2024 ASHRAE WINTER CONFERENCE

CHICAGO, JAN 20-24 | AHR EXPO, JAN 22-24

Seminar 14: Better Data and Analytics Enable Better Decision Making on Building Operations and Decarbonization

Metadata Ontologies for Data Discovery and Portable Analytics

Dr. Gabe Fierro

Colorado School of Mines

National Renewable Energy Laboratory

gtfierro@mines.edu

COLORADO SCHOOL OF MINES

Learning Objectives

- Understand FAIR principles of data: Findability, Accessibility, Interoperability, and Reusability
- Learn where to find open building performance datasets
- Understand how dataset and analytics can provide insights to improve building operations
- Learn various data tools (ontology, schema, models) used to represent semantic and metadata of building and system data

Acknowledgements

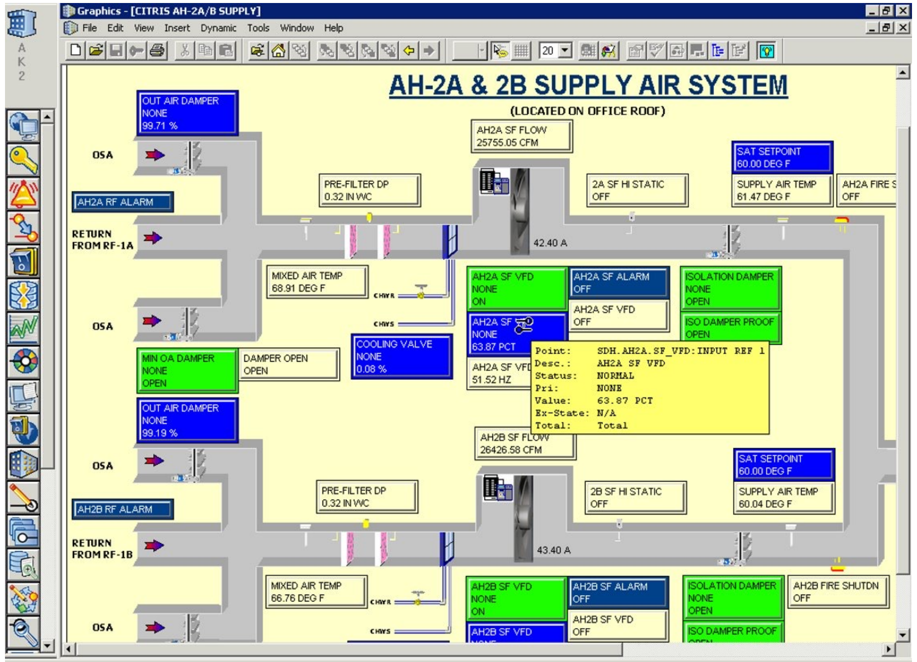
- Avijit Saha, NREL
- Matthew Steen, NREL
- Tobias Shapinsky, NREL
- Steve Bushby, NIST
- Joel Bender, Cornell University
- Brian Walker, DOE
- Amir Roth, DOE

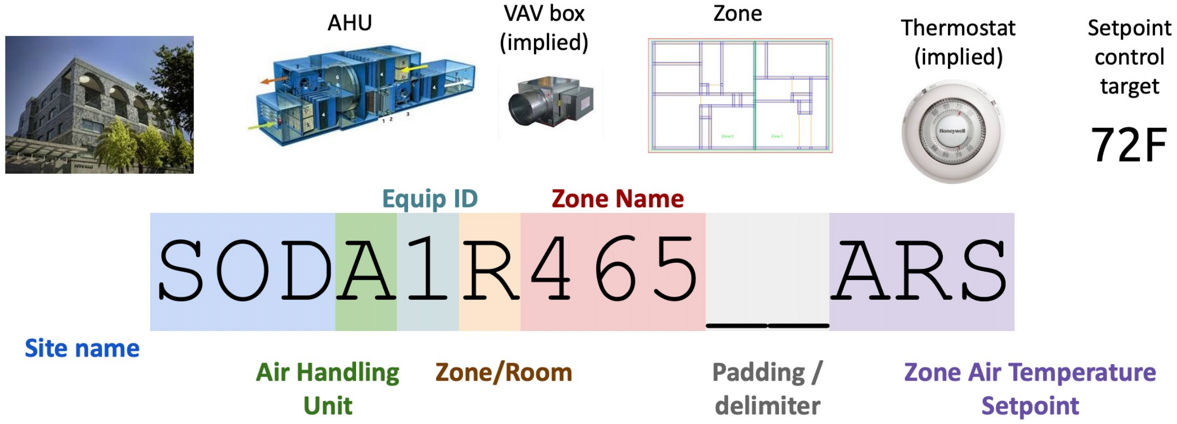
Outline/Agenda

- Overview of 223P and Brick ontologies
- Modeling buildings and building data with ontologies
- Accessing data through ontologies
- Using data to build fault detection and thermal models

Framing the Problem

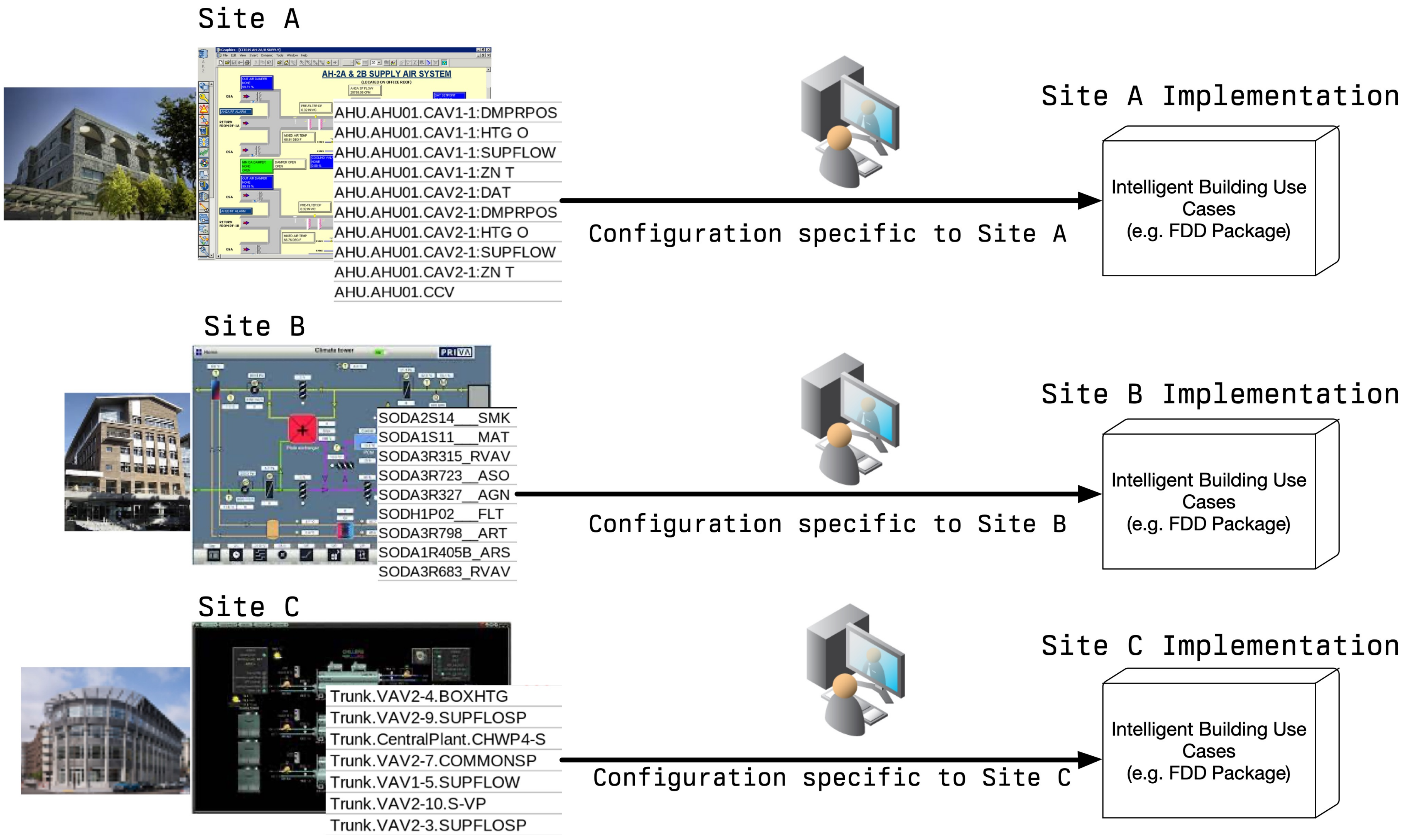
- *Data rich, but information poor*
- Building data often hidden away behind proprietary silos, semi-structured naming conventions





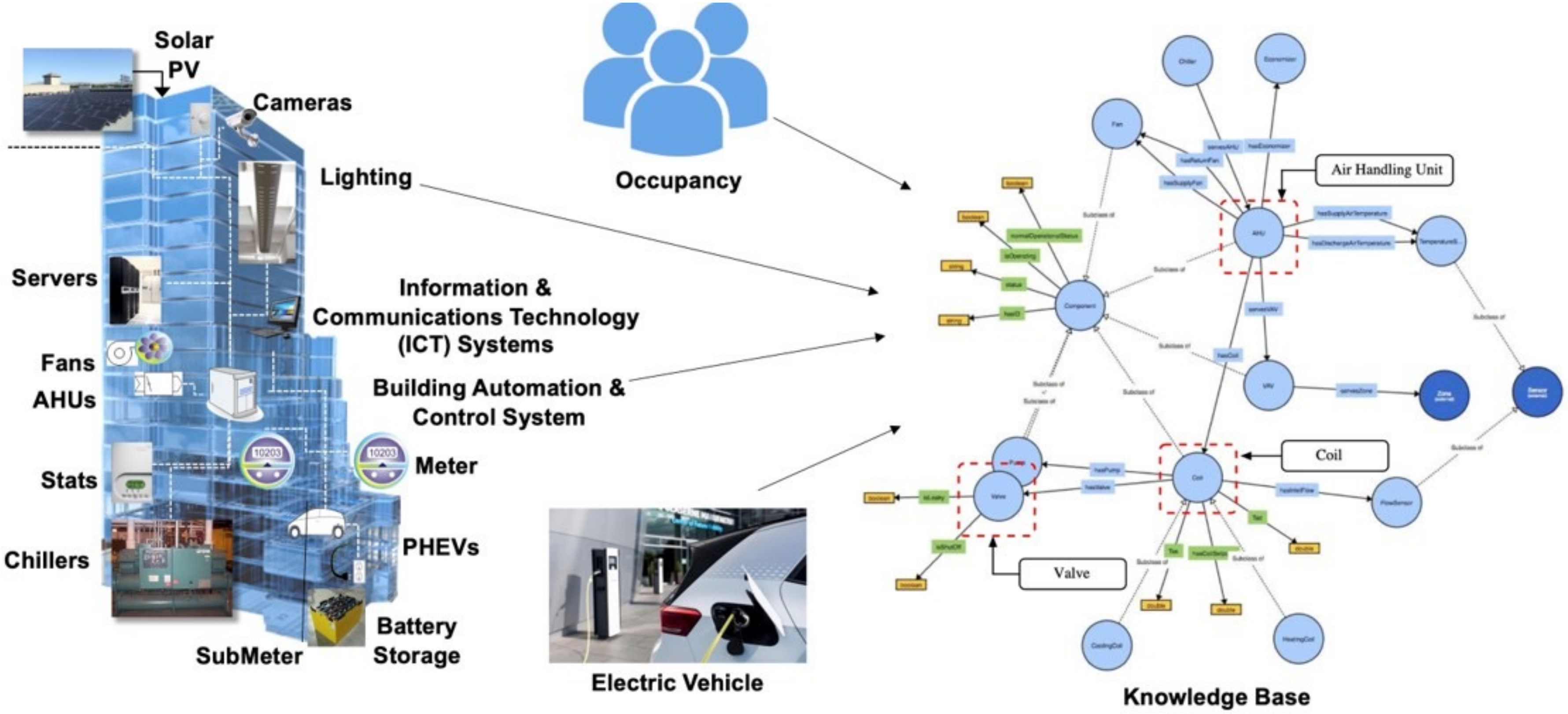
- Focus on operational data in this talk
- BMS point labels (above) are the primary identifiers of data

Implementing Software Requires Manual Configuration



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.

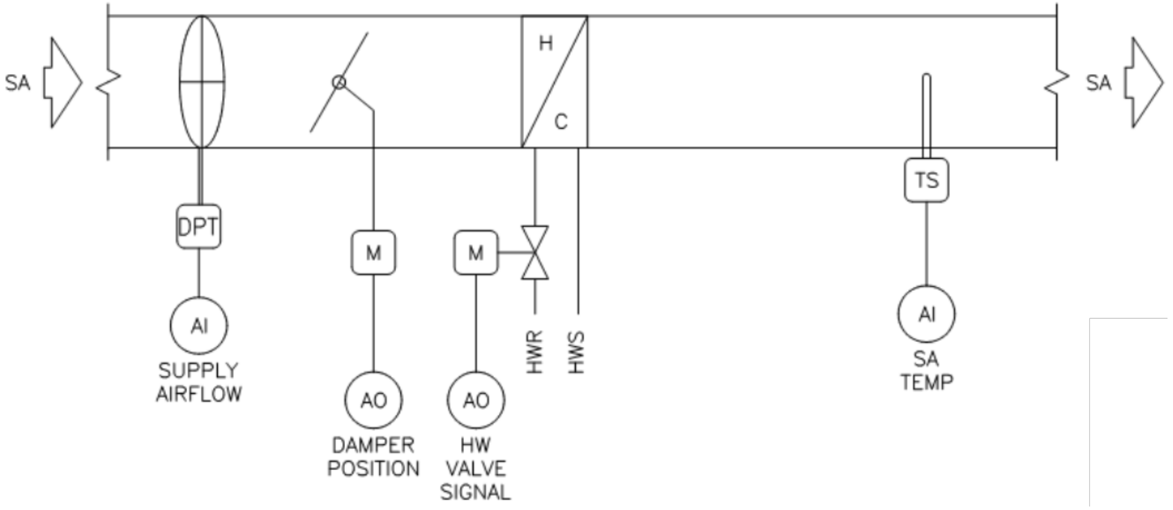


- Control
- Energy Auditing
- Fault Detection and Diagnostics
- Commissioning
- Smart Grid Interactions
- **ASHRAE 231P**

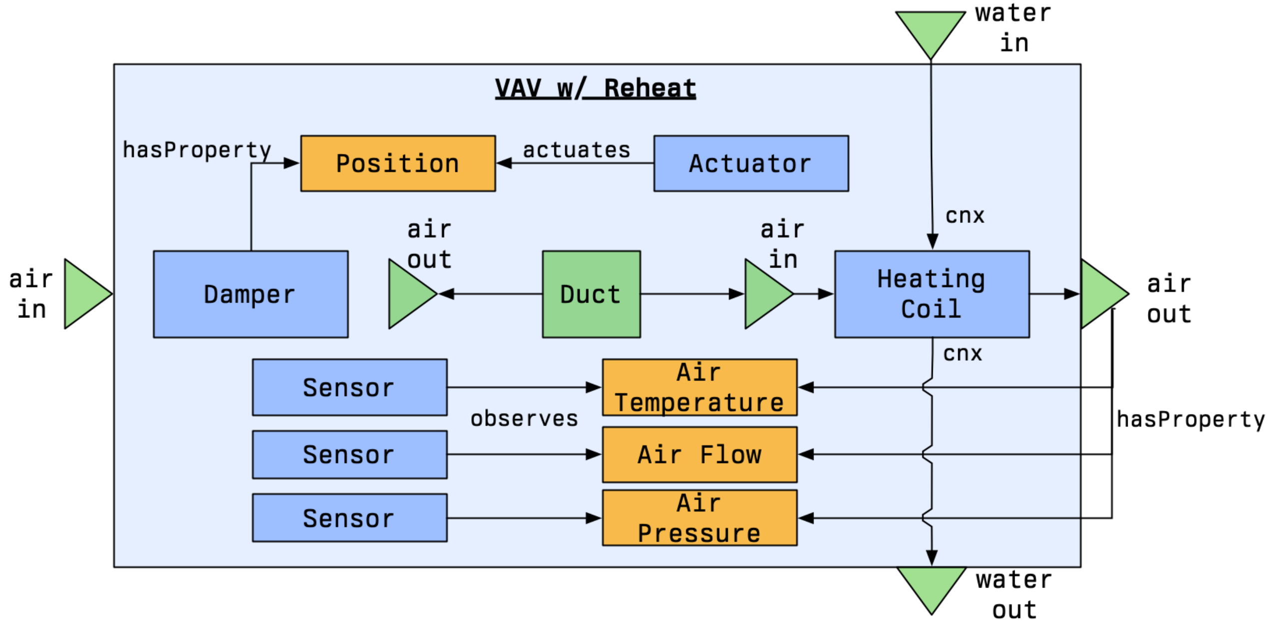
Physical, logical systems ← represents ← **Semantic Model** ← access ← **Applications**

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

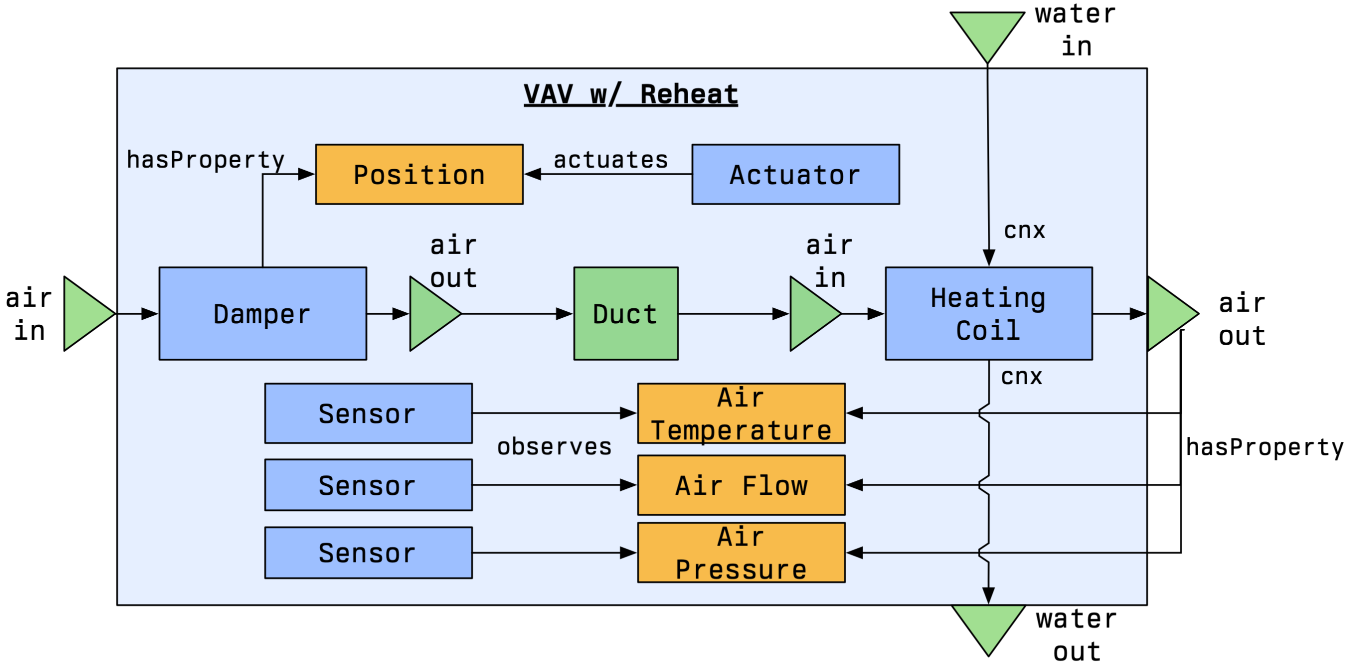


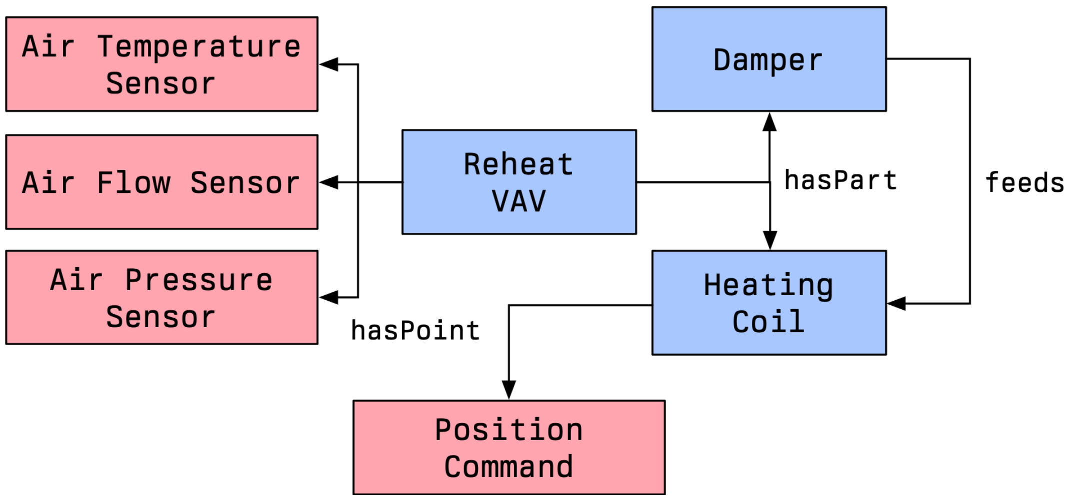
VAV with Reheat Mechanical Diagram



Graphical representation of 223P model

Brick is a Simplification of 223P



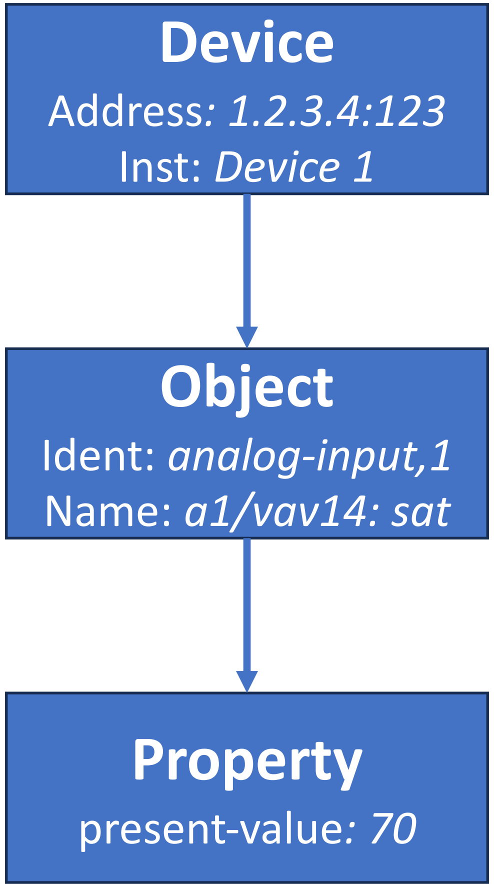


- Detailed topological information in 223P is helpful for some apps, but unnecessary for many others
- Use familiar terminology found in point lists, application descriptions
- Smaller models, easier to query → works better for common systems
- Can always reach into 223P for concepts not covered by Brick

Ontologies and Semantic Models

- ASHRAE 223P and Brick are **ontologies**
  - 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

How do Ontologies Work with Building Data?



BACnet

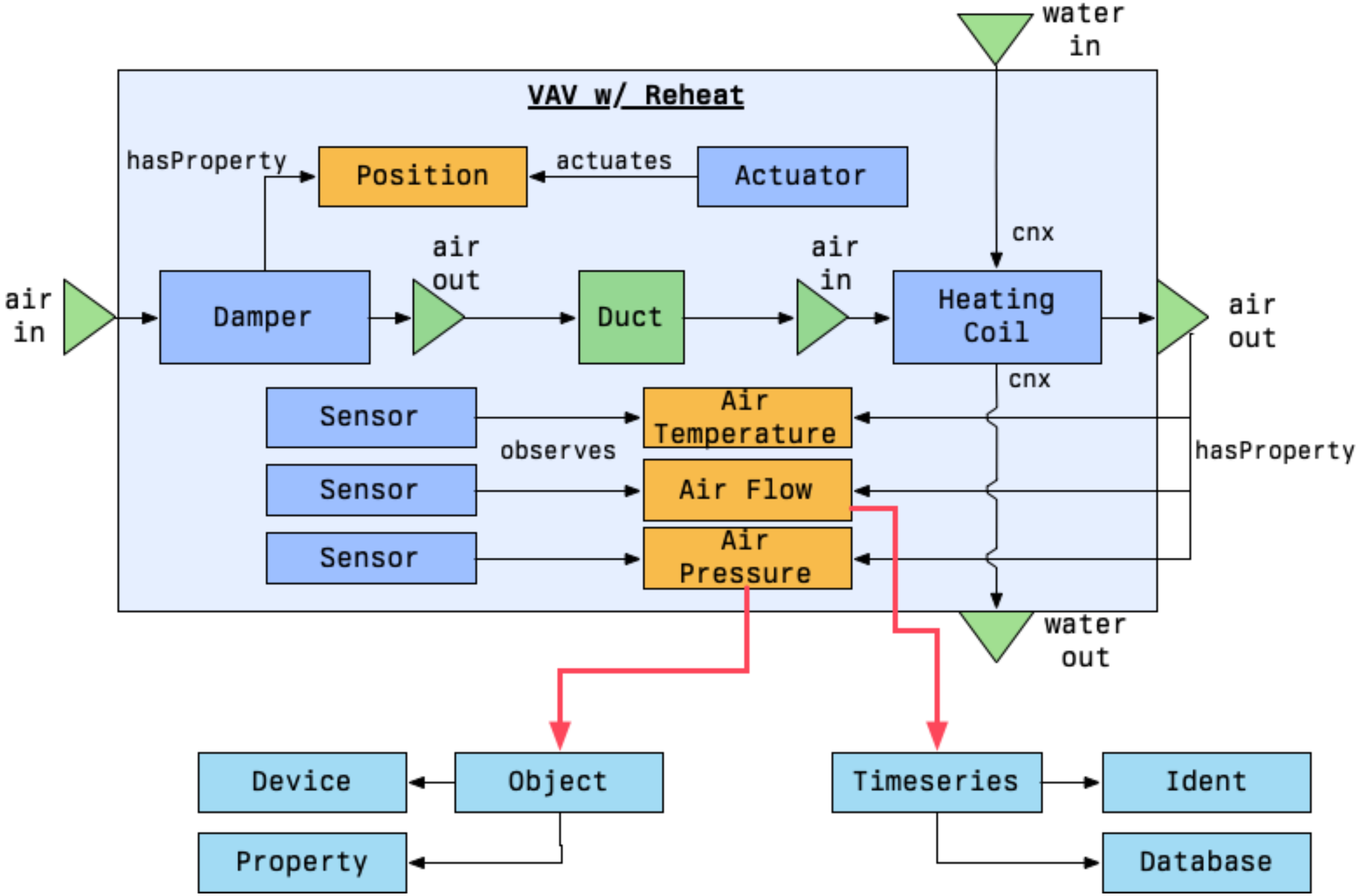
Ident	Time	Value
a1/vav14: sat	2024-06-21T10:15:00	70
a1/vav14: sat	2024-06-21T10:30:00	70
a1/vav14: sat	2024-06-21T10:45:00	70.5
a1/vav14: sat	2024-06-21T11:00:00	70.1

Timeseries DB

- Consider two sources of data
- **Live data** through interacting with BACnet objects
- **Historical data** through an archival service
- Nothing fundamental here:
  - Easily support MQTT, BACnet-SC, Modbus, …

How do Ontologies Work with Building Data?

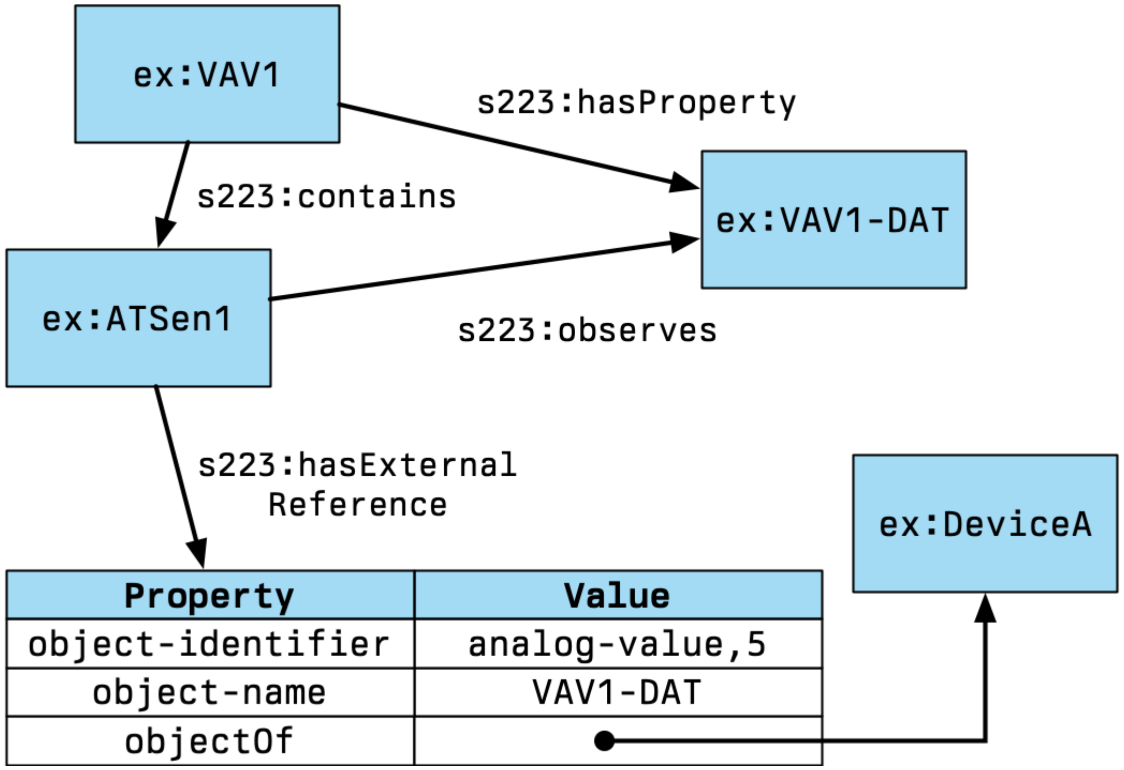
- Explicit representation of data context
  - What kind of point, who hosts it does it relate to equipment, how those equipment connect, etc
- Move away from naming conventions and "ctl-f"
- Use queries to describe and discover data by "meaning" and "context", not by name
  - "Early" vs "late" binding
  - We use google.com, not an IP address



Linking to data

- 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

- Query the external reference for a property to find the data



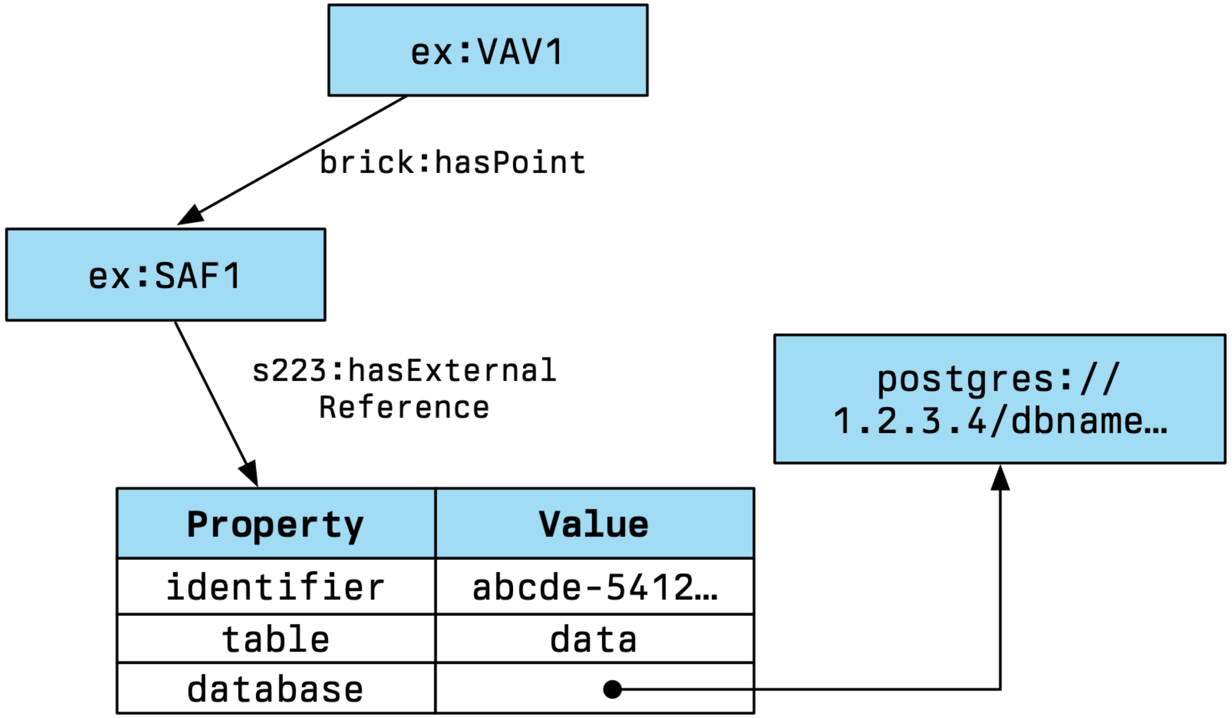
BACnet example

Linking to data (2)

- External References are *also* supported in the Brick ontology

- Query the external reference for the foreign key, table, and database URI for historical data

- The “secret” is just including the necessary client parameters required to connect



BACnet example

Queries to find data

- Same pattern for Brick/223P
- Use the right level of detail for the job:
    - "supply air flow sensor on a vav" → Brick
    - "all temperature sensors on the condenser water loop" → 223P or Brick
    - "all temperature sensors on the condenser water loop, and their measurement locations" → 223P

Types of points (quantity kind, substance, input/output)

External reference

Association of point to assets, equipment, connections

Context or identity of those assets, equipment, connections

Using Queries to Find Data

Types of points (quantity kind, substance, input/output)

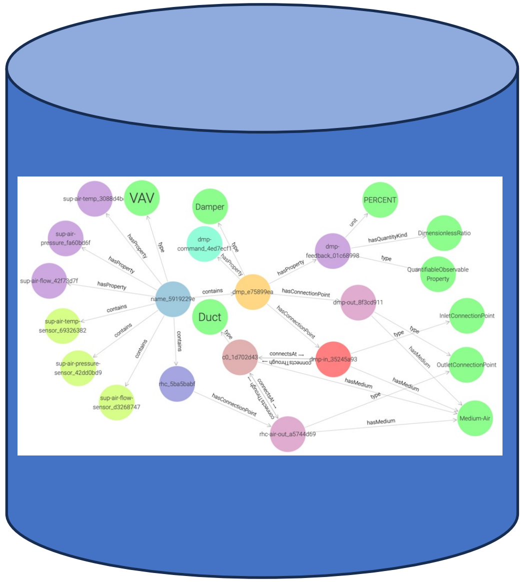
External reference

Association of point to assets, equipment, connections

Context or identity of those assets, equipment, connections

```
SELECT * WHERE {

   ?sensor rdf:type/rdfs:subClassOf* s223:Sensor ;
           s223:observes ?property .
   ?property qudt:hasQuantityKind qk:Temperature ;

           s223:hasExternalReference ?ref .
   ?ref bacnet:isObjectOf ?device ;
        bacnet:object-identifier ?ident .

   ?device rdf:type s223:TerminalUnit ;
           s223:contains ?sensor .

   ?device s223:connectsFrom building:AHU1 .

}
```

Using Queries to Find Data



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

Building Applications with Data

- 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

<u>Need to find:</u>

- 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 .
}
```

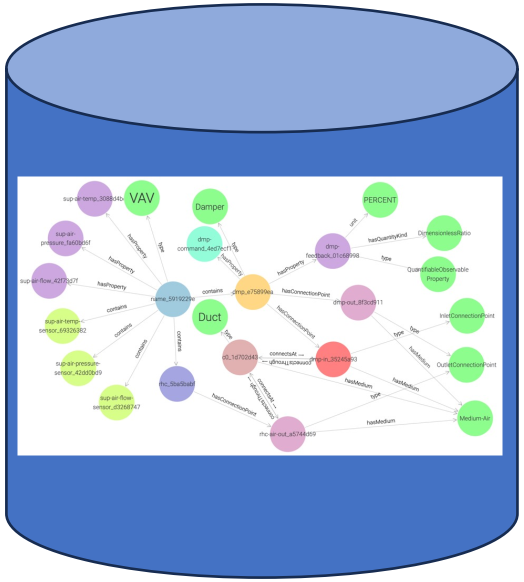

```
SELECT ?oat ?oatId ?mat ?matId ?rat ?ratId ?inst WHERE {
    ?ahu rdf:type s223:AirHandlingUnit .
    ?bacnet a bacnet:BACnetDevice ;
        bacnet:device-instance ?inst .
    # Outside Air Temperature Sensor
    ?oat rdf:type s223:Sensor ;
         s223:observes ?outsideAir .
    ?outsideAir rdf:type 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

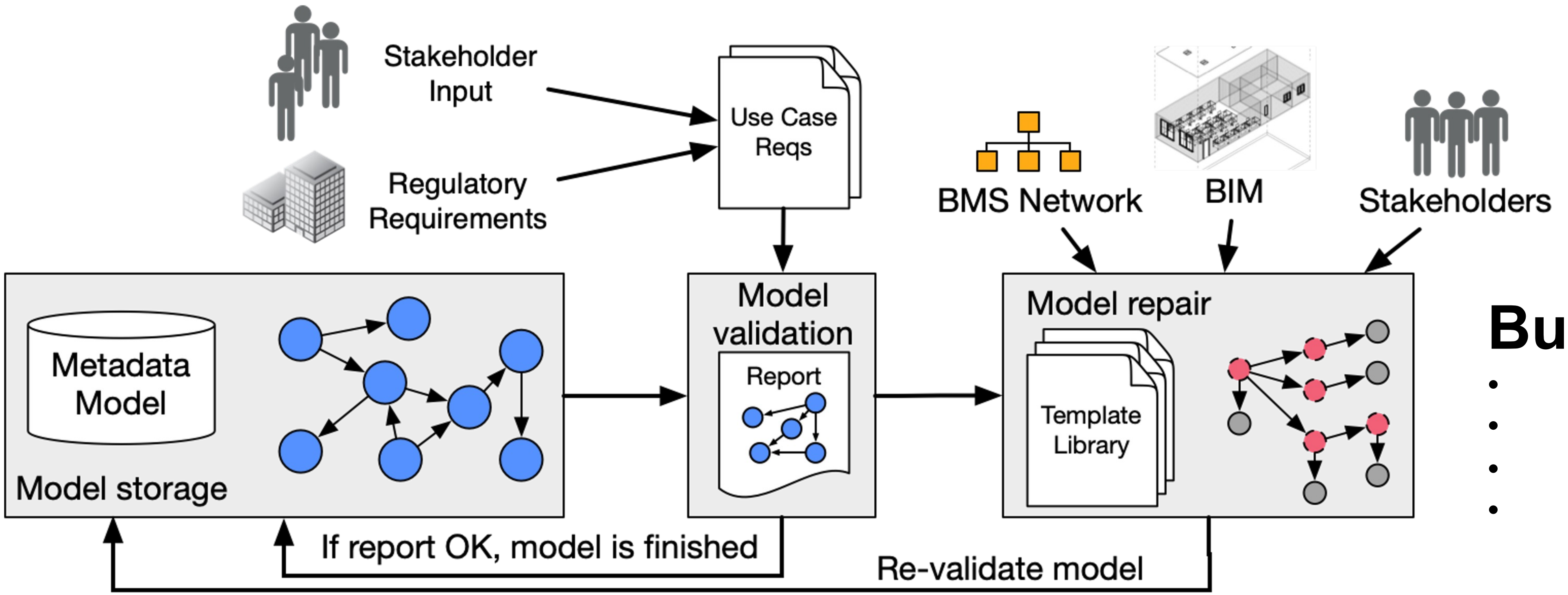
- Write out FDD rule as a Python function
- Use query results to generate a dataset with the correct column names
- Run the function!

```
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 Semantic Models



BuildingMOTIF

- Open source, BSD-licensed
- Developed by NREL
- Available on GitHub
- *"Software 1"*

- 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

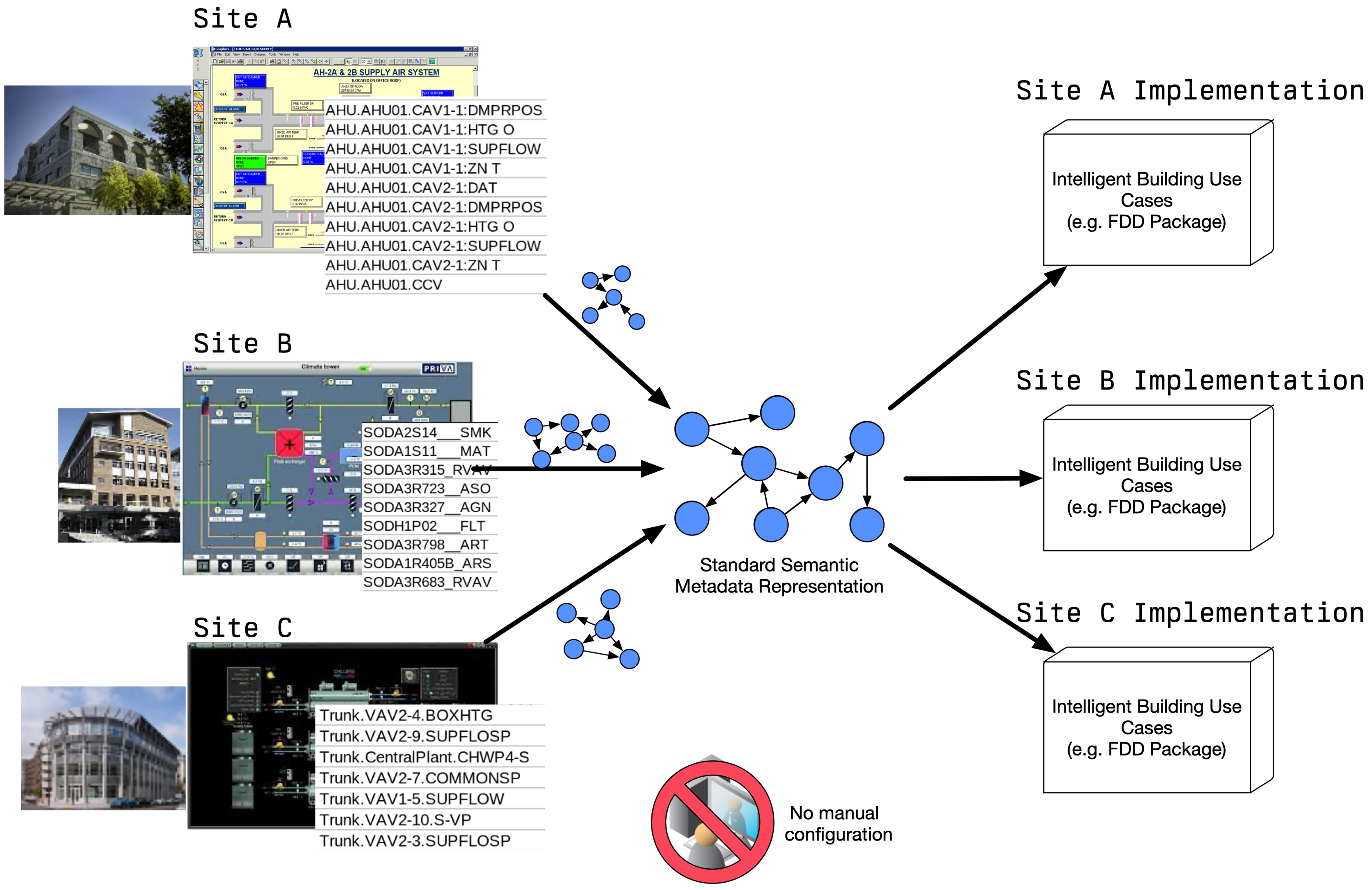
Deriving Queries from "Shapes"

```
:zone-temp-model a sh:NodeShape ;
    sh:targetClass brick:RVAV ;
    sh:property [
        sh:path brick:hasPoint ;
        sh:name "hvac_mode" ;
        sh:qualifiedMinCount 1 ;
        sh:qualifiedValueShape [ sh:class brick:HVAC_Mode_Command ] ;
    ] ;
    sh:property [
        sh:path brick:hasPoint ;
        sh:name "supply_air" ;
        sh:qualifiedMinCount 1 ;
        sh:qualifiedValueShape [ sh:class brick:Supply_Air_Temperature_Sensor ] ;
    ] ;
    sh:property [
        sh:path brick:feeds
        sh:qualifiedMinCount 1 ;
        sh:name "room" ;
        sh:qualifiedValueShape [
            sh:class brick:Room ;
            sh:property [
                sh:path brick:hasPoint ;
                sh:qualifiedMinCount 1 ;
                sh:name "room_temp" ;
                sh:qualifiedValueShape [ sh:class brick:Air_Temperature_Sensor ] ;
            ] ;
        ] ;
    ] ;
    sh:property [
        sh:path brick:hasPart ;
        sh:qualifiedMinCount 1 ;
```

```
SELECT * WHERE {
  ?target     rdf:type/rdfs:subClassOf* brick:RVAV ;
        brick:hasPoint          ?supply_air ;
        brick:feeds          ?room ;
        brick:hasPart          ?coil .
  ?room     rdf:type/rdfs:subClassOf* brick:Room ;
        brick:hasPoint          ?room_temp .
  ?coil     rdf:type/rdfs:subClassOf* brick:Heating_Coil ;
        brick:hasPoint          ?heating_valve .
  ?supply_air     a brick:Supply_Air_Temperature_Sensor .
  ?room_temp     a brick:Air_Temperature_Sensor .
  ?heating_valve     a brick:Position_Command .
}
```

- Shape is a specification of what information **must** be contained within a model
  - Check for required points, equipment, annotations
  - *Software 1* evaluates requirements on a model and suggests fixes
- Think of a shape as a **schema** for a desired dataset
- *Software 1* can convert shapes (validation) into queries (retrieval)

Portable Analytics



Conclusion

- Semantic models like Brick and 223P can provide rich contextual annotations to building data sets
- Queries retrieve data sources from models using properties and characteristics, *not identifiers*
- Semantic models can link to external data sources like BACnet objects and timeseries historian services
- All of this is built on open standards; open-source software provides simpler and easier interfaces

Questions?

Gabe Fierro

gtfierro@mines.edu