Open-Source Tool for Haystack Model Validation and Harmonization with RDF-based Schemas

Avijit Saha
Senior Research Engineer
NREL

Gabe Fierro
Assistant Professor
Colorado School of Mines
Joint Appointment at NREL

Photo from iStock-627281636

Motivation

We all agree on the need for **standardization** and **validation…**

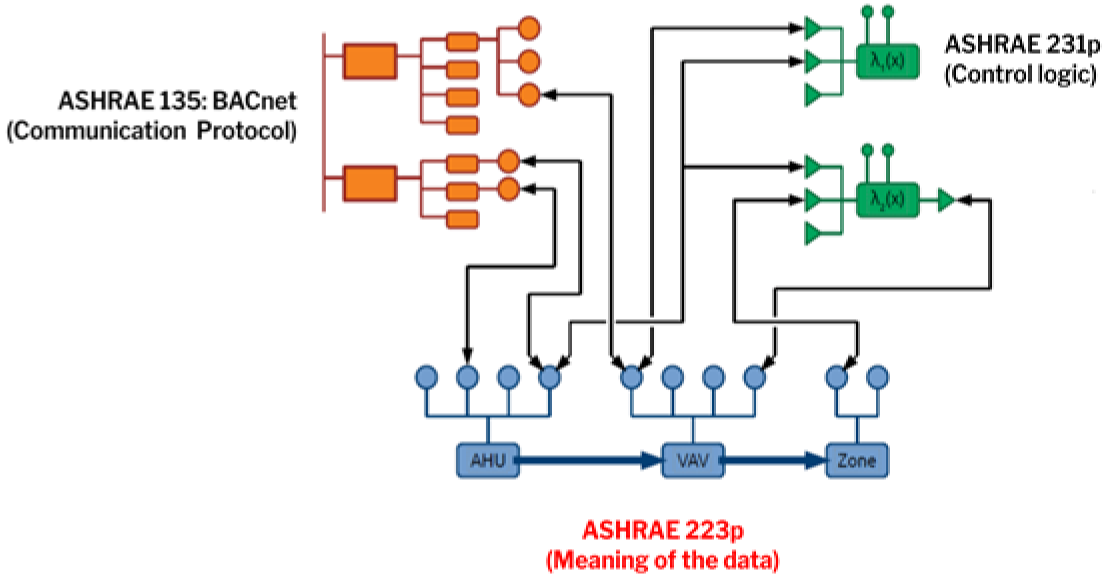
… but how do we provide this in a world with Haystack, Brick, 223P, and other metadata models?

Haystack:
Tags
Xeto
Proto
Specs

RDF:
ASHRAE 223
Brick
RealEstateCore
SPARQL
SHACL

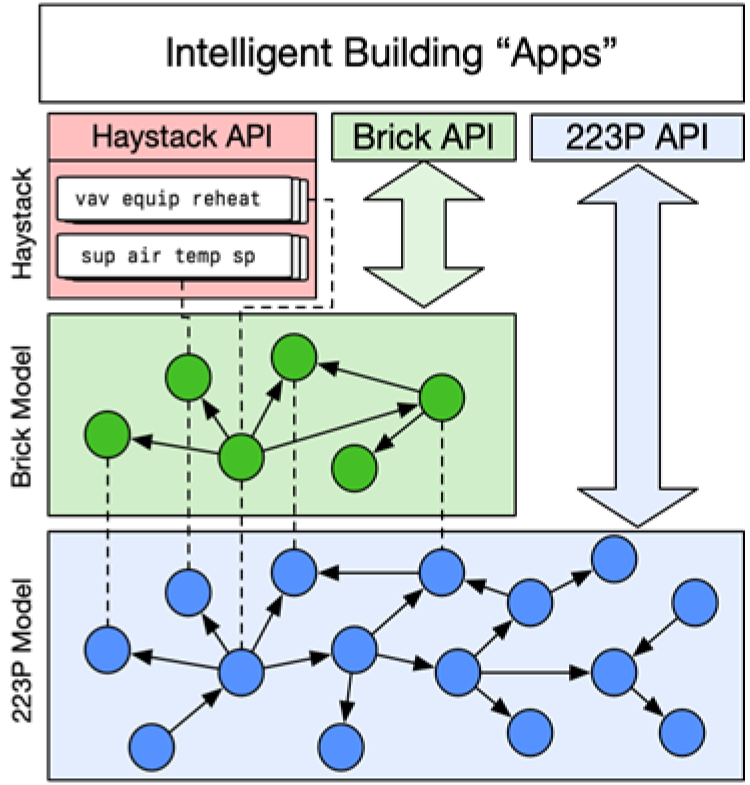
A Vision of Harmonization

- Building Semantic Standard (ASHRAE 223) works in conjunction with other ASHRAE standards to provide a standard process of control delivery and validation.
- RDF-based models integrate multiple perspectives on same building



- *How does Project Haystack fit into this picture?*

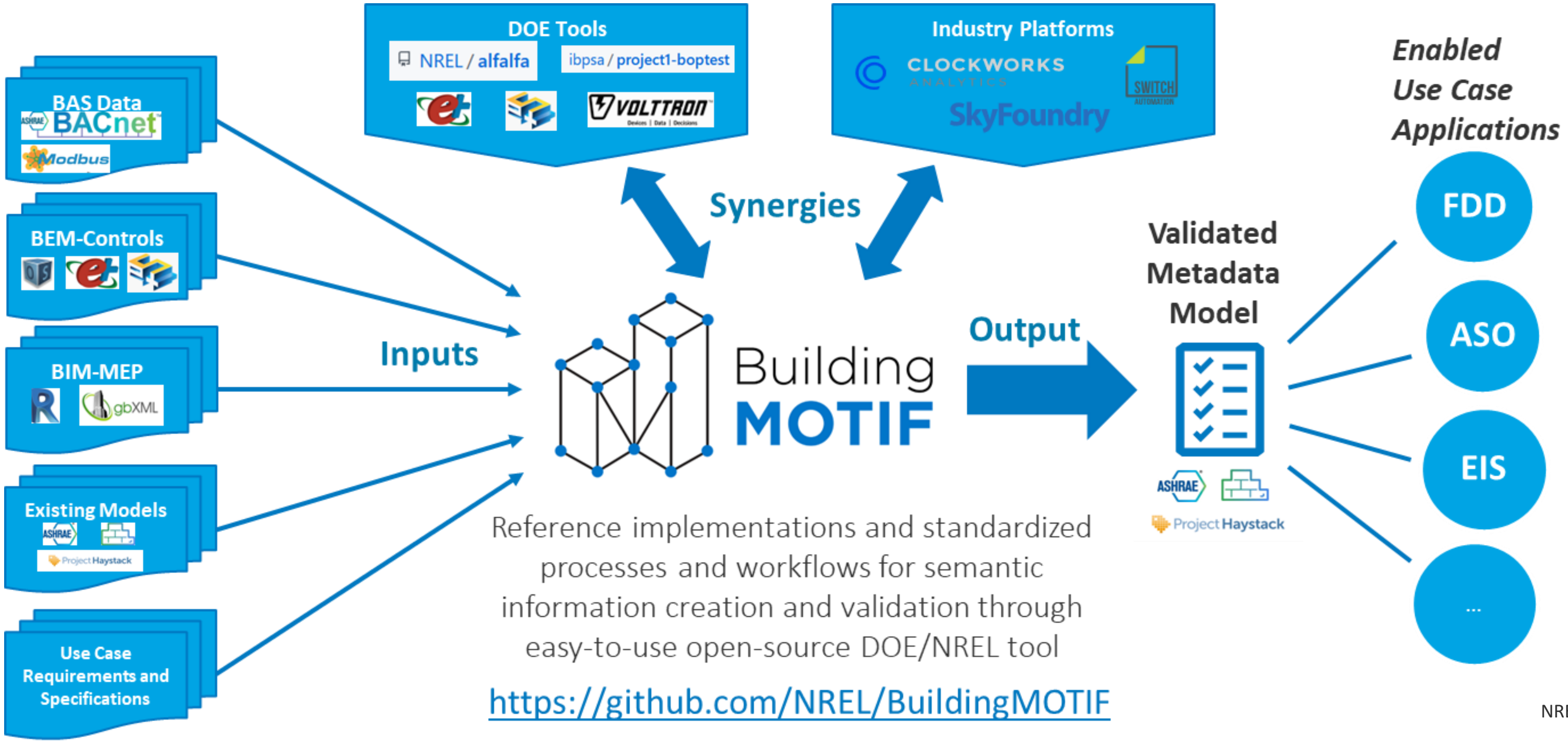
A Vision of Harmonization



- **Goal:** Interoperability among metadata solutions, not necessarily make them same/similar
  - *Embrace why they are different*
- **ASHRAE 223P Standard**:
  - Extremely detailed and precise, and therefore…
  - Verbose to express/query building systems
- **Haystack/Brick**:
  - Lean on common understandings and shared vocabularies
  - Less precise, but easier to query and use
- **Proposed approach:**
  - Haystack and Brick used by field implementers, software developers (as always) – **preserve existing tech investments**
  - Map into 223P standard for validation and exchange

DOE/NREL Open-Source Tool: BuildingMOTIF

MOTIF = **M**etadata **O**n**T**ology **I**nteroperability **F**ramework



Reference implementations and standardized processes and workflows for semantic information creation and validation through easy-to-use open-source DOE/NREL tool

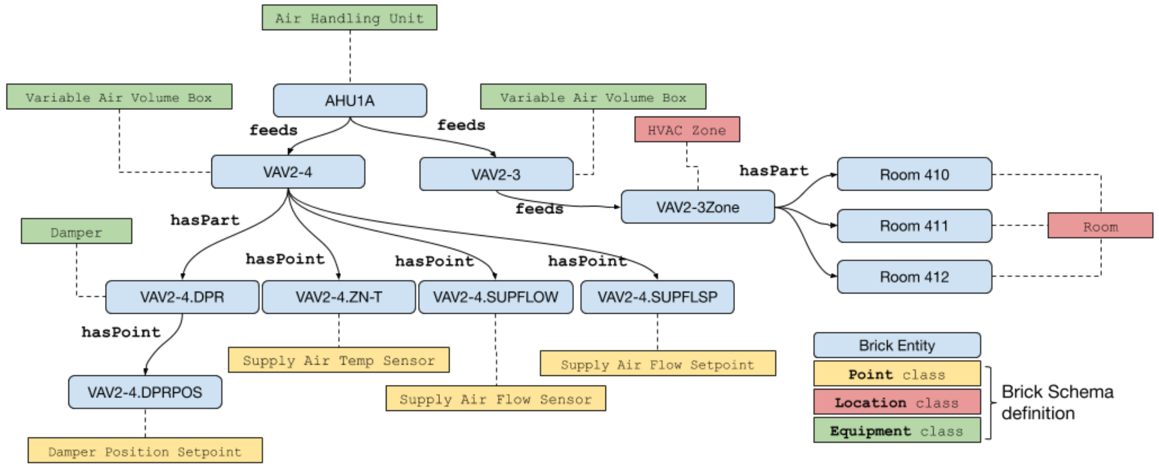
https://github.com/NREL/BuildingMOTIF

Primer on RDF Graphs

The Resource Description Framework (RDF) is a framework for expressing information about resources. An RDF statement expresses a relationship between two resources: a subject and an object through a predicate which is the relationship. These relationships together form a graph which is called an RDF graph.

<Subject>	<Predicate>	<Object>
AHU-1A	feeds	VAV2-4

Brick Example:

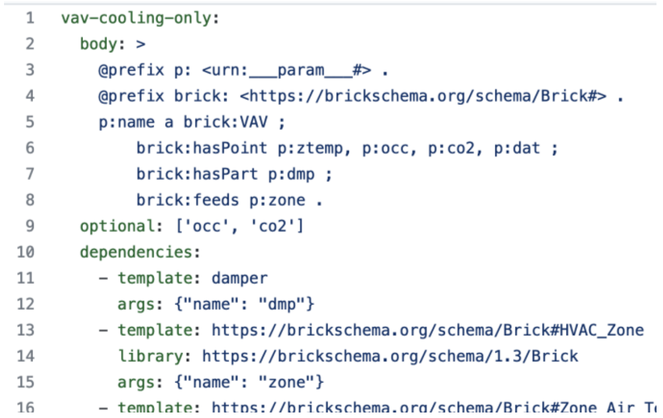


BuildingMOTIF Abstracts Away RDF

- BuildingMOTIF provides programmatic abstractions that hide RDF implementation details
  - Graph generation → **templates**
  - Graph validation → **shapes**
- RDF as a standard exchange of concepts, rules, metadata and entities
- Informs Haystack/Brick harmonization process:
  - More details on this later in the talk!

Graph-Based Templates in BuildingMOTIF

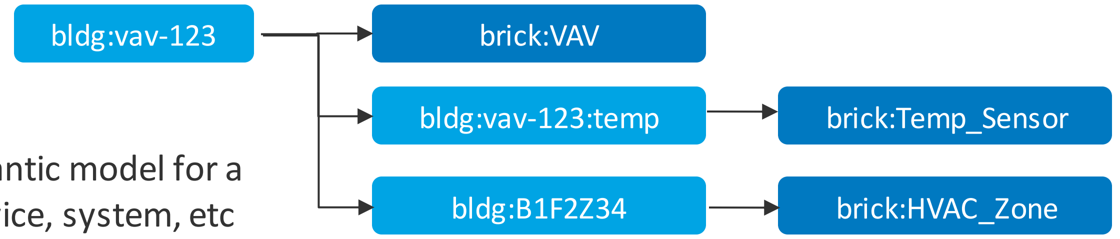
Template: parameterized specification for generating common subgraphs



```
vav-cooling-only:
  body: >
    @prefix p: <urn:___param___#> .
    @prefix brick: <https://brickschema.org/schema/Brick#> .
    p:name a brick:VAV ;
        brick:hasPoint p:ztemp, p:occ, p:co2, p:dat ;
        brick:hasPart p:dmp ;
        brick:feeds p:zone .
  optional: ['occ', 'co2']
  dependencies:
    - template: damper
      args: {"name": "dmp"}
    - template: https://brickschema.org/schema/Brick#HVAC_Zone
      library: https://brickschema.org/schema/1.3/Brick
      args: {"name": "zone"}
    - template: https://brickschema.org/schema/Brick#Zone_Air_T
```

User- or tool-provided *bindings* from CSV, XSV, BACnet scan...

Parameter	Binding
name	urn:bldg/vav-123
ztemp	urn:bldg/vav-123:temp
zone	urn:bldg/B1F2Z34



Consistent semantic model for a component, device, system, etc

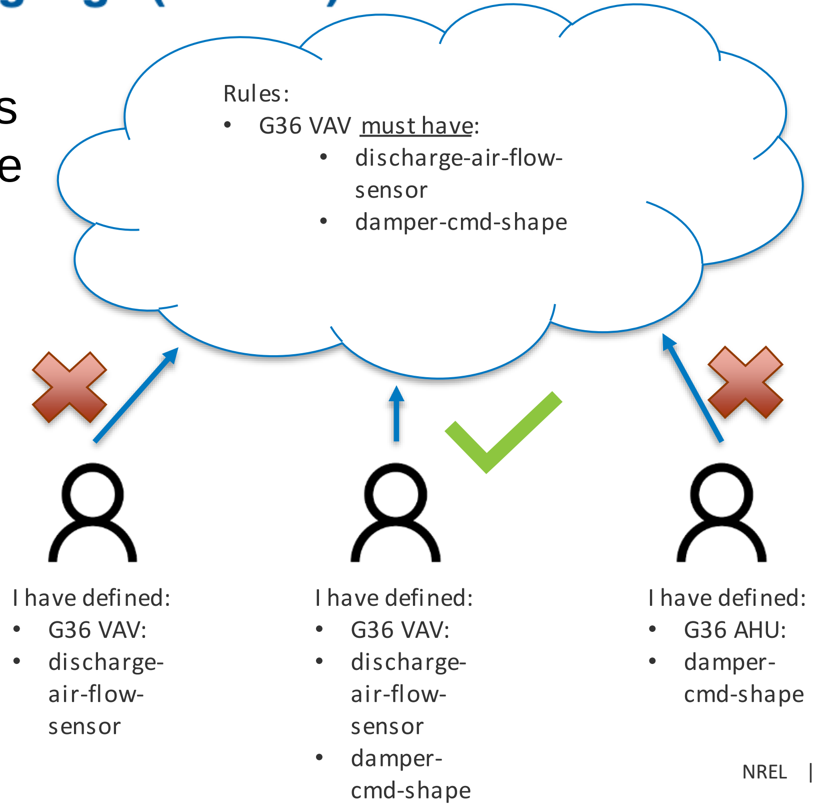
Primer on SHACL Shapes

Shapes Constraint Language (SHACL)

SHACL is a language for validating RDF graphs against a set of conditions. These conditions are provided as shapes and other constructs expressed in the form of an RDF graph.

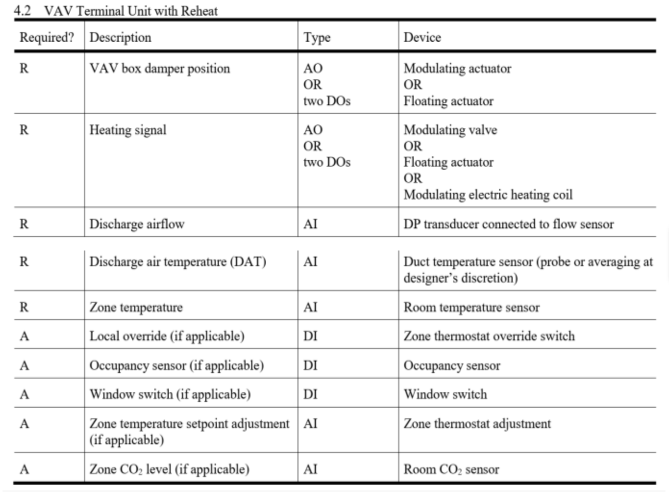
"One motivation for SHACL is Application Integration, where different software components, potentially maintained by different organizations, need to function together smoothly."

https://www.w3.org/TR/shacl-ucr/#scope-and-motivation



Shapes in BuildingMOTIF

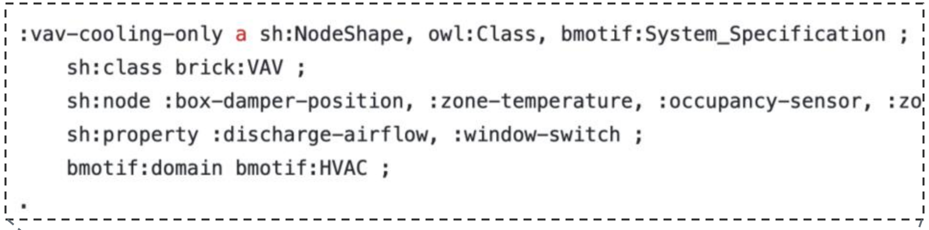
Shape: sets of constraints, conditions that validate part of a metadata model



4.2 VAV Terminal Unit with Reheat

Required?	Description	Type	Device
R	VAV box damper position	AO OR two DOs	Modulating actuator OR Floating actuator
R	Heating signal	AO OR two DOs	Modulating valve OR Floating actuator OR Modulating electric heating coil
R	Discharge airflow	AI	DP transducer connected to flow sensor
R	Discharge air temperature (DAT)	AI	Duct temperature sensor (probe or averaging at designer's discretion)
R	Zone temperature	AI	Room temperature sensor
A	Local override (if applicable)	DI	Zone thermostat override switch
A	Occupancy sensor (if applicable)	DI	Occupancy sensor
A	Window switch (if applicable)	DI	Window switch
A	Zone temperature setpoint adjustment (if applicable)	AI	Zone thermostat adjustment
A	Zone CO_2 level (if applicable)	AI	Room CO_2 sensor

Represent point lists, system configurations, etc...



```
:vav-cooling-only a sh:NodeShape, owl:Class, bmotif:System_Specification ;
    sh:class brick:VAV ;
    sh:node :box-damper-position, :zone-temperature, :occupancy-sensor, :zo
    sh:property :discharge-airflow, :window-switch ;
    bmotif:domain bmotif:HVAC ;
.
```

Target	Condition
brick:VAV	hasPoint 1 brick:Zone Temp Sensor
brick:VAV	hasPart 1 brick:Damper
brick:Damper	hasPoint 1 brick:Damper Pos Cmd

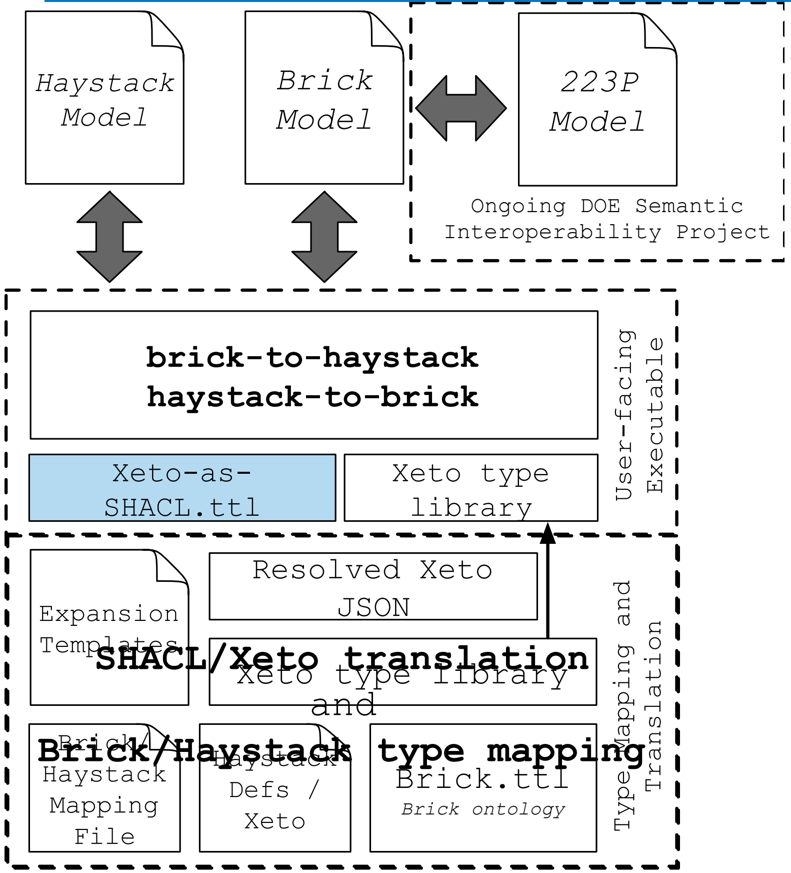
- Built using SHACL W3C standard
- Composable, expressive, declarative language
- Computational validation of metadata models
- **Ontology agnostic**

Applying BuildingMOTIF to RDF/Haystack

- There is plenty more to say on BuildingMOTIF!
  - Check out repository + documentation if you are curious
  - https://github.com/NREL/BuildingMOTIF

- Let's look at how BuildingMOTIF enables interoperability between the RDF world (Brick, 223P, etc) and Haystack

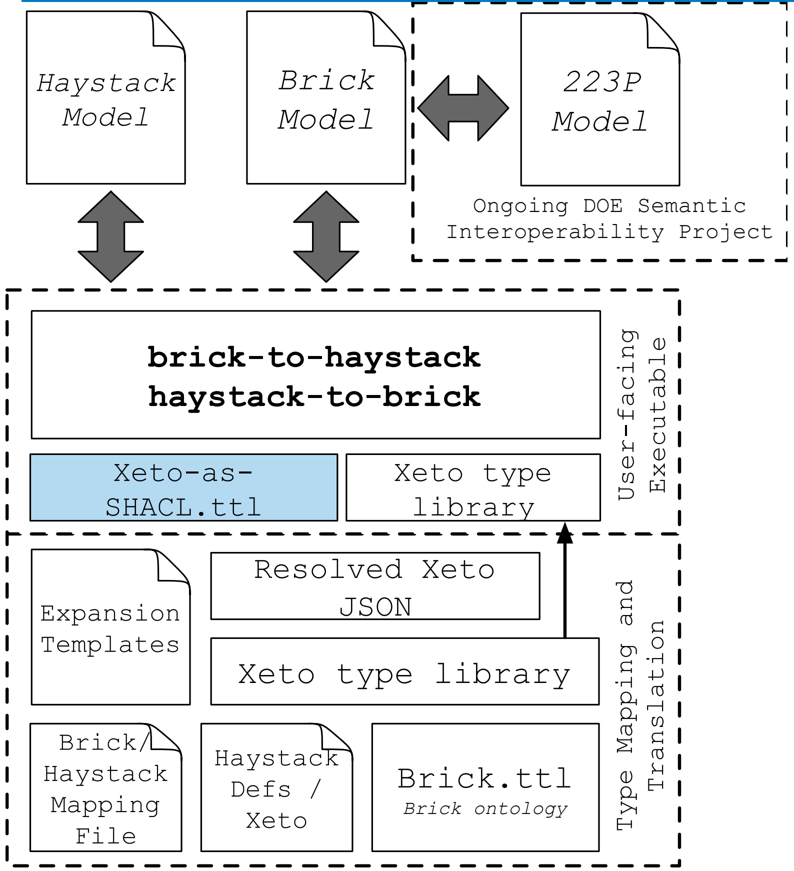
Approach: Xeto – RDF/SHACL Harmonization



- **Two goals**
- Haystack → RDF:
  - Produce Brick and 223P descriptions of Haystack entities
  - Validate Haystack models against 223P standard
- RDF → Haystack:
  - Use Xeto type system to add Haystack tags to Brick, 223P entities
  - Produce valid Haystack models from valid Brick, 223P models

- Along the way:
  - Resolve differences between Brick/Haystack types

Behind the Scenes, Briefly



Some technical details:

- Express Xeto type system using RDF SHACL:
  - SHACL: constraint language for RDF
  - Validates graphs against constraints
  - Can also infer/add new information via rules
- SHACL does three things for us in this context
  - Add Brick (or 223P) types based on existence/structure of Haystack tags
  - Add Haystack tags based on Brick/223P type
  - Ensure that properties, tags, relationships, etc are consistent with the stated types

Behind the Scenes: Haystack → RDF

- Developed a SHACL-based “ontology” for Haystack:
  - Representing Haystack data model in RDF (no semantics!)
  - Allows BuildingMOTIF and SHACL to interact with Haystack

- Xeto → SHACL translator:
  - Express Xeto type definitions as SHACL constraints

```
// Guidline 36 Fan Powered Terminal Unit
G36FanPoweredTerminalUnit : G36Vav {
  fanPowered
  hotWaterHeating
  singleDuct
  points: {
    DischargeFanSpeedCmd
    DischargeFanRunSensor
    DischargeFanRunCmd
    DischargeDamperCmd
    HotWaterValveCmd
    DischargeAirFlowSensor
    DischargeAirTempSensor
  }
}
```

Translating ASHRAE Guideline 36 Xeto types to Brick/RDF

```
<urn:brick-haystack-xeto/ashrae.g36::G36FanPoweredTerminalUnit> a owl:Class,
        sh:NodeShape ;
    sh:property [ a sh:PropertyShape ;
            sh:path ph:hasMarkerTag ;
            sh:qualifiedMinCount 1 ;
            sh:qualifiedValueShape [ sh:hasValue "equip" ] ],
        [ a sh:PropertyShape ;
            sh:path ph:hasMarkerTag ;
            sh:qualifiedMinCount 1 ;
            sh:qualifiedValueShape [ sh:hasValue "vav" ] ],
        # ... etc ...
        [ a sh:PropertyShape ;
            sh:node <urn:brick-haystack-xeto/ph.points::DischargeFanSpeedCmd> ;
            sh:path brick:hasPoint ],
        [ a sh:PropertyShape ;
            sh:node <urn:brick-haystack-xeto/ph.points::DischargeAirTempSensor> ;
            sh:path brick:hasPoint ] ;
        # ... etc ...
```

Behind the Scenes: Haystack → RDF

- Xeto → SHACL translator:
  - Express Xeto type definitions as SHACL constraints
- **Also allows definition of Brick types using Xeto**
  - Haystack constructs → Brick constructs
  - Built on community-constructed “mapping file”
  - Leverage Xeto type system to infer additional equivalences
  - **Import this Xeto library to ensure compatibility with Brick (and thus RDF)**

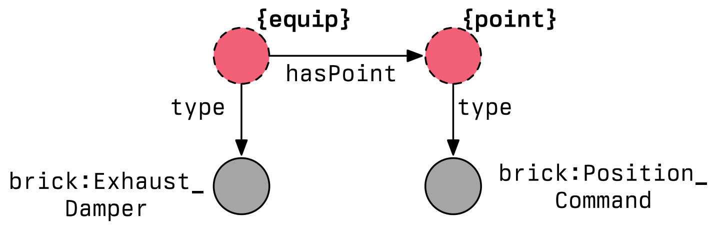
Additional tags inherited from the parent type

```
Brick_Discharge_Air_Temperature_Sensor : Brick_Air_Temperature_Sensor
  <uri:"https://brickschema.org/schema/Brick#Discharge_Air_Temperature_Sensor">
{
  air,
  discharge,
  sensor,
  temp
}
```

Fully qualified Brick “type” embedded in Xeto metadata

Behind the Scenes: Haystack → RDF

- Challenge: how to handle "flat" Haystack entities
  - **Haystack** (sometimes flat): *`exhaust damper position cmd point`*
  - **Brick** (equip, points separate): *`Exhaust Damper hasPoint Damper Position Command`*
- Solution: use BuildingMOTIF templates to generate RDF graphs
  - URI "minting" and Axon lookups to generate reasonable names
  - `id` assigned to the 'point' parameter in the template



BuildingMOTIF template to generate Brick subgraph

```
Brick_Exhaust_Damper_Position_Command : Brick_Command
  <template:"Brick_Exhaust_Damper_Position_Command",
   uri:"urn:___param___#Brick_Exhaust_Damper_Position_Command">
{
  air,
  damper,
  exhaust
}
```

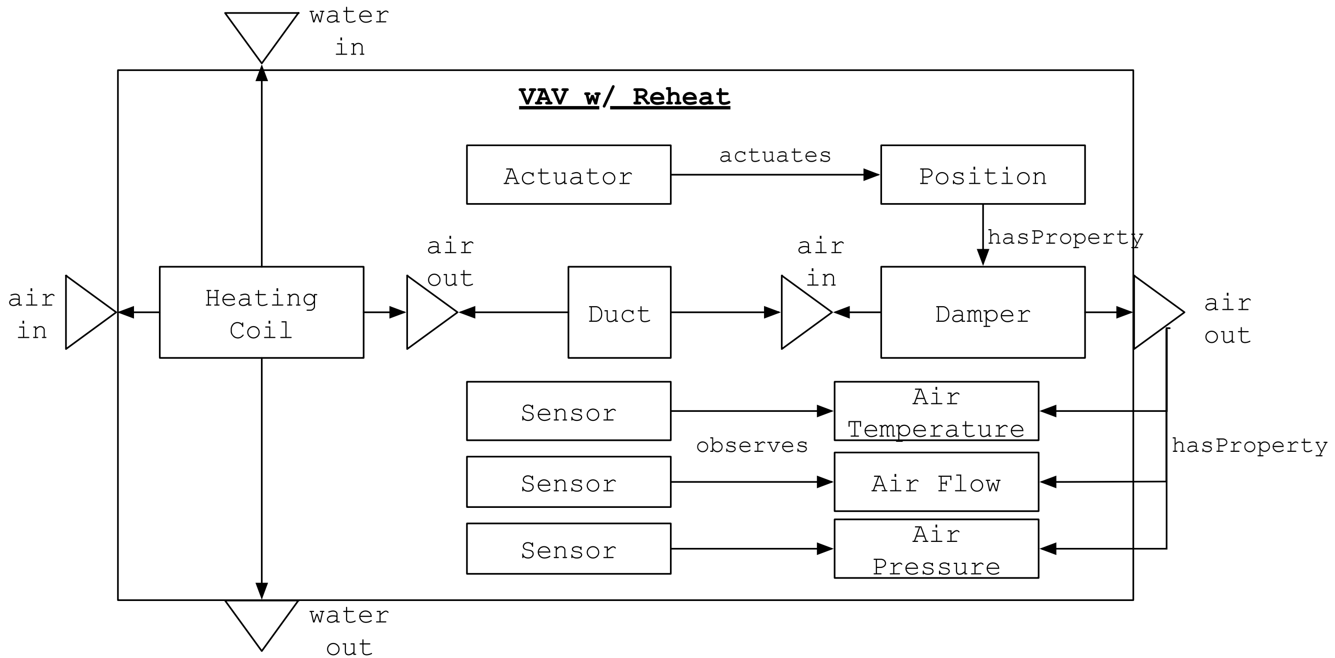
Two POC: Haystack → RDF and RDF → Haystack

- Scenario 1:
  - Have: Existing RDF model (223P and/or Brick)
  - Want: Valid Haystack model

- Scenario 2:
  - Have: Valid Haystack model
  - Want: Brick and/or 223P model

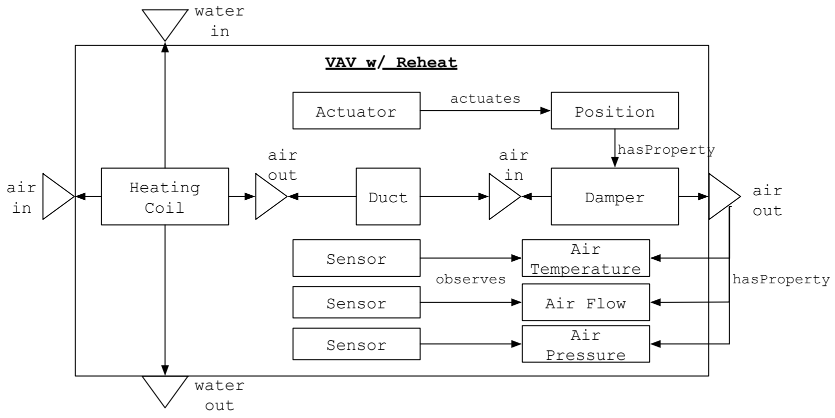
- Not showing running code due to time constraints
  - Online "demo-quality" code at https://github.com/gtfierro/Brick-Haystack-harmonization

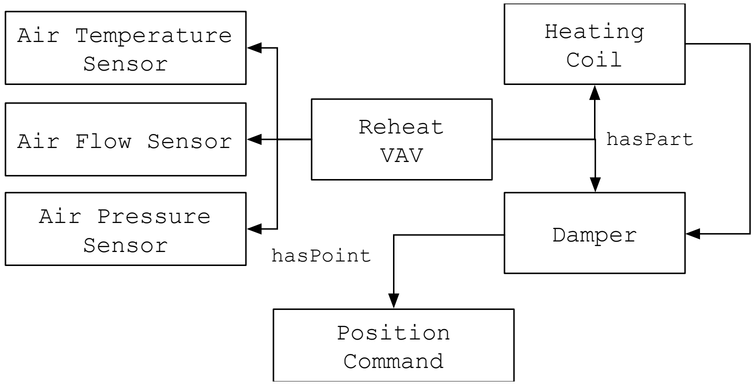
VAV w/ Reheat: 223P



- Directionality, substance, properties propagated through process flows
- Equipment composition, explicit sensor observation relationships
- *Close-to-finished 223P with some details removed*

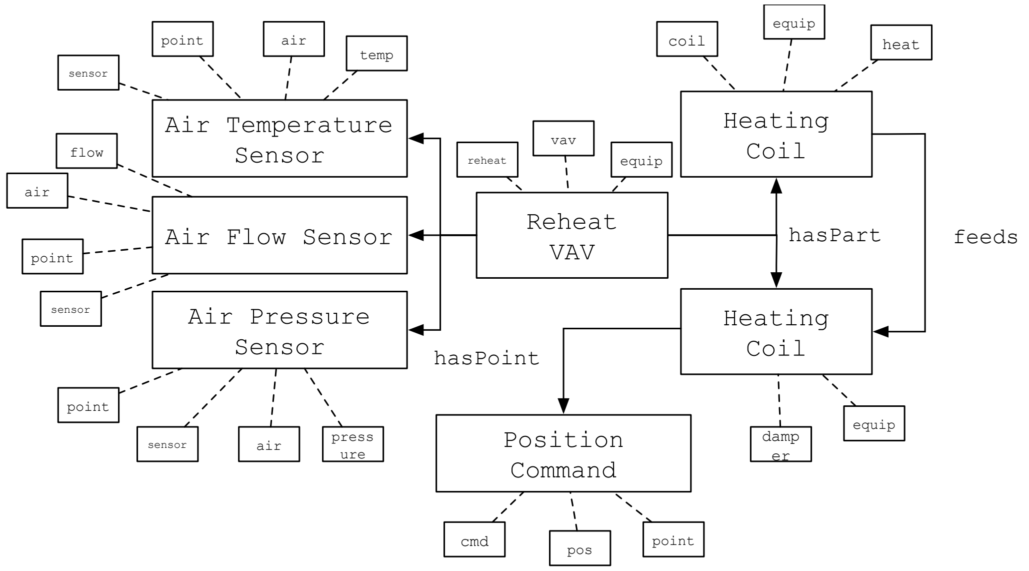
VAV w/ Reheat: 223P → Brick

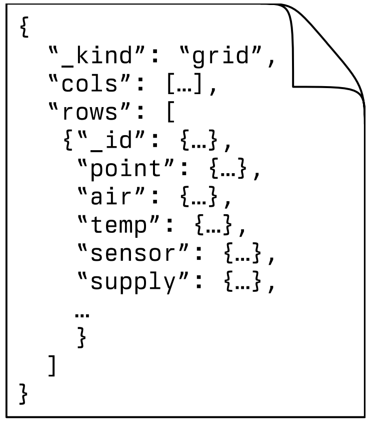




- Brick model captures composition, flow, relationship to BMS points
  - Simplification of the 223P model with more specific names
- Brick model is programmatically generated from the 223P model
  - Uses SHACL to infer types, add necessary relationships
- **Level of abstraction for Brick / Haystack is basically the same**
  - Just need to express the Brick types as Haystack types --- easy thanks to Xeto!

VAV w/ Reheat: 223P → Brick → Haystack

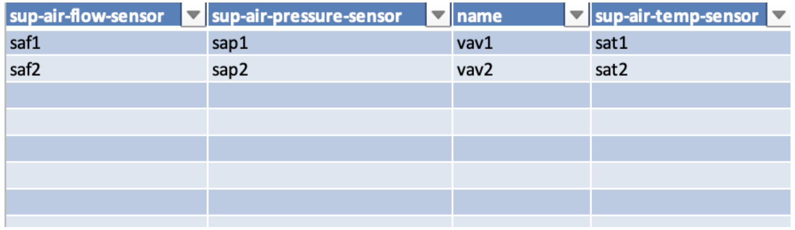




```
{
  "_kind": "grid",
  "cols": […],
  "rows": [
   {"_id": {…},
    "point": {…},
    "air": {…},
    "temp": {…},
    "sensor": {…},
    "supply": {…},
    …
    }
  ]
}
```

- Application of SHACL rules adds Haystack tags to Brick entities
  - Rules constructed automatically from Xeto type definitions
  - Adds ref tags and (*soon!*) value tags
- Haystack import formats (e.g. JSON) generated from the augmented RDF model

End-to-end: BuildingMOTIF-enabled RDF to Haystack



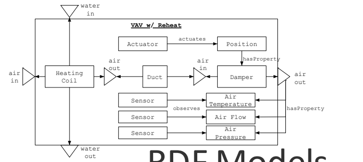
sup-air-flow-sensor	sup-air-pressure-sensor	name	sup-air-temp-sensor
saf1	sap1	vav1	sat1
saf2	sap2	vav2	sat2

Spreadsheet input

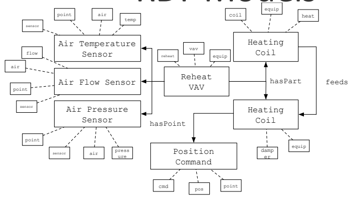
(or BACnet scan, BMS dump, etc…)

Evaluate template

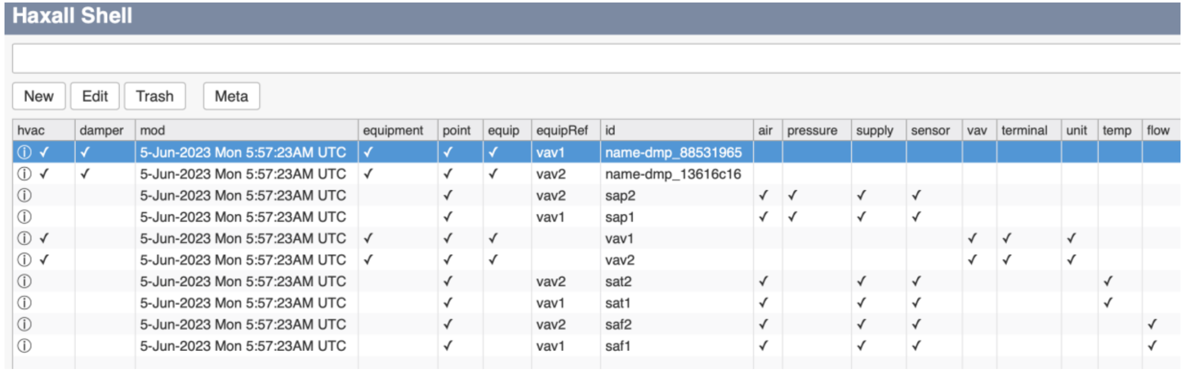
BuildingMOTIF



RDF Models



Xeto-enabled SHACL inference



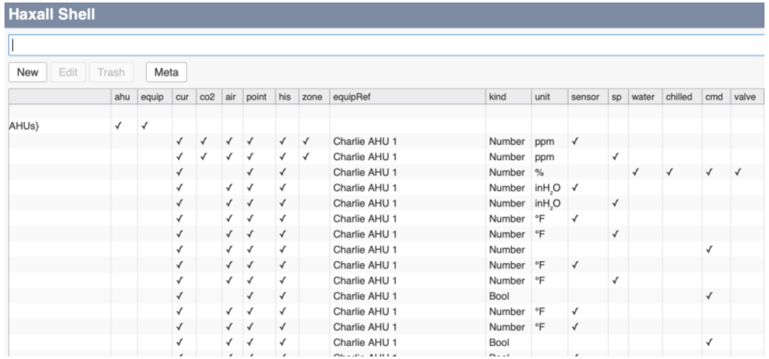
Haxall Shell

New | Edit | Trash | Meta

hvac	damper	mod	equipment	point	equip	equipRef	id	air	pressure	supply	sensor	vav	terminal	unit	temp	flow
ⓘ ✓	✓	5-Jun-2023 Mon 5:57:23AM UTC	✓	✓	✓	vav1	name-dmp_88531965									
ⓘ ✓	✓	5-Jun-2023 Mon 5:57:23AM UTC	✓	✓	✓	vav2	name-dmp_13616c16									
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav2	sap2	✓	✓	✓	✓					
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav1	sap1	✓	✓	✓	✓					
ⓘ ✓		5-Jun-2023 Mon 5:57:23AM UTC	✓	✓	✓		vav1					✓	✓	✓		
ⓘ ✓		5-Jun-2023 Mon 5:57:23AM UTC	✓	✓	✓		vav2					✓	✓	✓		
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav2	sat2	✓		✓	✓				✓	
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav1	sat1	✓		✓	✓				✓	
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav2	saf2	✓		✓	✓					✓
ⓘ		5-Jun-2023 Mon 5:57:23AM UTC		✓		vav1	saf1	✓		✓	✓					✓

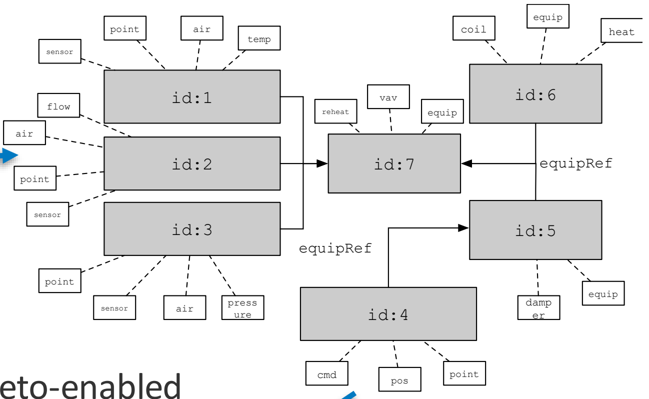
Haystack model

End-to-end: BuildingMOTIF-enabled Haystack to RDF

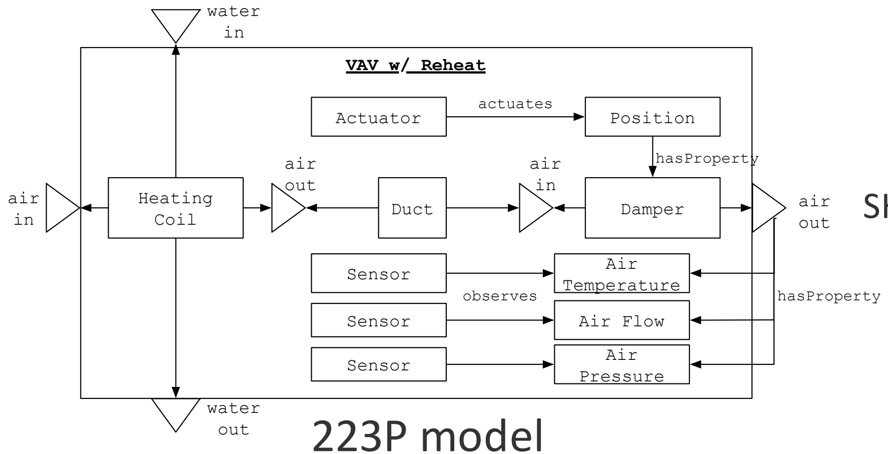


	ahu	equip	cur	co2	air	point	his	zone	equipRef	kind	unit	sensor	sp	water	chilled	cmd	valve
AHUs}	✓	✓															
			✓	✓	✓	✓	✓	✓	Charlie AHU 1	Number	ppm	✓					
			✓	✓	✓	✓	✓	✓	Charlie AHU 1	Number	ppm		✓				
			✓			✓	✓		Charlie AHU 1	Number	%			✓	✓	✓	✓
			✓		✓	✓	✓		Charlie AHU 1	Number	inH$_2$O	✓					
			✓		✓	✓	✓		Charlie AHU 1	Number	inH$_2$O		✓				
			✓		✓	✓	✓		Charlie AHU 1	Number	°F	✓					
			✓		✓	✓	✓		Charlie AHU 1	Number	°F		✓				
			✓		✓	✓	✓		Charlie AHU 1	Number						✓	
			✓		✓	✓	✓		Charlie AHU 1	Number	°F	✓					
			✓		✓	✓	✓		Charlie AHU 1	Number	°F		✓				
			✓			✓	✓		Charlie AHU 1	Bool						✓	
			✓		✓	✓	✓		Charlie AHU 1	Number	°F	✓					
			✓		✓	✓	✓		Charlie AHU 1	Number	°F	✓					
			✓		✓	✓	✓		Charlie AHU 1	Bool						✓	

Haystack RDF model

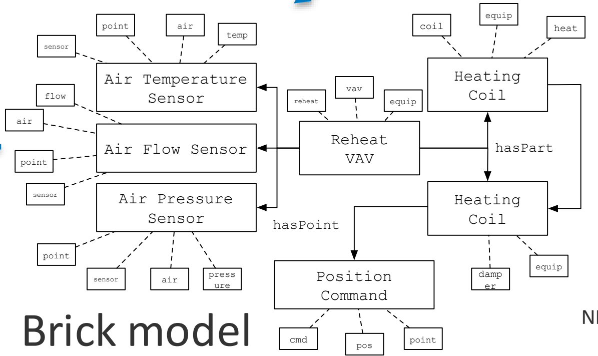


Xeto-enabled SHACL inference



SHACL inference

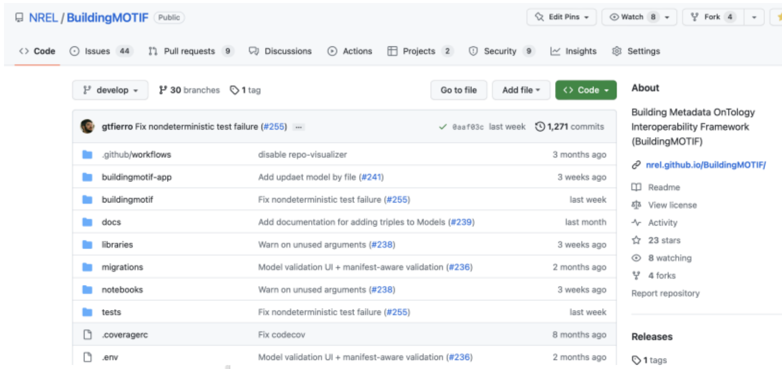
223P model



Brick model

Thank You!

- Please check out BuildingMOTIF! https://github.com/NREL/BuildingMOTIF
- Still in "alpha" but tutorials, Jupyter Notebooks, MVP web interface already available
- Support for Brick, 223P, RealEstateCore and Haystack underway
- Raise issues, leave comments, send emails
  gtfierro@mines.edu
  Avijit.Saha@nrel.gov


NREL / BuildingMOTIF Public

Edit Pins · Watch 8 · Fork 4

Code · Issues 44 · Pull requests 9 · Discussions · Actions · Projects 2 · Security 9 · Insights · Settings

develop · 30 branches · 1 tag · Go to file · Add file · Code

gtfierro Fix nondeterministic test failure (#255) ··· ✓ 0aaf03c last week · 1,271 commits

.github/workflows	disable repo-visualizer	3 months ago
buildingmotif-app	Add updaet model by file (#241)	3 weeks ago
buildingmotif	Fix nondeterministic test failure (#255)	last week
docs	Add documentation for adding triples to Models (#239)	last month
libraries	Warn on unused arguments (#238)	3 weeks ago
migrations	Model validation UI + manifest-aware validation (#236)	2 months ago
notebooks	Warn on unused arguments (#238)	3 weeks ago
tests	Fix nondeterministic test failure (#255)	last week
.coveragerc	Fix codecov	8 months ago
.env	Model validation UI + manifest-aware validation (#236)	2 months ago

About

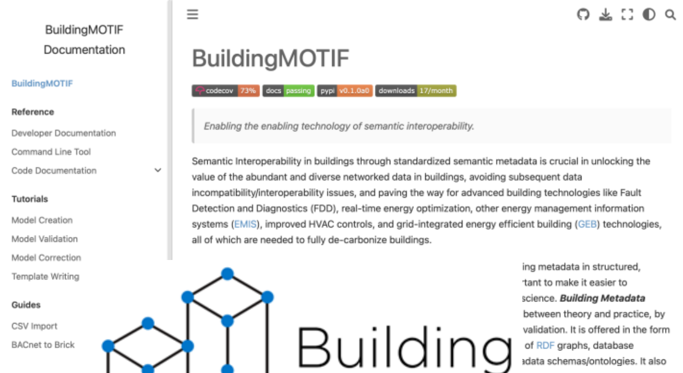
Building Metadata OnTology Interoperability Framework (BuildingMOTIF)

nrel.github.io/BuildingMOTIF/

Readme · View license · Activity · 23 stars · 8 watching · 4 forks · Report repository

Releases

1 tags


BuildingMOTIF Documentation

BuildingMOTIF

Reference
- Developer Documentation
- Command Line Tool
- Code Documentation

Tutorials
- Model Creation
- Model Validation
- Model Correction
- Template Writing

Guides
- CSV Import
- BACnet to Brick

BuildingMOTIF

codecov 73% · docs passing · pypi v0.1.0a0 · downloads 17/month

Enabling the enabling technology of semantic interoperability.

Semantic Interoperability in buildings through standardized semantic metadata is crucial in unlocking the value of the abundant and diverse networked data in buildings, avoiding subsequent data incompatibility/interoperability issues, and paving the way for advanced building technologies like Fault Detection and Diagnostics (FDD), real-time energy optimization, other energy management information systems (EMIS), improved HVAC controls, and grid-integrated energy efficient building (GEB) technologies, all of which are needed to fully de-carbonize buildings.

ing metadata in structured,
tant to make it easier to
science. ***Building Metadata***
between theory and practice, by
validation. It is offered in the form
of RDF graphs, database
data schemas/ontologies. It also

Building
MOTIF