Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning

Bram Steenwinckel [a,*], Dieter De Paepe [a], Sander Vanden Hautte [a], Pieter Heyvaert [a], Mohamed Bentefrit [b], Pieter Moens [a], Anastasia Dimou [a], Bruno Van Den Bossche [b], Filip De Turck [a], Sofie Van Hoecke [a], Femke Ongenae [a]

[a] IDLab, Ghent University-imec, Technologiepark 126, 9052 Zwijnaarde, Belgium
[b] Televic Rail NV, Leo Bekaertlaan 1, 8870 Izegem, Belgium

A R T I C L E   I N F O

Article history:
Received 23 December 2019
Received in revised form 11 September 2020
Accepted 15 October 2020
Available online 19 October 2020

Keywords:
Anomaly detection
Root cause analysis
Machine learning
Semantic web
Internet of Things
Fused AI
User feedback

A B S T R A C T

Anomalies and faults can be detected, and their causes verified, using both data-driven and knowledge-driven techniques. Data-driven techniques can adapt their internal functioning based on the raw input data but fail to explain the manifestation of any detection. Knowledge-driven techniques inherently deliver the cause of the faults that were detected but require too much human effort to set up. In this paper, we introduce FLAGS, the Fused-AI interpretabLe Anomaly Generation System, and combine both techniques in one methodology to overcome their limitations and optimize them based on limited user feedback. Semantic knowledge is incorporated in a machine learning technique to enhance expressivity. At the same time, feedback about the faults and anomalies that occurred is provided as input to increase adaptiveness using semantic rule mining methods. This new methodology is evaluated on a predictive maintenance case for trains. We show that our method reduces their downtime and provides more insight into frequently occurring problems.

1. Introduction

Sensor monitoring systems are transforming the industry, steered by the so-called Internet of Things (IoT) and Artificial Intelligence (AI) fields, through game-changing applications in, e.g., transportation [1], security [2], ventilation [3] and health-care [4]. For example, in the railway domain the number of sensors deployed on a single train bogie ranges from 10 to 50. A wide variety of sensors are used, such as accelerometers to monitor vibrations, ultrasonic, inductive and draw-wire range sensors, shock pulse sensors or gyroscopes for rotational speed. Monitoring these sensors can deliver valuable insights into the physical assets, the performance, and the interaction with the environment. For example, a bogie monitoring system can be used to assess the state of the wheel bearings, fatigue in the bogie, driving comfort for the passengers and the train body tilting.

Sensor monitoring systems analyze so-called sensor networks and can be used to detect faulty or deviating system behavior using methodologies such as Anomaly Detection (AD), Fault Recognition (FR) and Root Cause Analysis (RCA).

In the area of sensor networks and streaming data, AD consists of finding unknown patterns or outliers in unlabeled data when something unusual occurs or when the conditions deviate from the normal behavior [5]. FR captures the stronger patterns as the condition develops, and the system's operation deteriorates towards failure. Once a pattern for a specific fault has been identified, it can be referenced in the future when the pattern emerges again. This approach can be used to explain what is currently going wrong [6].

RCA is the process of deducing and understanding the underlying cause of the occurring anomalies or faults [7].

Combined, AD, FR and RCA allow end-users to accurately pinpoint problems, mediate them and prevent further escalations. For example, proprietary analysis tools are used by train maintenance personnel to analyze the accelerometer data from bogie-mounted sensors and to identify possible wheel issues.

Two main techniques exist to perform AD, FR and RCA in sensor networks: those that are data-driven and those that are knowledge-driven. The first technique derives anomalies or recognizes faults directly from the streaming data by identifying unusual patterns using machine learning (ML). The second technique encodes expert knowledge about the systems, e.g., expected sensor ranges during normal behavior, to detect known or unknown behavior in the data streams.

* Corresponding author.
E-mail address: bram.steenwinckel@ugent.be (B. Steenwinckel).

As discussed in detail in Section 3, both have limitations. The knowledge-driven techniques are highly interpretable, context-aware and have a low false-positive rate. However, they cannot update new fault- or anomaly-related knowledge on the fly. They are also unable to learn new anomalies automatically and require much human effort to construct and maintain. When used in sensor monitoring systems, data-driven techniques are adaptive and require minimal human effort to start analyzing a data stream. Nevertheless, the data-driven techniques lead to many false positives as they are not context-aware and are often uninterpretable.

The drawbacks of both techniques cancel each other out. The fuse of both allows extracting interpretable alerts in highly dynamic environments with a reduction in human involvement. However, developing an efficient methodology that combines both data- and knowledge-driven techniques remains a considerable challenge.

Therefore, the objective of this paper is the introduction of a methodology to efficiently combine data- and knowledge-driven techniques towards optimizing AD, FR and RCA for sensor monitoring systems. This fused methodology tackles the above mentioned drawbacks of current AD, FR and RCA systems. A prototype implementation of this methodology, called FLAGS (Fused-AI interpretabLe Anomaly Generation System), is also presented. FLAGS is evaluated on a predictive maintenance use case to illustrate its benefits, i.e., decreased number of falsely generated alerts, almost no human involvement needed to adapt towards new environments and providing interpretable causes for the occurred anomalies.

Our approach makes the following contributions:

- The output of both the data- and knowledge-driven techniques are combined to (i) pinpoint the unwanted behavior and (ii) reduce the number of falsely generated and missed alerts when compared to the data-driven techniques, while (iii) requiring less human involvement than the knowledge-driven techniques.
- Expert knowledge is fused into data-driven techniques, resulting in RCA algorithms that deliver the most likely causes of the anomalies or derive interpretations to start data-driven FR.
- AD, FR & RCA are combined and take the available context into account to readily adapt to new contexts of deployment and configurations, while reducing the amount of required data about this new context.
- Dashboard applications can gather non-intrusive user feedback on the detected faults, outliers and their causes. Here, user feedback is crucial for dealing with the continually changing or unknown deployment environments and enables the optimization of AD, FR, and RCA algorithms. This feedback is used by many components to improve the whole monitoring system automatically.

The remainder of the paper is structured as follows. Section 2 gives an overview of the requirements needed to fulfill a fused AD, FR and RCA methodology. Section 3 gives an overview of the two most common techniques to perform AD, FR and RCA and describes why not all of the requirements mentioned in the previous section can be met. The proposed end-to-end methodology for the derivation of highly accurate anomalies and their interpretable causes from sensor monitoring streams through adaptive and context-aware AD and RCA is discussed in Section 4. The methodology itself is evaluated using a train bogie monitoring use case in Section 5. Section 6 shows the results in terms of efficiency and performance of the FLAGS methodology based on the train monitoring use case. Section 7 discusses the methodology with respect to the requirements while Section 8 summarizes the approach and lists future work.

2. Requirements

Several requirements should be met to analyze anomalies, faults and determine their causes in a sensor network.

Sensor monitoring systems are deployed on a plethora of devices, with various configurations and within varying contexts. This makes AD, FR & RCA a challenging task. No upfront knowledge is available about the actual configuration or the complex environments or domain they are being deployed. However, knowledge about this deployment environment and used configuration can have a severe impact on whether an anomaly or fault has been occurred. For example, bogie monitoring systems are deployed on various types of trains, driving on a wide variety of tracks all over the world. The threshold indicating whether a temperature observation is anomalous varies widely between these locations.

Moreover, knowledge also influences the assessment of these anomalies and provides more information about a possible cause. For example, when multiple trains report vibration anomalies at the same location, the track is probably faulty and not the trains.

The knowledge about the anomaly or fault, together with the raw sensor data, gives greater interpretability to those who follow up on the problem posed. For example, maintenance can be requested faster when a temperature sensor of the wheel axle reports high values given the context that the train is riding through Siberia during the winter months.

As more information is provided to the end-users or operators, their actions on the detected anomalies and faults provide useful information for future investigations. As sensor environments can change rapidly, the sensor monitoring systems must be updated regularly with this new information to guarantee the correct functioning of the monitoring units. In our example, whether or not the operator requests maintenance for a particular detected anomaly or fault can indicate the urgency of a detected anomaly. However, a lot of human involvement is required when the operator has to label all the data points corresponding to the occurring event. These tasks are even seen as useless to the operator when no direct improvement is notified while relabeling.

Fusing knowledge into data-driven techniques can improve the detection rate of interesting anomalies. In the previous example, incorporating knowledge about the external temperature would improve the detection rate of wheel axle faults or anomalies. Analysis of a plethora of industry defined use cases for AD, such as the one outlined above, have led to the following requirements:

- **Precise:** The detection of anomalies should be accurate to reduce the number of false positives, which usually overwhelms the operator with annoying alerts. These alerts now lead to less timely interventions, increased stress and less detailed investigations of real problems. On the other hand, accurately detecting anomalies should also reduce the number of false negatives, as this lead to undetected problems that can escalate. In critical domains, the false positive rate should be lower than 70% to reduce the current burden and stress of the operators trying to resolve them [8].
- **Require minimal human involvement:** Human involvement should be minimized to deal with the lack of continuous access to the deployment environment. Industry partners state that the operators should be able to steer the functioning of the monitoring unit by providing simple feedback on the system's outcomes for more than 50% of the time.
- **Context-aware:** Providing context to data-driven detections results in the reduction of false positives as more information is available to discriminate wrong from correct behavior. Industry partners estimated that context incorporation

is only beneficial when the reduction in false positives can be noticed.

- **Adaptive:** An effective AD, FR & RCA sensor monitoring system should be capable of adapting the detection behavior to changing conditions in the deployment environment or system configuration while still recognizing anomalous or faulty activities. If not, the system is either not operable in a streaming environment or large numbers of fault positives will be generated due to the lack of adaptability [9]. The system should be able to adapt its detection behavior in less than 10 s after the operator or end-user has indicated changes in behavior. This upper limit is required to avoid losing the users' attention completely [10].
- **Interpretable:** The detection model is highly interpretable if end-users, i.e., operators, are able to quickly plan the appropriate mediation actions for the detected anomalies. Interpretable results should increase the efficiency of an operator's intervention effectiveness with more than 15% [11].

To adhere to all requirements, on the one hand, we need to deliver cost-effective and value-adding AD, FR & RCA techniques. On the other hand, these techniques must be combined and made to reinforce each other. By doing this, the human involvement can be reduced which is required to tune them for long-term tasks.

3. Related work

This section gives an overview of the two current techniques, i.e., data- & knowledge-driven, to perform AD, FR & RCA in the context of sensor networks. We compare both approaches with respect to the requirements of Section 2.

3.1. Knowledge-driven FR & RCA

When knowledge is provided by field experts to describe a particular system problem, such a problem is referred to as known faults. Knowledge-driven FR & RCA methodologies consist of two steps, as shown in Fig. 1: knowledge acquisition and knowledge transformation. The first aims to capture the existing guidelines and knowledge of domain experts on the expected normal behavior of a certain system or the faults that can occur with their possible underlying causes. During the second step, this information is transformed into software, e.g., rules, that can extract insights from the incoming data. These insights are the description of the detected faults and their causes.

Knowledge acquisition can be performed through risk analysis. Two types of risk analysis are prevalent, i.e. Failure Mode and Effects Analysis (FMEA) [12] and Fault Tree Analysis (FTA) [13]. Both are visualized in Fig. 1(a) and (b), respectively. FMEA is an inductive technique[1] that captures the potential failures in the system components, together with their underlying causes and effects. By contrast, FTA is a deductive technique that uses Boolean logic to analyze the undesired states of a system to see which lower-level event caused it.

Constructing these FMEA and FTA documents is a time-consuming process as many experts are involved. As each of them has expertise in other parts of the system, they interpret the risk analysis differently. All these different interpretations result in ambiguities, inconsistencies and duplicates. Moreover, no links between the specified anomalies, causes, the expected system behavior and the contextual deployment are defined within these documents. Without such links, it is rather hard to get a clear overview of the system's functioning.

Ontology-based risk analysis methods have been designed to overcome these problems [14]. They provide high-level ontologies, e.g., the FOLIO ontologies,[2] and rules that allow to semantically model the expert knowledge from FMEA & FTA analyses. By employing the Linked Data approach, this expert knowledge can then easily be related to knowledge on the incoming data by enriching these streams through domain & system ontologies, e.g. the Semantic Sensor Network (SSN) ontology [15]. Combining SSN and FOLIO enables the consolidation of the data streams and makes the device properties and the gathered context explicit. However, ontology engineers are required to improve and update these models with new domain knowledge constantly. As system experts are not familiar with ontology design, methods have been investigated to transform the existing FMEA tables, FTA trees or even free text automatically to ontological models and inference rules [16].

When the constructed ontologies and rules have been generated, they can be used to annotate incoming raw data. The appropriate metadata is used to link them to the available background knowledge, as shown in the bottom part of Fig. 1. This results in a so-called Knowledge Graph (KG), where the data is linked with the domain metadata. Commonly available semantic reasoners, e.g., Hermit [17] and Pellet [18], can be used to interpret this semantic data to detect possible faults and infer causes using the ontology and rules.

In sensor networks, reasoning on a stream of semantic data is more commonly known as Semantic Complex Event Processing (SCEP) [19]. The available rules can describe the semantic complex event, which in our case signifies a fault. A full comparison of all available SCEP systems is out of scope for this paper. There are already exist systems which can process complex events over high-velocity streams consisting of up to hundreds of events per second [20].

As these models are interpretable, they can explain how they derive conclusions or offer more insight for the end-user. Moreover, the in-depth knowledge of the domain experts, their evidence and the carefully curated and maintained guidelines lead to a low number of false positives. No data on normal system behavior is required to build the FR and RCA system. By explicitly linking the defined faults and causes to the system properties and the context in which they are valid, the operating systems can readily adapt to other known contexts and deployments.

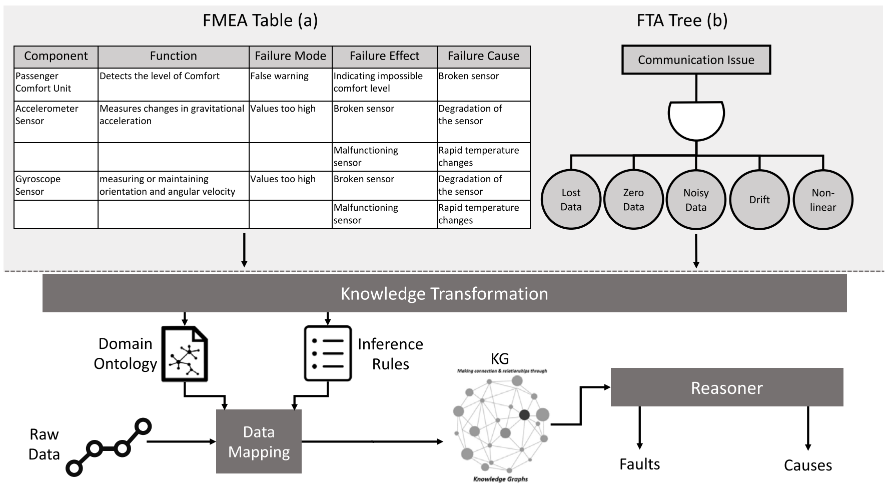
This methodology also has its limitations. The logic models can range from relatively simple rules, e.g., setting thresholds on measured parameters and statistically derived features, to highly complex, e.g., intricate ontologies with associated rule sets. A profound understanding of the domain and a large amount of human effort are required to construct and maintain these ontologies and rules. Moreover, these systems cannot learn new anomalous behavior without providing new knowledge manually. They are unable to adapt automatically to dynamic environments or previously unknown contexts. Not being able to adapt the detection behavior leads to undetected anomalies and the constant need for human involvement in changing the logical model based on new expert insights, their know-how and their efforts.

3.2. Data-driven AD & RCA

Data-driven AD and RCA employ ML to directly learn a model from the data collected by the sensor monitoring systems [21]. A preprocessing step can be required to transform the data into a set or vector of useful features. The overview in Fig. 2 shows how

1 An inductive method starts with many observations, to find a few, powerful statements. It is the opposite of a deductive technique, which starts with a few true statements (axioms) to prove many true statements (theorems) that logically follow from them.

2 https://github.com/IBCNServices/Folio-Ontology.



Component	Function	Failure Mode	Failure Effect	Failure Cause
Passenger Comfort Unit	Detects the level of Comfort	False warning	Indicating impossible comfort level	Broken sensor
Accelerometer Sensor	Measures changes in gravitational acceleration	Values too high	Broken sensor	Degradation of the sensor
			Malfunctioning sensor	Rapid temperature changes
Gyroscope Sensor	measuring or maintaining orientation and angular velocity	Values too high	Broken sensor	Degradation of the sensor
			Malfunctioning sensor	Rapid temperature changes

Fig. 1. The knowledge-driven AD & RCA methodology. The top part illustrates an FMEA table (a) and FTA tree (b). The lower part shows how this knowledge from both risk documents and the raw sensor data is incorporated through a mapping script into a KG. At last, a rule-based reasoner can be performed to detect faulty behavior with the accompanying cause.

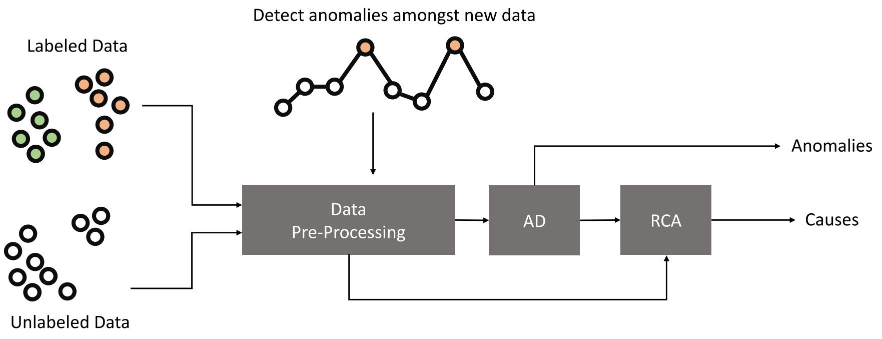


Fig. 2. The data-driven AD & RCA methodology. After prepossessing either labeled or unlabeled data, the corresponding features can be given to the AD module which detects anomalies. The RCA module takes both the features and anomalies to generate causes.

the raw data must be first preprocessed before both the AD and RCA modules can use it.

Specialized ML techniques are required to perform AD in the context of sensor networks. As most IoT systems provide streaming sensor data, methods are required to cope with time series data either directly or by preprocessing the time series to discrete events [22]. Anomalies occur infrequently and can be either uni- or multivariate. Techniques are needed that can cope with such inherent imbalance. For multivariate anomalies, multiple sensor streams need to be investigated simultaneously to make a correct decision. Based on these properties, three general categories of ML-based AD can be discerned, i.e. supervised, unsupervised and semi-supervised [23].

Supervised AD requires data where the instances have been labeled as normal or anomalous. Note that more than one normal or anomalous class can exist. Traditional classification techniques can be used by leveraging methods to deal with the inherent imbalance of the data set. Common methods are undersampling of the normal data, oversampling of the anomalous data or cost-sensitive learning punishing the misclassification of anomalies harder than that of normal samples. Supervised AD also assumes that new anomalies will be similar to past ones. Popular supervised AD methods that can deal with time-series data are Recurrent Neural Networks (RNN, e.g., LSTM) and Support Vector Machines (SVM) [24].

In many sensor streaming situations, class labels are not available due to the relatively large number of samples needed to train the model. Unsupervised AD techniques try to detect outliers by assuming that most of the data is normal behavior. Here, the objective is to assign a score (or a label) to each instance that reflects its degree of normalcy. For unsupervised AD to be successful, anomalies must be distinct from one another, as well as from the normal behavior. A prevalent methodology for performing unsupervised AD on time series is discovering motifs and discords by, e.g., Matrix Profiling [25] or HOT SAX [26]. Motifs are the best matching subsequences in time series, while discords are subsequences that maximally differ from the other ones, i.e., the anomalies. Autoregressive models (e.g., ARMA, ARIMA and VAR), isolation forests and clustering-based techniques (e.g., DBSCAN) are also popular [27].

Finally, datasets will sometimes only contain data that is labeled as normal. Semi-supervised AD methods construct a model representing the normal behavior from a given training data set with only positive labels and test the likelihood of a new sample to be generated by the learned model. The presence of many related outliers in the set of objects to be scored does not impact the model's evaluation. Popular methods for semi-supervised AD which can deal with time series are clustering algorithms [28].

The application of RCA for sensor systems and the IoT is rather under-explored in contrast to AD [29]. Different ML techniques for RCA are discussed in the literature [30–32]. Particularly association rule mining (ARM) and Bayesian networks are well-recognized data mining techniques for knowledge discovery and exploring relations between failure events embedded in large datasets. Techniques from the field of statistical process control, such as contribution plots based on principal component analysis (PCA) or partial least squares, can also be used for RCA.

Several advantages of these data-driven AD and RCA techniques can be noted. As they learn directly from the data, little human intervention is required. Moreover, they are able to discover new anomalies in the provided data that are not known to the domain experts yet.

Despite these advantages, huge amounts of data are required to learn the normal behavior accurately and distinguish it from the anomalies. Providing labeled data will have the advantage that several models can be trained or developed in parallel. When not available, more human interventions will be required to check whether or not the anomalies are of interest. Both approaches entail a huge effort to collect this data and correctly label it or provide useful feedback afterward. These data sets are hard to come by. ML-based AD in sensor streams is often trained only once on a limited, labeled dataset collected from the environment and deployed in various contexts and configurations. A limited data set can result in many false positives, mainly due to patterns that were not detected in the past. The problem of dealing with such missing patterns can be reduced by grouping the patterns together based on similar feature values [33]. However, this technique only reduces the false positives for those patterns that are available in the provided dataset and not for the new undetected ones.

Data-driven AD and RCA can take into account the environment they are operating in and know how to optimize their performance to relate the system components to each other [34,35]. However, this kind of context is limited to some additional features and makes it difficult to adapt them to new types of sources and environments. Many false positives are prone to overwhelm the user with redundant notifications, and anomalies that go undetected (false negatives) occur.

The lack of domain and background knowledge also makes it difficult to accurately pinpoint the real causes of anomalies. The best performing and most expressive ML-based AD methods for analyzing sensor streams are black-box and, therefore, not interpretable. This leaves the operators guessing why an alert was raised as no interpretation can be given by the model. Recent advances in eXplainable AI (XAI) have made it possible to give some interpretations of the trained models' outcome. Techniques such as Shapley values show which features are important for our ML model or on which parts of the input a decision is based [36]. However, such feature importance methods give only a small amount of interpretability and correlate features based on the given data. Moreover, the applicability of XAI is rather limited in streaming environments [37].

3.3. Comparison regarding the requirements

Both the data-driven and knowledge-driven AD, FR & RCA techniques can be summarized regarding the requirements listed in Section 2. As can be seen in Table 1, none of these two techniques met all stated requirements. As discussed above, the data-driven techniques adapt to new, unseen anomalies and applying such a model in the context of streaming data requires minimal effort. The incorporation of additional features makes these techniques contextual, but they cannot guarantee to fully incorporate the system or operating environment dynamics based on the data solely. By contrast, knowledge-driven techniques do include the inherent dynamics and provide an interpretable and precise FR mechanism. As a counterpart, new expert knowledge must be incorporated manually, which requires human expertise. Such human involvements make these techniques less attractive to operate in rapidly changing environments such as sensor networks.

From Table 1, it can be seen that combining both approaches could resolve the stated drawbacks. An efficient way to fuse both the data- and knowledge-driven AD, FR & RCA techniques for sensor streaming data in one methodology is proposed in Section 4.

Table 1
Overview of the data- and knowledge-driven techniques in comparison with the requirements of Section 2.

	Data-driven	Knowledge-driven
Precise		X
Minimal human involvement	X	
Context-aware	(X)	X
Adaptive	X	
Interpretable		X

3.4. Ad, FD & RCA within predictive maintenance

While the requirements are defined in the general context of AD, FD & RCA, they also impose several challenges within the predictive maintenance and IoT domain [38,39].

Adequate information about possible failures is challenging to acquire, as most system failures are rare or vary a lot for different types of systems and equipment [40]. For the data-driven paradigm, it is more common for models to be trained on the sensor data of one single machine instead of a more global approach as this is less effective. Predictive maintenance is therefore limited to health monitoring tasks, where they identify machine failures based on the deviations from the healthy machine or system data [41]. These health monitoring tasks are almost always unsupervised due to the unavailability of labeled data. Despite their effectiveness in both accurately finding anomalies and efficient usage of the available computational resources, still a lot of falsely introduced alerts or missed interventions occur. This almost always results in unwanted maintenance costs. The operators are unable to reduce these costs because their insights into the problems are limited to comparing the deviated signals with the healthy system behavior [42].

In contrast, expert systems have been the most used fault diagnostic technology during the previous decade as faults must be detected precisely and timely in critical systems [43]. However, experts have a rather global view on the system and its functioning. As more and more sensors are attached to subparts of the system, it becomes difficult for experts to give concrete explanations to the deviations in sensor values and how they affect all the other system components. Knowledge in the form of facts and rules is available from mainly two types of sources: diagnostic-based knowledge and documented information. Both introduce difficulties in assimilation and requires a lot of work to become computerized [44]. Tools exist to limit these drawbacks, but human efforts are still required to maintain the acquired knowledge base [16]. Knowledge-driven models are in this perspective complex, resulting in longer processing times, which makes them impractical for real-time or nearly real-time purposes.

To overcome the problem of knowledge acquisition, a procedure has recently been proposed to select the input variables for soft sensors based on both data-driven and knowledge-driven input selection methods [45]. Soft sensors are generally built through data-driven approaches that exploit industry historical databases [46]. This new procedure allows designing soft sensors with good prediction accuracy and a low number of inputs, which reduces the complexity of the model and increases its maintainability. Currently, this is the only technique that is able to combine expert information with raw data. However, by limiting the expertise to the selection of input variables, these soft sensors deliver only a limited amount of interpretability. Another drawback is a possible reduction in adaptiveness when different systems require different input variables.

4. Proposed methodology

To meet all the requirements stated in Section 2, we propose FLAGS, the Fused-AI interpretabLe Anomaly Generation System. This system operates in 3 phases:

- In the first phase, both data- and knowledge-driven techniques are used in parallel. They take as input the data streams provided by one or more sensors, together with case-specific context data. Faults (knowledge-driven) or outliers (data-driven) are outputted. If possible, an interpretation of the detected anomalies is provided. The output of both techniques is stored inside a KG.
- During the second phase, the detected anomalies are shown in a comprehensive dashboard. Both the associated raw data and an interpretation, if available, are shown as well. The user can then provide feedback, e.g., confirm the anomalies and faults, merge them, or edit them. The feedback is also stored inside the KG.
- In a third phase, the information in the KG, i.e., the detected anomalies, the feedback provided by the user through the dashboard and all contextual meta-information, is used to improve the data- and knowledge-driven AD, FR & RCA techniques. User feedback is used by the data-driven methods to derive new interpretations for the outliers, while the contextual information is used to adapt the detection algorithms itself. Semantic rule mining is employed by the knowledge-driven methods to generate new knowledge given the updated KG. The newly derived knowledge is of the form of rules that indicate when particular faults occur, based on the historical information stored in the KG. This new knowledge is used to update the knowledge-driven detection tools automatically.

To let the different phases interact with each other, the FLAGS framework is built around the consumer–producers' principles. Each component can input data of interest and make their results available for other components that benefit from this produced output. Details on how such a consumer–producer approach is implemented, are given in Section 5. The following subsections explain in more detail the various modules that make up these 3 phases and how they interact.

4.1. Phase 1: data- and knowledge-driven AD, FR & RCA resulting in semantic & interpretable anomalies and faults

In Fig. 3, Knowledge (light grey boxes) and data-driven (black boxes) AD, FR and RCA are applied separately, as shown in the top and bottom parts.

As detailed in Section 3, the knowledge-driven FR & RCA techniques rely on expert knowledge captured in FMEA & FTA documents. By employing ontology-based risk analysis methods, this know-how is captured in semantic models and linked to the available background information about the system and context in which it operates. This information, i.e., the KG, is stored in a semantic database, e.g., a triple store.

The Semantic Mapping module is responsible for enriching the incoming data streams through these domain models. Mapping raw data to semantic observations provides an additional interpretable layer, and is an effective way to give a richer semantic meaning to the sensor data. Semantic mapping realizes the sharing, reuse and fusion of that sensor data [47]. Different mapping languages and paradigms can be used in the FLAGS methodology. An example of such a mapping module is discussed in Section 5.2.

This enriched stream is fed to the Semantic FD/RCA module. This module keeps track of a window of semantic observations and uses a semantic reasoner on these windows to infer semantic rules provided by the experts. When such a rule is triggered, a known fault is detected. The semantic reasoner combines the semantic observations with the profound domain knowledge in this perspective. This profound knowledge is available in the semantic database as an ontology with accompanying rules. As these faults originate from given semantic rules, the detected faults can be explained by providing the rule that was fired. This description resembles to what is more commonly known as Semantic Stream Reasoning (SSR) or SCEP. Section 5.3.2 gives more information on how such an SSR module can be implemented and interacts in the FLAGS methodology. An evaluation of the most common SSR systems is left out of scope.

The semantic FD/RCA module outputs faults and accompanying causes, which are also added to the KG stored in the semantic database, such that they can be used in future reasoning tasks. This whole knowledge-driven approach is shown schematically in the lower part of Fig. 3.

The ML-based AD modules take the raw data as input and use a ML technique to derive unique patterns in the streaming data, assuming these are outliers. From this perspective, any ML module which follows the guidelines of being operational in streaming environments and output outliers or uncommon patterns can be used in the FLAGS methodology. Details about such a learning module are provided in Section 5.3.1. Note that the different modules can output different types of anomalies.

The output of all these ML AD techniques is fed to the ML RCA module, which tries to link the anomalies and give some basic data-driven explanations. The ML RCA module is triggered once enough anomalies, all sharing the same confirmation label, have been recorded. An anomaly is from this perspective an event with a timestamp. The ML RCA module searches which combination of events can predict the occurrence of this anomalous event. Pattern matching and association rule mining techniques are used to find the relations between these events and the specific triggering anomaly. Once such a rule or pattern has been found, an explanation can be given to the anomalies triggering this event. These alerts continue to pop up until a better rule has been found using new incoming events, or the confidence of the rule drops as new anomalies have been registered. The implementation of such a RCA module is explained in Section 5.3.1.

A Semantic Mapping module is used again to semantically enrich the outputs of the ML AD & RCA modules. This module links the causes to the associated anomalies and let them relate to the underlying correlated data, system and context in which they occurred. These enriched anomalies are also stored in the semantic database. The data-driven AD/RCA module is shown in the top part of Fig. 3.

At the end of phase 1, the anomalies and faults are mapped in semantic format and stored in a KG. Storing them inside the KG has the advantage that background knowledge can be linked to the detected anomalies, giving more information about how and why they happened. Also, each AD component can semantically query and filter anomalies, either ML or rule-based. This eases the uniform visualization of all these detected anomalies, which is important for phase 2. The different techniques might pick up similar or correlated anomalies, which can be detected by exploiting the created links in the KG. Overall, the KG adds a layer of interpretability to the detected anomalies and their causes.

4.2. Phase 2: Comprehensive dashboard to easily capture valuable user feedback

In the first phase, both the data- and knowledge-driven techniques operate separately. A dashboard can give an overview of all detected anomalies, faults and causes, as shown by the striped arrows and dark grey boxes in Fig. 4. When new anomalies or

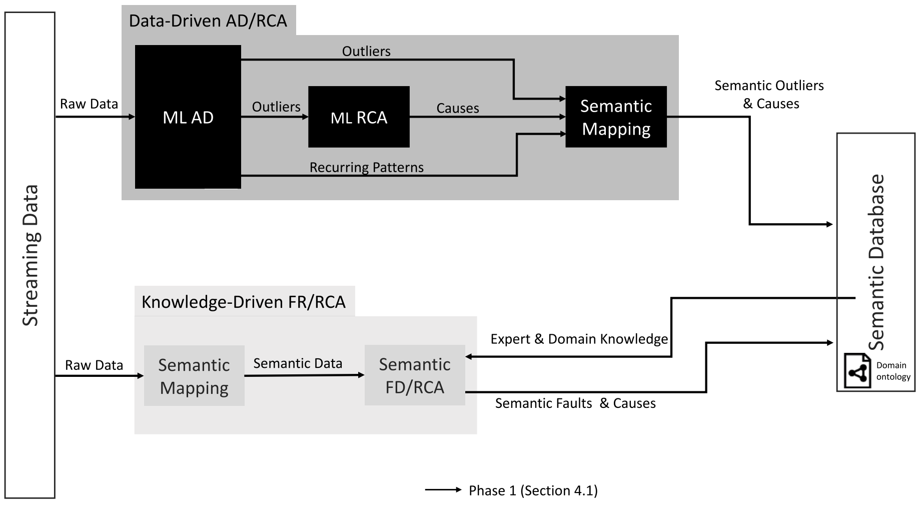


Fig. 3. Phase 1 of the FLAGS end-to-end methodology. Both the output of the data- and knowledge-driven techniques are combined in a semantic database.

faults are inserted in the KG, a dashboard visualization can be made to investigate the problem.

By exploiting all the semantic background information captured in the KG, comprehensive visualizations can easily be created. For example, anomalies can easily be grouped according to the context, system, configuration or the time frame they occur in. Such visualizations allow the end-users to quickly get a view on the system behavior without being overloaded with information.

Aided by the visualizations, the user can then quickly provide feedback on these anomalies, faults and causes by indicating their correctness or relabeling them based on the user's expert knowledge. By exploring the semantic links between the detected faults and outliers, e.g., whether they originate from the same sensor data, the user can also merge anomalies together. Merged anomalies are of huge interest for the ML RCA module as it provides similar events from which patterns can be found. The merge actions cause the outliers to be relabeled as known faults, which again is additional information to reduce possible false alerts.

All this feedback is also linked to the anomalies in the KG and stored in the semantic database. A fully functional dashboard that can provide feedback based on the visualization is discussed in depth in Section 5.4.

4.3. Phase 3: optimizing the data- and knowledge-driven AD & RCA through fused AI and user feedback

All the background knowledge, the context information, the anomalies, the causes and the feedback stored in the KG are used to optimize both the ML AD and Semantic FD/RCA module. These steps are indicated by the dotted arrows in Fig. 5.

For the data-driven techniques, contextual metadata is used first to enrich the preprocessing and feature extraction steps, by extracting additional features or by combining features into more informative ones. Background knowledge can also be used to choose the ML model type or tune this model to operate in varying contexts, i.e., use a separate model per context or configuration. In addition, the feedback generated by the user through the dashboard can also be taken into account. The outliers are (semi) automatically relabeled by merging them with detected faults for the same data points using the knowledge-driven techniques. This information can be used by the ML RCA to give a more detailed cause description of future detected anomalies of the same type. Another option is to use this information to build pattern detection tools to provide ML-based FR. Moreover, user feedback about the correctness of the anomalies is used by the ML AD module to further optimize its detection rate. The false positives and false negatives are now quickly addressed and stored alongside the input data.

Combining the outliers and faults in one KG also allows pinpointing faults that were not discovered by the semantic FD. These combinations of ML and Semantic FD output are ideal cases for finding new rules, i.e., new explanations for such new anomalies and making them detectable and explainable in the future. Semantic rule mining is employed on the KG containing all the semantic observations, their links to (merged) known (faults) and unknown (outliers) anomalies, and the links to the context and background knowledge. As indicated by the Semantic rule mining box in Fig. 5, this method derives inference rules describing the situations in which the outliers occur. These rules can then be added to the semantic FD and applied right after they are made available.

Semantic rule mining is done by applying sequential pattern mining techniques on the semantic data. Predefined support and confidence parameters are set as thresholds to filter those rules of interest. Different semantic rule mining approaches exist and can operate in the FLAGS framework. An implementation of such a semantic learner is discussed in detail in Section 5.5. A more in-depth analysis of all possible semantic mining frameworks is out of scope for this paper.

Again, these newly derived explained anomalies or faults, together with their accompanying context and inference rules when available, can be visualized to the end-user in an interpretable fashion. These visualizations enable a constant feedback loop between the detected events and the (learned) inference rules. The feedback loop allows the KG to evolve gradually with minimal human intervention.

5. Use case: Train bogie monitoring

The proposed methodology of Section 4 can be used in a wide range of applications. To showcase its potential, the methodology was applied to the railway domain, in close collaboration with the experts of Televic Rail.[3] They provided realistic datasets captured by their train monitoring systems, risk analysis information captured in FMEA & FTA documents and ML algorithms they already employ to perform AD. The following subsections first outline the

3 https://www.televic-rail.com/en.

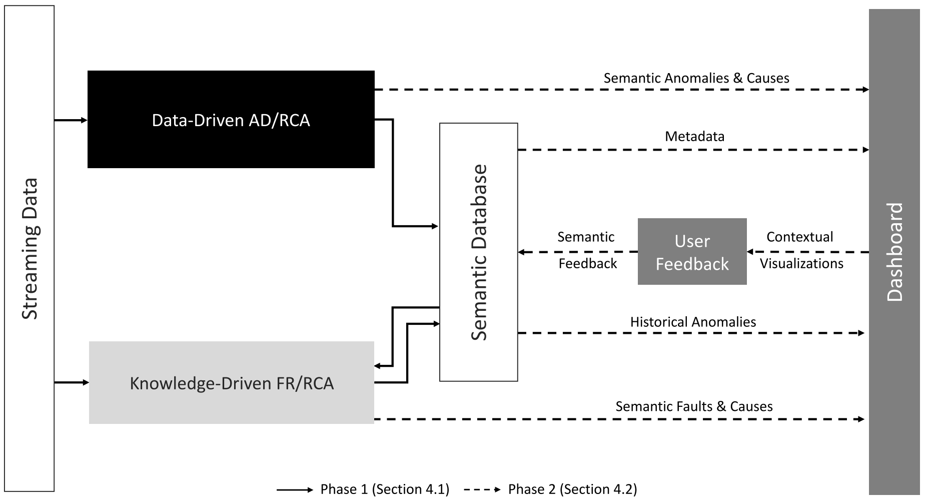


Fig. 4. Phase 2 of the FLAGS end-to-end methodology. The semantic faults, anomalies and causes are represented in a dashboard application. The end-users can provide feedback on the represented views.

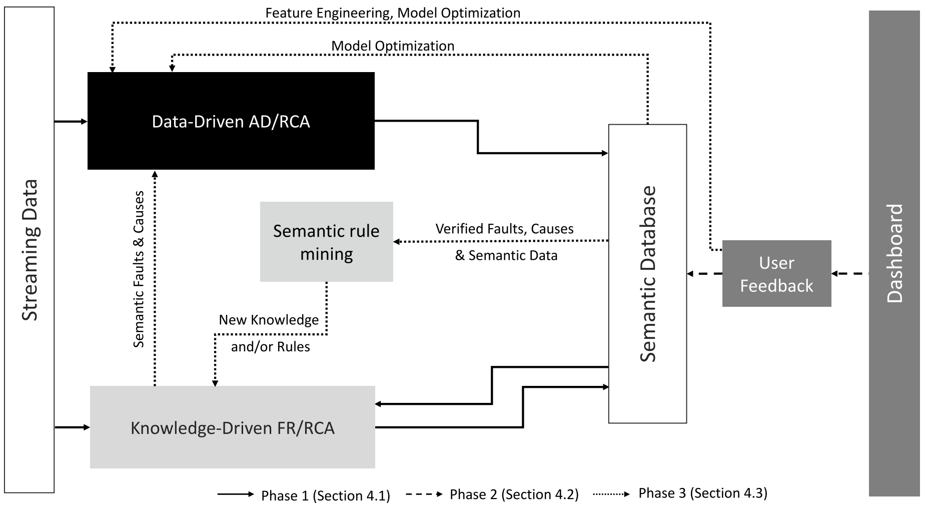


Fig. 5. Phase 3 of the FLAGS end-to-end methodology. The user feedback is used to update the data-driven detection behavior and to provide new expertise to the knowledge-driven FR using a semantic rule mining approach.

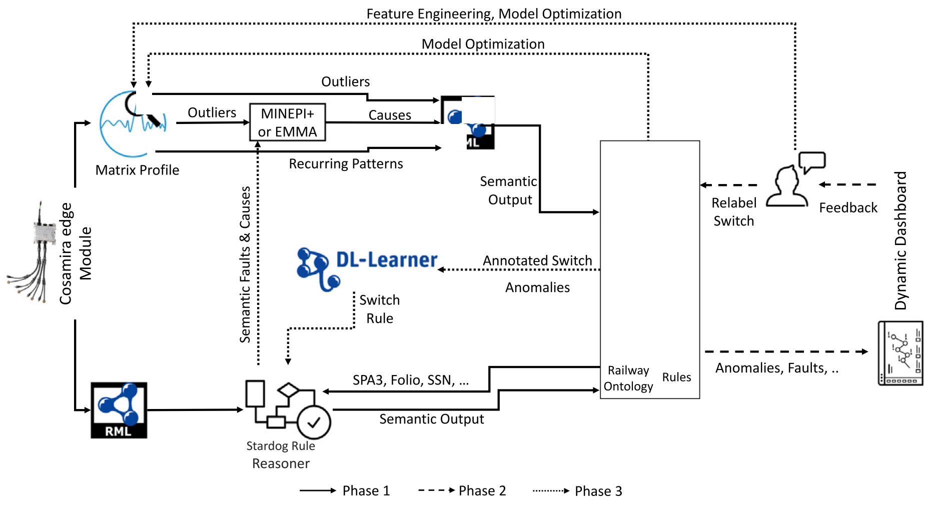


Fig. 6. Train monitoring use case overview, each component of the methodology in Section 4 is now specified by functional modules.

use case, followed by a detailed explanation of how the FLAGS methodology was employed and the various modules that were implemented to realize the optimization of AD, FR & RCA by fusing knowledge-driven algorithms, data-driven techniques with minimal user feedback. The application of the methodology to this specific use case is visualized in Fig. 6 and shows the different components of each phase in one diagram. As seen at the top right, the Matrix profile technique is used as an ML AD module to detect outliers in the raw data. At the bottom, the raw data is being semantified using RML and a Stardog Rule reasoner was built to detect known faults. All outliers and semantic metadata were stored inside a Stardog database, as shown on the right of Fig. 6. This Stardog database also communicates with a so-called Dynamic Dashboard [48]. Here, the database provides semantic information and stores the user feedback. At last, as rule mining techniques, we used some standard episode mining techniques, as well as DL-Learner, to discover new information. The rule mining interactions are shown in the center of Fig. 6. In the following sections, we will give more details about each of the used techniques.

As discussed in Section 4, this whole approach is designed around the principles of consumers and producers. In this use case, we used the Apache Kafka 2.4.1 platform[4] to let the data flow from one component to another. The components defined in this use case are either consumers, producers, or both as visualized in Fig. 7. The arrows show how these components interact with the available Kafka topics. All Kafka topics were divided into two main groups, one for the raw data and one for the semantic data. The raw.events topic is the one that is being used to let the company push its data. Both the data-driven ML modules and semantic mapping component consume the data from this raw.events topic. The mapping component transforms the raw events into semantic events and pushes these events to the semantic.events topic. Later on, the semantic FD module will consume the data from this semantic.events topic and produces events to the anomaly topic. The anomalies in this topic are also transformed semantically and added to the semantic.anomalies topic. This last topic is consumed by the semantic database, which stores all the semantic anomalies into the database. Similarly, the feedback and semantic.feedback topics are created to define the difference between raw and semantic feedback.

5.1. Use case description

Trains operate under various conditions. To ensure that both passenger comfort and equipment quality are adequate, more and more monitoring devices are being used to verify whether pre-defined standards are met [49]. One such sensor monitoring device is the Cosamira edge[5] designed by Televic Rail. This sensor monitoring unit is placed on the train bogie to capture the train's current location, the wheel axle temperature and the primary[6] & secondary suspension[7] of a train.

The data produced by this module can be analyzed to find irregularities in the train's behavior or the surrounding environment, e.g., to find anomalies. The main idea underlying this use case is that anomalies detected at the same location by different trains are most likely due to problems in the environment, e.g., faulty tracks or obstacles. The end-to-end methodology will make it possible to automatically discern anomalies due to degrading train equipment from the ones caused by irregularities in the environment and accurately pinpoint their cause, e.g., faulty wheel, bridge present, etc.

5.2. Knowledge graph and mappings

Televic prepared three datasets that each contain one hour of data. The datasets have been recorded at the same location, but for three different trains and moments in time. Every dataset contains:

- Latitude & longitude data sampled at 1 Hz (Global Navigation Satellite System or Gnss).
- Wheel axle temperature data sampled at 0.1 Hz.
- 3-axis accelerometers, data sampled at 1 kHz (Inertial Measurement Unit IMU).
- 3-axis gyroscopes, data sampled at 1 kHz (Inertial Measurement Unit or IMU).
- 1-axis shock pulse accelerometers, data sampled at 1 kHz (Shock Pulse Methods or SPM).

The datasets are known to contain track anomalies and to be different enough from one another in terms of accelerations. Three segments of five-minute data from these three different train rides are used here to show the benefits of our used approach. These segments are shown in Fig. 8. To provide a real-time detection mechanism, the sensor observations are streamed.

A railway ontology was made, in collaboration with the domain experts at Televic Rail who manufactured the Cosamira edge. This product-specific ontology extends the existing SPA3 and InteGRail ontologies [50]. In this ontology, the sensors, their interactions with the car body, wheel axle or track were defined.

Metadata of the trains and associated Cosamira modules, for which the data was collected, was also provided by Televic Rail. Track and train route meta information was captured using OpenStreetMap.[8] We created RML mapping scripts [51] to map all of this metadata about the tracks, trains and Cosamira module to the designed domain ontology in an automated fashion.

The OpenStreetMap railway-related nodes were transformed into semantic notations using the GEO-spatial functionalities to define their exact position and gather additional information such as crossings or stations provided by the SPA3 and InteGRail ontologies. We store the resulting KG in a semantic database. For this use case, Stardog[9] was used. The GeoSPARQL functionality was enabled in this semantic database to query location-based semantic data using the latitude and longitude samples.

Knowledge about possible anomalies, their causes and their effects on the train were provided by the Cosamira manufacturer using the discussed FMEA documents and FTA trees of Section 3. In this FMEA document, two main categories of anomalies were defined: track and train issues. A track issue is characterized by a certain irregularity on a specified location affecting more than one train. A train issue was specified by similar problems occurring multiple times, originating from the same Cosamira edge module (and so from the same train). By using the FOLIO ontology (see Section 3), these documents were translated into a semantic model linked to the railway domain ontology with accompanying rules.

A rule linking the location data from the GNSS sensor with the track anomalies and a rule specifying anomalies occurring at a single train was generated. Both of these rules are visualized in Listings 1 & 2. The anomaly information and the generated rules are stored in the same Stardog database as discussed above.

[4] https://kafka.apache.org.

[5] https://www.televic-rail.com/en/cosamira-rail-bogie-monitoring-data-analysis.

[6] The primary suspension stabilizes the running behavior of the bogie to ensure low track forces, low wear and proper behavior in curves.

[7] The secondary suspension measures the connection between the car body and bogie to isolate excitations transmitted from track irregularities via the wheelsets and bogie frames.

[8] https://www.openstreetmap.org/.

[9] https://www.stardog.com.

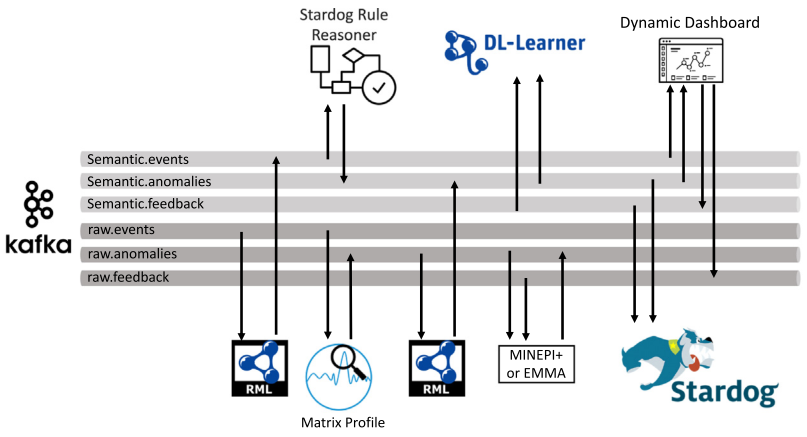


Fig. 7. Overview of each producer/consumer component and their interaction with the different Kafka topics.

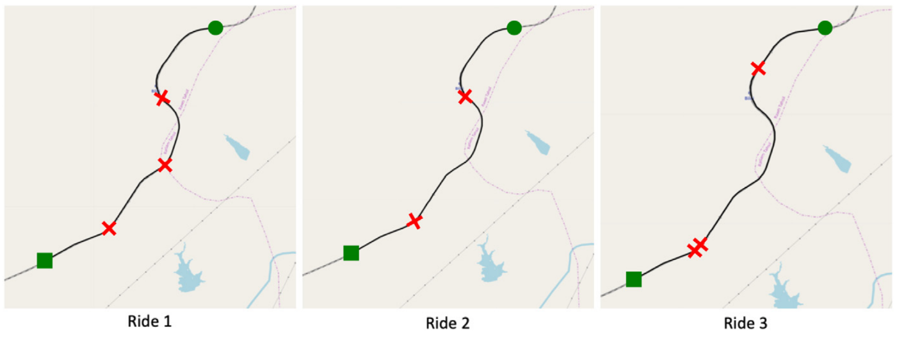


Fig. 8. Use case data for three train rides at the same location, at different moments in time performed by different trains. The train rides from the top of the figure (starting at the green circle) to the bottom (green square). The red crosses indicate the location of found anomalies by the ML AD technique specified in Section 5.3.

```
PREFIX folio: <http://IBCNServices.github.io/Folio—Ontology/Folio.owl>
PREFIX cosamira: <http://scope.icon.dyversify/televic/Cosamira#>
IF {
 ?a1 cosamira:fromTrain ?c1 .
 ?a2 cosamira:fromTrain ?c2 .
 Filter(?c1 != ?c2)

} THEN {
 ?a1 folio:cause 'Track␣problem' .
}
```

Listing 1: Track issue rule: comparing two anomalies or faults if they are from a different train. Whether these two anomalies occur near each other is defined by the GeoSPARQL functionality inside the semantic database.

```
PREFIX folio: <http://IBCNServices.github.io/Folio—Ontology/Folio.owl>
PREFIX cosamira: <http://scope.icon.dyversify/televic/Cosamira#>
IF {
 ?a1 cosamira:fromTrain ?c1 .
 ?a2 cosamira:fromTrain ?c2 .
 ?a1 folio:description ?d1 .
 ?a1 folio:description ?d2 .
 Filter(?c1 == ?c2 & ?d1 == ?d2)
} THEN {
 ?a1 folio:cause 'Train␣problem' .
}
```

Listing 2: Train issue rule: comparing two anomalies or faults if they are from the same train.

During the previous steps, schema information and expert knowledge were incorporated as an ontology with accompanying rules and populated with concepts borrowed from the railway and anomaly domain. RML mapping scripts were also created to map the incoming stream of raw observations from the Cosamira sensors to the SSN and the railway domain ontology. Concretely, all the sensor values listed above arrived at our platform in a JSON representation. Each JSON string included the actual data as a floating-point, the originating sensor, the corresponding metric and a timestamp when the value was generated. An example of such a JSON string is shown in Listing 3. A corresponding JSON-LD representation was made using the mapping file in Listing 4 to format these raw data samples. As a single number just specified the sensor ID, the corresponding JSON-LD makes the identifier link to additional sensor information and makes it possible for a semantic reasoner to infer knowledge by following this link. The other values, such as the timestamp, were matched in a similar fashion. As these transformations must occur in real-time, a streaming variant of the RML mapper was used [52]. The mapping script in Listing 4 was made manually using YARRRML,[10] which is a human-readable text-based representation for declarative Linked Data generation rules. It can be used to represent R2RML and RML rules in a human-friendly manner.

```
{
 "metricId": "sensor.axle.temperature::number",
 "timestamp": 1537060680000,
 "timeUnit": "MILLISECONDS",
 "sourceId": "COSAMIRA.BOX.0",
 "value": 23.81766845703125,
}
```

Listing 3: Example of a JSON value produced by the Cosamira Edge

A snippet of the resulting overall ontology is visualized in Fig. 9. In this snippet, the case-specific components, such as the

10 https://rml.io/yarrrml/.

```
prefixes:
 sosa: http://www.w3.org/ns/sosa/

mappings:
 observation:
 sources:
 - [input.json~jsonpath, $]
 s: http://.../observations/$(timestamp)
 po:
 - [a, sosa:Observation]
 - [sosa:observedProperty, http://.../$(metricId)~iri]
 - [sosa:resultTime, $(timestamp), xsd:dateTime]
 - [sosa:hasSimpleResult, $(value), xsd:float]
```

Listing 4: Example of a mapping file to transform the JSON value of Listing 3 to a semantic format.

Cosamira Shock sensor, are indicated as black boxes. Reusing concepts of the FOLIO ontology, the railway ontology and SSN show how the case-specific instances interact with each other and how they relate to the provided faults (a Train issue in this example). A simple rule is also visualized, showing the link between the observations and the AnomalyKnowledge concept.

5.3. Data- and knowledge-driven AD & RCA through semantic reasoning and matrix profiling

When all metadata is available in our system and sensor data streams in, both the data- and knowledge-driven modules can start detecting anomalies.

5.3.1. ML AD & RCA

Our main focus in this use case is to find either track or train issues. Therefore, the SPM signal is the most informative for this case as the producing sensors are closest to the track. The ML AD module aims to find abnormal patterns, i.e., discords, in the observed time series made by the sensors, and to match incoming patterns against previous confirmed to be anomalous by the dashboard. Matrix Profiling was selected because it is unsupervised, well suited to streaming data, has support for multidimensional data and can work in real-time. As visualized in Fig. 10, it works by moving a sliding window (W1) over the time series and tracking the best match to any other previous window (W2). The z-normalized euclidean distance is used as a distance measure to determine possible matches [26]. The minimum values for each of these matching fragments, in this case columns (F), are used for the resulting Matrix Profile vector.

After preprocessing the SPM signal, by subtracting the sensor offset and then squaring and smoothing it, the values are passed into a streaming-enabled Matrix Profiler that tracks the last 30 s of data (30k sensor values).

We implemented an additional constraint in the Matrix Profile so that only windows with a similar standard deviation could be compared. This ensures that the technique could distinguish between visually different patterns. As the input data is being processed, the resulting stream of distances is checked against a threshold, where high values indicates anomalies. Afterward, we merge nearby high values into a single, longer anomaly of up to 10 seconds. This technique was implemented as an extension of the Series Distance Matrix framework [25].

All train rides followed the same trajectory. Therefore, the main difference between these three rides was the train speed and corresponding vibrations along the way. The red crosses in Fig. 8 indicate anomalies detected for the 5 min use case by the Matrix Profiling.

Matrix Profiling was used to detect new outliers in the data, while the ML RCA module wants to provide an interpretation for the outliers of interest on the streaming data. To determine which outlier has to be mapped, rule mining techniques can be used to provide possible explanations, transforming them into more faults after anomalies have been detected.

Because anomalies in the SPM data consisted entirely of outliers rather than faults, the rule mining technique lacked proper input and did not deliver interesting results. This is a good example of where data-driven methodologies are limited in the amount of interpretation they can give or the knowledge they can derive.

Both the anomalies and causes derived by the ML AD and ML RCA module are semantified using RML mapping scripts and stored in the Stardog database. This semantification step is similar to the one described to semantify the raw data. The output of both the ML AD and ML RCA module is in a JSON format, and an RML mapping script transforms these JSON strings into a JSON-LD representation, based on the FOLIO ontology.

5.3.2. Semantic FD/RCA

As mentioned in Section 5.2, RML mapping scripts were created to semantically enrich the raw observations coming from the Cosamira sensors. To continuously map the incoming streaming data, the RMLStreamer [52] was used. All the raw events are stored in a data warehouse and it is not required to store all the semantic observations inside the semantic database. A replay of the data is possible on request. Therefore, the semantic database can be kept small and does not require to be reindexed when new data samples are coming in.

Data-driven AD is good at processing sensor data streams fast. Moreover, the domain experts had limited expert knowledge about which values or trends in the raw data streams indicated interesting events or anomalies. However, expert knowledge was available about how the detected anomalies could be related to the background knowledge, the context (e.g., the environment of the rails) and rolling stock (e.g., the dynamics of the train) to classify them as particular faults. Therefore, the semantic FD includes rules to reason directly on the anomalies generated from the ML AD module and classify them as faults. The code to perform such Semantic FD has been made open-source.[11] Once processed, the information of multiple outliers detected by the ML AD and train & track metadata becomes available inside the Stardog database. This semantic data can be used inside the semantic FD/RCA to filter generated anomalies or provide them with additional data such as more knowledge-based descriptions and causes. This filtering approach is visualised in Fig. 11.

More concretely, this filtering approach consumes the semantically transformed output of the ML-based Matrix Profiler. When the Matrix Profile module detected an anomaly, the coordinates of this anomalous event are used in a SPARQL query. The query which has been executed, is shown in Listing 5. This query finds all the information of interest in a predefined range around this anomaly. Among the possibilities are included:

- Previously occurred anomalies, which are confirmed or known faults.
- Semantically enhanced OpenStreetMap nodes which deliver railway information.
- Rules triggered by anomalies, which were composed based on Televic's expertise.

When any such additional information is available and gets triggered by the executed query, the inputted anomaly is enhanced with this information and stored semantically in the Stardog database.

In the perspective of the given use case, during the first train ride only additional information provided by the experts and

[11] Semantic Fault detector: https://github.com/IBCNServices/StardogStreamReasoning.

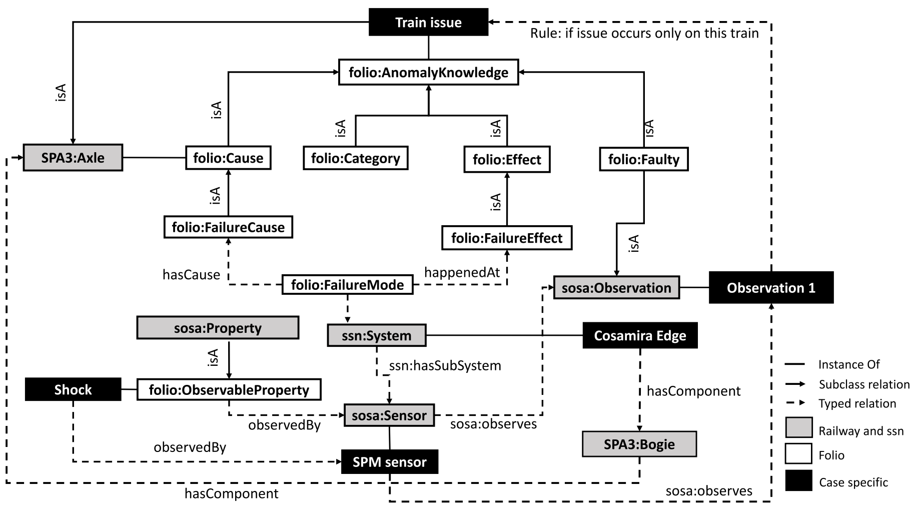


Fig. 9. A snippet of the overall ontology. This snippet shows how the SPM sensor observes Shock Observations and how it is connected to the Cosamira Edge system through the bogie and wheel axle. A rule which relates observations to train issues is also given.

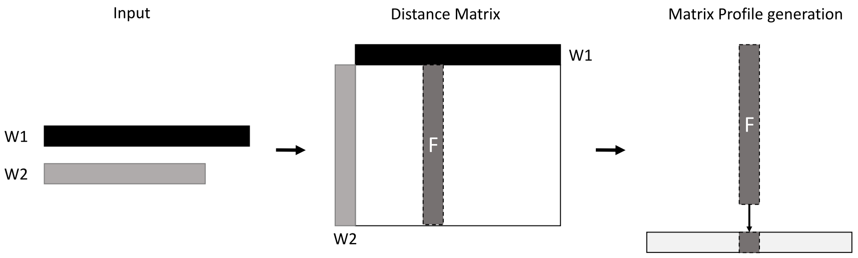


Fig. 10. Overview of the ML AD Matrix Profile approach. Starting from two input windows (W1, W2), the z-normalized Euclidean distance generator iteratively creates fragments (columns F) from the distance matrix of all subsequences. Each of these fragments is processed by the Matrix Profile consumer, storing the minimum value for each column in the resulting Matrix Profile vector..
Source: This Figure has been adapted from [25].

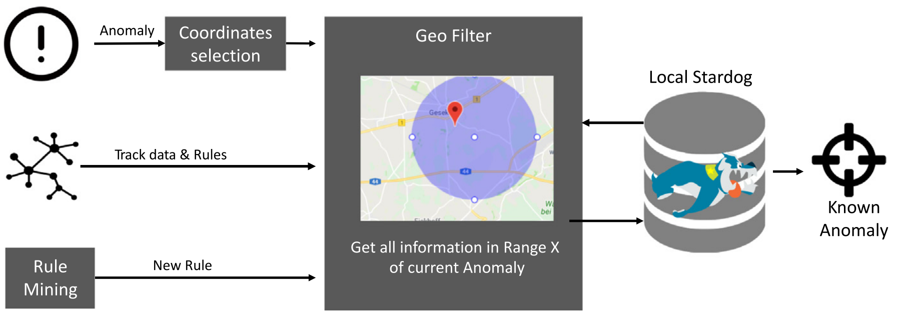


Fig. 11. Overview of the semantic AD/RCA approach. The coordinates of the detected anomaly, the track data, and accompanying rules are used inside the GEO Filter to determine whether or not an explanation can be given to the detected anomaly. A local Stardog database is used to filter those events. Additionally, a component is available to update newly mined rules when they become available.

extracted from the OpenStreetMap data can be delivered. The coordinates from the ML detected anomalies during the second train ride are given to the GEO Filter. This filter executes the query in Listing 5 on the semantic database. The rules specified in Listing 1 and 2, describing either a track or train issue, will be triggered to return the corresponding cause description. When no additional information can be provided, the anomaly is kept in its original state and reported, as is, in the dashboard.

The nodes and rules specified by the Geo Filter are all loaded into a separate local Stardog database to perform rule-based reasoning on a particular subset of the data. This Stardog database is a completely separate instance, isolated from the general semantic database which stores the semantic anomalies, faults, causes and user feedback. Loading a set of observations and

```
prefix geo: <http://www.opengis.net/ont/geosparql#>
prefix geof: <http://www.opengis.net/def/function/geosparql/>
prefix unit: <http://qudt.org/vocab/unit#>

SELECT distinct ?feature {
 {
 ?geom geof:nearby ( $lat $long $range unit:Meter) .
 ?geom <http://IBCNServices.github.io/dyversify/Televic#type> ?feature.
 } UNION {
 ?an <http://IBCNServices.github.io/dyversify/Televic#track> ?feature.
 }
}
```

Listing 5: Geo Filter query: This query filters on nearby nodes given a latitude, longitude and range. The lat and long parameters are extracted from the current occurring anomaly, the range parameter is set upfront to 100 m.

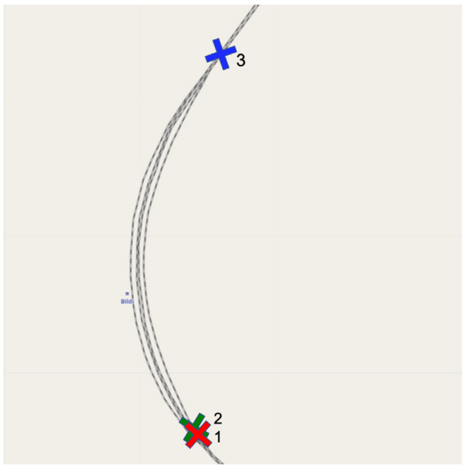


Fig. 12. Detailed overview of the three top anomalies from train ride 1 (red), 2 (green) and 3 (blue) in Fig. 8. They all occur near a switch. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the corresponding metadata in separate semantic databases has two main advantages. First, multiple local databases can be set up when multiple new anomalies arrive at the same time. This enabled us to perform reasoning in parallel to speed up the whole procedure. Multiple GEO filters can be made operational to query the additional data for multiple occurring anomalies as the Matrix Profiler can produce them faster than a single GEO filter can reason. Second, reasoning operations did not affect the general Stardog database, which is preferable when more than one application depends on it.

During a second train ride, some anomalies are close to the location of the anomalies detected in the first train ride. In the meantime, semantic background knowledge became available and is provided in the semantic database. The semantic FD will use the information about the anomalies detected during the first train ride, together with meta-information given by the railway experts to identify these new anomalies as track issues according to the rule shown in Listing 1. An example of such a triggered rule is shown in Fig. 12 (1 or red cross). The anomaly that occurred during the second train ride is close to the one that occurred during the first ride (2 or green cross). Since both anomalies originated from different trains, both were reclassified as track issues by our semantic FD module.

5.4. Dynamic semantic dashboard enabling user feedback

For visualization purposes, the so-called Dynamic Dashboard [48] is being used to show those visualizations that correspond with sensors selected by a user (user-driven visualization) and those visualizations that correspond with occurring anomalies or faults (anomaly-driven visualization).

In user-driven visualization, the Dynamic Dashboard helps the user select visualizations that are interesting to present, e.g., a user may want to visualize the route of a train on a map by selecting a widget for the sensor which publishes locations. The user is presented with the visualization options that the software deduced. Because all sensors and their corresponding metrics have a semantic description, each widget can reason whether or not it is useful to display the observations generated by the sensors. In this example, a gauge widget can be suggested to the operator to visualize the wheel axle's temperature sensor. More information about the workings of this dashboard can be found in Vanden Hautte, et al. [48].

In anomaly-driven visualization, the Dynamic Dashboard automatically creates a dashboard tab to investigate a selected anomaly. The dashboard tab is built with widgets that are found to be appropriate given the type of the selected anomaly and according to the sensor properties that are linked to the anomaly. The dashboard tab is built with appropriate widgets given the type of the selected anomaly and according to the linked sensor properties. Like the user-driven widgets, the anomalies are represented in a semantic format and can, therefore, interact with these widgets. In this example, the anomalies are all associated with a location and can thus be visualized on a map as shown in Fig. 12.

Creating a dashboard with multiple widgets of interest requires less time with this Dynamic Dashboard, since several suggestions are made using an underlying semantic reasoner. A possible dashboard for our use case can be seen in Fig. 13. Two different panes are visible: one left width five different widgets and one showing the listing on the right. Each of the five widgets represents a different sensor from our monitoring unit. From left to right and from top to bottom, the dashboard shows:

- The Y-axis acceleration of the IMU sensor as a time series.
- The Z-axis acceleration of the IMU sensor as a time series.
- The Yaw values of the IMU sensor as a bar plot.
- The temperature sensor as a raw value with a color-schemed background.
- The wheel acceleration of the SPM sensor as a time series.

The listing on the right side of this dashboard shows the occurring anomalies. An anomaly-driven dashboard tab will be opened once we click on one of the anomalies.

Creating widgets, interacting with these widgets and setting parameters all indicate possible interesting parts of the data an end-user deals with. These interactions can be useful feedback for our detection modules. Two types of feedback can be captured in this dashboard and are of interest during this evaluation. Confirming or rejecting anomalies will adapt the operating AD modules in our methodology. Merging anomalies will cluster similar anomalies together, which is useful for the rule mining modules.

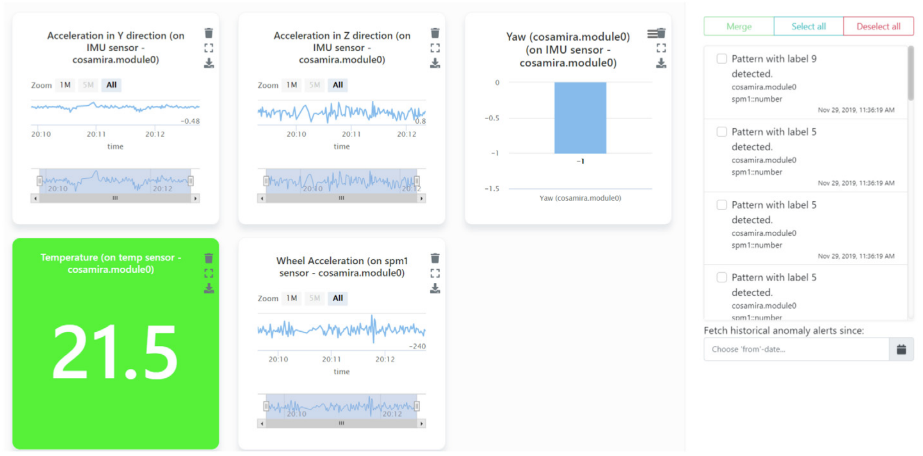
During the first train ride, only the ML AD can detect outliers, which are visualized to the end-users using anomaly-driven visualizations. This means that, based on the semantic annotations of the anomaly (in this case, unknown anomalies generated by the Matrix Profiler) and its origin (the SPM sensor with 1-axis accelerometer metric), a widget pops up on the dashboard making all the information visible to the end-user. The operator can already correct or label these outliers.

After the second train ride is finished, the semantic FD/RCA already outputs some faults, which differentiate between train and track faults. As the anomaly-driven widget visualizes the locations of the occurring anomalies as shown in Fig. 12, an operator was able to determine that some of the inferred track issues could be explained with the occurrence of a switch. The relabeled anomalies can then be used as input for the semantic rule mining procedure of phase 3.

At last, similar to the raw data, the user feedback captured in the dashboard is semantified using RML and stored in the semantic database. All feedback is provided in a JSON format and semantified to JSON-LD, a procedure similar to semantifying the anomalies or raw observations.

A newly designed feedback ontology was used to generate this semantic feedback and differentiates between intrusive and non-intrusive feedback.[12] Intrusive feedback comprises manual operations performed by the operator or dashboard user, such as relabelling anomalies. Examples of non-intrusive feedback are closing

[12] Also available at https://github.com/IBCNServices/Folio-Ontology.


Acceleration in Y direction (on IMU sensor - cosamira.module0)
Acceleration in Z direction (on IMU sensor - cosamira.module0)
Yaw (cosamira.module0) (on IMU sensor - cosamira.module0)
Temperature (on temp sensor - cosamira.module0)
21.5
Wheel Acceleration (on spm1 sensor - cosamira.module0)
Zoom 1M 5M All
time
Yaw (cosamira.module0)
Merge
Select all
Deselect all
Choose 'from'-date...

Pattern with label 9 detected.
cosamira.module0
spm1::number
Nov 29, 2019, 11:36:19 AM

Pattern with label 5 detected.
cosamira.module0
spm1::number
Nov 29, 2019, 11:36:19 AM

Pattern with label 5 detected.
cosamira.module0
spm1::number
Nov 29, 2019, 11:36:19 AM

Pattern with label 5 detected.
cosamira.module0
spm1::number

Fetch historical anomaly alerts since:

Fig. 13. Dynamic dashboard example for a train bogie monitoring case. The left pane shows 5 different widgets, displaying the sensor values. The right pane is populated with the detected anomalies.

and opening dashboard widgets. All the semantic feedback is stored in the Stardog database.

5.5. Optimizing the data- and knowledge-driven techniques through user feedback and semantic rule mining

Despite several occurrences of (geographically) nearby abnormal patterns detected by the Matrix Profiler over the three train rides, as visualized in Fig. 14, there was almost no similarity between their shapes. This made it challenging to build a pattern matching or fault detection tool purely based on the ML AD. The differences in these patterns are most likely due to the spiky nature of the signal and slight variations in train speed, causing high distances when comparing subsequent shapes [53]. Therefore, the feedback indicating switches occurred at the specified locations did not affect the ML AD module. This again shows the importance of combining the data-driven techniques with expert knowledge and user feedback to optimize AD, FR & RCA.

When enough anomalies have been labeled in the Dynamic Dashboard and become available in the Stardog database, the semantic rule mining component will become active to mine rules for those cases of interest. In this train use case, two anomalies from two different train rides occur close to each other and are labeled as switches through the Dynamic Dashboard, as shown in Fig. 12.

The rule miner starts to search for a possible explanation and an accompanying rule for these similar anomalies. The supervised framework DL-learner [54] was used to search for rules to describe the detected anomalies. DL-learner takes the labeled switch events, some nearby nodes linked to the anomalies and some non-related observations as input. It then searches a rule based on all this information, together with all available metadata of the track and train. The miner's input thus contains both positive samples (switch events), labeled as anomalies, and negative samples (other, normal events). The goal is to find the OWL class expression R such that all or many positive samples are instances of of the class C and as few as possible negative examples. As explained, R should be learned such that it generalizes to unseen individuals and is readable. R is seen as a rule in our system and is applied to find the positive events. The process described above and used by DL-Learner is called the Class Learning Problem (CLP).

The Geo filter set-up defined in Fig. 11 can consume new rules that were altered and applies them directly after the semantic rule miner has generated them.

The generalized rule, mined for these two switch events, is given in Listing 6 and defines a switch as a 'transition' in track

```
IF {
 ?w1 a <http://www.integrail.info/ont/SP3A.owl#Way>.
 ?w2 a <http://www.integrail.info/ont/SP3A.owl#Way>.
 ?w1 <http://IBCNServices.github.io/dyversify/Televic#Contains> ?node1 .
 ?w2 <http://IBCNServices.github.io/dyversify/Televic#Contains> ?node1 .
 FILTER(?w1 != ?w2)
} THEN {
 ?node1 <http://IBCNServices.github.io/dyversify/Televic#type> 'Switch' .
}
```

Listing 6: Newly generated rule applied after the second train ride. This rule was mined using the anomalies found in the first two train rides, as shown in Fig. 12 and combines track segment information available in the metadata.

segments using the OpenStreetMap's metadata. An additional component applying newly learned rules derived by the semantic rule miner was added, as shown in Fig. 11. Before the Geo filter is executed, this component is used to insert newly generated rules into the local Stardog databases. This update module ensures that newly mined information can automatically influence the derivation of anomalies in the future.

During a third train ride, a newly generated anomaly is classified as a switch event, as shown in Fig. 12. This is due to the newly applied rule (blue cross, or 3). With this rule, the semantic FR/RCA was able to deliver new insights provided by the metadata. A new type of fault can be filtered now, without needing additional human involvement.

5.6. Evaluation setup

To evaluate this use case, a Kubernetes[13] cloud setup was used to integrate each component mentioned in Fig. 6 and to let them communicate with each other. In total, two separate clusters were used:

- One cluster with 40 cores and with 40G RAM was used for all the ML and semantic AD, FR and RCA modules, the rule mining components, the Stardog database and dashboard. Resources were shared according to the needs of the modules.
- A second cluster with two cores and 24G RAM was used for the Kafka delegation.

This separation was needed to ensure the throughput of the streaming data.

13 https://kubernetes.io.

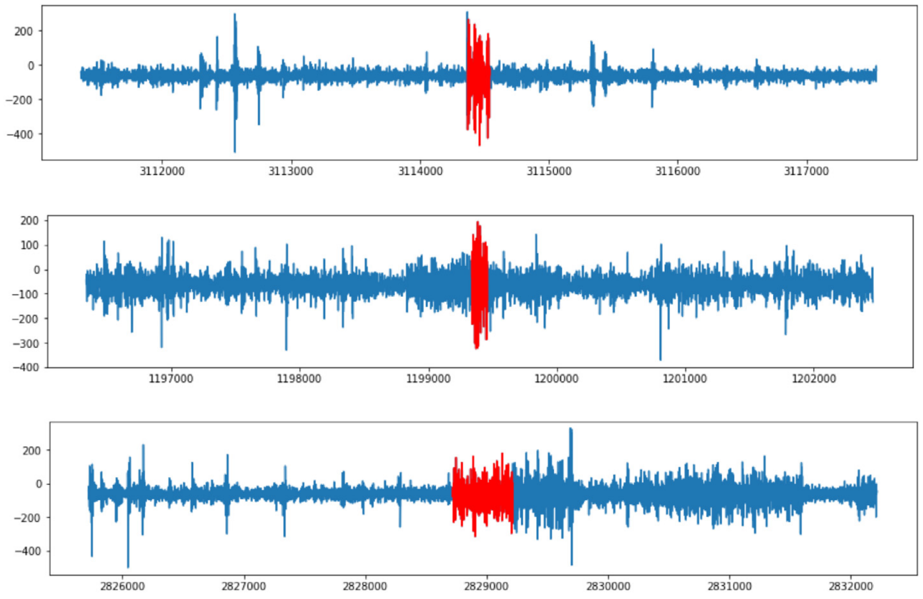


Fig. 14. Examples of detected anomalies for the SPM signal.

The five-minute raw sensor data for all three train rides took 855 MB (285 MB each). This dataset contained 1,200,000 SPM sensor values, which all have to be transformed into a semantic notation. For each such sensor value, five triples were generated by the RMLStreamer and pushed to the corresponding semantic Kafka topic. This RMLStremaer module, based on Apache Flink, was initialized with four task slots and one job manager with a heap size of 2048 MB to ensure the throughput. The manually constructed semantic metadata was stored inside the Stardog database and consisted of 326 586 triples. This Stardog database module was set with a heap size of 15G to ensure all queries could be executed. The Dynamic Dashboard module was initialized using the default configurations as described in [48].

Working within a streaming environment required some of the modules to be initialized upfront. The Matrix Profile module compared windows with a length of 100 ms and was initialized on the first 100 s of the original one-hour dataset. This initialization step is required to determine the sensor offset and noise parameters. Since the presence of noise can have a significant effect on the distance measure used by the Matrix Profiler, the noise elimination technique described in De Paepe, et al. [53] was used. The Matrix Profile itself is calculated at 25 ms intervals, to ensure sufficient throughput. This approach can process up to 6000 events per second, enough to match 4000 incoming train events every second (calculating the matrix profile at 25 ms interval, at a rate of 1 kHz).

The semantic FD module was initialized with the metadata and rules described in Section 5.3.2. Ten local Stardog modules were used within this module to ensure multiple anomalies could be filtered at the same time. The SL reasoning level and GeoSPARQL functionalities were enabled within these local Stardog databases. The default parameters were used for all other settings.

A configuration file was required to initialize the semantic rule mining by DL-Learner. Listing 7 shows this configuration. The default Class Expression Learner or Ontology Engineering (CELOE) algorithm was initialized with the Hermit reasoner and limited to return 300 results. The maximum execution time of this algorithm was set to 500 s. The rules were evaluated using the F1-measure.

```
// knowledge source definition
ks.type = "OWL␣File"
ks.fileName = "data.nt"

// reasoner
reasoner.type = "closed␣world␣reasoner"
reasoner.reasonerComponent = embeddedReasoner

embeddedReasoner.type = "OWL␣API␣Reasoner"
embeddedReasoner.reasonerImplementation = "Hermit"
embeddedReasoner.sources = { ks }

// CELOE
alg.type = "celoe"
alg.maxNrOfResults = 300
alg.maxExecutionTimeInSeconds = 500
alg.expandAccuracy100Nodes = true
alg.maxDepth = 45

// learning problem
lp.type = "clp"
lp.classToDescribe = folio:AnomalyKnowledge (iri)
accuracyMethod.type = "fmeasure"
```

Listing 7: DL-learner configuration file to relate the positive instances in the AnomalyKnowledge class together.

All defined parameters for these data- and knowledge-driven techniques were defined by using expert knowledge. Each module runs in a different Docker container,[14] and does not affect other ones' execution when enough resources are available. To measure the duration of the code within each module, the standard libraries within Python and JavaScript for time measurements were used. Grafana[15] was used to investigate both the performance in terms of used resources for each module, as well the lag and interactions between the multiple Kafka topics.

6. Results

In total eight anomalies were found in the three segments, visualized in Fig. 8. An operator of the industry partner inspected all of them. Three anomalies were found near bridges and were indicated as low priority track issues. Two anomalies were relabeled as switches as described above, which led the third one to be automatically labeled as a switch event. The other two anomalies could not be clarified, and as they did not reoccur

14 https://www.docker.com.
15 https://www.grafana.com.

during the other train rides, the operator classified them as low priority train issues for which further investigations are needed.

The Matrix Profiler found all these anomalies. The semantic FD classified 75% of the found anomalies as track issues as they occurred near each other. For the bridges and switches, the operator used the available context information of OpenStreetMap to verify whether the occurring anomalies could be explained. After the new switch rule has been applied, one anomaly was even explained automatically. In total, 75% of the anomalies received an explanation and the operator reduced their priority based on these explanations. 25% of the anomalies are now classified as false positives as the operators could not provide clear explanations for them.

To show the FLAGS system's adaptation rate, we investigated how long it would take to enable a new rule based on the feedback provided by the operator inside the Dynamic Dashboard. Time measures were taken to get an indication of the time it might take until the moment when the additional dashboard information becomes available in the semantic database. As discussed above, both the semantic rule miner and ML algorithms benefit from this feedback. The time between the moment the merge or relabel button was pressed until the moment the feedback was made available in the Stardog database was measured. On average, this action took 1694 ms (measured over 5 different runs std: 239.68 ms).

When enough semantic feedback is available, the semantic rule miner generates new knowledge to be incorporated in the semantic FD. This mining operation was limited in time, as indicated in Listing 7. Automatically applying a rule found by the rule mining module and making it active inside the semantic FD requires 31 ms on average (std: 2 ms over 5 rules). The semantic FD can filter the found anomalies based on the provided rules and metadata in 730 ms on average (std: 500 ms over 5 runs, using a cluster node with 8G ram and 10 CPUs).

7. Discussion

As stated in the requirements of Section 2, to deliver cost-effective and value-adding AD, FR & RCA, the proposed methodology must be adaptive, context-aware, precise, interpretable and should necessitate as little interventions needed as possible. The methodology that fulfills these requirements is FLAGS, which fuses the two known data- and knowledge-driven techniques together and incorporates valuable user feedback to adapt the models' internal functioning to new contexts or environments.

As explained in Section 3, relying on either one of the data- or knowledge-driven methodologies by itself would result in a less functional system. The knowledge-driven setup would not be able to detect faults because it cannot deal with the flood of sensor data and the experts do not have enough knowledge to verify manually which patterns in the raw data might lead to possible faults. The data-driven technique would be able to detect outliers, but almost all detected outliers are different from each other. Pattern matching tools, delivering some additional information by grouping the anomalies, gave no impressive results and obeyed the ML-based RCA to find useful causes. In the end, it comes down to a choice between almost no detected faults at all or giving anomalies to an operator without any explanation.

In this perspective, FLAGS fuses both approaches taking only their advantages. The evaluation of this system shows that most of the set requirements are met, as summarized in Table 2:

- **Precise:** The combination of knowledge-driven expertise with data-driven detections improved the detection rate.
  The Matrix Profiler operational in our evaluation use case will produce anomalies as outliers without updating its behavior after the initial 100 s. The semantic FD decides by using a GEO filter whether or not the detected outlier is of interest for an operator. FLAGS, therefore, filters a lot of false positives compared to the data-driven technique.
  The combination of both methods resulted in 75% correct filtered anomalies and only 25% of the anomalies were classified as misleading or false alerts. FLAGS again benefits from the combination of both the data-driven and knowledge-driven techniques. 100% of the found anomalies would have been false alerts using only the data-driven technique. The currently operating knowledge-driven approach would not even be able to make a single detection.
- **Require minimal human involvement:** FLAGS does not require to implement all the expert knowledge or provide fully trained ML models to generate useful insights. The evaluation section, for example, uses the Matrix Profiler instantiated on the first 100 s of the dataset. Almost all of the knowledge about possible faults was provided using FMEA tables and automatically mapped as described in Section 3. Fusing both a data- and knowledge-driven approach leads to better results than the approaches either would have exhibited on their own. Anomalies are filtered, reducing the need to improve the ML models themselves. New anomalies are merged and/or relabeled, resulting in automated adaptations in failure recognition, and thus enriching the knowledge-driven components.
  The evaluation shows the benefits of this fusion by demonstrating that no human effort is needed to retrain or adapt the knowledge to start detecting switch events after the first two train rides are finished. To thoroughly verify this requirement, deploying FLAGS within the operational environment is needed to perform realistic tests by the operators.
- **Context-aware:** The overall FLAGS methodology incorporates context information from the central KG in both the data- and knowledge-driven modules. From this perspective, the ML modules can benefit from the detected faults provided by the knowledge-driven parts and vice versa.
  In our evaluation, this contextual enhancement reduces the manual inspections needed for those anomalies related to the track issues. As Televic is interested in scheduling predictive maintenance for trains, other companies have to handle the anomalies related to track issues. However, detecting track issues reduces the number of falsely requested maintenance actions or, more in general, the number of false positives. FLAGS was able to explain four anomalies (two related to the bridge and two near the switch) based on the context of the ones that were found earlier.
  Without the use of context, all these anomalies would have been classified according to their signal shape as shown in Fig. 14. The Matrix Profile only considers the raw signals, and as already discussed, limited information could be extracted from these patterns due to the different behavior of the trains in this track segment.
- **Adaptive:** In general, the FLAGS methodology copes directly with the user-provided feedback and adapts the models by merging, deleting or relabeling the detected faults or anomalies.
  During the evaluation, relabeling two track anomalies as switches enabled the system to detect switch events for the third train ride. On average, 8.36 min are required to apply a newly learned rule in the semantic FD (1694 ms to store the user feedback in the database, 500 s to perform the rule mining and 31 ms to install this found rule). These are all non-blocking operations, so the operator was not aware of these delays at all.
  Without the FLAGS methodology, changing the behavior of the semantic FD would require more time to adapt. The

Table 2
Overview of the proposed FLAGS methodology regarding the requirements made in Section 2.

	Data-driven	Knowledge-driven	FLAGS
Precise		X	X
Minimal human involvement	X		(X)
Context-aware	(X)	X	X
Adaptive	X		X
Interpretable		X	X

operator would have to translate his findings into a user-defined rule, which must then be applied in the semantic FD rule base. In most cases, this would even require stopping the semantic FD for a while to make sure the update was sufficient. To adapt the behavior of the data-driven components without FLAGS requires retraining or even a reinitialization of the ML algorithms.

- **Interpretable:** The FLAGS approach combines all the output of the generated modules in a semantic format, which provides the end-user or operator with a lot more interpretability than the fault or anomaly could give by themselves.
  During the evaluation, all results were shown in the Dynamic Dashboard. As this dashboard benefits from the semantically generated output, an operator only has to perform several clicks to investigate the anomaly's occurrence. The preferred visualization was decided by reasoning about the occurrence of these anomalies. Out of the eight anomalies, the operator could easily identify which of the track issues were related to a switch event, as visualized in Fig. 8. Additionally, the semantic FD within FLAGS could cope with the third track issue and automatically relabel it as a switch event. This use case already showed the efficiency of investigating anomalous occurrences using FLAGS.
  For six of the eight anomalies, a useful interpretation was given by FLAGS. The data-driven AD module was unable to explain these cases, and they only became useful for the operator when the semantic FD filtered them.

However, the FLAGS system also has some limitations. As the adaptation of the whole approach is orchestrated by providing feedback, falsely pinpointing parts as normal behavior or wrongly classifying events as anomalies, would require multiple components to reset. This is mainly due to the fact that these components depend on the provided feedback and adapt their behavior based on them. Another drawback is the automatic application of new rules in the semantic FD. While this makes the knowledge-driven part adaptable, it reduces the experts' control of the system's internal functioning. At last, the amount of data floating through FLAGS is doubled as the data-driven approaches prefer raw instead of semantified data. Such huge amounts of data and the need to transform them into a semantic representation require high-end cloud set-ups.

8. Conclusion

To deliver cost-effective and value-adding AD, FR & RCA within streaming sensor environments, the used algorithms must be adaptive, context-aware, precise, interpretable and should require minimal human intervention. None of the currently available methodologies meet all these mentioned requirements. Knowledge-driven techniques are not adaptive and too time-consuming to construct & maintain. Data-driven methods are not context-aware and are often not interpretable. In this work, we have proposed FLAGS, a methodology that covers all those requirements.

First, we have combined the results of both data- and knowledge-driven techniques, and by using semantic filters enriched by metadata, a lot of the detected anomalies can be classified as known behavior. The opportunity to interpret occurring, data-driven anomalies by using semantic data, reduces the human involvement needed for the operators to find the correct alerts. Second, explaining the output of a data-driven model based on expert information is way more intuitive than deriving them through an experimental approach. Data-driven techniques are more expressive and can give valuable model insights when taking the metadata into account. Inspecting the data by visualizing the data with user- and anomaly-driven widgets gives the operators a necessary tool to investigate unknown system behavior.

Concretely, the whole end-to-end methodology with ML AD, ML RCA, Semantic FD/RCA, Semantic rule mining and a Dynamic dashboard which provides user feedback, is being tested using a predictive maintenance case in the railway domain. Differentiating between train or track problems is quite common in these domains. By applying a smart IoT device on multiple train rides, a single operator can already receive valuable insights. The proposed methodology gives the operator a new tool to investigate possible errors in the system. Their knowledge about the railway domain is used as the primary input, together with the feedback and information on possible anomalies and possible causes. Further evaluation of multiple cases is needed, as well as an in-depth evaluation to determine the reduction in human involvement. Additional future work will investigate how the rule mining component can be made more dynamic and can become visually available for the operator such that they can verify the newly generated knowledge and accompanied rules.

9. Code availability

The different components of the proposed FLAGS methodology are made available online in different repositories. This section gives an overview of each component/tool and the link where additional information can be found:

- Mappings tool (YARRML, RML Mapper, RMLStreamer):
  https://github.com/RMLio
- Matrix Profile code:
  https://github.com/IDLabResearch/seriesdistancematrix
- Semantic FD code:
  https://github.com/IBCNServices/StardogStreamReasoning
- FOLIO ontology + transformation scripts FMEA and FTA:
  https://github.com/IBCNServices/Folio-Ontology

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is part of the imec ICON project Dyversify, co-funded by imec, Belgium, VLAIO, Belgium, Renson Ventilation NV, Televic Rail & Cumul.io. Televic Rail, the manufacturer of the Cosamira Edge module, delivered the use case data, expert knowledge and additional metadata to evaluate the proposed methodology. Bram Steenwinckel (1SA0219N) is funded by a strategic base research grant of Fonds Wetenschappelijk Onderzoek (FWO), Belgium.

References

[1] S. Latif, H. Afzaal, N.A. Zafar, Intelligent traffic monitoring and guidance system for smart city, in: 2018 International Conference on Computing, Mathematics and Engineering Technologies (ICoMET), 2018, pp. 1–6.

[2] R. Saia, Internet of entities (IoE): a blockchain-based distributed paradigm to security, 2018, arXiv:1808.08809.

[3] J. Ren, S.-J. Cao, Incorporating online monitoring data into fast prediction models towards the development of artificial intelligent ventilation systems, Sustainable Cities Soc. 47 (2019) 101498.

[4] M.T. Mardini, Y. Iraqi, N. Agoulmine, A survey of healthcare monitoring systems for chronically ill patients and elderly, J. Med. Syst. 43 (3) (2019) 50.

[5] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing 262 (2017) 134–147.

[6] Using predictive analytics in anomaly detection and fault recognition, 2019, https://www.turbomachinerymag.com/using-predictive-analytics-in-anomaly-detection-and-fault-recognition/.

[7] M. Solé, V. Muntés-Mulero, A.I. Rana, G. Estrada, Survey on models and techniques for root-cause analysis, 2017, arXiv:1701.08546.

[8] S. Sendelbach, M. Funk, Alarm fatigue: a patient safety concern, AACN Adv. Crit. Care 24 (4) (2013) 378–386.

[9] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, J. Srivastava, A comparative study of anomaly detection schemes in network intrusion detection, in: Proceedings of the 2003 SIAM International Conference on Data Mining, SIAM, 2003, pp. 25–36.

[10] R.B. Miller, Response time in man-computer conversational transactions, in: Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I, 1968, pp. 267–277.

[11] E. Tjoa, C. Guan, A survey on explainable artificial intelligence (XAI): towards medical XAI, 2019, arXiv:1907.07374.

[12] D.H. Stamatis, Failure Mode and Effect Analysis: FMEA from Theory to Execution, ASQ Quality press, 2003.

[13] W.-S. Lee, D.L. Grosh, F.A. Tillman, C.H. Lie, Fault tree analysis, methods, and applications a review, IEEE Trans. Reliab. 34 (3) (1985) 194–203.

[14] I. Lykourentzou, K. Papadaki, A. Kalliakmanis, Y. Djaghloul, T. Latour, I. Charalabis, E. Kapetanios, Ontology-based operational risk management, in: 2011 IEEE 13th Conference on Commerce and Enterprise Computing, IEEE, 2011, pp. 153–160.

[15] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al., The SSN ontology of the W3C semantic sensor network incubator group, Web Semant.: Sci. Serv. Agents World Wide Web 17 (2012) 25–32.

[16] B. Steenwinckel, P. Heyvaert, D. De Paepe, O. Janssens, S. Vanden Hautte, A. Dimou, F. De Turck, S. Van Hoecke, F. Ongenae, Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses, in: 9th International Semantic Sensor Networks Workshop, Co-Located with 17th International Semantic Web Conference (ISWC 2018), Vol. 2213, 2018, pp. 17–31.

[17] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, Hermit: an OWL 2 reasoner, J. Automat. Reason. 53 (3) (2014) 245–269.

[18] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical owl-dl reasoner, Web Semant.: Sci. Serv. Agents World Wide Web 5 (2) (2007) 51–53.

[19] M. Schaaf, S.G. Grivas, D. Ackermann, A. Diekmann, A. Koschel, I. Astrova, Semantic complex event processing, Recent Res. Appl. Inf. Sci. (2012) 38–43.

[20] P. Bonte, R. Tommasini, E. Della Valle, F. De Turck, F. Ongenae, Streaming MASSIF: Cascading reasoning for efficient processing of IoT data streams, Sensors 18 (11) (2018) 3832.

[21] K.A. da Costa, J. ao P. Papa, C.O. Lisboa, R. Munoz, V.H.C. de Albuquerque, Internet of things: A survey on machine learning-based intrusion detection approaches, Comput. Netw. 151 (2019) 147–157.

[22] I. Cramer, P. Govindarajan, M. Martin, A. Savinov, A. Shekhawat, A. Staerk, A. Thirugnana, Detecting anomalies in device event data in the IoT, in: IoTBDS, 2018, pp. 52–62.

[23] M. Gupta, J. Gao, C.C. Aggarwal, J. Han, Outlier detection for temporal data: A survey, IEEE Trans. Knowl. Data Eng. 26 (9) (2013) 2250–2267.

[24] D.J. Hill, B.S. Minsker, Anomaly detection in streaming environmental sensor data: A data-driven modeling approach, Environ. Model. Softw. 25 (9) (2010) 1014–1022.

[25] D.D. Paepe, S.V. Hautte, B. Steenwinckel, F.D. Turck, F. Ongenae, O. Janssens, S.V. Hoecke, A generalized matrix profile framework with support for contextual series analysis, Eng. Appl. Artif. Intell. 90 (2020) 103487.

[26] C.-C.M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H.A. Dau, D.F. Silva, A. Mueen, E. Keogh, Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 1317–1322.

[27] M. Hushchyn, A. Sapronov, A. Ustyuzhanin, Machine learning algorithms for automatic anomalies detection in data storage systems operation, Adv. Syst. Sci. Appl. 19 (2) (2019) 23–32.

[28] N. Kant, M. Mahajan, Time-series outlier detection using enhanced k-means in combination with PSO algorithm, in: Engineering Vibration, Communication and Information Processing, Springer, 2019, pp. 363–373.

[29] P. Chemweno, L. Pintelon, P. Muchiri, et al., I-RCAM: Intelligent expert system for root cause analysis in maintenance decision making, in: 2016 IEEE International Conference on Prognostics and Health Management (ICPHM), IEEE, 2016, pp. 1–7.

[30] M. Shokoohi-Yekta, Y. Chen, B. Campana, B. Hu, J. Zakaria, E. Keogh, Discovery of meaningful rules in time series, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1085–1094.

[31] A. Alaeddini, I. Dogan, Using Bayesian networks for root cause analysis in statistical process control, Expert Syst. Appl. 38 (9) (2011) 11230–11243.

[32] P. Van den Kerkhof, J. Vanlaer, G. Gins, J.F. Van Impe, Contribution plots for statistical process control: Analysis of the smearing-out effect, in: 2013 European Control Conference (ECC), IEEE, 2013, pp. 428–433.

[33] R. Saia, S. Carta, D.R. Recupero, G. Fenu, M.M. Stanciu, A discretized extended feature space (DEFS) model to improve the anomaly detection performance in network intrusion detection systems, in: Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Vienna, Austria, 2019, pp. 17–19.

[34] H.H. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Ensembles of incremental learners to detect anomalies in ad hoc sensor networks, Ad Hoc Netw. 35 (2015) 14–36.

[35] H.H. Bosman, G. Iacca, A. Tejada, H.J. Wörtche, A. Liotta, Spatial anomaly detection in sensor networks using neighborhood information, Inf. Fusion 33 (2017) 41–56.

[36] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.

[37] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J.M. Moura, P. Eckersley, Explainable machine learning in deployment, in: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 2020, pp. 648–657.

[38] P. Kamat, R. Sugandhi, Anomaly detection for predictive maintenance in industry 4.0-a survey, in: E3S Web of Conferences, Vol. 170, EDP Sciences, 2020, p. 02007.

[39] A.H. Sodhro, S. Pirbhulal, V.H.C. de Albuquerque, Artificial intelligence-driven mechanism for edge computing-based industrial applications, IEEE Trans. Ind. Inf. 15 (2019) 4235–4243.

[40] S.S. Khan, M.G. Madden, One-class classification: taxonomy of study and review of techniques, Knowl. Eng. Rev. 29 (3) (2014) 345–374.

[41] V.M. Janakiraman, D. Nielsen, Anomaly detection in aviation data using extreme learning machines, in: 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2016, pp. 1993–2000.

[42] S.K. Bose, B. Kar, M. Roy, P.K. Gopalakrishnan, A. Basu, ADEPOS: anomaly detection based power saving for predictive maintenance using edge computing, in: Proceedings of the 24th Asia and South Pacific Design Automation Conference, 2019, pp. 597–602.

[43] G.F. Luger, Artificial Intelligence: Structures and Strategies for Complex Problem Solving, Pearson education, 2005.

[44] H. Long, Aircraft oil system fault detection expert system, in: 2015 International Conference on Electromechanical Control Technology and Transportation, Atlantis Press, 2015, pp. 196–199, http://dx.doi.org/10.2991/icectt-15.2015.38.

[45] F. Curreri, S. Graziani, M.G. Xibilia, Input selection methods for data-driven soft sensors design: Application to an industrial process, Inform. Sci. 537 (2020) 1–17.

[46] F.A. Souza, R. Araújo, J. Mendes, Review of soft sensor methods for regression applications, Chemometr. Intell. Lab. Syst. 152 (2016) 69–79.

[47] J. Liu, Y. Li, X. Tian, A.K. Sangaiah, J. Wang, Towards semantic sensor data: An ontology approach, Sensors 19 (5) (2019) 1193.

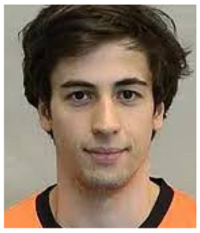
[48] S. Vanden Hautte, P. Moens, J. Van Herwegen, D. De Paepe, B. Steenwinckel, S. Verstichel, F. Ongenae, S. Van Hoecke, A dynamic dashboarding application for fleet monitoring using semantic web of things technologies, Sensors 20 (4) (2020) 32.

[49] Y. Jiang, B.K. Chen, C. Thompson, A comparison study of ride comfort indices between Sperling's method and EN 12299, Int. J. Rail Transp. 7 (4) (2019) 279–296.

[50] P. Umiliacchi, R. Shingler, G. Langer, U. Henning, A new approach to optimisation through intelligent integration of railway systems: the InteGRail project, in: Proceedings of the 7th WCRR, World Conference on Railway Research, 2006.

[51] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, R. Van de Walle, RML: a generic language for integrated RDF mappings of heterogeneous data, in: 7th Workshop on Linked Data on the Web, Proceedings, 2014, p. 5.

[52] G. Haesendonck, W. Maroy, P. Heyvaert, R. Verborgh, A. Dimou, Parallel RDF generation from heterogeneous big data, in: SBD2019, the International Workshop on Semantic Big Data, ACM Press, 2019, pp. 1–6.
[53] D. De Paepe, O. Janssens, S. Van Hoecke, Eliminating noise in the matrix profile, in: ICPRAM2019, the 8th International Conference on Pattern Recognition Applications and Methods, 2019, pp. 84–93.
[54] J. Lehmann, DL-learner: learning concepts in description logics, J. Mach. Learn. Res. 10 (2009) 2639–2642.



B. Steenwinckel holds a degree in Computer Science Engineering and is currently a Ph.D. student at Ghent-University-IDLab (since august 2017). His research focuses on the fusion of semantics and machine learning to enable adaptive, interpretable and accurate anomaly detection and root cause analysis.