HAL open science

Exploring Semantic Mediation Techniques in Feedback Control Architectures

Georgios M. Milis, Christos G. Panayiotou, Marios M. Polycarpou

To cite this version:

Georgios M. Milis, Christos G. Panayiotou, Marios M. Polycarpou. Exploring Semantic Mediation Techniques in Feedback Control Architectures. 9th Artificial Intelligence Applications and Innovations (AIAI), Sep 2013, Paphos, Greece. pp.657-666, 10.1007/978-3-642-41142-7_66 . hal-01459658

HAL Id: hal-01459658

https://inria.hal.science/hal-01459658v1

Submitted on 7 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Exploring semantic mediation techniques in feedback control architectures

Georgios M. Milis, Christos G. Panayiotou, and Marios M. Polycarpou

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

Abstract. *Modern control systems implementations, especially in large–scale systems, assume the interoperation of different types of sensors, actuators, controllers and software algorithms, being physical or cyber. In most cases, the scalability and interoperability of the control system are compromised by its design, which is based on a fixed configuration of specific components with certain knowledge of their specific characteristics. This work presents an innovative feedback control architecture framework, in which classical and modern feedback control techniques can be combined with domain knowledge (thematic, location and time) in order to enable the online plugging of components in a feedback control system and the subsequent reconfiguration and adaptation of the system.*

Keywords: Control system architecture, interoperability, scalability, semantic knowledge models, plug&play of components

1 Introduction

Nowadays, systems are designed and built not as monolithic entities but as collection of smaller physical and cyber components that, many times, can be considered as separate systems themselves with their own dynamics and objectives. This *system of systems* paradigm ([15]) necessitates the easy interaction and interoperability of the components that comprise a larger system. Components are expected to take informed decisions and act intelligently towards meeting (or balancing) the system's objectives. This description is valid also for modern control systems, where different types of components, being physical or cyber, interoperate in a larger control system implementation. However, in most cases, the design of feedback control systems is based on a fixed configuration of specific components, with certain knowledge of their specific characteristics. This causes lack of scalability and interoperability for the control system, thus considerably limiting its potential lifetime. There are cases where faulty sensors need to be replaced or additional sensors need to be installed (e.g. due to recent availability of this type of components or due to upgrading to new technology), and this should not require redesign of the overall feedback control system since such action would be impractical and costly.

The detection and identification of non-modelled events in linear and non-linear systems is currently addressed by the *fault diagnosis* research area. The authors in [8] and [5] provide a thorough overview on the classical algorithms that identify deviations from the normal behaviour of a system, attributed to faults or other external events. The *adaptive* and *fault-tolerant control* research areas address the design of intelligent control type algorithms, that aim to facilitate the flexibility of the control system with respect to on-line adaptation, and accommodation of faults, system uncertainties and/or time variations. Approaches to designing fault tolerant and reconfigurable control systems are presented in [4]. In [10] the author also addresses the issue of fault-tolerant components, while the authors in [7] provide methodologies for designing adaptive approximation-based control systems. Recent efforts in plug&play control ([17] and more recently in [3]), propose methodologies for the online identification of newly introduced dynamics when new components are plugged in a closed-loop system and the subsequent online adaptation of the feedback laws. Also, the authors of the present paper have recently presented initial results of their work [14] on the exploitation of ontology-based semantic mediation techniques in feedback control systems.

The main contribution of our work is the design of an innovative feedback control architecture framework, in which classical and modern feedback control techniques can be combined with domain knowledge (thematic, location and time) in order to enable the online plugging of components in feedback control systems and their subsequent reconfiguration and adaptation. The control system becomes able to make use of and enrich thematic, location and time related structured knowledge about the environment in which it operates.

The rest of the paper is organised as follows: Section 2 formulates the problem, to facilitate the presentation of the solution. Then, section 3 presents the proposed architecture and framework, followed by section 4 where a case-study scenario is given. Finally, section 5 shows a simulation with results and section 6 concludes the report.

2 Problem Formulation

Consider a closed-loop system with sensors measuring plant outputs, actuators acting on controlled inputs following instruction by a control law that considers an error trajectory. The actual plant states are estimated by an observer (e.g. a Luenberger observer [12]), to compensate for the case when some of them are missing, or for redundancy and noise cancellation.

Consider the following cases:

1 A deployed sensor fails and is replaced by a new one having different (and not compliant with the closed-loop system implementation) characteristics.
2 Sensor(s) enter the plant, at different locations and at different times. These sensors measure physical quantities that are already considered as states in the closed-loop system design.

3 Sensor(s) enter the plant as above, but this time some or all of them measure quantities not already taken into consideration for the initial design of the closed-loop system.

When changes happen in the components' synthesis of the closed-loop system, as explained in Section 2, the altered measurement vectors carry new sensing capabilities that can be potentially exploited using different models of the same plant. Therefore, at discrete time steps, the closed-loop system may assume a different model, with different types and/or dimensions of variables and parameters respectively. Without loss of generality, we assume this system is described by the state-space model in (1). The top-pointer $I = 1, 2, ...,$ is utilised to distinguish among different models.

$$\begin{aligned} \dot{x}^{(I)} &= A^{(I)}x^{(I)} + B^{(I)}u_a^{(I)} + G^{(I)}d^{(I)} \\ y_a^{(I)} &= C^{(I)}x^{(I)} + D^{(I)}u_a^{(I)} + H^{(I)}d^{(I)} + \nu^{(I)} \end{aligned} \tag{1}$$

where (avoiding the pointer I for simplicity): $x \in \mathcal{R}^n$ is the vector of system states, $u_a \in \mathcal{R}^m$ is the vector of controlled inputs, $d \in \mathcal{R}^q$ is the vector of uncontrolled inputs, $y_a \in \mathcal{R}^p$ is the vector of outputs (measurements), $\nu \in \mathcal{R}^p$ is the vector of measurement noise and A, B, G, C, D, H are the parameter matrices of proper dimensions and content.

The output part in (1), can be written as follows. Note that the signal is split into two parts to facilitate the analysis. Moreover, an extra top-pointer is used to indicate the signals that are changing between cases.

$$y_a^{(0)} = \begin{bmatrix} y_{a1}^{(0)} \\ y_{a2}^{(0)} \end{bmatrix} = \begin{bmatrix} C_1^{(0)} \\ C_2^{(0)} \end{bmatrix} x^{(0)} + \begin{bmatrix} D_1^{(0)} \\ D_2^{(0)} \end{bmatrix} u_a^{(0)} + \begin{bmatrix} H_1^{(0)} \\ H_2^{(0)} \end{bmatrix} g^{(0)} + \begin{bmatrix} \nu_1^{(0)} \\ \nu_2^{(0)} \end{bmatrix} \tag{2}$$

Then, the three cases identified above, lead to the equations:

$$y_a^{(1)} = \begin{bmatrix} y_{a1}^{(0)} \\ y_{a2}^{(1)} \end{bmatrix} = \begin{bmatrix} C_1^{(0)} \\ C_2^{(1)} \end{bmatrix} x^{(0)} + \begin{bmatrix} D_1^{(0)} \\ D_2^{(1)} \end{bmatrix} u_a^{(0)} + \begin{bmatrix} H_1^{(0)} \\ H_2^{(1)} \end{bmatrix} g^{(0)} + \begin{bmatrix} \nu_1^{(0)} \\ \nu_2^{(1)} \end{bmatrix} \tag{3}$$

$$y_a^{(2)} = \begin{bmatrix} y_{a1}^{(0)} \\ y_{a2}^{(0)} \\ y_{a2}^{(0)} \end{bmatrix} = \begin{bmatrix} C_1^{(0)} \\ C_2^{(0)} \\ C_2^{(0)} \end{bmatrix} x^{(0)} + \begin{bmatrix} D_1^{(0)} \\ D_2^{(0)} \\ D_2^{(0)} \end{bmatrix} u_a^{(0)} + \begin{bmatrix} H_1^{(0)} \\ H_2^{(0)} \\ H_2^{(0)} \end{bmatrix} g^{(0)} + \begin{bmatrix} \nu_1^{(0)} \\ \nu_2^{(0)} \\ \nu_2^{(0)} \end{bmatrix} \tag{4}$$

$$y_a^{(3)} = \begin{bmatrix} y_{a1}^{(0)} \\ y_{a2}^{(0)} \\ y_{a2}^{(3)} \end{bmatrix} = \begin{bmatrix} C_1^{(0)}x^{(0)} + D_1^{(0)}u_a^{(0)} + H_1^{(0)}g^{(0)} + \nu_1^{(0)} \\ C_2^{(0)}x^{(0)} + D_2^{(0)}u_a^{(0)} + H_2^{(0)}g^{(0)} + \nu_2^{(0)} \\ C_2^{(3)}x^{(3)} + D_2^{(3)}u_a^{(3)} + H_2^{(3)}g^{(3)} + \nu^{(3)} \end{bmatrix} \tag{5}$$

Equation 3 shows that a part of the sensing signals have been modified, comparing to specifications, resulting in a modified output vector. Equation 4

shows that the output vector has been modified not only in terms of content but also in terms of dimension, while still measuring same quantities. Finally, (5) shows that the output vector has been modified in terms of dimension and the newly introduced part measures different quantities. All described cases need to be properly accommodated in the closed-loop system by utilising available new knowledge and tools.

3 Proposed Architecture and Framework

In an earlier work, [14], the authors presented a basic introduction of the semantic interoperability concepts and the ontologies as a tool to implement knowledge models. Such models have been also used in domestic robotics (DOGont, [6]) to face the interoperation issues by implementing structured representations of domain knowledge. In this work, we adopt knowledge models in combination with control engineering mathematical representations. Efforts to represent the mathematical models in ontological knowledge models can be found in [18] and [11].

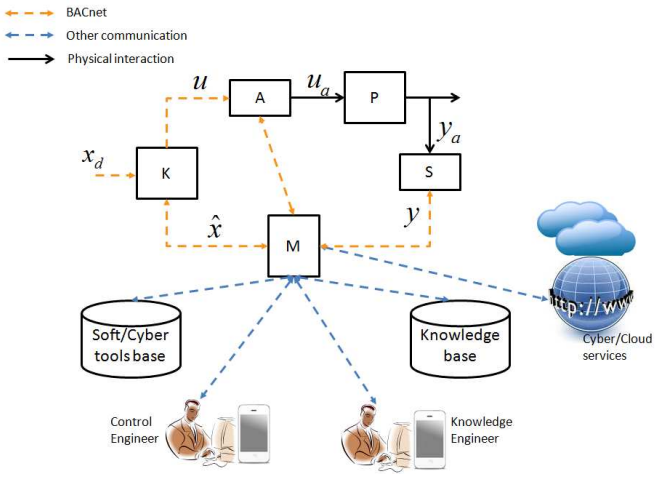


Fig. 1. Block diagram of proposed feedback–control architecture. Details about the content of this figure are given in section 3

The proposed architecture for the closed-loop system is depicted in fig. 1. As illustrated, the set-up comprises the: i) plant with its parameters and self-dynamics, ii) the physical control system components, like sensors (producing y) and actuators of different types (producing u_a), iii) a tools' base, which stores the implementations of software functions such as observer design implementations producing the state estimation $\hat{x}$, functions performing transformations among measurement units, etc., iv) the humans (e.g. Control Systems Engineer,

Knowledge Models Engineer), v) a communication infrastructure (the orange-dashed line shows the BACnet/IP protocol stack communication [2], whereas the blue-dashed line shows communication through any other protocol), vi) a semantic mediation module, M, which is responsible for the scalability of the control system and vii) a Knowledge Model, implemented as OWL ontology(ies) [1].

A critical component introduced here is M, which has a multi-fold scope as it implements the physical interaction interface among all components. The semantic mediation module strongly relies on the knowledge model to analyse each time's situation and take reasonable and optimal decisions for the operation of the system. It is therefore, of utmost important for the knowledge model to be well designed and defined based on the "closed-world" assumption [13]. We want the knowledge model to support control systems, that might comprise also safety-critical deployments, so the decisions taken should be based on explicit knowledge such as to avoid instability.

3.1 The knowledge model

The knowledge model comprises the agreement between all interacting physical and cyber components, about the interpretation of their environment.

This model is implemented as a set of objects' symbols, a set of classes/types for these objects and a set of properties of objects that also implement relations/mappings among them, that is, $\mathcal{A} = \{\mathcal{T}_H, \mathcal{C}_L, \mathcal{P}_R\}$. For the purpose of this work, we define specific objects, types of objects and properties. In order to keep it simple, we developed our own mini knowledge model. In future practical implementations, this model can be replaced by more complete efforts from the literature, such as combinations of the knowledge models in [9] to describe the environment and interactions of components, and the ones in [18] and [11] to describe the knowledge in mathematical representations.

The set of objects is defined as: $\mathcal{T}_H = \{o_i \mid i = 1, 2, ..., \}$, where o_i is the reference to an object's literal (e.g. the physical property "temperature") or to the real implementation of the object (e.g. "Sensor1" meaning the device with that identification).

The following classes of objects have been defined:

$\mathcal{C}_L = \{Plant, Model, State, ControlledInput, UncontrolledInput, Output, PlantLocation, PhysicalProperty, MeasurementUnit, Sensor, Actuator, Function\}$ where: $Plant$ is the set of plants served by the knowledge model, $Model$ is the set of system models (e.g. a state-space model of the system), $State$ is the set of system states, $ControlledInput$ is the set of controlled inputs, $UncontrolledInput$ is the set of uncontrolled inputs (disturbances to the plant), $Output$ is the set of measurable outputs of plant, $PlantLocation$ is the set of identified locations in the plant, $PhysicalProperty$ is the set of defined physical properties (e.g. temperature, energy), $MeasurementUnit$ is the set of units for the defined physical properties, $Sensor$ is the set of sensing devices deployed in the plant, $Actuator$ is the set of actuating devices deployed in the plant,

Function is the set of functions/mappings defined to represent the mathematical relations among variables.

Then, relations are defined, to represent the properties of objects. A relation is a mapping of the form: $relationName : \mathcal{C}_{L(i)} \times \mathcal{C}_{L(j)} \mapsto \{\top, \bot\}$. These may define whether an object belongs to a specific class, the relation between a plant and a model, the relation between a model and a state of the plant, the relation between an input/output of the plant and a physical property, the physical property that is measured in a specific unit, the location where a sensing/actuating device is located in, etc.

3.2 The controller and observer implementations

Upon a shift to a different model of the plant, as a result of the inference step, the implementations of the controller and the state-observer change. The new implementations, are either given and retrieved from the knowledge base or they are calculated online. We assume the actuators are driven by a simple proportional controller, while the system states are estimated (mostly for compensation of missing measurements) with a simple Luenberger full-state observer [12], as shown in (6).

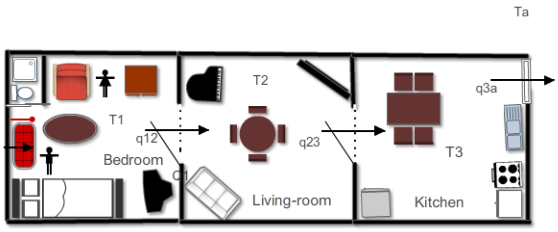
$$\begin{aligned} u &= Ke + u_0 \\ e &= x_d - \hat{x} \\ \hat{\dot{x}} &= A\hat{x} + Bu + G\hat{d} + LW(y - C\hat{x}) \end{aligned} \tag{6}$$

where $K \in \mathcal{R}^{m \times n}$ is the control gain matrix, $e \in \mathcal{R}^n$ is the error signal (difference between desired and estimated state values), $u_0 \in \mathcal{R}^m$ is the control bias that is used to cancel system disturbances, model uncertainties and retain the system at desired operation, $x_d \in \mathcal{R}^n$ is the desired system states' vector, $\hat{x} \in \mathcal{R}^n$ is the estimated system states' vector, $\hat{d} \in \mathcal{R}^q$ is the estimated uncontrolled inputs' vector, if such option exists, $L \in \mathcal{R}^{n \times n}$ is the observer gain, implemented such as the pair (A, WC) is stable and $W \in \mathcal{R}^{n \times p}$ is a weight matrix that represents the trust on each of the p measurements.

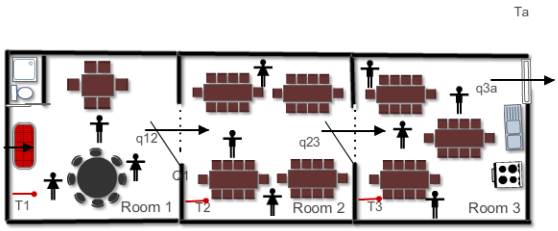
4 Case-Study Scenario

We assume an apartment with three rooms as shown in figure 2a. The apartment is equipped with a central heating installation, however, for budget reasons there is only one heating radiator in the bedroom, accompanied by one temperature sensor in the same room that measures in degrees Celsius. The equipment is used to regulate the temperature of the apartment at desired value. The design also assumes an uncontrolled input to the plant produced by the ambient temperature and modelled by a slightly open window (for simplicity we consider zero transfer of heat through the walls).

The case of replacing a sensor with another one of not compatible specifications, is described by (3) and has been specifically addressed in [14]. Here we



(a)



(b)

Fig. 2. a. The apartment plant. $Q1$ represents the heating input produced by the electric radiator, $T_i, i \in 1, 2, 3, a$ represent the temperature of the three rooms and the ambient respectively, q_{ij}, with $i, j \in 1, 2, 3, a$ representing the flow of heat among the rooms and the ambient, b. The office plant with open doors

consider the owner of the apartment buying a smart phone, which is equipped with temperature sensor. This mobile sensor is entering and leaving the apartment at different occasions during a day, therefore, at discrete sampling times the mediation component is retrieving more than one measurements. This case is described by (4). The knowledge base helps the mediation component to retrieve the measurements and also build the weight matrix W, while the observer continues to producing estimates of the state and the mediation component now feeds the control law with the fused sensors' measurements. This allows benefiting from the availability and accuracy of redundant information.

Later in time, the apartment is bought by an ICT company and is turned into an open-plan, as shown in fig. 2b. Soon after, they notice that people working in Room 3, do not feel comfortable and wear heavy clothes. So, they install temperature sensors in the other two rooms as well. In parallel, a control engineer is asked to design higher-order models of the apartment heating system. For simplicity we consider a manual design of the models, while an alternative would be for an adaptive algorithm like the one in [3] to be used in closed-loop operation. The closed-loop system now fully incorporates the sensing information available (increases the order of the model) and it is now able to maintain better temperature conditions across all rooms (of course with the limited capacity of the single actuator). This case is described by (5).

5 Simulations and Results

The devices are implemented as virtual BACnet/IP-enabled devices, using the BACnet4J API [16]. Their semantic descriptions (e.g. for a sensor, the measurement unit, its location in the plant, etc.) are created and stored in the knowledge base. Next, three plant models (1st-, 2nd- and 3rd-order) are created and stored in the knowledge base. We use the Newton's law of cooling, $Q = cA(x_i - x_j),\ i \neq j$, with Q the heat transfer in J/sec, c the heat transfer coefficient, A the area of the surface through which the heat flows and x_i, x_j the temperatures in the two sides of the surface, to derive linear state-space models of the closed-loop system in the form of (6). For each of the models, the ambient temperature is acting as an uncontrolled input. In addition, further scenario-related parameters are defined, like the steps of the electric radiator output, a model for the outside temperature, the simulation time (50 hours), the desired temperatures of rooms (25^0Celcius) and the initial temperatures of the rooms.

Initially, *sensor1* and *radiator1* are installed in *Room1* (bedroom). At time 08:00, *sensor1* breaks and is replaced by *sensor4*. At time 10:00 mobile *sensor5* enters *Room1*. At time 22:00, *sensor2* is installed in *Room2* and *sensor3* is installed in *Room3*. Finally, at time 36:00, *sensor6* and *sensor7* enter *Room3* and both leave at time 39:00. The simulation runs in 1-minute steps. During each step, the mediation component reads and stores the sensors' measurements together with their time-stamp. The processing of the measurements and the calculation of the control input is performed at 5-minute intervals. At each such step, the mediation component retrieves information about the current measurements. These are discarded if they were taken more than 2.5 minutes earlier. Moreover, in case a value is in a different unit than the one required by the current control law, the mediation component runs an inference rule [19] and retrieves the literal name of the function to invoke (from those in the Tools base) in order to perform the required transformation. The rule says: "*Find the name of the function that takes as input a real value of the given sensor's measurement unit and produces a real value in the desired measurement unit*". If no such function is returned, the measurement is discarded. In the implementation of the rule, the given and the desired measurement units are denoted as $o_1, o_2 \in MeasurementUnit$ and any symbols starting with "?" denote a variable that can take as value an object from the knowledge base of the class accepted as argument by the specific relation. The rule is written as:

$$z_1 \in \mathcal{Z} = Function(?x) \wedge hasDomain(?x, o_1) \wedge hasRange(?x, o_2) \longmapsto InferredInd(?x)$$

At that moment, in case there was any change in the sensors that comprise the closed-loop system, the mediation component retrieves the best available plant model to use for the operation of the controller, given the locations and the measurement properties/units. To this end, several inference rules are executed in the knowledge base. The first one is the:

$$z_2 \in \mathcal{Z} = Output(?x) \wedge [associatedWithLocation(?x, o_3) \vee [associatedWithLocation(?x, ?y) \wedge isPartOf(o_3, ?y)]] \wedge isPhysicalProperty(?x, o_4) \longmapsto InferredInd(?x)$$

where $o_3 \in Location$ and $o_4 \in PhysicalProperty$ are the given measurement location and the measured physical property of the sensor, respectively. The above rule means: "*Find all available outputs of models, that are either associated directly with the given location or are associated with a different location which is, however, defined as part of the given location, and that are associated with the given physical property (e.g. temperature)*". At the end, a decision algorithm is invoked which finds the model that is of the highest order, while still controllable and observable under current situation, and returns its constant parameter matrices as defined in (1) and (6). At this moment, the dimensions of all vectors and all parameters of the model to be used, are considered known. Given the new model, the mediation component invokes functions to calculate the observer's and the controller's gain matrices, $L^{(I)}$ and $K^{(I)}$ respectively. This is performed with simple pole placement for observability (pair A, WC) and stability (pair A, B). The new state vector estimation is based each time on the model and the designed observer. The observed state values are used by the controller to compute the next control input value. It is noted that the control input retains the previous value until a new one is produced.

The result is that the closed-loop system is able to transparently integrate any new component and use the new information to operate smoothly despite the events introduced during operation. No downtime or manual re-configuration are required.

6 Conclusions

We have presented a new architecture that can be adopted in the design of feedback control systems, in order to take advantage of the scalability characteristics offered by the combination of the classical control capabilities with a cyber infrastructure and semantic interoperability protocols and interfaces.

The scope of the work was not to advance the control algorithms as such. The current industrial practice suggests using standard controllers (e.g. PID) and applying the interoperability of components at higher application levels. We believe that much more advance intelligent control algorithms, already developed in the literature, can enormously impact the industrial applications if there is a framework for their deployment in large feedback control systems.

There is still lot of work to be done, before we can claim achieving the objectives of this work. Our immediate next steps will be the thorough investigation of the closed-loop system stability within the proposed architecture, as well as, the implementation of a demonstration setup that will pilot test the applicability in real-life scenarios.

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

References

1. Antoniou, G., Harmelen, F.V.: Web ontology language: Owl. In: Handbook on Ontologies in Information Systems. pp. 67—-92. Springer (2003), http://link.springer.com/chapter/10.1007/978-3-540-92673-3_4
2. ASHRAE: BACnet Website, http://www.bacnet.org/
3. Bendtsen, J., Trangbaek, K., Stoustrup, J.: Plug-and-Play ControlModifying Control Systems Online. I E E E Transactions on Control Systems Technology 21(1), 79–93 (2013)
4. Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M.: Diagnosis and fault-tolerant control. Springer Verlag (2003)
5. Chen, J., Patton, R.J.: Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers (1999)
6. E-Lite: DogOnt (2012), http://elite.polito.it/dogont
7. Farrell, J., Polycarpou, M.: Adaptive Approximation Based Control: Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches. J. Wiley (2006)
8. Gertler, J.: Fault detection and diagnosis in engineering systems. CRC (1998)
9. Holger, N., Compton, M.: The Semantic Sensor Network Ontology : A Generic Language to Describe Sensor Assets. In: 12th AGILE International Conference on Geographic Information Science, Workshop on Challenges in Geospatial Data Harmonisation. Hannover, Germany (2009), http://plone.itc.nl/agile_old/Conference/2009-hannover/shortpaper.htm
10. Isermann, R.: Fault-diagnosis systems: an introduction from fault detection to fault tolerance. Springer Verlag (2006)
11. Lange, C.: Ontologies and languages for representing mathematical knowledge on the semantic web. Semantic Web (i) (2013)
12. Luenberger, D.: Observers for multivariable systems. Automatic Control, IEEE Transactions on 11(2), 190–197 (Apr 1966)
13. Mazzocchi, S.: Closed World vs. Open World: the First Semantic Web Battle (2005), http://www.betaversion.org/~stefano/linotype/news/91/
14. Milis, G.M., Panayiotou, C.G., Polycarpou, M.M.: Towards a Semantically Enhanced Control Architecture. In: IEEE Multi-Conference on Systems and Control. Dubrovnik, Croatia (2012)
15. Samad, T., Parisini, T.: Systems of Systems. In: Samad, T., Annaswamy, A. (eds.) The Impact of Control Technology. available at www.ieeecss.org (2011)
16. Serotonin-Software: BACnet I/P for Java (2011), http://sourceforge.net/projects/bacnet4j/
17. Stoustrup, J.: Plug & Play Control: Control Technology Towards New Challenges. European Journal of Control 15(3-4), 311–330 (Aug 2009), http://ejc.revuesonline.com/article.jsp?articleId=13584
18. Suresh, P., Joglekar, G., Hsu, S., Akkisetty, P., Hailemariam, L., Jain, A., Reklaitis, G., Venkatasubramanian, V.: OntoMODEL: Ontological Mathematical Modeling Knowledge Management. Computer Aided Chemical Engineering pp. 985—-990 (2008)
19. W3C: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004), http://www.w3.org/Submission/SWRL/