Brick

BRICK SCHEMA
BUILDING BLOCKS FOR SMART BUILDINGS

MARCH 2019

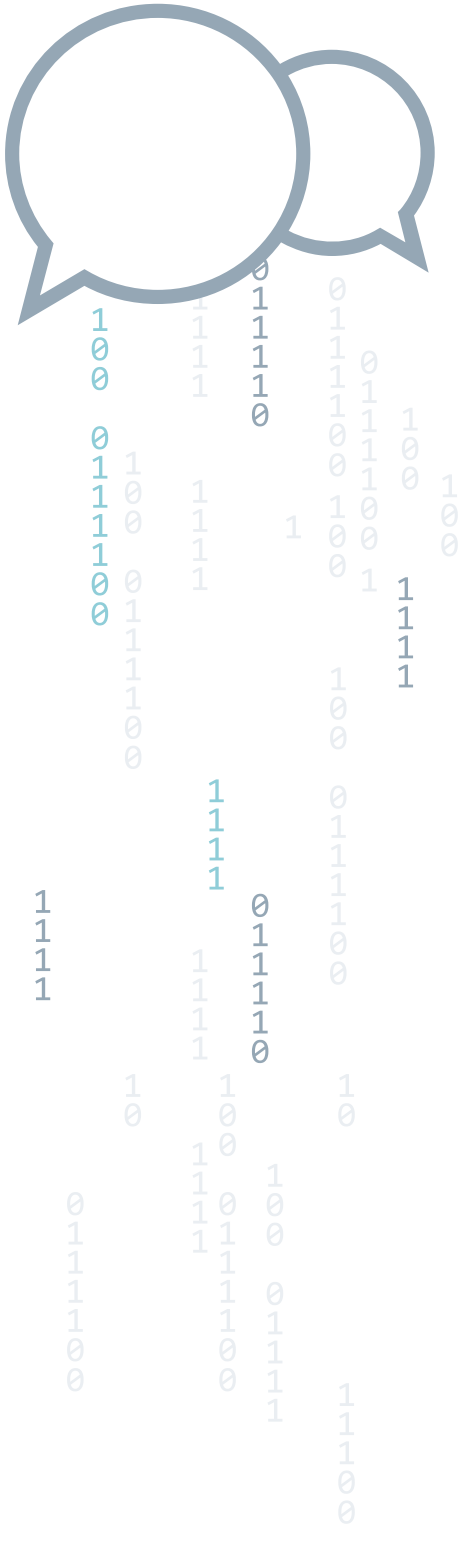
memoori

BRICKSCHEMA.ORG

Contents

1

We need to talk about data



The world is demanding more from buildings than it did a generation ago. Building owners and society at large want buildings that are cheaper to run and have a smaller environmental footprint.

Building occupants want more than just protection from the elements and basic comfort from the building - they want a building that responds to their requests and even anticipates their needs. Business owners want their buildings to be better aligned with their commercial or social mission.

We can enable these smart scenarios today. The IoT and **4th Industrial Revolution** are transforming buildings into sensor-rich environments. Our buildings can know to precool only the spaces that will host a meeting that day and make adjustments based on who physically attends, or a building occupant can use their voice assistant to override the lighting schedule for an after-hours event. We can enable spaces to be better optimized for usage and enable buildings to facilitate more productivity for its occupants.

However, because all of the building systems and sensors are siloed into different systems, each with their own model of the building data, the integrations needed to enable smart scenarios is time consuming, expensive, and not portable between buildings. Even though more and more buildings have the necessary sensors and actuators installed and the enterprise data captured, the integrations across all of the IT and OT systems to make a building smart is more often than not only partially completed, if at all.

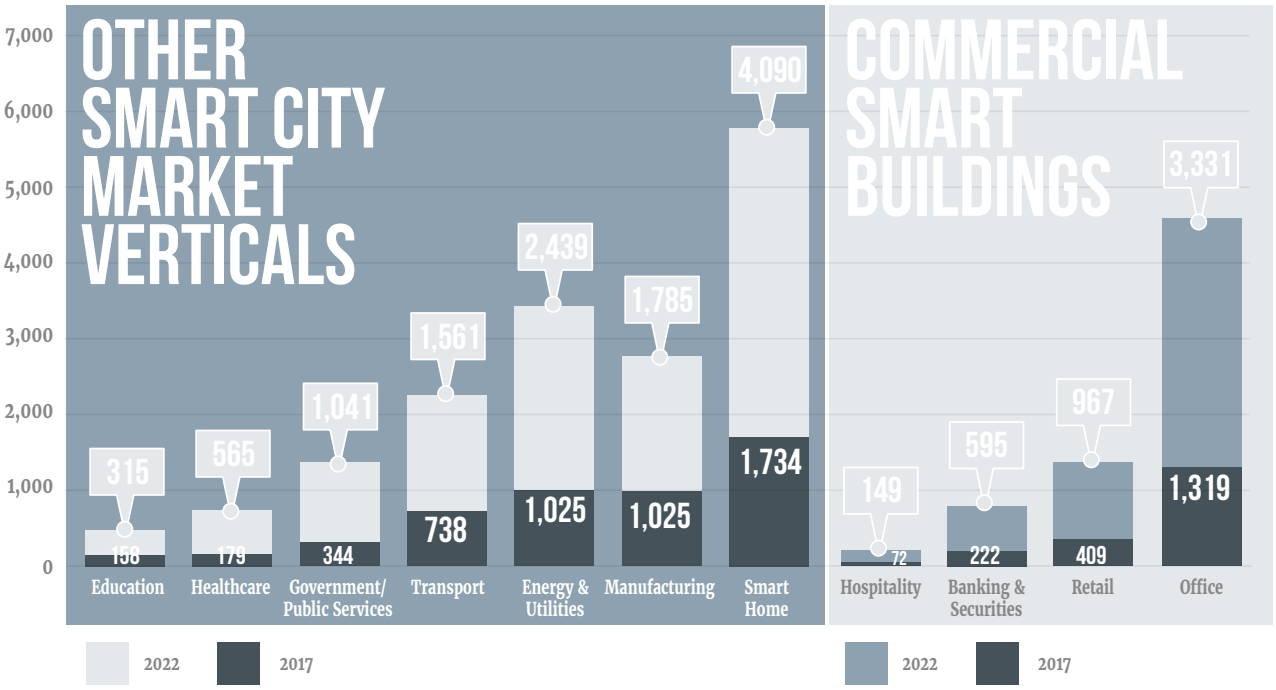
A common data schema that can represent all the critical aspects of the building - the people, places, and assets, and the relationships between them - would make integrations across building systems more reliable with less effort. Data can flow freely in and out of the building systems and enterprise IT systems and be translated to and from the common model and the native representation of each system. As the building undergoes changes through its lifecycle, the common schema makes it possible to keep a consistent view across the building systems as the underlying data sources change.

A common data schema turns integrations that were once complicated and expensive custom software development projects into data-driven applications that are portable and repeatable across buildings. A common schema is an important step towards making all buildings smart.

Different building systems, like the ones conceptualized below, are gathering masses of data which is offering unprecedented control to those who manage and operate commercial buildings; BUT if the true potential of the IoT is to be realized, data needs to be shared not just across different building systems but eventually across business enterprises and city wide platforms.

Memoori's research estimates that the global market for the Internet of Things in Buildings (BIoT) will continue to grow significantly, rising from $34.8Bn at the end of 2017 to $84.2Bn by 2022, representing a Compound Annual Growth Rate (CAGR) of 19.4%. Memoori also predicts that by 2022 there will be over 3 billion connected devices in commercial office space alone.

Smart City Related IoT Device Projections by Market Vertical
(Millions of Devices, 2017 to 2022)



Integrated data is a key enabler of better control and operations of a building, not only making it easier to maintain comfort with lower costs and resources, but integrated data is also critical to enabling better experiences and integration with the day to day business activities taking place inside the building.

The same data that makes it possible to tune the environmental controls to the personal preferences of meeting attendees helps power the AI that summarizes the outcomes of those meetings. The real-time usage data from an existing building informs the design concepts for a new construction project. Integrated data plays a crucial part of all aspects of the building life cycle, from concept to construction to occupancy to demolition and reconstruction.

WHY THE NEED FOR A SCHEMA?

Managing building data is challenging for a number of reasons but three of the most important are:

- Often building systems are commissioned in completely different ways depending on the engineers working on a project. And more often than not, they define how data from systems gets captured, stored and transmitted.
- Buildings systems generally last a long time (often between 15 and 20 years or even longer). As buildings change use and undergo renovations, data needs to be updated across multiple systems and integrations before wholesale replacement and refresh of the systems.
- Building data is not uniform. There are differences in the way data is modeled and thought about in different building systems and applications.

This last issue is stopping building data from being used more purposefully and seamlessly.

A lack of a common data model is not just preventing interoperability but also limiting development of cross-domain applications. There is research to suggest these issues are costing the industry an estimated $15 billion[1] annually. If everyone was speaking the same "language" or in this case the same "common descriptive schema," it would reduce the time and cost of developing and maintaining complex buildings systems.

A schema defines how the data is modeled and thus stored and is made available to the users and applications.

Currently, there are several data models that cover the building construction, commissioning and managing (including HVAC, Security, etc) applications. For example, IFC data model is used in the construction industry and HVAC control systems have their own proprietary data models. Integrating between multiple systems means that each application needs to understand multiple data models and account for differences in how the data is represented and the semantic meaning of the data. This means that there is a point-to-point integration from every application to every system, and any time a new application is added or a system is modified, there are many integrations to redo.

[1] Cost Analysis of InInteroperability in the US Capital Facilities Industry - https://www.nist.gov/publications/cost-analysis-inadequate-interoperability-us-capital-facilities-industry

A common data model can be the glue that enables data interoperability between the building subsystems and external data sources and uses. The existing systems are likely to continue to exist, the data model does not replace the BMS or the security system.

Instead, the common data model provides a format to extract data from the different subsystems, manage that common data in a unified way and only edit it once, and to put data back into the subsystems. The data extracted from the subsystems and combined at a building level can be pushed to emerging cloud solutions such as “digital twin” models of buildings and systems.

With the schema for a common data format, if a new system is added, the only integration required is for the system to publish and read from the common schema (if it can’t already) and all applications can use the new system. Similarly, if a new application is added, no new integrations are needed to any individual building system if the application can understand the common schema.

Why Brick Is The Solution

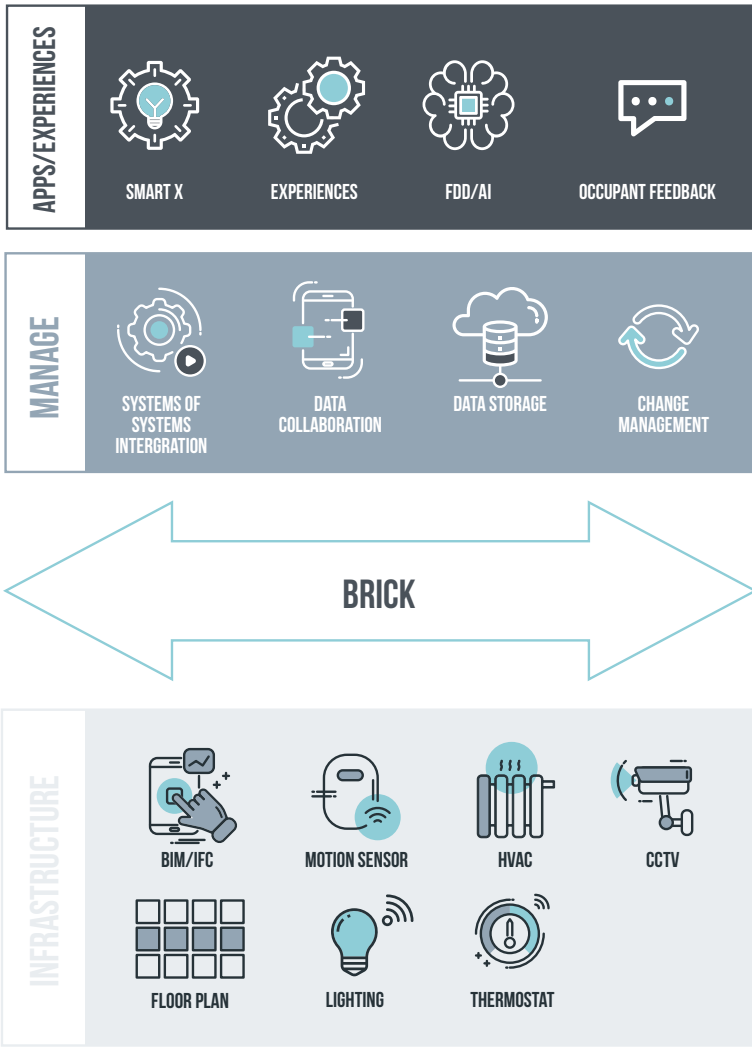
WHAT IS BRICK?

Brick is an open-source development effort to create a uniform schema (an ontology) for representing metadata in buildings

In layman's terms, Brick can describe resources in buildings, the underlying systems that control them, and form relationships among the resources; thereby allowing developers to write software applications that are not confined to a single technology or single building.

The Brick schema has been initially defined using technologies developed for the semantic web – the same technologies that help Google better understand data on the web and for scientific data repositories to work together, including the Resource Description Framework (RDF) and OWL, the Web Ontology Language. The Brick schema defines the kind of entities (defined by OWL classes) and the relationships among them (defined by OWL ObjectProperty) that are permitted in the data with the flexibility to define more concepts as needed. The entity types themselves are arranged in a hierarchy to fully capture the semantics of each entity type and the relationship between the entity types themselves.

The ontology defines a unique name for each concept applicable in the building domain, thus adopting the schema forces users to use a common "language". The open-source and extensible nature of the schema will let the participants define additional concepts as the need arises.



The diagram above shows how Brick might fit into a typical building system. Brick is not an application nor a building system nor even a new piece of software running as a server, but instead is a machine-interpretable language that describes the important aspects of a smart building.

WHY USE BRICK?

There are several reasons for using the Brick Schema;

- Brick was validated with many buildings across different parts of the world and has been used to model hundreds of buildings. As originally defined, it covers most of the HVAC and Lighting concepts that are used in any current BMS. Thus, importing data from existing BMS systems is fairly straightforward.
- It is extendable: Classes (a concept we explain in the next section) are composed of tags, so semantic reasoning about and creation of new classes is possible in a structured way.
- It is flexible: Classes are hierarchically defined so developers and building managers can express their data requirements and model their building components at different levels of abstraction, ensuring proper functionality.
- It is consistent: Brick classes guarantee maximum interoperability by preventing inconsistent usage such as different groupings of tags expressing the same concept. Furthermore, by giving a unique name to each concept there is no ambiguity in the data.
- It is useable: The Semantic-web (RDF) foundation of Brick makes for better interoperability as both the schema and the data is queryable from the standard RDF/OWL tools and technologies, e.g. RDFlib, SPARQL, Apache Jena. They can be used to support storage, querying, composition and visualization of data modeled in Brick schema. There are efforts underway to represent Brick data in JSON and to make it available through REST-like APIs, so it is easier for developers who are more familiar with those technologies to use data modeled in the Brick schema.

Johnson Controls has adopted the Brick schema as part of their Digital Vault platform. The JCI **Digital Vault**[2] provides a means of communication among various building sub-systems and devices so that their data can be used and combined more easily. The Digital Vault, powered by the schema defined in Brick, helps stakeholders collaborate with data from all parts of the built environment at all stages of the building lifecycle.

In summary, Brick is centered around the following concepts.

- Buildings and their internal spaces (e.g. Floor, Room)
- Things (e.g. HVAC equipment, security systems, AV equipment) in those spaces and how they relate to each other and the building spaces.
- The people who use the spaces
- Data sources/sinks (sensors and controllers represented as Points in HVAC) in those spaces and how they relate to the things and spaces.

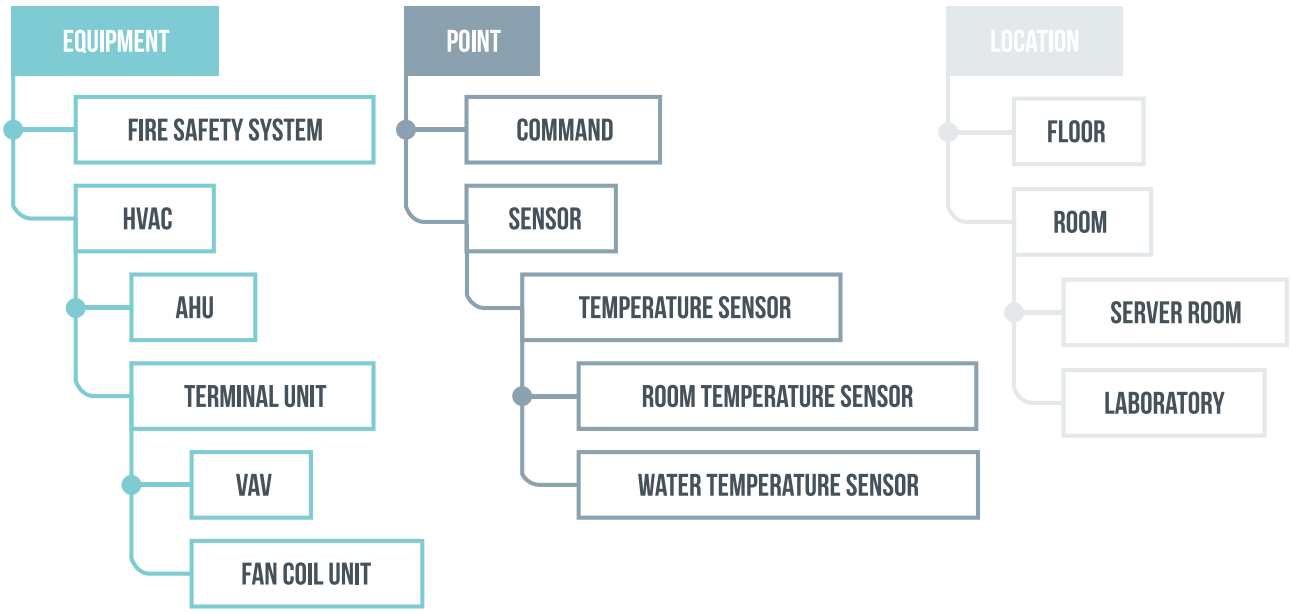
[2] https://www.johnsoncontrols.com/-/media/jci/be/united-states/campaigns/digitalvault/bts_brochure_digitalvault.pdf?la=en&hash=B6E736D09D8615CF11A998D04A7342BA24CFDB3B

BRICK ENTITIES & RELATIONSHIPS

Brick's design focuses on the metadata and data points found in real building deployments and requirements.

Each entity is an actual thing in the real world: the third floor of a building, or the thermostat on the wall of room 502, or a specific parking spot in a parking lot, and so on. Each entity is assigned to a class (similar to the classes of object-oriented programming) in the Brick model. These classes are defined under an extendable hierarchy for flexible usage.

The diagram below shows a sample of Brick's class hierarchy. For example, in this instance an Air Handling Unit (AHU) is a subclass of HVAC, which in turn is a subclass of Equipment. The entity for that represents the 3rd floor is assigned to class "Floor", which means that it is also class "Location".



Entities can have attributes beyond which class they belong to, for example, the entity for the 3rd floor might have an attribute that describes its area (say 9500), which has an attribute the describes the units that the area is measured in (perhaps square feet.)

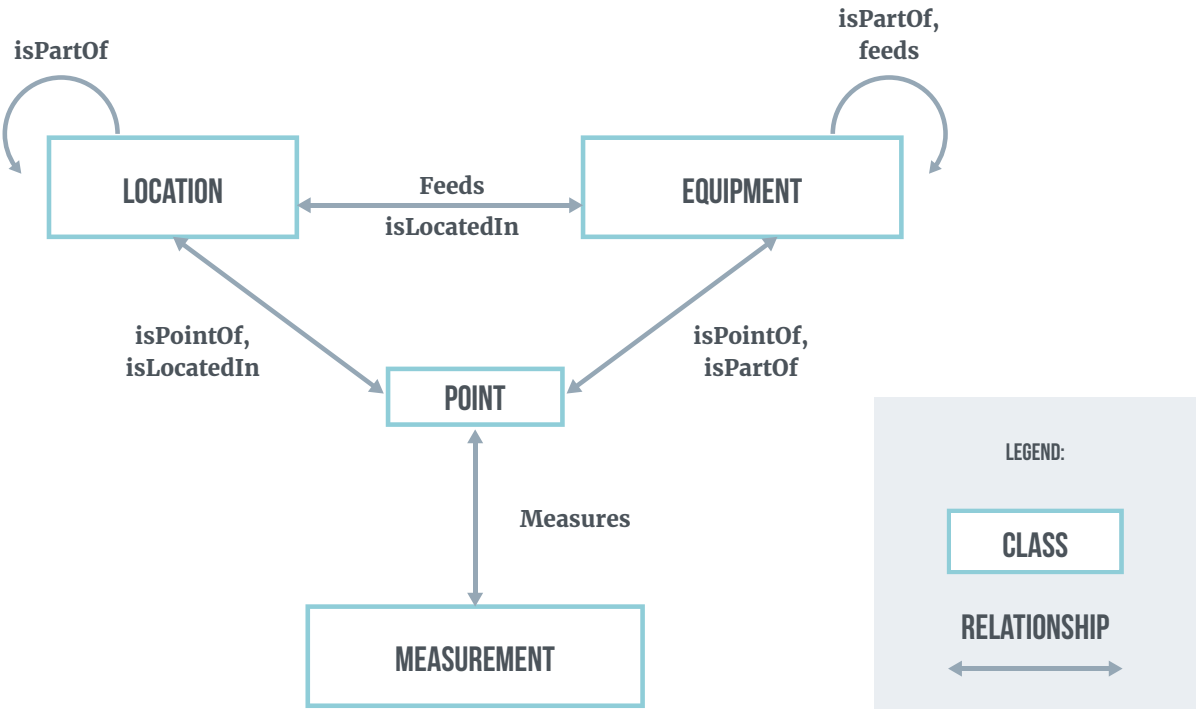
There are special attributes that establish relationships between different entities.

As an example, consider an HVAC fault detection application. It will need to know the room in which the temperature sensor is located (**location**), the corresponding temperature (**measurement**) and the status of the VAV that supplies conditioned air to the room (**equipment**).

Brick defines basic sets of relationships, which simplifies the process and makes it more intuitive. Here are 2 examples;

- The isPartOf relationship – This captures the composition of entities in the building. For example, a room isPartOf a floor.
- The feeds relationship – This captures different flows in the building, such as the flow of air from AHU to VAV.

The figure below shows some common Brick relationships between entities of class Location (and subclasses of Location, e.g. Floor or Room), class Equipment and its subclasses, and class Point and its subclasses.



Examples Of Brick IN ACTION!

Brick can play an important role in the smart building in different classes of use cases. It can enable real-time applications that cut across systems, for use cases that are focused more on immediate responses or direct requests from users.

Brick can also be used as a way to enable portable building analytics. Finally, Brick can be a translation layer to integrate data throughout the building life cycle.

Brick for real-time use cases:
In these scenarios, the Brick representation of the building is used to make direct requests from the building, and the common schema of Brick means that portable applications can be deployed at different buildings, regardless of the provider of the underlying building system.

Applications might include:

- Finding current energy usage of devices or building systems and adjusting them. The uniform Brick representation means that devices from all building systems can be queried and examined in one place, without having to do multiple integrations to each building subsystem. A load-shedding strategy that crosses into different building systems is easier to implement.
- Finding a conference room with specific requirements and real-time updates. Most conference room scheduling systems provide for some kind of requirements check, e.g. find a room that holds at least 6 people or has some type of AV equipment, but the querying power is usually limited. The full Brick model is a richer model that can be customized with a more powerful query system. Moreover, Brick enables integrations to other systems, so the Work Order system can integrate to Brick and the “find a room” query can avoid rooms that the Work Order system has marked as having non-working AV equipment.
- Lighting system override. Because Brick integrates naturally with locations, setting up lighting zones can be easy and transparent to users. The building occupants can refer to locations in the more familiar locations, such as “First floor” or “east wing”, and the Brick model of the building contains all the information necessary to find the associated lighting zones automatically.

- Smart meeting experience app. Once a meeting has been scheduled into a specific room, the smart meeting experience application can use Brick to find the particulars of the AV system and automatically establish the online meeting and start the appropriate presentations into the online meeting space and local displays, adjust the settings of the room for participant comfort (set the temperature, adjust the light shades, etc). If it appears meeting participants are traveling across campus to get to the meeting the application can send wayfinding directions to the user, and if necessary order a shuttle or other transportation. The smart meeting application is a powerful piece of software, and goes beyond Brick, but the Brick representation of the buildings made it easier to deploy the smart meeting application in an enterprise.

Brick for analytics: In these use cases, the building owner or building manager is interested in getting more insights out of their building.

The common Brick representation makes it easier to integrate all of the data into one place, and to deploy common applications to a building instead of having to rebuild the application for each building.

Some examples include:

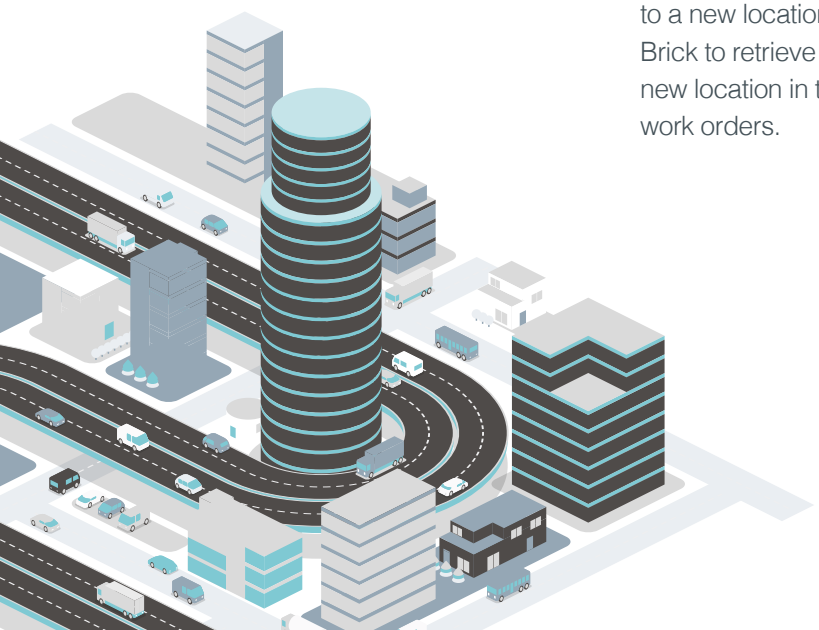
- Energy fault detection: This includes common scenarios like stuck dampers or HVAC zones where temperature or airflow setpoints are regularly exceeded or never met, or where airflow is reported correct but temperature targets are never met implying a sensing problem, and so on. Brick includes a way to identify the relevant zones and equipment and to establish the causal relationships between them.
- Predictive maintenance: There are many example algorithms to model and classify normal and abnormal behavior, but the challenge in deploying the models is integrating local data sources. Brick gives a uniform way for the algorithms to be deployed and obtain the necessary local data.
- Energy forecasting, space usage analytics: Similar to the other analytics, there are any number of example analytics but deploying them has always been a time-consuming effort of obtaining the necessary data.

- Novel local analytics: Having a common representation of data enables ad-hoc analytics that previously required far too much custom integration to obtain the data to consider. For example, imagine if a team had some measure of meeting productivity, like action items marked as resolved during a meeting. Are there environmental factors about the meeting room that might be helpful to predict if a meeting will be successful, such as "the marketing team creates 30% more pitches they rate as 'good' if the sunshades are drawn only to 30% when they meet"?

Brick for data integration and building lifecycle management: In these use cases, Brick serves as the common schema that can translate between the traditional tools and formats for data exchange.

This is similar to a data warehouse model, which provides a common schema to and from other databases and applications.

- Design to construction: New buildings typically have a BIM model created during design. That BIM can be used to create a first Brick model of the building that can then be used by intelligent commissioning tools to set up base configurations in the different building systems, from HVAC to lighting to security to wayfinding. This work is usually spread out across multiple contractors so a common data model that each contractor can read reduces overall effort.
- Regular audits and change detection: from each building subsystem, regular audits that export data back into Brick format allow for cross validation of each system. If a new entity appears in the HVAC system, for example, the building operator should at least be warned that other systems might need to be updated, or the audit software might even automatically push the changes down into the other systems.
- Brick integration into building processes: When an employee moves to a new location in the building, the HR workflow system can use Brick to retrieve all relevant details about the employee's old and new location in the building and automatically create the necessary work orders.



Information Interoperability

HOW DOES BRICK PLAY WITH BACNET & HAYSTACK?

ASHRAE (the association responsible for BACnet) announced last year their intent to integrate **Haystack** tagging and Brick data modeling concepts into the proposed ASHRAE Standard 223P for semantic tagging of building data[3]. This is a big step forward for the industry.

ASHRAE Standard 223P **"Designation and Classification of Semantic Tags for Building Data"** will provide a dictionary of semantic tags for descriptive tagging of building data including building automation and control data along with other associated systems, as well as rules for how tagsets (classes) should be constructed.

By integrating Haystack tagging and Brick data modeling concepts with the upcoming ASHRAE Standard 223P, the result will further enable interoperability on semantic information across the building industry, particularly in building automation.

THE FUTURE

The Brick schema is currently at version 1.0.3 and has a solid foundation for the HVAC aspects of the built environment and the general organization of a space. Brick is being actively extended into other domains, such as security systems, lighting, electrical, and transportation infrastructure. It is also extending into new data types and relationship types, for example, incorporating geospatial relationships and attributes expressed in common formats such as GeoJSON to enable applications such as location and positioning and asset tracking. Work is also being done to incorporate concepts from related efforts, such as bringing in the location concepts from Industry Foundation Classes (IFC), a commonly used format for Building Information Modeling (BIM). Brick is a simple yet powerful and expressive way to bring data together, and will be the foundation on which the smart environments of the future are built.

[3] https://www.ashrae.org/news/esociety/ashrae-bacnet-committee-works-with-other-organizations-on-new-standard

Resources & Brick Consortium

Brick was initiated by researchers from the academic community in 2015. Companies from across the industry are now actively getting involved with Brick.

If you are new to Brick, you can learn more on the official website and you can get started with some simple examples and tutorials

Brickschema.org ▸

Tutorials ▸

For those wishing to dive straight in, there is a link below to the main repository, and discussions / support takes place on the Brick User Forum.

Repository ▸

Brick User Forum ▸

Database ▸

Brick is open-source and is available under the BSD license.

It's easy to get involved with Brick.

WWW.MEMOORI.COM