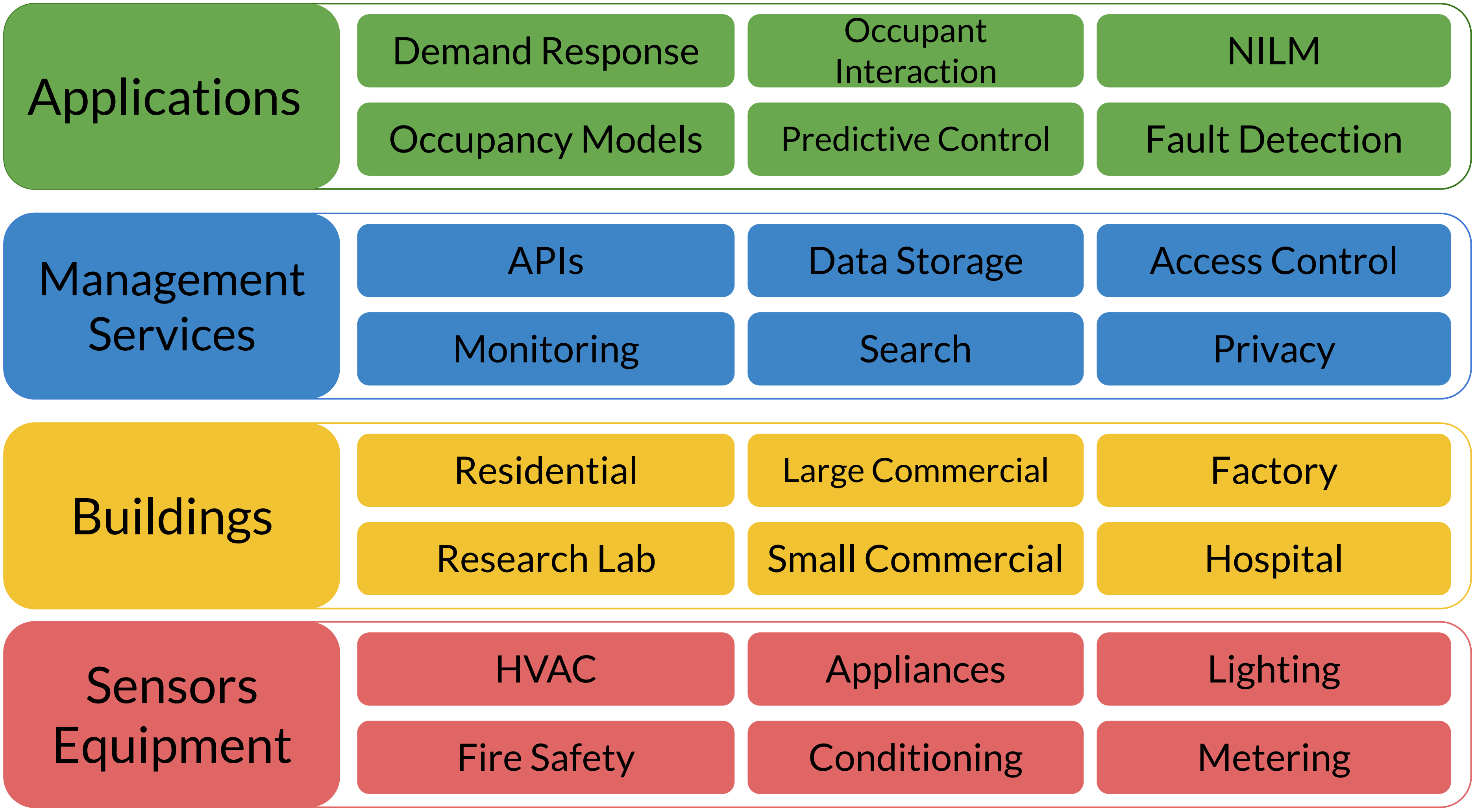
Why **Brick** is a Game Changer for Smart Buildings

Gabe Fierro
UC Berkeley

Jason Koh
UC San Diego

https://brickschema.org


Applications
Demand Response
Occupant Interaction
NILM
Occupancy Models
Predictive Control
Fault Detection
Management Services
APIs
Data Storage
Access Control
Monitoring
Search
Privacy
Buildings
Residential
Large Commercial
Factory
Research Lab
Small Commercial
Hospital
Sensors Equipment
HVAC
Appliances
Lighting
Fire Safety
Conditioning
Metering

Setting the Scene

- The built environment is characterized by extreme heterogeneity
  - Every building site is a “one-off”
  - Multitude of equipment vendors: different capabilities, features
  - BMS, SCADA systems custom to the deployment site
  - Custom-designed controls, architecture, use cases, etc
  - All of this changes over time

Buildings	Residential	Large Commercial	Factory
	Research Lab	Small Commercial	Hospital
Sensors Equipment	HVAC	Appliances	Lighting
	Fire Safety	Conditioning	Metering

Setting the Scene

- Fragmentation, heterogeneity require **effective interoperability standards**

- (2004) NIST Capital Facilities Interoperability Study:
    - *"Cost of inadequate interoperability in the U.S. capital facilities industry [is estimated at] $15.8 billion per year."*

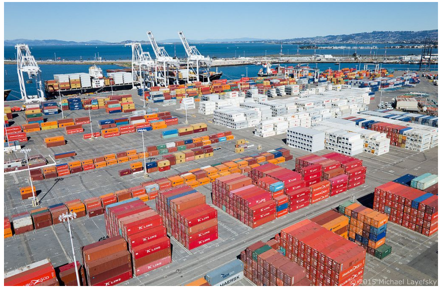
- (2017) Evaluation of U.S. Building Energy Benchmarking and Transparency Programs: Attributes, Impacts, and Best Practices:
    - *Limited deployment of energy efficiency applications constrains the ability to evaluate potential savings*

Benefits of Abstraction

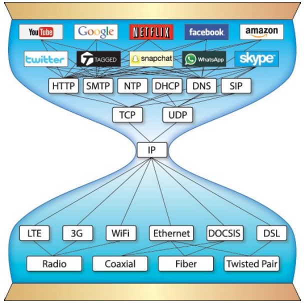
- Abstraction is key to interoperability

- Remove “irrelevant” details to focus on properties/attributes relevant to a task

- Abstraction facilitates scale



Shipping Containers



The Internet

Peripherals

Operating Systems

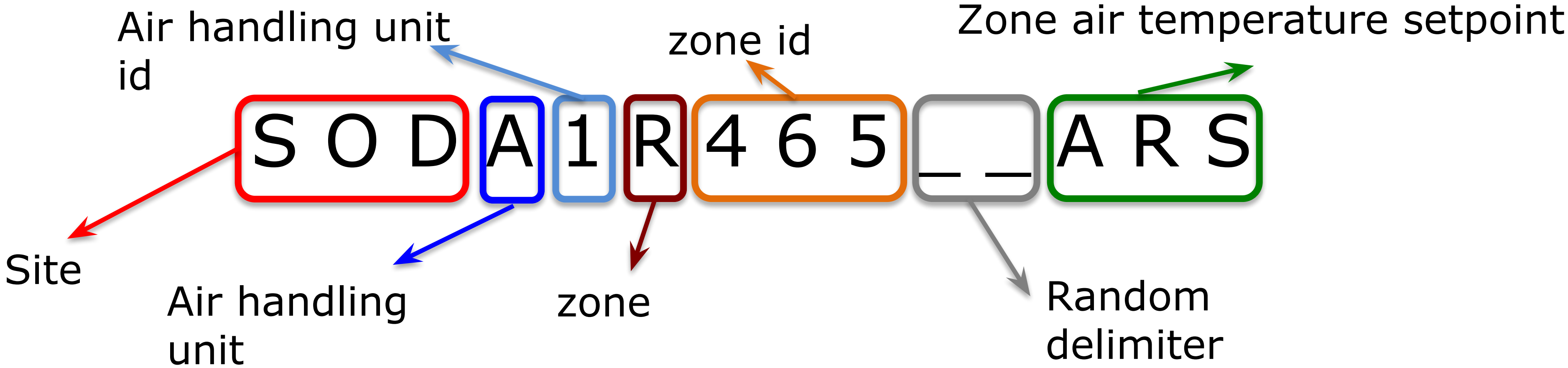
Data Interoperability in Buildings

- Most building telemetry resides within "data silos"
    - Proprietary, vendor-specific data repositories
- No common data representation:
    - Descriptions of buildings + subsystems dominated by informal and ad-hoc labels
    - Convention is fine for humans, but not for machines
    - Difficult to develop interoperable software
- Focus on point labels in this presentation

******** ANALOG INPUTS ********************************
"OA-T", Outside Air Temperature
"MA-T", Mixed Air Temperature
"DA-T" Discharge or Supply Air Temperature
"ZN-T" Zone or Space Temperature
"WC-ADJ", Warm/Cool Adjust (at the Wall sensor)
"RA-T", Return Air Temperature
"SA-P", Static Pressure Value (Duct Static)
******** ANALOG OUTPUTS ********************************
"DPR-O", Outside/Return Air Damper or Economizer Da
"HTG-O", Heating Valve Signal or analog signal to and e
"CLG-O", Cooling Valve Signal or analog signal to and e
"SF-O", Supply Fan Inlet Vane or VFD signal
******** BINARY INPUTS ********************************
"SF-S", Supply Fan Status
"RF-S", Return Fan Status
"SMK-S" or "SD-S", Smoke Detector Status (supervisory
"LL-S" or "LL1-S" Low Limit Status (aka. Freeze Stat)
"CHWP-S" Chiller Water Pump Status
"CWP-S" Condenser Pump Status
"HWP-S" Hot Water Pump Status

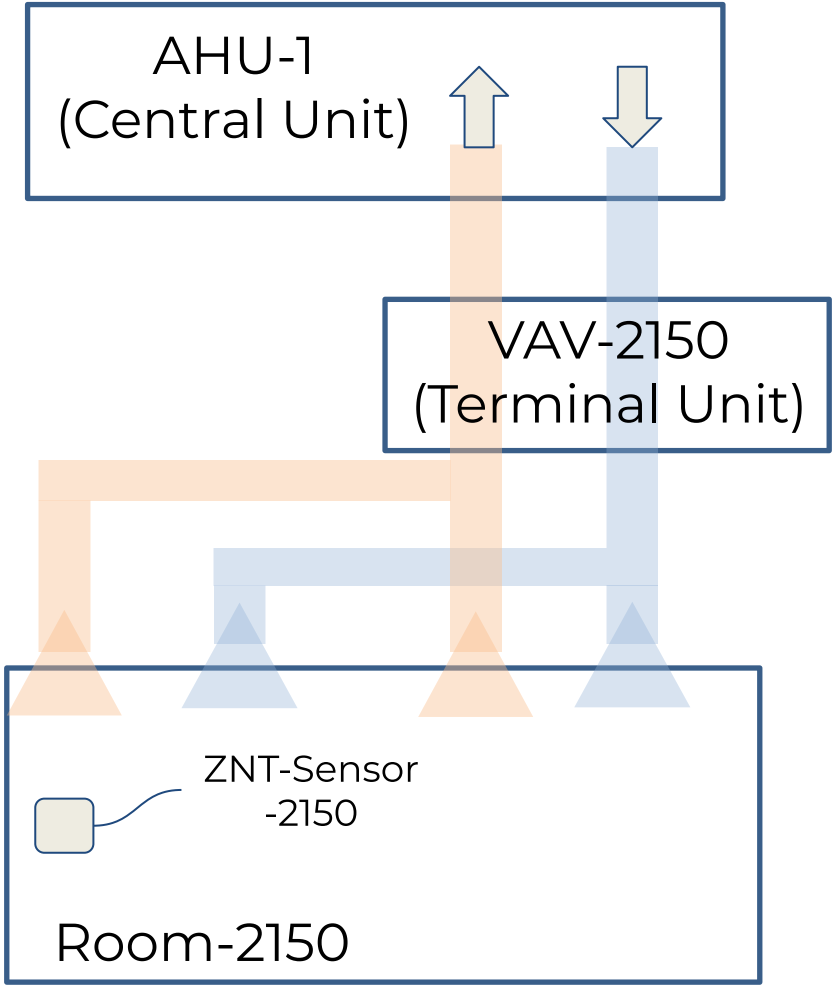
BMS Point Labels

- Points (sensors/actuators) combine location, function, related equipment, subsystem and related metadata in a single label
- Correct interpretation requires mix of site-specific conventions and implicit domain knowledge
- Established conventions are not consistent even within an enterprise



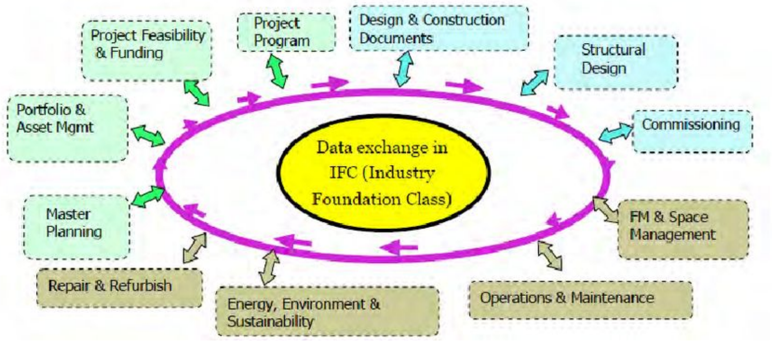
Building Metadata

- Broadly: “Data about data”
- In buildings:
  - Origin and context of collected telemetry
  - Sensor/setpoint/status/register the produced the data
  - What equipment or substance is controlled/monitored/regulated
  - Location of the data source
    - Physical location
    - Logical location
    - Position in a process or subsystem
    - Related equipment, points, etc



Existing Data Interoperability Standards

- Industry Foundation Classes:
  - Standardizes data exchange across design and construction phases of buildings
  - Focus on 3D geometrical modelling for space mgmt and asset tracking
  - Limited “data dictionary” defines generic assets used in building operations and mgmt
  - Inextensible data model, limited query mechanisms for software



Existing Data Interoperability Standards

- Project Haystack
  - Popular tagging system for building points and equipment
  - Replaces informal point labels with semi-structured sets of tags
  - Limited notion of “ref” tags for associating entities; relationships
  - Lack of formal rules for tag composition; improvement over unstructured labels
  - Tag dictionary partially covers HVAC, electrical subsystems

Project Haystack

```
id: @whitehouse
dis: "White House"
site
area: 55000ft²
geoAddr: "1600 Pennsylvania Avenue NW,  Washington, DC"
tz: "New_York"
weatherRef: @weather.washington
```

Existing Data Interoperability Standards

- (2015) study of 90 applications from building science literature
- 8 categories of applications:
    - Occupancy Modeling
    - Energy Apportionment
    - Web Displays + Dashboards
    - Model-Predictive Control
    - Participatory Feedback
    - Fault Detection and Diagnosis
    - Non-intrusive Load Monitoring
    - Demand Response
- Encapsulates state-of-the-art advances in modeling and control as well as standard industry practices

Bhattacharya, Arka, Joern Ploennigs, and David Culler. "Short Paper: Analyzing Metadata Schemas for Buildings: The Good, the Bad, and the Ugly." *BuildSys*, 2015.

Existing Data Interoperability Standards

- (2015) study of 90 applications from building science literature
- Identify the <u>entities</u> required:
    - What “things” does the application refer to?
- Identify the <u>relationships</u> required:
    - How do applications associate and find “things”?
- Existing standards do not meet the requirements of these applications

	IFC	SSN	Haystack
Tag Coverage	29%	11%	54%
Relationships	n/a	Only spatial	n/a

Bhattacharya, Arka, Joern Ploennigs, and David Culler. "Short Paper: Analyzing Metadata Schemas for Buildings: The Good, the Bad, and the Ugly." *BuildSys*, 2015.

Properties of an Effective Metadata Standard

- Represent **things** in buildings (physical, virtual, logical)
- Represent **relationships** between things
  - Things contained within other things
  - Things taking effect before other things in some process
  - Things affecting other things
  - Things fitting together to form a larger whole
- Extensible **classification** of things:
  - Named definitions
  - Add your own definitions, expand existing ones
- **Portability** and **consistency**:
  - Relationships and classifications should be generalizable to new situations
  - Obviate the need for site-/building-specific tagging and labeling schemes

Brick: a New Metadata Standard

- Graph-based metadata standard for smart buildings
- Capture physical, logical, virtual **entities** in buildings
- Define entities with an **extensible class hierarchy**
- Capture necessary **relationships** between entities

Brick

https://brickschema.org

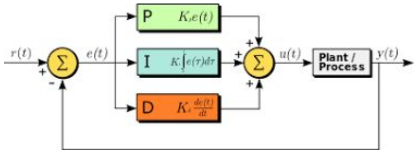
Brick Development Methodology

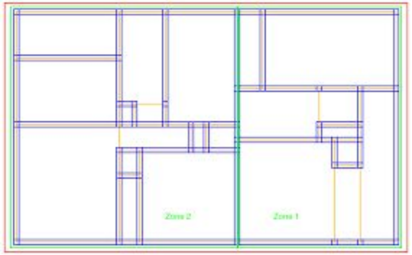
- 2015 BuildSys Conference: metadata is a problem
- Established working group of universities and companies working on metadata
    - “Bring a building”
    - BMS point dump + Ground truth
- Develop initial Brick class hierarchy and relationships
    - Derived empirically from actual BMS points, equipment
    - Relationships, classes driven by 2015 study’s application suite
    - Cross-validate Brick structure by implementing applications

Core Brick Concepts

Entity: abstraction of any physical, logical or virtual item; the “things” in a building

Thermostat A, Room 123 (physical); HVAC Zone 4, Temp. Sensor Class (logical), PID loop (virtual)

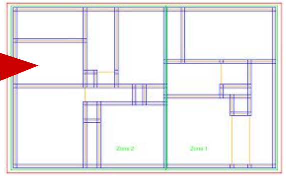




Core Brick Concepts

Relationship: defines the nature of a link between two related entities; includes encapsulation, composition, sequence, influence, control, instantiation, etc

Thermostat A is located in Room 123; AHU 1 is upstream of VAV 234; Thermostat A is a Thermostat



Thermostat definition

Core Brick Concepts

Class: a named category with intensional meaning (a definition) used for grouping entities; organized into a hierarchy; entities are instances of one or more classes

Thermostat, Temperature Sensor, Air Temperature Sensor, Room, VAV, HVAC Zone, Light, Meter

Blowdown_Water c

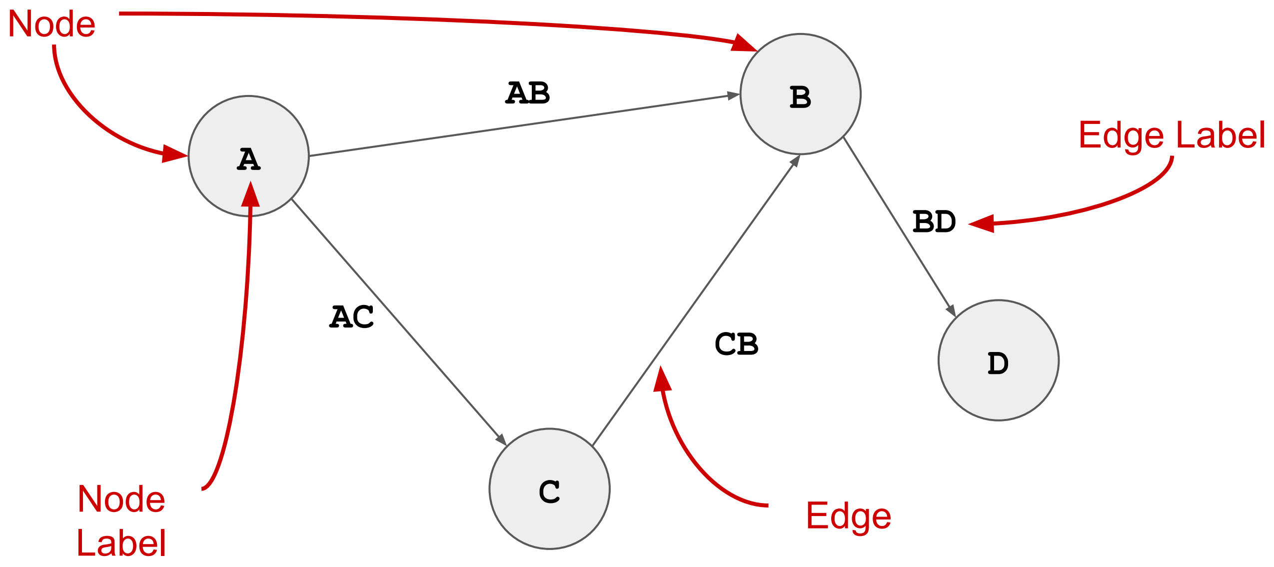
IRI	https://brickschema.org/schema/1.0.3/Brick#Blowdown_Water
Description	Water expelled from a system to remove mineral build up
Super-classes	brick:Water c

Root Classes

- Equipment
- Point
- Location
- Substance
- Quantity

Core Brick Concepts

Graph: an abstract organizational structure representing a set of entities (nodes) and relationships (edges)



Core Brick Concepts

Entity: abstraction of any physical, logical or virtual item; the “things” in a building

- Thermostat A, Room 123 (physical); HVAC Zone 4, Temp. Sensor Class (logical), PID loop (virtual)

Relationship: defines the nature of a link between two related entities; includes encapsulation, composition, sequence, influence, control, instantiation, etc

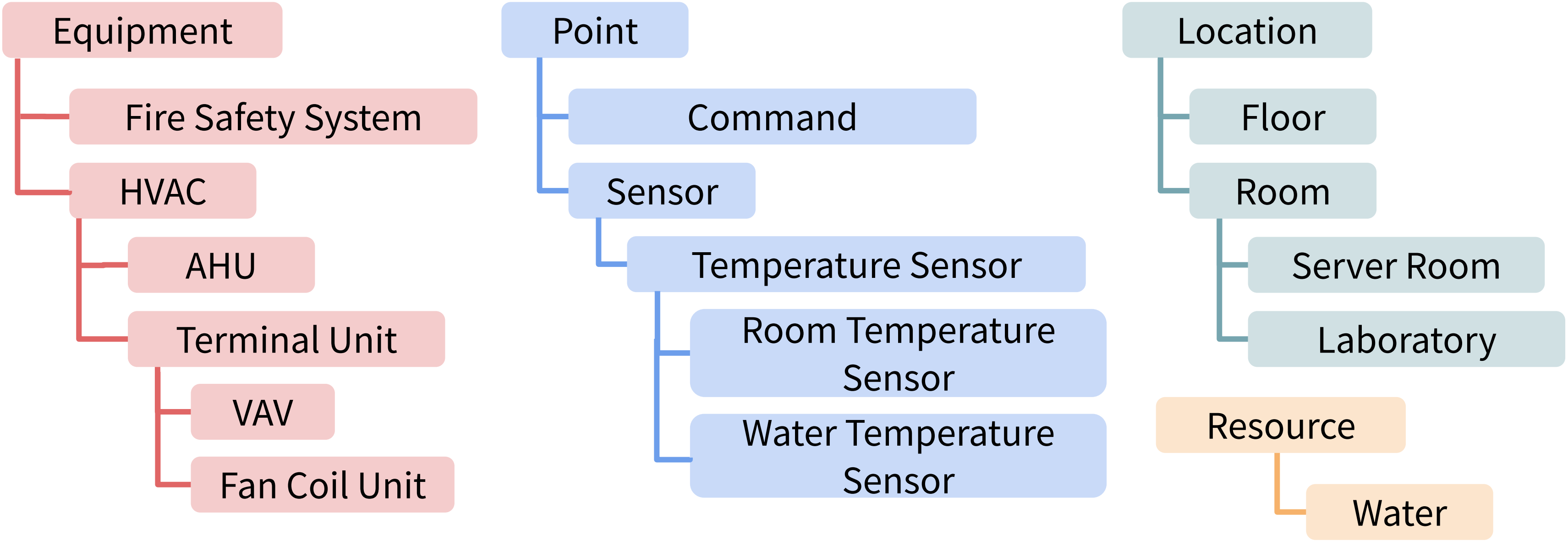
- Thermostat A <u>is located in</u> Room 123; AHU 1 <u>is upstream of</u> VAV 234; Thermostat A <u>is a</u> Thermostat

Class: a named category with intensional meaning (a definition) used for grouping entities; entities are instances of one or more classes

- Thermostat, Temperature Sensor, Air Temperature Sensor, Room, VAV, HVAC Zone, Light, Meter

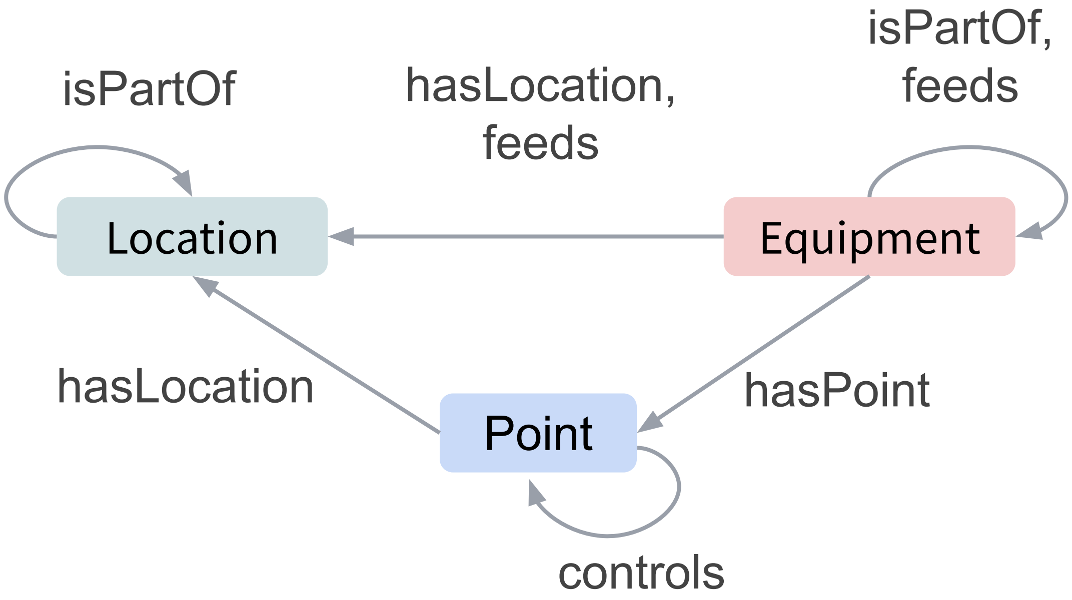
Graph: an abstract organizational structure representing a set of entities and relationships

Brick: Class Hierarchy



- Standardized class structure enables discoverability
- Extensible: allow site/deployment-specific classes

Brick: Relationships



- Relationships can be transitive, symmetrical
- Help Brick extend to cover new settings, equipment

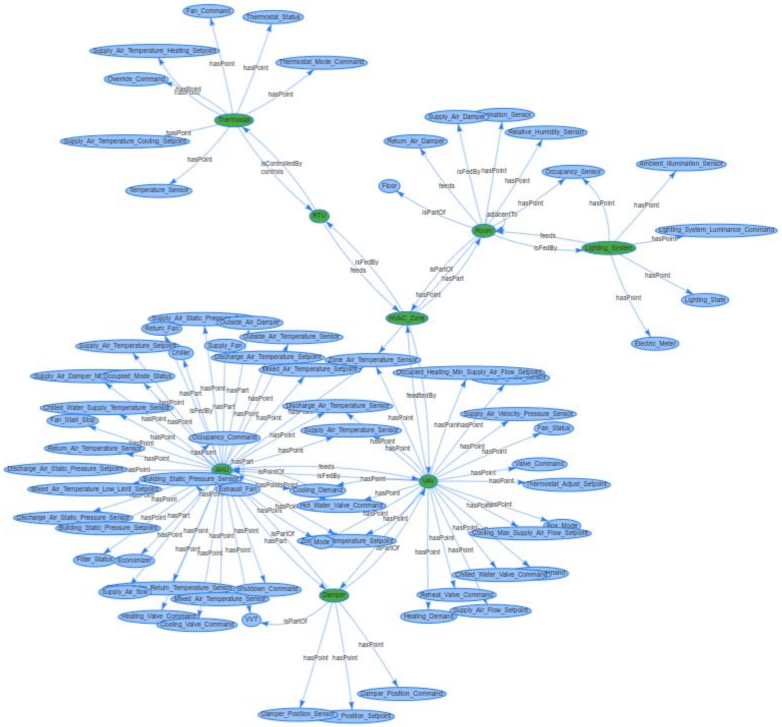
Brick: Relationships

Relationship	Definition	Domain	Range	Inverse	Transitive?
hasLocation	Subject is physically located in the object entity	*	Location	isLocationOf	yes
feeds	Subject conveys some media to the object entity in the context of some sequential process	Equipment Equipment	Equipment Location	isFedBy	yes
hasPoint	Subject has a monitoring, sensing or control point given by the object entity	Equipment Location	Point Point	isPointOf	no
hasPart	Subject is composed – logically or physically – in part by the object entity	Equipment Location	Equipment Location	isPartOf	yes
measures	Subject measures a quantity or substance given by the object entity	Sensor Sensor	Substance Quantity		no no
regulates	Subject informs or performs the regulation of the substance given by the object entity	Setpoint Equipment	Substance Substance		no
hasOutputSubstance	Subject produces or exports the object entity as a product of its internal process	Equipment	Substance		no
hasInputSubstance	Subject receives the object entity to conduct its internal process	Equipment	Substance		no

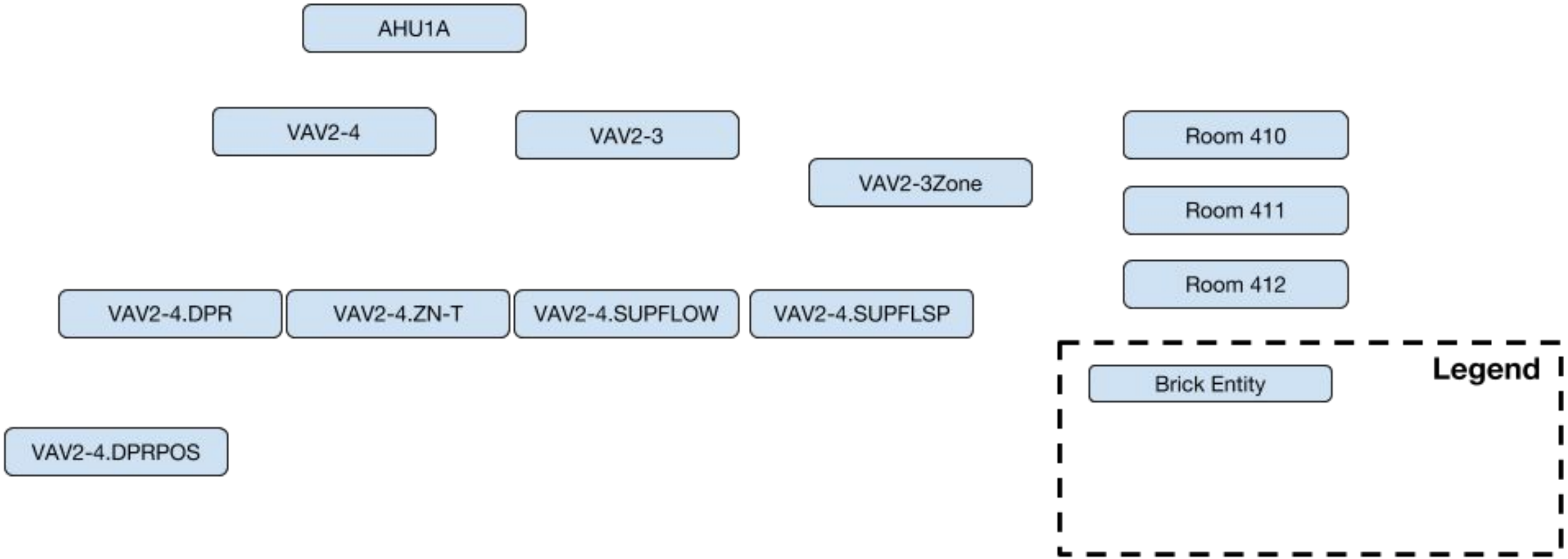
Modeling a Building with Brick

- A Brick model is a digital representation of a building

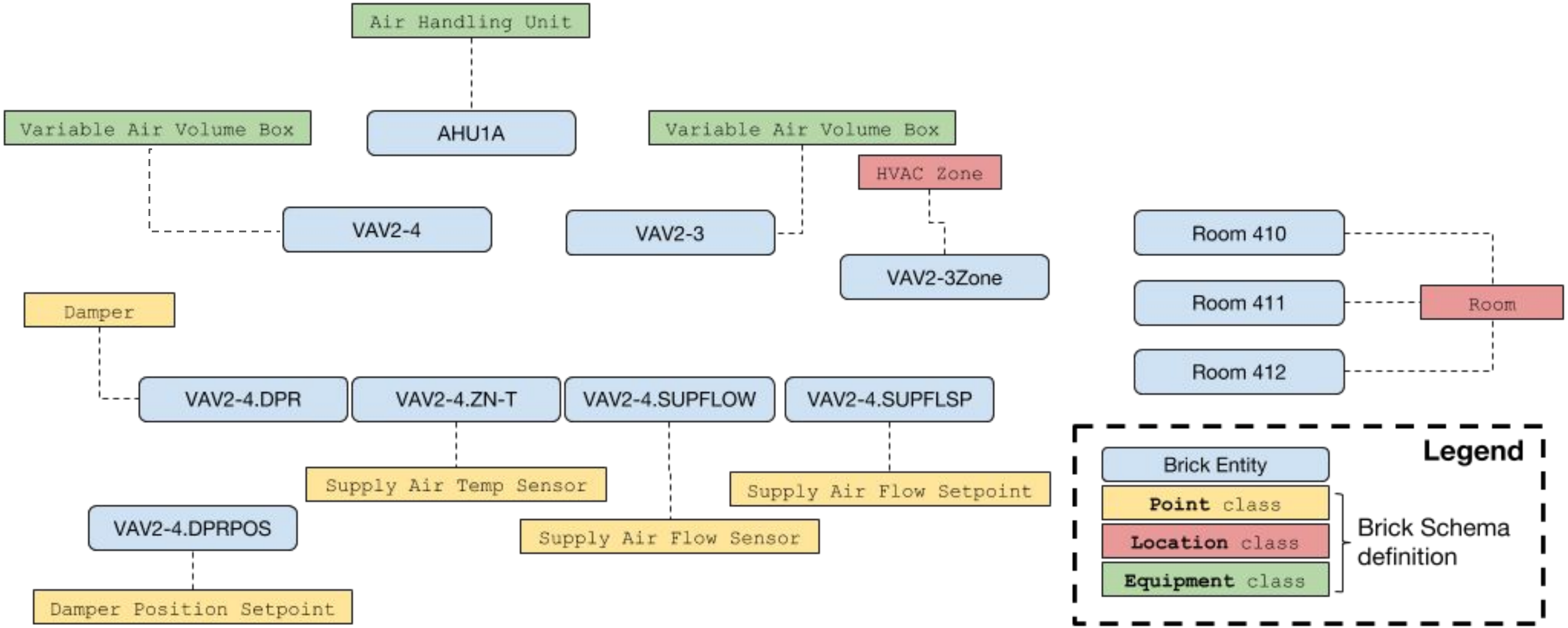
- **Nodes = “things”**
    - Building assets
    - Equipment
    - Subsystems
    - Class structure
- **Edges = “relationships”**
    - Location
    - Control
    - Connectivity
    - Composition
    - etc



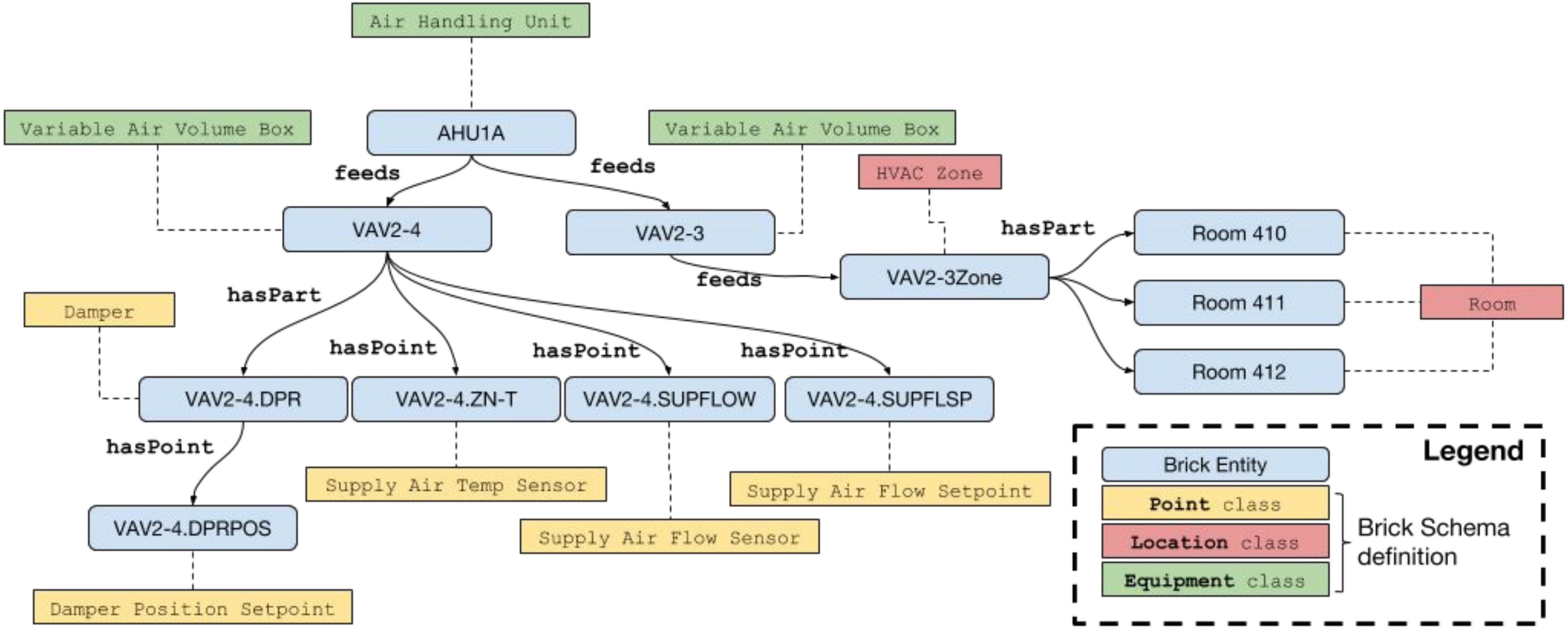
Modeling a Building with Brick

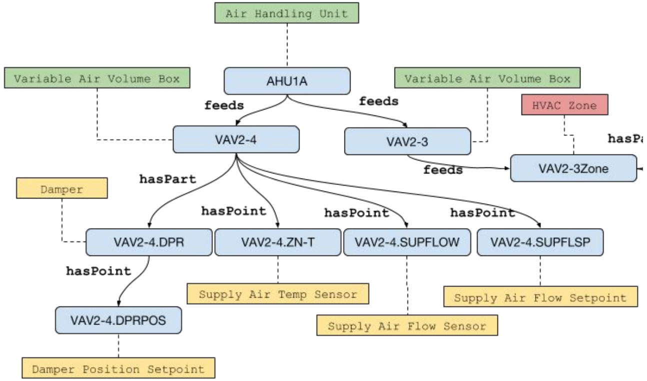


Modeling a Building with Brick



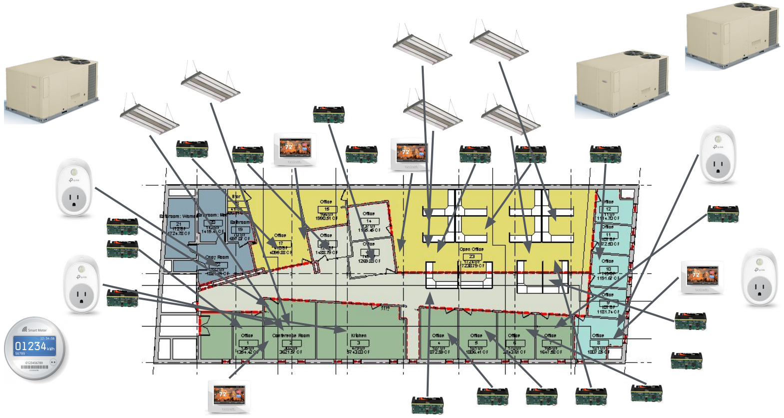
Modeling a Building with Brick





Brick model

represents



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

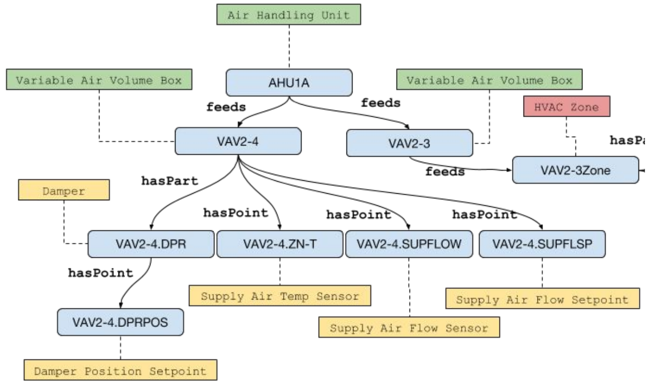
Query

```
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
};
```

python™

Application

queries



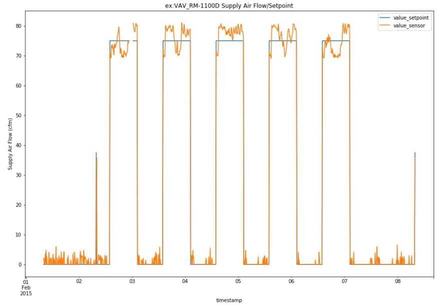
Brick model

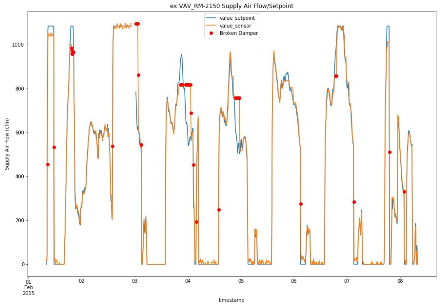
An application **queries** a Brick model to retrieve the data + configuration it needs

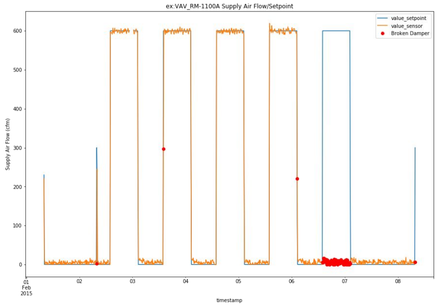
Query

```
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
};
```

executes



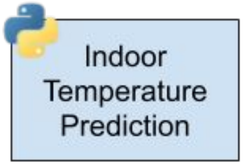




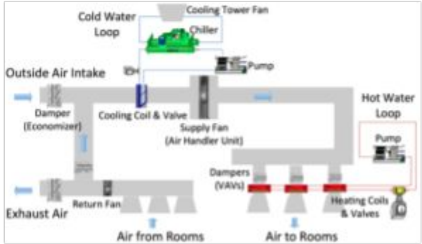
Queries allow apps to account for building heterogeneity and **customize their operation** to each building.

This is called **application portability**

The Utility of Portability

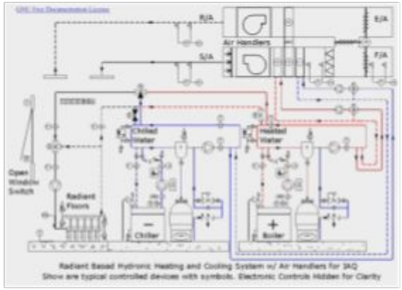


Indoor Temperature Prediction (AHU-based)



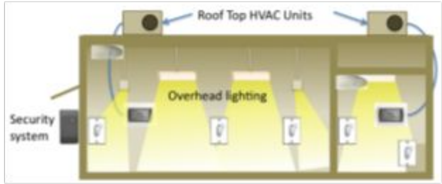
Air-based HVAC Systems

Indoor Temperature Prediction (Radiant)

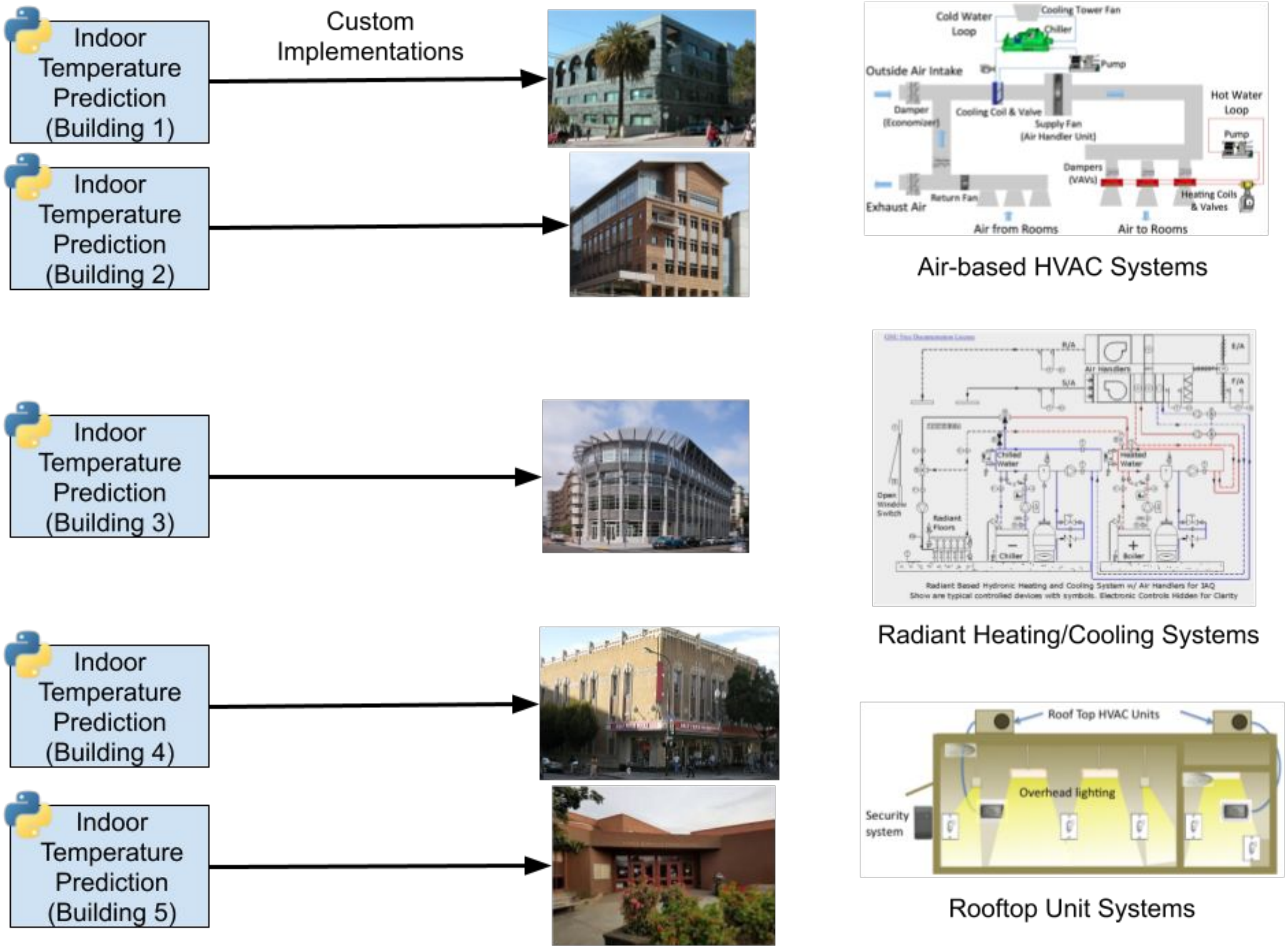


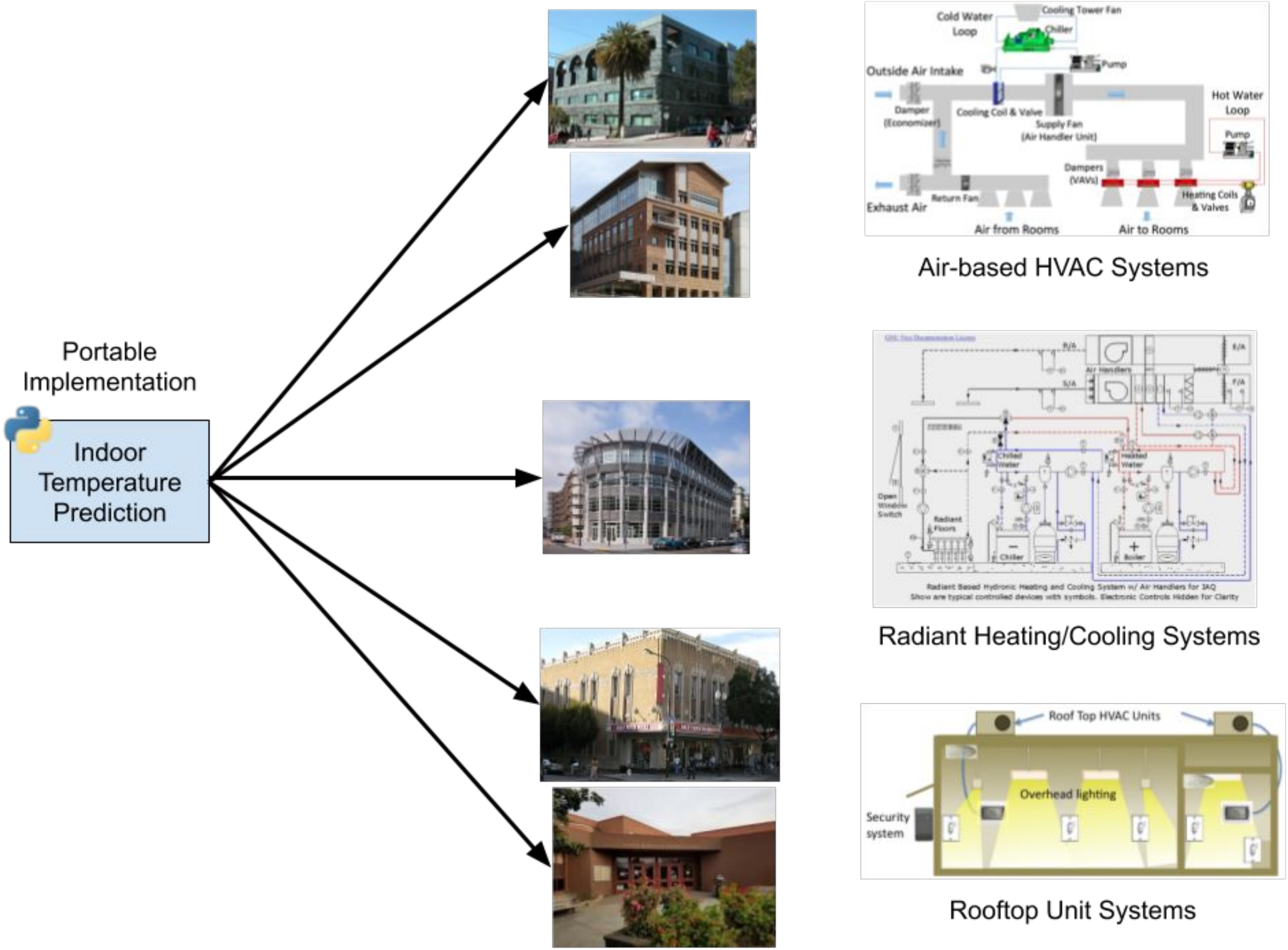
Radiant Heating/Cooling Systems

Indoor Temperature Prediction (RTU-basedl)



Rooftop Unit Systems


Custom
Implementations
Indoor
Temperature
Prediction
(Building 1)
Indoor
Temperature
Prediction
(Building 2)
Indoor
Temperature
Prediction
(Building 3)
Indoor
Temperature
Prediction
(Building 4)
Indoor
Temperature
Prediction
(Building 5)
Cold Water
Loop
Cooling Tower Fan
Chiller
Pump
Outside Air Intake
Hot Water
Loop
Damper
(Economizer)
Cooling Coil & Valve
Supply Fan
(Air Handler Unit)
Pump
Dampers
(VAVs)
Return Fan
Exhaust Air
Heating Coils
& Valves
Air from Rooms
Air to Rooms
Air-based HVAC Systems
Radiant Heating/Cooling Systems
Roof Top HVAC Units
Overhead lighting
Security
system
Rooftop Unit Systems


Portable Implementation
Indoor Temperature Prediction
Cold Water Loop
Cooling Tower Fan
Chiller
Pump
Outside Air Intake
Damper (Economizer)
Cooling Coil & Valve
Supply Fan (Air Handler Unit)
Hot Water Loop
Pump
Dampers (VAVs)
Return Fan
Heating Coils & Valves
Exhaust Air
Air from Rooms
Air to Rooms
Air-based HVAC Systems
Radiant Heating/Cooling Systems
Roof Top HVAC Units
Overhead lighting
Security system
Rooftop Unit Systems

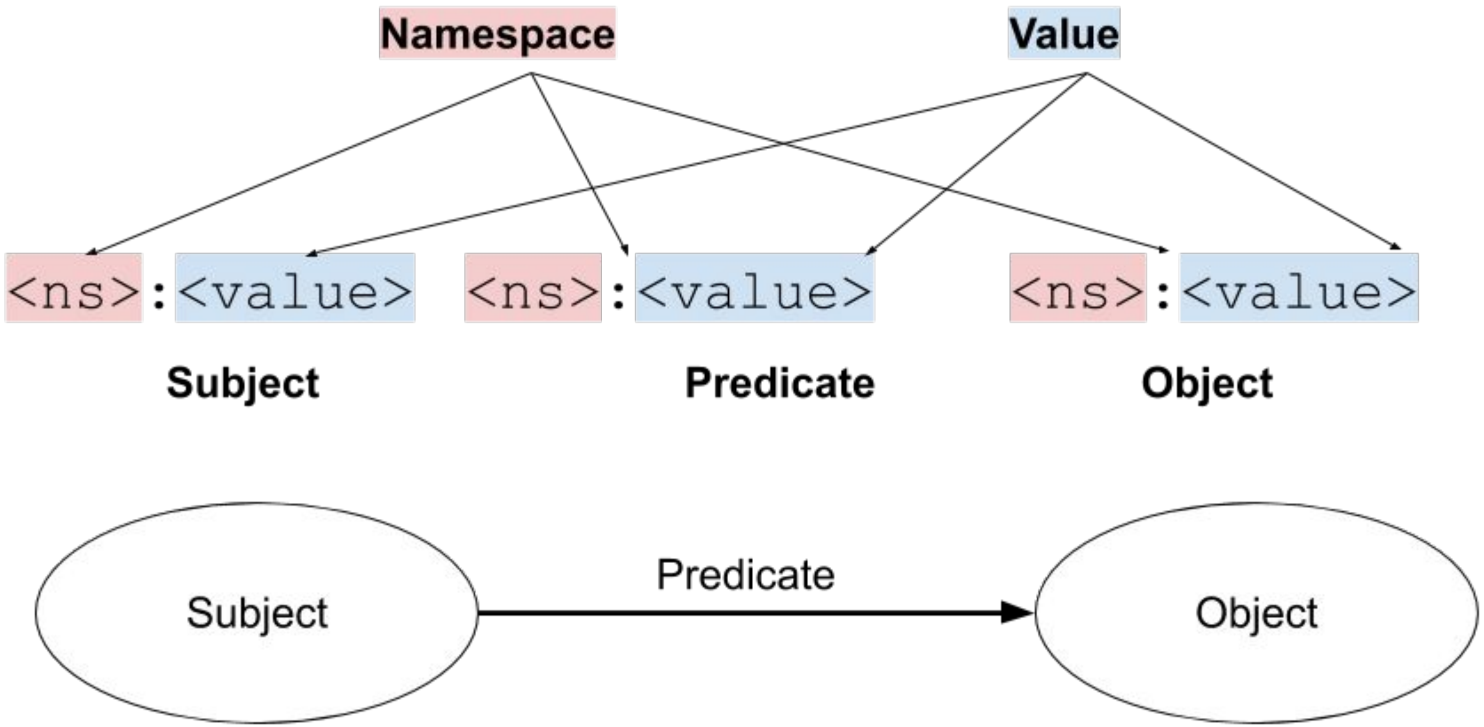
Data Model + Representation

- Brick models (graphs) represented with Semantic Web technology

- Resource Description Framework (RDF):
  - Structured statements about resources
  - Declare properties of resources, relationships to other resources
  - RDF statements are called **triples**
  - Set of triples defines the directed, labeled graph

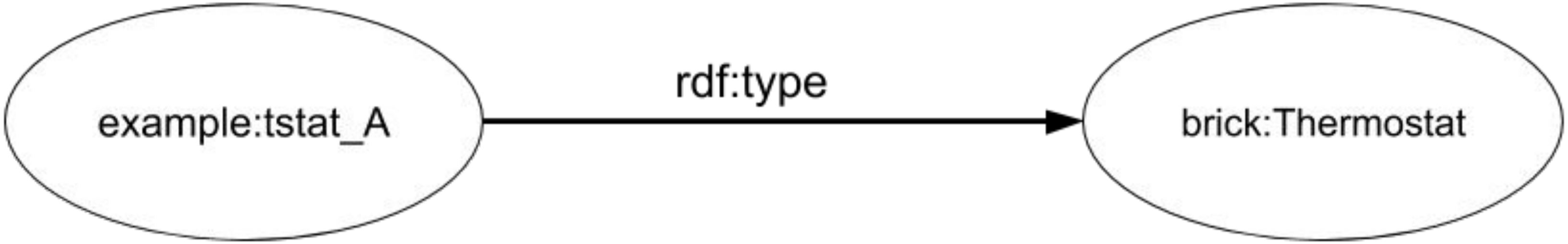
- “Triples” are 3-tuples of terms
  - Terms have **namespace** and a **value**
  - Namespaces provide scoping for values
  - Values are the names of entities

RDF Triple (Generic)

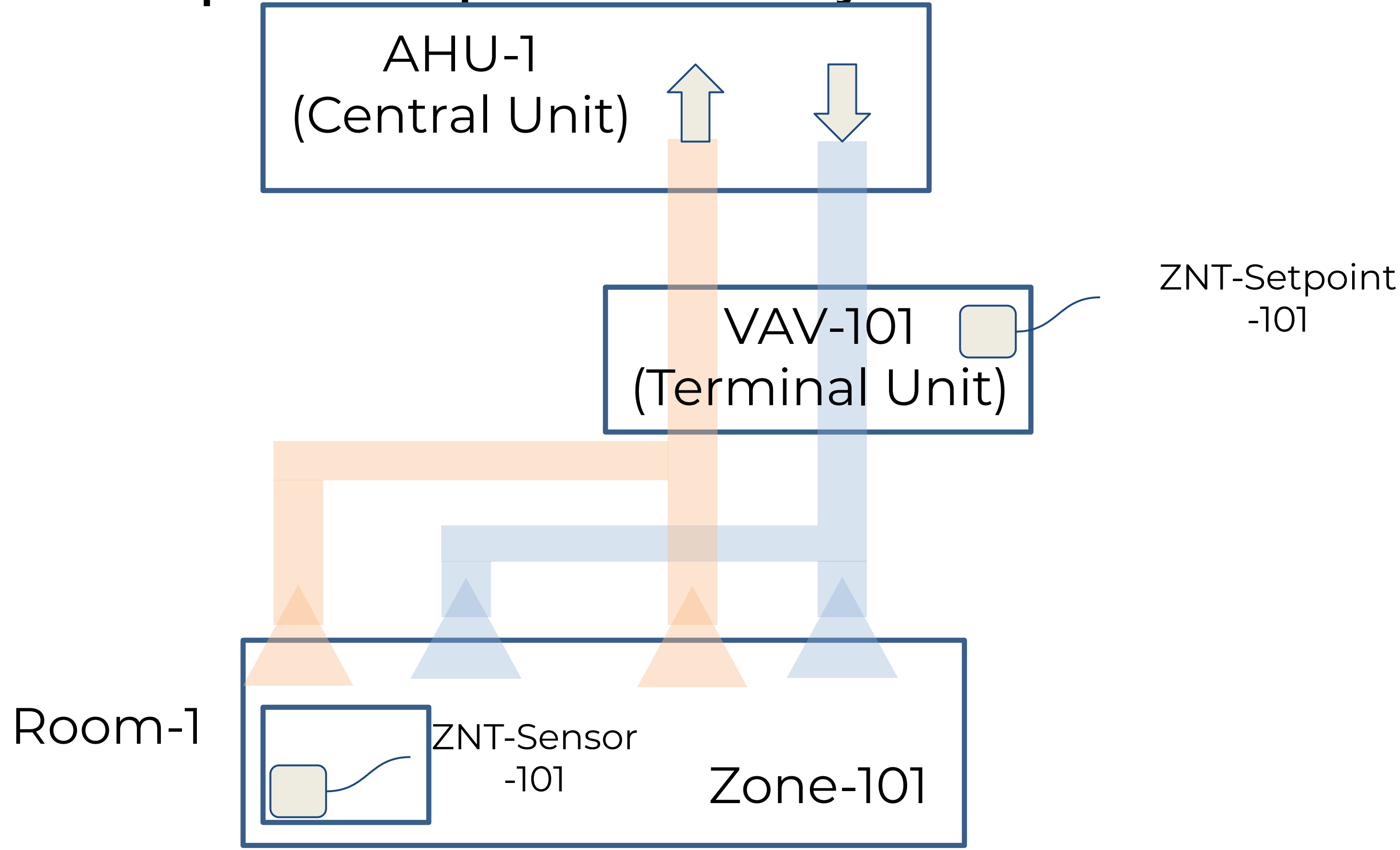


RDF Triple

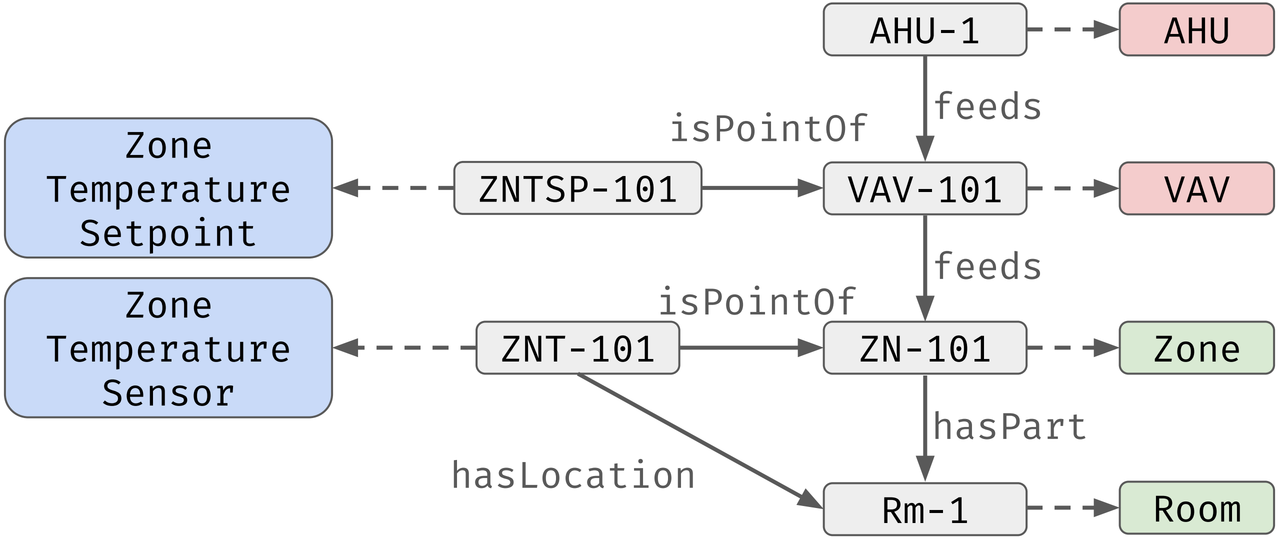
`example:tstat_A`	`rdf:type`	`brick:Thermostat`
Subject	**Predicate**	**Object**



Brick Example: Simple HVAC System



Brick Example: Simple HVAC System



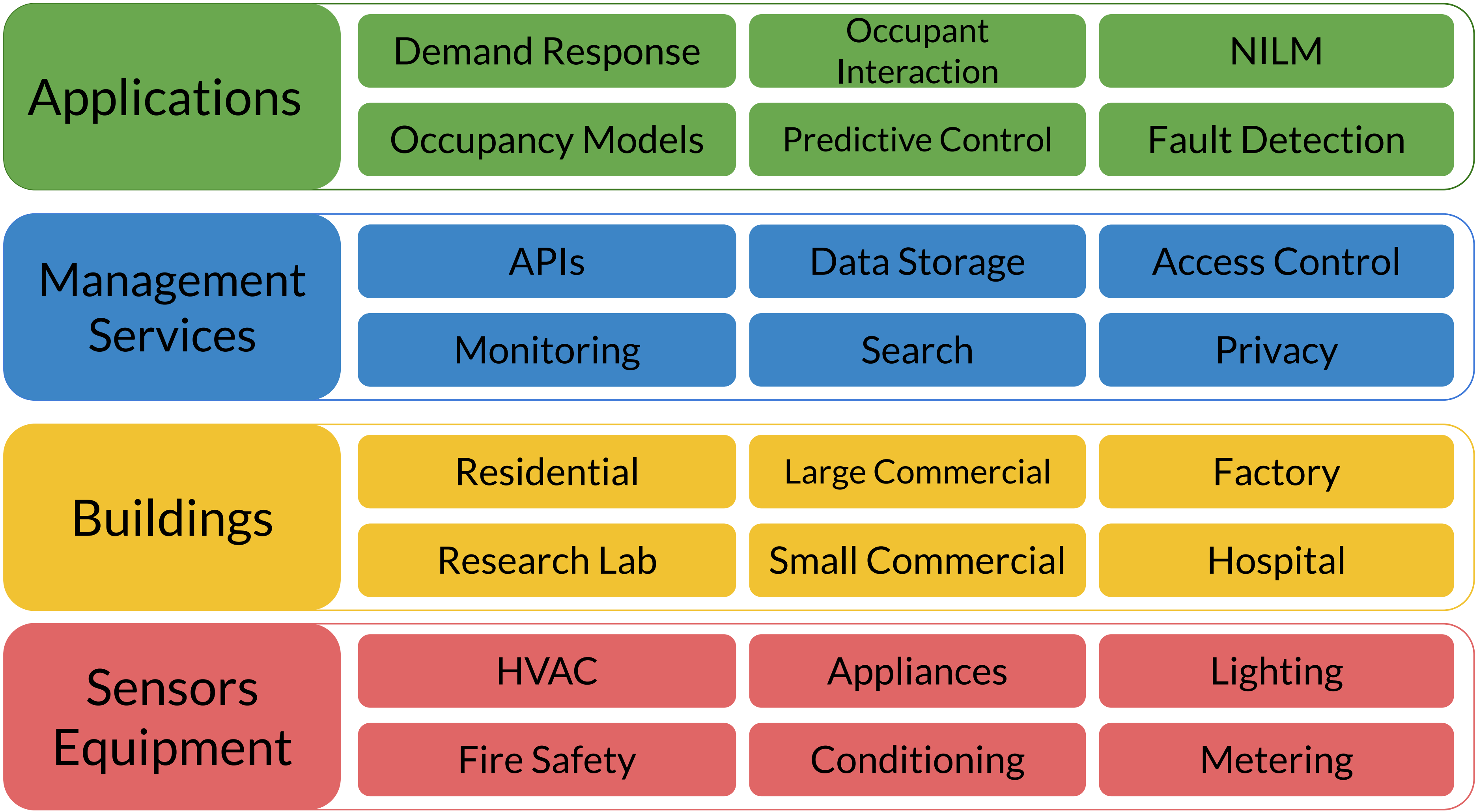
Brick Example: Simple HVAC System

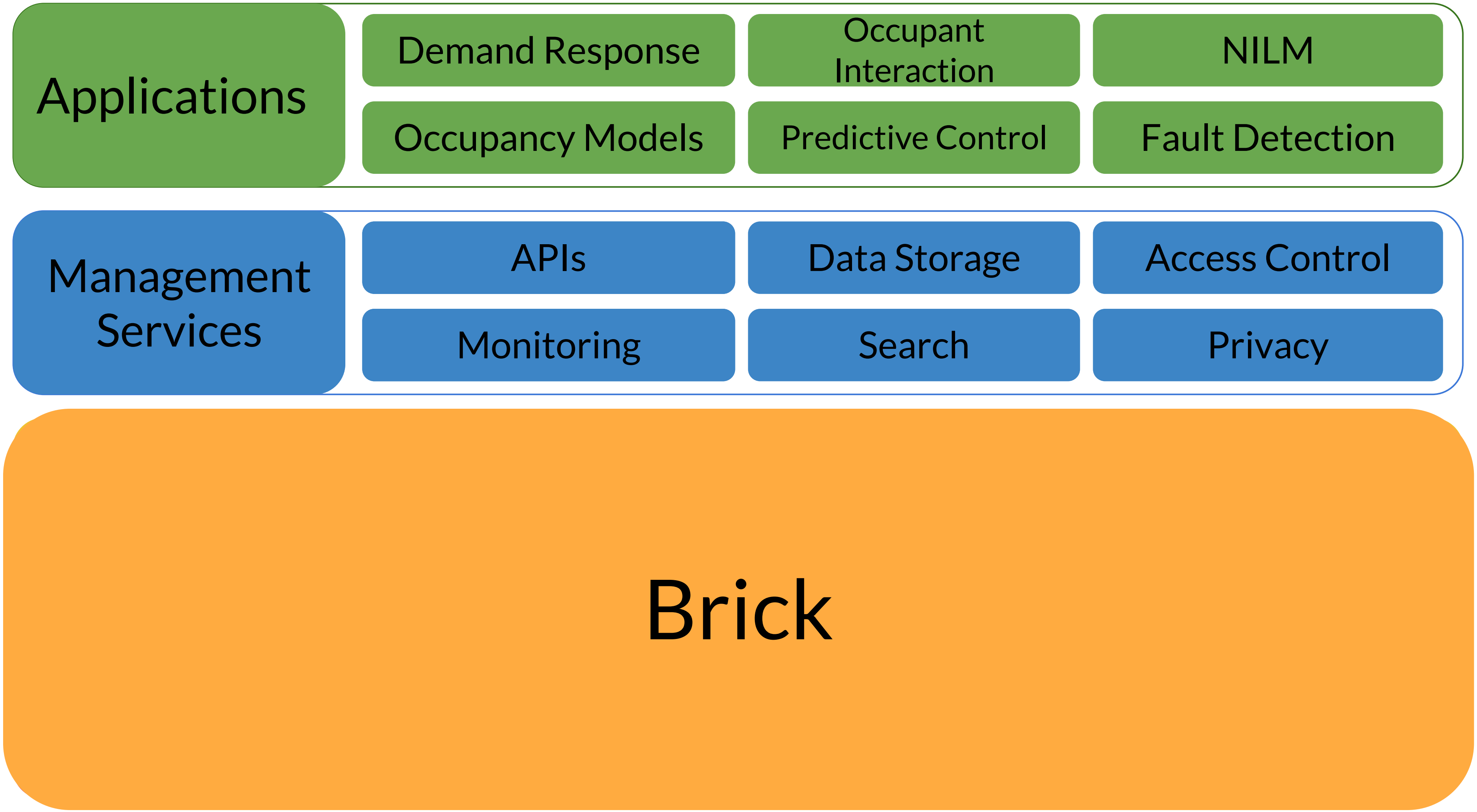
```
example:AHU-1         rdf:type                    brick:AHU
example:VAV-101       rdf:type                    brick:VAV
example:ZN-101        rdf:type                    brick:HVAC_Zone
example:Rm-1          rdf:type                    brick:Room
example:ZNTSP-101         rdf:type
brick:Zone_Temperature_Setpoint
example:ZNT-101       rdf:type                    brick:Zone_Temperature_Sensor

example:AHU-1         brick:feeds                 example:VAV-101
example:VAV-101       brick:feeds                 example:ZN-101
example:ZN-101        brick:hasPart               example:RM-1
example:ZNT-101       brick:isPointOf             example:ZN-101
example:ZNT-101       brick:hasLocation               example:ZN-101
example:ZNTSP-101         brick:isPointOf             example:VAV-101
```

Project Haystack (revisited)

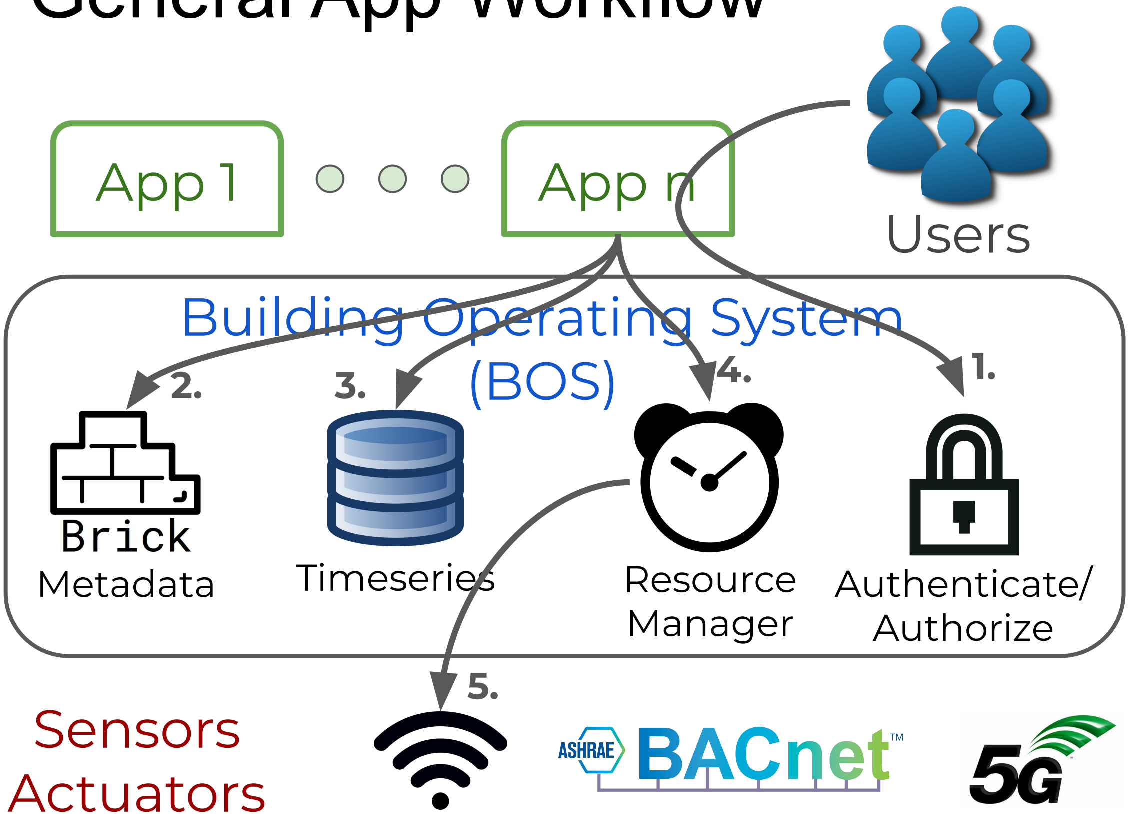
- Haystack uses **tags** to define entities
- Using tags for meaning trades **consistency** for **composability**
- Lack of formal rules for composition results in ambiguous interpretations (see table)
- Highly variable modeling practices; little consistency between buildings
- Lack of expressive relationships limits generalizability of Haystack models

	Tag	Desc. equip	Desc. point	Desc. mechanism	For AHU	For VAV	For Coil	For Valve	For Chiller	For Boiler
heating	heat	×	×				×	×		
	heating		×							
	hotWaterHeat	×		×	×					
	gasHeat	×		×	×					
	elecHeat	×		×	×					
	steamHeat	×		×	×					
	perimeterHeat	×				×				
reheating	reheat		×			×				
	reheating		×			×				
	hotWaterReheat	×		×		×				
	elecReheat	×		×		×				
cooling	cool	×			×	×	×			
	cooling		×							
	coolOnly	×			×					
	dxCool	×		×	×					
	chilledWaterCool	×		×	×					
	waterCooled	×							×	
	airCooled	×							×	


Applications
Demand Response
Occupant Interaction
NILM
Occupancy Models
Predictive Control
Fault Detection
Management Services
APIs
Data Storage
Access Control
Monitoring
Search
Privacy
Buildings
Residential
Large Commercial
Factory
Research Lab
Small Commercial
Hospital
Sensors Equipment
HVAC
Appliances
Lighting
Fire Safety
Conditioning
Metering


Applications
Demand Response
Occupant Interaction
NILM
Occupancy Models
Predictive Control
Fault Detection
Management Services
APIs
Data Storage
Access Control
Monitoring
Search
Privacy
Brick

General App Workflow



1. The user gets authenticated
2. The app queries resources. (e.g., SPARQL)
3. The app requests sensor data, or
4. The app set actuators/setpoints
5. The BOS sets the actuators

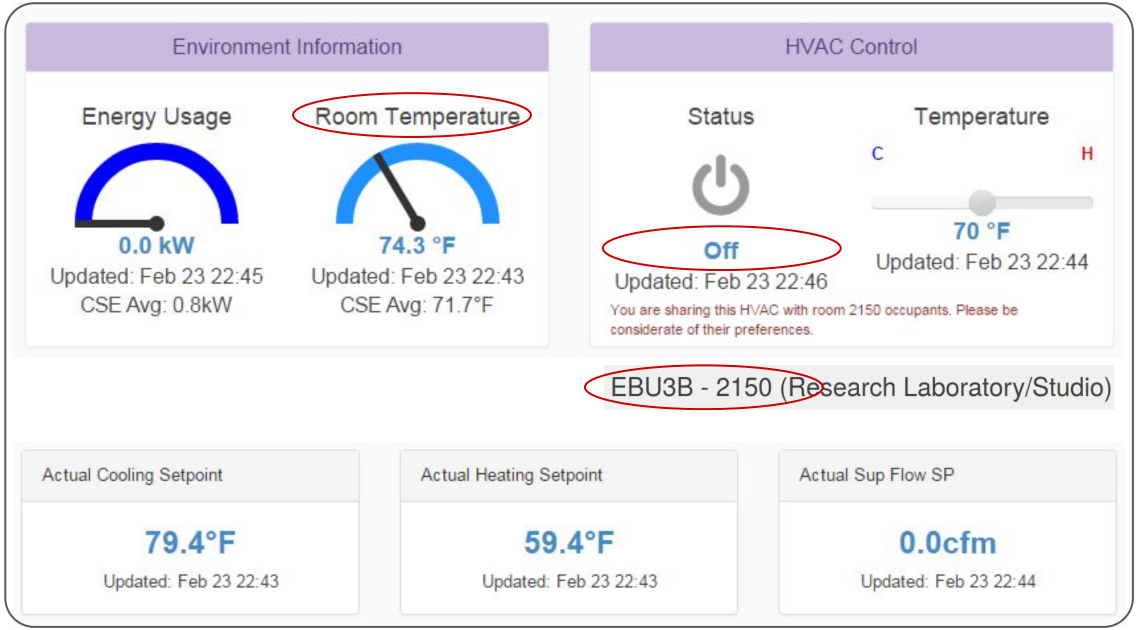
Building Applications (under dev)

- HVAC Web Interface: **Genie** https://github.com/jbkoh/genie-brickified
- Energy Disaggregation: **HVACMeter** https://github.com/jbkoh/hvacmeter
- Energy Visualizations: **VizEnergy** https://github.com/jbkoh/vizenergy
- Demand Response Model: **XBOS** https://github.com/SoftwareDefinedBuildings/xbos
- Data Analytics Framework: **Mortar** https://mortardata.org/

Brick App: *Genie*, a Web Thermostat

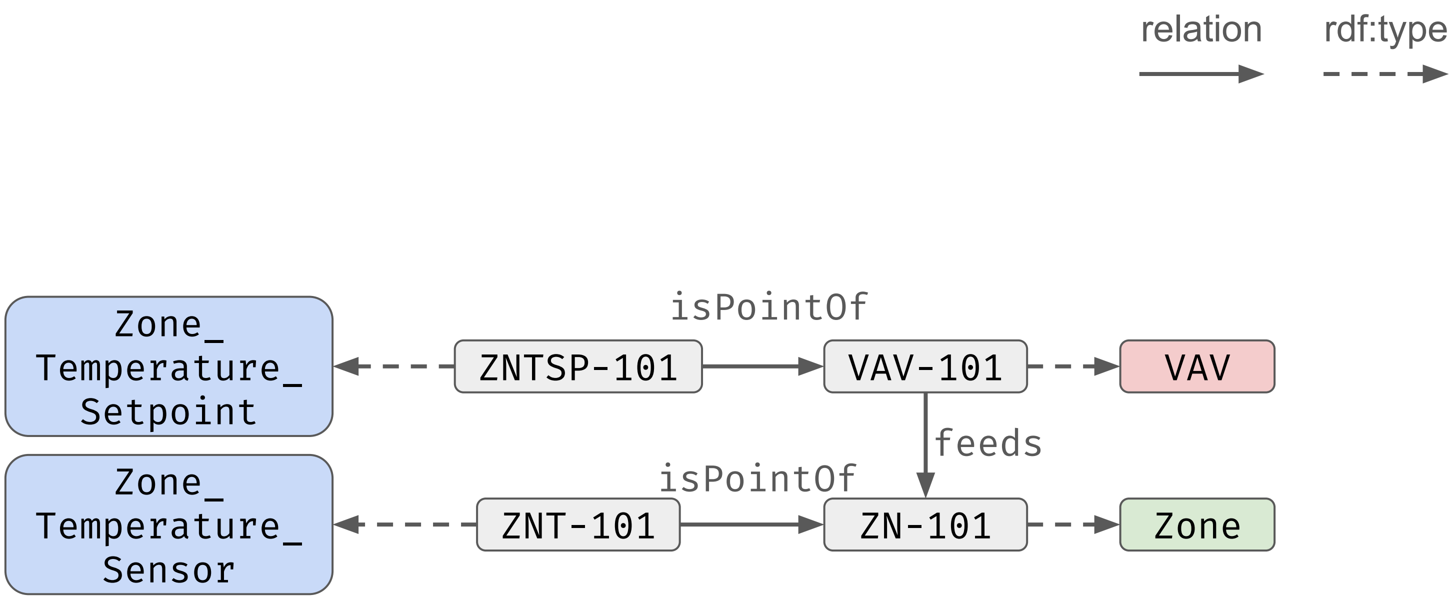
Why software thermostat?

1. Accessibility
   a. A physical thermostat in the next room
2. Configurability
   a. Schedules, setpoints are hard-coded in thermostats
3. Information Visualization
   a. Energy feedback, other parameters, etc.

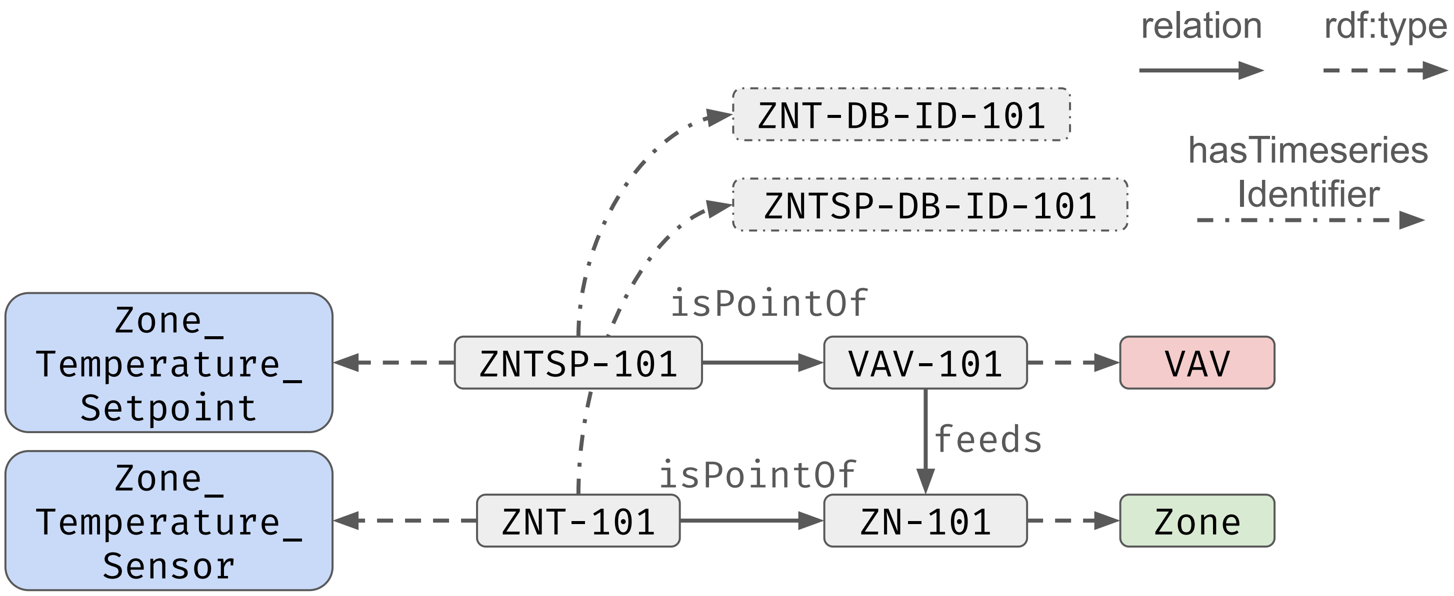


Genie's Goal: Understand the room's A/C status and control the A/C.

Example Building



Example Building



An entity can be associated with a pointer to timeseries data entry.

Brick Query Design Process

1. What types of entities are needed?
2. What are the relationships of the entities with others?
    a. What are the types of the other entities?
    b. What are the relationships of such entities with others?
        i. …
        ii. ...

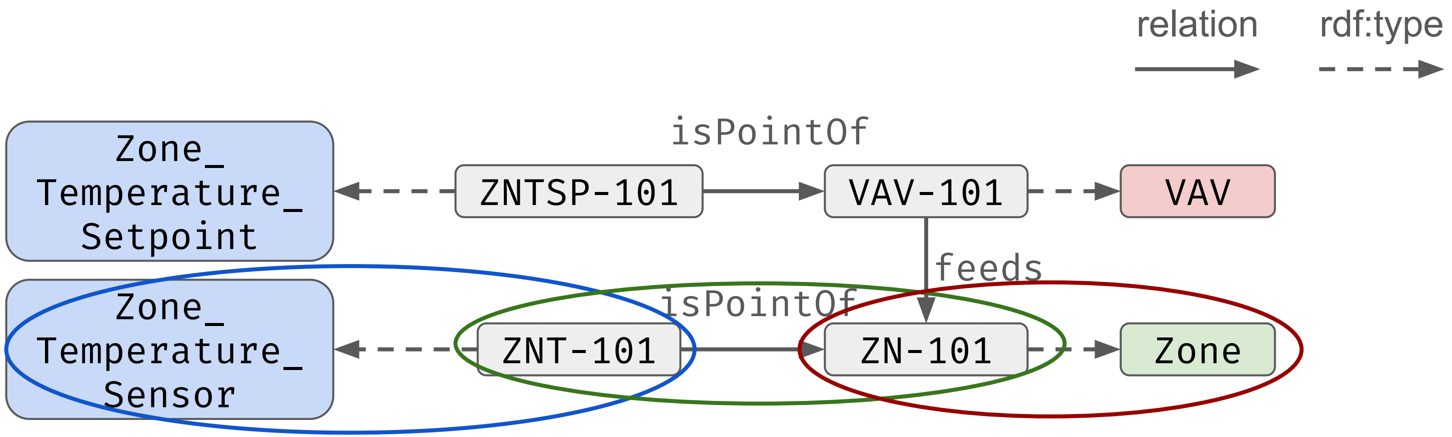
Query 1: get temperature: **What is the temperature sensor of a zone?**

1. What are the instances of Zone_Temperature_Sensor?
   - `?znt    rdf:type    brick:Zone_Temperature_Sensor.`

2. Which one among the instances measures a Zone?
   - `?znt    brick:isPointOf    ?zone.`

3. What are the instance of Zone?
   - `?zone    rdf:type    brick:Zone.`

Genie Query 1 Execution



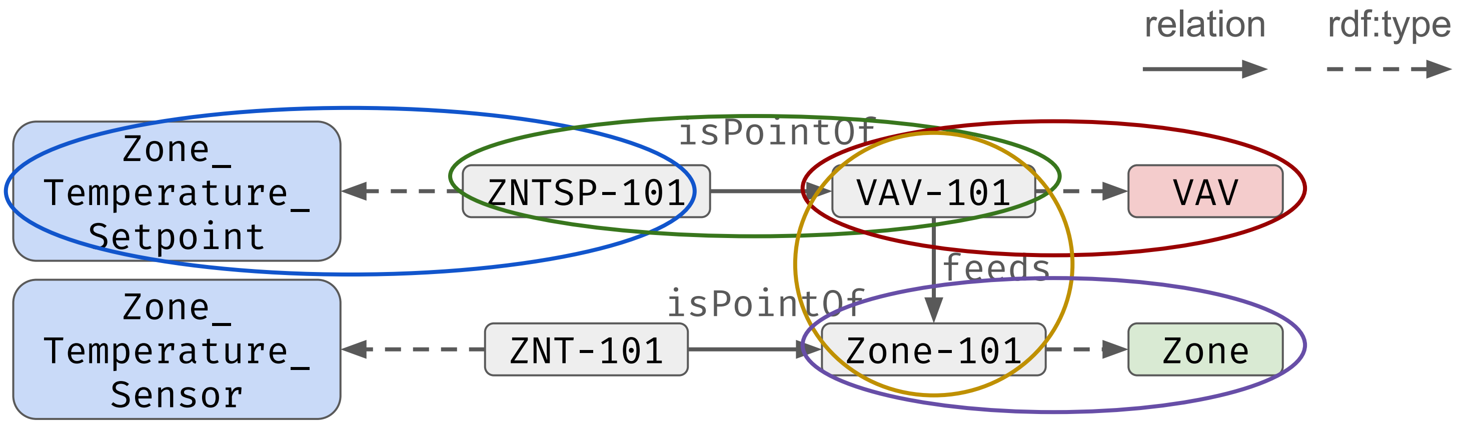

```
SELECT ?znt ?zone WHERE {
  ?znt    rdf:type         brick:Zone_Temperature_Sensor.  #1
  ?znt    brick:isPointOf  ?zone.                          #2
  ?zone   rdf:type         brick:Zone.                     #3
}
```

Query 2: set temperature:

What is the temperature setpoint of a terminal unit associated with this zone?

1. What are the instances of Zone_Temperature_Setpoint?
   - `?zntsp    rdf:type    brick:Zone_Temperature_Setpoint.`
2. Which ones among the instances do work for the A/C unit?
   - `?zntsp    brick:isPointOf    ?ac.`
3. If needed, constrain the type of A/C into VAVs (or any Terminal_Units)
   - `?ac    rdf:type    brick:VAV.`
   - `(?ac    rdf:type    brick:Terminal_Unit).`
4. What does feed air into the user's office?
   - `?ac    brick:feeds    ?zone.`
5. What does feed air into the user's office?
   - `?ac    rdf:type    brick:Zone.`

Genie Query 2 Execution




```
SELECT ?zntsp ?zone WHERE {
  ?zntsp  rdf:type          brick:Zone_Temperature_Setpoint.  #1
  ?zntsp  brick:isPointOf   ?ac.                              #2
  ?ac     rdf:type          brick:Terminal_Unit.              #3
  ?ac     brick:feeds       ?zone.                            #4
  ?zone   rdf:type          brick:Zone.                       #5
```

Genie Query Summary

1. Point Types
   a. `Zone_Temperature_Sensor, Zone_Temperature_Setpoint`
2. Local relationship
   a. `isPointOf Zone-101`
3. Functional relationship
   a. `feeds Zone-101`

Brick Toolchain

1. Brick-optimized Database: HodDB
    a. An external Tool: Virtuoso
2. Schema Viewer: BrickViewer, BrickWebsite
    a. An external Tool: Protege
3. Brick-enabled Building OSes: BrickServer, XBOS
4. Brick Authoring Tool: BrickStudio
5. Metadata Normalization Tool: Plaster
6. Open Testbed: Mortar
7. Apps: Genie, HVACMeter, VizEnergy, XBOS

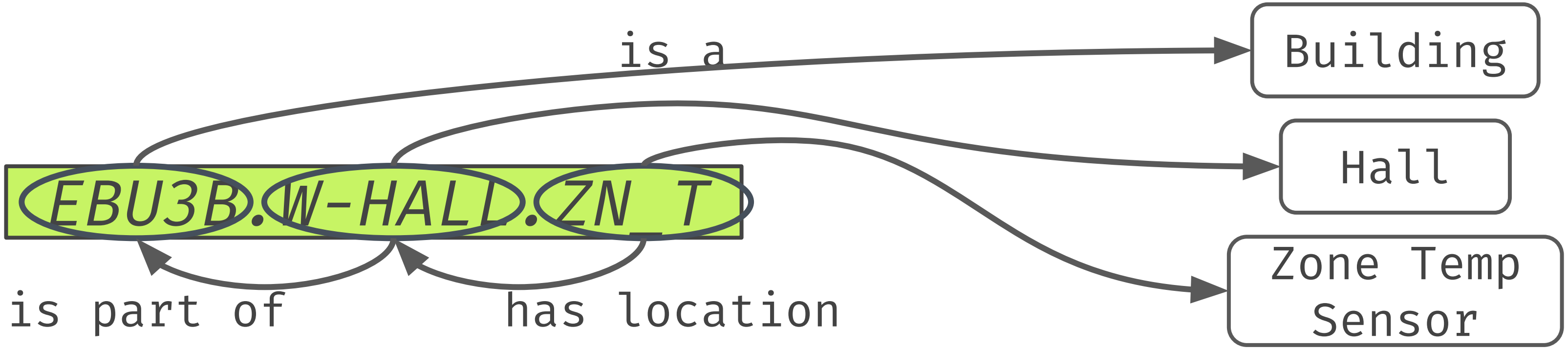
How to Convert Existing Buildings into Brick?

1. Buildings are heterogeneous
   a. Different devices, naming conventions, human errors, etc.
2. The instantiation process is highly manual
   a. Many vocabularies, diverse (but implicit) relationships

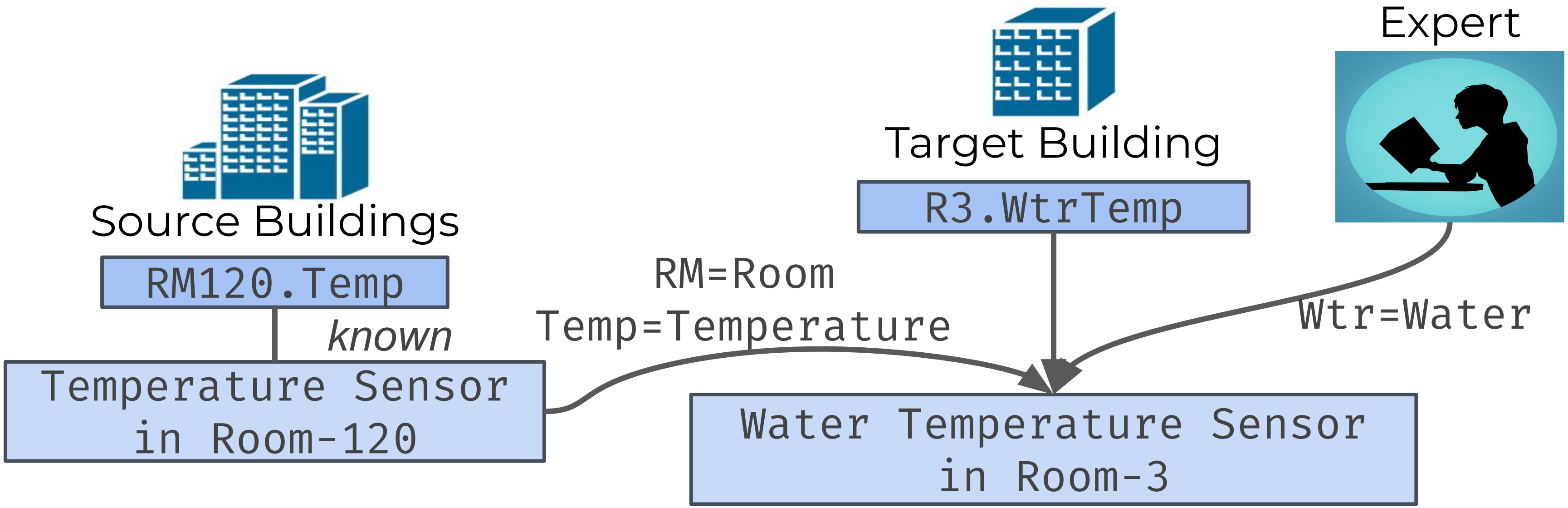
How to automate the conversion process?

1. Machine learning can help!
   a. Learning from various data sources
   b. Minimize the human effort

Parsing Metadata

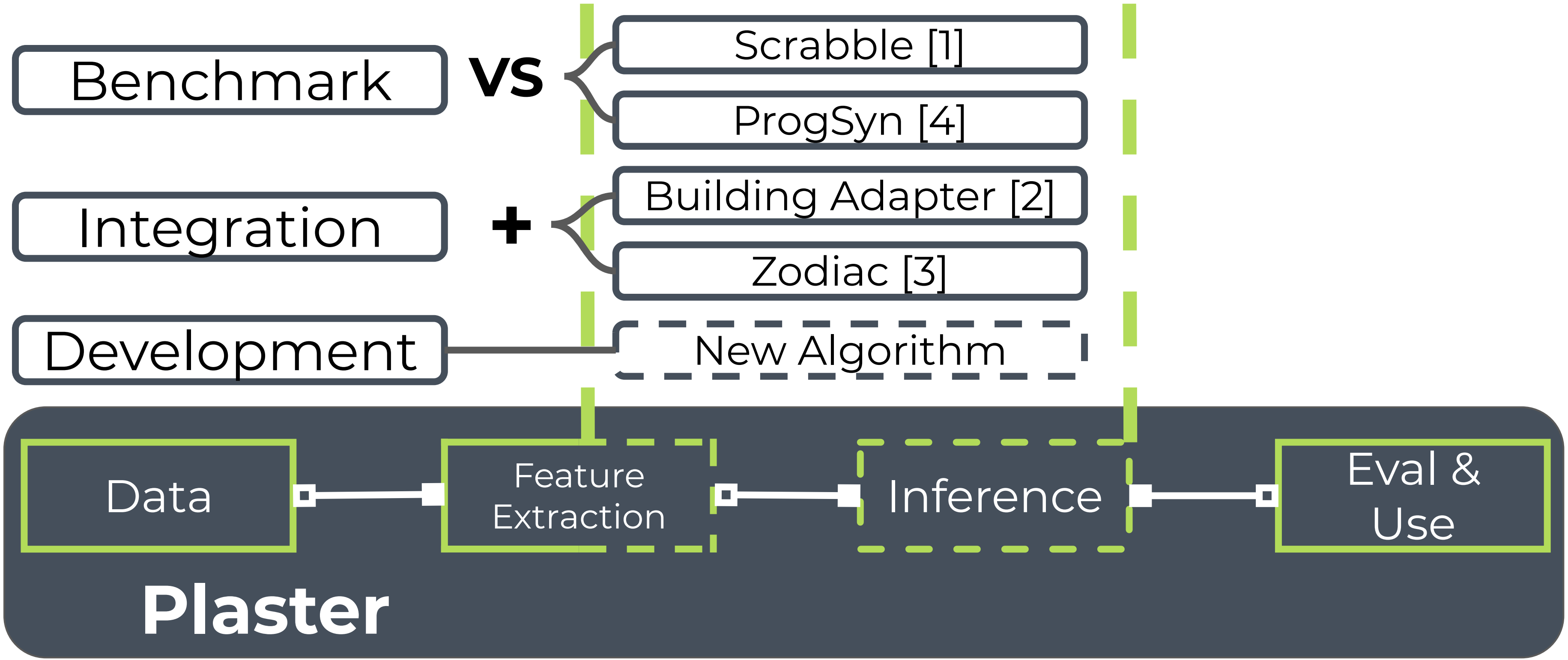


Reusing the Known Labels
+ Interactive Learning



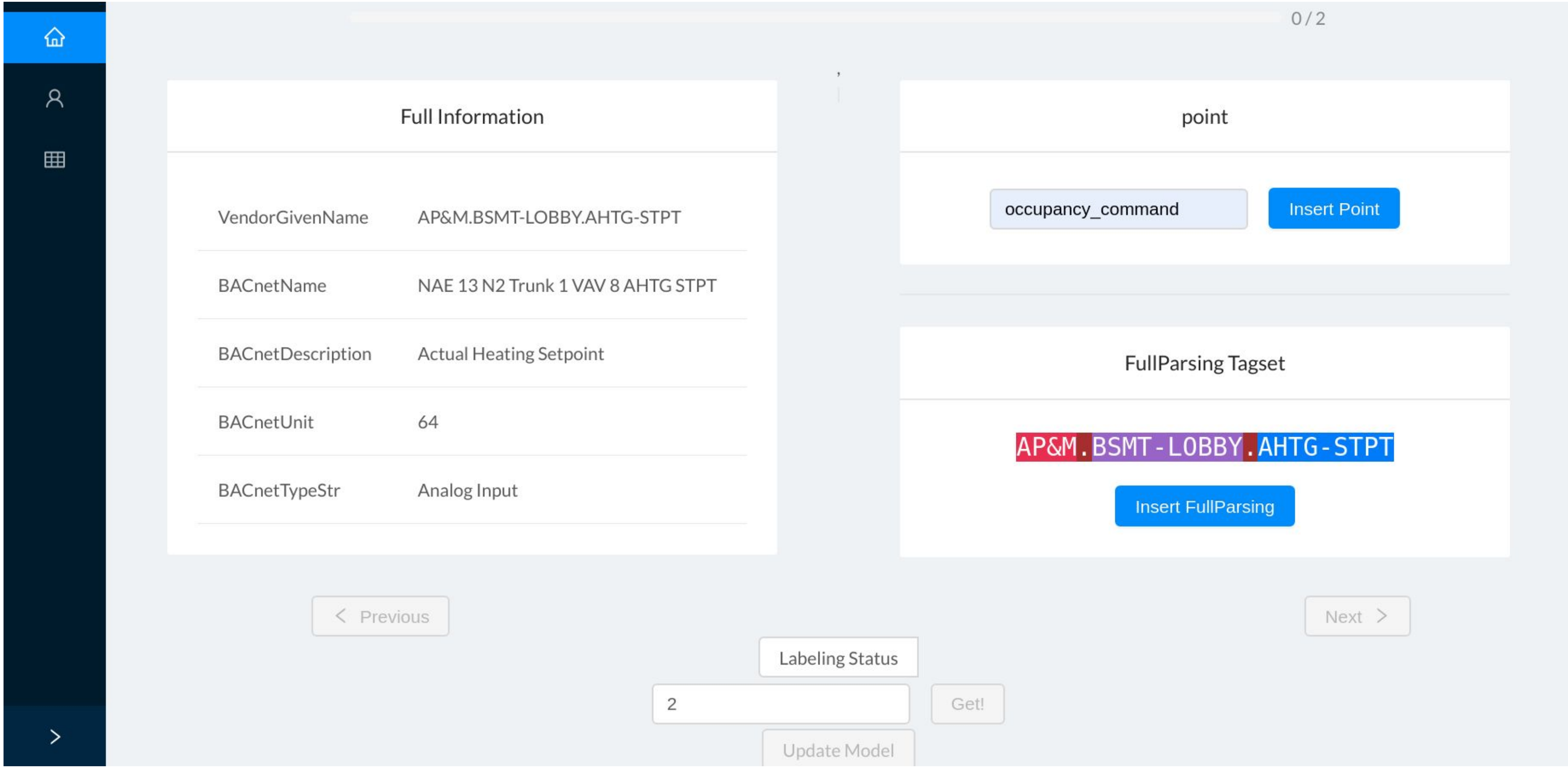
Goal: Brickify a target building with minimal human efforts

Plaster: Glueing Different Methods



[1] Koh et al. BuildSys2018
[2] Hong et al. BuildSys2015
[3] Balaji et al. BuildSys2015
[4] Bhattacharya et al. BuildSys2015

Plaster Web Service



Brick in Action

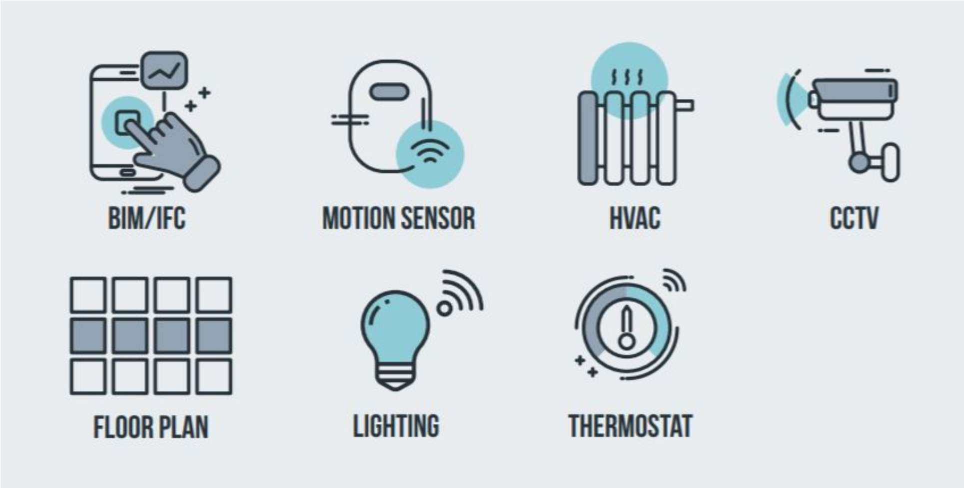
Johnson Controls uses Brick to address Smart Building use cases

Presented by:

Yiyi Guo

Product Manager, Johnson Controls

Brick Powered Common Data Model



- Prior to adopting Brick
- With Brick

Extension to Brick

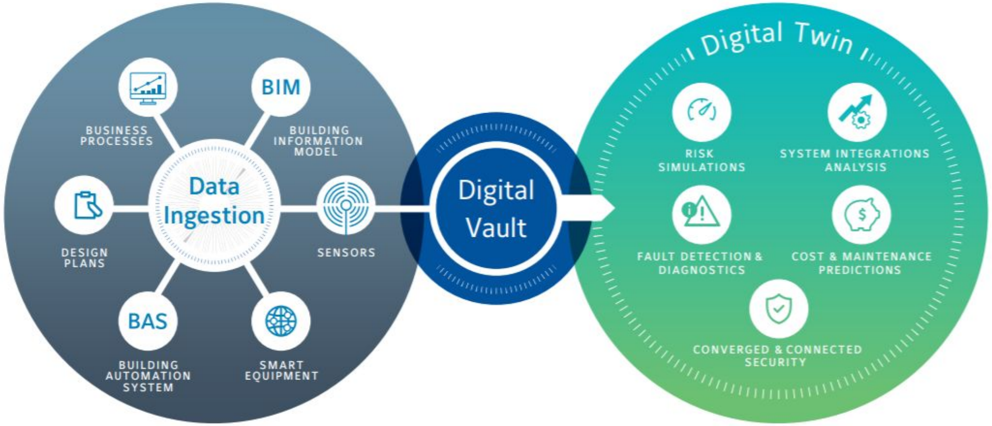
- JCI is applying Brick on other domains: Security and Access Control

- Brick is open for consumers to extend and include new concepts to Brick Schema

Analytics Use Cases for Smart Buildings

- JCI is using data modeled in Brick to power analytics use cases such as
  - Fault detection
  - Energy optimization
  - Risk analytics

Johnson Controls Digital Vault



- Bringing in sensor data through BACnet, OPC, and other protocols
- Ingesting location and equipment data from BIM, BMS systems, Access Control systems, etc.
- Storing and modeling building data with Brick schema

Connect with us

- Johnson Controls Digital Vault

https://www.johnsoncontrols.com/digital-solutions/digital-vault

- Johnson Controls Smart Buildings

https://www.johnsoncontrols.com/digital-solutions/smart-buildings

- Johnson Controls and Bee'ah - Office of the Future

https://www.johnsoncontrols.com/media-center/news/press-releases/2019/04/23/middle-east-sustainability-pioneer-bee-ah-selects-johnson-controls-microsoft-for-office-of-future

Brick

Johnson Controls

Brick Vision

1. Brick as a *Lingua Franca* for building lifecycle
    a. Accommodate other models; Haystack, IFC, BTO, etc.
2. Community-Driven
    a. Vocabulary Extension
    b. Tool Development
3. End-to-End Workflow
    a. device dev, building commissioning, system integration, app dev, and app deployment

Visit us: https://brickschema.org