The Role of Semantic Technologies in Diagnostic and Decision Support for Service Systems

Eleni Tsalapati
Loughborough University
E.Tsalapati@lboro.ac.uk

Thomas W. Jackson
Loughborough University
T.W.Jackson@lboro.ac.uk

William Johnson
Loughborough University
C.W.D.Johnson@lboro.ac.uk

Lisa Jackson
Loughborough University
L.M.Jackson@lboro.ac.uk

Andrey Vasilyev
Loughborough University
A.Vasilyev@lboro.ac.uk

Andrew West
Loughborough University
A.A.West@lboro.ac.uk

Lei Mao
Loughborough University
L.Mao@lboro.ac.uk

Ben Davies
Loughborough University
B.Davies2@lboro.ac.uk

Abstract

In this research, we utilize semantic technology for robust early diagnosis and decision support. We present a light-weight platform that provides the end-user with direct access to the data through an ontology, and enables detection of any forthcoming faults by considering the data only from the reliable sensors. Concurrently, it indicates the actual sources of the detected faults, enabling mitigation action to be taken. Our work is focused on systems that require only real-time data and a restricted part of the historic data, such as fuel cell stack systems. First, we present an upper-level ontology that captures the semantics of such monitored systems and then we present the structure of the platform. Next, we specialize on the fuel cell paradigm and we provide a detailed description of our platform's functionality that can aid future servicing problem reporting applications.

1. Introduction

Numerous industrial applications require real-time continuous monitoring of their performance and early diagnosis of any forthcoming failures. Most of the current diagnostic tools are limited to providing warning of impending failures without any explanation, thus preventing any specific mitigating action. At the same time, few diagnostic tools take into consideration the reliability of the sensors being used. In this research we suggest that the scalability and the integrating nature of semantic web technologies can facilitate these tasks.

We exploit the Ontology Based Data Access (OBDA) [1] technology for robust early diagnosis and decision support. OBDA enables the end-user direct access to the data through an ontology, which is a comprehensible semantic layer that constitutes a formal specification of the domain of interest. Roughly, ontologies constitute a formal representation of entities and of relationships between them that is both machine and human readable. This way, they can be processed by ontology reasoners. A great number of efficient OBDA reasoners are now available for this purpose; for example, PAGOdA [2], Ontop [3], and Hydrowl [4], but there are many more.

We propose the System Monitoring lightweight platform based on the OBDA technology, which focuses on providing diagnostic mechanisms for unavoidable failures and providing alerts about any forthcoming failures and suggestions appropriate mitigation actions. Our approach has two main benefits; firstly, it provides the end-user with direct access to the data through the ontology; secondly, it enables detection of any forthcoming faults by considering only the data of the reliable sensors. At the same time, the indication of the actual sources of the detected faults enables the suggestion for a respective mitigation action. Our work is focused on systems operating under static conditions where their diagnosis requires only real-time data and a restricted part of the historical data.

URI: http://hdl.handle.net/10125/50083
ISBN: 978-0-9981331-1-9

The proposed platform is applied in the Proton Exchange Membrane (PEM) fuel cell paradigm. PEM fuel cells provide an electrochemical source of virtually zero-emission energy conversion and power generation [5]. The cell design can be adapted to suit a diverse range of devices, either individually or combined in fuel cell stacks, to generate power for vehicles, portable and stationary units. The commercial success of fuel cell technology is largely dependent on establishing its durability and reliability. A drawback to most of the current fuel cell diagnostic tools is that their functionality lacks any identification mechanisms of the causes underpinning the occurring failures. Data-driven approaches (e.g. [6], [7], [8]) can detect the faults, but they may not further isolate the faults unless enough test data obtained from various faults is available, while model-based methods (e.g. [9], [10], [11], [12]) require the development of the accurate model incorporating different fault effects with mathematical equations, which is usually extremely complex and time-consuming.

After a brief introduction on the semantic technologies in Section 2, we describe in Section 3 the novel upper-level ontology System Monitoring ontology, which captures the basic knowledge related to system monitoring. In Section 4 we present the functionality of the System Monitoring platform. In Section 5 we focus on the PEM fuel cell paradigm. In particular, in Section 5.1 we present a brief introduction to the PEM fuel cell technology, in Section 5.2 the novel domain ontology Fuel Cell System Monitoring ontology is described. In Section 5.2 the functionality of the System Diagnosis platform once implemented in the fuel cell paradigm is presented. Finally, in Section 6 we present some results of this work and in Section 7 we discuss the conclusions and future work.

2. Preliminaries

The primary component of the semantic technology is the semantic *Knowledge Base* (KB), which comprises two components: the Terminology Box (*TBox*, or simply *ontology*) and the *Assertional Box* (*ABox*). The structural elements of an ontology in the OWL 2 Web Ontology Language [13] are the (atomic) classes, the individuals, the object and datatype properties. The *classes* represent abstract groups, sets, or collections of objects, e.g. the concepts System, Sensor are classes. The *individuals* refer to the real-world concrete objects, e.g. a specific system named $system_1$. The *object properties* relate objects to objects. For instance, given a specific individual, e.g. $system_1$ of the System class and a specific individual $sensor_1$ of the Sensor class, the property monitors($sensor_1$, $system_1$) indicates that the $sensor_1$ monitors the system $system_1$. Finally, the datatype properties assign data to objects, for example the assertion hasValue(Voltage, 23) states that the value of the Voltage is 23.

To describe thoroughly the domain of interest, we can define in the TBox subclass axioms, subproperty axioms or general concept inclusion axioms (GCIs). The components of the subclass axioms are atomic classes. For instance, the "Actuator SubClassOf System" is a subclass axiom that indicates that every individual that belongs to the Actuator class belongs also to the System class. It is important to note that a subclass always inherits the properties of its superclasses. Subproperty axioms are defined in the same manner. GCIs include more complex class expressions, for instance the GCI (expressed in Manchester syntax [14]):

System **and**(hasVoltage **some** LowVoltage)
SubClassOf isInDegradationMode **value** *flooding*

states that if a system has low voltage then it is in flooding mode. Depending on the complexity of the class expressions appearing in the GCI axioms, different profiles of the OWL 2 language are defined. Each profile has different expressiveness and enjoys different computational properties. In this paper we use the lightweight (PTime-complete [15]) OWL 2 EL profile [13].

Depending on the problem, different kinds of ontologies can be developed. *Upper-level* (or *foundation*) ontologies consist of general concepts that are common across different domains. This way, they facilitate the semantic interoperability among the *domain-specific* (or simply *domain*) ontologies, as the elements of the domain ontologies are *specializations* of the generic elements appearing in the upper-level ontology.

Ontology Based Data Access (OBDA) is a key reasoning service of the new generation information systems. In the OBDA paradigm, an ontology defines in a high-level of abstraction the schema of data sources in terms familiar to the domain experts. The data sources are related to the ontology either via *mappings*, which are declarative specifications, similar to view definitions in databases, or via their *RDFization*, i.e. their conversion into ABox assertions. This way, the user can query the KB without an IT's expert intervention. Usually OBDA systems can support *conjunctive queries* (*CQ*). A CQ is an expression of the form $Q(\vec{x}) \leftarrow f(\vec{x}, \vec{y})$, where f is a conjunction of function-free atoms, in which the predicate "Q" does not appear, containing only variables from $\vec{x}$ or from $\vec{y}$. An OBDA system

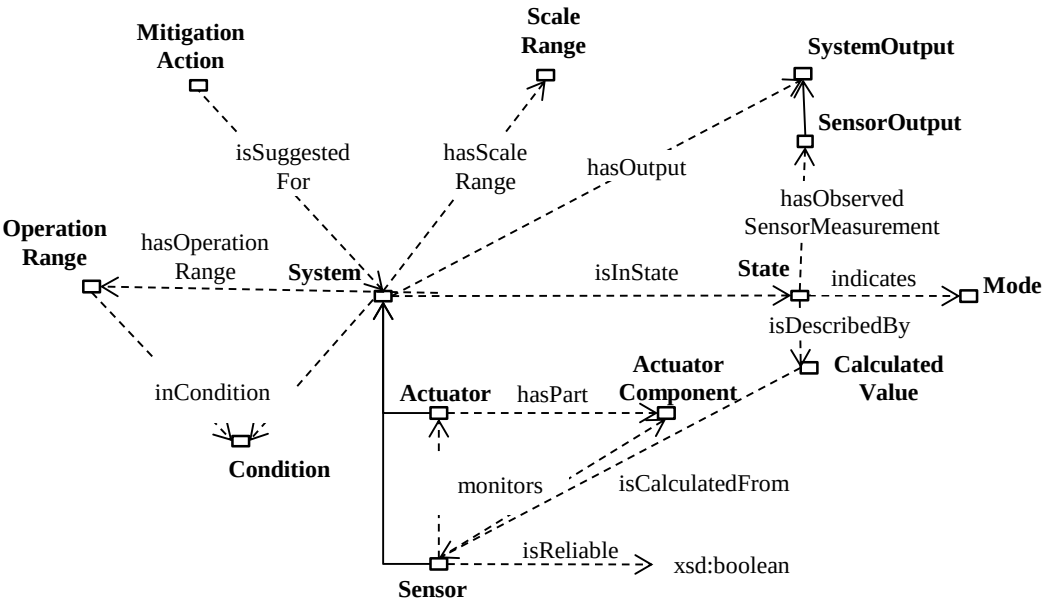


Figure 1 The System Monitoring Ontology

translates the queries and the ontology into the vocabulary of the data sources and then performs the actual query evaluation to a suitable query answering system. In this research, we use Hydrowl, which is one of the most efficient OBDA systems [4] and is partly based on GraphDB formerly known as OWLim [16]. OWLim performs the *materialization* technique, which provided the database and the ontology it computes all the implied assertions that can be inferred.

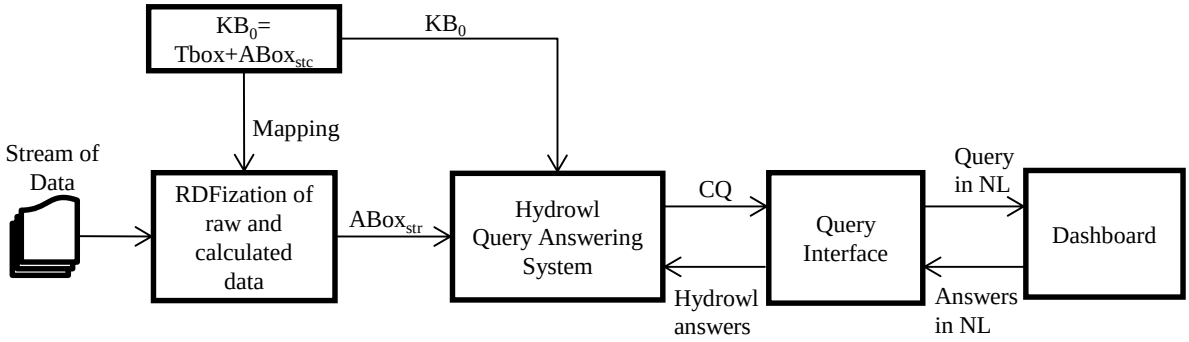
3. The System Monitoring Ontology

The purpose of the system monitoring ontology is to represent in a higher abstract level all the basic knowledge related to system monitoring. Thus, information about the monitored system, its components, the sensors monitoring the system, their outputs and their reliability must be represented in the ontology. Additionally, the ontology must contain terms related to its normal or abnormal operation, along with suggestions for mitigation action.

One of the most popular upper ontologies for the semantic representation of sensors and the information surrounding them is the Semantic Sensor Network (SSN) ontology [17], which is developed by the W3C Semantics Sensor Networks Incubator Group. However, it is rather impractical for the lightweight platform that we propose, as it has high level of expressivity. A more lightweight version of the SSN ontology is the Sensor, Observation, Sample, and Actuator ontology [18], developed by the same group, which however is too simple to capture the knowledge required for our platform. We have developed the System Monitoring (SM) ontology, which is an upper level lightweight ontology that captures the basic features related to system monitoring. The structure of the SM ontology is inspired by the SSN ontology but it is of lower expressivity (OWL 2 EL).

The SM ontology introduces a set of classes centred around the notions of *System* and *State*. We present a graphical representation of the SM ontology in Figure 1. The arrows with solid line represent the SubClassOf (ISA) relationships and the dashed arrows the object property relations between the classes or the datatype properties.

Classes. The core class of the ontology is the *System*, which has as subclasses the *Actuator* class, i.e. a device that performs a procedure that changes the state of the world and the *Sensor* class. All three classes are subclasses of the respective classes of the SSN ontology. A sensor can monitor either an actuator or an individual component of the actuator (*ActuatorComponent*). We suppose that both systems are in use, thus they are dynamic entities and as such can be described by a set of *States*. The instances of the class State are defined by the corresponding system and the time that is being monitored. For instance, the system "$system_1$" is in state "$system_1@t_1$", where "t_1" is a specific time value. This way, a distinction between the various states of a specific system is accomplished, avoiding any inconsistencies. Also, at each state it has a specific output with a result value, which is stored in the class *SystemOutput*. Additionally, from the output values of the system a set



of values, useful for the system diagnosis can be calculated. The class *CalculatedValue* contains these values. It is, also, important to capture the *Conditions* under which a *System* is operating, as these may affect its performance. Examples of such conditions include the weather conditions or the number of times that the system has been in operation. The class *Operation-Range* stores the maximum and minimum values that a system can operate under the specified conditions. This way, if the outputs of the sensors are not within these values a malfunction of the system can be derived. The class *ScaleRange* is used to store the thresholds with which the output values of the system or the calculated values are classified to very-high-high-medium-low to aid the early diagnosis of any forthcoming failure. The class *Mode* stores the different failing modes that the system is at every state. Given the mode a set of *MitigationActions* can be suggested for the particular system.

Figure 2 The System Diagnosis Platform

Object Properties. The object properties that interlink the classes of the SM ontology are presented in Figure 1, except for the property hasSensorOutput, which is subproperty of the property hasOutput, with domain the class Sensor and range SensorOutput. Additionally, for every property R with domain a class C_1 and range a class C_2 a respective property R_0 from C_2 to C_1 is defined. For instance, for the property hasSensorOutput a property isOutputOfSensor is also defined.

The property isCalculatedFrom that interlinks the concepts Sensor and CalculatedValue aids the reliable diagnosis of the system. This is achieved during prognostic process by taking into account only the calculated values that are calculated from reliable sensors.

Datatype Properties. The property *isReliable* that appears in Figure 1 states whether a sensor is reliable or not. As the focus of this work is to construct an overall basic framework for system diagnosis, sensor reliability is defined herein as a Boolean; however further generalization to consider fuzzy variability can also be considered and constitutes future outlook for this platform. Also, the following datatype properties do not appear in Figure 1: The class State is equipped with the datatype property *atTime*. The classes SensorOutput and CalculatedValue have the properties *hasValue*, *hasUnit*, *atTime*, through which the raw or calculated data are stored for each monitored moment. The class OperationRange has the properties *hasUpperValue* and *hasLowerValue* that define the upper and lower values that the system operates under the specified conditions. Finally, the class ScaleRange has four different datatype properties:

- *hasLowThreshold(ScaleRange,xsd:float)*
- *hasMediumThreshold(ScaleRange,xsd:float)*
- *hasHighThreshold(ScaleRange,xsd:float)*
- *hasVeryHighThreshold(ScaleRange,xsd:float)*

which are used to classify the output or the calculated values of the system. It is worth noting this classification process could be automatically performed be defining a set of SWRL rules [19], which enable the comparison of values, and using a reasoning that supports SWRL. However, SWRL rules introduce serious reasoning problems [20], especially when large datasets are included. Additionally, we regard that the threshold values is part of the knowledge that should be evident both for the engineer and the user when required.

ABox Assertions. Given the specification of the monitored system and its sensors, a set of static data related to its healthy operation is stored. In particular, the *ScaleRange* and the *OperationRange* classes are populated by the domain expert. Also, the different modes of the system with the respective mitigation actions can be stored in the ontology as part of the static information. Finally, all sensors are initially defined as reliable.

The SM ontology does not include any general inclusion axioms.

4. The System Diagnosis Platform

In this section we describe the System Diagnosis Platform, which given a system or a set of systems that are being monitored by a set of sensors, the user is being: (i) informed in a predefined frequency about the overall performance of the system, (ii) alerted about any forthcoming failures with their explanations to allow mitigating action, (iii) provided with the explanations of any failures that could not be prevented and (iv) enabled to perform queries.

In Figure 2 we illustrate the architecture of the platform. It takes as input the initial knowledge base KB_0 that consists of the SM ontology specialized in the monitored system enriched with the static data (reliability of sensors, operation ranges, scale ranges) and the real-time stream of sensor data. Then, it performs the following steps:

1. *Window of Data*. For a certain amount of time (predefined by the user) a set stream of data, i.e. a *window of data*, is used to check the reliability of the sensors.
2. *Sensor Reliability Checking.* First, all necessary calculations are performed from the set of raw data that correspond to the predefined timeframe. Next, the results are compared with the respective operating range values which have already been stored in the ontology. If for a sensor these values are violated for a predefined number of times, then the ABox assertion (*isReliable*) that stores its reliability is transformed from "true" to "false". Once a sensor is defined as unreliable then this information does not change. However, if a sensor is defined as reliable then it is continuously checked with respect to its reliability throughout the time of operation of the system.
3. *Knowledge Inference*. The window of data is mapped to the ontology and is transformed to the $ABox_{str}$, i.e. the streaming ABox. Also, for each monitored system $syst_i$ and for each monitored time t_j of the window of data the assertions $State(syst_i@t_j))$ and $isInState(syst_i, syst_i@t_j)$ are added to the $ABox_{str}$. The $ABox_{str}$ with the KB_0 is then loaded to the reasoner from which all implied assertions are inferred.
4. *Alerting*. A set of predefined conjunctive queries related to the healthy performance of the monitored system is automatically performed to the query answering system. If an abnormal behavior is noted, then the user is informed instantaneously about this malfunction and its causes in natural language by an alerting mechanism.
5. *Moving the Window of Data*. The first line of the window of data is replaced with the stream of new real-time data. This way, a moving window of data of fixed size is formed in first-in-first-out manner. The platform proceeds by repeating the steps 2-5 with the new data, until the system is stopped. It is important to note that at step 4 the user is alerted about some abnormal behavior of the system, only if this behavior has changed from the one in the previous state.

Additionally, the user communicates with the knowledge-based system through the dashboard, in which a set of plots over the sensor outputs or the calculated values appears, indicating the real-time performance of the system through time. The user is also provided with a set of (semi-)predefined queries in natural language which are automatically transformed to conjunctive queries and performed to the reasoner. The answers of the reasoner are then converted into natural language.

The size of the moving window of data determines the efficiency of the platform, as the biggest the size of the data the more inferences the reasoner will have to perform.

5. The PEM Fuel Cell Paradigm

Although a significant amount of research has been carried out on fault analysis within PEM fuel cells (e.g. [7]-[12], 0), the many components of a single fuel cell add to the complexity of understanding the root causes of fuel cell failure. Not least because the components are all subject to degradation over their operational lifecycle, sometimes making it harder to spot when expected degradation has spiraled into a fault. This complexity is increased when multiple fuel cells are combined into a stack.

The reliability and resilience of an operational fuel cell stack are key factors in making it a commercial success. However, this requires a number of crucial improvements in the monitoring and diagnostic capabilities. In order to achieve this there needs to be an informed method of dealing with the vast amount of raw data that is produced by the sensors monitoring the cells, which can be enabled by the semantic technology techniques.

After a short introduction to the PEM fuel cell technology, we describe the PEM fuel cell domain ontology which is mapped to the SM ontology and then we present the functionality of the platform for fuel cell diagnosis.

5.1. The PEM Fuel Cell

The core part of a PEM fuel cell consists of an ion conduction membrane, the electrolyte, bonded on each side by a porous electrode catalyst. These thin membrane electrode assemblies are usually bonded

together serially into a fuel cell stack [5]. A gas diffusion layer enables the reactants to pass across and interact with the electrodes. The stack can be seen as a distinct unit, with an outer anode electrode at one end and a cathode electrode at the other. The hydrogen fuel is fed into the anode end of the stack and the oxidization process produces electrons and protons. The former produces the output current of the cell, while the latter passes through the electrolyte membrane to the cathode. The protons and electrons combine with the oxygen through the electrolyte mem-

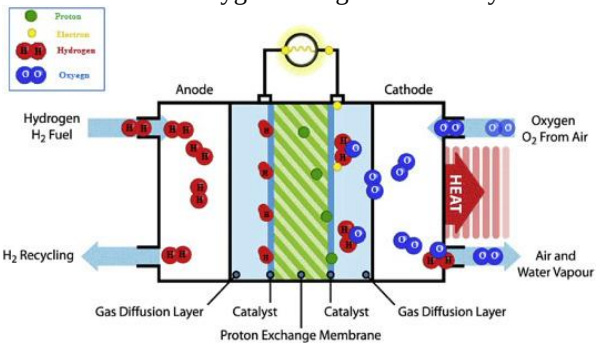


Figure 3 The PEM Fuel Cell [22]

brane to the cathode. The protons and electrons combine with the oxygen passed into the cathode to produce water and heat.

Each of the components of the fuel cells in a stack has an expected level of degradation within a given operational mode. But this degradation can be accelerated and critical faults occur. In addition to this, there is often very little time between, for example, a cell output voltage plummeting and the cell catastrophically failing. A stack can carry a certain amount of individual cell underperformance, but there is inevitably a trigger point. The aim is to recognize a failing stack before it fails, and to then take any remedial action or to close it down.

Table 1 Diagnostic rules base for PEM fuel cell water management issues [6]

IF	THEN
Stack temperature is cold OR Cathode humidity is high	Flooding is certain AND Dehydration is none
Stack temperature is normal OR Cathode humidity is normal	Flooding is null AND Dehydration is evidenced
Stack temperature is hot OR Cathode humidity is low	Flooding is null AND Dehydration is certain
Stack voltage is normal OR Stack voltage is high	Flooding is null
Stack voltage is low	Flooding is evidenced

A fuel cell may degrade due to several reasons: membrane chemical breakdown, catalyst dissolution, carbon support corrosion, flooding and dehydration to name only few. Davies et al [6] have formed a knowledge rules base of the form IF-THEN statements, which we exploit in the next section for the formation of the ontology. Due to space limitations we present in Table 1 only the rules related to water management, as it is one of the crucial mechanisms for the successful running and health of a PEM fuel cell [19[23]. Dehydration is usually associated with the anode electrode and can lead to a drop in output current and in severe cases mechanical breakdown or a reduction of the cell life expectancy; but it can be remedied by humidifying the cell. Fuel cell flooding is associated with the cathode where excess water can block the gas diffusion layer and lead to gas starvation.

In Table 1, a classification of both the sensor measurements and the degradation modes is presented. The classification of the sensor measurements is based on the operation of the sensors under normal conditions. The classification of the degradation modes expresses the level of agreement between the measured operating conditions and those necessary for a certain degradation. Hence, there is "null" dehydration (flooding) when the level of agreement is up to 15%. There is "evidenced" dehydration (flooding) when the level of agreement is up to 50% and "certain" when it is up to 90%.

5.2 The PEM Fuel Cell Ontology

In this section we describe the OWL 2 EL Fuel Cell System Monitoring (FCSM) ontology which constitutes a specialization of the SM ontology. To ensure the efficiency of the platform, we focus only on the representation of the basic components of a fuel cell system and of the diagnostic rules suggested by Davies et al [6].

In all, the FCSM ontology contains 75 classes, 20 object and 10 datatype properties. Also, it contains 127 subclass axioms and 15 GCIs. Due to space limitations we present only the elements relevant to water management issues. In Figure 4 we present this part of the FCSM ontology. The SM ontology is colored with grey and the specialized part of the FCSM which is

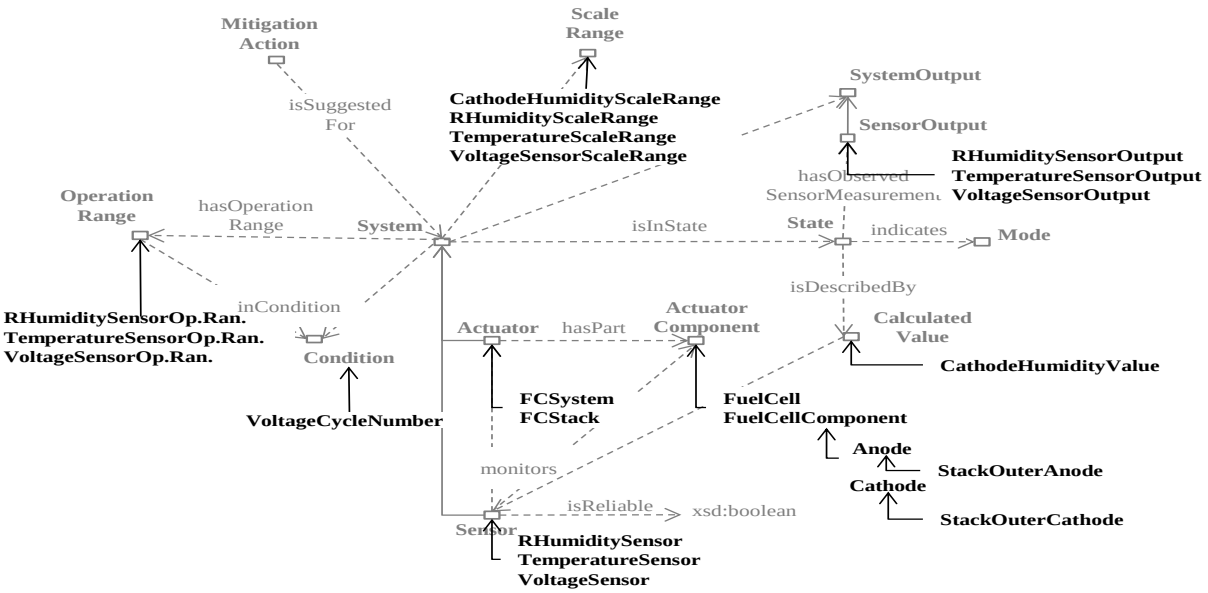


Figure 4 The Fuel Cell System Monitoring Ontology

related to the fuel cell technology is colored with black.

As it is shown in Figure 4, a fuel cell stack and a fuel cell system are regarded as Actuators and they consist of fuel cells, which have fuel cell components. The fuel cell components that are relevant to the water management issues are the outer anode and the outer cathode of the stack. Also, only the relevant humidity, temperature, and voltage sensors located to different parts (anode, cathode, etc.) of the fuel cell stack suffice to deduce if there is flooding or dehydration. Hence, these sensors have outputs (e.g. RHumiditySensorOutput), operating ranges, and scale ranges. Additionally, from the outputs of these sensors the value of the humidity of the outer cathode can be calculated, which also has a scale range. Additionally, following the Davies et al. method, we have classified both the sensor outputs and the calculated values to low-medium-high. For instance, the class TemperatureSensorOutput is superclass of the classes:

- LowTemperatureSensorOutput,
- Normal TemperatureSensorOutput
- HighTemperatureSensorOutput.

The several degradation modes have been incorporated to the class Mode. In particular, the class Mode contains the individuals: flooding, dehydration, evidenced flooding and evidenced dehydration.

The SM ontology also has been enriched by subclass axioms and GCIs. In order to take into account only the reliable sensors we have connected the sensor outputs and the calculated values with the respective sensors. For instance, the axiom:

HumidityValue **SubClassOf**
(isCalculatedFrom some RelativeHumiditySensor) and
(isCalculatedFrom some TemperatureSensor)

states that for the calculation of humidity value the relative humidity sensor and the temperature sensor were used. We have transformed the Table 1 rules to GCIs, by taking also under consideration the reliability of the sensors.

The first rule is decomposed to the following two axioms:

hasObservedSensorMeasurement **some**
(LowTemperatureSensorOutput **and**
(isOutputOfSensor **some** (TemperatureSensor **and**
(isReliable **value** true))))
SubClassOf indicates **value** *flooding*

isDescribedByCalculatedValue **some**
(HighHumidityValue **and**
(isCalculatedFrom **some**
(RelativeHumiditySensor **and**
(monitors **some** StacksOuterCathode) **and**
(isReliable **value** true)))
and
(isCalculatedFrom **some**
(TemperatureSensor **and**
(monitors some StacksOuterCathode) **and**
(isReliable **value** true))))
SubClassOf indicates **value** *flooding*

According to the first axiom, if a state has observed low temperature and the sensor that monitors the

temperature is reliable then the state indicates that there is flooding. According to the second axiom, if a state is described by high humidity, which is calculated from a reliable relative humidity sensor and a reliable temperature sensor both of them placed in the outer cathode of the stack, then this state indicates that there is flooding. The rest of the rules can be expressed in a similar way.

5.3. PEM Fuel Cell System Diagnosis with the System Diagnosis Platform

In this section we describe the functionality of the system diagnosis platform when it is used in the fuel cell paradigm.

The input KB_0 consists of the FCSM ontology and the static $ABox_{stc}$. The $ABox_{stc}$ is populated by assertions on the specific fuel cell systems (e.g. FuelCellSystem(s_1)), their components (e.g. FuelCellStack(stc_1), FuelCell(fc_1), FuelCell(fc_2)) the sensors monitoring them (e.g. TempSensor($ts_{airInlet}$), TempSensor(ts_{stack})) and the relations among them (e.g. monitors(ts_{stack}, st_1)). Also, it contains the low and upper values of the operating values of the systems and the thresholds for the classification of the output values of the sensors or the calculated values as they are defined by the specifications of the particular fuel cell systems.

Then, provided with the real-time data from the sensors the platform performs the steps 1-5 as described in Section 4. It is important to highlight that at step 1 besides the calculations performed (e.g. cathode humidity value), the sensor output and the calculated values are classified accordingly. In step 4, initially the platform identifies the stacks that are in some degradation mode by performing the query:

$$Q(x,y,z) \leftarrow \text{FuellCellStack}(x) \textbf{ and } \text{isInState}(x,y) \textbf{ and } \text{indicates}(y,z)$$

with which the kind of the failure of each stack stc_i at the state $stc_k@t_m$ will be returned. Then, for each degrading system according to the nature of the failure, e.g. flooding, evidenced flooding, etc., a set of different queries will be performed. Supposing that the first query returns that the fuel cell stack stc_k is at time t_m in the state $stc_k@t_m$ that indicates flooding, then the following queries will be performed:

$$Q(v,u) \leftarrow \text{hasObservedSensorMeasurement}(stc_k@t_m,x) \textbf{ and } (\text{LowTemperatureSensorOutput}(x) \textbf{ and } (\text{isOutputOfSensor}(x,y) \textbf{ and } (\text{TemperatureSensor}(y) \textbf{ and } \text{isReliable}(y, \text{true})))) \textbf{ and } \text{hasValue}(x,v) \textbf{ and } \text{hasUnit}(x,u)$$

$$Q(v,u) \leftarrow \text{isDescribedBy}(stc_k@t_m,x) \textbf{ and } \text{HighCathodeHumidityValue}(x) \textbf{ and } (\text{isCalculatedFrom}(x, y) \textbf{ and } (\text{RelativeHumiditySensor}(y) \textbf{ and } (\text{isReliable}(y, \text{true}))) \textbf{ and } (\text{isCalculatedFrom}(x, z) \textbf{ and } (\text{TemperatureSensor}(z) \textbf{ and } (\text{isReliable}(z, \text{true}))) \textbf{ and } \text{hasValue}(x,v) \textbf{ and } \text{hasUnit}(x,u)$$

The first query will check if the flooding is due to low temperature and the second if it is due to high humidity, by taking into account the reliability of the sensors. In every case the system will return the values (v) and their units (u) of the parameters responsible for the failure mode. In this way, apart from indicating the type of the degradation mode, the system provides also the explanations for this condition.

Finally, the user is also provided with a set of (semi-) predefined queries in natural language which are automatically translated to conjunctive queries and performed to the reasoner. Then, answers of the reasoner are transformed in natural language form. For example:

Q1: What was the degradation mode of the stack stc_1 *at the* 13^{th} *minute?*
A1: stc_1 was flooded
Q2: Why was stc_1 *flooded at the* 13^{th} *minute?*
A2: Due to very low temperature (19 C)

It is important to note though that the user can have access only to the part of the historic data that is within the moving window of data.

6. Evaluation

In this section we present a preliminary evaluation of System Diagnosis platform on the fuel cell paradigm. The evaluation is based on an early deployment of the platform. The evaluation was conducted on an Intel(R) Core (TM) PC running Windows 7 with a 3.20GHz processor and 8GB of RAM.

For the experimental evaluation we used a fuel cell system that contains only one fuel cell stack with two fuel cells. The experiment was set up to result in the flooding of the fuel cell stack. Figure 5 shows the progression of the current (in Ampere) through time.

The current version of the platform accepts in the input only the real-time data and not a window of data. Additionally, it does not check the sensors with respect to reliability, however by the results of the relative humidity sensors it was evident that they were unreliable (in many cases the sensors' outputs

exceeded the 120%). Hence, only the rules independent from humidity were used for the diagnosis of the fuel cell system.

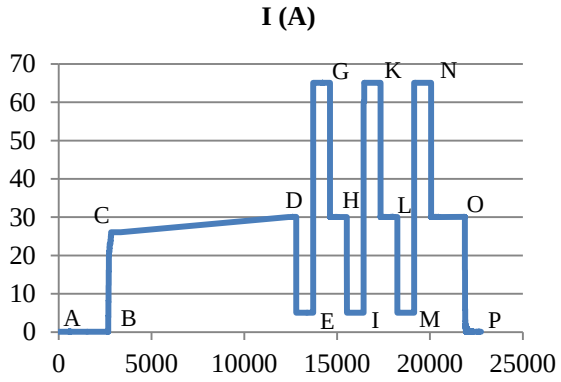


Figure 5 Current progression through time (sec)

As at each moment only a stream of real-time data was introduced to the platform the diagnosis was instantaneous. At the start-up (A-B segment) the non-verbose mode platform displayed the following:

ALERT! fuelCellStack1 is in [evidencedFlooding, flooding] mode.
-ALERT! Low Voltage: 0.0V
-ALERT! Low Temperature: 17.65C

While the verbose mode outputted the following:

Is some stack in some degradation mode?
- Yes, fuelCellStack1: [evidencedFlooding, flooding]
Is evidenced flooding due to voltage < 0.9V ?
- Yes, fuelCellStack1 0.0 V
Is flooding due to temperature < 20.0C ?
- Yes, fuelCellStack1 17.65 C

At the segments C-D, D-E, G-I, K-M, N-O there was no alert. At the segments E-G, I-K, M-N the non-verbose mode of the platform displayed:

ALERT! fuelCellStack1 is in [evidencedFlooding] mode.
-ALERT! Low Voltage: 0.7V

and the verbose mode displayed:

Is some stack in some degradation mode?
- Yes, fuelCellStack1: [evidencedFlooding]
Is evidenced flooding due to voltage < 0.9V ?
- Yes, fuelCellStack1 0.7 V

It is worth noting that at the initial stages of the segment I-K there were some values just above the threshold of 0.9V so the low voltage alert was not triggered consistently until after the initial phase transition. This could be fixed if a part of the historic data was also used for diagnosis. Finally, a special mechanism is required to identify the start-up process in order to avoiding triggering any alarms during this phase.

7. Conclusion-Future Work

In this research we proposed an ontology based platform for early diagnosis for monitored systems. The main objective of this work is to provide a user-friendly environment that will enable the identification of the trigger points that herald potential problems, deterioration and breakdown. Through this approach the system detects the fault and it also classifies it into a cause.

Within the proposed framework we presented the System Diagnosis platform and we introduced two novel ontologies; the System Monitoring ontology and the Fuel Cell System Monitoring ontology. Our approach also takes into consideration the reliability of the sensors. We validated an early deployment of the platform by applying it to the fuel cell paradigm with real raw data.

Although the proposed research framework is at a preliminary stage, the research findings have indicated the potential benefits of semantic technologies in their application to service analytics in the diagnostic processes within system monitoring domain. The semantic approach detailed in this paper has provided an alternative method, using a lightweight approach to improving the interpretation and understanding of basic service analytics by providing a semantic interpretation to the end user which will enable better mitigation intervention.

As a next step we are working towards performing fuzzy reasoning techniques to better reflect sensor reliability, thus resulting in more accurate and realistic system diagnosis. Finally, we are also interested in the new challenges introduced after the evaluation of the preliminary deployment of the proposed system. As it was observed, apart from the real-time data, their total rate of change should also be taken under consideration.

10. References

[1] Dolby, J., Fokoue, A., Kalyanpur, A., Ma L., Schonberg, E., Srinivas, K., and Sun, X.: Scalable grounded conjunctive query evaluation over large and expressive knowledge bases. In Proc. of the 7th Int. Semantic Web Conf. (ISWC 2008), 5318, pp. 403-418, Springer (2008).
[2] Zhou, Y., Cuenca, G. B. , Nenov, Y. , Kaminski, M., Horrocks, I.: PAGOdA: Pay-As-You-Go Ontology Query Answering Using a Datalog Reasoner. Journal of Artificial Intelligence Research 54(1), pp. 309-367, AI Access Foundation (2015).
[3] Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web 8(3), pp. 471-487, IOS Press (2017).
[4] Stoilos, G., Stamou, B. G.: Hybrid Query Answering Over DL Ontologies. Inf. Proc. of the 27th Int. Workshop on Description Logics, Vienna, Austria, July 17-20, 2014, pp. 336-339, CEUR-WS.org (2014).
[5] Larminie, J., Dicks, A., McDonald, M. S.: Fuel cell systems explained. Second Edition, John Wiley Sons, Chichester (2000).
[6] Davies, B., Jackson, L. M., Dunnett, S.J.: Development of a Fuzzy Diagnostic Model for Polymer Electrolyte. In Proc. of the 25th European Safety and Reliability Conference (ESREL 2015), pp. 2373-2378, CRC Press (2015).
[7] Li, Z., Outbib, R., Giurgea, S., Hissel, D., .Diagnosis for PEMFC systems: a data-driven approach with the capabilities of online adaptation and novel fault detection. Ind Electron IEEE Trans, 62 (8) , pp. 5164-5174 (2015).
[8] Gao, Z.; Cecati, C.; Ding, S.X. A survey of Fault diagnosis and fault-tolerant techniques-part II: Fault diagnosis with knowledge-based and hybrid/active approaches. IEEE Trans. Ind. Electron., 62, pp. 3768–3774 (2015).
[9] Mao, L., Jackson, L., Dunnett, S.: Fault Diagnosis of Practical Polymer Electrolyte Membrane (PEM) Fuel Cell System with Data driven Approaches. Fuel Cells, WILEY-VCH Verlag (2016).
[10] Isermann, R. Model-based fault-detection and diagnosis—Status and applications. Annu. Rev. Control, Vol. 29, pp. 71–85 (2005).
[11] Venkatasubramanian, V.; Rengaswamy, R.; Yin, K.; Kavuri, S.N. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Comput. Chem. Eng., 27, pp. 293–311 (2003).
[12] Damian-Ascencio, C. E., Saldaña-Robles A., Hernandez-Guerrero, A., Cano-Andrade, S. Numerical modeling of a proton exchange membrane fuel cell with tree-like flow field channels based on an entropy generation analysis, Energy, pp. 306-316 (2017).
[13] Hitzler, P., Krotzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds): OWL 2 Web Ontology Language: Primer. W3C Recommedation (2009).
https://www.w3.org/TR/owl-primer/
[14] Horridge, M., Nick Drummond, N., Goodwin, J., Rector, A. L., Stevens, R., Wang, H.: The Manchester OWL Syntax. In Proc. of the OWLED 06 Workshop on OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11, CEUR-WS.org (2006).
[15] Baader. F., Brandt, S., Lutz, C. Pushing the EL Envelope. In Proc. of the 19th Joint Int. Conf. on Artificial Intelligence (IJCAI, 2005).
[16] Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., & Velkov, R. OWLIM: A family of scalable semantic repositories. Semantic Web, 2(1), 33-42 (2011).
[17] Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., & Cox, S., et al. The SSN ontology of the W3C semantic sensor network incubator group. In Web Semantics: Science, services and agents on the World Wide Web, 17, pp. 25–32 (2012).
[18] Atkinson, R., Castro, R, G., Liebermann, J., Stadler, C. Semantic Sensor Network Ontology, W3C Working Draft (04 May 2017). https://www.w3.org/TR/vocab-ssn/
[19] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosofand, M. Dean, SWRL: A semantic web rule language combining OWL and RuleML, W3C Member Submission (May 2004).
URL http://www.w3.org/Submission/SWRL/
[20] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, Hermit: An OWL 2 reasoner, Journal of Automated Reasoning 53 (3) (2014) 245-269.
[21] Wang, H., Li, H., Yuan, X. Z. (Eds.): PEM Fuel Cell Failure Mode Analysis. CRC Press (2011).
[22] Fărcaş, A. C., Dobra, P.. Adaptive control of membrane conductivity of PEM fuel cell J Procedia Technol, 12, pp. 42-49 (2014).
[23] Ji, M., Wei, Z.: A review of water management in polymer electrolyte membrane fuel cells. Energies, 2(4), pp. 1057-1106 (2009).
[24] Al-Baghdadi, M. A. S.: A CFD model for analysis of performance, water and thermal distribution, and mechanical related failure in PEM fuel cells. Journal of Mechatronics, Electrical Power, and Vehicular Technology, 7(1), pp. 7-20 (2016).