BRICKS: Building's reasoning for intelligent control knowledge-based system

Gabriel Santos, Zita Vale, Pedro Faria*, Luis Gomes

GECAD – Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, Institute of Engineering – Polytechnic of Porto (ISEP/IPP), Rua Dr. António Bernardino de Almeida, 431, 4200-072, Porto, Portugal

Keywords:
Building management systems
Context-aware knowledge-base systems
Intelligent control
Interoperability
Semantic reasoning
Semantic rule-based systems

A B S T R A C T

Building energy management systems have been largely implemented, focusing on specific domains. When installed together, they lack interoperability to make them work correctly and to achieve a centralized user interface. The Building's Reasoning for Intelligent Control Knowledge-based System (BRICKS) overcomes these issues by developing an interoperable building management system able to aggregate different interest domains. It is a context-aware semantic rule-based system for intelligent management of buildings' energy and security. Its output can be a set of alarms, notifications, or control actions to take. BRICKS itself, and its features are the innovative contribution of the present paper. It is very important for buildings' energy management, namely in the scope of demand response programs. In this paper, it is shown how semantics is used to enable the knowledge exchange between different devices, algorithms, and models, without the need for reprogramming the system. A scenario is deployed in a real building for demonstration.

1. Introduction

Over the last decades, the industry has been developing and presenting various solutions in the area of energy management (Aduda, Labeodan, Zeiler, Boxem, & Zhao, 2016; Blaauwbroek et al., 2015; Brusco, Burgio, Menniti, Pinnarelli, & Sorrentino, 2014; Calvillo, Sánchez-Miralles, & Villar, 2016; Figueiredo & Costa, 2012; Jajac, Knezic, & Marovic, 2009; Marzband, Alavi, Ghazimirsaeid, Uppal, & Fernando, 2017; Marzband, Fouladfar, Akorede, Lightbody, & Pouresmaeil, 2018; Park, Ryu, Choi, Kim, & Kim, 2015; Shakouri & Kazemi, 2017; Simmhan et al., 2013; Su et al., 2019; Yu, Haghighat, & Fung, 2016; Zhou, Fu, & Yang, 2016). In particular, energy management systems for buildings have also undergone a technological revolution with the emergence of different technologies, protocols, and standards. However, it has been hard to find a solution that can integrate into a single system all the functionalities and decision support that guarantee optimized management of the available energy resources (Lee, Kwon, & Lee, 2014).

In fact, buildings are key components of smart grids (SGs) (Calvillo et al., 2016; Lawrence et al., 2016) and a very relevant source of flexibility in the context of demand response (DR) programs (Aduda et al., 2016; Faria, Spinola, & Vale, 2016; Park et al., 2015; Simmhan et al., 2013; Yu et al., 2016). While in the past a building wasn't enough to provide the minimum required reduction amount, nowadays an aggregator has a key role in collecting the individual building contributions for DR events (Faria & Vale, 2011; Marzband et al., 2018; Park et al., 2015). A building management system (BMS) can provide an automated response to a DR event, improving the building's efficiency (Figueiredo & Costa, 2012; Simmhan et al., 2013). Moreover, smart contracts techniques are also being applied in the power and energy sector (Andoni et al., 2019; Liu, Chai, Zhang, & Chen, 2019; Wang et al., 2019), which may be included in future BMS for a smarter ecosystem. The blockchain technology has proven to be resilient to tampering, which is very appealing especially when dealing with money transfers that must respect certain previously agreed conditions (Liu et al., 2019).

In what concerns energy efficiency, the limitations of BMS include the need for the detailed identification of the various points of energy consumption in buildings and need for the effective retrofitting of systems and equipment in buildings (Marzband et al., 2018; Park et al., 2015; Wijayasekara, Linda, Manic, & Rieger, 2014). This aspect will be more relevant with the rise of the need for the dynamic and intelligent management of a building, aggregating consumers and allowing them to participate actively in the management of consumption, taking into account the criticality of the loads that are in the surrounding context (Figueiredo & da Costa, 2012; Wijayasekara et al., 2014). Comfort issues are also very relevant in this field. The work presented in (Gómez-Romero, Molina-Solana, Ros, Ruiz, & Martin-Bautista, 2018) introduces a new concept of comfort as a service, as a new energy supply paradigm for providing comfort to residential users.

* Corresponding author.
E-mail addresses: gajls@isep.ipp.pt (G. Santos), zav@isep.ipp.pt (Z. Vale), pnf@isep.ipp.pt (P. Faria), log@isep.ipp.pt (L. Gomes).

Current systems rely on simplistic controls that require the user to enter information manually, such as operating times or desirable temperatures, while truly intelligent systems should calculate and understand all of these context variables autonomously. Moreover, solution developers continually redefine basic concepts rather than reuse widely accepted knowledge due to the lack of consensus and functionality gaps (Domingues, Carreira, Vieira, & Kastner, 2016). Additionally, with the considerable increase of renewable energy sources in places close to the points of consumption, BMS should be able to integrate such components, adopting a dynamic resource management philosophy, taking into account several time horizons and functioning in isolated mode or connected to the main network (Burmester, Rayudu, Seah, & Akinyele, 2017; Wang, Wang, Chu, Pota, & Gadh, 2016).

Critical buildings have a technological ecosystem replete with several closed systems that control various equipment and functions of a specific domain. For example, ventilation control, access control or the control and security of computer networks are likely to be carried out by different systems using different protocols and therefore do not allow the exchange of information between them, enabling little or no capacity for interoperability between them. It could be interesting to take advantage of different systems' knowledge to provide a more intelligent building management, such as controlling the windows shades and lights' intensity according to the available outside natural light; or using access control data to turn on the A/C and adjust the office temperature for a given user's preference. However, these systems are usually closed and do not allow the interaction with other systems. Given these limitations, it is the decision of the building managers to keep systems isolated by giving up interoperability, which results in increased manual tasks and possibly could lead to more errors (Somfy, 2019; ThinkEnergy, 2019). Additionally, these limitations also harden the adoption of such systems by residential users.

BMSs also have strong security limitations. An example is the vulnerabilities in the architecture of cyber-physical networks. The minimum level of structuring of cyber-physical networks should be the separation of safety networks, and access control of the work networks. Nowadays, Supervisory Control And Data Acquisition (SCADA) systems are not only subject to the inherent risks of personal computers, such as Transmission Control Protocol/Internet Protocol (TCP/IP), WWW, mail and others, but are also subject to threats which target personal computers such as viruses/worms, sabotage/hacking, and human failure (Leszczyna, 2018; Nicholson, Webber, Dyer, Patel, & Janicke, 2012). Recent attacks were exploiting vulnerabilities in energy meters, loss of confidentiality, loss of integrity (by manipulating data from the readability of the system), availability losses (by not guaranteeing the availability of data in the necessary time intervals), fraud in consumer information and misuse as attack platforms (Althunibat, Wang, & Granelli, 2016; Shang, Ding, Marianantoni, Burke, & Zhang, 2014).

Another important aspect to mention is the growing use of open-source tools (hardware/software) in the installation and development of BMS (Martinez et al., 2017; Mejías et al., 2017). The open-source movement promises high quality, flexibility, reliability, and lower cost. A survey made by 2010 showed that open-source tools were being used by 98% of the companies that responded to it (C|net Survey, 2017). This highlights the need to develop an intelligent and secure system that integrates energy management, communication network security, and promotes interoperability of all services. In this scope arises the "Building's Reasoning for Intelligent Control Knowledge-based System" (BRICKS). BRICKS main objective is to provide an intelligent, integrated, efficient, and optimized building management and control. BRICKS is not a SCADA system. It is a platform developed to be at a higher level where it can integrate different SCADA systems and/or smart appliances, and to apply rules composed by knowledge from different sources for an intelligent building management. This contributes to the evolution of the Smart Grid paradigm, as well as to the integration of renewable-based energy sources. It will optimize the use of renewable energy, take advantage of the flexibility of loads and allow the buildings to become an active player capable of reducing energy costs and even to adopt business models that allow obtaining profits through real-time management of its resources (GREEDi, 2017). BRICKS is modular and can easily be reused in many buildings, significantly reducing development costs. BRICKS is supported by a semantic context-aware rule-based system which is agnostic to the semantic model and rules; in this way, the model and rules may change without the need of reprogramming the system.

The main objectives and contributions of this work are:

- to develop a flexible, configurable, and context-aware system for building management;
- to overcome the difficulty of interaction between heterogeneous devices in a building;
- to keep the system abstracted from the hardware installed and respective communication protocols in order to avoid reprogramming the system every time a device is installed;
- to have a rule-based system agnostic to the rules and data, enabling advanced machine intelligence;
- to have a system that can be applied to different buildings by using it or parts of it in other buildings for which the same semantic model, semantic converter, and/or semantic rules apply;
- to centralize the interface to manage the heterogeneous cross-domains' monitoring and alarms.

After this introductory section, Section 2 presents the related work. Section 3 presents BRICKS, the context-aware semantic rule-based system for building management. A case study is presented in Section 4, and the results are shown in Section 5. The final conclusions and future work are stated in Section 6.

2. Related semantic approaches

This section starts by introducing relevant work developed on building semantic models and concludes with some discussion on semantic rule-based systems.

2.1. Buildings' semantic models

Several projects have been working on semantic models for building management, usually focusing on a specific aspect of the building. Both Brick Schema (Brick, 2018) and Haystack (Project Haystack, 2014) projects implement strong metadata models aiming at providing intelligence to building equipment while helping the implementation of realistic buildings descriptions considering energy management, Demand Response (DR), fault diagnosis, occupancy model, among others. However, the ontologies developed by these projects are more taxonomies with weak relationships between concepts, lacking the potential of actionable entities.

Regarding actionable entities, OneM2M's (Alaya, Medjiah, Monteil, & Drira, 2015) base ontology is the most consensual reference from a significant number of organizations. SAREF (Daniele (2016) ontology describes smart appliances control of energy consumption, control devices (meter, switch, sensor), and actionable command and services. It uses strong relations among its concepts and acts as a dynamic ontology with other ontologies such as DogOnt (Bonino & Corno, 2008) and SSN (W3C Semantic, 2018), which are relevant assets in the IoT context. SAREF is being extended to include alignments with IFC4 (buildingSMART, 2018) for building reference components and for energy management and DR capabilities with the support of Energy@Home project (Energy@home, 2019) and EEBus (EEBus, 2017).

Other relevant works have been done related to smart buildings. Examples of such are the SEAS (Smart Energy Aware Systems) (Lefrançois, Kalaoja, Ghariani, & Zimmermann, 2016) and FSGIM (Facility Smart Grid Information Model) (ASHRAE Project, 2018) ontologies. SEAS (Lefrançois et al., 2016) knowledge model has been

developed as modular ontologies. It is an enabler for semantic interoperability within SEAS ecosystem, covering different aspects of SGs while enabling the interaction between smart energy systems to optimize the overall energy usage. FSGIM (ASHRAE Project, 2018) has been developed for describing facilities and their assets, using the Unified Modelling Language (UML), and it can be translated into an ontology format. It's worth mentioning that it supports the participation of electrical energy consumers in SG networks.

Finally, the work developed in (Schachinger & Kastner, 2016) aimed at defining a semantic framework to support operational building management and has been extended in (Petrushevski et al., 2017) to enable its use for advanced data analysis for the building operation.

Since the above-mentioned models only consider subsets of key domains relevant for FUSE-IT project (Ahvar et al., 2017; Tamani et al., 2018), none of the models, by itself, is sufficient to deal with the four key domains identified (Energy, Security, Facility, and ICT). Therefore, FUSE-IT project built a reference model based on the gathering of some of the ontologies identified above, making the extension as needed to reflect the necessary knowledge, aiming at "*describing metadata to have access to data and make them actionable in an interoperable context of interconnected systems protected by secured services*" (Ahvar et al., 2017; Tamani et al., 2018).

2.2. Semantic rule-based systems

Semantic rule-based systems are being applied successfully in various distinct areas (Muñoz López, Fernández, Coronado, & Iglesias, 2016; Petrushevski et al., 2017; Rygaev, 2017; Subirats et al., 2013; Tamani et al., 2018; Teymourian & Paschke, 2009; Yuce & Rezgui, 2017). This subsection overviews some of the most relevant that can be found in the literature.

The work presented in (Petrushevski et al., 2017) concentrates efforts in the semantic representation of building systems' information to support advanced data analytics algorithms trying to improve the building energy efficiency. It extends the model developed in (Schachinger & Kastner, 2016) and implements rules to find anomalies in the building monitoring data. The rules are expressed as ontology concepts and are a conjunction of conditions and conclusions. In this approach, functions must be developed to execute the rules when they are triggered, not taking advantage of semantic reasoners.

In (Muñoz López et al., 2016), the use of ontologies is proposed to describe automation and an architecture that provides contextual services. The architecture is flexible to interconnect devices with web services, and the use of semantic vocabulary enables semantic interoperability and expressiveness in the automation rules modelling and definition. The architecture is implemented in a smart office scenario. However, the rules are written in JavaScript Object Notation (JSON)[1] and are saved in a MongoDB[2] database. This means that they are not taking advantage of the semantic rules potential. The semantics is only being used for describing and share knowledge. In turn, (Yuce & Rezgui, 2017) proposes a semantic mapping process to define the most prominent variables to reduce energy gap in near real-time. It uses an artificial neural network (ANN) to learn the semantic mapping patterns. Then it is used as the cost function of an optimization tool (genetic algorithm based) that generates the energy saving rules in multiple objectives and constraints. An evaluation process has also been developed to evaluate the generated rules, their limits, and underlying variables. In this work, the rules are written in the Semantic Web Rule Language (SWRL),[3] a declarative semantic rule language that can be written directly in OWL[4] documents. The rules are then fired by a semantic reasoner when the user requests it.

A different approach to implement semantic rules is presented in (Teymourian & Paschke, 2009), using SPARQL Protocol and RDF Query Language (SPARQL),[5] a Resource Description Framework (RDF)[6] query language for semantic databases. This work describes research on semantic rule-based complex events processing for event-driven systems. The idea is to use semantic rules to identify complex events from simple event notifications. SPARQL was used to write queries to identify event patterns, resulting in a constructive view of the events in the system. This filter allows to detect complex events derived from already existing events combined with knowledge already gathered.

Also, in (Rygaev, 2017) is demonstrated a rule engine based on SPARQL queries for SemETAP, a semantic text analyser. In addition, it implements a forward chaining algorithm for existential rules. SemETAP aims to accomplish deep semantic analysis of natural language texts. It converts the original text into a graph of concepts from the ontology and the relations between those concepts. After, the semantic rules are applied to infer new knowledge from the implicit information, so that the semantic structure is able to answer questions. (Subirats et al., 2013) uses semantic rule-based reasoning in the management of functional rehabilitation processes. It presents a clinical decision support system using SWRL and semantic annotations to reason on rehabilitation processes. The system is able to provide personalized therapies and to deal with different characteristics of a rehabilitation scenario. The modelling categories are based on well-accepted rehabilitation notions and the rule-based reasoner is used for the representation of processes' semantics. The authors believe that the solution can be exported to other domains.

Finally, (Tamani et al., 2018) introduces a rule-based model for smart buildings supervision and management. It is a proof of concept to highlight the potential of using declarative rules to deal with building management and supervision. A fragment of the logical framework has been implemented in two real-world use cases in the context of FUSE-IT, using SWRL rules. It uses FUSE-IT core data model (Ahvar et al., 2017) to identify the concepts and relations of the FUSE-IT four key domains. The use cases are focused on anomaly detection, but the authors argue that it can be extended to diagnostic, healing, and recovery.

3. BRICKS

The “Building's Reasoning for Intelligent Control Knowledge-based System” is a context-aware semantic rule-based system considering context-based profiles for intelligent management of buildings' energy and security. It is one of the outcomes of projects with companies. The system is not open source, nor publicly available, since it has been integrated as part of commercial products.

BRICKS is not a conventional BMS. It is a rule-based system that can integrate different BMSs, or devices, data; providing a centralized interface for the building monitoring and alarms. It enables the definition of rules that integrate data from different BMSs, devices or web services (such as weather data) to trigger alarms or automatic control. It must be stressed that BRICKS itself does not operate the grid. It is a semantic rule-based system using devices/sensors data to trigger alarms and automatic control, as defined by the system administrator. The data read in the devices/sensors is translated into the semantic model at each time step, to allow the use of the semantic rules. Moreover, the grid operation, such as DR events and/or smart contracts between the aggregator and the aggregated player, is performed by other modules, that work with the raw data read instead of using semantic models. However, it is being considered the update of these algorithms to accept the use of semantic data as input and output, enabling semantic interoperability besides the already existing syntactic one. For the moment,

[1] http://www.json.org.
[2] https://www.mongodb.com/.
[3] https://www.w3.org/Submission/SWRL/.
[4] https://www.w3.org/OWL/.
[5] https://www.w3.org/TR/rdf-sparql-query/.
[6] https://www.w3.org/RDF/.

the system can communicate using both Modbus TCP/IP[7] and HTTP REST[8] protocols, since those are the ones available in the researchers' laboratory for testing. It can be integrated with existing SCADA systems, such as the system introduced in (Figueiredo & Costa, 2012), as well as with IoT smart appliances. However, to be able to communicate with other higher-level facility/energy management systems such as the ones presented in (Blaauwbroek et al., 2015; Brusco et al., 2014; Calvillo et al., 2016; Jajac et al., 2009), these should provide REST communication interfaces to enable BRICKS to interact with them.

Building automation and management has its pros and cons (Somfy, 2019; ThinkEnergy, 2019). Taking a closer look at the cons, the most pointed out are: equipment cost; installation cost; complexity; and compatibility. Besides the cons above, when talking about rule-based systems for intelligent building management it is important to keep in mind that the implementation of the rules at the automation level depend on the devices used, their configuration, communications protocols, data types, etc. Meaning that every time the automation system needs to be updated/upgraded, by adding new devices or removing/changing existing ones, it will be necessary to rewrite the rules accordingly and recompile the system.

BRICKS tries to overcome these issues. Being a semantic rule-based system, BRICKS:

- uses semantic models, which enrich the information gathered by the different devices;
- is abstracted from the ontologies used and respective rules. In this way the knowledge model and/or the rules may change without the need to reprogram the system;
- does not depend on the installed devices nor communication protocols, since it implements the rules at a higher level, the software level.

Ontologies add semantic meaning to data, making it understandable by both humans and machines. The use of semantics enables: the sharing of a common understanding of the exchanged information; to reuse the domain knowledge; to make domain assumptions, and to separate domain from operational knowledge. In computer science, ontologies are used for: communications between computational systems and/or humans; knowledge reuse and organization; and computational inference (Gruninger & Lee, 2002). Computational inference provides deductive capabilities to add new knowledge to the already existing. Combining computational inference with a rule language improves the deductive reasoning capabilities of reasoners.

BRICKS translates the knowledge implicit in the devices data to the semantic model. Besides the real-time data, BRICKS also uses REST services of different algorithms to determine the correct profile at each time for a given context, which are also converted to ontology instances. The result of these translations are the ontology individuals or instances. Fig. 1 illustrates this scenario.

In the left side of Fig. 1 are identified different data sources that may be translated to semantic individuals. The devices' measurement data may be read from online databases or directly from the smart devices through REST requests. In the case of dummy devices, these may be read from a Programmable Logic Controller (PLC) to which they are connected to Fig. 1. It can be noticed that the rules are added in the semantic model. BRICKS allows the use of both SWRL rules and SPARQL queries to trigger the alarms and/or automatic control. The rules are also an input to the system.

BRICKS works in real-time depending on the time-step defined by the user. The devices' measurements, the contexts and context-based profiles are translated to the semantic model resulting in ontology individuals in the knowledge base. Then, the rules are applied by the rule engine, formalizing the system's output, as illustrated by Fig. 2.

The readings may be achieved by two means: using Modbus protocol; or REST web services. Ideally, BRICKS would use REST requests only for the devices real-time data, being agnostic to any hardware communication protocol. However, considering the use of dummy devices connected to a PLC, the authors decided to leave this option open once Modbus protocol is already available in the laboratory building of the authors' research centre. In fact, Modbus protocol had a large widespread use in communications at industrial level, which has motivated the authors to proceed with using it.

Currently, Open Platform Communications - Unified Architecture (OPC-UA) (CAS, 2010; OPC Foundation, 2018) is pointed as the most promising protocol to standardize communications in scopes such as Industry 4.0, IoT or SGs, however its specification is not straight forward, which results in incomplete implementations. In order to include OPC-UA protocol in BRICKS, a semantic converter similar to the one developed for the Modbus protocol is required.

The context-based profiles are achieved by means of a REST request to a web service. The service returns the adequate profile for each device in the current timestamp. The context is reached through a clustering algorithm (Madureira, Pinto, Fernandes, & Vale, 2017). Using this context-based profiles algorithm, the system is able to respond to the costumers' behaviour since it uses historic data from the past to adjust the profiles over time in different contexts, such as: the current season, month, weekday or weekend, and time of day. In this way, the user does not have the need to configure a custom profile, nor to use a predefined one, as the system will be able to learn from the costumer's behaviour. Given the real-time data and the context-based profiles, the rules are validated by matching the devices readings against the profiles. BRICKS starts by executing SWRL rules first, if defined by the user as input. After, it runs rules written as SPARQL queries. It is the responsibility of the system administrator to write the rules properly, to ensure the desired sequence of rules. Given the order of execution of the inference engines, a rule written in SWRL cannot wait for a result of a rule written in SPARQL, although the opposite is valid. The idea of executing SWRL rules first relates to the fact that the SWRL engine infers implicit knowledge from the available data which may be interesting to consider when executing SPARQL rules.

BRICKS output can be a set of alarms, notifications, or control actions to take. In addition, monitoring values are always output at each time-step to be presented by the BMS user interface as it is possible to observe in the example Fig. 3. In the case of automation control, this may also be achieved by means of REST requests previously defined in the set of rules.

At each time step BRICKS: (i) performs the devices readings, updating the semantic data model; (ii) requests the context-based profiles of each device to the predefined REST service; (iii) executes the semantic rules to validate the devices' data read with the respective profiles; and (iv) instantiates the "monitoring" and "alarms" outputs. The monitoring output is useful for BMS user interfaces. In case of any alarm triggered by a rule, BRICKS will act accordingly, i.e., depending on the rule. Rules can provide notification in the user interface, send an email, or apply automatic control.

BRICKS is being developed as a REST web service, so it can be used as a module by different BMSs, depending only on the configuration for each system. It enables access through a web browser. Secure communications over public networks must be assured, so, the use of Hyper Text Transfer Protocol Secure (HTTPS) with Transport Layer Security (TLS) protocol is suggested (Herberg, Mashima, Jetcheva, & Mirzazad-Barijough, 2014; Mohan & Mashima, 2014). The use of Firewalls, Intrusion Detection Systems (IDS), and Intrusion Prevention Systems (IPS) must also be considered. In the case of critical buildings, besides the above security measures, a secure Virtual Private Network (VPN) should be used to guarantee the network cyber-security (Actility Veolia, 2019; GTM, 2014).

Table 1 summarizes BRICKS' configuration and operation steps.

[7] http://www.modbus.org/.

[8] https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest.

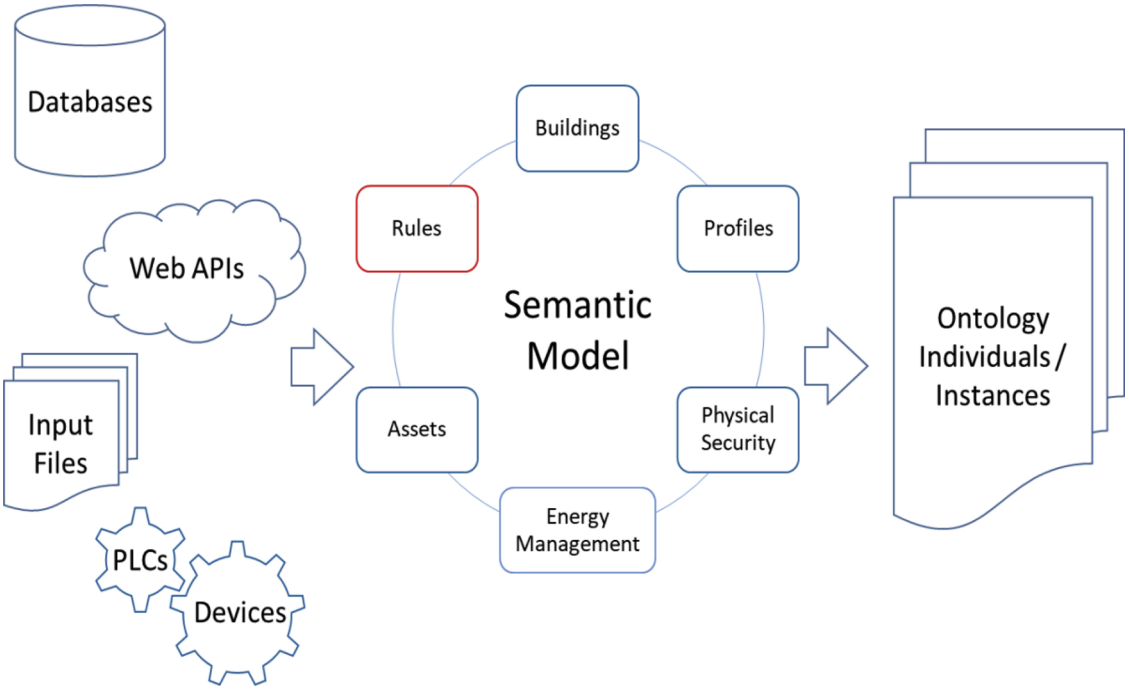

Fig. 1. BRICKS translation from raw data to the semantic model.

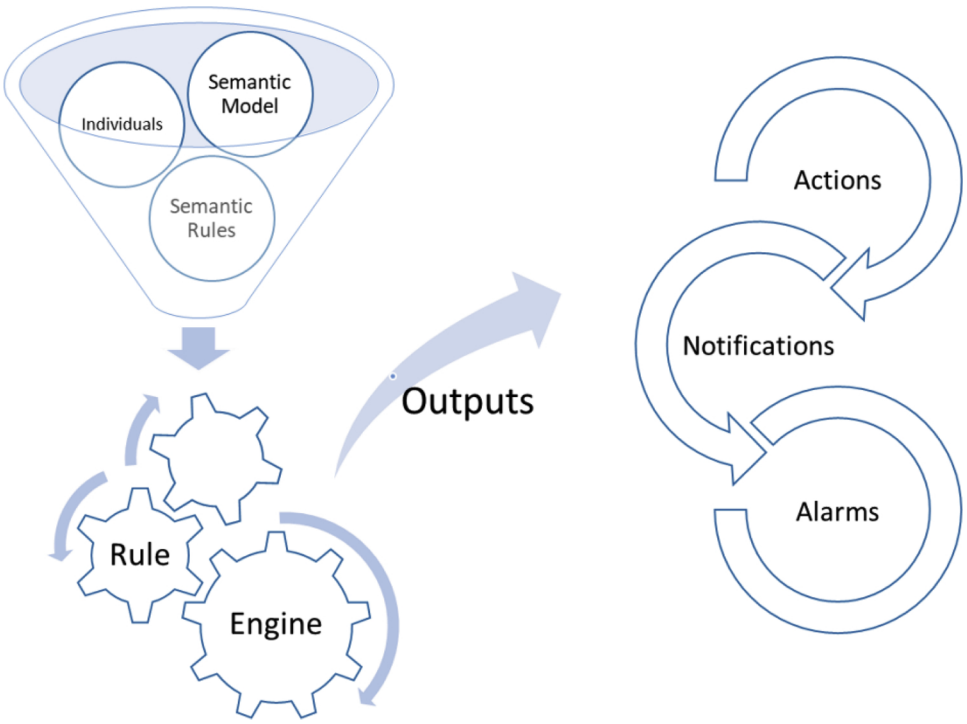

Fig. 2. BRICKS rule engine.

Fig. 4 introduces the overall framework flowchart of BRICKS execution explained above.

The main difficulty found in BRICKS development was to keep the system responsive when using both SWRL and SPARQL rules. SWRL is a semantic rule language written in OWL and directly interpreted from OWL reasoners, while SPARQL is a RDF query language that can only be interpreted by a SPARQL engine. This means that each type of rule must be executed by a different inference engine. Input/output streams have been implemented which do this process fast enough.

To implement BRICKS in a new site, the building can already have available smart appliances or PLCs to communicate with the system. It can be integrated with existing SCADA systems already available in the building. BRICKS will only be configured accordingly, ensuring the correct communication with the smart appliances and/or the PLCs. The time needed to configure BRICKS depends on the number of resources to be configured, and the rules that the administrator wants to write for triggering alarms and automatic control. Currently, this configuration is made manually in text files, which is a costly job. The configuration of the scenario present in the case study section in the present paper took about one week (5 business days). However, it is already in study the

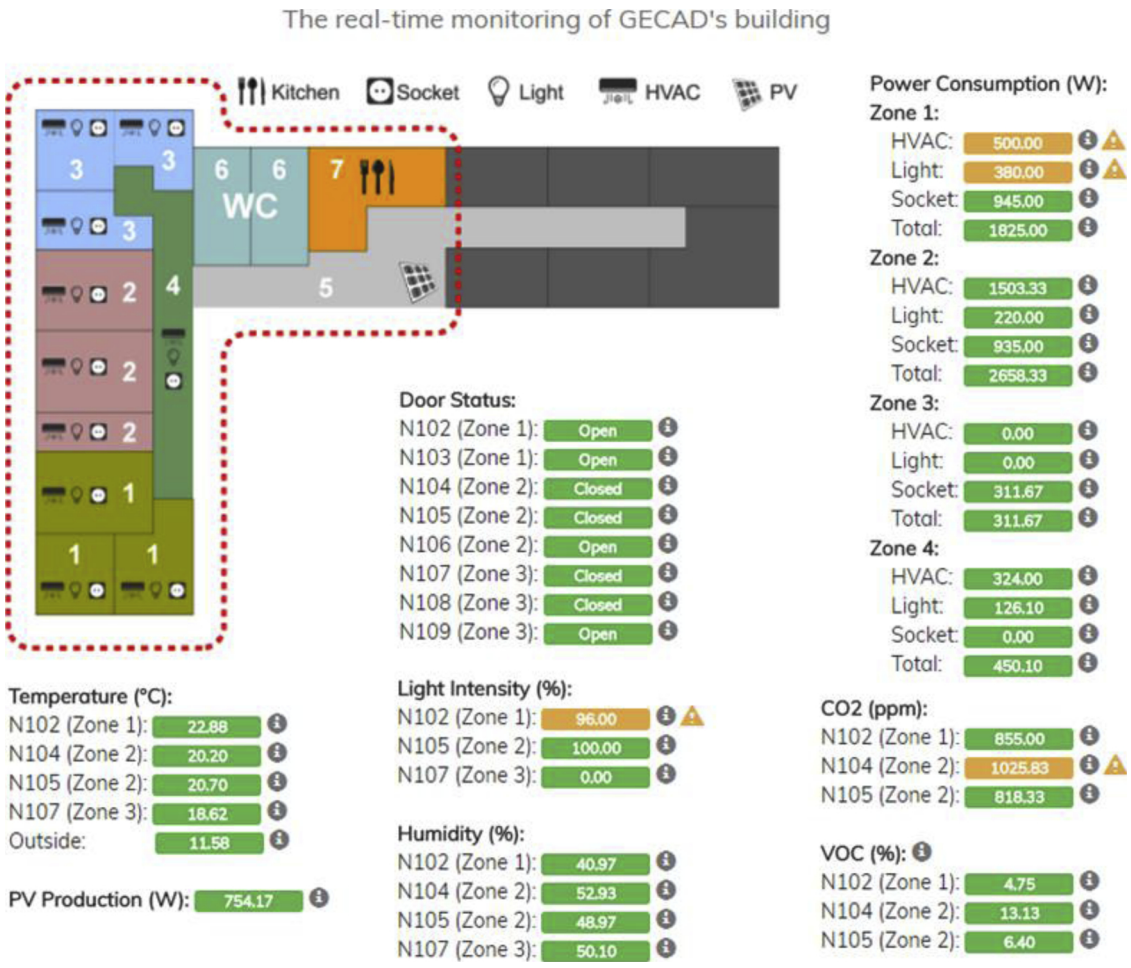

Fig. 3. Example of BMS user interface using BRICKS.

Table 1
BRICKS's steps.

Configuration	1. define necessary semantic model(s); 2. instantiate building and devices data, describing how to read the devices measurements and/or control them (e.g. Modbus or REST Service); 3. input/write SPARQL queries to read devices semantic data from the knowledge base; 4. input/write SPARQL constructs to update the knowledge base with the devices' measurements and context-profiles; 5. input/write rules (SWRL, SPARQL) to trigger alarms and automatic control
Operation *(at each time step)*	1. get context-profile for each device's measurement; 2. read devices' measurements (by Modbus or REST Service); 3. translate and include measurements in the knowledge base; 4. execute SWRL/SPARQL rules; 5. execute automatic control (if a given rule is triggered).

development of an intuitive graphical user interface to abstract the user from all the semantic configuration, i.e.: (i) building definition; (ii) devices instances; (iii) measurements individuals; (iv) rules definition; etc.

Regarding its maintenance, in a typical SCADA system, whenever a new device or rule is added or updated, the system must be reprogrammed and recompiled in order to function correctly. However, in BRICKS this will not be necessary, since the semantic model is loosely coupled from the software. BRICKS will only need to be reconfigured accordingly, saving time when compared to the typical SCADA.

4. Case study

The present section presents a case study in which a BRICKS configuration is illustrated. It is inspired by (GREEDi, 2017), using GECAD's facilities. This case study is only focused on the energy management scenario.

4.1. Scenario

The case study scenario considers a business building with three independent offices of three different companies. The building is managed by an administrator who also aggregates the different tenants for energy efficiency and DR programs. BRICKS is deployed for each entity (i.e., the administrator and the three offices). Each platform runs an instance of BRICKS. The scenario has been deployed in the left-side of GECAD's N building and uses real data gathered from the different devices through four PLCs (Khorram, Abrishambaf, Faria, & Vale, 2018). Figure 55 illustrates the left-side of GECAD's N building in the perspective of the building administrator.

The building is divided into 4 zones. Zones 1–3 represent the three offices of different companies; zone 4 is the common area. The building's administrator manages the common area and knows the information about each office's total consumption, for energy efficiency

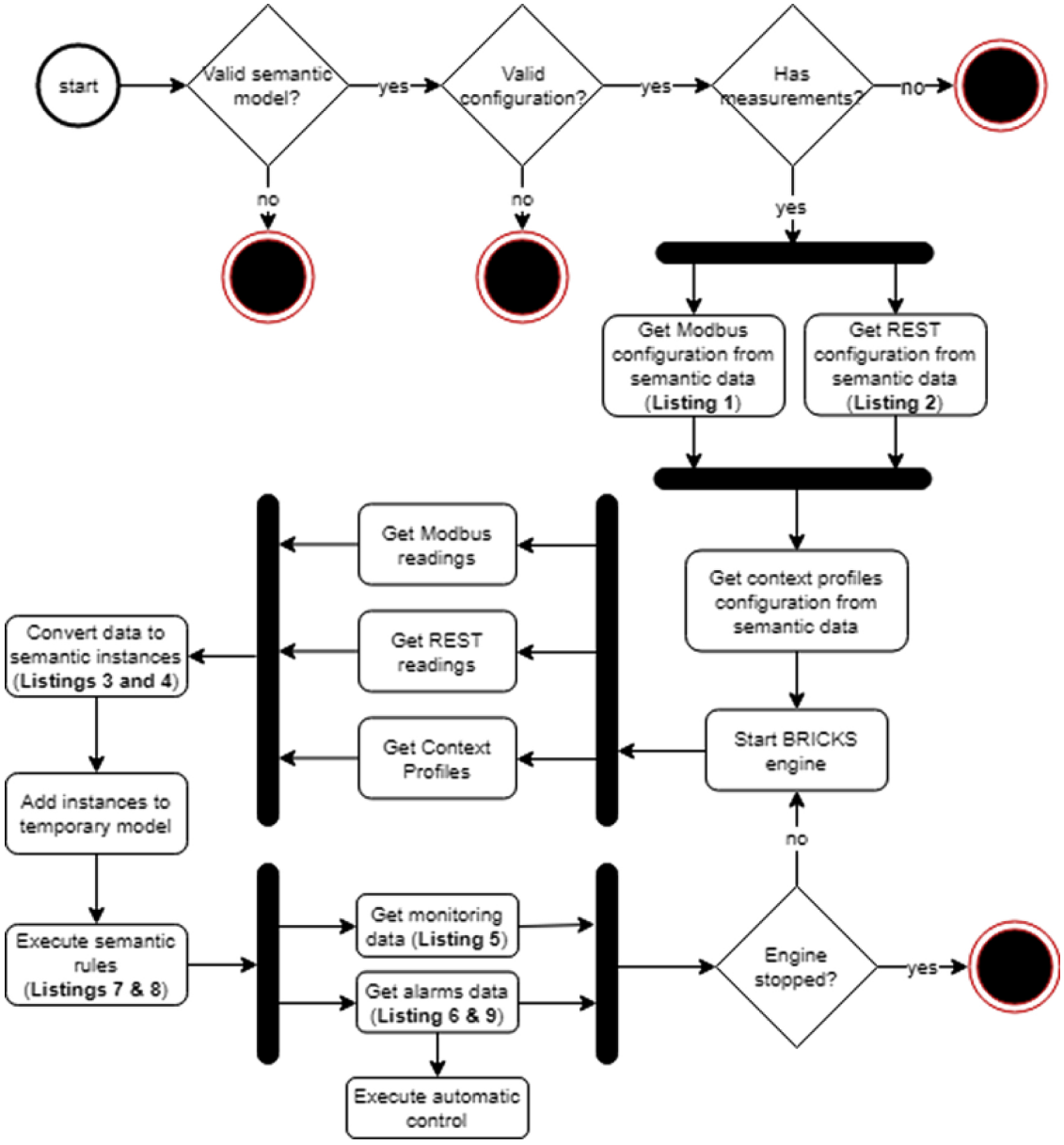

Fig. 4. Overall framework flowchart.

Table 2
Inputs installed.

Input	Zone 1	Zone 2	Zone 3	Zone 4
Photovoltaic (PV) panels with 7.5 kW rated power	–	–	–	√
Light intensity sensor	1	1	1	1
Indoor temperature sensor	1	2	1	1
Presence sensor	4	4	1	1
Humidity sensor	1	2	1	–
Outside temperature sensor	–	–	–	1
Door locker (actuator/sensor)	3	3	3	–
Air-conditioning device (I/O control + energy monitoring)	3	3	3	1
Fluorescent lamps controlled by DALI[a]	6	4	5	4
Energy analyser for the lights' total consumption	1	1	1	1
Energy analyser for the sockets' total consumption	1	1	1	1
Energy analyser for the total consumption	1	1	1	1

[a] Digital Addressable Lighting Interface: http://www.dali-ag.org.

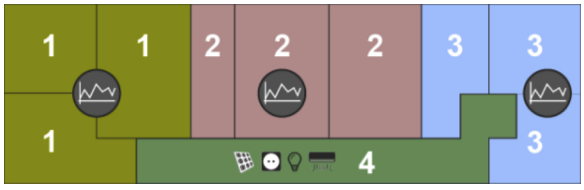

Fig. 5. Plant of the left side of building N of GECAD.

purposes and participation in DR programs. Table 2 shows the inputs installed in each zone, illustrated in Fig. 5 in khaki, pink, light blue, and dark green, respectively for Zone 1, Zone 2, Zone 3, and Zone 4. The locations of the devices, sensors and PLCs are undisclosed for security and privacy reasons.

The air-conditioning control was achieved by using an infra-red LED to send the control signals. It was used an Arduino Mega 2560 with Ethernet Shield and the control is possible by HTTP requests. The Arduino is able to receive HTTP GET requests that indicate the temperature and operation mode. This was a simpler approach to provide

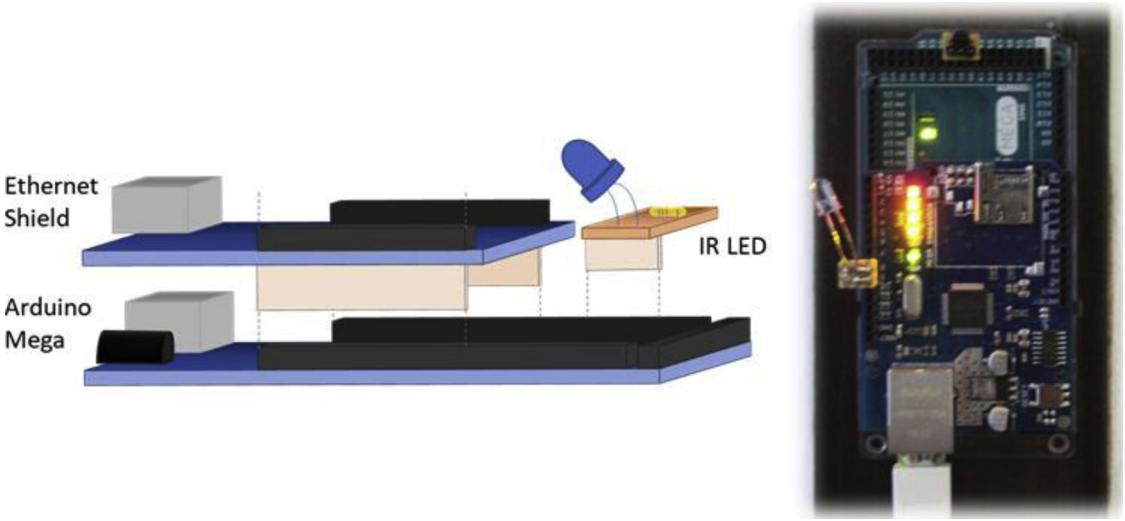

Fig. 6. Arduino with Ethernet Shield assembly.

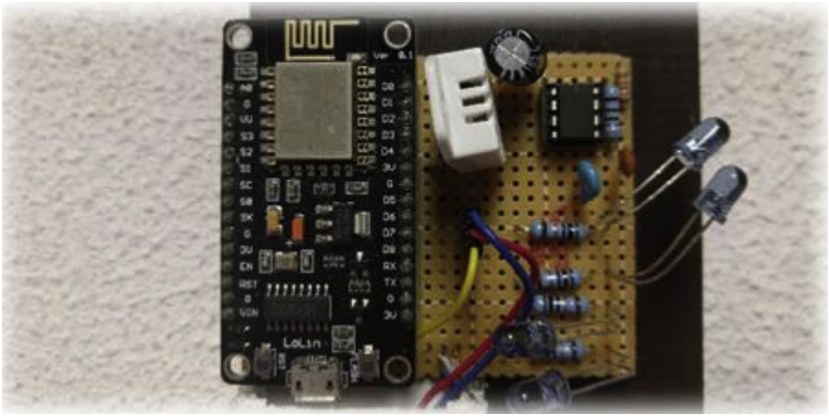

Fig. 7. Implementation of the air-conditioning controller using a NodeMCU module.

control over the air-conditioners and integration with BRICKS. Fig. 6 shows the assembly of the components.

It was also built a solution using the module NodeMCU that integrated Wi-Fi communication. In the NodeMCU approach it was integrated a DHT22 for temperature and humidity sensor and MQTT protocol. The MQTT protocol is used to publish the sensor data and subscribe to BRICKS control orders for the air-conditioning units. Fig. 7 shows the implementation of the air-conditioning controller using a NodeMCU module.

4.2. Configuration

The first step in BRICKS configuration is to define which semantic models are going to be used to describe the data models of each domain. To describe buildings and devices we defined an ontology.[9] To communicate with Modbus protocol, we added the GECAD's *PLC ontology*.[10] For REST services, we have also used a GECAD's preliminary *Web Services' ontology*.[11] Finally, for the context-based profiles, BRICKS uses by default the *Context-based Rules Matching Profile (CRMP) ontology*.[12] The next step is to instantiate the building in each BRICKS system accordingly, ensuring that the manager of each zone only has visibility of the device's data of its responsibility. The administrator configures semantically the building identifying the 4 zones, and the devices installed in the common area, namely zone 4. Each zone manager must define afterwards the devices installed in its office. When defining the devices, each manager must also define semantically how to read/write in it, using in this case the *PLC* or *Web Services* ontologies (since some devices accepts REST requests). In the specific case of the building administrator, he requires the use of a meteorologist web service to get the solar radiation value. This is important to check if the PV panels are working as expected, since PV energy generation depends on solar radiation.

Afterwards, the user must input the SWRL/SPARQL rules. Another important step to take is to write SPARQL queries to: (i) read the semantic configurations and instantiate the PLC/Web Services requests; and, after each time-step, (ii) translate the data from the different readings (i.e. Modbus or REST service) into the semantic model. In the former case, in order to ensure the correct functioning of the system and avoiding reprogramming it, the queries must output a JSON string respecting a predefined schema, for each case (i.e. Modbus/REST service). With those JSON objects BRICKS will be able to configure both Modbus and Web Services access and make readings requests at each iteration. Examples of both JSON objects are shown in Listing 1 and Listing 2. In the latter case, a JSON file must be filled to map the read values with the correct ontology instances. For such, the SPARQL construct files must use tags, that must be identified in the JSON file in order to replace them with the corresponding read values. An example

[9] Publicly available at: http://www.gecad.isep.ipp.pt/ieso/greedi.ttl.
[10] Publicly available at: http://www.gecad.isep.ipp.pt/ieso/plc.ttl.
[11] Publicly available at: http://www.gecad.isep.ipp.pt/ieso/ws.ttl.
[12] Publicly available at: http://www.gecad.isep.ipp.pt/ieso/crmp.ttl.

of this file is presented in Listing 3, and the corresponding SPARQL construct in Listing 4.

Listing 1. Example of Modbus JSON object.

```
{
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#photovoltaic-rooftop",
    "plc": {
        "id": 1,
        "name": "SAIA PCD",
        "ip": "192.168.2.25",
        "port": 502,
        "unitId": 2,
        "protocol": "TCP_IP",
        "readingTimeStep": 100,
        "resource": {
            "id": 1,
            "name": "Photovoltaic panels",
            "type": "GENERATOR",
            "status": "PRODUCTION",
            "register": {
                "number": 100,
                "name": "PV instant power generation",
                "type": "INTEGER",
                "count": 2,
                "dataArea": "SINGLE_REGISTER",
                "writable": false,
                "dataUnit": "W"
            }
        }
    }
}
```

The example of Listing 1 shows an example of output of a Modbus SPARQL query to the knowledge base, in order to BRICKS communicates correctly with the PLC.

Listing 2. Example of Web Services JSON object.

```
{
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#solar-radiation",
    "ws": {
        "request": {
            "url": "https://meteo.isep.ipp.pt/ramdisk/json/realtimeparameterlist.txt",
            "method": "GET"
        },
        "response": {
            "statusCode": 200,
            "contentType": "JSON",
            "parameter": "solarRad"
        }
    }
}
```

Likewise, Listing 2 presents an example of the output a SPARQL query for Web Services must return.

Listing 3. Example of JSON construct tags file.

```
[
    {
        "file": "construct.sparql",
        "readings": [{
            "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#photovoltaic-rooftop",
            "readers": [{
                "reader": "modbus:100",
                "tag": "[pv-value]"
            }]
        }, {
            "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#solar-radiation",
            "readers": [{
                "reader": "ws",
                "tag": "[solar-radiation-value]"
            }]
        }]
    }
]
```

Listing 3 exemplifies how the JSON construct tags file must be written. The JSON file holds a JSON array where each position is a JSON object identifying: (i) the SPARQL construct file where the tags must be replaced ("*file*" key); and (ii) an array of "*readings*" to be considered when replacing the tags with the respective values. The "*readings*" array is composed of JSON objects identifying: (i) the ontology individual ("*ind*" key) that describes the device; and (ii) an array of "*readers*" for that device. Each "*readers*" array position is a JSON object with: (i) the "*reader*", identifying the type of reader; and (ii) the "*tag*" (in blue) to be replaced by the corresponding value in the SPARQL

construct file. In case the "*reader*" is of type ”modbus” it must also have a suffix in the form ”:{modbus-register-number}”, such as in the example above: *"reader": "modbus:*100*"*.

Listing 4. Example of SPARQL construct file.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX plc: <http://localhost/ieso/plc.ttl#>
PREFIX greedi: <http://localhost/ieso/greedi.ttl#>
PREFIX : <http://localhost/ieso/i/greedi-demo-indiv.ttl#>

CONSTRUCT {
   # Read Instant #
   :read-instant rdf:type owl:NamedIndividual , greedi:Instant ;
       greedi:instant ?instant ;
       greedi:timeInMillis ?timeInMillis .

   # Photovoltaic Rooftop #
   :photovoltaic-rooftop-gen rdf:type owl:NamedIndividual , greedi:PowerMeasurement ;
       greedi:id "photovoltaic-rooftop-gen"^^rdfs:Literal ;
       greedi:unit ?pvUnit ;
       greedi:value "[pv-value]"^^xsd:double ;
       greedi:time :read-instant .
   :photovoltaic-rooftop greedi:produces :photovoltaic-rooftop-gen .

   # Solar Radiation ISEP #
   :solar-radiation-read rdf:type owl:NamedIndividual , greedi:SolarRadiationMeasurement ;
       greedi:id "solar-radiation-read"^^rdfs:Literal ;
       greedi:value "[solar-radiation-value]"^^xsd:double ;
       greedi:unit "W/m^2"^^rdfs:Literal ;
       greedi:time :read-instant .
   :solar-radiation greedi:measures :solar-radiation-read .

 } WHERE {

   # Read Instant #
   BIND( NOW() AS ?instant )
   BIND( HOURS( ?instant ) AS ?hour )
   BIND( MINUTES( ?instant ) AS ?min )
   BIND( SECONDS( ?instant ) AS ?sec )
   BIND( ( ( ( ?hour * 3600000 ) + ( ?min * 60000 ) ) + ( ?sec * 1000 ) ) AS ?tmt )
   BIND( STRBEFORE( STR( ?tmt ), "." ) AS ?temp )
   BIND( STRDT( ?temp, xsd:integer ) AS ?timeInMillis )

   # Photovoltaic Rooftop Unit #
   :photovoltaic-rooftop greedi:readThrough ?plc .
   ?plc plc:hasResource ?res .
   ?res plc:hasRegister ?reg .
   ?reg plc:dataUnit ?pvUnit .
 }
```

Listing 4 introduces an example of SPARQL construct file where the tags to be replaced by BRICKS are highlighted in blue, accordingly to the example exposed in Listing 3.

Finally, the user needs to set up the REST service that will return BRICKS the context-based profiles. This way BRICKS is flexible to change the service at any time. This configuration is also made using the *Web Services' ontology*. In order to BRICKS works correctly the REST service output must comply with a predefined JSON schema. Given this, BRICKS is able to translate the contextual profiles to the ontology, in the same way as the in readings.

4.3. Results

This subsection analyses BRICKS's outputs. The output is updated at every time-step, as explained before. For illustrative purposes a single iteration is analysed, where both ”monitoring” and ”alarms” outputs will be detailed.

It must be stressed that BRICKS is able to execute each time step in less than 30 s. The system is intended to be continuously in operation. It is error tolerant since if a reading fails at an iteration (e.g., because of hardware communication failure), the system ignores it in order to avoid a runtime exception that may cause the system to crash.

Table 3
Zone 4 monitoring and alarms results per device.

	Scenario 1		Scenario 2		Scenario 3		Scenario4	
	Before	**After**	**Before**	**After**	**Before**	**After**	**Before**	**After**
Device \ Time	10:00	10:05	10:00	10:05	22:00	22:05	22:00	22:05
PV generation (W)	880.00	880.00	0.00	0.00	0.00	0.00	0.00	0.00
Solar Radiation (W/m^2)	144	144	144	144	0	0	50	50
Outside temp. (ºC)	13.50	13.50	13.50	13.50	8.55	8.55	8.55	8.55
Indoor temp. (ºC)	18.50	18.50	14.00	17.50	28.10	20.10	12.50	12.50
Air-conditioning	On	On	Off	On	On	Off	Off	Off
Presence sensor	1	1	1	1	1	1	0	0
Lights status	Off	Off	Off	On	Off	On	On	Off
Light intensity (%)	50	50	20	50	0	50	50	0
Air-conditioning power (W)	1560.00	1560.00	0.00	1560.00	1560.00	1560.00	0.00	0.00
Lights power (W)	0.00	0.00	0.00	210.00	0.00	210.00	210.00	0.00
Sockets power (W)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Total power (W)	1560.00	1560.00	1560.00	1770.00	1560.00	1770.00	210.00	0.00

4.3.1. Monitoring output

As explained above, the monitoring output returns real time readings information to the BMS user interface. It identifies the device and its measurement read. Some devices have more than one type of measurement, as shown below. Listing 5 shows an example of the monitoring output for zone 4.

Listing 5. Example of monitoring output.


```
[
  {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#ac-zone4",
    "measurement": {
      "unit": "W",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#ac-zone4-cons"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#ac-zone4",
    "measurement": {
      "value": false,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#ac-zone4-on-off"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light-intensity-zone4",
    "measurement": {
      "unit": "%",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light-intensity-zone4-read"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light1-zone4",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light1-zone4-on-off-range"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light2-zone4",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light2-zone4-on-off-range"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light3-zone4",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light3-zone4-on-off-range"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light4-zone4",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#light4-zone4-on-off-range"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#lights-zone4",
    "measurement": {
      "unit": "W",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#lights-zone4-cons"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#outside-temperature",
    "measurement": {
      "unit": "º C",
      "value": 18.7,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#outside-temperature-read"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#photovoltaic-rooftop",
    "measurement": {
      "unit": "W",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#photovoltaic-rooftop-gen"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#sockets-zone4",
    "measurement": {
      "unit": "W",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#sockets-zone4-cons"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#solar-radiation",
    "measurement": {
      "unit": "W/m^2",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#solar-radiation-read"
    }
  }, {
    "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#total-zone4",
    "measurement": {
      "unit": "W",
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-indiv.ttl#total-zone4-cons"
    }
  }
]
```

As it can be seen in the first lines of Listing 5, the first two measurements are for the air-conditioning of zone 4. The first one is measuring its power consumption; while the second is reading its on/off state.

The monitoring output is a JSON array, where each element is a JSON object identifying: (i) the device ("*ind*" field); and (ii) the "*measurement*" in the ontology. The "*measurement*" is a JSON object consisting of: (i) the measurement individual identification ("*ind*" field); (ii) its "*value*"; and, when applicable, (iii) its "*unit*".

4.3.2. Alarms output

Regarding the alarms output, they depend on the rules defined by the manager. The output is set by querying the knowledge base in search of action individuals (*crmp:Action* class from *CRMP* ontology) and respective devices, measurements, and profiles. A device may have more than one measurement, and each measurement has its unique profile.

When using SWRL rules, due to the language limitations, *crmp:Action* individuals must be set before starting the system, since SWRL is not able to instantiate classes that are not present in the left-side of the rule. Otherwise the rule throws an exception. For this reason, the alarms output is always instantiated, so when a rule is triggered, the "*action*" field is available. The alarms output is also useful to update the BMS's user interface accordingly. Listing 6 presents a snippet of an alarm's output example for zone 4, with these two cases.

Listing 6. Snippet of alarms output.

```
[
  {
    "profile": {
      "profileUpperBound": 500,
      "type": "range",
      "profileLowerBound": 0
    },
    "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#ac-zone4",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#ac-zone4-cons"
    }
  }, {
    "profile": {
      "profileUpperBound": 7000,
      "type": "range",
      "profileLowerBound": 150
    },
    "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#photovoltaic-rooftop",
    "measurement": {
      "value": 0,
      "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#photovoltaic-rooftop-gen"
    },
    "action": {
      "alertLevel": "warning",
      "message": "The photovoltaic generation is lower than expected.",
      "commandURL": "http://localhost/BRICKSWS/sendmail"
    }
  }
  ...
]
```

Observing Listing 6 it can be seen that, in the case of the air-conditioning, no action is required. Moreover, the PV generation is lower than expected given the context-based profile, and available solar radiation. In this case, an action must be taken (highlighted in blue text). According to the rule defined, the zone manager will receive an email alerting that the PV is not functioning correctly.

Analysing the alarms output JSON array, each element is a JSON object identifying: (i) the device individual ("*ind*" field); (ii) its "*measurement*"; (iii) context-based "*profile*"; and, when a rule is triggered, (iv) the "*action*" to take.

The SWRL rule that triggers the action seen above is shown in Listing 7. It searches for a generator device' measurement and a solar radiation measurement of the same instant, and also for the lower bound of the device's profile to check if the device's generation value is lower than the profile's lower bound while the solar radiation is higher than zero. If both conditions are true the rule will be triggered, instantiating the *crmp:alertLevel*, *crmp:message*, and *crmp:commandURL* properties of *:a-photovoltaic-generation* (*crmp:Action*).

Listing 7. SWRL rule that triggers alarm.

```
greedi:GeneratorDevice(?d) ^ greedi:measures(?d, ?pm) ^ greedi:PowerMeasurement(?pm)
^ greedi:time(?pm, ?t) ^ greedi:value(?pm, ?pv) ^ crmp:RangeProfile(?rp)
^ crmp:individual(?rp, ?d) ^ crmp:individual(?rp, ?pm) ^ crmp:profileLowerBound(?rp, ?lb)
^ swrlb:lessThan(?pv, ?lb) ^ greedi:SolarRadiationMeasurement(?srm) ^ greedi:time(?srm, ?t)
^ greedi:value(?srm, ?srv) ^ swrlb:greaterThan(?srv, 0) ^ crmp:Action(?a)
^ crmp:individual(?a, ?d) ^ crmp:individual(?a, ?pm)
 ->
crmp:alertLevel(?a, "warning")
^ crmp:message(?a, "The photovoltaic generation is lower than expected.")
^ crmp:commandURL(?a, "https://localhost/BRICKSWS/sendmail")
```

Finally, Listing 8 introduces a rule to control the air-conditioning of zone 4 when someone forgets to shut it down after a working day. It searches for load devices with boolean measurements (such as on/off) and the value of the devices' boolean profiles to validate if the values are different. In case they are, it checks if the boolean measurement is different from "false", and if true it triggers the rule, constructing the values for *crmp:message* and *crmp:commandURL* properties.

Listing 8. SWRL rule that triggers automatic control.

```
greedi:LoadDevice(?d) ^ greedi:measures(?d, ?bm) ^ greedi:BooleanMeasurement(?bm)
^ greedi:value(?bm, ?bmv) ^ crmp:BooleanProfile(?bp) ^ crmp:individual(?bp, ?d)
^ crmp:individual(?bp, ?bm) ^ crmp:profileValue(?bp, ?bpv) ^ swrlb:booleanNot(?bmv, ?bpv)
^ swrlb:booleanNot(?bmv, false) ^ crmp:Action(?a) ^ crmp:individual(?a, ?d)
^ crmp:individual(?a, ?bm) ^ greedi:name(?d, ?dn) ^ swrlb:stringConcat(?tmp, "The ", ?dn)
^ swrlb:stringConcat(?msg, ?tmp, " should be shut down. Shutting it down...")
^ greedi:id(?d, ?did)
^ swrlb:stringConcat(?cmd, "https://localhost/BRICKSWS/shutdown/", ?did)
 ->
crmp:alertLevel(?a, "notice") ^ crmp:message(?a, ?msg) ^ crmp:commandURL(?a, ?cmd)
```

This rule being triggered results in the alarm output presented in Listing 9.

```
[
  {
    "profile": {
      "type": "range",
      "profileValue": false
    },
    "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#ac-zone4",
    "measurement": {
      "value": true,
      "ind": "http://localhost/ieso/i/greedi-demo-vtn-indiv.ttl#ac-zone4-on-off"
    },
    "action": {
      "alertLevel": "notice",
      "message": "The air-conditioning Zone 4 should be shut down. Shutting it down...",
      "commandURL": "https://localhost/BRICKSWS/shutdown/ac-zone4"
    }
  }
  ...
]
```

Listing 9. Alarm output example for SWRL rule of Listing 8.

Table 33 illustrates Zone 4 devices' alarms results in four different possible scenarios. All scenarios are for the same day, i.e., January 28th, 2019. Scenarios 1 and 2 demonstrate two possible occurrences around 10:00 o'clock, while scenarios 3 and 4 demonstrate other two possible situations around 22:00 o'clock. The light green cells identify monitoring data that do not trigger alarms. The yellow ones identify *notice* alarms. And the orange cells identify *warning* alarms.

Analysing Table 3, scenario 1 demonstrates an example where no alarm is triggered, and all monitoring values are within the expected intervals. Whereas scenario 2 triggers three different alarms: i) the PV generation is 0 (zero) while there's solar radiation; (ii) the A/C is off, and the area temperature is lower than previously defined by the system's administrator; and (iii) the light intensity drops to a lower than acceptable percentage. In the first case, the PV was not producing energy due to tree leaves that were covering it. Regarding the A/C, some user shut it down inappropriately, and the system was able to turn it on automatically, setting the temperature according to the users' preferences. In the case of the light intensity, a cloud covered sun light coming from the windows and the system turned the lights on to respect the light intensity preferences defined by the system's administrator.

Looking at scenario 3, two alarms are triggered: i) the area temperature is higher than the acceptable maximum value; and ii) the light intensity is too low, considering that the presence sensor identifies some people in the building. The first alarm automatically corrects the A/C temperature to an acceptable value. The second one, given the presence of some users in the building, automatically turns the lights on and adjusts the light intensity according to the users' preferences. It should be noticed that the system also produces a notification in the presence sensor, since it is not expected to have someone in the building at that time.

Finally, scenario 4 triggers 1 alarm and 1 notification. The notification warns the system's administrator that the solar radiation value is incorrect given the time of the day. Since this value comes from an online service, the system administrator can simply ignore it and try to find a new service to replace the current one. The alarm identifies that the lights are still on when nobody is in the building (according to the presence sensor), and in this case the system is able to turn the light off automatically.

5. Conclusions

Nowadays, building automation and BMS are more often used in new constructions and to improve energy efficiency in older ones, through retrofitting. There are several solutions for the most varied domains, such as: user comfort, building security, energy efficiency, among others. However, there are not yet in the market affordable solutions which are able to aggregate different services in the same centralized platform, in an interoperable way.

This paper presented BRICKS, a semantic rule-based system considering context-based profiles for intelligent building energy and security management. BRICKS is made available as a REST service. It tries to overcome the main building automation and management issues, such as: high costs, installation, complexity, and compatibility. Being a semantic rule-based system, it is abstracted from the semantic model and rules, as they are considered inputs for the system, and in this way the model and rules may change without the need of reprogramming the system. The context-based profiles are also an input to the system as they are defined externally. BRICKS matches the rules with the real-time measurements and the specific profiles. Being implemented in the software level, it does not depend on the used devices nor communication protocols. It uses a semantic model based on contexts for the validation of the rules according to the profiles to formalize the system's output.

BRICKS was designed to be an intelligent and secure system integrating both energy management and secure communications, while promoting interoperability between all services. It can be reused completely or partially in other buildings or parts of buildings for which the same semantic models, semantic converters, and/or semantic rules apply. This is very important namely to reduce development costs as well as for the feasibility of a possible product in the market. To keep the system abstracted from the semantic data model, SPARQL queries are used to get the necessary data from the knowledge base. In this way, BRICKS enables the user to define the ontologies to use that better fit the building needs. However, persistence is mandatory for BRICKS to work properly. Such is achieved by using JSON schemas. These schemas are used to validate the SPARQL queries' output which must be a JSON string. If the JSON string is valid against the corresponding schema, BRICKS will work properly, otherwise the system will ignore such knowledge. The use of JSON, instead of the traditional programming

languages' data models, eases its update since the developer does not need to reprogram the models every time the models change. In its turn, to add new knowledge to the knowledge base, SPARQL constructs are used. To make this process automatic at each iteration, string tags are used in the SPARQL constructs templates to be replaced by the corresponding measured values. This tags list is also an input for the system and depends on the resources allocated to BRICKS.

In accordance with the objectives listed in the introduction, BRICKS is a flexible system since it is not attached to any semantic data model. It is configurable to use any semantic model the user wants, which facilitates the process of evolution and maintenance of the data models. It also uses a context-aware algorithm to obtain the most adequate context profile for each iteration to validate the measurements in the rules. BRICKS overcomes the lack of interoperability between heterogeneous systems by being at a higher level, where it is able to aggregate different SCADA systems and/or smart appliances, and uses their measured data to trigger the rules, as well as their control. This higher level also keeps the system abstracted from the hardware installed and respective communication protocols, avoiding reprogramming the system every time a new device is installed. The system is abstracted from the rules and semantic models used, meaning that the user can change them at any time and the system does not need to be reprogrammed again. The richer the models, the better the knowledge inferred. BRICKS can be installed in any building with little effort, being only necessary to configure it accordingly. And finally, it centralizes heterogeneous cross-domains' data sources in a single user interface.

A case study explaining bricks configuration, execution, and outputs is also presented to clarify its functioning. The case study details the scenario deployed at the left-side of GECAD's building N. The examples presented only focus on the building's administrator perspective, and specifically in the energy management field, to avoid over-extension. This work overcomes the difficulty of interoperability between heterogeneous BMS in a building. The semantic rule-based system enables advanced machine intelligence. It is configurable in order to facilitate its deployment in different buildings and has a centralized interface to manage the cross-domains' monitoring and alarms. The use of SWRL rules have proven to be very costly in terms of performance as the number of instances in the knowledge base increases. This is explained by the fact that SWRL inference engines assert inferred facts to the knowledge base. The greater the number of facts, the more time it takes to infer new knowledge about the existing facts. And since this process is done in RAM, the greater the number of facts, the more memory is necessary for the inference, which can quickly result in the loss of performance, and consequently in an out of memory error. Using only SPARQL rules may be the best option in terms of performance, if no reasoner is used. In this case SPARQL engine will only use the RDF graph knowledge and do not infer implicit knowledge.

BRICKS will be continuously improved in order to abstract the system's user from the semantic models and rules, facilitating the building management. SPARQL "rules" will be explored to achieve more complex and intelligent results, as well as developing and evaluating security related rules.

Funding

This work has received funding from the European Union's Horizon 2020 research and innovation programme under project DOMINOES (grant agreement No 771066) and from FEDER Funds through COMPETE program and from National Funds through FCT under the projects UID/EEA/00760/2019, PTDC/EEI-EEE/28954/2017 (MAS-Society), and SFRH/BD/118487/2016.

References

Actility Veolia (2019). *Actility Veolia uses ThingPark Energy to monetize its flexible capacity through Demand Response*.

Aduda, K. O., Labeodan, T., Zeiler, W., Boxem, G., & Zhao, Y. (2016). Demand side flexibility: Potentials and building performance implications. *Sustainable Cities and Society, 22*, 146–163.

Ahvar, S., Santos, G., Tamani, N., Istasse, B., Praça, I., Brun, P., et al. (2017). Ontology-based model for trusted critical site supervision in FUSE-IT. *Proceedings of the 2017 20th conference on innovations in clouds, internet and networks (ICIN),* 313–315.

Alaya, M. B., Medjiah, S., Monteil, T., & Drira, K. (2015). Toward semantic interoperability in oneM2M architecture. *IEEE Communications Magazine, 53*, 35–41.

Althunibat, S., Wang, Q., & Granelli, F. (2016). Flexible channel selection mechanism for cognitive radio based last mile smart grid communications. *Ad Hoc Networks, 41*, 47–56.

Andoni, M., Robu, V., Flynn, D., Abram, S., Geach, D., Jenkins, D., et al. (2019). Blockchain technology in the energy sector: A systematic review of challenges and opportunities. *Renewable & Sustainable Energy Reviews, 100*, 143–174.

ASHRAE Project (2018). *Committee 201 facility smart grid information model.*

Blaauwbroek, N., Nguyen, H. P., Konsman, M. J., Shi, H., Kamphuis, I. G., & Kling, W. L. (2015). Decentralized resource allocation and load scheduling for multicommodity smart energy systems. *IEEE Transactions on Sustainable Energy, 6*(4), 1506–1514.

Bonino, D., Corno, F., et al. (2008). DogOnt – Ontology modeling for intelligent domotic environments. In A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, & T. Finin (Eds.). *Proceedings of the the semantic web – ISWC 2008* (pp. 790–803). Berlin, Heidelberg: Springer Berlin Heidelberg.

Brick (2018). *Brick schema homepage*.

Brusco, G., Burgio, A., Menniti, D., Pinnarelli, A., & Sorrentino, N. (2014). Energy management system for an energy district with demand response availability. *IEEE Transactions on Smart Grid, 5*, 2385–2393.

buildingSMART (2018). *buildingSMART homepage.*

Burmester, D., Rayudu, R., Seah, W., & Akinyele, D. (2017). A review of nanogrid topologies and technologies. *Renewable & Sustainable Energy Reviews, 67*, 760–775.

C|net Survey (2017). *C|net Survey: 98 percent of enterprises using open source*.

Calvillo, C. F., Sánchez-Miralles, A., & Villar, J. (2016). Energy management and planning in smart cities. *Renewable & Sustainable Energy Reviews, 55*, 273–287.

CAS (2010). *CAS welcome to the OPC unified architecture e-book*.

Daniele, L. (2016). *SAREF: The smart appliances REFerence ontology*.

Domingues, P., Carreira, P., Vieira, R., & Kastner, W. (2016). Building automation systems: Concepts and technology review. *Computer Standards & Interfaces, 45*, 1–12.

EEBus (2017). *EEBus homepage*.

Energy@home (2019). *Energy@home homepage*.

Faria, P., & Vale, Z. (2011). Demand response in electrical energy supply: An optimal real time pricing approach. *Energy, 36*, 5374–5384.

Faria, P., Spinola, J., & Vale, Z. (2016). Aggregation and remuneration of electricity consumers and producers for the definition of demand response programs. *IEEE Transactions on Industrial Informatics, 12*, 952–961.

Figueiredo, J., & Costa, J. S. (2012). A SCADA system for energy management in intelligent buildings. *Energy and Buildings, 49*, 85–98.

Figueiredo, J., & da Costa, J. S. (2012). A SCADA system for energy management in intelligent buildings. *Energy and Buildings, 49*, 85–98.

Gómez-Romero, J., Molina-Solana, M., Ros, M., Ruiz, M. D., & Martin-Bautista, M. J. (2018). Comfort as a service: A new paradigm for residential environmental quality control. *Sustainability, 10*, 3053.

GREEDi (2017). *GREEDi project homepage*.

Gruninger, M., & Lee, J. (2002). Ontology applications and design. *Communications of the ACM, 45*, 39–41.

GTM (2014). *GTM innovari wants to make demand response the same as an independent power plant*.

Herberg, U., Mashima, D., Jetcheva, J. G., & Mirzazad-Barijough, S. (2014). OpenADR 2.0 deployment architectures: Options and implications. *Proceedings of the 2014 IEEE international conference on smart grid communications (SmartGridComm),* 782–787.

Jajac, N., Knezic, S., & Marovic, I. (2009). Decision support system to urban infrastructure maintenance management. *OTMC, 1*(2), 72–79.

Khorram, M., Abrishambaf, O., Faria, P., & Vale, Z. (2018). Office building participation in demand response programs supported by intelligent lighting management. *International Journal of Energy Information and Communications, 1*, 9.

Lawrence, T. M., Boudreau, M.-C., Helsen, L., Henze, G., Mohammadpour, J., Noonan, D., et al. (2016). Ten questions concerning integrating smart buildings into the smart grid. *Build Environment, 108*, 273–283.

Lee, S., Kwon, B., & Lee, S. (2014). Joint energy management system of electric supply and demand in houses and buildings. *IEEE Transactions on Power Systems, 29*, 2804–2812.

Lefrançois, M., Kalaoja, J., Ghariani, T., & Zimmermann, A. (2016). *SEAS knowledge model*.

Leszczyna, R. (2018). Cybersecurity and privacy in standards for smart grids – A comprehensive survey. *Computer Standards & Interfaces, 56*, 62–73.

Liu, C., Chai, K. K., Zhang, X., & Chen, Y. (2019). Peer-to-peer electricity trading system: Smart contracts based proof-of-benefit consensus protocol. *Wireless Networks, 1*, 1–12.

Madureira, B., Pinto, T., Fernandes, F., & Vale, Z. (2017). Context analysis in energy resource management residential buildings. *Proceedings of the 2017 IEEE Manchester*

PowerTech, 1–6.
Martinez, B., Vilajosana, X., Kim, I. H., Zhou, J., Tuset-Peiró, P., Xhafa, A., et al. (2017). I3Mote: An Open Development Platform for the Intelligent Industrial Internet. *Sensors,* 17.
Marzband, M., Alavi, H., Ghazimirsaeid, S. S., Uppal, H., & Fernando, T. (2017). Optimal energy management system based on stochastic approach for a home Microgrid with integrated responsive load demand and energy storage. *Sustainable Cities and Society, 28*, 256–264.
Marzband, M., Fouladfar, M. H., Akorede, M. F., Lightbody, G., & Pouresmaeil, E. (2018). Framework for smart transactive energy in home-microgrids considering coalition formation and demand side management. *Sustainable Cities and Society, 40*, 136–154.
Mejías, A., Herrera, R. S., Márquez, M. A., Calderón, A. J., González, I., & Andújar, J. M. (2017). Easy handling of sensors and actuators over TCP/IP networks by open source hardware/software. *Sensors,* 17.
Mohan, A., & Mashima, D. (2014). Towards secure demand-response systems on the cloud. *Proceedings of the 2014 IEEE international conference on distributed computing in sensor systems,* 361–366.
Muñoz López, S., Fernández, A., Coronado, M., & Iglesias, C. A. (2016). Smart office automation based on semantic event-driven rules. In P. Novais, & S. Konomi (Eds.). *Proceedings of workshop on smart offices and other workplaces, colocated with 12th international conference on intelligent environments (IE'16)* (pp. 33–42). Vol. 21, ISBN 978-1-61499-690-3.
Nicholson, A., Webber, S., Dyer, S., Patel, T., & Janicke, H. (2012). SCADA security in the light of Cyber-Warfare. *Computers & Security, 31*, 418–436.
OPC Foundation (2018). *OPC foundation unified architecture*.
Park, S., Ryu, S., Choi, Y., Kim, J., & Kim, H. (2015). Data-driven baseline estimation of residential buildings for demand response. *Energies, 8*(9), 10239–10259.
Petrushevski, F., Gaida, S., Beigelböck, B., Sipetic, M., Zucker, G., Schiefer, C., et al. (2017). Semantic building systems modeling for advanced data analytics for energy efficiency. *Proceedings of the building simulation 2017,* 622–627.
Project Haystack (2014). *Project haystack homepage*.
Rygaev, I. (2017). Rule-based reasoning in semantic text analysis. *Proceedings of the RuleML + RR*.
Schachinger, D., & Kastner, W. (2016). Semantics for smart control of building automation. *Proceedings of the 2016 IEEE 25th international symposium on industrial electronics (ISIE),* 1073–1078.
Shakouri, G. H., & Kazemi, A. (2017). Multi-objective cost-load optimization for demand side management of a residential area in smart grids. *Sustainable Cities and Society, 32*, 171–180.
Shang, W., Ding, Q., Marianantoni, A., Burke, J., & Zhang, L. (2014). Securing building management systems using named data networking. *IEEE Network, 28*, 50–56.
Simmhan, Y., Aman, S., Kumbhare, A., Liu, R., Stevens, S., Zhou, Q., et al. (2013). Cloud-based Software Platform for Data-Driven Smart Grid Management. *IEEE/AIP Computing in Science & Engineering (CISE),* (Jul/Aug), 1–11.
Somfy (2019). *Somfy is home automation right for you*.
Su, H., Zio, E., Zhang, J., Chi, L., Li, X., & Zhang, Z. (2019). A systematic data-driven Demand Side Management method for smart natural gas supply systems. *Energy Conversion and Management, 185*, 368–383.
Subirats, L., Ceccaroni, L., Gómez-Pérez, C., Caballero, R., Lopez-Blazquez, R., & Miralles, F. (2013). On semantic, rule-based reasoning in the management of functional rehabilitation processes. In J. Casillas, F. J. Martínez-López, R. Vicari, & F. la Prieta (Eds.). *Proceedings of the management intelligent systems* (pp. 51–58). Heidelberg: Springer International Publishing.
Tamani, N., Ahvar, S., Santos, G., Istasse, B., Praca, I., Brun, P., et al. (2018). Rule-based model for smart building supervision and management. *Proceedings of the 2018 IEEE international conference on services computing (SCC),* 9–16.
Teymourian, K., & Paschke, A. (2009). Semantic rule-based complex event processing. In G. Governatori, J. Hall, & A. Paschke (Eds.). *Proceedings of the rule interchange and applications* (pp. 82–92). Berlin, Heidelberg: Springer Berlin Heidelberg.
ThinkEnergy (2019). *The pros and cons of home automation systems*.
W3C Semantic (2018). *W3C semantic sensor network ontology*. Available online: https://www.w3.org/TR/vocab-ssn/ (Accessed on 15 May 2018).
Wang, Y., Wang, B., Chu, C.-C., Pota, H., & Gadh, R. (2016). Energy management for a commercial building microgrid with stationary and mobile battery storage. *Energy and Buildings, 116*, 141–150.
Wang, N., Zhou, X., Lu, X., Guan, Z., Wu, L., Du, X., et al. (2019). When energy trading meets blockchain in electrical power system: The state of the art. *Applied Sciences-Basel, 9*, 1561.
Wijayasekara, D., Linda, O., Manic, M., & Rieger, C. (2014). Mining building energy management system data using fuzzy anomaly detection and linguistic descriptions. *IEEE Transactions on Industrial Informatics, 10*, 1829–1840.
Yu, Z., Haghighat, F., & Fung, B. C. M. (2016). Advances and challenges in building engineering and data mining applications for energy-efficient communities. *Sustainable Cities and Society, 25*, 33–38.
Yuce, B., & Rezgui, Y. (2017). An ANN-GA semantic rule-based system to reduce the gap between predicted and actual energy consumption in buildings. *IEEE Transactions on Automation Science and Engineering, 14*, 1351–1363.
Zhou, K., Fu, C., & Yang, S. (2016). Big data driven smart energy management: From big data to big insights. *Renewable & Sustainable Energy Reviews, 56*, 215–225.