1

SEMIoTICS: Semantically-enhanced IoT-enabled Intelligent Control Systems

Georgios M. Milis, Christos G. Panayiotou, and Marios M. Polycarpou
KIOS Research and Innovation Center of Excellence
Department of Electrical and Computer Engineering
University of Cyprus, Nicosia, Cyprus
Email: {milis.georgios, christosp, mpolycar}@ucy.ac.cy

***Abstract*—In today's "smart era" there is a growing ecosystem of Internet of Things (IoT)-enabled devices, which exploit (wireless) Internet connectivity and use standard communication protocols to interact with each other and the environment. As various IoT components are becoming widely available in the marketplace, a key challenge from a feedback control viewpoint is the ability to seamlessly integrate new IoT components or modify existing configurations in feedback control settings without having to halt the operation of the system and redesign the overall feedback control scheme. This article exploits technologies from the semantic web domain, for the design of a novel Semantically-enhanced IoT-enabled Intelligent Control Systems (SEMIoTICS) architecture. The proposed SEMIoTICS scheme incorporates a supervisor module able to facilitate the semantic modelling of IoT components and the subsequent online composition/reconfiguration of feedback control loops. We demonstrate the applicability of the SEMIoTICS architecture through illustrative scenarios from the Smart Buildings domain.**

I. Introduction

TODAY'S engineered systems more and more employ cyber and physical components with advanced communication capabilities [1], such as sensors for monitoring system properties, electrical and mechanical actuators, controllers and other software tools implementing algorithms for data and signal processing. This trend is becoming even more prominent due to the incorporation of Internet of Things (IoT)-enabled components in cyber-physical systems. IoT components have the capability to exploit (wireless) Internet connectivity and use standard communication protocols, e.g. MQTT [2], to interact with each other and the environment, exchanging data and information in a context-aware framework [3]. They are able to cover a broad range of functionalities, from observing and measuring properties of physical features of interest, to processing collected data and information, to acting and affecting these properties after receiving appropriate instructions.

Large-scale smart buildings are good candidate consumers of IoT components, due to their advanced monitoring and control needs, as well as due to the readily available Internet infrastructure. For instance, a smart building may consist of various sub-systems that control the lighting, the temperature and the humidity, monitor the quality of the air, supervise the fire alarm systems and many more. Current solutions for monitoring and control applications typically assume pre-deployed sensing and actuation devices, as well as pre-designed control intelligence. However, the IoT ecosystem is rapidly evolving and IoT components, including mobile ones and virtual ones running through a computer terminal or on the cloud, can be deployed and/or removed online, thus modifying the available sensing, actuation and processing capabilities (e.g., by measuring the occupancy of rooms or the openings of windows/doors, by correlating CO_2 concentration measurements to occupancy values, etc.). Building operators will expect their control systems to exploit seamless intelligence; i.e., be able to utilize the new measurements, as well as utilize the processing and actuation capabilities online, towards improving the occupants' comfort, reducing energy costs and so on. For example, an existing temperature regulation system could be reconfigured online so as to utilize the output of a newly installed occupancy sensor to lower the reference value of the room temperature when the room is not occupied, thus saving energy.

The control community has recently identified the above challenges [4], also suggesting the exploitation of technologies from the Web domain, that deal with online discovery and composition of services [5]. In previous work [6], the authors addressed some aspects of the above challenges by developing an initial design of a novel architecture that incorporates a supervisor module with logical inference capabilities. The supervisor uses declarative language (ontologies) to model the expert knowledge about individual components in a consistent way, so as to be exploitable by machines. It subsequently performs deductive reasoning to decide the online composition/reconfiguration of feedback control loops, thus addressing the design-operation continuum challenge, as recognised by the CPSoS EU project with relation to the cyber-physical systems of systems [7]. The present article significantly extends our previous work by developing the Semantically-enhanced IoT-enabled Intelligent Control Systems (SEMIoTICS) architecture. The specific contributions are summarized as:

- Incorporation of more types of control system components, i.e., "Processing Functions", allowing to offer online utilization of advanced functionality, e.g. fusion of measurements, amplifying of control decision signal,

This work is partially funded by: i) the European Research Council (ERC) under the project ERC-AdG-291508 "Fault-Adaptive Monitoring and Control of Complex Distributed Dynamical Systems" (FAULT-ADAPTIVE); ii) the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement No 739551 (KIOS CoE).

etc. Our previous work focused only on sensors, actuators and controllers, considering processing functions only for measurement unit transformations.
- Modification and extension of the semantic annotation models of the components, as well as further development of the operations of "semantic annotation", "semantic matching" and "semantic reasoning".
- Incorporation of a mechanism to extract the semantically valid feedback control system configurations and select the one with higher performance against pre-defined criteria.

The article is organized as follows. Section II provides an overview of the related state-of-art. Section III formulates the addressed problem. Section IV presents the refined SEMI-oTICS architecture, followed by Section V that describes the design of the semantically-enhanced supervisor. Then, Section VI presents application scenarios of our work in the Smart Buildings domain. Finally, Section VII discusses the overall impact of the work and concludes the document.

II. State-of-art overview

During the last two decades there has been significant progress in the development of fault tolerant control schemes to address the cases of components' faults [8], as well as the more general cases of changes in the dimension of input, output and/or state vectors during operation of the system (e.g., when new sensors or actuators are plugged in a closed-loop system) [9]–[11]. Still, it remains impractical, if at all possible, to design apriori a single controller that would have all the required adaptation intelligence at the design time. Several approaches have been considered, including the variable structure control method [12], where it is assumed that the control system switches among pre-defined control structures that take over when certain criteria are met.

The emergence of the IoT paradigm and the growing ecosystem of IoT-enabled devices that exploit (wireless) Internet connectivity, have created new opportunities, as well as new challenges for large-scale cyber-physical systems and subsequently for their effective monitoring and control. The IoT-enabled environments are inherently dynamic in terms of the number, types and capabilities of deployed devices. The vision is for the capabilities of IoT components, i.e., sensors, actuators and other processing units, to be modelled as "services" that are consumed by implementations of monitoring and control algorithms, thus offering more advanced and composite services [4]. The latter requires multi-disciplinary research work and basic knowledge from both the ICT and the control domains. The W^3C Semantic Sensor Network Incubator Group (SSN-XG), as well as the Open Geospatial Consortium (OGC) have been among the first who identified the challenge. They created the "Semantic Sensor Network (SSN)" [13] and the "SensorML" standard [14] respectively, for the semantic characterisation of sensors' operation. The standards have recently published updates that take into consideration also the modelling of actuation and other processing operations.

During the last few years, several groups have started working on top of the above standards, to address the interoperability, evolve-ability and online re-configuration challenges of IoT-enabled systems. The authors in [15] have worked on the conceptual matching of heterogeneous lexical sources and ontologies in the IoT and Smart Cities domains. The authors in [16] have used rule-based systems for the automatic construction of topical ontologies from real-world IoT sensor measurements. They also developed a distributed mechanism to index IoT resources and their data, based on their location, thus enabling online grouping by similarity [17]. Another group used knowledge models to process large amount of IoT data and produce added-value knowledge for Smart City applications [18]. Other works [19], [20] model IoT components as Web resources, thus enabling their online discovery and utilisation of their capabilities. An attempt to design self-configurable systems composed of IoT components has been also presented in [21]. There is also work from the "smart factory" domain and the Industry 4.0 concept, with teams working on modelling the information about factory objects (hardware or software) using semantic knowledge formalisms [22] and also modelling the IoT resources and processes towards optimal resource management [23].

The above presented work addresses spherically and extensively the services' and data semantic interoperability challenge in the IoT domain. It mainly focuses on the sensing and data collection and manipulation capabilities, which support IoT applications in general. However, IoT-enabled applications will not be able to offer effective monitoring and control services, if the semantic models do not explicitly consider the systems' modelling and control design details. The need to extend the semantic models towards conceptualising control and other processing entities as well, has been partially addressed in [24]. The author identified the need to model sensors, actuators and controllers, presenting a "Semantic Smart Gateway Framework" that acts as an interoperability service and mediator between IoT application providers and consumers. Although this work effectively models the semantics of control from an input-output perspective, it does not take a control-theoretic view, thus it is not opening the way to deal with the internal dynamics and online (context-aware) re-configuration of IoT-enabled control systems.

Specifically for the smart buildings domain, there has recently been work on designing semantic information models using ontologies [25], [26]. Other researchers have also acknowledged the need for additional flexibility in the IoT paradigm and started exploiting the Building Information Model [27], [28] and enriching it with Resource Description Framework (RDF)-based information concepts from the fault diagnosis in Building Automation domain [29]. This work addresses the challenge of enabling online self-configuration of the parameters of a fault-diagnosis methodology, using the ontological models specifically for the buildings domain.

When it comes to control systems, the explicit incorporation of semantic models in the design of control systems has been proposed back in 1988 [30], based on the concepts of semantic control systems discussed in [31]. In an IoT perspective, the key challenge still remains the implementation of the feedback control systems as compositions of components acting as providers and/or consumers of "services" thus allowing

flexibility in adapting to components' changes.

Before dealing with the (semantic) interoperability in IoT-enabled control systems, work needs to start from the characterization of control system entities using component-based architectures [32]. These techniques have been recently applied for the development of component-based portable control applications for buildings, decoupled from hardware and building specificity [33].

As mentioned in Section I In a recent work [6] we advance the above presented state-of-the-art, by entering into the details of the self-adaptation and reconfiguration need for the feedback control system and designing the basics of the semantic modelling and reasoning mechanisms. In that work, we propose the modelling of expert knowledge in a super-graph of many bipartite graphs formed by meaningful relations between modelled knowledge entities of pre-defined types [34], [35]. The switching among extracted control system configurations is decided and enforced through a discrete-state logic-based deductive reasoning process [36]. The valid control system configurations are not defined in advance; instead, they are configured online using components from an evolving database of IoT control system components.

III. Problem Formulation

Consider the plant in Fig. 1, characterised by a vector of state variables $x(k) \in \mathcal{R}^{n_x}$, with k being the time-step in a discrete-time formulation. In a Smart Building context, these can represent room temperature, humidity, lighting, air quality, etc.. We assume that the state variables need to be controlled to satisfy certain objectives and this is achieved by the design of a control system able to generate an appropriate plant input vector $v(k) \in \mathcal{R}^{m_v}$ and provide the expected control service to the plant. For generality, the vector $w(k) \in \mathcal{R}^{q_w}$ represents any uncontrolled inputs to the plant. In an IoT context, the plant may be equipped with multiple sensors able to measure its states, multiple actuators able to act and affect its states, multiple controllers able to consume the measurements and drive the actuators, as well as multiple other functions able to process and transform the produced signals when required. Assuming that at certain time-steps k_I, $I = 0, 1, 2, ...$ events happen that change the availability of IoT components in the plant, then in each period without changes there can be a finite number (n_I) of control system configurations able to offer the required control service.

We address the problem of designing a control system that will be able to "know" the available components after each change at time steps k_I, extract each time the n_I possible control system configurations and select one of them (based on pre-defined criteria) to be put in operation. The control service is formulated as: $v(k) = f_I(x(k_I^+), k_I^+; \sigma_I)$, where the function $f_I(\cdot)$ represents the implementation of the selected control system configuration among the n_I ones, k_I^+ denotes the discrete time steps for $k > k_I$ (following the event I) and σ_I represents the respective selection decision, which activates one of the configurations. The objective is to design a supervisor module Σ (light-green-highlighted module), able to generate online the decisions σ_I.

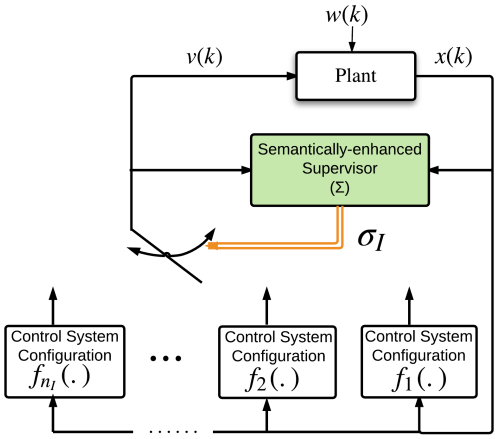

Figure 1. A plant with multiple IoT components available at time step k_I, forming n_I "Control System Configurations". A supervisor generates the respective selection signal σ_I

IV. System Architecture

The SEMIoTICS architecture is presented in Fig. 2. We consider each "Control System Configuration" $f_I(\cdot)$ as a composition of instances of specific types of components available in the plant as a result of the event I at time step k_I. Each composition consists of zero or more sensors producing the signal-vector $y^s(k_I^+)$, one controller producing the control decision $u^c(k_I^+)$, one or more actuators producing the plant input vector $v(k_I^+)$ and zero or more of other signal processing functions. The latter are further classified into: i) "Pre-control Functions" that transform plant state measurements to the signal-vector $y^y(k_I^+)$; ii) "Post-control Functions" that transform control decision signals to the signal-vector $u^u(k_I^+)$; and iii) "Parameter Functions" that utilize measurements of plant properties and produce/update values of general plant and control system parameters ζ_x, $x \in \{s, y, c, u, a, \zeta\}$ required by the respective types of participating components. In general, designs of feedback-control systems may require more types of components, e.g., "Online Learning" structures (neural network, polynomial function, radial-basis functions, wavelets, etc.); however, we do not consider other types of components at this stage of our work.

Each instance of the above types of control system components produces a certain output vector, after being provided with a certain input vector and, where necessary, a parameter vector. The inputs and outputs comprise the signals involved in the feedback loop continuous information flow (black continuous lines). On the other hand, the values of parameters (e.g., openings of windows in a building, set-points of heating elements, desired plant state trajectories, etc.) are considered retrieved on-demand and/or on-availability, from dedicated "Parameter Functions" (blue dashed lines). The internal dynamics of the components are generally unknown. For instance, a post-control processing function may implement an amplifier of a control signal so as to properly drive an available actuator; a pre-control processing function may act on the measured signals to perform data validation/reconstruction or

state-estimation, e.g., Kalman filter, Luenberger observer, etc.; the controller may be a simple On/Off function or a PID implementation or a more advanced non-linear and adaptive control algorithm. SEMIoTICS considers all components as syntactically modelled by general formulas. E.g., a controller is modelled by $u^c = f_c(y^y; \zeta_c)$, where u^c, y^y and ζ_c are vectors of appropriate dimensions and depend on the order of the system and the specific implementation of the controller. Certain internal parameters of the implementation can then be exposed through the controller semantic annotation. Future work can also consider online design of models and controllers in cases of an evolving plant topology. Whether a specific component will take part in a control system configuration, is defined by a Semantically-enhanced Supervisor module Σ through the selection decision σ_I (double orange line), as detailed in the sequel.

It is clarified that the assumption of one controller per plant is not restrictive since multiple instances of the SEMIoTICS architecture may run in parallel, to configure control systems for parts of a bigger plant.

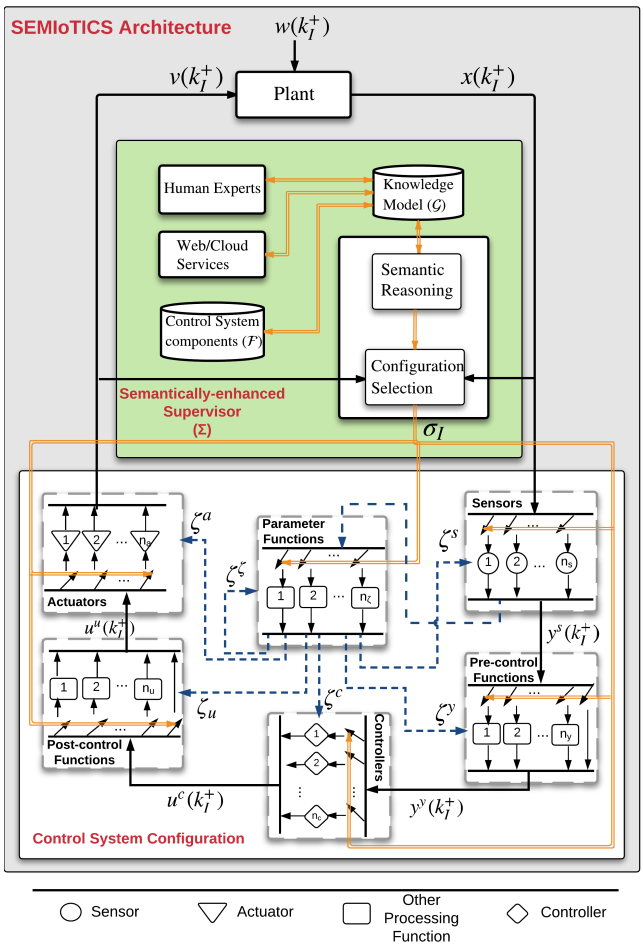

Figure 2. The SEMIoTICS architecture

V. Semantically-enhanced Supervisor

In this section we go into the design of the sub-modules and operations of the supervisor Σ. All operations of Σ are built on top of the "Knowledge Model" $\mathcal{G}$, which is a super-graph of many bipartite graphs formed by semantic relations between modelled knowledge objects of pre-defined types [6]. To model the knowledge we use declarative language, i.e. *ontologies*, building on top of: i) the W^3C "OWL-S" standardisation effort that deals with the semantic composition of (Web) services/components [37]; and ii) the "Semantic Sensor Network (SSN)" ontology which is a joint effort of the W^3C and the Open Geospatial Consortium (OGC) and describes sensors and actuators, as well as the features of interest and the specific properties they observe/affect [13].

Referring back to Fig. 2, it can be seen that the modelled knowledge can be updated/inserted online by human experts. For instance, an engineer may provide information about a new sensor, the topology of the plant, and so on. In addition, updates of the knowledge can be performed automatically through Web/Cloud services, e.g., retrieving information about a component from online sources, assuming that such information is kept in a remote database in the form of "semantic drivers". Finally, the Knowledge Model also retrieves information directly from the available IoT control system components, so as to become aware of their characteristics and capabilities. With the emerging maturity of IoT, this requirement is addressed by established message formatting standards and communication protocols, like MQTT [2]. For convenience, we define the database of "Control System Components" ($\mathcal{F}$). The operation of modelling the knowledge about the characteristics and capabilities of control system components is called "semantic annotation" and is presented in the sequel.

A. Semantic Annotation

The SEMIoTICS Knowledge Model, $\mathcal{G}$, provides explicit support for all the following types of control system components: Plant, Sensors, Actuators, Controllers, Processing Functions (Pre-Control, Post-Control and Parameter Functions). We further adopt additional parts of the OWL-S "Service Profile" model for the modelling of the service offered by each type of component. I.e., each component has inputs, outputs, parameters, as well as some additional information for its categorisation. For simplicity, at this stage we do not use the concepts of "pre-conditions" and "effects" [37]. The semantic characterisation of the control system components is mainly based on the SSN ontology.

The SSN ontology defines sensors and actuators as "Systems" that "observe"/"act-on" a certain "property" of a "feature of interest" of the environment in which they are deployed. For instance, a sensor may measure the property "temperature" of the feature of interest "room 1" in a given building. The same ontology defines that such a "System", in order to provide its intended service, implements a "Procedure" that has certain "Inputs" and "Outputs". In our work we extend this by modelling sensors, actuators, controllers, other processing functions and the plant itself as components that transform properties of the plant's features of interest, towards offering a collaborative control service. To avoid complexity and without loss of generality, we do not use the

SSN ontology in its full detail but we adopt the parts that help us define our semantic models. Moreover, we adopt the concept "feature of interest" to refer to specific objects in a physical plant, which correspond to specific "locations" in the plant. "Locations" here do not refer to a representation of coordinates in a geographic map; they refer to parts of the plant and objects in the plant that correspond to certain relative positions; e.g., "heater 1", "room 1" "window 1" are locations and subsequently features of interest in the plant. In order to model relations between locations, we adopt concepts of the GeoSPARQL model [38], e.g. "touches", "inside", "contains".

In order to model explicitly the services offered by control system components and facilitate their online invocation, we combine the "Procedure" concept of SSN with the "Service" concept of OWL-S. That is, beyond inputs and outputs we consider additional "parameters" required by each component. Also, we incorporate categorisation of the services (from OWL-S) in terms of their "capability" on the specific properties of features of interest (from SSN) that are represented by inputs, outputs and parameters. These allow us to model the knowledge about all produced/consumed signals using the "Five Ws and one H" method [39], which has been proposed for capturing and communicating the correct information about an entity in a reporting or decision making context. As an example, Fig. 3 shows the semantic annotation of an input, an output and a parameter of a component. It can be seen that the semantic annotation space Λ is defined by four dimensions: $\Lambda \equiv \mathcal{L} \times \mathcal{Q} \times \mathcal{P} \times \mathcal{M}$. That is, an element of the space Λ is represented by the specific values in a quadruple of respective variables:

- Variable l represents the plant's "feature of interest" and answers to the question "WHERE", taking values from the set $\mathcal{L} = \{\text{room 1}, \text{room 2}, \text{door}, \text{window}, \text{ambient}, \text{west wall 1}, \text{ceiling 1}, \text{heater}, ...\}$. The set can be the output of the building design using a CAD software.
- Variable q represents the studied property of the feature of interest and answers to the question "WHAT", taking values from the set $\mathcal{Q} = \{\text{temperature}, \text{energy}, \text{opening}, \text{flow rate}, \text{filtration rate}, \text{fan speed}, \text{time}, ...\}$. The values of this set, as well as of the measurement unit below, can be retrieved from existing models (e.g., the current version or future extensions of the Building Information Model [40]).
- Variable p represents the role of the signal or the parameter in the control system configuration and answers to the question "WHY", taking values from the set $\mathcal{P} = \{\text{state}, \text{stateMeasurement}, \text{controlDecision}, \text{disturb}, \text{referenceValue}, \text{topologyParameter}, \text{regulate}, \text{increase}, \text{decrease}, ...\}$. These values are given at the time of annotating the component, either manually selected by the engineer/technician or automatically by downloading the information from the Internet.
- Variable m represents the measurement unit of the property, where applicable, and answers to the question "HOW", taking values from the set $\mathcal{M} = \{\text{Celcius}, \text{Fahrenheit}, \text{kWatt}, \text{kilogramsPerSecond}, \text{percentage}, ...\}$

Note that the question "WHO" is explicitly answered through the link of inputs, outputs and parameters to specific components, whereas the question "WHEN" is out of the scope of the decision making performed by the supervisor Σ. The size of the above sets can change online, adding or removing elements, without affecting the operation of the system.

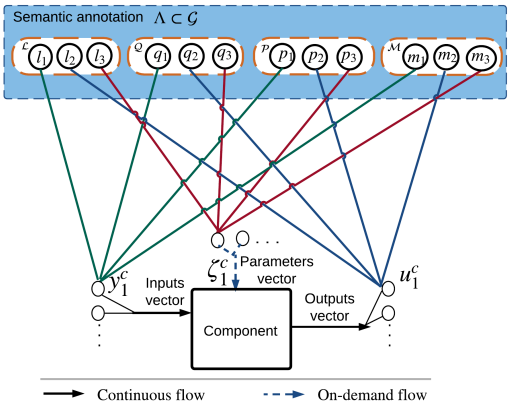

Figure 3. A control system component with an example semantic model of an input, an output and a parameter

The "Semantic Annotation" operation is then defined as: $\lambda(\cdot) : \mathcal{A} \mapsto \Lambda$, where $\mathcal{A}$ denotes the set of all inputs, outputs and parameters of components. For instance, the annotated input in Fig. 3 may represent the measured temperature state of "room 1" in degrees Celsius and denotes a point in the space Λ, as: $\lambda(y_1^c) = \{l = \text{room 1}, q = \text{temperature}, p = \text{stateMeasurement}, m = \text{Celsius}\}$. In the same way, the semantic annotations of the example output and parameter are: $\lambda(u_1^c) = \{l = \text{heater}, q = \text{flow rate}, p = \text{controlDecision}, m = \text{kilogramsPerSecond}\}$ and $\lambda(\zeta_1^c) = \{l = \text{door}, q = \text{opening}, p = \text{topologyParameter}, m = \text{percentage}\}$.

There are cases where a semantic annotation does not define a specific value for one or more of the variables. In such cases, the annotation is considered as covering an area of the semantic annotation space instead of a single point. In these cases, instead of a value we use the symbol of the respective set. For instance, if no specific measurement unit was defined for the above door opening, the semantic annotation would be: $\lambda(\zeta_1^c) = \{l = \text{door}, q = \text{opening}, p = \text{topologyParameter}, m = \mathcal{M}\}$.

We clarify that a component with multiple inputs, outputs and/or parameters is associated with a vector of semantic annotations, elements of Λ. For instance, in the case of a PID controller with its coefficients exposed as a parameter vector in SEMIoTICS, these will correspond to a (equal size) vector of semantic annotations. We further clarify that the modelled control system components do not necessarily coincide with single physical devices or software tools; i.e., a single device may implement more than one of these components, each one with its own syntactic profile and semantic annotations. For instance, a device that measures the occupancy and uses that

value to estimate the CO_2 concentration in a room, offering both values as output, will be modelled as two components: i) a "Sensor" receiving a "state" signal (e.g., occupancy of room) and producing a signal of type "stateMeasurement"; ii) A "Pre-control Function" that uses a "stateMeasurement" signal and produces a processed "stateMeasurement" signal. This helps us make a step towards breaking the dynamics of a control system into smaller components which cooperate to provide the overall control service.

B. Semantic Reasoning

The semantic annotations of all inputs, outputs and parameters of components are considered pre-defined in the Knowledge Model. The "Semantic Reasoning" then considers the "Semantic Annotations" and tries to evaluate their "Semantic Matching". We define the "Semantic Matching" operator as: $\rho : \Lambda \times \Lambda \mapsto \{\top, \bot\}$. This operator takes as input a pair of semantic annotations and returns '$\top$' (true) if the two annotations share at least one point in the space Λ, otherwise it returns '$\bot$' (false). The Semantic Matching operator is used in two different cases:

1) Output-Input semantic matching: it checks all individual outputs of components to all individual inputs, excluding the inputs and outputs of "Parameter Functions". For instance, a sensor output signal annotated with the quadruple $\{\text{room } 1, \text{temperature}, \text{"stateMeasurement"}, \text{Celsius}\}$ matches with the input of a controller annotated with the quadruple $\{\text{room } 1, \text{temperature}, \text{"stateMeasurement"}, \mathcal{M}\}$. This can be a simple "proportional controller" whose gain is not affected by the specific measurement unit but it only depends on the difference from a respective reference value. Note that the controller's input annotation is a sub-space containing the single-point annotation of the sensor's output.
2) Parameters semantic matching: it checks whether the parameters required by a component semantically match with the output of a "Parameter Function". For instance, a parameter required by a controller and annotated with the quadruple $\{\text{room } 1, \text{temperature}, \text{"referenceValue"}, \text{Fahrenheit}\}$, will match with the output of a "Parameter Function" annotated with the same quadruple.

Further to the direct matching between semantic annotations as described above, the "Semantic Reasoning" submodule explores also transformations of these annotations within the semantic annotation space. Typically, the control system signals are assumed in spaces of real numbers and the transformations happen between such spaces of different dimensions. This is convenient during the design of fixed-configuration control loops, where the knowledge about the involved components and their variables is implicitly passed in the implementations by the human expert. However, if we want a machine to perform configurations of closed loops online, then we have to model explicitly the representation of the signals, and subsequently the transformations, in a higher-dimension space. That is, the states vector can be denoted as $x \in \mathcal{R}^{n_x} \times \Lambda^{n_x}$ and the controlled inputs vector as $v \in \mathcal{R}^{m_v} \times \Lambda^{m_v}$ respectively. This way, each component not only transforms the real value of signals but it also affects their semantic annotations. For instance, the semantic annotation of a "state" signal given by $\{\text{room } 1, \text{temperature}, \text{"state"}, \mathcal{M}\}$, when measured by a temperature sensor with output in Celsius, will take the value $\{\text{room } 1, \text{temperature}, \text{"State"}, \text{Celsius}\}$.

Beyond the semantic transformations that happen to the signals when passing through certain control system components, the semantic annotation of a signal can be transformed also using "semantic rules" [31]. Such rules comprise the encoding of expert knowledge about whether we can move from one point of the semantic annotation space to another, without affecting the "meaning" of the variable in the subject domain usage. For instance, the semantic annotation of a sensor's output $\{\text{window } 1, \text{temperature}, \text{"state"}, \text{Celsius}\}$, can be transformed to $\{\text{room } 1, \text{temperature}, \text{"State"}, \text{Celsius}\}$ provided that the linguistic terms "window 1" and "room 1" are linked through the relation "within", defined using the concepts of the GeoSPARQL presented earlier. As defined in our previous work [6], these rules comprise composite relations as compositions of "relation graphs"; the edges of these graphs are not explicitly defined but they are implemented as paths of length > 2. The relations and the semantic rules are exploited by the supervisor Σ in the reasoning process, to evaluate whether certain semantic transformations can be applied for enabling the semantic matching.

At each execution of the "Semantic Reasoning" process, following an event I at time step k_I, the supervisor iterates within the controllers and tries to match their inputs and outputs considering that a component can be used only if all its inputs match with outputs of other components and also all its required parameters match with outputs of "Parameter Functions". The operation detects all semantically valid matchings between control system components and extracts the n_I candidate control system configuration options. From these valid options, one needs to be selected to operate the control system. The selection mechanism is described in the sequel, while examples are given in Section VI.

C. Configuration option selection

The final task of the Supervisor Σ is to explore the detected configuration options, apply some logic to select one of them and produce the decision σ, using pre-defined cost criteria. The "Configuration Selection" operation is formulated as in (1).

$$\underset{i}{\operatorname{argmin}} \sum_{j} w_j \times c_j \tag{1}$$

where i is an iterator of the set of the n_I valid configurations from which we want to keep the one that minimizes the argument on the right, $c_j, j = 1, ..., n_t$ define the pre-defined cost criteria utilised for the selection process and $w_j, j = 1, ..., n_t$ define the weights assumed for each criterion in the current implementation. The weights take values in the range $[0, 1]$ and

model the significance given to each criterion by the building operator.

To implement the logic of the configuration selection operation, a simple solution would be to use the first one of the detected configuration options. Another option would be to associate the control system configurations with a rating mechanism, such as for the components participating in "well-performing" configurations to have their rating score increased by a certain value. A third option would be to test each configuration option for a certain amount of time on the real system and choose the one with higher performance against the pre-defined criteria. A fourth option would be to test each configuration in a simulation, using pre-defined models for the plant and the actuators. Each option has its own advantages and drawbacks; the first two fail to provide high confidence about the real system operation and performance, while the third requires that untested configuration options are put in operation, with unpredicted behaviour. The fourth option makes the assumption that the installation of a SEMIoTICS architecture is accompanied by test-models of the plant and actuating components. These are not trivial to obtain, especially for the plant. However, recent work [11] already considers online generation of models for a fault diagnosis application in buildings.

In this work, we chose and implemented the fourth option. That is, each of the valid configuration options is passed through a 1-hour simulation test and is evaluated against the criteria. The simulation test uses a pre-defined (test) model of the plant, instantiated with certain parameters so as to offer evaluation on equal basis. In addition, the actuators are also assumed accompanied by their model implementations for testing purposes. We define two cost-criteria ($n_t = 2$), as follows:

1) *Performance cost:* $c_1 = \frac{1}{N_{k_s}} \sum_{k_s} (\frac{(x(k_s) - r(k_s))}{r(k_s)})^2$
2) *Energy cost:* $c_2 = 1 - \frac{1}{N_{k_s}} \sum_{k_s} (\frac{(v(k_s) - v_{max}(k_s))}{v_{max}(k_s)})^2$

where N_{k_s} represents the simulation time in discrete time-steps, k_s is the simulation time step iterator, $x(k_s)$ is the vector of plant states, $r(k_s)$ is the vector of the respective reference values, $v(k_s)$ is the vector of simulated plant inputs and $v_{max}(k_s)$ is a vector of the theoretical maximum input to the plant by each actuator.

We use the "normalised mean squared error (NMSE)" to measure how far the performance of the feedback control configuration is from the reference state values. A real implementation may choose to use a different formula. The second criterion calculates the accumulated plant input as a percentage of a theoretical maximum. For simplicity and without loss of generality, we also assume that the output of all actuators is transformed to a reference scale to become comparable, and that the energy cost of operating the components is proportional to their output.

VI. Application scenarios

Consider the small building of Fig. 4 consisting of two rooms with certain openings (plant f_1^p), which is equipped with a temperature regulation control system in the room at the top ("room 1").

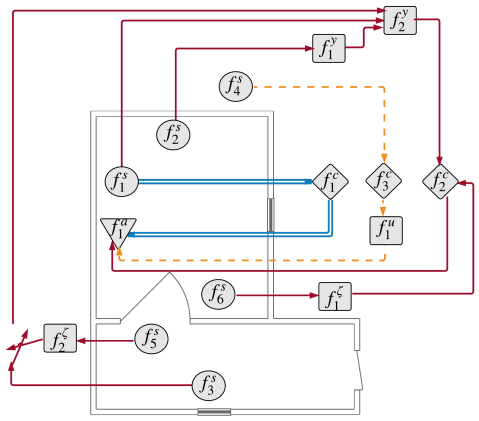

Figure 4. A building with two adjacent rooms. Several IoT components are gradually deployed in the building and potentially being utilized in different control loops to regulate the temperature of the room at the top.

We describe the gradual deployment of IoT-enabled cyber-physical control system components at time steps k_I, $I = 0, 1, 2, ...$, and each time we show the result of the reasoning performed by the supervisor Σ to extract the semantically valid configuration options and facilitate the subsequent selection of the configuration $f_I(\cdot)$ to be put in operation. For the convenience of the reader, Table I presents the complete database $\mathcal{F}$ of control system components shown in Fig. 4, together with their respective semantic annotations, adopting the formulation given in Section V-A.

At $k = k_0 = 0$ (i.e., the initial deployment and configuration of components $I = 0$ is assumed happening at $k = 0$), the control system is operating with a small set of components, comprising one temperature sensor f_1^s, one heating element f_1^a and one controller f_1^c that reads the room temperature and drives the heating element. From Table I, it can be seen that the first output of the plant f_1^p semantically matches the input of the sensor f_1^s, since $\lambda(x_1^p) = \lambda(x_1^s) = \{\text{room 1}, \text{temperature}, \text{state}, \mathcal{M}\}$. Then, the output of f_1^s semantically matches the input of f_1^c, since $\lambda(y_1^s) = \lambda(y_1^c) = \{\text{room 1}, \text{temperature}, \text{stateMeasurement}, \text{Celsius}\}$. Following the same logic we check the semantic matching of the output of the controller f_1^c to the input of the actuator f_1^a. It can be seen that the output of the controller does not define a specific "location", whereas the actuator's input instantiates the respective variable as $l =$ 'west wall 1'. That is, the input's annotation comprises a point within the area formed by the output's annotation, thus the respective semantic matching is confirmed. Then, checking the potential matching of f_1^a output to the (controlled) input of the plant, we see that the first defines a point that does not lie within the area formed by the second. However, "room 1 contains the west wall 1" and this connection between the two locations is assumed captured in the knowledge model through the relation concept 'contains'. This enables the semantic transformation of the annotation of the actuators' output to a point that lies within the area formed by the annotation of the plant's input. The result is that the reasoning extracts a single valid configuration ($f_0(\cdot)$), as recorded in Table II and also shown in Fig. 4

with double blue-coloured line. The same table shows, in addition, the evaluation cost of each configuration option, assuming two scenarios in relation with the selection criteria of subsection V-C; one with the two criteria given equal significance and another with the energy saving criterion given 80% significance comparing to 20% of the comfort criterion. The time needed for the execution of the reasoning process and the configuration decision making is also given. It can be seen that for a single configuration option it takes 1.8 minutes. The time-to-decision is considered an important factor when it comes to the adoption of this solution in real environments, since it may affect the stability properties of plants with fast dynamics. The latter is a subject of our ongoing research.

Later during the operation of the system, at time-step k_1, the building operator buys and installs sensor f_4^s, which measures the ambient temperature. The operator also "downloads": i) two more controllers, f_2^c, f_3^c; ii) the Post-control Function f_1^u that transforms a control signal given in percentage, to the control value expected by the heating element. Supervisor Σ is then able to recommend two control system configuration options, as presented in Table II. Looking at the semantic annotations in Table I, it can be seen that the controller f_c^3 is able to consume the ambient temperature measurement and drive the actuator f_a^1 when its output is processed by the function f_1^u. The selected configuration is illustrated in Fig. 4 with dashed orange-coloured line. Table II shows also the total costs of each option for each of the two cases of criteria-weights and highlights the selected one (minimum cost in bold fonts) in each case. The time-to-decision appears slightly higher, which is expected since the reasoning operation needs to consider additional components and their semantic annotations.

At time step $k_2 > k_1$, the building operator installs more sensing capabilities: sensor f_3^s measures the temperature of the second room ("room 2"), sensor f_5^s measures the opening of the door in the wall separating the two rooms, and sensor f_6^s measures the occupancy of "room 1". During that time, a user of the first room has an activated temperature measurement on her mobile device (f_2^s), with the output being in degrees Fahrenheit. We also assume that in the meantime the database of components has been enriched with few more functions: i) f_1^y that transforms Fahrenheit to Celsius; ii) f_2^y that performs an averaged fusion of input signals; iii) f_1^ζ that lowers the reference value of the temperature to 18^oC instead of the default 25^oC, when the room is not occupied; and iv) f_ζ^2 that reads the door opening measurement and updates the knowledge model by connecting the two rooms with the relation "within" when the opening exceeds 80%. Table II shows a configuration option where function f_1^ζ utilizes the occupancy measurement and updates the temperature reference value, allowing the controller f_2^c to save energy. Also, the temperature measurement of the mobile device can be fed to the fusion function f_2^y, the output of which can be given to any of the controllers f_1^c and f_2^c. Finally, the temperature of "room 2", output of sensor f_3^s, can be also fused together with the other measurements when the opening of the door causes the change in the relation between the two rooms. The result is that the new situation enables the supervisor Σ to recommend nine configuration options for the feedback control loop and make its decision in 6 minutes, as shown in Table II. The selected configuration for the case of equal significance of the evaluation criteria is illustrated in Fig. 4 with dark red-coloured line.

Table I Components' database and Semantic Annotations

Plant		
f_1^p	Inputs:	$\lambda(v_1^p) = \{$room 1, heat energy, increase, $\mathcal{M}\}$
		$\lambda(w_1^p) = \{$ambient, temperature, $\mathcal{P}$, $\mathcal{M}\}$
	Outputs:	$\lambda(x_1^p) = \{$room 1, temperature, state, $\mathcal{M}\}$
		$\lambda(x_2^p) = \{$room 1, occupancy, state, $\mathcal{M}\}\}$
		$\lambda(x_3^p) = \{$window, opening, state, $\mathcal{M}\}\}$
		$\lambda(x_4^p) = \{$door, opening, state, $\mathcal{M}\}$
		$\lambda(x_5^p) = \{$room 2, temperature, state, $\mathcal{M}\}$
		$\lambda(x_6^p) = \{$ambient, temperature, state, $\mathcal{M}\}$
	Parameters:	$\lambda(\zeta_1^p) = \{$window, opening, topologyParameter, percentage$\}$
		$\lambda(\zeta_2^p) = \{$door, opening, topologyParameter, percentage$\}$
Sensors		
f_1^s	Inputs:	$\lambda(x_1^s) = \{$room 1, temperature, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_1^s) = \{$room 1, temperature, stateMeasurement, Celsius$\}$
f_2^s	Inputs:	$\lambda(x_2^s) = \{$ceiling 1, temperature, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_2^s) = \{$ceiling 1, temperature, stateMeasurement, Fahrenheit$\}$
f_3^s	Inputs:	$\lambda(x_3^s) = \{$room 2, temperature, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_3^s) = \{$room 2, temperature, stateMeasurement, Celsius$\}$
f_4^s	Inputs:	$\lambda(x_4^s) = \{$ambient, temperature, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_4^s) = \{$ambient, temperature, stateMeasurement, Celsius$\}$
f_5^s	Inputs:	$\lambda(x_5^s) = \{$door, opening, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_5^s) = \{$door, opening, stateMeasurement, percentage$\}$
f_6^s	Inputs:	$\lambda(x_6^s) = \{$room 1, occupancy, state, $\mathcal{M}\}$
	Outputs:	$\lambda(y_6^s) = \{$room 1, occupancy, stateMeasurement, $\{\top, \bot\}\}$
Pre-control processing functions		
f_1^y	Inputs:	$\lambda(y_1^y) = \{$room 1, temperature, stateMeasurement, Fahrenheit$\}$
	Outputs:	$\lambda(y\prime_1^y) = \{$room 1, temperature, stateMeasurement, Celsius$\}$
f_2^y	Inputs:	$\lambda(y_2^y) = \{$room 1, 'temperature', stateMeasurement, Celsius$\}$
		$\lambda(y_3^y) = \{$room 1, 'temperature', stateMeasurement, Celsius$\}$
		$\lambda(y_4^y) = \{$room 1, 'temperature', stateMeasurement, Celsius$\}$
	Outputs:	$\lambda(y\prime_2^y) = \{$room 1, temperature, stateMeasurement, Celsius$\}$
Controllers		
f_1^c	Inputs:	$\lambda(y_1^c) = \{$room 1, temperature, stateMeasurement, Celsius$\}$
	Outputs:	$\lambda(u_1^c) = \{$room 1, flow rate, controlDecision, kilogramsPerSecond$\}$
f_2^c	Inputs:	$\lambda(y_2^c) = \{\mathcal{L}$, temperature, stateMeasurement, Celsius$\}$
	Outputs:	$\lambda(u_2^c) = \{\mathcal{L}$, flow rate, controlDecision, kilogramsPerSecond$\}$
	Parameters:	$\lambda(\zeta_1^c) = \{$room 1, temperature, referenceValue, Celsius$\}$
f_3^c	Inputs:	$\lambda(y_3^c) = \{$ambient, temperature, stateMeasurement, Celsius$\}$
	Outputs:	$\lambda(u_3^c) = \{$room 1, flow rate, controlDecision, percentage$\}$
Post-control processing functions		
f_1^u	Inputs:	$\lambda(u_1^u) = \{$room 1, flow rate, controlDecision, percentage1$\}\}$
	Outputs:	$\lambda(u\prime_1^u) = \{$room 1, flow rate, controlDecision, kilogramsPerSecond$\}$
Actuators		
f_1^a	Inputs:	$\lambda(u_1^a) = \{$west wall 1, flow rate, controlDecision, kilogramsPerSecond$\}\}$
	Outputs:	$\lambda(v_1^a) = \{$west wall 1, heat energy, increase, Joule$\}$
Parameter functions		
f_1^ζ	Inputs:	$\lambda(\zeta_1^\zeta) = \{$room 1, occupancy, stateMeasurement, $\{\top, \bot\}\}$
	Outputs:	$\lambda(\zeta_1) = \{$room 1, temperature, referenceValue, Celsius$\}$
f_2^ζ	Inputs:	$\lambda(\zeta_3^\zeta) = \{$door, opening, stateMeasurement, percentage$\}$

VII. Conclusions and Future Work

We presented SEMIoTICS, a new control system architecture, which enables the utilization of logic-based reasoning over declarative language models of IoT-enabled control system components, for the online re-configuration of feedback control systems. We showed that the system is able to extract configuration options online and implement some logic to evaluate them against pre-defined cost criteria so as to choose the best performing option for the operation of the control system. It is emphasised that there is no pre-requisite for the system to know in advance the instances of the components and their semantic annotations, since these are given online upon deployment. The use of such systems in large-scale buildings can considerably increase the flexibility of adding IoT components in control loops, either through

Table II Results of configuration options and selection

Conf.	Conf. Options	Cost (50-50)	Cost (80-20)	Time (min)
$f_0(\cdot)$	$\{f_1^a, f_1^c, f_1^s\}$	**0.2075**	**0.083**	1.81 – 2.55
$f_1(\cdot)$	$\{f_1^a, f_1^c, f_1^s\}$	0.2065	**0.083**	2.43
	$\{f_1^a, f_1^u, f_3^c, f_4^s\}$	**0.1789**	0.2716	3.67
$f_2(\cdot)$	$\{f_1^a, f_1^c, f_1^s\}$	0.2075	0.0830	
	$\{f_1^a, f_1^u, f_3^c, f_4^s\}$	0.1787	0.2716	
	$\{f_1^a, f_1^c, \{f_3^s, f_5^s\} f_1^\zeta\}$	0.2089	0.0829	
	$\{f_1^a, f_1^c, f_1^y, f_2^s\}$	0.2083	0.0831	
	$\{f_1^a, f_1^c, f_2^y, \{f_1^s, f_2^s, f_3^s, f_4^s\}\}$	0.2085	**0.0828**	5.94
	$\{f_1^a, f_2^c, f_1^s, f_1^\zeta\}$	0.1601	0.2071	
	$\{f_1^a, f_2^c, \{f_3^s, f_5^s, f_4^s\}\}$	0.1757	0.2627	
	$\{f_1^a, f_2^c, f_1^y, f_2^s\}$	0.1594	0.1887	
	$\{f_1^a, f_2^c, f_1^y, f_2^y, \{f_1^s, f_2^s, f_3^s, f_4^s\}, f_1^\zeta, f_2^\zeta\}$	**0.1561**	0.1808	6.58

physical installations or through downloading/importing software functions. It reduces the need for human intervention to components' changes and potentially increases the operation lifetime of a feedback control system.

SEMIoTICS architecture and system has been tested through illustrative scenarios from the smart buildings domain, however, it is potentially applicable to a range of domains where IoT-enabled feedback control loops can be considered. The semantic reasoning process that helps making the re-configuration decision is a combinatorial problem; it involves searching through combinations of linguistic variables' values in a graph, to satisfy certain logical constraints, which comes with increased computational cost and time. Although improvements can be achieved by combining the implementation with appropriate scaling and parallelization of computing resources, the solution may not be applicable to systems that cannot tolerate the re-configuration cost (in terms of time and computational load). It is also clarified that the application of SEMIoTICS in different domains requires the prior update of the knowledge model (not the core schema) so as to consider the domain features of interest and their properties.

Some future topics that we are currently pursuing include: i) Studying the computational complexity of the semantic reasoning process and how this is affected by the number of involved IoT components and their annotations, aiming to understand the scalability characteristics of the solution; ii) Exploiting the capabilities offered by the SEMIoTICS modules so as to achieve the online design and use of controllers for certain control objectives in certain plants with evolving topology and parameters.

REFERENCES

[1] P. J. Antsaklis, B. Goodwine, V. Gupta, M. McCourt, Y. Po Wu, M. Xia, H. Yu, and Z. Feng, "Control of cyberphysical systems using passivity and dissipativity based methods," *Eur. J. Control*, vol. 19, no. 5, pp. 379–388, 2013.

[2] Machine-to-machine Internet-of-Things Connectivity Protocol. Accessed: 2017-01-25. [Online]. Available: http://mqtt.org/

[3] D. Jung and A. Savvides, "Estimating Building Consumption Breakdowns using ON/OFF State Sensing and Incremental Sub-Meter Deployment," in *8th ACM Conf. on Embedded Networked Sensor Systems (SenSys)*, 2010.

[4] T. Samad, "Control systems and the internet of things [technical activities]," *IEEE Control Systems*, vol. 36, no. 1, pp. 13–16, Feb 2016.

[5] M. Mrissa, C. Ghedira, D. Benslimane, and Z. Maamar, *Context and Semantic Composition of Web Services*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 266–275. [Online]. Available: http://dx.doi.org/10.1007/11827405_26

[6] G. Milis, C. Panayiotou, and M. Polycarpou, "Semantically-Enhanced Online Configuration of Feedback Control Schemes," *IEEE Transactions on Cybernetics*, 2017, [to be published]. [Online]. Available: http://ieeexplore.ieee.org/document/7891022/

[7] B. Copigneaux, S. Engell, R. Paulen, M. Reniers, C. Sonntag, and H. Thompson, "Proposal of a European Research and Innovation Agenda on Cyber-physical Systems of Systems – 2016-2025," CPSoS EU FP7-ICT Project (contract no. 611115) Consortium, Tech. Rep., 04 2016.

[8] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and Fault-Tolerant Control*. Berlin Heidelberg: Springer-Verlag, 2016.

[9] J. Stoustrup, "Plug & play control: Control technology towards new challenges," *European Journal of Control*, vol. 15, no. 3-4, pp. 311–330, Aug. 2009. [Online]. Available: http://ejc.revuesonline.com/article.jsp?articleId=13584

[10] J. Bendtsen, K. Trangbaek, and J. Stoustrup, "Plug-and-Play Control—Modifying Control Systems Online," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 1, pp. 79–93, 2013.

[11] S. Riverso, F. Boem, G. Ferrari-Trecate, and T. Parisini, "Plug-and-Play Fault Detection and Control-Reconfiguration for a Class of Nonlinear Large-Scale Constrained Systems," *IEEE Trans. Autom. Control*, vol. 61, no. 12, pp. 3963–3978, Dec 2016.

[12] J. Y. Hung, W. Gao, and J. C. Hung, "Variable structure control: a survey," *IEEE Trans. Ind. Electron.*, vol. 40, no. 1, pp. 2–22, Feb 1993.

[13] A. Haller, K. Janowicz, S. Cox, D. L. Phuoc, K. Taylor, M. Lefrançois, R. Atkinson, R. García-Castro, J. Lieberman, and C. Stadler. Semantic Sensor Network Ontology. [Online]. Available: https://www.w3.org/TR/vocab-ssn/

[14] Sensor Model Language (SensorML). Accessed: 2016-10-21. [Online]. Available: http://www.opengeospatial.org/standards/sensorml

[15] X. Liu, B. Cheng, J. Liao, P. Barnaghi, L. Wan, and J. Wang, "Omidl: An ontology matching framework," *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 580–593, July 2016.

[16] F. Ganz, P. Barnaghi, and F. Carrez, "Automated semantic knowledge acquisition from sensor data," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1214–1225, Sept 2016.

[17] Y. Fathy, P. Barnaghi, and R. Tafazolli, "Distributed spatial indexing for the internet of things data management," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 1246–1251.

[18] R. Toenjes, D. Kuemper, and M. Fischer, "Knowledge-based spatial reasoning for iot-enabled smart city applications," in *2015 IEEE International Conference on Data Science and Data Intensive Systems*, Dec 2015, pp. 736–737.

[19] D. Guinard, V. Trifa, and E. Wilde, "A resource oriented architecture for the web of things," in *2010 Internet of Things (IOT)*, Nov 2010, pp. 1–8.

[20] S. K. Datta, R. P. F. D. Costa, and C. Bonnet, "Resource discovery in internet of things: Current trends and future standardization aspects," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 542–547.

[21] I. Chatzigiannakis, H. Hasemann, M. Karnstedt, O. Kleine, A. Kröller, M. Leggieri, D. Pfisterer, K. Römer, and C. Truong, "True self-configuration for the IoT," in *2012 3rd IEEE Int. Conf. on the Internet of Things*, Oct 2012, pp. 9–15.

[22] I. Grangel-González, L. Halilaj, G. Coskun, S. Auer, D. Collarana, and M. Hoffmeister, "Towards a semantic administrative shell for industry 4.0 components," in *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, Feb 2016, pp. 230–237.

[23] K. Suri, W. Gaaloul, A. Cuccuru, and S. Gerard, "Semantic framework for internet of things-aware business process development," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, June 2017, pp. 214–219.

[24] K. Kotis and A. Katasonov, "Semantic interoperability on the internet of things: The semantic smart gateway framework," *Int. J. Distrib. Syst. Technol.*, vol. 4, no. 3, pp. 47–69, Jul. 2013. [Online]. Available: http://dx.doi.org/10.4018/jdst.2013070104

[25] D. Bonino and F. Corno, *DogOnt - Ontology Modeling for Intelligent Domotic Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–803. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88564-1_51

[26] A. Pablo, R. Valiente, and A. Lozano-Tello, "Ontology and SWRL-Based Learning Model for Home Automation Controlling," in *Ambient Intelligence and Future Trends-Int. Symposium on Ambient Intelligence (ISAmI 2010)*, J. C. Augusto, J. M. Corchado, P. Novais, and C. Analide, Eds. Berlin: Springer Berlin Heidelberg, 2010, pp. 79–86.

[27] P. Pauwels and D. Van Deursen, "IFC-to-RDF: adaptation, aggregation and enrichment," in *First Int. Workshop on Linked Data in Architecture and Construction, Abstracts*, 2012, pp. 1–3. [Online]. Available: http://multimedialab.elis.ugent.be/ldac2012/documents/LDACworkshopreport.pdf

[28] L. Z. Raja and R. A. Issa, "Ontology-based partial building information model extraction," *Journal of Computing in Civil Engineering*, vol. 27, no. 6, pp. 576–584, 2013.

[29] R. M. G. Ferrari, H. Dibowski, and S. Baldi, "A message passing algorithm for automatic synthesis of probabilistic fault detectors from building automation ontologies," in *20th IFAC World Congress*, Jul 2017, pp. 4268–4274.

[30] E. Y. Rodin, "Semantic Control Theory," *Applied Mathematics Letters*, vol. 1, no. 1, pp. 73–78, 1988.

[31] C. Joslyn, "Semantic control systems," *World Futures: Journal of General Evolution*, vol. 45, no. 1-4, pp. 87–123, 1995.

[32] M. Boasson, "Control systems software," *IEEE Trans. Autom. Control*, vol. 38, no. 7, pp. 1094–1106, 1993.

[33] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building Operating System Services," in *Proc. 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013.

[34] B. Russel, "Mathematical logic as based on the theory of types," *American Journal of Mathematics*, vol. 30, pp. 222–262, 1908.

[35] J. Ferreiros, "The road to modern logic-an interpretation," *Bulletin of Symbolic Logic*, vol. 7, no. 4, pp. 441–484, 12 2001. [Online]. Available: http://projecteuclid.org/euclid.bsl/1182353823

[36] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., *The Description Logic Handbook*. Cambridge University Press, 2003.

[37] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. (2004) OWL-S: Semantic Markup for Web Services. Accessed: 2017-07-24. [Online]. Available: https://www.w3.org/Submission/OWL-S/

[38] GeoSPARQL - A Geographic Query Language for RDF Data. [Online]. Available: http://www.opengeospatial.org/standards/geosparql

[39] C. Griffiths and M. Costi, *GRASP : the solution*. Cardiff, UK: Proactive Press, 2011.

[40] D. Conover, D. Crawley, S. Hagan, D. Knight, C. Barnaby, C. Gulledge, R. Hitchcock, S. Rosen, B. Emtman, G. Holness, D. Iverson, M. Palmer, and C. Wilkins, *An Introduction to Building Information Modeling (BIM) - A Guide for ASHRAE Members*. Amer. Soc. of Heating, Refrig. and Air-Cond. Eng., 2009.