Semantically-Enhanced Online Configuration of Feedback Control Schemes

Georgios M. Milis, Christos G. Panayiotou and Marios M. Polycarpou, *Fellow, IEEE*

***Abstract*—Recent progress towards the realization of the "Internet of Things" has improved the ability of physical and soft/cyber entities to operate effectively within large-scale, heterogeneous systems. It is important that such capacity be accompanied by feedback control capabilities sufficient to ensure that the overall systems behave according to their specifications and meet their functional objectives. To achieve this, such systems require new architectures that facilitate the online deployment, composition, interoperability and scalability of control system components. Most current control systems lack scalability and interoperability because their design is based on a fixed configuration of specific components, with knowledge of their individual characteristics only implicitly passed through the design. This work addresses the need for flexibility when replacing components or installing new components, which might occur when an existing component is upgraded or when a new application requires a new component, without the need to readjust or redesign the overall system. A semantically-enhanced feedback control architecture is introduced for a class of systems, aimed at accommodating new components into a closed-loop control framework by exploiting the semantic inference capabilities of an ontology-based knowledge model. This architecture supports continuous operation of the control system, a crucial property for large-scale systems for which interruptions have negative impact on key performance metrics that may include human comfort and welfare or economy costs. A case-study example from the smart buildings domain is used to illustrate the proposed architecture and semantic inference mechanisms.**

***Index Terms*—cyber-physical systems, feedback control, internet of things, semantic composition, semantic knowledge models.**

WE currently live in the "smart era" where people and machines intelligently interact in work and social ecosystems [1]. This interaction is facilitated by a variety of smart machines, from small personal devices to hardware and software-equipped smart buildings [2], [3], to even larger and more complex systems such as electric power grids [4]. The high penetration of smart portable and embedded devices connected to networks, suggest a future where machines will interact with each other and the environment, in a context-aware framework [5]. Thus, new sensing, actuation and control capabilities can be created.

In this context, consider the case of a smart building, equipped with various sensors, actuators and controllers, designed to maintain the comfort and safety of the inhabitants, while reducing operating costs. A smart building may consist of various sub-systems that control the lighting, the temperature and the humidity, the condition of the air, fire alarm systems, sprinkler systems and many more. In most cases, the design of these systems is based on a fixed configuration of specific components, with certain knowledge of their specific characteristics. For example, specific temperature sensors and heaters may be part of the control operation of a heating, ventilation and air-conditioning (HVAC) system, but in general any other system in the same building ignores their existence. This causes lack of reconfigurability and interoperability for the control components, thus considerably limiting the potential capabilities and lifetime of the overall control system. A truly smart control system should be able to respond to an evolving environment by reconfiguring itself online and effectively utilizing the available components so that the technical and economic control objectives are not compromised. For instance, the control system may utilize an independent electric heater if it becomes aware that, at a certain time, the HVAC requires that actuation capacity to effectively control the temperature. The emerging maturity of the Internet of Things (IoT) paradigm and the subsequent proliferation of embedded systems with wireless connectivity has made it possible to meet this requirement.

The main contribution of this work is the design of an innovative semantically-enhanced architecture for online configuration of feedback control schemes. The proposed architecture can be adopted in a class of dynamic systems and it is general enough to be applicable to various application domains, though in this paper emphasis is given to smart building applications. We present a mechanism, which allows the systematic modelling of available expert knowledge about the domain in which the control system operates, as well as about the control system design variables. The mechanism enables the on-line selection of components and can be exploited for automatic (semantically viable) closed-loops' reconfiguration when changes are detected in the pool of components.

The article is organized as follows. Section I offers insights into related work. Section II formulates the problem being considered, with reference to the introduced architecture. Then, Section III discusses the relevant semantic (knowledge) modelling. This is followed by Section IV that presents the respective semantic models of individual control system components, as well as the pairwise composition models in a feedback loop orchestration. Section V follows, presenting specific composition example cases from the buildings domain. Finally, Section VI concludes the article and discusses future directions.

The authors are with the KIOS Research Center for Intelligent Systems and Networks, Department of Electrical and Computer Engineering, University of Cyprus, {milis.georgios, christosp, mpolycar}@ucy.ac.cy

This work is partially funded by the European Research Council (ERC) under the project "Fault-Adaptive Monitoring and Control of Complex Distributed Dynamical Systems".

I. Related Work

A lot of work has been undertaken to date by the control community towards designing control algorithms with online adaptation capabilities that aim to facilitate the flexibility of the control system to accommodate system uncertainties and/or time variations. For instance, approaches to designing fault-tolerant control systems are presented in [6], while the textbooks [7] and [8] comprise comprehensive information sources on approaches to design adaptive controllers. Combining adaptation capabilities with online learning of unknown system dynamics, has been also addressed in research; the textbook [9] discusses general methodologies for designing adaptive approximation-based control systems. Adaptive approximation-based techniques have been utilised also recently, using neural networks [10], as well as fuzzy logic [11], [12] in designing controllers and/or approximators for certain classes of nonlinear systems with unknown parameters. The authors in [13], [14] and more recently in [15], have worked towards adapting to and accommodating online changes in the system's dynamics and order (e.g., when new components are plugged in a closed-loop system), for linear systems. A more recent approach to the control reconfiguration and fault detection in non-linear systems can be seen in [16].

The work discussed above, deals with the design of controllers with pre-defined structure, which however pose the required adaptation capabilities so as to maintain satisfactory performance in the presence of certain system uncertainty and changes. Nevertheless, there are cases where the design of a single-structure controller is not practical or it cannot satisfy the control objectives. Such cases are addressed by another research area, the "variable structure control" [17]. This area assumes that the control system switches among several predefined control structures that take over when certain criteria are met (e.g., when a regulated state slides to the bounds of specific areas in the state-space). The switching between control structures is either based on deterministic criteria or even follows a stochastic process [18]. This type of systems are a special case of "hybrid dynamical systems" that have been studied extensively in the literature [19], also in terms of their reconfiguration strategies for active fault accommodation [20].

An essentially complementary approach from a different perspective, is the design of modular architectures. An early effort towards introducing the need for a modular architecture for the control system design and the concept of structuring the data exchanged based on their basic semantic description, has been discussed in [21]. More recent efforts [22], though not focusing specifically on the control problem, address the modelling of cyber-physical systems as collections of services offered by specific physical devices and cyber tools within certain time and domain context. These services are then invoked, based on their standard descriptions for online composition of services, to perform specific tasks. Moreover, the authors in [23] are taking advantage of the cloud computing paradigm to offer middleware solutions that allow abstraction of services of physical units for online invocation. An equally interesting approach is presented in [24], [25] where a *Building Application Stack (BAS)* and a *Building Operating System Services (BOSS)* architecture are introduced, allowing the development of portable control applications for buildings, decoupled from hardware and building specificity. That work deals with the interoperability of building components, implementing metadata models and fuzzy queries for abstraction.

In order to deal with the interoperability in the modular (component-based) architectures discussed above, the research community investigated the use of semantic knowledge models. The explicit incorporation of semantics in the control systems design has been proposed in [26]. The semantic knowledge models are considered in that article as an artificial intelligence tool, required to be used for the modelling of knowledge about plants, control system components and control goals. An architecture is proposed, which allows the control system to reason upon the modelled knowledge and select appropriate control systems from a pre-existing pool. The concept of expert knowledge modelling in control systems is also used in [27], mainly focusing on the semantic validation of simulation software. A more system theoretic approach to presenting the concepts of semantic control systems can be found in [28], which discussed the notion of semantic rules as mappings between physical phenomena and explained that control is in general rule-following, be it functional or semantic rules. Furthermore, concrete examples of using semantic knowledge models in the smart buildings domain, are observed in the literature. For instance, the "DOGont" ontology [29] deals with the current issues of vendors' and technologies' heterogeneity in domotic environments. Also, the authors in [30] describe an ontology-based expert system that is able to transparently control the home automation processes by learning from the human users' behaviour.

Our work is complimentary to what discussed above and advances the state of art by combining existing tools from different areas and offering explicit solutions to identified gaps. We address in general a variable structure control problem, since we assume that there is no pre-designed single controller that poses all required adaptation capabilities to learn and accommodate the changes (e.g., in case of faults) in the controlled plant and/or the control system components. We consider a plug-and-play capability in our proposed solution, however, not in terms of built-in adaptation capability to changes due to newly plugged components; we implement a mechanism that enables the control system to switch online between different configurations. Unlike typical switching systems where the transition happens when the system state enters a pre-defined region of the state-space, our switching decision is explicitly made and enforced through a discrete-state logic-based system. The explicit design of this mechanism is a key contribution. This mechanism assumes that the available control modes (configurations) are not defined in advance; they are composed online using components from an evolving pool of control system components, able to interoperate in a modular architecture but without hard-coded interfaces. At discrete events, the logic-based system exploits modelled knowledge and subsequent (standards' based) semantic characterisations of components' interfaces, as well as inductive inference rules to produce the switching signal. These are further clarified in

the sequel.

II. Problem Formulation

Large-scale complex systems typically consist of a large number of components and subsystems and they evolve over time as new technology becomes available, leading to the addition of new or replacement of existing components. For example, in the context of a smart building, new sensors (e.g., occupancy sensors or CO and CO_2 sensors) or new actuators (e.g., a heater or a humidifier) may be plugged-in and become available. In this case, it is desirable that the building control system reconfigures itself and integrates the new components in the overall control scheme, so as to ensure continuous operation and satisfaction of control specifications (e.g., user safety and comfort).

In order to achieve the above objectives, a *cognitive agent*, denoted as Σ, is proposed. This agent is responsible for maintaining an expert knowledge base, performing logical reasoning tasks, making decisions and configuring various components such that the operation objectives are met. The agent Σ first detects and identifies the changes in the availability of components. With the emerging maturity of the "Internet of Things", this requirement is addressed by established message formatting standards and communication protocols. Subsequently, Σ should become aware of the characteristics and capabilities of the new components. This is exactly the emphasis of this work, proposing the use of *ontological knowledge models* and *semantic characterisation and reasoning* techniques [31], [32]. Thus, each component can be annotated with certain "tags" that describe its characteristics and capabilities. Once this information is received by Σ, it is integrated with the existing knowledge available about the overall system, together with information that can be obtained from experts, normal users and/or the Internet. The new knowledge is then utilized to reconfigure the existing feedback control scheme, considering currently available components.

The general architecture is shown in Fig. 1, where $I \in \{0, 1, 2, ...\}$ is the index of the possible feedback control scheme configurations. In the proposed architecture, a control scheme for a given "Plant" is considered (e.g., a room of a larger building, characterised by certain state variables like temperature, humidity, lighting, air quality, etc.) that need to be controlled to satisfy certain control objectives, e.g., maintain safe and comfortable conditions for the users. Other example of a "plant" can be a window or door characterised by its opening state. The plant's state is monitored by a set of sensors $\mathcal{S}^{(I)}$, e.g., thermistors, occupancy sensors, limit switches, smoke detectors and so on. Furthermore, a set of actuators $\mathcal{A}^{(I)}$ is considered, able to act on the plant and affect its state. For instance, a heating element may be used to adjust the temperature, a lamp to adjust the lighting, a motor to open a window.

In practice, the actuators are typically driven by automatic control schemes which take measurements from sensors and compare them to reference signals in order to generate the control signals to be fed to the actuators. The control algorithms can range in complexity from a simple ON/OFF control

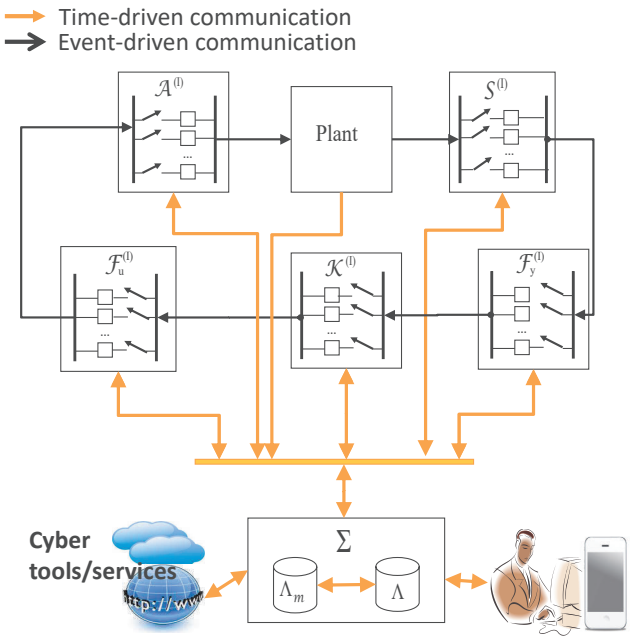


Figure 1. System Architecture for the configuration I. The large boxes indicate the set of plants $\mathcal{P}^{(I)}$ that are controlled, the set of used sensors $\mathcal{S}^{(I)}$, the set of used actuators $\mathcal{A}^{(I)}$, the set of used controllers $\mathcal{K}^{(I)}$ and the sets of used transformation functions $\mathcal{F}_y^{(I)}$ and $\mathcal{F}_u^{(I)}$. Small boxes indicate the individual components in these sets. Switches indicate the selection possibility at each different configuration I. The time-driven communication concerns vectors of signals each time with appropriate dimension. The event-driven communication comprises cyber two-way communication with components and it also transfers the selection decisions to switches. Note that communication with the plant is one-way since any knowledge about it, comes from human or cyber sources)

scheme to PID control to adaptive control algorithms, etc. In certain cases, the signal produced by a sensor may not be compatible with that which is required by a controller, or the signal produced by a controller may not be compatible with the signal required by an actuator. This may be particularly relevant when a new component is added, e.g., when a new temperature sensor measuring in degrees Fahrenheit is added to a closed-loop control system configured to use degrees Celsius. In such case, the sensor signal can be processed through a transformation function, that is an element of a set of functions denoted by $\mathcal{F}_y^{(I)}$, before fed to the controller(s) from the set $\mathcal{K}^{(I)}$. The control signal can also be processed through another set of transformation functions denoted by $\mathcal{F}_u^{(I)}$ before fed to the actuators. Typically, the control of a plant, given certain control objectives, may be comprised of several feedback loops as indicated in Fig. 1 for the configuration I.

Expert knowledge about the available components is given to the *cognitive agent* Σ and is stored in its semantic database Λ. This database may also contain additional information obtained from users and the Internet, as well as new implicit knowledge created by applying logical inference mechanisms on pre-existing knowledge. To facilitate the latter, the agent Σ has access to a second database, Λ_m, which contains semantic rules that model experts' reasoning through semantic mappings and support the inference mechanism. Basic definitions about the notion of these rules can be found in [28].

Every time a component is introduced or removed, agent Σ becomes aware of the change. In the case of introducing a new component, Σ obtains information about its functionality, properties and characteristics (through semantic annotations), as well and adds it to its semantic database. The information may be coming from the component itself, through appropriate communication protocols and/or from the users or discovered via the Internet. The communication of agent Σ with all components is event-driven and is indicated in Fig. 1 by orange lines. At the occurrence of every relevant event, where new knowledge about components becomes available, Σ guides the transition from configuration I to $I+1$ by selecting the appropriate sensors, actuators, controllers and transformation functions and setting their relevant parameters.

In summary, in order to perform the change in the configuration from I to $I+1$, the tasks of agent Σ are broken down as follows: **Task 1:** Understand the context/environment where the entire system operates; **Task 2:** Detect and identify the available components; **Task 3:** Acquire information about the functionality, properties and characteristics of each component; **Task 4:** Turn the available information into knowledge that can be used to determine what components to use in order to synthesize the necessary feedback-control loops; **Task 5:** Select the appropriate components and set their parameters such that certain performance objectives are met.

In the scope of this work, for the purpose of illustrating the proposed architecture in an application domain instead of in an abstract formulation, it is assumed that the context is a smart building with a floor-plan already available. Furthermore, it is assumed that each component has wireless connectivity and implements some node discovery and communication protocol. Thus, this work considers **Task 1** and **Task 2** solved a-priori, since information about the environment is assumed to have been imported in advance, while detection of new components is assumed solved by using standard network communication protocols. The emphasis is on **Task 3** and **Task 4**. **Task 3** is addressed through the semantic modelling and characterization of components described in Sections III and IV. Semantic characterizations are then exploited to solve **Task 4** as presented in Section V. Finally, exploring additional automation methods to address **Task 5** is for future investigation, while in this work some static solutions and assumptions are adopted.

III. Knowledge Model

The reader is taken through an illustrative scenario aiming at clarifying the way the semantic database is built and how the semantic inference is performed over the stored knowledge-facts to implement the decision mechanism for the online composition of the feedback-loop scheme.

Consider the scenario of an office as in Fig. 2, where a digital sensor device (s_1) is deployed, measuring the temperature of the office (T_1) in degrees Celsius. The office is also equipped with a heating device (a_1) which introduces heat to the office at some energy rate in kW. In addition, a controller k_1 is deployed to regulate the temperature of the office by controlling the operation of the actuator. There is also a window on the east wall of the office, through which energy losses (E_{loss}) can potentially appear, however, it is initially considered fully closed and made of material with zero heat transfer coefficient.

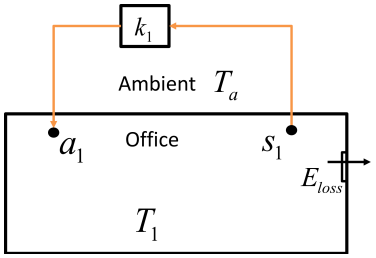


Figure 2. An open-plan office with inner temperature T_1, ambient temperature T_a, a deployed temperature sensor s_1, one heating actuator a_1 and a temperature controller k_1

Considering the office with the dynamics of its temperature state, we describe next the "things" that this plant contains. There is one temperature sensing device, one heating actuator, one controller, as well as the physical properties "temperature" and "energy", the measurement units "Celsius" and "kW", as well as the locations "office" and "ambient". All these are linguistic terms used by humans to refer either to cyber-physical entities or other types of entities and real world phenomena and concepts. From a knowledge perspective, any plant can be considered as a set of "things" (knowledge objects), defined as follows:

Definition 1. *Things*: $\mathcal{T} = \{t_i | i = 1, \cdots, n_{\mathcal{T}}\}$ is the finite set of all "things" (linguistic terms) which an expert would use as a convention, to describe what exists in a subject plant. (The terms "individuals" and "instances" are also used to refer to these "things", in the ontological engineering domain.)

The "things" are not all of the same type. The type of each "thing" is an important part of the knowledge which experts assume in order to share the same meaning. For instance, one would not be able to understand the description without knowing that the linguistic term "Celsius" refers to a unit of measurement, while the linguistic term "office" denotes a location. Therefore, a convention about the types of "things" is necessary and defined as follows:

Definition 2. *Classes*: $\Omega = \{\omega_i | i = 1, \cdots, n_\Omega, \omega_i \subseteq \mathcal{T}\}$ is the set of all types/classes to which the "things" belong. Note that each element of Ω is a subset of the set $\mathcal{T}$.

Elements in Ω could be, for instance, the set of sensors ($\mathcal{S}$), the set of actuators ($\mathcal{A}$), the set of controllers ($\mathcal{K}$), the set of locations ($\mathcal{L}$), the set of physical properties ($\mathcal{Q}$), as well as the set of measurement units ($\mathcal{M}$). Since this work is dealing with online compose-ability of control system schemes, these are further described in terms of their "inputs" and "outputs", therefore, two additional classes of "things" are defined for convenience: Set $\mathcal{U} \in \Omega$ represents all inputs of components, while set $\mathcal{Y} \in \Omega$ represents all outputs of components. The use of these two sets will be clarified in Section IV. In general, each "thing" may belong to more than one classes, however, for simplicity it is considered in this work that each one

belongs to only one class.

Further to the classes of "things", the full understanding of the meaning is facilitated by the (semantic) relations between them. That is, "Celsius" ($m_1 \in \mathcal{M}$) is a measurement unit of "temperature" ($q_1 \in \mathcal{Q}$), while $s_1 \in \mathcal{S}$ is a "sensor" that measures "temperature". Such logical relations, comprise a third convention assumed by experts when sharing knowledge. In general, relations comprise mappings between "things" of one class to "things" of another and their definition is given below:

Definition 3. *Relation-graph*: $G(\mathcal{V}^o, \mathcal{V}^d, E_n^{(\mathcal{V}^o,\mathcal{V}^d)})$ defines a non-balanced graph with vertices being the elements of the sets $\mathcal{V}^o, \mathcal{V}^d \in \Omega$ and edges being the elements of the set $E_n^{(\mathcal{V}^o,\mathcal{V}^d)} = \{(v_i^o, v_j^d) | v_i^o \in \mathcal{V}^o, v_j^d \in \mathcal{V}^d, i \in \{1, ..., n_{\mathcal{V}^o}\}, j \in \{1, ..., n_{\mathcal{V}^d}\}\}$, which represent the arcs connecting elements of the origin set $\mathcal{V}^o$ to elements of the destination set $\mathcal{V}^d$. (The term "properties" and "relations" are used in the ontological engineering domain).

Typically, a "Relation-graph" is bipartite, i.e., the sets $\mathcal{V}^o$ and $\mathcal{V}^d$ comprise distinct classes of "things". However, in the case when the "Relation-graph" describes mapping between "things" of the same class (e.g., two distinct locations), then the origin and destination sets of vertices converge to one. Furthermore, in the case where multiple relations are required and defined between the same pair of classes, a subscript $n = 1, 2, \cdots$ is used with the set of edges E in the above definitions, in order to differentiate between the relations.

Fig. 3 shows some example "Relation-graphs" that represent various relations between "things". The nodes of the graphs represent "things" while their classes are shown with dashed-line rectangle containers. An edge that connects two "things" represents the relation that exists between them. The shown edges represent six different "Relation-graphs", however, for readability purposes, four of them are highlighted with different colours.

Multi-edge paths between "things" represent "composite" relations, i.e., relations not explicitly defined in advance. For example, the bipartite graph $G(\mathcal{Y}, \mathcal{Q}, E^{(\mathcal{Y},\mathcal{Q})})$ can be considered as formed with vertices from the sets $\mathcal{Y}$ and $\mathcal{Q}$ and edges the paths of length 2 from nodes in $\mathcal{Y}$ to nodes in $\mathcal{Q}$ (e.g., the path (y, m, q). This is a relation that is derived from the composition of two other relations.

Since the relations are defined from specific origin class to specific destination class, the respective graphs are directed in general. However, in most cases the converse relation also exists, which results in bidirectional (or non-directional) graphs. For instance, considering the illustrative example introduced earlier, saying that sensor s_1 is "located in" $l_1 =$ office, would have the same meaning as saying l_1 "contains" sensor s_1. Wherever the converse of relations utilized in this work are meaningful, the edges are shown as non-directional.

Several relations can be defined between different classes of "things". The union of all resulting "Relation-graphs" defines a super-graph, which in combination with the "things" and their "classes" comprises the semantic database Λ introduced earlier.

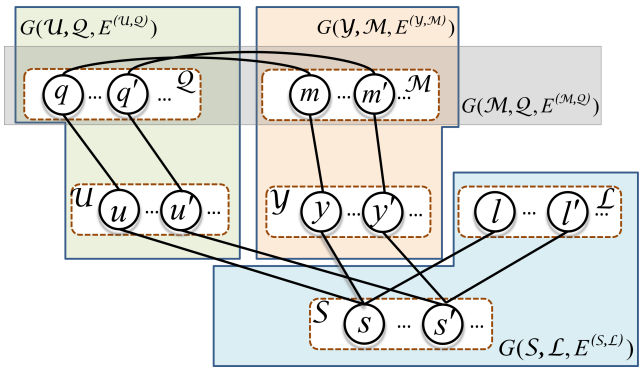


Figure 3. Representation of relation graphs: The graph $G(\mathcal{S}, \mathcal{L}, E^{(\mathcal{S},\mathcal{L})})$ is highlighted with light blue colour, the graph $G(\mathcal{U}, \mathcal{Q}, E^{(\mathcal{U},\mathcal{Q})})$ is highlighted with light green, the graph $G(\mathcal{Y}, \mathcal{M}, E^{(\mathcal{Y},\mathcal{M})})$ is highlighted with light orange and the graph $G(\mathcal{M}, \mathcal{Q}, E^{(\mathcal{M},\mathcal{Q})})$ is highlighted with light grey. The "prime" superscript is used to illustrate the potential of having multiple "things" in each class.

IV. Semantic Models of Components in a Feedback Control Scheme

The knowledge (semantics) about the components of a feedback control scheme, is encoded in a layered hierarchical view. To determine this view, the input-output nature of all components is exploited. The following four layers are used: i) the "Scheme Components" layer which contains the "things" that represent the actual implementations of control-related components (i.e., plant, sensor, controller, actuator), ii) the "Inputs-Outputs-Locations" layer which contains all inputs and outputs of every component, as well as the "Locations" sub-layer that encodes the physical location associated with each component directly or indirectly, iii) the "Thematic Knowledge" layer which contains "things" encoding all other domain specific knowledge (physical properties and measurement units in this work). This is used to characterise the components and subsequently their inputs and outputs, and iv) the "Functions" layer which hosts "things" representing any type of signal processing functions that, e.g., transform one measurement unit to another.

In order to introduce and help clarify the semantic composeability of feedback control schemes, the semantic models of components are first graphically introduced in the following subsection. Additional emphasis is given to the "Locations" sub-layer, since locations comprise a key property in identifying the cyber-physical components that are more appropriate to be connected together.

A. Semantic models of components

1) Plant: A core type of component in a control scheme is the "plant". In general, a plant is a component described by the dynamics of a certain set of state-variables. These dynamics are affected through input-variables (controlled and/or uncontrolled) and the effect is measured via certain output-variables (measurable functions of state-variables and/or parameters) through a certain transfer function. The controlled input-variables are computed by a controller and produced by actuators, while the output-variables can be measured by sensors.

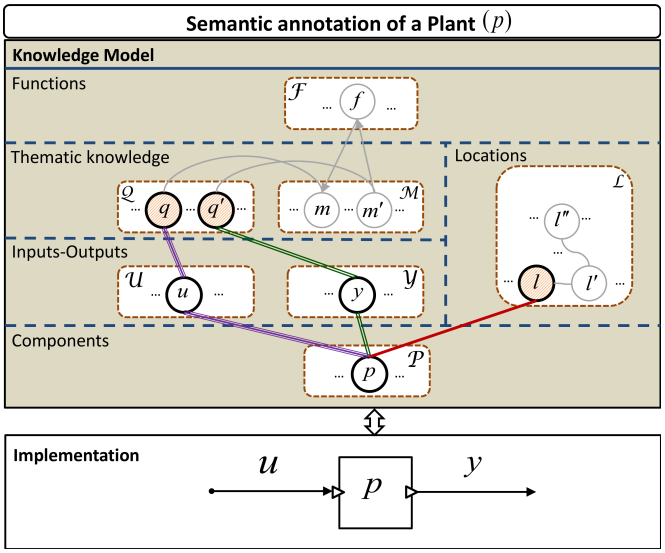


Figure 4. The semantic model of a certain "plant" p, comprising a tree with one input-branch (through a certain input u), one output-branch (through a certain output y) and one location-branch (through a certain location l).

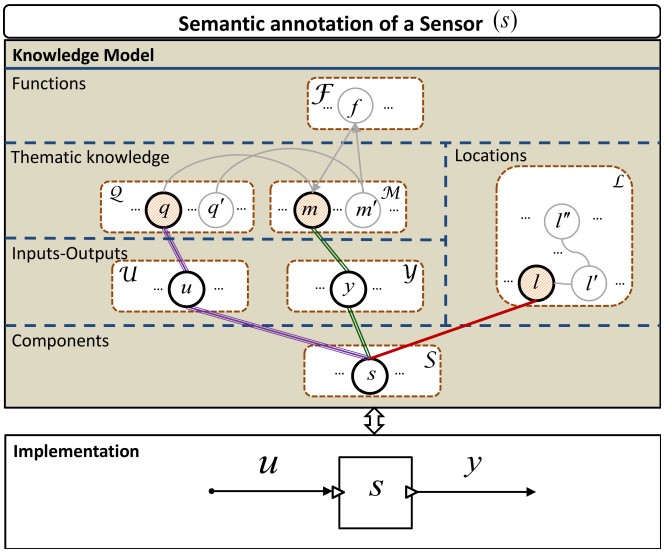


Figure 5. The semantic model of a certain "sensor" s, comprising a tree with one input-branch (through a certain input u), one output-branch (through a certain output y) and one location-branch (through a certain location l).

To simplify the presentation of semantic modelling for feedback control schemes, this work considers plants with one input-variable, which is mapped to an output-variable and is associated with a certain location. This is represented by the snapshot of the semantic database shown in Fig. 4. In general, multiple input-variables and output-variables can be considered, simply by introducing more nodes of class $\mathcal{U}$ and $\mathcal{Y}$, respectively. The semantic model of a plant consists of a *tree* that has as *root* the plant node. For single-input, single-output (SISO) plants, the tree has three branches, the "input-branch" (marked by triple purple edges) the "output-branch" (marked by double green edges) and the "location-branch" (marked by single red edges). The input branch identifies the input-variable to the plant, the output branch identifies the output-variable, while the location branch identifies the location associated with the plant. In the figure, nodes q and q' are referred to as the "input-branch-leaf-node" and "output-branch-leaf-node" respectively, while the location node l is referred to as the "location-branch-leaf-node" of the plant model tree (all leaf nodes are filled with light orange diagonal pattern). For example, let the plant ($p = p_1$) be the office of Fig. 2 with its temperature dynamics. This has a single input and a single output. The output y may represent the measurable office temperature and the input u may represent the heat introduced in the office. The plant is also associated with the location $l = l_1 =$ office.

2) Sensor: Adopting the same input-output view of components, a sensor can be viewed as a component that receives as input some physical property and produces as output a signal or a time series measured in some measurement units. Fig. 5 illustrates the semantic model of a sensor s. Similarly to the "plant" the semantic model tree associated with the sensor has "thing" s as a root node and three branches: the input-branch with the physical property that the sensor measures as an input-branch-leaf-node q, the output-branch with output-branch-leaf-node being the measurement unit m produced by the sensor and the location-branch with l, the location where the sensor is installed, as the location-branch-leaf-node. For example, the semantic model of the temperature sensor in the office has its input associated with the physical property $q = q_1 =$ temperature, its output associated with the measurement unit $m = m_1 =$ Celsius, while the device is located in the office ($l = l_1 =$ office).

In general, each measurement unit is always associated with a physical property, therefore there is also a formed "Relation-graph" that involves objects of type physical property and objects of type measurement unit. The edges of this graph are indicated by gray line in Fig. 5, since they are not participating in the semantic model of the sensor itself.

3) Actuator: An actuator is a device that receives an input command or an input signal by a controller and produces an output signal that affects a physical property. For example, a heater, depending on its capabilities, may receive an input command "0 or 1" to simply turn on or off or in the range "$0, \cdots, 100$" and produce a signal in kW, representing the energy in the form of heat. The semantic model of the actuator is not represented graphically, however, it is very similar to the model of a sensor. Its input-branch has a leaf node m which characterizes the input signal that the actuator can receive, the output-branch has as leaf node the measurement unit m' that characterises the output signal produced and the location-branch has as leaf node the location where the actuator is installed l.

4) Controller: The controller is also viewed as an input-output component, where the time series input is usually a signal produced by a sensor and generates an output command or an output signal, fed to the actuator. The semantic model of a controller is shown in Fig. 6.

The controller "thing" k is the root node and there is one input- and one output-branch, as well as a location-branch. The input corresponds to a signal coming e.g., from

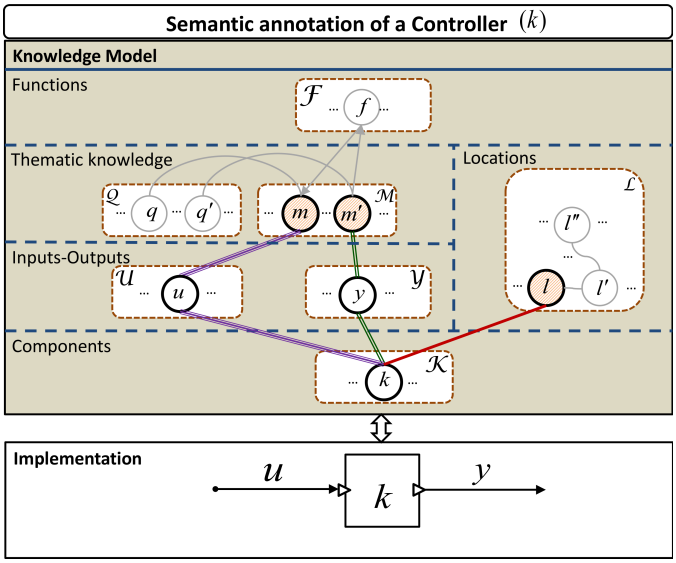


Figure 6. The semantic model of a certain "controller" k, comprising a tree with one input-branch (through a certain input u), one output-branch (through a certain output y) and one location-branch (through a certain location l.

a sensor output, thus an input-branch-leaf-node m of type measurement unit is considered. The output corresponds to a signal/command to be given to an actuator, therefore another measurement unit m' is considered as the output-branch-leaf-node. The controller component is assumed to be a software, with no location property of its own. However, the controller is typically considered in relation with a specific plant (either due to its design or due to its potential deployment) and as such it inherits the location properties associated with the plant p. This results in the use of the location-branch-leaf-node l. For instance, the model of plant $p = p_1$ discussed earlier, already defines the office as its associated location and this will be inherited by any controller adopted in the feedback control scheme, giving $l = l_1 =$ office. For completeness, it is noted that beyond the time series' inputs, controllers may have additional inputs for re-configurable parameters, as well as special inputs for reference signals. The values for these "parameter-inputs" and reference signals may be measured by installed sensing devices or given (updated) by humans and/or third party (Internet) services.

For the purposes of this work, a small set of simple controllers is considered, which are introduced in the sequel. Incorporating more advanced controllers with different criteria is also possible, and in fact, this is one of the main advantages of the proposed methodology and architecture. It is also emphasised that the instantiations of the semantic models of the components can be either imported manually or retrieved from pre-defined Internet sources hosting the structured information in the form of "semantic drivers" of components (e.g., during design of buildings). In essence, what is achieved by the proposed modelling, is the encoding of the required knowledge in machine readable format. Assuming a repository of given (cyber and/or physical) components with their semantic models, the objective is for the agent Σ to be able to choose a subset of them and ensure viable compositions of feedback control schemes that also meet the pre-defined application and control objectives. The compositions (connections) of the components are discussed in the next sub-section.

The semantic database developed and used in this work for the presentation of the concept, has been implemented as a lightweight, application-independent and easily manageable OWL/RDF ontology [33], that is, a standard format (XML-like text) representing knowledge facts in the form of triplets $\langle$ *subject - property - object* $\rangle$. The developed semantic database allows the incorporation of existing standard ontology frameworks that allow semantic composition of services/-components, e.g., OWL-S, [34], or frameworks for semantic annotation of RESTfull services [35]. It may also be combined with ontologies dedicated for sensor annotations such as the "SSN" by the $W3C$ Semantic Sensor Network Incubator Group (SSN-XG) [36] and the "SensorML" standard [37] by the Open Geospatial Consortium (OGC), as well as other domain specific ontologies (e.g., DOGont, [29] in the smart buildings context), so as to serve a wider range of intelligent applications.

B. Semantic composition of components

The semantic composition of the components, first of all considers the classes (types) of the components and their typical role in the closed-loop scheme as indicated in Fig. 1. That is, a sensor will use as input a certain physical signal of the plant, a plant's (controlled) input will be produced by a relevant actuator and so on. Second, the composition considers the "semantic matching" of the components, that is, the output-input matching for consecutive pairs of components, based on their semantic models. In general, the existence of a path from one "thing" (node) to another when traversing the semantic database (super-graph Λ), indicates a semantic relation for the two "things". A human expert would identify the useful paths (relations) that should exist in order to form a meaningful connection between an output of one component to the input of another. This cognitive process of the human expert, needs to be codified in "semantic rules" [28], in order to enable the agent Σ to reason upon the semantic database and infer the meaningful connections. The database of available rules is denoted as Λ_m. At this stage in our work we do not attempt to present the rules in a formal language; instead, we use the graphical representation of the semantic database to describe the rules, as presented in the sequel.

The "semantic matching" is essentially exploited on an output-input basis and is achieved *if and only if* there are meaningful relation paths from the output-branch-leaf-nodes of one component to the input-branch-leaf-nodes of the other component *and*, in addition, there is a meaningful relation path between the location-branch-leaf-nodes of the two components through the existing relation graphs that connect different locations in the semantic database. Possible relations between physical properties and measurement units (including composite relations) can be utilized, while for the location matchings, relations between location nodes may be utilized as demonstrated next.

1) Location relations: Given an overall system (e.g., a building), a number of relative locations and the relations between these locations may be introduced. For example, a building is located in some region and it consists of several rooms. Within each room, several sub-locations are identified. Then, one relation may represent that a location "is part of" another location while its inverse relation, i.e., a location "contains" another location is also defined. This relation is represented by $G(\mathcal{L}, \mathcal{L}, E_1^{(\mathcal{L},\mathcal{L})})$, using the notation of "Definition 3". Another relation is that a location "is adjacent to" some other location which is also bi-directional and is represented by $G(\mathcal{L}, \mathcal{L}, E_2^{(\mathcal{L},\mathcal{L})})$. Other relations between the location objects may also be defined as needed. Location-relations comprise a very important property of cyber-physical components, as their operation is often highly associated with their spatial properties.

2) Actuator-Plant-Sensor Matching: The first task of the agent Σ is to find the actuators able to affect the inputs of the plants that require control, as well as the sensors that can actually measure the outputs and other required parameters. The actuator-plant composition is shown in Fig. 7. It can be seen that the matches are shown by formed loops, while the leaf-nodes that participate in the loops are filled with orange colour instead of the diagonal pattern (nodes q', m' and l here) and edges that participate in the paths to the leaf nodes are also marked with orange dashed lines. Finally, transparent orange dashed lines start from the virtual edges and end on an AND gate that virtually controls the switch, meaning that the switch only closes if all candidate matches are confirmed. All other graph elements that do not have any role in the semantic matching, are illustrated with thin grey lines.

The plant-sensor composition is not shown, however, it is very similar to the actuator-plant composition discussed earlier. Note that if the agent Σ cannot find a sensor that directly matches the location of the plant through the $G(\mathcal{L}, \mathcal{L}, E_1^{(\mathcal{L},\mathcal{L})})$ relation-graph, it may explore other options, e.g., find a sensor in another location associated with the plant's location through a path of the "is adjacent to" relation graph ($G(\mathcal{L}, \mathcal{L}, E_2^{(\mathcal{L},\mathcal{L})})$. Such sensor will be used depending on the specifications of the considered controllers, e.g., considering whether there is an open physical air-flow path between the two locations.

3) Controller matching: Once the agent Σ has found appropriate actuators and sensors, it is required to identify an appropriate controller. In this case, the matching is confirmed if the input-branch-leaf-node of the controller's semantic model is the same as (or is connected through a path of the knowledge model with) the output-branch-leaf-node of the sensor's semantic model *and* the location-branch-leaf-nodes of the models of the two components are either the same or they are connected through appropriate location-graphs. In the example of this work, the sensor output, as well as the controller's input represent signals in some measurement units $m \in \mathcal{M}$. Detailed presentation of this composition, as well as the controller-actuator composition are omitted, as they can be derived from understanding the previous models.

It has been mentioned that there are cases where the output signals of some components need to be passed through processing functions before being fed as inputs to other components. For instance, a physical property may be associated with several measurement units and there might be a function that converts one measurement unit to another measurement unit. This piece of expert knowledge can be included in the semantic database and be utilized as required by the agent Σ. For example, an appropriate function can be used to transform degrees Celsius (^oC) to degrees Fahrenheit (^oF) and vice versa when required for temperature measurements. Fig. 8 illustrates the intervention of such function to transform the measurement unit of the signal produced by the sensor (e.g., "Celsius" represented by "thing" m) before this is fed to the controller that expects a signal in "Fahrenheit" represented by the "thing" m''.

It can be seen that the semantic matching of the sensor with the controller only becomes feasible through the transformation function $f' = f_{CF}(C) = C * \frac{9}{5} + 32$ that transforms degrees Celsius to Fahrenheit. Note the orange-dashed-line marking of the edges that pass through the function and facilitate the matching. These edges are also shown as directed, to capture the flow of transformation from a unit to another and avoid non-meaningful use of the relations. The transformation of control signals before being fed to an actuator can be modelled in a very similar way and is not presented here.

C. Closed-loop Online Composition Decision Algorithm

Summarising the above, as soon as a change appears in the availability of components, an overall closed-loop online configuration algorithm is executed by the agent Σ (see Algorithm 1), so as to enable the shift of the control scheme's configuration from (I) to $(I + 1)$. Since this work is not addressing the selection of controllers based on performance criteria, it is assumed that the already available controllers are ranked offline according to pre-defined performance criteria. This assumption supports the selection and adoption of one controller (the one with higher ranking) in case more than one comprise valid (semantic) options for closing the loop.

It is noted that the discussed procedure for output-input semantic matching is executed as many times as required during the execution of the algorithm, to find matches between outputs and inputs of components. Also, the cases where more than one sensors measure (and/or, more than one actuators act on) the same physical property, are assumed pre-solved in this work, adopting existing sensor fusion or actuation signal fusion techniques. In future implementations, an effort will be made to automate the encoding of knowledge related to the performance ranking, as well as to the fusion functions, thus further automating the decision process. Moreover, it is emphasized that the method is currently considered applicable to systems and processes with "slow" (enough) dynamics that leave time for the agent Σ to complete the execution of the algorithm and connect the components appropriately, not compromising the control objectives.

The following section presents specific online composition cases, to enhance the understanding and illustrate the applicability of the method.

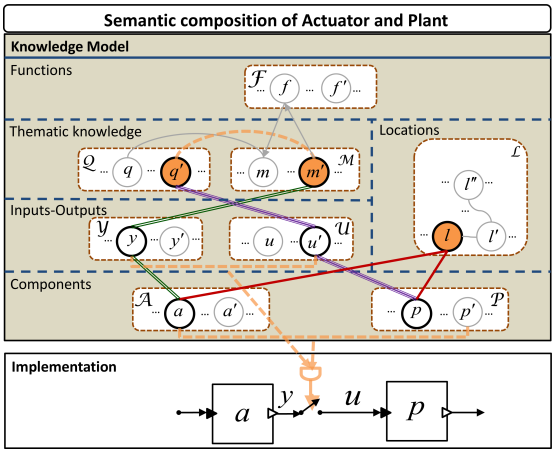


Figure 7. The actuator-plant composition model

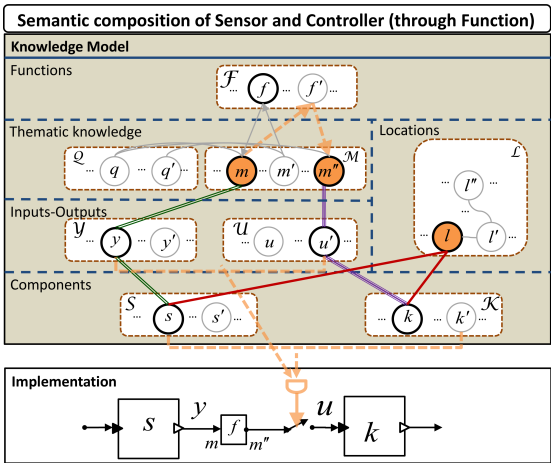


Figure 8. The sensor-function-controller composition model

Algorithm 1 Agent Σ Algorithm

procedure START EXECUTION
Find all actuators that are capable of acting on the plant's inputs
Find all sensors that are capable of measuring the plant's outputs.
for each available controller starting with the one of higher performance ranking **do**
Check if it can drive the actuators found in previous step.
Exploit also available transformation functions for the control signals.
if successful **then**,
Find whether the sensors identified earlier measure the controller's (mandatory) inputs.
Exploit also available transformation functions for the sensor's output signals.
if successful **then**,
Match is confirmed, therefore close the loop with the matching components and continue operation of the control system for the specific plant.
else
Allow option to drive fewer actuators than available and iterate with checking the next controller
end if
end if
end for
end procedure

V. EXAMPLE ONLINE COMPOSITION SCENARIOS

A feedback control scheme of the architecture in Fig. 1 has been deployed in the plant of the example given in Section III, with the aim to regulate the temperature at a desired value.

For the purposes of presentation clarity of the work, it is assumed that all utilized actuators receive inputs of the form "ON/OFF" or of the form of a range "$[u_{min}, u_{max}]$" and produce an action proportional to those values. In practice, each of these commands may be represented by different actual values, e.g., "$0/1$" meaning '1' for ON and '0' for OFF. All these will be represented by different "things" from the class of measurement units, $\mathcal{M}$ and there can be appropriate functions in the class of function "things", $\mathcal{F}$, that can transform one unit (control decision) to another. Moreover, the contribution of the work is focusing on the knowledge modelling of control system components and subsequent configuration inference and not on the specific implementation of the controllers. Therefore, for the sake of the example, it is assumed that agent Σ can select only among three types of controllers; a periodic open-loop controller, a threshold controller and a proportional controller, as presented below. In future work, we plan to present the applicability of the method with more advanced controllers, as well as online incorporation of criteria for selecting the appropriate controller to use.

The specifics of the three utilised controllers are discussed below:

i. *Periodic Open Loop Controller:* An open-loop controller which produces an "on/off" signal to an actuator based on internal cycle configuration. The controller operates with a default cycle, exchanging between an "off" time-period followed by an "on" time-period. It also offers a parameter-input through which the "on" and "off" periods can be configured. This controller is given by (1).

$$u_1(t) = \begin{cases} 1 & \text{if } nT \leq t \leq (n+a)T \\ 0 & \text{if } (n+a)T \leq t \leq (n+1)T \end{cases} \quad (1)$$

where T defines one operation cycle, i.e. the time between two consecutive turn-on events of the controller, $a \in (0,1)$ is the parameter that configures the "on" and "off" periods in the cycle, $u_1(t)$ is the control decision produced by controller k_1 and $n = 0, 1, 2, 3, ...$ Such controller might be utilised for the control of a ventilation fan, to clean the inside air at fixed cycles.

ii. *Threshold Controller:* A bang-bang controller which produces an "on" signal when the measured value of a physical property is less than its desired value and an "off" signal when the value is greater. This controller receives a measurement value of some physical property

as an input. The controller is given by (2), where a dead-zone is also considered to avoid reactions to measurement noise or other high-frequency oscillations.

$$u_2(y(t), r(t), t) = \begin{cases} 1 & \text{if } (r(t) - y(t)) > \epsilon \\ 0 & otherwise \end{cases} \quad (2)$$

where $u_2(.)$ is the control decision produced by controller k_2, $y(t) > 0$ is the measurement of the state assumed by the controller, $r(t) > 0$ is the reference/desired value of the state and ϵ is a small positive value that defines the size of the dead-zone. Such controller may be used for maintaining the temperature of a room close to a reference value.

iii. *Proportional Controller:* A controller that produces a signal proportional to the tracking error $y_c(t) - r(t)$ which defines the level of operation of an actuator. The controller receives as input the measured value of a physical property, while it is also given a reference value and, optionally, a vector of other parameters of the plant (e.g., opening of the window) as parameter-inputs, which help in calculating the gain for stability. By default, the parameters are considered constant and are given to the controller upon deployment. This controller is given by:

$$u_3(y_c(t), r(t), t) = \Gamma(\theta)(y_c(t) - r(t)), \quad (3)$$

where $u_3(.)$ is the control decision produced by the controller, $y_c(t)$ is the measurement of a plant's state, as assumed by the controller, $r(t)$ is the desired value of that state of the plant, Γ is the proportional gain function and θ is a vector of parameters which affect the gain calculation for stable control. Such controller may be used for the regulation of the temperature in a room at specific value.

Moreover, for the implementation of the semantic matching and the algorithm, the interface of agent Σ with the semantic database is implemented through the API of the Apache Jena project [38], while the queries performed in the semantic database, are written using the SPARQL protocol [39].

The following list summarises the knowledge objects currently introduced in the knowledge model:

- $p_1 \in \mathcal{P}$, representing the plant (i.e., the office and its temperature dynamics) subject to control
- $s_1 \in \mathcal{S}$, representing the temperature sensor deployed in the plant
- $a_1 \in \mathcal{A}$, representing the actuator (heating device) deployed in the plant
- $k_1, k_2, k_3 \in \mathcal{K}$, representing the controllers introduced earlier
- $m_1, m_2, m_3, m_4 \in \mathcal{M}$, representing the "Celsius" units produced by the sensor, the kW units produced by the actuator, as well as the "on/off" and the "$[0, 100]$" signals respectively, produced by controllers
- $q_1, q_2 \in \mathcal{Q}$, representing the "temperature" physical property and the "Heat Energy" property affected by the actuator, respectively
- The corresponding semantic models of the plant, sensors, actuators and controllers, as discussed in Section IV

Given the currently available components and semantic database and considering the plant p_1 that is subject to control, Algorithm 1 runs as follows:

Step 1: The first step focuses on the composition of the available actuator a_1 and the plant p_1. The plant has an input u_1 representing the introduction of heat energy in the office, while the actuator has output y_1 representing the produced heat energy. The respective composition is illustrated as part of Fig. 9. Note that the colouring convention has been adjusted here to facilitate the illustration of the semantic matching for a complete closed-loop. The output-branch of a_1, the input-branch of p_1, as well as the path between the respective leaf-nodes are marked with thick gold line. The location-branch is marked with a red line and it can be seen that the respective leaf-nodes of the components coincide. The actuator is therefore confirmed as acting on the plant input since it is located in the appropriate location and produces a signal that affects the required physical property.

Step 2: In a similar way, Fig. 9 shows the composition of plant p_1 and sensor s_1, where the respective output-branch and input-branch are marked with double blue line.

Step 3: Exploiting whether actuator a_1, confirmed in previous step, can be driven by the controller k_3 given by (3), which has the highest performance ranking. This controller produces a signal in the range $[0, 100]$, represented by measurement unit m_4) It is also assumed that the gain of the controller is calculated based on prior knowledge of the plant parameters at deployment, which implies that the controller is associated with the location $l_1 =$ office. On the other hand, the actuator is assumed receiving an on/off signal as input and this knowledge is modelled by measurement unit m_3. The composition can be examined in Fig. 9. Following the light grey line from the controller k_3, it can be seen that its output-branch-leaf-node does not coincide with the input-branch-leaf-node of the actuator, which means that the components cannot be used together in the closed-loop scheme. The algorithm suggests checking for the next controller in the rank, that is, k_2. This controller does produce an on/off signal, which makes the matching confirmed as shown by the branches marked with dashed purple line.

Step 4: Exploiting whether sensor s_1 confirmed earlier, measures the input of controller k_2. Apart from the measured input, the controller requires a reference value. It is assumed that the desired room temperature is read from a pre-defined source, in degrees Celsius. Therefore, the input u_4 of the controller inherits the location $l_1 =$ office from the plant and the measurement unit $m_1 =$ Celsius from the reference value. On the other hand, the sensor is annotated as measuring in m_1 and its location is already given as l_1. The leaf-nodes of the output-branch of the sensor (for its output y_4) and the input-branch of the controller (for its input u_4), as well as the leaf-node of the location-branches of both components, coincide. This leads to the conclusion that sensor s_1 does measure the input of controller k_2. This composition is also illustrated in the same figure, with the output and input branches marked with triple green line.

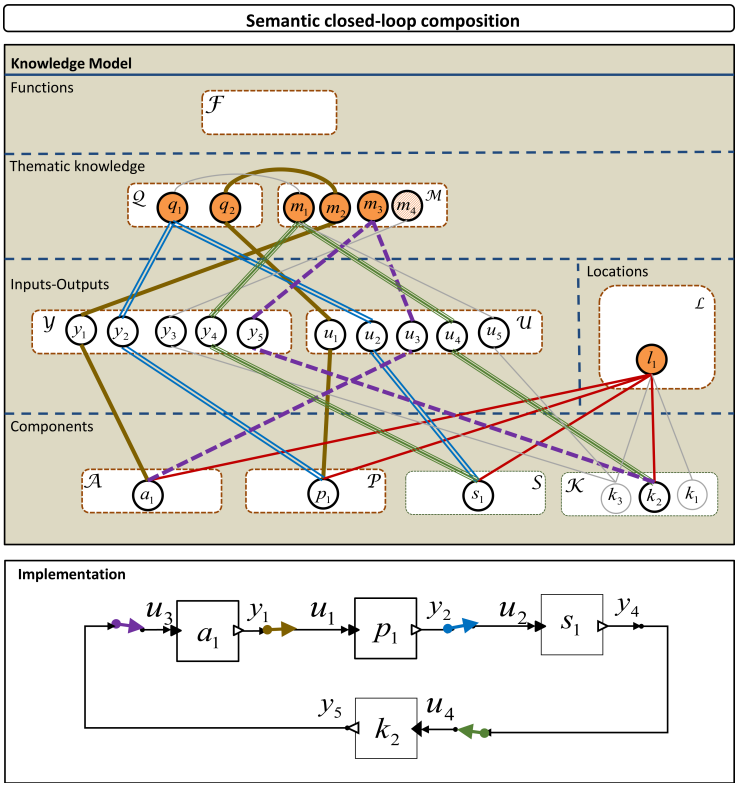


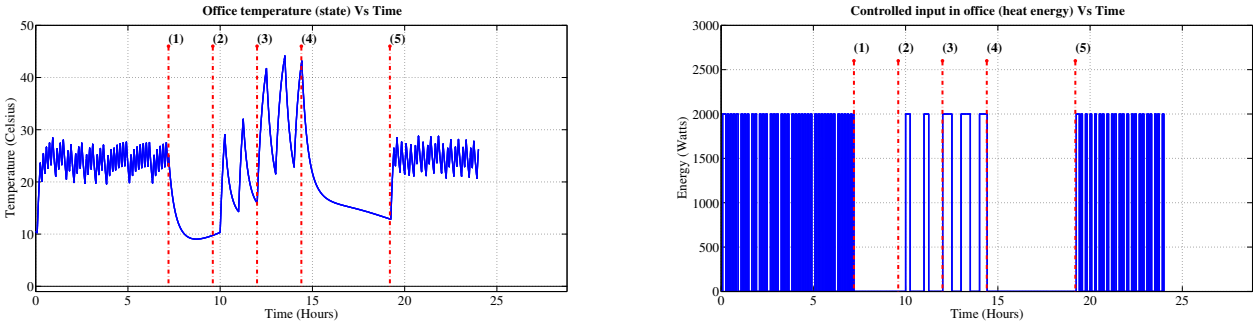
Figure 9. The composition of a closed loop with plant p_1, actuator a_1, sensor s_1 and controller k_2

A. *Replacing and adding sensors and processing functions*

At some time-point, the sensor s_1 stops transmitting. In order to help emphasise on the contribution of the agent Σ, we assume that it is initially non active when the change happens. The result of this event is captured in Fig. 10, marked with "(1)" and a red dashed line. The figure illustrates a 24-hour simulation of the plant's operation. Sub-fig. 10a, shows the temperature of the office in degrees Celsius, versus the simulation time, while Sub-fig. 10b shows the corresponding control input in $Watts$. The current situation is such that there is no measurement feedback and we assume that the controller is configured so as to stop operating. As a consequence, and since there is no active semantic mediation, the office temperature is left to drop (note: the ambient temperature is modelled as varying between 6 and 15 degrees Celsius in a 24-hour cycle). Then, at time-point marked with "(2)" we assume that the agent Σ becomes active again, thus executing the algorithm to re-configure the feedback control scheme. Reviewing the execution of the algorithm, it is derived that k_2 is ruled out of options since its input is no longer measured. The algorithm iterates and selects the controller k_1, which can still drive a_1, while not expecting any inputs since it operates in open loop (we assume a default cycle configuration with $a = 0.4$, defining a 40% "on" period). This configuration is the $I = 1$. We can see that the control of the plant resumes with the use of the open-loop controller k_1 and the temperature is regulated accordingly, even with degraded performance given the capabilities of the employed controller.

While k_1 operates on a default cycle, a human user updates its semantic model, specifying that its parameter-input a represents the property "thermal comfort" in the location "ambient", defined as a percentage value in the range (0,1). A software service is also available and semantically annotated as reading the local "ambient" temperature from a weather service on the Internet and transforming it to a "thermal comfort" signal, mapping the thermal comfort range $(25-10^oC)$ to the range (0,1) (ambient temperatures below 10^oC are mapped to the lower end of the comfort range). These annotations enable the agent Σ to match the output of the software service to the parameter-input a of k_1, which subsequently adjusts its operation cycle. The change happens at time-point marked with "(3)" and shows that the controller k_1 increases the "on" period responding to the very low ambient temperature.

At a future event, a new temperature sensor s_2 is deployed to replace the faulty one, which however measures in degrees Fahrenheit (modelled by $m_5 \in \mathcal{M}$). With the opportunity, the heating device is also replaced by another one ($a_2 \in \mathcal{A}$), which accepts as input a step signal in the range "$[0, u_{max}]$", where u_{max} is the maximum power at which the device can operate. In the meantime, two functions become available as well: i) Function $f_{FC}(F) = \frac{5}{9}(F-32)$, which transforms degrees Fahrenheit (F) to degrees Celsius; ii) Function f_{KW} which scales the $[0, 100]$ control signal to the steps and range of the power signal accepted by the actuator a_2. The semantic database is subsequently updated with the "things" $f_{CF}, f_{KW} \in \mathcal{F}$ and their relations to the domain and range measurement units (here m_5 and m_4 to m_1 and m_2 respectively). Note: these "things"/functions can be also retrieved online from dedicated cyber sources.



(a) (b)

Figure 10. A 24-hour simulation of the plant's control operation. Event (1): sensor s_1 stops transmitting while semantic mediation is off. Event (2): Semantic mediation is back on. Event (3): parameter a of controller k_1 is annotated and subsequently fed by a measurement of ambient temperature. Event (4): Sensor s_2 is deployed, measuring the temperature in degrees Fahrenheit. Event (5): an additional function is imported, transforming the measurement units

The system detects the changes in the components. Given the current situation, the composition of actuators and the plant finds a_2 capable of producing the signal for plant input u_1, since the output characteristics of the new actuator have not changed. Moreover, the composition of plant and sensors remains unchanged as well, since s_2 measures $q_1 =$ temperature, which is the same physical property. The algorithm continues by checking the composition of the controller k_3 (that is ranked higher in terms of performance) and the available actuator a_2. Fig. 11 illustrates all resulting compositions. It can be seen that through the function f_{KW} a path is formed from m_4, the output-branch-leaf-node of k_3, to m_2, the input-branch-leaf-node of a_2. The components have the same location-branch-leaf-nodes. This leads to the conclusion that the controller k_3 is able to drive the actuator a_2. Next, the algorithm checks whether sensor s_2 measures the input of controller k_3. Figure shows that there is a path created through the function f_{FC}, from $m_5 =$ Fahrenheit, the output-branch-leaf-node of s_2, to $m_1 =$ Celsius, the input-branch-leaf-node of k_3. Locations are again the same, therefore, the composition is confirmed. This leads to the completion of the algorithm and the controller k_3 being adopted to close the loop in a new scheme configuration $I = 2$.

The simulation results are shown again in Fig. 10. Event marked with "(4)", captures the deployment of the new sensor, initially also assuming no availability of the measurement units' transformation function. We choose to show this result in order to emphasise on the ability to incorporate new functionality online. It can be seen that the problematic interpretation of the temperature measurements in the time period between events "(4)" and "(5)", causes the controller to operate wrong. Finally, event "(5)" marks the incorporation of the unit transformation function, which causes the control system to return back to normal operation.

Although the above examples of retrieving temperature measurements from Internet sources, transforming Fahrenheit to Celsius degrees or adjusting between signal ranges are simple for presentation purposes, the same mechanism can be used for more advance transformation paths, e.g., given the occupancy of a room, derive implicit knowledge about the increase in temperature and/or Carbon dioxide concentration. It is emphasized that with the proposed architecture, such knowledge is not necessarily available in advance, but it can be incorporated in the control scheme online through the update of the semantic database. The following sub-sections offer insight to additional cases to show the strengths of the approach, however, details about the compositions are not given, due to high complexity of the graphs and space limitations.

B. Exploring new plant knowledge

Later in time, the knowledge about the plant is enriched, adding another location $l_2 =$ window. A third sensor s_3 is also deployed that measures the physical property "opening" of the window (defined as q_3) in the measurement unit "percentage", represented by the "thing" m_6. The plant semantic model is also enriched with a new output-branch that represents the measurable window opening. The execution of the algorithm will lead to configuration $I = 3$, where sensor s_3 will be confirmed as measuring the window opening, while controller k_3 may be adjusted to consume that measurement through an optional input for parameters, to update its gain using the new plant knowledge.

C. Exploring relations between locations

Later, a fourth sensor s_4 is deployed outside of the office (represented by location l_3), measuring the ambient temperature in degrees Celsius. The knowledge model is also adjusted to be able to update the adjacency relations between locations to define that l_3 is "adjacent to" l_1 whenever the opening of the window (q_3) is measured more than 80%. The reasoning behind this, is that the office temperature becomes almost equal to the ambient temperature if the heat transfer coefficient of the opening is big enough.

Executing the algorithm, sensor s_4 can be confirmed by Σ, through the locations' relation, as measuring the office temperature. In this case, the values produced by sensors s_2 and s_4 can be properly fused (e.g., applying a weighted average considering the accuracy of devices given by manufacturers) and be fed into the controller k_3 in a configuration $I = 4$. Alternatively, in case of big openings and subsequently

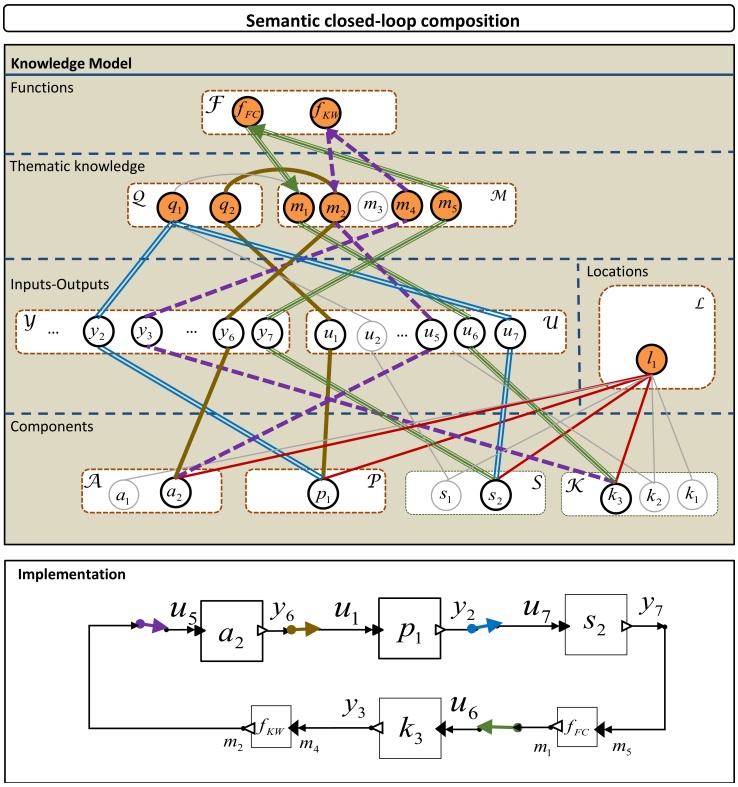


Figure 11. The composition of a closed-loop with plant p_1, actuator a_2, sensor s_2 and controller k_3, adopting also available signal transformation functions

potentially big heat energy losses, Σ may also be given the intelligence to shut-down the controllers to avoid high energy cost (it is left outside the scope of this work).

D. Discussion of results

It can be seen from the presented illustrative scenarios and simulations that the designed system is able to respond to changes in the measurement and/or actuation capability and switch to a different (semantically valid) closed-loop configuration. Further to the positioning of this work, addressed in Section I, the key advantages and contributions are summarised as: i) Unlike typical switching control systems, there is no pre-requisite to know in advance the possible components and configurations or their capabilities; These can be semantically described as they become available; ii) The switching decision is made online, by the agent Σ, which is equipped with a logic-based system that exploits expert knowledge and inference rules and helps producing an explicit decision signal; The expert knowledge comprises knowledge about the application domain, standards-compatible modelling of components, and closed-loop system configuration rules; Current related work in literature, either does not address the re-configuration challenge at the closed-loop level or it does not provide an explicit mechanism for making the switching decisions when the configurations are not known in advance. For instance, existing solutions would neither have been able to feed a controller with the signal of a new sensor, online, nor to incorporate a transformation function online.

Beyond the advantages, there are some key constraints that limit the applicability of our solution: i) although in theory the method works with wired components as well, its applicability is more straightforward with wireless IoT-enabled components; ii) new components deployed in the system must first be semantically described using the pre-defined models, otherwise they cannot be used effectively; iii) the logic-based switching decision-making is computationally intensive, therefore, it may not be applicable to systems with fast dynamics; iv) the system can only use types of components already considered in the modelling, i.e., current implementation cannot use a state-estimator; v) the supported measurement and actuation capability is limited by the modelled domain knowledge, i.e., the system cannot use an occupancy sensor if the property "occupancy" is not already described in the model.

VI. Concluding Remarks and Future Plans

This article presented a new method and architecture, that can be adopted in the design of feedback control schemes within large-scale systems, in order to take advantage of the online re-configurability characteristics offered by combining cyber-physical control systems with semantic composition techniques.

The scope of the work was neither to advance the control algorithms as such nor to extent the theory behind semantic composition of services. It was rather to develop a mechanism that will allow the application of these techniques within the configuration of switching closed-loop schemes. The applicability of the proposed architecture and methodology has been tested with an illustrative scenario from the smart buildings domain. It has been shown that the methodology allows the

transparent plugging-in of control system components without the need to shut-down and re-design the closed-loop.

The results presented here, are considered as the first milestone of our work, since they comprise a complete first-draft implementation of the framework. Next steps will focus on: i) semantic modelling of more components, e.g., observers, online learning functions; ii) investigation of closed-loop stability during switching (the current work assumed only systems the states of which remain bounded during switching); iii) methods for optimal selection of closed-loop configuration, given a set of returned semantically valid options; iv) online knowledge acquisition, combining semantic modelling with data driven techniques (in current work the system is pre-populated with expert knowledge, while it can also incorporate new expert knowledge online, i.e., in cases of new components installed in the large-scale system, but it cannot generate knowledge on its own; v) the implementation of a demonstration setup that will pilot-test the applicability in real-life scenarios.

REFERENCES

[1] M. Webb, "SMART 2020: Enabling the low carbon economy in the information age," The Climate Group, London, Tech. Rep., 2008.

[2] D. Snoonian. (2003) IEEE Spectrum - Smart Buildings. Accessed: 2016-10-21. [Online]. Available: http://spectrum.ieee.org/green-tech/buildings/smart-buildings

[3] T. Weng and Y. Agarwal, "From Buildings to Smart Buildings - Sensing and Actuation to Improve Energy Efficiency," *IEEE Design Test of Computers*, vol. 29, no. 4, pp. 36–44, Aug 2012.

[4] IEEE SmartGrid. Accessed: 2016-10-21. [Online]. Available: http://smartgrid.ieee.org/

[5] 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.

[6] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki, *Diagnosis and fault-tolerant control*. Springer Berlin Heidelberg, 2006.

[7] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Prentice Hall, 1994.

[8] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[9] J. Farrell and M. Polycarpou, *Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*, N. J. W. Hoboken, Ed. J. Wiley, 2006.

[10] H. Li, L. Bai, L. Wang, Q. Zhou, and H. Wang, "Adaptive neural control of uncertain nonstrict-feedback stochastic nonlinear systems with output constraint and unknown dead zone," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. PP, no. 99, pp. 1–12, 2016.

[11] Y. Li and S. Tong, "Adaptive Fuzzy Output-Feedback Control of Pure-Feedback Uncertain Nonlinear Systems With Unknown Dead Zone," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 5, pp. 1341–1347, Oct 2014.

[12] H. Li, S. Yin, Y. Pan, and H.-K. Lam, "Model reduction for interval type-2 Takagi–Sugeno fuzzy systems," *Automatica*, vol. 61, pp. 308–314, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109815003465

[13] T. Knudsen, "Awareness and its use in Plug and Play Process Control," in *2009 European Control Conf. (ECC)*, Aug 2009, pp. 4078–4083.

[14] 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

[15] 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.

[16] 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.

[17] 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.

[18] H. Li, P. Shi, D. Yao, and L. Wu, "Observer-based adaptive sliding mode control for nonlinear Markovian jump systems ," *Automatica*, vol. 64, pp. 133–142, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0005109815004719

[19] J. Lygeros, C. Tomlin, and S. Sastry, *Hybrid Systems: Modeling, Analysis and Control*, 2008.

[20] K. Tsuda, D. Mignone, G. Ferrari-Trecate, and M. Morari, "Reconfiguration strategies for hybrid systems," in *Proceedings of the 2001 American Control Conf. (Cat. No.01CH37148)*, vol. 2, 2001, pp. 868–873 vol.2.

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

[22] J. Huang, F. Bastani, I. L. Yen, J. Dong, W. Zhang, F. J. Wang, and H. J. Hsu, "Extending service model to build an effective service composition framework for cyber-physical systems," in *IEEE Int. Conf. on Service-Oriented Computing and Applications, SOCA' 09*, 2009.

[23] P. Glotfelter, T. Eichelberger, and P. J. Martin, "PhysiCloud : A Cloud-Computing Framework for Programming Cyber-Physical Systems," in *2014 IEEE Conference on Control Applications (CCA)*, 2014.

[24] A. Krioukov, G. Fierro, N. Kitaev, and D. Culler, "Building application stack (BAS)," in *Proc. of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings - BuildSys '12*. New York, USA: ACM Press, 2012, p. 72. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2422531.2422546

[25] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler, "BOSS: Building Operating System Services," in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013. [Online]. Available: http://www.cs.berkeley.edu/~stevedh/pubs/nsdi13boss1.pdf

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

[27] Y. Lirov, E. Y. Rodin, B. G. McElhaney, and L. W. Wilbur, "Artificial Intelligence Modelling of Control Systems," *The Society for Modeling and Simulation International*, vol. 50, no. 1, pp. 12–24, Jan. 1988. [Online]. Available: http://dx.doi.org/10.1177/003754978805000103

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

[29] 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

[30] A. Pablo, R. Valiente, and A. Lozano-Tello, "Ontology and SWRL-Based Learning Model for Home Automation Controlling," in *Ambient Intelligence and Future Trends-International 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.

[31] S. Staab and R. Studer, Eds., *Handbook on Ontologies*. Springer-Verlag Berlin Heidelberg, 2009.

[32] G. M. Milis, C. G. Panayiotou, and M. M. Polycarpou, "Towards a Semantically Enhanced Control Architecture," in *IEEE Multi-Conference on Systems and Control*, Dubrovnik, Croatia, 2012.

[33] A. Gómez-Pérez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. First Edition*, 1st ed. London: Springer-Verlag London, 2004.

[34] 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. (2008) OWL-S: Semantic Markup for Web Services. Accessed: 2016-10-21. [Online]. Available: http://www.ai.sri.com/daml/services/owl-s/1.2/overview/

[35] J. Davis and M. S. Rajasree, "RESTDoc: Describe, Discover and Compose RESTful Semantic Web Services using Annotated Documentations," *Int. journal of Web & Semantic Technol.*, vol. 4, no. 1, pp. 37–49, Jan. 2013. [Online]. Available: http://www.airccse.org/journal/ijwest/papers/4113ijwest03.pdf

[36] A. Sheth, C. Henson, and S. S. Sahoo, "Semantic Sensor Web," *IEEE Internet Comput.*, vol. 12, no. 4, pp. 78–83, Jul 2008. [Online]. Available: http://knoesis.org/library/publications/SHS08-ICColumn-SSW.pdf

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

[38] The Apache Jena project. Accessed: 2016-10-21. [Online]. Available: http://jena.apache.org/

[39] W3C. (2004) Semantic Web Query Standards. Accessed: 2016-10-21. [Online]. Available: http://www.w3.org/standards/semanticweb/query

George M. Milis (S'08) was born in Cyprus in 1977. He received an MSc in Advanced Computing with specialisation in Artificial Intelligence from the Department of Computing, Imperial College London, and a Diploma in Electrical and Computing Engineering from the Aristotle University of Thessaloniki, Greece.

Mr Milis has been employed as a researcher in Intelligent Systems and Software Engineering Lab in Thessaloniki and then as a Senior ICT Consultant with European Dynamics SA in Greece. Currently he is a PhD candidate at the Department of Electrical and Computer Engineering of the University of Cyprus and is working as a Researcher at the KIOS Research Center for Intelligent Systems and Networks. His research interests include the study of semantically-enhanced architectures for the online composition of feedback control and fault detection systems, as well as interoperability aspects of Critical Infrastructure Systems, and Systems of Systems in general. He has extensive experience in knowledge extraction and management, applications of intelligent control in security and social welfare, as well as, semantically-enhanced interoperable services in e-Government, Education and beyond. He also has a wide acquaintance with software project management, analysis and consulting for efficient adoption of technology in collaborative environments.

Christos G. Panayiotou (SM'06) is an Associate Professor with the Electrical and Computer Engineering (ECE) Department at the University of Cyprus (UCY). He is also the Deputy Director of the KIOS Research Center for Intelligent Systems and Networks for which he is also a founding member. Christos has received a B.Sc. and a Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts at Amherst, in 1994 and 1999 respectively. He also received an MBA from the Isenberg School of Management, at the aforementioned university in 1999. Before joining UCY in 2002, he was a Research Associate at the Center for Information and System Engineering (CISE) and the Manufacturing Engineering Department at Boston University (1999 - 2002). His research interests include distributed and intelligent control systems, wireless, ad-hoc and sensor networks, computer communication networks, fault diagnosis, optimization and control of discrete-event systems, resource allocation, transportation networks and intelligent buildings.

Prof. Panayiotou has published more than 190 papers in international refereed journals and conferences and is the recipient of the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He is an Associate Editor for the Conference Editorial Board of the IEEE Control Systems Society, the IEEE Transactions on Control Systems Technology, the Journal of Discrete Event Dynamical Systems and the European Journal of Control. He held several positions in organizing committees and technical program committees of numerous international conferences.

Marios M. Polycarpou (F'06) is a Professor of Electrical and Computer Engineering and the Director of the KIOS Research Center for Intelligent Systems and Networks at the University of Cyprus. He received the B.A degree in Computer Science and the B.Sc. in Electrical Engineering, both from Rice University, USA in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Southern California, in 1989 and 1992 respectively. His teaching and research interests are in intelligent systems and networks, adaptive and cooperative control systems, computational intelligence, fault diagnosis and distributed agents. Dr. Polycarpou has published more than 300 articles in refereed journals, edited books and refereed conference proceedings, and co-authored 7 books. He is also the holder of 6 patents.

Prof. Polycarpou is a Fellow of IEEE and IFAC. He is the recipient of the 2016 IEEE Neural Networks Pioneer Award and the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He has served as the President of the IEEE Computational Intelligence Society (2012-2013), and as the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems (2004-2010). He is currently the Vice President of the European Control Association (EUCA). Prof. Polycarpou has participated in more than 60 research projects/grants, funded by several agencies and industry in Europe and the United States, including the prestigious European Research Council (ERC) Advanced Grant.