domain **and**
with classificationType="REFERENCE_DOCUMENT", that it is not in the list of ReferenceDocuments.

Why is it not a `ReferenceDocument` entity?

c	classificationType	overlaps with entity
29	"DOMAIN"	`Domain`
18	"REFERENCE_DOCUMENT"	`ReferenceDocument`

All entities should have URL

All significant classes should have `ID`, which in the case of RDF data is the node URL.

However, many bSDD classes don't have such a field:

- `Domain, Property, Classification` do have `namespaceUri`
- `Country, Language, Unit` don't have an ID but have a field (`code, isocode`) that can be used to make an `ID`, when prepended with an appropriate prefix.

URL for ClassificationProperty

- `Property` and `ClassificationProperty` are two different classes, but the latter does not have a distinct URL(*) in GraphQL and JSON.
- The same URL is overloaded to identify entities of both classes.
- `ClassificationProperty` are not returned separately by the JSON or RDF entity API, but only as part of the respective `Classification`

E.g., IfcCableSegment.CABLESEGMENT classification has **ACResistance** as a ClassificationProperty, but

```
curl
https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4.
3/class/IfcCableSegmentCABLESEGMENT/ACResistance
```

returns

```
{"":["Classification with namespace URI

'https://identifier.buildingsmart.org/uri/buildingsmart/ifc-4
.3/class/IfcCableSegmentCABLESEGMENT/ACResistance' not
found"]}
```


ontotext

MODELLING ISSUES

Unify Solutions to Model Complex Properties

The key attribute `propertyValueKind` has values COMPLEX and COMPLEX_LIST used in combination with `connectedProperties`. These key values are defined for `Property` and `ClassificationProperty`

> propertyValueKind: PropertyValueKind
>
> Indicates kind of value: Single, Range (2 values expected), List (multiple values expected), Complex (use in combination with ConnectedProperties), ComplexList

- However, `connectedPropertyCodes` is defined only for `Property`
- More importantly, these key values are never used
- `connectedProperty` is used only on **7** `Properties` (and not `ClassificationProperties`)
- Instead of using `connectedPropertyCodes` to describe complex properties, some providers have used classifications with the type COMPOSED_PROPERTY.

ontotext

Improve Modelling of Dynamic Properties

12385 `Properties` are declared with `isDynamic=true`

135250 are not.

However, the field `dynamicParameterPropertyCode` (used to compute the dynamic property) is **always** empty, so how can one know which “sub-properties” to use?

Additionally, `dynamicParameterPropertyCodes` is `String`, but should be `[Property]`, i.e. an array of `Properties`.

Improve Relations Between Entities

is a classification field (String)	should be
connectedPropertyCodes	[Property]
countriesOfUse	[Country]
countryOfOrigin	Country
creatorLanguagecode	Language
documentReference	ReferenceDocument
dynamicParameterPropertyCodes	[Property]
example	PropertyValue
languageCode	Language
predefinedValue	PropertyValue
relatedClassificationUri	Classification
relatedIfcEntityNames	a relation to some IFC Classification
relatedPropertyUri	Property
units	[Unit]

Add More Entities

bSDD includes numerous string attributes (codes or URLs) that should be converted to relations (object fields) to improve the connectedness of the bSDD GraphQL graph

is a classification field (String)	should be
`physicalQuantity`	(New) `PhysicalQuantity`
`propertySet`	(New) `PropertySet`
`subdivisionsOfUse`	(New) `CountrySubdivision`
`version`	(New) `DomainVersion`
`replaced/(-ing)ObjectCodes`	some kind of objects. Currently they are empty

Use Class Inheritance

`Property` and `ClassificationProperty`: have **46** fields in common, differ in only 5 fields:

belongs uniquely to `Property`	belongs uniquely to `ClassificationProperty`
`connectedPropertyCodes` (String)	`isRequired` (Boolean)
`relations` (PropertyRelation)	`isWritable` (Boolean)
	`predefinedValue` (String)
	`propertySet` (String)
	`symbol` (String)

Since there are *no rules* on which fields of `Property` to reuse in `ClassificationProperty`, the latter type copies most of the fields from the former

Improve `PropertyValues`

- `PropertyValue` and `ClassificationPropertyValue` are structured values with rich fields:

`code, value, namespaceUri, description, sortNumber`

- However, most structured values we've seen have only

`code, value`

- This has multiple problems:
  - Individual values have no description (`description` is not filled out)
  - Some values are described in the property definition, intermingling multiple descriptions together
  - The "standard" values NOTKNOWN, OTHER, UNSET are not described at all.
  - Values have no `namespaceUri`, precluding unique identification.

Improve `predefinedValue`

- `allowedValues` store structured values (`ClassificationPropertyValue`)
- However, their “sibling” property `predefinedValue` holds just a String, which means that even in the future, `predefinedValue` cannot be an enumeration value identified globally with a URL

Improve Multilingual Support

- bSDD is advertised as a multilingual dictionary
- In the GraphQL API, one can specify a desired language when fetching classifications and properties
- However, currently most domains are present in one language only (*unilingual*).

DATA QUALITY

ontotext

Data Quality Issues

- Leading, trailing, consecutive whitespace
- Improve physical quantities and units
- No rules on missing data
- Unicode problems
- Unresolved HTML entities
- Bad classification relations (broken links)

Implementing Improvements

We implemented a lot (but not all) of the improvements suggested above by using the following process:

- Fetch bSDD data as JSON
- Draft SOML schema
- Convert it to RDF using SPARQL Anything
- Load it to GraphDB
- Refactor the RDF using SPARQL Update

The results are available at the SPARQL endpoint and in GraphQL

ontotext

Conclusions and Future Work

Here are further ideas for improvement:

- improve bSDD ontology
- implement more radical data model refactoring to convert “strings” (countries, reference documents, etc.) into “things”
- link bSDD units of measure to QUDT ontology
- perform deeper data quality analysis using SHACL shapes generation and validation provided by Ontotext Platform Semantic Objects
- address and resolve more data quality issues, including
  - seek correlation between dimension vectors, units of measure and physical quantity,
  - parse out enumeration values from `Property/ClassificationProperty` descriptions and create corresponding `PropertyValue` lists
- make more graph visualizations
- obtain more interesting statistics using SPARQL

Acknowledgements

Funding: ACCORD project, Horizon Europe, grant #101056973

- Data: buildingSMART Data Dictionary (Leon van Berlo, Artur Tomczak, Erik Baars)
- Powered by:
  - Ontotext GraphDB
  - Ontotext Platform Semantic Objects