A Semantic Based Multi-Platform IoT Integration Approach from Sensors to Chatbots

Charbel El Kaed*, André Ponnouradjane*, Dhaval Shah†
*Digital Services Platform-Innovation, †Buildings BU
Schneider Electric
Boston One Campus
800 Federal Street, Andover, MA, USA
Email: first.last@schneider-electric.com

***Abstract*—The Internet-of-Things paradigm has brought exciting opportunities to increase productivity and efficiency but also to enhance customer experience. Such things are expected to connect and exchange information enabling applications to propose a value added services. However, due to their heterogeneous nature, things communicate in different semantics making integration a laborious and costly to achieve. We propose in this paper an end to end semantic-based integration approach from sensors to chatbot. Our approach provides integration between two different IoT platforms and offers a coherent view of a building facility.**

Keywords—Smart Data, IoT, Ontology, Chatbot

I. Introduction

The Industrial Internet of Things paradigm is expected to tackle many challenges in the industry but also to bring new exciting opportunities. Those challenges and new opportunities involve interconnecting massive things in order to provide value-added services in various domains and segments such as energy, transportation, healthcare, retail, and many others. Promises of the IIoT paradigm involve increasing productivity, achieving efficiency, and enhancing customer experience.

Things ranging from sensors, devices, gateways, and systems deployed in a wide variety of segments and domains are expected to be interconnected in order to achieve and apply one or more strategies such as energy optimization, maintenance operation or safety. In addition to their inter-connectivity, those things are also expected to push their sensed and acquired data to a cloud platform where a massive storage and computational power are available to store and analyze the collected data.

Things are designed for various purposes in the industry, such diversity is reflected in their computational, storage, and communication capabilities. In addition, due to their various purposes, these things communicate data in different formats, syntax, and semantics according either to home grown data models or industry standards. Such heterogeneity of data representation is one of the most challenging problems in the IoT domain, it prevents interoperability and inter-cooperation between things but also add additional burden on data analytics and applications to understand the semantics of the data [1].

Semantic technology, is one of the most promising fields in the knowledge representation domain, expected to enable interoperability in the IoT. The World Wide Web Consortium (W3C) defines a set of standards , such as RDF, OWL and SPARQL [2], [3], [4], to represent semantics and query linked data, offering an ideal ecosystem and opportunity to tackle the heterogeneity challenge in the IoT.

In this paper, we propose to rely on semantic technology to abstract and represent things and their context in an integrated knowledge approach. This approach extends our previous work [5] by relying on our model driven approach OLGA [6] and our federated query engine FOrTÉ [7]. We also propose a chatbot application to interact with our infrastructure.

The rest of the paper is organized as follows: Section II depicts our industrial context and our requirements. Our proposal is detailed in section III along with the architecture. We cover the implementation along with results in section IV. The related work is reviewed in V, then we conclude in section VI.

II. Industrial Context: Workplace Environment

This work is applied on two of our facilities : our North American headquarter and Swedish sites where we partnered with two 3rd parties: a floor mapping service and an indoor sensing service. These two cloud services provide the following types of data:

1) Our indoor sensing partner deployed various sensors in our workplace environment. Such sensors were attached to desks, cubicles, and offices measuring various information such as temperature, humidity, occupancy, and CO2.

2) Sensor location: each sensor is associated with a textual representation manually assigned during the commissioning phase for a particular zone or area such as *2C-WS080* indicating the second floor, sector C and cubicle number 80.

3) The floor mapping partner is specialized in providing a digital plan of the building which includes detailed information for every floor which depicts desks, meeting rooms, offices, such as their surface and their occupation's capacity. In addition, items of interest are also represented such as coffee machines, external cardiac massage machines, or printers.

Such data is provided by two different partners through two different infrastructures which makes the information still trapped in silos. For example, to measure the CO2 per square foot, one needs to combine the CO2 measurement from the Indoor Sensing partner in a given area with the square footage information provided by our floor mapping partner.

The combined data sources provides new capabilities to facility managers and employees in fulfilling the space and

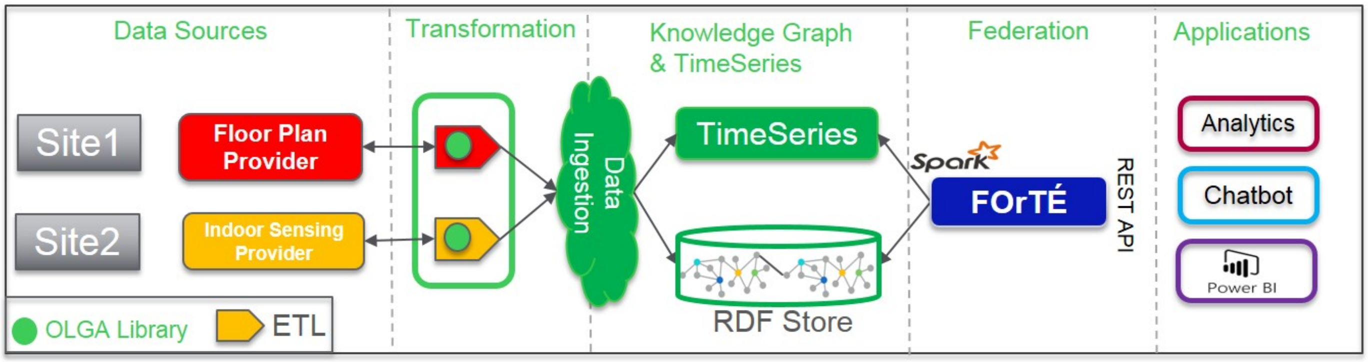


Fig. 1. Overall Architecture

occupancy management in addition to comfort and well-being in the workplace environment. We enumerate below a non exhaustive list of queries collected from interviewed facility managers $[Q1..Q5]$ and on site employees $[Q6..Q8]$. Once fulfilled, such queries will provide facility managers with better insights enabling them to take well informed decisions and provide employees with a comfortable workplace environment.

Q1: How many workstations are there in this building?
Q2: List out the average temperature of all the rooms recorded today?
Q3: List out the max humidity of all the rooms?
Q4: List out the minimum temperature of all the rooms recorded today?
Q5: What is the average temperature by day of the 1st floor?
Q6: Which is the largest meeting room on the 2nd floor?
Q7: List out the meeting rooms that are over 100SqFt on 1st Floor?
Q8: Where is the conference room ”Atomic”?

In order to unlock the potential of data and answer such queries, we detail next our integrated approach where we merge and combine data by relying on a semantic representation. In this work, we extend our previous work [5] by first proposing an ontology model to capture the data representation of our partners instead of our own internal systems. Then, we rely on our model driven approach [6] to instantiate our ontology model and our federation engine for data retrieval. Finally, we propose a chatbot to enable interaction with the two buildings where this approach is deployed.

III. Proposal & Overall Architecture

In this section, we overview our proposal and architecture shown in Fig 1. Then, we detail each part of the architecture and its role in the data flow ranging from transformation, storage, to query federation, and its usage in applications.

A. Ontology Model of Multi Data Sources

In order to enable data integration from different sources, we first model a semantic model which integrates and represents the concepts from the various data sources and the relations between, as shown partially in Fig 2. For example, the concept of `Comfort` sensor is a subclass of `Sensor` which

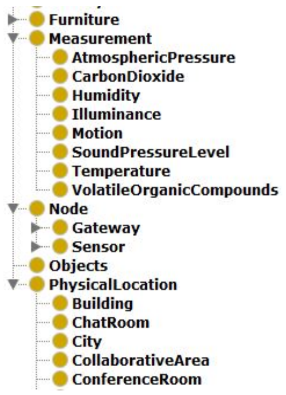

- Furniture
- Measurement
  - AtmosphericPressure
  - CarbonDioxide
  - Humidity
  - Illuminance
  - Motion
  - SoundPressureLevel
  - Temperature
  - VolatileOrganicCompounds
- Node
  - Gateway
  - Sensor
- Objects
- PhysicalLocation
  - Building
  - ChatRoom
  - City
  - CollaborativeArea
  - ConferenceRoom

Fig. 2. Part of the Ontology Model in Protégé [8]

have a measurement and a location. A location can be one of the `PhysicalLocation`.

In addition, the model allows to represent possible relations and constraint, for example a `Building` concept is allowed to have several `Floor` concepts. However, a `Floor` can only be `locatedIn` exactly one `Building`. Furthermore, an inference engine [9] part of an ontology store, allows to query additional data non explicitly entered. For example, consider a `sensor1` is located in Room `202A` which is located in the `West2` area, on the floor `L2` of `buildingA`, since `locatedIn` is declared as transitive, the inference engine will deduce that the `sensor1` is actually located in `buildingA`.

B. Data Transformation with a Model Driven Approach

Once the ontology model is represented, data transformation can take place. In our previous work [6], we proposed OLGA, an Ontology Library GenerAtor which follows a model driven code generation approach. OLGA, as shown in Fig. 3, takes as input an ontology model and generates a library which is conform to the model. The aim of the generated code [10] is to abstract the low level details for developers and allow them to easily develop ontology based applications without any required knowledge in the RDF [2] and OWL [3] languages.

The generated library is conform to the ontology model reflecting the same concepts, possible relations, and constraints. It will be used by a developer when implementing the ETL (Extract Transform Load) process which consists in retrieving the data from our 3rd party APIs and applying a translation using the generated library. The output of the ETL process will

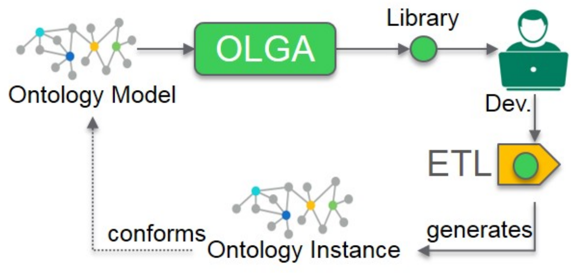
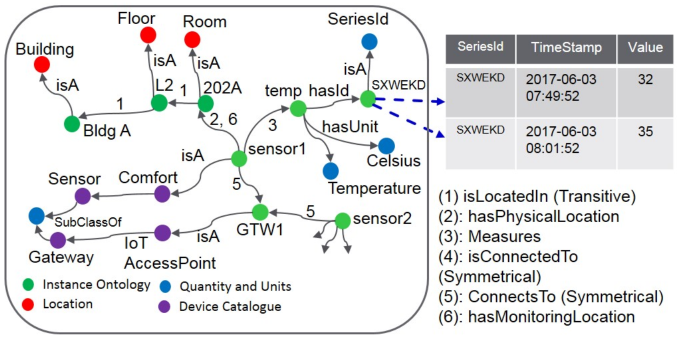


Fig. 3. Transformation based on a Model Driven Approach

generate an ontology instance file conform to the ontology model. The generated file is sent to our cloud infrastructure through the data ingestion layer as shown in Fig 1. The transformation layer handles the ontology types enumerated in section II by relying on two ETL processors. The first, handles the floor plan provider data, while the second handles the topology of sensors in addition to their timeseries data. Each ETL process generates an ontology instance, which are linked by reference, the location identifier.

C. Data Storage

Our third layer, consists of storing the generated ontology instances (topology), model, and timeseries data. As shown in Fig 1, the data is provided by the ingestion layer which routes and stores the generated ontology instances and timeseries. The



SeriesId	TimeStamp	Value
SXWEKD	2017-06-03 07:49:52	32
SXWEKD	2017-06-03 08:01:52	35

Fig. 4. Ontology Instance and Model Example

content of the knowledge graph and the timeseries stores is depicted in Fig 4. The left part of Fig 4 shows a specific ontology instance along with the ontology concepts such as the location, device catalogue, and the quantities and units. The right part of Fig 4 overviews a table with timeseries entries collected from the sensors according to sampling interval defined or tuned by the our indoor sensing partner. The two representations are linked through the *SeriesId*. The ontology is expected to change slowly based on new types of sensing added or upon a sensor replacement. However, the timeseries data is expected to grow with time to reach billions of entries. Due to the difference in the data velocity, two different stores are used to persist the data, as detailed in our previous work [7]. The ontology store allows to represent a very flexible graph based structure, in addition to the inference engine which allows to deduce and extract additional information. The timeseries store handles a high velocity and volume of data.

D. Query Federation with FOrTÉ

The data architecture relies on two separate stores, therefore, a federation engine is needed to retrieve the data in a scalable manner with an adequate infrastructure to avoid bottleneck on applications. Therefore, we rely on FOrTÉ [7], our Federated Ontology and Timeseries Query Engine which takes a federated standard SPARQL [4] query and handles a query targeting one or both of the stores. A federated query contains an topology and a timeseries query. FOrTÉ queries and federates the results in a standard SPARQL format [11].

Consider for example *Q5* presented in section II, it consists of retrieving the average temperature by day of the 1st floor. When *Q5*, expressed in a SPARQL federated query, is received by FOrTÉ, the ontology part is extracted and submitted to the Rdf store. The result contains all temperature sensors of the 1st floor and their timeseries identifiers. FOrTÉ extracts such ids and generates a timeseries query by injecting the ids in addition to average operation by day on each timeseries entry. FOrTÉ then aggregates and returns the results in a standard SPARQL response format. FOrTÉ's architecture is detailed in our previous work [7]. From the list of queries introduced in section II, $[Q2, Q3, Q4, Q5]$ requires a federation between the knowledge graph and the timeseries stores, while queries $[Q1, Q6, Q7, Q8]$ will only target the knowledge graph.

E. Butlor: A Chatbot Application

Applications consume the generated data and create value since they represent the link between the customer and/or user and the IoT technical infrastructure, as shown in Fig. 1. Having a data centric architecture based on a semantic representation provides a lot of flexibility since the ontology model allows to capture the overall knowledge. It is then up to the applications to retrieve parts of such knowledge and transform it into an added value to the end user or customer.

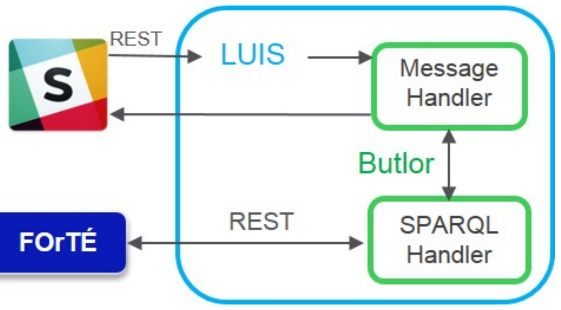


Fig. 5. Butlor Architecture

In our previous work [5], we connected a visualization tool such as Power BI to our semantic repository. We also demonstrated in our recent work [12], an analytics energy-based shaving application integrated with FOrTÉ.

In this work, we propose to integrate our chatbot *Butlor* specifically targeting the workplace environment. *Butlor* is designed as a personal assistant to be used by workplace occupants and facility managers to accomplish various tasks in the workplace such as booking a room or guiding an occupant to the nearest printer. *Butlor*, as shown in Fig 5, is dependent on a Language Understanding Service, *LUIS*[1] which receives

[1]https://www.luis.ai/,

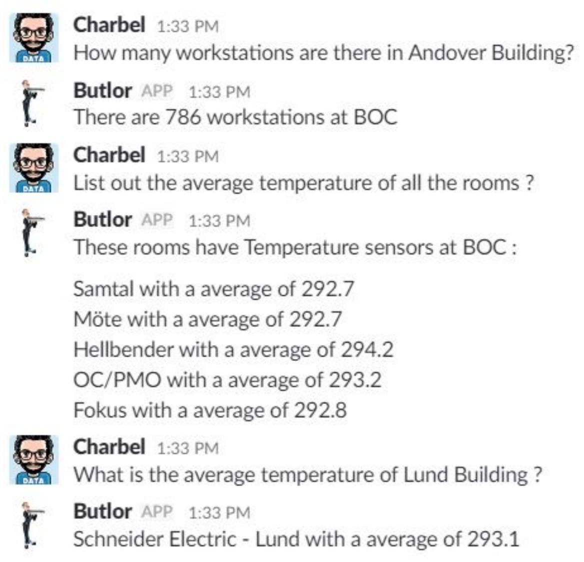
human queries from various channels such as Slack[2].

A language understanding service allows to define a set of intents. Each intent is composed of one or several entity sets. For example, one can arrange all queries $[Q1..Q8]$ under one intent or split them between Knowledge intents $[Q1, Q6, Q7, Q8]$ only and federation intents combining knowledge and timeseries $[Q2, Q3, Q4, Q5]$. We designed the intents structure and entity sets by relying on our ontology concepts. We detail next two intents covering queries $Q1$ and $Q2$ with their entities such as *Quantity* and *PhysicalLocation*.

$$\overbrace{How\ many}^{\text{Quantity}}\ \overbrace{workstations}^{\text{PhysicalLocation}}\ \overbrace{are\ there}^{\text{ignored}}\ \overbrace{in}^{\text{In}}\ \overbrace{Lund}^{\text{Name}}\ \overbrace{Building}^{\text{PhysicalLocation}} \tag{1}$$

$$\overbrace{List\ out}^{\text{QueryCmd}}\ \overbrace{the\ min}^{\text{AggOp}}\ \overbrace{temperature}^{\text{Measure}}\ \overbrace{of\ all}^{\text{NoFilter}}\ \overbrace{the\ rooms}^{\text{PhysicalLocation}}\ \overbrace{today}^{\text{TemporalOp}} \tag{2}$$

Once an facility manager or an end-user inputs a natural text, such as $[Q1..Q8]$ via Slack, for example. The user input is sent to LUIS which parses the text and based on machine learning techniques it matches and generates as an output an intent and its entities as shown in the two previous examples.


Charbel 1:33 PM
How many workstations are there in Andover Building?

Butlor APP 1:33 PM
There are 786 workstations at BOC

Charbel 1:33 PM
List out the average temperature of all the rooms ?

Butlor APP 1:33 PM
These rooms have Temperature sensors at BOC :

Samtal with a average of 292.7
Möte with a average of 292.7
Hellbender with a average of 294.2
OC/PMO with a average of 293.2
Fokus with a average of 292.8

Charbel 1:33 PM
What is the average temperature of Lund Building ?

Butlor APP 1:33 PM
Schneider Electric - Lund with a average of 293.1

Fig. 6. Butlor Answering Queries $[Q1, Q2, Q5]$ through Slack

The generated output is sent to our `Message Handler`, as shown in Fig 5, which parses, extracts the entity sets and forwards it to our `SPARQL Handler` to generate a SPARQL query. The generated SPARQL query is sent to FOrTÉ which executes the query and return the results in a standard format [11]. The returned message is parsed and sent to our `Message Handler` which returns it to the caller with the adequate format, such as the Slack application to render the message, as shown in Fig 6 for the queries $[Q1, Q2, Q5]$[3].

In the following section, we go through our implementation and evaluation of our approach.

[2]https://slack.com

[3]The returned temperature results are in Kelvin Unit

IV. Implementation & Evaluation

To validate and evaluate our approach, we deployed the two services (sensors and floor mapping) at our site in Sweden (SWD) and only the floor mapping service in USA (BoC).

A. Ontology Model

The workplace ontology is modelled with Protégé [8] based on the models provided by our two partners. Table I depicts a summary of the metrics.

Class	Object Prop.	Data Prop.	Axioms
54	49	15	412

TABLE I. Ontology Metrics Summary

B. Data Transformation

The workplace ontology is then used as an input to OLGA [6] to generate a java library containing 54 classes and interfaces of the ontology where each object and data property of the ontology is transformed into methods. The generated library is similar to the one published in our previous work [10]. OLGA generates, compiles, and packages the workplace library in 9 seconds on a Windows 7 machine with 32 GB RAM and an Intel i7 core with 2.80 GHz.

```
...
case "room" :
    Room room = new Room(jsonFacilityName);
    room.Id = jsonFacilityId;
    room.Name = jsonFacilityName;
    room.Surface = UTIL.calculateSurface(jsonFacilityObject);
    ...
    currentFloor.AddRoom(room);
  break;
```

Listing 1: ETL Code Snippet

The generated OLGA library is used in the two ETL processors as shown in Figure 1. The ETL processor requests data from our partners APIs by site ids and receives a JSON file. We implemented a logic in each process to transform incoming data from our partners into ontology instances (topology). Each element found in the JSON file has its correspondence in the ontology model and therefore the generated library. Listing 1 shows part of the business logic. Each processor generates an ontology instance (topology) file wrapped in a header detailing the tenant unique identifier, the ontology model version used and the payload type (timeseries or ontology).

Site	Topology (Floor)	Topology (Sensors)	Timeseries/Measure
SWD	12.69 sec	230 ms	122 ms
BoC	15.54 sec	NA	NA

TABLE II. Topology and Timeseries Generation Time

Table II depicts the time to generate timeseries and ontology instances (topology) for our site in Sweden (SWD) and the topology generation for our site in US (BoC). Our Sensing partner samples the data every minute and pushes it to its own cloud platform, for each measurement it takes 122 ms to retrieve 2000 values and transform it into a message/file.

C. Data Ingestion and Storage

The generated files are then sent to the ingestion layer consisting of an Azure IoT Hub [4] and an IoThub processor. The processor handles the received messages based on the headers and tenant identifier, it inserts the ontology (topology) in an ontology (rdf) store by tenant identifier, or the timeseries data in an SQL store. Stardog community edition 5.0.3 [13] is used as an ontology store and Azure MS SQL as a timeseries database with 250 GB storage and 10 DTU[5]. The IoTHub processor and the rdf store are deployed on the same Linux VM with 7 GB of RAM, 2 cores and a 100 GB SSD disk.

Once transformed, the retrieved data from our partners consists in topology and timeseries data. The number of triples in the SWD site topology is 14,090 and the number of timeseries entries to date is 48,165 which corresponds to sensors data sampled by 15 minutes interval. In the SWD site there are 128 sensors and 117 sensed assets (chairs and tables) which generate 370 measurements of occupancy, temperature, and relative humidity. The Number of Triples in the BoC site is 40,737. The difference in the triples size between the two buildings is due to the difference in the square footage.

Site	Topology (Floor)	Topology (Sensors)	Timeseries/Measure
SWD	1.67 sec	1.25 sec	1 min 13 sec / 2000 entries
BoC	3.28 sec	NA	NA

TABLE III. TOPOLOGY AND TIMESERIES INSERTION TIME

Table III overviews the data insertion performance by our IoTHub Processor for both the topology data retrieved from our floor flan and sensing providers, as well as the timeseries sensor data. The insertion time for the topology data from the floor provider on the SWD site reaches 1.67 seconds and 1.25 seconds to insert sensors information. The two insertion time are needed to ingest 14,090 triples in the RDF DB. As for the timeseries on the SWD site, our processor retrieves 2000 entries for each measurement and inserts those entries in the Azure MS SQL in around 1 min 13 seconds. The BoC site has more square footage and items therefore the number of triples is higher than the SWD site. It takes the IoTHub processor 3.28 seconds to insert 40,737 triples.

D. Query Federation and Chatbot Application

FOrTÉ [7] is used to federate queries from both the RDF and timeseries stores. We implemented FOrTÉ in Java with a dependency on Apache Spark[6] to easily federate data retrieval in a scalable manner. FOrTÉ can run in a standalone (single node) or cluster mode. The performance evaluation of FOrTÉ is detailed in [7]. In this work, we deployed FOrTÉ on the same VM as the RDF store. Table IV outlines the query execution of the queries $[Q1..Q8]$ in milliseconds end-to-end from the Slack channel following the architecture in Fig. 5. The following queries $[Q2, Q3, Q4, Q5]$ require data federation therefore, their execution time is higher than the others which only require data from the topology.

[4]https://azure.microsoft.com/en-us/services/iot-hub

[5]https://docs.microsoft.com/en-us/azure/sql-database/sql-database-what-is-a-dtu

[6]https://spark.apache.org/

Site	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
BoC	45	NA	NA	NA	NA	43	21	24
SWD	20	2,355	1,819	2,709	2,267	30	32	53

TABLE IV. QUERY ANSWERING TIME BY BUTLOR IN MS

Butlor is written in C# and relies on the Azure Bot Framework[7]. It receives natural language queries from the user. The Bot Framework relies on Azure LUIS[1] framework which allows intents to be trained by applying machine learning techniques. LUIS returns a JSON file containing the matched intent and the entities. As mentioned previously, the entities are inspired by the ontology model. The returned JSON file is parsed by Butlor and the returned entities are injected in adequate intent SPARQL query templates, as depicted in Listing 2. Butlor embeds a SPARQL template corresponding to one or several trained intents in LUIS. For example template Listing 2 allows Butlor to answer a user query where any given item of the facility is to be returned based on its type and square footage on any given floor of the building.

```
Select ?Name {
?Building a ws:Building.
?Building ws:hasFloor ?Floor.
?Floor ws:hasName \"" + floor + "\"^^xsd:string
?Floor ws:has" + facility + " ?Facility.
?Facility a ws:" + facility + ". "
?Facility ws:hasName ?Name.
?Facility ws:hasSquareFeet ?number.
filter( ?number >" + sqft + ")"
} GROUP BY ?Building ?Floor ?Facility ?Name
```

Listing 2: SPARQL Query Template Example: Q7

In the following, we depict related work similar to ours.

V. RELATED WORK

In our previous work [5], we proposed an approach to extract data from our Building Management System and Power Monitoring System by applying transformation at a cloud connector level. Our BMS expert had to manually implement all the ontology concepts (class, relations) at the cloud connector level. Such implementation is considered a hard and error prone task since our BMS expert had to gain skills in ontology development along with understanding RDF and OWL. In this current approach, OLGA generates a library based on an ontology model which is used by a developer. OLGA allows to remove complexity and accelerate the implementation time for ontology based application development. In addition, in this work, we rely on FOrTÉ to federate the two sources of data providing a much better performance while in our previous approach, the federation was handled at the application level, Power BI. Butlor is also added in this work making our approach more interactive to the user with his workplace instead of just a static reporting tool.

Youngmin *et al.* [14] provides a data architecture similar to ours where the data is split into two categories: the contextual data and the timeseries data. They propose a runtime stream processing to calculate metrics during runtime information regarding occupancy such as number of people in the room based on CO2 levels. In our approach, we rely on the sensing capabilities of our partners to measure such KPIs instead

[7]https://dev.botframework.com

of inferring them on the fly. Their semantic lifter is highly coupled with the SAREF ontology [15] and it is based on specific hard coded BAS naming rules. Our approach relies on OLGA which is more generic tool capable of generating a library to be used in the semantic lifting making our approach more generic and faster to be put in the hands of actual developers. Youngmin [14] proposes a visualization front-end interface while we rely on Power BI in our previous work [5] and we offer a chatbot for a more interactive experience.

SemanticBMS [16] proposes a BMS ontology inspired from the SSN Ontology [17] where the authors aim to represent BIM (Building Information Model) [18], and a BMS model. SemanticBMS separates between topology and timeseries data, however, no federation mechanism is put in place, it is left to the application to federate data. They propose *Semantic Providers* similar to our ETL processors in order to transform data into semantic representation. However, there is no clear mention how their semantic providers translate the models. In addition, the *Data Providers* handling extraction of timeseries are still underdevelopment. The SemanticBMS project exposes in REST a *Semantic API* to query ontology data by translating rest with parameters queries into SPARQL. One would argue on the choice of reducing the expressivity of SPARQL to Rest parameters based queries.

Other similar work propose an ontology model to represent information in the building such as [19] which proposes monitoring categories of the building. Ploennings [20] relies on an extended sensor network model to enhance building diagnosis. Schachinger [21] proposes a similar approach to ours by relying on an ontology centric model to represent information in a building management system with a focus around automation and control. In comparison to the related work, we consider our approach is more generic and relies on tools such as OLGA [6] and FOrTÉ [7] which reduces complexity and friction for developers to use in their ontology based applications and data federation implementations.

In addition, ontology models around smart buildings are starting to emerge such as Haystack ontology [22] or Brick [23]. Since our architecture is loosely coupled with the ontology model, such ontologies can replace our proposed ontology when maturity, and industrial adoption are achieved.

VI. Conclusion & Future Work

We presented in this paper our experimental work to integrate data from two different IoT platforms we are investigating a partnership with. First, we modelled their data in an ontology. Then, we relied on our model driven approach to generate an object oriented library conform to the model. The generated library is then used in an ETL processor to transform our partners data into semantic information which are stored in an ontology database. Our query engine is then used to federate data between the ontology and timeseries stores. Butlor our chatbot application is integrated in our approach to provide a basic interactive user experience with the environment to answer natural language based queries.

For the next steps, we will tackle commissioning and consistency of data among our partners. So far, we manually ensure the integrity of the references. For example, the room names and identifiers from the Floor Plan provider are consistent with the sensors' location provided by the Indoor sensing provider. In addition, we intent to extend our query capabilities with stream handling capabilities to support more dynamic visualization and alerts.

References

[1] D. Sculley and et al, "Hidden technical debt in machine learning systems," in *NIPS*, 2015.

[2] W3C, "Resource description framework," http://www.w3.org/RDF, 1999.

[3] B. Sean, v. H. Frank, H. Jim, H. Ian, M. Deborah, P.-S. Peter, and A. Stein, "Web ontology language," http://www.w3.org/TR/owl-features/, 2004.

[4] P. Eric and S. Andy, "Sparql query language for rdf," www.w3.org/TR/rdfsparql-query, 2004, sPARQL Query Language for RDF, W3C.

[5] C. E. Kaed, B. Leida, and T. Gray, "Building management insights driven by a multi-system semantic representation approach," in *IEEE WFIoT*, 2016.

[6] C. E. Kaed and A. Ponnouradjane, "A model driven approach accelerating ontology-based iot applications development," in *SIS-IoT: Semantic Interoperability and Standardization in the IoT workshop, part of the 13th Semantics conference*, 2017.

[7] C. El Kaed and M. Boujonnier, "Forte: A federated ontology and timeseries query engine," in *The 3rd IEEE International Conference on Smart Data*. IEEE, 2017.

[8] "Protege," http://protege.stanford.edu, Online- accessed 9-July-2017.

[9] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51 – 53, 2007, software Engineering and the Semantic Web.

[10] "Saref generated by olga: Libraries and examples," https://github.com/InnovationSE/SAREF-Generated-By-OLGA, 2017.

[11] W3C, "Serializing sparql query results in json)," https://www.w3.org/TR/rdf-sparql-json-res/, 2013.

[12] C. Domitille, E. K. Charbel, G. Ayush, H. Chris, H. Jonathan, and S. Stuart, "Energy efficiency driven by a storage model and analytics on a multi-system semantic integration," in *IEEE BIG Data Workshops, Boston*, 2017.

[13] "Stardog," http://www.stardog.com, Online- accessed 9-July-2017.

[14] J. Youngmin, C. Woosuk, O. Kisu, and A. Jooyoung, "Intelligent building using hybrid inference with building automation system to improve energy efficiency," in *Proceedings of the Second Workshop on Semantic Web Technologies for the Internet of Things co-located with 16th International Semantic Web Conference*, 2017.

[15] "Saref," http://ontology.tno.nl/saref, Online- accessed 9-July-2017.

[16] A. Kuera and T. Pitner, "Semantic bms: Allowing usage of building automation data in facility benchmarking," *Advanced Engineering Informatics*, vol. 35, pp. 69 – 84, 2018.

[17] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic sensor web," *IEEE Internet Computing*, vol. 12, no. 4, pp. 78–83, Jul. 2008.

[18] C. Eastman and et al, *BIM Handbook Introduction, in BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors*. John Wiley, 2008.

[19] A. Mahdavi and M. Taheri, "An ontology for building monitoring," *Journal of Building Performance Simulation*, vol. 10, no. 5-6, pp. 499–508, 2017.

[20] J. Ploennigs and et al, "Adapting semantic sensor networks for smart building diagnosis," in *The Semantic Web – ISWC 2014*, 2014.

[21] D. Schachinger and W. Kastner, "Semantics for smart control of building automation," in *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 2016.

[22] C. Victor *et al.*, "An ontology design pattern for iot device tagging systems," *5th International Conference on the Internet of Things*, 2015.

[23] B. Bharathan et al, "Brick: Towards a unified metadata schema for buildings," in *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. ACM, 2016.