

Tooling for Creation, Validation and Integrated Use of ASHRAE 223P, Brick and Haystack Semantic Metadata Models

Dr. Gabe Fierro

<https://gtf.fyi>

Learning Objectives

- - Attendees will be able to conceptualize how the ASHRAE 223P, Brick, Haystack, and RealEstateCore ontologies can cooperatively model a building to support diverse use cases
- - Attendees will gain basic familiarity with NREL's BuildingMOTIF tool and learn how to use BuildingMOTIF to create a metadata model using 223P
- - Attendees learn how to use BuildingMOTIF to validate a semantic metadata model against a set of representative use cases

Semantic Metadata Models for Buildings

Haystack

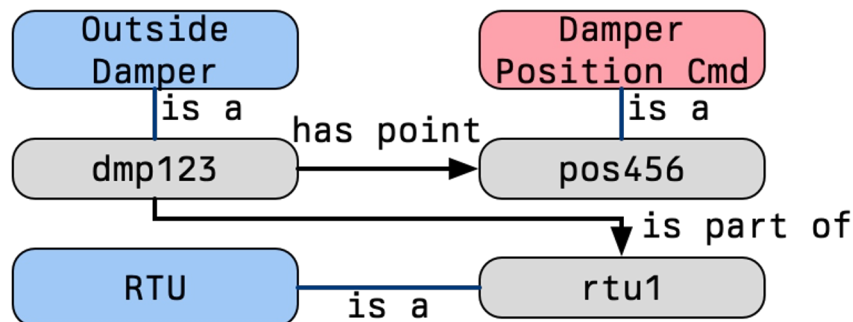
- Uses tags to flexibly describe data/assets
- Lacks consistency between implementations
- Meaning requires *human interpretation*

id: 'd83664ec RTU-1 OutsideDamper'

air: ✓
 cmd: ✓
 cur: ✓
 damper: ✓
 outside: ✓
 point: ✓

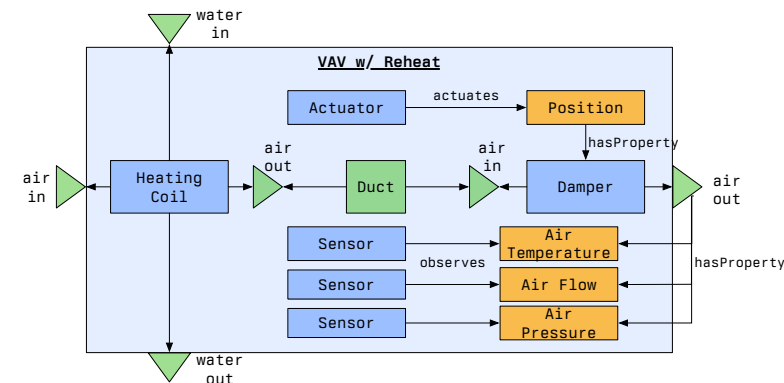
Brick and RealEstateCore (Rec)

- Explicit and formal descriptions of assets
- Expressive relationships
- Extensible and comprehensive
 - Hundreds of classes
- Not *standardized*



ASHRAE 223P

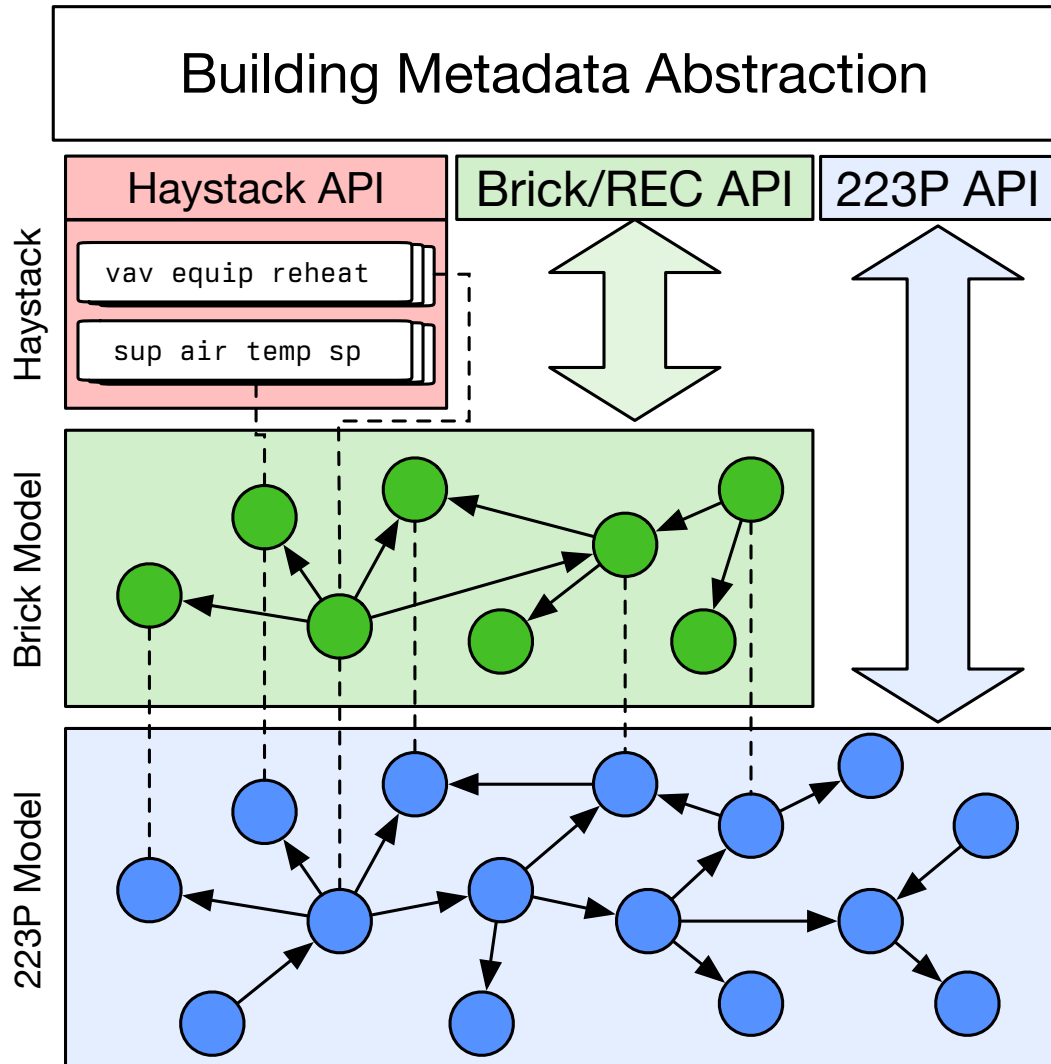
- Fully elaborated connection topologies
- Cross-subsystem and inter-domain models:
 - Electrical, mechanical, plumbing, ...
- Emphasizes precision to enable semantic interop
- Harder to extend due to standards process



Metadata Models can be Complementary

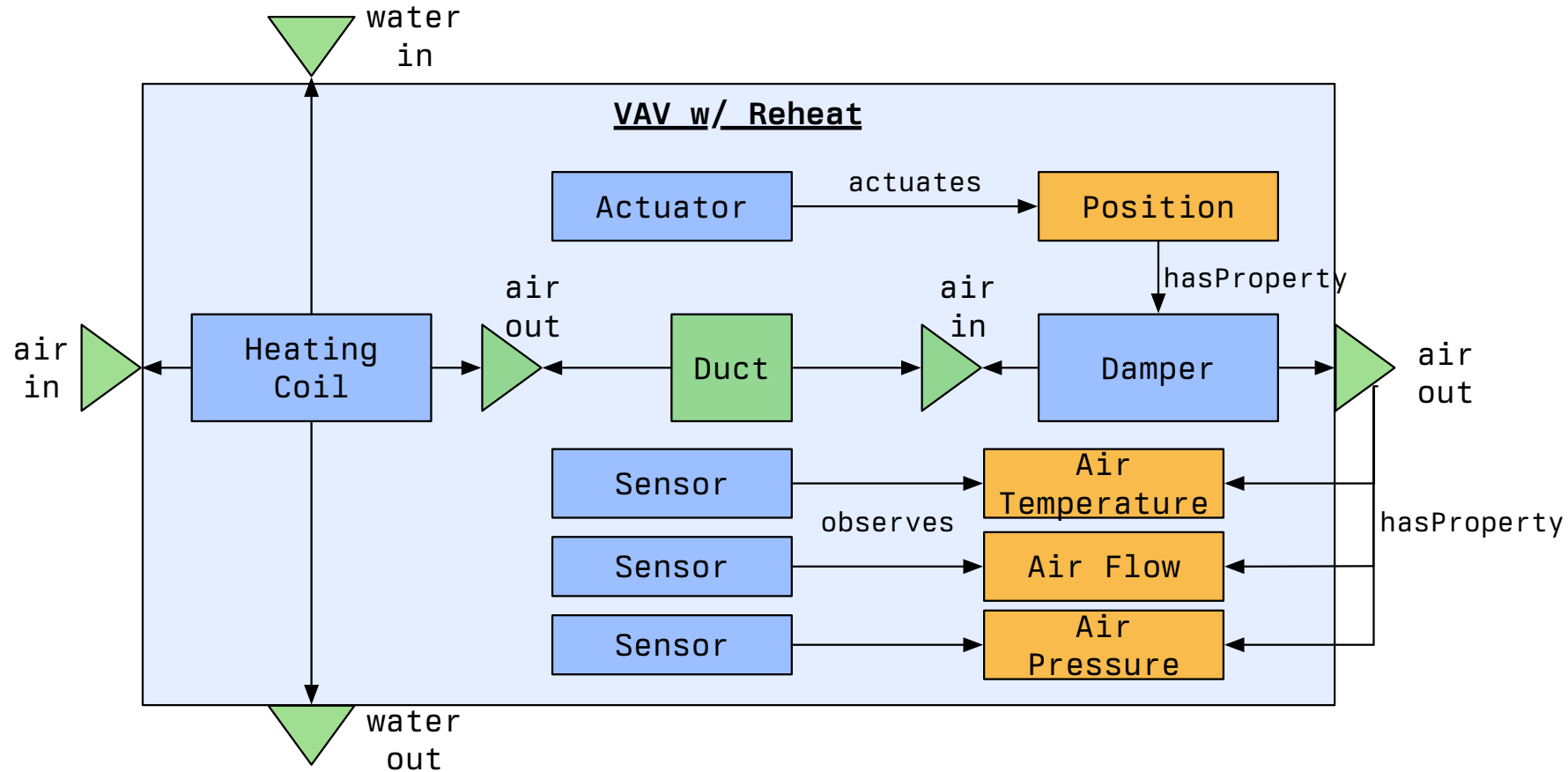
- Key idea is to model a building at multiple levels of abstraction
 - Reach for as much detail as you need to get the job done
- **ASHRAE 223P**
 - Detailed and precise
 - → larger models and more complex queries
- **Brick and RealEstateCore**
 - Lean on common understandings and nomenclature
 - → less precise, but easier to query
- **Project Haystack**
 - Simple tag/document-based data model
 - → simpler query + API, but not as expressive

Complementary Metadata Designs



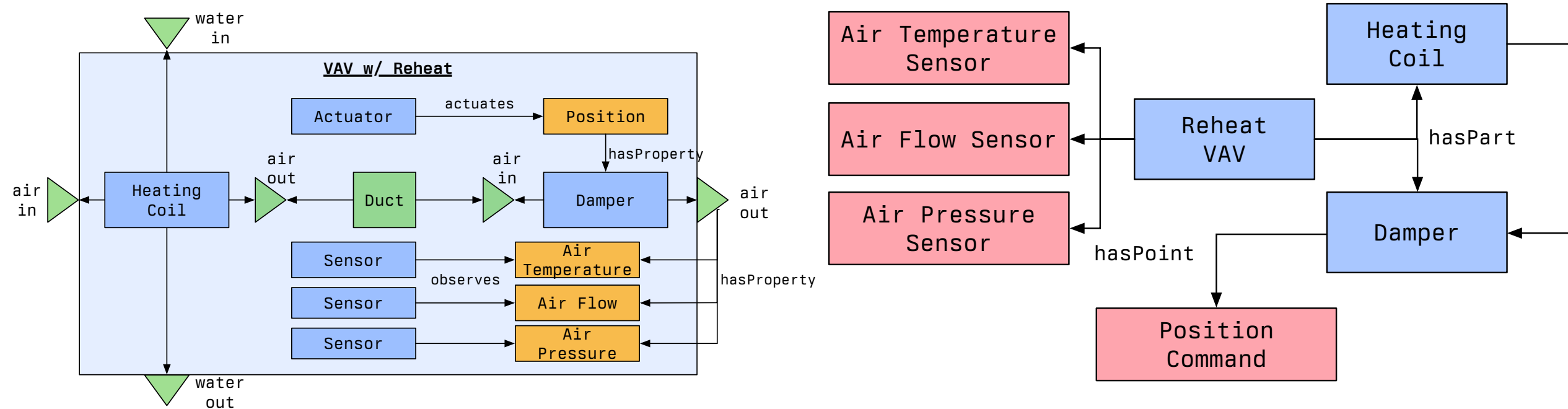
- 223P models the fundamental components of a building
- Brick/REC provide human-facing application-centric vocabulary
 - **Programmatically generated from 223P**
- Haystack provides one possible high-level abstraction over RDF / graph models
 - **Programmatically generated from Brick/REC**

VAV w/ Reheat: 223P



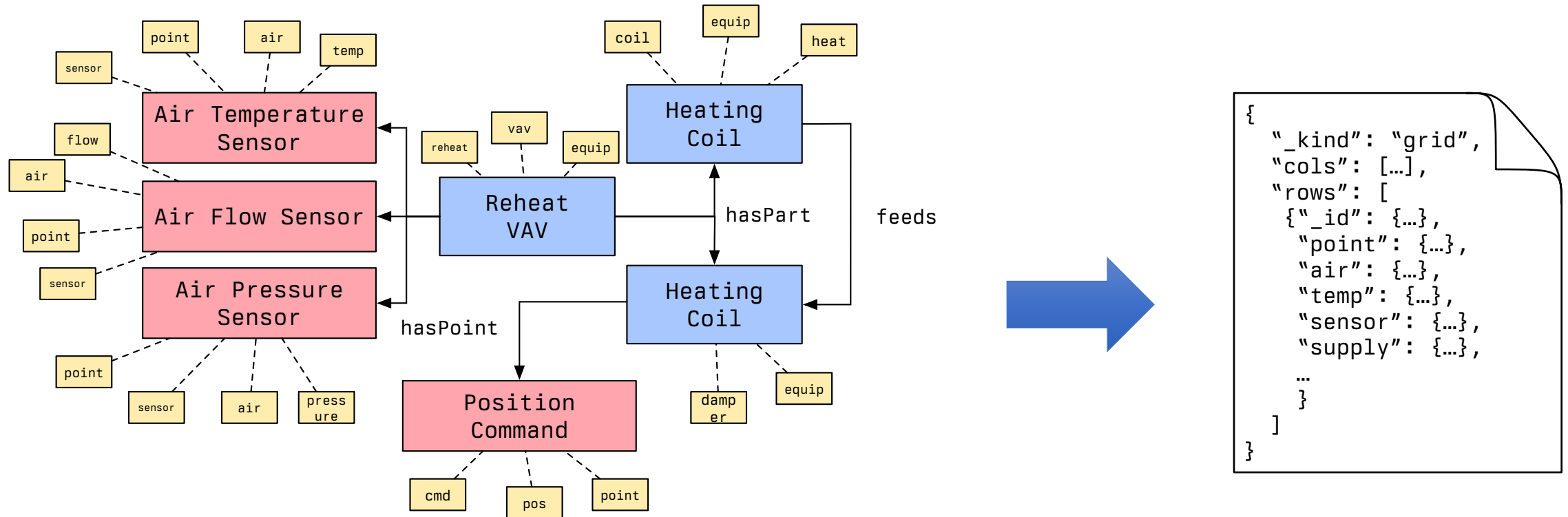
- Directionality, substance, properties propagated through process flows
- Equipment composition, explicit sensor observation relationships
- *Some details removed for ease of visualization*

VAV w/ Reheat: 223P → Brick/REC



- 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

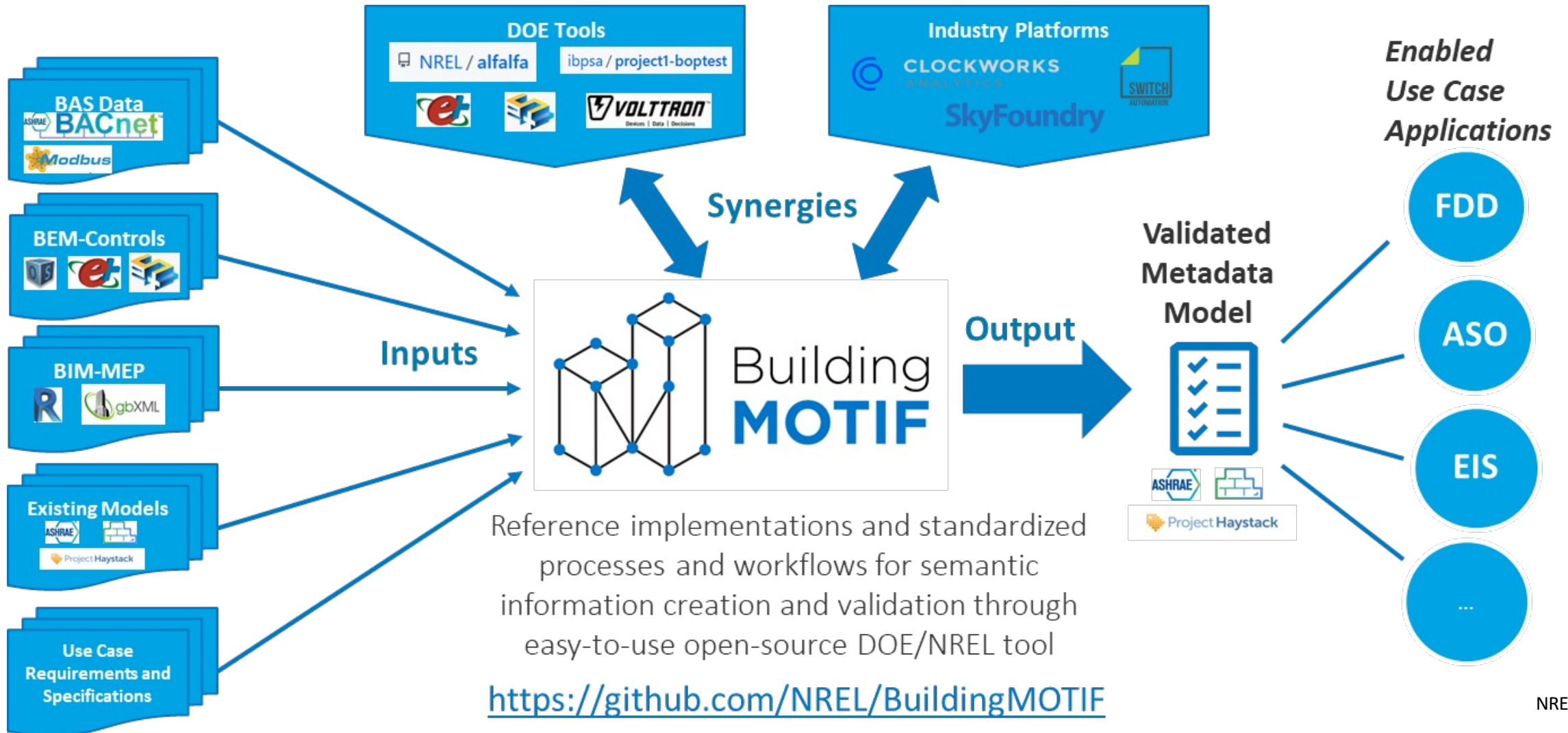
VAV w/ Reheat: 223P → Brick/REC → Haystack



- 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

DOE/NREL Open-Source Tool: BuildingMOTIF

MOTIF = Metadata OnTology Interoperability Framework



How to Work With Semantic Metadata Models

- Creating a semantic metadata model of a building
 - Hide complexity of working with RDF models
 - Help to direct/prioritize human effort
- Validating a semantic metadata model of a building
 - Ensure model contains correct / enough metadata to configure controls, FDD rules, digital twins, ...
- *Two parts of a larger workflow*

Shapes: Formalizing Application Metadata Requirements

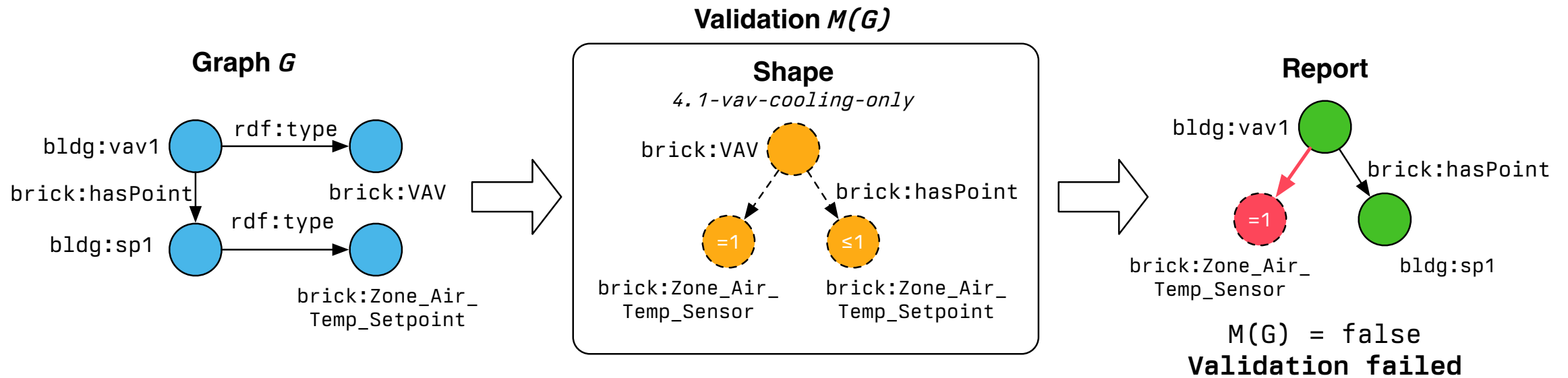
4.2 VAV Terminal Unit with Reheat

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

- Come in many different forms...
- **At right:** point lists from ASHRAE Guideline 36
- SOO for high-perf forced-air systems
- How to ensure 223P, Brick, REC, etc model contains the right metadata?

Shapes: Formalizing Application Metadata Requirements

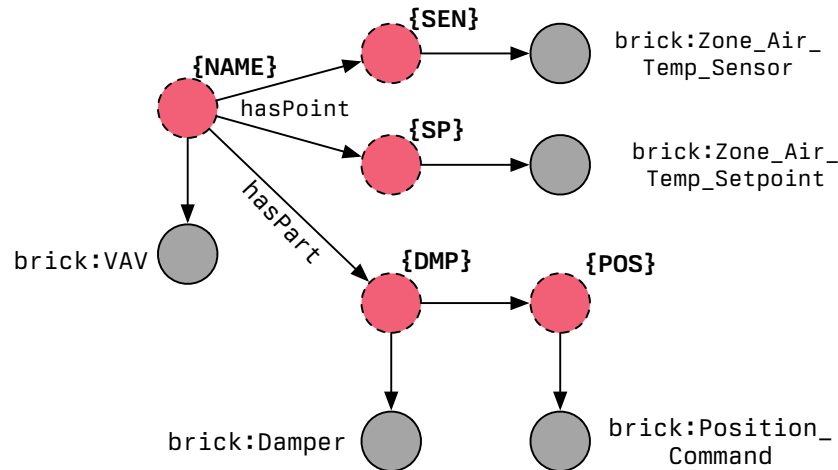
- **Shape:** set of constraints, requirements, conditions on parts of the graph
- Function returns T/F when evaluated on a graph
 - Plus some information about "what went wrong"



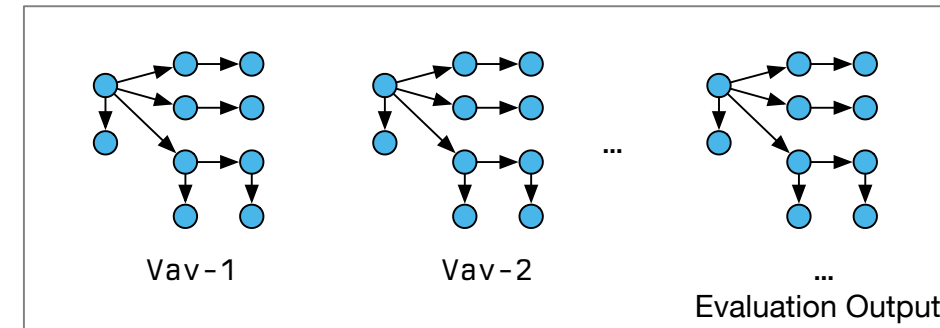
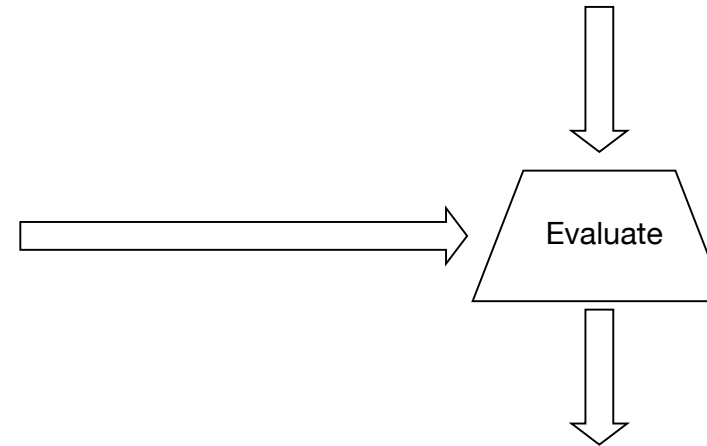
- Defined using W3C SHACL standard

Templates: Simplifying Model Creation

- **Template:** function that generates a graph



{NAME}	{SEN}	{SP}	{DMP}	{POS}
Vav-1	ZTS-A	ZTSP-A	DMP-12	POS-12
Vav-2	ZTS-B	ZTSP-B	DMP-23	POS-23
...



- Simplifies RDF graph creation
 - Tabular input: web forms! Spreadsheets!
 - Easier to integrate (pull from BACnet, etc)

Quick Demonstration

1. Use BuildingMOTIF to create a 223P model for a simple HVAC system:
 - 1 Makeup Air Unit feeding 1 VAV with reheat
 - Include the VAV hot water coil
 - Include all the filters, fans, coils, etc in the MAU
 - *Without touching any RDF!*
2. Use BuildingMOTIF to ensure the model contains sufficient metadata to support a Guideline 36 sequence

```
from rdflib import Namespace
from buildingmotif import BuildingMOTIF
from buildingmotif.dataclasses import Library, Model, Template
from buildingmotif.namespaces import bind_prefixes

# connect BuildingMOTIF to a SQLite database (also supports Postgres)
bm = BuildingMOTIF("sqlite://")

# load in 223P definitions
s223 = Library.load(ontology_graph=r"ontologies/223p.ttl")

# load in an NREL template library
nrel = Library.load(directory="libraries/ashrae/223p/nrel-templates")

# load in helper library to assist with common model constraints
constraints = Library.load(ontology_graph="constraints/constraints.ttl")

# find relevant templates by browsing BuildingMOTIF web interface
mau_tmpl = nrel.get_template_by_name("makeup-air-unit")
rvav_tmpl = nrel.get_template_by_name("vav-reheat")
zone_tmpl = nrel.get_template_by_name("hvac-zone")
duct_tmpl = nrel.get_template_by_name("duct")
```

```
from rdflib import Namespace

# create a namespace to 'contain' all building entities
BLDG = Namespace("urn:ashrae_example/")
# create a new metadata model in BuildingMOTIF
bldg = Model.create(BLDG)

# instantiate the templates to create the 3 main entities
# These templates already contain the common components -- we can give them names if we want
MAU1 = mau_tmpl.evaluate({'name': BLDG.MAU1}, warn_unused=False)
RVAV1 = rvav_tmpl.evaluate({'name': BLDG.RVAV1}, warn_unused=False)
ZoneA = zone_tmpl.evaluate({'name': BLDG.MAU1_ZoneA}, warn_unused=False)

# It's all just plain Python, so I can create new functions that automate
# the "boring" stuff

def make_air_connection(t1: Template, from_param: str, t2: Template, to_param: str):
    t1, duct = link(t1, from_param, duct_tmpl, 'a')
    t2, duct = link(t2, to_param, duct, 'b')
    add_to_model(duct)
    return t1, t2

# link the MAU to the RVAV
MAU1, RVAV1 = make_air_connection(MAU1, 'air-supply', RVAV1, 'air-in')
# link the RVAV to the Zone
RVAV1, ZoneA = make_air_connection(RVAV1, 'air-out', ZoneA, 'air-in')

add_to_model(MAU1, RVAV1, ZoneA)
```



```
res = bldg.validate([s223, constraints])
print(f"Is model valid? {res.valid}")
print("Reasons why not:")
for r in res.diffset:
    print(r.reason())
```

Is model valid? False

Reasons why not:

urn:ashrae_example/outside-air_5f30 needs exactly 1 instance of s223:Connection on path s223:connectsThrough


```

# ensure 1 AHU connected to 1 RVAV as defined above
manifest:ahu-count a sh:NodeShape ;
  sh:message "Model has 1 AHU" ;
  sh:targetNode manifest: ;
  constraint:exactCount 1 ;
  constraint:node [
    a sh:NodeShape ;
    sh:class s223:AirHandlingUnit ;
    sh:property [
      sh:path s223:connectedTo ;
      sh:qualifiedValueShape [ sh:class manifest:ReheatVAV ] ;
      sh:qualifiedMinCount 1 ; sh:qualifiedMaxCount 1 ;
    ] ;
  ] ;

```

create our own "reheat VAV" type

```

manifest:ReheatVAV a sh:NodeShape, owl:Class ;
  sh:class s223:SingleDuctTerminal ;
  sh:property [
    sh:path s223:contains ;
    sh:qualifiedValueShape [ sh:class s223:HeatingCoil ] ;
    sh:qualifiedMinCount 1 ; sh:qualifiedMaxCount 1 ;
  ] ;
  sh:property [
    sh:path s223:connectedTo ;
    sh:qualifiedValueShape [ sh:class s223:Zone ] ;
    sh:qualifiedMinCount 1 ; sh:qualifiedMaxCount 1 ;
  ] ;

```

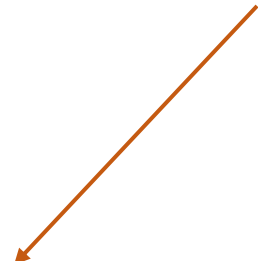
```

sh:node g36:pointlist_4_2 ; # require that the RVAVs support G36 pointlist

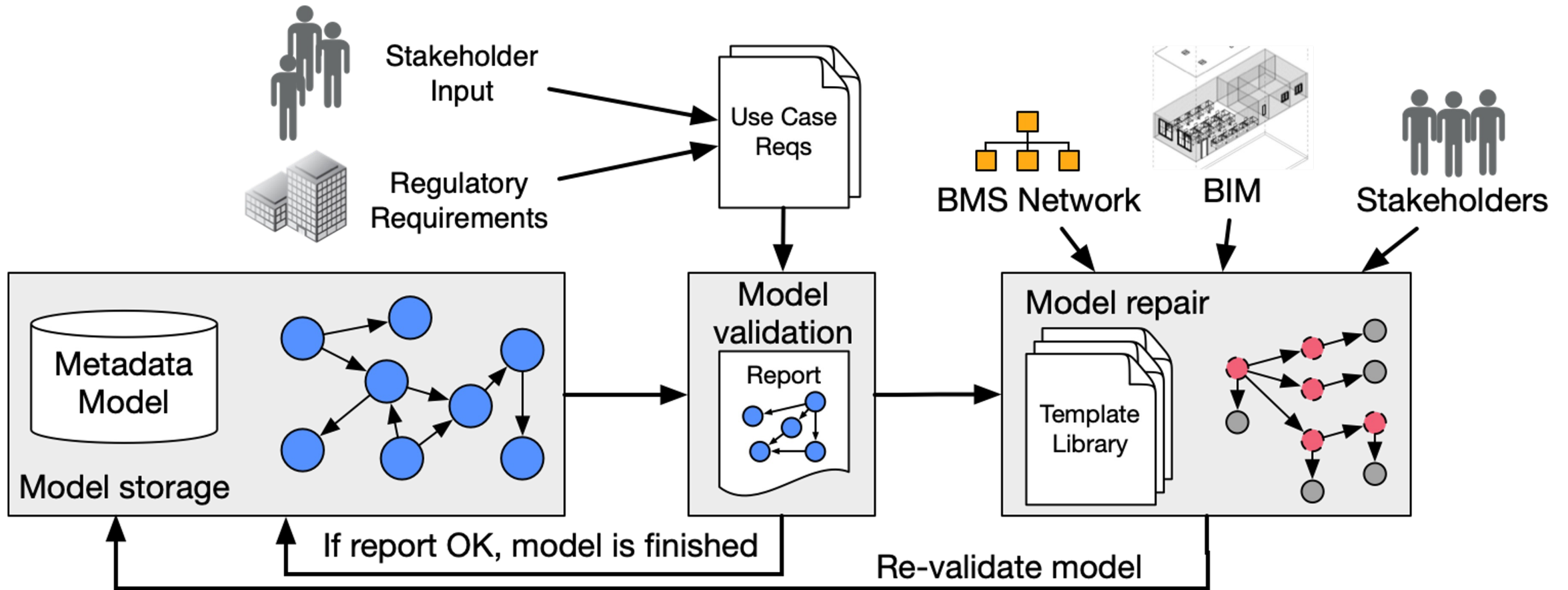
```

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



BuildingMOTIF Model Create/Validate Workflow



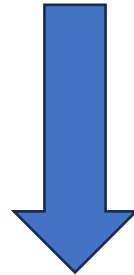
- Incorporate formal use case requirements into iterative workflow
- Ensure that delivered metadata model fulfills all use cases
- Automate / simplify authoring through templates, imports from other sources

```
manifest = Library.load(ontology_graph='manifest.ttl')
res = bldg.validate([s223, constraints, manifest])
print(f"Is model valid? {res.valid}")
print("Reasons why not:")
for r in res.diffset:
    print(r.reason())
```

Is model valid? False

Reasons why not:

- urn:ashrae_example/RVAV1 needs exactly 1 instance of g36:ZoneTemperatureSensor on s223:contains
 - urn:ashrae_example/RVAV1 needs exactly 1 instance of g36:DischargeAirTemperatureSensor on s223:contains
 - urn:ashrae_example/RVAV1 needs exactly 1 instance of g36:HeatingSignal on s223:contains
- ...



BuildingMOTIF generates automatic
“repairs” for the model

```
for r in res.as_templates():
    print(r.name, r.parameters)
```

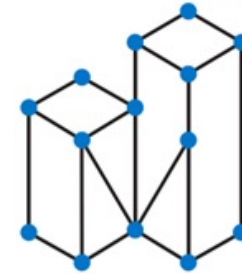
```
fix-missing-ZoneTemperatureSensor {'name', 'zone'}
fix-missing-DischargeAirTemperatureSensor {'name', 'terminal_unit'}
...
```

End-to-end: Building MOTIF-enabled RDF to Haystack

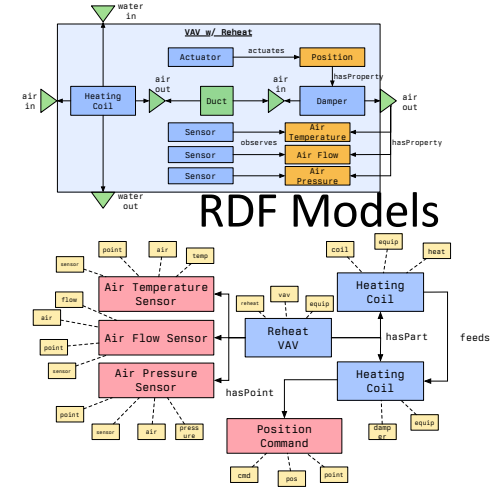
sup-air-flow-sensor	sup-air-pressure-sensor	name	sup-air-temp-sensor
saf1	sap1	vav1	sat1
saf2	sap2	vav2	sat2

Spreadsheet input
(or BACnet scan, BMS dump, etc...)

Evaluate
template



Building
MOTIF



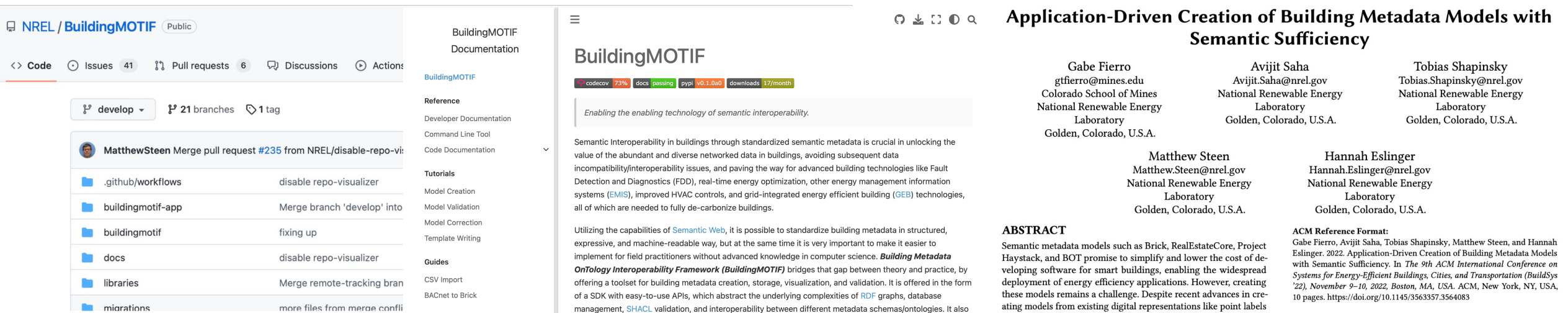
Xeto-enabled
SHACL inference

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

Haystack model

Open Source Code!

- BuildingMOTIF is open source (*and in pre-alpha*)
 - <https://github.com/NREL/BuildingMOTIF>
 - Released under permissive BSD 3 Clause license
 - Tutorials, notebooks, demo web interface
 - Support for Brick, 223P, RealEstateCore and Haystack underway
- Check out other tools on <https://open223.info>



The screenshot shows the GitHub repository for BuildingMOTIF on the left and a paper abstract on the right. The GitHub page includes repository statistics (73% code coverage, 17 downloads/month), a list of files (e.g., .github/workflows, buildingmotif-app), and a pull request from MatthewSteen. The paper abstract is titled "Application-Driven Creation of Building Metadata Models with Semantic Sufficiency" and lists authors Gabe Fierro, Avijit Saha, Tobias Shapinsky, Matthew Steen, and Hannah Eslinger, all from the National Renewable Energy Laboratory. The abstract discusses the importance of semantic interoperability in buildings and the role of BuildingMOTIF in addressing this challenge.

BuildingMOTIF
Documentation

BuildingMOTIF
73% codecov 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.

Utilizing the capabilities of *Semantic Web*, it is possible to standardize building metadata in structured, expressive, and machine-readable way, but at the same time it is very important to make it easier to implement for field practitioners without advanced knowledge in computer science. *Building Metadata Ontology Interoperability Framework (BuildingMOTIF)* bridges that gap between theory and practice, by offering a toolset for building metadata creation, storage, visualization, and validation. It is offered in the form of a SDK with easy-to-use APIs, which abstract the underlying complexities of *RDF* graphs, database management, *SHACL* validation, and interoperability between different metadata schemas/ontologies. It also

Application-Driven Creation of Building Metadata Models with Semantic Sufficiency

Gabe Fierro
gferro@mines.edu
Colorado School of Mines
National Renewable Energy
Laboratory
Golden, Colorado, U.S.A.

Avijit Saha
Avijit.Saha@nrel.gov
National Renewable Energy
Laboratory
Golden, Colorado, U.S.A.

Tobias Shapinsky
Tobias.Shapinsky@nrel.gov
National Renewable Energy
Laboratory
Golden, Colorado, U.S.A.

Matthew Steen
Matthew.Steen@nrel.gov
National Renewable Energy
Laboratory
Golden, Colorado, U.S.A.

Hannah Eslinger
Hannah.Eslinger@nrel.gov
National Renewable Energy
Laboratory
Golden, Colorado, U.S.A.

ABSTRACT
Semantic metadata models such as Brick, RealEstateCore, Project Haystack, and BOT promise to simplify and lower the cost of developing software for smart buildings, enabling the widespread deployment of energy efficiency applications. However, creating these models remains a challenge. Despite recent advances in creating models from existing digital representations like point labels

ACM Reference Format:
Gabe Fierro, Avijit Saha, Tobias Shapinsky, Matthew Steen, and Hannah Eslinger. 2022. Application-Driven Creation of Building Metadata Models with Semantic Sufficiency. In *The 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22)*, November 9–10, 2022, Boston, MA, USA. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3563357.3564083>

Thanks!

- Questions, feedback, comments?
 - <https://home.gtf.fyi>
 - gtfiero@mines.edu