A Cognitive Fault-Detection Design Architecture

Georgios M. Milis, Demetrios G. Eliades, Christos G. Panayiotou, and Marios M. Polycarpou
KIOS Research Center for Intelligent Systems and Networks
Department of Electrical and Computer Engineering
University of Cyprus
Nicosia, Cyprus
Email: {milis.georgios, eldemet, christosp, mpolycar}@ucy.ac.cy

***Abstract*—This paper presents a novel architecture for the design of fault-detection schemes, aiming to automate the cognitive process performed by human experts when designing fault detection schemes for certain systems. The work starts with the identification of types of cyber-physical components participating in a fault-detection scheme. These are semantically characterized, adopting a model driven by previous efforts of the World Wide Web Consortium on the semantic composition of Web services. The semantic characterizations of the components are then exploited by a Cognitive Agent with semantic reasoning capabilities, to achieve the configuration of a fault-detection scheme, given a set of specifications and available components. The Cognitive Agent has access to a knowledge representation model and is able to interact with human operators and with the components to enrich its knowledge for making and enforcing decisions about the configuration. The applicability of the architecture and the reasoning steps are demonstrated through the configuration of a water contamination event-detection scheme with learning capabilities within a smart water distribution network.**

I. Introduction

Today's engineered systems consist of several cyber and physical components, like sensors for monitoring system states, electrical and mechanical actuators, controllers and a number of other software components such a fault-detection and online-learning implementations. Over time, it is inevitable that one or more of these components will fail or system dynamics will move out of expected bounds due to external events, necessitating the utilization of fault/event detection and isolation mechanisms [1], [2]. By enabling the detection and diagnosis of miss-operation of systems, these mechanisms can help, e.g., in saving energy, in reducing economic cost and/or in avoiding critical consequences of cascading effects due to inter-dependencies with other systems.

During the last two decades, various methodologies have been developed and proposed for detecting, identifying, isolating and accommodating faults [1]–[3]. In general, fault detection methods can be classified into model-free (or data-driven) and model-based methods.

Model-free methods are the most commonly used, since they can be developed without the requirement of understanding the underline system's model [1]. Examples are, quantitative methods (e.g., neural networks, statistical classifiers), and qualitative methods (e.g., expert systems, fuzzy logic, pattern recognition, trend analysis) [4].

Model-based methods, on the other hand, require additional modeling and calibration effort, since a model with physical significance has to be developed using *a-priori* knowledge of the system. Again, there are examples of quantitative methods (e.g., observer-based/Kalman-filter state and parameter estimation, parity space) and qualitative methods (e.g., fault-trees and other causal models) [4].

Fault diagnosis methodologies with learning capabilities have also been proposed in the past years, which combine model-based analytical redundancy and computational intelligence tools, i.e. neural networks, to detect faults and to learn the unknown fault dynamics [5]–[7]. By learning the unknown fault dynamics, isolating the type of fault and identifying its magnitude, it is possible to change the control input to accommodate the fault, during operation [8].

Designing a fault-detection scheme for a certain system is a complicated procedure which relies on the knowledge and reasoning capabilities of a human expert. In practice, a human expert should have a broad background knowledge of tools (e.g., state-of-the-art fault-detection methods, online learning methods based on computational intelligence, state-estimation methods, etc.) and in which situations these are best suited, in order to make an informed selection that fully exploits the available measurements, constraints and objectives. Depending on the system which needs the fault-detection service, as well as the preferred fault-detection method, different schemes can be designed, consisting of smaller components. In practice, it is very rare, if not impossible, to find and employ a human expert of such breadth of knowledge whenever a fault-detection scheme is required for a certain system. An additional drawback in current practices is the lack of mechanisms to allow online (and where possible automatic) replacement of individual components or of the overall fault-detection scheme. Therefore, in case new and advance methods become available and are implemented as components, they can only be deployed by expert engineers who will re-design the overall fault detection scheme. This may inhibit industry adoption of advanced fault-detection methodologies and may act as a barrier to the exploitation of research results.

The above motivate our proposal for a design of an architecture which is able to utilize expert knowledge and cognitive reasoning based on semantics, to reproduce part of the reasoning procedure of a human expert. This would allow new components (e.g., new on-line learning algorithms or new fault-detection algorithms) to be gradually deployed as they become available, by automatically configuring the

components of the fault-detection scheme. The aim is to achieve the best composition and to facilitate interoperability with the other available components using a common framework for efficient exchange of data and knowledge. It is emphasized that our work is not focusing on the design of any new fault-detection algorithm or components, but rather on the online configuration of fault-detection schemes using existing components. Moreover, it is envisioned that the adoption of a component-based fault-detection design will facilitate the faster exploitation, testing and demonstration of academic research in industrial applications. To demonstrate the application of the proposed architecture, a case-study of configuring a contamination event detection scheme with online learning capabilities for drinking water distribution networks is presented.

The paper is organized as follows: Section II formulates the problem by revisiting and formulating the fault-detection theory and also breaking the fault-detection scheme into a set of individual components. Section III then presents the design of the proposed architecture, the knowledge graph with the semantic characterization of the components and the cognitive agent which utilizes semantic reasoning towards the automatic configuration of fault-detection schemes. Section IV demonstrates the architecture in the configuration of a scheme for the detection of contamination events in water distribution networks. Finally, Section V concludes the paper and discusses possible future directions.

II. Problem formulation

Fault detection is defined as the problem of determining whether a system is operating under normal or abnormal conditions (e.g., due to the occurrence of a system, actuator or sensor fault). Typically, a fault-detection algorithm is specifically designed for a certain system, taking into account the system's measurable variables, known dynamics and other information available. The output of a fault-detection algorithm at discrete time k, is given by:

$$d(k) = f(y(k), u(k); \zeta), \tag{1}$$

where $f(\cdot)$ is a fault-detection composite function, $y(k)$ and $u(k)$ are the measured output vector and the known input vector of the system respectively and ζ is a set of parameters related to the system and the considered fault-detection implementation. In general, the detection signal $d(k)$ can be a vector corresponding to a set of various fault-level classes. Depending on the specifications, the detection signal $d(k)$ can be: i) binary, i.e., of the form $\{0, 1\}$ or {True, False}, thus informing of the detection of a fault or not; ii) a real number, e.g., representing the probability or the risk of fault existence; iii) a more generic class of values, e.g., a fault class type, a color-based risk-level scheme, a linguistic variable, a fuzzy value, etc.

Two families of fault-detection algorithms are typically considered: the "model-free" and the "model-based" methods [1]. The former process the measured output signals plus other known or computed signals that may be required, in order to generate certain features (e.g., operation state). These features are passed through a "detection logic" component and are compared to their value in non-faulty operation. Typical examples are the limit-checking approach, the change-detection approach (such as the CUSUM) [9] and other statistical-based approaches. In addition, learning methodologies have also been applied within a model-free fault detection context [10]. On the other hand, the "model-based" fault-detection methods process the measured output and known input signals utilizing also a known system-model, in order to generate certain features (e.g., state estimation residuals). As in the model-free case, these features are then passed through a "detection logic" module and are compared to their values in non-faulty operation. Typical examples are the analytical redundancy fault-detection schemes, utilizing tools such as state-estimation, filtering, parametric uncertainty learning and adaptive approximations [5], [8], [11]–[14].

In the general case, the fault-detection algorithm $f(\cdot)$ can be considered as composed of sub-components, some of which are basic (mandatory) for all implementations of fault-detection while others are required only in certain cases. All these components are discussed in the sequel.

A. Basic Components

The first basic component of a fault-detection scheme is the "Detection logic", given by the function:

$$d(k) = f_d(r(k), t(k); \zeta_d), \tag{2}$$

where $d(k)$ is the detection signal defined also in (1), $f_d(\cdot)$ is the detection logic implementation, $r(k)$ is a feature signal which is computed by a separate function in order to be compared against a threshold signal $t(k)$, and ζ_d is a set of other parameters required by the adopted detection-logic method.

The second basic component in a fault-detection scheme is the "Feature" given by the function:

$$r(k) = f_r(y(k), u(k), \hat{x}(k); \zeta_r), \tag{3}$$

where $r(k)$ is the feature signal defined in (2), $f_r(\cdot)$ is the implementation of a method to derive the signal, $y(k)$ and $u(k)$ are the measurable outputs and the known inputs of the system respectively, if available, $\hat{x}(k)$ is the estimated system state (optionally used) and ζ_r is a set of parameters required by the adopted method.

As an example, in a model-free fault-detection scheme, the detection logic function f_d may be a limit-check, e.g., comparing the measured state $r(k) \equiv y(k)$ or its difference $r(k) \equiv \frac{y(k)-y(k-1)}{\Delta\tau}$ with a given upper and/or lower bound $t(k)$, such that $|r(k)| \leq t(k)$. Another example is the CUSUM change-detection method, where the cumulative sum of the differences of the measured state from a pre-defined parameter (e.g., the statistical mean of the state signal) are compared with a given threshold. In a model-based fault-detection scheme, the detection logic function f_d may be comparing the state-estimation error $r(k) \equiv y(k) - \hat{x}(k)$ with a given or computed threshold signal.

A third basic component of a fault-detection scheme is the "Threshold", which undertakes the task of generating the threshold signal against which to compare the detection-logic features and is given by the function:

$$t(k) = f_t(g(\cdot), y(k), u(k); \zeta_t), \tag{4}$$

where $t(k)$ is the threshold signal defined in (2), $f_t(\cdot)$ is the implementation of a method to derive a threshold, $g(\cdot)$ is a function representing new (additive) system dynamics, $y(k)$ and $u(k)$ are the measurable outputs and known inputs of the system respectively (optionally available) and ζ_t is a set of parameters required by the adopted implementation of the function. For instance, the threshold parameters may correspond to bounds on parameters or on parts of the system state dynamics, derived from expert knowledge about the system operation or from the offline processing of historical data, etc.; it may be the output of a stochastic process on the detection logic features or it may be computed given knowledge about a system model with parameter uncertainty or function uncertainty. In all cases, the threshold may be a constant value or a time-varying adaptive signal.

In summary, at a minimum, the fault-detection scheme is composed of the functions specified above, such that $f \equiv f_d(f_r(\cdot), f_t(\cdot), \zeta)$. That is, the detection logic component compares measured or computed features of the system, against a pre-selected or computed threshold signal.

B. Advanced Components

In addition to the three basic components described in the previous sub-section, additional components may be required by certain model-based fault-detection schemes. For instance, in some model-based cases the estimation signal of the system states $\hat{x}(k)$ is required. For this, a "State-Estimation" component is required, given by:

$$\hat{x}(k+1) = f_e(g(\cdot), y(k), u(k); \zeta_e), \tag{5}$$

where $\hat{x}(k+1)$ is the estimated system states signal at the next time step, $f_e(\cdot)$ is the adopted implementation of the state-estimation, $y(k)$ and $u(k)$ are the vectors of system's measured output and known system inputs respectively, ζ_e is a set of other parameters required by the adopted implementation and $g(\cdot)$ is a function representing a new additive part of the system state dynamics. For instance, the State-Estimation component may correspond to a "Kalman filter" which produces estimates based on some prior knowledge about the states, a measurement vector and certain parameters of measurement and state's uncertainty; it can also be a "Luenberger observer" which, based on a known model of system dynamics and the available measurements, produces estimates of the state. The state estimation may be also implemented as a black-box by a system simulation, e.g., using the EPANET software for simulating water distribution systems[1] or CONTAM for simulating contaminant propagation in buildings[2].

[1]http://www.epa.gov/water-research/epanet

[2]http://www.nist.gov/el/building_environment/contam_software.cfm

Furthermore, in the case of having a system model with unknown dynamics $g(\cdot)$, a "Learning Component" can be utilized, to learn the unknown function using a suitable approximation structure (e.g., neural network, polynomial function, radial-basis functions, wavelets, fuzzy systems, etc), such that $g \equiv \hat{g}$. This module undertakes the task to learn an unknown part of the overall state dynamics function and can be described in general by:

$$\hat{g}(k) = f_\theta(y(k), u(k), \zeta_\theta) \tag{6}$$

where $\hat{g}(k)$ is the estimated value of the unknown function, $f_\theta(\cdot)$ is the adopted online learning implementation and ζ_θ are any other parameters required by the adopted implementation (e.g., the convergence rate, knowledge about the structure of the function, etc.). The output of this component may be used as input to components allowing update of the system model on which they base their implementation (e.g., certain State-Estimation or Threshold components).

C. Components Database

All implementations of components (functions) of the types discussed above, can be considered as being elements of a function-set $\mathcal{F}$, thus forming a database of components. The set $\mathcal{F}$ is defined as a super-set of the following sub-sets of components:

- $\mathcal{F}_d = \{f_d^i | i = 1, ..., n_{F_d}\}$: all implementations of the Detection-logic function, with cardinality n_{F_d}
- $\mathcal{F}_r = \{f_r^i | i = 1, ..., n_{F_r}\}$: all implementations of the Feature function, with cardinality n_{F_r}
- $\mathcal{F}_t = \{f_t^i | i = 1, ..., n_{F_t}\}$: all implementations of the Threshold function, with cardinality n_{F_t}
- $\mathcal{F}_e = \{f_e^i | i = 1, ..., n_{F_e}\}$: all implementations of the State-Estimation function, with cardinality n_{F_e}
- $\mathcal{F}_\theta = \{f_\theta^i | i = 1, ..., n_{F_\theta}\}$: all implementations of the Online Learning function, with cardinality n_{F_θ}

D. Cognitive Agent

Depending on the system and the given fault detection specifications, an expert engineer would have selected and designed a fault-detection scheme using implementations of all basic components and possibly utilized additional components for state-estimation and learning, depending on the availability of measurements, models of state dynamics, as well as specific domain knowledge expertise. In other words, for the decision, the expert engineer relies on reasoning which considers the available knowledge about the domain and the fault-detection systems engineering, including the associated semantics of each component.

The challenge addressed in this work is to design an architecture with the ability to utilize pre-modeled expert knowledge and a set of fault-detection specifications (including performance criteria) and automatically design and configure a suitable fault-detection scheme, for a large class of systems. This is written as:

$$\sigma = f_\sigma(\mathcal{G}, \mathcal{S}, \mathcal{F}), \tag{7}$$

$$f \equiv f_F(\sigma, \mathcal{F}_d, \mathcal{F}_r, \mathcal{F}_t, \mathcal{F}_e, \mathcal{F}_\theta), \tag{8}$$

where function f is the constructed fault-detection scheme, σ is a set of decision signals that correspond to the selection of specific components from the subsets of $\mathcal{F}$ defined earlier, $f_\sigma(\cdot)$ is a function implementing the reasoning and the decision on the design, $f_F(\cdot)$ is the function which enforces the decision and the subsequent construction of the function $f(\cdot)$, $\mathcal{G}$ is the graph modeling the available knowledge, $\mathcal{S}$ is the set of fault-detection specifications given to the function (e.g., the preferred form for the detection signal). The elements of σ are index-vectors for the sets of components defined earlier, such that $\sigma = \{\sigma_{f_d}, \sigma_{f_r}, \sigma_{f_t}, \sigma_{f_e}, \sigma_{f_\theta}\}$, where $\sigma_{f_d} \in \{1, ..., n_{F_d}\}$, $\sigma_{f_r} \in \{1, ..., n_{F_r}\}$, $\sigma_{f_t} \in \{1, ..., n_{F_t}\}$, $\sigma_{f_e} \in \{1, ..., n_{F_e}\}$ and $\sigma_{f_\theta} \in \{1, ..., n_{F_\theta}\}$.

The next sections provide details about the proposed architecture for the implementation of the functions f_σ and f_F, emphasizing on the knowledge graph and the reasoning mechanism.

III. Architecture Design

The fault-detection architecture, which implements the functions described in the previous section, is depicted in Fig. 1. The top-part of the figure illustrates the system on which fault-detection is performed. Assuming, for generality, a controlled plant by a closed-loop configuration, the system comprises the Plant with its states $x(k)$ measured by a set of Sensors, the control configuration and the driven set of Actuators that act on the Plant. The middle-part of the figure shows the composite fault-detection scheme, comprising all components discussed in previous section, with their input/output connections. The inputs to the fault-detection scheme from the system are the measured outputs $y(k)$ and the known controlled inputs $u(k)$. The output is the resulted detection signal $d(k)$. It is noted that the white boxes refer to the sets of available components of each type and not to specific implementations. The selection of specific implementations is performed through the decision signal σ given by the Cognitive Agent (blue dashed line). The bottom layer illustrates the design of the cognitive agent. As discussed in Section II, the cognitive agent function f_σ utilizes the stored knowledge in the knowledge graph $\mathcal{G}$ (including the characterization of components discussed in subsequent sub-sections) and any given specifications $\mathcal{S}$ and produces a decision as to what implementations of components to adopt for the fault detection. The decision signal σ is passed to the function f_F that invokes the selected implementations found in the database of components $\mathcal{F}$. An important added-value feature, which is out of the scope of this work, is that the stored semantic knowledge can be also enriched from internet-based remote services with access to the database.

The following sub-sections go into details on the implementation of the Cognitive Agent.

A. The Knowledge Graph

The definitions and the design of the knowledge objects and the subsequent knowledge graph $\mathcal{G}$ have been already described in a recent work published by the same authors [15], with application in "Smart Water Networks". In summary, "knowledge objects" are defined and grouped in dedicated "type-sets" and relations between objects of pairs of type-sets are defined as bipartite graphs. The combination of these "relation-graphs" results is a large graph of knowledge objects (the "knowledge graph") which can be traversed in order to extract more complex knowledge facts and reason about relevant semantic queries. The formal representation of the knowledge graph achieves the encoding of the required knowledge in machine readable format, to facilitate the automation of the cognitive process for the design of the fault-detection scheme. These definitions are adopted in this work and are not repeated here.

The "graphical language" used for the representation of the semantic characterization of components is also adopted and extended in this work. That is, the physical representation of the fault-detection scheme is shown in an "Implementation" layer, while the knowledge part is shown in a "Knowledge graph" layer. The latter is split into: i) the "Components" sub-layer where the knowledge objects representing the implementations of fault-detection components are hosted; ii) the "Profile" sub-layer where the knowledge objects representing the inputs $\mathcal{A}$, the outputs $\mathcal{W}$ and other parameters $\mathcal{Z}$ of the components' implementations lie; iii) the "Domain characterization" sub-layer that hosts the more detailed (domain-relevant) knowledge about the types $\mathcal{Q}$ of the inputs/outputs/parameters, as well as their characterization in terms of represented physical properties $\mathcal{P}$, measurement units $\mathcal{M}$ and locations $\mathcal{L}$; and iv) the "Measured-known parameters" sub-layer which hosts the knowledge about measured ($\mathcal{Y}$) and known (Θ) variables of the system. For convenience, Fig. 2 illustrates an example of the semantic characterization of a "Detection logic" component. In addition, it is noted that the knowledge objects are shown with circles and the type-sets are shown with dashed-line rectangle containers. The semantic relations between the objects are shown by the edges.

The semantic characterization of a component is defined as:

$$\mathcal{G}^\omega(\mathcal{V}^\omega, \mathcal{E}^\omega) = f_\lambda(\omega) \tag{9}$$

where $\omega \in \mathcal{F}$ is a knowledge object representing a certain component and $f_\lambda(.)$ is a function representing the mapping of the object to a tree sub-graph of the overall knowledge graph, such that $\mathcal{G}^\omega \subseteq \mathcal{G}$, and has a set of nodes $\mathcal{V}^\omega$ and a set of edges $\mathcal{E}^\omega$.

The semantic characterization tree consists of "input- output- parameter- branches" which start from the component and pass through inputs, outputs and parameters, respectively, associated with it and end on relevant leaf-nodes (objects) in the domain characterization sub-layer. The input-, output- and parameter- branches in Fig. 2 are shown with thick green, blue and red edges and may pass through a finite number of input, output and parameter nodes respectively (none is also an option).

For visualization purposes, the example is limited to showing only one object of each type-set in the "Profile" sub-layer. In addition, the measured and known variables of the system are characterized by trees marked with thick purple and black

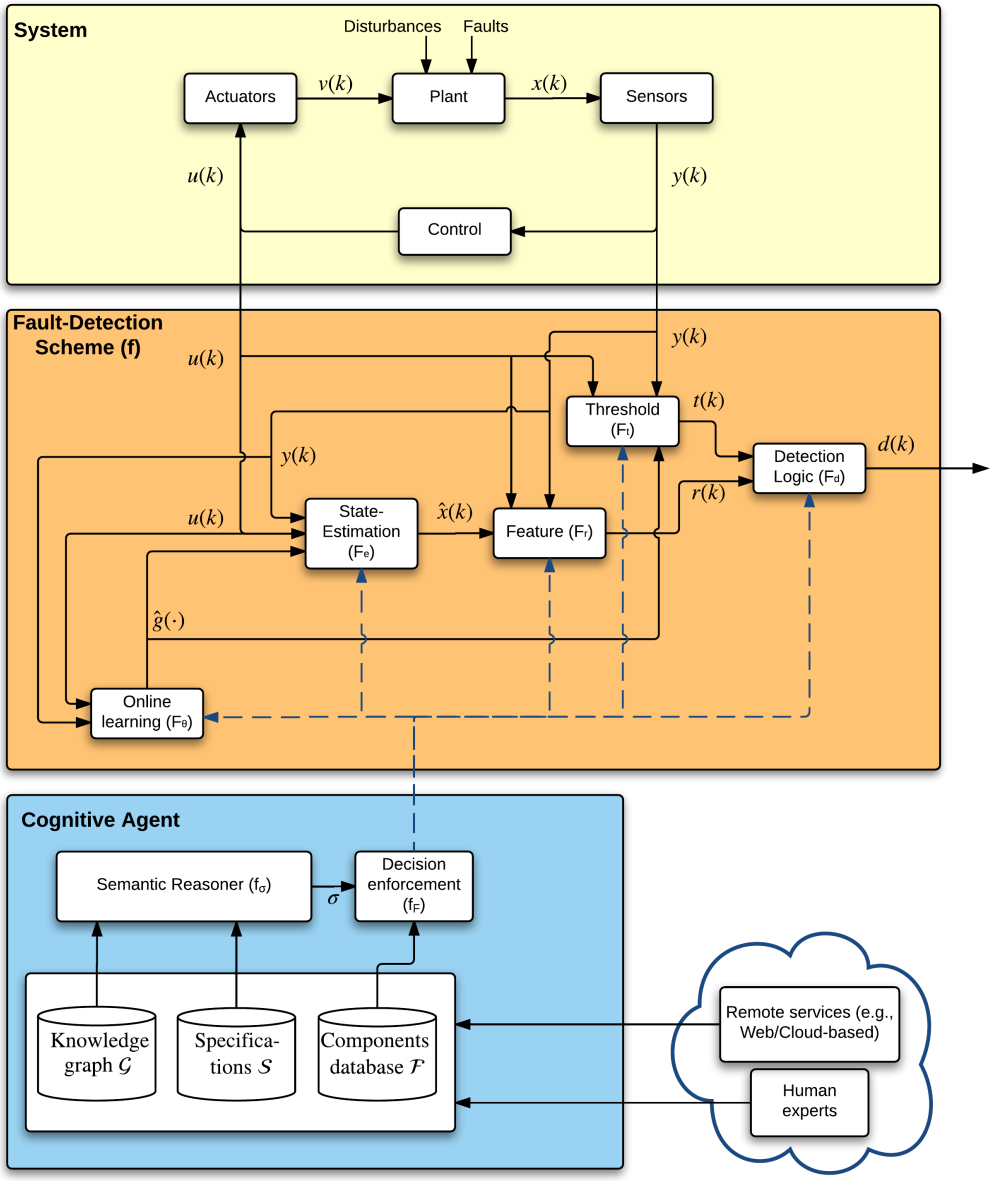

Fig. 1: Block diagram of the architecture. Top: the System on which fault detection is performed. Middle: the Fault-Detection Scheme. Bottom: the Cognitive Agent.

edges respectively. Finally, each specification given in advance and that needs to be met by the fault-detection scheme, is filling-in the relevant parts of the knowledge graph with a different color.

For instance, the specification marked with light blue color asks that a component must be available of which at least one input must be of type q. Another example is the specification marked with light green color, which says that a component is needed that has some parameter associated with location l.

The following subsection describes how the above semantic characterizations of components, in combination with measured-known parameters and given specifications, can be utilized in reasoning about what types of components and what specific implementations of them to adopt in a fault-detection scheme.

B. Semantic Reasoning Algorithm

The objective of the reasoning algorithm can be briefly described as: *Given the measured-known variables of the underline system, find the required types of components and the exact implementations of them that meet the pre-defined specifications*.

Since the algorithm seeks to connect components together in terms of their input-output relations, a key part of the reasoning is to decide whether the output of one component

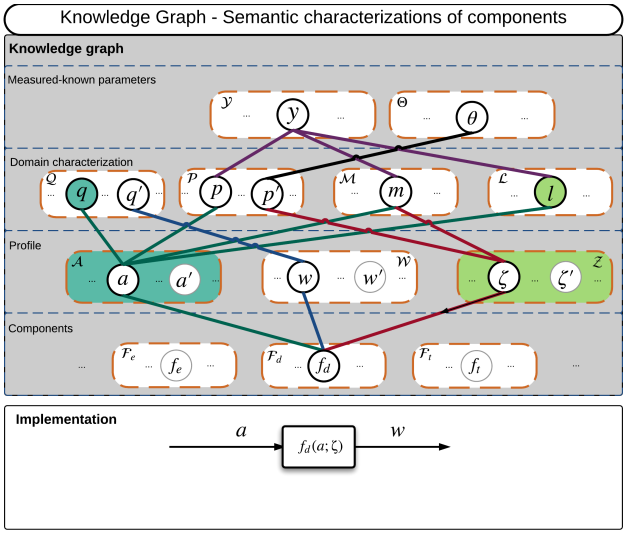

Fig. 2: A "Knowledge Graph" showing the semantic characterization tree of a detection logic component f_d

can be used as the input of another component, creating a meaningful connection. This sub-process is called "output-input semantic matching" and is performed several times during the execution of the algorithm (presented later). Assuming the characterization tree of the object f_d, with an input a, the leaf nodes of the input branch of this tree that lie in the "Domain characterization" sub-layer, that is, the "input-branch-leaf-nodes", are the elements of a subset of nodes, $\mathcal{V}_\sigma^{(f_d,a)}$. Similarly, the "output-branch-leaf-nodes" and the "parameter-branch-leaf-nodes" are defined. These subsets of nodes are very important for the implementation of the "output-input semantic matching". Assuming the algorithm checks the semantic matching of an output $w\prime$ of the component f_t to an input a of a component f_d, the matching is confirmed if *the set of "output-branch-leaf-nodes" of the output is a super-set of the set of "input-branch-leaf-nodes" of the input*. In other words, $\mathcal{V}_\sigma^{(f_t,w\prime)} \supseteq \mathcal{V}_\sigma^{(f_d,a)}$.

Having defined the above, the reasoning is implemented by the algorithm which is outlined below:

- Start with a "Detection logic" component
- Find a "Feature" component with outputs that match semantically with the "Detection logic" inputs. If there are remaining inputs and/or required parameters, find measured-known parameters of the system that match semantically with them.
- Repeat the above steps for a "Threshold" component, then sequentially for a "Feature", a "State-Estimation" and an "Online Learning" component.
- In all cases, check if specifications are met by the selections, otherwise re-iterate within components.
- If all successful, create the decision signal σ such as to enforce the selection of the matching components.

It is noted that this work assumes an offline ranking of components based on their known "quality of service", which, in the cases of multiple options, leads to the selection of the one with higher ranking. It is also noted that the above procedure is iterative, that is, the checks are repeated for all components of the same type in the order defined by their offline ranking. In case of inability to find an appropriate component at any relevant step, the algorithm terminates and informs inability to configure a fault-detection scheme with currently available components. An illustrative case-study is presented in the following section, to clarify the procedure.

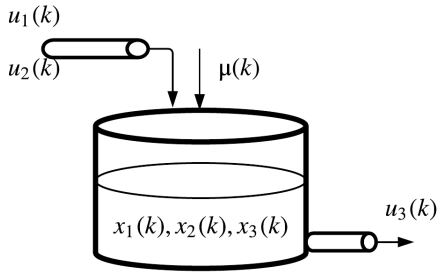

Fig. 3: A water storage tank utilised in the illustrative case-study

IV. Illustrative Case-Study

A. Water-Tank Contamination Event Detection

As mentioned in Section I, faults can be defined in general as any shifts from the normal operation of systems. The illustrative use case, described in this section, is related to the problem of contamination event detection in drinking water distribution networks, exploiting the dynamics and measurements of chlorine concentrations, as approached in [16]. The underlying assumption is that contaminants injected in drinking water will react with and affect the concentration of chlorine [17]. For instance, a bacterial toxin may decrease the concentration of free chlorine. The low cost of sensors measuring chlorine concentration turns them appropriate for wide use by water utilities in monitoring for contamination events.

In most of the cases, the actual chlorine reaction dynamics are not known, resulting to the use of empirical models [18]. Chlorine concentration dynamics depend on a reaction rate coefficient, which in turn depends on several pipe parameters. Different studies have proposed various models based on laboratory experiments, ranging from first-order linear models to more complex second-order ones.

Consider a water storage tank of cylindrical shape in a water network, as depicted in Fig. 3. Water treated with chlorine is added into the tank from a pipe at the top and water is removed from the tank from a pipe situated at the bottom of the tank. Let k denote the discrete time with sampling interval $\Delta\tau$ measured in hours. The system state is denoted as $x(k) = [x_1(k), x_2(k), x_3(k)]^\top$ and the controlled inputs are denoted as $u(k) = [u_1(k), u_2(k)]^\top$, where $x_1(k)$ is the tank volume in liters (L), $x_2(k)$ is the chlorine concentration of the tank

water in (mg/L) and $x_3(k)$ is the concentration of a certain contaminant in (mg/L); $u_1(k)$ is the volume of water entering the tank in liters (L), controlled through a valve and $u_2(k)$ is the chlorine inflow in the tank measured in (mg), controlled using a set-point chlorination booster system. There is also an uncontrolled input $u_3(k)$ representing the volume of water exiting the tank, driven by consumer demands. The state-space formulation of the water tank system is as given in [16].

B. Cognitive Agent Reasoning Example

This subsection starts with presenting the components currently available in the functions database (set $\mathcal{F}$), as well as the respective knowledge introduced in the graph $\mathcal{G}$ by their semantic characterization. For clarity of the concept's proof, only one measured system output (chlorine concentration) and only one known system input (chlorine inflow) are assumed; the inclusion of the rest can be addressed in a similar way.

- **Detection logic**: Two components, f_d^1 and f_d^2. The first component, f_d^1, produces an output d^1 of type q^1 = "detection", that represents the property p^1 = "binary" in measurement units m^1 = "true-or-false". It requires an input r^1 of type q^2 = "feature" and an input t^1 of type q^3 = "threshold". The second component, f_d^2, produces an output d^2, of type q^1 = "detection", that represents the property p^2 = "probability" in measurement units m^2 = "in-range-[0,1]". It requires an input r^2, of type q^2 = "feature" and an input t^2, of type q^3 = "threshold".
- **Feature**: Two components, f_r^1 and f_r^2. The first component, f_r^1, produces an output r^3, of type q^2 = "feature". It requires an input y^1, of type q^4 = "measured-state". The second component, f_r^2, produces an output r^4, of type q^2 = "feature". It requires an input y^2, of type q^4 = "measured-state" and an input $\hat{x}^1$, of type q^5 = "estimated-state". The two inputs inherit the characterization of the measurement to be fed, for instance, they may represent the property p^3 = "chlorine-concentration" in measurement units m^3 = "mg/L" at location l^1 = "tank-1".
- **Threshold**: Two components, f_t^1 and f_t^2. The first component, f_t^1, produces an output t^3, of type q^3 = "threshold", representing a constant bound. It requires a parameter ζ^1, of type q^5 = "historical-state-data". The second, f_t^2, produces an output t^4, of type q^3 = "threshold". It requires an input y^3, of type q^4 = "measured-state" that represents the property p^3 = "chlorine-concentration" in measurement units m^3 = "mg/L" at location l^1 = "tank-1", an input u^1, of type q^5 = "known-system-input" that represents the property p^4 = "chlorine-inflow" in measurement units m^3 = "mg/L" at location l^1 = "tank-1" and an **optional** input g^1 of type q^6 = "system-dynamics-function".

The specifications given for the design of the fault-detection scheme are given by: $\mathcal{S}$ = {detection signal as m^1 = "true-or-false", at location l^1 = "tank-1", for the property p^3 = "chlorine-concentration"}.

The execution of the algorithm is as follows:

- The "Detection logic" component f_d^1 is selected since the component f_d^1 violates the specification regarding the output in units m^1 = "true-or-false".
- The "Feature" component f_r^1 is selected since it does produce an output that matches the input of the selected detection logic component ($\mathcal{V}_\sigma^{(f_r^1,r^3)} = \{q^2\} \supseteq \mathcal{V}_\sigma^{(f_d^1,r^1)} = \{q^2\}$) and its input is matched with the measured system state ($\mathcal{V}_\sigma^{(y)} = \{q^4, p^3, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_r^1,y^1)} = \{q^4\}$). On the other hand, although it also matches the input of the detection logic component, there is no component producing an output of type q^5 = "estimated-state" which is required as input by the component f_r^2.
- The "Threshold" component f_t^2 is exploited first as it is assumed ranked higher and it is selected since it produces an output that matches with the input of the detection logic component ($\mathcal{V}_\sigma^{(f_t^2,t^4)} = \{q^3\} \supseteq \mathcal{V}_\sigma^{(f_d^1,t^1)} = \{q^3\}$). Moreover, its two inputs are matched by measured and known variables of the system ($\mathcal{V}_\sigma^{(y)} = \{q^4, p^3, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_t^2,y^3)} = \{q^4, p^3, m^3, l^1\}$ and $\mathcal{V}_\sigma^{(u)} = \{q^5, p^4, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_t^2,u^1)} = \{q^5, p^4, m^3, l^1\}$). The input that corresponds to the system dynamics function is not matched since there is no component producing a relevant output, however, it is mentioned as optional.
- At this stage, a possible configuration of the fault-detection scheme has been found, with component f_d^1 producing the detection signal and receiving inputs from components f_r^1 and f_t^2. It can be seen that all specifications are met by this configuration since the knowledge objects associated with the specifications are included in the characterizations of the selected components. Therefore, the configuration is confirmed and the algorithm terminates.

Some time in the future, two more components become available in the database $\mathcal{F}$, as follows:

- **State-Estimation**: One component, f_e^1, which produces an output $\hat{x}^2$, of type q^5 = "estimated-state", that represents the property p^3 = "chlorine-concentration" in measurement units m^3 = "mg/L" at location l^1 = "tank-1". It also has an input u^2, of type q^6 = "known-system-input" that represents the property p^4 = "chlorine-inflow" in measurement units m^3 = "mg/L" at location l^1 = "tank-1" and an input g^2 of type q^7 = "system-dynamics-function".
- **Online Learning**: One component, f_θ^1, which produces an output $\hat{g}^1$, of type q^7 = "system-dynamics-function", that is associated with the property p^3 = "chlorine-concentration" in measurement units m^3 = "mg/L" at location l^1 = "tank-1". It requires an input y^4, of type q^4 = "measured-state" that represents the property p^3 = "chlorine-concentration" in measurement units m^3 = "mg/L" at location l^1 = "tank-1" and an input u^3, of type q^6 = "known-system-input" that represents the property p^4 = "chlorine-inflow" in measurement

units m^3 = "mg/L" at location l^1 = "tank-1".

The availability of the new components causes the re-execution of the algorithm, which will produce the following results:

- The "Detection logic" component f_d^1 is selected again for the same reason as in the first case.
- The "Feature" component f_r^2 is exploited first and is selected this time since it produces an output that matches the input of the selected detection logic component ($\mathcal{V}_\sigma^{(f_r^2,r^4)} = \{q^2\} \supseteq \mathcal{V}_\sigma^{(f_d^1,r^1)} = \{q^2\}$) and its inputs are now both matched, the first with the measured system state as for the f_r^1 ($\mathcal{V}_\sigma^{(y)} = \{q^4, p^3, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_r^2,y^2)} = \{q^4\}$) and the second by the output of the State Estimation component ($\mathcal{V}_\sigma^{(f_e^1,\hat{x}^2)} = \{q^5, p^3, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_r^2,\hat{x}^1)} = \{q^5\}$).
- Then the same "Threshold" component f_t^2 is selected as before. This time its input that corresponds to the system dynamics function is also matched by the output of the Online Learning component that became available ($\mathcal{V}_\sigma^{(f_\theta^1,\hat{g}^1)} = \{q^7, p^3, m^3, l^1\} \supseteq \mathcal{V}_\sigma^{(f_t^2,g^1)} = \{q^7\}$).
- The "Online Learning" component f_θ^1 is finally selected as well. It has been seen above that its output matches with the input of the selected State Estimation component, while both its inputs are matched by the measured system output and the known system input respectively.
- A possible configuration of the fault-detection scheme comprises the component f_d^1 producing the detection signal and receiving inputs from components f_r^2 and f_t^2, while the component f_r^2 and f_t^2 receive input from the component f_e^1 and the latter receives input from the component f_θ^1. All specifications are met by this configuration, as well, therefore it is confirmed and the algorithm terminates.

The graphical representation of the semantic matches above is also possible, adopting the graphical language introduced in Fig. 2, however it is not presented here due to space limitations.

V. Concluding Remarks and Future Plans

The work presented in this paper, introduced an effort to automate the design process of a fault-detection scheme. The expert engineering and domain knowledge have been modeled using existing knowledge representation techniques and an agent has been developed with the ability to reproducing part of the cognitive process and reasoning performed by the engineer when designing the fault-detection algorithm. The primary impact of these results is the fact that it defines clear semantic interfaces between parts of the fault-detection algorithm and enables industrial setups and/or academic prototypes to allow online plugging-in of new implementations of these parts (components) without the need to re-design the whole of the fault-detection scheme. This is the first important milestone of the work, while future plans foresee the enrichment of the knowledge graph, the automation of the selection of components in case of multiple matching configurations and the extension of the work to provide a unified architecture for monitoring and control schemes.

Acknowledgment: This work is partially funded by the European Research Council (ERC) under the project ERC-AdG-291508 "Fault-Adaptive Monitoring and Control of Complex Distributed Dynamical Systems" (FAULT-ADAPTIVE).

References

[1] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. New York, NY, USA: Springer-Verlag, 2006.

[2] J. Chen and R. Patton, *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Norwell, MA, USA: Kluwer Academic Publishers, 1999.

[3] I. Hwang, S. Kim, Y. Kim, and C. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *Control Systems Technology, IEEE Transactions on*, vol. 18, no. 3, pp. 636–653, May 2010.

[4] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[5] M. M. Polycarpou and A. J. Helmi, "Automated fault detection and accommodation: a learning systems approach," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 25, no. 11, pp. 1447–1458, 1995.

[6] X. Zhang, M. M. Polycarpou, and T. Parisini, "A robust detection and isolation scheme for abrupt and incipient faults in nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 4, pp. 576–593, Apr. 2002.

[7] R. M. G. Ferrari, T. Parisini, and M. M. Polycarpou, "Distributed fault detection and isolation of large-scale discrete-time nonlinear systems: An adaptive approximation approach," *IEEE Trans. Autom. Control*, vol. 57, no. 2, pp. 275–290, 2012.

[8] V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Adaptive approximation for multiple sensor fault detection and isolation of nonlinear uncertain systems," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 137–153, 2014.

[9] M. Basseville and I. V. Nikiforov, "Detection of abrupt changes: theory and application," 1993.

[10] C. Alippi, *Intelligence for Embedded Systems: A Methodological Approach*. Cham: Springer International Publishing, 2014, ch. Fault Diagnosis Systems, pp. 249–270.

[11] V. Reppa and A. Tzes, "Fault detection and diagnosis based on parameter set estimation," *IET control theory & applications*, vol. 5, no. 1, pp. 69–83, 2011.

[12] V. Reppa, M. M. Polycarpou, and C. G. Panayiotou, "Multiple sensor fault detection and isolation for large-scale interconnected nonlinear systems," in *Proc. Eur. Control Conf.*, Zurich, Switzerland, 2013, pp. 1952–1957.

[13] J. Gertler, "Analytical redundancy methods in fault detection and isolation," in *Preprints of IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS'91*, 1991, pp. 9–21.

[14] R. Isermann, "Supervision, fault-detection and fault-diagnosis methods—an introduction," *Control engineering practice*, vol. 5, no. 5, pp. 639–652, 1997.

[15] G. M. Milis, D. G. Eliades, C. G. Panayiotou, and M. M. Polycarpou, "Semantic mediation in smart water networks," in *Computational Intelligence, 2015 IEEE Symposium Series on*, Dec 2015, pp. 617–624.

[16] D. Eliades, C. Panayiotou, and M. Polycarpou, "Contamination event detection in drinking water systems using a real-time learning approach," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, July 2014, pp. 663–670.

[17] J. Hall, A. Zaffiro, R. Marx, P. Kefauver, R. Krishman, R. Haught, and J. Herrmann, "On-line water quality parameters as indicators of distribution system," *Journal of the American Water Works Association*, vol. 99, no. 1, pp. 66–77, 2007.

[18] F. Hua, J. West, R. Barker, and C. Forster, "Modelling of chlorine decay in municipal water supplies," *Water Research*, vol. 33, no. 12, pp. 2735–2746, Aug. 1999.