Demo Abstract – Portable Queries Using the Brick Schema for Building Applications

Bharathan Balaji[1], Arka Bhattacharya[2], Gabe Fierro[2], Jingkun Gao[3], Joshua Gluck[3], Dezhi Hong[4], Aslak Johansen[5], Jason Koh[6], Joern Ploennigs[7], Yuvraj Agarwal[3], Mario Berges[3], David Culler[2], Rajesh Gupta[6], Mikkel Baun Kjærgaard[5], Mani Srivastava[1], Kamin Whitehouse[4]

[1]UCLA, [2]UC Berkeley, [3]Carnegie Mellon University, [4]University of Virginia, [5]University of Southern Denmark, [6]UC San Diego, [7]IBM Research - Ireland

ABSTRACT

Sensorized commercial buildings are a rich target for building a new class of applications that improve operational and energy efficiency of building operations that take into account human activities. Such applications, however, rarely experience widespread adoption due to the lack of a common descriptive schema that would enable porting these applications and systems to different buildings. Our demo presents *Brick* [4], a uniform schema for representing metadata in buildings. Our schema defines a concrete ontology for sensors, subsystems and relationships among them, which enables portable applications. Using a web application, we will demonstrate real buildings that have been mapped to the Brick schema, and show application queries that extracts relevant metadata from these buildings. The attendees would be able to create example buildings and write their own queries.

CCS Concepts

•**Information systems** → **Ontologies;** *Process control systems;*
•**Computer systems organization** → **Sensors and actuators;**

Keywords

Smart Buildings, Building Management, Schema, Ontology

1. INTRODUCTION

As of 2012, 14% of the commercial buildings in the U.S. deployed Building Management System (BMS) [3] to manage data collection and remote actuation of the connected building infrastructure. Innovations in "Internet of Things" (IoT) devices have led to connected lights, power meters, occupancy sensors and appliances that are capable of interfacing with BMSes. New networked devices thus present an opportunity for a building "applications plane" to provide new capabilities to building operators and occupants alike. However, as a platform, even the most modern BMS present a cacophony of data and information flows that vary by buildings, vendors and across locations. The lack of a common data representation prevents interoperability between buildings and limits scalability of applications as developers need to map the heterogeneous data of each building to a common format.

We designed the Brick schema [4] to capture the essential building metadata required by smart building applications. Our design is guided by metadata incorporated in BMSes across a heterogeneous set of buildings and canonical applications published in the literature [6]. In our demo, we demonstrate the efficacy and usability of Brick with a web application that allows attendees to write queries on real buildings that have been mapped to our schema. Our demo will show how the existing buildings have been mapped to Brick, and smart building application queries that work on these buildings without modifications. Here we present a brief summary of the Brick schema, and provide details about the buildings and application queries that will be presented in our demo.

BuildSys '16 November 16-17, 2016, Palo Alto, CA, USA

ACM ISBN 978-1-4503-4264-3/16/11.

DOI: http://dx.doi.org/10.1145/2993422.2996411

2. BRICK SUMMARY

The Brick schema has three parts: (i) a class hierarchy of entities describing the various building subsystems and their entities; (ii) a minimal and principled set of relationships for connecting these entities into a directed graph describing the structure of the building; (iii) a method of encapsulation in the form of Function Blocks to abstract away the details of complex equipment and control loops.

Brick builds on the concept of *tags* from Project Haystack [1] and enriches the schema with an underlying ontology that crystallizes the concepts defined by the tags. We introduce the concept of a *tagset* that groups together relevant tags to represent entities such as a temperature sensor or an HVAC equipment. The concept of tagsets works well with an ontology class hierarchy - a `room temperature sensor` is a subclass of a generic `temperature sensor`, and will inherit its properties.

We introduce *Point* as a class, with subclasses defining specific types of points: Sensor, Setpoint, Command, Status, Alarm. Each point can have several attributes, which we divide into *properties* and *relationships*. Properties are attributes like units and data type, while relationships link entities: a sensor is associated with its location, equipment it belongs to, and so on. We define three concepts as high level classes to which a Point can be related to: Location, Equipment and Measurement (Figure 1). Each concept has a class hierarchy to concretely identify each entity in the building. For example, the Equipment class has subclasses HVAC, Lighting and Power, each of which has its own subclasses.

We have designed Brick relationships to be minimal, multipurpose and intuitive so that a user can easily specify them. For ex-

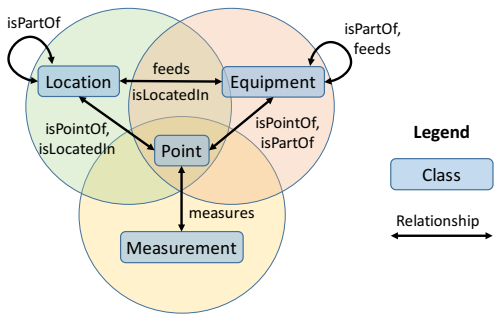

Figure 1: Information concepts in Brick and their relationship to a data point.

```
example:myVAV rdf:type brick:VAV
example:myTempSensor rdf:type brick:Zone_Temperature_Sensor
example:myVAV brick:hasPoint example:myTempSensor
```

Figure 2: RDF triples instantiating a VAV and a Temperature Sensor and declaring that the VAV measures temperature via that sensor.

ample, the `isPartOf` relationship is designed to capture the compositions among the entities in the building – a room `isPartOf` a floor. The `feeds` relationship captures the different *flows* in the building,e.g., the flow of air from AHU to VAV.

Buildings equipment and points are often grouped by multiple logical views such as control view. We use *Function Blocks* to encapsulate details of such logical groups that expose an interface through named inputs and outputs. These are defined through `isInputOf` and `isOutputOf` relations to the particular function block acting as context. Function Blocks may encapsulate other Function Blocks via the `isPartOf` relation.

Brick adheres to the RDF (Resource Description Framework) data model, which represents knowledge as a graph expressed as tuples of *subject-predicate-object* known as *triples*. For example, the collection of triples in Figure 2 gives the representation of the connection of an equipment called VAV to a Temperature Sensor using the `hasPoint` relationship. Applications query the Brick graph for entities and relationships using SPARQL [2]. SPARQL queries specify constraints and patterns of triples, and traverse an underlying RDF graph to return those that match.

3. BUILDINGS AND APPLICATIONS

Our demo will showcase five buildings across the US and Europe whose metadata have been mapped to Brick. These buildings vary by age, climatic conditions, equipment installed and their BMS vendors. We effectively map the metadata of these buildings to the Brick schema. To show that applications running on these buildings are indeed portable, we pick eight canonical applications from the literature [6] and demonstrate that their application queries work on all the five buildings. Table 1 summarizes the results of our application queries across these buildings.

For example, ZonePAC [5] incorporates monitoring and modeling of HVAC zone behavior and power usage with occupant feedback to provide a platform for occupants to directly contribute to the efficacy and efficiency of a building's HVAC system. The application requires metadata mapping the HVAC VAV unit with its corresponding rooms and airflow sensors. Figure 3 shows the 11 line SPARQL query that can be used to obtain this metadata for any building mapped to Brick. Our demo will consist of similar app queries, and attendees can create their own queries as well.

	Building					
Application	EBU3B	GTH	GHC	IBM	Rice	Soda
Occupancy	244	139	366	821	11	232
Energy Apportionment	-	302	-	397	4	-
Web Displays	697	81	65	835	106	513
MPC	482	69	428	324	110	482
Participatory Feedback	-	253	-	386	-	-
FDD	229	12	229	728	-	136
NILM	6	82	-	1348	-	-
Demand Response	1428	24	2490	608	4	144

Table 1: Number of matching triples in each building for the SPARQL queries across eight applications. Entries with '–' indicate lack of relevant points in the building. For more details please refer to [4].

```
SELECT ?airflow_sensor ?room ?vav
WHERE {
  ?airflow_sensor rdf:type/rdfs:subClassOf*
    brick:Supply_Air_Flow_Sensor .
  ?vav rdf:type brick:VAV .
  ?room rdf:type brick:Room .
  ?zone rdf:type brick:HVAC_Zone .
  ?vav brick:feeds+ ?zone .
  ?room brick:isPartOf ?zone .
  ?airflow_sensor brick:isPointOf ?vav .
}
```

Figure 3: ZonePAC query for airflow sensors and rooms for VAVs. The query returns all relevant triples for ZonePAC to bootstrap itself to a new building.

Acknowledgments

Our work is supported by the following grants – National Science Foundation grants: CPS-1239552, NSF-1636879, IIS-1636916, CSR-1526237, CNS-1526841, NSF-1305362; U.S. Department of Energy grant: DE-EE0006353; King Abdullah University of Science and Technology award: Sensor Innovation Award #OSR-2015-Sensors-2707; Innovation Fund Denmark award: COORDICY(4106-00003B); EU H2020 grant: TOPAs (676760); and support from Intel.

4. REFERENCES

[1] Project Haystack. http://project-haystack.org/.

[2] SPARQL Query Language. https://www.w3.org/TR/rdf-sparql-query/.

[3] U. E. I. Administration. User's guide to the 2012 cbecs public use microdata file. *Commercial Buildings Energy Consumption Survey (CBECS)*, page 33, May 2016.

[4] B. Balaji, A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, J. Ploennigs, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M. Kjaergaard, and K. Whitehouse. Brick v1.0 - towards a unified metadata schema for buildings. In *BuildSys*. ACM, 2016.

[5] B. Balaji, H. Teraoka, R. Gupta, and Y. Agarwal. Zonepac: Zonal power estimation and control via HVAC metering and occupant feedback. In *BuildSys*, pages 1–8. ACM, 2013.

[6] A. Bhattacharya, J. Ploennigs, and D. Culler. Short paper: Analyzing metadata schemas for buildings: The good, the bad, and the ugly. In *BuildSys*, pages 33–34. ACM, 2015.