2024 ASHRAE WINTER CONFERENCE
CHICAGO, JAN 20-24 | AHR EXPO, JAN 22-24

Seminar 55: True Building Controls Interoperability: New digital solutions enabled by proposed ASHRAE standards 223P and 231P

Digitizing the delivery of controls and analytics through the proposed ASHRAE standards 223p and 231p

Dr. Gabe Fierro

Colorado School of Mines

National Renewable Energy Laboratory

gtfierro@mines.edu

Learning Objectives

- Learn about the key features of the upcoming interoperability ASHRAE standards 223p and 231p.
- Understand how these two standards will be integrated and synergize with ASHRAE 135 (BACnet) and Guideline 36 (Best in class control sequences).
- Learn about practical examples of the implementation of these standards in real buildings.
- Explore opportunities to use these standards in your products, services or workflows.

Acknowledgements

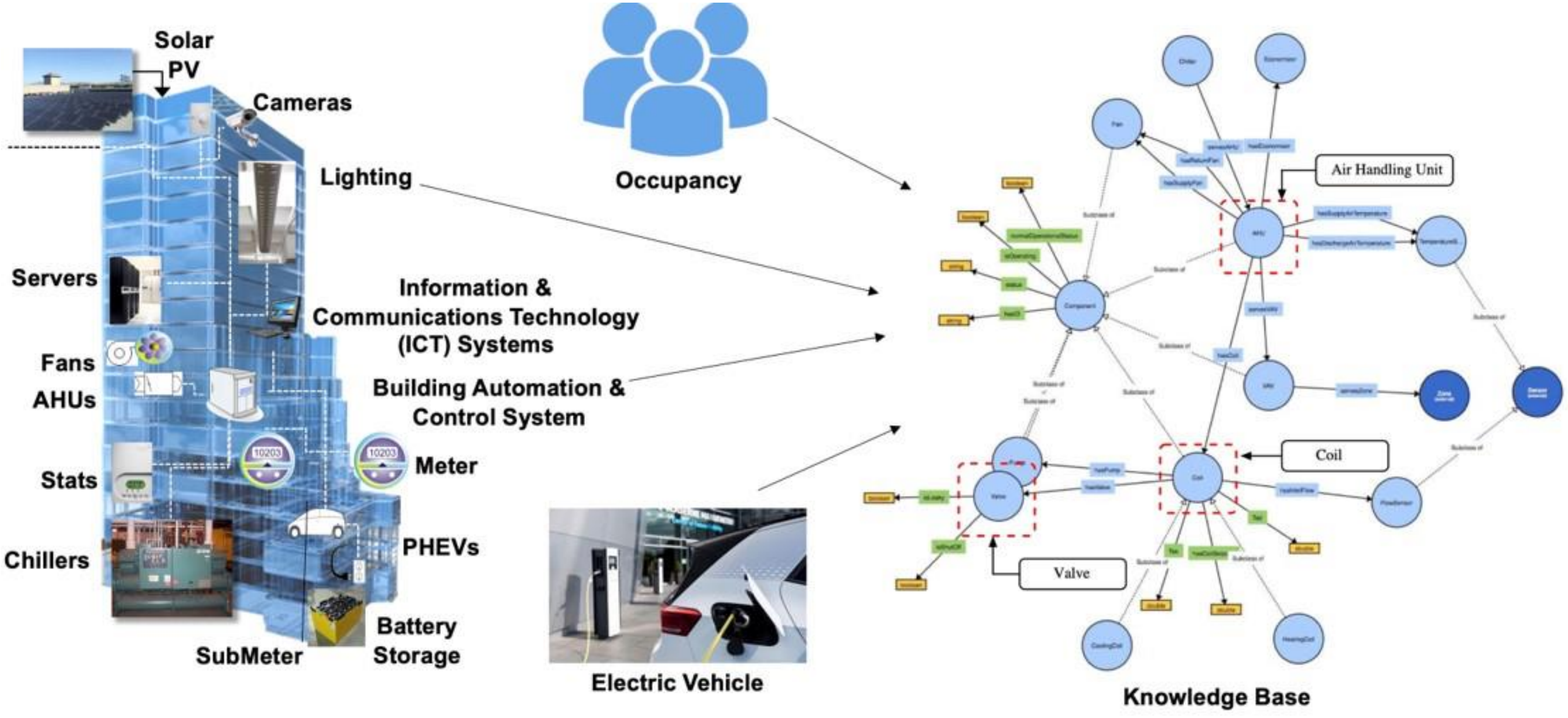
- Avijit Saha, NREL
- Matthew Steen, NREL
- Tobias Shapinsky, NREL
- Steve Bushby, NIST
- Joel Bender, Cornell University
- Brian Walker, DOE
- Amir Roth, DOE

Outline/Agenda

- What is 223P?
- Modeling a simple HVAC system
- 223P provides context for data
- Example FDD application

What is ASHRAE 223P?

ASHRAE 223P standard defines concepts and methodologies to create interoperable, **machine-readable semantic models** for representing building information for analytics, control, and automation.



- Control
- Energy Auditing
- Fault Detection and Diagnostics
- Commissioning
- Smart Grid Interactions
- **ASHRAE 231P**

Physical, logical systems ← represents ← **Semantic Model** ← access ← **Applications**

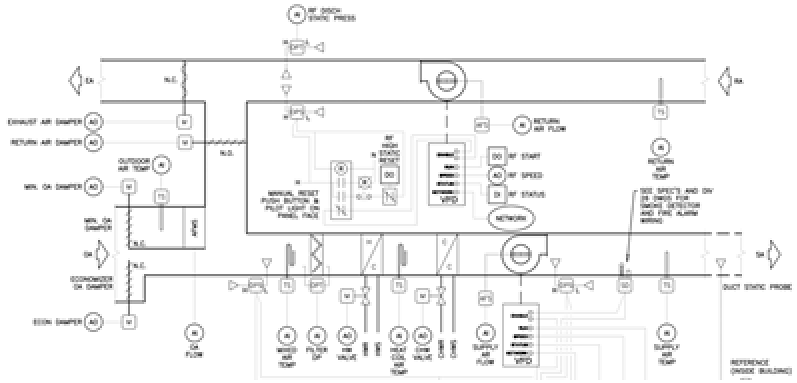
Framing the Problem

- **Applications** define data, system requirements using convention-driven, but non-standard language
  - *Example (right): point lists from ASHRAE Guideline 36*

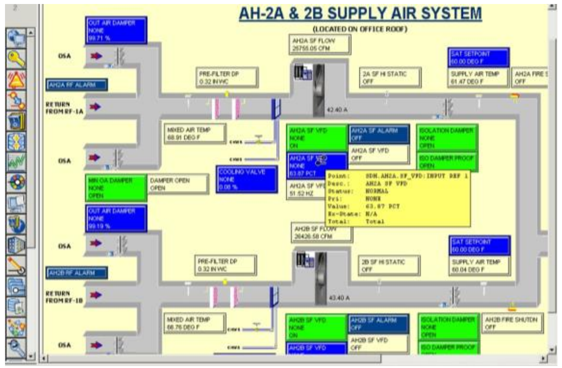
- Necessary data is spread over many (potentially) noisy, non-standard, and possibly non-digital sources

4.2 VAV Terminal Unit with Reheat

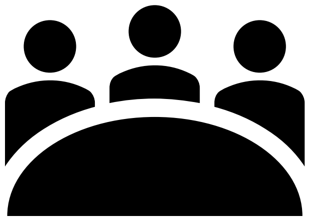
Required?	Description	Type	Device
R	VAV box damper position	AO OR two DOs	Modulating actuator OR Floating actuator
R	Heating signal	AO OR two DOs	Modulating valve OR Floating actuator OR Modulating electric heating coil
R	Discharge airflow	AI	DP transducer connected to flow sensor
R	Discharge air temperature (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



Mechanical Diagrams: human-readable but non-standard



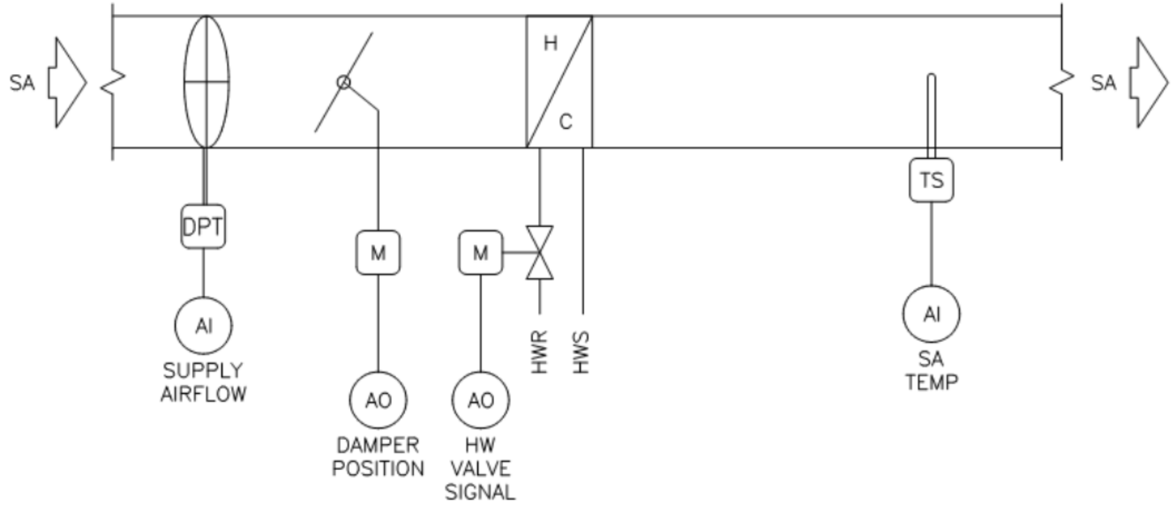
BMS labels and graphics: undocumented naming conventions



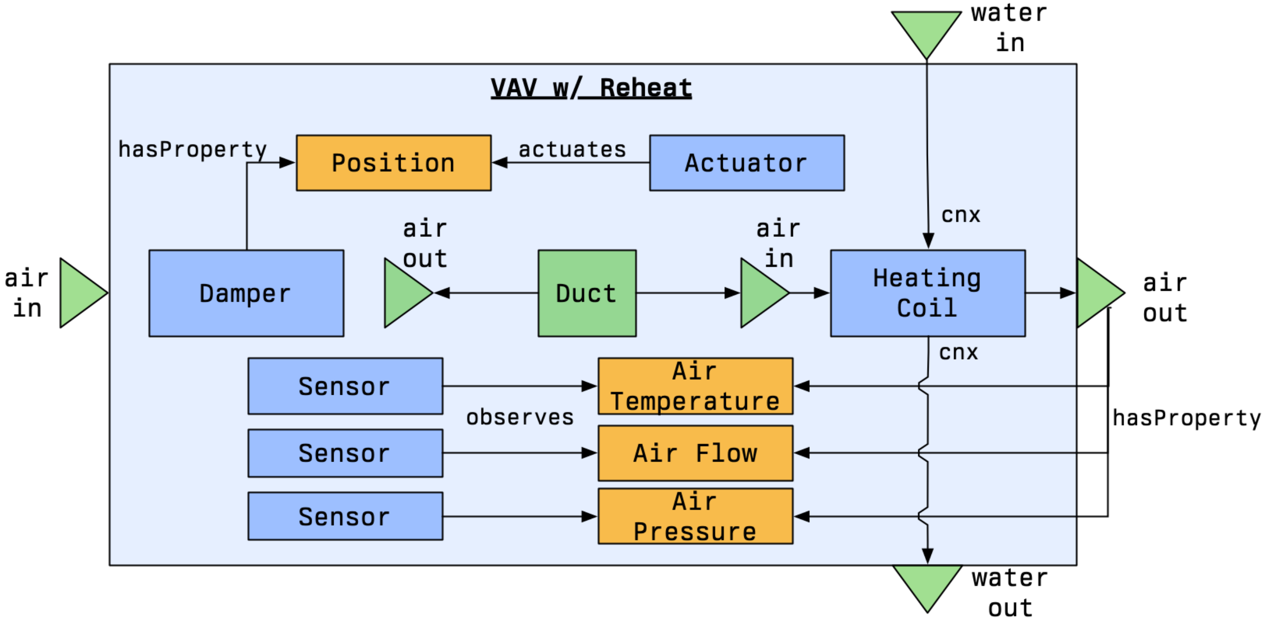
Facility managers, maintenance staff, and others hold implicit knowledge

Ontologies and Semantic Models

- A **semantic model** is a digital graph-based representation of a building
  - *Entities*: equipment, sensors, actuators, properties, connections
  - Includes useful attributes of these entities
  - Models how entities relate to each other and compose into systems



VAV with Reheat Mechanical Diagram

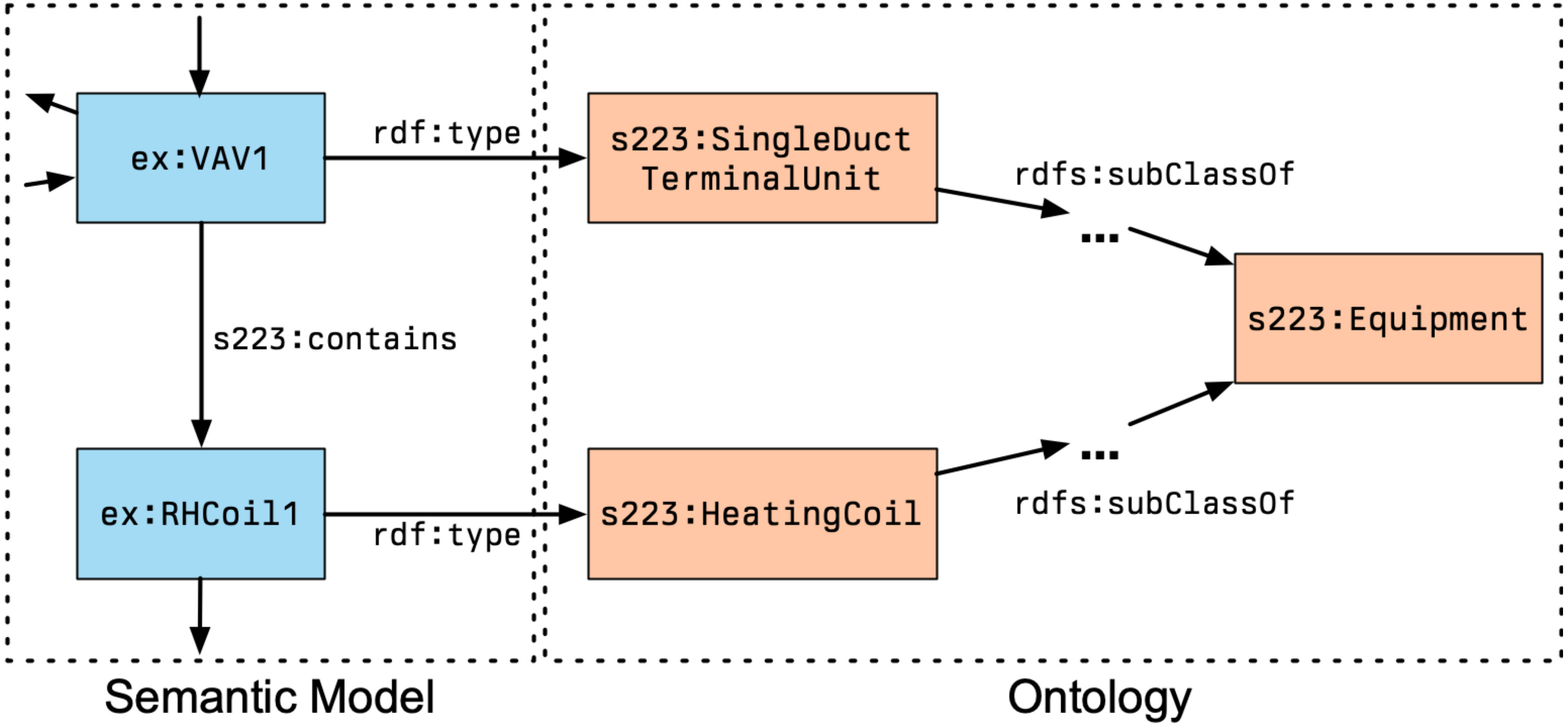


Graphical representation of 223P model

Ontologies and Semantic Models

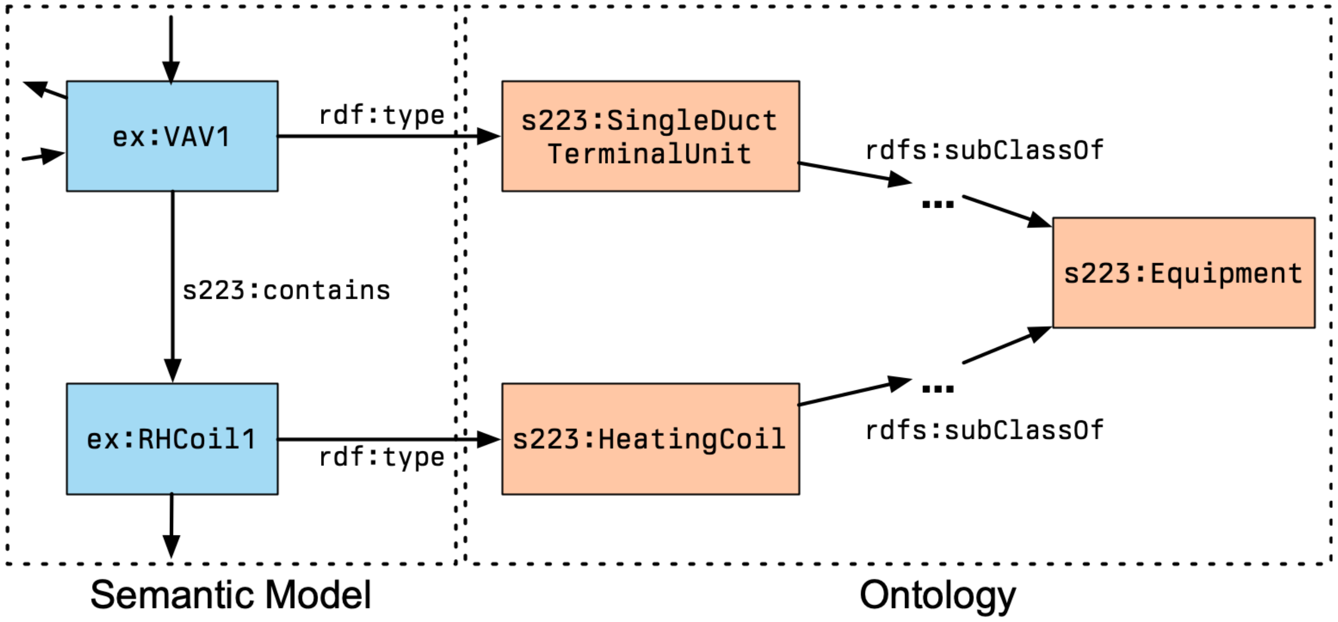
- ASHRAE 223P is an **ontology**
  - Formal definition of directed, labeled graph data structure
  - Analogous to a schema (think XML, databases, etc)
- Provides structure to semantic models, enabling
  - Automated verification/validation of semantic models
  - (Semi-)automated configuration of applications
  - (Semi-)automated creation and maintenance of semantic models
- Builds on open standards
  - **RDF (Resource Description Framework):** W3C standard for directed graphs
  - **SPARQL**: W3C standard query language for graphs
  - **SHACL**: W3C standard constraint language for graph validation

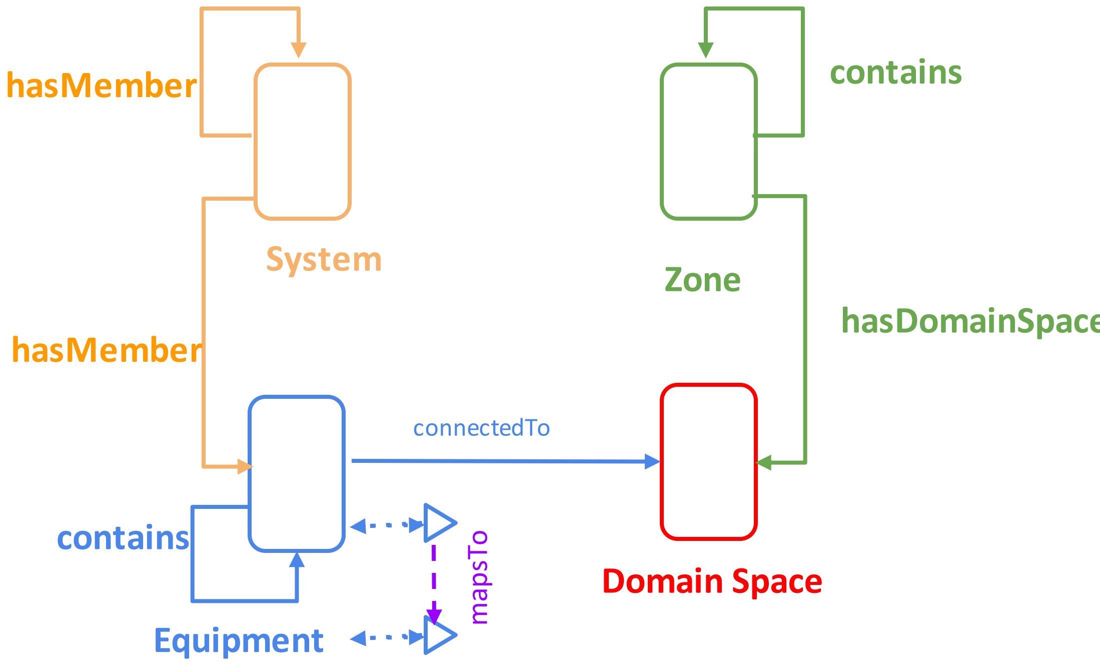
What's in a 223P Model?



- A semantic model is a graph containing labeled **nodes** and **edges** representing **entities** and their **relationships**
  - "Type" of an entity (tells applications what properties, etc to expect)
  - How entities are connected
  - Available sensors, actuators, BMS points
  - etc

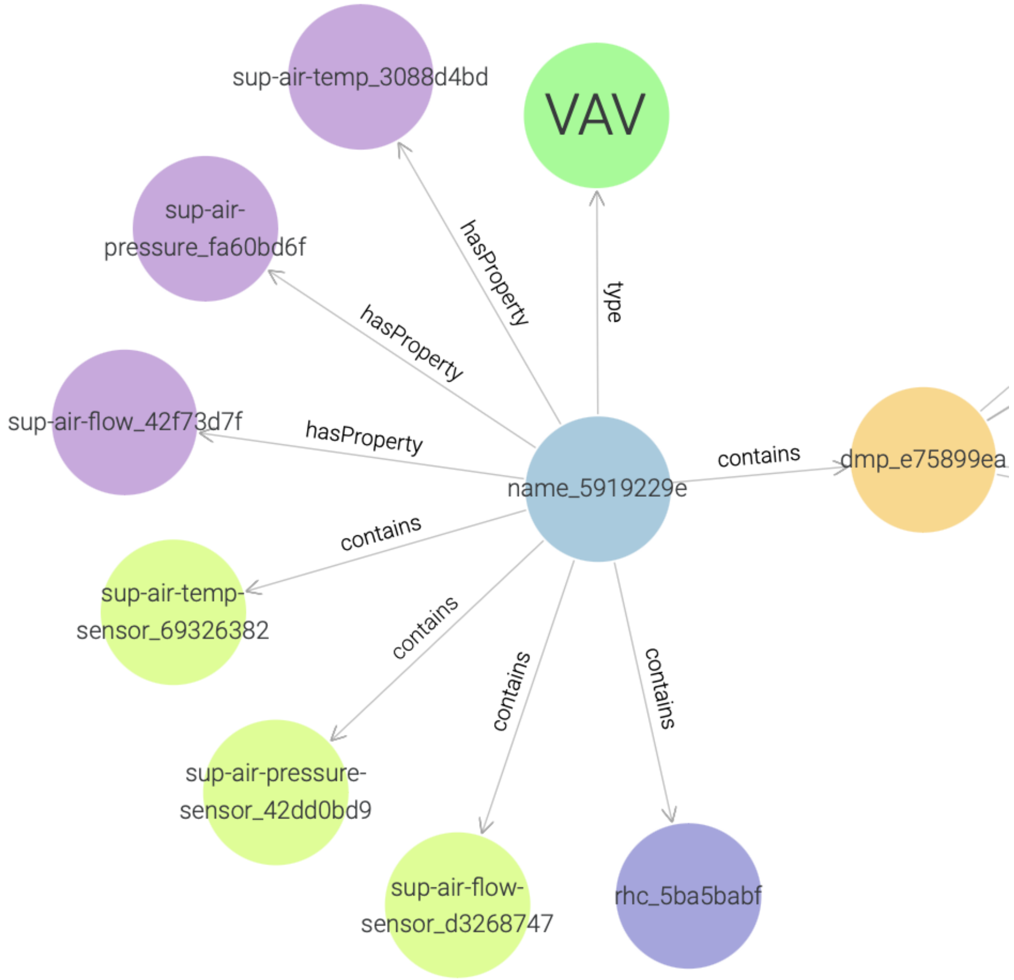
What's in the 223P Ontology?





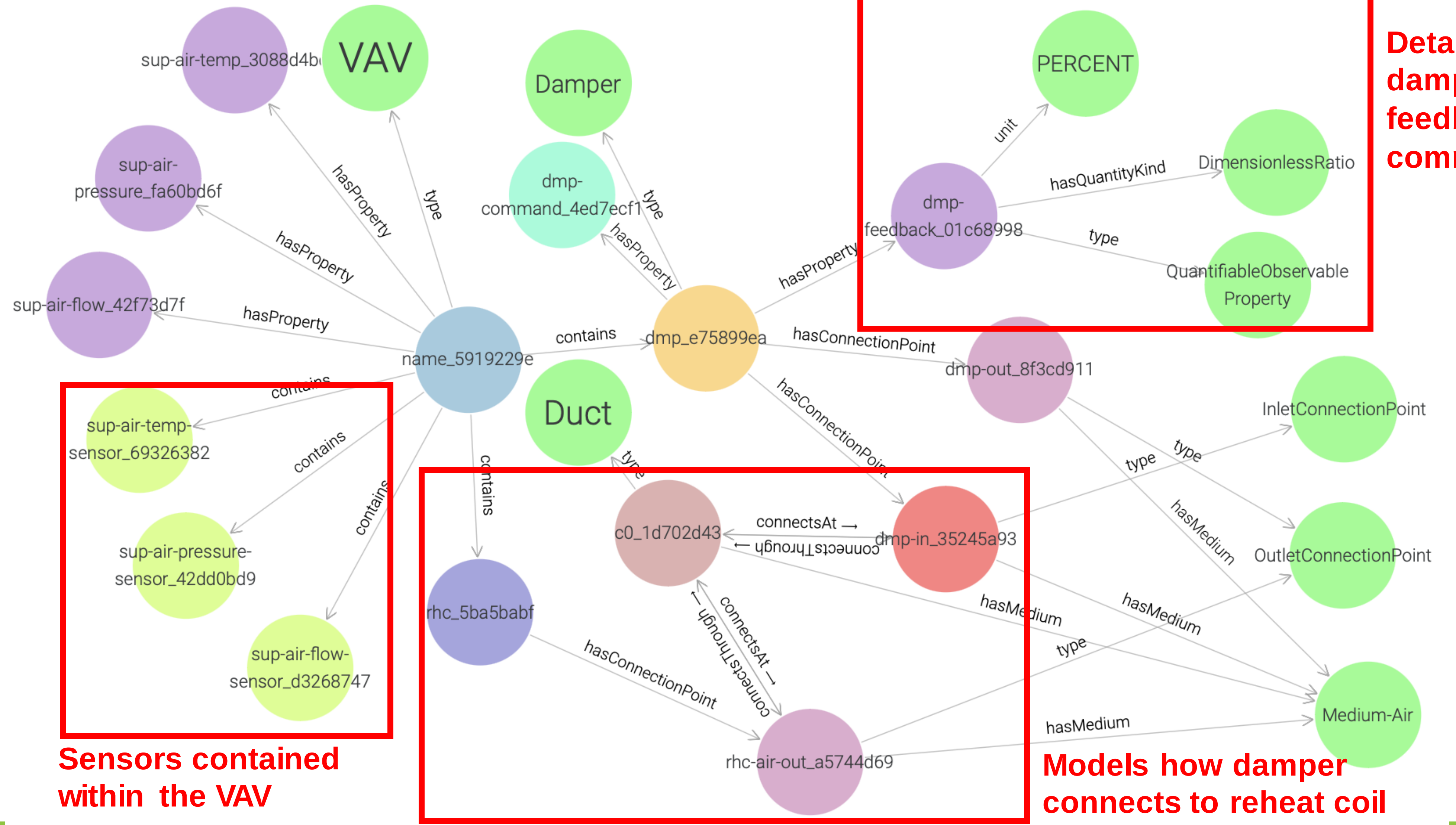
- **The 223P ontology is also a graph**
- Defines what relationships different types of entities can have
- Defines inference rules for generating new information about the model
- Includes constraints to ensure *consistency* in modeling and therefore *interoperability*

Example: VAV with Reheat



- *Name_5919229e* is the name of our RVAV
- **contains** relationship models internal equipment
  - Damper and reheat coil
  - Multiple sensors (air temp, pressure, flow)
- **hasProperty** relationship models properties which can be observed/actuated
- **type** relationship tells us which 223P class this entity is

Example: VAV with Reheat

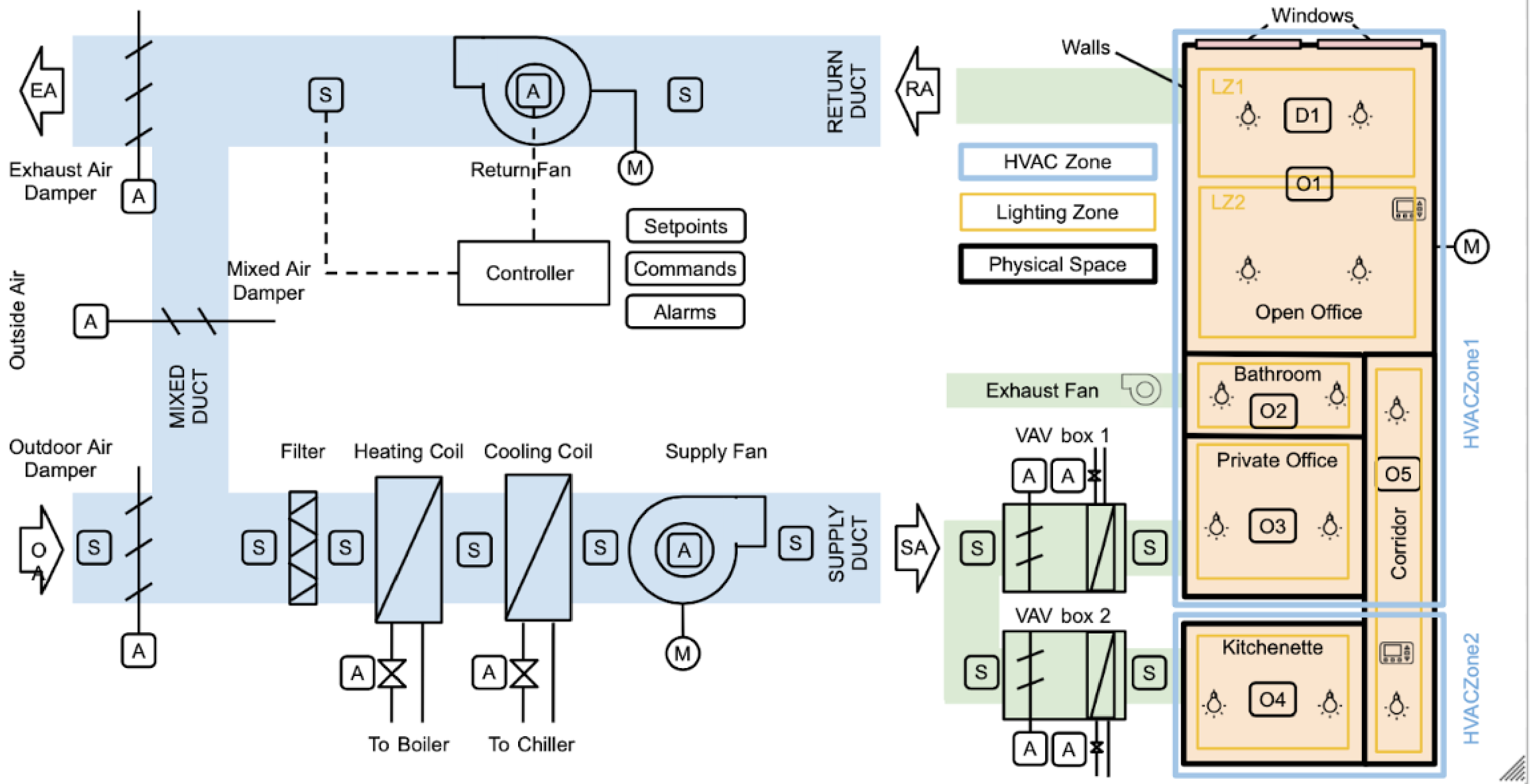


Details on damper feedback command

Sensors contained within the VAV

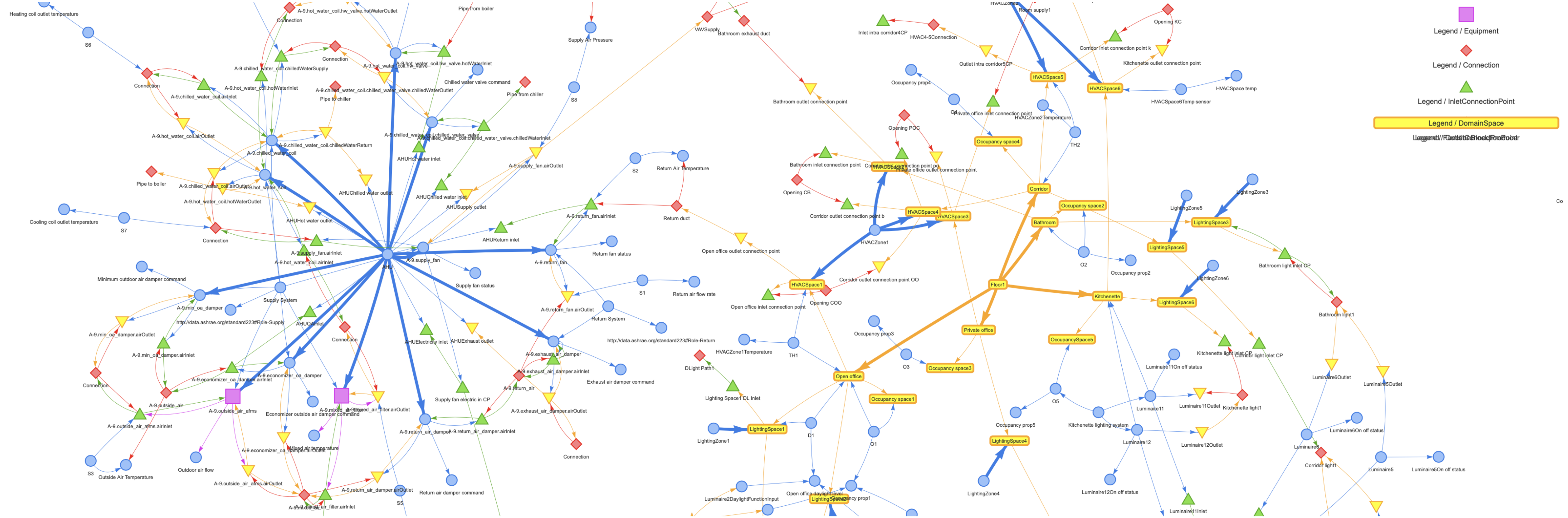
Models how damper connects to reheat coil

Example: Multi-zone AHU



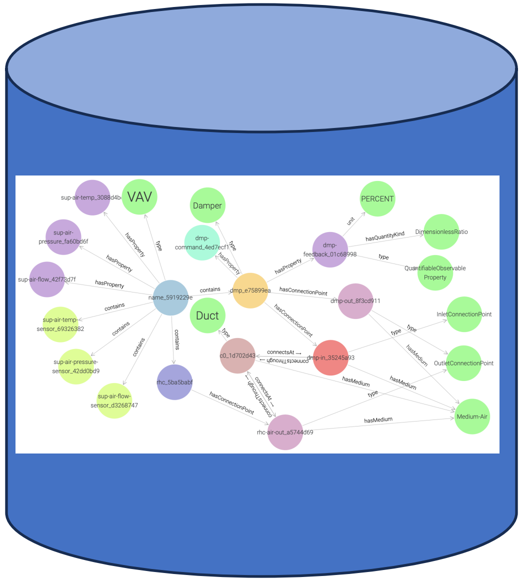
Notes: DaylightSensor1 (D1) is an input to the control for LightingZone1 (LZ1); OccSensor1 (O1) is an input to the control for LightingZone1 (LZ1) and LightingZone2 (LZ2)

Example: Multi-zone AHU



Note that this is a different visualization tool! Building on open standards like RDF makes it possible to use many different pieces of software

Asking questions about our model



Semantic model stored in graph database

- **SPARQL queries** retrieve information from semantic models
- Example: retrieving all temperature sensors and where they observe temperature

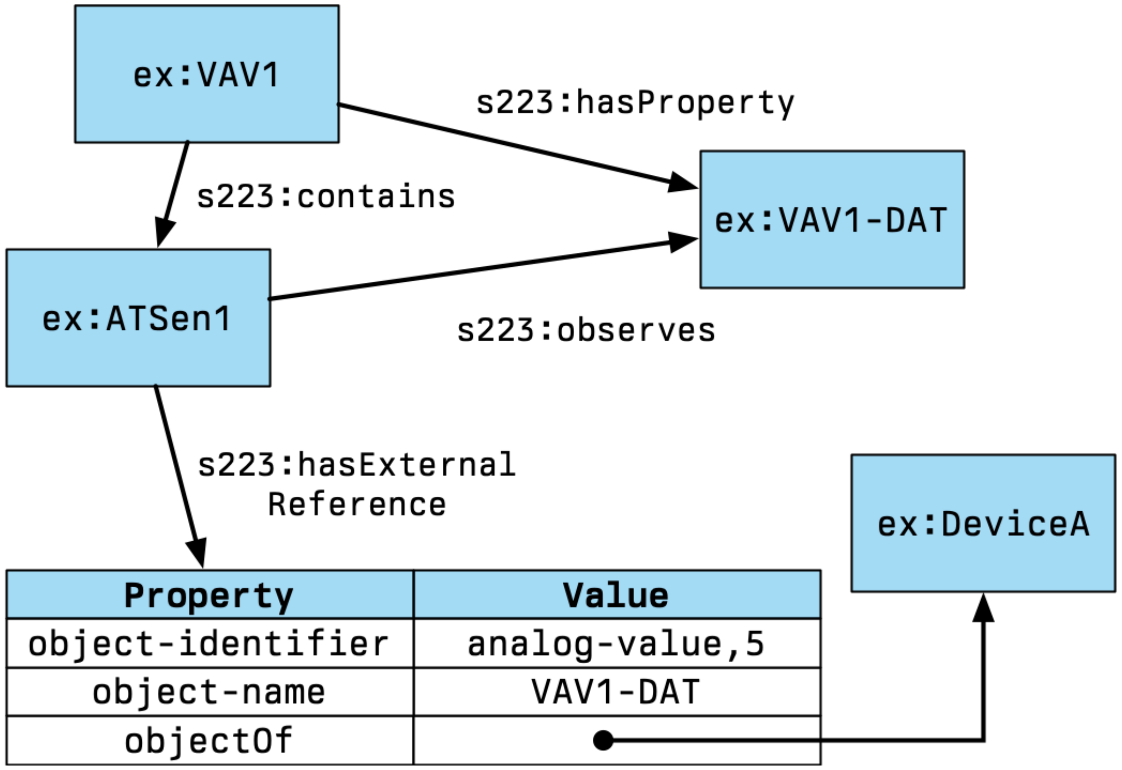
```
SELECT ?sensor ?location WHERE {
    ?sensor rdf:type/rdfs:subClassOf* s223:Sensor .
    ?sensor s223:observes ?property .
    ?property qudt:hasQuantityKind quantitykind:Temperature .
    ?sensor s223:hasObservationLocation ?location
}
```

?sensor	?location
sup-air-temp-sensor_69326382	vav_out_name_a871635f
rhc-ret-water-temp-sensor_40bc	rhc-valve-in_5060b895

From our earlier VAV Reheat example

Using queries to build applications

- *First question*: how to find the actual data?
  - Solution: **External References**
- External References are 223P entities which contain the necessary properties required to retrieve data
  - From a timeseries database…
  - From a BACnet object…
  - From a Modbus register…
  - etc



Using queries to build applications

- Example: Fault Condition from ASHRAE Guideline 36
  - Low Mixed Air Temperature detection for single zone VAVs

FC #2 (omit if no MAT sensor)	**Equation**	$MAT_{AVG} + \varepsilon_{MAT} < \min[(RAT_{AVG} - \varepsilon_{RAT}), (OAT_{AVG} - \varepsilon_{OAT})]$
	Description	MAT too low; should be between OAT and RAT
	Possible Diagnosis	RAT sensor error MAT sensor error OAT sensor error

- Use a SPARQL query to
  a) identify all locations in the model (building) where this rule can run
  b) retrieve the data necessary to run the rule
- Write the rule itself in the Python programming language

Example FDD rule application

FC #2 (omit if no MAT sensor)	**Equation**	$MAT_{AVG} + \varepsilon_{MAT} < \min[(RAT_{AVG} - \varepsilon_{RAT}), (OAT_{AVG} - \varepsilon_{OAT})]$
	Description	MAT too low; should be between OAT and RAT
	Possible Diagnosis	RAT sensor error MAT sensor error OAT sensor error

Need to find:

- Mixed air temperature
- Return air temperature
- Outside air temperature

```
PREFIX s223: <http://data.ashrae.org/standard223#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX qudt: <http://qudt.org/schema/qudt/>
PREFIX quantitykind: <http://qudt.org/vocab/quantitykind/>
PREFIX bacnet: <http://data.ashrae.org/bacnet/2020#>

SELECT ?oat ?oatId ?mat ?matId ?rat ?ratId ?inst WHERE {
    ?ahu rdf:type s223:AirHandlingUnit .
    ?bacnet a bacnet:BACnetDevice ;
        bacnet:device-instance ?inst .
    # Outside Air Temperature Sensor
    ?oat rdf:type s223:Sensor ;
         s223:observes ?outsideAir .
    ?outsideAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Outside ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?oatId .

    # Mixed Air Temperature Sensor
    ?mat rdf:type s223:Sensor ;
         s223:observes ?mixedAir .
    ?mixedAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Mixed ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?matId .

    # Return Air Temperature Sensor
    ?rat rdf:type s223:Sensor ;
         s223:observes ?returnAir .
    ?returnAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Return ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?ratId .
}
```


```
SELECT ?oat ?oatId ?mat ?matId ?rat ?ratId ?inst WHERE {
    ?ahu rdf:type s223:AirHandlingUnit .
    ?bacnet a bacnet:BACnetDevice ;
        bacnet:device-instance ?inst .
    # Outside Air Temperature Sensor
    ?oat rdf:type s223:Sensor ;
         s223:observes ?outsideAir .
    ?outsideAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Outside ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?oatId .
```


SPARQL query retrieves names and BACnet object IDs for all sensors

Example FDD rule application

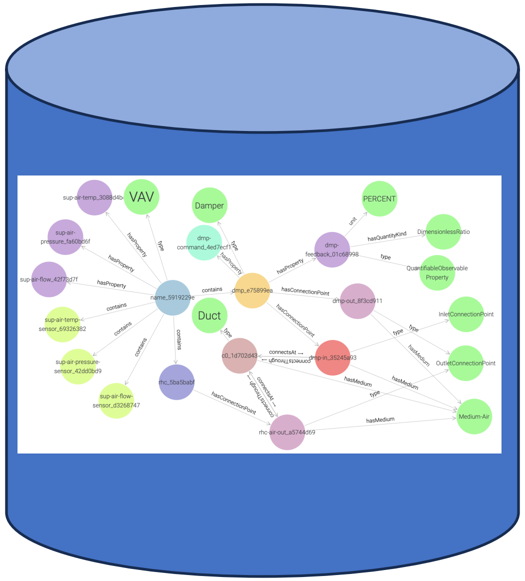
```
PREFIX s223: <http://data.ashrae.org/standard223#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX qudt: <http://qudt.org/schema/qudt/>
PREFIX quantitykind: <http://qudt.org/vocab/quantitykind/>
PREFIX bacnet: <http://data.ashrae.org/bacnet/2020#>

SELECT ?oat ?oatId ?mat ?matId ?rat ?ratId ?inst WHERE {
    ?ahu rdf:type s223:AirHandlingUnit .
    ?bacnet a bacnet:BACnetDevice ;
        bacnet:device-instance ?inst .
    # Outside Air Temperature Sensor
    ?oat rdf:type s223:Sensor ;
         s223:observes ?outsideAir .
    ?outsideAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Outside ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?oatId .

    # Mixed Air Temperature Sensor
    ?mat rdf:type s223:Sensor ;
         s223:observes ?mixedAir .
    ?mixedAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Mixed ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?matId .

    # Return Air Temperature Sensor
    ?rat rdf:type s223:Sensor ;
         s223:observes ?returnAir .
    ?returnAir rdf:type s223:QuantifiableObservableProperty ;
        s223:hasAspect s223:Role-Return ;
        qudt:hasQuantityKind quantitykind:Temperature ;
        s223:hasExternalReference/bacnet:object-identifier ?ratId .
}
```

Execute query against semantic model



ahu	inst	matId
urn:ex/single-zone-ahu	123	analog-value,6

oatId	ratId
analog-value,5	analog-value,7

Query Results

- Now we have all the information necessary to read live data from our BACnet network!
- External references also let us read data out of databases, etc

Example FDD rule application

FC #2 (omit if no MAT sensor)	**Equation**	$MAT_{AVG} + \varepsilon_{MAT} < \min[(RAT_{AVG} - \varepsilon_{RAT}), (OAT_{AVG} - \varepsilon_{OAT})]$
	Description	MAT too low; should be between OAT and RAT
	Possible Diagnosis	RAT sensor error MAT sensor error OAT sensor error

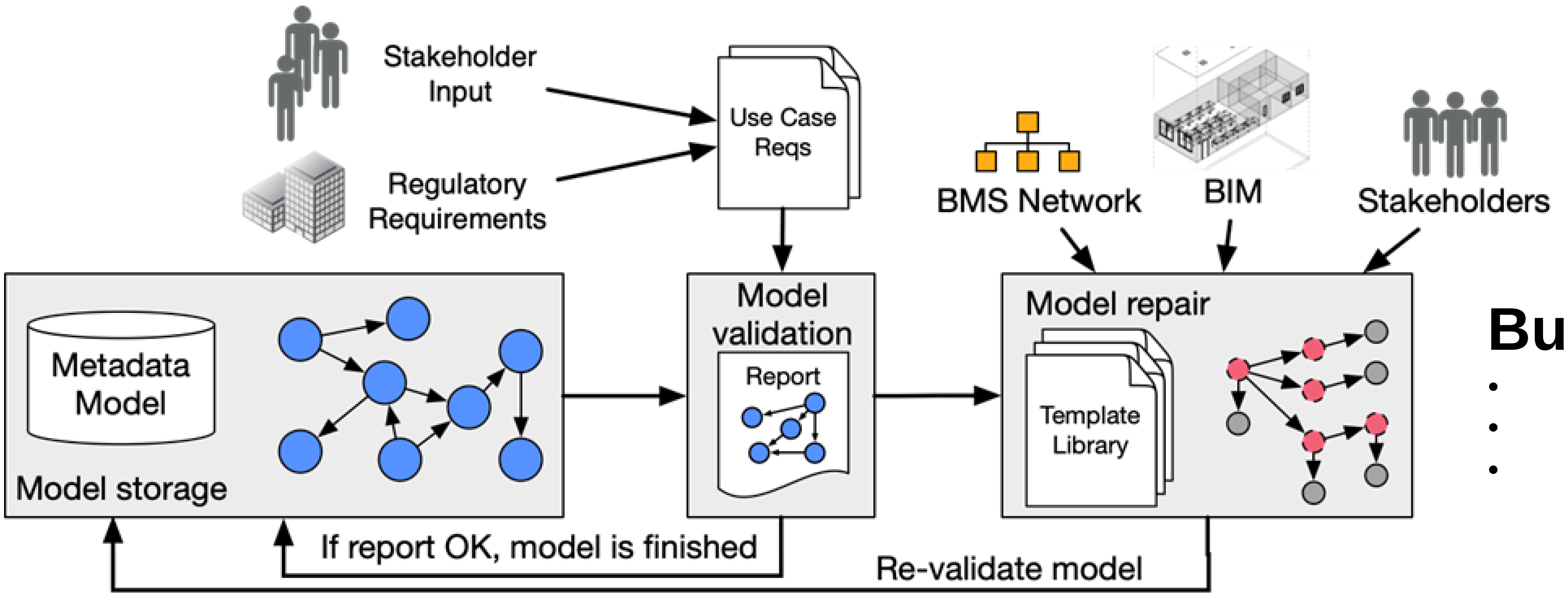
- Write out FDD rule as a Python function
- Use query results to generate a dataset with the correct column names
- Run the function!

```
def run_fc2(df):
    """
    Check MAT + ε_MAT < min[(RAT − ε_RAT), (OAT − ε_OAT)] at each timestamp and
    print out the timestamps where the inequality is true.
    """
    # Assuming ε values as constants, they can be changed as per actual values
    epsilon_MAT = epsilon_RAT = epsilon_OAT = 1
    # List to store timestamps where the inequality holds true
    timestamps_where_true = []
    # Iterate over the dataframe
    for index, row in df.iterrows():
        # Check the inequality condition for each row
        if row['mat'] + epsilon_MAT < min(row['rat'] - epsilon_RAT, row['oat'] - epsilon_OAT):
            timestamps_where_true.append(index)
    # Print out the timestamps
    for timestamp in timestamps_where_true:
        print(f"Fault condition true at: {timestamp}")
```

[19]:
```
run_fc2(df)
```

```
Fault condition true at: 2023-01-01 06:30:00
Fault condition true at: 2023-01-01 08:30:00
Fault condition true at: 2023-01-01 09:45:00
```

Open-Source Software for 223P Semantic Models



BuildingMOTIF

- Open source, BSD-licensed
- Developed by NREL
- Available on GitHub

- 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
- Generate SPARQL queries from application requirements

Conclusion

- ASHRAE 223P models buildings, their assets, and data sources as a directed graph
- Applications, such as FDD suites and control sequences, can query 223P models for useful configuration information
- ASHRAE 223P provides a standardized framework for data interoperability, allowing the software applications to exchange information seamlessly.

Questions?

Gabe Fierro

gtfierro@mines.edu