PhD Thesis Defence

A Metamodel for Mechanical, Electrical, and Plumbing Systems: Enabling Interoperability and Control Integration in Building Management

Peter Yefi

Supervisors
Dr. Yann-Gaël Guéhéneuc
Dr. Ursula Eicker

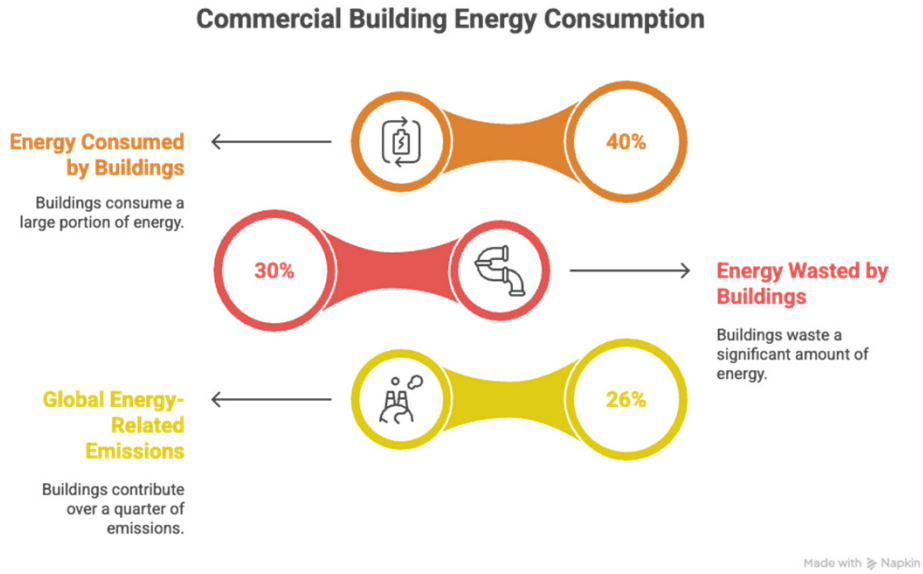
Date: December 9, 2025

Overview

- Context
- Thesis Statement
- Research Approach
- Object-oriented Modelling of MEP subsystems
- Interoperable Data Access of MEP Subsystems
- Integrated Control Logic in Object-oriented MEP Models
- Conclusion
- Future Work

Context

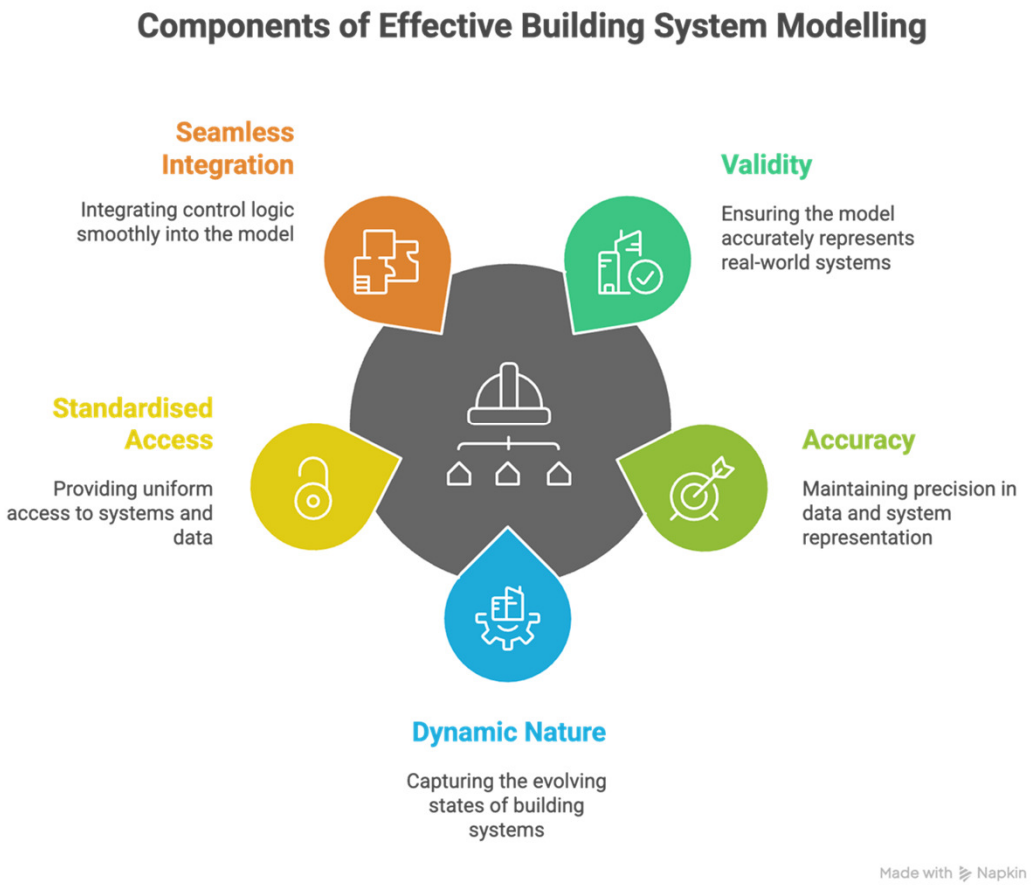
- Buildings are complex systems with multiple mechanical, electrical and plumbing (MEP) subsystems
- Building management systems (BMS) monitor and control MEP subsystems to
  - ensure optimal indoor environment conditions (IEC)
  - efficient resource utilisation



Context

- Proprietary BMS representations of MEP subsystems hinder solution transferability [6]
- Researchers have investigated ontology-based approaches to enable vendor-agnostic representations of MEP subsystems
- Ontologies excel at representing semantic domain knowledge, but they fall short in modelling the operational aspects of MEP subsystems [4]
- Commercial BMS remain largely closed, constraining the integration of advanced control and analytics capabilities [5]

Context



Achieve building energy efficiency through dynamic accurate modelling of building systems and seamless integration of advanced solutions

Thesis Statement

RQ1 & RQ2

RQ3

RQ4

Can an **object-oriented modelling approach represent mechanical, electrical, and plumbing (MEP) subsystems** in buildings, provide **consistent and standardised access to their data**, and enable the implementation and **deployment of control strategies** through predefined classes, validation mechanisms, and structured interfaces that facilitate integration with Building Management Systems (BMS)?

Research Categorisation

Object-oriented Modelling of MEP Subsystems

- **RQ1**: How does an object-oriented metamodel enable the creation of accurate, inherently validated models compared to existing ontology-based approaches?
- **RQ2**: How can an object-oriented metamodel represent mechanical, electrical, and plumbing (MEP) systems and their complex interactions?

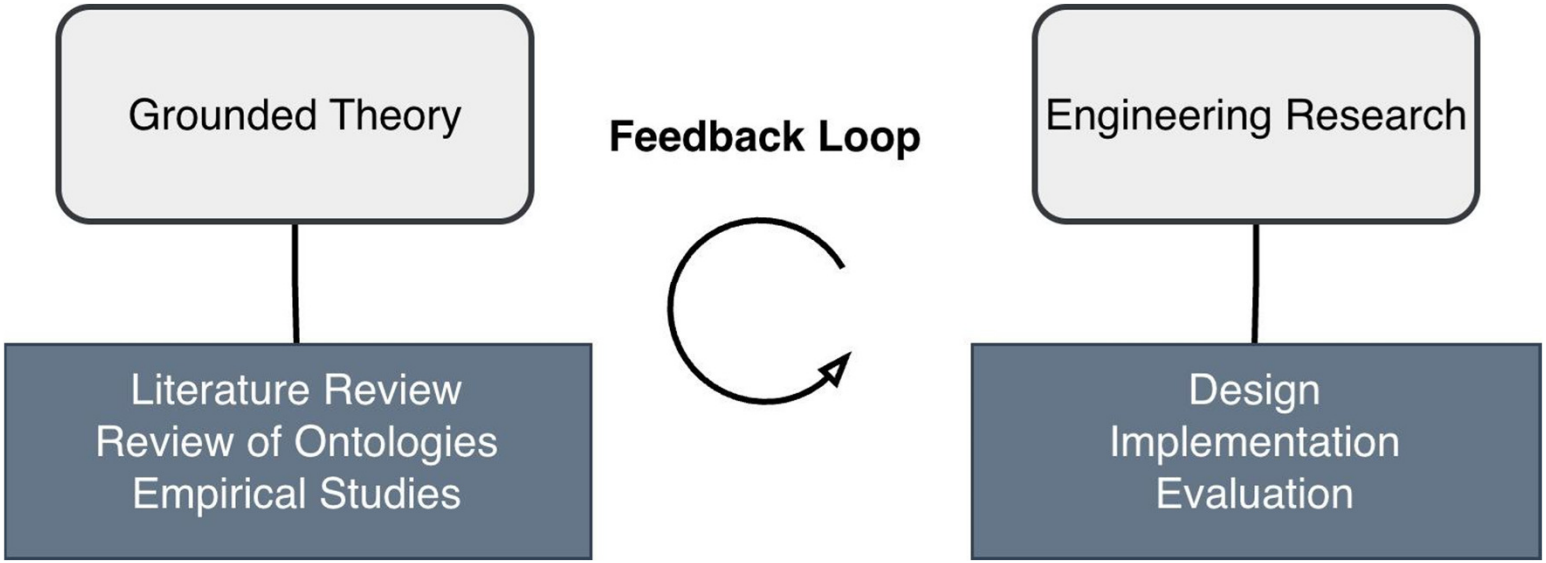
Interoperable Data Access of MEP Subsystems

- **RQ3**: How can practitioners and researchers access interoperable data of mechanical, electrical, and plumbing (MEP) systems in buildings?

Integrating Control Logic in Object-oriented MEP Models

- **RQ4**: How does an object-oriented modelling approach support the integration of control logic and promote broader adoption in practical energy-efficient applications?

Approach


Grounded Theory
Feedback Loop
Engineering Research
Literature Review
Review of Ontologies
Empirical Studies
Design
Implementation
Evaluation

RQ1 & 2: Object-Oriented Modelling of MEP Systems

Object-Oriented Modelling of MEP Systems

RQ1: How does an object-oriented metamodel enable the creation of accurate, inherently validated models compared to existing ontology-based approaches?

- RQ1.1: What are the limitations of current built environment modelling approaches, particularly ontology-based systems?
- RQ1.2: How can an object-oriented modelling paradigm address these limitations and offer practical advantages for representing and validating MEP systems?

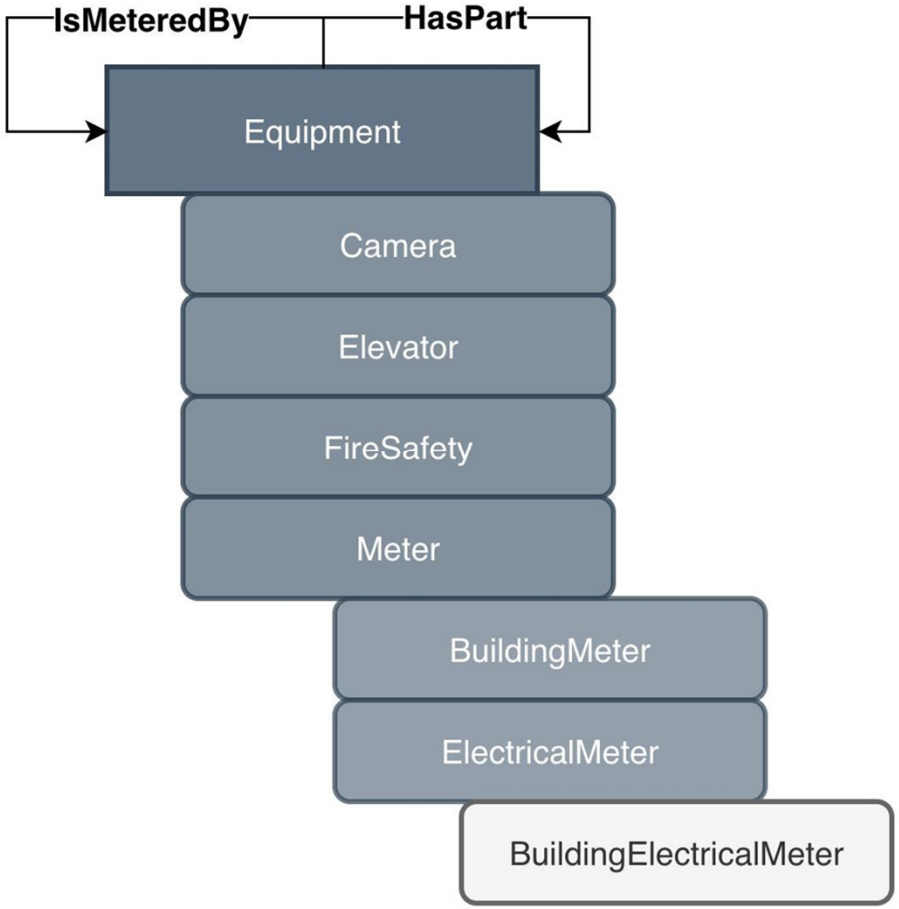
RQ2: How can an object-oriented metamodel represent mechanical, electrical, and plumbing (MEP) systems and their complex interactions?

RQ1.1: Limitation of Existing Approaches (L1 - L4)

- L1: Possible Ambiguity in Defining an Entity (e.g., boiler)
- L2: Possibility of Composing Flawed Entity Relationships
- L3: Lack of Entities for Ducts, Pipes, and Detailed Spatial Context
- L4: Difficulty Validating Entities and Relationships
- L5: Inconsistent Data and Structure Semantics
- L6: Complex and Deep Inheritance Hierarchies
- L7: Inability to Represent System Behaviours
- L8: Static Artefacts and Limited Interactivity

Yefi, P., Menon, R. P., Eicker, U., & Guéhéneuc, Y. G. (2024). MetamEnTh: An Object-Oriented Metamodel for IoT Systems in Buildings. IEEE Internet of Things Journal, 11(15), 25818-25838.
Yefi, P., Menon, R., & Eicker, U. (2023, May). Building IoT systems modeling: A object-oriented Metamodeling approach. In 2023 IEEE/ACM 5th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT) (pp. 1-8). IEEE.

RQ1.1: Limitation of Existing Approaches (L1, L2, L4)



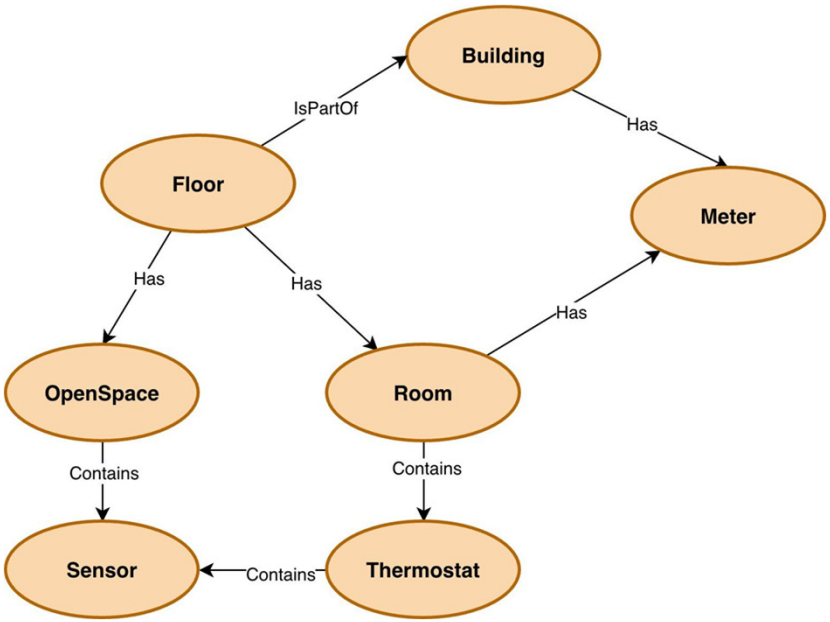
Brick Class Hierarchy

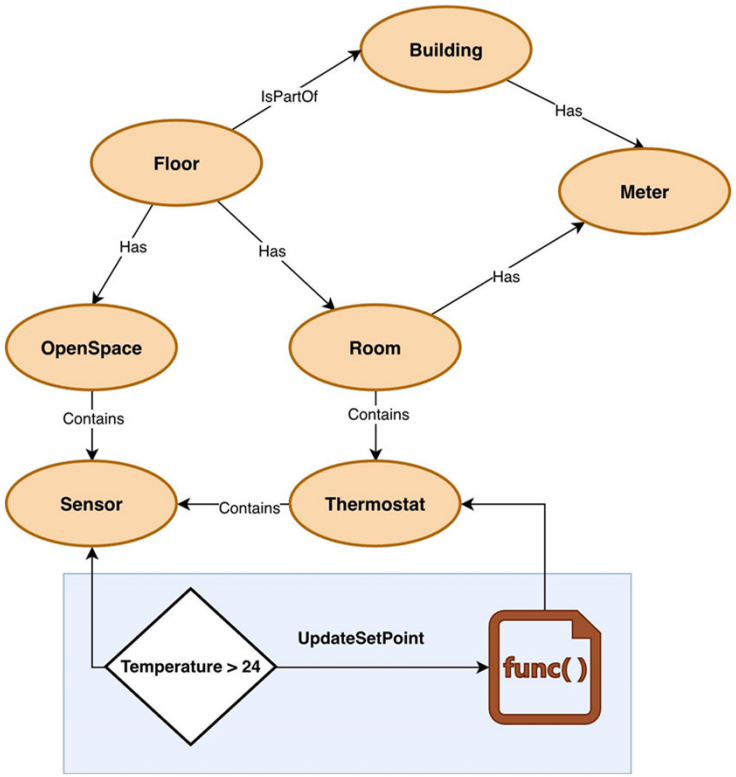
```
id: @388f-83efa-323de
point
dis: "Discharge Air Temperature"
sensor
air
temp
discharge
```

Project Haystack Tags

RQ1.1: Limitation of Existing Approaches

- Ontologies can model complex entity relationships for varying system configurations
- They are not suitable to model dynamic operational aspects of MEP subsystems





Object-Oriented Modelling of MEP Systems

RQ1: How does an object-oriented metamodel enable the creation of accurate, inherently validated models compared to existing ontology-based approaches?

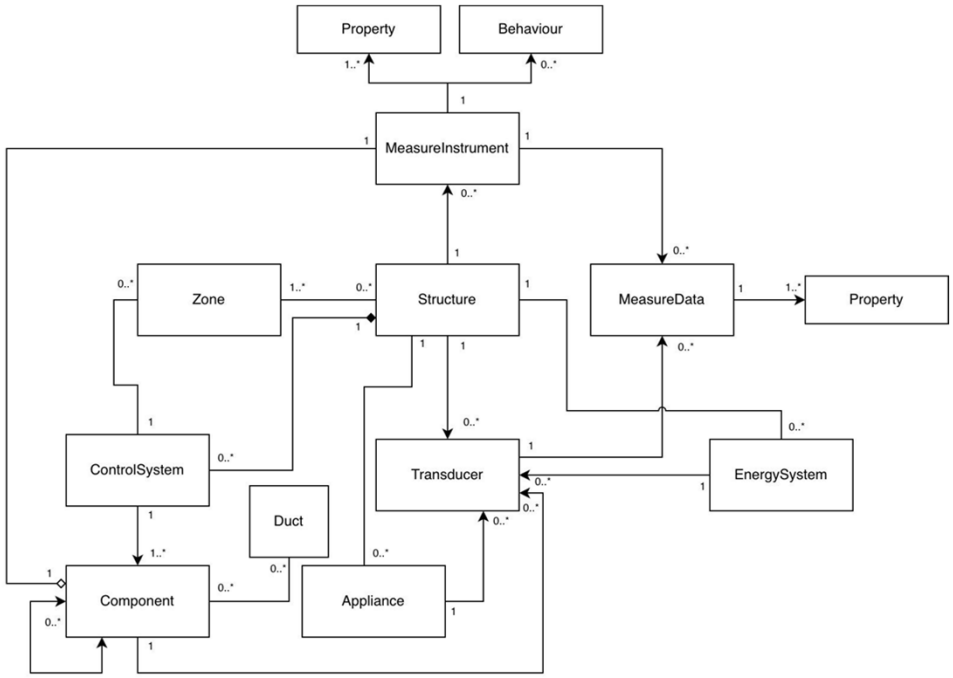
- RQ1.1: What are the key limitations of current built environment modelling approaches, particularly ontology-based systems?
- RQ1.2: How can an object-oriented modelling paradigm address these limitations and offer practical advantages for representing and validating MEP systems?

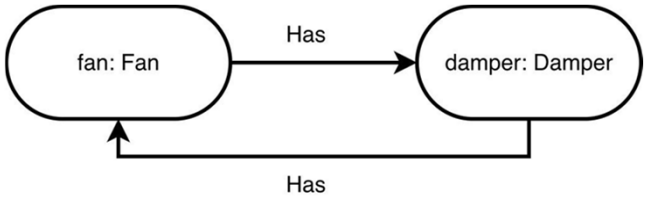
RQ2: How can an object-oriented metamodel represent mechanical, electrical, and plumbing (MEP) systems and their complex interactions?

Yefi, P., Menon, R. P., Eicker, U., & Guéhéneuc, Y. G. (2025a). Implementation of an Object-Oriented Metamodel for Modelling Building Systems. Scientific Reports (Manuscript submitted for publication)

RQ1.2 & RQ2: How Object-Oriented Models Represent MEP Systems

- A metamodel is a "model of models" [9]
- A model is an illustration of a specific aspect of reality [10]
- The expressiveness of a metamodel determines what an instantiated model can describe
- Metamodels **enforces built-in** constraints for all instantiated models





RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

Targeted needs assessment with practitioners and researchers

- **30–45 minutes** Interview with **2** facility managers
  - **34 combined years of experience** managing multiple commercial buildings
- Survey with **10** respondents
  - **6** student researchers, **2** facility managers, **1** professor and **1** industrial researcher
  - Participants worked in multiple countries: Canada, Lebanon, Iraq, Turkey, Syria, the United States, Australia, Chile, Colombia, Ecuador, Germany, India, Mexico, and Spain

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

Interview Discussions

- If you were creating building models for simulation to improve energy efficiency, which building systems would you prioritise?
- Based on your experience, is it acceptable to make minor errors in building models intended for energy simulation?
- Could you explain why modelling ventilation ducts, the components within them (e.g., fans, dampers), and their connections (e.g., to chillers) is important?
- Have you worked with BMS data outside the BMS itself? If so, what data did you use, for what purpose, and in what format?
- Is it important for a user to determine which time-series (sensor) data to include in a model, and the required data duration?

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

Interview Insights

- **Modelling Priorities**: Mass flow of air and water to accurately represent components necessary for calculating conductive and convective heat losses
- **Tolerance for Modelling Error**: Avoid duct and piping configuration errors that propagate into inaccurate performance assessments
- **Data Inclusion and Scope**: Allow flexible configuration of data inclusion thresholds
- **Use of BMS Data**: Trend logs data exported into Excel or other analytical tools to perform calculations on energy flux, entropy changes in heating systems, or power consumption in electrical systems

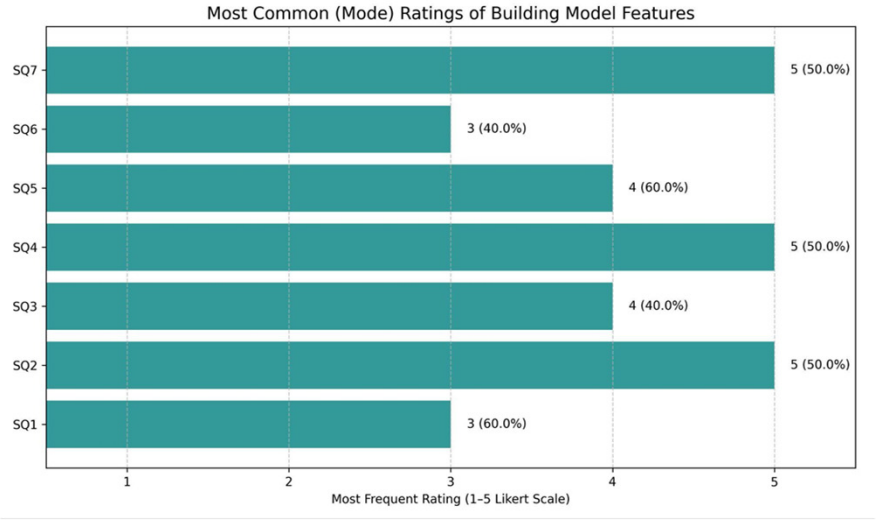
RQ1.2 & RQ2: How Object-Oriented Models Represent MEP Systems

Surveys Questions

- SQ1: Difficulty of creating building models from existing ontologies/metamodels
- SQ2: Importance of accurately modelling relationships between building systems and entities
- SQ3: Importance of modelling building-entity behaviour in a metamodel
- SQ4: Importance of distinguishing static vs. dynamic data in a building model
- SQ5: Importance of specifying the amount of historical sensor data in a model
- SQ6: Importance of explicitly modelling ductwork and associated systems
- SQ7: Importance of semantic organisation (spaces, floors, zones) for model usability

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

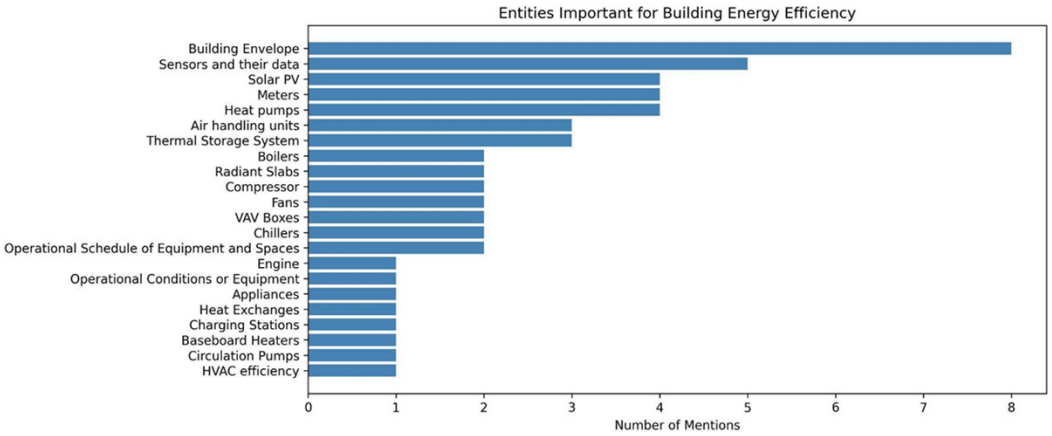
Surveys Insights

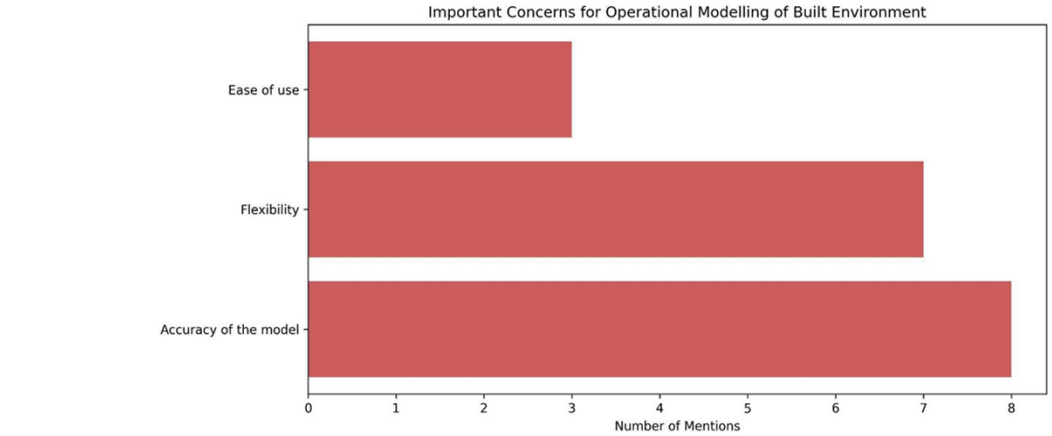


- **SQ2** (accurate models), **SQ4** (static vs dynamic data), and **SQ7** (semantic organisation) received a **rating of 5 from 50% of respondents**, highlighting their high importance
- **SQ1** (model creation) had a **modal rating of 3** and **SQ5** (amount of historial data) a **modal rating of 4** (each selected by 60%)
- **No item scored below 3**, indicating all aspects were viewed as at least moderately relevant

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

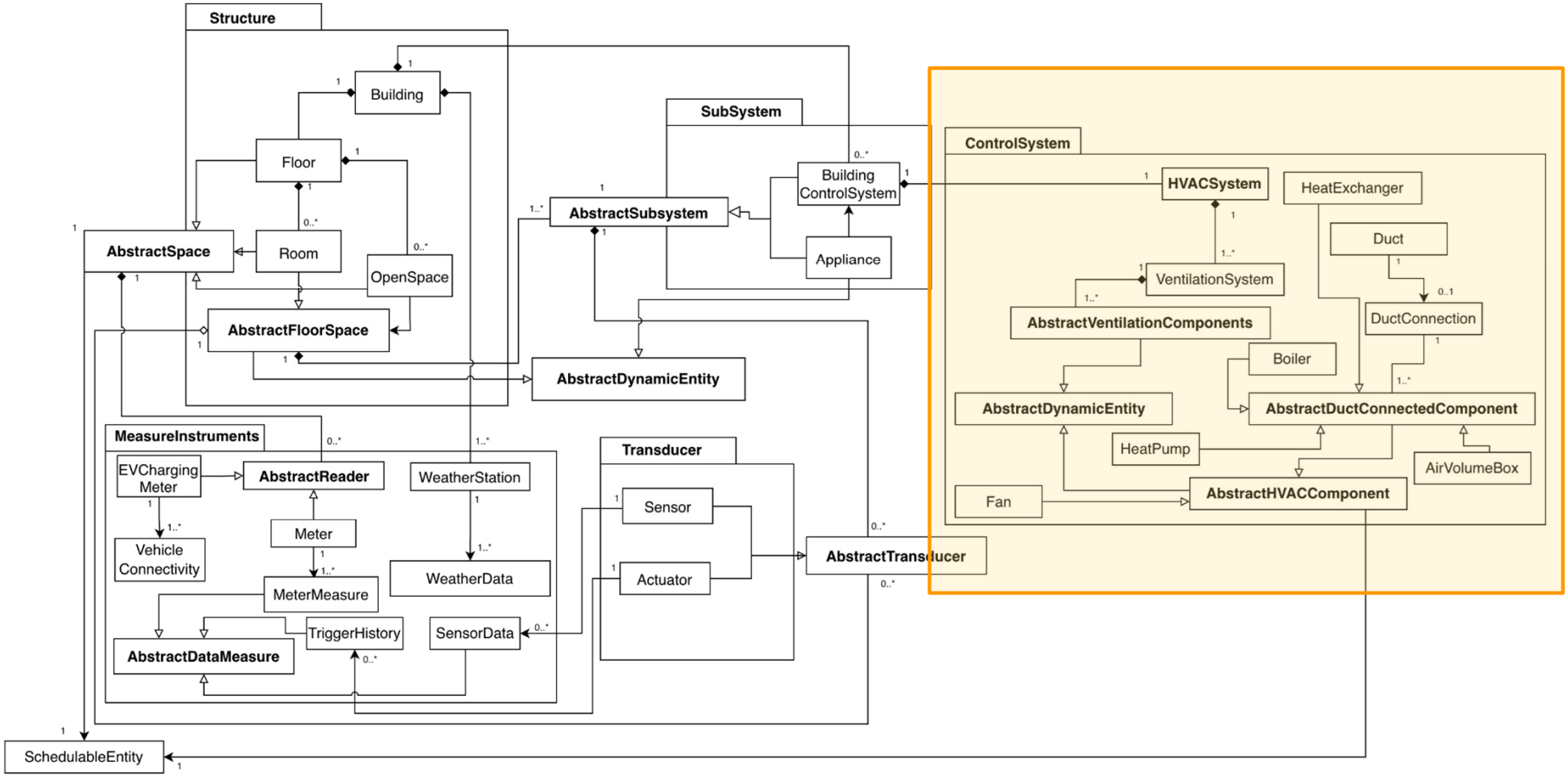
Surveys Insights





- The survey results indicate that the building envelope is viewed as the most essential entity for energy efficiency **(8 respondents)**, followed by sensors and their data **(5)**, solar PV **(4)**, meters **(4)**, heat pumps **(4)**, and air handling units **(3)**
- **Model accuracy** also emerged as the top concern in building energy modelling, cited by eight respondents

RQ2: How object-oriented models represents MEP subsystems



RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

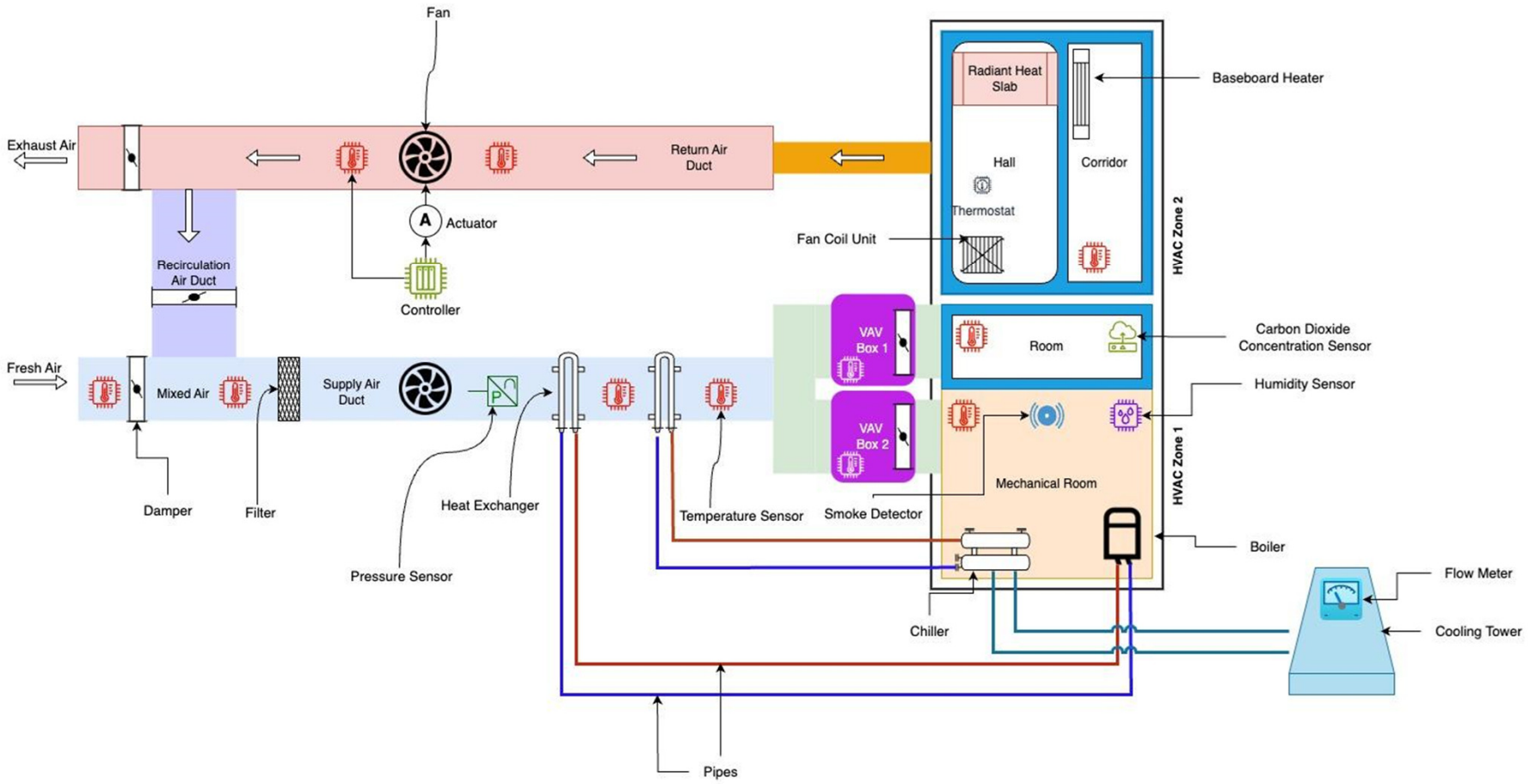
Empirical Evaluation

- Participants model an experimental HVAC system to Comparison MetamEnTh with Brick
- **10** participants
- **5** software engineers, **3** students, **2** researchers
- **80%** have **3** or more years of programming experience

Experimental Tasks

- Task One: Participants creates Brick and MetamEnTh models of the experimental HVAC system
- Task Two: Definition of four MEP domain relationships in models from Task One

RQ1.2 & RQ2: How Object-Oriented Models Represent MEP Systems



Experimental HVAC systems adapted from Pritoni et al. [10]

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

Table 3.1: Summary statistics for questionnaire responses comparing Brick and **MetamEnTh**

Question	Approach	Mean	Median	Min	Max	Std Dev
General Detail of Model	Brick	3.9	4.0	2	5	0.99
	MetamEnTh	3.8	4.0	2	5	1.03
Modelling Difficulty	Brick	2.5	2.0	1	5	1.08
	MetamEnTh	3.6	4.0	2	5	0.84
Model Accuracy	Brick	3.9	4.0	2	5	1.10
	MetamEnTh	4.2	4.0	3	5	0.63
Detail in Modelling HVAC Entities	Brick	3.6	4.0	1	5	1.35
	MetamEnTh	4.3	4.5	3	5	0.82

Table 3.2: Results of statistical tests comparing Brick and **MetamEnTh** responses

Question	Test Used	p-value	Significant ($p < 0.05$)?
General Modelling Detail	Paired t-test	0.859	No
Modelling Difficulty	Wilcoxon	0.027	**Yes**
Modelling Accuracy	Paired t-test	0.541	No
HVAC Modelling Detail	Wilcoxon	0.100	No

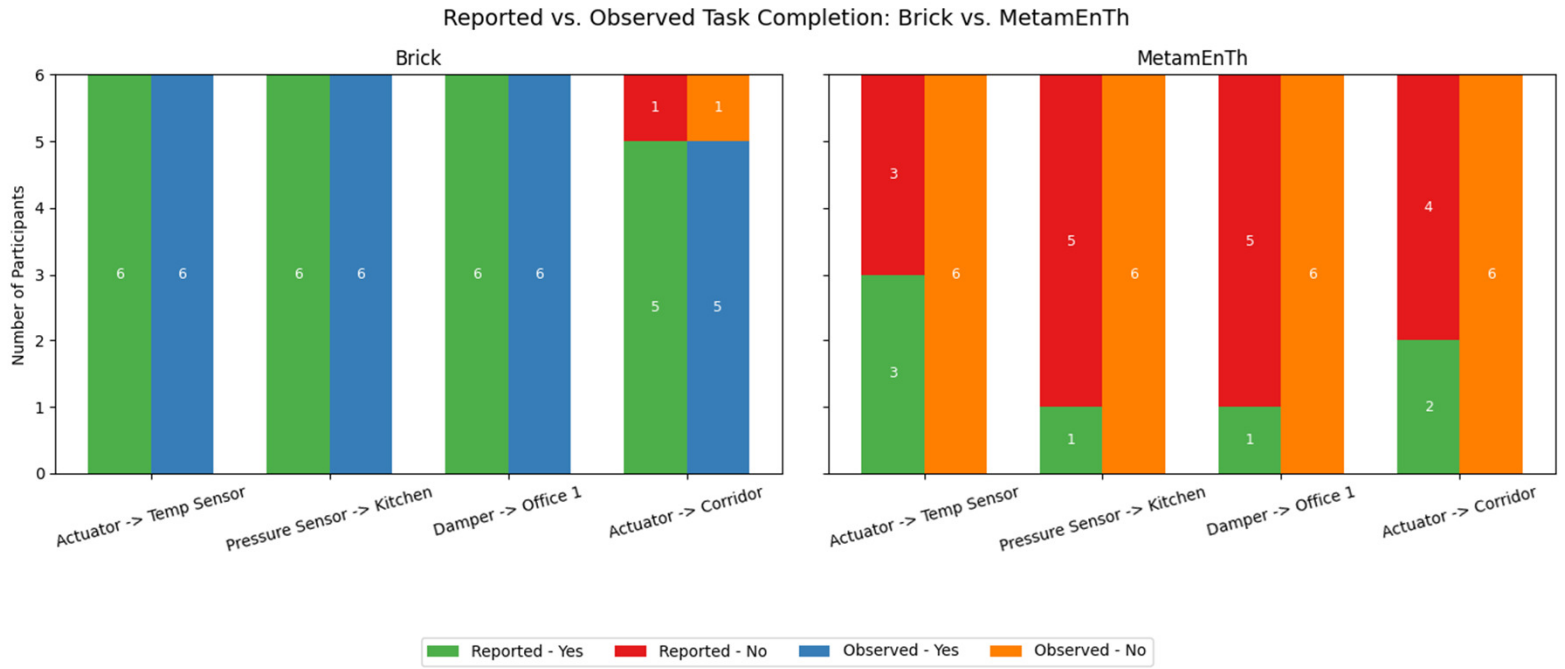
Paired test, when differences between paired responses follow a normal distribution

Non-parametric Wilcoxon test, otherwise

Treated ordinal data as interval for central tendencies and variability [12]

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

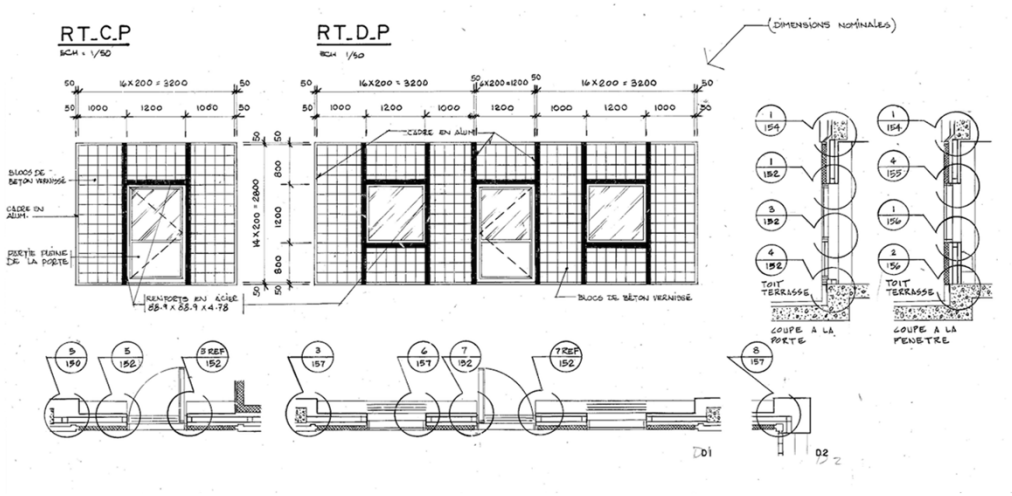
RQ1.2: **MetamEnTh prevents the creation of inaccurate models**



- **Brick: 6** participants created the four relationships, except one
- **MetamEnTh:** none implemented all four correctly

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

Practical Evaluation



Architectural Drawings

Table 3.3: Layer composition of cover RT-B1 of the LB Building

Name	Length (mm)	Height (mm)	Thickness (mm)	Material	Layer Order	Description
RT-B1	3,200	2,800	16	ordinary gypsum board	1	Window Area
RT-B1	3,200	2,800	42	metal half-timbering	2	Window Area
RT-B1	3,200	2,800	92	semi-rigid insulation	4	Window Area
RT-B1	3,200	2,800	16	ordinary gypsum board	5	Window Area
RT-B1	3,200	2,800	38	semi-rigid insulation	6	Window Area
RT-B1	3,200	2,800	21	air gap	7	Window Area
RT-B1	3,200	2,800	94	varnished concrete block	8	Window Area
RT-B1	2,000	2000	6	glass	1	Window
RT-B1	2,000	2000	12	window gas	2	Window
RT-B1	2,000	2000	6	glass	3	Window
RT-B1	2,000	2000	12	window gas	4	Window
RT-B1	2,000	2000	6	glass	5	Window

Extracted Envelope Layers

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

```
webster_library = BuildingCreator('webster_library.json',   LBDynamicDataReader(LIBRARY_FLOOR))
webster_library.create_building()

webster_library.add_building_envelope(
    BuildingEnvelopeProcessor('lb-layer-catalog.xlsx'),'LB Envelope')

envelope = webster_library.building.get_envelope_by_name('LB Envelope')

first_floor_north_covers = envelope.get_covers({'floor_number': 0,
                           'building_orientation': BuildingOrientation.NORTH.value})

glass_layers = first_floor_north_covers[0].get_layers({'material.material_type': MaterialType.GLASS.value})

envelope_obj = EnvelopeAnalysis(first_floor_north_covers)

cover_u_values = envelope_obj.calculate_envelope_u_values()

integrity_checks = envelope_obj.check_integrity()
total_area = envelope_obj.calculate_total_cover_area()
```

RQ1 & RQ2: How Object-Oriented Models Represent MEP Systems

- L1, L2, L4: MetamEnTh defines classes with **built-in properties, behaviours, and relationships**, along with **validation rules and constraints** that ensure accurate and easily verifiable models
- L3: Classes such as **Duct**, **DuctConnection**, and **HVACComponent** support detailed modelling of airflow systems
- L7, L8: **Memory-resident objects** and **methods** model dynamic behaviours

Limitations

- L1: Possible Ambiguity in Defining an Entity
- L2: Possibility of Composing Flawed Entity Relationships
- L3: Lack of Entities for Ducts, Pipes, and Detailed Spatial Context
- L4: Difficulty Validating Entities and Relationships
- L7: Inability to Represent System Behaviours
- L8: Static Artefacts and Limited Interactivity

Object-Oriented Modelling of MEP Systems: Threats to Validity

- **Internal Validity**
  - Participants had different levels of OOP and building-systems experience
  - Documentation was provided, but prior experience still likely affected task performance and perceived difficulty
- **External Validity**
  - Study involved only 10 participants
  - Findings may not generalise to industry practitioners or other regions
- **Construct Validity**
  - Model quality was defined by semantic correctness, relationship accuracy, and completeness using predefined rules
  - Different definitions or criteria could change how accuracy and detail are interpreted
- **Conclusion Validity**
  - Small sample size limited statistical power
  - Larger, more diverse samples are needed to draw stronger conclusions

RQ3: Interoperable Data Access for MEP Systems

Interoperable Data Access for MEP Systems

RQ3: How can practitioners and researchers access interoperable data of mechanical, electrical, and plumbing (MEP) systems in buildings?

- RQ3.1: What capabilities do BMS APIs provide to interact with building systems and the data they generate?
- RQ3.2: How can practitioners and researchers interact with building systems and their data in a standardised way?

Interoperable Data Access for MEP Systems

RQ3.1: Capabilities of BMS APIs

- We evaluate and group the API capabilities of **5** BMS vendors into **12** functional groups
- **BMS vendors**: Siemens (Desigo), Honeywell (EBI), Schneider Electric (Ecostruxture), Delta Controls (Enteliweb), Johnson Controls (Metasys)
- **12 functional groups**: alarms and events, audits, equipment, network devices, objects, space, time-series data, energy, carbon dioxide emissions, assets, energy LEED scoring, energy modelling

Yefi, P., Menon, R., & Eicker, U. (2024). Evaluation of APIs for data exchange with building management systems. In Proceedings of the ACM/IEEE 6th international workshop on software engineering research & practices for the internet of things (pp. 1–6)

RQ3.1: BMS API Capabilities

None of the BMS provide endpoints covering the 12 functional groups

- Schneider Electric's EcoStruxure — **10/12** functional groups
- Johnson Controls' Metasys — **9/12** functional groups
- Siemens' Desigo — **9/12** functional groups
- Delta Controls' Enteliweb — **6/12** functional groups
- Honeywell's Enterprise Building Integrator — **2/12** functional groups

- Data requirements of **7** common building applications [16]
  - Occupancy modelling, energy apportionment, model predictive control (MPC), participatory feedback, fault detection and diagnosis (FDD), non-intrusive load monitoring, demand-response

Yefi, P., Menon, R. P., Eicker, U., & Guéhéneuc, Y. G. (2025b). A study of the capabilities of building management system APIs and the limitations of their practical usage. ACM Transactions on Cyber-Physical Systems (Manuscript submitted for publication)

RQ3.1: BMS API Capabilities

Building entities (MEP) and their relationships:

- *ER1*: Sensor-to-Spatial Location
- *ER2*: Sensor-to-HVAC Component or Appliance
- *ER3*: Person-to-Spatial Location
- *ER4*: Spatial Hierarchy
- *ER5*: HVAC Component-to-Spatial Location
- *ER6*: HVAC Component-to-HVAC Component
- *ER7*: Meter-to-Sensor
- *ER8*: Meter-to-HVAC Component
- *ER9*: Meter-to-Spatial Location
- *ER10*: Appliance-to-Person

RQ3.1: BMS API Capabilities

Table 4.4: Common Building Applications, Building Entities and their Relationships, and API Requirements

Common Application	Entity Relationship	API Category
Occupancy Modelling	ER1, ER2, ER3, ER4, ER5, ER6, ER8, ER10	Equipment, Objects, Space, Time-series (occupancy)
Energy Apportionment	ER1, ER2, ER4, ER7, ER9	Objects, Space, Equipment, Time-series (metering data), Energy
Model Predictive Control	ER1, ER2, ER5, ER6	Objects, Space, Time-series, Equipment
Participatory Feedback	ER1, ER2, ER5, ER6	Objects, Space, Time-series, Equipment
Fault Detection and Diagnosis	ER1, ER2, ER5, ER6	Objects, Space, Time-series, Equipment
Non-Intrusive Load Monitoring	ER7, ER9	Time-series (energy), Objects
Demand Response	ER1, ER2, ER5, ER6	Objects, Space, Time-series, Equipment

Occupancy modelling requires data for nearly all entity relationships plus **Equipment**, **Objects**, **Space**, and **Time-series** APIs

All vendors expose **Objects** and **Time-series**

Only **Metasys** and **Desigo** provide the full APIs for advanced applications like demand response, FDD, participatory feedback, and MPC

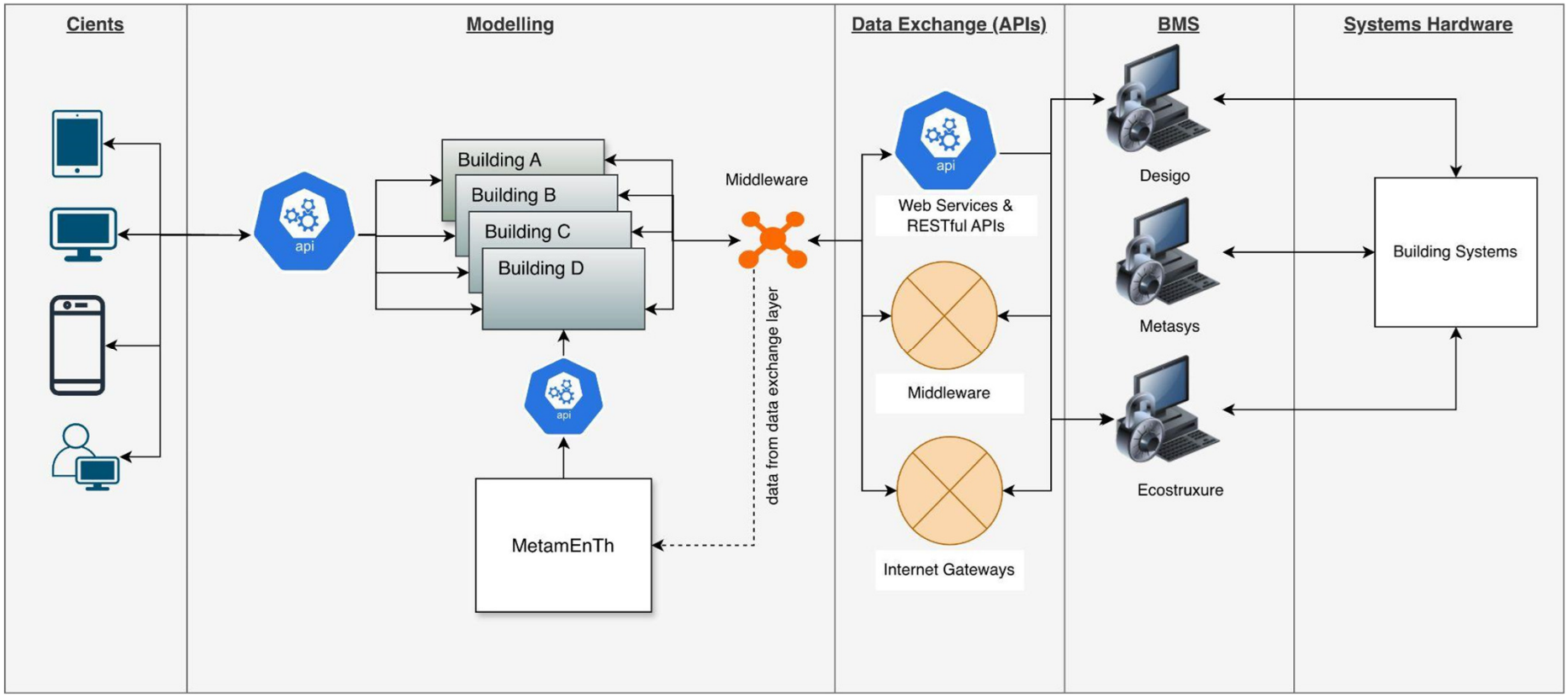
ER7: Meter-to-Sensor
ER9: Meter-to-Spatial Location

Interoperable Data Access for MEP Systems

RQ3: How can practitioners and researchers access interoperable data of mechanical, electrical, and plumbing (MEP) systems in buildings?

- RQ3.1: What capabilities do BMS APIs provide to interact with building systems and the data they generate?
- RQ3.2: How can practitioners and researchers interact with building systems and their data in a standardised way?

RQ3.2: Standardised MEP Access



Yefi, P., Ejaz, S., Menon, R. P., Eicker, U., & Guéhéneuc, Y. G. (2024, October). An Architectural Approach for Enhanced Data Interoperability Across Building Systems. In 2024 7th Conference on Cloud and Internet of Things (CIoT) (pp. 1-8). IEEE.

RQ3.2: Standardised MEP Access

Evaluation

- Multiple data sources: architectural drawings, Archidata, BMS (Enteliweb, Desigo)
- Middleware
  - Parsing and type conversion with **BuildingStructure**
  - Model creation with **BuildingCreator**
  - Historical sensor data with **DynamicDataReader**
  - Envelope data integration with **BuildingEnvelopeProcessor**
- A MetamEnTh model each for the Webster Library (LB) and Varennes Library
- A single temperature analysis client

RQ3.2: Standardised MEP Access

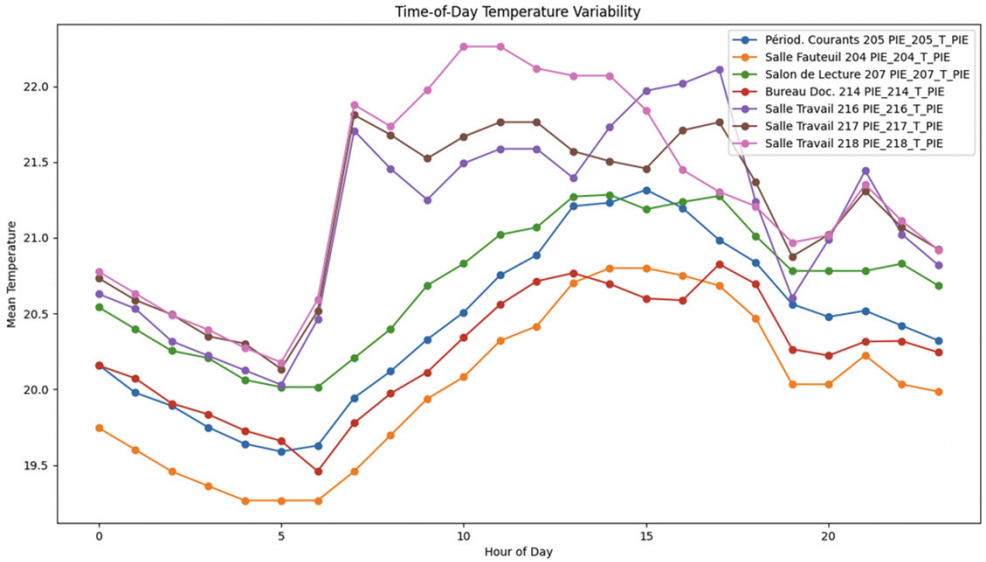
Webster Library (LB Building)

```
webster_library = BuildingCreator('webster.json', LBDynamicDataReader(LIBRARY_FLOOR))
webster_library.create_building()
webster_library.add_sensor_data()
data = webster_library.building.get_floor_by_number(LIBRARY_FLOOR).get_rooms({'room_type': 'Library'})
tempAnalysis = TemperatureAnalysis(data)
tempAnalysis.time_of_day_variability()
```

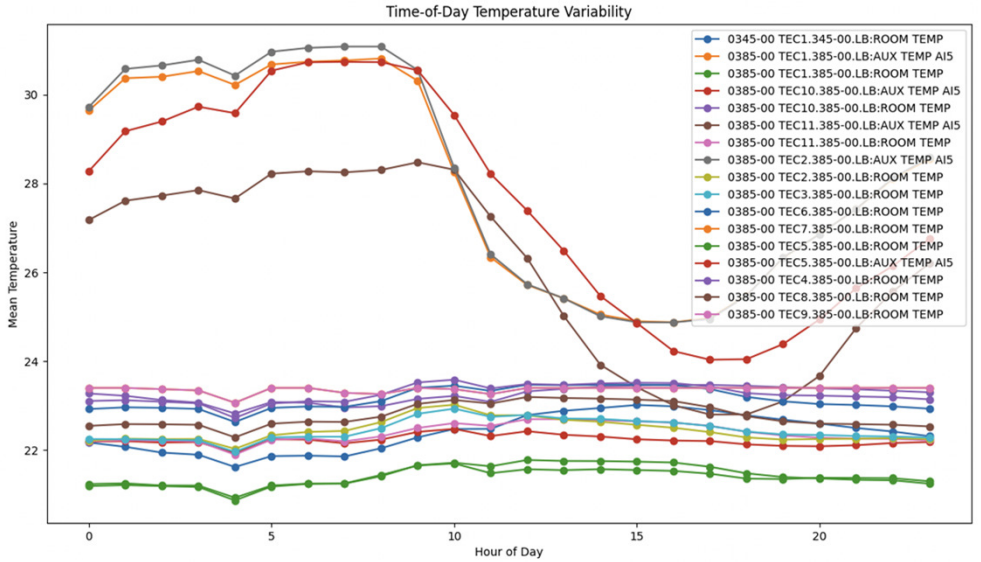
Varennes Library

```
varennes_library = BuildingCreator('varennes.json', VLDynamicDataReader())
varennes_library.create_building()
varennes_library.add_sensor_data()
data = varennes_library.building.get_floor_by_number(1).get_rooms()
tempAnalysis = TemperatureAnalysis(data)
tempAnalysis.time_of_day_variability()
```

RQ3.2: Standardised MEP Access



Time of Day Temperature Variability, Varennes



Time of Day Temperature Variability, LB

Interoperable Data Access for MEP Systems: Threats to Validity

- **Construct validity**
  - Relied on API documentation, which may not reflect real system behaviour
- **Internal validity**
  - Limited access to live BMS installations; manual extraction may introduce errors or miss real-world nuances
- **External validity**
  - Findings based on five vendors and selected use cases; applicability may vary across building types, regions, and software versions
- **Conclusion validity**
  - API–requirement mappings rely on literature and expert judgment; different categorizations could yield different conclusions

RQ4: Integrating Control Logic in Object-oriented MEP Models

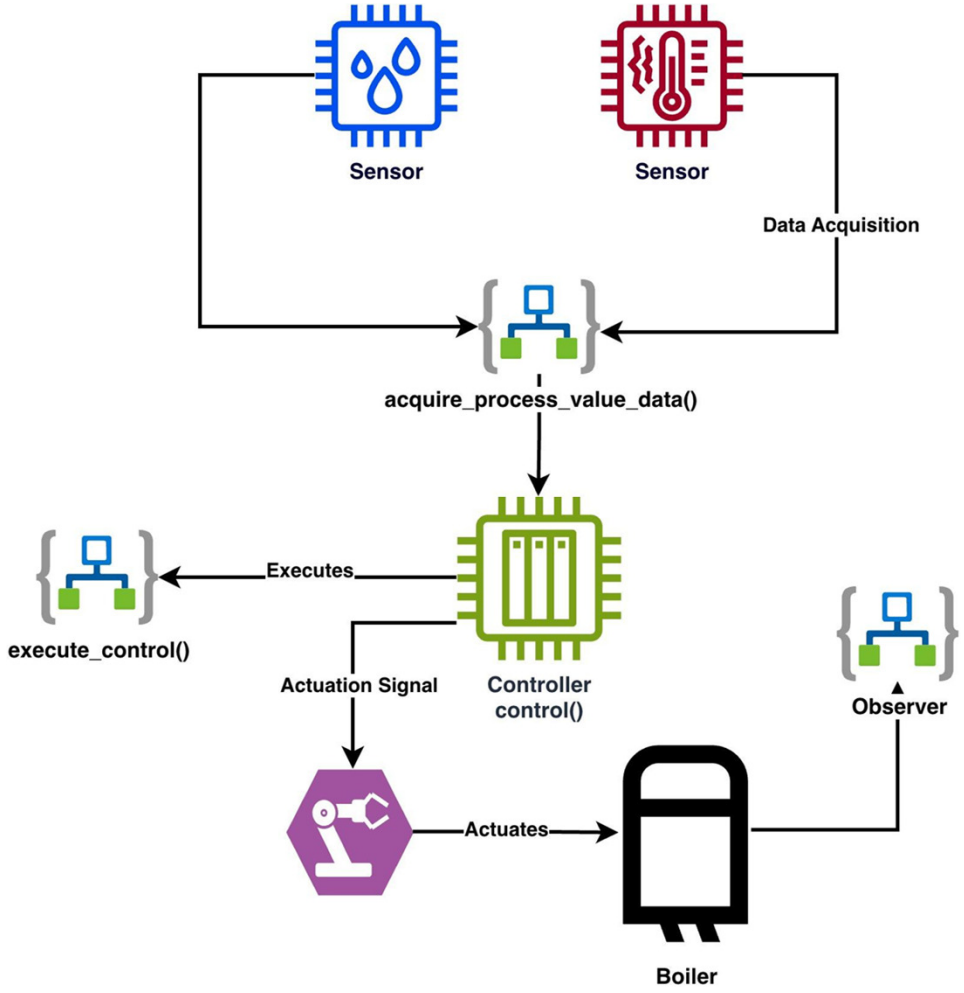
Integrating Control Logic in Object-oriented MEP Models

RQ4: How does an object-oriented modelling approach support the integration of control logic and promote broader adoption in practical energy-efficient applications?

- **Data Acquisition Challenge:** Advanced controls need high-quality real-time data, but BMSs use heterogeneous formats and proprietary interfaces, complicating integration
- **Architectural Complexity:** Traditional BMS control modules are vendor-specific and inflexible
- Interfaces to embed control logic into MetamEnTh models using OOP principles
- Demonstrate on/off control integration through two real building use cases

Peter Yefi, Reiner Braun, Philipp Regel, Ramanunni Parakkal Menon, and Ursula Eicker. Enabling Digital Twins in Built Environments with Object-Oriented Models. In Proceedings of 8th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT 26). ACM

Integrating Control Logic in Object-Oriented MEP Models



Integrating Control Logic in Object-Oriented MEP Models

Evaluation: LB Building

- Implemented **on/off control** for the Living Lab Room 345
- Control logic retrieves **temperature** and **CO_2** values via **Desigo BMS API**
- **Boiler ON**: temperature ≤ 16 °C and CO_2 ≥ 420 ppm
  - indicates occupancy & low temperature
- **Boiler OFF**: temperature ≥ 24 °C or CO_2 ≤ 400 ppm
  - space warm or unoccupied
- Lack of BMS write access
  - actuation simulated via console output

Integrating Control Logic in Object-Oriented MEP Models

Evaluation: LB Building

```
LIBRARY_FLOOR=3
temp_sensor    = (library.building.get_floor_by_number(LIBRARY_FLOOR).get_room_by_name('345-00')
                  .get_transducer_by_name('TEC1.345-00.LB:ROOM TEMP'))
co2_con_sensor = Sensor('TEC1.345-00.LB:ROOM C02', SensorMeasure.CARBON_DIOXIDE, MeasurementUnit.PARTS_PER_MILLION,
                  SensorMeasureType.NON_DISPERSIVE_INFRARED, 900)
library.building.get_floor_by_number(LIBRARY_FLOOR).get_room_by_name('345-00').add_transducer(co2_con_sensor)

temp_set_point    = MeasureFactory.create_measure(RecordingType.CONTINUOUS.value,
                    Measure(MeasurementUnit.DEGREE_CELSIUS, 16, 24))
co2_con_variable = MeasureFactory.create_measure(RecordingType.CONTINUOUS.value,
                    Measure(MeasurementUnit.PARTS_PER_MILLION, 400, 420))

boiler     = Boiler('CTRL.BL', BoilerCategory.NATURAL_GAS, PowerState.ON)
controller = Controller('CTR')
actuator   = Actuator("Boiler.ACT", boiler)
controller.add_transducer(temp_sensor)
controller.add_transducer(co2_sensor_con_sensor)
controller.add_transducer(actuator)
controller.add_set_point(temp_set_point, (temp_sensor.name, actuator.name))
controller.add_set_point(co2_con_variable, (co2_sensor_con_sensor.name, actuator.name))

binary_control = LBBinaryControl([temp_sensor, co2_con_sensor], actuator, [temp_set_point, co2_con_variable])
controller.control(binary_control)
```

Integrating Control Logic in Object-Oriented MEP Models

Evaluation: Mixed-Use Building, Ludwigsburg, Germany

- IoT-based setup monitors and controls a thermostat in Unit 3 (office)
- Devices communicate via LoRaWAN and are programmed using Node-RED
- FIWARE REST API (HTTP) exposes FIWARE Data Model of devices
- Conditional logic ensures setpoints remain within **20–26 °C** to prevent overheating/cooling
- **PATCH** request updates valve position through real-time monitoring
- Physical actuation not connected to HVAC

Integrating Control Logic in Object-Oriented MEP Models

Evaluation: Mixed-Use Building, Ludwigsburg, Germany

```
def execute_control(self, process_value: Dict):
    if process_value["temperature"] >
        self.control_thresholds[self.SETPOINT_INDEX].maximum:

            self.process_actuator.actuated_component.add_damper_position(DamperPosition(
            self._original_set_point - self.VALVE_UPDATE_VAL))

    elif process_value["temperature"] <
          self.control_thresholds[self.SETPOINT_INDEX].minimum:

            self.process_actuator.actuated_component.add_damper_position(DamperPosition(
            self._original_set_point + self.VALVE_UPDATE_VAL))
```

Integrating Control Logic in Object-Oriented MEP Models: Threats to Validity

- **Construct validity**
  - No expertise claimed in designing control strategies
  - Focus on demonstrating integration, not optimising control logic
- **Internal validity**
  - Current implementation lacks interfaces for advanced strategies
  - Extending to complex controls may require class modifications
- **External validity**
  - Results may not generalize to all control scenarios
- **Conclusion validity**
  - Conclusions apply only to integration mechanisms, not control performance or correctness
  - Broader inferences beyond integration are unsupported

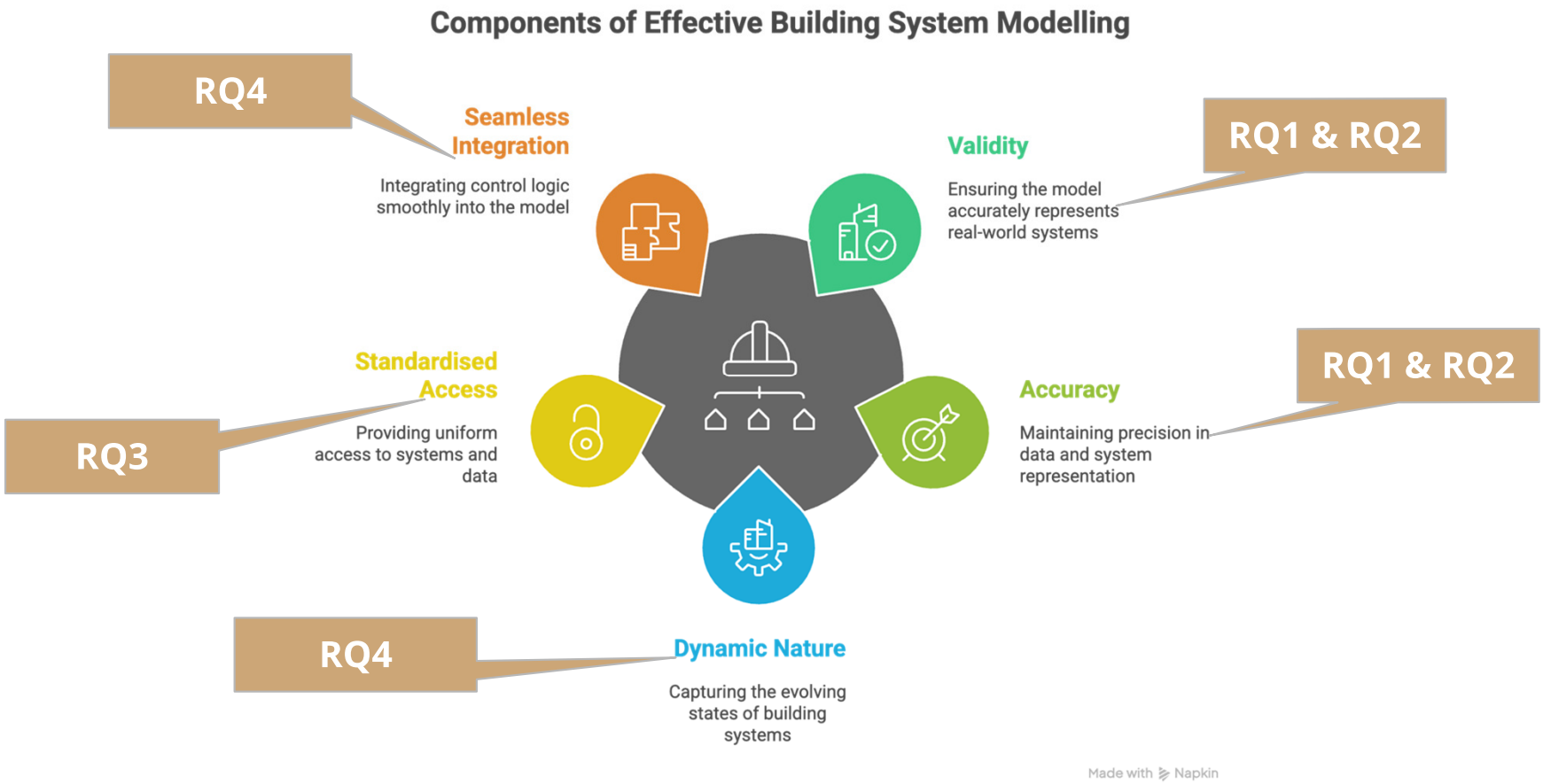
Conclusion

- Buildings rely on complex systems managed by BMS, yet still consume **~30%** more energy than necessary due to inefficiencies and lack of interoperability
- Deploying solutions for building efficiency goes beyond static semantic representation of MEP systems with ontologies
- Alternative object-oriented metamodels to address limitations of ontologies

RQ1 & RQ2 **RQ3** **RQ4**

Can an **object-oriented modelling approach represent mechanical, electrical, and plumbing (MEP) subsystems** in buildings, provide **consistent and standardised access to their data**, and enable the implementation and **deployment of control strategies** through predefined classes, validation mechanisms, and structured interfaces that facilitate integration with Building Management Systems (BMS)?

Conclusion: Answer



Future Work

Threats to Validity	Short Term
• Small sample size for survey and interview • 10 developers involved in experiment • Perceived difficulty in using MetamEnTh	• Broader evaluation of MetamEnTh • Implement Building Information Modelling (BIM) importers • Inclusion of utility methods for energy-efficiency related tasks • Interfaces for advanced control strategy integration
	Mid-Term
• Perceived difficulty in using MetamEnTh • Lack of write API access • Evaluation limited to 2 buildings	• User-friendly editors and plugins • Cross-building pilots • Adoption of MetamEnTh for digital twins
	Long Term
	• Alignment MetamEnTh with semantic standards, e.g., ASHRAE (223P), IFC, Brick

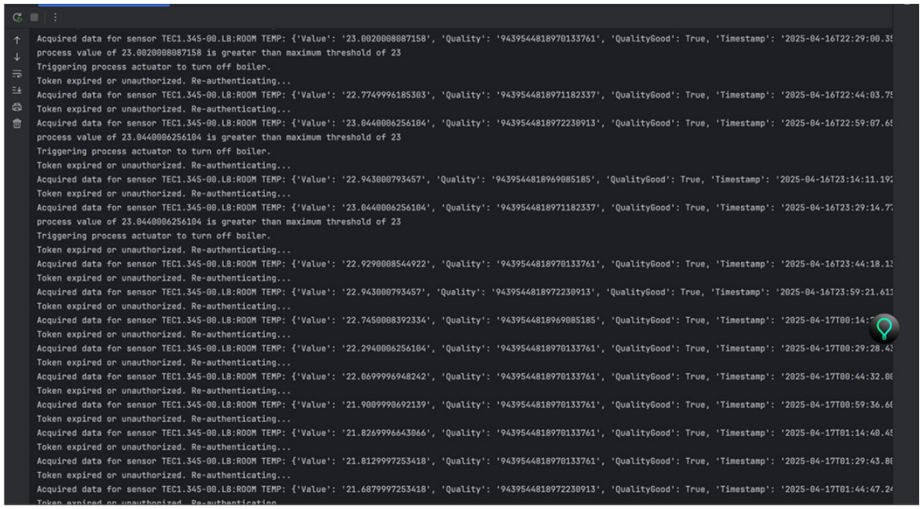
References

1. https://www.energy.gov/sites/prod/files/2017/03/f34/qtr-2015-chapter5.pdf
2. Wang, Y., Kuckelkorn, J., & Liu, Y. (2017). A state of art review on methodologies for control strategies in low energy buildings in the period from 2006 to 2016. *Energy and Buildings*, *147*, 27-40
3. Buildings - Energy System - IEA
4. Daniele, S. (2025). Ontologies for the Reconfiguration of Domestic Living Environments: A Systematic Literature Review. *Information*, *16*(9), 752.
5. Khabbazi, A. J., Pergantis, E. N., Premer, L. D. R., Papageorgiou, P., Lee, A. H., Braun, J. E., ... & Kircher, K. J. (2025). Lessons learned from field demonstrations of model predictive control and reinforcement learning for residential and commercial HVAC: A review. *arXiv preprint arXiv:2503.05022*
6. Donovan, P. (2020). Three Essential Elements of Next Generation Building Management Systems (BMS) (White Paper). 35 Rue Joseph Monier, 92500 Rueil-Malmaison, France: Schneider Electric - Science Center
7. Stolk, S., & McGlinn, K. (2020, June). Validation of IfcOWL datasets using SHACL. In *LDAC* (pp. 91-104).
8. Wang, C., Zhang, L., & Yan, W. (2024). Enhancement and validation of IFCOWL ontology based on Shapes Constraint Language (SHACL). *Automation in Construction*, *160*, 105293.
9. Favre, J. M. (2005). Foundations of meta-pyramids: Languages vs. metamodels–Episode II: Story of thotus the baboon. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.
10. Yonglin, L., Zhi, Z., & Qun, L. (2020). An ontological metamodeling framework for semantic simulation model engineering. *Journal of Systems Engineering and Electronics*, *31*(3), 527-538.
11. Pritoni, M., Paine, D., Fierro, G., Mosiman, C., Poplawski, M., Saha, A., ... & Granderson, J. (2021). Metadata schemas and ontologies for building energy applications: A critical review and use case analysis. *Energies*, *14*(7), 2024.
12. Norman, G. (2010). Likert scales, levels of measurement and the "laws" of statistics. *Advances in health sciences education*, *15*(5), 625-632.
13. McNemar, Q. (1947). Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, *12*(2), 153-157.
14. Afram, A., & Janabi-Sharifi, F. (2014). Theory and applications of HVAC control systems–A review of model predictive control (MPC). *Building and environment*, *72*, 343-355.
15. Khabbazi, A. J., Pergantis, E. N., Premer, L. D. R., Papageorgiou, P., Lee, A. H., Braun, J. E., ... & Kircher, K. J. (2025). Lessons learned from field demonstrations of model predictive control and reinforcement learning for residential and commercial HVAC: A review. *arXiv preprint arXiv:2503.05022*.
16. Bhattacharya, A., Ploennigs, J., & Culler, D. (2015, November). Short paper: Analyzing metadata schemas for buildings: The good, the bad, and the ugly. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments* (pp. 33-34).

Integrating Control Logic in Object-Oriented MEP Models

Evaluation: LB Building

```
Acquired data for sensor TEC1.345-00.LB:ROOM TEMP: {'Value': '24.0020008087158', 'Quality':
'9439544818970133761', 'QualityGood': True, 'Timestamp': '2025-04-16T22:29:00.35'}
Process value of 24.0020008087158 is greater than maximum threshold of 24
Triggering process to turn off boiler.
Token expired or unauthorized. Re-authenticaing…

Acquired data for sensor TEC1.345-00.LB:ROOM TEMP: {'Value': '23.7720008087158', 'Quality':
'9439544818970133761', 'QualityGood': True, 'Timestamp': '2025-04-16T22:44:.03.75'}
Token expired or unauthorized. Re-authenticaing…
```



Conclusion: Core Contributions

- **RQ1**: Identified **8** limitations of existing approaches
- **RQ1**: MetamEnTh addresses limitations using a grounded theory foundation, iterative practitioner-informed design, and engineering-based implementation
- **RQ1&2**: Empirical and practical evaluation with Webster Library, Varennes Library and facility managers
- **RQ1&2**: Complete envelope data of LB
- **RQ3**: BMS APIs and their capabilities to support **7** common building application
- **RQ3**: Architecture for accessing standardised data of MEP systems
- **RQ4**: MetamEnTh interfaces for control logic integration
- **RQ4**: Practical demonstration of control logic integration with LB and mix-used building in Germany

Interoperable Data Access for MEP Systems

RQ3.2: Standardised access to building systems

- We propose a **5** layer architecture for a standardised access to MEP systems and their data
- **5 layers**: Clients, Modelling, Data Exchange, BMS, Systems Hardware (MEP)
- BMS APIs exist in the Data Exchange Layer
- MetamEnTh exists in the Modelling Layer

Conclusion: Answer

- Object-oriented modelling effectively represents MEP systems **(RQ1 & RQ2)** and integrates heterogeneous BMS data **(RQ3)** for seamless control deployment **(RQ4)**
- Encapsulation, interfaces, and design patterns make the metamodel flexible, extensible, and practical for real-world use
- MetamEnTh lays the foundation for next-generation digital twins, simulations, and real-time control in the built environment

Integrating Control Logic in Object-Oriented MEP Models

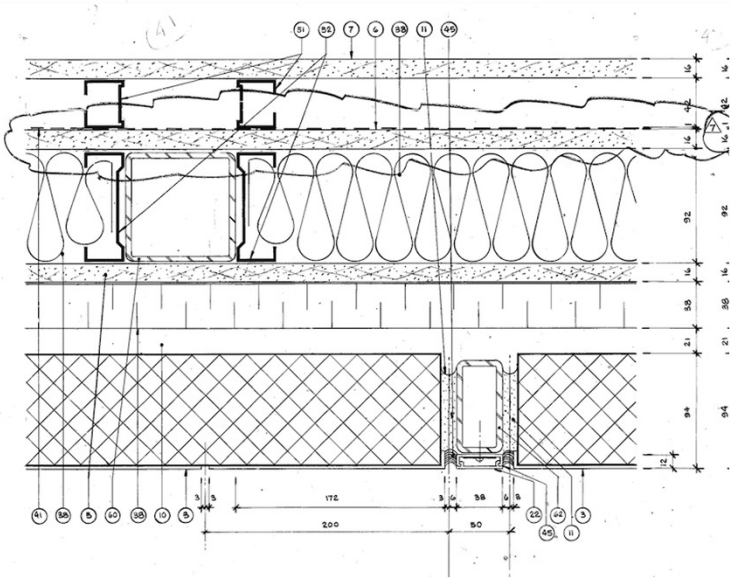
AbstractBinaryControl

- **Attributes:** *process_value_sensors: List[Sensor], actuator: Actuator, control_thresholds: List[ContinousMeasure], run_duration: float*
- **Methods:**
  - *acquire_process_value_data(self) -> Dict*: Encapsulates data acquisition logic
  - *execute_control(self, process_value: Dict) -> None*: Encapsulates control logic

Controller

- **Attributes:** *set_points: Dict[str, ContinuousMeasure] (e.g. {"sensor_name:actuator_name": ContinuousMeasure}), controller_entities: List[Union[AbstractHVACComponent, Appliance]]*
- **Methods:**
  - *control(self, control_obj: AbstractControl) -> Dict*: Orchestrates the execution of control logic

RQ3.2: Standardised MEP Access



```
webster_library = BuildingCreator( file_path: 'webster_library/data/webster.json',  dynamic_data_reader: None)
webster_library.create_building()
webster_library.add_sensor_data()

webster_library.add_building_envelope(BuildingEnvelopeProcessor('../webster_library/data/lb-layer-catalog.xlsx'),
                                      envelope_name: 'LB Envelope')

envelope = webster_library.building.get_envelope_by_name('LB Envelope')
first_floor_north_covers = envelope.get_covers({'floor_number': 0,
                                                'building_orientation': BuildingOrientation.SOUTH.value})
bishop_roof_covers = envelope.get_covers({'floor_number': 6,
                                          'building_orientation': BuildingOrientation.TOP.value})
filtered_layers = first_floor_north_covers[0].get_layers({'material.material_type': MaterialType.GLASS.value})
```

```
Cover(UID: 5424d339-b85a-4d8b-8a38-69e1eadf4334, Cover Type: CoverType.ROOF, Building Orientation: Top, Floor Number: 6, Layers:
Layer(UID: 3a81c80a-56a2-478a-8dc2-6eeff482e78b, Height: 36000 MeasurementUnit.MILLIMETERS, Length: 22775 MeasurementUnit.MILLIMETERS, Thickness: 20 MeasurementUnit.MILLIMETERS,
Layer(UID: 8ce16f28-defe-4a99-840b-b6ba870f169a, Height: 36000 MeasurementUnit.MILLIMETERS, Length: 22775 MeasurementUnit.MILLIMETERS, Thickness: 102 MeasurementUnit.MILLIMETERS,
Layer(UID: a3eb5250-2805-444e-bd41-6761a3e915c6, Height: 36000 MeasurementUnit.MILLIMETERS, Length: 22775 MeasurementUnit.MILLIMETERS, Thickness: 180 MeasurementUnit.MILLIMETERS,
Layer(UID: 66931335-a062-48aa-bb1a-af774d1acd7d, Height: 36000 MeasurementUnit.MILLIMETERS, Length: 22775 MeasurementUnit.MILLIMETERS, Thickness: 280 MeasurementUnit.MILLIMETERS,
```

```
Material: Material(UID: 1303f172-d580-4436-a151-77cda9f15dc3, Description: crushed stone ballast, Type: CrushedStoneBallast, Density: 0 kg/m3, Heat Capacity: 0 J/K, Thermal Trar
Material: Material(UID: a142b4e8-fefc-44d9-9d1b-09e629512ab6, Description: rigid insulation, Type: RigidInsulation, Density: 0 kg/m3, Heat Capacity: 0 J/K, Thermal Transmittance
Material: Material(UID: e0945928-03ba-4f86-8c20-27485ebd77f1, Description: rubber membrane, Type: RubberMembrane, Density: 0 kg/m3, Heat Capacity: 0 J/K, Thermal Transmittance:
terial: Material(UID: 7a3f23e7-bd6a-4018-b758-2f795cbe708e, Description: concrete, Type: Concrete, Density: 0 kg/m3, Heat Capacity: 0 J/K, Thermal Transmittance: 0 W/(m2.K), Ther
```

RQ1.1: Limitation of Existing Approaches (L1, L2, L4)

Open-world assumption [7]:

- Resource Description Framework (RDF), Web Ontology Language (OWL) ontologies are designed for inference
- Restrictions are not data constraints
- Missing information are just not captured yet, or they are elsewhere; they are unknown

Shape Constraint Language (SHACL) closed-world assumption [8]:

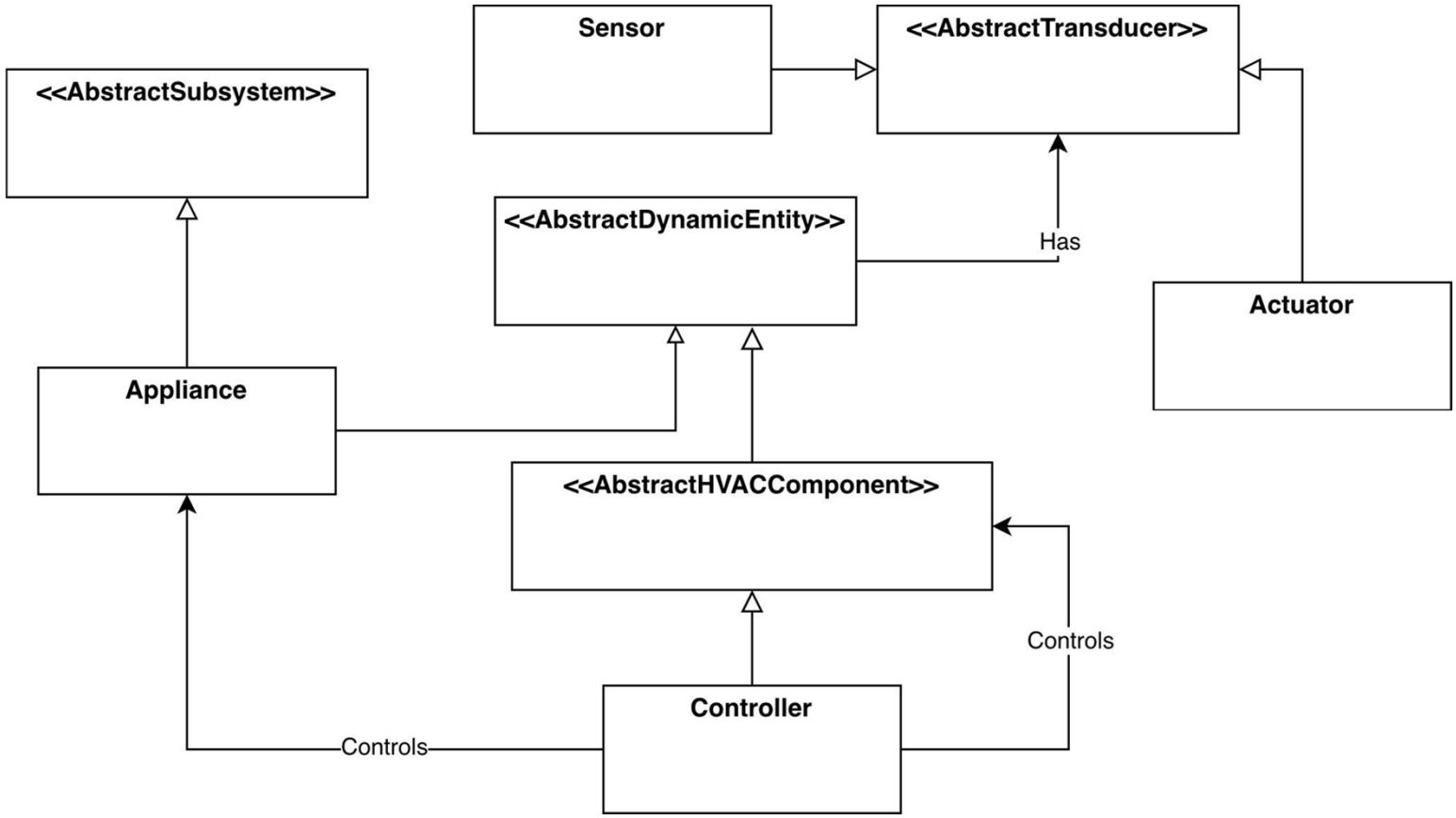
- Adds constraints to triples in RDF graphs
- Any data or fact not stated is considered false
- External to the model, and not enforced

Integrating Control Logic in Object-Oriented MEP Models

Evaluation: Mixed-Use Building, Ludwigsburg, Germany

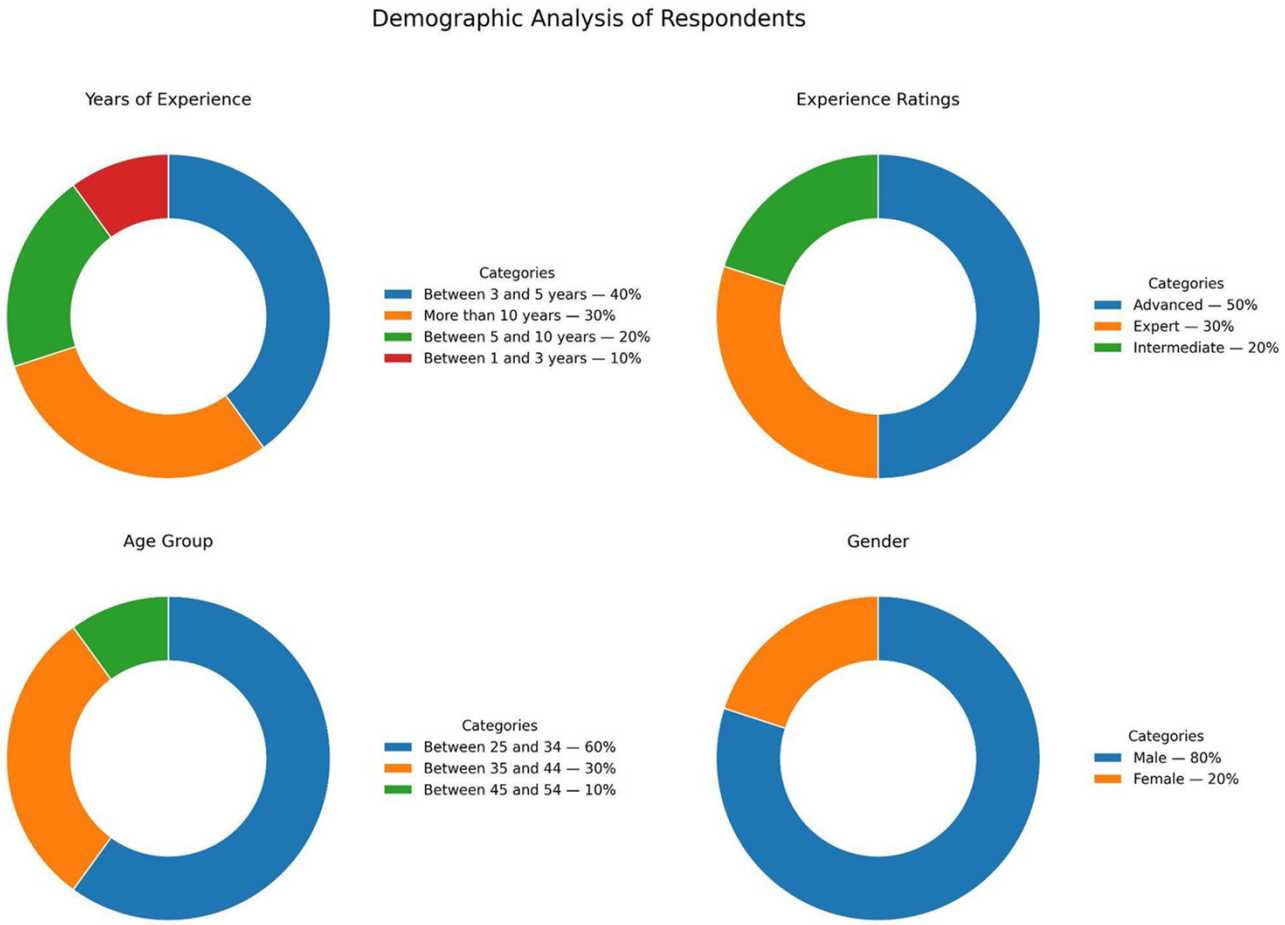
```
def valve_update_observer(self, action, valve_position):

    uri = f'entities/{self.process_actuator.actuated_component.name}/attrs'
    json_data = {
        "setPointTemperature": {
            "type": "Property",
            "value": valve_position.value
        },
        "@context": [
            "https://smartdatamodels.org/context.jsonld"
        ]
    }
    self._iot_api.make_authenticated_request(uri, 'PATCH', json_data)
```

Integrating Control Logic in Object-Oriented MEP Models



MetamEnTh classes for control integration

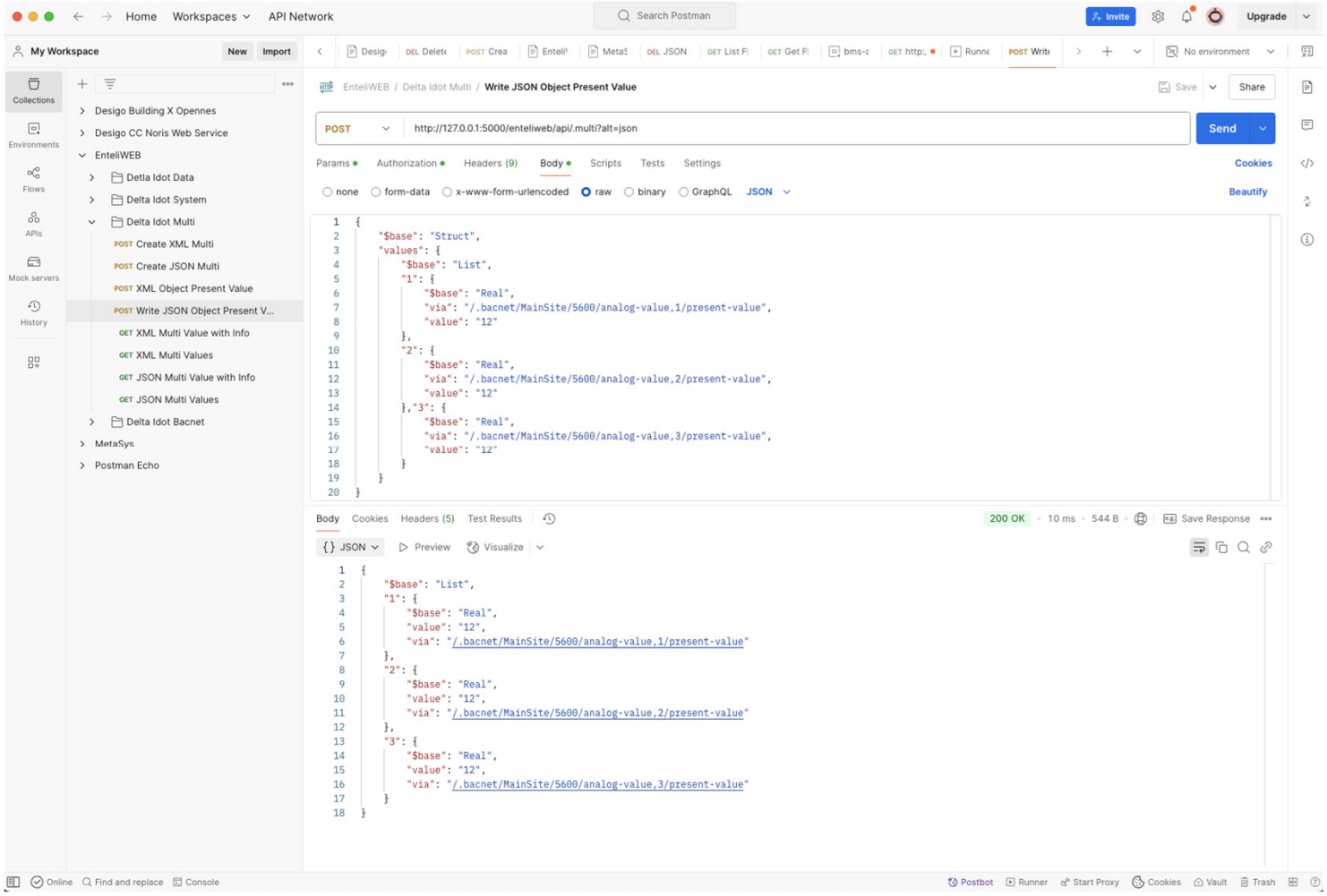
RQ1.2 & RQ2: How Object-Oriented Models Represent MEP Systems



RQ2: Practitioners and Researchers Interactions with BMS APIs

- We mocked selected API endpoints from EnteliWEB, Desigo and Metasys
- We evaluated the BMS APIs on four criteria beyond their functional capabilities:
  - Standardisation: HTTP verbs, status codes
  - Error handling: error status codes and HTTP verbs
  - Security: authentication and authorisation
    - Use of OAuth tokens
    - American Society of Heating Refrigeration, and Air conditioning Engineers (ASHRAE) Addendum 135-2012am
  - Interoperability: Data and URI formats
    - ASHRAE Addendum 135-2012am

RQ2: Practitioners and Researchers Interactions with BMS APIs

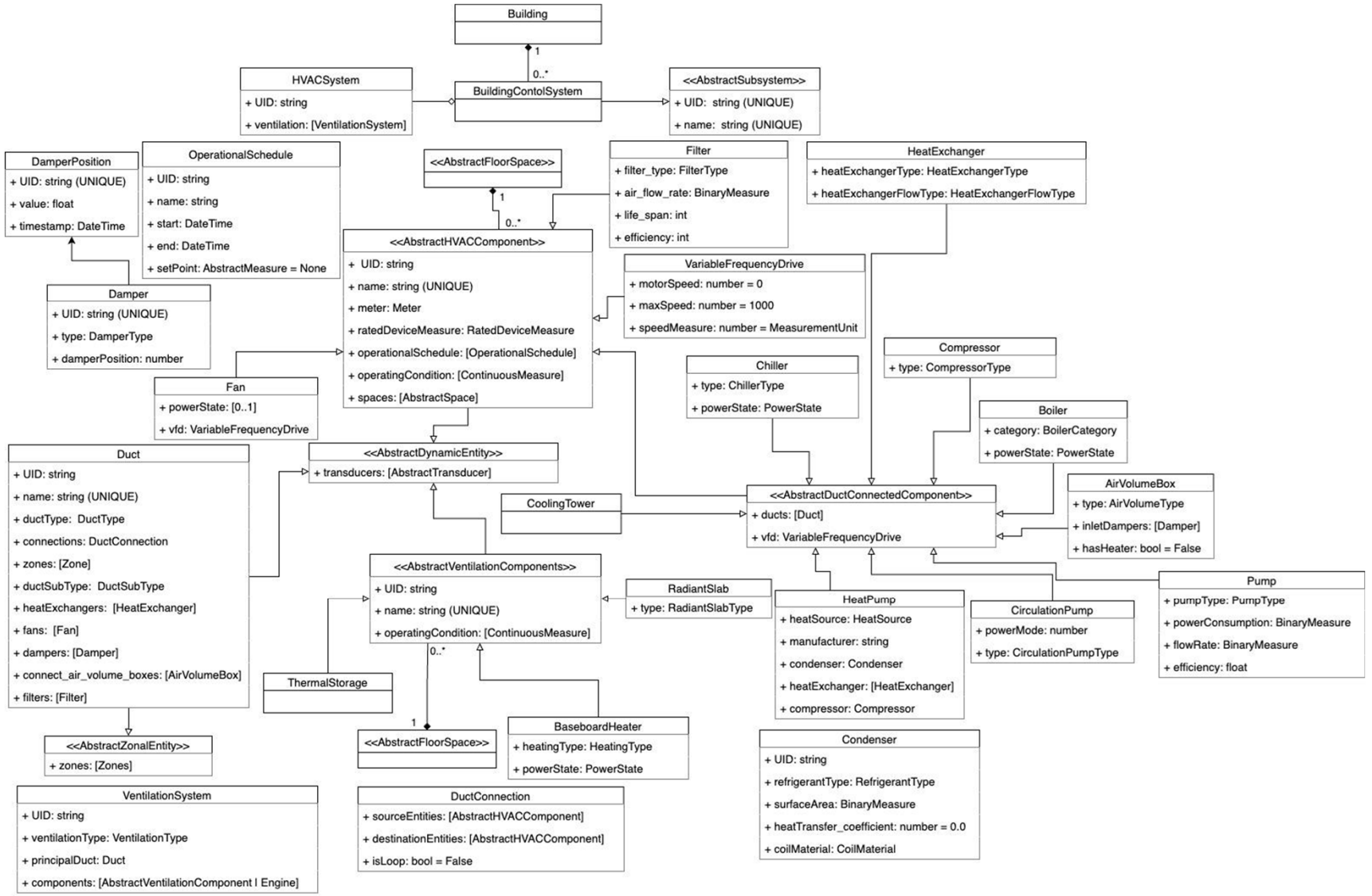


RQ2: Practitioners and Researchers Interactions with BMS APIs

Assessment

- Most of APIs endpoints do not use PATCH and PUT correctly
- Only EnteliWEB conforms to data and URI formats specified in the ASHRAE Addendum 135-2012am
- APIs may partially conform to security specifications of ASHRAE Addendum 135-2012am
- APIs do not always use the appropriate HTTP error codes, e.g., Metasys uses 405 PATCH requests that try to discard already discarded audits

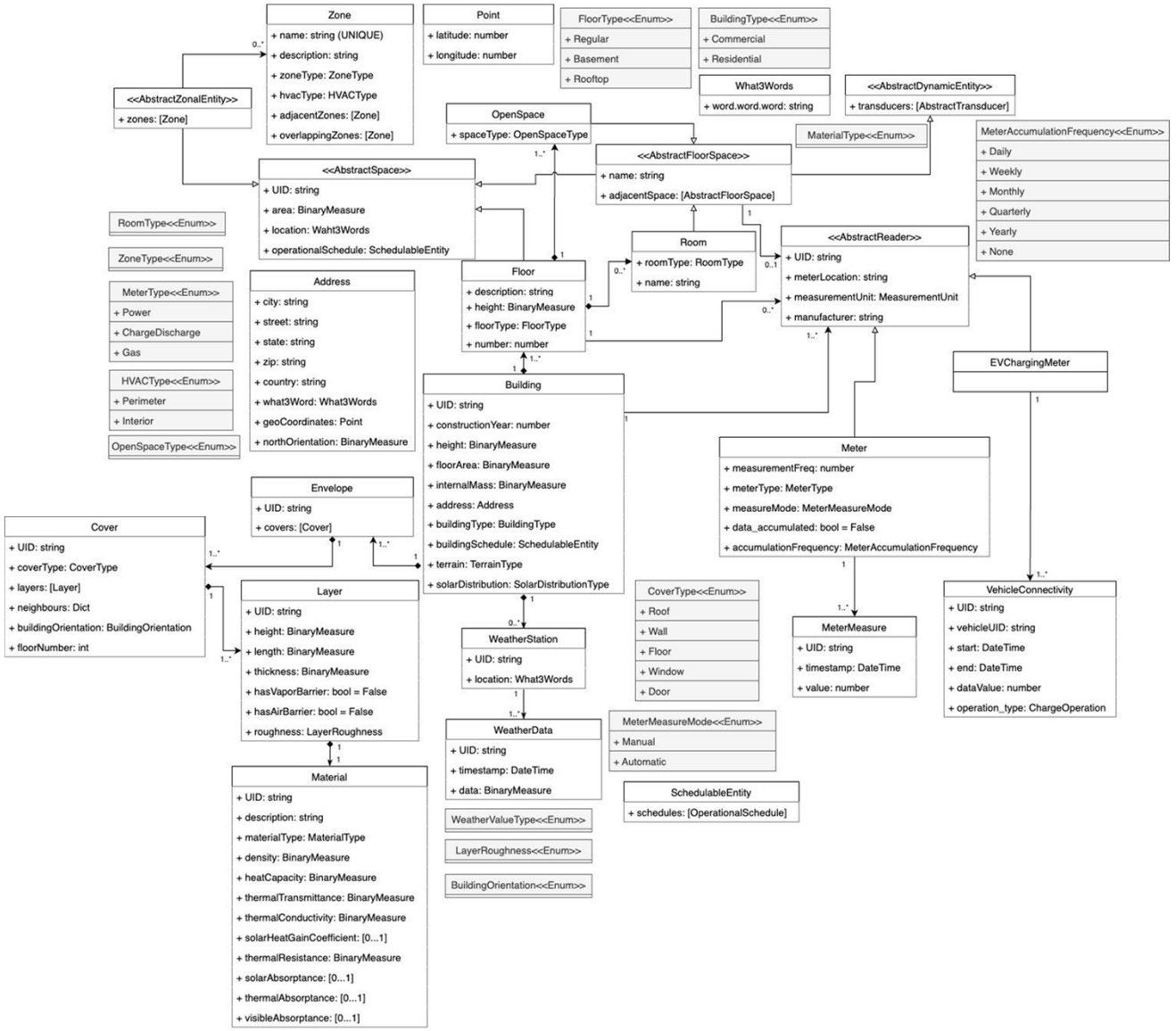
Metamodel for Energy Things



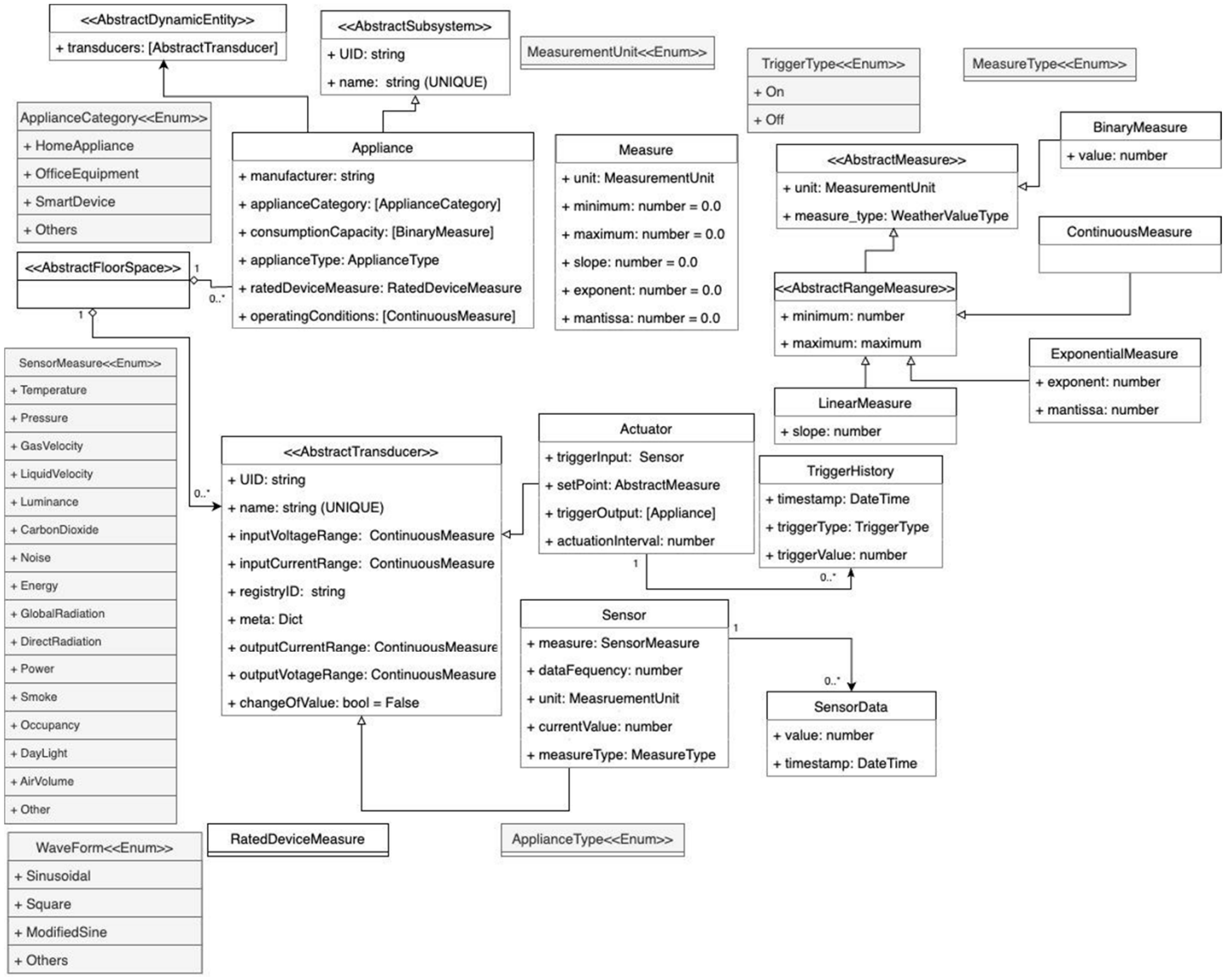
Varennes Library Use Case

```
57              "rooms": [
58                  {
59                      "name": "Room 001",
60                      "room_type": "storage",
61                      "area": {"value": 25.99, "unit": "square feet"},
62                      "sensors": [
63                          {
64                              "name": "CO2_SENSOR_001",
65                              "log_type": "Polling",
66                              "meta_data": {},
67                              "data": [],
68                              "measure": "carbon dioxide concentration",
69                              "measure_range": {"minimum": 0, "maximum": 2000},
70                              "unit": "ppm",
71                              "data_frequency": 600
72                          },
73                          {
74                              "name": "TMP_SENSOR_001",
75                              "log_type": "Polling",
76                              "measure": "temperature",
77                              "measure_range": {"minimum": -40, "maximum": 150},
78                              "input_voltage_range": {
79                                  "minimum": 0,
80                                  "maximum": 10,
81                                  "unit": "volts"
82                              },
83                              "output_voltage_range": {
84                                  "minimum": 0,
85                                  "maximum": 10,
86                                  "unit": "volts"
87                              },
```

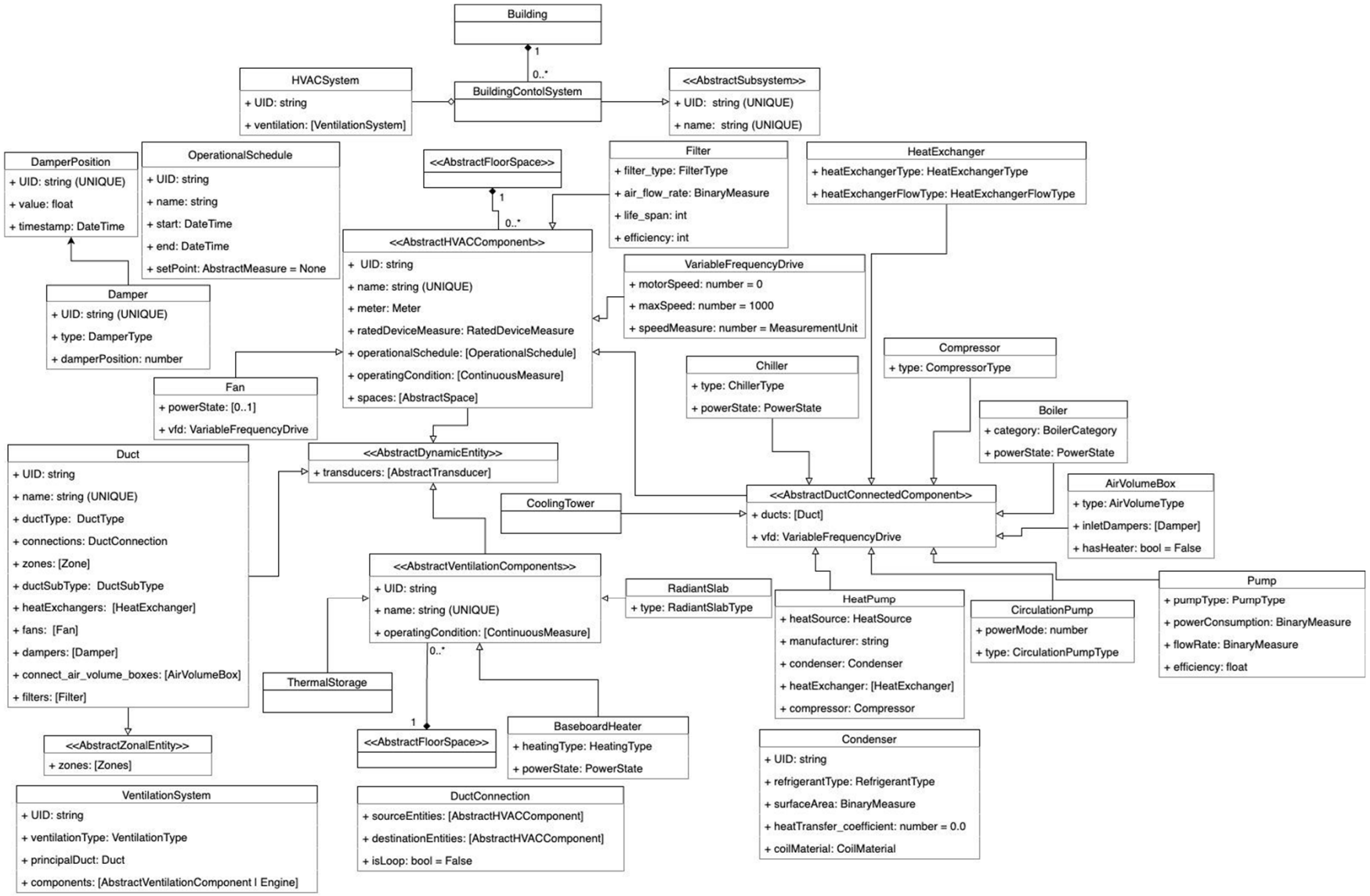
Metamodel for Energy Things: Classes



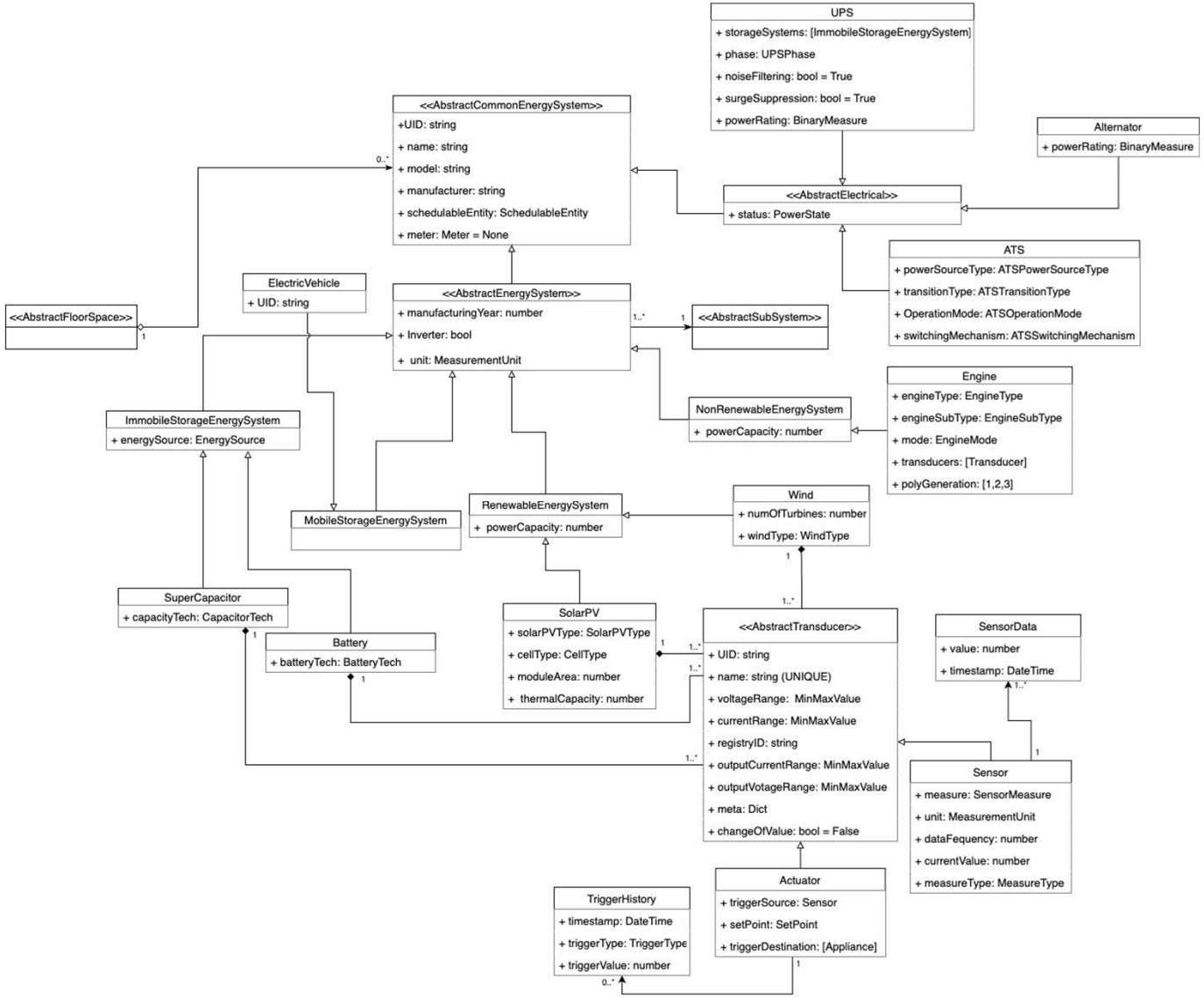
Metamodel for Energy Things: Appliance



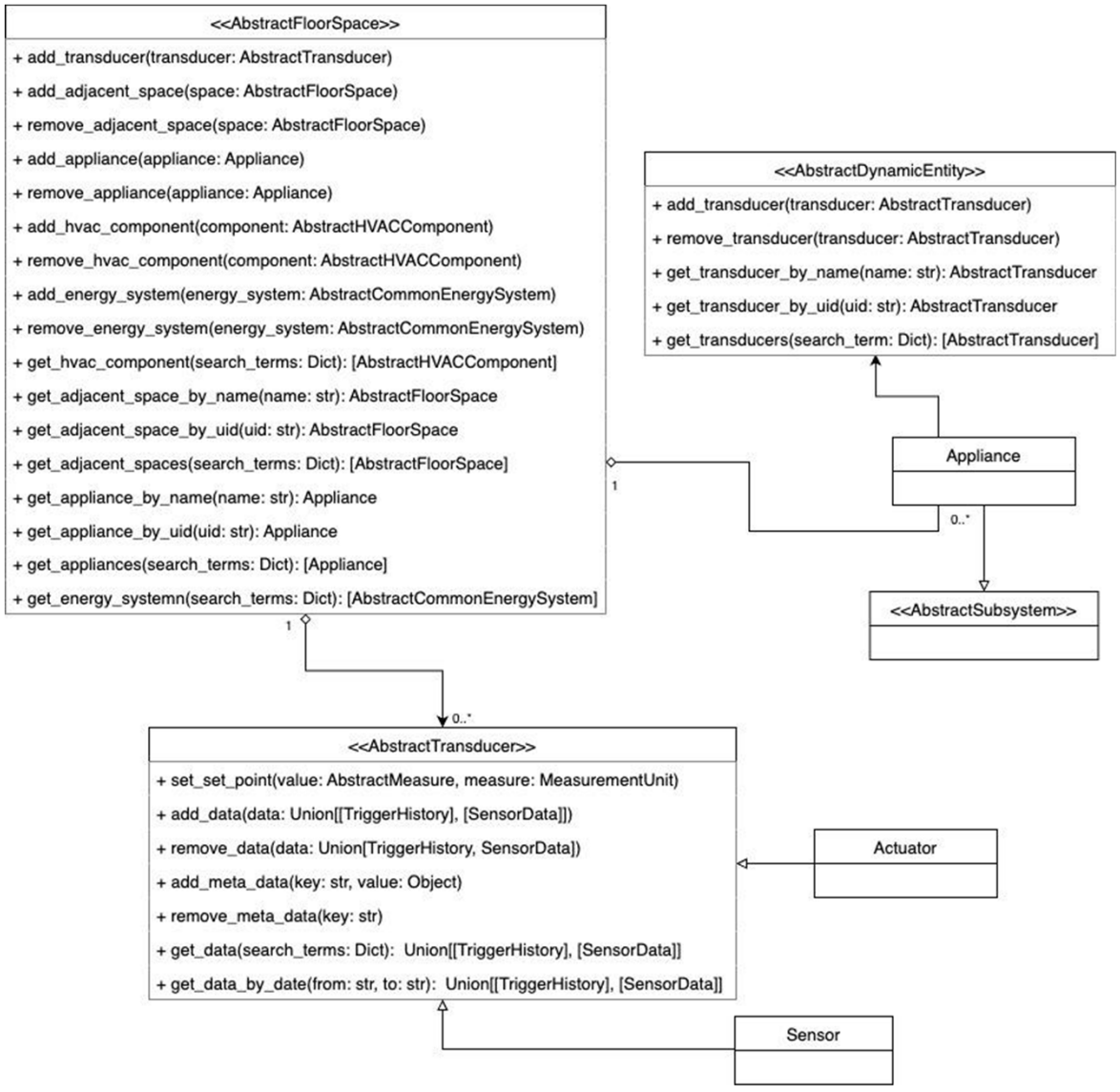
Metamodel for Energy Things: Control System



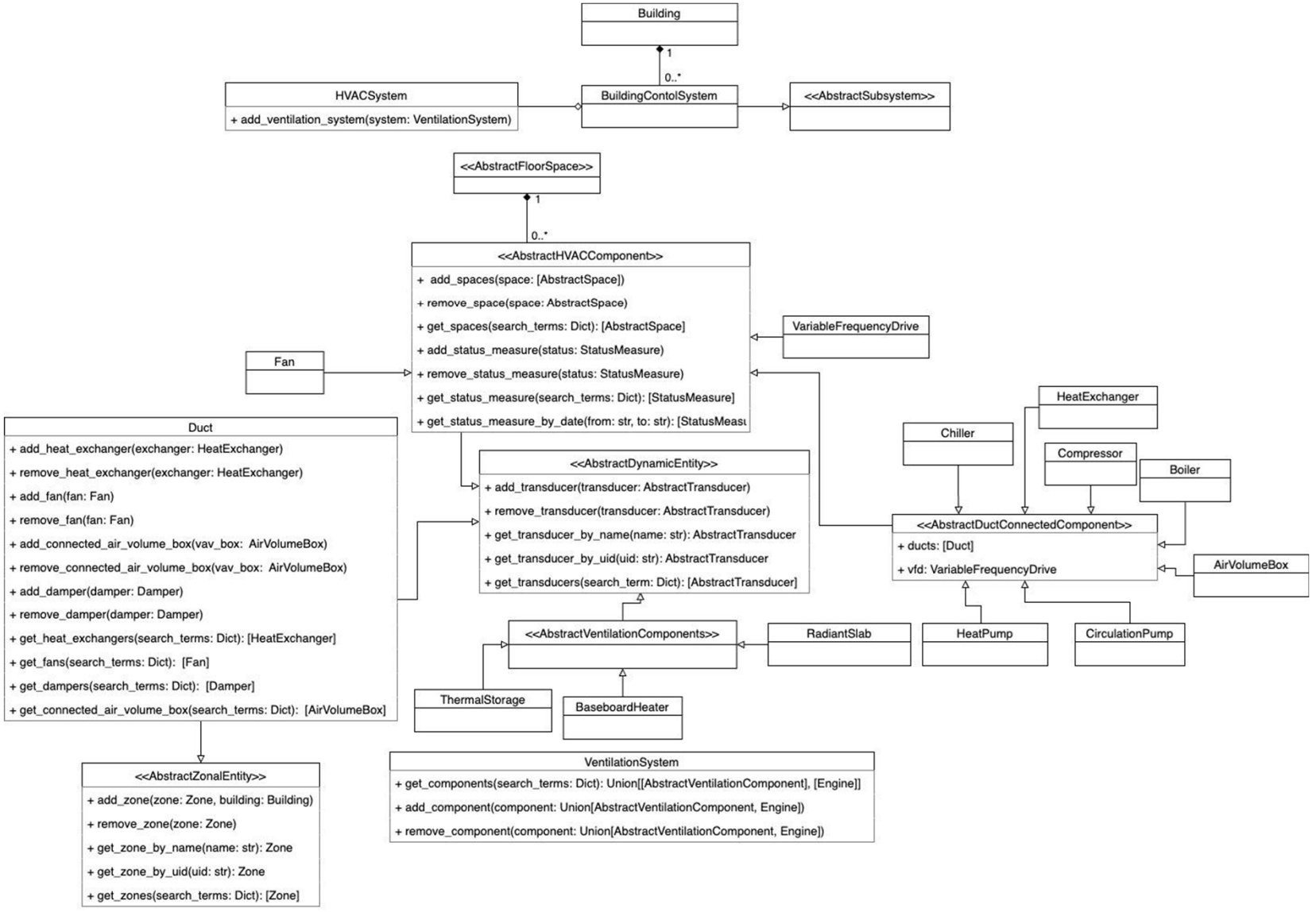
Metamodel for Energy Things: Energy System



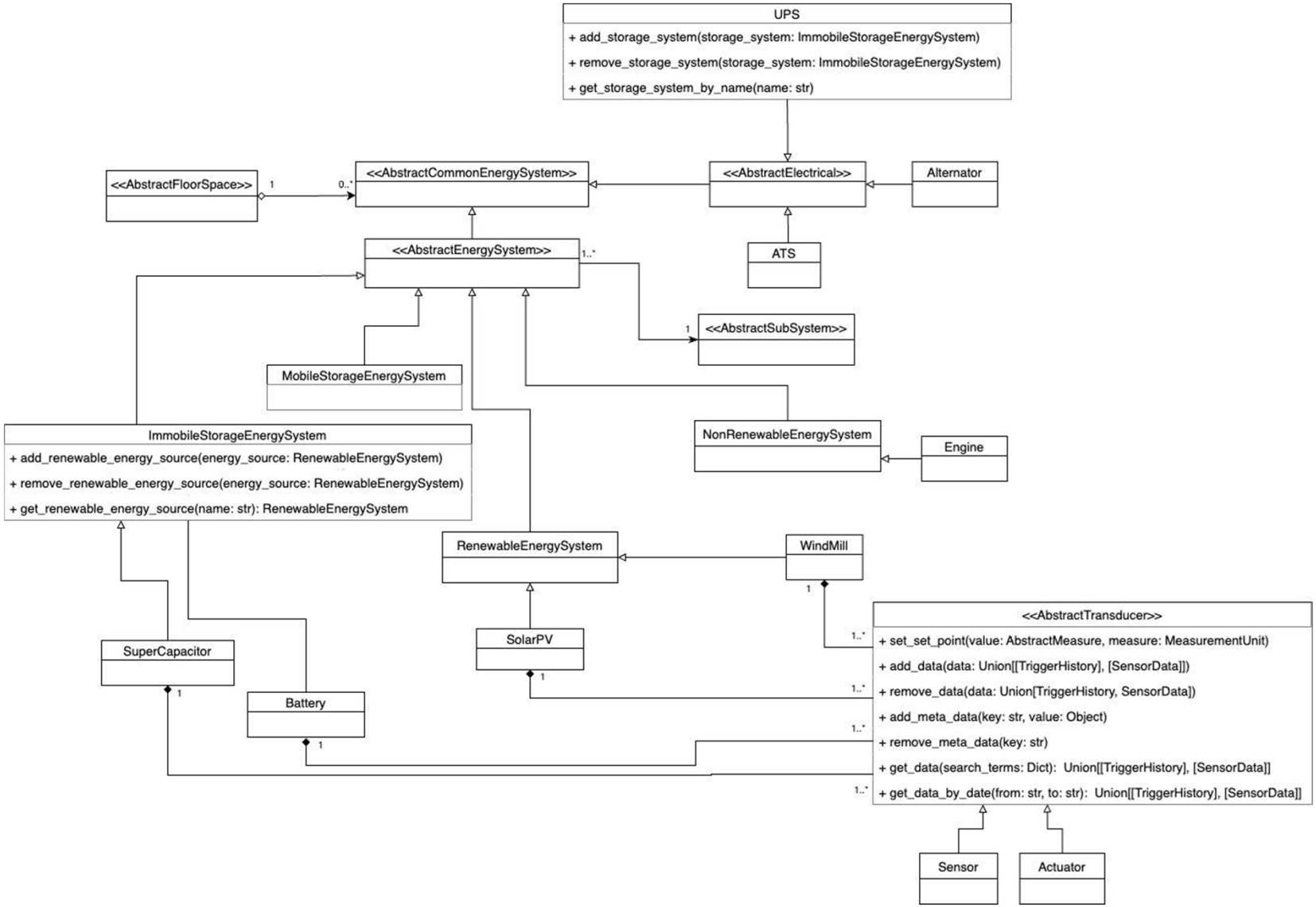
Metamodel for Energy Things: Methods



Metamodel for Energy Things: Methods

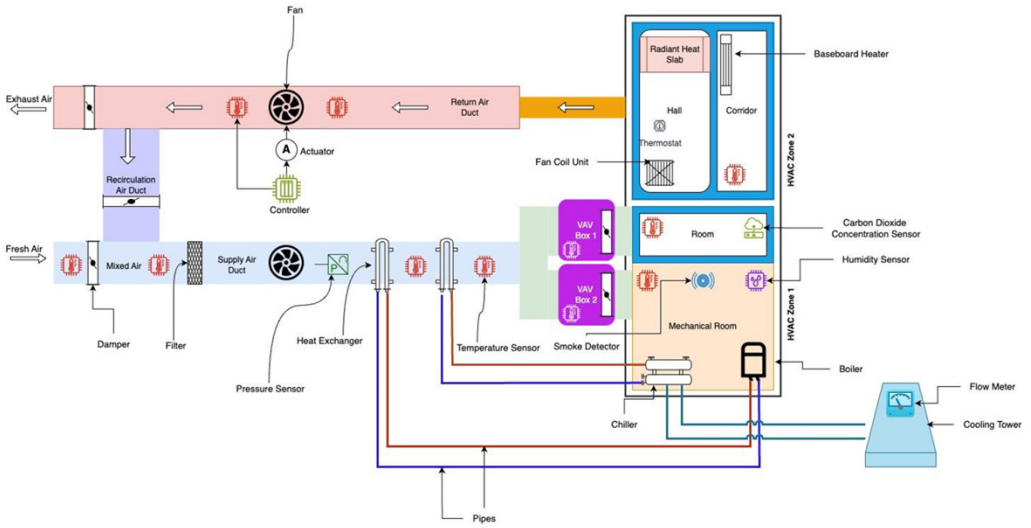


Metamodel for Energy Things: Methods



Metamodel for Energy Things: Example

```
bcs = BuildingControlSystem("EV Control System")
hvac_system = HVACSystem()
bcs.hvac_system = hvac_system

# Create the fresh air duct for the ventilation system
supply_air_duct = Duct("SUPP.VNT.01", DuctType.AIR)
supply_air_duct.duct_sub_type = DuctSubType.FRESH_AIR

# create the return air duct for the ventilation system
return_air_duct = Duct("RET.VNT.01", DuctType.AIR)
return_air_duct.duct_sub_type = DuctSubType.RETURN_AIR

# create the recirculation air duct
recirculation_air_duct = Duct("REC.VNT.01", DuctType.AIR)
recirculation_air_duct.duct_sub_type = DuctSubType.RETURN_AIR

# connect the fresh air, return air and recirculation air ducts
duct2duct_conn = DuctConnection()
# add the return air duct as the source of the connection
duct2duct_conn.add_entity(DuctConnectionEntityType.SOURCE, return_air_duct)
# add the fresh air duct as the destination of the connection
duct2duct_conn.add_entity(DuctConnectionEntityType.DESTINATION, supply_air_duct)
# set the connection object as a property of the recirculation air duct
recirculation_air_duct.connections = duct2duct_conn

# set the recirculation air duct as the source for the supply air duct
supp_duct_conn = DuctConnection()
supp_duct_conn.add_entity(DuctConnectionEntityType.SOURCE, recirculation_air_duct)
supply_air_duct.connections = duct2duct_conn

# create the ventilation system with the supply air duct as the principal duct
ventilation_system = VentilationSystem(VentilationType.AIR_HANDLING_UNIT, supply_air_duct)

# set the ventilation system for the HVAC system
hvac_system.add_ventilation_system(ventilation_system)
```



BMS API Capabilities and their Practical Usage: RQ2

TABLE V: Selected endpoints of the mocked BMS APIs, their HTTP verbs, responses, and HTTP status codes. The responses and status code are limited to successful HTTP requests.

BMS	API Category	Description	Endpoint	Response	Status Code	HTTP Verb
Metasys	Objects	List all objects	`POST` /objects	Object list	200	POST
		Create object	`POST` /objects	Location header	200	POST
		Get object	`GET` /objects/{objectId}	Object	200	GET
		Send command	`PUT` /objects/{objectId}/commands/{commandName}	Success	200	PUT
		Edit object	`PATCH` /objects/{objectId}		200	PATCH
		Delete object	`DELETE` /objects/{objectId}		204	DELETE
EnteliWEB	Delta Idot Bacnet	Get object list	`GET` /.bacnet/{siteName}/{deviceNumber}	Object list	200	GET
		Write object value	`PUT` /.bacnet/{siteName}/{deviceNumber}/{objectType},{instance}/{propertyName}	Message object	200	PUT
		Delete object	`DELETE` /.bacnet/{siteName}/{deviceNumber}/{objectType},{instance}	Message object	203	DELETE
	Delta Idot Multi	Create multi	`POST` /.multi	Multi object (with error message)	201	POST
		Write object present value	`POST` /.multi	Multi object (with error message)	200	POST
	Delta Idot System	Write multiple system properties	`PUT` /systems/{systemName}	Property object (with error message)	200	PUT
Desigo Building X Openness	Building Operations	Get all devices	`GET` /devices	Device list	200	GET
		Create device	`POST` /devices	Device object	201	POST
		Update device attributes	`PATCH` /devices/{deviceId}		202	PATCH
		Update point	`PATCH` /points/{pointId}		204	PATCH
		Update point tags	`PUT` /points/{pointId}/tags		204	PUT
	Building Structure	Delete location	`DELETE` /locations/{locationId}		204	DELETE

RQ3.1: BMS APIS Capabilities

TABLE III: APIs of five (5) BMS vendors and their capabilities grouped into twelve (12) categories

Criteria/BMS	Metasys	EnteliWEB	EcoStruxure	Desigo	EBI
Alarms and Events	List events and alarms. Subscribe and unsubscribe to alarms. Perform batch operation on events. Update alarms (to discard or acknowledge them)	Get alarms and notifications. Create alarm details and categories. Get event notifications. Acknowledge event notifications. Get alarm details for event notifications	Get all alarms. Get (ranked and binned) alarm occurrences. Acknowledge alarms. Create and update alarm instances and occurrences. Manage events (CRUD). Get alarm transitions	Retrieve all events and their transition histories. Create, get and update alarm configurations. Delete alarm configuration.	N/A
Audits	List all audits. Subscribe and unsubscribe to audits. List audit users. Add audit annotations. Edit to discard audits	No endpoints for audits, however, there are endpoints to get all event and trend logs	N/A	N/A	N/A
Equipment	Retrieve equipment hosted by network devices. Retrieve equipment serving other equipment or spaces. Retrieve equipment points	Get all systems. Retrieve system properties. Write property values of a system	Get asset hierarchies with parent-child relationships. Get equipment associated with network devices. Get equipment of a particular type.	List all equipment. Create and update equipment. Delete equipment.	N/A
Network Devices	List network devices serving a space. List children of network devices. Delete offline child network devices	Get a list of node types. Get a list of protocol nodes. Get a list of network nodes. Get a list of device nodes	Get communication device hierarchies	List all devices. Create and update devices. List devices behind gateways.	N/A
Objects	Manage (CRUD) objects. List object attributes. List commands an object supports. Send commands to an object. Perform batch operations on objects. List object points	List object nodes. List objects of a device. Get object lists and properties. Get object property values. Write object properties (commands). Create and delete objects. Get the value of multiple object properties using multi	Retrieve a list of all objects. Get object properties. Update object properties (commands).	Get a list of all objects. Update object values (commands). Add object values. Create, get and update groups of points.	Get objects and properties
Space	Get all spaces. Get spaces served by network devices. Get spaces within other spaces	Get a list of all sites (buildings)	N/A	List buildings in a partition (e.g., campus). List building parts (e.g., rooms). Create location hierarchy (e.g., room on a floor). Get, update, and delete a location. Create, get, update and delete addresses. Get a 2D geometry of floors.	N/A

APIs of five (5) BMS vendors and their capabilities grouped into twelve (12) categories - continued from the previous page

Criteria/BMS	Metasys	EnteliWEB	EcoStruxure	Desigo	EBI
Time-series Data	Get data for objects. Get network device attributes that have time-series data. Get data for network device attributes	Get log records from trend log objects. Get a list of historical or trend log objects	Get device time-series data. Get aggregated time-series data. Get binned energy usage data (value and cost)	Get time-series data for points. Add point values.	Retrieve time-series data for points
Energy	N/A	N/A	Get energy production and consumption data. Get energy intensity data. Get energy usage cost data. Get meters and their measurements. Get active energy data. Get utility (gas, water flow rate) data	List all medium (space and equipment) consumption. Read consumption per location or meter. Read consumption cost per location or meter	N/A
Carbon Dioxide Emissions	N/A	N/A	Get aggregated carbon dioxide data (requires emission factor configuration)	Read emission data per location	N/A
Assets	N/A	N/A	Get asset details, including service level, maintenance, and health indexes. Get asset tickets. Create, get, update and delete asset ticket subscriptions. Create, get, update and delete asset health subscriptions. Create, get, update and delete site risk-level subscriptions	Manage (get and update) asset locations and their relationships	N/A
Energy LEED Scoring	N/A	N/A	Get a building performance score. Create performance scoring requests. Request performance scoring status	N/A	N/A
Energy Modelling	N/A	N/A	Create an energy model. Apply the existing model to predict building energy consumption. Assess model quality savings	N/A	N/A

TABLE VI: Comparison of Brick and Project Haystack Classes and Tags for Representing BMS Data Ac Across Twelve Functional Categories

BMS API Category	Brick	Project Haystack
Alarms and events	Alarm class (which extends the Point class). Multiple Alarm types, e.g., Air_Alarm, C02_Alarm, Failure_Alarm, Cycle_Alarm, Humidity_Alarm, etc. Event class (from RealEstateCore)	Alarm tag for conditional notification. The standard event tag, however, a custom tag to events can be included
Audits	Does not possess classes for BMS audits (e.g., compliance audits) but may have classes to model audit data generated by building systems, e.g., energy usage data. Agent class could model building interactions with users, which is important in compliance audits	Does not possess classes for BMS audits (e.g ance audits) but may have classes to model generated by building systems, e.g., energy us
Equipment	Equipment class with multiple equipment types, e.g., ICT_Equipment, Elevator, Fire_Safety_Equipment, Meter, PV_Panel, Lighting_Equipment, etc	Equip tag with multiple subtags, e.g., actuato dlingEquip, airTerminalUnit, ates, battery, boi chiller, circuit, coolingTower, etc.
Network Devices	ICT_Equipment class with multiple types, e.g., AudioVisualEquipment, Controller, Gateway, ICTHardware, ITRack, and SensorEquipment	Device, network, networking-device, network and networking-switch tags
Objects	Does not have a direct class for BACnet objects but has classes for the entities modelled as BACnet objects. For example, the Point class can model point objects like sensors and setpoints. The Command class, which extends the Point class, can model command objects	Does not have direct tags for BACnet objects, tags for the entities modelled as BACnet ob example, the point class can model data poin sensors or actuators and cmd tags for comma
Space	Space class (from RealEstateCore) with multiple types, e.g., Architecture, Region, and Wing. The Architecture class has subclasses such as Building, Room, Level, Site, OutdoorSpace, Zone, and SubBuilding	The space tag has multiple subtags, such as d floor, room, and zone-space. The Floor has sul as the ground-floor, roof-floor and subterranea
Time-series Data	Sensor classes (e.g. C02_Sensor) have attributes for lastKnownValue and aggregate of data. It is not meant to store multiple time-series data	curVal tag for current values of points. unit t the unit of measurement. The curStatus tag the status of a point's value. currErr indicate condition. The his tag is used for historical hisMode indicates the way historical data is c
Energy	The Meter Class measures usage and consumption of electricity with properties like measuredPowerInput, measurePowerOutput, ratedCurrentOutput, ratedCurrentInput, ratedVoltageInput, ratedVoltageOutput, ratedPowerInput, ratedPowerOutput, etc.,	The meter, electric-meter, ac-electric-meter, d meter, and equip tags models a kind of meter. be used with the point and unit tags to measu related phenomena.
Carbon Dioxide Emissions	The Outside_Air_C02_Sensor class has properties like lastKnownValue and aggregate (for aggregated emissions)	co2 and point tags with quantities tags such a tration, flow, and level to measure emissions