# A Semantic Framework for Condition Monitoring in Industry 4.0 based on Evolving Knowledge Bases

Franco Giustozzi [a,*], Julien Saunier [a] and Cecilia Zanni-Merk [a]
[a] *Normandie Université, INSA Rouen, LITIS, Rouen 76000, France*
*E-mails: franco.giustozzi@insa-rouen.fr, julien.saunier@insa-rouen.fr, cecilia.zanni-merk@insa-rouen.fr*

**Abstract.** In Industry 4.0, factory assets and machines are equipped with sensors that collect data for effective condition monitoring. This is a difficult task since it requires the integration and processing of heterogeneous data from different sources, with different temporal resolutions and underlying meanings. Ontologies have emerged as a pertinent method to deal with data integration and to represent manufacturing knowledge in a machine-interpretable way through the construction of semantic models. Ontologies are used to structure knowledge in knowledge bases, which also contain instances and information about these data. Thus, a knowledge base provides a sort of virtual representation of the different elements involved in a manufacturing process. Moreover, the monitoring of industrial processes depends on the dynamic context of their execution. Under these circumstances, the semantic model must provide a way to represent this evolution in order to represent in which situation(s) a resource is in during the execution of its tasks to support decision making.

This paper proposes a semantic framework to address the evolution of knowledge bases for condition monitoring in Industry 4.0. To this end, firstly we propose a semantic model (the COInd4 ontology) for the manufacturing domain that represents the resources and processes that are part of a factory, with special emphasis on the context of these resources and processes. Relevant situations that combine sensor observations with domain knowledge are also represented in the model. Secondly, an approach that uses stream reasoning to detect these situations that lead to potential failures is introduced. This approach enriches data collected from sensors with contextual information using the proposed semantic model. The use of stream reasoning facilitates the integration of data from different data sources, different temporal resolutions as well as the processing of these data in real time. This allows to derive high-level situations from lower-level context and sensor information. Detecting situations can trigger actions to adapt the process behavior, and in turn, this change in behavior can lead to the generation of new contexts leading to new situations. These situations can have different levels of severity, and can be nested in different ways. Dealing with the rich relations among situations requires an efficient approach to organize them. Therefore, we propose a method to build a lattice, ordering those situations depending on the constraints they rely on. This lattice represents a road-map of all the situations that can be reached from a given one, normal or abnormal. This helps in decision support, by allowing the identification of the actions that can be taken to correct the abnormality avoiding in this way the interruption of the manufacturing processes. Finally, an industrial application scenario for the proposed approach is described.

Keywords: Semantic Technologies, Ontology, Context Modeling, Stream Reasoning, Condition Monitoring, Industry 4.0

## 1. Introduction

Industry 4.0 aims to improve the production and associated services through the digitalization and automation of manufacturing processes [1, 2]. To ensure high productivity, availability, and efficiency of manufacturing processes, the detection of abnormal conditions of production lines is a crucial issue for manufacturers [3]. In order to tackle this issue, factories rely on condition monitoring. This is the task of monitoring all the equipment involved in a man-

*Corresponding author. E-mail: franco.giustozzi@insa-rouen.fr.

ufacturing process for early detection of abnormal behaviors or situations that could lead to anomalies, which affect its performance, energy-use or reliability. Through the early detection of these situations, proactive decisions can be made to avoid production downtime. These decisions can imply, for example, changing the process parameters to adapt the processes behavior.

An important characteristic for industrial production in Industry 4.0 is that physical items such as sensors, actuators, machines and company assets are connected to each other and to the Internet. More particularly, most modern machines have a large number of sensors installed by the Original Equipment Manufacturer (OEM), as well as additional sensors added by the manufacturing train operator. In this environment, devices and sensors generate an increasing amount of data. Such data is available to operators and engineers through SCADA (Supervisory Control and Data Acquisition), OSI-Pi[4] and other interfaces, and can be used for effective condition monitoring of equipment. In order to detect abnormal situations, the data collected by the sensors must be interpreted. This is a challenging task as it requires the integration and processing of heterogeneous data coming from different sources, with different temporal resolutions and different underlying meanings. Moreover, a key point to consider is that the monitoring of industrial processes should depend not only on their internal state and on user interactions but also on the context of their execution. Making industrial processes context-aware allows to provide added-value information to improve their performance. Context is any information that can be used to characterize the situation of an entity [5]. It is usually considered as a mixture of geospatial data, environmental sensor inputs, service descriptions, among others. In manufacturing, a dynamic context model should not only take into account the context of tools, machines, parts and products, but also the planning of the manufacturing processes, the specification of resources and the control system configuration. Context data is subject to constant change and can be highly heterogeneous.

Semantic Web technologies have proved their efficiency to deal with this data integration issue. The Semantic Web is an extension of the World Wide Web that combines knowledge engineering and AI methods to represent, integrate, and reason upon data and knowledge through ontologies and rules. In computer science, an ontology is considered as *"an explicit specification of a conceptualization for a domain of interest"* [6]. Ontologies emerge as a pertinent method to represent knowledge of any kind (in particular, manufacturing knowledge) in a machine-interpretable way. An ontology describes knowledge at a schema level, *i.e.*, in terms of conceptual taxonomies and general statements, whereas if we think more in terms of data, we talk about a knowledge base that contains instance information about particular scenarios [7]. More formally, a knowledge base is constituted by two components called TBox and ABox. The TBox stores a set of universally quantified assertions, inclusion assertions that declare general properties of concepts and roles. The ABox comprises assertions about individual objects instance assertions. Ontologies are used to structure knowledge in knowledge bases [7]. In this way, in the manufacturing domain it provides a virtual representation of the resources of a manufacturing plant as well as of the relevant situations associated to them. From this point of view, an ontology is a piece of knowledge that can be used by a knowledge-based application, for example, to perform reasoning on it at both schema-level and instance-level information. Furthermore, reasoning on ontologies through its rule-based extensions allows transforming raw observations collected by sensors into higher-level abstractions, such as situations of interest, that are meaningful for humans and provide a better understanding about the physical world to support decision making tasks. For example, observations from sensors can be used to optimize power consumption in a production line to avoid failures.

Current solutions for reasoning on ontologies were traditionally developed for static or slow changing data. The highly dynamic nature of data in the industrial domain introduces new issues. To tackle these, a number of recent works propose to unify reasoning and stream processing, giving rise to the research field of stream reasoning. Stream reasoning supports decision systems based on the continuous processing of data streams together with rich background knowledge [8]. It can find application in several domains, for example, in [9] the authors introduce a stream reasoning system for monitoring vessel activity in geographical areas in order to detect different types of dangerous, suspicious and potentially illegal vessel activity. Moreover, it can also be found in applications from smart cities to e-health, from the Internet of Things to social network analytics [10, 11].

Under these circumstances, the knowledge base itself must evolve in order to represent in which situation(s) the resource(s) is during the execution of its tasks to support decision-making. The knowledge base can evolve due to different reasons. The first possibility is related to a change in the structure of the model itself, *i.e.* addition/removal of concepts and relations. This type of change is studied in the field of ontology evolution [12–15] and is not addressed in this paper. The second possibility is related to the addition of instances to concepts already defined

and to the addition of relations over existing instances. One example can be the addition of a physical resource to the factory that would be reflected as a new instance of the corresponding resource concept in the semantic model. Another example can be the addition of a detected situation and the fact that a certain resource is involved in that situation (meaning a new link between two instances through an existing relation).

Situations of interest in the industrial domain depend on sensor data and domain knowledge. Besides the static information about industrial entities, dynamic information, such as the way the industrial processes are executed, has to be described. Therefore, the data collected from the sensors deployed in a factory needs to be explicitly represented to describe the process state. This is the first requirement for the semantic model to be build.

Another requirement is the representation of temporal relations among processes and of spatial relations between entities and locations in the factory. These relations are used to identify different situations of interest. This implies handling information that evolves in time, such as the changes of situation(s) a machine can go through or the changes in the values of a machine parameter according to the different decisions made. Consequently, the semantic model must provide enough expressiveness to capture high level knowledge, such as situations, from time annotated data coming from sensors. More specifically, the requirements that the semantic model must meet are:

- **Data Integration.** The integration of information that comes from different sources is a key requirement. Moreover, the integration of data coming from sensors (data streams) with background knowledge that describes the application domain is also required (*e.g.*, the streaming data collected from a machine is combined with background knowledge about different constraints that indicate abnormal behavior).
- **Time representation.** Time plays a central role in condition monitoring tasks. This demands for a suitable representation of time, where data can be annotated with their time of occurrence and validity.
- **Efficiency.** Regarding situation detection, processing efficiency is related to the ability to cope with large amount of data being collected to timely generate new results (e.g. abnormal situations need to be promptly detected for enabling counter actions).
- **High level decision support.** Once a situation that may lead to failures is detected, an action needs to be undertaken to counter-act it. Actions change the state of the system and lead to other situations. In order to help in decision making and choose the most appropriate action, it is necessary to understand the relationships among situations.

*Main contributions of this article*

These requirements motivate the proposal of a novel semantic framework to address the evolution of knowledge base in Industry 4.0. The proposed framework uses semantic technologies to represent the manufacturing domain with special emphasis on modeling the context of the resources involved in a factory. The proposed framework aims at detecting situations that can lead manufacturing processes in Industry 4.0 to failures and their possible causes. Through the detection of these situations and their causes, appropriate decisions can be made to avoid the interruption of manufacturing processes.

The first contribution is related to the development of the knowledge base; and the other ones are related to the management of the evolution of the knowledge base:

- An ontological knowledge model for the manufacturing domain, COInd4, is proposed to provide a declarative, abstract representation of the resources, processes, sensors and the relationships among them. Furthermore, the model is strongly oriented towards modelling the context of the resources involved in a manufacturing process. Relevant situations that combine sensor observations with domain knowledge are also represented in it.
- An approach that uses stream reasoning to detect relevant situations that lead to potential failures is introduced. This approach enriches data collected from sensors with contextual information using the proposed semantic model. The use of stream reasoning facilitates the integration of data from different data sources, different temporal resolutions as well as the processing of these data in real time. This allows to identify situations from low-level context and sensor data streams combined with background knowledge. Furthermore, our proposal uses classical reasoning approaches for determining the possible causes that generate the situation when stream reasoning results are not sufficient.

– Detected situations can trigger actions to adapt the process behavior, and in turn, this change in behavior can lead to the generation of new contexts leading to new situations. To organize the situations based on the constraints they rely on, the building of a lattice is proposed. This lattice represents a road-map of all the situations that can be reached from a given one, normal or abnormal. This helps in decision support, by allowing the identification of the actions that can be taken to correct the process behavior.

The rest of this paper is structured as follows. Section 2 provides a review of existing ontology-based models and systems developed for condition monitoring. Section 3 presents a framework to deal with the evolution of knowledge bases for condition monitoring in Industry 4.0. The approach includes the use of an ontological model for the representation of the entities involved in a manufacturing process, emphasizing the modeling of the context. Each of the components of the proposed framework are detailed as well as the interactions among them. In Section 4 an industrial application scenario for the proposed approach is studied. By means of it, we verify that (i) the proposed semantic model is generic and extensible to accommodate a wide spectrum of manufacturing processes, and that its modular architecture allows the description of manufacturing resources with different levels of modularity; (ii) the use of stream reasoning together with classical reasoning approaches allows the detection of abnormal situations as well as of their possible causes in a suitable way; and (iii) the exploitation of the lattice helps in decision making, by allowing the identification of the actions that can be taken to correct the abnormal behavior of the process. Finally, Section 5 gives some concluding remarks as well as some perspectives for future work.

## 2. Related Work

Monitoring is a crucial activity in industry. Any unexpected machine failure can degrade or even interrupt the manufacturing processes of a company. Therefore, it is fundamental to develop a well-implemented and efficient monitoring strategy to prevent unplanned stoppages of production, improve reliability and reduce operating costs. The use of the technologies associated to Industry 4.0 such as Artificial Intelligence (AI), Robotics, the Internet of Things (IoT) and Cyber Physical Systems (CPS), makes condition monitoring more efficient and flexible [16, 17]. It employs advanced and online analysis of collected data for the early detection of the occurrence of possible machine failures, and supports technicians during the maintenance interventions by providing decision support. However, data from sensors attached to the machines alone are often not sufficient for condition monitoring; metadata about the asset, its operating environment and external variables that influence the asset's degradation would also be needed [18]. In other words, changes in the asset's environmental variables must be taken into account in order to do condition-monitoring effectively. Over the last decades, a considerable amount of research efforts has been undertaken to address the development of different models for industrial condition monitoring, using the key technologies associated to Industry 4.0.

In this section, we first review approaches related for fault detection and diagnosis in condition monitoring systems. These approaches can be categorized into data-driven and knowledge-based approaches. For the latter, the most relevant ontologies for modeling the manufacturing domain are also reviewed. We analyse them and their rule-based extensions that are relevant to condition monitoring highlighting their advantages as well as their drawbacks according to the requirements presented in the introduction of this paper: data integration, temporal representation, efficiency and high-level decision support.

### *2.1. Data-driven approaches to condition monitoring*

With the development of big data related techniques and the ever-increasing availability of data, data-driven predictive maintenance is becoming more and more attractive. To extract useful knowledge and help to make appropriate decisions from huge amount of data, machine learning and deep learning techniques have been regarded as powerful solutions. The models obtained by applying those techniques perform like a black box that learn the behavior of physical assets directly from their operation data [19]. They have the ability for feature learning, fault classification and fault prediction. Within a data-driven approach, knowledge about machines are extracted internally from machine operation data, instead of externally from domain experts. In the following, we present works that are representative of the domain.

In [20], the authors present an Artificial Neural Network (ANN)-based fault diagnosis for rolling element bearings using time-domain features. Five vibration signals from sensors are used to extract five time-domain features as the input of the designed ANN. The experimental results show that the accuracy can reach 62.5%-100%. Benkercha et al. [21] propose an approach based on the building of a Decision Tree to detect and diagnose the faults in grid connected photovoltaic system (GCPVS). The features used include temperature, irradiation and power ratio, and the class labels contains free fault, string fault, short circuit fault or line-line fault. Experimental results indicate that the diagnosis accuracy can reach 99.80%. In [22], Soualhi et al. apply the Hilbert-Huang transform (HHT)[23] to extract health indicator from vibration signals and utilized SVM to achieve fault classification of bearings. A method based on the evidential k-NN (EKNN) rule in the framework of evidence theory to achieve a condition monitoring and early warning in power plants is developed and presented in [24].

Several surveys on different data-driven approaches for fault detection and diagnosis exist. Zhao et al. [25] provide an overview of Deep Learning based machine health monitoring systems. All the systems presented are aimed at fault identification and classification. Zhang et al. [26] present more recent results on Intelligent fault diagnosis with small & imbalanced data, which refers to build intelligent diagnosis models using limited machine faulty samples to achieve accurate fault identification. In addition, a series of survey papers focus on the fault diagnosis for a specific type of components, e.g., bearing, rotating machinery, wind turbines [27]. In [28], Zhang et al. systematically summarize the existing literature employing machine learning (ML) and data mining techniques for bearing fault diagnosis. Liu et al. [29] provide a comprehensive review of AI algorithms in rotating machinery fault diagnosis from the perspectives of theories and industrial applications. In the same direction, a more recent review of the role of AI in rotor fault diagnosis is presented in [30]. In particular, it is based on fault-wise analysis, providing a proper categorization of rotor failures, rather than on component-wise analysis.

The aforementioned works have given interesting contributions related to the field of fault detection and diagnosis, but most of them are equipment specific. They monitor the behavior of different properties of a machine. They are suitable and efficient when consuming data streams to detect abnormal patterns in the values of a machine's property (e.g. temperature, vibration, etc.). However, they have two drawbacks: (i) the need, in advance, for a huge amount of annotated data for model training [31, 32], most of the research is conducted in laboratories with equipment in test benches or using one of the few public datasets; and (ii) the lack of explicit model to explain decisions. It is therefore difficult to interpret the data and it also complicates the interoperability and re-usability of the models.

### *2.2. Knowledge-based approaches to condition monitoring*

Many traditional condition monitoring systems and fault diagnosis systems make use of a priori expert knowledge and deductive reasoning processes [33]. Expert knowledge is exploited to build an ontology that formally describes concepts and relationships that exist in the industrial domain. Different ontologies have been built for monitoring tasks such as predictive maintenance and health assessment systems [34]. Ontology-based modeling allows: (1) knowledge sharing between computational entities by having a common set of concepts, (2) logic inference by exploiting various existing logic reasoning mechanisms to deduce high-level concepts from raw data, and (3) knowledge reuse by reusing well-defined ontologies of different domains. Ontologies can be employed to represent different machinery systems and devices, and can be combined with various existing rule-based reasoning algorithms to achieve monitoring and fault diagnosis. This is done based upon the evaluation of on-line monitored data according to a set of rules which is pre-determined by expert knowledge. Usually, rules are expressed in the form: **IF** $\langle antecedent \rangle$ **THEN** $\langle consequence \rangle$. In this way, the consequence can make changes that affect the system that is being monitored, if the conditions in the antecedent are met. In the following, we make a review on several knowledge-based approaches for condition monitoring.

Recently, Gul et al. [35] propose a condition monitoring based approach for risk assessment in a rail transportation system by using a fuzzy rule-based expert system. In [36], Kharlamov et al. design a rule-based language for equipment diagnosis in ontology mediated data integration scenarios such as industrial IoT. Its advantages include the easy formulation of diagnosis programs with hundreds of pieces of complex equipment. Berredjem et al. [37] propose a fuzzy expert system to localize bearing faults diagnosis as well as distributed faults. The fuzzy rules are automatically induced from numerical data using an improved range overlaps method.

A more recent approach in this direction is the one proposed in [38], the authors introduce a novel hybrid approach that combines data mining and semantics to facilitate predictive maintenance tasks in manufacturing processes. This approach uses chronicle mining to predict the future failures of the monitored industrial machinery, and a Manufacturing Predictive Maintenance Ontology (MPMO) with its rule-based extension is used to predict temporal constraints of failures and to represent the predictive results formally. However, this approach does not consider the possibility to represent the evolution of a manufacturing process and the rule base. This makes it difficult for the predictive maintenance system to be able to adapt to dynamic situations over time, *e.g.* change of context. Although these approaches could be considered as data-driven, data mining technologies are here used to elicitate knowledge that will be exploited later.

Another rule-based model named Adaptive Neuro-Fuzzy Interference Systems (ANFIS) is applied for monitoring wind turbine SCADA (Supervisory Control and Data Acquisition) signals in [39, 40]. In order to obtain turbine condition statements, the authors implement rules given by an expert who is familiar with the behavior of the turbine, typical faults and their root causes. There are two types of rules: generic rules used to highlight anomalies, and specific rules providing specific condition or potential root cause. Schmidt et al. [41] develop a semantic framework, using an ontology-based approach for data aggregation, to support cloud-enabled maintenance of manufacturing assets. Xu et al. [42] propose an ontology-based fault diagnosis model to integrate, share and reuse fault diagnosis knowledge for loaders. A loader is a self-propelled heavy machinery having for main function to push and lift (load) ground pieces.

Ontologies provide a way to integrate, share and reuse of domain knowledge, but other reasoning methods have to be integrated with ontologies to achieve predictive maintenance. Rule-based reasoning can be employed in order to achieve monitoring and diagnosis. These rules are built from expert knowledge, or are extracted from the analysis of large data sets, as mentioned above. However, knowledge-based approaches have difficulties in dealing with new (unknown by experts) faults and acquires complete knowledge to build a reliable set of rules.

*Ontological models for the manufacturing domain*

A review of the existing knowledge-based condition monitoring systems is given in the previous section. In this section, we give an overview on the development and usage of ontologies in the manufacturing domain. These ontologies were defined with distinct purposes and, therefore, describe different types of information related to that domain. The development of a knowledge-based condition monitoring system requires domain knowledge about manufacturing processes to be represented in a formal way, thus making this knowledge usable by the system. To achieve this goal, semantic technologies, especially ontologies with their rule-based extensions, have shown promising capabilities for formalizing knowledge in various domains [43, 44].

A typical manufacturing system can be characterized according to three main notions: Product, Process and Resource [45]. In [46], authors have developed a product ontology, named ONTO-PDM. It provides a semantic layer to business, design and manufacturing product-related information. ONTO-PDM harmonizes the product-related knowledge and standards, and this harmonization has shown positive results in solving the interoperability problems among different enterprises and applications. Other works in this direction are: OntoSTEP (ONTOlogy of Standard for the Exchange of Product model data) [47], which allows the description of product information mainly related to geometry; and MCCO (Manufacturing Core Concepts Ontology) [48] that focuses on interoperability across the production and design domains of product life-cycle. It provides some core classes in categories such as `ManufacturingProcess`, `ManufacturingFacility`, `ManufacturingResource` and `Feature`.

Like ONTO-PDM, the Process Specification Language (PSL) ontology [49] is another notable development work in the area of product information modelling. The PSL ontology covers several domains such as manufacturing, engineering and business processes. In this section we only focus on the manufacturing domain. Even though it mainly focuses on fundamental concepts for representing manufacturing processes, the ontology also provides a basis for the formal description of elements and entities that constitute a Process. The foundational elements of the core of the PSL ontology are four primitive classes (`Activity`, `Activity-occurrence`, `Timepoint`, `Object`), three primitive relations (`participates-in`, `before`, `occurrence-of`) and two primitive functions (`beginof`, `endof`). From the manufacturing product point of view, the notion Product could be deemed as a sub-concept under the core concept `Object` in the ontology. The PSL ontology provides a robust semantic foundation for mod-

elling manufacturing product information. Furthermore, as indicated by the name, the PSL ontology is a powerful approach for the representation of manufacturing processes.

Other ontologies have been developed to enhance the performance of product information modeling in the manufacturing domain. As the PSL ontology, these ontologies include the notions of resource and process. Among them, the MASON [50] and the ADACOR [51] ontological models are considered as pertinent. The MASON ontology presents a detailed conceptualization mainly focused on the products because it was used for cost estimation of the production of mechanical parts. MASON is an upper ontology for representing what the authors consider the core concepts of the manufacturing domain: products, processes and resources. As a result, the main classes of MASON are `Entity`, for specifying the product; `Operation`, for describing all processes linked to manufacturing; and `Resource`, for representing concepts regarding machines, tools, human resources and geographic resources. The ADACOR Ontology, developed as a part of the ADACOR architecture, provides a refined conceptualization to model operations, production plans and work orders regarding customer orders. The ontology is not openly available. Hence to be applied outside of the ADACOR project, the ontology has to be rebuilt with the information provided in the documentation. The SIMPM (Semantically Integrated Manufacturing Planning Model) ontology [52] is an upper ontology that is also focused on the representation of production plans. It models the fundamental constraints of manufacturing process planning: manufacturing activities and resources, time and aggregation.

Besides the ontologies presented above, there exist other ontologies that are more focused on modeling manufacturing processes and resources. The P-PSO (Politecnico di Milano Production Systems) ontology [53] considers three aspects in the manufacturing domain: the physical aspect (the material definition of the system), the technological aspect (the operational view of the system) and the control aspect (the management activities), for information exchange, design, control, simulation and other applications. Thus, its main classes are `Component`, `Operation` and `Controller`, which model the aforementioned three aspects, as well as part, `Operator` and `Subsystem`. The MSDL (Manufacturing Service Description Language) ontology [54] allows to describe manufacturing services. More precisely, a `ManufacturingService` is seen as a service that is provided by a `Supplier` and that has some `ManufacturingCapability`, which is enabled by some `ManufacturingResource` and delivered by some `ManufacturingProcess`. Another ontology with similar purposes to MSDL is MaRCO (Manufacturing Resource Capability Ontology) [55], that defines capabilities of manufacturing resources. Its main class is `Capability`, which is specialized to cover both simple capabilities (e.g. Fixturing, SpinningTool) and combined capabilities (those that require a combination of two or more simple capabilities, e.g. PickAndPlace, which requires Finger-Grasping or Vacuum Grasping, Moving and Releasing). Another interesting work is the one presented in [56]. It proposes a base ontology to represent a production line. The base ontology integrates four ontologies: (1) the device ontology, with concepts such as `Machine`; (2) the process ontology, with a taxonomy of the different `Operations` performed by the technical equipment; (3) the parameter ontology, with concepts such as `QualityofService`; (4) the product ontology, with the product information.

The presented ontologies define terms and relations for modeling the manufacturing domain, such as machine, process and the possible relations between them. The ontologies previously described do not allow representing the sensors attached to the company's resources and the properties they measure, although the sensors are also fundamental for anomaly detection on production lines and reflect the correctness of mechanical system conditions. In addition, these ontologies do not allow the representation of the context of the machines and processes. This is important to determine whether the observations collected by sensors represent abnormal behavior or not.

As mentioned in the introduction, IoT is a key element of Industry 4.0. Nowadays, it is possible to use sensor networks to detect and identify a wide variety of observations, from simple phenomena to complex events and situations. However, the lack of integration between these sensor networks often isolates important or relevant data. To deal with this issue, the Semantic Sensor Web (SSW) approach provides tools that allow the integration of data from multiple data sources [57]. It introduces semantic annotations for describing: (1) the data produced by the sensors, introducing spatial, temporal, or situation/context semantics; and (2) the sensors and the sensor networks that provide such data. Furthermore, there are also works on defining suitable ontologies for data and sensors to enable both the integration of data from multiple sensor networks and external sources, and reasoning on such data. As an example, the W3C Semantic Sensor Network Incubator Group [58] developed an ontology to describe sensors and sensor networks, the Semantic Sensor Network Ontology (SSN). Another work in this direction is

Table 1
Comparison of manufacturing ontologies.

| | Manufacturing | | | Context | | | | Change over time | IoT | Monitoring and Diagnosis | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Ontologies | Product | Process | Resource | Identity | Activity | Time | Location | | Sensor | Situation | Cause |
| MASON | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| ADACOR | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| PSL | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| MSDL | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| SIMPM | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| MaRCO | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ |
| ONTO-PDM | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| P-PSO | ✗ | ✓ | ✗ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| MCCO | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| [56] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| MPMO | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ |
| OntoSTEP | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

SAREF4INMA [59], that pursues interoperability with industry standards. The SSN ontology is further detailed in Section 3.1.

### *2.3. Discussion*

Data-driven approaches, based on annotated data, and knowledge-based approaches, based on expert knowledge, for condition monitoring are discussed in this section. Furthermore, we have reviewed existing ontologies and their rule-based extensions for knowledge-based condition monitoring systems. Table 1 summarizes the domain coverage of the discussed ontologies. We evaluate the domain coverage and scopes of these knowledge models by examining whether the key concepts required for condition monitoring in Industry 4.0 exist and are formally described. We categorized these key concepts into five sub-domains: Manufacturing, Context, Change over time, IoT, and Monitoring and Diagnosis. For the Manufacturing sub-domain, the key concepts are Product, Process and Resource. For the Context sub-domain, the key concepts are Identity, Activity, Time, and Location. In this case, the term Activity, used in the definition of context proposed by Dey et. al. [5], refers to industry-related processes. Therefore, we consider Activity and Process as two related concepts. For the IoT sub-domain, Sensor and Observation are the key concepts. For the Monitoring and Diagnosis sub-domain, the key concepts are Situation and Cause. These concepts are the columns of Table 1, and the ontological models are the rows. If a check mark is placed in the table then the concept exists in the corresponding ontology. Otherwise, a cross mark is assigned.

After reviewing the ontological models mentioned previously, we recognize that none of them provides a satisfactory knowledge representation of the five required sub-domains. Some of these knowledge models focus on a narrow field, such as manufacturing resource planning and work orders, and they do not formalize context-related concepts, *e.g.*, Activity and Location. Also, none of the existing ontologies provides a way of representing knowledge that evolve in time. To perform smart condition monitoring, the knowledge base should contain not only the machine-interpretable knowledge for characterizing the manufacturing entities or processes which are being monitored but also the knowledge about abnormal situations that are associated to failures. This motivates us to develop a more expressive and complete ontological model that provides a rich representation of the domain knowledge in the fields of manufacturing considering the notion of context.

In addition, the ontological models of the smart systems reviewed in this section are rather static. As described in the introduction, machines perform manufacturing processes in different contexts and these contexts change over time. According to the context in which a manufacturing process is executed, the rules that manage the process can change. In order to represent the fact that a machine is performing a process in these contexts, the knowledge base needs to evolve in time to represent this changing knowledge.

## 3. A novel knowledge-based framework for condition monitoring

The goal of the proposed semantic framework is to address the evolution of knowledge bases (as explained in the introduction) in Industry 4.0 for performing condition monitoring. The main components of our proposal are shown

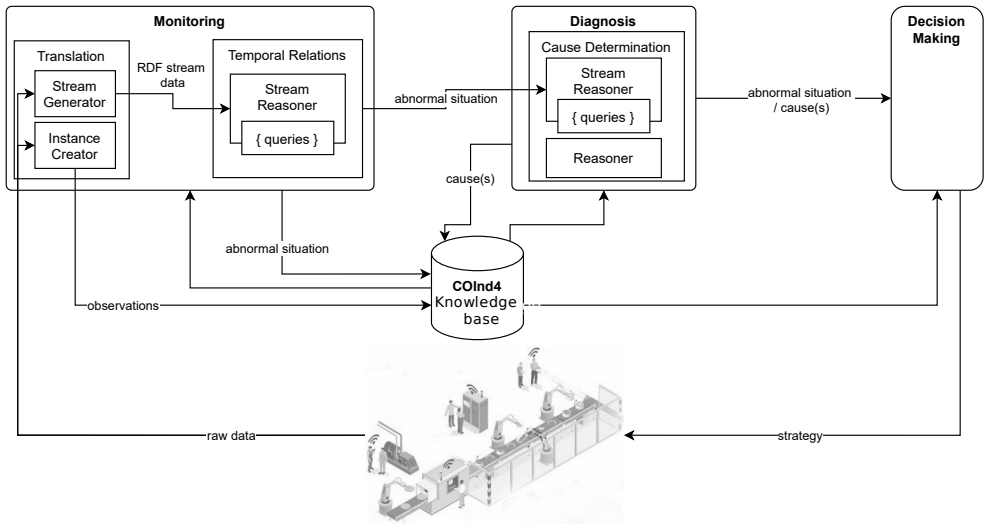


Fig. 1. The main components of the proposed framework.

in Figure 1: (1) the Monitoring component, which is in charge of detecting abnormal behaviors of the production system; (2) the Diagnosis component, which identifies the possible cause(s) that generate an abnormal behavior; and (3) the Decision making component, which determines the adaptation strategies that are needed to correct abnormal behaviors of the production system.

All the components rely on a formal model to semantically enrich data representation and processing. Each of the components operates as an expert does. They can be seen as smart systems to solve complex problems by reasoning. Each one uses the knowledge base to perform its functions and also makes the knowledge base evolve by adding their own results in it. In this way, the tasks of each component are performed considering the updated knowledge base which is a virtual representation of what happens in the real factory.

### *3.1. The Context Ontology for Industry 4.0 (COInd4)*

The development of a smart system requires that the domain knowledge is represented in a formal way. To achieve this objective, ontologies have been widely used to formalize knowledge about the industrial domain [42, 43].

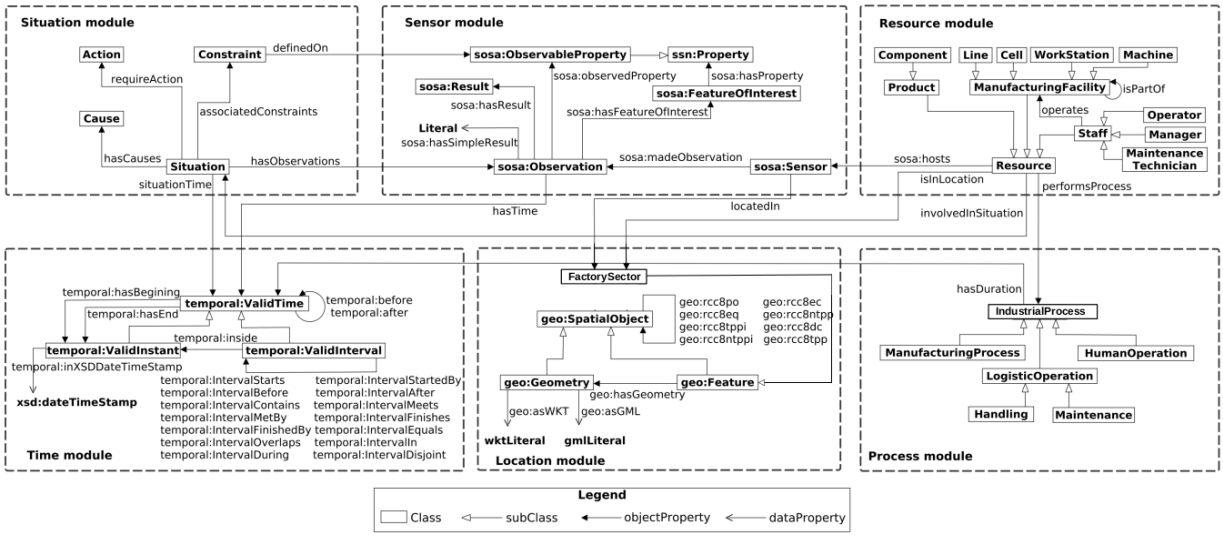


Fig. 2. The Context Ontology for Industry 4.0 (COInd4).

However, as indicated in the related work, most of these existing ontological models focus on a very specific field and sometimes lack a formal representation of context.

Our COInd4 ontology is a foundation of our approach. It represents the elements of the real factory, such as machines, processes and sensors, with special emphasis on modeling the context of operation of these elements. Relevant situations representing abnormal behaviors are also represented in the model. The goal of this semantic model is to represent the concepts and relations in the industrial domain to enable context representation and reasoning.

The ontological model is developed according to the methodology proposed in [60]. An overview of the classes and properties from the proposed ontological model is shown in Figure 2. This ontological model is composed of six modules. They are described in more detail below together with the definition of their main concepts using Description Logics (DL) [61].

### *3.1.1. The Resource module*

The *Resource* module aims to provide a comprehensive representation of manufacturing products and resources which are physical objects used for executing a set of operations during the manufacturing processes. The `Resource` class represents resources of a company, such as physcial assets, staff and products. The `ManufacturingFacility` is a subclass of the *Resource* concept and it represents the machines and physical assets of a company. The `ManufacturingFacility` class is further decomposed in `Machine`, `Workstation`, `Cell` and `Line`. This decomposition enables to represent different Industry 4.0 processing sequences and flexible or re-configurable production lines. Another relevant point is that by using this taxonomy, it is possible to describe the context of a `ManufacturingFacility` at different nested levels (*e.g.* characterising the context of a `Line` and depicting the context of a `Workstation` that belongs to that `Line`). The DL axioms for defining the most relevant classes of the *Resource* module are:

$$\begin{aligned}
\texttt{ManufacturingFacility} &\sqsubseteq \texttt{Resource}\\
\texttt{Staff} &\sqsubseteq \texttt{Resource}\\
\texttt{Product} &\sqsubseteq \texttt{Resource}\\
\texttt{ManufacturingFacility} &\sqsubseteq \exists \texttt{operates}^{-1}.\texttt{Staff}\\
&\quad \sqcap (\texttt{Line} \sqcup \texttt{Cell} \sqcup \texttt{WorkSation} \sqcup \texttt{Machine})
\end{aligned}$$

### *3.1.2. The Process module*

The *Process* module gives a formal representation of processes related to manufacturing. A `IndustrialProcess` represents a task or a set of tasks performed by one or more resources and is specified with its contextual information, *e.g.* occurring time, place and related resources. The domain concepts given in the *Process* module represent a taxonomy of manufacturing processes. For example, processes include control operations and assembly operations. The `IndustrialProcess` class is sub-categorized into `LogisticOperation`, `HumanOperation` and `ManufacturingProcess`. This last kind of process includes cutting, drilling, milling, among others. The DL axioms used for defining the classes from the *Process* module are:

$$\begin{aligned}
\texttt{LogisticOperation} &\sqsubseteq \texttt{IndustrialProcess}\\
\texttt{HumanOperation} &\sqsubseteq \texttt{IndustrialProcess}\\
\texttt{ManufacturingProcess} &\sqsubseteq \texttt{IndustrialProcess}\\
\texttt{IndustrialProcess} &\sqsubseteq \exists \texttt{performsProcess}^{-1}.\texttt{Resource}\\
&\quad \sqcap \exists \texttt{hasDuration.time:ValidTime}
\end{aligned}$$

### *3.1.3. The Sensor module*

The *Sensor* module considers knowledge about sensors and the observations generated by them. The main goal of this module is to enrich the heterogeneous sensor data with a formalized semantics in order to improve interoperability among different systems and re-usability. This is done by adding metadata about contextual information (such as locations or timestamps). The conceptualization of sensor measurements related to the sensing activity itself and to observations come from the Semantic Sensor Network[1] (SSN) ontology presented in [58]. One of the most relevant concepts of the SSN ontology that we reuse in our model is the `sosa:Sensor` class. A `sosa:Sensor` is a device that detects and responds to some type of input from the physical environment. To do this, sensors implement a spe-

[1] available at: http://www.w3.org/ns/ssn/

cific method that results in the calculation of the value of the observed property (`sosa:ObservableProperty` class). Another concept of the SSN ontology that is key to our model is the `sosa:Observation` class. It provides the structure to represent a single observation. Therefore, an instance of `sosa:Observation` is related to a single measurement (`sosa:Result` class) performed by a given sensor (`sosa:Sensor` class) on a single property of some entity (`sosa:ObservableProperty` and `sosa:FeatureOfInterest` classes, respectively). To specify the observation time, SSN has the `sosa:resultTime` datatype property that relates a `sosa:Observation` to a time instant. It indicates the time at which the observation was made.

### *3.1.4. The Location module*

The *Location* module provides concepts and relations that allow representing the abstraction of physical spatial places. The `geo:SpatialObject` class contains two sub-classes called `geo:Feature` and `geo:Geometry`. The `geo:Feature` class represents 3D-objects or 2D-areas and can be assigned geometries that describe them through the `geo:hasGeometry` property. Therefore, a `Resource` is a `geo:Feature` (`Resource` $\sqsubseteq$ `geo:Feature`). In order to represent sectors of the factory, the class `FactorySector` is defined as a subclass of `geo:Feature`. These classes are reused from the GeoSPARQL ontology [62]. GeoSPARQL uses the Region Connection Calculus, introduced in [63]. More specifically, it uses RCC-8 (the version with eight relations) that is well-known for representing mereo-topological relationships between spatial regions. RCC-8 consists of 8 basic relations that are possible between two regions: disconnected (DC), externally connected (EC), equal (EQ), partially overlapping (PO), tangential proper part (TPP), tangential proper part inverse (TPPi), non-tangential proper part (NTPP), non-tangential proper part inverse (NTPPi). From these primitives relations, more complex combinations can be built. For example, a proper part (PP) is the union of TPP and NTPP.

### *3.1.5. The Time module*

The *Time* module enables a consistent representation of temporal information in the industrial system. Temporal modelling is key in the manufacturing domain because both sensor measurements and process representations need to be positioned in a temporal dimension so that they can be correctly interpreted by the industrial system. This module provides a way to represent temporal entities (*e.g.* time instants and time periods) as well as to interpret temporal relations (*e.g.* before, after, during, etc.). The *Time* module comprises all information related to the current time and allows time-stamping all the context information that may change over time.

The SWRL Temporal Ontology[2] (SWRLTO) proposed in [64] is reused as a temporal model. It provides a simple and efficient solution to operate with temporal information in queries and rules. This ontology has a main class named `temporal:ValidTime`, which has two sub-classes, `temporal:ValidInstant` and `temporal:ValidInterval`. The `temporal:ValidInstant` class denotes a point on a timeline (an instant) and the `temporal:ValidInterval` class models the time between two instants. These are declared by two data properties, `swrlto:hasStartTime` and `swrlto:hasFinishTime`. Besides, the `swrlto:Granularity` class describes the unit of measurement of the time reference (e.g., months, days, hours). Another important class is `temporal:Fact`, which corresponds to the `swrlto:ExtendedProposition` class. It models an entity that can extend over time and that is associated to the `temporal:hasValidTime` property, indicating the time period during which the associated information is true. The range of this property is the `temporal:ValidTime` class. Regarding temporal reasoning and querying, SWRLTO implements a series of predicates embedded in SWRL that allows operating with temporal relations. Most of them are based on Allen's time algebra [65]. In addition, SWRLTO offers some SWRL operators to perform granularity conversions and duration calculations in different time units. The possible values are: *Years*, *Months*, *Days*, *Hours*, *Minutes*, *Seconds*, and *Milliseconds*. There are no specific spatio-temporal built-ins, however, the combination of spatial and temporal operators allows the construction of spatio-temporal built-ins.

### *3.1.6. The Situation module*

The *Situation* module aims at representing relevant states of affairs associated to a particular scenario of interest and can consist of resources, observations and processes. A `Situation` defines a state of affairs associated to a particular scenario of interest. In this case, an abnormal situation is a specific scenario in which the system state

[2] http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl

shows a particular combination of sensed values for its attributes (observations) that are not desirable and could lead to a failure. Thus, an abnormal situation describes "intermediate" or "abnormal" manufacturing conditions that are described with expert knowledge, in the form of constraints. Consequently, from a conceptual point of view, a situation involves a combination of at least, one resource, eventually associated to its location, and at least one sensor measurement, fulfilling the constraints set by the expert. The whole can be linked through spatial, temporal and/or spatio-temporal relationships.

It should be noted, that a situation is actually a general description (concept) about a specific scenario and there may be several instances (individuals) of these situations occurring at different points in time. For example, an abnormal behavior of a particular machine can be represented by a situation. This situation can happen more than once on the same machine or even on another machine of the same type; *i.e.* which means that there can be different instances of the abnormal situation in the ontology. These situations take place over a period of time, and their duration is represented by the property `situationTime`.

The `Situation` class is a subclass of the `temporal:Fact` class, since a situation occurs over an interval of time (`Situation` $\sqsubseteq$ `temporal:Fact`), therefore, it has a valid time. Other relevant concepts related to the concept of situation are `Cause`, `Action` and `Constraint`. The `Cause` class represents the possible causes that provoke a situation. A situation is linked to its causes through the `hasCauses` property. The `Action` concept represents actions (preventive or reactive) needed to mitigate the situation or its severity. The actions can be diverse, such as tasks performed by operators or remote changes in the operating parameters of the machines. They are linked to a situation through the `requiredAction` property. In both cases, the link between a situation and its causes and the required actions is obtained from expert knowledge. Finally, the `Constraint` concept represents defined restrictions on certain observable properties, such as a maximum threshold for one value. As a situation is expressed as a combination of constraints, this is represented through the `associatedConstraints` relation. Considering the concepts defined above, the concept of situation is described by the following DL axiom:

$$\begin{aligned}\texttt{Situation} \sqsubseteq\ & \texttt{temporal:Fact} \\ & \sqcap \exists \texttt{hasConstraint.Constraint} \\ & \sqcap \exists \texttt{hasObservation.Observation} \\ & \sqcap \exists \texttt{situationTime.time:TemporalEntity} \\ & \sqcap \exists \texttt{involvedInSituation}^{-1}\texttt{.Resource}\end{aligned}$$

### *3.1.7. Integration of all the modules*

The ontological modules described above reuse others ontologies that have been developed to address specific needs in different domains and following different approaches. Therefore, it is necessary to semantically integrate them in order to have an ontological model that allows context modelling in the Industry 4.0 domain. The integration among the modules is done mainly through the use of relationships (object properties, equivalence ($\equiv$) and subsumption ($\sqsubseteq$)) that link concepts from one module with concepts from other modules. In Table 2 we present and defined the main relations linking concepts from different modules.

Table 2

Relations (object properties) that link the modules of the COInd4 ontology

| Object Property | Domain | Range | Description |
|---|---|---|---|
| `sosa:hosts` | `Resource` | `sosa:Sensor` | It asserts that one or more sensors are attached to a Resource. |
| `performsProcess` | `Resource` | `IndustrialProcess` | It asserts that a resource performs one or more processes. |
| `isInLocation` | `Resource` | `FactorySector` | It asserts that a resource is located in a specific location. |
| `involvedInSituation` | `Resource` | `Situation` | It asserts that a resource is involved in one or more particular situation(s). It holds during the interval in which the situation occurs. |
| `hasDuration` | `IndustrialProcess` | `temporal:ValidPeriod` | It asserts the duration of a process. |
| `happensIn` | `IndustrialProcess` | `FactorySector` | It asserts the location where a process occurs. |
| `concernedBySituation` | `IndustrialProcess` | `Situation` | It asserts that a process is concerned by one or more situation(s). |
| `hasTime` | `Observation` | `temporal:ValidTime` | It asserts the time when an observation was made. |
| `locatedIn` | `sosa:Sensor` | `FactorySector` | It asserts that a sensor is deployed in a certain location. |
| `definedOn` | `Constraint` | `sosa:ObservableProperty` | It asserts that a constraint is defined on a particular property. |
| `hasObservation` | `Situation` | `Observation` | It asserts that a situation has one or more observations. |
| `situationTime` | `Situation` | `temporal:ValidTime` | It asserts the duration of a situation. |
| `occursIn` | `Situation` | `FactorySector` | It asserts the location or locations where a situation occurs. |

It is worth highlighting that new knowledge can be inferred or made explicit from these relations, as well as many of these relations can be inferred from prior knowledge. For example, the location of where a sensor is located can be inferred from the location of the resource that hosts that sensor. Another interesting example is that from the observations we can obtain the necessary information to determine the resources involved in a situation as well as the processes concerned by the situation, its location and its time.

Each of the six ontological modules that constitute the COInd4 ontology and its integration, enable to represent situations of interest that take into account the context of the resources. Furthermore, our ontological model provides a way to capture the dynamic changes and the evolution of knowledge in time, such as the different situations a machine can go through. This is a key point to deal with the dynamics of manufacturing processes as mentioned in the introduction of this paper. The proposed model is written in OWL DL, a standard language with formal semantics based on logic [66] and is available[3].

### *3.2. The Monitoring component*

Certain situations that may lead to machine failures can be detected by interpreting observations in their contexts, for example, if an observation has an abnormal value it may be because another parameter is having abnormal values too. Let us consider the case where the temperature of a machine and the temperature of one of its components are being monitored. It is known that an increase in the temperature of the machine may be due to an increase in the temperature of its component, or vice versa. This allows the exploitation of expert knowledge about relationships between the values of certain parameters from the machines, from the processes and from their context for interpreting observations. Through the early detection of situations, the maintenance schedule can be adapted or further measures to prevent unexpected downtime can be taken.

The **Monitoring** component is responsible for collecting data from sensors in order to detect abnormal situations. It uses the knowledge base and modifies it as well. This component mainly uses the model for enriching data collected from sensors with contextual information. This allows to derive situations from context and sensor information of lower-level abstraction. Once an abnormal situation is detected, the situation together with the resources and the processes involved in it are added to the model; so that the model also represents which resources and processes are involved in the situation. This allows the model to continuously represent what is happening in the real manufacturing process.

In the following we explain the *Translation* and *Temporal Relations* modules. These modules, which are part of the **Monitoring** component as shown in Figure 1, are involved in the detection of situations. The interaction between them, as well as with the ontological model, is also described.

- The *Translation* module is responsible for (i) converting acquired data from sensors to RDF streams, and (ii) inserting them as instances into the ontology. Both tasks are performed respectively by the *Stream Generator* and the *Instance creator* sub-modules.
  The *Stream Generator* sub-module performs semantic enrichment of the acquired data, using the concepts and relations among them as defined in the ontological model. This allows the module to stream out semantically enriched data streams that are then consumed by the *Stream Reasoner*. The output streams are RDF streams. An RDF stream is an ordered sequence of pairs, where each pair is constituted by an RDF triplet and its timestamp $t$, (<Subject,Predicate,Object>,$t$).
  The *Instance creator* sub-module creates instances from the received data and inserts them in the ontology, *i.e.* it is in charge of populating the ontological model with observations and their corresponding metadata, such as the sensor(s) which made the observation(s), the observed property and the timestamp.
- Once the data from the distributed and heterogeneous data sources are available in a homogeneous, contextualized and temporally ordered representation, the streams are explored by the *Temporal Relations* module to generate new knowledge. A set of queries, which combine background knowledge extracted from the ontology and some relevant parts of the streams, is registered and executed by the *Stream Reasoner* over the data streams. These queries represent particular situations to be identified and they include mainly temporal dependencies

[3] https://gitlab.insa-rouen.fr/fgiustozzi/STEaMINg-Ontology

between observations (that can be normal ones or anomalies). When this module detects situations, they can be converted into RDF streams and be returned as output, *i.e.* this module produces streams of situations as output. This output feeds another stream reasoner in the *Cause determination* module. In this way, the *Temporal Relations* module itself can be seen as an advanced sensor able to produce high level data. Furthermore, the detected situations are stored in the ontological model indicating also the resources involved in that situation. This is represented by the `involvedInSituation` relation.

In Section 4, the case study enables to explain in more detail how each of the modules of the Monitoring component works for data enrichment using context information as well as how the detection of situations defined by expert knowledge is performed.

### *3.3. The Diagnosis component*

The **Diagnosis** component is responsible for determining the possible causes of the abnormal behavior. The crucial difference between monitoring and diagnosis relies on the nature of the output. Monitoring observes a discrepancy between the expected and detected behavior without exploration of the cause or fault underlying it [67]. However, in many domains such as Industry 4.0, monitoring and diagnosis are tightly coupled tasks: when monitoring observes an anomaly in the behavior, a diagnosis task is started, using the monitored information as input.

As the monitoring component, the Diagnosis component both uses the knowledge base and modifies it. This component mainly uses the model to determine the cause(s) of a detected abnormal situations. The association between the causes and the abnormal situations are obtained from expert knowledge. Once the possible cause(s) associated to the detected abnormal situation are identified, these cause(s) are linked to the detected situation in the knowledge base. This allows the model to represent which are the causes of an abnormal situation detected from the real manufacturing process.

The Diagnosis component has only one module called *Cause Determination* to determine the possible causes that caused an abnormal situation. The purpose of the *Cause Determination* module is to identify the possible causes that generated a situation detected by the *Temporal Relation* module. For this, two components are used separately: a *Stream Reasoner* and a *Reasoner*, as can be seen in Figure 1.

Stream reasoning is more suitable to highly dynamic data than classical reasoning approaches. Thus, in the case where causes need to be determined in real-time, the *Stream Reasoner* is used to identify the causes. The association between the situations and their possible causes are stored in the ontological model and they are exploited by this module to return them. However, it is possible that some situations do not have identified causes in a real scenario, in which case the system notifies that the causes are unknown.

Therefore, in order to determine the possible causes of a situation classical reasoning approaches can be used. They provide other inferences that can help in determining the causes of a situation. More complex queries can be performed to the ontological model in order to extract information useful for cause determination. For example, it can be necessary to consider the Open World Assumption, which states that the absence of a statement alone cannot be used to infer that the statement is false. In this case, the *Reasoner* can be also used over the ontology to infer the causes, if the real-time requirement is not needed. This last option has some advantages over the previous one. If the cause is identified later, it is added as an instance to the ontological model and linked to the situation for future use. This is detailed in the section presenting the proof of concept.

In both cases, the *Diagnosis* component provides the *Decision Making* component with the detected situation together with the possible causes inferred by the reasoner(s).

### *3.4. The Decision Making component*

Manufacturing processes are not always executed under optimal conditions. Nevertheless, they can sometimes continue their execution in degraded conditions without being completely stopped. Expert knowledge enables to describe these *"intermediate"* manufacturing conditions. The associated abnormal situations can have different levels of severity, and be nested in different ways. They can impact other processes or resources that participate in these processes and they can trigger other situations that represent a risk of major interruption of the manufacturing

process or a risk of accident. Once an abnormal situation and its causes have been detected, the Decision Making component is responsible for determining and applying the adaptation strategies that are needed to improve the behavior of the production system. These strategies may include changes to the operation parameters of a machine or the launching of a maintenance task. This component also uses the knowledge base to determine which actions are possible to execute to improve the operating conditions.

In order to support the Decision Making component of our framework, it is relevant to consider which other situations can be reached from the current situation, to choose the most appropriate action. Therefore, we propose an approach to establish an order among the situations. The order among the situations is a hierarchy that depends on how situations' constraints are correlated. This order represents a road-map of all the situations, normal or abnormal, that can be reached from a given one. In this way, it is possible to identify the actions that can be taken to correct the abnormality, considering that certain actions can modify the value of a property and thus change the state of the system, either by satisfying another constraint or, on the contrary, by not satisfying constraints anymore. The idea driving this proposal is to formally represent a hierarchy among situations leading to failures.

The proposed approach uses the lattice theory [68–70] to provide an expressive formalization to order the situations in a taxonomic way. In this way, the hierarchy of situations is formally extracted from the situations definitions. In the following, we firstly introduce the definitions that are necessary for the construction of the hierarchy of situations, and secondly, we describe the steps to automatically build this hierarchy from the situations definitions and its associated lattice.

### *3.4.1. Definitions*

In order to formally represent a hierarchy of situations, let us consider the following structure $\langle \mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{T} \rangle$ where:

- $\mathcal{S} = \{s_1, s_2, s_3, \ldots, s_n\}$ is the set of all the situations,
- $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_m\}$ is the set of all the constraints,
- $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{C}$ is a binary relation that links a situation with a constraint, and
- $\mathcal{T} \subseteq \mathcal{C} \times \mathcal{C}$ is a binary relation that links a constraint with another constraint.

It is worth mentioning two aspects about situations that were defined in Section 3.1. The first one is that a situation defines an state of affairs that represents a particular scenario of interest and involves observations linked through spatio-temporal relationships, resources and processes. The second one is that situations are abstract, meaning that there may be different instances of a given situation. Instances of the same situation can happen during different periods of time and involve several resources, but they all satisfy the same constraints.

Set $\mathcal{C}$ contains all the constraints that are associated with one or more situations in the set of situations $\mathcal{S}$. These constraints concern properties of the processes, machines, resources and of the environment in which the tasks are executed. For example, if we consider variables `MC1_Temp` and `MC2_Temp`, corresponding to the temperature of a component of a machine and the temperature of another component of the same machine, then `MC1_Temp` $< 40°C$ and `MC1_Temp>MC2_Temp` are constraints defined on them.

The binary relation $\mathcal{R}$ is used to establish that a constraint is involved in a situation. We write $s_1 \mathcal{R} c_1$ to indicate that the constraint $c_1$ is involved in the situation $s_1$. The $\mathcal{R}$ relation is therefore built from the relationships between the situations and the constraints that are extracted from expert knowledge.

The sets $\mathcal{S}$ and $\mathcal{C}$ correspond to the `Situation` class and the `Constraint` class of our ontological model, respectively. The association between a situation and its constraints is represented through the `hasConstraint` relation in the ontological model, represented by the binary relation $\mathcal{R}$.

The example below is used to illustrate the definition of the structure and the operators. Let us consider the following sets of constraints and of situations, with six constraints and six situations, respectively: $\mathcal{C} = \{c_1, c_2, c_3, c_4, c_5, c_6\}$ and $\mathcal{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$.

The corresponding $\mathcal{R}$ relation, built from $\mathcal{S}$ and $\mathcal{C}$, is shown in the first two columns of Table 3. For example, the fourth line in this table indicates that situation $s_4$ happens (first column) if constraints $c_2$ and $c_5$ are satisfied (second column). Another way to show relation $\mathcal{R}$ is depicted in Figure 3. In this figure, we can observe how different situations share a part of the constraints in their definition. For example, $s_1$ and $s_2$ share constraints $c_1$ and $c_6$.

Some constraints can be more general than others, *i.e.* include others. Such is the case with, for example, constraints $c_1$ and $c_2$ defined as `MC1_Temp` $< 40°C$ and `MC1_Temp` $< 60°C$, respectively. If $c_1$ is satisfied, then $c_2$

Table 3
Example of situations and associated constraints (in bold face the constraints that are implied by other constraints)

| Situations | Constraints | $\mathcal{T}$ Constraints |
|---|---|---|
| $s_1$ | $c_1$, $c_3$, $c_6$ | $c_1$, $\mathbf{c_2}$, $c_3$, $\mathbf{c_4}$, $\mathbf{c_5}$, $c_6$ |
| $s_2$ | $c_1$, $c_4$, $c_6$ | $c_1$, $\mathbf{c_2}$, $c_4$, $\mathbf{c_5}$, $c_6$ |
| $s_3$ | $c_2$, $c_4$, $c_6$ | $c_2$, $c_4$, $\mathbf{c_5}$, $c_6$ |
| $s_4$ | $c_2$, $c_5$ | $c_2$, $c_5$ |
| $s_5$ | $c_3$, $c_6$ | $c_3$, $\mathbf{c_4}$, $\mathbf{c_5}$, $c_6$ |
| $s_6$ | $c_1$, $c_6$ | $c_1$, $\mathbf{c_2}$, $c_6$ |

is necessarily also satisfied. Furthermore, some constraints may imply other constraints due to physical properties extracted from expert knowledge or observations. For this reason, the $\mathcal{T}$ relation is defined to indicate that if a constraint is satisfied, then another one is also satisfied. We write $c_1 \mathcal{T} c_2$.

Consider the implications among the constraints from the set of all constraints $\mathcal{C}$ shown in Figure 3 (arrows labeled with $\mathcal{T}$). These implications are inferred from the order relations ($<$,$>$,$\leqslant$,$\geqslant$), from observations, or extracted from expert knowledge. Taking into account these relations, the constraints associated with each situation are established as shown in Table 3 (third column). This allows to associate both explicit and implicit (implied) constraints to a situation.

In order to extend the formalization between a situation and a constraint to a set of situations and a set of constraints, two operators are defined below, based on the use of both $\mathcal{R}$ and $\mathcal{T}$ relations. These operators are defined on a set of situations or constraints because each situation can involve several constraints, and several situations can have constraints in common.

The first operator $\lceil \mathcal{X} \rceil$ enables the retrieval of the set of constraints associated to a set of situations.

**Definition 1.** *For a situation set $\mathcal{X}$, $\mathcal{X} \subseteq \mathcal{S}$, let*

$$\lceil \mathcal{X} \rceil := \{c \in \mathcal{C} | \forall x \in \mathcal{X} : x \mathcal{R} c \vee \exists c' \in \mathcal{C} : x \mathcal{R} c' \wedge c' \mathcal{T} c\}$$

Considering situations $s_1$ and $s_2$ of the example presented above, the constraints related to both situations are $\lceil \{s_1, s_2\} \rceil = \{c_1, c_2, c_4, c_5, c_6\}$.

The second operator $\lfloor \mathcal{Y} \rfloor$ conversely enables the retrieval of the set of situations involving a set of constraints $\mathcal{Y}$.

**Definition 2.** *For a constraint set $\mathcal{Y}$, $\mathcal{Y} \subseteq \mathcal{C}$, let*

$$\lfloor \mathcal{Y} \rfloor := \{s \in \mathcal{S} | \forall y \in \mathcal{Y} : s \mathcal{R} y \vee \exists c' \in \mathcal{C} : s \mathcal{R} c' \wedge c' \mathcal{T} c\}$$

This operator retrieves all the situations involving at least all the constraints in the set $\mathcal{Y}$. Therefore, considering the example presented above, if the constraints $c_2$ and $c_5$ are considered, then the situations involving those constraints are $\lfloor \{c_2, c_5\} \rfloor = \{s_1, s_2, s_3, s_4\}$. Let us note that these situations may involve other constraints, *e.g.* $s_3$ with $c_4$ and $c_6$.

### *3.4.2. The situation hierarchy construction*

Using the elements of the structure $\langle \mathcal{S}, \mathcal{C}, \mathcal{R}, \mathcal{T} \rangle$ and the two operators $\lceil . \rceil$ and $\lfloor . \rfloor$ previously defined, the construction of the lattice representing the hierarchy of situations is detailed below.

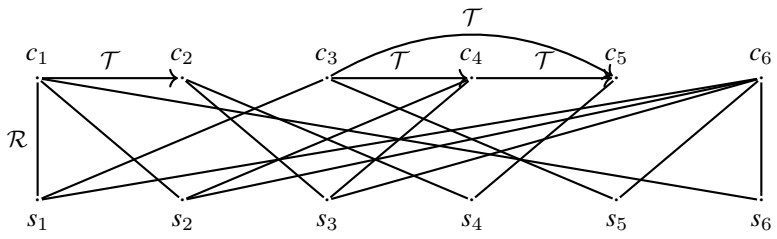


Fig. 3. Situations with constraints ($\mathcal{R}$) and implications among the constraints ($\mathcal{T}$).

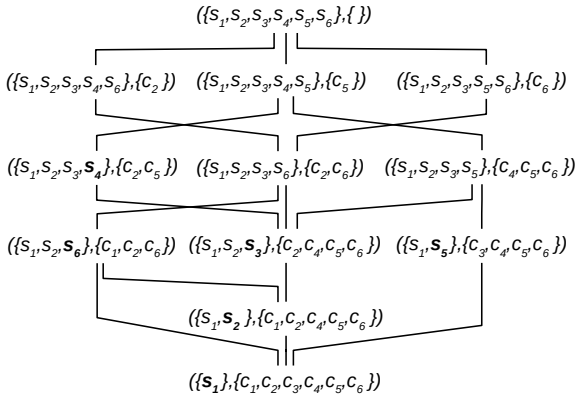


Fig. 4. Lattice representing the hierarchy of situations (the situations defined exactly by all the constraints of the second component of the pair in the node are shown in bold face)

In our approach, we group situations and their constraints as ordered pairs $(\mathcal{X},\mathcal{Y})$ where $\mathcal{X}$ is a set of situations and $\mathcal{Y}$ is a set of constraints such that $\lceil\mathcal{X}\rceil=\mathcal{Y}$ and $\lfloor\mathcal{Y}\rfloor=\mathcal{X}$. The first component of the pair is a set including all the situations that share all the constraints belonging to the second component. The second component is therefore the set including all the common constraints among the situations of the first component.

In order to find all the pairs and thus the nodes of the lattice, given a set of situations $\mathcal{S}$, a set of constraints $\mathcal{C}$, and relations $\mathcal{R}$ and $\mathcal{T}$, Algorithm 1 is applied. Firstly, $\lceil\{s\}\rceil$ is computed for each situation $s\in\mathcal{S}$ (lines 3-5). Then, for any two sets in this set of sets (*consSet*), their intersection is calculated. If this intersection is not yet contained in *consSet*, it is added to it (lines 6-12). This step is repeated until no new sets are generated. If set $\mathcal{C}$ of all the constraints and the empty set ($\{\}$) are not in *consSet*, they are also added to it (lines 13-18). These two sets yield the minimum and maximum nodes of the lattice, respectively. Finally, for every set $\mathcal{O}$ in *consSet*, $\lfloor\mathcal{O}\rfloor$ is computed (lines 19-21). In the end, set (*setofPairs*) of all the $(\mathcal{X},\mathcal{Y})$ pairs that satisfy $\lceil\mathcal{X}\rceil=\mathcal{Y}$ and $\lfloor\mathcal{Y}\rfloor=\mathcal{X}$ is obtained.

Considering the example in Figure 3 and Table 3, if the set of situations $\mathcal{S}$ and the set of constraints $\mathcal{C}$ are given as input to the algorithm, then the output are the nodes of the lattice shown in Figure 4. Let us note that since the situations are predefined, the search for all pairs can be done offline. If a situation needs to be added, it is either added to a node or a new node is created. The addition of a new situation to the hierarchy has an impact on the structure of the hierarchy.

**Algorithm 1** Calculate all the pairs $(\mathcal{X},\mathcal{Y})$ where $\mathcal{X}\subseteq\mathcal{S}$, $\mathcal{Y}\subseteq\mathcal{C}$, $\lceil\mathcal{X}\rceil=\mathcal{Y}$ and $\lfloor\mathcal{Y}\rfloor=\mathcal{X}$ (*setofPairs*)

**Require:** a set of Situations $\mathcal{S}$ and a set of Constraints $\mathcal{C}$
**Ensure:** $\{(\mathcal{X},\mathcal{Y})|\mathcal{X}\subseteq\mathcal{S}\wedge\mathcal{Y}\subseteq\mathcal{C}\wedge\lceil\mathcal{X}\rceil=\mathcal{Y}\wedge\lfloor\mathcal{Y}\rfloor=\mathcal{X}\}$
1: *consSet* $\leftarrow\{\}$ //*consSet is a set of constraint sets*
2: *setofPairs* $\leftarrow\{\}$
3: **for all** $s\in\mathcal{S}$ **do**
4: *consSet* $\leftarrow$ *consSet* $\cup\{\lceil\{s\}\rceil\}$
5: **end for**
6: **for all** $\mathcal{O}_1\in$ *consSet* **do**
7: **for all** $\mathcal{O}_2\in$ *consSet* **do**
8: **if** $\mathcal{O}_1\cap\mathcal{O}_2\notin$ *consSet* **then**
9: *consSet* $\leftarrow$ *consSet* $\cup\{\{\mathcal{O}_1\cap\mathcal{O}_2\}\}$
10: **end if**
11: **end for**
12: **end for**
13: **if** $\mathcal{C}\notin$ *consSet* **then**
14: *consSet* $\leftarrow$ *consSet* $\cup\{\mathcal{C}\}$
15: **end if**
16: **if** $\{\}\notin$ *consSet* **then**
17: *consSet* $\leftarrow$ *consSet* $\cup\{\{\}\}$
18: **end if**
19: **for all** $\mathcal{O}\in$ *consSet* **do**
20: *setofPairs* $\leftarrow$ *setofPairs* $\cup\{(\lfloor\mathcal{O}\rfloor,\mathcal{O})\}$
21: **end for**

Once all the pairs are found, the next step is to order them in a lattice to build the hierarchy of situations. A **lattice** is an algebraic structure that consists of a partially ordered set in which every two elements have a unique *supremum* (also called *least upper bound* or *join*) and a unique *infimum* (also called *greatest lower bound* or *meet*). A **partial order** is a pair $(\mathcal{P}, \preceq)$ where $\mathcal{P}$ is a set and $\preceq$ is a binary relation over $\mathcal{P}$ where $\preceq$ is reflexive, anti-symmetric and transitive. For our lattice, we consider $\mathcal{L} = \{(\mathcal{X}, \mathcal{Y}) | \mathcal{X} \subseteq \mathcal{S} \wedge \mathcal{Y} \subseteq \mathcal{C} \wedge \lceil \mathcal{X} \rceil = \mathcal{Y} \wedge \lfloor \mathcal{Y} \rfloor = \mathcal{X}\}$ and the following binary relation defined over it:

**Definition 3.** *Let $(\mathcal{X}, \mathcal{Y})$ and $(\mathcal{X}', \mathcal{Y}')$ be two pairs where $\mathcal{X}$, $\mathcal{X}'$ are sets of situations and $\mathcal{Y}$, $\mathcal{Y}'$ are sets of constraints. The situations in $\mathcal{X}'$ are reachable from $\mathcal{X}$ if the constraints in $\mathcal{Y} \cap \mathcal{Y}'$ are satisfied, noted as $(\mathcal{X}, \mathcal{Y}) \preceq (\mathcal{X}', \mathcal{Y}') \Leftrightarrow \mathcal{X} \subseteq \mathcal{X}' \wedge \mathcal{Y}' \subseteq \mathcal{Y}$.*

The hierarchy of situations has been proved to be a lattice [71]. The fact that the hierarchy is a lattice allows us to know the situations that can be reached from the current situation, knowing also the intermediate situations (this is inherited from the fact that a lattice is a partial order). In addition, let us suppose that two situations are happening simultaneously and these situations are represented in two different nodes of the lattice. Each set of situations from those nodes have a larger common set of situations with the constraints that are common to all situations in both sets of the two nodes. Dually, each set of situations have a smaller common subset of situations, which comprises all the constraints that all the situations in both sets of the nodes have. This allows to know from two situations that are happening, which situations can be reached by the production system: situations that satisfy the smaller number of constraints between the two situations that are happening (*supremum*) or situations that satisfy the greater number of constraints in common between the two situations that are happening (plus some other constraint(s)) (*infimum*).

The interpretation and exploitation of the lattice to support the decision making is detailed in the following section.

## 4. Proof of concept

In order to apply our approach, we have developed a software prototype. The software uses deductive approaches, domain ontologies and ontology reasoning, and stream reasoning to analyze industrial data and to detect abnormal situations that can lead to failures. The system is open source[4].

For the development of each component of the proposed framework, several software and tools are used. As already mentioned, the proposed framework is a smart system in which each component can also be seen as a smart system. The proposed framework is implemented in Java 1.8. The main technologies used are C-SPARQL[5], OWLAPI[6] and SWRLAPI[7]. The OWLAPI is a Java API for creating, manipulating and serialising OWL Ontologies. The SWRLAPI is also a Java API for working with the SWRL language. The SWRLAPI uses the OWLAPI to manage OWL ontologies and provides a Drools-based SWRL rule engine implementation to execute SWRL rules. C-SPARQL is both a query language to process RDF streams and an engine that provides continuous query capabilities. It supports timestamped RDF triplets as input and uses a periodic execution strategy to continuously execute queries over these RDF streams. It has the capability of integrating both RDF streams and static background knowledge, also represented as RDF triplets. Given that streams are intrinsically infinite, data are usually read through time windows using the CQL window concept [72]: queries are executed on all the triplets which happen during a given time interval.

An illustrative case study is described in this section to highlight how the proposed framework can be used and the advantages it offers to support decisions that need to be made when an abnormal situation is detected in an industrial framework.

[4] https://gitlab.insa-rouen.fr/fgiustozzi/STEaMINg-SR_SitDet
[5] http://streamreasoning.org/resources/c-sparql
[6] https://github.com/owlcs/owlapi
[7] https://github.com/protegeproject/swrlapi

Table 4
Constraints definition.

| Set of constraints $\mathcal{C}$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| ID | Properties | Restriction | Device | ID | Properties | Restriction | Device |
| $c_1$ | Oil temp. | > 40°C | $M_1$ | $c_{13}$ | Power output | < 500 kW | $PL_1$ |
| $c_2$ | Oil temp. | > 60°C | $M_1$ | $c_{14}$ | Power output | < 200 kW | $PL_1$ |
| $c_3$ | Transformer temp. | > 45°C | $M_1T_1$ | $c_{15}$ | Conv. water temp. | > 60°C | $M_3Cv_1$ |
| $c_4$ | Controller temp. | > 40°C | $M_1Ct_1$ | $c_{16}$ | Conv. water temp. | > 80°C | $M_3Cv_1$ |
| $c_5$ | Generator curr. | < 800 A | $M_1G_1$ | $c_{17}$ | Trans. grid temp. | < 35°C | $M_3T_1$ |
| $c_6$ | Platform temp. | < 35°C | $PL_1$ | $c_{18}$ | Generator temp. | > 45°C | $M_3G_1$ |
| $c_7$ | Platform temp. | > 40°C | $PL_1$ | $c_{19}$ | Converter temp. | > 60°C | $M_3Cv_1$ |
| $c_8$ | Gearbox temp. | > 40°C | $M_2GB_1$ | $c_{20}$ | Converter temp. | > 80°C | $M_3Cv_1$ |
| $c_9$ | Gearbox temp. | > 60°C | $M_2GB_1$ | $c_{21}$ | Rotor speed | < 200 rpm | $M_4R_1$ |
| $c_{10}$ | Generator speed | < 500 rpm | $M_2G_1$ | $c_{22}$ | Rotor speed | < 100 rpm | $M_4R_1$ |
| $c_{11}$ | Environment temp. | < 25°C | $PL_1$ | $c_{23}$ | Rotor Pitch angle | < 5° | $M_4R_1$ |
| $c_{12}$ | Power output | > 2000 kW | $PL_1$ | | | | |

## 4.1. Case study description

The case study is based on a manufacturing production line (Figure 5), named PL1, composed of four machines: M1, M2, M3 and M4. These machines and the production line are equipped with sensors. The sensors collect data on the properties described in Table 4. This scenario is formally represented using the ontological model introduced in this paper and it is shown in Figure 6.

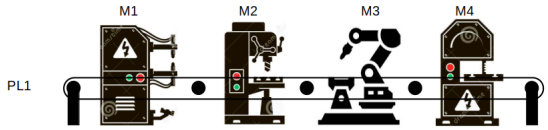


Fig. 5. Production line.

Several abnormal situations that lead to failures in this scenario are defined by experts. Each of them is expressed as a set of constraints. We focus on situations covering the following types of failures: machines malfunctions and global malfunctions of the production line including hydraulic oil leakage and cooling system filter obstructions. The abnormal situations are describe in Table 5. The situations indicate different levels of severity. For example, situations $s_1$ and $s_2$ both represent situations that could lead to the same failure but $s_2$ indicates a higher severity since the temperature threshold is higher than the temperature threshold for $s_1$. In this case, higher impact actions should be taken.

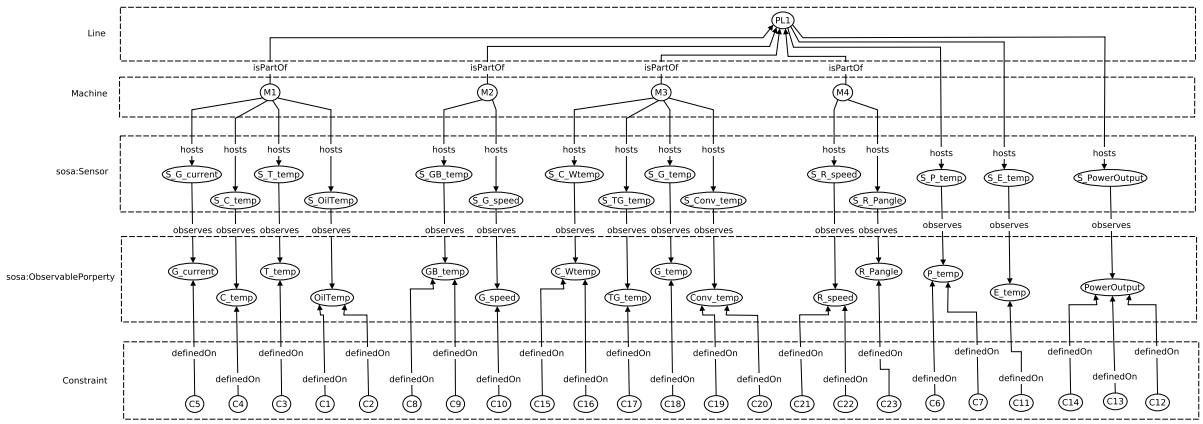


Fig. 6. Representation of the scenario using our semantic model.

Table 5
Situations and their concerned constraints (the constraints that are implied by other constraints are in bold face)

| | Set of situations $\mathcal{S}$ | |
|---|---|---|
| Sit. | Const. ($\mathcal{T}$) | Description |
| $s_1$ | $c_1$,$c_3$,$c_4$,$c_5$,$c_6$ | If the constraints are satisfied at least once in a time period of 20 seconds then $s_1$ represents an oil leak in $M_1$. |
| $s_2$ | **$c_1$**,$c_2$,$c_3$,$c_4$,$c_5$,$c_6$ | If the constraints are satisfied at least once in a time period of 20 seconds then $s_2$ represents a more severe oil leak in $M_1$. |
| $s_3$ | $c_6$,$c_8$,$c_{10}$,$c_{11}$ | If the constraints are satisfied at least once in a time period of 25 seconds then $s_3$ represents an increase of $M_2$ oil temp. |
| $s_4$ | $c_6$,**$c_8$**,$c_9$,$c_{10}$,$c_{11}$ | If the constraints are satisfied at least once in a time period of 25 seconds then $s_4$ represents a more severe increase of $M_2$ oil temp. |
| $s_5$ | $c_7$,**$c_8$**,$c_9$,$c_{10}$,$c_{11}$ | If the constraints are satisfied at least once in a time period of 25 seconds then $s_5$ represents an extreme increase $M_2$ oil temp. |
| $s_6$ | $c_{15}$,$c_{17}$,$c_{18}$ | If the constraints are satisfied at least once in a time period of 15 seconds then $s_6$ represents a filter obstruction in $M_3$. |
| $s_7$ | **$c_{15}$**,$c_{16}$,$c_{17}$,$c_{18}$ | If the constraints are satisfied at least once in a time period of 15 seconds then $s_7$ represents a nearly total filter obstruction in $M_3$. |
| $s_8$ | $c_6$,$c_{17}$,$c_{19}$ | If the constraints are satisfied at least once in a time period of 20 seconds then $s_8$ represents a slight sign of malfunction of $M_3Cv_1$. |
| $s_9$ | $c_6$,$c_{17}$,**$c_{19}$**,$c_{20}$ | If the constraints are satisfied at least once in a time period of 20 seconds then $s_9$ represents a more pronounced malfunction of $M_3Cv_1$. |
| $s_{10}$ | **$c_{15}$**,$c_{16}$,$c_{17}$,$c_{18}$,$c_{19}$ | If the constraints are satisfied at least once in a time period of 20 seconds then $s_{10}$ represents a malfunction in $M_3Cv_1$ that puts $M_3$ at risk. |
| $s_{11}$ | $c_{12}$,$c_{21}$,$c_{23}$ | If the constraints are satisfied at least once in a time period of 30 seconds then $s_{11}$ represents a $M_4R_1$ malfunction. |
| $s_{12}$ | $c_{12}$,**$c_{21}$**,$c_{22}$,$c_{23}$ | If the constraints are satisfied at least once in a time period of 30 seconds then $s_{12}$ represents a more pronounced $M_4R_1$ malfunction. |
| $s_{13}$ | $c_{13}$,$c_{21}$,$c_{23}$ | If the constraints are satisfied at least once in a time period of 35 seconds then $s_{13}$ represents a global malfunction of the PL. |
| $s_{14}$ | **$c_{13}$**,$c_{14}$,$c_{21}$,$c_{23}$ | If the constraints are satisfied at least once in a time period of 35 seconds then $s_{14}$ represents a very high risk of failure for all the PL. |

## 4.2. Abnormal situation detection and cause determination

The situations defined are detected through C-SPARQL queries. In this section, we describe a particular query and see the effect it has on the ontological model when the situation is detected.

We must emphasize that for this case study all the data streams of the properties (`ObservableProperties`) defined in Table 4 are generated using the *RDFStream* class provided by C-SPARQL to generate RDF streams. Queries can also be executed on data streams that are generated and published by other systems or users, however, it is necessary to know the structure of the RDF stream to be able to execute queries on them.

The C-SPARQL query presented in Listing 1 has the purpose of detecting the situation `S6`, defined previously. The query name is registered on line 1 and prefixes used in the query are declared on lines 2 and 3. The query is executed on RDF streams that correspond to the properties `C_Wtemp`, `TG_temp` and `G_temp` in the time frame of 15 seconds, sliding the window by 5 seconds (line 5-7). The chosen time frame is arbitrary and can be changed as desired. It produces pairs of values (line 4): the machine name (`?m`) and the production line of which it is a part of (`?pl`). In order to obtain the production line to which the machine belongs, we indicate in the query that the C-SPARQL engine must use our ontological model as background knowledge (line 8). Line 10 enables to obtain the

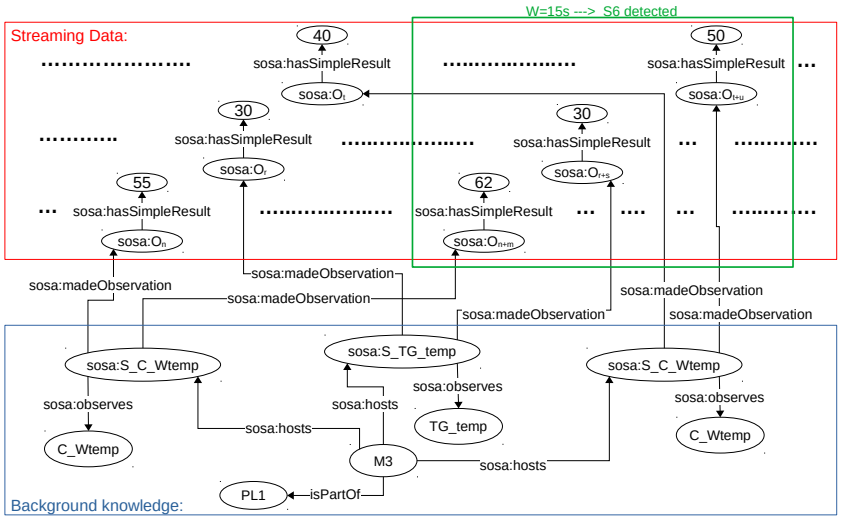


Fig. 7. Background knowledge and streaming data for S6 situation detection.

production line to which the machine belongs, as shown in Figure 7. To get the observation's values, `?o1`, `?o2` and `?o3` individuals are respectively bound with the data values `?v1`, `?v2` and `?v3` through the appropriate properties (`sosa:madeObservation` and `sosa:hasSimpleResult`) (line 11-19). Finally, the list of output pairs are filtered out to include only the ones where the observation's values satisfy the restrictions in the `FILTER` clause (line 20-23).

```
1  REGISTER QUERY S6-detection AS
2  PREFIX : <http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
3  PREFIX sosa: <http://www.w3.org/ns/sosa/>
4  SELECT ?m ?pl
5  FROM STREAM <Stream_C_Wtemp> [RANGE 15s STEPS 5s]
6  FROM STREAM <Stream_TG_temp> [RANGE 15s STEPS 5s]
7  FROM STREAM <Stream_G_temp> [RANGE 15s STEPS 5s]
8  FROM <http://semanticweb.org/STEaMINg/ContextOntology-COInd4#>
9  WHERE {
10   ?m        :isPartOf        ?pl .
11   ?m        sosa:hosts       sosa:S_C_Wtemp .
12   :S_C_Wtemp sosa:madeObservation ?o1 .
13   ?o1       sosa:hasSimpleResult ?v1 .
14   ?m        sosa:hosts       sosa:S_TG_temp .
15   :S_TG_temp sosa:madeObservation ?o2 .
16   ?o2       sosa:hasSimpleResult ?v2 .
17   ?m        sosa:hosts       sosa:S_G_temp .
18   :S_G_temp sosa:madeObservation ?o3 .
19   ?o3       sosa:hasSimpleResult ?v3 .
20 FILTER (
21   ?v1 > 60.0 &&
22   ?v2 < 35.0 &&
23   ?v3 > 45.0 ) .
24 }
```

Listing 1: C-SPARQL query to detect the S6 situation.

Once the situation is detected, it is added as an instance to the ontology as well as the relationships representing the resources concerned by this situation. Figure 8 shows a part (relevant to this case) of the ontological model before and after the detection of the `S6` situation.

The next step is to determine the possible cause(s) of the `S6` situation. The `S6` situation is associated with a failure of the cooling system of the `M3` machine. By expert knowledge we know that this situation can be caused by polluted filters in the cooling system. Therefore, the following rule is used to add this fact to the ontology. Figure 8(b) shows the added cause associated with the detected situation.

$$\texttt{Situation-S6(?sit)} \Rightarrow \texttt{hasCause(?sit,:PollutedFilters)}$$

It is worth mentioning that in case the cause is not known beforehand, it can be added to the ontological model. Thus, the next time the situation is detected the cause is automatically determined.

Once the situation and its possible cause(s) have been detected, the next step is to make a decision to avoid the failure associated with the situation. To determine what action to make, we use the approach proposed in Section

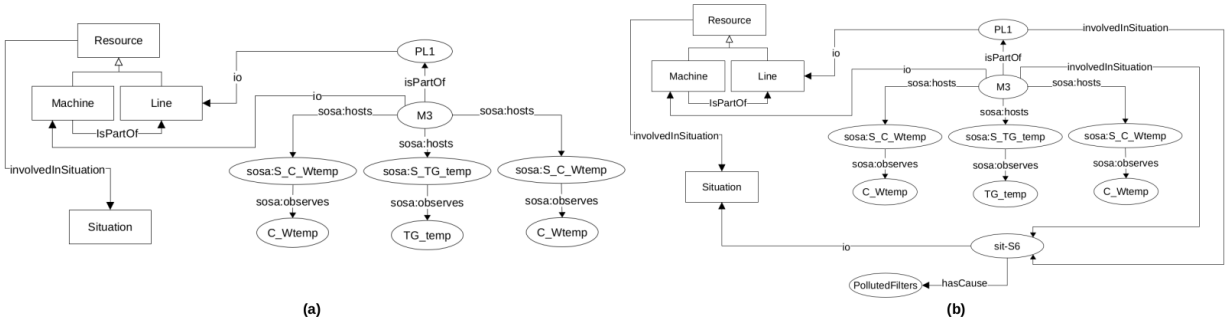


Fig. 8. Part of the ontological model: (a) before detecting the S6 situation; and (b) after detecting it.

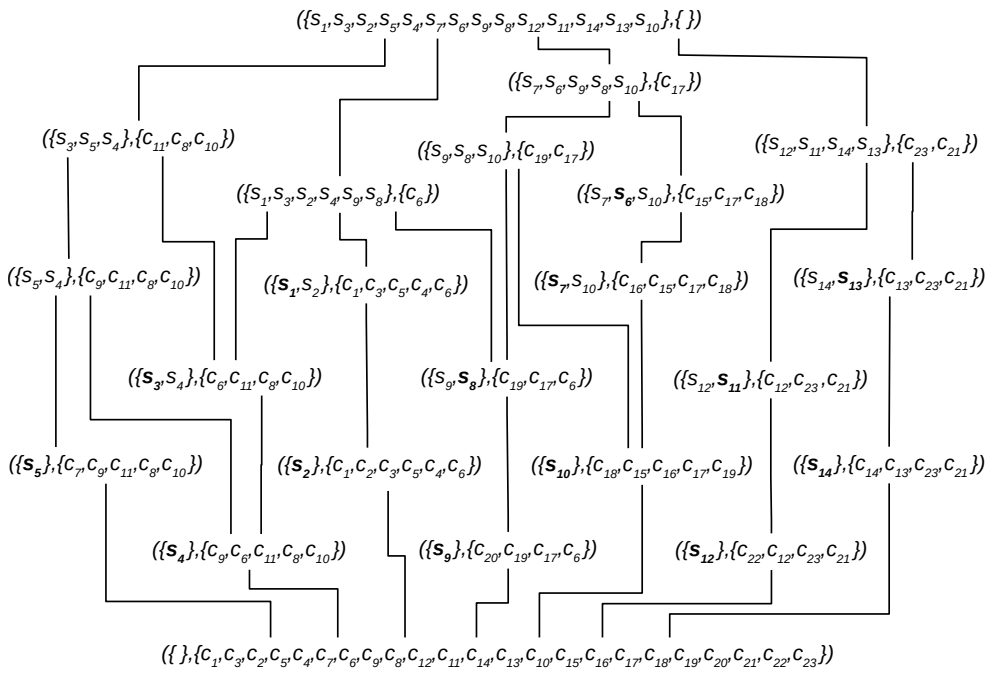


Fig. 9. Hierarchy of the situations defined in the illustrative case study.

3.4 to build the hierarchy of situations, based on the constraints on which the situations rely. The Figure 9 shows the hierarchy of situations defined in the case study.

Having the situation detected and its possible cause, the Decision making component can use both to build a strategy to correct the abnormal behavior. For example, one action could be to replace the polluted filters which would lead to the constraints associated with the `S6` situation no longer being met, *i.e.* correction of the abnormal behavior. In case this action cannot be carried out, the system or the operators can decide that the production line continues with the execution of its processes. This could lead to the `S7` situation, satisfying the `C_Wtemp` $> 80°$`C` ($c_{16}$) constraint, and thus requiring more urgent actions since the `M3` machine would be closer to a failure. By observing the hierarchy, we can see that the behavior of the `M3` machine could get even worse, since the `S10` situation could also be reached if the `Conv_temp` $> 60°$`C` ($c_{19}$) constraint is also satisfied. Unlike situation `S6` and `S7` which are associated with a failure of the `M3` machine's cooling system, situation `S10` indicates that the `M3` machine is having a global malfunction. So the strategy to follow could be more drastic as for example stopping the execution of the processes of the `M3` machine, halting also the production of the whole production line (`PL1`).

Our approach provides the Decision making component with high level information such as situations and their possible causes from raw data. In addition, the hierarchy of situations allows visualizing the possible intermediate situations that the production system can reach. In this way, it is sought that the Decision making component takes appropriate actions to maintain the reliability and availability of the production line.

### *4.3. Discussion of the proposed framework*

First, we evaluate the COInd4 ontology and then we discuss the limitations of the proposed framework for performing condition monitoring in Industry 4.0. We then discuss some of the limitations of the monitoring and diagnostic components of the proposed framework.

#### *4.3.1. COInd4 ontology evaluation*

We evaluate the ontology to check that it does not contain pitfalls and that it covers all the requirements identified in the introduction. In order to detect common mistakes done when developing ontologies we have used OOPS!

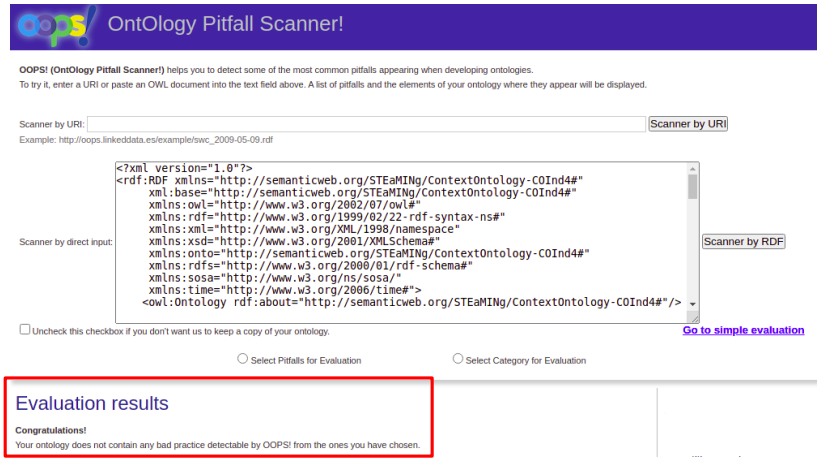


Fig. 10. Screenshot of the proposed ontology evaluation results by OOPS!

[73]. In OOPS!, ontology pitfalls are classified into three categories: structural, functional, and usability-profiling. Under each category, fine-grained classification criteria are provided to cope with specific types of mistakes. The ontology is evaluated according to the following three categories:

- **Structural dimension** focuses on mistakes detection on syntax and formal semantics.
- **Functional dimension** considers the intended use and functionality of the proposed ontology.
- **Usability-profiling dimension** evaluates the level of ease of communication when different users use the same ontology.

The evaluation of COInd4 with OOPS! has yield some minor pitfalls, that do not affect the consistency, reasoning or applicability of the ontology.

A first issue mainly regards ”missing domain or range” errors inherited from the SSN ontology. The documentation of SSN states that not adding the domain or range to certain properties is a design decision not to be restrictive with them. However, we have decided to add the missing range and domains. Another minor issue is the case of the ”recursive definitions”. In our case, it was needed to define several recursive relations, such as ”A production equipment can contain another production equipment”. Therefore, we have not considered this pitfall as a mistake. The final evaluation of the COInd4 ontology by OOPS! appears in Figure 10.

The proposed ontology is generic and extensible to cover a wide spectrum of manufacturing services. Its modular architecture also allows the description of the capabilities of manufacturing resources at different levels of modularity namely, `Machine`, `Workstation`, `Cell` and `Line`.

### *4.3.2. Evaluation of the proposed framework*

The technologies and tools used for the development of the proposed framework are introduced as well as the implementation of the core functionalities of it.

An illustrative case study from the industrial domain is presented to demonstrate the application of the framework. The goal is to show the interaction between the different components and how they modify the knowledge base, making it evolve for the representation of what happens in the real factory. This proof of concept application of our framework with simulated data is encouraging. Unfortunately, it has not been possible to validate the whole approach on real industrial data, even offline.

The main limitation of the proposed approach is that not all situations associated with failures are known in advance. Therefore, if these failures happen they are not detected. It is thus required to consult domain experts for decisions about these undetected failures. The domain experts asses the current state of the production system and provide appropriate decisions that can be further capitalized. In this way, new queries capitalizing the experts' experience need to be registered to the stream reasoner to update the initial set of queries. Thus, when in the future a similar situation needs to be detected, the updated set of queries will detect that situation. Ideally, it would also be

interesting to test our proposal in a real production line. This would allow to verify if the decisions made based on the use of our proposal would improve its efficiency and reliability.

## 5. Conclusion

This paper proposes a novel semantic framework to address the evolution of knowledge bases in Industry 4.0. The proposed framework allows to automate and facilitate condition monitoring and diagnosis, and support decision-making in the manufacturing domain. To this end, firstly we propose an ontological model for the manufacturing domain that represents the resources and processes that are part of a factory, with special emphasis on modeling the context of these resources and processes. Relevant situations that combine sensor observations with domain knowledge are also represented in the model. In this way, background knowledge from one manufacturer may be shared with another who may benefit in condition monitoring and decision making for the same type of processes and resources. The proposed framework enriches data collected from sensors with contextual information using the ontological model and uses stream reasoning to allow abnormal situation detection in real-time. Furthermore, it also uses classical reasoning approaches to compensate for possible non-detection of the causes by the stream reasoning method. Through the detection of these situations and their causes, appropriate decisions can be made to avoid the interruption of the monitored process. In addition, to support decision-making the proposed framework provides a hierarchy of situations that represents a road-map of the situations that can be reached from a given one, normal or abnormal. This allows the identification of the actions to take to correct the abnormal behavior, by avoiding or minimizing in this way the interruption of the manufacturing processes.

The contributions presented in this paper enable to identify several research perspectives. Future works can be oriented to the enrichment of the proposed semantic model. New types of relations can be explored aside from the spatial and temporal ones addressed in this work, in order to allow the representation of richer contextual information. For example, representing the relationship that two production lines execute the same process could be used to divert the production from one production line to the other if the first one is behaving abnormally. Another possibility can be the representation of relations between production lines and work orders, so that depending on whether the production line is close to finishing a work order and whether the deadline is coming up, the decision to continue the production until the order is finished or to stop the production line can be made. Another work to be completed in order to enable ontology-based interoperability is the alignment of the semantic model with top-level ontologies since several of the reused ontologies are already aligned with top-level ontologies, such as SSN with DUL[8].

Since the manufacturing domain is highly-dynamic, how to process real-time and heterogeneous data streams is a crucial concern. In order to test scalability and complexity issues to detect situations in real-time, a broader case study with more complex situations, involving more properties observations that are temporal and spatially related, has to be explored. Different implementations of stream reasoning engines should be evaluated in terms of efficiency and scalability, considering also the size of the knowledge base.

Another interesting direction to investigate is the identification of causality patterns that could enable the definition of generic queries for certain types of anomalies. These generic queries could be reused in different cases, and perform complex operations among the sensed values.

Regarding the refinement and identification of new situations, it can be interesting to apply machine learning methods to exploit the data stored in the knowledge base. As seen in this paper, the queries are executed during certain time windows every certain intervals of time, therefore the application of machine learning methods can help to determine the duration of the time windows in which a query needs to be executed or to dynamically adapt the duration of the time windows. Another possibility is the extraction of new situations from the application of machine learning methods and their integration in the semantic model. This could give rise to interesting problems such as situations more general or specific than others, contradictory situations and therefore it would be necessary to select which of the situations should remain as part of the model.

[8] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

There are also perspectives associated to the construction of the hierarchy of situations from real industrial data. Firstly, this construction also raises scalability and complexity issues to build the lattice. Therefore, tests need to be performed with different variations of Algorithm 1 to evaluate their relative efficiency. In addition, another possible future work is to investigate how to add new situations or the refined ones without having to rebuild the hierarchy from scratch.

Another perspective to explore is to create an ontological representation of the hierarchy of situations taking advantage of the fact that the situation R is already in the ontology and the relation T should be added accordingly. In this way, this ontological representation of the hierarchy could be exploited and not only be a visual road-map of the different situations that can be reached by the system.

Finally, a more exhaustive and detailed study can be made on the constraints implications that represent dependencies among constraints for the identification of new situations or the refinement of existing ones. In this way, different hierarchies of situations can be obtained according to the chosen $\mathcal{T}$ relation, that can be used alone or combined with others, allowing different strategies for decision making.

## Acknowledgements

These works are funded by the Normandy Region (France) in the framework of the STEaMINg (Semantic Time Evolving Models for Industry 4.0) project.

## References

[1] C. Santos, A. Mehrsai, A.C. Barros, M. Araújo and E. Ares, Towards Industry 4.0: an overview of European strategic roadmaps, *Procedia Manufacturing* **13** (2017), 972–979, Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain. doi:https://doi.org/10.1016/j.promfg.2017.09.093. http://www.sciencedirect.com/science/article/pii/S235197891730728X.

[2] S. Wang, J. Wan, D. Li and C. Zhang, Implementing Smart Factory of Industrie 4.0: An Outlook, *International Journal of Distributed Sensor Networks* **12**(1) (2016), 3159805. doi:10.1155/2016/3159805.

[3] H.M. Hashemian, State-of-the-Art Predictive Maintenance Techniques, *IEEE Transactions on Instrumentation and Measurement* **60**(1) (2011), 226–236.

[4] https://docs.osisoft.com/bundle/overview-of-pi-interfaces/page/pi-interfaces.html, Accessed on September 2022.

[5] A.K. Dey and G.D. Abowd, Towards a better understanding of context and context-awareness, *Proceedings of the CHI 2000 Workshop on The What, Who, Where, When and How of Context Awareness* **4** (2000), 1–6. ISBN 978-3-540-66550-2. doi:3-540-66550-1.

[6] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge Acquisition* **5**(2) (1993), 199–220. ISBN 1042-8143. doi:10.1006/knac.1993.1008.

[7] G. Stephan, H. Pascal and A. Andreas, *Knowledge Representation and Ontologies*, in: *Semantic Web Services: Concepts, Technologies, and Applications*, R. Studer, S. Grimm and A. Abecker, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 51–105. ISBN 978-3-540-70894-0. doi:10.1007/3-540-70894-$4_3$.

[8] H. Stuckenschmidt, S. Ceri, E. Della Valle, F. Harmelen and P. Milano, Towards expressive stream reasoning, *Proceedings of the Dagstuhl Seminar on Semantic Aspects of Sensor Networks* (2019).

[9] G.M. Santipantakis, A. Vlachou, C. Doulkeridis, A. Artikis, I. Kontopoulos and G.A. Vouros, A Stream Reasoning System for Maritime Monitoring, in: *25th International Symposium on Temporal Representation and Reasoning (TIME 2018)*, N. Alechina, K. Nørvåg and W. Penczek, eds, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 120, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, pp. 20:1–20:17. ISSN 1868-8969. ISBN 978-3-95977-089-7. doi:10.4230/LIPIcs.TIME.2018.20. http://drops.dagstuhl.de/opus/volltexte/2018/9785.

[10] D. Dell'Aglio, E. Della Valle, F. van Harmelen and A. Bernstein, Stream reasoning: A survey and outlook, *Data Science* **1**(1–2) (2017), 59–83.

[11] M. Bourgais, F. Giustozzi and L. Vercouter, Detecting Situations with Stream Reasoning on Health Data Obtained with IoT, *Procedia Computer Science* **192** (2021), 507–516, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021. doi:https://doi.org/10.1016/j.procs.2021.08.052. https://www.sciencedirect.com/science/article/pii/S1877050921015398.

[12] L. Stojanovic, A. Maedche, B. Motik and N. Stojanovic, User-driven ontology evolution management, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2002, pp. 285–300.

[13] L. Stojanovic, Methods and tools for ontology evolution (2004).

[14] N.F. Noy and M. Klein, Ontology evolution: Not the same as schema evolution, *Knowledge and information systems* **6**(4) (2004), 428–440.

[15] P. Plessers, O. De Troyer and S. Casteleyn, Understanding ontology evolution: A change detection approach, *Journal of Web Semantics* **5**(1) (2007), 39–49.
[16] A. Cachada, J. Barbosa, P. Leitño, C.A. Gcraldcs, L. Deusdado, J. Costa, C. Teixeira, J. Teixeira, A.H. Moreira, P.M. Moreira et al., Maintenance 4.0: Intelligent and predictive maintenance system architecture, in: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vol. 1, IEEE, 2018, pp. 139–146.
[17] K. Wang, Intelligent predictive maintenance (IPdM) system–Industry 4.0 scenario, *WIT Transactions on Engineering Sciences* **113** (2016), 259–268.
[18] D. Kwon, M.R. Hodkiewicz, J. Fan, T. Shibutani and M.G. Pecht, IoT-Based Prognostics and Systems Health Management for Industrial Applications, *IEEE Access* **4** (2016), 3659–3670. doi:10.1109/ACCESS.2016.2587754.
[19] K. Javed, R. Gouriveau and N. Zerhouni, State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels, *Mechanical Systems and Signal Processing* **94** (2017), 214–236. doi:10.1016/j.ymssp.2017.01.05.
[20] B. Samanta and K. Al-Balushi, Artificial neural network based fault diagnostics of rolling element bearings using time-domain features, *Mechanical systems and signal processing* **17**(2) (2003), 317–328.
[21] R. Benkercha and S. Moulahoum, Fault detection and diagnosis based on C4. 5 decision tree algorithm for grid connected PV system, *Solar Energy* **173** (2018), 610–634.
[22] A. Soualhi, K. Medjaher and N. Zerhouni, Bearing health monitoring based on Hilbert–Huang transform, support vector machine, and regression, *IEEE Transactions on Instrumentation and Measurement* **64**(1) (2014), 52–62.
[23] D. Guo and Z.K. Peng, Vibration analysis of a cracked rotor using Hilbert–Huang transform, *Mechanical Systems and Signal Processing* **21**(8) (2007), 3030–3041. doi:https://doi.org/10.1016/j.ymssp.2007.05.004. http://www.sciencedirect.com/science/article/pii/S0888327007000787.
[24] X.-l. Chen, P.-h. Wang, Y.-s. Hao and M. Zhao, Evidential KNN-based condition monitoring and early warning method with applications in power plant, *Neurocomputing* **315** (2018), 18–32.
[25] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang and R.X. Gao, Deep learning and its applications to machine health monitoring, *Mechanical Systems and Signal Processing* **115** (2019), 213–237.
[26] T. Zhang, J. Chen, F. Li, K. Zhang, H. Lv, S. He and E. Xu, Intelligent fault diagnosis of machines with small & imbalanced data: A state-of-the-art review and possible extensions, *ISA Transactions* **119** (2022), 152–171. doi:https://doi.org/10.1016/j.isatra.2021.02.042. https://www.sciencedirect.com/science/article/pii/S0019057821001257.
[27] S. Baltazar, C. Li, H. Daniel and J.V. de Oliveira, A review on neurocomputing based wind turbines fault diagnosis and prognosis, in: *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, IEEE, 2018, pp. 437–443.
[28] S. Zhang, S. Zhang, B. Wang and T.G. Habetler, Machine learning and deep learning algorithms for bearing fault diagnostics-a comprehensive review, *arXiv preprint arXiv:1901.08247* (2019).
[29] R. Liu, B. Yang, E. Zio and X. Chen, Artificial intelligence for fault diagnosis of rotating machinery: A review, *Mechanical Systems and Signal Processing* **108** (2018), 33–47.
[30] A.G. Nath, S.S. Udmale and S.K. Singh, Role of artificial intelligence in rotor fault diagnosis: A comprehensive review, *Artificial Intelligence Review* **54**(4) (2021), 2609–2668.
[31] D. Corrêa, A. Polpo, M. Small, S. Srikanth, K. Hollins and M. Hodkiewicz, Data-driven approach for labelling process plant event data, *International Journal of Prognostics and Health Management* **13**(1) (2022). doi:10.36001/ijphm.2022.v13i1.3045.
[32] A. Theissler, J. Pérez-Velázquez, M. Kettelgerdes and G. Elger, Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry, *Reliability Engineering & System Safety* **215** (2021), 107864. doi:https://doi.org/10.1016/j.ress.2021.107864. https://www.sciencedirect.com/science/article/pii/S0951832021003835.
[33] Y. Peng, M. Dong and M.J. Zuo, Current status of machine prognostics in condition-based maintenance: a review, *The International Journal of Advanced Manufacturing Technology* **50**(1–4) (2010), 297–313.
[34] D.L. Nuñez and M. Borsato, OntoProg: An ontology-based model for implementing Prognostics Health Management in mechanical machines, *Advanced Engineering Informatics* **38** (2018), 746–759.
[35] M. Gul and E. Celik, Fuzzy rule-based Fine–Kinney risk assessment approach for rail transportation systems, *Human and Ecological Risk Assessment: An International Journal* **24**(7) (2018), 1786–1812.
[36] E. Kharlamov, G. Mehdi, O. Savković, G. Xiao, E.G. Kalaycı and M. Roshchin, Semantically-enhanced rule-based diagnostics for industrial Internet of Things: The SDRL language and case study for Siemens trains and turbines, *Journal of Web Semantics* **56** (2019), 11–29.
[37] T. Berredjem and M. Benidir, Bearing faults diagnosis using fuzzy expert system relying on an improved range overlaps and similarity method, *Expert Systems with Applications* **108** (2018), 134–142.
[38] Q. Cao, A. Samet, C. Zanni-Merk, F. De Bertrand de Beuvron and C. Reich, Combining chronicle mining and semantics for predictive maintenance in manufacturing processes, *Semantic Web* **11** (2020), 927–948. doi:10.3233/SW-200406.
[39] M. Schlechtingen and I.F. Santos, Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 2: Application examples, *Applied Soft Computing* **14** (2014), 447–460. doi:https://doi.org/10.1016/j.asoc.2013.09.016. http://www.sciencedirect.com/science/article/pii/S1568494613003104.
[40] M. Schlechtingen, I.F. Santos and S. Achiche, Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 1: System description, *Applied Soft Computing* **13**(1) (2013), 259–270. doi:https://doi.org/10.1016/j.asoc.2012.08.033. http://www.sciencedirect.com/science/article/pii/S1568494612003821.
[41] B. Schmidt, L. Wang and D. Galar, Semantic framework for predictive maintenance in a cloud environment, in: *10th CIRP Conference on Intelligent Computation in Manufacturing Engineering, CIRP ICME'16, Ischia, Italy, 20-22 July 2016*, Vol. 62, Elsevier, pp. 583–588.

[42] F. Xu, X. Liu, W. Chen, C. Zhou and B. Cao, Ontology-based method for fault diagnosis of loaders, *Sensors* **18**(3) (2018), 729.
[43] P. Papadopoulos and L. Cipcigan, Wind turbines' condition monitoring: an ontology model, in: *2009 International Conference on Sustainable Power Generation and Supply*, IEEE, 2009, pp. 1–4.
[44] J. Schwarzenbach, L. Wilkinson, M. West and M. Pilling, Mapping the remote condition monitoring architecture, *Research Programme. Rail Safety and Standards Boards (RSSB) LTD. RSSB Core Report* (2010).
[45] P. Martin and A. D'Acunto, Design of a production system: an application of integration product-process, *International Journal of Computer Integrated Manufacturing* **16**(7–8) (2003), 509–516.
[46] H. Panetto, M. Dassisti and A. Tursi, ONTO-PDM: Product-driven ONTOlogy for Product Data Management interoperability within manufacturing process environment, *Advanced Engineering Informatics* **26**(2) (2012), 334–348.
[47] R. Barbau, S. Krima, S. Rachuri, A. Narayanan, X. Fiorentini, S. Foufou and R.D. Sriram, OntoSTEP: Enriching product model data using ontologies, *CAD Computer Aided Design* (2012). doi:10.1016/j.cad.2012.01.008.
[48] Z. Usman, R.I.M. Young, N. Chungoora, C. Palmer, K. Case and J. Harding, A manufacturing core concepts ontology for product lifecycle interoperability, in: *Lecture Notes in Business Information Processing*, 2011. ISSN 18651348. ISBN 9783642196799. doi:10.1007/978-3-642-19680-5$_3$.
[49] M. Grüninger, Using the PSL Ontology, in: *Handbook on Ontologies*, 2009. doi:10.1007/978-3-540-92673-3$_1$9.
[50] S. Lemaignan, A. Siadat, J.-Y. Dantan and A. Semenenko, MASON: A proposal for an ontology of manufacturing domain **2006** (2006), 195–200. ISBN 0-7695-2589-X.
[51] S. Borgo and P. Leitão, Foundations for a Core Ontology of Manufacturing, Vol. 14, 2007, pp. 751–775.
[52] D. Šormaz and A. Sarkar, SIMPM – Upper-level ontology for manufacturing process plan network generation, *Robotics and Computer-Integrated Manufacturing* (2019). doi:10.1016/j.rcim.2018.04.002.
[53] M. Garetti and L. Fumagalli, P-PSO ontology for manufacturing systems, in: *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2012. ISSN 14746670. ISBN 9783902661982. doi:10.3182/20120523-3-RO-2023.00222.
[54] F. Ameri and D. Dutta, An upper ontology for manufacturing service description, in: *Proceedings of the ASME Design Engineering Technical Conference*, 2006. ISBN 079183784X. doi:10.1115/detc2006-99600.
[55] E. Järvenpää, N. Siltala, O. Hylli and M. Lanz, The development of an ontology for describing the capabilities of manufacturing resources, *Journal of Intelligent Manufacturing* (2019). doi:10.1007/s10845-018-1427-6.
[56] H. Cheng, P. Zeng, L. Xue, Z. Shi, P. Wang and H. Yu, Manufacturing Ontology Development Based on Industry 4.0 Demonstration Production Line, 2016, pp. 42–47. doi:10.1109/TSA.2016.17.
[57] A. Sheth, C. Henson and S.S. Sahoo, Semantic Sensor Web, *IEEE Internet Computing* **12**(4) (2008), 78–83.
[58] A. Haller, K. Janowicz, S.J. Cox, M. Lefrançois, K. Taylor, D. Le Phuoc, J. Lieberman, R. García-Castro, R. Atkinson and C. Stadler, The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation, *Semantic Web* **10**(1) (2019), 9–32.
[59] M. de Roode, A. Fernández-Izquierdo, L. Daniele, M. Poveda-Villalón and R. García-Castro, SAREF4INMA: a SAREF extension for the Industry and Manufacturing domain (2019).
[60] M. Uschold and M. Grüninger, Ontologies: Principles, methods and applications, *The Knowledge Engineering Review* **11** (1996).
[61] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi and P.F. Patel-Schneider (eds), *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, USA, 2003. ISBN 0521781760.
[62] M. Perry and J. Herring, OGC GeoSPARQL - a geographic query language for RDF data, *OGC Implementation Standard* (Sept, 2012).
[63] D.A. Randell, Z. Cui and A.G. Cohn, A Spatial Logic based on Regions and Connection, *3rd International Conference On Knowledge Representation And Reasoning* (1992), 165–176. ISBN 9781558602625. doi:10.1.1.35.7809.
[64] M.J. O'Connor and A.K. Das, A Method for Representing and Querying Temporal Information in OWL, in: *Biomedical Engineering Systems and Technologies*, A. Fred, J. Filipe and H. Gamboa, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 97–110. ISBN 978-3-642-18472-7.
[65] B.P. Allen, Case-based reasoning: Business applications, *Communications of the ACM* **37**(3) (1994), 40–43.
[66] F. Baader, I. Horrocks and U. Sattler, *Description Logics as Ontology Languages for the Semantic Web*, in: *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, D. Hutter and W. Stephan, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 228–248. ISBN 978-3-540-32254-2. doi:10.1007/978-3-540-32254-2$_1$4.
[67] G. Schreiber, H. Akkermans, A. Anjewierden, R.D. Hoog, N.R. Shadbolt and B. Wielinga, *Knowledge Engineering and Management: The CommonKADS Methodology*, Vol. 99, 2000, p. 455. ISSN 09333657. ISBN 0262193000. doi:10.1016/S0933-3657(01)00090-2. http://eprints.ecs.soton.ac.uk/2270/.
[68] J.B. Nation, Notes on lattice theory, Citeseer, 1998.
[69] H.A. Davey B. A. Priestley, *Introduction to Lattices and Order*, 2nd edn, Cambridge University Press, 2002. doi:10.1017/CBO9780511809088.
[70] G. Gratzer, *Lattice Theory: Foundation*, 2011. ISBN 978-3-0348-0017-4. doi:10.1007/978-3-0348-0018-1.
[71] F. Giustozzi, J. Saunier and C. Zanni-Merk, Towards the use of Situation Hierarchies for supporting Decision Making: A Formal Lattice-Based Approach, in: *In Proceedings of the 14th International Rule Challenge (RuleML+ RR 2020) as part of Declarative AI 2020*, Vol. 2644, 2020, pp. 73–86.
[72] A. Arasu, S. Babu and J. Widom, CQL: A Language for Continuous Queries over Streams and Relations, in: *Database Programming Languages*, G. Lausen and D. Suciu, eds, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 1–19. ISBN 978-3-540-24607-7.
[73] M. Poveda-Villalón, A. Gómez-Pérez and M.C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation, *International Journal on Semantic Web and Information Systems (IJSWIS)* **10**(2) (2014), 7–34.