Programming Smart Buildings with Knowledge Graphs

Playground Tutorial

SenSys 2025

Dr. Gabe Fierro
https://gtf.fyi
Assistant Professor, Computer Science
Joint Appointment @ NREL

Outline

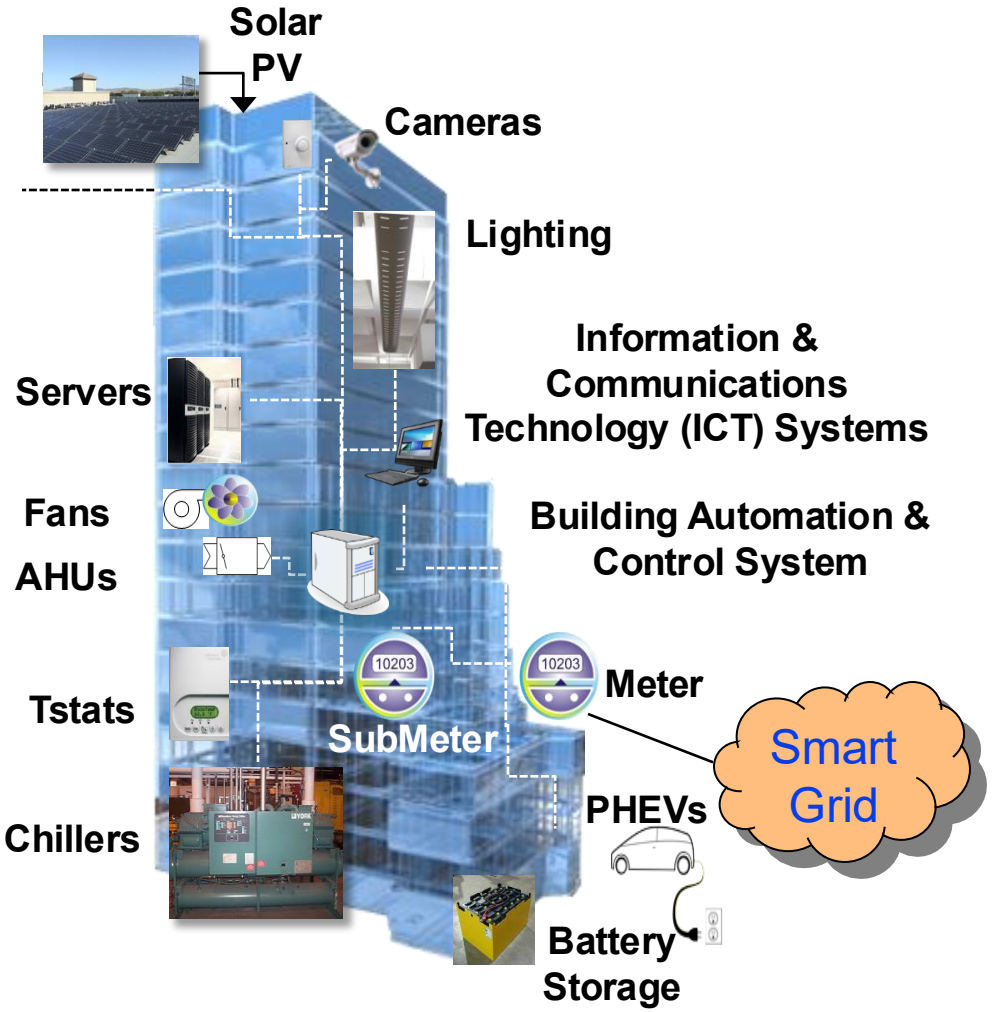
- Why we want to program buildings
- Why is programming buildings difficult?
- Addressing data discovery with knowledge graphs
- Programming smart buildings with knowledge graphs
- Specification-driven development

Setting the Stage

- Modern buildings have large digital control/sensing surfaces

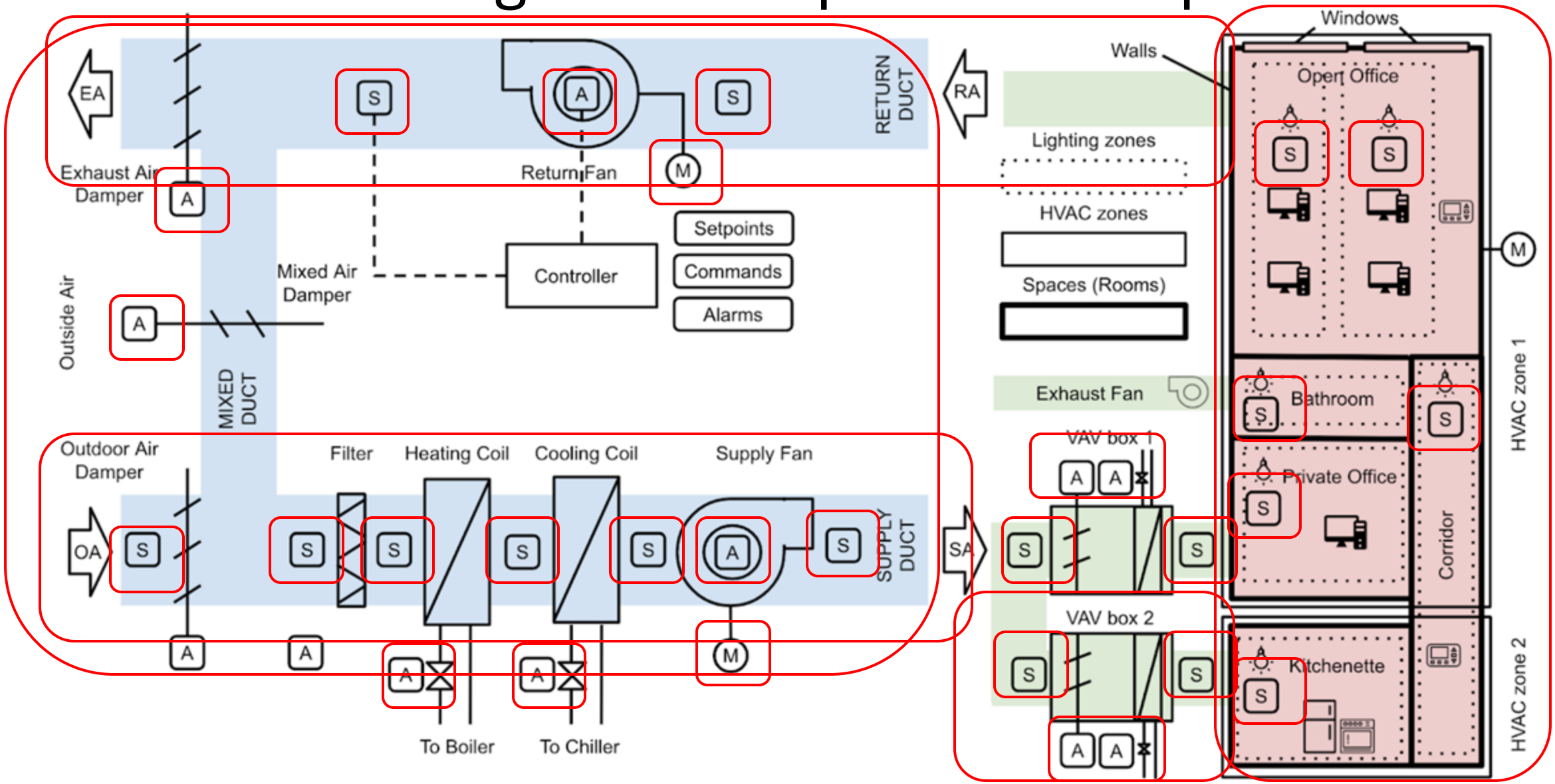
- Building management is increasingly complex and software driven
  - Renewables → bi-directional power flow
  - Dynamic energy/power tariffs
  - Personalized comfort settings
  - Advanced controls and fault detection

- **Adoption of "smart" software lags adoption of the hardware** (CBECS 2018)



Why is it difficult to build smart building software?

Reason 1: Buildings are Complex and Bespoke CPS

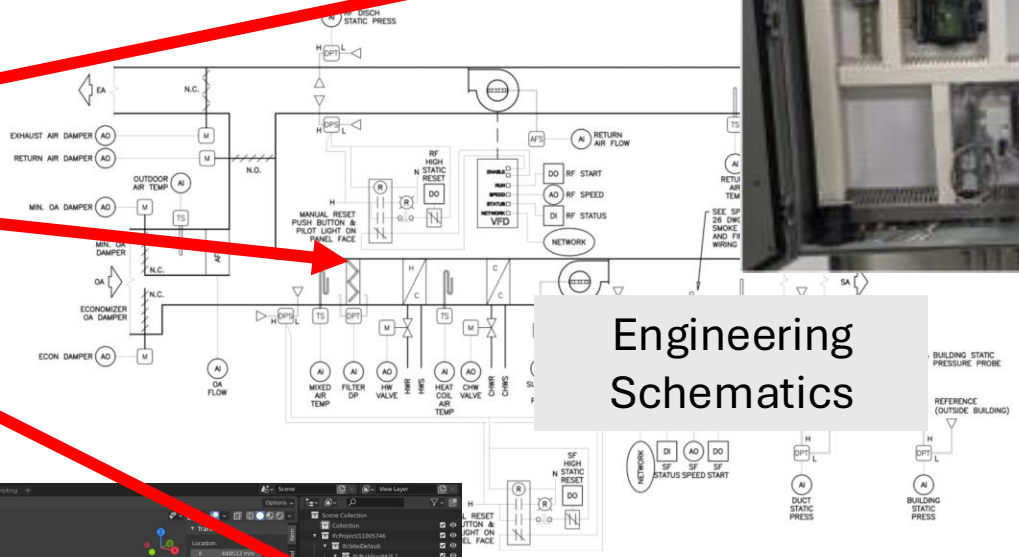


Not just home automation: simple HVAC system for a commercial office building

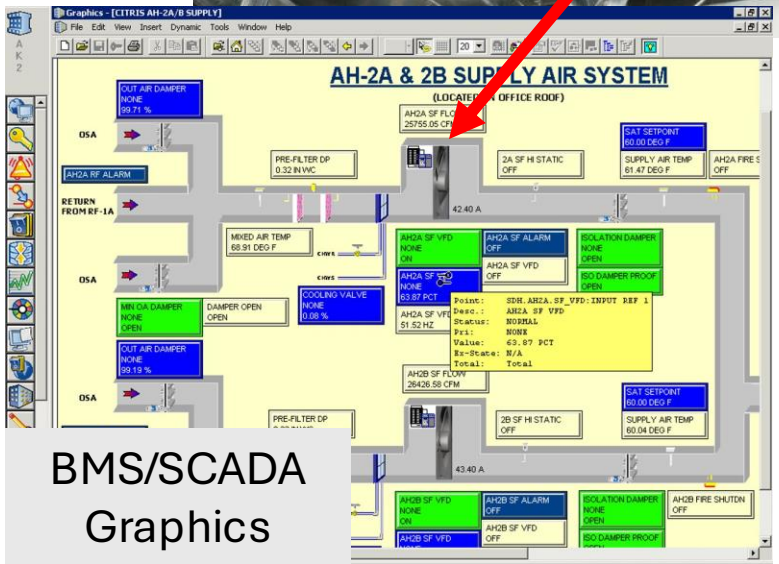
Reason 2: Disconnected Digital Models

Field Controllers / PLC

Physical system

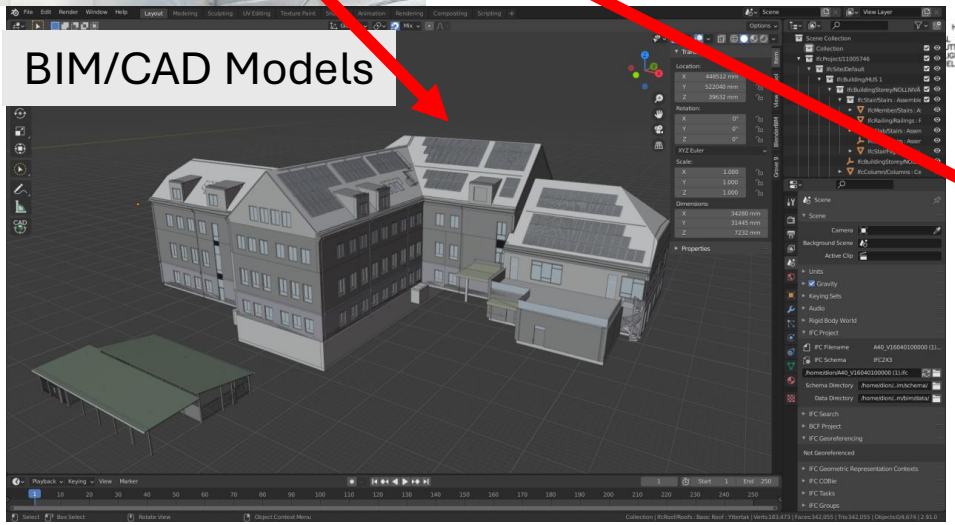


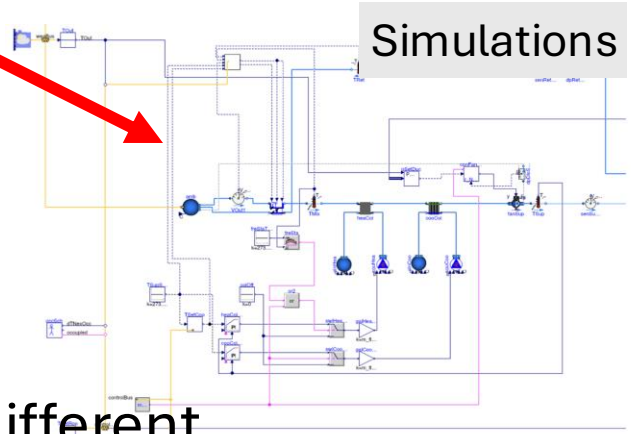
Engineering Schematics



BMS/SCADA Graphics

BIM/CAD Models

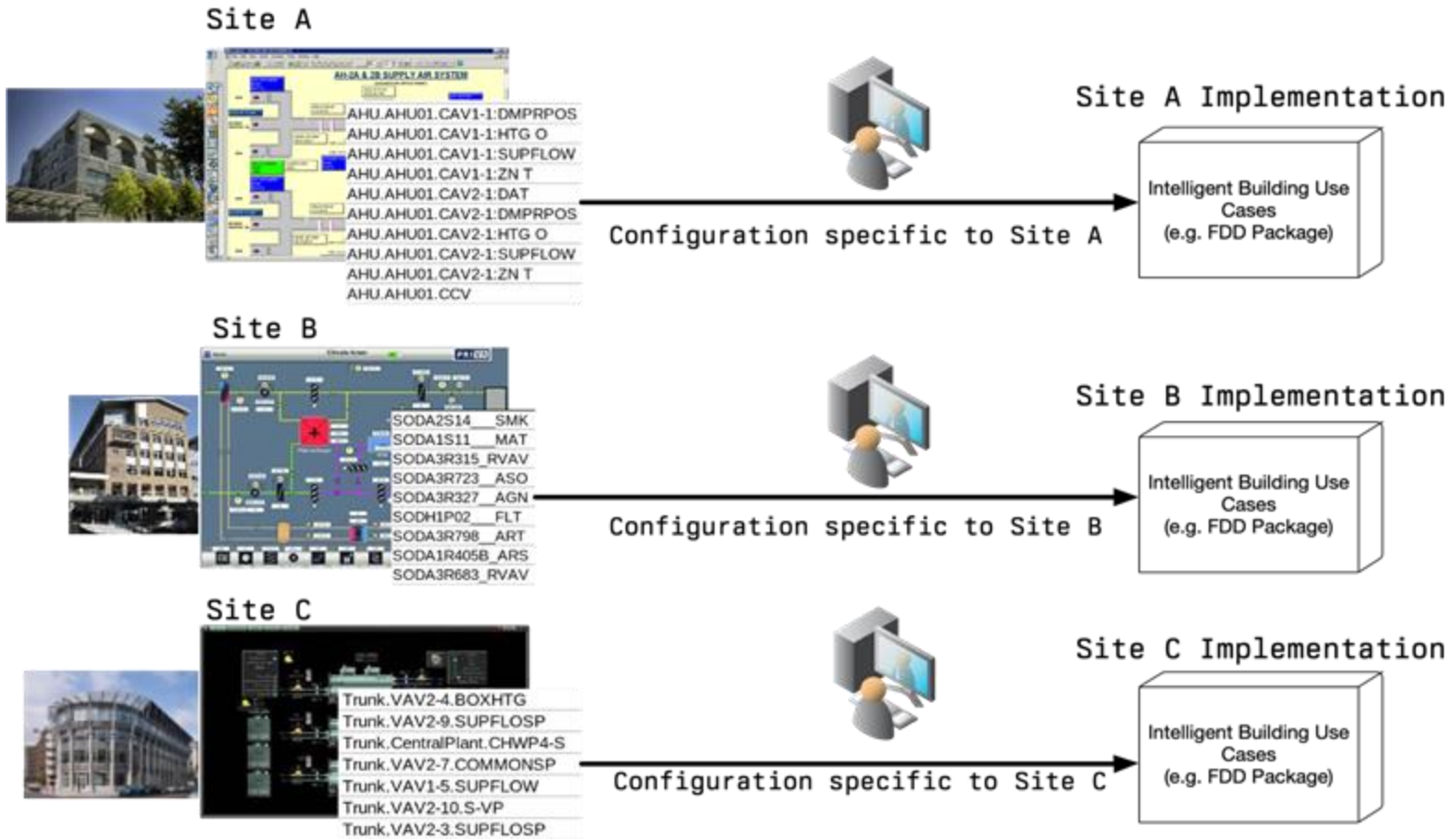


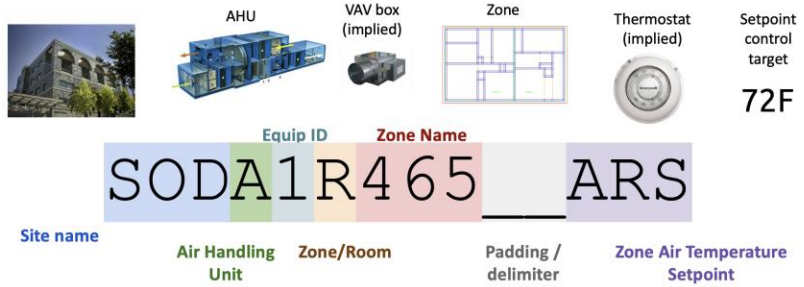


Simulations

Finding data often requires cross-referencing between different representations of the same system

Reason 3: Each Building is Unique

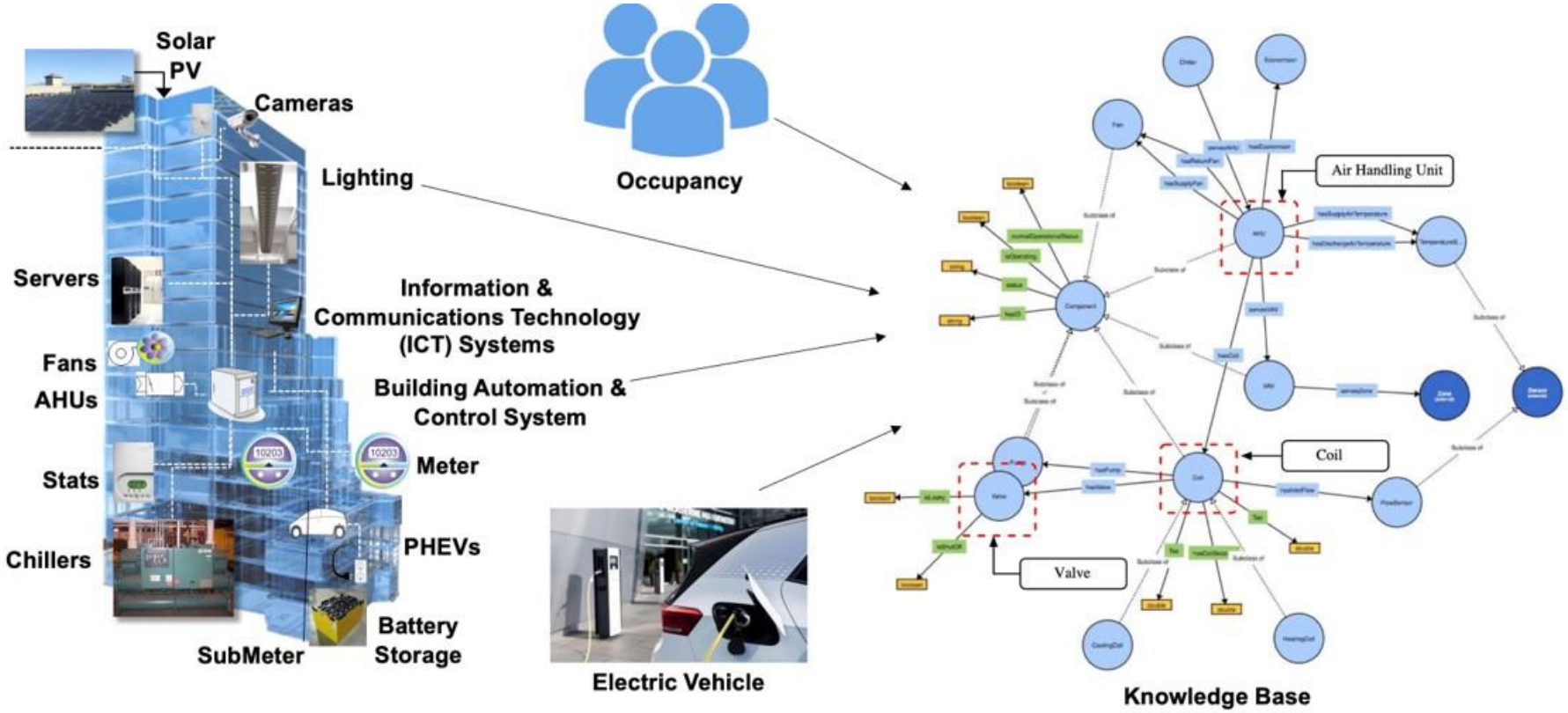




Naming conventions and protocol "soup"

KNX LonWorks ASHRAE BACnet X10 Modbus enocean matter zigbee

- Lack of standardization, interoperability increases soft costs associated with <u>developing</u> and <u>deploying</u> data-driven solutions



- Controls
- Energy Auditing
- Fault Detection and Diagnostics
- Commissioning
- Smart Grid Interactions
- Dashboards
- Digital Twins
- Sensor Calibration

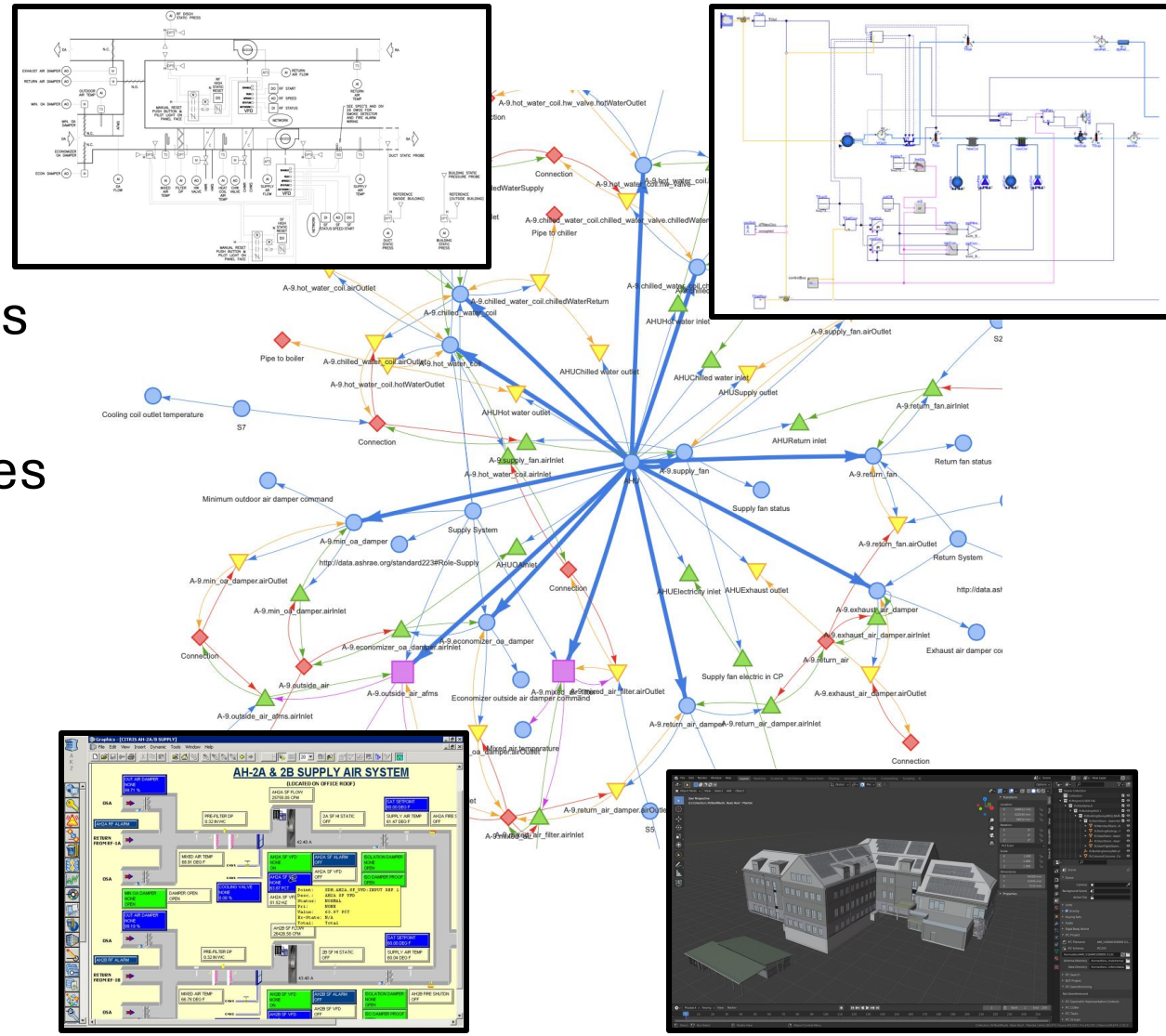
Physical, logical systems ← represents — **Knowledge Graph** — access → **Applications**

Two parts to solution:

- **Knowledge graphs**: a shared representation of devices, data sources, and environment
- **Portable programming models**: new ways of expressing data-driven logic using KGs

Cyber-Physical Knowledge Graphs for Buildings

- **Knowledge graphs** store building metadata
  - Assets and their connections
  - Sensors, setpoints, actuators, alarms
  - Links to SCADA and data stores
  - Nameplate and other static properties
- Digital representation of what's in other digital models and diagrams
- Software queries the knowledge graph for configuration/discovery

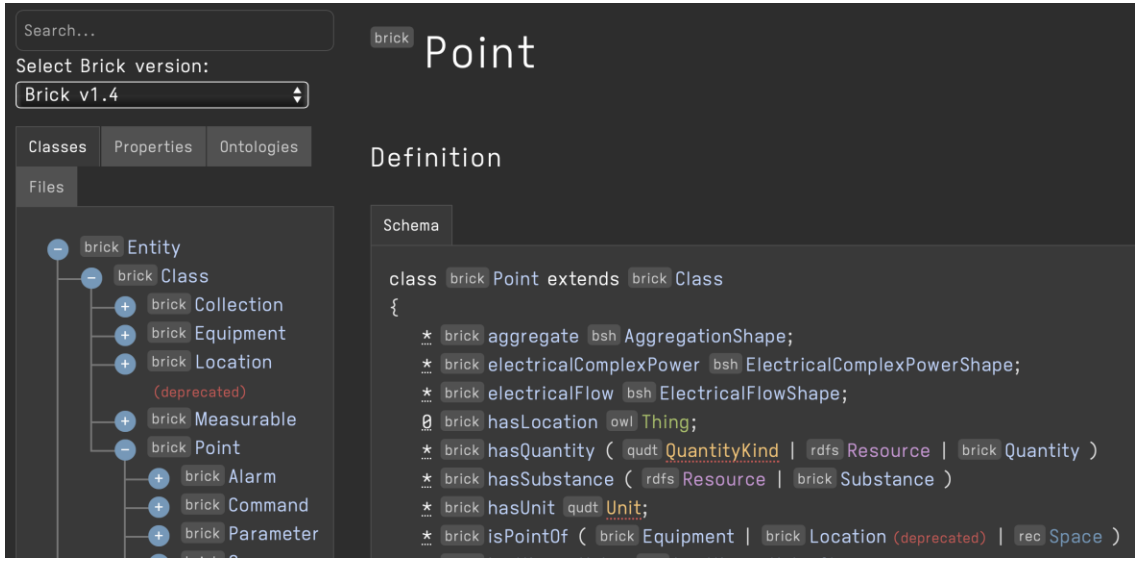


The Brick Ontology

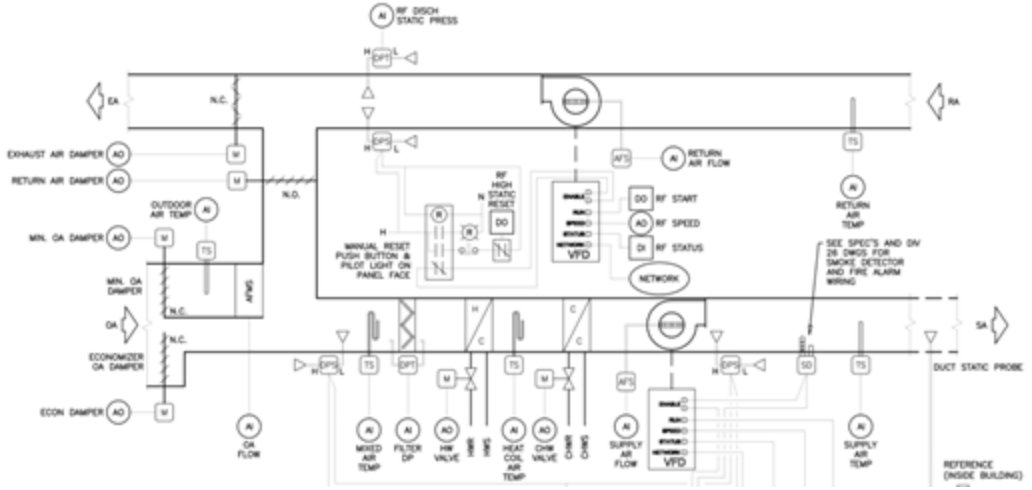
- Created in 2016 by researchers in the BuildSys community
- Standardizes names, definitions of common building assets, architectural elements, data sources
- Now a 501(c)(6) with 7 industrial partners

https://brickschema.org

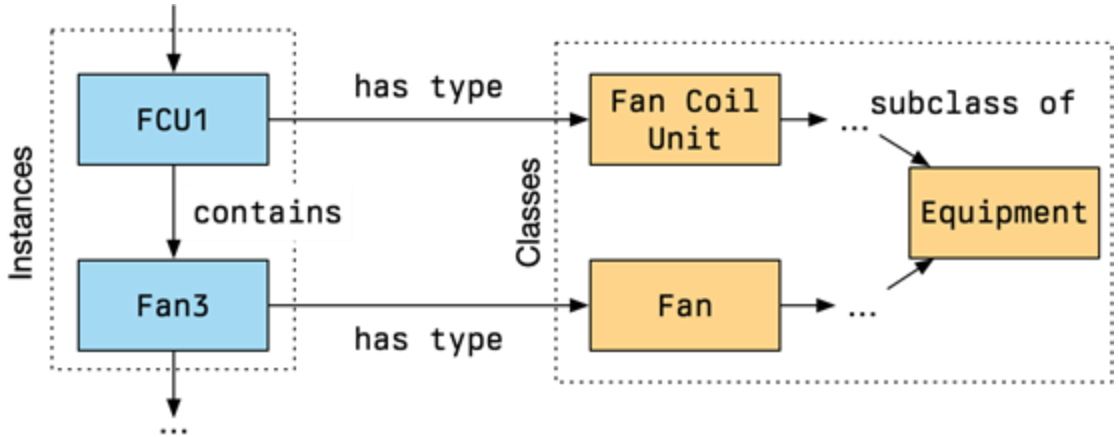
Original Founding Members



Building on Open Standards for Semantic Metadata Graphs



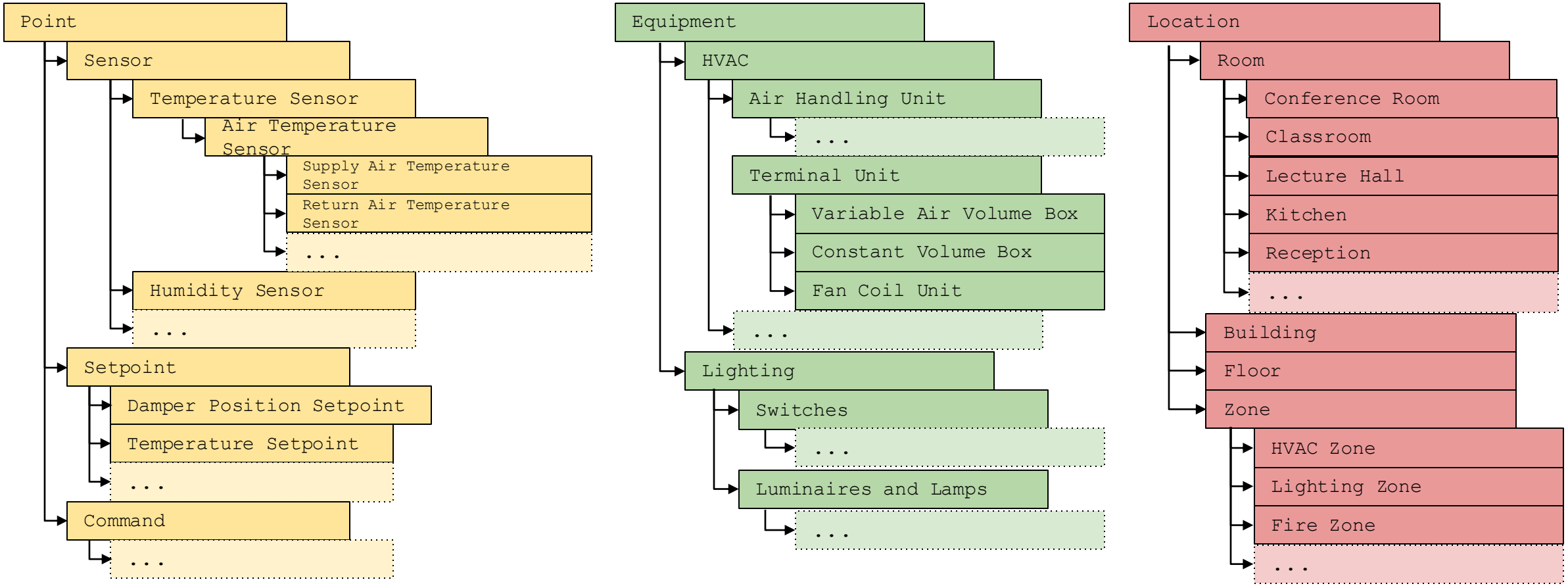
Mechanical Diagrams: human-readable and non-standard



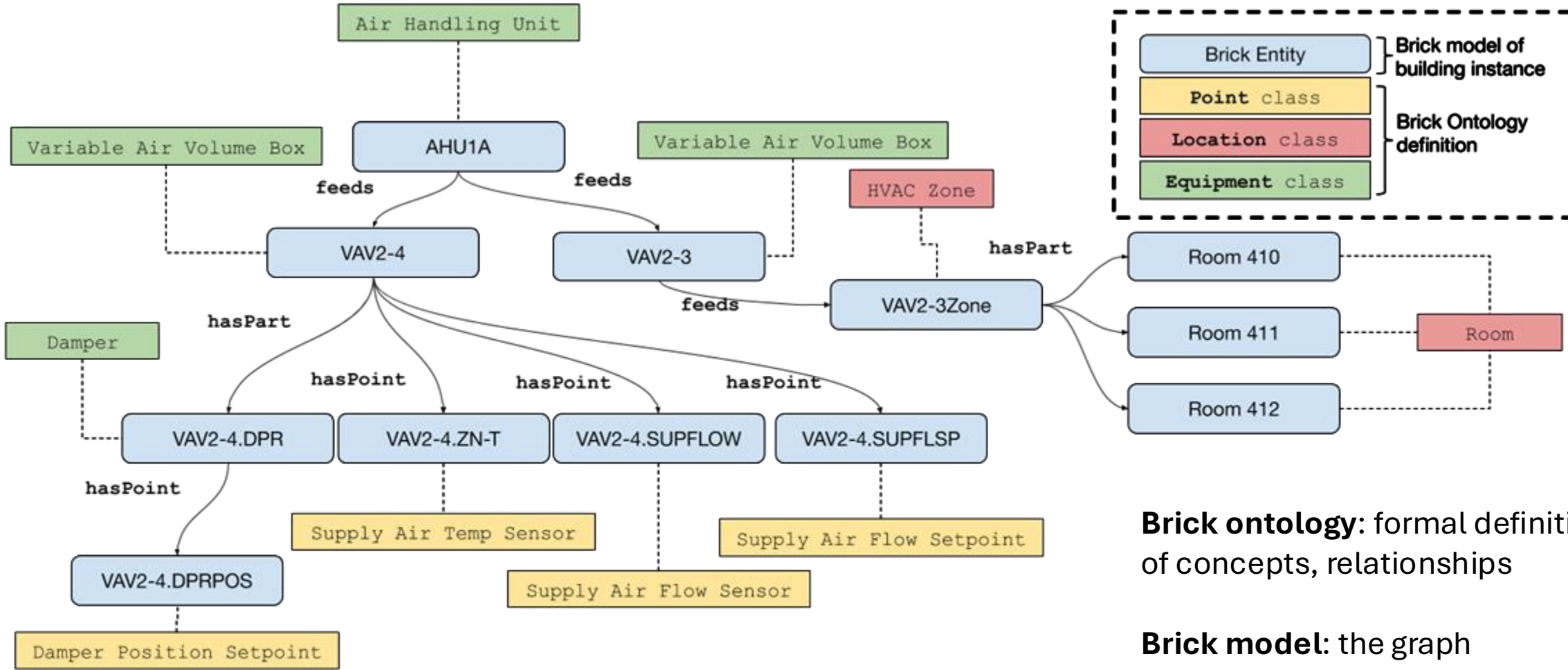
RDF Graphs: standard machine/human-readable models

- Build on **R**esource **D**escription **F**ramework (**RDF**) W3C standard for directed, labeled graphs
  - Tap into existing open-source and commercial ecosystems of tools
  - Supports sophisticated search and discovery
- **SPARQL**: Standard graph query language
  - Retrieve information from graphs
- **SHACL**: Constraint language for graphs
  - Enable automated validation of models
  - Acts as a “schema” for graphs

What's in the Ontology?



- Class hierarchy covering over 800 major equipment types, architectural elements
- **Point** classes cover different types of I/O data
  - Sensors (r), setpoints (r/w), commands (r/w), alarms (r), statuses (r)…



Brick ontology: formal definitions of concepts, relationships

Brick model: the graph representing a particular building

- Relate Brick Point instances to timeseries data
- We can now use knowledge graphs for configuring our software

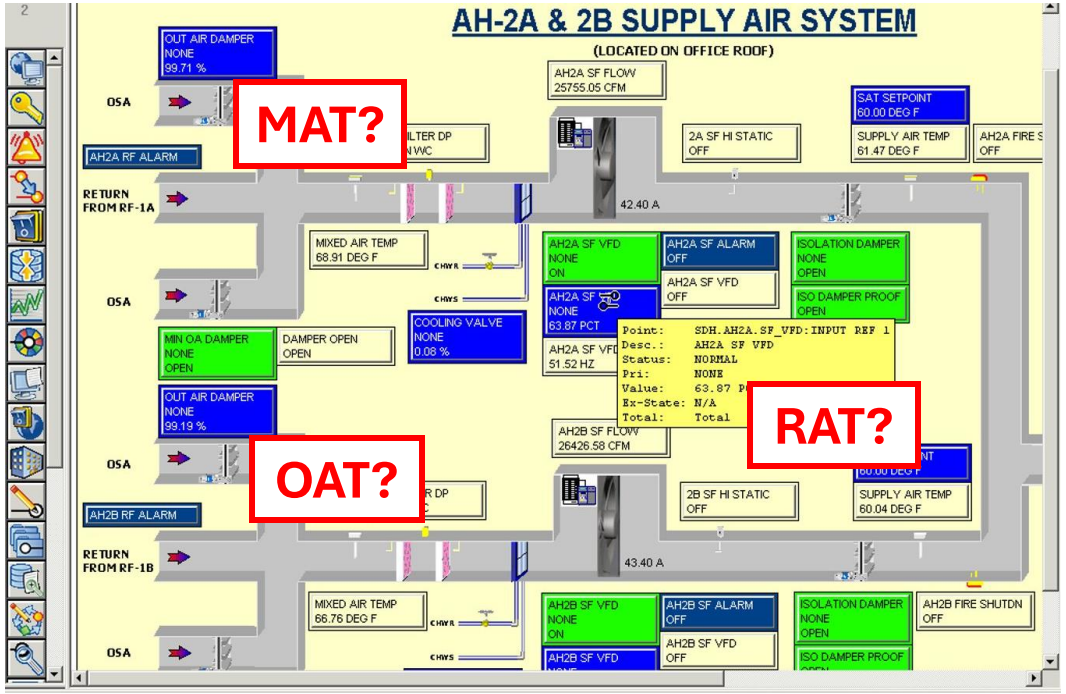
Portable Cyber-Physical Programming

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

MAT?

RAT?

OAT?



Example Fault Detection Rule:

- Runs on a particular type of asset
- Needs specific sensors
- **Sensors and data sources may have different names, or not be installed**
- Most rule implementations <u>hard code</u> these names

AHU.AHU01.CAV1-1:DMPRPOS	SODA2S14___SMK
AHU.AHU01.CAV1-1:HTG O	SODA1S11___MAT
AHU.AHU01.CAV1-1:SUPFLOW	SODA3R315_RVAV
AHU.AHU01.CAV1-1:ZN T	SODA3R723__ASO
AHU.AHU01.CAV2-1:DAT	SODA3R327__AGN
AHU.AHU01.CAV2-1:DMPRPOS	SODH1P02___FLT
AHU.AHU01.CAV2-1:HTG O	SODA3R798__ART
AHU.AHU01.CAV2-1:SUPFLOW	SODA1R405B_ARS
AHU.AHU01.CAV2-1:ZN T	SODA3R683_RVAV
AHU.AHU01.CCV	SODA1R405B_ART
AHU.AHU01.CHWHHW.UNT:CHW FLOW	SODA3R311__AGN
AHU.AHU01.CHWHHW.UNT:HW FLOW	SODH1_____L_L
AHU.AHU01.Cooling Enable	SODC1SP03__FLT
AHU.AHU01.ECM	SODA4R645_RVAV
AHU.AHU01.HP.UNT:ZN T	SODA1R288__AGN
AHU.AHU01.HSP	SODA3R419__AGN
AHU.AHU01.LSP	SODA3C611__ASO
AHU.AHU01.LTD	SODA2S14_P__VR
AHU.AHU01.MAX.ZONE.DAMPER	SODA4S1832_STA
AHU.AHU01.MAX.ZONE.HEATING	
AHU.AHU01.MIN OA	
AHU.AHU01.Mixed Air Damper Position	
AHU.AHU01.Mixed Air Temp	

Which variables correspond to

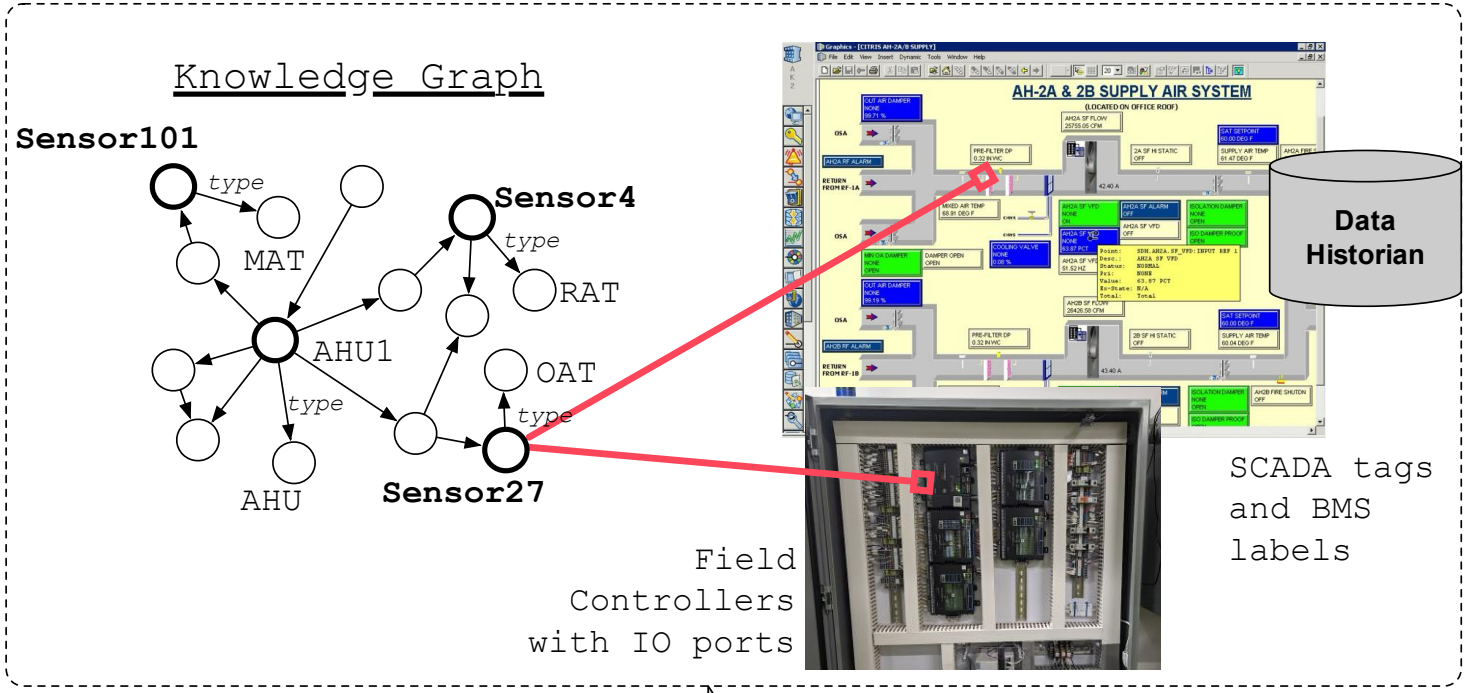
MAT?

RAT?

OAT?

Hundreds of non-standard variable names…

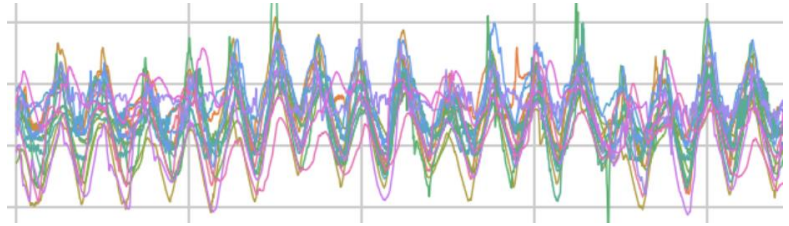
Portable Cyber-Physical Programming



AHU	Air Handling Unit (Equipment)
MAT	Mixed Air Temperature (Sensor)
RAT	Return Air Temperature (Sensor)
OAT	Outside Air Temperature (Sensor)

When `MAT` < `min(RAT, OAT)`
on an `AHU` then raise **error**

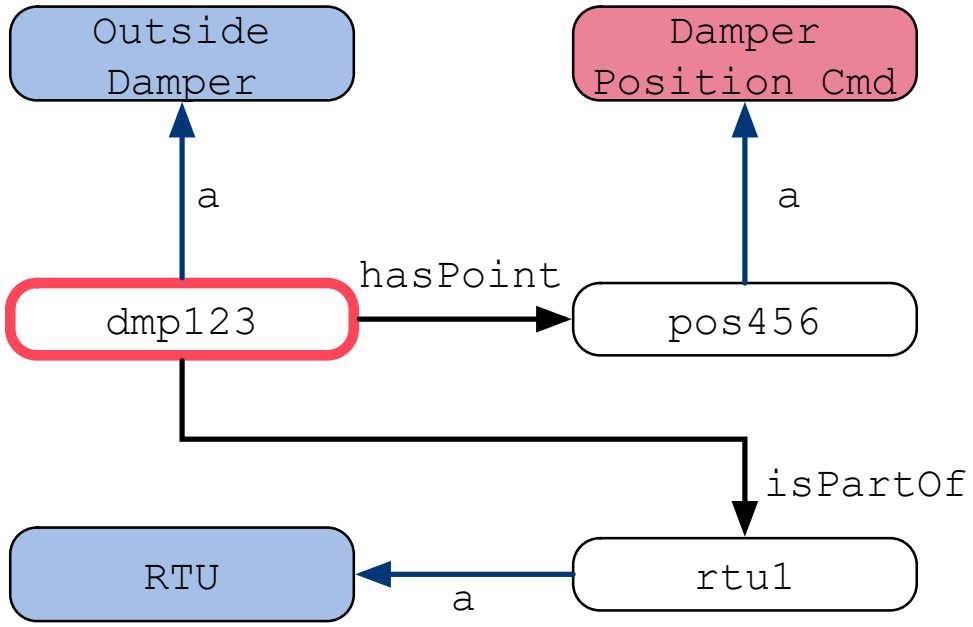
Graph-driven Specialization



When `Sensor101` < `min(Sensor4, Sensor27)`
on `AHU1` then raise **error**

- Solution: don't use variable names!
- Replace variables with **graph queries**
- System runs the queries to bind variables to the available data sources

Graph Queries on Knowledge Graphs

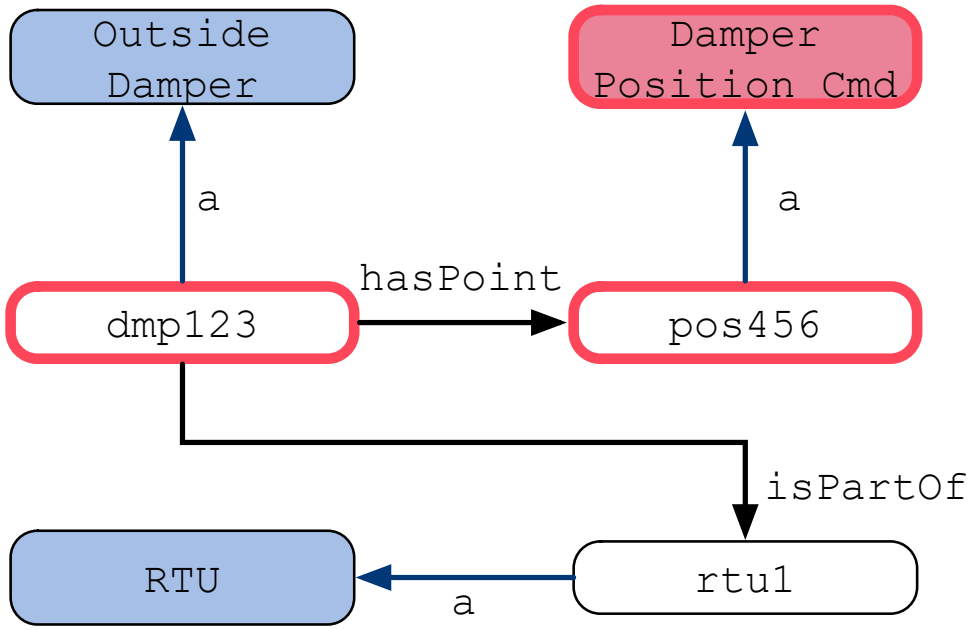



```
SELECT * WHERE {
  ?x a brick:Outside_Damper .
}
```

- Pattern matching on *node-edge-node* **triples**
- ?variables bind to the missing parts of the pattern
- Share variables to JOIN different patterns

- Common patterns:
  - **Find instances of a type** (or supertype)
  - Relate instances of different types together

Graph Queries on Knowledge Graphs




```
SELECT * WHERE {
  ?x a brick:Outside_Damper .
  ?x brick:hasPoint ?p .
  ?p a ?point_class
}
```

- Pattern matching on *node-edge-node* **triples**
- ?variables bind to the missing parts of the pattern
- Share variables to JOIN different patterns

- Common patterns:
  - Find instances of a type (or supertype)
  - Relate instances of different types together

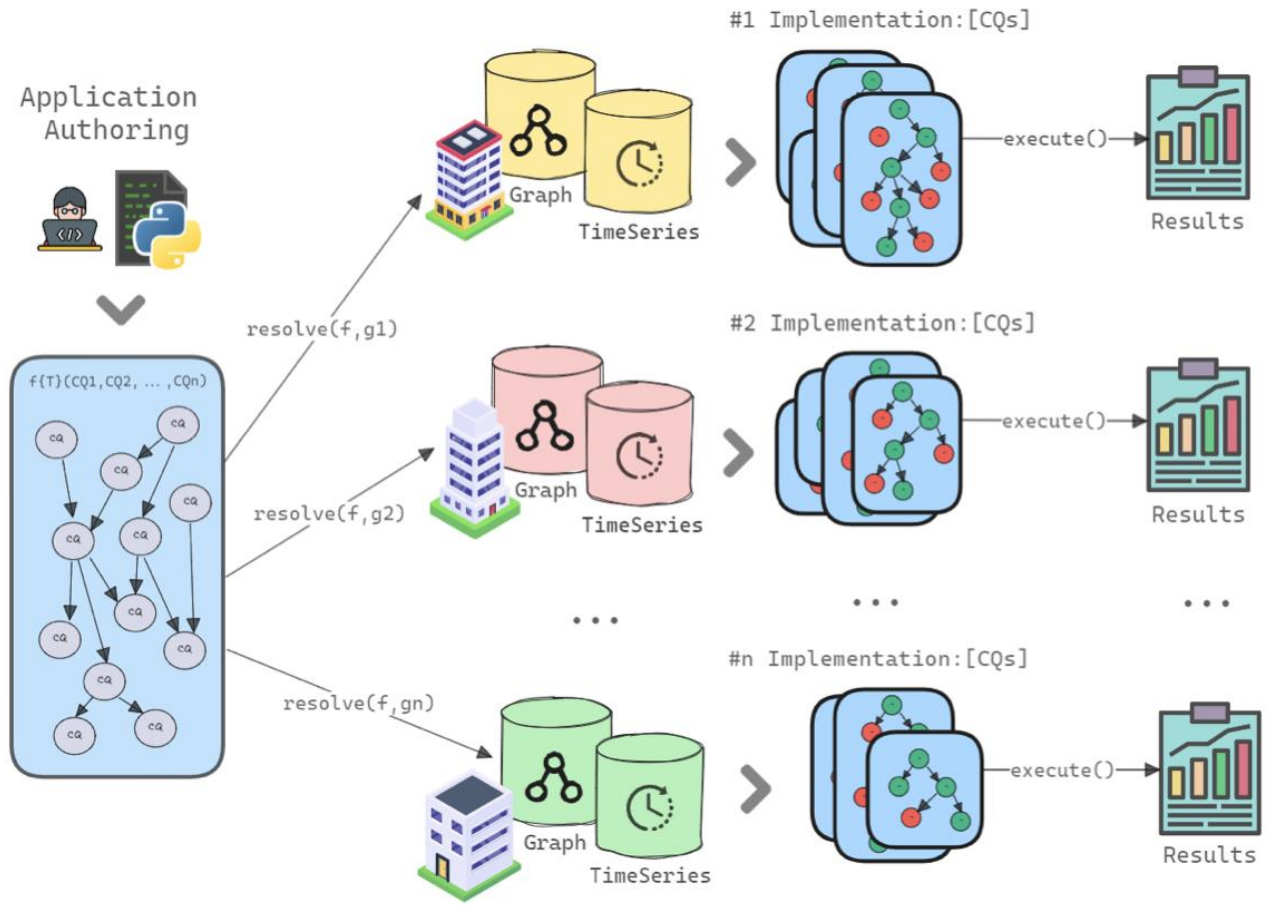
SeeQ: New Programming Model for CPS Code

- Write Python applications against concepts defined by knowledge graph
- SeeQ “compiles” the Python code against the metadata model for each building
  - Generates site-specific implementation

```python
from SeeQ import *
from pandas import DataFrame
from G36.CQs import Dmp_Pos, Fsa, Fsp_clg, Fan_s
from APAR.CQs import Tsa, Tma, DelTsf, Hc_pos, Epsilon_t

def APAR_R1(sup: Tsa, mix: Tma, drop: DelTsf, heat_coil: Hc_pos, e: Epsilon_t):
    is_heating: DataFrame = heating_coil.df > 0
    supply_air_low: DataFrame = sup.df < (mix.df + drop.df - error.df)
    violating_records = is_heating & supply_air_low
    # returns fault if more than 10 violating samples
    if len(violating_records) > 10:
        return "fault detected"

def G36_Dmp_Leaking(pos: Dmp_Pos, sup_flow: Fsa, cool_sp: Fsp_clg, fan: Fan_s):
    if ((pos.df == 0) and (sup_flow.df > max([0.1*cool_sp.df, 50]) \
    and (fan.df == "ON")).for_time(600):
        return "Level 4 alarm"
```

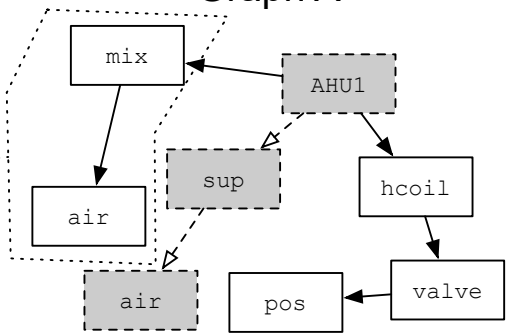


Portable Programming Model

Original Rule

```python
def rule1(ahu) -> bool:
    is_heating = ahu.hcoil.valve.pos > 0
    if not is_heating: return False
    sup_air_temp = ahu.sup_air.temp
    mix_air_temp = ahu.mix_air.temp
    fan_temp_drop = 1.1 # degC, approx
    eps = 0.1
    return sup_air_temp < mix_air_temp +
           fan_temp_drop - eps
```

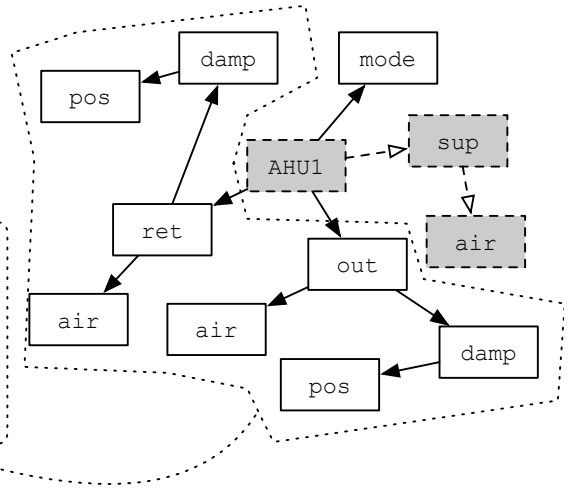
Graph A



Specialized Implementation

```python
def rule1(ahu) -> bool:
    # !! Different detection of heating mode
    is_heating = ahu.hvac_mode == 'heating'
    if not is_heating: return False
    sup_air_temp = ahu.sup_air.temp
    # approximate mixed air with available sensors
    ret_damper_pos = ahu.ret_damper.pos
    out_damper_pos = ahu.out_damper.pos
    ret_air_temp = ahu.ret_air.temp
    out_air_temp = ahu.out_air.temp
    mix_air_temp = ret_air_temp * ret_damper_pos +
                   out_air_temp * out_damper_pos
    fan_temp_drop = 1.1 # degC, approximated
    eps = 0.1
    return sup_air_temp < mix_air_temp +
           fan_temp_drop - eps
```

Graph B



- Compiled Python code detects different subgraphs
- Adjusts data discovery and application logic automatically
- Generate **site-specific** implementations with zero human configuration

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

Encoding Application Requirements

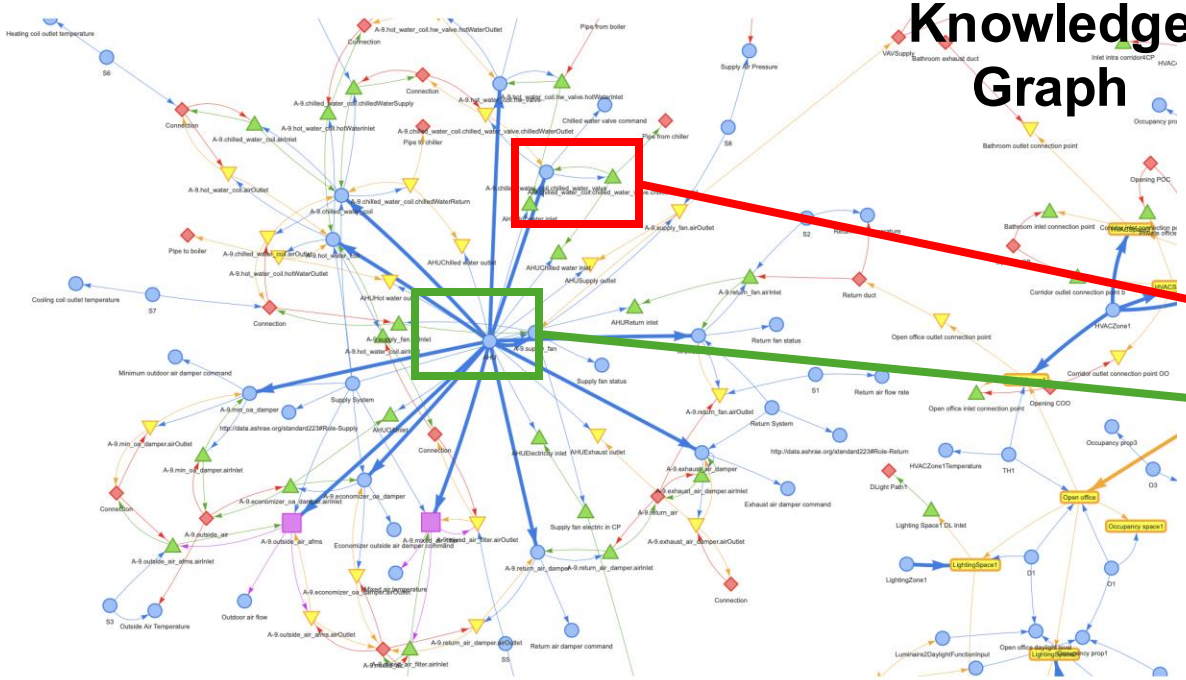
4.2 VAV Terminal Unit with Reheat

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
- Required I/O points for a high-performance control sequence
- How to ensure Brick model contains the right metadata?

- **Validate** the knowledge graph against application requirements
- Report lists which parts of the building support which applications

Building Knowledge Graph



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

Application Data Requirements

SHACL Validation Process

Shapes validate KG content

FCU503	Passes Validation
FCU510	**Fails Validation:** - missing Room Air Temp Sensor - missing Heating Coil - missing Filter

Validation Report

- Use validation results to fix the knowledge graph (support more apps!)

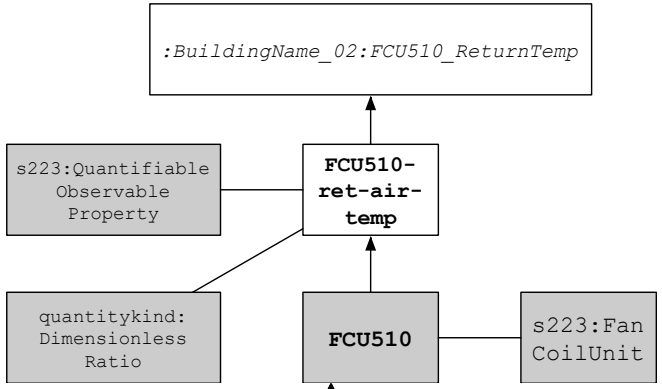
Using Validation to Fix a Knowledge Graph

From validation process

FCU503	Passes Validation
FCU510	**Fails Validation:** - missing Room Air Temp Sensor - missing Heating Coil - missing Filter

Validation Report

To validation process



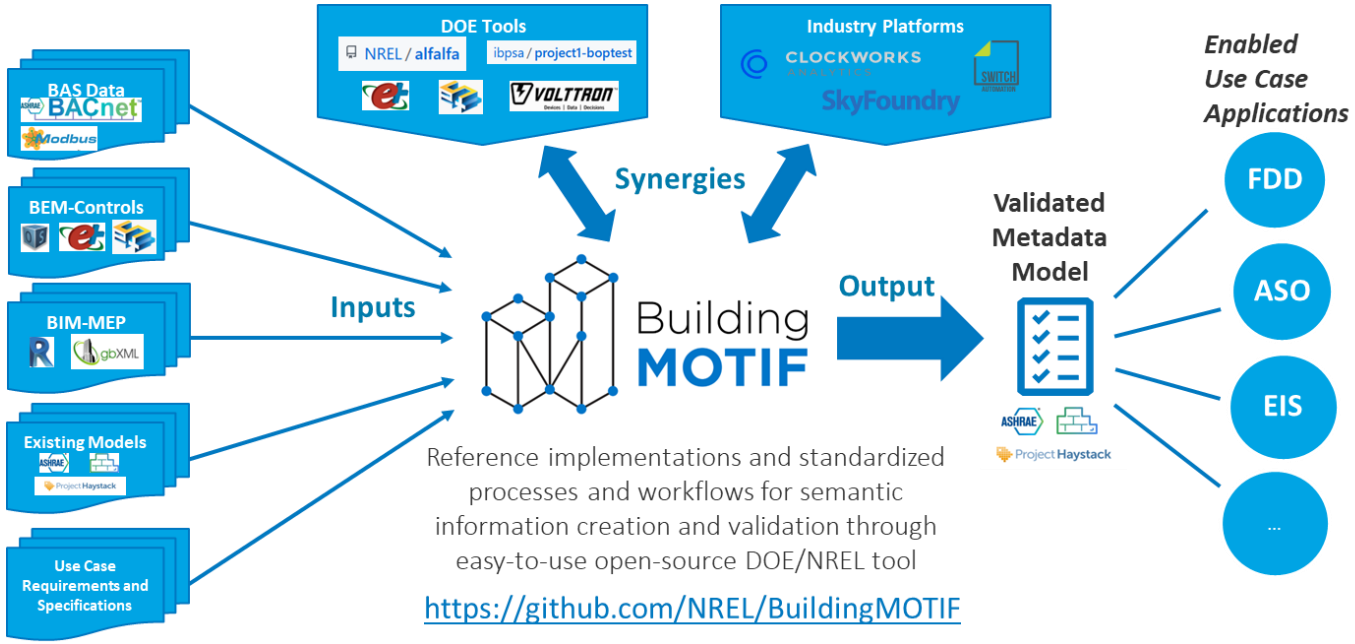
Model Patch

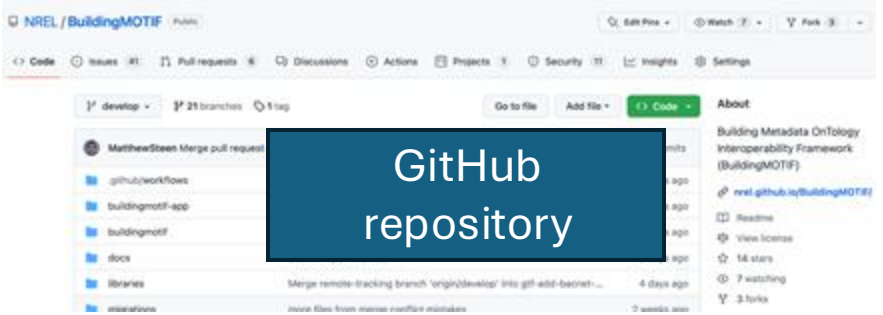
Possible fixes

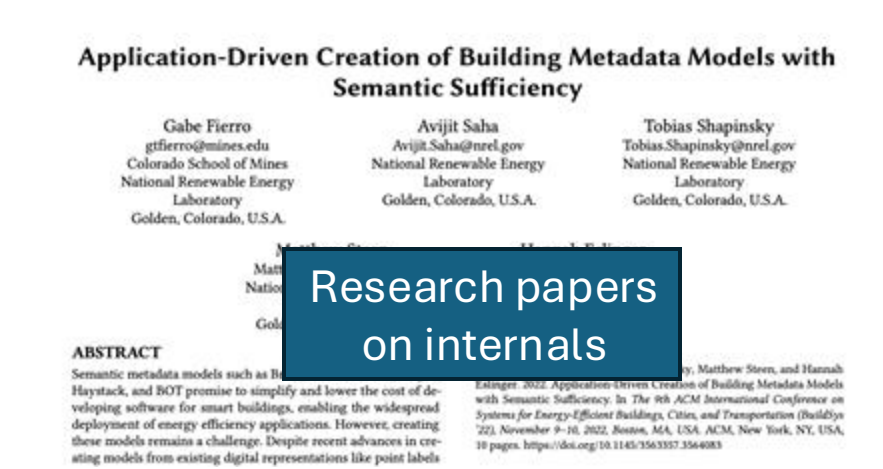
- Extend point label naming convention script
- Re-scan network for more points
- Extract metadata from BIM
- Prompt user for specific inputs

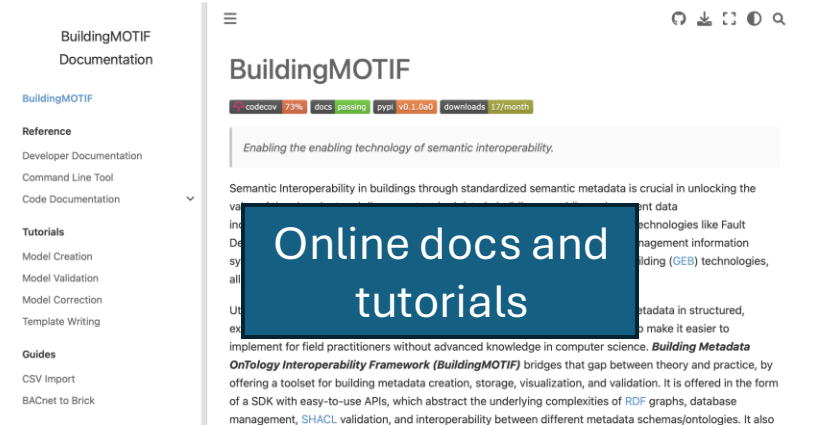
BuildingMOTIF SDK

- **KG construction, validation + App configuration**
  - Available on GitHub under NREL organization
  - Released under permissive BSD 3 Clause license
  - Tutorials, notebooks, demo web interface
  - Supports Brick and other ontologies
  - *Simplifies building metadata-driven software*









Thank you!

- My current research/projects: https://gtf.fyi
  - Contains links to all the works I've mentioned in this talk
  - Most have an open-source GitHub repository associated with them
- Brick ontology project: https://brickschema.org
- ASHRAE 223P development: https://open223.info
  - new smart building ontology!
- NREL-developed semantic metadata platform: https://github.com/NREL/BuildingMOTIF