Welcome!
Brick Consortium Kickoff

Winter 2021

Agenda

- Intro to Brick, the Consortium, and this kickoff
- A double-click on Brick, and how it works under the hood
- The Brick Working Groups
- The Brick Roadmap
- The way forward and getting involved

What is Brick?

Brick Schema is:

- A uniform metadata schema that defines how the building data is modeled.
- Standardized semantic descriptions of the physical, logical, and virtual entities in buildings and the relationship between them.
- Focus on interoperability, not replacement of other specialized standards
- Open Source project –
  all development is on Github
- Initiated by researchers from the academic community in 2015.

A Brick Model is a digital representation of a building that adheres to the Brick schema.

What industry-wide customer problems does it solve?

- **Allows** owner of the built-environment to really own the data
- **Empowers** the customer to bring in the best-in-class third party applications; Manufacturer and service-provider can be different
- **Allows** third-party applications to work independent of the data silos
- **Provides** an integration “platform” for data from disparate data sources
- **Allows** for true “plug and play” even if you have multiple manufacturers’ systems in the building environment

Brick v1.2 and what's next

- Brick 1.2.0 released Feb 20th!
  - Main artifact: a schema defined in RDF – Brick.ttl
  - Typical open source project: BSD license, hosted on Github, community discusses on mailing list and then creates issues, developers create pull requests, software released through appropriate channels
  - Lots of contributors, thanks!
  - Future development will continue this way
- Also tools, datasets, and best practices documentation

- Issues to address moving forward
  - How do we grow the ecosystem and solve industry problems?
  - Standards exist over decades, how do we go that long?
  - How does a community fairly govern a standard?

Introducing the Brick Consortium

- The Brick Consortium, Inc. is a non-profit membership corporation whose purpose is to encourage the research and development of Brick Schema Specifications for the built environment and any supporting tooling, documentation, and best practices necessary to promote Brick.
- The consortium develops the Brick Schema Specifications as open source through member participation, addressing an important industry and societal need by helping to make data of the built environment interoperable.
- The consortium is made up of researchers, technology providers, integrators, and building operators, and will serve as an organization that can provide the long-term support necessary to maintain and enhance the Brick schema until 2040 and beyond.
- The consortium provides governance for the Brick Schema specifications and conformance testing protocols and provides tooling and a repository of reference models and canonical use cases.
- The consortium also funds the research of work related to Brick and the built environment and works to evangelize the use of the Brick schema. Governance and development of Brick Schema, tooling, conformance testing, and canonical use case and reference solution

Brick Consortium Membership

- Membership in the Brick Consortium is open to commercial entities, universities and non-profit research institutions, and individual academic researchers.
- Membership for commercial entities is available as a full member or a contributing member.
- A contributing member does not have a vote for the Steering Committee or can chair committees but is otherwise fully eligible to participate.
- Universities and non-profit institutions do not pay any membership fees. In some cases, individual academic researchers may join in lieu of their institution joining.
- Membership is not required to participate in the open source development efforts

Brick Development with the Consortium

- Schema development (e.g. the standard): open source development, just like today
  - New: members vote before a release, each member org get a vote
- Work happens in working groups, natural evolution of current effort
- The elected Technical Committee creates new work groups and coordinates between the groups

Agenda for today

- Brick in detail – Gabe Fierro, UC Berkeley
  - Review of Brick
  - What's new in Brick 1.2
  - How we develop Brick
  - Software to support Brick
- Brick Working Groups
  - Full organization structure and how to join
- The Brick Roadmap – a discussion
  - Specifics of Brick 1.2.1 and 1.3
  - Thinking bigger: moving the industry forward

Brick Overview and Refresher

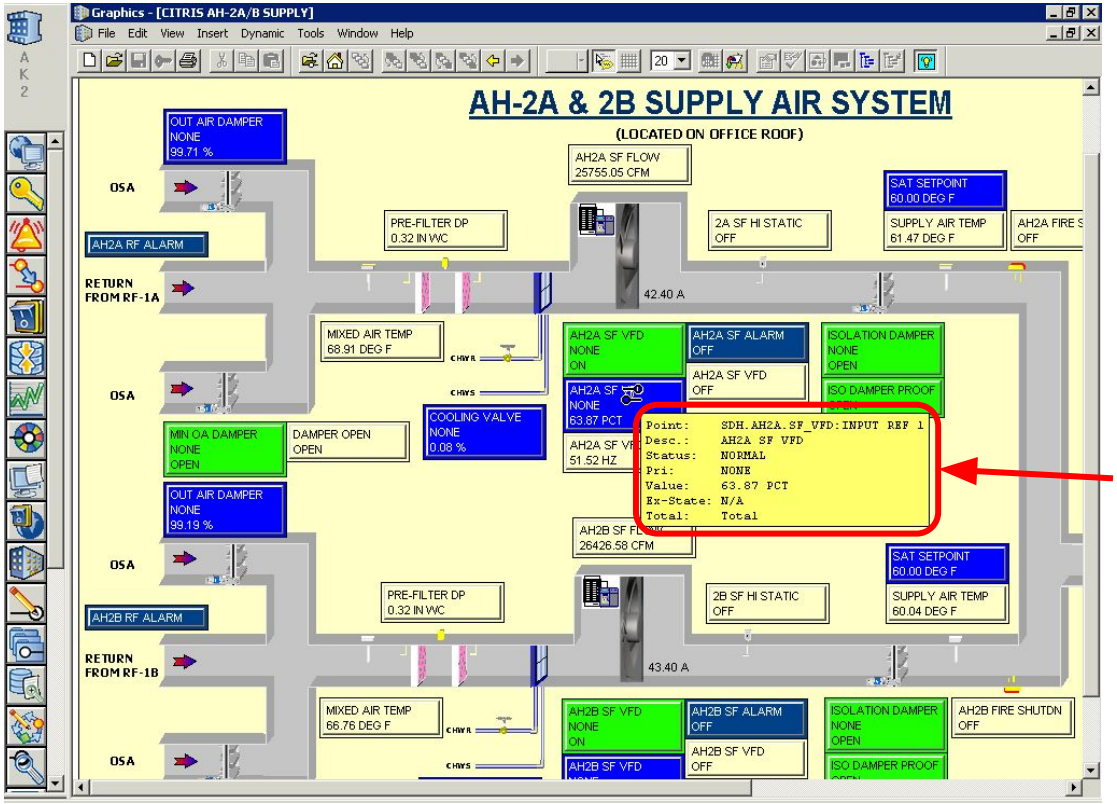
Gabe Fierro

*Working with data in buildings is **hard***

… because there is no consistent, machine-readable representation of <u>data sources</u> and their <u>context</u>

We need **semantic metadata**

State of Building Metadata



Point: a source of data from a cyberphysical system

10s or 100s of thousands of points in a typical CPS

Building Management System (BMS): the “operating system” for your building

State of Building Metadata

- Open up your **building management/automation system**, look at the point names

SODA2S14___SMK
SODA1S11___MAT
SODA3R315_RVAV
SODA3R723__ASO
SODA3R327__AGN
SODH1P02___FLT
SODA3R798__ART
SODA1R405B_ARS
SODA3R683_RVAV
SODA1R405B_ART
SODA3R311__AGN
SODH1_____L_L
SODC1SP03__FLT
SODA4R645_RVAV
SODA1R288__AGN
SODA3R419__AGN
SODA3C611__ASO
SODA2S14_P__VR
SODA4S1832_STA

AHU.AHU01.CAV1-1:DMPRPOS
AHU.AHU01.CAV1-1:HTG O
AHU.AHU01.CAV1-1:SUPFLOW
AHU.AHU01.CAV1-1:ZN T
AHU.AHU01.CAV2-1:DAT
AHU.AHU01.CAV2-1:DMPRPOS
AHU.AHU01.CAV2-1:HTG O
AHU.AHU01.CAV2-1:SUPFLOW
AHU.AHU01.CAV2-1:ZN T
AHU.AHU01.CCV
AHU.AHU01.CHWHHW.UNT:CHW FLOW
AHU.AHU01.CHWHHW.UNT:HW FLOW
AHU.AHU01.Cooling Enable
AHU.AHU01.ECM
AHU.AHU01.HP.UNT:ZN T
AHU.AHU01.HSP
AHU.AHU01.LSP
AHU.AHU01.LTD
AHU.AHU01.MAX.ZONE.DAMPER
AHU.AHU01.MAX.ZONE.HEATING
AHU.AHU01.MIN OA
AHU.AHU01.Mixed Air Damper Position
AHU.AHU01.Mixed Air Temp

Trunk.VAV2-12.OCCHTGFL
Trunk.CentralPlant.HWP2-RST
Trunk.VAV2-4.BOXHTG
Trunk.VAV2-9.SUPFLOSP
Trunk.CentralPlant.CHWP4-S
Trunk.VAV2-7.COMMONSP
Trunk.VAV1-5.SUPFLOW
Trunk.VAV2-10.S-VP
Trunk.VAV2-3.SUPFLOSP
Trunk.VVT-4.UNOCDB
Trunk.VAV2-10.BOXHTG
Trunk.VVT-5.ZN-T
Trunk.CentralPlant.HWP2-A.Alarm1
Trunk.VVT-1.ZN-T
Trunk.VAV2-8.COMMONSP
Trunk.VAV1-1.BOXMODE
Trunk.AHU-3.MA-T

- 3 different buildings/BMS/subsystems → 3 (or more) different labeling/naming schemes

Brick
https://brickschema.org

Make Working with Building Data Easier

- Most building data resides in **opaque data silos**
  - Unclear, inconsistent, hard-to-interpret labels
  - *(if you have access to it at all)*

- Existing metadata standards focus on other perspectives of the building
  - Design, construction
  - Asset management
  - Commissioning, Auditing

- Need a metadata representation designed for **data-driven building software**
  - Unlock potential of building data
  - *Preserve existing investments*

Brick Schema

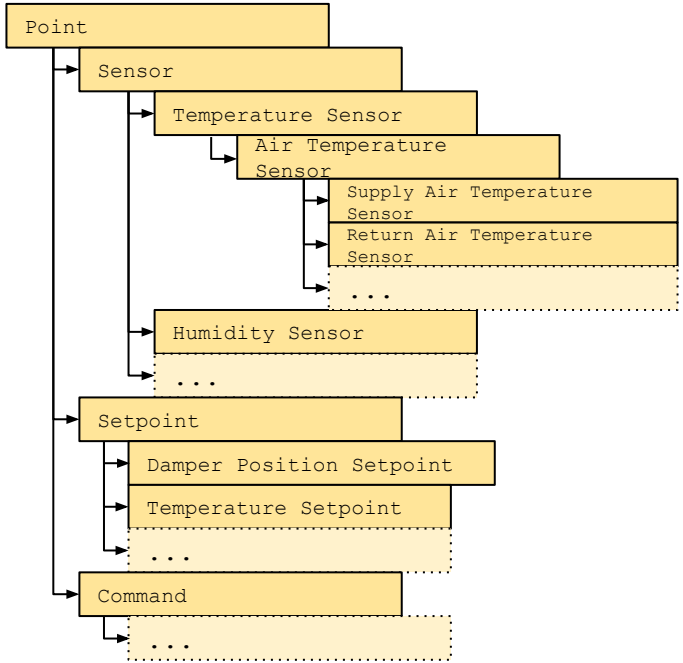
- Graph-based metadata schema for smart buildings
- Capture physical, logical, virtual **entities** in buildings using a **class hierarchy**
- Capture the necessary **relationships** between them
- Use Brick to describe timeseries data and its context

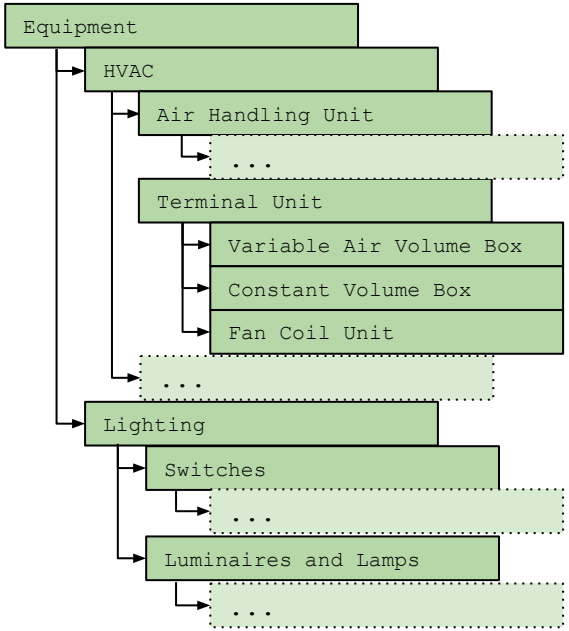
Brick

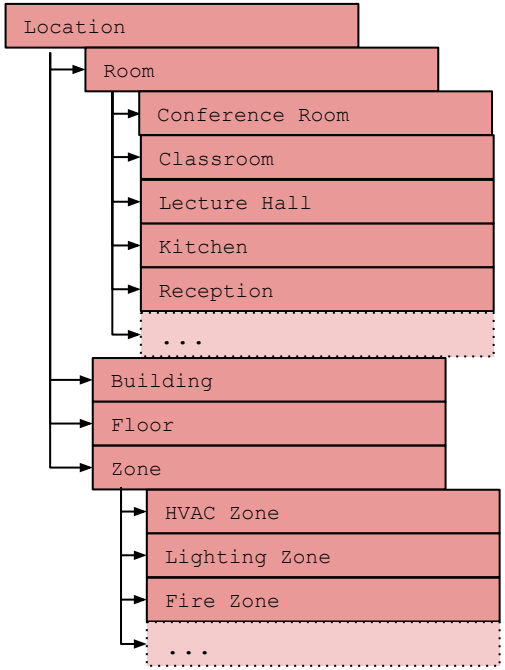
https://brickschema.org

Berkeley UNIVERSITY OF CALIFORNIA · Carnegie Mellon University · Johnson Controls · IBM · UC San Diego · UCLA · SDU · University of Virginia

Brick Classes



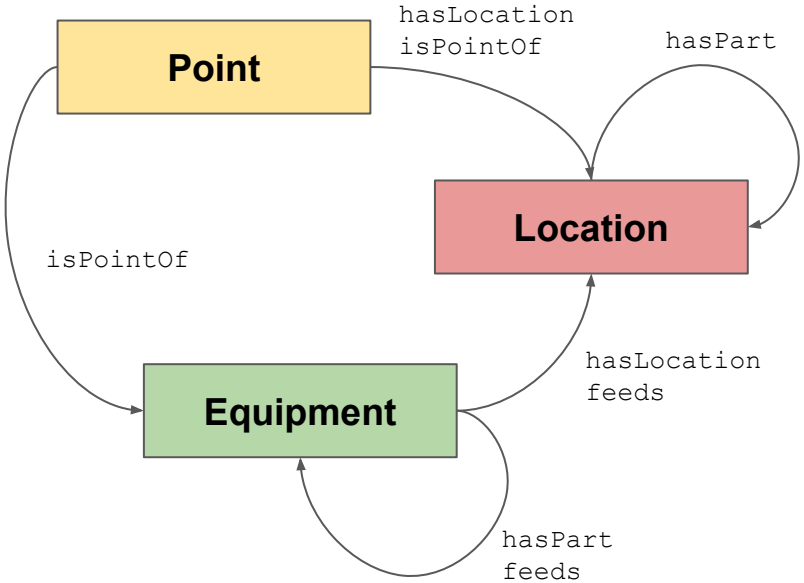


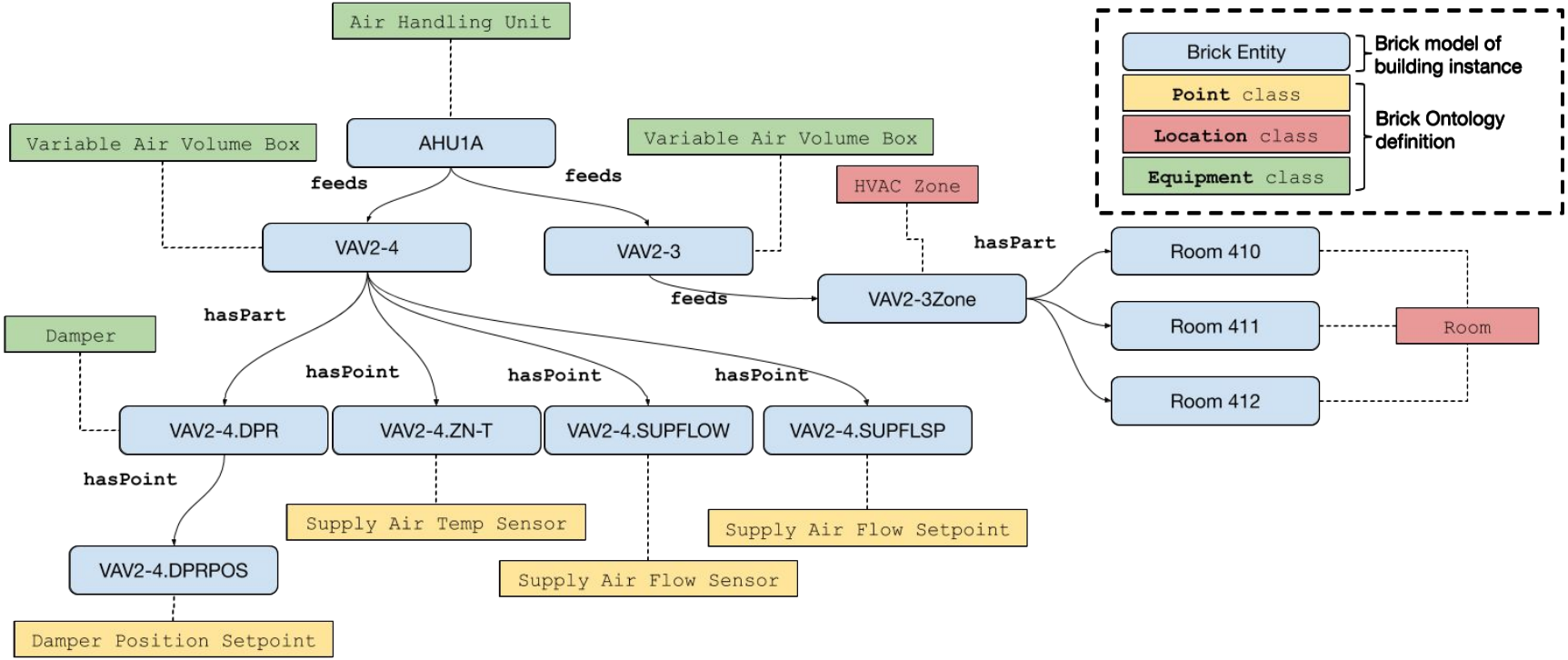


Brick
https://brickschema.org

Brick Overview

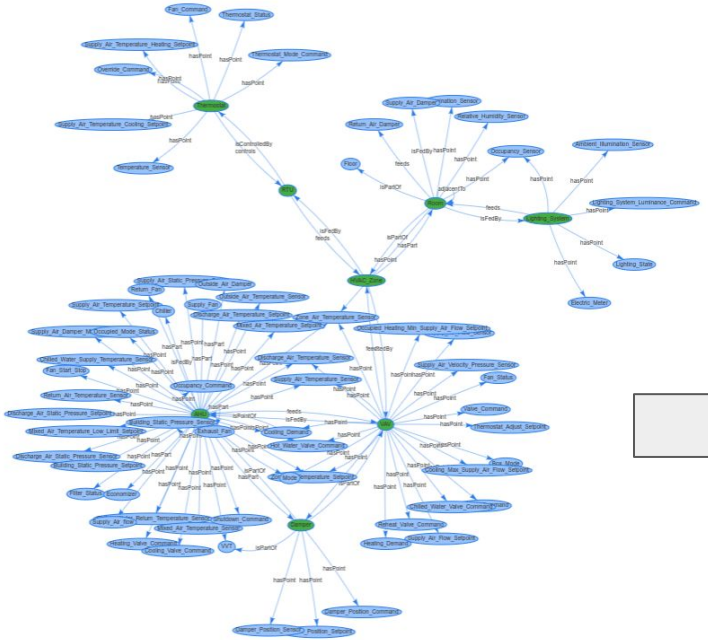
- Three main concepts, each the root of their own **class hierarchy**
- **Classes** provide definition, organization to **entities**
- **Entities** are the physical, logical and virtual “things”
- **Relationships** dictate how entities correspond and relate to each other





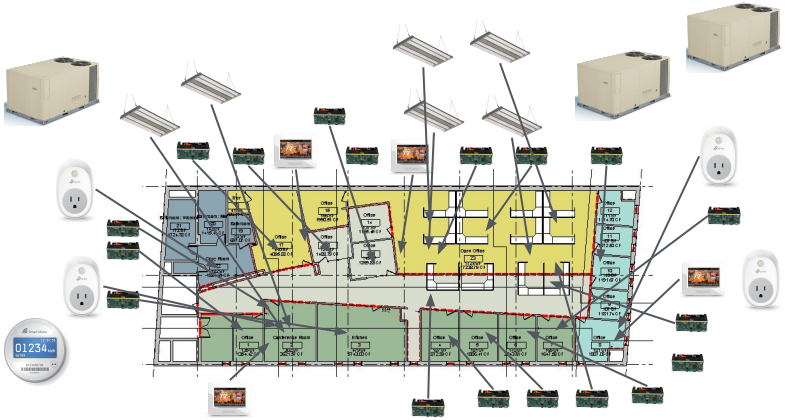
- Have a set of physical, virtual "*things*" and points that an application wants to refer to
- Brick defines a **hierarchical class structure** to define **standard names** for equipment, points, locations, etc
- Brick defines a set of **standard relationships** that describe how things are **connected**

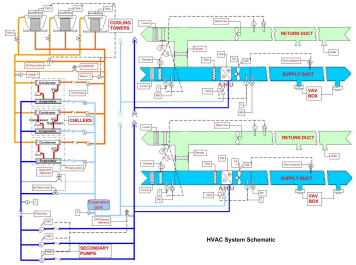
Brick
https://brickschema.org



Brick model

represents





A **Brick model** represents the **assets** and **relationships** and **data** in a building

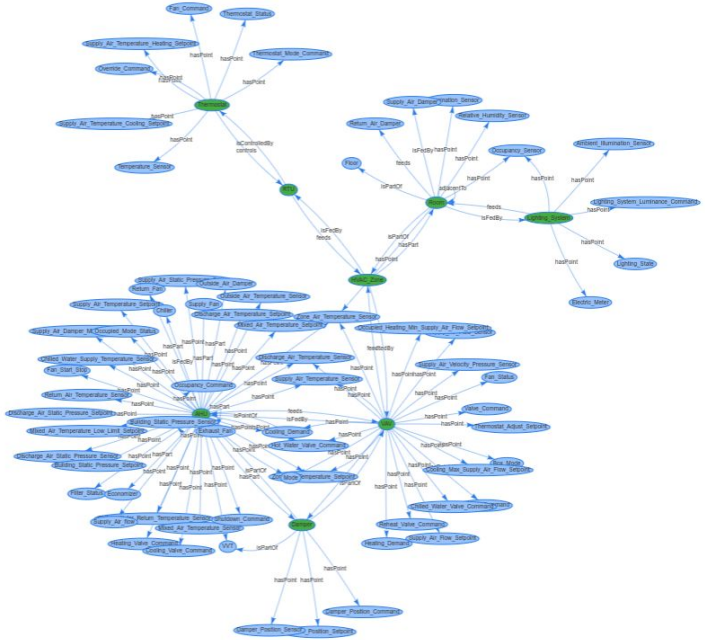
Brick
https://brickschema.org

Query

```
SELECT ?tstat ?zone ?tstat_uri ?state_uuid ?temp_uuid ?weather_uuid FROM ciee WHERE {
    ?zone rdf:type brick:HVAC_Zone .
    ?rtu bf:feeds+ ?zone .

    ?tstat rdf:type/rdfs:subClassOf* brick:Thermostat .
    ?tstat bf:controls+ ?rtu .
    ?tstat bf:uri ?tstat_uri.

    ?tstat bf:hasPoint ?state .
    ?state rdf:type brick:Thermostat_HVAC_Operation_Status .
    ?state bf:uuid ?state_uuid .

    ?tstat bf:hasPoint ?temp .
    ?temp rdf:type/rdfs:subClassOf* brick:Temperature_Sensor .
    ?temp bf:uuid ?temp_uuid .

    ?tstat bf:hasSite ?site .
    ?weather_sensor bf:hasSite ?site .
    ?weather_sensor rdf:type/rdfs:subClassOf* brick:Temperature_Sensor .
    ?weather_sensor bf:uuid ?weather_uuid
};
```

Application

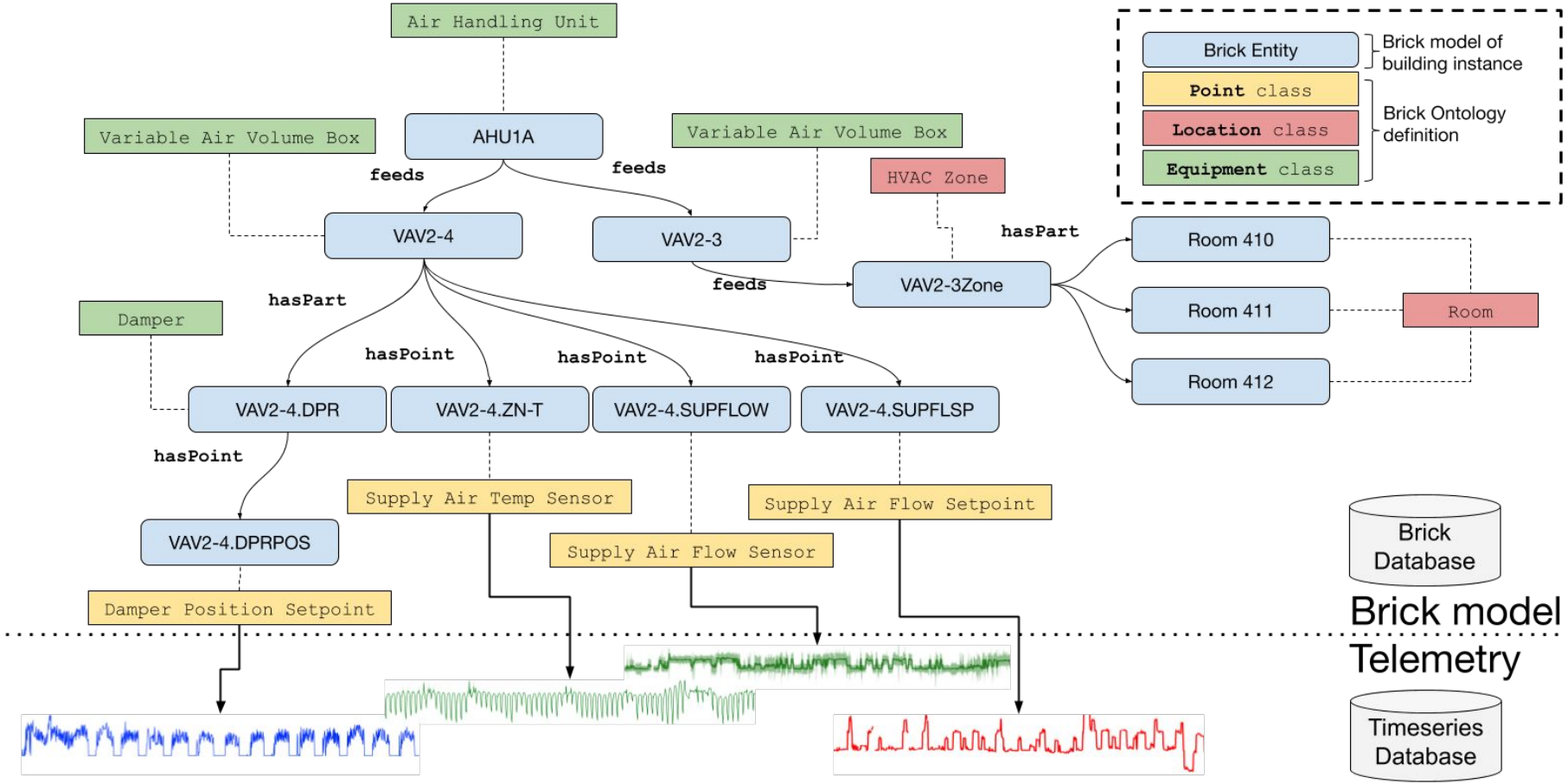
queries



Brick model

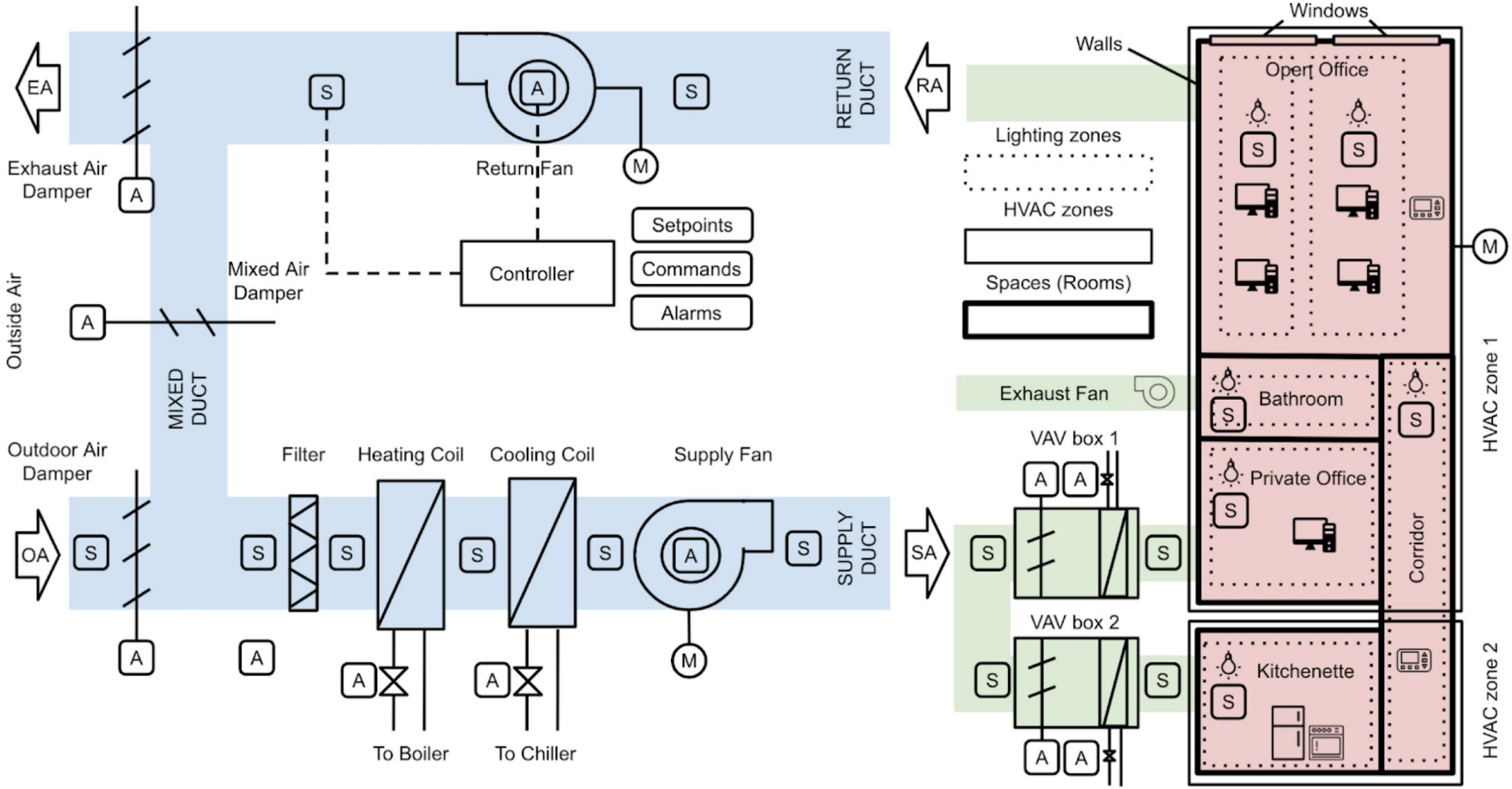
An application **queries** a Brick model to retrieve the data + configuration it needs

Brick
https://brickschema.org

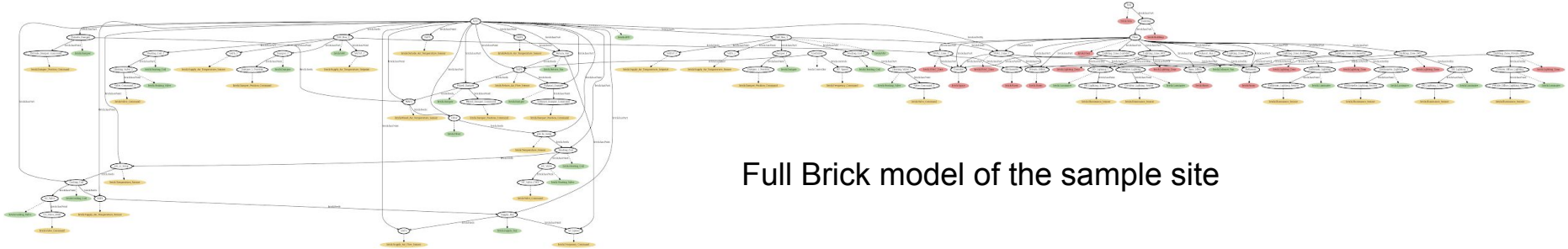


- Relate Brick Point classes to timeseries data
- Contextualize data in existing datastores

Brick
https://brickschema.org

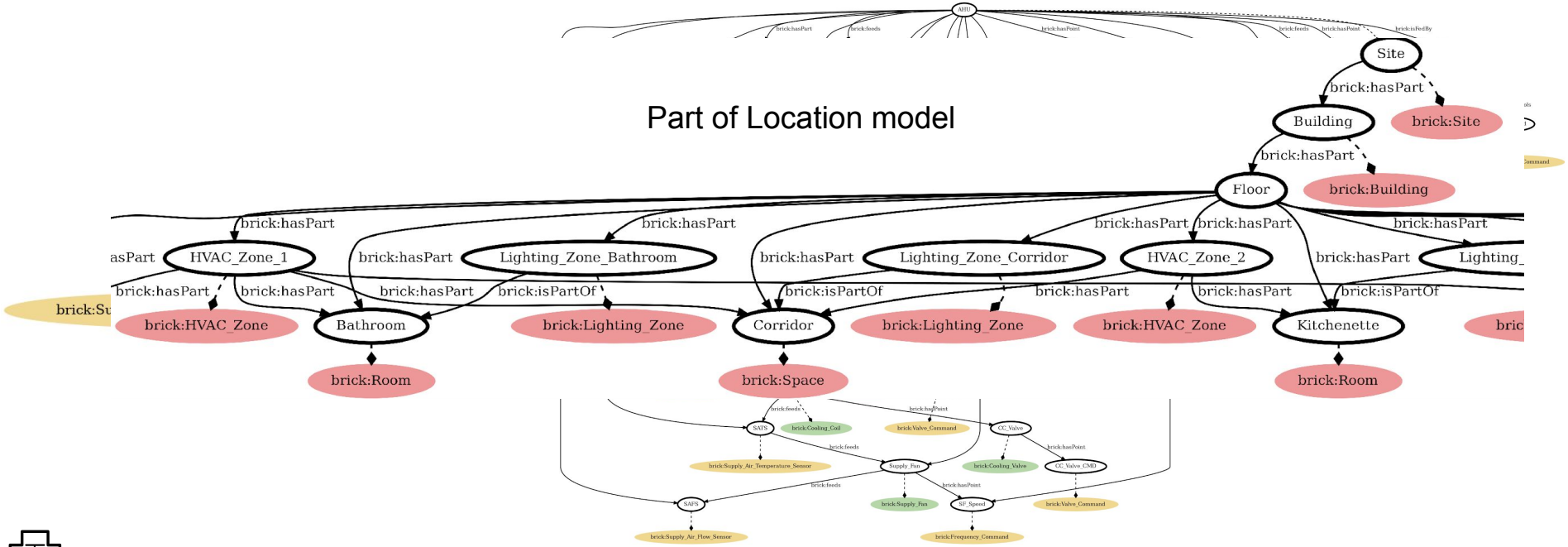


Brick
https://brickschema.org



Full Brick model of the sample site

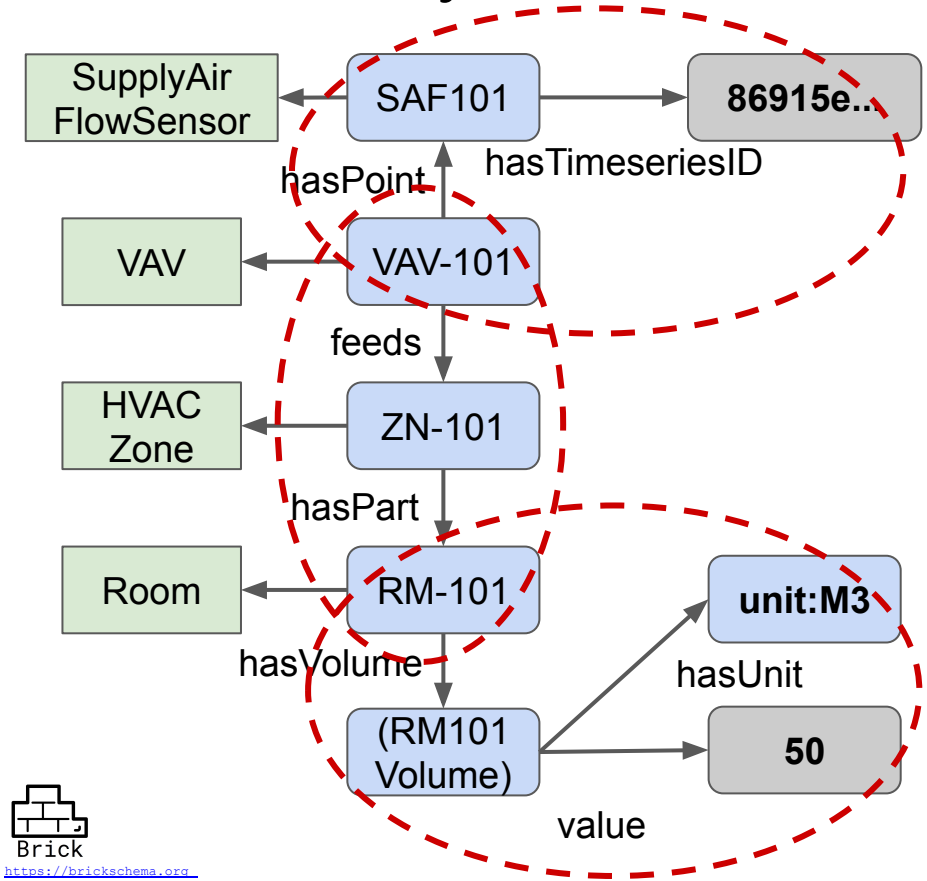
Part of Location model



Brick

https://brickschema.org

Brick Query: Minimum Airflow Requirements




```
select ?saf_ref ?volume ?unit where {

# Identify the room's VAV
?zone      brick:hasPart      <RM-101>.
?zone      rdf:type           brick:HVACZone.
?device    brick:feeds        ?zone.
?device    rdf:type           brick:TerminalUnit.

# Get the air flow sensor's timeseries ref.
?device    brick:hasPoint     ?saf.
?saf       rdf:type           brick:AirFlowSensor.
?saf       brick:hasTimeseriesID  ?saf_ref.

# Get the room's volume.
?room      brick:hasVolume    ?quantity.
?quantity  brick:value        ?volume
?quantity  brick:hasUnit      ?unit
}
```

Brick
https://brickschema.org

Uncerscores are omitted due to the limited space

Brick Model Validation

User-defined "templates" to validate Brick models

Model_XYZ_VAV

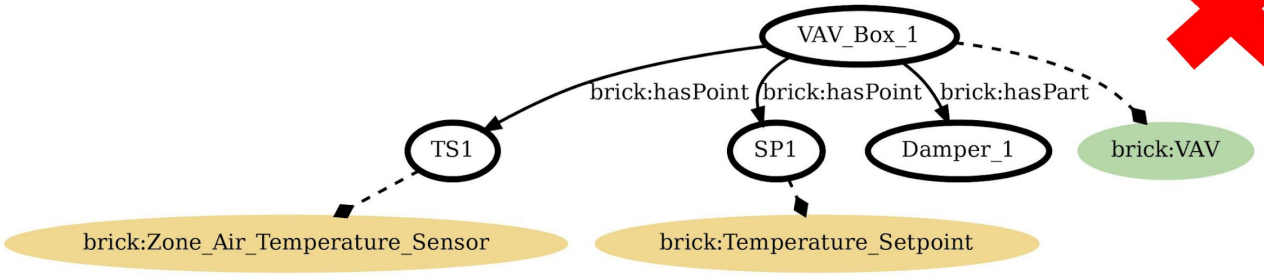
```
instance of brick:VAV
1 brick:Zone_Air_Temperature_Sensor
1 brick:Temperature_Setpoint
1 model_ABC_damper
```

Model_ABC_Damper

```
instance of brick:Damper
1 brick:Damper_Position_Command
```

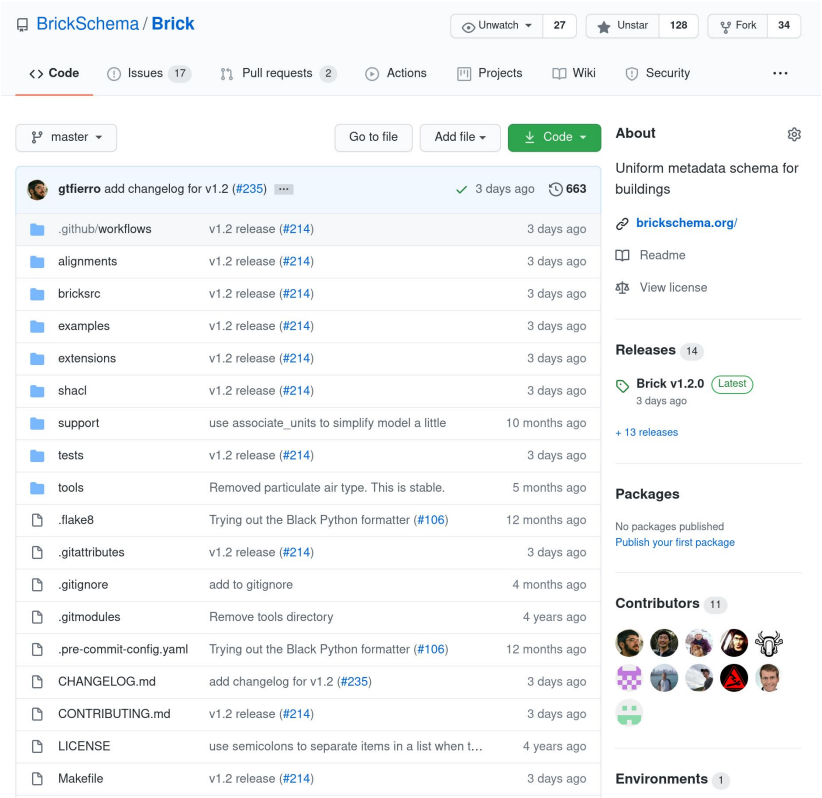
Templates ensure that Brick models match expectations, requirements

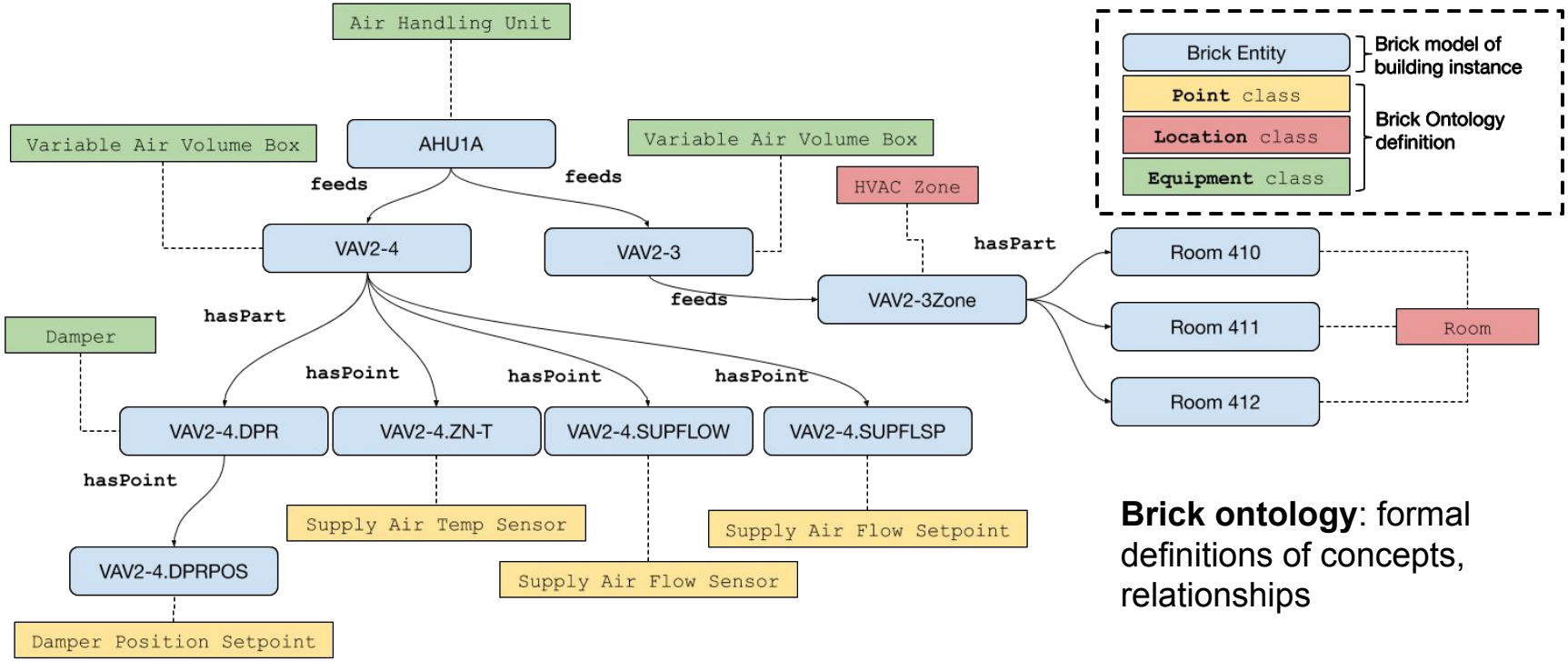
Verify that Brick is being used correctly



Brick Resources

- Developer-focused documentation:
  - https://docs.brickschema.org/
- Concept documentation:
  - https://brickschema.org/ontology
- Download Brick and examples:
  - https://brickschema.org/resources
- Python library:
  - https://brickschema.readthedocs.io/



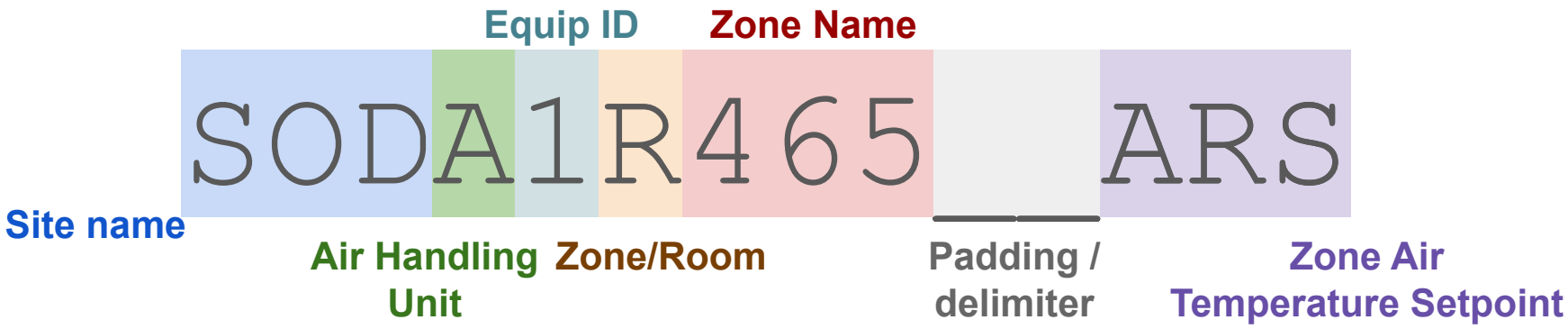


Brick ontology: formal definitions of concepts, relationships

Brick model: the graph representing a particular building

Brick
https://brickschema.org

What's in a Name?

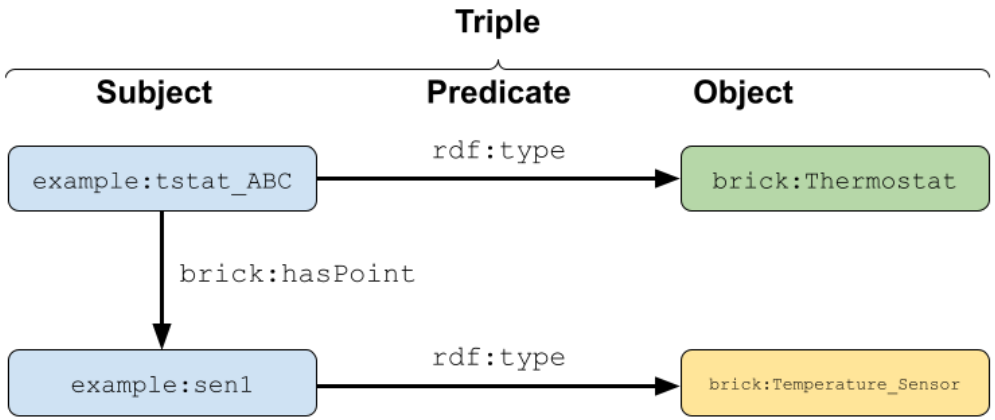


- Labels like these often the *only* up-to-date and readily-available source of metadata
- Site-specific conventions; sometimes no labels at all
- **No consistent, standard metadata for cyberphysical system data**

OUTLINE

1. Brick Implementation
   - The Brick Distribution
   - Extending Brick
2. Brick Tooling
   - `brickschema` Python package
   - `brick-builder` and OpenRefine

```
example:tstat_ABC        rdf:type          brick:Thermostat ;
                         brick:hasPoint    example:sen1 .
example:sen1             rdf:type          brick:Temperature_Sensor .
```



SIMPLE BRICK CLASS

```
1
2  brick:Temperature_Sensor a owl:Class ;
3      rdfs:label "Temperature Sensor" ;
4      rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
5          brick:Sensor ;
6      owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
7                            owl:hasValue brick:Temperature ;
8                            owl:onProperty brick:measures ] ) ] ;
9      skos:definition "Measures temperature: the physical property of matter that quantitatively
10     brick:hasAssociatedTag tag:Point,
11         tag:Sensor,
12         tag:Temperature .
```

SIMPLE BRICK CLASS

```
1
2  brick:Temperature_Sensor a owl:Class ;
3      rdfs:label "Temperature Sensor" ;
4      rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
5          brick:Sensor ;
6      owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
7                            owl:hasValue brick:Temperature ;
8                            owl:onProperty brick:measures ] ) ] ;
9      skos:definition "Measures temperature: the physical property of matter that quantitatively
10     brick:hasAssociatedTag tag:Point,
11         tag:Sensor,
12         tag:Temperature .
```

SIMPLE BRICK CLASS

```
1
2  brick:Temperature_Sensor a owl:Class ;
3      rdfs:label "Temperature Sensor" ;
4      rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
5          brick:Sensor ;
6      owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
7                            owl:hasValue brick:Temperature ;
8                            owl:onProperty brick:measures ] ) ] ;
9      skos:definition "Measures temperature: the physical property of matter that quantitatively
10     brick:hasAssociatedTag tag:Point,
11         tag:Sensor,
12         tag:Temperature .
```

SIMPLE BRICK CLASS

```
1
2  brick:Temperature_Sensor a owl:Class ;
3      rdfs:label "Temperature Sensor" ;
4      rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
5          brick:Sensor ;
6      owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
7                            owl:hasValue brick:Temperature ;
8                            owl:onProperty brick:measures ] ) ] ;
9      skos:definition "Measures temperature: the physical property of matter that quantitatively
10     brick:hasAssociatedTag tag:Point,
11         tag:Sensor,
12         tag:Temperature .
```

SIMPLE BRICK CLASS

```
brick:Temperature_Sensor a owl:Class ;
    rdfs:label "Temperature Sensor" ;
    rdfs:subClassOf [ owl:intersectionOf ( _:has_Point _:has_Sensor _:has_Temperature ) ],
        brick:Sensor ;
    owl:equivalentClass [ owl:intersectionOf ( [ a owl:Restriction ;
                        owl:hasValue brick:Temperature ;
                        owl:onProperty brick:measures ] ) ] ;
    skos:definition "Measures temperature: the physical property of matter that quantitatively
    brick:hasAssociatedTag tag:Point,
        tag:Sensor,
        tag:Temperature .
```

EXTENDING BRICK

BrickSchema / **Brick**

Unwatch 27 | Unstar 128 | Fork 34

Code | Issues 17 | Pull requests 2 | Actions | Projects | Wiki | Security | Insights | Settings

master | **26** branches | **14** tags | Go to file | Add file | Code

gtfierro add changelog for v1.2 (#235) — 45909db 3 days ago — **663** commits

.github/workflows	v1.2 release (#214)	3 days ago
alignments	v1.2 release (#214)	3 days ago
bricksrc	v1.2 release (#214)	3 days ago
examples	v1.2 release (#214)	3 days ago
extensions	v1.2 release (#214)	3 days ago
shacl	v1.2 release (#214)	3 days ago
support	use associate_units to simplify model a little	10 months ago
tests	v1.2 release (#214)	3 days ago
tools	Removed particulate air type. This is stable.	5 months ago
.flake8	Trying out the Black Python formatter (#106)	12 months ago
.gitattributes	v1.2 release (#214)	3 days ago
.gitignore	add to gitignore	4 months ago
.gitmodules	Remove tools directory	4 years ago
.pre-commit-config.yaml	Trying out the Black Python formatter (#106)	12 months ago
CHANGELOG.md	add changelog for v1.2 (#235)	3 days ago
CONTRIBUTING.md	v1.2 release (#214)	3 days ago
LICENSE	use semicolons to separate items in a list when the items contain commas	4 years ago
Makefile	v1.2 release (#214)	3 days ago

About

Uniform metadata schema for buildings

brickschema.org/

Readme

View license

Releases 14

Brick v1.2.0 Latest
3 days ago

+ 13 releases

Packages

No packages published
Publish your first package

Contributors 11

Environments 1

github-pages Active

EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
```

EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
```

EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
```

EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
```

EXTENDING BRICK

```
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Air_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Air],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Air],
                        ]
```

EXTENDING BRICK

How to add a `Glycol Temperature Sensor`

1. Find an appropriate parent class
2. Add the class definition to the Python file
3. Add textual definition to `definitions.csv`
4. Compile and run tests

EXTENDING BRICK

How to add a `Glycol Temperature Sensor`

- Find an appropriate parent class
- Add the class definition to the Python file

```python
# bricksrc/sensor.py
sensor_definitions = {
    "Sensor": {
        "tags": [TAG.Point, TAG.Sensor],
        "subclasses": {
            "Temperature_Sensor": {
                "tags": [TAG.Point, TAG.Sensor, TAG.Temperature],
                "substances": [[BRICK.measures, BRICK.Temperature]],
                "subclasses": {
                    "Glycol_Temperature_Sensor": {
                        "tags": [TAG.Point, TAG.Sensor, TAG.Temperature, TAG.Glycol],
                        "substances": [
                            [BRICK.measures, BRICK.Temperature],
                            [BRICK.measures, BRICK.Glycol],
                        ],
```

EXTENDING BRICK

How to add a `Glycol Temperature Sensor`

- Add textual definition to `definitions.csv`

```
# bricksrc/definitions.csv
https://brickschema.org/schema/Brick#Generator_Room,"A room for electrical equipment, specifica
https://brickschema.org/schema/Brick#Glycol_Temperature_Sensor,A sensor which measures the temp
https://brickschema.org/schema/Brick#HVAC_Equipment,See Heating_Ventilation_Air_Conditioning_Sy
```

EXTENDING BRICK

How to add a `Glycol Temperature Sensor`

- Compile using the Makefile

```
(venv) gabe@arkestra:~/src/Brick$ make
mkdir -p extensions
python generate_brick.py
Checking 58 containers
Non-zero exit code 1 with message b'\nAllegroGraph Server Edition 7.1.0 (rc6), built on Januar
Checking 58 containers
Non-zero exit code 1 with message b'\nAllegroGraph Server Edition 7.1.0 (rc6), built on Januar
2021-02-22:12:01:30,297 WARNING [generate_brick.py:296] Property 'tags' not defined for https:
2021-02-22:12:01:31,388 WARNING [generate_brick.py:296] Property 'tags' not defined for https:
2021-02-22:12:01:32,676 WARNING [generate_brick.py:519] WARNING: Close_Setpoint does not exist
2021-02-22:12:01:32,720 WARNING [generate_brick.py:519] WARNING: Unoccupied_Cooling_Supply_Air
2021-02-22:12:01:32,743 WARNING [generate_brick.py:519] WARNING: Unoccupied_Heating_Supply_Air
2021-02-22:12:01:32,754 WARNING [generate_brick.py:519] WARNING: Unoccupied_Heating_Discharge_
2021-02-22:12:01:32,771 WARNING [generate_brick.py:519] WARNING: Unoccupied_Cooling_Supply_Air
2021-02-22:12:01:32,793 WARNING [generate_brick.py:519] WARNING: Unoccupied_Heating_Supply_Air
```

EXTENDING BRICK

How to add a `Glycol Temperature Sensor`

- Run unit and integration tests

```
(venv) gabe@arkestra:~/src/Brick$ make test
tests/test_class_structure.py::test_subclasses PASSED
tests/test_conversions.py::test_queries PASSED
tests/test_definitions.py::test_class_definitions PASSED
tests/test_definitions.py::test_relationship_definitions PASSED
tests/test_definitions.py::test_obsolete_definitions PASSED
tests/test_generate_shacl.py::test_domainProperties PASSED
tests/test_generate_shacl.py::test_rangeProperties PASSED
tests/test_measures_inference.py::test_measurable_hierarchy PASSED
tests/test_no_inference.py::test_query_equipment PASSED
tests/test_no_inference.py::test_query_points PASSED
tests/test_no_inference.py::test_query_sensors PASSED
tests/test_no_inference.py::test_query_downstream_temperature PASSED
tests/test_no_inference.py::test_query_room_temp_sensors_ahu1 PASSED
tests/test_quantities.py::test_measurables_defined PASSED
```

EXTENDING BRICK

https://github.com/BrickSchema/Brick/blob/master/CONTRIBUTING.md

Submitting Changes to Brick

Changes to Brick are performed through Pull Requests. It is recommended that you become familiar with how to fork a repository and create a pull request.

Setting up Development Environment

Brick requires Python >= 3.6. We recommend using virtual environments to manage dependencies. We use pre-commit hooks to automatically run code formatters and style checkers when you commit.

1. Check out the Brick repository (or your own fork of it)

```
git clone https://github.com/BrickSchema/Brick
cd Brick
```

2. Install the virtual environment and set up dependencies

```
# creates virtual environment
python3 -m venv venv

# activates virtual environment; do this every time you develop on Brick
source venv/bin/activate

# install dependencies
pip install -r requirements.txt

# install pre-commit hooks
pre-commit install
```

3. Run tests to make sure the build is not broken

```
make test
```

4. Whenever you commit, the `pre-commit` script will run the Black code formatting tool and the flake8 style checker. It will automatically format the code where it can, and indicate when there is a style error. **The tools will not commit unformatted code**; if you see a "Failed" message, please fix the style and re-commit the code. An example of what this looks like is below; the failed flake8 check results in a short error report at the bottom.

```
gabe@arkestra:~/src/Brick$ git commit -m 'adding changes to Alarm hierarchy'
[WARNING] Unstaged files detected.
[INFO] Stashing unstaged files to /home/gabe/.cache/pre-commit/patch1581700010.
Check Yaml.....................................(no files to check)Skipped
```

BRICK TOOLING: `brickschema`

```
pip install brickschema
```

Simple Python package for managing and programming against the Brick ontology

- Load, validate Brick models
- Query, update Brick models
- Simple inference support, extensions, alignments

BRICK TOOLING: `brickschema`

```
import brickschema
# empty graph with no triples
g = brickschema.Graph()
# creates a new rdflib.Graph with a recent version
# of the Brick ontology preloaded.
g = brickschema.Graph(load_brick=True)
# OR use the absolute latest Brick:
g = brickschema.Graph(load_brick_nightly=True)
# OR create from an existing model
g = brickschema.Graph(load_brick=True).from_haystack(...)
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
# empty graph with no triples
g = brickschema.Graph()
# creates a new rdflib.Graph with a recent version
# of the Brick ontology preloaded.
g = brickschema.Graph(load_brick=True)
# OR use the absolute latest Brick:
g = brickschema.Graph(load_brick_nightly=True)
# OR create from an existing model
g = brickschema.Graph(load_brick=True).from_haystack(...)
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
# empty graph with no triples
g = brickschema.Graph()
# creates a new rdflib.Graph with a recent version
# of the Brick ontology preloaded.
g = brickschema.Graph(load_brick=True)
# OR use the absolute latest Brick:
g = brickschema.Graph(load_brick_nightly=True)
# OR create from an existing model
g = brickschema.Graph(load_brick=True).from_haystack(...)
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
# empty graph with no triples
g = brickschema.Graph()
# creates a new rdflib.Graph with a recent version
# of the Brick ontology preloaded.
g = brickschema.Graph(load_brick=True)
# OR use the absolute latest Brick:
g = brickschema.Graph(load_brick_nightly=True)
# OR create from an existing model
g = brickschema.Graph(load_brick=True).from_haystack(...)
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
g = brickschema.Graph(load_brick=True)
# load in data files from your file system
g.load_file("mybuilding.ttl")
# ...or by URL
g.parse("https://brickschema.org/ttl/soda_brick.ttl", format="ttl")
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
g = brickschema.Graph(load_brick=True)
# load in data files from your file system
g.load_file("mybuilding.ttl")
# ...or by URL
g.parse("https://brickschema.org/ttl/soda_brick.ttl", format="ttl")
```

The `graph` is the unit of management for Brick

BRICK TOOLING: `brickschema`

```
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.expand(profile="owlrl") # also supports 'rdfs', 'shacl', 'vbis'
g.expand(profile="owlrl+shacl") # create "schedules" of reasoning
```

Reasoning adds implied information to the Brick graph

BRICK TOOLING: `brickschema`

```
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.expand(profile="owlrl") # also supports 'rdfs', 'shacl', 'vbis'
g.expand(profile="owlrl+shacl") # create "schedules" of reasoning
```

Reasoning adds implied information to the Brick graph

BRICK TOOLING: `brickschema`

```
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.expand(profile="owlrl")
valid, _, resultsText = g.validate()
if not valid:
    print("Graph is not valid!")
    print(resultsText)
```

Validate the Brick ontology is being used correctly

BRICK TOOLING: `brickschema`

```python
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.get_extensions()
# => ['shacl_tag_inference']
g.load_extension('shacl_tag_inference')
g.expand("shacl") # usually run reasoning after loading extension

g.get_alignments()
# => ['VBIS', 'REC', 'BOT']
g.load_alignment('BOT')
g.expand("owlrl") # usually run reasoning after loading alignment
```

Extensions and alignments add features, enable integration with other ontologies and metadata representations

BRICK TOOLING: `brickschema`

```python
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.get_extensions()
# => ['shacl_tag_inference']
g.load_extension('shacl_tag_inference')
g.expand("shacl") # usually run reasoning after loading extension

g.get_alignments()
# => ['VBIS', 'REC', 'BOT']
g.load_alignment('BOT')
g.expand("owlrl") # usually run reasoning after loading alignment
```

Extensions and alignments add features, enable integration with other ontologies and metadata representations

BRICK TOOLING: `brickschema`

```python
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
# perform SPARQL queries on the graph
res = g.query("""SELECT ?afs ?afsp ?vav WHERE  {
    ?afs    a       brick:Air_Flow_Sensor .
    ?afsp   a       brick:Air_Flow_Setpoint .
    ?afs    brick:isPointOf ?vav .
    ?afsp   brick:isPointOf ?vav .
    ?vav    a   brick:VAV
}""")
for row in res:
    (air_flow_sensor, air_flow_setpoint, vav) = row
    print(row)
```

SPARQL queries retrieve parts of the Brick graph for applications

BRICK TOOLING: brickschema

```python
import brickschema
g = brickschema.Graph(load_brick=True)
g.load_file("mybuilding.ttl")
g.serve()
```

Apply OWLRL Reasoning | Apply RDFS Reasoning | Apply SHACL Reasoning | Apply Brick Tag Reasoning | Apply VBIS Reasoning

Query

http://localhost:8080/query

```
PREFIX unit: <http://qudt.org/vocab/unit/>
PREFIX quantitykind: <http://qudt.org/vocab/quantitykind/>
PREFIX qudt: <http://qudt.org/schema/qudt/>
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX brick: <https://brickschema.org/schema/1.1/Brick#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT * WHERE {
  ?sensor a brick:Temperature_Sensor .
  ?sensor brick:isPointOf ?vav .
  ?vav a brick:VAV .
} LIMIT 10
```

Table | Response | 10 results in 0.106 seconds

Filter query results | Page size: 50

	vav	sensor
1	<https://brickschema.org/schema/1.0.2/building_example#vav_R310>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R310>
2	<https://brickschema.org/schema/1.0.2/building_example#vav_C500A>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_C500A>
3	<https://brickschema.org/schema/1.0.2/building_example#vav_R179>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R179>
4	<https://brickschema.org/schema/1.0.2/building_example#vav_R347>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R347>
5	<https://brickschema.org/schema/1.0.2/building_example#vav_R288>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R288>
6	<https://brickschema.org/schema/1.0.2/building_example#vav_C700B>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_C700B>
7	<https://brickschema.org/schema/1.0.2/building_example#vav_R465H>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R465H>
8	<https://brickschema.org/schema/1.0.2/building_example#vav_R537>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R537>
9	<https://brickschema.org/schema/1.0.2/building_example#vav_R327>	<https://brickschema.org/schema/1.0.2/building_example#temp_sensor_hvac_zone_R327>

BRICK TOOLING: `brick-builder`

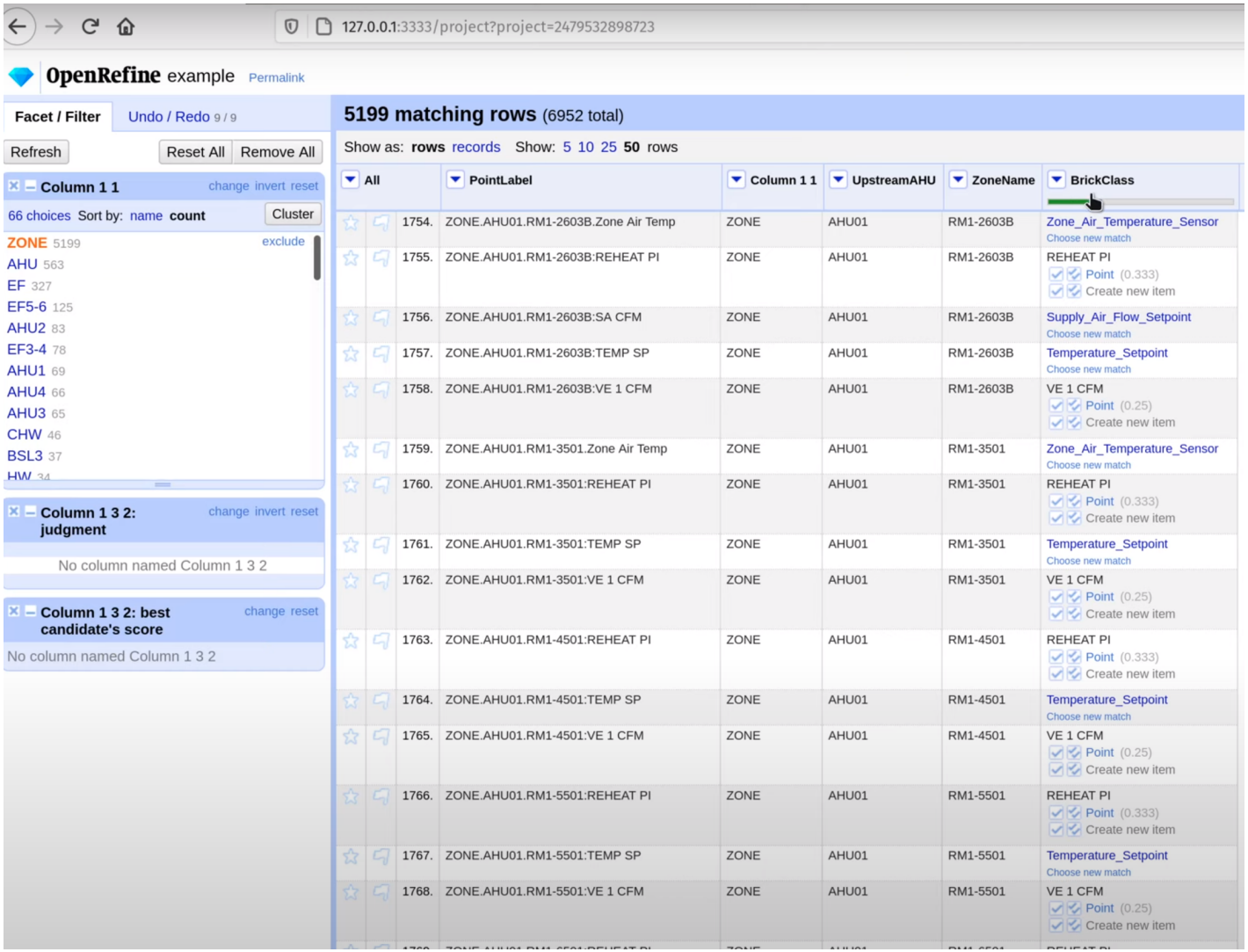
BRICK TOOLING: `brick-builder`

SODA2S14___SMK
SODA1S11___MAT
SODA3R315_RVAV
SODA3R723__ASO
SODA3R327__AGN
SODH1P02___FLT
SODA3R798__ART
SODA1R405B_ARS
SODA3R683_RVAV
SODA1R405B_ART
SODA3R311__AGN
SODH1_____L_L
SODC1SP03__FLT
SODA4R645_RVAV
SODA1R288__AGN
SODA3R419__AGN
SODA3C611__ASO
SODA2S14_P__VR
SODA4S1832_STA

AHU.AHU01.CAV1-1:DMPRPOS
AHU.AHU01.CAV1-1:HTG O
AHU.AHU01.CAV1-1:SUPFLOW
AHU.AHU01.CAV1-1:ZN T
AHU.AHU01.CAV2-1:DAT
AHU.AHU01.CAV2-1:DMPRPOS
AHU.AHU01.CAV2-1:HTG O
AHU.AHU01.CAV2-1:SUPFLOW
AHU.AHU01.CAV2-1:ZN T
AHU.AHU01.CCV
AHU.AHU01.CHWHHW.UNT:CHW FLOW
AHU.AHU01.CHWHHW.UNT:HW FLOW
AHU.AHU01.Cooling Enable
AHU.AHU01.ECM
AHU.AHU01.HP.UNT:ZN T
AHU.AHU01.HSP
AHU.AHU01.LSP
AHU.AHU01.LTD
AHU.AHU01.MAX.ZONE.DAMPER
AHU.AHU01.MAX.ZONE.HEATING
AHU.AHU01.MIN OA
AHU.AHU01.Mixed Air Damper Position
AHU.AHU01.Mixed Air Temp

Trunk.VAV2-12.OCCHTGFL
Trunk.CentralPlant.HWP2-RST
Trunk.VAV2-4.BOXHTG
Trunk.VAV2-9.SUPFLOSP
Trunk.CentralPlant.CHWP4-S
Trunk.VAV2-7.COMMONSP
Trunk.VAV1-5.SUPFLOW
Trunk.VAV2-10.S-VP
Trunk.VAV2-3.SUPFLOSP
Trunk.VVT-4.UNOCDB
Trunk.VAV2-10.BOXHTG
Trunk.VVT-5.ZN-T
Trunk.CentralPlant.HWP2-A.Alarm1
Trunk.VVT-1.ZN-T
Trunk.VAV2-8.COMMONSP
Trunk.VAV1-1.BOXMODE
Trunk.AHU-3.MA-T

BRICK TOOLING: `brick-builder`



BRICK TOOLING: `brick-builder`

- Export parsed data to CSV files
- Write a **brick-builder template** (below)

```
brick = https://brickschema.org/schema/1.1/Brick#
rdf = http://www.w3.org/1999/02/22-rdf-syntax-ns#
bldg = http://example.org/building#

bldg:$1 rdf:type brick:VAV
bldg:$1 brick:hasPoint bldg:$2
bldg:$2 rdf:type brick:Temperature_Sensor
bldg:$1 brick:hasPoint bldg:$3
bldg:$3 rdf:type brick:Temperature_Setpoint
$4? bldg:$1 rdf:type brick:RVAV
```

BRICK TOOLING: `brick-builder`

- Builds on OpenRefine tool, Reconciliation API
- Tutorial Video: https://www.youtube.com/watch?v=LKcXMvrxXzE
- NLP-based, other inference techniques under development

Working Group 1: Schema

- What most folks think of when we say “developing Brick”
- Responsible for stewardship of the Brick Schema
- Managing Brick ontology – new classes, new properties and relationships, data modeling approaches
- Extend Brick to new domains
- Create reference models

Working Group 2: Tooling

- More software development than modeling
- Tools to make it easier to create, manage, and exchange Brick models
- Extend/rewrite research prototypes into production tools
- Examples
  - Py-brickschema
  - Brick-builder (and next-gen version from CMU)
  - “Shepherding” tool from Buildsys 2020

Working Group 3: Applications

- Probably more analytics than applications at the beginning
- “Applying” Brick
- Coordinate with Schema WG – for a given use case, what’s missing from the model?
  - Similarly with tooling working group
- May also create “reference models”

Working Group 4: Datasets

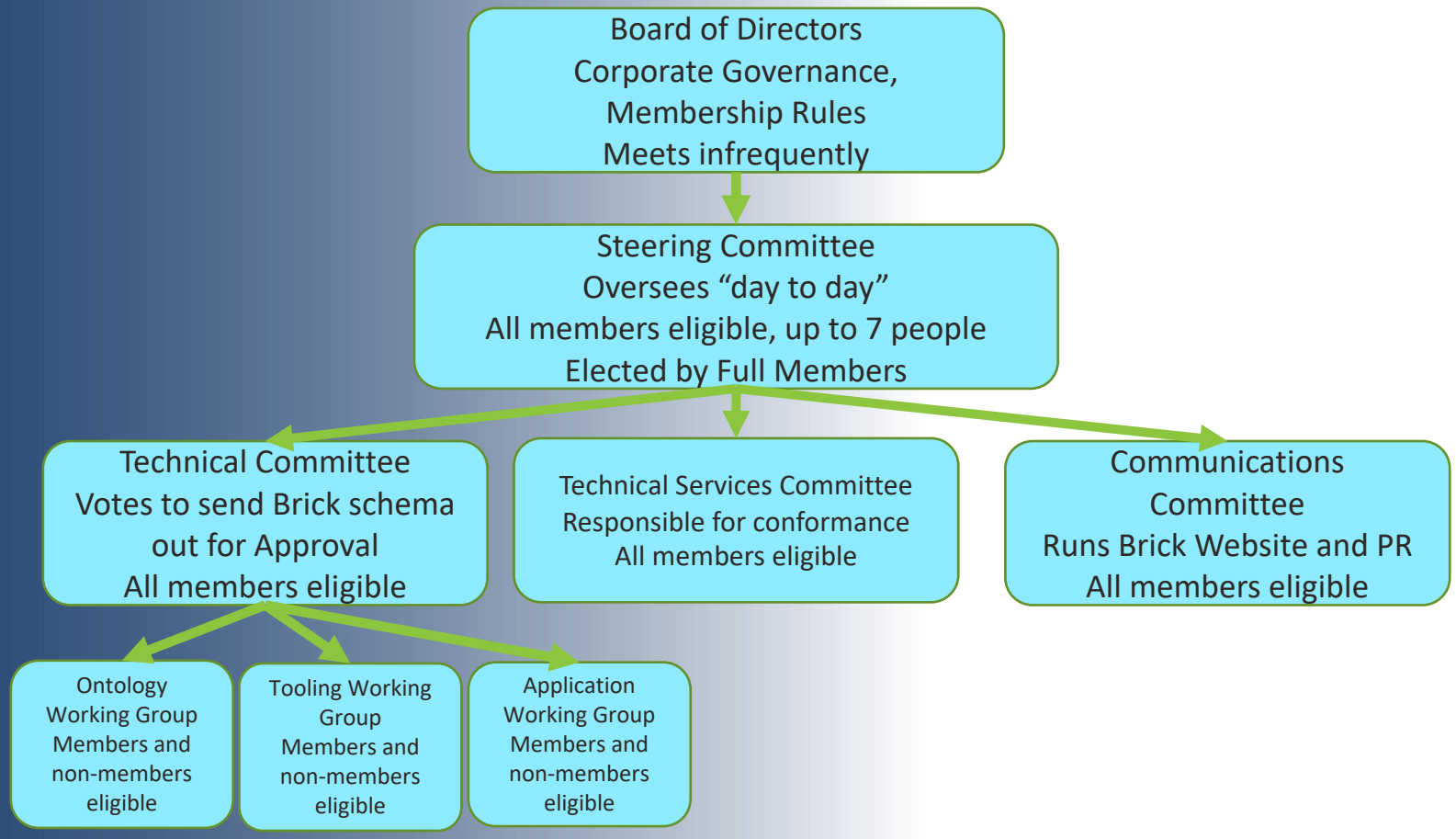
- Interest in collecting anonymized datasets of buildings
- Important for research community and important for validating schema and applications
- Work closely with tooling – a serving platform can be a reference implementation
  - Also ingestion and model creation, with one extra anonymization step

Working Groups: Next Steps

- Two other groups likely on the horizon
  - Conformance
  - Data Exchange
- Groups will kick off in March
- Please add your name to interest survey if you haven’t already!
  - https://groups.google.com/g/brickschema/c/OOHSCDEZwnc

The Brick Consortium Organization

- **Steering Committee**: Sets rules and membership fee, approves new “technical areas” for Brick to work on
  - Has a role in IP consideration
- **Technical Committee**: Responsible for creating Brick, but mostly through Working Groups. Decides when to send Schema out for a vote for release, 30 day clock for vote
- **Other Committees**: Will likely stay on-hold for the first few months



Brick and IP

- Brick Consortium takes IP and IP Disclosure very seriously. We do not want anyone to be surprised!
- Any member or participant who proposes enhancements to the Brick schema must disclose if they have relevant IP – unless they are willing to grant a “non-remunerative license” for that IP.
  - Detailed in section 5 of the membership rules –go by what that says!
  - Inspired by IETF but closes some loopholes
- Also required to disclose if you are aware of IP even if you don’t own it
- Steering Committee can approve the inclusion into the standard parts that would require a license
- Please review section 5 of the membership rules carefully

To Join Brick Consortium

- Choose appropriate membership level
- Complete Membership Form from website
- Submit 2021 dues

	Vote for Steering Committee	Chair any committee	Serve on any other committee	Vote for any other committee	2021 Membership Fee
Full Member	Yes	Yes	Yes	Yes	$50,000
Institutional Member	No	Yes	Yes	Yes	$0
Academic Individual Member	No	Yes	Yes	Yes	$0
Contributing Member	No	No	Yes	Yes	$5000

Brick Public Roadmap Discussion
http://roadmap.brickschema.org/

Brick Roadmap Discussion

- If you are a potential adopter of Brick, what do you think is important to see in Brick?
- If you are a potential adopter of Brick, what's necessary for you to get Brick into your products?
- What are you willing to work on?

Thank You