BEMSERVER: AN OPEN SOURCE PLATFORM FOR BUILDING ENERGY PERFORMANCE MANAGEMENT

Pierre Bourreau[1], Richard Chbeir[2], Yudith Cardinale[3], Aitor Corchero[4], Khouloud Salameh[5], Jérôme Lafréchoux[1], David Frédérique[1], Gulben Calis[6], Lara Kallab[1,2] and Rafael Constantinou[7]
[1]Nobatek/INEF4 Innovation Center, Talence, France
[2]Université de Pau et des Pays de l'Adour, Anglet, France
[3]Universidad Simón Bolívar, Caracas, Venezuela
[4]Eurecat Technology Centre, Barcelona, Spain
[5]American University of Ras Al Khaimah, United Arab Emirates
[6]Ege University, Izmir, Turkey
[6]Cyric Research Center, Nicosis, Cyprus

Abstract

In the era of new technologies and with the growing need for reliable ecological energy supplies, researchers eyes are diverted towards reducing the energy consumption of buildings which becomes a major concern that requires a variety of actors being involved (e.g., building owners, energy/facility managers, occupants, data scientist), who have different expertise and need to use different tools

This paper introduces BEMServer, an open source Building Energy Management system solution built to ease the integration of data analytic and visualization services into building and energy domain applications. It handles heterogeneous data in a transparent way by providing a clear and unambiguous semantic representation of the modeled resources. To do so, BEMServer integrates the OntoH2G formal ontology at its core. The architecture of BEMServer, the design and use of OntoH2G in the solution, as well as the suitability of BEMServer as a middleware in the H2020 project Hit2Gap are described

Introduction

It is stated that buildings account for more than 40% of the energy consumed worldwide and for more than 30% of the global CO2 emission, as already shown in Shaikh et al. (2014), Costa et al. (2013), and the Commission (2018). Reducing energy consumption in buildings is challenging in many respects. First of all, it is necessary to involve all building actors, from experts (energy manager, ESCOs, facility managers) to non-experts (occupants, building owners). Different users need different tools, therefore a building energy management system needs to be modular. Secondly, most of the currently monitored buildings are equipped with a Building Management System (BMS) to collect data from different sources, meters, sensors, actuators, or set points. BMSs are closed system, which are not service oriented, with poor evolution capacities: extracting and adding values to data stored in BMSs is often a tedious work; communicating with legacy sensors requires handling specific protocols such as BACNet, Modbus, or LonWorks, to mention some. Deploying solutions in various buildings can only be achieved by adopting open standards instead of proprietary technologies. Finally, developers of services need description on the data captured by the monitoring system in order to select those of interest for post-processing or visualization; for buildings which are equipped with hundreds of sensors, the task of energy or facility managers must also be facilitated by attaching a clear and unambiguous semantic to the different data sources.

On one hand, the amount of available information from buildings is dramatically increasing; on the other one, data science has brought novel solutions to add value to such a massive amount of information. Bridging data extraction and data treatment is required to accelerate the deployment of energy applications at a bigger scale. Moreover, all actors must be part of making buildings greener and the situation requires adapting tools to different types of users, while offering interoperability and the capacity of integrating different tools, applications, and protocols, through a semantic data model. To address the aforementioned drawbacks, BEMServer is presented as an open source platform to deploy Building Energy Management Systems (BEMS), with multiple layers of functionalities (see Figure 1), aimed at providing:

- *Data abstraction.* BEMServer integrates a semantic data model to represent the monitored data collected from different sources; it also offers REST APIs to access, manipulate, and visualize those data; thus developers and end users do not need to know how data are collected but only focus on how valuable they are to them; the Semantic layer in Figure 1 acts as an abstraction layer;

DOI:10.35490/EC3.2019.176

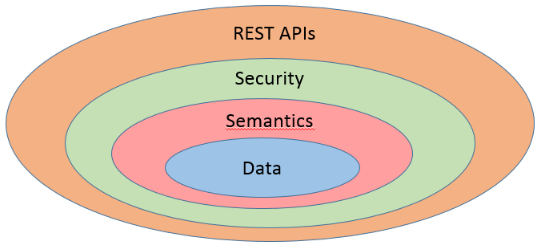

Figure 1: BEMServer layers of functionalities

- *Services integration and interaction*. BEMServer ensures interaction between services, since results generated by different services can be reused by other external modules; this interaction facilitates a better collaborative work between complex modules (for instance, machine learning based services, energy simulation, front-end applications . . .), and the power of composing services to deliver the best output;
- *Data access control*. To protect confidential and sensible data, BEMServer implements a security layer that ensures a secure connection in a multi-tenant environment: users can only access data on specific buildings, and from specific services, and can perform operations according to the rights conceded.

The paper is structured as follows. We first provide a state of the art of solutions similar to BEMServer, and other ontologies similar to OntoH2G, the core data model used in BEMServer. We then provide details on the architecture and the main functionalities of BEMServer, with a focus on the semantic layer. The third section is dedicated to a demonstration of the suitability of the solution through its use in the Hit2Gap project[1], in which BEMServer is used to deploy smart monitoring solutions on four different sites. Finally, we point out conclusions and future works.

Related works

In this section, we provide a comparative study of different data models used for building monitoring purpose, as well as building energy management software.

Data models

The use of linked data has gained interest in recent years as a solution to the complexity of modelling buildings with information relevant to different domains (architecture, energy, structural engineering, . . .). As such, ifcOWL in Beetz et al. (2009), is an RDF translation if the IFC schema: it is therefore an important reference thanks to the high level of modelled information, but suffers from being too verbose and not domain specific, just as IFC files do.

More recently, the Building Ontology Topology (BOT) [2] model was proposed by Rasmussen et al. (2017) within the Linked Building Data W3C working group as the angular ontology that could be used to develop building-related domain ontologies on top of it.

At the operational level, different models have been proposed to describe sensors and building equipment that are monitored. The SOSA ontology proposed by Janowicz et al. (2018) describes sensors and actuators, and the SSN one in Compton et al. (2012) extends it to sensors networks. SOSA and SSN are now widely accepted as standards and are W3C recommendations. They also integrate concepts to store actual measure points, which is a questionable choice since triple stores (i.e. RDF-dedicated storage such as Jena or Stardog) perform poorly when dealing with a big amount of data, like monitoring time series data, while specific time series storage solutions exist.

Several ontologies focus on modeling building equipment. The BASOnt ontology, proposed by Ploennigs et al. (2012), integrates building infrastructure, devices, and data points but does not seem to address the role of occupant information in the overall building management, which could help authorities to elaborate newer management strategies or policies focused on specific building usage at different levels (operational, managerial, or technical). Additionally, it does not seem to be open.

The SAREF ontology, first designed by Daniele et al. (2015), integrates elements related to building equipment, such as HVAC systems, and monitoring devices. While many of these concepts can easily be used to align the SAREF ontology with ifcOwl or SOSA, the ontology seems quite incomplete in the description of building equipment, sensors locations, or occupants integration.

Haystack is a tagging model led by the Corporation (2014) and used to represent the metrology building information in a lightweight strategy. It is far from obvious how to link it with existing building data models like ifcOwl or BOT, since it is a tagging model and not a formal ontology (although an OWL translation of Haystack was proposed in Charpenay et al. (2015)). Moreover, no inference engine can be run on it in a straightforward way to create new relations.

The Brick Schema, presented by Balaji et al. (2016), can be assumed to be the most complete model. It is an RDF formatted ontology for building equipment and monitoring devices. A great number of concepts are described in the model; it is already aligned with both ifcOwl and BOT. Additionally, contributors of the model provide a bridge between Brick and Haystack, which is a popular system in commercial applications.

More recently, Terkaj et al. (2017) proposed a building au-

[1]http://www.hit2gap.eu/

[2]https://w3c-lbd-cg.github.io/bot/

Table 1: Specific Domain Ontologies

Name	Dimensions	Language	Building Domain
ifcOWL	20644 (Axioms) 1315 (classes)	OWL	Generic
BOT	439 (axioms) 9 (classes) 14 (properties)	RDF/OWL	Generic
Haystack	–	–	Operation - Monitoring
Brick Schema	19365 (axioms) 313 (classes)	RDF	Operation - Monitoring
SAREF	1111 (axioms) 92 (classes) 36 (properties)	TTL	Operation - Monitoring
BACS	2165 (axioms) 244 (classes) 95 (properties)	OWL	Operation - Monitoring
CTRLont	144 (axioms) 13 (classes) 11 (properties)	TTL/OWL	Operation - Monitoring

tomation and control systems (BACS) ontology to model automation systems, building appliances, physical devices and their relations. It is aligned with other ontologies (such as SOSA, SSN, and BOT) and offers a detailed representation of automation states and functioning. The CTRLOnt, proposed by Schneider et al. (2017) also models building automation systems (i.e., devices that have the possibility to collect data, but also to control devices).

Table 1 shows a summary of these ontologies.

Energy building monitoring platforms

While many ontologies were developed to describe building monitoring systems, few energy management applications built on top of these models seem to exist.

McGibney et al. (2016), propose OpenBMS as an IoT-based architecture to manage blocks of buildings. An extension of the BASOnt ontology is used at the core of OpenBMS. BASOnt being aligned with ifcOwl, it makes OpenBMS a BIM-compliant system, although the integration of BIM data is not explicitly mentioned as a functionality of the platform. Additionally, there is no reference to the integration of some preprocessing services and how the data quality is handled in OpenBMS.

The BOSS (Building Operating System Services) architecture, proposed by Dawson-Haggerty et al. (2013), is divided into five layers: (i) the hardware presentation layer, that handles a direct communication with different on-site devices to expose the very same data to some standard REST APIs; (ii) the hardware abstraction layer that semanticizes collected data; (iii) the time series management layer to store and preprocess data points; (iv) the control transaction layer to be used by external services that can control buildings systems through a BOSS implementation; and (v) the authorization layer that handles control access. The integration of semantic into BOSS is not fully detailed and does not seem to rely on W3C recommendations, both on the technologies used (i.e. SPARQL query language) and on the models used (i.e. SOSA, ifcOwl).

While formal ontologies are relevant to associate some semantic to measure points, triple stores performance is low when handling actual time series values. To solve this issue, a hybrid architecture is presented in Hu et al. (2016), where both semantic data and actual values of measure points are handled. The architecture is generic in the sense that different storage solutions can be handled and described into an RDF-based ontology (RDF-DB), to ease the integration of new storage solutions; in this work, time series data are stored in relational databases.

Other solutions, freely available or commercial, offer services to collect, analyze, and process data for energy management, but they do not implement a semantic data model. Such are the cases with the open source solutions from Al Faruque and Vatanparvar (2016) and Pan et al. (2015), and the commercial Skyspark [3] solution, which limits these solutions in terms of interoperability, and evolution. Nevertheless, Skyspark is interesting in many aspects: it has a capacity to handle a huge variety of proprietary protocols (e.g. KNX, BACNet) that are used in building monitoring systems, and to offer the possibility for end users to configure their own data visualization modules. It also integrates a rule-based language to generate alerts.

Finally, some initiatives to push the community towards using formal ontologies at the core of building energy monitoring applications exist. For instance, the Brick's consortium encourages to do so, but exposing data through a SPARQL endpoint is a barrier for application developers that would need to develop their own SPARQL queries, instead of using more popular SOAP or REST APIs.

BEMServer - Technical description

In this section, the overall architecture of BEMServer is introduced, its main functionalities are presented, and some insights on the semantic data model are given.

Overall architecture

BEMServer aims at being a core solution to deploy smart energy building solutions that are: (i) modular; (ii) extensible; and (iii) secure. A variety of data needs to be stored and exposed to end-users and connected services. Three main categories of data were identified: (i) time series

[3]https://skyfoundry.com/product

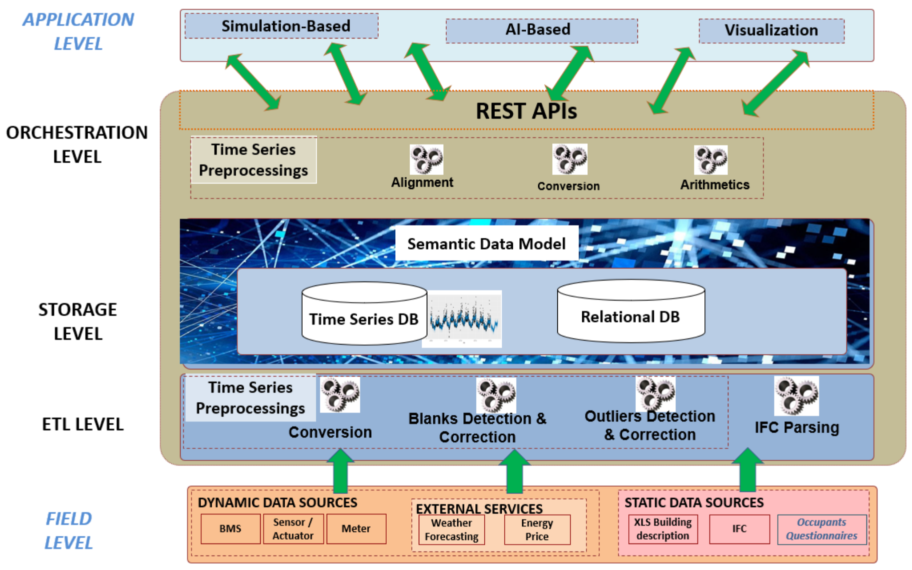

Figure 2: BEMServer architecture

data (i.e. sequences of timestamps/values pairs), which are data points collected by sensors/meters, set points on HVAC (Heating, Ventilation, and Air Conditioning) systems, and potentially data generated by services (such as predictive services); (ii) events which are elements generated by additional services, such as errors detected on sensors or identified potential energy savings; and finally (iii) metadata, which is information used to attach semantic to the data stored and exposed; as an example, metadata consist of all the data that is required to describe the value points collected by a sensor measuring the temperature of an exhaust air zone of an air handling unit. The management of metadata is one of the core functionalities of BEMServer, and it is detailed in the next subsection.

The functional architecture of BEMServer is depicted in Figure 2. It is composed of four layers: (i) the **Field Level**, where on-site data is collected from meters, sensors, BMSs, etc., but also from documents like IFC[4] files; (ii) the **Extract-Transform-Load (ETL) Level**, that is used to connect the BEMServer services with the Field Level; this layer is in charge of collecting on-site data, from smart sensors/meters, to data stored in traditional BMSs, but also building descriptions through IFC files or *ad hoc* Excel files, and occupants answers to questionnaires; (iii) the **Storage Level** in charge of keeping track and semanticizing data; in this level, several technologies are used to deal correctly and efficiently with the heterogeneity of the data stored: a Time Series Database (TSDB) solution is used to store continuous monitoring data; the current implementation makes use of the HDF5 file format; a relational database (RDB) is used to store events; and (iv) the **Management (Orchestration) Level**, where different services can be plugged to the BEMServer so as to deliver some added value to the data stored and shared within the platform; all the data handled in BEMServer are exposed via standard REST APIs, running over the HTTP protocol. REST APIs are widely adopted in the SaaS industry, and it ensures a homogeneous data access to BEMServer clients, hiding the complexity of using three different data storage solutions.

[4]http://www.buildingsmart-tech.org/specifications/ifc-overview

The secure access is provided through: (i) SSL encryption over HTTP communications; (ii) the delivery of cookie sessions to BEMServer clients; and (iii) the implementation of a Role-Based Access Control to reduce the scope of available resources and operations for registered users.

BEMServer has been developed to manage all these data in a transparent way for end-users and to connect the two sides of the same smart building coin: the Field Level and the Management Level.

The Semantic Data Model

Numerical values returned by sensors are useless without a meaning. In the double-blind architecture proposed through the BEMServer, it is necessary to attach semantic to the data extracted from sensors, smart devices, questionnaires, etc. OntoH2G, the formal ontology used at the core of BEMServer (presented in Chbeir et al. (2018)), allows to describe building infrastructure, including physical and digital entities, as well as user-building interactions not only in the form of activities but also comfort requirements, user preferences, and other aspects that motivate users to interact with the building. In order to do so, different domain specific ontologies are aligned, among which:

- SOSA (Sensor, Observation, Sample, and Actuator) ontology, a W3C recommendation used to describe sensors, actuators, and measures, while offering various concepts to ease the alignment with other ontologies;
- ifcOwl, used to describe building infrastructures and equipment; ifcOwl is the OWL version of usual EXPRESS formatted IFC files; as such it offers concepts covering many areas of interest in building descriptions, from infrastructure, energy equipment, measurement devices, to the geometry;
- Haystack, a tagging system developed by major BMSs vendors, and it is being increasingly adopted by smart building software industries, with the goal of becoming a standard so that new generations of smart measurement devices can integrate tags and attach them to the data points sent.

SOSA has the expressive and flexibility power of formal ontologies and can easily be linked to other ontologies, but it does not contain any information on building equipment and does not describe any of the relations that can be useful to monitor complex building heating or cooling assets, for instance. Thus, OntoH2G aligns SOSA with ifcOwl, to complement the semantic needed in BMSs. Additionally, ifcOwl is augmented with some features, and simplified to represent walls adjacency or space adjacency for instance. While SOSA can be used to store actual measures performed by sensors, those are stored in a time series storage solution, for performance reason. Each measure is then related to an ID to access actual time series, in the OntoH2G model.

The ifcOwl ontology, even if rich of concepts, is not specific to the building monitoring domain. In particular, relating an Observation with a particular element of an HVAC system cannot be done in an obvious way. To fill this gap, OntoH2G is augmented with concepts extracted from the Haystack project. Haystack is a simple tagging system, not a formal ontology and is therefore hard to bind to a formal building description, but it contains description of complex measurement types, that can be made in the context of building systems monitoring. Figure 3[5] provides an example of an AHU (Air Handler Unit) discharge air temperature measure, using the Haystack tagging system. The measure is associated to a specific part of an equipment, (i.e., the discharge part of an air handling unit), a kind (i.e., number), a unit (i.e., degree Fahrenheit), a physical element (i.e., air), and an observation type (i.e., temperature).

[5]The example is taken from `https://project-haystack.org/tag/point`

```
id:  @whitehouse.ahu3.dat
dis:  "White House AHU-3
DischargeAirTemp"
point
siteRef:  @whitehouse
equipRef:  @whitehouse.ahu3
discharge
air
temp
sensor
kind:  "Number"
unit:  "°F"
```

Figure 3: Example of a measure representation in Haystack

In the scope of BEMServer, coupling SOSA and Haystack is done considering concepts in ifcOwl which share some concepts with both SOSA (i.e., Sensors) and Haystack (i.e., building HVAC equipment: air handling units, coils, heaters, boilers).

Our proposal for OntoH2G does not rely on a simple alignment of the Haystack ontology in Charpenay et al. (2015) with SOSA and ifcOwl, for efficiency reasons: both ifcOwl and the Haystack are verbose, being direct translation of some formats into OWL triples. This direct translation would have an impact on the complexity of an instance creation (because intermediate relations would need to be created) and on the response time on queries. Thus, the adaptation of OntoH2G for BEMServer, instead, is focused on enriching SOSA and ifcOwl with relations, classes, and individuals of relevance for BEMServer based on the information described in Haystack.

The following concepts from Haystack and ifcOwl are integrated into OntoH2G:

- *Spatial elements*: an observation may be related to some specific zones; it is particularly true for ambient sensors, i.e., those who perform some measurements like temperature, humidity, CO2 rate, in a specific zone of a building
- *Equipment*: in particular representing HVAC equipment, which are heavy energy consumers. Those systems are often monitored in big buildings to ensure they are not deteriorated, or even not badly programmed.
- *The type of observation*: observation type entails the physical measurement (e.g., temperature, energy, power) and also the physical medium (e.g., air, oil, water).
- *The unit*: that needs to be coherent with the physical measurement.

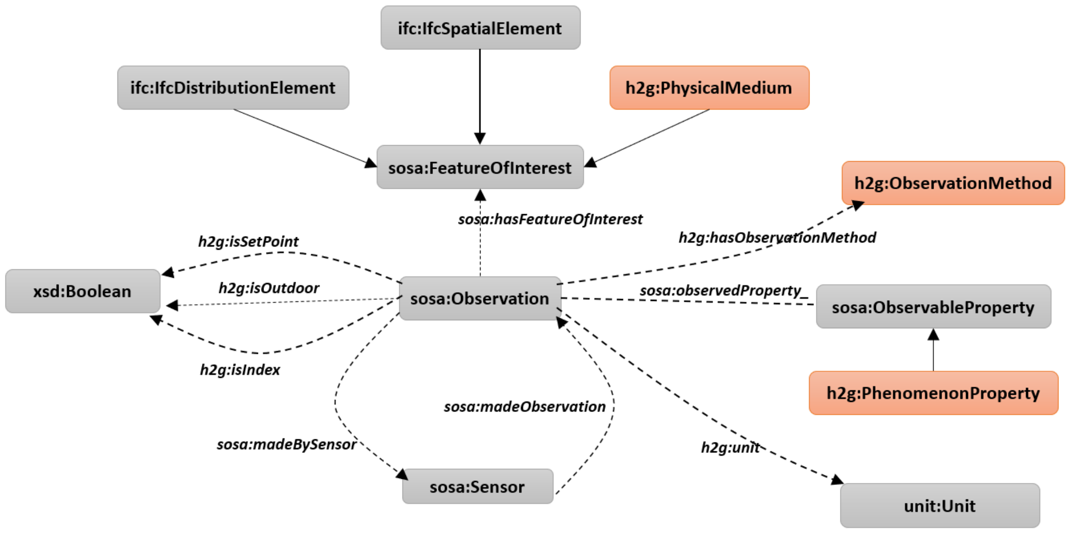

Figure 4: Classes and relations related to measures in OntoH2G

Additionally, more information needs to be modeled to assist end-users and services running on the time series to discriminate data:

- Is the observation a set point? A set point is a programmed value, not an observation (for instance, it can be the minimal temperature at which the heating system needs to be turned on). For services running simulations or energy consumption prediction, this data is particularly important and can be integrated in models.

- How is the observation made? Two modes are modeled in BEMServer ontology: observations performed at a fix frequency rate and those performed when a change of value is observed (for instance, a sensor checking the open/close status of a window, or valve). Such data are important in the BEMServer architecture and in particular for the missing data detection/correction services.

The final proposition is represented in Figure 4. The `sosa:FeatureOfInterest` concept was extended to model spatial elements, systems, and physical mediums on which the measure is performed. The `sosa:ObservableProperty` was extended to express the physical phenomenon that are measured. Finally, units are not modelled through the `sosa:hasResult` relation as expressed in the online documentation of SOSA because numerical values, are not modelled in the present ontology. Additionally, if some of the sensors can perform various measures, no measure can change units at runtime, but units need to to be predefined and fixed at the installation of the sensors.

Services and data preprocessing

The Storage level of BEMServer is in charge of attaching a semantic to the data collected on sites. This, in addition to the REST APIs of the Orchestration level, provides an easy access to the data relevant to applications at the top level of the architecture. For instance one can obtain the list of air temperature measures in a certain zone using correct filters in a REST route: `https://h2g-platform-core.nobatek.com/api/v0/measures/?location_id=XXXX&observation_type=Temperature&medium=Air`[6].

Such a functionality eases the integration of new applications; nevertheless, monitored data can also suffer from different issues that can prevent them from correctly running: sensors can be offline for some periods, can diverge from their initial calibration, or simply be measuring data with some undesired noise. These situations can cause corresponding time series to contain blanks (i.e., when no data is erroneously collected) and outliers (i.e., data that contain abnormal values). Developers of services at the Application Level, may not be energy building experts (for instance, front-end developers) and they should just care about working with values that are as-correct-as possible. In order to assist them in their tasks, the BEMServer embeds some preprocessing services so that:

- *Data cleansing*: blanks and outliers can be detected and corrected;

- *Resampling*: time series data can be resampled at a desired frequency;

- *Conversion*: time series values can be converted to a different value unit;

[6]The full list of APIs is available at `https://h2g-platform-core.nobatek.com/api/v0/api-docs/redoc`

- *Operations*: basic arithmetic operations can be performed so as to retrieve the minimum or maximum value of a time series on a given time range; to compute the mean value or the sum of a time series on a given time range; or to sum multiple time series.

In the case of arithmetic operations on multiple time series, all the data must be resampled at the very same frequency. Hence, the resampling and arithmetic services need to be combined. In a micro-service architecture, this issue is common and requires a services composition functionality. In BEMServer, this functionality is being developed in a semi-automatic way: services are described using an extension of the Hydra vocabulary presented in Lanthaler and Gütl (2013) and an API is available for end-users to specify the way services can be composed. This way, end-users can describe how to compose different services as workflows to get the desired preprocessed data. For example the data cleansing service can be first invoked, its output transferred to the resampling service, which prepares the data for the target arithmetic operation service. This tool is currently being integrated in the solution and has not been tested within the Hit2Gap project.

The Hit2Gap project: A Case Study

In this section, a description of real case study using BEMServer is presented, and some related conclusions are provided.

Description

The Hit2Gap project aims at reducing the energy gap at the exploitation phase of non-residential buildings. Four different pilot sites provide case studies in four different climate regions (Paris, Warsaw, Galway, and San Sebastian). The types of the sites vary from educational buildings (NUIG in Galway and Nanogune in San Sebastian), to an administrative office (Wilanow Town Hall in Warsaw) and a commercial private office (Challenger in Paris). BEMServer is the central solution used to monitor those four different buildings through the deployment of different types of services: (i) AI-based services for energy consumption prediction developed by Massana et al. (2016), fault and detection diagnosis developed by Burgas et al. (2018), and behavior modelling from Calis et al. (2016); (ii) building performance simulation for the identification of potential energy savings and discomforts, model calibration by Monari and Strachan (2017); and (iii) data visualization services for different actors, from experts (i.e., facility or energy managers) to non-experts (building occupants).

A single instance of BEMServer was deployed to monitor the 4 sites, and 16 modules were plugged at the Management Level. Another module was plugged at the Field Level as a bridge between proprietary communication protocols (e.g., Modbus, BACNet) and the HTTP protocol. Additionally, BEMServer is connected to OpenWeatherMap[7], an external service that provides current and predicted weather data (for buildings that are not equipped with a weather station).

Table 2 gives an overview of the amount of data gathered by BEMServer in the Hit2Gap project. Note that more descriptive data (i.e. floors, spaces, walls, windows, surfaces, volumes...) are available for NUIG's building, than for the other three sites, since the information was extracted from an available IFC file for NUIG, while the other sites were manually described. In particular, information on façades, walls, slabs, windows, and windows covering could be retrieved in a straightforward way, while the same information is not modeled in BEMServer for the other sites.

The number of measure points also differs from one site to another: as shown in Table 2, a big amount of data is collected on the French site, which is the one equipped with an important number of sensors and meters (see column Sensors in Table 2). In the case of the Irish site, a high quantity of data points has been recorded because this site was connected to the BEMServer solution almost a year before the other ones.

Feedbacks from the experiment

The presentation of all the data handled in the storage layer through REST APIs was of great importance during the integration of the different modules, as the technical complexity was hidden to end users. For BEMServer developers, coding such APIs on top of a triple store is nevertheless time consuming. Several approaches are maturing to solve this issue, like the use of GraphQL for Linked Data in Taelman et al. (2018) or the automatic generation of REST APIs on top of Linked Data as presented in Schröder et al. (2018).

Preprocessing services reveal to be an important feature of BEMServer to help software editors to develop services, in particular for those with no expertise on building monitoring. It is important to remark that raw data is still available, so that experts can use their own preprocessing services.

The main efforts when deploying BEMServer is to collect metadata on building infrastructure and sensors. Such a task requires a close cooperation with building managers to get accurate data on measure types, units and locations. Extracting this data from an IFC file is most of the time important, as sensors are often not reported in BIM models, even though the schema offers concepts to do so.

The connection with the Field Level is not entirely man-

[7]`https://openweathermap.org/`

Table 2: Data storage in the Hit2Gap project

Site	Building Metadata	Number of Sensors	Number of Data points
Challenger, Paris	199	2494	57.061.659
Nanogune, San Sebantián	139	576	4.952.007
Wilanow, Warsaw	78	136	19.616.834
NUIG, Galway	9546	350	70.694.192

aged by BEMServer, but an intermediate software is used for its ability to connect to the main sensors protocols. Despite these limitations, this use case shows that BEMServer is suitable and adaptable to handle multi-sites and multi-tenant data in a secure way. A more complete connection to the Field level will be studied in the future, through the use of gateways between local networks protocols and web APIs, such as the one in Bonino et al. (2008).

Some performance leaks were also revealed during the project. For instance, HDF5 pilots do not handle concurrency, and a move to other solutions, such as InfluxDB or DruidIO, is being studied. Also, SPARQL queries will be optimized to offer a better response time.

Conclusion

BEMServer is an open source solution to deploy modular, secure, and semantic building energy management solutions. It is based on different standards from the Web (REST APIs, linked data), to ease the work of web services developers and different storage solutions to handle heterogeneous data. BEMServer is based on a semantic data model (OntoH2G), to describe measure points, as well as occupants and user information. Thus it offers a holistic model to represent the building and the human interaction impacting energy consumption. The deployment of the solution in the H2020 project Hit2Gap shows its suitability to monitor non residential buildings, and its robustness in a real case study. Its usage in residential buildings as the use of a semantic layer is perhaps not useful when few measure points are managed.

In the future, some performance issues need to be tackled to make BEMServer more efficient. Additionally, the use of the ifcOwl ontology made queries complex because of the complexity of this model, thus we wish to align our model to ontologies from the Linked Building Data W3C group, in particular the BOT ontology, which is more concise and made of direct relations between concepts, while keeping compliance with the IFC model. Additionally, the connection of BEMServer and Dog-Gateway will be evaluated to handle proprietary protocols. Finally, more preprocessing services will be developed to offer more quality of service to end-users, like blanks and outliers correction based on machine learning techniques.

Acknowledgements

The HIT2GAP project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 680708.

References

Al Faruque, M. A. and Vatanparvar, K. (2016). Energy management-as-a-service over fog computing platform, *ieee Internet of Things Journal* pp. 161–169.

Balaji, B. et al. (2016). Brick: Towards a unified metadata schema for buildings, *Proceedings of the* 3*rd ACM International Conference on Systems for Energy-Efficient Built Environments*, pp. 41–50.

Beetz, J., van Leeuwen, J. and de Vries, B. (2009). Ifcowl: A case of transforming express schemas into ontologies, *AI EDAM* pp. 89–101.

Bonino, D., Castellina, E. and Corno, F. (2008). The dog gateway: enabling ontology-based intelligent domotic environments, *ieee Transactions on Consumer Electronics* pp. 1656–1664.

Burgas, L. et al. (2018). N-dimensional extension of unfold-pca for granular systems monitoring, *Engineering Applications of Artificial Intelligence* pp. 113 – 124.

Calis, G. et al. (2016). A methodology to develop a user-behaviour tool to optimize building users' comfort and energy use, *Proceedings of the International Conference on Sustainable Built Environment*.

Charpenay, V. et al. (2015). An ontology design pattern for iot device tagging systems, *Proceedings of* 5*th International Conference on the Internet of Things*, pp. 138–145.

Chbeir, R. et al. (2018). Ontoh2g: A semantic model to represent building infrastructure and occupant interactions, *Proceedings of the International Conference on Sustainability in Energy and Buildings*, pp. 148–158.

Commission, E. (2018). Energy efficiency of buildings, `https://ec.europa.eu/energy/en/topics/energy-efficiency/buildings`. Online; accessed January 2019.

Compton, M. et al. (2012). The ssn ontology of the w3c semantic sensor network incubator, *Web Semantics: Science, Services and Agents on WWW* pp. 25 – 32.

Corporation, H. (2014). Project haystack, `https://project-haystack.org`.

Costa, A. et al. (2013). Building operation and energy performance: Monitoring, analysis and optimisation toolkit, *Applied Energy* pp. 310 – 316.

Daniele, L., den Hartog, F. and Roes, J. (2015). Created in close interaction with the industry: the smart appliances reference (saref) ontology, *Proceedings of the International Workshop Formal Ontologies Meet Industries*, pp. 100–112.

Dawson-Haggerty, S. et al. (2013). Boss: Building operating system services, *Proceedings of the* 10*th Conference on Networked Systems Design and Implementation*, pp. 443–458.

Hu, S. et al. (2016). Building performance optimisation: a hybrid architecture for the integration of contextual information and time-series data, *Automation in Construction* pp. 51–61.

Janowicz, K. et al. (2018). Sosa: A lightweight ontology for sensors, observations, samples, and actuators, *Journal of Web Semantics* .

Lanthaler, M. and Gütl, C. (2013). Hydra: A vocabulary for hypermedia-driven web apis, *Proceedings of the* 2013 *Workshop on Linked Data on the Web*.

Massana, J. et al. (2016). Short-term load forecasting for non-residential buildings contrasting artificial occupancy attributes, *Energy and Buildings* pp. 519 – 531.

McGibney, A., Rea, S. and Ploennigs, J. (2016). Open bms - iot driven architecture for the internet of buildings, *Proceedings of the* 42*nd Annual Conference of the ieee Industrial Electronics Society*, pp. 7071–7076.

Monari, F. and Strachan, P. (2017). Calibro : an r package for the automatic calibration of building energy simulation models, *Proceedings of Building Simulation* 2017.

Pan, J. et al. (2015). An internet of things framework for smart energy in buildings: designs, prototype, and experiments, *ieee Internet of Things Journal* pp. 527–537.

Ploennigs, J. et al. (2012). Basont - a modular, adaptive building automation system ontology, *Proceedings of the* 38*th Annual IEEE Conference on Industrial Electronics Society*, pp. 4827–4833.

Rasmussen, M. H. et al. (2017). Proposing a central aec ontology that allows for domain specific extensions, *Proceedings of the Joint Conference on Computing in Construction* (*JC*3), pp. 237–244.

Schneider, G. F., Pauwels, P. and Steiger, S. (2017). Ontology-based modeling of control logic in building automation systems, *ieee Transactions on Industrial Informatics* pp. 3350–3360.

Schröder, M. et al. (2018). Simplified sparql rest api - crud on json object graphs via uri paths, *Proceedings of the* 15*th Extended Semantic Web Conference* (*Posters and Demos*).

Shaikh, P. H. et al. (2014). A review on optimized control systems for building energy and comfort management of smart sustainable buildings, *Renewable and Sustainable Energy Reviews* pp. 409 – 429.

Taelman, R., Vander Sande, M. and Verborgh, R. (2018). Graphql-ld: Linked data querying with graphql, *Proceedings of International Semantic Web Conference* 2018 - *posters and demos papers*.

Terkaj, W., Schneider, G. F. and Pauwels, P. (2017). Reusing domain ontologies in linked building data : the case of building automation and control, *Proceedings of the Joint Ontology Workshops* 2017 *Episode* 3: *The Tyrolean Autumn of Ontology*, p. 12.