A Reference Architecture for Smart Building Digital Twin

Zoé Chevallier[1], Béatrice Finance[1], and Benjamin Cohen Boulakia[2]

[1] DAVID laboratory, University of Versailles-Saint-Quentin-en-Yvelines, Versailles, France
zoe.chevallier@isty.uvsq.fr beatrice.finance@uvsq.fr
[2] LINEACT laboratory, CESI, Nanterre, France
bcohen@cesi.fr

Abstract. This paper proposes a software architecture for creating and managing unambiguous descriptions of a Smart Building, allowing its Digital Twin to be deployed. These descriptions include static semantic structural description and dynamic time-series sensors data. Moreover, it is designed to be able to include data from various simulation tools, represented in the same way as sensors data. Thus, relying on these descriptions allow easy implementations of both monitoring and interaction tools that can the be interfaced with the Smart Building. Before the recent emergence of Smart Buildings, there was very few advantages in applying this concept to the field of construction and building management. Non-connected buildings don't produce real-time data, and can't be monitored nor operated from computer systems. But now that more and more buildings include IoT devices that allow them to expose data and services, the benefits of creating a Digital Twin to interact with the building, without interrogating the object itself, only by querying its Digital Twin, appears obvious. It then becomes crucial to be able to expose both a static description of the building's components (including IoT devices structural and functional description) and dynamic data that represent the evolution of the building's states over time.

Keywords: Sensor data management · Smart Building · BIM ontology · IoT · Digital Twin

1 Introduction

Buildings have many diverse components such as building blocks, electricity network, water pipes... The need to optimize a building's function, from the minimization of energy consumption while maintaining comfort, to preventive or predictive maintenance, led to considering the entire lifecycle management of the building. When considering a non-connected building, the lifecycle management can only rely on data that are maintained by human actions, which offers very few opportunities of innovating solutions. But buildings now become more and more equipped with IoT devices, leading to the appearance of buildings meant to be fully automatized known as Smart Building. Subsequently, the industry

now shows a growing interest in deploying Digital Twins of Smart Buildings. The Digital Twin of a physical system is a notion that first appeared in the early 2000s [5]. The term became widespread at the beginning of the years 2010, and it is one of the topics for which different industrial sectors show a strong interest. It is also considered by public authorities as a lever for improving governance [1]. Among the different industrial sectors that could benefit from such tool, the fields of construction and civil engineering are the ones that academic research has considered the least so far.

Several definitions has been proposed for a Digital Twin, not specifically applied to Smart Buildings. The first one comes from Grieves and Vickers who proposed the following definition [2]: "The digital twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level". Glaessgen and Stargel [3] consider a Digital Twin to be "an integrated Multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available structural models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin". More recently, Liu et al. [4] state that "The digital twin is actually a living model of the physical asset or system, which continually adapts to operational changes based on the collected online data and information, and can forecast the future of the corresponding physical counterpart". All these definitions consider a Digital Twin as a set of data that describe the state of a system, and its evolution over time. Those data may correspond to the real state of the system, but they can also be related to a hypothetical state, which could be for example the result of simulations that use data from the Digital Twin, and produce data that are integrated in return, thus allowing to predict its future state. This description includes static data, such as the physical structure of the system, but also temporal aspects, such as the evolution of this physical structure, and the data describing the dynamic state of the system (temperatures, occupancy...) When applied to Smart Buildings, the static data is the structure of the building, whether structural (building blocks, fluids and energy networks...) or functional (IoT software, space partitioning...). The temporal data is of two types: the evolution of the static description over time, and the data corresponding to the sensors and actuators that are deployed in the building.

Much work has been done on the static representation of a building. In 1987, a Building Information Modeling (BIM) has been proposed to facilitate collaborative work in both construction and evolution building's phases. Nowadays many industrial BIM software exist like ArchiCAD, Tekla or Revit [15]. In order to facilitate the interoperability between these various software and their data sources, an industrial exchange format, named IFC (Industry Foundation Classes) has been proposed [16]. Its schema representation is tree-based structure specified in the EXPRESS language [17]. The entities described in an EXPRESS file are instantiated by an IFC file. The IFC standard is continuously evolving, many versions are available. Some are obsolete and some others concern future releases not yet supported by the industry. Nowadays each BIM software can

export data in IFC standard. IFC4, the last version, is quite loosely supported by most of the industry softwares.

In the literature, we now find more and more research work that focus on BIM ontologies. Indeed it is known that ontologies facilitate the interoperability between many applications. For instance, Berstein and Young in [19], were the first to used a limited BIM ontology in order to estimate the construction cost of a Smart Building. Later, Liu and Jiang in [6], also used another BIM ontology and gave a more reliable cost estimation. More recently, Pauwels and Terkaj in [8] introduced the ifcOWL ontology which is the OWL format of IFC and became a buildingSMART standard. However the BIM industry is not yet ready to support the complexity of IoT and in particular the richness of the most prominent domain ontologies promoted by the W3C, such as the SOSA and SSN ontologies described in [9]. Dealing with many domain ontologies in the same project requires to align them as explained in [10]. This task might not be easy to achieve because existing BIM software, even if they supposedly support the IFC4 standard, do not properly export the IoT information. Some use generic IFC objects, making the alignment almost impossible.

In this paper, we show how BIM and IoT ontologies (and even additional ontologies, like BOT) can be created and aligned, then used to automatically create and update data flow from sensors and actuators to databases, providing a full Digital Twin software architecture to manage both static and dynamic description of a Smart Building. This software architecture is able to alleviate the issue with BIM silo data and provide a global view of a Smart Building. Such data are core components of the Digital Twin of the Smart Building, and can then expose and integrate data both from real IoT devices and tools such as digital simulation tools to BMS (Building Management System) or intelligent control systems.

This paper is organized as follows. In Section 2, we present the reference architecture we propose. In Section 3, its proof of concept is presented. Finally, Section 4 concludes the paper.

2 Towards a Smart Building Digital Twin

In order to build a Smart Building Digital Twin, we propose a reference architecture as depicted in Figure 1. This reference architecture aims to be independent from tools and serves as a reflection tool to discuss the global approach and to identify the different phases required for reaching our goal. These phases are formally described below. In our proof of concept, we will describe how these phases have been implemented.

(1) *Building the TripleStore for RDF data*

The first phase consists of building a TripleStore that doesn't rely on industrial tools but is based on various domain ontologies such as: (a) ifcOWL for the infrastructure, (b) SSN (Semantic Sensor Network) and SOSA (Sensor, Observation, Sample, and Actuator) for IoT description. Additional standard ontologies like BOT (Building Ontology Topology) for the topological representation of the

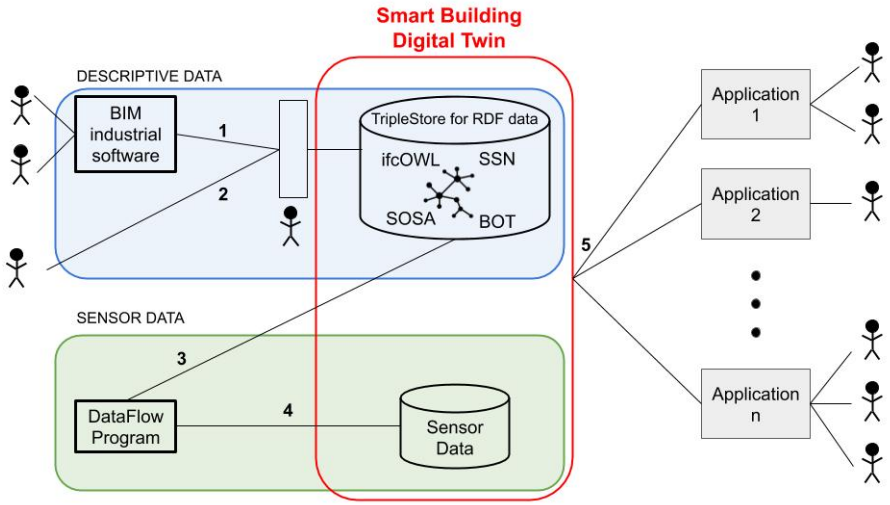

Fig. 1. The Reference Architecture

building can be used to facilitate the querying when topological data is required (such as fire simulation).

Existing BIM industrial software do not use domain ontology, they rather use either their own proprietary format, or the industrial data exchange format IFC that we presented previously. In order to follow the recent evolution towards Smart Buildings, the industry prefers to define new releases that are not easily integrated in the products and that bring a lot of inconsistencies and errors. In particular the last IFC release does not support IoT in a proper way, it mainly focuses on the physical object and does not take into account its behaviour.

As said before, Smart Buildings are complex and costly to build, they require the expertise of many professionals who have been trained to use specific industrial software. Thus these software cannot be replaced easily and it may take years before the industry replaces its IFC standard by RDF. This is why some adaptors (translators) need to be defined to fill the gap between the industry and domain ontologies.

(2) *Data Enrichment and Consistency*
In [7, 10], the authors follow the same approach about these domain ontologies, however they do not address how to align these different ontologies. We believe that the way these ontologies should be aligned should be based on name identifiers that are usually created in industrial BIM software to identify in an unique way the various objects of the building. Instead of introducing yet another naming convention, we decided to keep these names that serve as link between the various ontologies. For instance, before inserting a specific sensor into the building, one needs to manually define where it will be located in the building. Thus someone needs to query the ifcowl dataset to retreive the object on which the

sensor will be attached. In turn, when adding a sensor, one may need to add cables, make a hole in the wall to fix it, thus updating the ifcowl ontology. It is obvious that these links cannot be found automatically, but should be added manually through a human intervention.

When many different experts participate to the creation of a Smart Building Digital Twin through the means of domain ontologies, many problems can occur, such as information that is incomplete, redundant, in contradiction or imperfect. For example, each IoT object defined in SSN should be associated (linked) to its physical object in the ifcOWL ontology, and vice versa. Moreover, one can export IFC data and topology data (BOT) from existing industrial software, such as Revit, but this will introduce some redundancy since IFC already contains some topological information. However for SSN and SOSA data to be instantiated, it requires IoT experts to define sensors and actuators in a proper way.

Once the Digital Twin is created and represented with many RDF triples, it will continue to be updated. The update of the building's description, which usually takes place in industrial software, should be made by means of the SPARQL language, on the Digital Twin. For instance, when the work of many experts is integrated in a BIM industrial software a list of inconsistencies can be produced and given back to the experts, it's then up to them to correct them. In some cases, industrial software already allow this, for instance physical collision detection when merging several BIM files. Thus we believe that smart building complex integrity constraints should be defined as rules on the TripleStore for RDF data. This is still an open research problem as we do not know how difficult it will be to express them on ontologies.

Moreover a Digital Twin does not only reflect one state of a Smart Building, it should also cover the whole life cycle of the Smart Building. If supporting complex integrity constraints allows incremental updates, it does not provide versioning capability. To our knowledge, there does not really exist research tools nor industrial software that support Smart Building versioning. However it is crucial as the French law imposes a decennial building contract that needs to be kept in order to prove some wrongdoing. Simulation tools need to produce different hypothetical states (versions) of a Digital Twin in order to decide which material should be better to use. Rasmussen in [11] proposed a method to manage the model evolution.

Finally for the architecture to be complete, the TripleStore for RDF data should support concurrent access and transactions. Some TripleStores like AllegroGraph (a proprietary TripleStore) or JenaTDB have ever their transaction management system. Others, like Blazegraph do not support them.

(3) *Data Flow Program Generation*

Up to now, only one part of the Smart Building Digital Twin has been created. In order to be complete, all the information produced by sensors which reflect the state of a Smart Building over time should be integrated. Thanks to the SSN and SOSA ontologies, the infrastructure of sensors network is fully described. Thus they can be queried in SPARQL to retrieve the needed information. For instance, a first query retrieves the device name and protocol. The second uses

the protocol name to determine what protocol-specific information to retrieve. In the case of Modbus, it is the register offset corresponding to the device.

Thus, following the step presented above, it is possible to automatically generate both the schema of the sensor database by using the device name and the data flow program that is going to collect the sensors data and store them in the appropriate tables. In order to do so, one needs to choose a database for storing a huge amount of time series data coming from hundred of sensors. Once the database is chosen, one needs to define the physical schema and in particular the data distribution. Finally, many workflows should be defined. For each system of sensors there is a specific workflow associated, which is a graph. An arc between two nodes represents the data flowing between the two tasks. Those tasks include connecting to the device at regular intervals, retrieving data from the device, processing it (for example converting from a format to another), associating it with the current timestamp, and inserting it into the right table in the database.

(4) *System at run time*

Once the Data Flow Programs are deployed, data are collected in the sensors database mainly as time-series. As said earlier, tables names are linked to concepts in the IoT ontology. Thus in order to get data from a specific sensor, the ontology must be queried first to find the sensor's name, then the sensor database must be queried to retrieve the data from this sensor. These links between the two databases need to be maintained in a consistent way, as sensors can be removed or added in the Smart Building over time.

(5) *Applications*

Smart Buildings applications are developed to manage, to follow the evolution or to have a real-time view of the building. In order to enable an adequate performance of these applications, an easy access to data issued from sensors and about the building's infrastructure is required. Once built, the Digital Twin contains all the data required by advanced applications.

3 Proof of concept

The implementation we present here is a part of the Digital Twin of the real Smart Building. This building belongs to CESI [18] which is an engineer school offering engineering courses in industry and services, construction and computer sciences. CESI also conducts research in the domains of Smart Buildings and Smart Cities. This building is located on the CESI campus in Nanterre, and is 220m^2 surface, composed of four rooms and his own weather station. Various sensors are placed in each room: thermostat, luminary sensors, temperature-humidity sensors, motion sensors, window opening sensors. For windows, one single system captures and sends window states (i.e. open or closed). Finally, there is a continuous mandatory ventilation (CMV) system throughout the building. The building is composed of 90 sensors in total.

(1) *Building the TripleStore for RDF data*

The CESI Smart Building BIM, containing the whole description of the building

and including furniture, has been designed using the Autodesk Revit software [15]. Even if Autodesk claims that Revit supports IFC4, the integration and export of sensors information in IFC4 is not yet implemented. Furthermore, there is no IFC4 parameter for describing functional information about IoT devices, like URLs or device ID. Thus, IoT information had to be described separately, and linked with structural data described in IFC. As the IFC4 standard is not well-supported in Revit, the model is exported using IFC2x3.

Once the IFC file is obtained, it is translated into RDF in order to instantiate the ifcOWL2X3 ontology (available in [13]). The RDF instances are obtained using existing converters as proposed in [12]. One of those converters takes as input the IFC file and produces as output the RDF file. Note that there also exists another converter [12] that does the opposite, *i.e.* it converts RDF instances from the ontology into IFC files. This can be useful to keep the compatibility with many existing applications that only support IFC input file. In our prototype, we did not focus on the BOT ontology, but there is a Revit-BOT-exporter plugin proposed in [14], allowing to obtain RDF instances of BOT ontology about the building. The size of the IFC file of the CESI Smart Building is 70 MB. Once converted we obtain a RDF file of 702,1 MB, containing more than 9 millions RDF triples. By way of comparison, the Navitas dataset [7] that corresponds to a building of 38000m^2 surface, generated more than 20 millions RDF triples. Thus, traditional main memory ontology tools such as Jena or Protégé could not be used, instead we used the Blazegraph TripleStore.

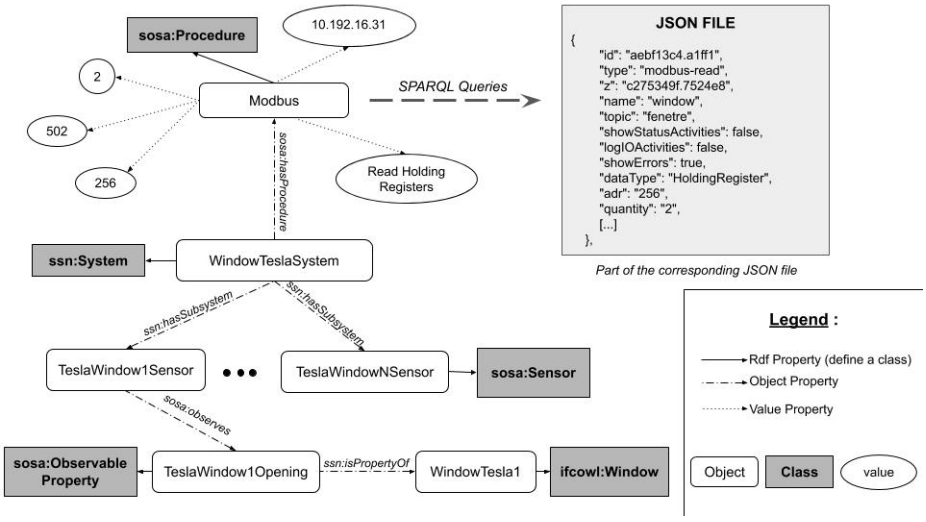

Fig. 2. Window System classes and corresponding JSON

Concerning IoT, two separate parts are identified: the structural one (such as thermostat housing and his location on the wall), which is already contained

in the IFC file and the functional one which is not and which corresponds to protocols, data access such as URLs, logical ID, register offsets, links between sensors and actuators... To add functional information into our prototype, we used the information in technical documents, such as the name of the device, its type and location, the protocol and the protocol-specific information... Those files allow to define new classes and instantiate them. Thus the IoT is fully represented into several ontologies: SOSA/SSN for the functional part and ifcOWL for its structural representation. As said earlier, we used naming conventions to link the domain ontologies, which allow us to use SPARQL queries to associate the functional part to its corresponding physical objects. In our case, the naming convention is composed of : Room name + type of the reference object + number of this object + an optional IoT object (for instance TeslaWindow3Sensor).

Let's consider the example of the windows system for a specific room which is called Tesla. Sensors are placed into each window but a unique system is collecting windows state using the Modbus protocol. The Modbus class is defined as a subclass of sosa:Procedure, with specific characteristics of this protocol. Then a WindowsTeslaSystem instance of ssn:System is created, with an instance of Modbus procedure as depicted in figure 2. We then add to all the windows in the room the sosa:Sensor class, and attached them to the global system. Instances can have multiple classes, such as windows that are both ifcowl:Window, sosa:Sensor and sosa:Actuator (as we can see in the figure 2) which allow us to link the structural and functional part. As depicted in figure 2, we built a JSON file with SPARQL queries about the Modbus protocol characteristics. This JSON file is then injected into the NodeRed server that manages data flows. This part is more detailed later. Note that in this current prototype we do not yet support complex integrity constraints, nor versioning, as these are really dependant upon the BlazeGraph TripleStore functionalities that we are using.

(2) *Generating the Data Flow Program*

One major contribution of our work is to automatize as much as possible the sensors data acquisition. Indeed, a Smart Building is complex and can contain hundred heterogeneous sensors and actuators, making the work of the manager of the building cumbersome and repetitive if some changes occur such as adding or removing sensors from the building. The idea we defend here consists in using the description of the IoT objects found in our TripleStore to generate the schema of the sensor database, as well as to write the different data flow programs for acquiring all sensors time series data to be stored in the database.

In our prototype we decided to use the NoSQL CASSANDRA database that can deal with large amount of data and also for its horizontal scalability. Any time series database can be used, it must be adapted to the data and possible treatments of the use case. Data issued from a same controller is gathered in the same table. For instance, the various sensors data of the weather station managed by one controller are all stored in one table. In the same manner, all lights sensors of a single room are in another table. The partitioning is made with a "year-month" key. For our scenario, if the 90 sensors sense every minute it will produce approximately more than 47 millions tuples per year. This data is

```
prefix ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
prefix sosa: <http://www.w3.org/ns/sosa>
prefix ssn: <http://www.w3.org/ns/ssn>

SELECT ?window ?procedure ?registerNumber ?ipAdress ?automatonPort
WHERE {
  ?window rdf:type ifcowl:IfcWindow.
  ?system ssn:hasSubSystem ?window;
          sosa:hasProcedure ?procedure.
}
```

Fig. 3. Find the protocol(s) of the windows

```
prefix ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
prefix sosa: <http://www.w3.org/ns/sosa>
prefix ssn: <http://www.w3.org/ns/ssn>

SELECT ?window ?procedure ?registerNumber ?ipAdress ?automatonPort
WHERE {
  ?window rdf:type ifcowl:IfcWindow,
          :hasRegisterNumber ?registerNumber.
  ?system ssn:hasSubSystem ?window;
          sosa:hasProcedure ?procedure.
  ?procedure :hasIpAdress ?ipAdress,
             :hasAutomatonPort ?automatonPort.
}
```

Fig. 4. Windows data needed for Modbus Protocol

only stored in the CASSANDRA database and not in the triple store to alleviate the TripleStore for RDF data (ref Figure 1).

Once the database schema is created, by querying the TripleStore for RDF data and an IoT middleware, we are able to collect the real data and store them in the database. In our prototype we use Node-RED. Node-RED is a programming tool that allows the creation of workflows for connecting together hardware devices, APIs and online services. Its advantages include many IoT protocol connectors already implemented and usable, a readable flow description file format based on JSON and a WebServices API that allows modifications of the deployed workflows.

Let's illustrate a little bit more how this works by considering the example of a workflow that detects a window opening. The connection to the window uses the IoT protocol name used by the window sensor, and some protocol-specific data needed for the connection to this specific window (such as the sensor ID, or a device register offset) obtained by querying the TripleStore for RDF data in SPARQL (Figure 3). In this example, the ModbusTCP/IP protocol is identified. The Modbus protocol needs the IP and port address of the server, and a the register number specific to the window (since in the system that is installed, all window states are retrieved in a unique frame), thus another query is issued to retrieve the needed data as shown in the query (Figure 4). Note that this second query is specific to the ModbusTCP/IP protocol, in the case of another protocol it would be different. The second node is responsible for recurrent device interrogation. The third node retrieves the current time and date for inserting in the database. The three following nodes are used for data processing. The last one generates a CQL query, used for inserting data into the CASSANDRA

database. In this query, the table in which the data is inserted is determined using the window name that is obtained by querying our TripleStore.

(3) *Simulations*

A Smart Building Digital Twin aims to analyze and predict the building's behaviour during its life cycle with some multi-domain simulations. Many applications, such as predictive maintenance for example, needs the structural description of the building, that contains the list of each object subject to maintenance, but also need to know the use time of each of these objects, in order to predict their failure probability over time. Another example is the Intelligent Control System, that in order to manage simultaneously and consistently thermal and light comfort, must have access to the list of all windows and lamps, but also to the current temperature and natural light. In case of low light, the system can determine whether it's better to open blinds or to turn on the lights. In our prototype, we focused on the environmental part which is a valid application of a multi-domain simulation: structural description, dynamic time-series sensors and meteorological data are necessary. Many other simulation applications can be added to have a full representation, they will all have the same principle and need the same data.

For predicting the evolution of temperature of the buildings rooms we used OpenModelica software and a simulator changing actuators state (like radiator heating or not) as shown in figure 5. OpenModelica tool automatically builds a multi-domain digital simulation model, using the structural description of a building (including its geometry and the building material), and the historical thermal and meteorological data (acquired from the building's sensors). We built a Modelica model by querying ifcOWL instances to obtain structural characteristics of a room (as walls location, thickness, material used, ...). We detected some incompatibilities (spaces not closed, ...) in our first model then we built a list of the constraints that must be respected to have a coherent BIM representation accepted by OpenModelica software.

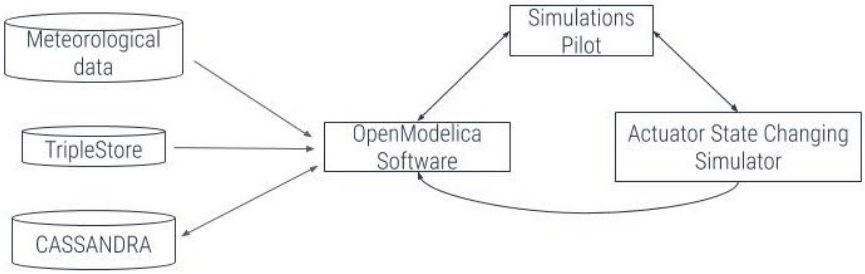

Fig. 5. Simulation

To start a simulation, several input are needed: meteorological data which are historical or predictive data not included in our prototype but more issued

from different sources in one part. In a second part, a subdomain of the ifcowl which are relevant for simulation (rooms, walls, windows, ...), and IoT data. More information are necessary for simulation, such as operating mode of the heat pump or heating trip temperature which are not include in the dataset because they are not defined in SOSA/SSN ontology. They could be added in extensions of those ontologies for example. Finally, during simulation not all sensors are analyzed, that is why some data are replaced by real ones which are stored into the CASSANDRA database (explained in section 3). Then the simulation pilot starts the simulation. Depending on the temperature evolution of the room, actuators states change due to thermostat rule by the dedicated simulator. This change is also modified in the state of actuator in the input of OpenModelica as shown in figure 5. Resulting data are stored into new tables in the Cassandra Database. Those tables store some parameters of the simulation configuration (such as representation and input datasets).

4 Conclusion

In this paper, we have presented a reference architecture for creating and managing a Digital Twin that provides a global view of a Smart Building (i.e. structural/static and temporal/dynamic data). Thus it will facilitate the development of various applications that were not possible before due to the numerous silo data. By offering a TripleStore with its multi-domain ontologies promoted by the W3C, and its associated sensor database both built in good intelligence and consistency, it is now possible to tackle more advanced functionalities and applications.

We proved with our prototype based on a real Smart Building that it is a very promising way to follow. It enables us to develop a multi-domain digital simulation tool that is going to be used to train and educate future engineers in the domain of Smart Building. Although for our approach to be complete and fully effective, some perspective issues such as complex integrity constraints and versioning, should be addressed by the research community. We also hope that the industry will benefit from this work and will make some efforts in that direction.

5 Acknowledgments

We would like to thank Yann Richou and Houssine Idoudi for their work in this project. This work is part of the LaVI&Co project (*Laboratoires Virtuels pour l'Industrie et la Construction* – Virtual laboratories for industry and construction), 40% funded by the Ile-de-France FEDER Regional Operational Program *Strengthening of new digital uses and content in the fields of e-education and e-health*. Its aim is to validate and develop two Digital Twin models for educational use, one in the construction field and based on the Smart Building mentioned above, and one in the industry field, based on an autonomous metal additive manufacturing unit.

References

1. Malhotra, Charru : Role of Digital Technologies in Governance. https://iipa.org.in/upload/Role%20of%20Digital%20Technologies%20in%20Governance.pdf
2. Grieves, Michael: Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management. Space Coast Press (2011)
3. Glaessgen, Edward & Stargel, David: The digital twin paradigm for future NASA and U.S. air force vehicles. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 1818 (2012).
4. Liu, Z., Meyendorf, N., Mrad, N.: The role of data fusion in predictive maintenance using digital twin. Annual Review Of Progress In Quantitative Nondestructive Evaluation (2018)
5. Aidan Fuller and Zx Fan and Charles Day: Digital Twin: Enabling Technology, Challenges and Open Research. ArXiv abs/1911.01276 (2019)
6. Liu, Xin and Jiang, Shaohua : Ontology-Based Representation and Reasoning in Building Construction Cost Estimation in China.Future Internet(8), 39 (2016)
7. Rasmussen, Mads Holten and Frausing, Aaskov and Hviid, Christian Anker and Karlshøj, Jan : Integrating Building Information Modeling and Sensor Observations using Semantic Web. In: 9th International Semantic Sensor Networks Workshop, pp. 1–2. Publisher, Location (2018)
8. Pauwels, Pieter and Terkaj, Walter : EXPRESS to OWL for contruction industry: Towards a recommendable and usable ifcOWL ontology. Automation in Construction(63), 100–133 (2016)
9. W3C - Semantic Sensor Network Ontology, https://www.w3.org/TR/vocab-ssn/. Last accessed 2017
10. Terkaj, Walter and Schneider, Georg Ferdinand and Pauwels, Pieter : Reusing Domain Ontologies in Linked Building Data: the Case of Building Automation and Control. In: Proceedings of the 8th International Workshop on Formal Ontologies Meet Industry,
11. Rasmussen, Mads Holten and Lefrançois, Maxime and Pauwels, Pieter and Anker-Hviida, Christian and Karlshøja, Jan: Managing interrelated project information in AEC Knowledge Graphs. Automation in Construction(108), (2019)
12. IfcSTEP to IfcOWL converters, https://github.com/BenzclyZhang/IfcSTEP-to-IfcOWL-converters
13. Pieter Pauwels, Walter Terkaj - IfcOWL 2X3 TC1 ontology, https://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL/index.html
14. Mads Holten - Revit BOT exporter, https://github.com/MadsHolten/revit-bot-exporter
15. Autodesk - Revit Software, https://www.autodesk.fr/products/revit/overview
16. ISO 16739-1:2018 - Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries, https://www.iso.org/standard/70303.html
17. ISO 10303-11:2004 - The EXPRESS language reference manual, https://www.iso.org/standard/38047.html
18. CESI Smart Building, https://recherche.cesi.fr/ville-futur-batiment-futur/
19. Norbert W Young, Stephen A Jones, Harvey M Bernstein, and John E Gudgel: The business value of bim-getting building information modeling to the bottom line. Bedford, MA : McGraw-Hill Construction(51), (2009)