Enabling Scalable Smart-Building Analytics

Arka Bhattacharya

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-201
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-201.html

December 15, 2016

Enabling Scalable Smart-Building Analytics

by

Arka Bhattacharya

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor David Culler, Chair
Professor Randy Katz
Professor Edward Arens

Fall 2016

The dissertation of Arka Bhattacharya, titled Enabling Scalable Smart-Building Analytics, is approved:

Chair _____ Date _____

_____ Date _____

_____ Date _____

University of California, Berkeley

Enabling Scalable Smart-Building Analytics

Abstract

Enabling Scalable Smart-Building Analytics

by

Arka Bhattacharya

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor David Culler, Chair

Modern buildings are being integrated with myriad (often >1000s) networked sensors to improve convenience, occupant comfort accessibility and energy-efficient operations. These technological improvements hold the promise of significant advances in centralized operation and management, fault diagnosis, and integration to an emerging smart grid. As of 2012, 14% of the buildings in the U.S. deployed Building Management Systems (BMS) to provide some kind of programmatic interface to the the sensors, actuators, and historical data management. Innovations in "Internet of Things" (IoT) devices have further led to connected lights, power meters, occupancy sensors and appliances that are capable of interfacing with the underlying BMS systems used in building automation. New buildings are installed with a BMS by design, and older buildings are being continuously retrofitted with networked systems for improved efficiency.

However, whether provided by novel sensor networks or legacy instrumentation, extracting meaningful information from sensor data and taking actions based on that data depends fundamentally on the metadata available to interpret it. There are more than 5 million commercial buildings in the US, with the sensors in each building set up with customized and obscure metadata. One cannot achieve scalable deployments of software analytics and applications across buildings if deploying them requires vendors and domain experts spending spending 100s of hours fixing each building. Today even well-established applications do not get deployed at scale because of this very reason. Thus, the major challenge is scalability , i.e a paradigm where an application can be written once and deployed on 100s or 1000s of buildings.

This thesis evaluates the challenges with existing metadata of sensors in smart-buildings and proposes ways to normalize it to uniform standard that would allow scalable (write-once and deploy everywhere) application development. We develop three empirical criteria for successful metadata schemas — (a) completeness, or the ability to capture all sensors, (b) ability to capture all relationships between sensors required by state-of-the-art applications, and (c) flexibility in incorporating novel sensors and applications, and usability. We empirically demonstrate that no existing smart-building sensor metadata schema satisfy these properties and develop a schema based on an underlying graphical data model, Brick, that does. We validate Brick across 6

large and diverse commercial buildings (comprising more than 17,000 sensors) in two different continents and set up by different BMS vendors.

We also develop a human-in-the-loop synthesis technique which uses syntactic and data-driven steps to parse legacy metadata into a common schema. This technique allows building-experts, who might not be conversant with sophisticated regular expression programs, to parse more than 70% of the legacy metadata in a building to a common schema by providing example parses of only about 1% of the sensors. We also show how to use active perturbations of subsystems in a building to construct functional relationships between subsystems that may have been missing or incorrectly captured in legacy metadata with an accuracy of 80%. Finally, we demonstrate the power of our normalized smart-building metadata schema paradigm by using the standardized sensor and relationship representations to implement both simple (e.g. finding errant zones, identifying inefficient air handling subsystems to save energy) and sophisticated applications (e.g finding rough occupancy estimates) that scale across real-world smart-buildings.

Contents

List of Figures

List of Tables

To Mom and Dad

Acknowledgments

My graduate school journey at UC Berkeley has been an incredible experience. I have had the good fortune of engaging with and learning from myriad people who were all exceedingly smart, wise and helpful in equal measure. This thesis would not have materialized without their help and I would like to take this opportunity to thank a few of them.

First and foremost I would like to thank David Culler, who has been my advisor and mentor throughout my stay in Berkeley. He was always incredibly patient with me, affording me the time and freedom to explore and grow as a researcher and as a person. David was an invaluable resource in helping me learn the art of scoping, framing and making progress on seemingly open-ended problems. I have tried to learn from his remarkable ability to reason through large complex systems, his expertise across almost every area of computer science, and his ability to quickly identify the crux of an idea and apply it to problems in completely different contexts. I rarely had a meeting with David from which I did not leave more invigorated and with many more ideas than I had gone in with.

I would also like to acknowledge Randy Katz, Ed Arens and Ras Bodik for being the nicest and most helpful committee members, helping me focus on the right questions and always giving me great suggestions. Thanks to Kamin Whitehouse, Yuvraj Agarwal and Eugene Wu for helping make some of the ideas in this thesis more concrete. I am especially thankful to Aman Kansal for not only getting me started down the research path when I was an undergraduate student, but also being the best collaborator I could have hoped for during my first couple of years in graduate school.

I would like to thank the old RADLab/AMPLab folk for the frequent fun discussions, and always inspiring me with their brilliance — Matei, TD, Zats, Radhika, Shivaram, Sameer, Mosharaf, Rachit and Vyas. I will be eternally grateful to Ganesh, Ali and Prashanth for being role-models, friends and invaluable mentors, often going out of their way to help me find my footing during my first few years in Berkeley. Panda, a veritable human encyclopedia, was always my go-to resource for help and advice when Google failed; I am thankful to him for being so generous with his time. I would also like to thank my academic "brothers" — Jay, Steve, Andrew and Jorge for taking me under their wing, and demonstrating that research could be both intellectually challenging and societally meaningful! Thanks to the 410 crew — Gabe, Michael, Shankari, Kaifei and Jack — for always keeping my spirits up and providing frequent constructive feedback. Special thanks to Albert for all the late-night chats, all the birthday cakes, and being almost super-naturally helpful and nice in every possible way (including teaching me a lot about cars). I also want to thank Harry Stark for allowing me to hack and play with the massive building under his charge. Thanks to Sara, Alex, Caroline, Marco, Omid, Bharathan, Jason, Dezhi and Joern for being great collaborators and teaching me so much along the way.

A special thank you to all my friends, many of whom made the transition from undergrad in the small, sleepy town of Kharagpur, India, to life in the bustling Bay Area alongside me — Kamble, Jog, Madhura, Kedia, Anuj, Debanjan, Bhandaru, Raj, Nishank, Dibyendu, Sabrina, Momo, Shaunak, Aastha, Ajith, Amit, Arindam, Shubham, Diptesh, Sayantan, Himanshu, De-

bkishore and Aamod. Navigating the new set of experiences together was thoroughly enjoyable and memorable.

Finally, I would like to thank my parents who have been constant pillars of support and guidance from halfway across the world — my dad, who was perennially anxious about my progress in grad school, balanced by my mom who helped me relax and see the bigger picture. My brother always seemed to have infinite faith in my abilities, and I thank him for providing a sense of safety that I knew I could always fall back on.

Berkeley is a weird and funny town, full of warm, interesting people, great food and great weather. I could not have asked for a better place or a better school to do my PhD. As this chapter of my life, sometimes frustrating but always rewarding, comes to a close I want to thank everyone (some of whom I might have forgotten to mention) who helped make this a wonderful journey.

Chapter 1

Introduction

1.1 Trends In Building Intelligence

There is a big trend towards making large commercial buildings more responsive, energy efficient and easy-to-maintain. There are about 5 million commercial buildings in the U.S alone, and they consume 19% of all U.S energy production. 90% of the consumed energy comes from non-renewable sources, making curtailment of building energy consumption a requisite step towards the worldwide effort to mitigate global warming and ensuring a more sustainable future. Also, an increase of renewable energy penetration into the electricity grid and a heightened emphasis on supply-following loads puts the onus on commercial buildings to be more responsive to signals from the grid and possibly curtail its energy usage on demand. Additionally, commercial buildings need to be responsive to occupant comfort and desires, since occupants typically spend 40-60 hours a week in them. Finally, commercial buildings typically contain large mechanical and electrical systems controlling its different aspects, viz. its environment, lighting, power, etc. These systems need constant monitoring and supervision to identify faults which may lead to inefficient energy-use or adversely affect occupant comfort.

To gain more insight into building operation, modern buildings are increasingly being integrated with a variety of networked sensors and equipment to improve convenience, accessibility and energy-efficient operations. These technological improvements hold the promise of significant advances in centralized operation and management, fault diagnosis, and integration to an emerging smart grid. As of 2012, 14% of the buildings in the U.S. deployed Building Management Systems (BMS) [12] 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 the underlying SCADA systems used in building automation. New buildings are installed with BMS by design, and older buildings are being continuously retrofitted with networked systems for improved efficiency. New networked devices thus present an opportunity for a building "applications plane" to provide new capabilities to building operators and occupants alike.

The current state of the art to improve the operation and responsiveness commercial building,

termed manual building commissioning, is inefficient and does not scale. It involves the facilities management having to spend between $10,000-$80,000 on a commissioning team comprising 5-10 professional experts to periodically (at intervals of 5-10 years) physically come to the building premises and spend 100s of hours manually inspecting all building systems, data from the networked sensors and blueprints to suggest improvements in building operation and control, identify faults or fix mechanical and electrical subsystems. Due to the high costs involved in the process most commercial buildings skip the periodic commissioning process or do it much less frequently than recommended. Moreover, this approach of individually 'fixing' a building does not scale to the millions of buildings in the U.S alone.

We posit that most of the utility of manual building commissioning can be obtained at scale through novel software. As we show later in the thesis, automated software can monitor data from the large sensor deployments for faults or inefficiencies, give the building manager more insight into building operation, and implement actuation that makes a building more responsive to occupants as well as the grid. Moreover, monitoring and self-correction can be done at intervals of hours or days rather than the 5-year timelines of manual commissioning teams. In theory, software applications can be deployed across the entire commercial building stock and achieve the scale that current manual commissioning teams cannot handle.

However, deploying novel software applications on one building, let alone across the entire smart-building stock is challenging. One of the main obstacles is the lack of uniform metadata to semantically interpret the data associated with the networked devices. Most BMS systems contain only limited, customized, obscure and often inaccurate metadata on its sensors. Even the most modern BMS systems present a cacophony of data and information flows that vary by buildings, vendors and across locations. Also, the existing metadata is often attached to specific visual graphics, where only an expert with enough context can infer the locational, functional and physical attributes of a sensor. 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. NIST estimates that the U.S. building industry loses $15.8 billion annually due to lack of interoperability standards [101]. Attempts have been made to address this problem. Building Information Models (BIM) [47] were introduced to address the interoperability concerns both for the design and operation of buildings. The resultant schemata – Industry Foundation Classes (IFC) [31] and Green Building XML (gbXML) [193] – were oriented towards design and construction efforts. As a consequence, only limited support was provided for BMS information. More recently, several schemata, e.g. Project Haystack [4], SAREF [63], have emerged to highlight the importance and use of building *metadata*.

Writing novel and useful software applications on building sensor deployments require not only information about the properties and characteristics of each individual sensor, but also knowledge about relationships between sensors and various subsystems in a building. For example, a fault detection algorithm trying to identify rogue zones in a building (i.e thermal zones which are always much hotter than the desired setpoints) need to identify all zone air temperature and setpoint sensors in a building, and be able to relate which zones these sensors are in, and which setpoint corresponds to which temperature sensor. Similarly other application might

require identifying the functions of other sensors, the location of a sensor, information about which subsystem a sensor is a part of, etc. Since this information is represented in different ways in each building, scalably deploying these applications face an insurmountable hurdle.

1.2 Problem Statement

Given the heterogeneity in the type and metadata of sensors in existing large sensor networks in commercial buildings, this thesis tries to answer the question — Is it possible to deploy applications scalably across smart-buildings comprising large apriori sensor networks ?

There are several aspects to this question. First and foremost is our notion of smart-building applications. We draw from the wide variety of applications that have been proposed in the industry and prior academic literature. These applications can be broadly classified into a few categories such as occupancy modeling, energy apportionment, web displays, model-predictive control, participatory feedback, fault detection and diagnosis, non-intrusive load monitoring or demand response. While it is hard to predict what kind of applications will be prevalent in the digital world of the future where every aspect of a building's control system and occupant interaction will be monitored through networked sensors, taking into account all existing applications provide a firm ground to make progress about the kinds of sensor relationships building metadata services need to provide.

Deploying applications on the smart-buildings assumes the presence of an operating system managing the sensors, a safe and robust application environment and an efficient metadata and data archiver system. There is a lot of existing work in this area, and such capabilities are provided to a limited extent by existing building management systems and by novel systems such as [65, 224]. This thesis makes contributions to sensor metadata management, and the ability for applications to express and query the relationships it needs to run, assuming that the above-mentioned systems are in place.

We achieve scale by proposing a normalized metadata schema and a way to express applications, such that we can write an application once and deploy it without any additional effort or customization to 1000s of buildings. It is important to note that not all applications can scale throughout the building stock because they might require control subsystems, sensors or relationships that are not available in every smart-building. However, even scaling the deployment to a large number of buildings with similar configuration is a substantial improvement on the state of the art.

Finally, it is not merely enough to propose a normalized metadata schema for all networked sensor systems in smart-buildings going forward. More than 14% of existing commercial buildings in the U.S alone already have legacy sensors with obscure metadata. Manually re-mapping the metadata of these sensors is an arduous task. We propose automated human-in-the-loop program synthesis and data-based intrusive and non-intrusive techniques to transform the metadata of legacy BMS systems into a normalized metadata namespace.

1.3 Thesis Roadmap

The remainder of this thesis addresses the solution to the problem statement. In Chapter 2 we provide detailed background to the existing state-of-the-art. We design an empirical methodology to evaluate the effectiveness of existing metadata schemas for sensors deployed in smart-buildings. We compare the most commonly cited existing schemas along three categories — (1) completeness, or how many of the existing sensors can they capture, (2) ability to capture relationships , or whether they can capture relationships required by applications that have already been proposed by the community, and (3) flexibility and ease-of-use , i.e whether they can capture potential novel sensors which might be built in the future and whether they are easily understandable and usable by practitioners. We show that none of the existing metadata schemas, viz Project Haystack, Industry Foundation Classes, or Semantic Sensor Web schemas satisfy these requirements.

We then present a technique to transform the obscure, terse and inconsistent legacy metadata of existing building sensors to a normalized schema in Chapter 3. We develop an automated synthesis technique which when combined with simple data-driven machine learning learns how to transform and *normalize* legacy metadata tags into a well-formed representation using a small number of examples from an expert, e.g., the building manager. Such building managers understand the metadata tags, but they are unlikely to be adept at writing complex regular expression programs to transform them to a common, understandable namespace. The transformation to such a namespace yields *semantic relationships between sensors*, which enables analytics applications to be deployed without a priori building-specific knowledge. We demonstrate our technique on three large commercial buildings comprising 1586, 2522 and 1865 sensors respectively and come from completely different institutions with different building systems, installers, and BMS vendors. We show that a few examples are sufficient to produce rules to parse a large fraction of the tags in each building. Our technique is able to normalize the metadata of 70% of all sensors in just 24, 15 and 43 examples for the three buildings. The synthesis technique is robust enough to handle the presence of obscure and noisily encoded sensor metadata.

To tackle missing legacy metadata that capture relationships between sensors in a building, in Chapter 4 we present a novel subsystem perturbation and voting-based data-analysis technique. We show that common techniques in existing academic literature are not effective in this context due to the characteristics of the data (response lags, nested control loops, tight variable boundaries). Our algorithm utilizes perturbations of subsystem variables and guarantees that the building systems operate within normal operating regimes. We were able to identify missing functional relationships correctly in 80% of the cases.

Chapter 5 describes our novel metadata schema , Brick, which satisfies the three empirical criteria we lay out in Chapter 2. Brick has an underlying graphical data model, and defines nodes and edges to capture all possible sensor information and relationships in smart-buildings. Our design of Brick is grounded by the information from BMS across five buildings spread across two continents, comprising more than 615,000 sq-ft of floor space and more than 15,700 data points, whose BMS systems were set up by different vendors, and have vastly varying subsystems and sensors. We further refine our design requirements using eight canonical building

applications that require integrated information across commonly isolated building subsystems: HVAC, lighting, spatial and power infrastructure. We demonstrate that 98% of BMS data points across our five buildings can be mapped to Brick, and our eight applications can easily query the mapped building instances for required information. We open source the Brick schema files, the BMS metadata from our buildings, the application queries that run on top of Brick and tutorials on how to map existing building metadata to Brick.

We demonstrate the power of normalized sensor metadata across large commercial buildings in Chapter 6. We first show results from three applications which identify opportunities for large energy savings in buildings — finding errant rogue zones, finding stuck dampers and identifying inefficient air handling units — on 10 large commercial buildings. These diagnostic applications only require access to archived sensor data and exploit relationships expressed between sensors and its subsystems. We then demonstrate a non-intrusive occupancy detection technique which can be applied to diverse smart-buildings in order to quantify possible energy savings in its HVAC systems through implementation of simple schedules. Specifically, we employ a step change detection algorithm for identifying step edges of the occupancy indicative signal, which are then associated with occupancy start and end times, and consider the application of an empirical decomposition technique for removing the effect of noise and other dominant factors. We evaluate the efficacy of these techniques in three large commercial buildings in the United States and show through simulations that huge energy savings can be obtained using simple schedules that can be easily programmed into legacy HVAC systems. Should facilities managers want to further increase the energy savings, we propose adaptive schedules that can track occupancy of each zone for a given tolerance for occupant discomfort.

We conclude and discuss future avenues of research in Chapter 7.

Chapter 2

Background

2.1 Legacy Building Metadata

While advances in cyberphysical systems have provided new infrastructures for monitoring and interacting with physical environments, traditional automation and control infrastructures dominate the building stock and must also advance. The monitoring and actuation networks that are wired into commercial buildings for their basic operation are increasingly accessible through the BMS (Building Management System) or SCADA (Supervisory Control and Data Acquisition) systems that host higher level control, retain historical data, and provide visualization. Many of these systems provide some kind of programmatic interface to the sensors, actuators, and historical data under their management [16, 65, 224]. But, whether provided by novel networks or legacy instrumentation, extracting meaningful information from sensor data and taking actions based on that data depends fundamentally on the metadata available to interpret it. While development of effective metadata schema has become an active topic for emerging systems, it has long been a core challenge in the legacy setting.

Often, a critical step in the deployment and engineering of large automation or building systems is formulating consistent naming conventions so that the many aspects of a "point" — its function, type, position, role, and so on — are represented in its "tag", typically a highly constrained alphanumeric string ([82, 45]). These encodings are often quite sophisticated , as they have to convey many distinct attributes and relationships, i.e., metadata, in a compact representation that is interpreted by various engineers over many years.

However, this terse metadata is designed to be used by specially trained engineers in the field; it is not designed for machine translation. Typically, tags are attached to various screens as part of the human-machine interface of BMS and SCADA systems, so engineers can check status and plot trends. With knowledge of the intention of the naming scheme and the ad hoc association to various views, the syntax and semantics of the tag are aparent to the well-trained engineer or facilities manager. But, developing an algorithm to parse the tag and soundly identify each of the semantic attributes in it is an altogether different story. There may be no field delimiters or multiple, spurious ones; symbol definition may be context dependent; different schemas may

encode the same type of sensor; and each vendor or each deployment may follow different rules.

Thus, even with programmatic access to tags, data, and other descriptive information, scaling analytics or intelligent control across the commercial building stock to, say, improve energy efficiency is likely to be intractable, as long as the basic steps in interpreting the metadata involve labor intensive manual efforts by highly trained professionals with deep knowledge of each building. Sophisticated applications may be developed for a particular building, but require customized building-specific logic and queries, which are not portable or scalable across buildings.

There have been extensive efforts to standardize and automate of the management of sensor metadata in SCADA and related systems. However, tag naming remains heterogenous and inconsistent between commercial vendors/agents [227]. Some tools (e.g [54, 83]) have been created to automate the generation of tag namings, but are largely oriented to making the tags more "human readable" for later manipulation rather than making them interpretable by computers for direct analysis.

2.2 Building Applications and Required Semantic Relationships

Typically the vendor (e.g JCI, Siemens) contracted to set up the digital control systems of a particular building uses company and deployment-specific guidelines to "tag" sensor points. Often, the only metadata accompanying a sensor stream is its tag ([82]); wherein the vendor and the facilities manager try to encode all the pertinent information for a particular sensor. In our test buildings a sensor tag `BLDA1R465__ART` encodes the following information: `BLD` denotes the *site* name, `A1` indicates it is part of the first air handling unit, `R465` indicates it is located in room 465, and `ART` indicates it is an air temperature sensor. Another sensor labelled `BLDA1R465__ARS` indicates that it is in the same room 465 (`R465`), part of the first air handling unit (`A1`) but is a room temperature setpoint (`ARS`). Note, the two example tags not only encode the location and function of the sensors, but also the semantic relationship between them and other sub-systems in the building.

These encodings typically vary between buildings (often, even those deployed by same the vendor) — for instance, in another building in our data set, a tag looks like this: `BLD.S2-06:CTL STPT:PRIORITY` with `06` indicating that the sensor was part of the 6th variable air volume unit on the second floor (`S2`), and `CTL STPT:PRIORITY` means that it is an air temperature setpoint.

Such custom, condensed encodings are widespread. We surveyed several different BMS vendors (e.g [16, 202, 125]) and found many variants of such encodings and no other available metadata for the sensors. This makes it hard to infer a sensor's context uniformly across buildings and precludes the development of applications that can scale across buildings. This non-uniform metadata problem is exacerbated by the custom metadata schemas of wireless sensor networks and novel sensors (such as BLE temperature and humidity sensors, etc.) deployed

in buildings, making it extremely hard to deploy new applications like energy visualization, demand response, energy disaggregation, occupancy modeling, model predictive control, or anomaly detection in an unknown building, and porting the same application across buildings.

2.3 Shortcomings of Prior Normalized Smart-Building Metadata Schema Proposals

There have been many efforts to come up with a common metadata schema for all buildings. We evaluate three of these schemas: (1) Project Haystack [4], (2) Industry Foundation Classes [121], and (3) Semantic Sensor Networks [59].

Project Haystack [4] aims to address heterogeneity in buildings using *Tags* to label different entities such as sensors or subsystems. For example, a temperature sensor in a particular thermal zone in a building would have the Tags: `[zone, temperature, sensor]`. Tags provide a flexible and easy to use framework for annotating metadata to building data points. An application can identify the set of concepts or sensors in a building by querying for tags. Haystack provides a vocabulary of tags that describes building equipment, weather, different types of data points and properties such as unit and data type. Haystack captures relationships between subsystems using references (akin to foreign keys in databases). The referential system is not generic and one can express only a few kinds of relationship in Haystack, e.g which zone is served by which variable air volume unit, or which air volume unit is served by which air handling unit. It defines tagging models, data formats and data structures to exchange data over HTTP using REST APIs. Their documentation provides guidelines for which specific tags and references can be used, how to use the tags to create hierarchies and relationships between different entities of interest. Haystack uses a customized data format (Zinc) and there are no standard tools that enforce or verify the myriad set of rules laid out in their documentation. The lack of tools and a number of key missing concepts, such as rooms and floors, make it difficult to map existing buildings to Haystack.

IFC [31] is a standardized Building Information Model (BIM) that developed from the need to have a common exchange model for 3D architectural drawings. Common exchange formats include Industry Foundation Classes (IFC), COBie, and gbXML. IFC is the most comprehensive format of the three. It was first standardized in 2000 as ISO 16739 [121] and supports explicit modeling of sensors, actuators and controllers since version 4 (published in 2013). COBie is a subset (view) of IFC that focusses on simple export formats such as Excel [75]. The Green Building XML schema gbXML [100] is another format that concentrates on energy performance analysis tools and has rudimentary elements to model sensors, and no dedicated taxonomy of semantic types for sensors. In this thesis, we only evaluate IFC. They are well designed to capture relationships within building components. For example, IFC is good at capturing space related information. However, the concept of *Sensors* was only added in the latest version and is still in its infancy. Thus, IFC lacks many of the common metadata attributes found in a typical BMS. Further, the focus on building CAD models makes IFC too verbose for a concise representa-

tion that application developers can grasp easily. To determine functional relationships between different subsystems in a building, one would have to traverse the spatial relationships.

Semantic ontologies provide an alternative approach, with a formal language to represent essential concepts, domain hierarchies and relationships between the concepts. Ontology models the semantics of a certain domain. They usually base on a very atomic concept of defining objects (nouns) and their relationships (verbs) in terms of triples. Ontology models usually have a open world-assumption, that states that the knowledge is never fully defined and extensible by default. This results in the behaviour that many ontologies reference to other so called upper ontologies that specify specialize vocabulary that is reused. The Semantic Sensor Network (SSN) ontology is a set of domain independent concepts to model sensors [59], defined by the W3C consortium. It is derived from several scientific ontologies and strongly orients on the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) standard. The SSN ontology is domain independent and, therefore, provides no specific taxonomy to model domain specific sensor tags. The classes range from the physical devices (*Sensor*) that sample individual values (*Observation*) of a physical variable (*Property*) belonging to the system monitored (*FeatureOfInterest*). This enables various function like classifying sensors, searching for sensors, composing sensors, transforming data, collating data, infering domain knowledge, and producing and analysing events. Considerable approaches from the Sensor Web domain are SensorML [42], OntoSensor [195], and the Semantic Sensor Network Ontology (SSN) [59]. Other domain ontologies such as [178, 41] provide similar concepts to describe sensors and the observed context. Also IFC can be represented as ontology due to the similarity in representation [32]. There exist also domain specific ontology that provide approaches to describe building sensors with ontologies. They primarily try to overcome the long history of interoperability problems in buildings due to the diversity of systems and technologies [178]. There is a wide research community is concentrated on developing middleware solutions. The often add a special kind of intelligent gateway to the network that convert messages from heterogeneous devices to achieve interoperability. Other approaches assume special device architectures that support agents or Web Services where ontologies are used here to model the system in an abstract, platform independent way. The main focus of the system model lies on interoperability as well as plug and play functionality at runtime. Other approaches use ontologies to model devices for context awareness or Ambient Intelligence features. Most of the examples are of an academic nature such as eHome [190], DogOnt [41], LinkSmart [81], SENSO [149], CONON [106], IntelliDomo [216], Charatsis et al. [48], AMIGO [102], eDIANA [18] and A3ME [114]. The first commercial approach is BrightCore [146]. Some of these approaches use own definitions of the building structure, but, not one considers a connection to a BIM. Only the ThinkHome project [189] intended to use a standardized BIM (IFC), but results have not yet been published. A number of ontologies focus on realizing specific applications like controlling things [41], energy management [142], or automated design and operation [181]. Daniele et al. [63] combined these ontology modeling efforts in collaboration with industry to create a simple but unified model called SAREF. They identify 20 recurring concepts in homes and buildings across these ontologies, and lay out the steps to convert SAREF to a custom ontology.

Haystack Tagging Ontology (HTO) [49] maps the Haystack tags to an ontology, with each

tag corresponding to an ontology class. Thus, HTO is able to combine the flexibility of tags and the formal modeling of ontologies to define essential BMS metadata and the relationships between entities. However, HTO confines the ontology to the defined tags, and the building entities which are a collection of tags (e.g. zone temperature sensor) are not modeled. HTO also does not provide a way to compose complex subsystems in a building and relies on Haystack tagging for mapping raw metadata to the ontology.

2.4 Empirical Evaluation of Existing Schemas

We use sensor metadata from three large commercial buildings, each set up by a different Building Management System (BMS) vendor, and a list of 87 published smart-building applications[2] to evaluate the three schemas based on the following criteria:

- *Completeness:* Could all the distinct sensor metadata information (such as a sensor's location, type, etc.) contained in these buildings be represented by these schemas?
- *Ability to Capture Relationships:* Is it possible to express all the sensor relationships (required by the applications we study) through these schemas?
- *Flexibility:* How flexible are these schemas to capture uncertainty in the metadata (e. g. uncertainty over whether an air pressure sensor is located before or after a damper), or the emergence of new sensors (e. g. Apple iBeacon, Kinect, card swiping machines) and subsystems (e. g. a smart couch) ?

Our results show that none of the three schemas capture all the available sensor metadata, nor express all the relationships required by novel applications, nor capture any notion of uncertainty, nor allow easy extensibility to model novel sensors.[1]

We evaluate the effectiveness of the metadata schemas on three buildings set up by different BMS vendors [10]. One of the buildings (henceforth referred to as `DUB`) is located in Dublin , Ireland and two (`SODA` and `SDH`) in Berkeley, California, U.S.A. The metadata of the BMS sensors in these buildings were manually mapped against a set of Haystack tags to the extent possible. In cases where Haystack did not have the relevant tags, we developed our own consistent tag names.

Terminology: We define a *sensor label* as a collection of tags capturing a sensor's metadata. For example, in one of the buildings in our testbed, a sensor was encoded with the metadata `SODA1R465__ART`. This indicated that the sensor was a *zone air temp sensor* (`ART`) in *zone* 465 (*zoneRef* : `465`) which is served by *ahu* 1 (*ahuRef* : `1`) in the *site* SOD (`SOD`). Hence its sensor label is *'site ahu ahuRef zone zoneRef zone return temp sensor'* , each of the terms in the label being a 'tag'. Sensor labels indicate which tags the building vendors put together while framing the custom metadata of a sensor. The three buildings have 2028, 2551, and 1586 sensors respectively, comprising of 510, 148, and 281 distinct sensor labels.

[1] This work was done in collaboration with Joern Ploennigs, IBM Research Ireland [39]

The results for the three buildings, labeled `SODA, SDH and DUB` are shown in Figure 2.1. The histogram of the tags is shown in Fig. 2.1a. Many tags, such as *site, zone, sensor* are common accross the buildings. This illustrates that a taxonomy of common tags exists between various buildings. However, Fig. 2.1b shows that none of the three studied schemas has a taxonomy that includes all the tags used in these buildings. Hence, none of the three schemas are complete. The most common sensor labels (or aggregation of tags) in our dataset are shown in Fig. 2.1c. Although tags are common across buildings, sensor labels are not, showing that different BMS vendors use different combination of tags while describing a sensor which might perform the same function. Also note that the frequency of occurrence of individual tags are Pareto distributed, implying that getting a common taxonomy of a few tags can result in normalizing a large amount of building metadata.

We quantify the extent to which the tags identified in our testbed are expressible in the three studied metadata schemas. We also analyze a set of 87 applications from the building application literature[2] to quantify whether the schemas can capture the required sensor relationships.

The main classes of applications, their metadata relationship requirements and whether these relationships can be expressed in Project Haystack, IFC and Semantic Web is shown in Table 2.1. The first column lists the needed relationships between different dimensions of metadata. For example, 'sensor $\leftrightarrow$ function' states that a link between the sensor entity and the functional semantic of the sensor is required. Hierarchical relationships are possible such as 'location $\leftrightarrow$ location' refers to relating the sub-locations to a location. The second column provides an intuitive query example for each relationship. the applications listed in the columns 3 to 10. An 'X' indicates that a specific relationship is required to automate the application. The last three columns show which relationships are supported by the studied metadata schema. The last three rows in the table then compute the *applicability* of each schemata as the fraction of the required relationships the is captured by a particular schema.

2.4.1 Project Haystack

Completeness: Figure 2.1b shows the tags supported by Haystack in our three testbed buildings. Project Haystack supports 54 % of the unique tags used in the three datasets. Weighted by the occurrence frequency of the tags in the dataset, Haystack supports 63 % of the tags. While Haystack has well-defined tags to capture very commonly used sensor types, e. g. zone temperature sensors, its tags are incapable of representing (a) building-specific sensors such as a fault-detecting sensor that is monitoring the status of a pump in the condensor water loop, (b) common sensors outside the HVAC system, such as whether a light array is monochromatic or has hue controls, or whether an entire light panel is controlled by a single sensor or by multiple sensors.

Ability to Capture Relationships: Table 2.1 compares the relationships required by common applications toward the support in Haystack. Although Project Haystack is capable of capturing relationships between HVAC subsystems, it cannot model relationships between spatial elements, such as the list of rooms in a floor. Also, Haystack is unable to capture relationships

Table 2.1: Relationships required and expressibility of apps existing metadata schemas

Needed Relationships		Common Applications								Schemata		
Relationships	Example	Occupancy Modelling[1]	Energy Apportionment[2]	Web Displays[3]	Model-Predictive Control[4]	Participatory Feedback[5]	Fault Detection and Diagnosis[6]	Non-Intrusive Load Monitoring[7]	Demand-Response[8]	Haystack	IFC	Semantic Web
sensor ↔ function	sensorsOf(type)	X			X	X	X			X	X	X
sensor ↔ location	sensorsIn(room)	X	X		X	X	X		X	X	X	X
location ↔ location	roomsIn(building)	X	X								X	X
asset ↔ location	ahuOf(room)	X			X	X	X		X	X	X	
sensor ↔ asset	sensorsOf(AHU)	X	X		X	X	X		X	X	X	
asset ↔ asset	ahuSupplying(vav)	X			X	X	X		X	X	X	
location ↔ persons	occupantOf(room)	X										
location ↔ organisation	ownerOf(room)	X	X								X	
gadget ↔ persons	macAddrOfPhone(user)	X										
gadget ↔ location	computerIn(room)	X										
meter ↔ location	meterOf(room)		X					X		X	X	X
meter ↔ gadget	meterOf(computer)	X										
meter ↔ asset	meterOf(ahu)	X										
meter ↔ sensor	sensorUnder(meter)		X					X				
Applicability	Haystack	42%	50%	75%	100%	100%	100%	50%	100%	77 %		
	IFC	58%	83%	100%	100%	100%	100%	50%	100%		86 %	
	Semantic	25%	50%	100%	25%	25%	25%	50%	25%			41 %

References: [1][139, 127]; [2][98, 122]; [3][24]; [4][173, 207]; [5][110, 124]; [6][179, 220]; [7][150]; [8][222]
A more complete set of citations is available in [2].
Applicability denotes what fraction of the required relationships is captured by a particular schema.

between different views (spatial, HVAC, power) of a building, e. g. which set of rooms/floors an HVAC zone comprises of.

Flexibility: The schema has no way to capture any uncertainty, does not capture any novel sensors/electronic gadgets (such as computers, iPhones, etc.), and requires consensus among the community to include new sensor tags into the vocabulary.

2.4.2 Building Information Models (IFC)

Completeness: IFC supports 11 predefined semantic types, such as CO2, heat, temperature and sound sensors, with additional properties specifying units, values, type, etc. 22 generic measurement types such as count, electric current, length and time are available. IFC provides concepts for 29 % of the unique tags in our dataset (Figure 2.1b), and 60% when weighted by tag occurrence frequency.

Ability to Capture Relationships: IFC has its foundation in 3D geometrical modelling and provide comprehensive ways to model spatial and asset relationships (Table 2.1). The main shortcoming with IFC is that metadata is primarily related via 3D objects, making it hard to

query even simple things, such as if two devices are located within the same room. It is also not possible to assign multiple sensor types to a device, e. g. modeling multi-sensors that sample semantically different values. Concepts for humans and novel sensors do not exist.

Flexibility: Although, IFC and gbXML formats are built to be extensible, an extension to the standardized core model requires consensus from a community primarily focused on building design. Also it is unable to capture uncertainty of 3D objects such as room size, wall thickness, etc.

2.4.3 Semantic Sensor Web

We us the Smart Appliances REFerence (SAREF) ontology([62]) in our comparison.

Completeness: SAREF classifies device functionality roughly into Sensing, Actuating, Metering. For each function a specific sensor type can be defined as a literal. By default 5 types (Temperature, Occupancy, Humidity, Motion, Smoke, Pressure) are provided. Meter types such as Water, Gas, Pressure, Energy, and Power are provided. This small set of default tags results in a coverage of only 11% of the tags in our dataset, (8% when weighted by frequency).

Ability to Capture Relationships: The SAREF ontology allows modeling hierarchies of spatial elements, but does not specify modeling of assets (such as AHUs, VAVs, etc.). This strongly reduces the applicability of the SAREF ontology to the use cases defined in Table 2.1. Units are specified using the external ontology – Units of Measure (OM). SAREF links 20 ontologies that define concepts for other areas. This demonstrates an inherent strength of the semantic web, as additional ontologies can be linked and reused to model aspects such as organizational structure, novel devices, etc. However, a strict guideline of which ontology should be used is not provided by SAREF.

Flexibility: Ontologies differentiate between classes (kind of things) with attributes (properties of things) and individuals (instances of classes) with relationships (links between instances). While such ontologies are helpful in defining a clear and verifiable meta-model, it only allows capturing uncertainty on an individual level, such as multi-lingual names. Also, addition of novel sensors and taxonomies requires a consensus by the standards body.

2.5 Related Work on Normalizing Legacy Metadata Tags Into a Uniform Smart-Building Metadata Schema

2.5.1 Extracting Information From Legacy Metadata Schemas

There have been various data-driven efforts to capture the contextual and semantic relationships between sensors in order to build applications, such as type classification of sensors [169] and finding spatial relationships between sensors ([91, 116, 141]). However, these techniques either classify sensors into broad categories (type classification), and do not capture semantic (or functional) relations between sensors (e.g which air temperature sensor is related to which setpoint

sensor), and hence are not useful in writing applications which depend on the semantic relationships between sensors. Even if techniques like [91, 116, 141] can predict that an air flow sensor and a temperature sensor are in the same room, it cannot determine the air handling system the room is a part of — information often encoded in the sensor metadata tag directly.

There has also been prior work in substring extraction[107], and log record manipulation[131] using examples from a human. However, spreadsheet and log data comprise records encoded in very few schemas or formats, requiring only two or three human examples to synthesize programs to extract all the required fields. These techniques fail to parse building sensor metadata, which present a far more heterogenous and noisy dataset, containing many different schemas and hundreds of encoded fields (comparison shown in Figure 3.3). We achieve the required robustness using a combination of clustering and domain-specific language contructs. We draw from the boolean classification techniques in [107] to avoid over-generalization of synthesized rules. In the building domain, industrial softwares like PI from OSIsoft [170] give users the ability to generate wildcard regular expressions to select a set of tags. Our domain-specific language provides for much more powerful regular expressions than such software. [201, 180] use string matching to find the most likely fields in building tags. This approach breaks down when fields are represented by only one or two characters, as shown in our previous examples.

2.5.2 Challenges in Normalizing Legacy Metadata

Experts, often facility managers or maintenance professionals, are not well-equipped to construct the correct regular expression programs themselves to parse legacy metadata into a normalized schema. Moreover, inconsistencies in the metadata structure including obscure and noisy tag encodings require hundreds of very complex regular expressions, which would make manual regex generation error-prone, if not impossible. Automatically synthesizing regular expression programs that transform primitive metadata into a common desired namespace using examples from the expert addresses both these concerns.

Unlike machine opcodes, the language of the primitive metadata was not created with the intention of being machine decode-able. Machine opcodes generally have specific fields which specify how to parse a particular sequence of characters/bits (*fixed field encodings*) which make designing a language for parsing tractable. On the other hand, primitive sensor metadata may suffer from the following inconsistencies which make parsing hard[2] :

- Context-dependence: Different fields may be coalesced in a context-dependent way. For instance, the sixth letter — `C` — in
  `BLDA1C600A_ART` denotes the value for the field *room*, while in
  `BLDC1C2_____TMR` it denotes the value for field *chiller*.
- Multiple Schemas: Tags within a particular building may comprise several different schemas. For instance, a sensor with the metadata
  `BLDA1C600A_ART` should be parsed as

[2]we illustrate examples from one building in our dataset. These challenges appear in all the three buildings in our dataset

| BLD | A | 1 | C | 600A_ | ART |, each token representing the value of a different field. In the same building, there exists sensors with metadata such as
`BLDS03AR179ART`, which is tokenized as
| BLD | S | 03A | R | 179 | ART |.

- Variable Delimiters: The metadata schema also does not depend on specific delimiters. As shown in the example
| BLD | A | 1 | C | 600A_ | ART | , letters themselves can be the delimiters for some tokens, and underscore characters for others.
- Spurious Delimiters : Some tokens may have delimiters as a part of the token. For instance, in the same dataset a sensor with the tag
`BLDA2S14SASA_M`, for example, should be parsed as
| BLD | A | 2 | S | 14 | SASA_M |.
- Multiple values for the same field: some fields may be expressed by multiple different values. For instance, the field *room* may be denoted by an `R` or a `C`, while the field *damper valve position* maybe expressed as either `VAV` or `VP`.
- Noisy metadata: The tags of some sensors may have misplaced or wrong tokens borne out of human error.

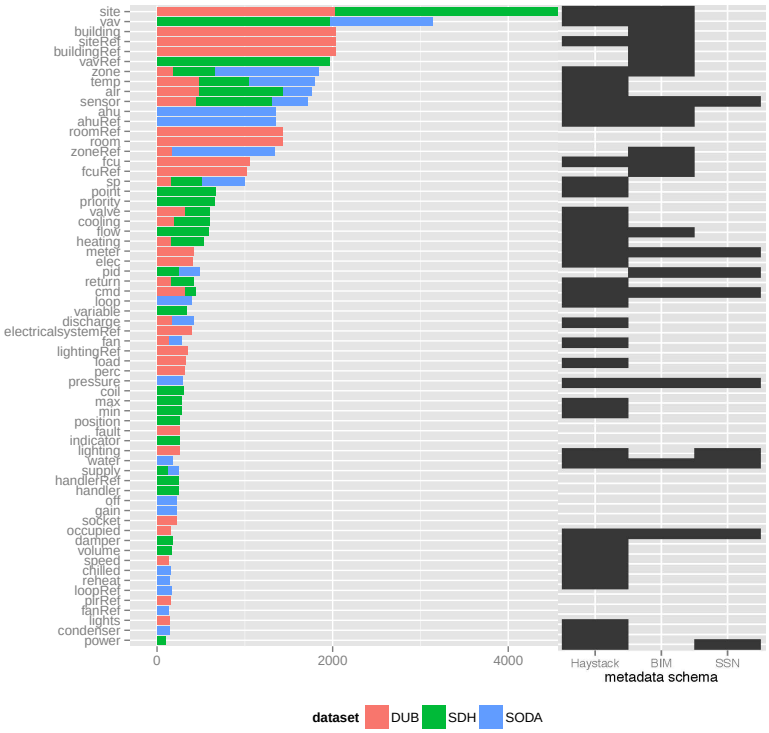
There have been various data-driven efforts to capture the contextual and semantic relationships between sensors in order to build applications, such as type classification of sensors [169] and finding spatial relationships between sensors ([91, 116, 141]). However, these techniques either classify sensors into broad categories (type classification), and do not capture semantic (or functional) relations between sensors (e.g which air temperature sensor is related to which setpoint sensor), and hence are not useful in writing applications which depend on the semantic relationships between sensors. Even if techniques like [91, 116, 141] can predict that an air flow sensor and a temperature sensor are in the same room, it cannot determine the air handling system the room is a part of — information often encoded in the sensor metadata tag directly.

2.6 Technical Problem Statement

Given the background developed in Sections 2.3, 2.4 and 2.5 we articulate our problem statement in more detail — "Is it possible to normalize legacy smart-building sensor metadata with little effort into an expressive, complete, usable and uniform metadata schema such that novel applications leveraging the resulting standardized sensor and relationship representations can be run unmodified across diverse smart-buildings ?"

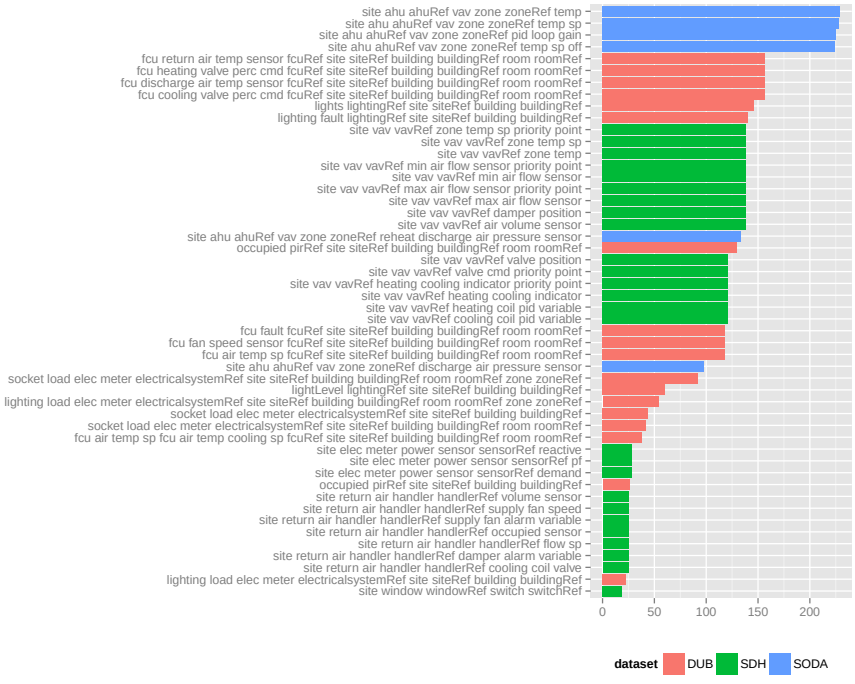
In the following chapters, we describe a human-in-the-loop synthesis technique coupled with perturbation of sensors and careful analysis of the associated data to normalize the legacy metadata of existing building sensors. Our technique uses clustering and data-based sensor association to circumvent the challenges related to machine-parsing described in Section 2.5. We also design a metadata schema based on a general graphical data model that satisfies the empirical criteria laid out in Section 2.4. Finally, we demonstrate this normalized metadata paradigm where sensors, its attributes and inter-relationships have standard representations can enable

applications which can not only diagnose common problems in buildings, but can measure response of building subsystems to occupant behavior which can then be used to improve occupant comfort and mitigate energy loss.



(a) Most frequent tags.

(b) Tags supported by metadata schema.



(c) Most frequent sensor labels.

Figure 2.1: Analysis of tags and sensor labels in the three buildings.

Chapter 3

Normalizing Legacy Smart-Building Metadata

In this chapter we develop an automated synthesis technique that learns how to transform and *normalize* legacy metadata tags of sensors in large-commercial buildings into a well-formed representation (cf., [4]) using a small number of examples from an expert, e.g., the building manager. As mentioned in Section 2.5 building managers understand the tags, but they are unlikely to be adept at writing complex regular expression programs to transform them to a common, understandable namespace. The transformation to such a namespace yields *semantic relationships between sensors*, which enables analytics applications to be deployed without a priori building-specific knowledge.

3.1 Brief Overview of Technique

We draw inspriation from programing-by-example techniques developed in [107]. To our knowledge, this work is the first attempt to apply these techniques to buildings, and find that the techniques must be fundamentally rethought to work well in building systems. Our approach is shown in Figure 3.1. To reduce the complexity of parsing the varied and inconsistent schemas contained within a building, we first cluster the sensor metadata into chunks which are more likely to share a common schema. An example is selected from one of the clusters and provided to an expert to parse into a common namespace (In practice, a simple GUI is provided for this). Based on the example parse, we synthesize rules from a domain-specific language which are consistent with the expert-provided example. We then apply these rules to parse the field encodings of the remaining sensor tags in that cluster, i.e., to generalize the example to a program that can parse many of the tags. Based on the resultant parsing, a new example is selected to be presented to the expert, and so on. Ideally, a few iterations of this example-driven synthesis loop should produce rules that correctly parse a large fraction of the tags. In practice, this is the case, but several important factors and subtleties arise, which we study in the sequel.

Once progress has been made in qualifying tags through these syntactic methods, if we have

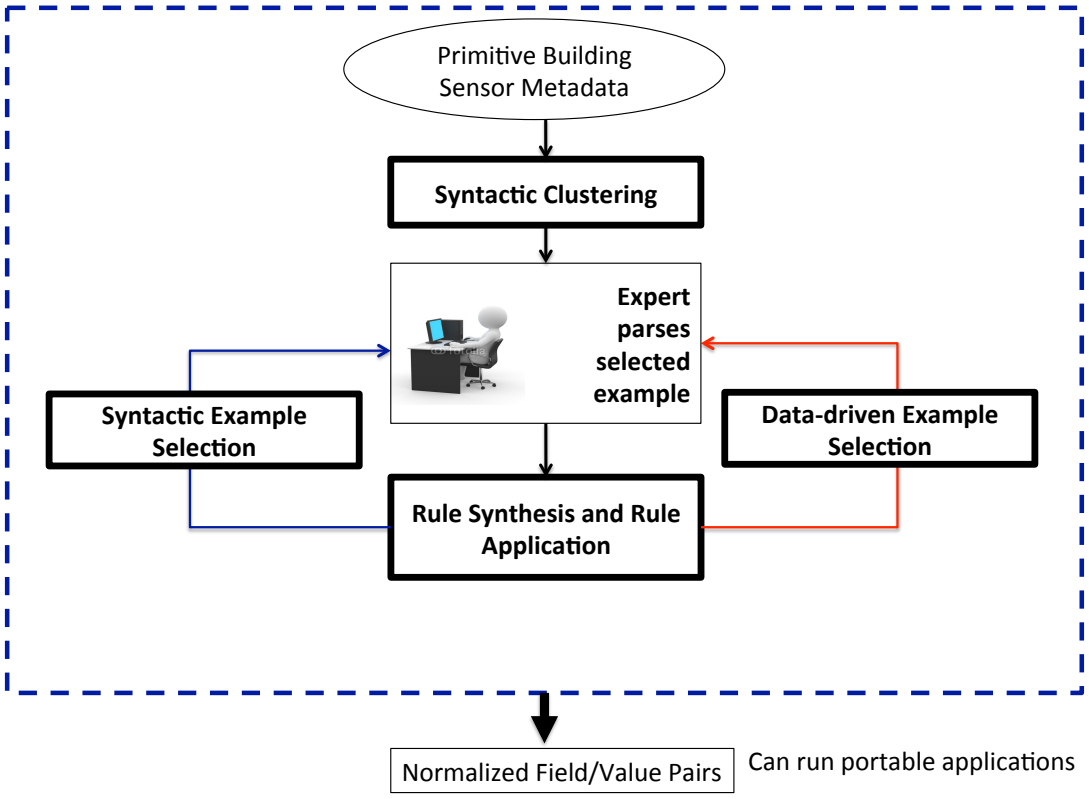


Figure 3.1: High-level view of major steps in our Algorithm

access to the data streams for the points associated with the tags, we can employ learning on the data to attempt to find points that are semantically related, but with syntactically distinct tags. This 'boosting' is represented by the second loop in the figure. In either loop, an important question is how to select the next example to present to the expert, since the human in the loop is the precious resource. And, a critical question is when to stop. Typically, most of the tags in a building conform to a few simple encoding formats, but many indiosyncratic formats are present with few tags each. With proper clustering, selection, rule synthesis and generalization, the vast majority of tags are resolved with a few examples, but a long tail of obscure ones remains. On the other hand, the real goal of this process is to enable portable applications on buildings and any such application only requires certain types of points. Thus, we also study how many examples are needed to resolve all the points that are relevant to certain important applications.

The three large commercial buildings used in our study have 1586, 2522 and 1865 sensors respectively and come from completely different institutions with different building systems, installers, and BMS vendors. We find that, indeed, a few examples are sufficient to produce rules to parse a large fraction of the tags in each. Our technique is able to normalize the metadata of 70% of all sensors in just 24, 15 and 43 examples for the three buildings. The synthesis technique is robust enough to handle the presence of obscure and noisily encoded sensor metadata.

However, the pre-clustering step is essential in some buildings to avoid over-generalization of the synthesized rules. We study the criteria used to select the next example to present to the expert and find that random selection generally performs better than application-specific heuristics, and choosing a random example from the cluster with the largest number of yet-unqualified tags is robust.

Each building has a long list of obscure tag formats which slows convergence; to parse the entire set of tags requires in 161, 116 and 196 examples respectively. However, the applications of interest generally do not require normalizing the metadata of all sensors in a building, but only specific sensor types. That is not to say they are uniformly encoded throughout — in one of our buildings, the zone temperature sensors were encoded six different ways.

The Data-Driven Example Selection builds a random forest classifier for the set of sensors required for an application from the set of sensors already normalized, and applies it to the remaining sensors in the building to identify similar sensors that have not yet been presented to the expert. This classifier is built using a feature vector computed from the physical data associated with the sensors. For three applications on the two of our buildings with accessible data streams the required sensors are parsed with an order of magnitude fewer examples than with only syntactic example selections.

The techniques developed here are likely to be applicable to the other large legacy sensor networks, such as industrial processing, or urban monitoring, and provide a metadata framework that can be adopted without need for such learning-based transformations in emerging sensor networks.

We describe our program synthesis and example selection techniques in Section 3.2. We evaluate our synthesis technique on three large commerical buildings, and run three portable efficiency applications on them in Section 3.3.

3.2 Automated Metadata Construction Techniques

We now describe the techniques used to perform each of the four main components of our system , i.e apriori Syntactic Clustering (3.2.1), Rule Synthesis and Application (3.2.2), selecting an example for the expert using only the sensors' metadata syntax (3.2.3) or through data (3.2.4).

3.2.1 Syntactic Clustering

Given a building's sensor tags, we perform syntactic clustering on them. This preconditioning step has three advantages — (a) tags in a resulting cluster are more regular, and hence helps our rule synthesis algorithm converge , (b) the cluster with the most number of unqualified tags is a good metric to decide which example to next select for an expert's parse , (c) the rule synthesis and application happens only on the tags in the same cluster as the expert-provided example, and is computationally fast.

In constructing the feature vector, we aim to cluster tags which resemble fixed field encodings together. Unlike clustering text documents, we have no apriori notion of delimiters or

words. We assume all non-alphanumeric character to be a potential delimiter. We replace contiguous runs of alphabets, numerals and special characters with a single number — alphabets are denoted by `1` , numerals by `2` and each special character as an independent but consistent number. Thus, the tag `BLDA1R465__ART` is denoted in our feature space as $\langle 1, 2, 1, 2, 3, 1\rangle$, and `BLDS03AR179ART` as $\langle 1, 2, 1, 2, 1\rangle$[1]. Intuitively, points which are close together in this space have the same relative positioning of alphanumeric characters, and thus can be parsed by a similar synthesized program. The feature vector corresponding to a particular sensor is then padded with `0`s to make them have the same number of dimensions.

We perform agglomerative clustering based on the jaccard distance between the feature vectors as the distance metric[2]. We define the hetereogeneity metric within a cluster to be the average of all-pair jaccard distances, and at each step merge two clusters if the heterogeneity of the resultant cluster is below a specified threshold[3]. Clusters thus formed are more regular (since they have similar positioning of alphabets, numerals and delimiters). Tags with idiosyncratic schema form their own clusters.

3.2.2 Rule Synthesis and Rule Application

Our synthesis technique selects a sensor tag and presents the example to an expert for a parse. We first introduce terminology that we will use throughout this section, followed by an overview and a description of the synthesis technique.

Terminology: The expert is expected to point out *(Field, Value, Value Type)* tuples in the sensor tag. A *field* is mapped on to a substring of the tag, which is called its *value*. A field can have a constant or a variable value. A value is a *constant* if it is not specific to that particular tag, and *variable* otherwise.

Sample Input: Suppose the expert is presented with an example `BLDA1R465__ART`. This tag indicates that it is in Building `BLD`, is part of the first air handling unit, indicated by the character `A1`, in room `465` (`R465`) and it is the area temperature sensor (`ART`). The expert should provide the parse as: `BLDA1R465__ART` : (site, `BLD`, const), (ahu, `A`, const), (ahuRef[4], `1`, var), (zone, `R`, const), (zoneRef, `465`, var), (zone air temp sensor, `ART`, const). The *site* field's value is `BLD`, which is not specific to that particular sensor tag. Hence, the expert should mark it as a constant. On the other hand, the value of the *zoneRef* field is specific to that sensor, and hence should be marked as variable.

Sample Output: The synthesis technique should be able to identify the learned fields in a new tag automatically. For example, given a new tag `BLDA5R577A__ART`, it should output the set of tuples: `BLDA5R577A__ART` : (site, `BLD`), (ahu,`A`), (ahuRef,`5`), (zone,`R`), (zoneRef,`577A`), (zone air temp sensor, `ART`, const). If there exists a portion of the tag for which the synthesis technique has not yet received an example, it should remain unmapped to any field-value tuple.

[1] since BLDA is a continuous run of alphabets, it is replaced by a single `1`

[2] so that strings with a higher number of common coordinates would be clustered together

[3] We set the threshold to 0 in our experiments

[4] ahuRef, zoneRef are idioms from the Haystack taxonomy

We term each of these tuples as a *qualification*, because it qualifies a set of alphanumeric characters into field-value pairs in a common namespace. A tag is *fully qualified*, if every alphanumeric character in it was correctly *qualified* by the set of outputted *field-value* pairs. The goal of the expert should be to use fields from the set of markers and idioms defined in Project Haystack [4]. There might be cases where the correct field (such as specific alarms, etc) is not part of the Haystack taxonomy. In these cases, we expect the expert to use an easily understandable long-form field name, which is consistent[5] across the entire building.

Synthesis technique overview (within each cluster[6]) : The high-level aim of the technique is to learn two sets of information from the given input-output examples — (a) which fields are applicable on a particular sensor tag, and (b) what is the set of regular expressions that transform the tag to the value of the corresponding field.

From each expert example, and for each field in the expert-provided qualification, the set of all expressions from the language (shown in Figure 3.2), that could extract the required field's value is computed. If there are multiple examples for the same field, the substring extraction rules of the multiple examples are intersected to obtain a more concise set of expressions. If the substring extraction rules cannot be intersected, they are maintained as two disjoint sets, which we shall hereby term as a *partitions*.

Finally, for each field and each disjoint set of extraction rules/regular expressions therein, a classifier in the form of **If Then ... Else** statements is built, where the conditions are boolean in the Disjunctive Normal Form (DNF)[7]. These classifiers dictate whether a particular field is applicable to a particular sensor tag, and which regular expression partition should be applied to it. Thus, we can independently consider each *field* to be a potential output for a sensor tag. If a field is deemed to be applicable by its classifier, then the value of that field would be generated by the regular expressions synthesized by our technique.

Learning a classifier for each field separately has two advantages. The classifier is able to identify and extract fields from other sensor tags, irrespective of the formatting of the rest of the tag. Suppose two zone temperature sensors have the tags `BLDA1R465__ART` (expansion described above) and `BLD_300___ART` : (site, `BLD`), (zoneRef, `300`), (zone air temp sensor, `ART`) . In both cases, the first three characters denotes the value of the *site* field, and the substring `ART` denotes that it was a *zone air temp sensor*. Learning classifiers for the fields *site* and *zone air temp sensor* would enable us to gain useful information by automatically applying these fields on the second sensor, even though we might not know its *zoneRef*. Thus, we are able to transform as much of the metadata as possible without having to depend on another example from the expert.

The language : The language is designed to take into account various possible metadata encodings that can occur in sensor tags. We assume that the substring corresponding to the value of a field can be obtained by either (a) extracting substrings between two constant indices, (b) extracting the substring between two other fields or regular expressions, and (c) as a constant

[5]This can be achieved by presenting the expert a set of fields that has already been used in that building to qualify that particular substring.

[6]the clusters formed in Section 3.2.1

[7]same technique as in [107]

Transformation Program **P**	:=	if b_1 then e_1 else if b_2 then e_2 ….. else tag not exist in string
Boolean classifier $\mathbf{b_i}$	:=	$d_1 \vee d_2 \vee \ldots \vee d_n$
Conjunct $\mathbf{d_i}$	:=	$p_1 \wedge p_2 \wedge \ldots \wedge p_n$
Predicate $\mathbf{p_i}$	:=	**Occurs**(v, r, k) | **OccursAtPos**(v, r, c)
Occurs(v, r, k)	:=	True, iff regular expression r occurs in string v , k times
OccursAtPos (v, r, c)	:=	True, iff regular expression r occurs in string v at index c
Extraction Rule **e**	:=	**Substring** (v, p_1, p_2)
Substring(v , p_1, p_2)	:=	Substring of string v between positions p_1 and p_2
Position $\mathbf{p_i}$	:=	**Constant**(k) | **PrecedeSucceed**(r_1, r_2, c) | **ConstantWidth**(k)
PrecedeSucceed(r_1, r_2,c)	:=	Index at the c^{th} intersection of regular expression r_1 and r_2
ConstantWidth(k)	:=	Index p_1 + k, where p_1 is starting index of substring
Regular Expression **r**	:=	**Tokens**(T_1, …. T_n)
Token **T**	:=	Alphabets| Numeric | specialToken | ε | constant tag value entered by expert

Figure 3.2: Language for learning substring extraction

width substring if the left index is identified. The classifiers, which are constructed to determine whether a particular field is applicable on a sensor's tag, is based on the general format of the string — (a) whether a particular regular expression occurs at a particular index, or (b) how many times a particular regular expression occurs in the sensor tag.

The top level expression of the language is the classifier — the *If* b_i *Then* e_i structure, which applies the substring expression e_i to the input only if it matches the boolean expression b_i. The boolean function is in the Disjunctive Normal Form and is composed of predicates of the form **Occurs**(v_i, r, k) , which evaluates to true, iff the input v_i has k occurrences of the regular expression r, or
OccursAtPos(v_i, r, c) which evaluates to true iff the input v_i has a regular expression r which occurs at index c.

The Substring expression **SubString**(v_i, p_1, p_2), evaluates to the substring between positions p_1 and p_2 of the string v_i. **Constant**(k) denotes the integer position k in the substring. A position expression **PrecedeSucceed**(r_1, r_2, c) when applied on a string s evaluates to an integer position t in the subject string s such that r_1 matches some suffix $s[0..t]$ and r_2 matches some prefix of $s[t...l]$ (where l = Length(s)). Also, t is the cth such match starting from the left end of the string. If such an position t does not exist in the string, this operator fails. **ConstantWidth**(k) is an operator which indicates a constant offset index from the position p_1. The regular expressions are either just a single token τ, or a token sequence, **Tokens**$(\tau_1..\tau_n)$, or ϵ (which matches the empty string). The tokens τ comprise of a single token to denote alphabetic characters (referred to as *AlphTok*) , one for numeric characters (referred to as *NumTok*), one for each special character, and one for each constant value entered by the user. The output is obtained by applying the resultant **SubString**(v_i, p_1, p_2) operation.

We provide a couple of examples to elucidate how the field-value extraction technique works.

Example 1: Tag : `BLDA1R465__ART`, desired output value : `ART` (for the field *zone air temperature sensor*).

Possible programs synthesized: SubString(s, Constant(11), Consant(14)), or SubString(s, PrecedeSucceed(*UnderscoreToken*, *ART*,1), ConstantWidth(3)) .

Example 2: Suppose the synthesis algorithm has seen two examples (a) `BLDA1R465__ART`, for which the value for field *ahuRef* is `1` and (b) `BLD_300___ART`, in which the field *ahuRef* does not exist, and hence should not be applied.

Possible programs synthesized to extract the value of the field *zoneRef* is : **If** b_1 **Then** e_1, where b_1 = OccursAtPos(s, (A), 3), e_1 = Substring(s, Constant(4), ConstantWidth(1)). This program ensures that the field *ahuRef* is only applied to tags similar to the former, and not to the latter.

Example 3: Consider the following tags from our testbed, in which the first sensor is a status indicator connected to supply fan `4`, while the second is a variable air volume unit airflow sensor in room `5871`.

Tag 1 : `BLDA4S1831_STA` : [(site, `BLD`, const), (ahu, `A`, const), (ahuRef,`4`, var), (supply fan,`S`, const), (supply fanRef,`1831`, var), (status point,`STA`,const)] ; and

Tag 2: `BLDA3R5871_VAV` : [(site, `BLD`, const), (ahu, `A`, const), (ahuRef, `3`, var), (zone, `R`, const), (zoneRef, `5871`, var), (vav, `VAV`, const)]

Both these sensor tags have the exact same arrangement of numeric and alphabetic characters, and special symbols, and no classifier comprising only tokens for alphanumeric and special characters would be able to discern between the two. This can result in erroneous extra fields being applied to sensor names.

Token Set: To solve this problem, and get a more expressive set of tokens, we utilize the values marked as *constant* in the examples provided by the expert as special tokens. In the example above, the tag `BLDA4S1831_STA` is treated as a set of tokens — (`BLD`), (`A`), *NumTok*, (`S`), *NumTok*, *NumTok*, *NumTok*, *NumTok* ,*UnderscoreTok*, (`STA`).

Note that utilizing the *constant* values as tokens enables us to have a different token set for each building. The set of tokens increase as the expert gives more examples. Note that the new tokens provide enough expressibility for the regular expressions to differentiate between the two tags `BLDA4S1831_STA`, and `BLDA3R5871_VAV`.

3.2.3 Syntactic Example Selection

We choose the cluster[8] with the maximum number of not-yet-fully qualified sensors, and choose one of them as the next example to present to the expert. We evaluated four different techniques

[8] as described in Section 3.2.1

to choose an unqualified sensor from that cluster, the results of which are presented in our evaluation (Section 3.3.2).

- *Random:* Select an example at random.
- *MinLeft :* Select the example with the minimum tag length left to qualify. The intuition is to complete partial parse of sensors.
- *MaxLeft :* Select the example with the maximum tag length left to qualify. The intuition is to help the synthesis technique cover the space of unseen fields (i.e the long tail in Figure 3.4).
- *SameLeft :* Select an example with the most frequent unqualified substring. This method seeks to find a commonly occurring field.

3.2.4 Data-driven Example Selection

Often the same sensor types in a building are specified in the form of multiple different schemas. This may occur because of mistakes on part of the vendor, later addition of such sensors, etc. For instance, zone temperature sensors in Building 1 in our testbed was encoded in 6 disparate schemas, comprising of 78%, 11%, 6%, 2% , 2% and 1% of the sensors. It is unlikely that a purely syntactic example selection method will happen upon the tags with rare encodings. In contrast, because the characteristics in the associated data of sensors of the same type tend to be similar, data-based selection can identify the rare-encoding sensors as being similar to sensors with a more common encoding, and present one of these rare examples to the expert for parsing.

To utilize this approach, an expert should specify which are the sensor types an application needs (henceforth termed as *Required Sensors*). Our technique first transforms each sensor stream into two new data streams M and V that contain the running median and variance values of the original data stream, using a 45-minute long sliding window. The length of the window is set to 45 minutes to smooth over any transient phenomena and noises. We compute the *minimum*, *maximum*, *median* and *variance* of each of the two streams M and V, resulting in the following 8-tuple: $\langle min(M), max(M), median(M), var(M),$
$min(V), max(V), median(V), var(V)\rangle$.[9]

This 8-tuple is then used as a feature vector to train a random forest classifier based on sensors which have already been fully qualified, and to classify the yet-unqualified sensors. We choose a random forest classifier because it is an ensemble learning algorithm which is more robust to noise and in general outperforms other learning techniques like SVM or linear regression. We choose the sensor which the classifier classified as a *required sensor* with maximum likelihood and select that example be parsed by the expert.

3.2.5 Running Portable Building Applications

We study three important applications, that with normalized metadata can run portably across buildings — Finding Rogue Zones, Finding Zones with Stuck Dampers, and Finding Inefficient

[9]Described in more detail in [115]. In our experiments, we use the same one week time window of data from the month of July for all sensors.

Air Handling Units.

1. Finding Rogue Zones : A thermal zone is *rogue* if its air temperature is constantly above its required setpoint, i.e it requires constant cooling. Rogue zones are typically caused by high thermal load, incorrect setpoints, or faulty sensors, and should be rectified before implementing further sophisticated efficiency techniques. Rogue zones are an artifact of poor planning or wrong setpoints, and can be fixed if brought to the attention of a building manager. This application queries for sensors having the *zone air temp sensor* field, and for each such sensor, queries for a sensor with the *zone air temp setpoint* field having the same field-value for the *zoneRef* field, and checks whether the temperature is always more than its respective setpoint (factoring in a tolerance factor of 2F).

2. Finding Stuck Dampers: Finding zones where the dampers are stuck, i.e they do not modulate the amount of chilled air entering a zone. This application queries for all sensors with the field *zone damper* and the corresponding zone number (*zoneRef*), and checks whether its data stream remains constant or shows any variation.

3. Finding Inefficient Air Handling Units (AHUs): An AHU is considered "inefficient" if it serves rogue zones as well as over-cooled zones. Typically, a hot rogue zone drives the AHU to supply air that is too cold, resulting in other zones supplied by the same air handler always being too cold and uncomfortable. Identifying such AHUs may lead to making some over-cooled zones more comfortable. This application first computes whether a zone is over-heated or over-cooled using the same technique as the Rogue Zone application, and then queries for *ahuRef* (the air handling unit ID) field of each over-heated rogue zone, and checks if any of the other zones served by the same air handling unit is over-cooled. This application requires the same sensors as the Rogue Zone application, but requires an extra *ahuRef* relationship between the zones.

3.3 Evaluation

In this section we evaluate each part of our technique on the legacy metadata tags of three large commercial buildings. We manually ground truth-ed all sensor tags in three buildings with BMS' installed by different vendors, having 1586, 2522 and 1865[10] sensors respectively. We simulate the role of the expert in our experiments. When the simulated expert is asked for a parse, it consults the ground truth and provides the correct parse. A sensor tag is considered fully qualified if (a) the correct fields and field-values are extracted by the synthesized rules (b) No extra incorrect field is identified, and (c) the field-values explain every alphanumeric character of the sensor tag.

We first evaluate the advantage of our a priori clustering approach (3.3.1), and compare convergence with existing spreadsheet synthesis techniques. We next evaluate the effectiveness

[10] we did not have access to sensor stream data for this building

and convergence of our algorithm for different syntactic example selection mechanisms (3.3.2). We then compare and contrast the efficacy of the syntactic example selection method (3.3.3) to the data-driven example selection method (3.3.4) to parse enough sensor tags to run a particular application, following which we report the results of the portable applications on the buildings in our testbed (3.3.5).

3.3.1 Clustering

We evaluate the number of expert examples required for full qualification of all sensors in a building with and without apriori syntactic clustering. With clustering, the synthesized rules from the expert's parsed example are only applied to the cluster the example is in. Without clustering, the synthesized rules are applied to all sensors. We present the results from Building 3 in our dataset. We also compare the results to the parse generated by the spreadsheet synthesis algorithm presented in [107] (Flash-Fill).

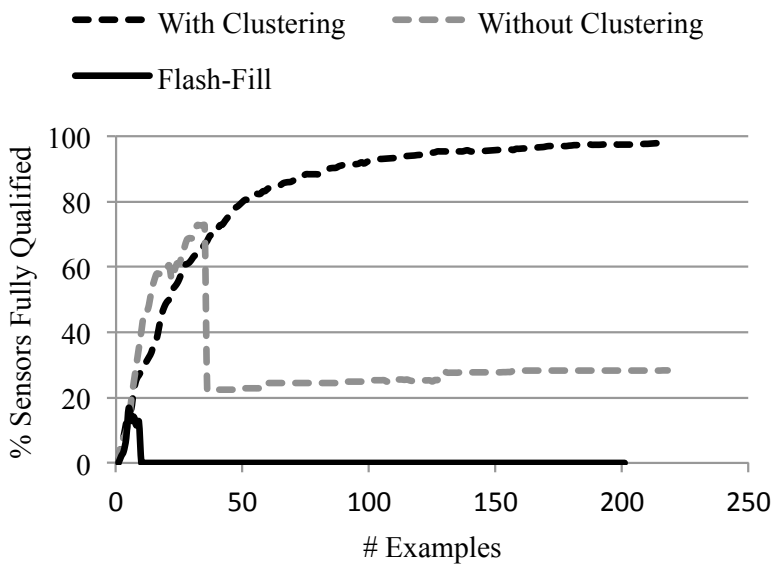


Figure 3.3: Advantage of apriori clustering in Building 3 : Rules are not over-generalized

Figure 3.3 shows the rate of full qualification in Building 3. Without clustering there is a sharp drop in the number of sensors fully qualified because erroneous fields start getting applied to sensor tags which were previously fully qualified. Figure 3.4 provides the intuition for this behavior. In all the buildings, a few fields (about 20 in each building) are applicable on a lot of sensors[11], while there is a long tail of fields applicable only to a handful of sensors. In the

[11]This is pretty common in commercial buildings, where a majority of the sensors are related to zone information. Thus, fields such as *zone temp setpoint*, *zone airflow*, etc are very common

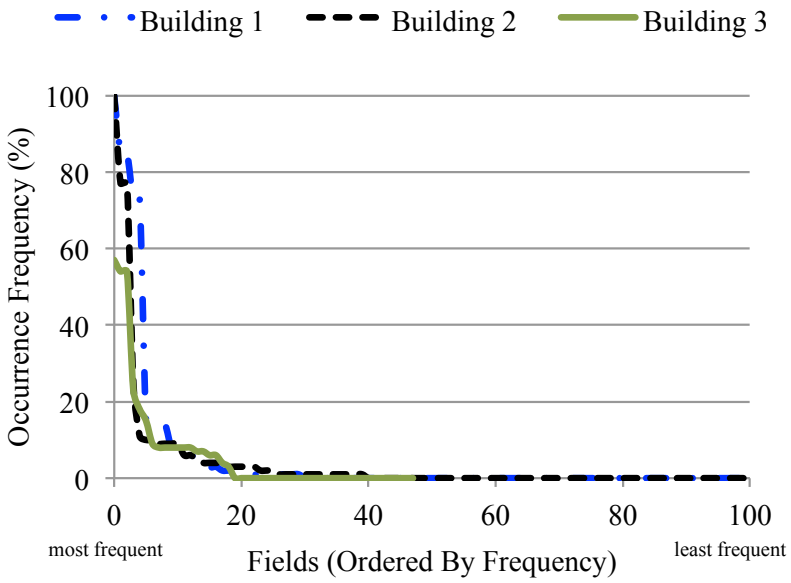


Figure 3.4: Percentage of sensor names (tags) each field appears in. The x-axis is sorted according to the frequency of occurrence of a field

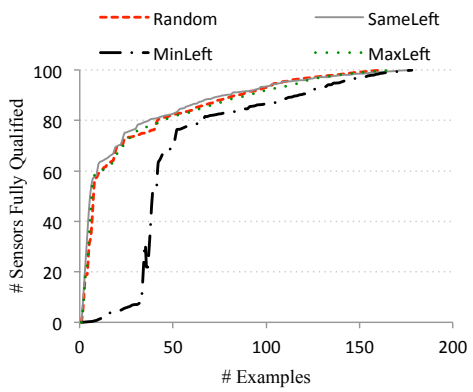
absence of clustering, the synthesis technique over-generalizes rules for these rare or erroneous fields. With clustering, 100% of the sensors are fully qualified. The apriori clustering step avoids over-generalization of the synthesized rules by restricting the rules to the cluster in which a particular example parsed by the expert lies. The spreadsheet-based synthesis technique (FlashFill) fails after 10 examples, because its underlying language and tokens is not robust enough to disambiguate the large number of fields.

Without the apriori clustering, Building 1 also showed a drop in the number of fully qualified sensor tags after 100 examples, settling at full qualification of 90% of the sensors. Building 2's sensors' schemas were much less noisy, and our synthesis technique could parse all sensor tags without clustering (these results are omitted due to space constraints).

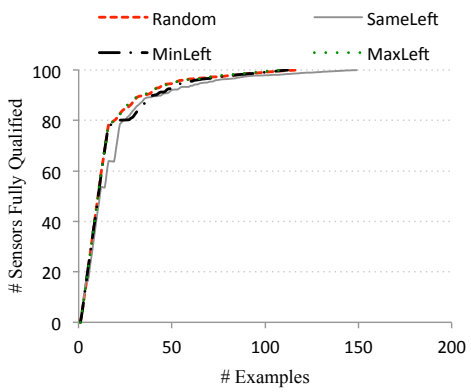
3.3.2 Syntactic Example Selection

We now study the selection methods (described in Section 3.2.3) used to select which example the expert should parse following the initial clustering.

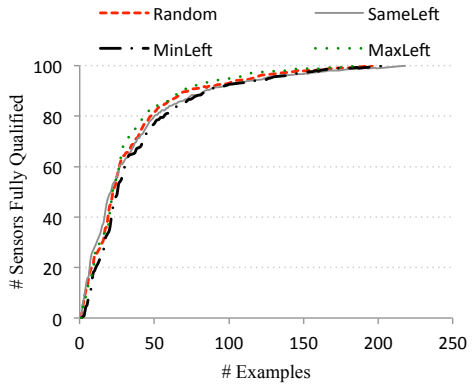
Figure 3.5 shows the rate of sensor qualification for each of the four example selection methods on all buildings. With the exception of *MinLeft* in Building 1, all the selection methods qualify sensors at roughly the same rate — they correctly classify the most frequently occurring sensors within a handful of examples. With addition of more examples, the rate of new sensors fully qualified decrease because the synthesis technique starts encountering obscure fields which are not applicable widely. *MinLeft* performs poorly for Building 1 because its metadata is very



(a) Building 1



(b) Building 2



(c) Building 3

Figure 3.5: Sensor qualification rate for the three buildings. The *Random* generator achieves 70% full sensor name qualification within 24 examples for Building 1, 15 examples for Building 2 and 43 examples for Building 3. It takes substantially more examples for Building 3 because its subsystems had no similarity in metadata because they were installed by different vendors.

noisy, and the approach gets stuck trying to fully qualify idiosyncratic sensors with obscure fields.

Thus, the convergence of the synthesis technique (i.e its ability to fully qualify all sensors) is not affected by decision of the example selection criteria. Also, the apriori clustering enables all the four selection criteria to quickly qualify the most frequently occurring sensors, since all of them seek out examples from the biggest clusters first.

The number of examples required for full qualification of all sensor tags in the three buildings using the *Random* method was 161, 116 and 196 respectively, and that for the *SameLeft* method was 176, 149 and 218 respectively. We use the *SameLeft* method in all the studies in the following subsections because of its deterministic nature.

3.3.3 Application-Oriented Qualification with Syntactic Example Selection

We present the number of examples required to obtain full-qualification of all sensors for the Rogue Zone application (developed in Section 3.2.5) in Building 1. The results for other applications on other buildings are similar, except when stated explicitly. The Rogue Zone application's set of *required sensors* are sensors with the fields *zone temp sensor* or *zone temp setpoint*. In Building 1 there are 462 such sensors, and their tags are in 10 different schemas, some of which are very frequent, while others not so.

Figure 3.6 shows that 147 examples are required to fully qualify all the *required sensors* for the Rogue Zone application (compared to 176 examples required for full qualification of all sensors). The first two examples fully qualify the *required sensors* (denoted by **P** in the Figure) with the most frequent schemas. *Required sensors* encoded in more obscure schemas with infrequent fields require more examples because the aim of the syntactic selection method is to select examples from large clusters. As the expert parses more examples, the number of fully qualified non-required (**N**) sensors increase steadily. Even though fewer expert examples are required for application-oriented qualification, the number of examples required is still prohibitively large to enable easy deployment of the Rogue Zone application.

The number of examples required to fully qualify all sensors for the Rogue Zone application on Building 2 was 67 (compared to 149 examples for qualifying all sensors). Similarly the Identifying Stuck Dampers application took 137 and 1 example for Buildings 1 and 2 respectively (all dampers in Building 2 were encoded with the same schema).

3.3.4 Application-Oriented Qualification with Data-Driven Example Selection

We now evaluate the number of examples required to fully qualify sensors for the same application and building as in the previous experiment (i.e Rogue Zone application on Building 1). We use the syntactic selection method to select the first five examples, so that the data-driven classifier has positive and negative instances in its training set, and thereon apply our data-driven classifier, and select the maximum likelihood *required sensor* to present to the expert for parsing.

Figure 3.7 shows that only 24 expert examples are required to obtain full qualification of all *required sensors*. After step 5, the data-driven classifier has 356 positive and 472 negative examples of fully qualified sensors in its training set. Out of the remaining 758 sensors (its test set, out of which 106 are *required sensors*), it classifies 231 sensors as *required sensors*, out of which 102 are true positive (denoted by **TP** in the Figure). It selects the example which it has classified as *required* with maximum likelihood.

Thus, data-driven example selection (after 5 steps of syntactic example selection) leads to a 6x reduction in the number of expert examples required compared to purely syntactic example selection, making it feasible to deploy the Rogue Zone application on Building 1 within a few minutes.

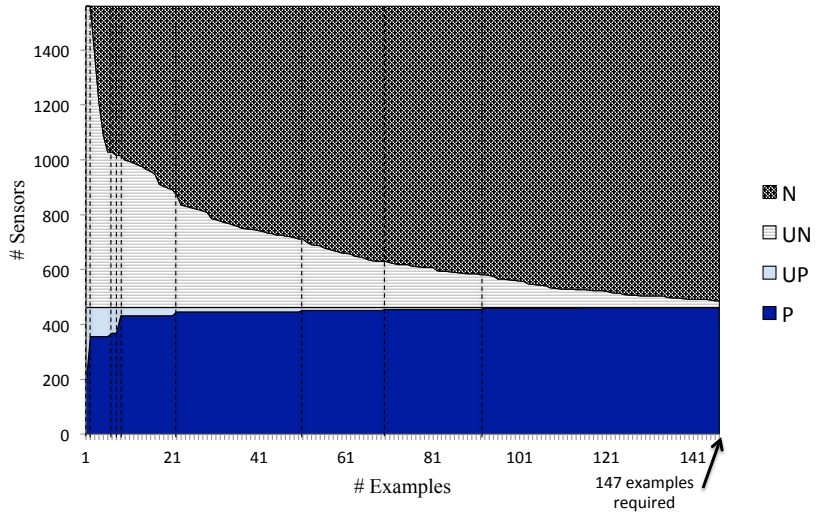


Figure 3.6: Rate of full qualification of *required sensors* for the Rogue Zone application on Building 1 using *sameLeft* example selection method. (**P**) denotes number of *required sensors* fully qualified (positive), **N** shows number of unrequired sensors fully qualified(negative), **UP** shows the number of *required sensors* yet to fully-qualified (unkown positive), and **UN** shows the number of non-required sensors yet to be fully-qualified (unkown negative). The dotted vertical line shows the steps in which a remaining *required sensor* example was presented to the expert.

However, one cannot apply the data-driven technique until the syntactic technique has identified at least one positive example for each required sensor. Figure 3.8 shows the number of examples needed[12] for full qualification of all *required sensors* for the Rogue Zone and Stuck Dampers application as a function of the initial number of syntactic example selections for Buildings 1 and 2. In general, fewer syntactic steps gave better results. This result was a surprise to us, because our intuition was that our algorithm would need to perform several syntactic steps to build up an adequate training set for the data-driven classifier. However, the *SameLeft* syntactic selection method always chose the first few example in a way to provide a sufficiently large training set for the data-driven example selection method to progress. Building 2's dampers were all encoded in the same schema, and hence all *required sensors* were parsed by the first syntactic example.

Note that in this experiment, our algorithm stops when the last *required sensor* is qualified. In practice, a heuristic would be required to infer that further positive examples are unlikely to be obtained. Thus, a small number of of additional examples would be required ensure convergence.

[12]the sum of data-driven example selection steps and syntactic example selection steps

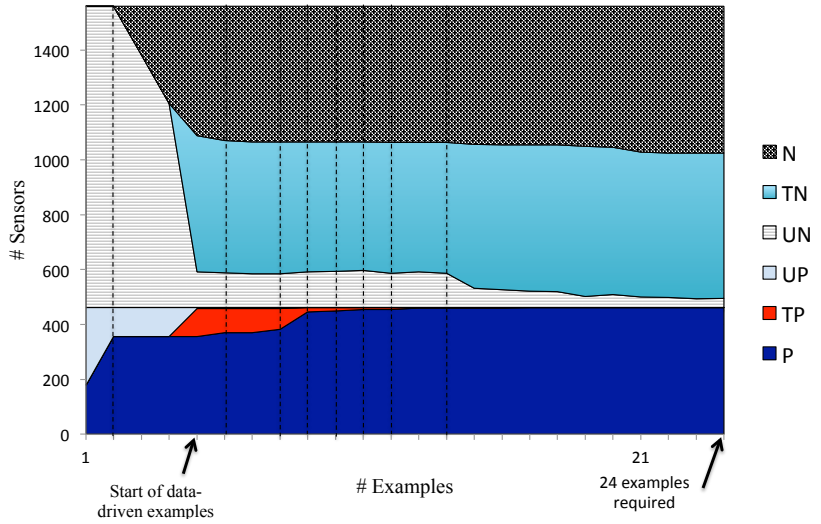
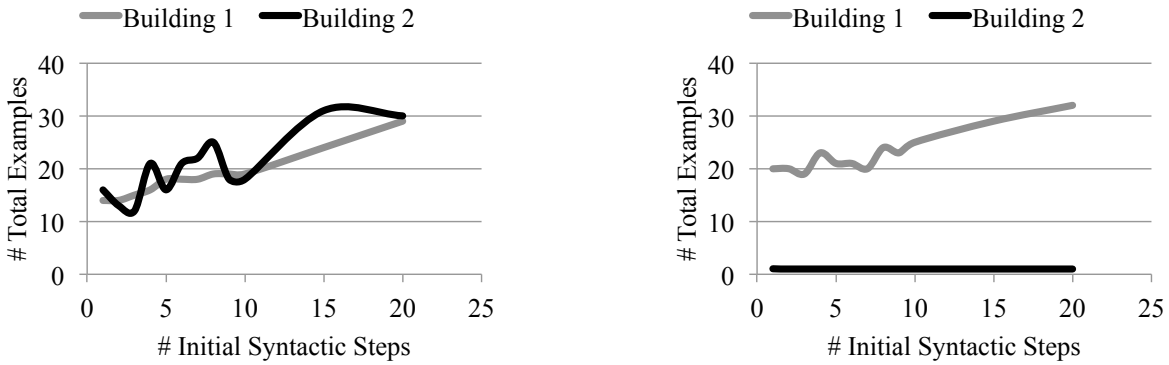


Figure 3.7: Rate of full qualification for Rogue Zone application in Building 1 using data-driven example selection after 5 steps of syntactic example selection. **P**, **N**, **UN** and **UP** is defined in Figure 3.6. **TP** indicates true-positive required sensors that have been labelled by the data-driven classifier at that step. **TN** shows the number of true negative *required sensor* identifications made by our data-driven classifier. In this case, **UP** is the same as false negatives and **UN** same as false positives, but this is unknown to the data-driven classifier. The dotted vertical line shows at which step a remaining *required sensor* example was presented to the expert.



(a) Rogue Zone Application

(b) Stuck Dampers Application

Figure 3.8: Total number of expert examples required to qualify all the *required sensors* for an application, as a function of the number of initial syntactic selection steps (the remaining steps have data-driven example selection).

3.3.5 Results of Applications

We ran our portable applications for finding rogue zones, stuck dampers and inefficient AHUs to Buildings 1 and 2 (Table 3.1), after the expert had parsed all the required sensors.

Results : We were able to identify *all* the zones and dampers in the two buildings. Building 1 was the more inefficient building with 5 hot rogue zones, and 17 over-cooled zones, and 4 zones having stuck dampers. We identified two inefficient air handling units in Building 1 which were trying to cool down extremely hot electrical closets and in the process over-cooling multiple office spaces. The inefficient AHU application could not run on Building 2 because none of the sensors encoded the relationship between zones and their corresponding AHUs.

Table 3.1: Application Results

	Building 1	Building 2
Number of Thermal Zones	201	78
Number of Rogue Zones	5	2
Number of Over-cooled Zones	17	0
Number of Zones with Dampers	175	55
Number of Zones with Stuck Dampers	5	0
Number of Air Handling Units	4	NA
Number of Inefficient AHUs	2	NA

3.4 Conclusion

In order to build meaningful applications at scale for buildings with disparate sensor metadata schemas, existing building sensor metadata schemas should be *augmented* and *normalized* to a common namespace to the extent possible. The normalization helps capture the semantic relationships between sensors, which is critical in enabling such applications.

We developed an approach which can normalize the primitive metadata of each building to field/field-value pairs in a desired common namespace using a few examples from an expert (e.g the facilities manager, often the only person familiar with the existing building metadata). We demonstrated that our synthesis technique is robust and achieves full sensor qualification for all sensors for 3 different BMS systems, even when presented with obscure and noisy tags. Our technique takes very few examples to fully qualify the most commonly occurring sensors (24, 15 and 43 examples for the three buildings in our testbed for qualifying 70% of the tags). We used the relationships inferred in the transformed metadata to run three unmodified analytics applications on two of the buildings. The data-driven example selection method reduces the effort to deploy a new application in an unknown building. The three applications could be deployed in under 30 examples from the building manager.

Chapter 4

Capturing Underspecified Legacy Metadata Through Active System Perturbation

The legacy metadata in buildings capture most, but often not all of the relationships required to run desired applications. For example, certain BMS systems may not capture which Air Handling Unit (AHU) is connected to which Variable Air Volume (VAV) Unit, or which chiller is connected to which AHU. There may be multiple AHUs, chillers, VAVs, or other subsystems in a large commercial building and the knowledge of their inter-relationships is required by many applications, e.g identifying errant AHUs or finding the root cause of thermal discomfort in certain zones.

The reasons for omission of these relationships from legacy BMS metadata can vary — it may be simple oversight, or maybe the pre-installed applications which were shipped with the BMS system did not require these relationships. In certain cases, these inter-relationships are captured but may even be incorrect. In this chapter we describe a general technique to deduce inter-relationships between various mechanical subsystems in a smart-building through careful impulse-response analysis. This work was done in collaboration with Marco Pritoni, UC Davis [185].

4.1 Overview Of Technique

Functional relationships between HVAC subsystems imply some physical media (such as air or water) is transferred from one subsystem to the other. Our insight is that if we perturb the control parameters of the upstream subsystem in a way that changes the property of the physical media reaching the subsystem downstream, the latter's control parameters will change noticeably to compensate for the change. For example, consider an AHU that supplies air at a certain temperature and pressure to a particular VAV, which ,in turn, supplies it to a specific room. The VAVs are generally equipped with dampers and local reheat to supply the air at just the right tempera-

ture and volume required to maintain the room's desired temperature setpoint. If we perturb the temperature or pressure of the air that the AHU sends the VAV, the latter would have to change its damper and reheat control parameters to compensate, since its control loop has been set up to maintain a constant room temperature. For each perturbation in the output of an upstream subsystem, we can quantify which downstream subsystem's control parameters changed and thus identify which pair of subsystems are connected. This technique can run during the normal operation of a building, totally transparently to its occupants.

Although our intuition is simple, there are certain challenges associated with perturbing the upstream subsystems (e.g an AHU) and picking up the resulting signals in the downstream subsystems (e.g a VAV). First, the perturbations have to be small so as not to affect the daily operation of the subsystems. Second, there can be confounding factors such as external environment, occupancy, etc that obscure the results of the perturbation in each subsystem. Finally, the signal to noise ratio may be low, making it really hard to pick up the resulting signal in the downstream subsystem when the upstream system is perturbed only by a small amount.

We solve the first two challenges through careful experiment design and the third through repeated perturbations. Our experiment takes in the maximum deviation of control parameters of the upstream system allowed by a building manager. We perform two step changes — the first setting the value of the control parameter to the lowest allowable limit, and then a day later to the highest allowable limit. This allows us to get an intermediate step change of twice the allowed deviation. Operating each regime over an entire day and over consecutive days also helps us eliminate confounding variables such as outside air temperature, occupancy, etc to a large extent. Finally, we repeat this experiment, once for each upstream subsystem. We attribute the downstream subsystem to the upstream subsystem whose perturbation has caused the maximum change in the latter's control parameters.

We next describe details of our technique in the context of finding the functional relationships between AHUs and VAVs in a large commercial building.

4.2 Problem Instance

We analyze data from a large commercial building with 3 chillers, 4 air-handling units (AHU) and 179 thermal zones. A variable air volume (VAV) box modulates the airflow from the AHU to each zone. One AHU is associated to multiple VAVs. VAV modulation is achieved through a combination of two control actions: adjusting the air damper position (`DMP`) and regulating the local reheat valve (`RVP`) (Figure 4.1). The temperature in each zone (T_{zone}) is influenced also by the supply air temperature (T_{sa}) and flow (FLW_{sa}) coming from the AHU. All these points are monitored and recorded by the BMS system, and represent the datasets used here. In addition, the room temperature is impacted by uncontrolled variables, such as weather (T_{oat} is the outdoor air temperature), internal gains, and other thermal gains. This configuration is typical of a large number of buildings. The association between AHU and VAV is not stored in the BMS system; however, for the purpose of the experiment, ground truth association was collected from building blueprints to verify the results of the proposed method.

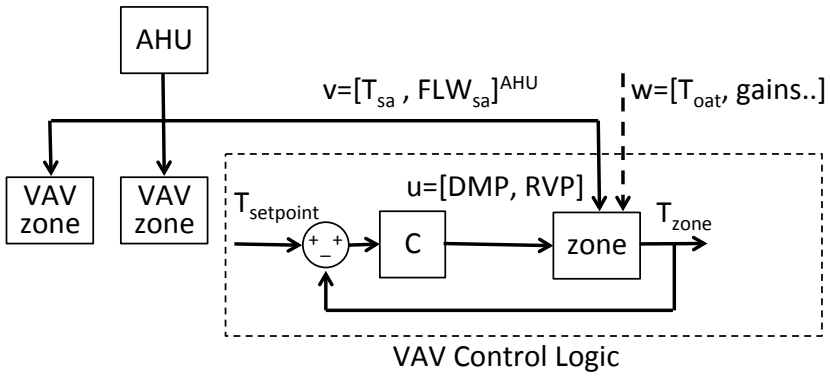


Figure 4.1: AHU and VAV configuration in our building testbed. On the right, the VAV control logic is represented. T_{zone} is kept around the $T_{setpoint}$ by the control inputs $u = [\texttt{DMP},\texttt{RVP}]$. T_{zone} it is also influenced by parameters controlled by the AHU ($v = [T_{sa}, FLW_{sa}]^{AHU}$) and external disturbances w=[T_{oat}, internal gains].

4.3 Prior Techniques

Techniques commonly employed to detect relationships in data streams include: correlation of raw data [141], correlation of transformed data [119], principal component analysis (PCA) and clustering [161, 117], statistical process control [226], and model-based system identification (i.e., building a model and looking for the best fit from a VAV to an AHU). We also tested conventional correlation methods to find relationships between AHU and the corresponding VAV boxes. Table 4.1 shows an example of these coefficients. Results show very poor correlation. Desired values inside the bold boxes should be higher, in absolute value, than the corresponding values in the other rows. The same test was repeated with data resampled at 5 min, 15 min and daily, yielding similar results. Thus, this method does not allow identifying which VAV boxes are connected to which AHU.

Table 4.1: Correlation matrix (showing raw data from two AHU and two VAV boxes).

		Example VAV_{AHU3}			Example VAV_{AHU5}		
		T_{zone}	DMP	RVP	T_{zone}	DMP	RVP
AHU 3	T_{sa}	-0.06	0.06	0.11	-0.11	0.24	0.29
AHU 5	T_{sa}	-0.20	-0.19	-0.06	-0.04	-0.01	0.02

Unlike prior research [119], the variable measured in this test are pressures, temperatures, flows and actuator positions, which show delayed and attenuated responses to changes in input variables, thus reducing correlation. Further, AHU and VAV boxes/zones are physically distant (differently from [141]), and variables in the latter are significantly influenced by additional measured and non-measured disturbances (Figure 4.1). For this reason, sensor values show a small signal to noise ratio. In addition, sensor readings are frequently constrained between

physical limits (e.g. max damper position) and kept around setpoints by nested control loops. Also, cross-talk between systems (i.e., zones might influence each other) and similarity in the way different AHU are controlled (setpoints and daily behavior are similar) make correlation of raw data ineffective. Next, we construct feature vectors for the data of each VAV. Features included were T_{zone}, DMP, RVP, $T_{setpoint}$, measured flow (FLW), flow setpoint (FWS), day of the week, time of the day. We apply Principal Component Analysis to these feature vectors to identify the two principal components and correlate them to the T_{sa}. Unfortunately, results of this analysis is not very different from what we obtain with the raw data (Table 4.1). Again, this method is ineffective in inferring the desired functional relationship.

Finally, we test a completely different and novel approach involving system identification (SID) techniques. SID is often used in control engineering to find a mathematical relationship (model) between inputs and outputs variables in an observed system [5]. We construct a physics-inspired black-box dynamical model to predict T_{zone}, based on the available sensor data:

$$T_{zone,t} = \beta_1 T_{zone,t-1} + FLW_t * \sum_{t}^{t-k} \beta_{2i} RVP_i + \beta_3 FLW_t * T_{sa,t}^{AHU} + \beta_4 * T_{oat,t}$$

where variable names are defined above, β_i are the statistical coefficients, and t stands for time. The equation is in the form of an autoregressive (AR) time series model with exogenous inputs and interactive effects (some variables are multiplied). The term with the sum represents the lag in the effect of the reheat valve. The model is based detailed physical knowledge of the heat transfer processes in VAV boxes. Note that all the variables in this equation belong to the VAV with the exception of ($T_{sa,t}^{AHU}$), that represents the supply air temperature controlled by the AHU at time t. The idea is that using the $T_{sa,t}^{AHU}$ from the AHU actually connected to each VAV would improve the model fit. Both linear regression and lasso [211] are used to fit the model over 15-min resampled data.

While this model fits the data very well (R^2=76-95% depending on the zone), it fails to capture the difference in AHU. Plugging in different $T_{sa,t}^{AHU}$ did not change the fit of the model as we had expected, because the majority of the variation in the output variable $T_{zone,t}$ is captured by the first term of the model (zone temperature at the previous time step) and the remaining β_i coefficients are relatively small. With the calculated coefficients, the input variable $T_{sa,t}^{AHU}$ would have to change by more than 20F to produce measurable effects in the output variable. Such large temperature differential never occurs spontaneously in our recorded data, and if artificially produced would seriously compromise occupantsâĂŹ comfort.

4.4 Mechanism of Perturbation

To obtain the functional relationships between AHUs and VAVs, we perturb each AHU so that we can generate a distinguishable signal in the connected VAV boxes. In practice, by changing the supply air temperature in an AHU (input), the VAV box will respond by changing some controlled variables (reheat and damper position) to maintain its desired output, i.e temperature

setpoint for a particular zone. [1] The AHU supply air temperature was chosen over the AHU flow rate as it is practically easier to tune. Also significantly perturbing flow rates might provide insufficient ventilation or damage the ducts due to high pressure.

We took care to ensure that the perturbation had no impact on occupant comfort. The temperature setpoint for the supply air in the AHU is set one day to 52F (cold mode) and the following day to 60F (hot mode), thus deviating a maximum of 5F from the normal setpoint for a week-day of 57F. Since thermal systems have a significant response lag, the perturbation was sustained for a full day in each mode. This also helps reduce the effect of confounding factors such as occupancy and external environment effects. We then computed the average daily values for `RVP`, `DMP` and T_{zone} for each zone.

The perturbation algorithm can be described in the following steps:

1. Perturb the supply air temperature of an AHU at the time for two consecutive weekdays, one day in cold mode (52F) and one day in hot mode (57F).
2. Collect data for each VAV for the following sensors: `RVP`, T_{zone}, `DMP`.
3. Obtain the daily average for each of these sensors for every VAV.
4. For each VAV and each day of perturbation combine data into average daily vectors of the form [`RVP`, `DMP`, T_{zone}].
5. For each VAV calculate the Euclidian distance between vectors on the hot mode day and cold mode day. At the end of this step each zone will have a metric for the AHU hot-cold perturbation.
6. Repeat steps 1-5 for each AHU.
7. For each VAV take a vote to select the AHU whose perturbation has a highest metric (calculated in step 5). The AHU selected with this method is associated with that VAV box.

Even though, in theory, VAVs which are not connected to the perturbed AHU should show no change in their average values over the two days, we found that this did not hold up in practice. External confounding factors resulted in most VAVs showing some deviation in their behavior during each 2-day experiment. Hence, we adopt a voting method where we attribute a VAV to the AHU whose perturbation caused the largest deviation between the hold and cold mode days.

[1] If the operating conditions fall outside a VAV box's control regime and it is unable to maintain its desired output and keep up with zone cooling/heating load, the zone temperature (T_{zone}) will be affected and change

4.5 Results

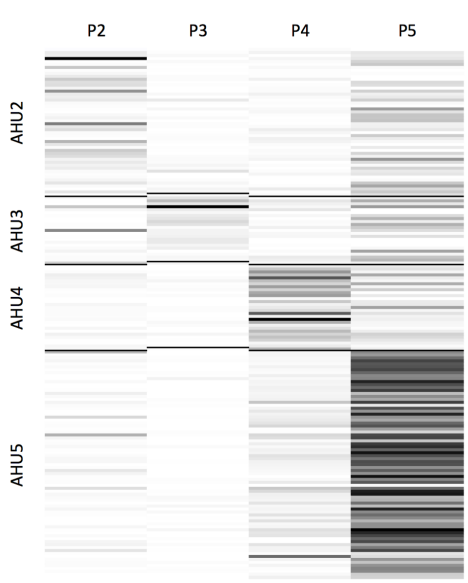
We applied the algorithm described in the previous section to our testbed building. As shown in Figure 4.2 our algorithm correctly identified the relationship between VAV box and AHU in 79% of the cases, a 2x improvement over all prior methods.

Technique	AHU Attribution Details	Accuracy
Random	Randomly allocate VAV boxes to AHUs	25%
Correlation (Raw)	AHU whose supply air temperature has max. correlation to VAV sensors	38%
Correlation (PCA)	Same as above but with principal components	32%
State Identification	AHU whose supply air temp gives lowest error	32%
Perturbation + Voting	Perturb all AHUs and attribute VAV to AHU which caused max internal perturbation	79%

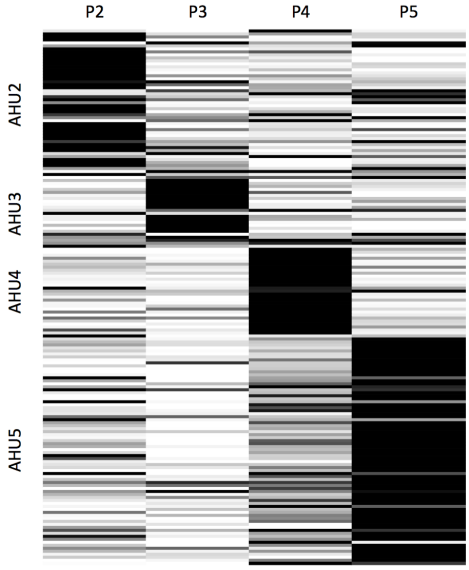
Figure 4.2: Comparison of our technique against prior methods

Figure 4.3 shows the advantage of using voting to attribute VAVs to AHUs. Figure 4.3a shows a case where we simply attributed a relationship without voting, if our difference metric exceeded a pre-defined threshold magnitude. Choosing the right threshold is challenging and hard to generalize. We show results for a threshold magnitude of 2, but our results show similar performance for other thresholds as well. Even though a lot of zones were in AHU5 showed large deviations when AHU5 was perturbed, the same cannot be said for AHUs 2, 3 and 4. With voting however, it becomes much more apparent which AHU a VAV belongs to (Figure 4.3b), because we are taking into account zone characteristics and comparing the difference metric across different perturbation periods for the same zone.

There were some zones that were misclassified though. These zones could be grouped into three categories: (a) VAVs that showed no response to any perturbation, (b) VAVs that had a unexpected behavior, such as counter-intuitive responses to perturbations, and (c) VAVs that responded more strongly to the perturbation of the 'wrong' AHU. For instance a zone had very frequent oscillations of its damper and reheat valve position (about 20 cycles per day) ranging between values of 0% and 40%. This behavior shows that the local VAV control system is misconfigured, resulting in unexpected behavior during our perturbation experiments. Some



(a) Each perturbation as a standalone experiment. Our metric – the deviation magnitude of the vector — is normalized across all zones for the same perturbation experiment.



(b) Application of voting across perturbation periods. Our metric – the deviation magnitude of the vector — is normalized for each zone across its values for the different perturbation periods, and not against all other zones.

Figure 4.3: Overview of perturbation statistics across all zones in a building. The zones are sorted and grouped according to their ground-truth AHU data on the y-axis. The x-axis shows the 4 different perturbation experiments, Pi is the perturbation experiment for AHU i.

zones were seldom occupied and had control parameters set to never maintain a reasonable zone temperature. Our perturbation experiment had little effect on those zones.

4.6 Conclusion

We showed a novel algorithm to infer functional relationships between HVAC components of large commercial buildings. Our technique does not impact the occupants of the building under perturbation, and is able to identify the required relationships correctly in 80% of the cases.

Chapter 5

Designing BRICK- A Combined, Unified Metadata Schema

Designing a comprehensive schema for all IoT sensors in order to run any possible application in any context is a difficult problem. Instead, we focus on creating an information exchange platform that is focused on commercial buildings where interactions among devices and people are core to sophisticated applications. In building such a platform, we are guided by the sensors, attributes and relationships that have been shown to be useful in the published literature with a view towards composability and extensibility as laid out in Section 2.4. Our schema — Brick, is evaluated on the same empirical criteria, which we restate here:

- **Completeness:** Can Brick represent all the sensor metadata information (such as a sensor's location, type, etc.) contained in a building's BMS?
- **Expressiveness:** Can Brick capture all important relationships between sensors that are (a) (overtly or tacitly) mentioned in a building's BMS, and (b) expressed in canonical smart-building applications in published academic literature?
- **Usability:** Can Brick represent the information in a way that is easy to use for both the domain expert and the application developer? Can the schema provide support automation with machine readable data formats and querying tools?

Brick builds upon prior work in several ways. We utilize the tagging concept of Haystack and extend it with mechanisms to model relationships and entities. We use the location concepts from IFC. We use a semantic representation to utilize its flexibility and extensibility properties. The semantics allows us to formalize, restrict, and verify the usage of tags, entities, and relationships.

Our design of Brick is grounded by the information from BMS across five buildings spread across two continents, comprising more than 615,000 sq-ft of floor space and more than 15,700 data points, whose BMS systems were set up by different vendors, and have vastly varying subsystems and sensors. We further refine our design requirements using eight canonical building applications that require integrated information across commonly isolated building subsystems: HVAC, lighting, spatial and power infrastructure.

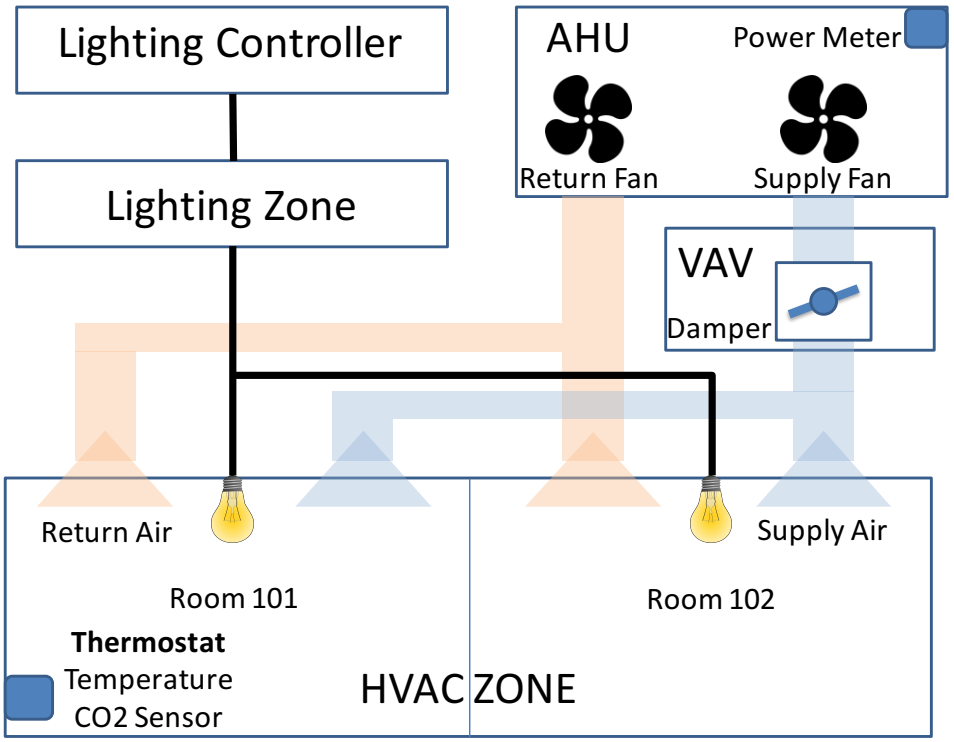


Figure 5.1: A simple example building that highlights the components to be modeled in a building schema.

We demonstrate that 98% of BMS data points across our five buildings can be mapped to Brick, and our eight applications can easily query the mapped building instances for required information. We open source the Brick schema files, the BMS metadata from our buildings, the application queries that run on top of Brick and tutorials on how to map existing building metadata to Brick. Brick was developed in collaboration with researchers at UC San Diego, UC Los Angeles, CMU, IBM Research in Ireland, University of Virginia, Southern Denmark University and UC Berkeley [23].

5.1 An Example Building - Sensors and relationships

We start with a hypothetical building to understand the requirements of uniform building data representation, outlining the current state of the art. Figure 5.1 shows the major components of this building that are of interest: an Air Handler Unit (AHU) supplying conditioned air to a Variable Air Volume Box (VAV), which modulates the air provided to an HVAC zone consisting of two rooms. The HVAC zone has a thermostat that contains a temperature and CO2 sensor. The same two rooms are part of a Lighting Zone, and the building Lighting Controller controls the zone lights based on a schedule. We model the HVAC and lighting systems as examples

because these are the systems commonly found in a modern BMS.

At the very minimum, a schema should be able to model the components illustrated in Figure 5.1 as well as their relevant points such as temperature sensors and their related control parameters. More realistically, we should be able to model diverse infrastructure systems such as HVAC, lighting, water, and express the specifics of each installation as per end use and vendor requirements. For example, the AHU in an HVAC system can consist of equipment such as fans, pumps, heat exchangers, humidifiers, valves and dampers. Each component could have further types, e.g., fans can have types: supply fan, return fan, exhaust fan, and each fan would have its associated sensors measuring its speed, air flow and power consumption. In addition to this heterogeneity, the vendor may choose not to install certain sensors, or to expose esoteric data points whose functionality is unclear to others.

5.2 Schema Design

5.2.1 Design Principles

Brick's design focuses on the metadata and data points found in real building deployments and requirements defined by end use applications. We obtain ground truth information from five diverse buildings across the US and Europe, which have 15,700 data points and five different vendors in total (Table 5.4). We pick eight popular applications from the list of smart building applications compiled by Bhattacharya et al. [39], and formulate metadata queries for these applications to drive the basic requirements of Brick as well as evaluate how well our building metadata can be mapped to Brick. Section 5.5 contains our initial findings for the five buildings evaluated thus far.

We have used terminology and organized the basic concepts in a way that is consistent with BMS deployments in our buildings and the vocabulary used by the building managers at our respective institutions. We follow standard ontology design methods so that developers can leverage available tools for data formatting (e.g., Turtle [9]) and querying (e.g., SPARQL [7]).

5.2.2 Tags and Tagsets

We borrow the concept of *tags* from Project Haystack [4] (Section 6.2.4) to preserve the flexibility and ease of use of annotating metadata. We enrich the tags with an underlying ontology that crystallizes the concepts defined by the tags and provides a framework to create the hierarchies, relationships and properties essential for describing building metadata. With an ontology, we can analyze the metadata using standard tools and place restrictions to prohibit arbitrary tag combinations or relationships. For example, we can restrict the units of temperature sensors to Fahrenheit and Celsius.

We introduce the concept of a *tagset* that groups together relevant tags to represent entities. With Haystack and related tagging ontologies [49], an entity such as `zone temperature sensor` from Figure 5.1 is defined by its individual tags, so its properties and relationships

Relationship / Inverse	Transitive?	Definition	Endpoints
`contains` / `isLocatedIn`	Yes	`A` physically encapsulates `B`	Loc. / Sensor Loc. / Equip.
`controls` / `isControlledBy`	No	`A` determines or affects the internal state of `B`	Function Block / Equip.
`hasPart` / `isPartOf`	Yes	`A` has some component or part `B` (typically mechanical)	Equip. / Sensor Equip. / Equip. Loc. / Loc.
`hasPoint` / `isPointOf`	No	`A` is measured by or is otherwise represented by point `B`	Equip. / Sensor Loc. / Sensor
`feeds` / `isFedBy`	Yes	`A` "flows" or is connected to `B`	Function Block / Equip. Equip. / Equip.
`hasInput` / `isInputOf`	No	Function `A` has an input `B`	Function Block / Sensor
`hasOutput` / `isOutputOf`	No	Function `A` has an output `B`	Function Block / Sensor

Table 5.1: List of the Brick relationships and their definitions. All definitions follow the form `A <relationship> B`, where `relationship` is the first one listed, not the inverse. All Brick relationships are asymmetric, and transitive where marked. If a relationship $\rightarrow$ is transitive, then if $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$ is a valid relation. Asymmetric simply means that if $A \rightarrow B$, then $B \rightarrow A$ is invalid.

with other entities can only be specified at the tag level. With tagsets, we have a cohesive concept of a `zone temperature sensor`, and we can specify that the temperature is maintained between its `cooling setpoint` and `heating setpoint`. The concept of tagsets works well with an ontology class hierarchy - a `zone temperature sensor` is a subclass of a generic `temperature sensor`, and will automatically inherit all its properties. Further, we avoid use of complex tags such as `chilledWaterCool` and `hotWaterReheat` in Haystack. The vocabulary of Brick is defined by its list of tagsets.

5.2.3 Class Hierarchies

We define several high level concepts that provide the scaffolding for Brick's class hierarchy. As the central emphasis of our design is on representing points in the BMS, we introduce *Point* as a class, with subclasses defining specific types of points: Sensor, Setpoint, Command, Status, Alarm. Each point can have several attributes, and we divide them into *properties* and *relationships*. Properties are attributes that provide specifics about the point: units, data type, etc. Relationships are attributes that relate the data point to other classes: its location, equipment it belongs to, etc.

We define three concepts as high level classes to which a Point can be related to: Location, Equipment and Measurement (Figure 5.2). We can expand these concepts in future versions to expand the metadata covered by Brick (e.g. Network, People). 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 have their own subclasses. Figure 5.3 showcases a sample of Brick's class hierarchy.

It is common in a domain to use multiple terminologies for the same entity. For example, in HVAC systems, `Supply Air Temperature` and `Discharge Air Temperature` are used interchangeably. We identify these synonyms from our ground truth buildings, and mark

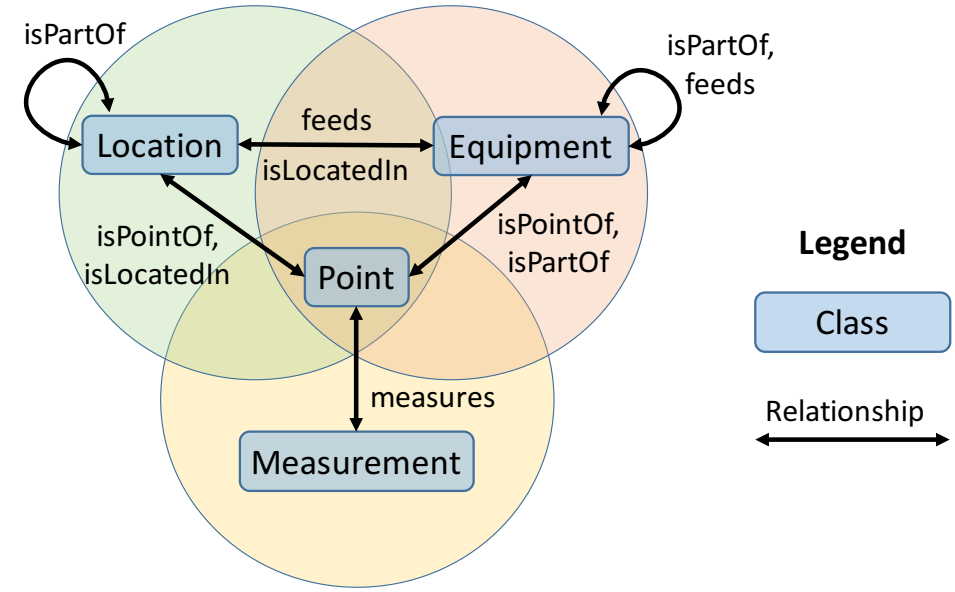


Figure 5.2: Information concepts in Brick and their relationship to a data point.

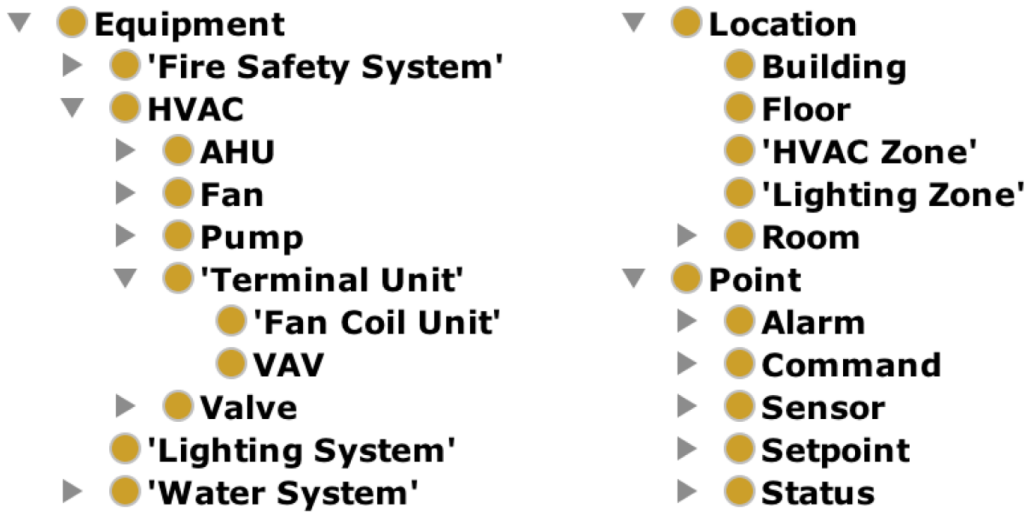


Figure 5.3: A subset of the Brick class hierarchy

the corresponding tagsets as being equivalent classes in Brick. Note that the class hierarchy does not strictly follow a tree structure, and we use multiple inheritance when appropriate. For example, a desk lamp can be a subclass of both the lighting system and office appliance classes.

5.2.4 Fundamental Relationships

Relationships connect the different entities in the building and are essential to providing adequate context for many applications. For example, an HVAC fault detection app running on our example building (Figure 5.1) needs to know the room in which the temperature sensor is located, the corresponding temperature setpoint and the status of the VAV that supplies conditioned air to this room.

Table 5.1 defines the basic set of relationships in Brick. We have designed these relationships to be minimal, multipurpose and intuitive so that it is easy for a user to specify a particular relationship. The `isPartOf` relationship is designed to capture the compositions among the entities in the building. For example, a room `isPartOf` a floor, an AHU `isPartOf` the HVAC system. The `feeds` relationship captures the different *flows* in the building - flow of air from AHU to VAV, flow of water from a tank to a tap or flow of electricity from a circuit panel to an outlet. Each of these relationships can have sub-properties. For instance, `feeds` can be extended to `feedsAirTo`, `feedsWaterTo`, etc. Figure 5.4 shows the relationships for a subset of example building in Figure 5.1.

The Brick schema includes possible relationships among classes as a guideline for users to add relationships to their instances. For example, using ontology class restrictions we say that a VAV can have points like `zone temperature sensor`, `discharge air flow setpoint`, `reheat valve command`, and it can have other equipment as its components such as `damper` and `reheat valve`. These can be exploited by a user interface to guide users while tagging raw metadata or while establishing relationships between entities. Note that we do not enforce these restrictions to enable the flexibility to compose building metadata as per user requirements.

5.2.5 Function Blocks

The tags, tagsets, class hierarchies and fundamental relationships provide sufficient expressiveness to describe our building metadata and direct relationships. However, 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.

Consider a heat exchanger as an example: a heat exchanger is an equipment that transfers heat between two fluid (air/water/steam) flows – a primary and a secondary. The primary is intended to heat or cool the secondary. It may be considered a simple piece of equipment with primary input, primary output, secondary input and secondary output. The primary input `feeds` the primary output, and it `transfersHeatTo` the secondary output.

A heat exchanger can be defined as a subclass of a function block and modeled as illustrated in Figure 5.5. Notice how the instance objects receive relations through the class they are in-

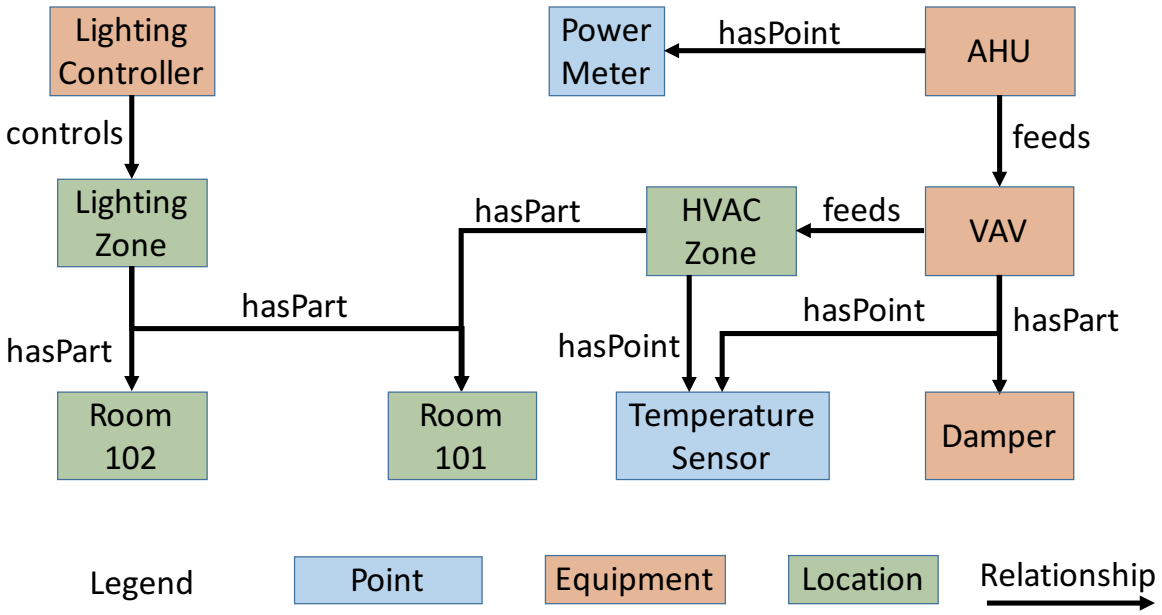


Figure 5.4: Brick classes and relationships for a subset of the example building in Figure 5.1.

stance of. Similarly, more complex equipment, like VAVs, can be constructed as function blocks and manipulated at this level.

5.3 RDF and SPARQL

Brick represents knowledge as a graph of entities (nodes) connected by relationships (directed edges). This section briefly describes how Brick uses the RDF format to represent its knowledge, and how this knowledge is traversed and queried using SPARQL.

5.3.1 Representing Knowledge in RDF

Brick adheres to the RDF (Resource Description Framework) data model [148], which represents knowledge as a graph expressed as tuples of *subject-predicate-object* known as *triples*. All buildings in Brick consist of a collection of such triples. A triple states that some *subject* entity has some relationship *predicate* to some other entity *object* — essentially a directed edge in a graph. This simple structure enables the succinct and elegant composition of the large, interconnected structures typical of building subsystems.

All entities and relationships exist in some namespace, indicated by a `namespace:` prefix. Brick takes advantage of the standard RDF [5], RDFS [6] and OWL [3] namespaces, which come with their own graphs defining entities, relationships and restrictions.

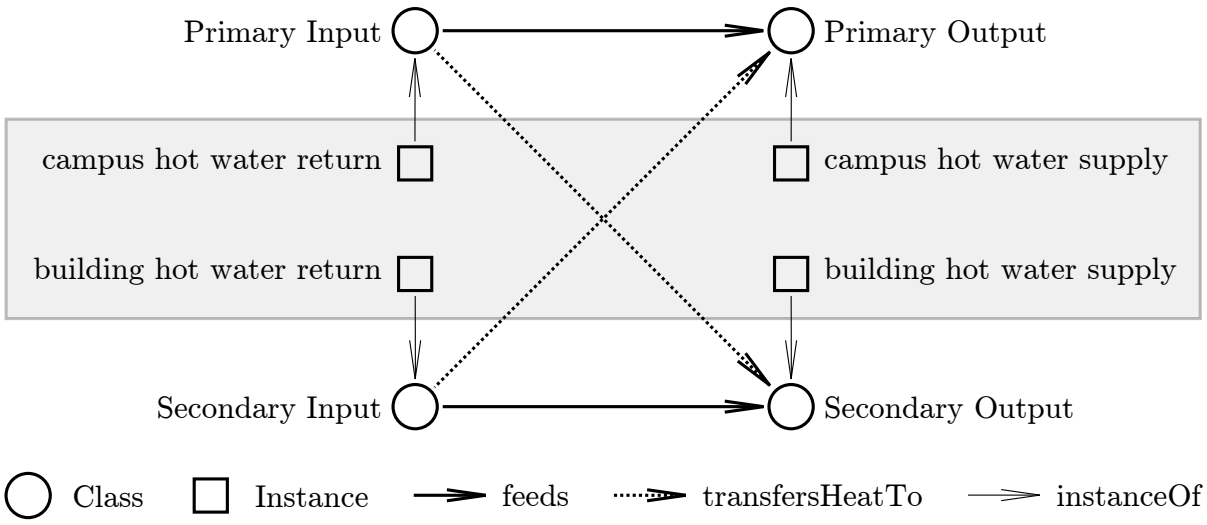


Figure 5.5: An example of a heat exchanger modeled in gray as a function block.

```
1 example:myVAV rdf:type brick:VAV
2 example:myTempSensor rdf:type brick:Zone_Temperature_Sensor
3 example:myVAV brick:hasPoint example:myTempSensor
```

Figure 5.6: RDF triples instantiating a VAV and a Temperature Sensor and declaring that the VAV measures temperature via that sensor.

The collection of triples in Figure 5.6 gives the representation of the connection of the VAV to the Temperature Sensor using the `hasPoint` relationship from the building graph in Figure 5.4. Line 1 declares an entity identified by the label `example:myVAV`: this creates the `myVAV` entity in the `example` namespace. All entities are implicitly created the first time they are mentioned. `brick:VAV` is a class defined by the Brick ontology that represents a variable air-volume box. The use of the `rdf:type` relationship declares that `example:myVAV` is a `brick:VAV`. Similarly, line 2 of Figure 5.6 instantiates a Zone Temperature Sensor. Line 3 uses the Brick relationship `brick:hasPoint` to declare that `example:myVAV` is associated with the given temperature sensor.

5.3.2 Querying Knowledge with SPARQL

Applications query the Brick graph for entities and relationships using SPARQL (SPARQL Protocol and RDF Query Language) [7]. SPARQL queries specify constraints and patterns of triples, and traverse an underlying RDF graph to return those that match. For Brick applications, this underlying graph consists of all the entities and relationships in that building.

While SPARQL has many features, Brick is simple enough to support all of our intended applications with a simple subset of SPARQL. Figure 5.7, a query for retrieving all rooms which

```
1 example:myVAV rdf:type brick:VAV
2 example:myTempSensor rdf:type brick:Zone_Temperature_Sensor
3 example:myVAV brick:hasPoint example:myTempSensor
```

Figure 5.7: A simple SPARQL query for retrieving all rooms connected to a given Air Handling Unit (AHU).

are connected to a given AHU, contains a representative example of each of these features.

Lines 1-3 declare the prefixes for the various namespaces to shorten the references to entities; for length, we omit these from all later queries in this work. Line 4 contains the `SELECT` clause, which states that the variables `?ahu` and `?room` should be returned (the `?` prefix indicates a variable). The `WHERE` clause determines the types and constraints on these variables. Line 6 states that `?zone` is any entity in the graph that is an instance of the class `brick:HVAC_Zone`. Likewise, line 7 declares `?room` to be an instance of a `brick:Room`.

Brick provides both generic (such as **AHU**) and specific classes of equipment (such as a **RoofTop-Unit** AHU). A building represented in Brick can specify the specific subclasses, or if that information is not available, instantiate a generic class. Line 8 is a common construct in Brick queries which accounts for this type of uncertainty in how Brick represents buildings. This sub-query returns all entities `?ahu` that are either an instance of a subclass of `brick:AHU` or an instance of `brick:AHU` itself.

An application that does not require specific features of such subclasses may want to query for the generic class rather than exhaustively specify every possible subclass. Because SPARQL and RDF do not support the object-oriented programming style of classes, the SPARQL query itself must specify the semantics of the type-inheritance: entities that instantiate the generic class directly, or entities that instantiate a subclass of the generic class.

After declaring the types of the entities involved, the query restricts the set of relationships between the entities on lines 9 and 10 to determine which pairs of entities are connected. Line 9 finds all HVAC zones downstream of a particular AHU by following a chain of `brick:feeds` relationships (the + indicates that 1 or more edges can be traversed as long as the edges are of type `brick:feeds`). Line 10 links the identified HVAC zones with the rooms they contain. The correct relationships to use can be determined from the Brick relationship list (Table 5.1).

This example query also illustrates an important quality of Brick queries: establishing a link between two entities (even across different subsystems such as HVAC and spatial) does not require explicit knowledge of all intermediary entities. Rather, the query denotes the relevant entities and relationships: the query in Figure 5.7 is indifferent to whatever building-specific equipment and details lie between an Air Handler Unit and the end zones. This is possible because the relationships between those entities all use Brick's `brick:feeds` relationship. What's more, the query accomplishes this using only a few, straightforward expressions to return the relevant triples from the collection of thousands of entities and relationships present in the building.

5.4 Applications

In this section, we construct this set by pulling a representative example from each of the eight common application dimensions identified in Section 2.4, specifically in Table 2.1. We determine the effectiveness of the schema to be how many of these entities and relationships it can capture.

Entities	Occupancy Modeling [128]	Energy Apportionment [123]	Web Displays [25]	Model-Predictive Control [208]	Participatory Feedback [143]	Fault Detection and Diagnosis [198]	NILM [153]	Demand-Response [223]
Sensors								
Temp Sensor	X					X		
CO2 Sensor	X							
Occ Sensor	X	X			X			
Lux Sensor		X			X			
Power Meter	X	X	X		X		X	X
Airflow Sensor			X					
Equipment								
Generic							X	X
HVAC	X	X	X			X		
Lighting	X	X			X			
Reheat Valve			X			X		
VAV			X	X				
AHU				X		X		
Chilled Water			X			X		
Hot Water			X			X		
Locations								
Building				X		X		
Floor				X	X		X	
Room	X	X	X	X	X		X	
HVAC Zone	X		X	X				
Lighting Zone	X				X			
Relationships								
Sensor `isLocIn` Loc.	X	X			X		X	
Equip `isLocIn` Loc.		X			X		X	X
Loc. `hasPart` Loc.		X		X	X			
Loc. `hasPoint` Sensor	X	X			X		X	X
Equip `hasPoint` Sensor	X		X			X	X	X
Equip `hasPart` Sensor			X			X	X	X
Equip `feeds` Zone	X			X	X			
Equip `feeds` Room	X			X			X	
Equip `feeds` Equip			X	X			X	
Zone `hasPart` Room	X			X	X			

Table 5.2: This table shows at a high level which entities and relationships are required by each of the eight representative applications.

5.4.1 Designing Relationships

We use these representative applications to establish the set of required relationships as well as the domains of those relationships. Relationships define how entities are associated, which for a given entity may include:

- **Taxonomy:** what class or classes of things define an entity
- **Location:** which building, floor and room an entity is in, but also where in the room it is
- **Equipment Connections:** what equipment an entity is connected to, and how it is connected
- **Equipment Composition:** what equipment an entity is a part of, or what equipment is a part of it
- **Subsystem:** how the entity is situated in a building subsystem such as HVAC, Electrical or Lighting
- **Monitoring:** what measures the entity or what it measures

Portability and orthogonality are two primary concerns in designing the set of relationships to include in an effective ontology. When describing or reasoning about a building, the set of possible relationships between any two entities (i.e. set of named edges between any two nodes) should be small enough and well-defined such that the "correct" relationship should be obvious. This *orthogonality* reduces the possibility of inconsistency across buildings. Taken to its extreme, orthogonality informs a set of relationships that are specific and non-redundant, which can lead to overfitting the set of relationships for a particular building or subsystem. To support the goal of designing a unified metadata across many buildings, these relationships must also be sufficiently generic to be *portable* to many buildings.

Resolving these two tensions leads to the set of relationships listed in Table 5.1. We demonstrate here that this set of relationships is sufficient to cover the requirements of the representative applications. The specific entities and relationships each application requires are listed in Table 5.2. We implemented the eight applications in Table 5.2 and ran them on the five buildings; the results are collated in Table 5.3. The actual points exposed for each building by the BMS are the primary limiting factor for whether or not each application runs on a building: if a BMS exposes no lighting points, then a lighting application cannot run. In addition, applications have to account for the diversity of points across buildings: Brick defines synonym tagsets where possible, but there will always be a degree of disambiguation that is application-specific.

Brick allows applications to write *portable queries* that identify relevant resources in a building-agnostic manner. An application can then adapt its behavior to the set of returned resources, likely using some API to interact with the required points. For this reason, we implement each of the applications as a set of SPARQL queries that return the set of relevant entities and relationships.

	Building				
Application	Soda	EBU3B	GTH	GHC	Rice
Occupancy [128]	232	244	139	366	11
Energy Apportionment [123]	-	-	302	-	4
Web Displays [25]	513	697	81	65	106
MPC [208]	482	482	69	428	110
Participatory Feedback [143]	-	-	253	-	-
FDD [198]	136	229	12	229	-
NILM [153]	-	6	82	-	-
Demand Response [223]	144	1428	24	2490	4

Table 5.3: Number of matching triples in each building for the SPARQL queries consisting the eight applications. A non-zero number indicates that the application successfully ran on the building. Buildings with '−' did not have any relevant points exposed in the BMS.

```
1 example:myVAV rdf:type brick:VAV
2 example:myTempSensor rdf:type brick:Zone_Temperature_Sensor
3 example:myVAV brick:hasPoint example:myTempSensor
```

Figure 5.8: ZonePAC query for airflow sensors and rooms for VAVs. The query returns all relevant triples for ZonePAC to bootstrap itself to a new building.

5.4.2 Results

We implement eight applications — one from each of the application categories in [39] — as a set of SPARQL queries identifying the relationships in Table 5.2. These queries do not contain the full operating logic of the application, but rather serve as a bootstrapping step for the application to discover the set of available and relevant resources. Table 5.3 contains the results of running these queries over the five buildings for each of the applications.

The applications that ran on the majority of buildings did so because they rely on HVAC and construction/spatial information readily exposed by the BMS. This includes VAVs, AHUs, HVAC zones, relevant sensors, and how these connect to each other. The Participatory Feedback application operates entirely on lighting controls, which rarely appear in a BMS, thus limiting its portability. Likewise, the NILM application relies on power meters, which also may not be integrated into the BMS.

The primary challenge in developing portable queries was accounting for the variance in relationships across buildings. For example, a zone temperature sensor may have *either* of the two connections indicated in Figure 5.4: it may have an `isPointOf` relationship with an HVAC zone entity or a VAV entity. These inconsistencies arise from differences in building construction and the representation of the points in the BMS. It is possible to account for these differences in SPARQL to construct truly portable queries.

5.4.3 Example Application: ZonePAC

The ZonePAC [25] application 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. ZonePAC requires the following relationships:

- the mapping of VAVs to HVAC zones and rooms
- the heating and cooling state of all VAVs in the building
- the mapping of VAV airflow sensors to rooms
- all available power meters for heating or cooling equipment

Immediately, the requirements of this application outstrip the features provided by other metadata solutions. ZonePAC needs to relate entities across subsystems typically isolated or ignored in modern BMS: the spatial construction of the building, the functional construction of the HVAC system, and the positioning of power meters in that infrastructure. Brick simplifies this cross-domain integration and makes it possible to retrieve all relevant information in a few simple queries.

To identify the airflow sensors and rooms served for each VAV, the application uses the query in Figure 5.8. The application uses Brick's synonyms to capture both `Discharge Air Temperature Sensor`s as well as `Supply Air Temperature Sensor`s. Airflow sensors have an `isPointOf` relationship with the VAVs, and the rest of the relationships in the application mirror those in Figure 5.4. The "Web Displays" row of Table 5.3 contains the results of running ZonePAC over the five buildings.

5.5 Case Studies

We showcase the effectiveness of our schema by converting five buildings with a wide range of BMS, metadata formats, and building infrastructure into Brick. We discuss the challenges faced in converting various buildings into Brick to demonstrate Brick's robustness as well as to provide guidance for those facing similar challenges when using Brick.

Table 5.4 contains a summary of the construction and infrastructure of the five buildings, and how well Brick was able to capture their exposed BMS points. To evaluate the effect of "overfitting" Brick's tagsets to the set of known BMS points, we examined the % of BMS points covered by Brick's tagsets for Rice Hall and Soda Hall both before and after we incorporated their specialized points into Brick. Using an unaltered Brick, we matched 93.5% and 93.1% of Rice and Soda Hall's BMS points respectively. After incorporating the BMS-specific points, they scored 98.5% and 98.7% respectively, using Brick's class hierarchy to avoid compromising generalizability. Thus, we can conclude that Brick's tagsets do not overfit the set of five buildings. Examining Table 5.4, we can see that Brick matches the majority of points in all five buildings.

Building Name	Location	Year	Size (ft^2)	# of Points	% Tagsets Mapped	# Relationships Mapped
Gates Hillman Center (GHC)	Carnegie Mellon Univ., Pittsburgh, PA	2009	217,000	8,292	99%	35,693
Rice Hall	Univ. of Virginia, Charlottesville, VA	2011	100,000	1,300	98.5%	2,158
Engineering Building Unit 3B (EBU3B)	UC San Diego, San Diego, CA	2004	150,000	4,594	96%	8,383
Green Tech House (GTH)	Vejle, Denmark	2014	38,000	N/A	N/A	N/A
Soda Hall	UC Berkeley, Berkeley, CA	1994	110,565	1,586	98.7%	1,939

Table 5.4: Case Study Buildings Information. GTH does not expose any BMS points, so numbers are not available.

5.5.1 Gates Hillman Center at CMU

The Gates and Hillman Center (GHC) at Carnegie Mellon University is a relatively new building, completed in 2009, with 217,000 square feet of floor space, 9 floors, and 350+ rooms of various types (offices, conference rooms, labs), and contains over 8,000 BMS data points for various HVAC sensors, setpoints, alarms, and commands. CMU contracts with Automated Logic for building management.

The GHC includes 11 AHUs of different sizes serving multiple zones: three small AHUs serve one giant auditorium, one big laboratory and three individual rooms respectively. Eight large AHUs supply air to more than 300 VAVs. GHC's HVAC system also contains computer room air conditioning (CRAC) systems which are equipped with additional cooling capacity to maintain the low temperature in a computer room and fan coil units systems to provide cooling and ventilation functions. Despite the existence of these more esoteric subsystems, Brick matched 99% of GHC's BMS points, with the remaining points being too uncommon to be required by most applications (such as a `Return Air Grains Sensor` which measures the mass of water in air). The direct translation of BMS tags into Brick was relatively simple, only requiring a mapping between the human-readable BMS data points and Brick for each unique data point type.

The major challenge in converting the GHC to Brick was determining the relationships between pieces of equipment, which were not encoded in the BMS's labels. While the information is available through an Automated Logic GUI representation of the building, there was no machine readable method of understanding which VAV was related to which AHU. This required examining the building plans directly to incorporate these relationships (of which there were over 400). While a barrier to generating a Brick representation of a building, this example also shows the benefits that such a representation provides. Instead of being reliant upon manually examining a GUI to determine relationships between equipment, the Brick representation shows these relationships in both human and machine readable formats once represented in Brick.

5.5.2 Rice Hall at UVA

Rice Hall hosts the Computer Science Department at the University of Virginia. The building consists of more than 120 rooms including faculty offices, teaching and research labs, study areas and conference rooms distributed over 6 floors with more than 100,000 square feet of floor space. The building contracts with Trane for building management.

Rice Hall contains 4 AHUs associated with more than 30 Fan Coil Units (FCU) and 120 VAVs serving the entire building. Besides the conventional HVAC components, the building features several different new air cooling units, including low temperature chilled beams and ice tank-based chilling towers, an enthalpy wheel heat recovery system, and a thermal storage system. The building also contains a smart lighting system including motorized shades, abundant daylight sensors and motion sensors. Rice Hall's BMS points are easily interpretable for conversion to Brick despite it containing some uncommon equipment such as a heat recovery and thermal storage systems, as part of the building design as an energy-efficient "living laboratory". Moreover, the set of relationships defined by Brick sufficiently captured how the uncommon equipment related to other components of the HVAC system.

A few of these points, such as `Ice Tank Entering Water Temperature Sensor`, are specific to Rice Hall among the set of five buildings we examined. Nonetheless, Brick's structure allows for the clean integration of new tagsets into the hierarchy without disrupting the representation of existing buildings.

5.5.3 Engineering Building Unit 3B at UCSD

The Engineering Building Unit 3B (EBU3B) at University of California, San Diego hosts the department of Computer Science and contains offices, conference rooms, research laboratories, an auditorium and a computer room. The building was constructed in 2004 and has 150,000 square feet of floor space with over 450 rooms. The BMS of EBU3B is provided by Johnson Control Inc., and contains more than 4500 data points, most of which belong to the HVAC system and power metering infrastructure.

The HVAC system consists of a single AHU that supplies conditioned air to 200+ VAV units and some FCUs. There is a CRAC system serving the computer room and there are exhaust fans for all kitchens and restrooms. The HVAC system also consists of Variable Frequency Drives (VFD), valves, heat exchangers and cooling coils to facilitate operation of AHU and CRAC. Brick's schema provides the necessary tagsets and relationships to account for all of these components and their data points. The university central power plant provides the hot and cold water necessary for the HVAC and domestic hot water system. The corresponding sensors that measure the hot and cold water use were modeled in Brick, but the central plant was left out as it was not part of the building. The building contains meters that measure power consumption of various subsystems: lighting, computer room, HVAC system and elevator. The meters were associated to the corresponding systems with `isPointOf` relationship as required by the applications.

An issue that arose in mapping EBU3B to Brick was that the AHU supply air was divided into two parts that supplied air to two wings of the building. Brick currently does not provide a means to model this division of supply air which has proven relevant to the diagnosis of various faults. Moving forward, Brick can address this by modeling the AHU discharge air as a *resource*, which can also help model other concepts such as cold water supply from a central plant. Alternatively, the discharge air can be attributed to the cooling coil that modulates its temperature, and the cooling coil can be said to feed discharge air to the terminal units.

Additionally, EBU3B's BMS contains data points corresponding to Demand Response events, which exposes an interesting conflation of the representation and operation of the building. Because BMS are typically written as monolithic applications over building-specific representations, they must incorporate external signals such as Demand Response into the set of BMS points. Conversely, Brick decouples the resources and infrastructure of a building from the processes managing the building.

5.5.4 Soda Hall at UC Berkeley

Soda Hall, constructed in 1994, houses the Computer Science Department at UC Berkeley. It comprises mostly of closed small to medium sized office spaces, where either faculty or groups of graduate students sit. The BMS system, provided by the now-defunct Barrington Systems, exposes only the sensors in the HVAC system.

The HVAC system of the building runs on pneumatic controls, and comprises 232 thermal zones. The zones on the periphery of the building have VAVs with reheat, while the other zones do not. For a VAV with reheat, the same control setpoint indicates both the amount of reheat and the amount of air flowing into a zone, by using a proprietary value mapping mechanism. While the value mapping is building-specific, Brick can express the fact that the same sensor controls both the reheat and air flow by labeling the point as a subclass of both reheat and airflow tagsets. The logic for communicating with the point correctly would be handled by some other system; Brick simply identifies the available points.

Unique to the set of buildings presented here, the operational set of Soda Hall's HVAC components is not static. Soda Hall contains a redundant configuration of chillers, condensers and cooling towers. At any point of time, one of each of these systems is operational, while the others are kept as hot standby. An isolation valve setpoint indicates which of the redundant subsystems is currently operating. Brick completely expressed the redundant subsystem arrangement, but the equipment contained several unique points such as `On Timer` for the chiller subsystem that had to be added to Brick's tagsets.

5.5.5 Green Tech House

The Green Tech House (GTH), constructed in 2014, is a 38,000 square feet office building that houses a range of organizations and companies with different business purposes. It is a three-story building containing 50 rooms comprising office spaces, a cafeteria, meeting rooms and bathrooms. GTH's BMS limits our access to a subset of the available points, so calculations of how well Brick covered the points (Table 5.4) are not applicable.

GTH's BMS exposes some lighting points, but has a substantially different HVAC system. The HVAC system heats air centrally in order to distribute air to zones with cooling capabilities. Although the building documentation don't refer to groups of equipment as AHUs or VAVs, equivalents are present.

A single AHU recovers heat from the return air using a rotary heat exchanger. This heat exchanger is the first in a cascade of two between outside air and supply air. Outside and return

air are never mixed. The pressure of return and supply air of the north and south side of the building are measured separately.

AHU heating relies on the second heat exchanger which uses a hot water loop. Additionally, most rooms have either radiators or floor heating. These are supplied by independent hot water loops, heated by district heating.

The main challenge in converting GTH's points to Brick was accounting for the acute difference in infrastructure compared to Brick's construction. The current version of Brick assumes that an HVAC system contains VAVs and AHUs, which is not strictly true for GTH. For the purposes of running Brick applications, we mapped GTH's HVAC components onto Brick's model; however, this is not an ideal solution. It is not Brick's goal to disguise the construction of building subsystems, but rather to abstract away the intricacies of subsystem composition between buildings. Future versions of Brick will account for such variation in subsystem equipment and construction.

5.6 Discussion

This chapter has addressed an important open problem that was referenced in [26, 39, 40, 95, 118] — Can there be a building metadata schema that is complete, expressive and usable? Completeness entails that the schema expresses the vast majority of the points found in large commercial buildings that facilities managers thought worthy to include in the BMS tags, as well as the points and relationships necessary for important building applications. Expressiveness entails that the schema's namespace is well-defined, and one can implement applications using it rather than using the ad-hoc namespace of the particular building. Usability entails that the schema is understandable, and the total amount of work required to convert the existing buildings into the schema is limited and bounded, and can perhaps be automated at a reasonable scale using solutions such as [26] and [40].

We have defined a schema, Brick, that we believe is a strong candidate to solving this open problem. Brick builds upon prior work and introduces a number of novel concepts that we believe adresses this open problem. Brick uses clear tags and tagsets to specify sensors and subsystems in a building. It defines an ontology and a class hierarchy for this list of tags and tagsets. Relationships are represented as triples, which allows us to leverage existing tools to build and query the resulting building representations. Brick proposes Functional Blocks to abstract out complexity but also aid in system composition and hierarchies. Finally, Brick uses the notion of synonyms to equate sensors and subsystems similar in function.

Brick is complete, capturing an average of 98% of BMS data points across five diverse buildings comprising almost 15,700 data points and 615,000 sq-ft of floor space. Brick is expressive, successfully running eight canonical applications on these buildings. Four applications ran on all five buildings, while the remaining applications ran on buildings whose BMS exposed the requisite points. Brick is usable, as converting each of the buildings' legacy metadata to the normalized schema took no more than 20 man-hours. The resulting schema is understandable and easy to query as shown in Figures 5.6, 5.7 and 5.8.

Brick tries to maintain orthogonality in describing tagsets and relationships, i.e. there should be a single straightforward way to describe an entity, collection of entities and their inter-relationships. The functional block model in Brick proves very helpful in abstracting out the complexity and heterogeneity of particular subsytems in buildings, while aiding system composition and hierarchies. For instance, functional blocks' easy replication, instantiation and composition can help Brick quickly specify the numerous VAV zones and its associated points in a building. Functional blocks also help hide the subtle differences between complex subsystems such as an AHU.

Brick distinguishes itself from other industry standards in the building-metadata domain by virtue of its process of the production of open reference implementations on real buildings serving as a means of evaluating the effectiveness of the solution. Developing a reference standard through such a process has been successful in other fields, most notably the IETF [8] for internet protocols and algorithms. The code, schema, and reference implementations of all the buildings in our testbed are available at [1].

We hope that our solution to this well-defined open metadata problem lays the foundation for industry and academic collaboration to produce bonafide standards that could be transformative in producing energy efficient buildings and portable applications.

Chapter 6

Enabling Non-Trivial Scalable Building Applications

Transformation of the primitive metadata into a common metadata schema yields *semantic relationships* between the sensors, and enable the development of applications which are portable across the building stock.

The results of such analytics applications may be used to better inform the manual commissioning, or equipment purchase process, and give the facilities manager insights into problems plaguing a building. In this chapter, we describe a few applications that were run unmodified on multiple commercial buildings.

6.1 Simple Diagnostics Applications

Our applications first searched for required sensors in a building, and if matching sensors were found, it proceeded to perform operations on the corresponding data streams.

Rogue Zones: A thermal zone (or room) is *rogue* if its air temperature is constantly above their required setpoints, i.e it requires constant cooling[1]. The air handling unit, in an effort to cool these rogue zones down, tries to circulate as much cold air as it possibly can, leading to energy wastage. Thus, rogue zones are an artifact of poor planning or wrong setpoints, and can be rectified if brought to the attention of a building manager. To find the rogue zones in a building, one needs to search for *zone air temp sensors*, and for each such sensor, find the corresponding*zone air temp setpoint*, and check whether the temperature is always more than its respective setpoint (factoring in a tolerance factor of 2F).

Inefficient Air Handling Units: A hot rogue zone may drive the air handler to supply air that is too cold, resulting in other zones supplied by the same air handler always being too cold and uncomfortable. It may be possible that fixing some rogue zones might result in other zones being more comfortable. This application requires searching for the the air handling unit ID of each rogue zone, and check if any of the other zones served by the same air handling unit was

[1] Such hot zones are often caused by server/electrical loads

Hot Rogue Zones

Zone	Avg (zone temp – zone setpoint)	Air Handler Id	Zone Temperature Setpoint (F)	Avg Zone Temp (F)
330B	10.6	1	72	82.6
333	5.5	4	72	77.5
288	5.0	1	70	75.0
627	53.8	3	70	73.8
342	2.7	1	71	73.7

Over-cooled Zones

Zone	Avg (zone temp – zone setpoint)	Air Handler Id	Zone Temperature Setpoint (F)	Avg Zone Temp (F)
340	-2.7	1	75	72.2
544	-3.4	1	75	771.5
180	-3.5	1	75	71.4
300T	-3.5	4	75	71.4
420A	-3.6	1	75	71.3
678	-4.0	1	72	67.9
444	-4.1	1	72	67.8
384	-4.3	1	74	69.6
530	-4.5	1	75	70.4
682	-4.6	1	72	67.3
626	-4.9	1	75	70.0
405A	-5.3	1	72	66.6
405B	-5.9	1	72	66.0
300B	-6.0	1	75	68.9
420	-7.0	1	75	67.9
684	-7.3	1	75	67.6
287	-8.7	1	70	61.2

Figure 6.1: List of hot and cold rogue zones generated through a scalable building efficiency application on Building 1 in our testbed.

over-cooled (i.e constantly had a lower temperature than its corresponding setpoint factoring in the a tolerance factor of 2F). This analysis fails to run on buildings where the metadata does not provide enough information to associate thermal zones to its corresponding air handling unit.

Building Id	Year of Construction	BMS Vendor	Num. of Sense Points	Num. of Thermal Zones	Num. of Hot Rogue Zones	Num. of over-cooled zones	Num. of AHUs	Num. of Inefficient AHUs
1	1994	1	1586	201	5	17	4	2
2	2009	2	2522	78	2	0	NA	NA
3	1961	1	367	42	28	1	2	1
4	1968	1	132	12	1	0	2	0
5	1941	1	417	48	8	4	6	2
6	2007	1	6169	368	35	5	NA	NA
7	NA	1	164	8	3	0	6	0
8	1950	1	421	20	0	2	1	0
9	1982	1	277	9	2	0	3	0
10	1996	1	730	57	10	0	1	0
Total			12813	843	94	29	25	5

Figure 6.2: Evaluation of our portable applications on all 10 buildings in our testbed. (AHU : Air handling units)

Nighttime setback: To save energy, buildings may opt for more conservative temperature and airflow setpoints during non-office hours and weekends. Absence of nighttime setbacks helps identify a simple way to reduce building energy consumption. This analysis searches for setpoints , e.g *zone temp setpoints* and verifies whether or not they had reported different data values for office and non-office hours.

6.1.1 Applying Applications on One Building

We present a detailed of one of the buildings in our testbed — Building 1. Figure 6.1, shows the result of our applications on Building 1. This building had more than 1586 sensors, 201 thermal zones and 4 air handling units. Three of the hot zones were served by Air Handler 1, which resulted in a lot of over-cooled zones under the same air handler. We verified with the facilties manager that zone 330B was a communications closet which produced a lot of heat. Air Handling Unit 1 in a best effort to cool this zone down, was over-cooling a whole list of other office spaces, resulting in the occupant discomfort and energy wastage. Also, as can be seen from the table, none of the zones implemented NightTime Setback, which results in further energy wastage.

6.1.2 Porting Application to Other Buildings

Table 6.2 shows the summary of the application of our portable applications on each of the 10 buildings in our dataset. The results show that presence of rogue zones irrespective of the age or BMS of the buildings. Only one building (Building 2) implemented night-time setbacks. We were able to identify two other buildings with the same inefficiency in air handling units as was pointed out in Building 1. Our *Identifying Inefficient Air Handling Units* could not be run on two buildings because the metadata did not contain enough information to capture the relationship between zones and air handling units.

6.2 Occupancy Detection

Modern commercial buildings are instrumented with thousands of sensors sampling several environmental parameters, including temperature, humidity, pressure, supply air velocity, and damper and valve positions in various locations at regular intervals. These measurements are communicated to a Building Management System (BMS), which monitors building operations and controls indoor climate using nested control loops. Traditionally, most climate control systems use temperature and humidity as primary inputs in determining heating, cooling, and ventilation requirements, assuming constant maximum occupancy [21]. A climate control system that conditions rooms based on their actual usage is indeed more efficient; however, accurate occupancy information is not normally available in a building, and existing approaches aiming to incorporate occupancy into the control loops require retrofitting the building with numerous sensors, which is intrusive, overly costly, and error prone given the scale.

Despite the lack of direct occupancy sensing in most commercial buildings, the large volume of data available through the centralized BMS provides ample opportunities to learn the recurring occupancy pattern of each zone through analyzing the measurements of the variable air volume system (*e.g.*, the damper or reheat valve position sensor) that maintains the zone temperature around its setpoint. The inferred occupancy patterns can be incorporated into the scheduling of each zone, thereby optimizing the HVAC energy consumption. Additionally, abnormalities in the occupancy pattern of a zone over several weeks could potentially indicate a wide range of equipment faults and operational inefficiencies, such as duct leakage, sensor drift, stuck dampers, leaky valves, and improper setpoints, which can be located and addressed by the facilities manager.

This work investigates the possibility of using scalable unobtrusive time series analysis techniques to develop customized per-zone schedules (which dictate night-time setbacks) that incorporate a *rough* estimate of occupancy obtained from coarse-grained measurements of *occupancy indicative sensors* available through the BMS. Specifically, we employ a step change detection algorithm for identifying step edges of the occupancy indicative signal, which are then associated with occupancy start and end times, and consider the application of an empirical decomposition technique for removing the effect of noise and other dominant factors. We evaluate the efficacy of these techniques in three large commercial buildings in the United States and show through simulations that huge energy savings can be obtained using simple schedules that can be easily programmed into legacy HVAC systems. Should facilities managers want to further increase the energy savings, we propose adaptive schedules that can track occupancy of each zone for a given tolerance for occupant discomfort. This work was done in collaboration with Omid Ardakanian [19].

6.2.1 Background and Feasibility

A large commercial building's internal environment is comprised of multiple *thermal zones*[2]. Each thermal zone typically has its own specific operating conditions and performance requirements, and these may vary across zones. For instance, a thermal zone housing a cluster of servers might require low humidity, low temperature and high airflow conditions, whereas a thermal zone comprising an occupant office might require low humidity, moderate temperature and moderate airflow. The environment of each zone is controlled by a different control loop.

The major sources of heat gain and loss in a zone are — (a) heat gain from the presence of occupants[3], (b) heat gain from (possibly) periodic external sources such as solar irradiation, (c) heat gain from internal sources such as servers, displays, incandescent lights, etc, and (d) conduction heat transfer through the external building envelope, or through walls, floor, ceiling, windows, and open doors. The Heating Ventilation and Air Conditioning (HVAC) system in a building extracts or supplies the balance heating or cooling required to maintain the internal environment of the thermal zone at its specified operating condition.

In this section, we give a short overview of the HVAC system in a building and its associated sensors, how the internal environment of each thermal zone is maintained by various control loops in conjunction with those sensors, provide an intuition as to why it is possible to extract occupancy simply from the HVAC sensors, and make a case for using these occupancy estimates to design smarter schedules to optimize building-wide energy consumption.

HVAC System

We study large commercial buildings whose HVAC system comprises central cooling and distributed reheat, which is common in moderate climates yet not universal. Such an HVAC system typically consists of one or more Air Handling Units (AHUs), which supply cool air through ductwork to a large number of Variable Air Volume (VAV) systems, each of which controls the local environment of a thermal zone. If the zone requires cooling to balance the heat gained from occupancy and external sources, the VAV units open its dampers to the required extent to allow cooler air to flow into the zone. Conversely, if a zone requires heating to maintain its operating point, the VAV units reheat the cold air by opening the reheat valve and passing the air through reheat coils, which maybe simple electric heaters or pipes with hot water running through them, before supplying it to a zone.

The heating or cooling action of a VAV unit is determined by a control loop, whose job it is to maintain the temperature of a zone as close to its setpoint as possible, while maintaining the required minimum airflow for its size and occupancy, as specified in ASHRAE standards [21]. To do this, the VAV control loop monitors the zone temperature, and actuates the air inflow/outflow dampers and reheat valves of the VAVs. The Building Management System (BMS) typically

[2]A thermal zone may comprise a single room, multiple rooms, or may even span floors.

[3]The total heat generated by an average person per hour is 105Wh, almost equivalent of the heat produced by a 100W light bulb [61].

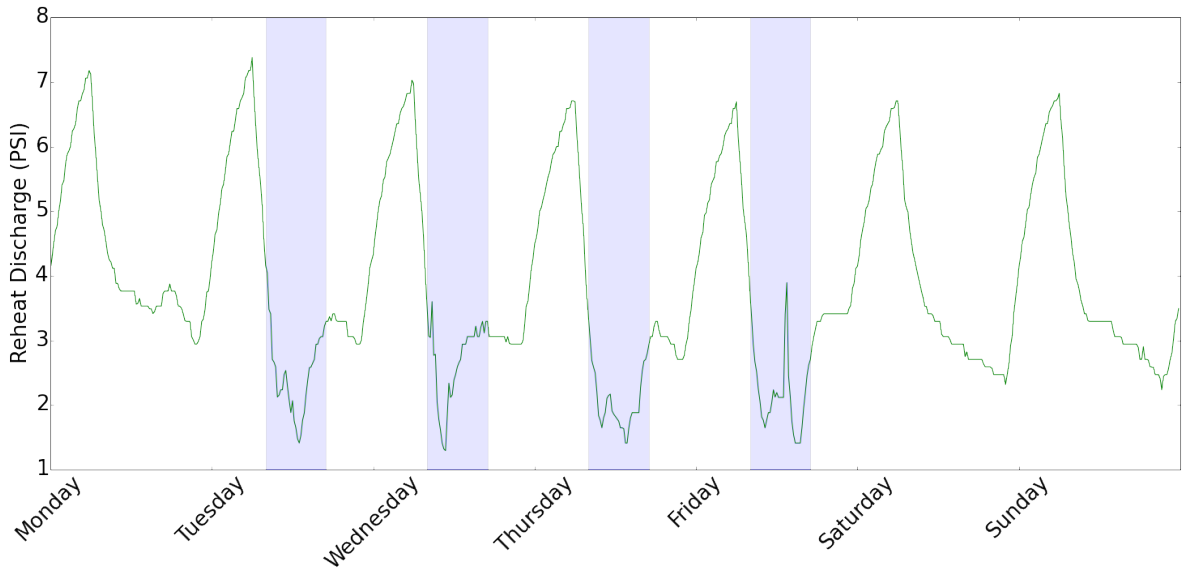


Figure 6.3: Occupancy indicative sensor readings and the ground truth occupancy (shaded intervals) of an office over 7 days.

logs the instantaneous values of the sensors and actuators associated with each of these subsystems.

Impact of Occupancy on VAV Control

An entry of a human occupant raises the amount of heat load in a zone. This is due to the heat dissipated from the human body as well as the heat generated by appliances that are used by the occupant, collectively known as occupancy-induced loads. The VAV control loop for that zone has to react to this change to maintain operating conditions either by reducing the amount of reheat for the cold air supplying a zone, or by increasing the airflow into a zone. It is this response of the HVAC system that our study aims to measure to determine a rough estimate of when a zone was occupied, and design schedules, which work by implementing night-time setbacks which widen the guard-band the zone temperature is allowed to float in, based on that zone's occupancy characteristics.

In every thermal zone, at least one sensor (*e.g.*, reheat valve or damper position sensor) reflects this response to the heat load change when an occupant enters or exits the zone. We refer to this sensor as *occupancy indicative sensor* and its readings constitute *occupancy indicative signal*. Figure 6.3 illustrates the correlation between the reheat valve position and logged occupancy hours of an office during a week. The office has been occupied during regular office hours, *i.e.*, 8:00am–5:00pm, on all weekdays except for Monday which was a holiday. It can be readily seen that the reheat valve position varies drastically as the occupancy state of the room changes. In particular, the reheat valve closes to a certain extent when the room gets occupied and opens once it becomes empty again, highlighting the potential for detecting occupancy from

the occupancy indicative signal. Note that the reheat valve also closes to a lesser extent on nonworkdays; this can be attributed to conduction heat transfer and solar radiation.

Saving Energy Wasted due to Reheat

The AHU and VAV settings often lead to over-cooling of the zones they serve because of two primary reasons. First, since each AHU supplies a large number of zones, the supplied air has to be cooled sufficiently to offset the largest heat gain in the building in a reasonable amount of time. Second, ASHRAE standards regulate the minimum amount of airflow that has to be let into a zone at any point of time. To precisely control the temperature within a zone, VAVs typically have reheat mechanisms which reheat the air so that the zone is not over-cooled.

This over-cooling of the air at central AHUs and subsequent reheating of the air at terminal VAVs at all times, irrespective of occupancy, results in a huge wastage of energy, which could be mitigated by shutting off the dampers or simply not reheating the air when there is no occupancy. However, occupancy sensors are seldom deployed in all zones in legacy buildings, resulting in VAVs and AHUs running on a static and fixed schedule, if any, which leads to huge energy wastage. Hence, applying a schedule based on a rough estimate of occupancy could result in substantial energy savings. Energy could also be saved by reducing flow into unoccupied zones, permitting pressure and temperature resets at the AHU, which we do not evaluate in this work.

Challenges

Several confounding effects, such as solar radiation, equipment load, outside air temperature, noise, and other transient effects, contribute to the change of heat load in a zone. Hence, the effect of occupancy is not always pronounced in the occupancy indicative signal, making it extremely difficult to estimate the exact occupancy state of a zone from this signal at a given point in time. However, computing energy-efficient schedules only requires a rough estimate of per-zone occupancy start and end times, which can be obtained using statistical techniques as long as the zone occupancy pattern exhibits some periodicity and the heat emitted by occupants is significant enough compared to the energy delivered by the HVAC system and other sources. Moreover, the longer we observe these confounding effects in a zone, the easier it is for the statistical techniques to weed them out. The occupancy indicative signal can also be passed through a number of filters to remove the noise and other unwanted effects. We expand on these ideas in Section 6.2.2.

6.2.2 Methodology

This section describes our methodology for assessing the potential of unobtrusive analytics to obtain a rough estimate of occupancy at the level of individual zones despite the many sources of noise in the underlying signals.

Testbed

To demonstrate and evaluate our techniques, we use a testbed comprising three large campus buildings with Building Management Systems installed by different vendors, henceforth referred to as Building 1, 2 and 3. These buildings contain 117, 109 and 270 zones, covering an area of 110,565, 141,000, 305,641 sq.feet, respectively, and comprise mostly faculty and student offices. The zones in these buildings contain respectively 2, 4 and 3 sensors (and multiple other setpoints/command points). Buildings 2 and 3 have local VAV control with local reheat with different sensors and setpoints devoted to controlling air flow and reheat, while Building 1 has a pneumatically controlled VAV where one setpoint is used to control both local reheat and air flow. The data from the zone-based sensors is collected at an approximate sampling rate of one reading every 10 minutes. We run our analysis on three months of data collected between the months of March and June. The occupancy indicative sensors used for the three buildings are — (a) the single pneumatic control sensor in Building 1, (b) air flow sensor in Building 2, (c) the reheat sensor in Building 3.

Existing Energy Saving Strategies

Building 2 implemented a setback strategy which allows the temperature to drift when the zones are presumably empty. The nighttime setbacks were in effect every day from 7pm–5am during the time that we were collecting sensor data. The other two buildings neither had a setback strategy nor ran any other building-wide or zone-specific energy saving schedule, mostly due to the critical function of only a small fraction of their zones. More specifically, a subset of zones in Building 3 contain heat-generating equipment that operate uninterruptedly and have strict temperature requirements. These zones are required to be conditioned at all times. Lacking customized per-zone schedules, these buildings are extremely inefficient in terms of energy consumption.

Ground Truth Data

In general, we expect that in a large building it will not be possible to groundtruth occupancy of the many zones, but in developing the solution stratified sampling of occupancy is essential. To this end, we manually log the occupancy hours of 7 private and shared offices over the period of two weeks where these rooms are selected from our three buildings such that we have at least one interior and one perimeter zone in each building. The occupants of each room are asked to record on a daily basis the times that they come in and leave, and any time range in between that the room had no occupancy for longer than an hour. In addition to the manual occupancy logging, we have access to a security camera installed in a large lab with a lot of heat-generating equipment. The occupancy hours of that lab are also extracted from the video recordings.

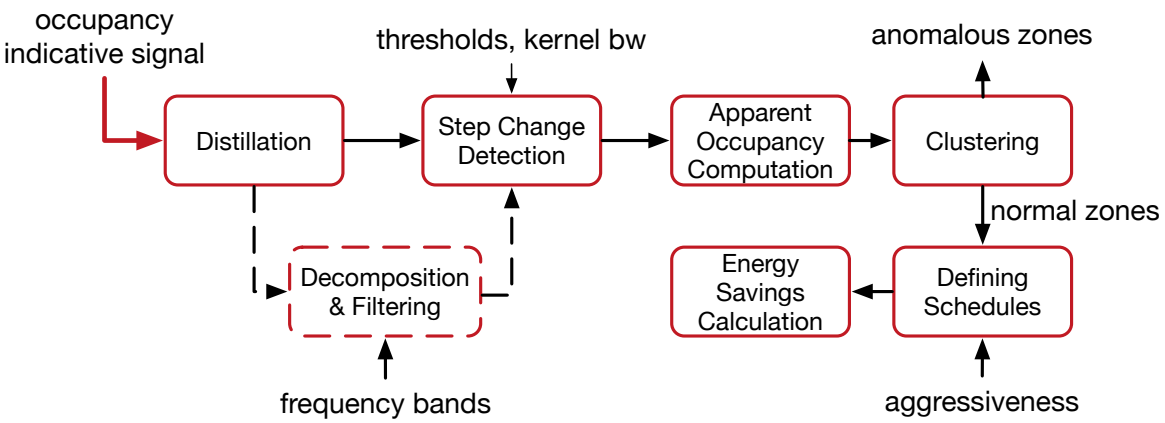


Figure 6.4: Proposed unobtrusive occupancy detection approach.

Approach

Our approach is comprised of a few time series analysis techniques that are applied to occupancy indicative signals from our testbed to roughly estimate the occupancy intervals of individual zones as shown in Figure 6.4. These techniques include (a) a data cleansing algorithm for correcting and amending incorrect and incomplete data, (b) an empirical decomposition technique for decomposing a time series into its intrinsic modes and subsequently removing high and low frequency modes that pertain to noise, and diurnal and seasonal effects, (c) a step change detection algorithm for identifying the step changes of a time series, and (d) statistical techniques for calculating apparent occupancy of the zones from their estimated occupancy intervals and comparing the apparent occupancy of different zones. We cluster the zones based on their apparent occupancy and examine the resulting cluster of anomalous zones for potential faults. Various schedules can be computed for the zones that belong to the normal cluster, which are specified in Section 6.2.3. We expand on each of these steps in the following.

Distillation

Distillation is the process of removing outliers from the occupancy indicative signal and correcting erroneous readings. Outliers are detected through comparison with their neighbouring points and substituted with the median of a window of certain length that surrounds them. Apart from data cleansing, in Building 1 where a single sensor monitors both damper and reheat valve, the sensor readings are split into two separate signals, one represents the damper position and the other one represents the reheat valve position. One of these two signals which is more indicative of occupancy is treated as our occupancy indicative signal.

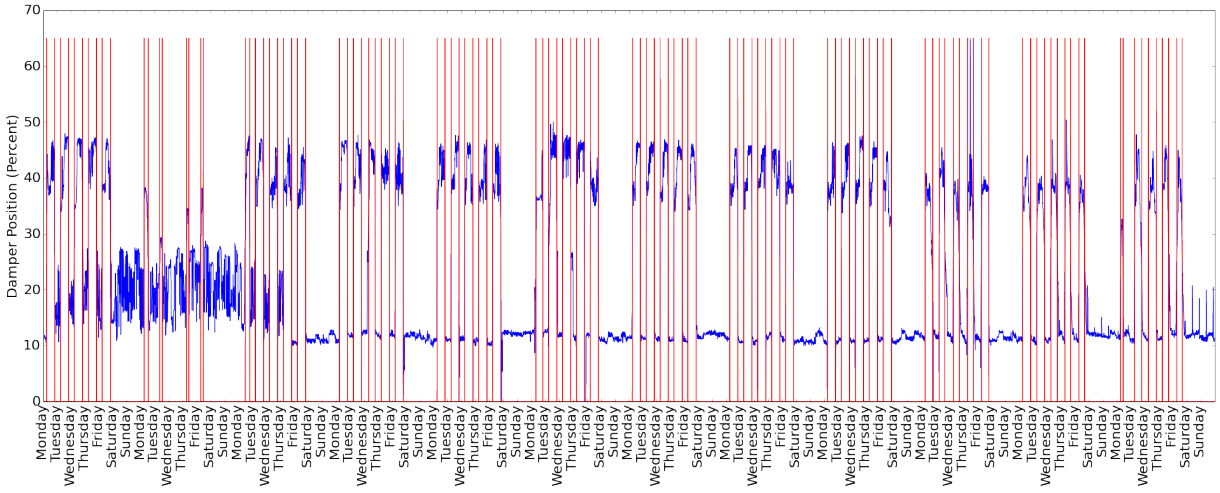


Figure 6.5: The damper position sensor readings of a cafeteria in Building 2 (blue) and upward and downward edges that are detected by Canny edge detector (vertical red lines), representing the occupancy start and end times, respectively.

Frequency Decomposition & Filtering

The harmonic components of the occupancy indicative signal that best reveal the zone occupancy pattern can be identified and extracted in the frequency domain. We adopt Empirical Mode Decomposition (EMD) [120], which is an adaptive and a posteriori method for decomposing non-stationary data into intrinsic patterns, trends, and noise with the basis of the decomposition being derived from the data. This iterative algorithm can be applied to decompose a signal into a small number of oscillatory, yet not necessarily uniform components termed *intrinsic mode functions* (IMFs). An IMF contains the same number of extrema and zero-crossings (or they differ at most by one), and at any point the envelopes defined by its local maxima and minima are symmetric with respect to zero. This means that IMFs admit Hilbert transform and the notion of *instantaneous frequency* can be defined for them at any point in time.

Following the technique proposed in [92], we apply a variant of EMD, called Complete Ensemble EMD, to decompose our occupancy indicative signal into a small number of IMF components. Once the instantaneous frequency is computed for each IMF, the IMFs are grouped into low, medium, and high frequency bins based on their average instantaneous frequency. Specifically, the IMFs with an average period greater than 18 hours, between 18 and 2 hours, and less than 2 hours constitute the low, medium, and high frequency bins, respectively. These bins are defined such that the low frequency IMF components capture diurnal and seasonal effects, while the high frequency IMF components mostly capture noise and oscillations of the control system. We then add the medium frequency IMF components to obtain a filtered occupancy indicative signal that can be used instead of the original occupancy indicative signal in the next step. However, our experiments suggest that using filtered occupancy indicative signal does not improve the accuracy of our approach in most zones, implying that a change in

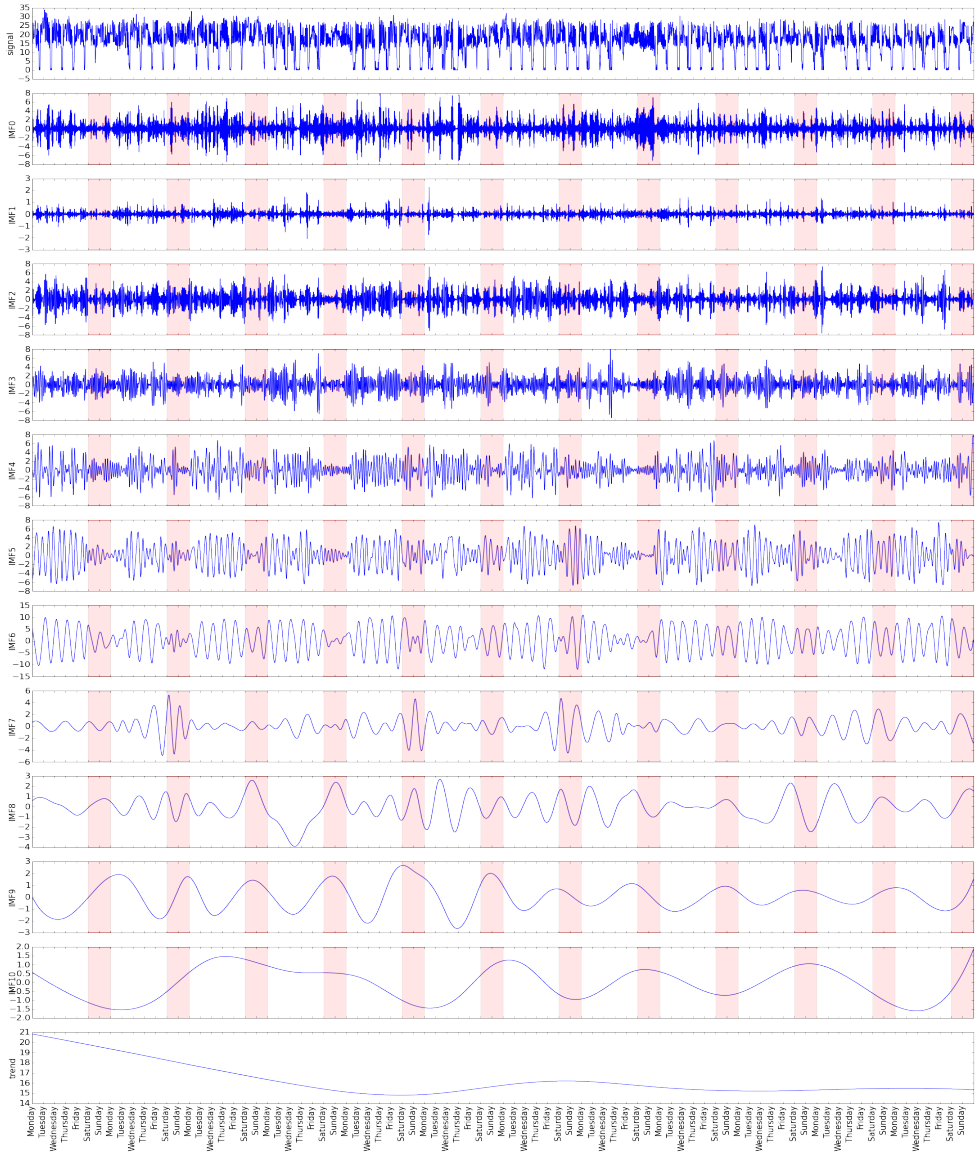


Figure 6.6: The occupancy indicative signal representing the reheat valve position of a zone over 12 weeks of measurements (top), the intrinsic mode functions, and the trend (bottom) extracted by Complete Ensemble EMD algorithm. The weekends are represented in red.

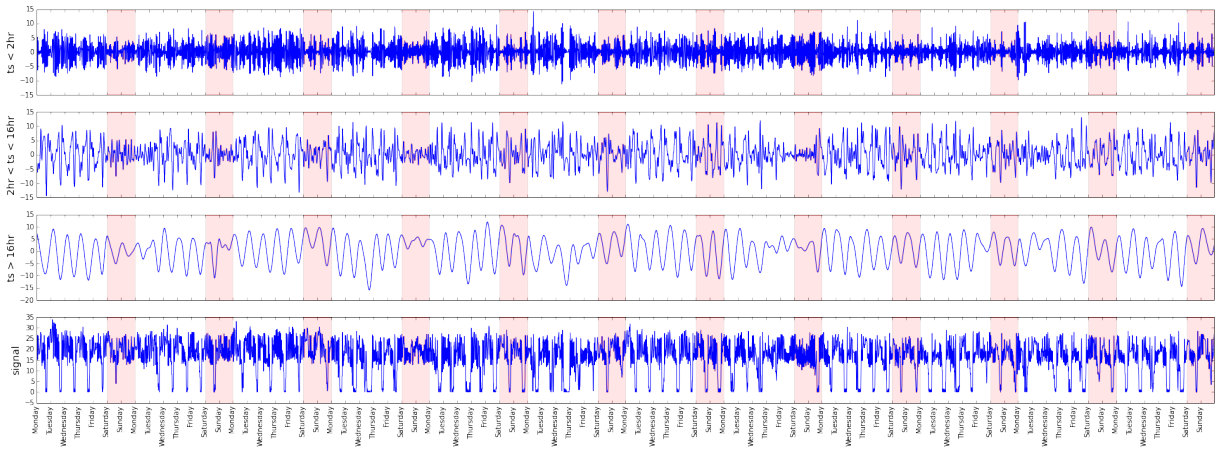


Figure 6.7: The aggregated IMF components for low, medium, and high frequency bins and the occupancy indicative signal representing the reheat valve position (bottom). The weekends are represented in red.

the occupancy state of a zone produces a detectable signature in the occupancy indicative signal that can be identified, despite the noise and other confounding effects. For this reason, we do not run the edge detector on the filtered signal, but this technique can be used to possibly remove the compounding effects, such as solar heat gain, in buildings where such effects are dominant.

Step Change Detection

A significant *step-edge*[4] in the occupancy indicative signal can be attributed to a change in the occupancy state of the corresponding zone. Thus, detecting these step-edges is key in inferring occupancy. We distinguish two types of step-edges — an *upward step-edge* is a step-edge where the mean of the signal increases after the edge, whereas a *downward step-edge* is the one where the mean of the signal decreases after the edge.

Several step change detection algorithms have been proposed in the literature. In this work, we use Canny edge detection [46] to identify step-edges of occupancy indicative signals. This algorithm has been extensively used in signal processing and computer vision, where it is generally applied to 2D signals, *e.g.*, images. The Canny detection algorithm consists of the following steps: (a) input signal is smoothed by a Gaussian filter, (b) gradient of the smoothed signal is computed, (c) local optima of the gradient of the smoothed signal are found and considered as candidate edges (d) a simple thresholding technique is applied to preserve strong edges among the candidate edges, and (e) weak edges that are linked to the strong edges are added to the set of identified edges. It has been shown that the first derivative of Gaussian kernel is an optimal edge detector in the sense that it approximates the operator that optimizes the product of signal-to-noise ratio and localization [46]. To see this, suppose S is our occupancy indicative signal, G is a Gaussian filter, D denotes the differentiation operator, and $*$ denotes the convolution operator.

[4] A step-edge is defined as an abrupt change in the mean of the signal.

Since convolution is associative, we can write

$$D * (G * S) = (D * G) * S$$

where $D * G$ would be the derivative of Gaussian filter.

We employ a variant of this algorithm which is capable of distinguishing between upward and downward step-edges of time series data. Specifically, the occupancy indicative signal is convolved with the first derivative of Gaussian kernel to produce a local maximum at each upward step-edge and a local minimum at each downward step-edge. The obtained local optima are treated as candidate edges of our occupancy indicative signal, determining the beginning and the end of occupancy periods. To err on the side of caution, we neglect the weaker upward (downward) step-edge if two consecutive local maxima (minima) are detected. Figure 6.5 shows the step-edges of the occupancy indicative signal of a zone in Building 2 detected by Canny edge detector. When the occupancy periods are found, a binary vector is constructed for the inferred occupancy of each zone, where zeros and ones denote unoccupied and occupied states, respectively.

We note that our step-edge detector might pick up a significant edge that is not due to occupancy or miss a true edge that falls below the specified threshold (*i.e.*, a weak edge). These errors will lead to spurious detections; nevertheless, these rough occupancy estimates are reasonably accurate for creating customized per-zone schedules as discussed in Section 6.2.2.

Computing Apparent Occupancy

Most zones in a commercial building are expected to have periodic occupancy patterns as people tend to maintain consistent weekly schedules. This periodicity can be leveraged to generate per-zone schedules that repeat every week, can be easily programmed into the building BMS, and will reduce wastage of energy due to reheating the zones that are believed to be empty. Computing such schedules requires having a distribution of the zone occupancy for each day of the week. This can be generated by taking the weekly average of the inferred binary occupancy vector of each zone over a certain number of training weeks. We refer to this as *apparent occupancy* of a zone. Figure 6.8 shows the apparent occupancy of a zone computed using 12 weeks of occupancy indicative sensor readings. The apparent occupancy encodes the probability that a zone is occupied at a given time on a particular day of the week.

Clustering

The apparent occupancy of a zone offers a lot of insight into how it is used on each day of the week. For example, a zone whose apparent occupancy peaks between 8am-5pm on weekdays and is zero at other times, is probably an office occupied during regular office hours. In addition to the type and function of a zone, the apparent occupancy can also reveal an equipment fault or a control error. For instance, constant apparent occupancy during the night or throughout the week might imply a stuck damper or actuator, a temperature sensor drifted out of calibration, or improper coefficients defined for the control loop. These potential faults, which contribute to

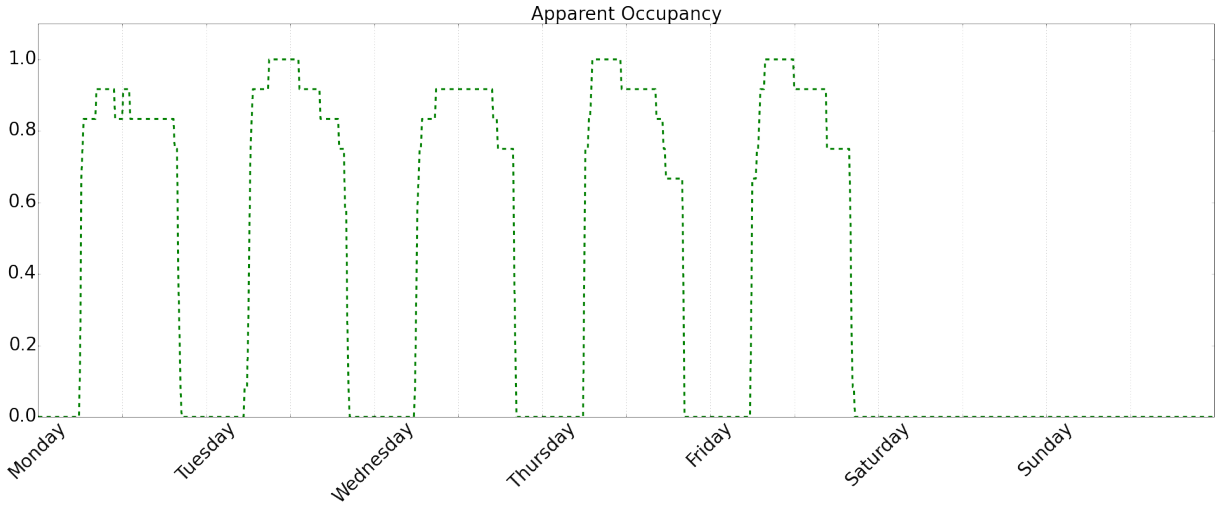


Figure 6.8: The apparent occupancy of a cafeteria in Building 2 strongly suggests that it has always been closed on weekends.

	Building 1	Building 2	Building 3
normal zones	99	76	135
anomalous zones	10	6	106
mostly unoccupied	8	27	29
total	117	109	270
pct. anomalous zones	5.6%	5.7%	39.2%

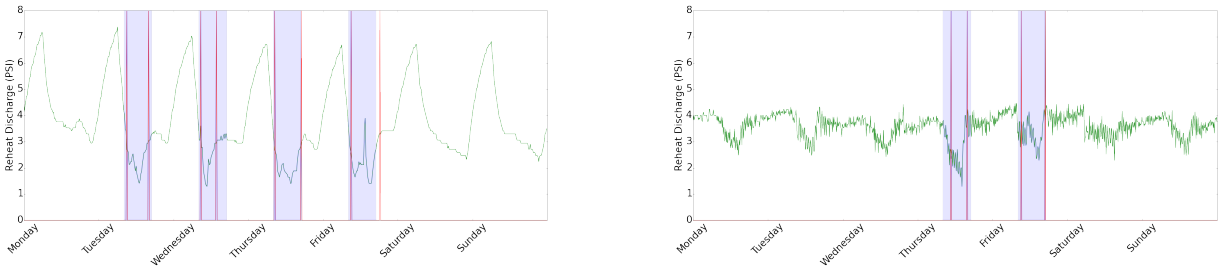
Table 6.1: Size of the clusters formed in each building

energy wastage in commercial buildings, should be detectable from apparent occupancy of the zones.

We divide the zones in a building into two broad categories, namely *anomalous* and *normal* zones, based on their apparent occupancy. To do this, we first identify the zones which are mostly unoccupied, *i.e.*, their apparent occupancy does not exceed 125% of its minimum level more than 90% of the time. These zones are removed from the set of the zones that will be clustered in the next step as they might be confused with anomalous zones. We then use agglomerative hierarchical clustering to group the remaining zones that show a similar pattern in their apparent occupancy between midnight and 6am of each day. Two features are selected to separate these anomalous zones from the rest of the zones: the average, and the peak to average ratio of apparent occupancy between midnight and 6am of each day of the week. We adopt complete-linkage clustering where the distance between two clusters is defined as the maximum distance between their members:

$$d(C_1, C_2) = \max_{\vec{x} \in C_1, \vec{y} \in C_2} \|\vec{x} - \vec{y}\|$$

The clustering algorithm is stopped when two clusters are formed.



(a) Zone A in Building 1

(b) Zone B in Building 1

Figure 6.9: Occupancy indicative signal (green curves), ground truth occupancy intervals (shaded areas), and inferred occupancy intervals (intervals between each pair of vertical red lines) of two offices during a week.

Table 6.1 shows the number of zones that were almost always unoccupied, as well as the number of zones that are clustered as normal and abnormal in each building. The unoccupied zones together with the normal zones are considered for applying energy-efficient schedules and the anomalous zones will be inspected by the facilities manager to locate, diagnose, and possibly correct the faults Note that Building 3 has a large number of zones housing biotechnology laboratories that must be cooled at all times. This results in their reheat valve being closed almost always, which translates into high apparent occupancy throughout the day, explaining the high percentage of anomalous zones in this building.

In an attempt to evaluate the clustering results and map atypical occupancy patterns to the known faults, the anomalous zones in Building 3 are checked with the facilities manager. These anomalous zones fall into four categories — (a) zones that contain heavy heat-generating equipment and therefore do not use any local reheat, *e.g.*, zones containing electron microscopes and pumps, (b) zones that are supposed to be kept at an extremely cold temperature, *e.g.*, zones containing freezers and microbial cultures, (c) corridors and closets which do not house occupants, and (d) faulty zones including a zone with stuck damper, two neighboring zones which exhibited simultaneous heating and cooling, and a zone where the occupants keep heat-generating equipment right next to the zone temperature sensor.

Evaluation

Full validation of the proposed unobtrusive occupancy detection approach is impractical owing to the lack of pervasive occupancy monitoring in our three commercial buildings. Relying on our knowledge of the approximate occupancy pattern of the zones to validate this approach will also be anecdotal. In light of this, we attempt to sanity check the occupancy hours identified by this approach and assess its efficacy in creating reasonable occupancy schedules through (a) matching the ground truth occupancy to the inferred occupancy intervals of the 8 zones where ground truth data were available, and (b) comparative analysis of the overall weekday and weekend occupancy profile of the buildings in our testbed.

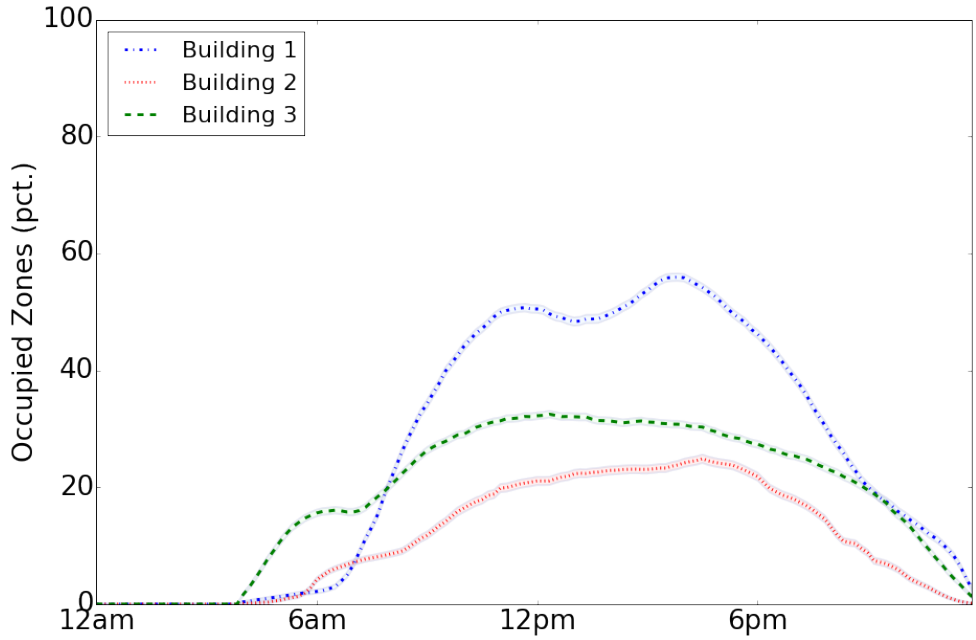


Figure 6.10: The average weekday occupancy of Buildings 1, 2, and 3 estimated over three months. The shaded boundary of each curve shows the 95% confidence interval of the average number of occupied zones divided by the total number of zones in that building.

Figure 6.9 depicts measurements of the reheat valve position of two of these zones over the period of one week, the ground truth occupancy of these zones, and the identified occupancy intervals[5]. It can be observed that our approach successfully identifies all occupancy intervals, and the inferred start and end times are within a few hours from the actual occupancy start and end times. These errors will not impact the inferred occupancy schedule of a zone since we have several weeks of measurements and exploit percentile statistics to define the schedules (described in Section 6.2.3) rather than choosing the earliest start the latest end times that are detected, which are more sensitive to noise and spurious occupancy detections.

We now inspect the aggregate occupancy profile of the zones within a building. Figure 6.10 shows the average number of zones that are identified as occupied on weekdays in a building, normalized by the total number of zones in that building, and the upper and lower 95% confidence limits. We see that at least 40%, 75%, and 67% of the zones are unoccupied at any time in Buildings 1, 2, and 3, respectively. Furthermore, the total number of occupied zones is higher between 4-6pm than any other time intervals in Buildings 1 and 2, while Building 3 reaches its maximum occupancy just before noon.

Figure 6.11 shows the start and end times of apparent weekday and weekend occupancy of the zones in Building 2, where the start and end times are defined as the 10th percentile and the 90th percentile of start times and end times, respectively. It also shows the number of times a

[5]The identified occupancy intervals and ground truth occupancy data were in agreement in the other 6 zones, but we do not show these graphs due to space constraint.

zone has been occupied at a particular time on a weekday or a weekend. Note that the vertical lines represent the average start and end times of the identified occupancy intervals. It can be seen that (1) several zones have never been occupied on weekends, (2) the zones were occupied on average for a longer period of time on weekdays than weekends, and (3) the average start time of occupancy intervals moves toward the afternoon on weekends as compared to weekdays, implying that people who come to their office on weekends tend to arrive later than the time they usually arrive on weekdays. These observations are perfectly reasonable for our buildings and suggest that rough occupancy estimates of the zones can serve the purpose of computing zone-specific schedules as discussed next.

6.2.3 Energy Savings Potential

An HVAC schedule to control the period of operation of a VAV can be programmed into a building's BMS. Most commercial buildings work on a fixed schedule, if any[6]. These schedules are set using the facilities manager's intuition and apply to all thermal zones regardless of their occupancy pattern. For instance, a facilities manager may arbitrarily decide that the HVAC system should maintain the environment comfortable between 6am and 11pm on weekdays. These schedules are, by design, conservative, because the facilities manager does not want to sacrifice occupants' comfort when they are present in the building.

We use the apparent occupancy profile for each thermal zone computed in Section 6.2.2 to develop possible VAV operation schedules, and compute the aggregate energy savings that can be obtained in a building. We report two metrics for each schedule in our experiments: Percentage Energy Saved on Reheat, and Percentage Occupant Comfort Violations.

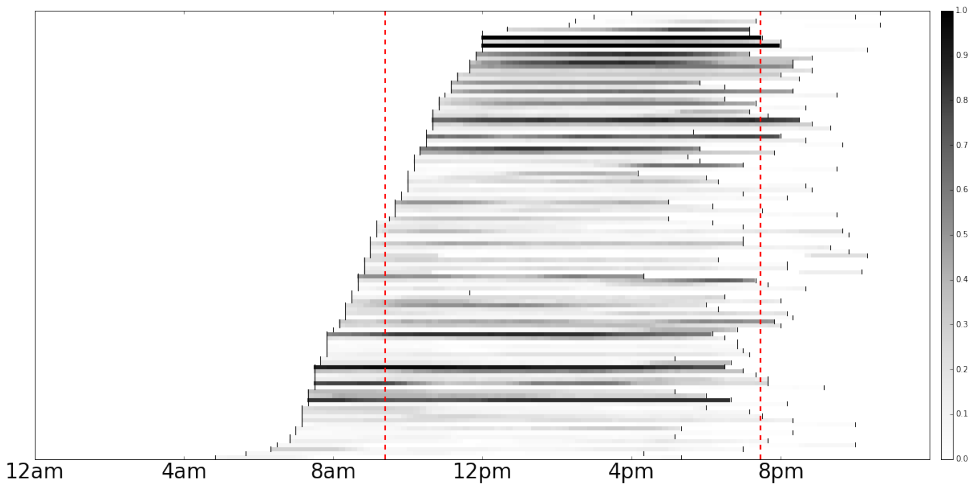
The Percentage Energy Saved on Reheat is the ratio, averaged over all zones in a building, of the amount of reheat used during the operation of a VAV under a schedule to a baseline when no schedule was in operation. When a schedule is not in operation, we assume the reheat energy used is zero. If a computed schedule mis-estimates occupancy, we consider the period of operation of the VAV to be the union of the apparent occupancy and schedule operation interval. Note that we do not report the overall building energy savings because we do not model the other aspects of HVAC energy consumption (such as AHU energy usage), and the buildings in our testbed did not provide fine-grained energy measurements for the HVAC system.

The Percentage Occupant Comfort Violations quantifies the mismatch between a schedule and actual occupancy, and is computed as the ratio, averaged over all zones, of the period of time a schedule mis-estimated occupancy[7] to the total duration of our study, *i.e.*, 3 months. Note that the schedule violations are only approximate since we do not consider the time that it takes to bring the zone temperature back to its setpoint when a schedule starts and the time that it takes for the zone temperature to deviate from the setpoint when a schedule ends.

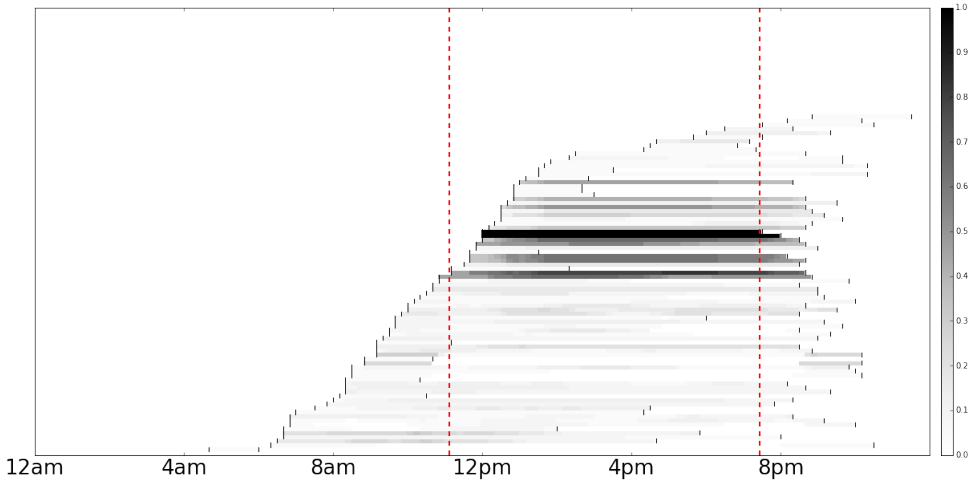
These two metrics expose a trade-off space for a facilities manager. A schedule could be overly aggressive and condition a zone only when there is a strong guarantee of occupancy. This

[6]In our testbed, only one building operated on a schedule where the VAVs maintained a zone's environment comfortable from 6am–11pm.

[7]That is if a schedule starts after or ends before a period of apparent occupancy.



(a) Weekdays Occupancy Profile



(b) Weekends Occupancy Profile

Figure 6.11: Occupancy profiles of Building 2 summarize the rough occupancy estimate of every zone in this building. Each horizontal line depicts the apparent occupancy of a zone and its 10th percentile and 90th percentile of start time and end time distributions, and vertical red lines represent the average occupancy start and end times of all zones. The darker a point is, the greater would be the number of days that the corresponding zone has been occupied at that particular time over the observation period.

would lead to higher energy savings at the expense of increasing the number of times an occupant arrives to find her zone unconditioned. On the other extreme, too conservative a schedule could ensure that zones are always conditioned when occupants are even remotely likely to be present, achieving much lower energy savings but resulting in much happier occupants.

As Figure 6.11 shows, occupancy of zones varies widely, and hence having a single building-

	Reheat Energy Savings		Violations	
Building	Learned Static	Naive	Learned Static	Naive
1	57.2%	52.7%	1.9%	12.7%
2	57.0%	37.2%	2.8%	2.9%
3	47.9%	49.9%	2.4%	11.0%

Table 6.2: Energy saved on reheat and occupant comfort violations for two static schedules

wide schedule would lead to significant inefficiency. Hence, we investigate customized per-zone schedules. In particular, we investigate *static schedules*, *i.e.*, zone-specific schedules which do not vary throughout our period of observation, and *adaptive schedules*, *i.e.*, zone-specific schedules which are continuously updated based on some window of observation.

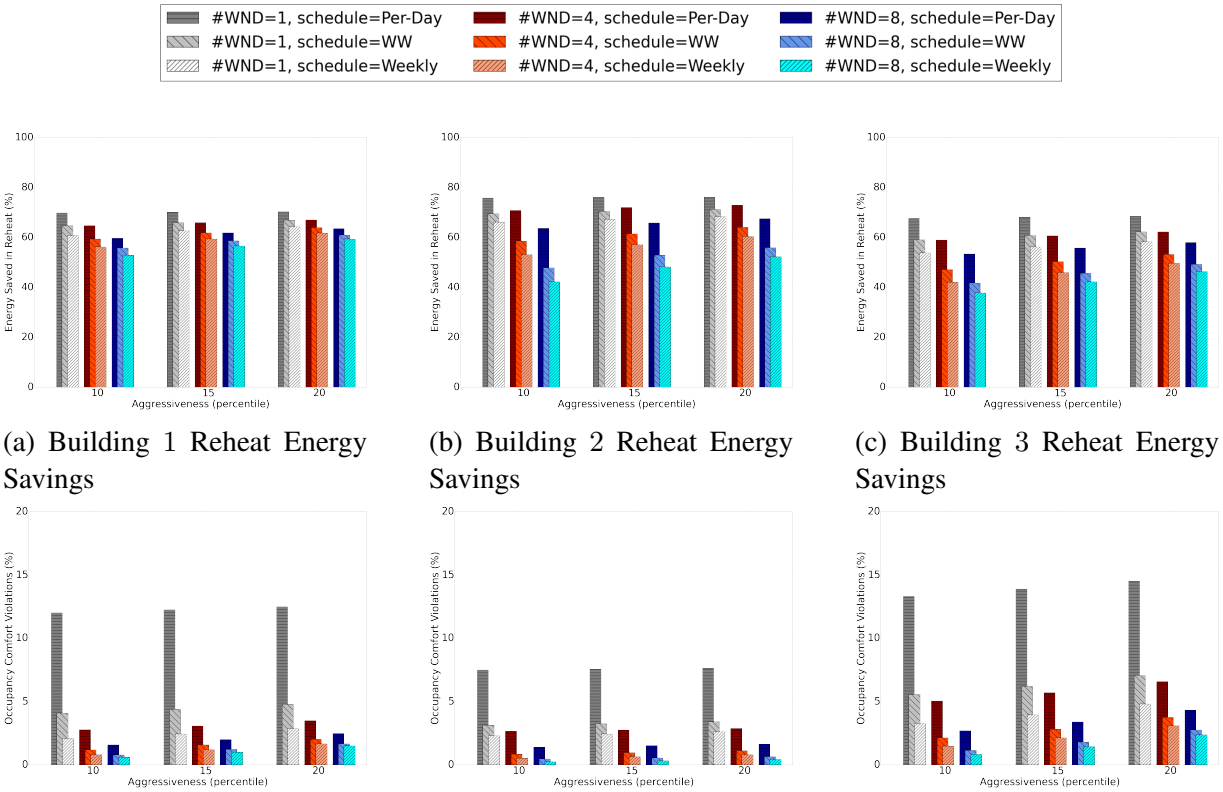
Energy Savings of Static Schedules

Static schedules are easy to implement in a commercial building, requiring only a one-time effort by a certified technician or domain expert to program the building BMS. In this subsection, we explore two kinds of static schedules —

- **Naive Schedule**: This schedule ensures that a VAV is operated during normal business hours, and only on weekdays. In our experiment, we evaluate the energy savings of a building under a predefined naive schedule where HVAC zones would only be conditioned from 6am-11pm on weekdays.
- **Learned Static Schedule**: This schedule is based on actual inferred zone occupancy data over a short period of time. Most existing BMS solutions allow for limited storage/trend capacity for HVAC sensor data, making such an analysis feasible. In our experiment, we learn per-zone schedules from a randomly chosen two-week duration of data.

Table 6.2 shows the results of simulating these two schedules on the three buildings in our testbed. Building 1 and Building 3 save roughly the same percentage of energy in reheat using either schedule. However, the percentage of occupant comfort violations is an order of magnitude higher under the Naive Schedule as compared to the Learned Static Schedule. This is because a Naive Schedule applies the same schedule across all zones and hence cannot account for the heterogeneity in occupancy periods across zones. Also a Naive Schedule assumes the building is unoccupied during weekends, when that may not be the case. For Building 2, the energy saved by the Learned Static Schedule is 20 percent higher than a Naive Schedule while resulting in a nearly similar percentage of occupant comfort violations. Building 2's occupancy for all zones on weekdays are mostly centered around normal occupancy hours (refer to Figure 6.11a). However, most occupants come in later than 6am and leave much earlier than 11pm, resulting in the Naive Schedule saving much lesser energy than a simple schedule that learned this pattern over the period of two weeks, while achieving the same percentage of comfort vi-

olations. The building is sparsely occupied on weekends (as evident from the light shades in Figure 6.11b), which does not lead to comfort violations under the Naive Schedules.



(a) Building 1 Reheat Energy Savings

(b) Building 2 Reheat Energy Savings

(c) Building 3 Reheat Energy Savings

(d) Building 1 Occupant Comfort Violations

(e) Building 2 Occupant Comfort Violations

(f) Building 3 Occupant Comfort Violations

Figure 6.12: Energy Savings on Reheat and Occupant Comfort Violations for three different levels of aggressiveness and different durations of training window (WND=1, 4, and 8 weeks) for three adaptive schedules. In general, higher energy savings and lower violations are favored.

Energy Savings of Adaptive Schedules

A building occupancy profile may change over time resulting in static schedules that do not provide optimal energy savings. In this subsection, we explore adaptive schedules which adjust the schedule timings based on the apparent occupancy of each zone in a sliding window that spans a fixed time interval in the past. Such schedules are capable of tracking the variable occupancy pattern of a zone.

Given historical data and the time series analysis techniques introduced in the previous section, we can obtain the empirical distribution function of the identified start and end times of apparent occupancy of every zone. This data, together with two adjustable parameters (a) the *aggressiveness* of the start and end times of the per-zone schedule, (b) the length of *training*

window used for learning the schedules, gives a more sophisticated way to model the potential energy savings in a building. The aggressiveness metric would determine which percentile of the start (end) times distribution would be used as the schedule's start time (end time)[8]. This is a parameter the facilities managers can adjust depending on the amount of energy savings they are willing to achieve and their tolerance for occupant discomfort. The size of the sliding window determines the amount of training data available for estimating the start and end time distributions of the apparent occupancy of a zone. The larger the window, the better the estimate of schedule timings is expected to be, provided that the actual zone occupancy is stationary during this window.

We explore three types of adaptive schedules —

- **Weekly Schedule**: A single schedule would be in operation for the entire week.
- **Per-Day Schedule**: A different schedule is calculated for each specific day of the week. This helps capture day-specific occupancy patterns, *e.g.*, in the case that the occupant works from home on Fridays.
- **Weekday-Weekend (WW) Schedule**: A single schedule would be in operation on weekdays and a different one would be in operation on weekends. This schedule is expected to result in more energy savings in zones that are mostly unoccupied on weekends.

Figure 6.12 shows the results of applying these three schedules on the buildings in our testbed for different levels of aggressiveness and different duration of the training window. The following observations can be made based on these results— First, the Weekday-Weekend schedules offer a sweet-spot between the two other schedules. Applying a Weekday-Weekend schedule saves more energy than a Weekly Schedule which does not take into account the inherent difference in occupancy over weekends, and saves less energy than a Per-Day Schedule; however, it results in remarkably lower percentage of occupant comfort violations than Per-Day Schedules. Second, while Weekly Schedules save less energy than Per-Day Schedules, they also result in lower comfort violations. This is expected as Per-Day Schedules are computed over lesser training data as compared to the other schedules. Third, the increase in the duration of the training window results in better occupancy estimates and hence lower occupant comfort violations and lesser energy savings. Finally, increasing the aggressiveness of a schedule generally leads to larger energy savings but larger occupant comfort violations as well.

Figure 6.13 shows the average occupancy profile of Building 1 along with its 95% confidence interval represented by a shaded band around it, and the normalized aggregate energy consumed to reheat all the zones over our period of observation, *i.e.*, when no schedule was in effect. It can be seen from the comparison of the shape of these two curves that a lot of energy is wasted to reheat the building overnight when it is mostly unoccupied. If a Per-Day or Weekly schedule is applied, the resultant reheat energy consumption profile almost resembles the occupancy profile.

[8] If the xth percentile is used for determining the schedule start time, (100-x)th percentile is used for determining its end time.

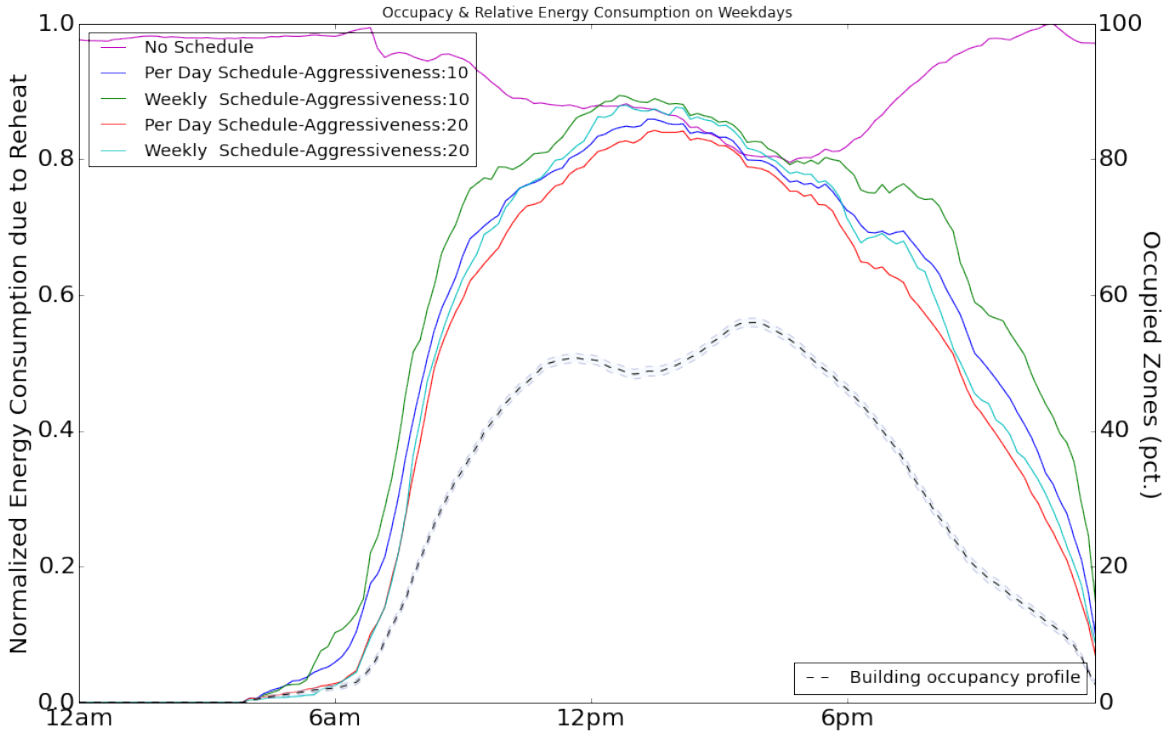


Figure 6.13: Weekday occupancy profile of Building 1 (y-axis on the right) and the normalized energy consumption of the building due to reheat at the present time (no schedule), and under Per-Day and Weekly schedules using 8 weeks of training data. The energy consumption profiles are normalized by the maximum energy consumption that was recorded when the building does not run a schedule. The potential energy saved on reheat for a given schedule is the area between its curve and the baseline No-Schedule curve.

This indicates that (1) these schedules do reflect the occupancy of each zone and (2) most energy savings come from closing the reheat valve during the night.

Trade-offs in Schedule Complexity

There is a three-way trade-off between complexity of schedules, their energy savings potential, and their occupancy comfort violations. A hypothetical per-zone scheduler that relies on an oracle that accurately predicts occupancy hours of the zones will attain maximum energy savings and minimum violations. Since the prescribed schedules must be updated every day, this would be the most difficult one to implement. On the other hand, static schedules are the easiest to implement; however, they lead to a considerable number of violations unless the occupancy pattern of the zones is highly predictable and time invariant. Adaptive schedules are expected to be better in terms of occupant comfort violations and can also result in higher saved energy on reheat, though they are more difficult to apply than the static schedules.

Interestingly, the Static Learned Schedules seem to be the sweet spot for our testbed. In particular, they achieve more than 48% reduction in reheat energy consumption across the buildings, while resulting in less than 3% violations. This implies that the occupancy pattern of the zones across the three buildings is stationary over these three months; hence, the random choice of the training weeks does not cause significant violations. Sophisticated adaptive schedules could achieve between 37%–76% energy savings, and between 1%–8% comfort violations across the buildings.

6.2.4 Related Work

In recent years, substantial efforts have been devoted to reducing the demand of residential and commercial buildings by adding intelligence to appliances, optimizing HVAC controls, and detecting faults in buildings. The knowledge of occupancy, at different scales, is essential to achieve the energy saving targets with imperceptible impact on building operations and human comfort. This has given rise to many systems designed for sensing occupancy of the spaces within a home or a commercial building.

The extensive body of literature on building occupancy monitoring can be classified into two categories: methods that require deployment of additional sensors and those that leverage existing infrastructure. The intrusive methods, which are the former category, aim at sensing occupancy by fusing data from one or multiple types of sensors, including passive infrared (PIR) motion sensors, carbon-dioxide (CO_2) sensors, temperature sensors, cameras, magnetic reed switches, acoustic sensors, differential pressure sensors, and plug-in electricity meters [174, 171, 152, 13, 74, 129, 85, 145, 35, 126]. For instance, Dong et al. [74] deploy multiple sensors (acoustic, CO_2, light, and motion) to estimate the number of occupants and occupancy duration using Hidden Markov Models. These methods require retrofitting buildings with dedicated hardware for occupancy detection, posing several challenges from sensor placement and calibration to ensuring that the sensors have a reliable network connection and power supply.

To address these limitations, several methods have been proposed that leverage existing infrastructure to estimate occupancy. Specifically, WiFi access points, smart meters, HVAC sensors, and calendar feeds are used to infer occupancy. These methods are scalable, more affordable, and less intrusive. For example, an occupancy based HVAC actuation system is developed in [28] relying on inferred occupancy from WiFi network logs and building occupant metadata. The system has a false negative detection rate of 6.2% in personal spaces and achieves savings of 17.8% in HVAC energy consumption of a commercial building. However, the proposed occupant inference algorithm assumes that smart phones are continuously connected to the wireless network, requires information about all wireless capable devices used by an occupant, and does not apply to shared spaces, *e.g.*, meeting rooms and lobbies.

Coarse-grained electricity data produced by smart meters are used to infer home occupancy in [51]. Using a simple threshold-based algorithm that detects changes in three statistical metrics of smart grid data, namely average, standard deviation, and range, the authors generated a continuous track of daytime household occupancy. This approach does not require any training data for occupancy and appliance loads. In a similar line of work, home occupancy is detected from

smart meter data using a supervised learning algorithm [136]. Using 35 features and various classifiers, the authors reported a detection accuracy between 83%-94% in five households. The major drawback of both approaches is that they cannot be readily applied to large commercial buildings with many zones and a vast number of appliances.

In recent work, measurements of environmental signals, such as CO_2 concentration, room temperature, and ventilation actuation levels are used to estimate the occupancy levels [76]. Despite the novelty of this approach, it requires a burdensome initial training phase to relate the number of occupants with the CO_2 concentration level. Moreover, CO_2 sensor readings are often not available in HVAC systems. Existing computing and security infrastructure in commercial buildings is also exploited in [103] to infer occupancy. In particular, the authors use minutely data collected from area access badges, WiFi access points, calendars, and instant messaging clients for occupancy estimation. In most commercial buildings, this information is not available for all zones and therefore cannot be incorporated into the HVAC control. Finally, a simple HVAC optimization application is implemented in [67] that uses only time of day and calendar feed to estimate room occupancy and adjust the ventilation rate as a function of the number of people in a room.

Aswani et al. [22] also employ semi-parametric regression to estimate the heat load from occupancy, equipment, and solar radiation using measurements of the zone temperature and VAV control signals. This approach cannot separate the effect of occupancy from heat-generating equipment, solar heating, and noise, rendering it of limited practical value. Moreover, model parameter estimation for every single zone is error prone.

6.2.5 Conclusions

We investigates the potential of unobtrusive occupancy detection techniques to produce a rough estimate of occupancy at the level of individual zones from coarse-grained measurements of the VAV system. Our experiments on three large commercial buildings over a period of three months corroborate that the estimated occupancy patterns are sufficiently accurate to compute customized per-zone schedules that can significantly reduce the energy consumption of the VAV systems with only a small number of occupant comfort violations. The proposed approach can be readily applied to any building with a BMS that archives data from HVAC sensors, thereby enabling facilities managers to quantify and explore the complexity, comfort, and energy savings trade-off. The plausibility of our results underlines that much value can be extracted from existing building data streams through careful analytics and justifies the effort to collect ground truth data on a reasonably large set of buildings where occupancy sensors are pervasive. Such a data set could further validate and refine the proposed techniques.

Chapter 7

Conclusion

7.1 Contributions

The techniques developed in this thesis answered the question of how to deploy applications scalably across diverse smart-buildings comprising large apriori deployed sensor networks, each set up with deployment-specific, obscure metadata. We presented techniques to normalize the existing metadata of such deployments into a common namespace, designed a common namespace schema which can capture all required sensors and relationships, and then demonstrated the wide array of applications which could exploit the resulting normalized metadata schema and run unmodified across smart-buildings.

Specifically, we make five major contributions in this thesis. First, we design an *empirical* criteria to evaluate the effectiveness of existing smart-building metadata schemas. Prior schemas, as well as newly proposed ones can now get a score on each of the metrics we laid out, which are (1) completeness, (2) ability to capture relationships, and (3) flexibility and ease-of-use. This contribution helps eliminate subjective comparisons of smart-building metadata schemas, and provides a firm grounding to evaluate the efficacy of schemas going forward. We show that all of the existing popular existing metadata schemas (Project Haystack, Industry Foundation Classes, and Semantic Sensor Webs) fall well short of solving the metadata challenges in smart-buildings.

Second, we develop techniques to transform legacy metadata of building sensors to a normalized schema. We showed how to circumvent the inherent challenges of parsing the inconsistent and noisy structure of legacy metadata via an automated synthesis technique coupled with data-driven clustering and classification. Our technique can transform legacy metadata into a well-formed representation using a small number of examples from an expert, e.g., the building manager. The transformation to such a namespace yields *semantic relationships between sensors*, which enables analytics applications to be deployed without a priori building-specific knowledge. We demonstrated that our technique is robust and achieves full sensor qualification for all sensors for 3 different buildings sensor systems, even when presented with obscure and noisy metadata tags. Our technique takes very few examples to fully normalize the most

commonly occurring sensors (24, 15 and 43 examples for the three buildings in our testbed for qualifying 70% of the tags).

Third, we presented a technique to capture missing sensor/subsystem relationships in a smart-building using active perturbations of relevant subsystems and performing a voting-based analysis on the resulting data. Our perturbation technique is totally transparent to building occupants and can correctly identify missing functional relationships in roughly 80% of the cases.

Fourth, we developed and defined a schema, Brick, that we believe is a strong candidate to solving uniform metadata standards problem. Brick builds upon prior work and introduces a number of novel concepts that we believe adresses this open problem. Brick has an underlying graphical data model, and defines nodes and edges to capture all possible sensor information and relationships in smart-buildings. Brick uses clear tags and tagsets to specify sensors and subsystems in a building. It defines an ontology and a class hierarchy for this list of tags and tagsets. Relationships are represented as triples, which allows us to leverage existing tools to build and query the resulting building representations. Brick proposes Functional Blocks to abstract out complexity but also aid in system composition and hierarchies. Finally, Brick uses the notion of synonyms to equate sensors and subsystems similar in function. Brick is complete, capturing an average of 98% of BMS data points across five diverse buildings comprising almost 15,700 data points and 615,000 sq-ft of floor space. Brick is expressive, successfully running eight canonical applications on these buildings. Four applications ran on all five buildings, while the remaining applications ran on buildings whose BMS exposed the requisite points.

Finally, we showed a diverse set of applications, ranging from simple diagnostics, e.g finding errant rogue zones, finding stuck dampers and identifying inefficient air handling units, to complex occupancy modeling which can now be run at scale, thanks to normalized metadata of sensors across smart-buildings. We demonstrate the results of each of the applications on multiple diverse buildings, each comprising 1000s of sensors.

7.2 Future Work

There are several avenues of research which were not addressed in this thesis. Although we proposed a normalized metadata Brick schema which satisfies certain empirical objectives and techniques to augment transform legacy existing sensor metadata in millions of buildings, making this process efficient still requires additional work. There is a need to perform usability studies with human experts, to devise intuitive ways to enable them navigate through the large building datasets (related research efforts include [37, 155]) and come up with techniques to help the expert and make our system robust to errors in the expert's input. Brick also needs an equivalent of a model-checker to ensure the correctness of the generated metadata. Manually debugging why certain applications were not running on certain buildings in our case study required quite a bit of effort, and the answer was invariably some error in translating existing labels to tags/tagsets, misuse of tags when they should not have been used, or expressing semantically wrong relationships.

Even though our testbed comprised of 10s of buildings comprising different vendors, subsystems and geographic locations, scaling these techniques to even larger scale will probably require modifications to the schema, and additional techniques to normalize their existing metadata. How to choose the right metadata constructs and the right set of techniques to normalize legacy metadata in each particular setting is still an open question.

Third, our metadata schema proposal is mostly aimed at providing a uniform namespace for applications that do not require detailed simulation of a building. Building simulators require more information than simply sensors and their inter-relationships and this information needs to be scraped from other information sources, such as blueprints and user manuals. Devising a way to feed the data obtained through our model back into the detailed simulation tools will not only lead to better adoption our proposed schemas, but also mitigate a lot of the data modeling concerns in the simulation tools.

Finally, on the systems side we need to enable transparent deployment of applications on both existing BMS systems and novel sensor network deployments. Often commercial buildings have a mix of both, and effectively and seamlessly navigating this heterogeneity in sensor interaction and data collection requires further research. Also, this thesis does not deal with provenance, versioning and data integrity issues that arise out of changes in metadata that arise in a building's lifecycle. Additionally, we need to address data associated with occupants and their interaction with building sensors as a first-class citizen in the building information plane, something that was not addressed in this work.

Bibliography

[1] Brick Schema. https://github.com/BuildSysUniformMetadata/GroundTruth.

[2] List of building applications. https://ibm.biz/UCB-IBM-Apps.

[3] OWL Namespace. http://www.w3.org/2002/07/owl#.

[4] Project haystack. http://project-haystack.org/.

[5] RDF Concepts Namespace. http://www.w3.org/1999/02/22-rdf-syntax-ns#.

[6] RDF Schema Namespace. https://www.w3.org/2000/01/rdf-schema#.

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

[8] The Internet Engineering Task Force (IETF). https://www.ietf.org/.

[9] Turtle. https://www.w3.org/TR/turtle/.

[10] Used building metadata. https://ibm.biz/UCB-IBM-Data.

[11] ENERGY INFORMATION ADMINISTRATION. Commercial buildings energy consumption survey,. Technical report, 1999.

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

[13] Y. Agarwal, B. Balaji, S. Dutta, R. K. Gupta, and T. Weng. Duty-cycling buildings aggressively: The next frontier in HVAC control. In *IPSN*, pages 246–257, April 2011.

[14] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *BuildSys*, pages 1–6. ACM, 2010.

[15] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *BuildSys*, pages 1–6. ACM, 2010.

[16] ALC. Automated logic corporation. `http://www.automatedlogic.com/`.

[17] American Society of Heating, Refrigerating and Air-Conditioning Engineers. ASHRAE Standard 135-1995: BACnet. ASHRAE, Inc., 1995.

[18] N. Arana, A. Noguero, T. Padilla, and MJ. Mtz. de Lizarduy. D2.2 - a ontology for device awareness, 2009.

[19] Omid Ardakanian, Arka Bhattacharya, and David Culler. Non-intrusive techniques for establishing occupancy related energy savings in commercial buildings. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pages 21–30. ACM, 2016.

[20] Pandarasamy Arjunan, Nipun Batra, Haksoo Choi, Amarjeet Singh, Pushpendra Singh, and Mani B Srivastava. Sensoract: a privacy and security aware federated middleware for building management. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 80–87. ACM, 2012.

[21] ASHRAE. Standard 90.1-2013. `https://www.ashrae.org/resources--publications/bookstore/standard-90-1`.

[22] A. Aswani, N. Master, J. Taneja, V. Smith, A. Krioukov, D. Culler, and C. Tomlin. Identifying models of HVAC systems using semiparametric regression. In *American Control Conference (ACC)*, pages 3675–3680, June 2012.

[23] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Towards a unified metadata schema for buildings. In *Proceedings of the ACM International Conference on Embedded Systems for Energy-Efficient Built Environments (BuildSys). ACM*, 2016.

[24] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In *BuildSys*. ACM, 2013.

[25] Bharathan Balaji, Hidetoshi Teraoka, Rajesh Gupta, and Yuvraj Agarwal. Zonepac: Zonal power estimation and control via hvac metering and occupant feedback. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[26] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 13–22. ACM, 2015.

[27] Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *BuildSys*, pages 13–22. ACM, 2015.

[28] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: Occupancy based HVAC actuation using existing wifi infrastructure within commercial buildings. In *SenSys*, pages 17:1–17:14. ACM, 2013.

[29] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 17. ACM, 2013.

[30] Nipun Batra, Manoj Gulati, Amarjeet Singh, and Mani B Srivastava. It's different: Insights into home energy consumption in india. In *BuildSys*. ACM, 2013.

[31] Vladimir Bazjanac and DB Crawley. Industry foundation classes and interoperable commercial software in support of design of energy-efficient buildings. In *Proceedings of Building SimulationâĂŹ99*, volume 2, pages 661–667, 1999.

[32] J. Beetz, J. Van Leeuwen, and B. De Vries. IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 23(01):89–101, 2009.

[33] Gowtham Bellala, Manish Marwah, Martin Arlitt, Geoff Lyon, and Cullen E Bash. Towards an understanding of campus-scale power consumption. In *BuildSys*, pages 73–78. ACM, 2011.

[34] Gowtham Bellala, Manish Marwah, Martin Arlitt, Geoff Lyon, and Cullen E Bash. Towards an understanding of campus-scale power consumption. In *BuildSys*, pages 73–78. ACM, 2011.

[35] Alex Beltran, Varick L. Erickson, and Alberto E. Cerpa. Thermosense: Occupancy thermal based sensing for HVAC control. In *BuildSys*, pages 11:1–11:8. ACM, 2013.

[36] Alex Beltran, Varick L Erickson, and Alberto E Cerpa. Thermosense: Occupancy thermal based sensing for hvac control. In *BuildSys*. ACM, 2013.

[37] Arka Bhattacharya, David Culler, Dezhi Hong, Kamin Whitehouse, and Jorge Ortiz. Writing scalable building efficiency applications using normalized metadata: demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 196–197. ACM, 2014.

[38] Arka Bhattacharya, David E. Culler, Jorge Ortiz, Dezhi Hong, and Kamin Whitehouse. Enabling portable building applications through automated metadata transformation. Technical Report UCB/EECS-2014-159, EECS Department, University of California, Berkeley, Aug 2014.

[39] Arka Bhattacharya, Joern Ploennigs, and David Culler. 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*, pages 33–34. ACM, 2015.

[40] Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. Automated metadata construction to support portable building applications. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 3–12. ACM, 2015.

[41] Dario Bonino and Fulvio Corno. DogOnt – ontology modeling for intelligent domotic environments. In *ISWC - Int. Semantic Web Conf.*, volume 5318, pages 790–803. 2008.

[42] M. Botts and A. Robin. OpenGIS sensor model language (SensorML) implementation specification, 2007. OpenGIS Implementation Specification OGC.

[43] Mic Bowman, Saumya K. Debray, and Larry L. Peterson. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.*, 15(5), November 1993.

[44] Johannes Braams. Babel, a multilingual style-option system for use with latex's standard document styles. *TUGboat*, 12(2):291–301, June 1991.

[45] Brown and Caldwell. Scada analysis. `http://sswd.org/modules/showdocument.aspx?documentid=966`.

[46] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[47] Tomo Cerovsek. A review and outlook for a âĂŸbuilding information modelâĂŹ(bim): A multi-standpoint framework for technological development. *Advanced engineering informatics*, 25(2):224–244, 2011.

[48] K. Charatsis, A.P. Kalogeras, M. Georgoudakis, J. Gialelis, and G. Papadopoulos. Home / building automation environment architecture enabling interoperability, flexibility and reusability. In *ISIE - IEEE Int. Symp. on Ind. Electronics*, volume 4, pages 1441–1446, 2005.

[49] Victor Charpenay, Sebastian Kabisch, Darko Anicic, and Harald Kosch. An ontology design pattern for iot device tagging systems. In *Internet of Things (IOT), 2015 5th International Conference on the*, pages 138–145. IEEE, 2015.

[50] Tanushyam Chattopadhyay and Sangheeta Roy. Human localization at home using kinect. In *2013 ACM Conf. on Pervasive and ubiquitous computing adjunct publication*, pages 821–828. ACM, 2013.

[51] Dong Chen, Sean Barker, Adarsh Subbaswamy, David Irwin, and Prashant Shenoy. Non-intrusive occupancy monitoring using smart meters. In *BuildSys*, pages 9:1–9:8. ACM, 2013.

[52] Dong Chen, Sean Barker, Adarsh Subbaswamy, David Irwin, and Prashant Shenoy. Non-intrusive occupancy monitoring using smart meters. In *BuildSys*. ACM, 2013.

[53] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. Indoor localization without the pain. In *Proceedings of the Sixteenth Annual International Conference on Mobile Computing and Networking*, MobiCom '10, pages 173–184. ACM, 2010.

[54] CitectSCADA. Tagging tutorial. `http://www.citect.com.tw/download/files/1hr_Quickstart_Tutorial.pdf`.

[55] Malcolm Clark. Post congress tristesse. In *TeX90 Conference Proceedings*, pages 84–89. TeX Users Group, March 1991.

[56] Industry Foundation Classes. Industry foundation classes. `http://www.ifcwiki.org/index.php/Main_Page`.

[57] M. Compton, C. Henson, L. Lefort, H. Neuhaus, and A. Sheth. A survey of the semantic specification of sensors. In *SSN - Semantic Sensor Networks*, pages 17–32, 2009.

[58] Michael Compton, Payam Barnaghi, Luis Bermudez, RaúL GarcíA-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[59] Michael Compton, Payam Barnaghi, Luis Bermudezc, and et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2012.

[60] Drury B. Crawley, Curtis O. Pedersen, Linda K. Lawrie, and Frederick C. Winkelmann. EnergyPlus: Energy simulation program. *ASHRAE Journal*, 42:49–56, 2000.

[61] CUErgo. Ambient environment: Thermal conditions. `http://ergo.human.cornell.edu/studentdownloads/DEA3500notes/Thermal/thcondnotes.html`, 2016, retrieved.

[62] Laura Daniele, Frank den Hartog, and Jasper Roes. Study on semantic assets for smart appliances interoperability, March 2015.

[63] LM Daniele, FTH den Hartog, and JBM Roes. Study on semantic assets for smart appliances interoperability: D-s4: Final report. Technical report, European Union, 2015.

[64] Anish Das Sarma, Aditya Parameswaran, Hector Garcia-Molina, and Jennifer Widom. Synthesizing view definitions from data. In *Proceedings of the 13th International Conference on Database Theory*, pages 89–103. ACM, 2010.

[65] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. smap: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 197–210, New York, NY, USA, 2010. ACM.

[66] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. smap: A simple measurement and actuation profile for physical information. In *SenSys*, pages 197–210. ACM, 2010.

[67] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David Culler. Boss: Building operating system services. In *NSDI*, pages 443–458. USENIX Association, 2013.

[68] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David E Culler. Boss: Building operating system services. In *NSDI*, volume 13, pages 443–458, 2013.

[69] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David E Culler. Boss: Building operating system services. In *NSDI*, volume 13, pages 443–458, 2013.

[70] Hong Dezhi. Sensor type classification in buildings. `http://www.abc.com/`.

[71] Luigi Di Caro, K Selçuk Candan, and Maria Luisa Sapino. Using tagflake for condensing navigable tag hierarchies from tag clouds. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1069–1072. ACM, 2008.

[72] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A Tomlin, et al. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM, 2003.

[73] Colin Dixon, Ratul Mahajan, Sharad Agarwal, A.J. Brush, Bongshin Lee, Stefan Saroiu, and Paramvir Bahl. An operating system for the home. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, pages 337–352, San Jose, CA, 2012. USENIX.

[74] Bing Dong and Khee Poh Lam. Building energy and comfort management through occupant behaviour pattern detection based on a large-scale environmental sensor network. *Building Performance Simulation*, 4(4):359–369, 2011.

[75] E William East. Construction operations building information exchange (Cobie). Technical report, DTIC Document, 2007.

[76] A. Ebadat, G. Bottegal, D. Varagnolo, B. Wahlberg, and K. H. Johansson. Regularized deconvolution-based approaches for estimating room occupancies. *IEEE Transactions on Automation Science and Engineering*, 12(4):1157–1168, Oct 2015.

[77] Afrooz Ebadat, Giulio Bottegal, Damiano Varagnolo, Bo Wahlberg, and Karl H Johansson. Estimation of building occupancy levels through environmental signals deconvolution. In *BuildSys*. ACM, 2013.

[78] Afrooz Ebadat, Giulio Bottegal, Damiano Varagnolo, Bo Wahlberg, and Karl H. Johansson. Estimation of building occupancy levels through environmental signals deconvolution. In *BuildSys*, pages 8:1–8:8. ACM, 2013.

[79] Echelon Corporation. LonTalk Protocol Specification. Echelon Corp. 1994.

[80] Energy Efficiency. Buildings energy data book. *US Department of Energy. http://buildingsdatabook. eere. energy. gov/*, 2009.

[81] M. Eisenhauer, P. Rosengren, and P. Antolin. A development platform for integrating wireless devices and sensors into ambient intelligence systems. In *SECON - 6th An. IEEE Com. Soc. Conf. on Sensor, Mesh and Ad Hoc Com. and Networks Workshops*, pages 1–3, 2009.

[82] DLT & V Sytems Engineering. Arapahoe county water and wastewater authority electrical, instrumentation and scada system design standards. `http://www.arapahoewater.org/documents/SCADA_Standards.pdf`.

[83] Flatirons Engineering. Scada tagging standards, wastewater treatment division, cincinati. `http://bit.ly/1LvfJQn`.

[84] V. L. Erickson, M. Á. Carreira-Perpi nán, and A. E. Cerpa. OBSERVE: Occupancy-based system for efficient reduction of HVAC energy. In *IPSN*, pages 258–269, April 2011.

[85] Varick L. Erickson, Stefan Achleitner, and Alberto E. Cerpa. POEM: Power-efficient occupancy-based energy management system. In *IPSN*, pages 203–216. ACM, 2013.

[86] Varick L Erickson and Alberto E Cerpa. Occupancy based demand response hvac control strategy. In *BuildSys*, pages 7–12. ACM, 2010.

[87] Varick L Erickson and Alberto E Cerpa. Thermovote: participatory sensing for efficient building hvac conditioning. In *BuildSys*, pages 9–16. ACM, 2012.

[88] Varick L Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E Cerpa, Michael D Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *BuildSys*, pages 19–24. ACM, 2009.

[89] Varick L. Erickson, Yiqing Lin, Ankur Kamthe, Rohini Brahme, Amit Surana, Alberto E. Cerpa, Michael D. Sohn, and Satish Narayanan. Energy efficient building environment control strategies using real-time occupancy measurements. In *BuildSys*, pages 19–24. ACM, 2009.

[90] Katja Filippova and Keith B. Hall. Improved video categorization from text metadata and user comments. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 835–842, New York, NY, USA, 2011. ACM.

[91] Romain Fontugne, Jorge Ortiz, and David Culler. Empirical mode decomposition for intrinsic-relationship extraction in large sensor deployments.

[92] Romain Fontugne, Jorge Ortiz, Nicolas Tremblay, Pierre Borgnat, Patrick Flandrin, Kensuke Fukuda, David Culler, and Hiroshi Esaki. Strip, bind, and search: A method for identifying abnormal energy consumption in buildings. In *IPSN*, pages 129–140. ACM, 2013.

[93] Ge Gao and Kamin Whitehouse. The self-programming thermostat: optimizing setback schedules based on home occupancy patterns. In *BuildSys*, pages 67–72. ACM, 2009.

[94] Ge Gao and Kamin Whitehouse. The self-programming thermostat: Optimizing setback schedules based on home occupancy patterns. In *BuildSys*, pages 67–72. ACM, 2009.

[95] Jingkun Gao, Joern Ploennigs, and Mario Berges. A data-driven meta-data inference framework for building automation systems. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 23–32. ACM, 2015.

[96] Peter Xiang Gao and S Keshav. Optimal personal comfort management using spot+. In *BuildSys*. ACM, 2013.

[97] Peter Xiang Gao and S. Keshav. Optimal personal comfort management using spot+. In *BuildSys*, pages 22:1–22:8. ACM, 2013.

[98] Daniel Garnier-Moiroux, Fernando Silveira, and Anmol Sheth. Towards user identification in the home from appliance usage patterns. In *2013 ACM Conf. on Pervasive and ubiquitous computing adjunct publication*, pages 861–868. ACM, 2013.

[99] Daniel Garnier-Moiroux, Fernando Silveira, and Anmol Sheth. Towards user identification in the home from appliance usage patterns. In *UbiComp Adjunct*, pages 861–868. ACM, 2013.

[100] GBXML. Green building xml. `http://www.gbxml.org/`.

[101] NIST GCR. Cost analysis of inadequate interoperability in the US capital facilities industry. *National Institute of Standards and Technology (NIST)*, 2004.

[102] N. Georgantas, S.B. Mokhtar, Y. Bromberg, V. Issarny, J. Kalaoja, J. Kantarovitch, A. Gerodolle, and R. Mevissen. The Amigo service architecture for the open networked home environment. In *WICSA - 5th Working IEEE/FIP Conf. on Softw. Archit.*, pages 295–296, 2005.

[103] S. K. Ghai, L. V. Thanayankizil, D. P. Seetharam, and D. Chakraborty. Occupancy detection in commercial buildings using opportunistic context sources. In *Pervasive Computing and Communications Workshops, IEEE International Conference on*, pages 463–466, March 2012.

[104] Google. Google earth 3d buildings. `http://www.google.com/earth/explore/showcase/3dbuildings.html`.

[105] Jessica Granderson, Mary Ann Piette, Ben Rosenblum, and Lily Hu. Energy information handbook: Applications for energy-efficient building operations. 2013.

[106] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18, 2005.

[107] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, pages 317–330, New York, NY, USA, 2011. ACM.

[108] Sumit Gulwani. Synthesis from examples. *WAMBSE (Workshop on Advances in Model-Based Software Engineering) Special Issue, Infosys Labs Briefings*, 10(2), 2012. Invited talk paper.

[109] Sumit Gulwani, William R. Harris, and Rishabh Singh. Spreadsheet data manipulation using examples. In *In Communications of the ACM*, 2012.

[110] Lam Abraham Hang-yat and Dan Wang. Carrying my environment with me: A participatory-sensing approach to enhance thermal comfort. In *BuildSys*. ACM, 2013.

[111] William R. Harris and Sumit Gulwani. Spreadsheet table transformations from examples. In *Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '11, pages 317–328, New York, NY, USA, 2011. ACM.

[112] Simon Hay and Andrew Rice. The case for apportionment. In *BuildSys*, pages 13–18. ACM, 2009.

[113] Maurice Herlihy. A methodology for implementing highly concurrent data objects. *ACM Trans. Program. Lang. Syst.*, 15(5):745–770, November 1993.

[114] A. Herzog, D. Jacobi, and A. Buchmann. A3ME - an agent-based middleware approach for mixed mode environments. In *UBICOMM - 2nd Int. Conf. on Mobile Ubiq. Comp., Syst., Serv. and Techn.*, pages 191–196, 2008.

[115] Dezhi Hong, Jorge Ortiz, Arka Bhattacharya, and Kamin Whitehouse. Sensor-type classification in buildings. *arXiv preprint arXiv:1509.00498*, 2015.

[116] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 13:1–13:8, New York, NY, USA, 2013. ACM.

[117] Dezhi Hong, Jorge Ortiz, Kamin Whitehouse, and David Culler. Towards automatic spatial verification of sensor placement in buildings. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[118] Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. The building adapter: Towards quickly applying building analytics at scale. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 123–132. ACM, 2015.

[119] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 1998.

[120] Norden E. Huang, Zheng Shen, Steven R. Long, Manli C. Wu, Hsing H. Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H. Liu. The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995, 1998.

[121] ISO 16739:2005 - Industry Foundation Classes, Release 2x, Platform Specification (IFC2x Platform), 2005.

[122] Marco Jahn, Tobias Schwartz, Jonathan Simon, and Marc Jentsch. Energypulse: tracking sustainable behavior in office environments. In *2nd Int. Conf. on Energy-Efficient Computing and Networking*, pages 87–96. ACM, 2011.

[123] Marco Jahn, Tobias Schwartz, Jonathan Simon, and Marc Jentsch. Energypulse: tracking sustainable behavior in office environments. In *Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking*, pages 87–96. ACM, 2011.

[124] Farrokh Jazizadeh and Burcin Becerik-Gerber. Toward adaptive comfort management in office buildings using participatory sensing for end user driven control. In *BuildSys*, 2012.

[125] JCI. Johnson controls building management. `http://www.automatedlogic.com/`.

[126] Ming Jin, Nikolaos Bekiaris-Liberis, Kevin Weekly, Costas Spanos, and Alexandre Bayen. Sensing by proxy: Occupancy detection based on indoor CO2 concentration. In *Proceedings of the 9th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 1–10, 2015.

[127] Deokwoo Jung, Varun Badrinath Krishna, Ngo Quang Minh Khiem, Hoang Hai Nguyen, and David KY Yau. Energytrack: Sensor-driven energy use analysis system. In *BuildSys*. ACM, 2013.

[128] Deokwoo Jung, Varun Badrinath Krishna, Ngo Quang Minh Khiem, Hoang Hai Nguyen, and David KY Yau. Energytrack: Sensor-driven energy use analysis system. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[129] Ankur Kamthe, Varick Erickson, Miguel Á. Carreira-Perpiñán, and Alberto Cerpa. Enabling building energy auditing using adapted occupancy models. In *BuildSys*, pages 31–36. ACM, 2011.

[130] Ankur Kamthe, Varick Erickson, Miguel Á Carreira-Perpiñán, and Alberto Cerpa. Enabling building energy auditing using adapted occupancy models. In *BuildSys*, pages 31–36. ACM, 2011.

[131] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372. ACM, 2011.

[132] Hyungsul Kim, Manish Marwah, Martin Arlitt, Geoff Lyon, and Jiawei Han. Unsupervised disaggregation of low frequency power measurements. In *SIAM International Conference on Data Mining*, pages 747–758, 2011.

[133] Younghun Kim, Rahul Balani, Han Zhao, and Mani B Srivastava. Granger causality analysis on ip traffic and circuit-level energy monitoring. In *BuildSys*, pages 43–48. ACM, 2010.

[134] Younghun Kim, Thomas Schmid, Mani B Srivastava, and Yan Wang. Challenges in resource monitoring for residential spaces. In *BuildSys*, pages 1–6. ACM, 2009.

[135] Wilhelm Kleiminger, Christian Beckel, Anind Dey, and Silvia Santini. Using unlabeled wi-fi scan data to discover occupancy patterns of private households. In *SenSys*, pages 47:1–47:2. ACM, 2013.

[136] Wilhelm Kleiminger, Christian Beckel, and Silvia Santini. Household occupancy monitoring using electricity meters. In *UbiComp*, pages 975–986. ACM, 2015.

[137] Wilhelm Kleiminger, Christian Beckel, Thorsten Staake, and Silvia Santini. Occupancy detection from electricity consumption data. In *BuildSys*. ACM, 2013.

[138] Wilhelm Kleiminger, Christian Beckel, Thorsten Staake, and Silvia Santini. Occupancy detection from electricity consumption data. In *BuildSys*, pages 10:1–10:8. ACM, 2013.

[139] Wilhelm Kleiminger, Silvia Santini, and Friedemann Mattern. Smart heating control with occupancy prediction: How much can one save? In *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*, pages 947–954. ACM, 2014.

[140] Wilhelm Kleiminger, Silvia Santini, and Friedemann Mattern. Smart heating control with occupancy prediction: How much can one save? In *UbiComp Adjunct*, pages 947–954. ACM, 2014.

[141] Merthan Koc, Burcu Akinci, and Mario Bergés. Comparison of linear correlation and a statistical dependency measure for inferring spatial relation of temperature sensors in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 152–155. ACM, 2014.

[142] Mario J Kofler, Christian Reinisch, and Wolfgang Kastner. A semantic representation of energy-related information in future smart homes. *Energy and Buildings*, 47:169–179, 2012.

[143] Andrew Krioukov, Stephen Dawson-Haggerty, Linda Lee, Omar Rehmane, and David Culler. A living laboratory study in personalized automated lighting controls. In *Proceedings of the third ACM workshop on embedded sensing systems for energy-efficiency in buildings*, pages 1–6. ACM, 2011.

[144] Andrew Krioukov, Gabe Fierro, Nikita Kitaev, and David Culler. Building application stack (bas). In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 72–79. ACM, 2012.

[145] Varun Badrinath Krishna, Deokwoo Jung, Ngo Quang Minh Khiem, Hoang Hai Nguyen, and David K. Y. Yau. Energytrack: Sensor-driven energy use analysis system. In *BuildSys*, pages 38:1–38:2. ACM, 2013.

[146] Kurtalj Ltd. Brightcore products, 2012.

[147] Leslie Lamport. *LaTeX User's Guide and Document Reference Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[148] Ora Lassila and Ralph R Swick. Resource description framework (rdf) model and syntax specification. 1999.

[149] C. Legat, C. Seitz, and B. Vogel-Heuser. Unified sensor data provisioning with semantic technologies. In *IEEE ETFA - Int. Conf. on Emerging Technol. and Factory Autom.*, 2011.

[150] Ting Liu, Yulin Che, Yuqi Liu, Zhanbo Xu, Yufei Duan, and Siyun Chen. A user demand and preference profiling method for residential energy management. In *ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*, pages 911–918. ACM, 2014.

[151] Lennart Ljung. System identification: Theory for the user. *PTR Prentice Hall Information and System Sciences Series*, 198, 1987.

[152] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The smart thermostat: Using occupancy sensors to save energy in homes. In *SenSys*, pages 211–224. ACM, 2010.

[153] Alan Marchiori and Qi Han. Using circuit-level power measurements in household energy management systems. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 7–12. ACM, 2009.

[154] Alan Marchiori and Qi Han. Using circuit-level power measurements in household energy management systems. In *BuildSys*, pages 7–12. ACM, 2009.

[155] Mikal Mayer, Gustavo Soares, Maxim Grechkin, Vu Le, Mark Marron, Oleksandr Polozov, R Singh, B Zorn, and S Gulwani. User interaction models for disambiguation in programming by example. UIST, 2015.

[156] George A Miller, Claudia Leacock, Randee Tengi, and Ross T Bunker. A semantic concordance. In *Proceedings of the workshop on Human Language Technology*, pages 303–308. Association for Computational Linguistics, 1993.

[157] Archan Misra and Henning Schulzrinne. Policy-driven distributed and collaborative demand response in multi-domain commercial buildings. In *1st Int. Conf. on Energy-Efficient Computing and Networking*, pages 119–122. ACM, 2010.

[158] Andrés Molina-Markham, Prashant Shenoy, Kevin Fu, Emmanuel Cecchet, and David Irwin. Private memoirs of a smart meter. In *BuildSys*, pages 61–66. ACM, 2010.

[159] Srinarayana Nagarathinam, Shiva R. Iyer, Arunchandar Vasan, Venkata Ramakrishna P., Venkatesh Sarangan, and Anand Sivasubramaniam. On the utility of occupancy sensing for managing HVAC energy in large zones. In *Proceedings of the 2015 ACM Sixth International Conference on Future Energy Systems*, e-Energy '15, pages 219–220. ACM, 2015.

[160] Srinarayana Nagarathinam, Arunchandar Vasan, Venkata Ramakrishna P, Shiva R. Iyer, Venkatesh Sarangan, and Anand Sivasubramaniam. Centralized management of HVAC energy in large multi-AHU zones. In *BuildSys*, pages 157–166. ACM, 2015.

[161] Balakrishnan Narayanaswamy, Bharathan Balaji, Rajesh Gupta, and Yuvraj Agarwal. Data driven investigation of faults in hvac systems with model, cluster and compare (mcc). In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 50–59. ACM, 2014.

[162] Balakrishnan Narayanaswamy, Bharathan Balaji, Rajesh Gupta, and Yuvraj Agarwal. Data driven investigation of faults in HVAC systems with model, cluster and compare (mcc). In *BuildSys*, pages 50–59. ACM, 2014.

[163] NEST. Nest labs. `http://www.nest.com`.

[164] Guy R Newsham and Benjamin J Birt. Building-level occupancy data to improve arima-based electricity use forecasts. In *BuildSys*, pages 13–18. ACM, 2010.

[165] Guy R. Newsham and Benjamin J. Birt. Building-level occupancy data to improve ARIMA-based electricity use forecasts. In *BuildSys*, pages 13–18. ACM, 2010.

[166] Next10. Untapped Potential of Commericial Buildings: Energy Use and Emissions, 2010.

[167] H. Ochiai, M. Ishiyama, T. Momose, N. Fujiwara, K. Ito, H. Inagaki, A. Nakagawa, and H. Esaki. Fiap: Facility information access protocol for data-centric building automation systems. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 229–234, 2011.

[168] Christopher Olston and Marc Najork. Web crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246, 2010.

[169] Jorge Ortiz. *A Platform Architecture for Sensor Data Processing and Verification in Buildings*. PhD thesis, EECS Department, University of California, Berkeley, Dec 2013.

[170] OSIsoft. Pi system. `http://www.osisoft.com/`.

[171] Kumar Padmanabh, Adi Malikarjuna, V, Sougata Sen, Siva Prasad Katru, Amrit Kumar, Sai Pawankumar C, Sunil Kumar Vuppala, and Sanjoy Paul. isense: A wireless sensor network based conference room management system. In *BuildSys*, pages 37–42. ACM, 2009.

[172] Kumar Padmanabh, Adi Malikarjuna V, Sougata Sen, Siva Prasad Katru, Amrit Kumar, Sunil Kumar Vuppala, Sanjoy Paul, et al. isense: a wireless sensor network based conf. room management system. In *BuildSys*, pages 37–42. ACM, 2009.

[173] Alessandra Parisio, Damiano Varagnolo, Daniel Risberg, Giorgio Pattarello, Marco Molinari, and Karl H Johansson. Randomized model predictive control for HVAC systems. In *BuildSys*, 2013.

[174] Shwetak Patel, Matthew Reynolds, and Gregory Abowd. Detecting human movement by differential air pressure sensing in HVAC system ductwork: An exploration in infrastructure mediated sensing. In *Pervasive*, pages 1–18. Springer-Verlag, 2008.

[175] Shwetak N. Patel, Thomas Robertson, Julie A. Kientz, Matthew S. Reynolds, and Gregory D. Abowd. At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In *UbiComp*, pages 271–288. Springer-Verlag, 2007.

[176] Daniel Perelman, Sumit Gulwani, Thomas Ball, and Dan Grossman. Type-directed completion of partial expressions. In *Proceedings of the 33rd ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '12, pages 275–286, New York, NY, USA, 2012. ACM.

[177] D. Pfisterer, K. Romer, D. Bimschas, and et al. SPITFIRE: toward a semantic web of things. *IEEE Commun. Mag.*, 49(11):40–48, 2011.

[178] J. Ploennigs, B. Hensel, H. Dibowski, and K. Kabitzsch. BASont - a modular, adaptive building automation system ontology. In *IEEE IECON*, pages 4827–4833, 2012.

[179] Joern Ploennigs, Bei Chen, Anika Schumann, and Niall Brady. Exploiting generalized additive models for diagnosing abnormal energy use in buildings. In *BuildSys*. ACM, 2013.

[180] Joern Ploennigs, Bernard Gorman, Niall Brady, and Anika Schumann. Bead-building energy asset discovery tool for automating smart building analytics: demo abstract. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 194–195. ACM, 2014.

[181] Joern Ploennigs, Burkhard Hensel, Henrik Dibowski, and Klaus Kabitzsch. Basont-a modular, adaptive building automation system ontology. In *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society*, pages 4827–4833. IEEE, 2012.

[182] Joern Ploennigs, Anika Schumann, and Freddy Lecue. Adapting semantic sensor networks for smart building diagnosis. In *ISWC - Int. Semantic Web Conf.*, 2014.

[183] Joern Ploennigs, Anika Schumann, and Freddy Lecue. Extending semantic sensor networks for automatically tackling smart building problems. In *ECAI/PAIS - Eu. Conf. on Artificial Intelligence - Prestigious Applications of Intelligent Systems*, 2014.

[184] Marco Pritoni, Arka A. Bhattacharya, David Culler, and Mark Modera. A method for discovering functional relationships between air handling units and variable-air-volume boxes from sensor data. In *BuildSys*, pages 133–136. ACM, 2015.

[185] Marco Pritoni, Arka A Bhattacharya, David Culler, and Mark Modera. Short paper: A method for discovering functional relationships between air handling units and variable-air-volume boxes from sensor data. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 133–136. ACM, 2015.

[186] Chuan Qin, Xuan Bao, Romit Roy Choudhury, and Srihari Nelakuditi. Tagsense: A smartphone-based approach to automatic image tagging. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 1–14, New York, NY, USA, 2011. ACM.

[187] Erhard Rahm and Philip A Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350, 2001.

[188] Niranjini Rajagopal, Patrick Lazik, and Anthony Rowe. Visual light landmarks for mobile devices. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 249–260. IEEE Press, 2014.

[189] Christian Reinisch, MarioJ Kofler, Felix Iglesias, and Wolfgang Kastner. ThinkHome energy efficiency in future smart homes. *EURASIP Journal on Embedded Systems*, 2011(1), 2011. Article ID 104617.

[190] Daniel Retkowitz and Monika Pienkos. Ontology-based configuration of adaptive smart homes. In *ARM - 7th Workshop on Reflective and Adaptive Middleware*, pages 11–16, 2008.

[191] Alejandro Gomez Rivera, Burcu Akinci, and Mario Berges. *Exploratory Study Towards Streamlining the Identification of Sensor Locations Within a Facility*, chapter 226, pages 1820–1827.

[192] Kurt W Roth, Detlef Westphalen, Patricia Llana, and Michael Feng. The energy impact of faults in U.S. commercial buildings. In *Int'l Refrigeration and Air Conditioning Conference*, 2004.

[193] S Roth. Open green building xml schema: A building information modeling solution for our green world, gbxml schema (5.12). 2014.

[194] Anthony Rowe, Mario Berges, and Raj Rajkumar. Contactless sensing of appliance state transitions through variations in electromagnetic fields. In *BuildSys*, pages 19–24. ACM, 2010.

[195] D.J. Russomanno, C. Kothari, and O. Thomas. Sensor ontologies: from shallow to deep models. In *SSST Southeastern Symp. on System Theory*, pages 107–112, 2005.

[196] S.L. Salas and Einar Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.

[197] Jeffrey Schein, Steven T. Bushby, Natascha S. Castro, and John M. House. A rule-based fault detection method for air handling units. *Energy and Buildings*, 38(12):1485–1492, dec 2006.

[198] Jeffrey Schein, Steven T Bushby, Natascha S Castro, and John M House. A rule-based fault detection method for air handling units. *Energy and Buildings*, 38(12):1485–1492, 2006.

[199] Anthony Schoofs, Declan T Delaney, Gregory MP O'Hare, and Antonio G Ruzzelli. COPOLAN: non-invasive occupancy profiling for preliminary assessment of hvac fixed timing strategies. In *BuildSys*, pages 25–30. ACM, 2011.

[200] Anthony Schoofs, Alex Sintoni, Antonio G Ruzzelli, and Greg MP O'Hare. Netbem: business equipment energy monitoring through network auditing. In *BuildSys*, pages 49–54. ACM, 2010.

[201] Anika Schumann, Joern Ploennigs, and Bernard Gorman. Towards automating the deployment of energy saving approaches in buildings. In *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, pages 164–167. ACM, 2014.

[202] Siemens. Siemens building management system. `http://www.buildingtechnologies.siemens.com/bt/global/en/market-specific-solutions/airports/hvac-plant-room/building-management-system/pages/building-management-system.aspx`.

[203] Rishabh Singh and Sumit Gulwani. Learning semantic string transformations from examples. *Proc. VLDB Endow.*, 5(8):740–751, April 2012.

[204] Rishabh Singh and Sumit Gulwani. Synthesizing number transformations from input-output examples. In *Proceedings of the 24th International Conference on Computer Aided Verification*, CAV'12, pages 634–651, Berlin, Heidelberg, 2012. Springer-Verlag.

[205] Rishabh Singh and Sumit Gulwani. Synthesizing number transformations from input-output examples. In *24th Int. Conf. on Computer Aided Verification*, CAV'12, pages 634–651, Berlin, Heidelberg, 2012. Springer-Verlag.

[206] Deke Smith. An introduction to building information modeling. *Journal of Building Information Modeling*, pages 12–14, November 2007.

[207] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Semi-automated modular modeling of buildings for model predictive control. In *BuildSys*, pages 99–106. ACM, 2012.

[208] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Semi-automated modular modeling of buildings for model predictive control. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 99–106. ACM, 2012.

[209] Z Cihan Taysi, M Amac Guvensan, and Tommaso Melodia. Tinyears: spying on house appliances with audio sensor nodes. In *BuildSys*, pages 31–36. ACM, 2010.

[210] Brian L. Thomas and Diane J. Cook. Carl: Activity-aware automation for energy efficiency. In *UbiComp Adjunct*, pages 939–946. ACM, 2014.

[211] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[212] M. E. Torres, M. A. Colominas, G. Schlotthauer, and P. Flandrin. A complete ensemble empirical mode decomposition with adaptive noise. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4144–4147. IEEE, May 2011.

[213] US Department of Energy. 2011 Buildings Energy Book, 2012.

[214] U.S. Department of Energy, Office of Energy Efficiency and Renewable Energy. Buildings energy data book, 2011.

[215] U.S. Environmental Protection Agency. Buildings Energy Data Book, 2010.

[216] Pablo Valiente-Rocha and Adolfo Lozano-Tello. Ontology-based expert system for home automation controlling. In *Trends in Applied Intelligent Systems*, volume 6096 of *Lecture Notes in Computer Science*, pages 661–670. 2010.

[217] Chen Wang and Martin De Groot. Managing end-user preferences in the smart grid. In *1st Int. Conf. on Energy-Efficient Computing and Networking*, pages 105–114. ACM, 2010.

[218] He Wang, Dimitrios Lymberopoulos, and Jie Liu. Local business ambience characterization through mobile audio sensing. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 293–304, New York, NY, USA, 2014. ACM.

[219] Jingjing Wang, Changsung Kang, Yi Chang, and Jiawei Han. A hierarchical dirichlet model for taxonomy expansion for search engines. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 961–970, New York, NY, USA, 2014. ACM.

[220] James Weimer, Seyed Alireza Ahmadi, José Araujo, Francesca Madia Mele, Dario Papale, Iman Shames, Henrik Sandberg, and Karl Henrik Johansson. Active actuator fault detection and diagnostics in HVAC systems. In *BuildSys*, pages 107–114. ACM, 2012.

[221] James Weimer, Seyed Alireza Ahmadi, José Araujo, Francesca Madia Mele, Dario Papale, Iman Shames, Henrik Sandberg, and Karl Henrik Johansson. Active actuator fault detection and diagnostics in HVAC systems. In *BuildSys*, pages 107–114. ACM, 2012.

[222] Thomas Weng, Bharathan Balaji, Seemanta Dutta, Rajesh Gupta, and Yuvraj Agarwal. Managing plug-loads for demand response within buildings. In *BuildSys*, pages 13–18. ACM, 2011.

[223] Thomas Weng, Bharathan Balaji, Seemanta Dutta, Rajesh Gupta, and Yuvraj Agarwal. Managing plug-loads for demand response within buildings. In *Proceedings of the Third ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 13–18. ACM, 2011.

[224] Thomas Weng, Anthony Nwokafor, and Yuvraj Agarwal. Buildingdepot 2.0: An integrated management system for building analysis and control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, BuildSys'13, pages 7:1–7:8, New York, NY, USA, 2013. ACM.

[225] Thomas Weng, Anthony Nwokafor, and Yuvraj Agarwal. Buildingdepot 2.0: An integrated management system for building analysis and control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[226] Donald J Wheeler, David S Chambers, et al. *Understanding statistical process control*. SPC press, 1992.

[227] Ian Wiese. The integration of scada and corporate it. `http://www.iinet.net.au/~ianw/integration.doc`.

[228] Ian Wiese. The integration of scada and corporate it. `http://isawwsymposium.com/wp-content/uploads/2013/08/WWAC2013_Loncar_CodeGeneratorBenefits-for-Standards_slides_1up.pdf`.

[229] Rizhen Zhang, Bharathan Balaji, Yan Zhang, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. Buildingsherlock: Fault management framework for HVAC systems: Demo abstract. In *BuildSys*, pages 202–203. ACM, 2014.