



Title	Proposal of a Scalable Building Operating System Architecture and Data Model Toward Software-Defined Building
Author(s)	Owaki, Koichi; Yamamoto, Matsuki; Techasartikul Nattaon et al.
Citation	Proceedings - 2024 IEEE 48th Annual Computers, Software, and Applications Conference, COMPSAC 2024. 2024, p. 1586-1591
Version Type	AM
URL	https://hdl.handle.net/11094/98234
rights	© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

<https://ir.library.osaka-u.ac.jp/>

The University of Osaka

Proposal of a Scalable Building Operating System Architecture and Data Model Toward Software-Defined Building

Koichi Owaki*, Matsuki Yamamoto*, Nattaon Techasartikul*, Yuichi Ohsita*, Hideyuki Shimonishi*,
Kouki Higashioka†, Yoshihisa Toshima†, Takanori Ogura†, and Shinji Shimojo‡

*Osaka University, Japan. †Daikin Industry, Japan ‡Aomori University, Japan

koichi.owaki@ist.osaka-u.ac.jp

{yamamoto.matsuki, techa.nattaon, yuichi.ohsita, shimonishi}.cmc@osaka-u.ac.jp

{kouki.higashioka, yoshihisa.toshima, takanori.ogura}@daikin.co.jp

shinji.shimojo@aomori-u.ac.jp

Abstract—The development of Internet of Things and smart building technologies will accelerate many real-world entities (e.g., equipment, people flow, comfort levels) to be managed as digital data. This enables a Software-Defined Building in which advanced and complex applications realize a broad range of value propositions for smart buildings. To enable a scalable platform that can easily add and integrate a wide range of data and applications, the main issue addressed is the difficulty in unifying existing building equipment data models used in BACnet or others, or any related information in the building. Based on the Web of Things architecture, we propose a data model that accommodates both the information within buildings and the broad range of information expected to increase in the future. To enable the proposed data model to be efficiently managed by a building OS, we also propose a hierarchical mechanism that incorporates modules to convert the group of devices connected by different protocols into a unified data model. We develop a prototype building OS platform using Eclipse Ditto to evaluate the proposed model and mechanism. We confirm that the platform can correctly manage device information and that its performance is sufficient.

1. Introduction

With the advancement of the Internet of Things (IoT), smart buildings with complex cyber-physical systems are being developed [1]. The concept of smart buildings is expected for various value propositions, including energy efficiency, safety, comfort, etc. Smart buildings perform closed-loop control by collecting and analyzing data from various environmental sensors and then controlling the devices within the building, such as air conditioners and lighting equipment [2]. Building Operating System (building OS) has been receiving a lot of attention as the platform that facilitates these capabilities.

Building Automation and Control Networking (BACnet) [3] is a standard device connectivity protocol for intelligent building networks, defined as a standard by ASHRAE, ANSI, and ISO. With BACnet, more specifically, BACnet/IP, it is possible to collect data or control building equipment

using Internet Protocol (IP) communication. Although BACnet specifies the communication protocol for this purpose, the data model of individual devices is not yet standardized. Different naming rules or equipment groupings are applied for different buildings, and thus generalization and interoperability are issues. Since BACnet is already widely used in many buildings, proposing and migrating to a unified data model would be realistically challenging. In such a device connectivity layer, there are also other protocols; however, there is no protocol capable of handling all devices with a unified data model. In the future, we expect that more IoT devices or even robots will be connected to the building OS, and various information such as room reservations and people flows will also be handled by the building OS. As a result, the value of smart buildings will expand dramatically into many different areas; not only energy efficiency or comfort, but also new value propositions to any stakeholder including owner, tenant, visitors, and so on. The building OS will play a critical role in supporting various types of smart building applications, and thus the unified data model becomes extremely important and challenging.

To explore new value propositions in a building, all stakeholder must be able to freely and easily develop and test their applications within proper access control and resource management. Since this is as if the existing building evolves through software, we call this *Software-Defined Building*, where the building OS is expected to be an open platform to create an open ecosystem of various activities. In this paper, while there are many challenges in realizing Software-Defined Building, we start with proposing the basic architecture of the building OS and the data model to enable a wide range of value-added services. Based on the model, we will further discuss important functions such as discovering and sharing resources, access control, and security in future work.

In this study, we keep the device connectivity layer unchanged and define a wide range of building data models based on the Web of Things (WoT). Our proposed model employs a hierarchical data structure that allows multilayer abstraction so that applications can access arbitrary data at any level. In the building OS, the data structure is created

and managed by a series of conversion services based on a microservices approach, so that the data model obtained from the device connectivity layer can be converted into the next-level abstract model. We adopt Eclipse Ditto [4], the open digital twin platform, to prototype the proposed building OS architecture. Subsequently, we validate the prototype's operation and evaluate its performance.

2. Related Work

Research on smart buildings has been conducted extensively including energy management [5], [6], [7], comfort [8], access control [9], and more. This research enhances the value propositions of smart buildings, and we expect that the building OS will host many applications based on this research in real-world implementations.

Different organizations have proposed various data models for building OS. For example, Building Information Modeling (BIM) for smart buildings has been proposed [10]. In addition, Industry Foundation Classes (IFC) is a neutral and open CAD data model specification of a building [11]. As introduced above, the BACnet is a standard protocol for intelligent building networks. These models and protocols focus on the hardware aspect of the building lifecycles from design, construction, and operation. In the realm of building data ontology, Brick [12] and Real estate Core [13] have been proposed. These represent the physical composition of buildings, their relationships and their connections to real estate. When we build a broader data model, it would be important to employ these ontology models already established in the right places.

For more general data modeling to employ various IoT and ICT equipment and any information associated with the buildings, we discuss *Digital Twin* of a building. For example, a digital twin architecture for smart buildings is proposed, called BIM-IoTDI (BIM-IoT Data Integration) to enable semantic interoperability among smart building applications [14]. As a data model for the digital twin, DTDL used in Azure Digital Twin and WoT models advocated by W3C for standardization [15] are proposed. Furthermore, digital twin platforms such as Azure Digital Twin, Azure IoT Hub, AWS IoT Core, and Eclipse Ditto in open source [16] would be useful for enabling building OS.

3. Software-Defined Building

The concept of Software-Defined Building is shown in Figure 1. Various building assets, including equipment, ICT devices, environment, or any other kind of information associated with the building, are managed by the building OS so that any stakeholder of the building can pursue various value propositions with the application ecosystem.

In the following, we first present a hierarchical data model to represent a whole building as a digital twin. Then, we describe an architecture and its mechanism for recursively converting data models.

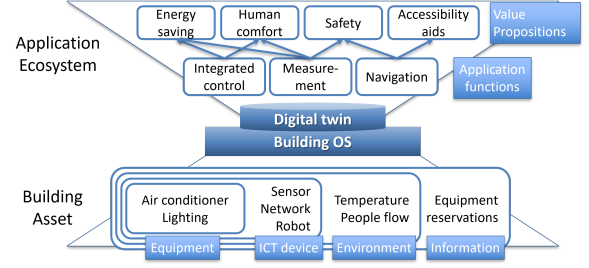


Figure 1. Software-Defined Building Concept

3.1. Digital twin data model

3.1.1. Basic data model. Figure 2 shows the overall of the proposed data model. The top of this figure illustrates the relationship between the structure of the building, starting from a building to a floor, a room, and any objects related to the room. The blue arrows show the relationship of the hierarchical structure. There is also a virtualized structure in which tenants have delegated authority over a subset of the building.

Connected to a room, or directly to a floor or a building, we define three major parent classes; *Device*, including building equipment or any ICT/IoT devices, *Mobile object*, including human and robot that move around in a room/floor/building, and *Space*, including the three-dimensional structure of temperature distribution, radio waves (WiFi/5G), or human flow. The black arrows show the inheritance relationships with the children's classes.

In this study, we define a unified data model using the WoT Thing Description (TD) so that anyone can easily develop their applications.

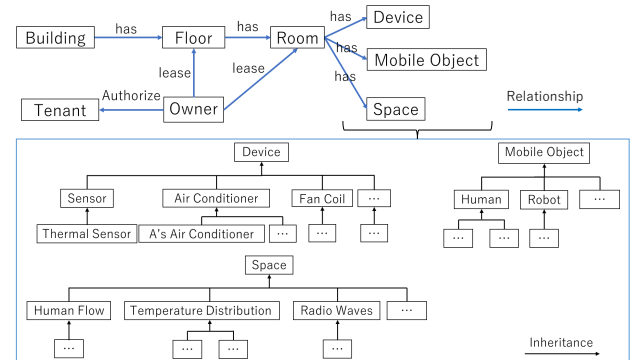


Figure 2. Proposed data model

3.1.2. Hierarchical data abstraction. Building equipment is connected to a building OS through device connectivity protocols such as BACnet, and its data is converted to the WoT TD model. However, this is not a simple conversion from equipment information to a fixed data model; rather, it involves establishing a hierarchical structure of abstractions that allows for the easy addition and integration of many types of data anywhere in the hierarchy.

For example, general high-level applications may require a highly abstracted model, which allows them to access similar equipment or information in a unified way to ensure the generality of the applications (e.g., “Sensor” and “Thermal Sensor” in Figure 2). Conversely, a specific application may require a raw equipment model to exploit vendor-specific extensions within the equipment (e.g., “Air Conditioner” and “A’s Air Conditioner” in Figure 2).

In other scenarios, an abstracted model at a lower layer, such as a BACnet device model, may contain dozens of Fan Coil Units (FCUs). In this case, it is separated into dozens of independent models. Alternatively, multiple lighting units in a room may be integrated into a single model that represents a lighting set of a room. This multilayer abstraction model allows applications to use arbitrary data at any level; an application may interact with a highly abstracted object to easily handle things, or with lower abstracted objects for more detailed control.

3.1.3. Multilevel access management and security. The hierarchical data model also allows flexible permission settings at any layer of the hierarchy. For example, it is possible to assign access rights to individual devices, as well as to grant access to groups of devices by assigning permissions to the abstracted objects post-conversion. Access rights can also be set collectively to a room, a floor, or a building. Additionally, by creating an object called a tenant that encompasses specific devices within leased spaces, it becomes easy to implement tenant-specific access control.

This mechanism is also used for security or business competition. For example, in Figure 2, general users have access to the generalized object, “Air Conditioner”, while privileged users have access to low-level objects, “A’s Air Conditioner”, to use the functions that may have hazardous consequences or vendor-specific differentiation.

3.2. Building OS Architecture

3.2.1. Overall architecture. Figure 3 shows the overall architecture of the proposed building OS. The main functions of the building OS include the following four functions:

- 1) Device connectivity: Device connectivity protocol (BACnet) and messaging protocol (MQTT/HTTP) handling
- 2) Device and data abstraction: WoT TD and abstraction functions
- 3) Data management: Search and access control for data storage
- 4) Application platform: APIs and access authentication

This overall architecture is rather general, and, especially for functions 1), 3), and 4), we can use the functions provided by digital twin platforms such as Ditto and Azure Digital Twin, as well as various protocol libraries. Therefore, in the following, we focus on the discussions on device and data abstraction.

Existing architecture proposals such as BIM-IoTDI in [14] have a straight data flow between devices and data storage, and between data storage and applications. On the other

hand, in the proposed architecture, we can create arbitrary feedback loops between data storage and the abstraction function to enable a hierarchical data structure, as explained below.

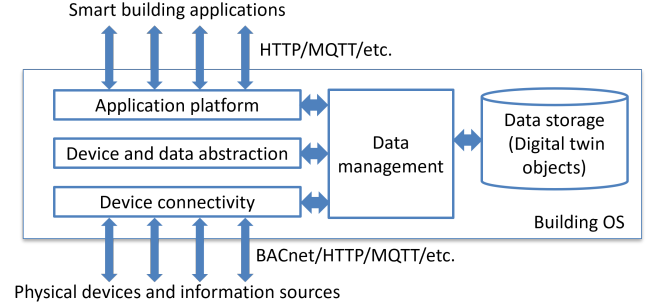


Figure 3. The proposed architecture of the Building OS

3.2.2. Device and data abstraction. Due to the hierarchical data model and its abstraction schemes varying across buildings and devices, it is impossible for a building OS to handle all such data conversions as its internal functions. Therefore, we propose to incorporate conversion services into the building OS based on a microservices approach to convert the data model obtained from BACnet into the next-level abstract model. This conversion can be cascaded up to highly abstract data model expression and sometimes involves merging or separating objects as introduced in the previous subsection.

The mechanism for the device and data abstraction function is shown in Figure 4. It consists of three features:

- Blue area: Raw device objects are created through the device connectivity function.
- Orange area: Abstracted objects are created from raw device objects by the conversion service. In addition, new abstracted objects can be recursively created from other abstracted objects.
- Green area: The conversion service converts one or more objects having its data models to one or multiple objects in other data models, possibly in both directions. This conversion is performed to create the hierarchical data structure or the application functions.

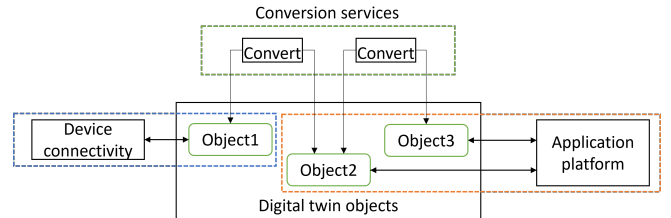


Figure 4. Mechanism for device and data abstraction

4. Prototype Building OS

4.1. Basic design

To verify the feasibility of the proposed architecture and to evaluate its performance, we designed our prototype implementation as shown in Figure 5. In this implementation, we designed that internal communication within a building OS uses MQTT messaging. This enables various components, such as conversion service and device connectivity, to be flexibly attached to the central part of the building OS, namely data management and data storage. Actually, we assume that conversion services may be provided by various stakeholders, such as building owner, operator, tenant, etc., and device connectivity may use various protocols including BACnet, so they should not be tightly bound to the central parts.

As a rapid prototyping, we utilized Eclipse Ditto for the application platform, which incorporates NGINX and HTTP API for the data storage, and Mongo DB for managing digital twin objects. The data management provides core functions, including the Mosquitto server as an MQTT broker. In device connectivity, we have implemented a BACnet-WoT converter with HTTP access and a connectivity server that bridges HTTP and MQTT. Conversion services are responsible for managing any object stored in data storage.

The specifications of the servers and devices used in our prototype are shown in Table 1. We assume that the application server could be located in a cloud data center, so a regular PC was used for this server. We assumed that other servers were located in the building, so a fanless small-box computer with a small CPU was used. The BACnet server is implemented on a separate box from the Building OS server, as it needs to be located in close proximity to the building equipment.

4.2. Sample use case

In this paper, we use Fan Coil Unit (FCU) operation as a sample use case for the building OS. We have used a SIEMENS programmable controller to implement the FCU emulator and integrate the BACnet protocol. This emulator is configured to simulate 10 FCUs, each with 8 properties, concurrently. It's designed to manage all these FCUs collectively as a BACnet device.

In this use case, we test the interconversion between objects of different abstraction levels. An object obtained through the BACnet protocol (referred to as pre-conversion object), containing 10 FCU instances, is converted to 10 independent objects (referred to as post-conversion object), containing an FCU instance. The former object may be used by an application that is in charge of the total air-conditioning of a building, whereas the latter objects may be used by individual air conditioning controllers at each floor/room. The flow of data collection from FCUs shown in Figure 6 consists of the following paths:

- 1.1) Device data are collected by the BACnet-WoT conversion module at one minute intervals.

- 1.2) Device data collected by the BACnet-WoT conversion module are fetched by the connectivity service module at one minute intervals.
- 1.3) The connectivity service module sends the device data to the data storage, and the pre-conversion object is updated.
- 1.4) Upon the update of the pre-conversion object, the conversion service receives the object and updates the post-conversion object accordingly.

The flow of device control shown in Figure 6, consists of the following paths:

- 2.1) The application sends an HTTP request to control the device and the post-conversion object is updated to reflect the control request.
- 2.2) When the post-conversion object is updated, the pre-conversion object is updated accordingly by the conversion service.
- 2.3) Updates on the pre-conversion object are sent to the connectivity service.
- 2.4) The connectivity service sends a message to the BACnet-WoT conversion module, and the message is finally sent to the BACnet device.

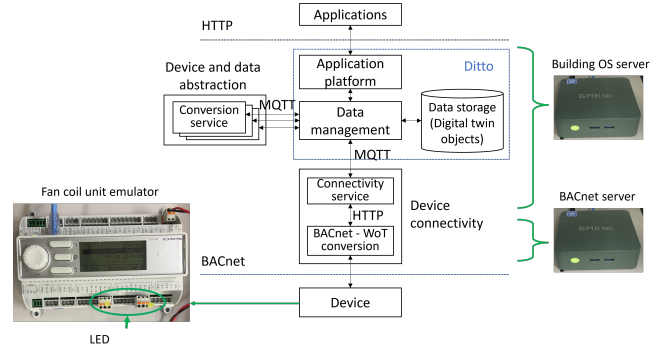


Figure 5. Prototype implementation

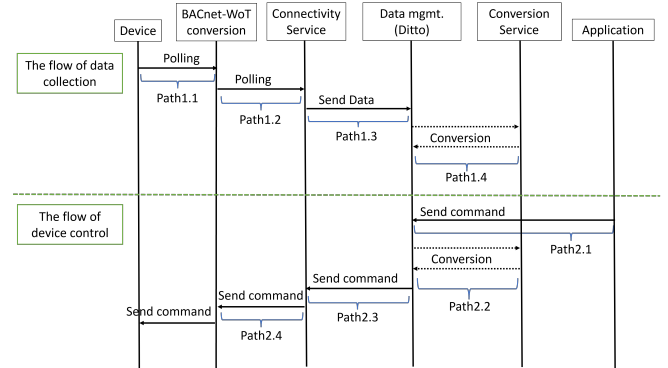


Figure 6. Flow of data collection and device control

5. Evaluation

We conducted an evaluation using the prototype presented in the previous section. The results of this evaluation

TABLE 1. SPECIFICATION OF SERVER

Names	Items: Specification
Building OS server & BACnet server	CPU: Intel 12th Alder Lake- N100 RAM: 16 GB OS: Ubuntu 22.04.3
Application server	CPU: 13th Gen Intel(R) Core(TM) i7-1360P RAM: 16 GB OS: Windows 11 Home
FCU emulator	Manufacturer: SIEMENS Product name: POL688.80/MCQ controller

are presented below.

5.1. Data Conversion Behavior

As shown in Figure 7, before the conversion, the information from ten FCUs is defined as a single object. By performing the conversion process and splitting it into ten objects, the structure becomes more aligned with real-world information and easier for the applications to handle the devices.

To verify that the conversion service is functioning correctly, we tested whether the data was properly reflected when the building OS collects data from the device and when an application manipulates objects using the HTTP API (Figure 6). Each operation was performed 60 times to confirm that the post-conversion objects and the actual device (in this case, the LEDs connected to the FCU emulator) were synchronized as designed. The results show that synchronization was achieved 100%, indicating reliability. In this way, the application can intuitively operate devices and retrieve data by interacting with the unified data model after conversion, without being aware of the data model of the original objects.

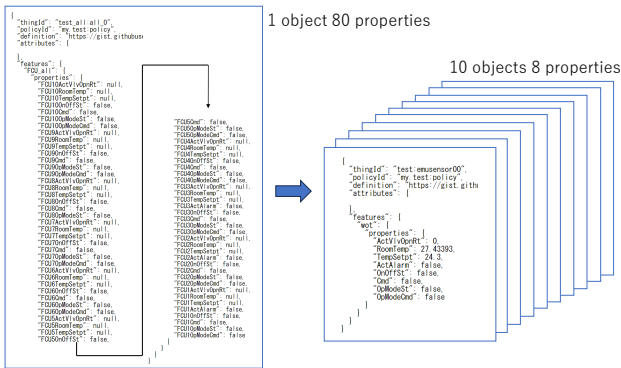


Figure 7. Before the conversion, information from ten devices is defined as a single object in JSON format. The conversion service then splits this object into ten separate parts, transforming it into a structure that is easier for humans to understand.

5.2. Data Conversion Delay

The proposed architecture incurs overhead due to object conversion. To assess the overhead numerically, the flow

of the entire data processing was divided into detailed flows, and we measured the processing time for each flow. We conducted the experiment ten times and measured the average time for each flow, as well as the total time. The results are presented in Table 2.

Regarding data collection, it is evident that while polling occurs, the processing itself is completed in a few milliseconds. Furthermore, the overhead caused by the conversion (Path1.4) is about 0.098 seconds, indicating that the overhead of the conversion is almost negligible. Regarding device control, there is a total delay of 1.67 seconds due to the implementation specification of the BACnet-WoT conversion, but the crucial conversion process (Path2.2) is completed in about 0.037 seconds, which is also negligible.

For more complex applications, conversions may be performed more than twice. As in this case, even if the conversion is performed ten times, the latency can be kept to around one second.

TABLE 2. THE DELAY OF DATA PROCESSING

Data Collection	Path1.1	Path1.2	Path1.3	Path1.4	Total
Average times (sec)	0.62	0.0075	0.013	<u>0.098</u>	0.74
Device Control	Path2.1	Path2.2	Path2.3	Path2.4	Total
Average times (sec)	0.038	<u>0.037</u>	0.14	1.45	1.67

5.3. System Scalability

When we apply the building OS to larger buildings, the number of objects can range from several hundred to several thousand. In addition, the delay of the conversion services increases according to the levels of abstraction hierarchies. Therefore, in this section, we discuss the scalability aspects of the proposed architecture.

Since it is not feasible to prepare hundreds of actual physical devices, we implemented virtual devices in the connectivity service, which generates ten objects per device; thus, we prepared up to 100 FCU emulators and a maximum of 1000 converted objects. Then, we measured the time it takes for objects to be reflected by increasing the number of managed devices.

The evaluation results are shown in Figure 8. When the number of managed devices is small, performing conversion processes does not result in a significant difference. However, as the number of devices increases, the conversion time increases accordingly. However, when we assume that the device status is monitored per minute, an update time of less than 5 seconds is considered acceptable enough to manage 1000 objects.

6. Future Works

This paper focused on considerations at the lower layers close to the devices so that applications can easily handle the abstracted devices. Moving forward, we also aim to examine higher layers closer to applications, such as integrated control of various equipment, real-time measurement

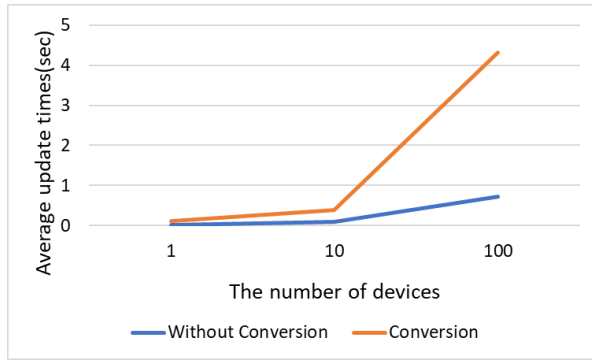


Figure 8. Comparison of update times when the number of devices increases. The blue line represents times without conversion. The orange line represents times with conversion processes.

of various environmental information, navigation of humans and robots, etc., as shown in Figure 1. In addition to building equipment, various ICT devices, any environmental information, any other related information in the building, are integrated within the building OS, as shown in the same figure. We will develop various value-added smart building applications, and their common conversion modules that will accelerate the development of various applications, to evaluate the advantage of the proposed architecture.

Furthermore, to manage large buildings or multiple buildings, we will investigate the distributed architecture of the building OS in which various components are distributed to a number of servers located in a cloud or local premises.

7. Conclusion

In this study, we proposed a hierarchical data model for building OS that responds to advances in IoT and smart building technologies. Using established standards such as BACnet and emerging frameworks such as WoT, we have designed a solution that manages building resources and seamlessly integrates various IoT devices and applications within the building environment. The feasibility and effectiveness of our proposed model and mechanisms were demonstrated through the evaluation of a prototype implementation. The architecture has been confirmed to have the ability to properly manage device information, and its performance was found to be sufficient. Furthermore, the hierarchical data model, including conversion services, enables application developers to progress more efficiently in their development efforts. Ongoing research and development in the field of smart buildings is expected to contribute to the realization of smarter, more efficient, and sustainable building environments.

References

[1] Alex H Buckman, Martin Mayfield, and Stephen BM Beck. What is a smart building? *Smart and Sustainable Built Environment*, 3(2):92–109, 2014.

[2] Daisuke Sato, Uran Oh, Kakuya Naito, Hironobu Takagi, Kris Kitani, and Chieko Asakawa. Navcog3: An evaluation of a smartphone-based blind indoor navigation assistant with semantic features in a large-scale environment. In *Proceedings of the 19th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 270–279, 2017.

[3] Steven T Bushby and H Michael Newman. Bacnet today. *ASHRAE journal*, 10:10–18, 2002.

[4] Eclipse ditto. <https://eclipse.dev/ditto/index.html>. Accessed: 2024-04-14.

[5] Daniel Minoli, Kazem Sohraby, and Benedict Occhiogrosso. Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, 4(1):269–283, 2017.

[6] Yuvraj Agarwal, Bharathan Balaji, Rajesh Gupta, Jacob Lyles, Michael Wei, and Thomas Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, page 1–6, New York, NY, USA, 2010. Association for Computing Machinery.

[7] Hooman Farzaneh, Ladan Malehmirchegini, Adrian Bejan, Taofeek Afolabi, Alphonse Mulumba, and Precious P. Daka. Artificial intelligence evolution in smart buildings for energy efficiency. *Applied Sciences*, 11(2), 2021.

[8] Pervez Hameed Shaikh, Nursyarizal Bin Mohd Nor, Perumal Nallagownden, Irraivan Elamvazuthi, and Taib Ibrahim. A review on optimized control systems for building energy and comfort management of smart sustainable buildings. *Renewable and Sustainable Energy Reviews*, 34:409–429, 2014.

[9] Nao Takizaki, Yoshiyuki Kido, Yoshiyuki Masuda, Yoshihisa Toshima, Matsuki Yamamoto, and Shinji Shimojo. Ontology-based access control framework for smart building iot devices. In *2023 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–2. IEEE, 2023.

[10] Christiana Panteli, Angeliki Kylili, and Paris A. Fokaides. Building information modelling applications in smart buildings: From design to commissioning and beyond a critical review. *Journal of Cleaner Production*, 265:121766, 2020.

[11] Ifc formats. <https://technical.buildingsmart.org/standards/ifc/ifc-formats/>. Accessed: 2024-04-14.

[12] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. Brick: Metadata schema for portable smart building applications. *Applied energy*, 226:1273–1292, 2018.

[13] Karl Hammar, Erik Oskar Wallin, Per Karlberg, and David Hälleberg. The realestatecore ontology. In *The Semantic Web—ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*, pages 130–145. Springer, 2019.

[14] Dagimawi D. Eneyew, Miriam A. M. Capretz, and Girma T. Bitsuamlak. Toward smart-building digital twins: Bim and iot data integration. *IEEE Access*, 10:130487–130506, 2022.

[15] Dominique Guinard, Vlad Mihai Trifa, and Erik Wilde. Architecting a mashable open world wide web of things. *Technical Report/ETH Zurich, Department of Computer Science*, 663, 2010.

[16] Vignesh Kamath, Jeff Morgan, and Muhammad Intizar Ali. Industrial iot and digital twins for a smart factory: An open source toolkit for application design and benchmarking. In *2020 global internet of things summit (GIoTS)*, pages 1–6. IEEE, 2020.