

ONTOLOGY-BASED SENSOR HEALTH MONITORING: A DYNAMIC ONTOLOGY-BASED FAULT DETECTION AND CLASSIFICATION FOR TIMESERIES DATA

Colin Klein[1], Emy Arts[1], Bogdan Atanasoaie[1], Hendrik Meyer[1], Florian Raddatz[1] & Zamira Daw[2]

[1]German Aerospace Center (DLR), Institute of Maintenance, Repair and Overhaul, 21129 Hamburg, Germany
[2]University of Stuttgart, Institute of Aircraft Systems, Pfaffenwaldring 27, 70569 Stuttgart, Germany

Abstract

Predictive Maintenance methods are essential to the efficient operations of aircraft and mission-critical systems. Adoption of these methods, however, is still hindered by bottlenecks such as uncertainty quantification, data harmonization, and data quality, as the literature mainly focuses on the development of methods based on select, cleaned datasets or synthetic data. In the Predictive Maintenance domain, there currently are no known Sensor Health Monitoring (SHM) methods for harmonizing data and determining data quality that tackle these issues systemically. This work presents a method for monitoring sensor health using an ontology-based representation of structured system metadata. The proposed approach uses ontologies to store static model-based simulation information for the Sensor Health Monitoring, including redundant sensor subsystems relationships. In addition, aircraft system information from tabular and hierarchical data sources is converted into ontologies to increase semantic expressiveness and support automated reasoning. The developed ontology-based Sensor Health Monitoring (OSHM) algorithm is validated on experimental data from the DLR's ISTAR, a Falcon 2000LX aircraft, which shows the detection of various faults in the aircraft's experimental sensors. The results show that the framework is able to integrate metadata from various sources and can be scaled for any asset producing time series data and having structured metadata information available. Contrary to most data-driven methods, this approach has the advantage of being deterministic, allowing method interpretation of results for Predictive Maintenance analysis and uncertainty quantification. Additionally, the ontology provides a system level overview and allows capturing expert knowledge in an expressive and reusable form.

Keywords: Sensor Health Monitoring; Ontologies; Fault Detection; Prognostics and Health Management; Aircraft

1. Introduction

Predictive Maintenance (PM) enables safer, more efficient, and environmentally responsible aircraft operations by predicting failures and estimating the remaining useful life (RUL) of components using simulations and reduced system models [1]. This allows maintenance actions to be scheduled proactively, avoiding the limitations of reactive and time-based preventive strategies and reducing unplanned downtime [2]. With global maintenance, repair and overhaul (MRO) costs reaching approximately US $94 billion in 2023, equaling about 11% of airline operating costs, the economic relevance of PM is substantial [3]. PM can reduce maintenance expenses by up to 30%, translating into 1–3% lower operating costs and improved fleet availability [4]. While sensor-level PM in the form of Sensor Health Monitoring (SHM) has been demonstrated [5], system-wide solutions and scalable data management remain open research challenges. These challenges are increasingly addressed within the research domain of digital twins.

Digital twins support PM by enabling efficient storage, modeling, and analysis of large and heterogeneous datasets [6]. However, their implementation faces challenges such as data harmonization,

data quality assurance, and accurate modeling of system relationships. Integrating previously unused data sources remains largely manual, time-consuming, and error-prone. Furthermore, ensuring sensor data quality typically requires extensive expert involvement. Despite emerging approaches that use large language models (LLMs), maintaining a system-wide, linked metadata overview is still not nearly fully automated. [7]

Currently, few methods in the domain of SHM exist. These SHM methods mainly focus on fault detection and diagnosis (FDD), with limited integration of metadata and ontologies. Ontologies offer a structured way to represent complex system interactions and lifecycle relationships. These relationships are challenging to capture effectively in conventional databases of tabular (SQL, Excel) or hierarchical (MongoDB, XML) structure. Where relationships become complicated, ontologies enable semantic knowledge graphs. One method for describing ontologies is by using ontology web language (OWL) to write and SPARQL protocol and RDF query language (SPARQL) to retrieve information. In this domain, ontology-based SHM remains underexplored, as current ontology-based time series processing is generally based on rules and lacks further system information. Previous work proposed a mixed method approach to SHM that integrated fault detection mechanisms towards health index (HI) and RUL estimation that was built on conventional data formats. [8]

This work proposes an ontology-based method for system-wide SHM using structured metadata. A brief overview of the related work in the field of SHM, ontology-based fault detection, and simulation methods is presented first. Within the ontology-based Sensor Health Monitoring (OSHM) development process, the methods for constructing the ontology-based system are illustrated, detailing the data mapping and querying. Then, the implementation and used artifacts are shown and the use case is described. The use case drives the results that are shown, and discussed next. Last, we conclude and look to future developments.

2. Related Work: State of Sensor Health Monitoring and Ontologies

Current contributions for data quality assurance by SHM include methods such as fault detection (FD) [9], fault classification (FC) [10], fault prediction (FP) [11], and RUL estimation [12]. The contributions are mostly rule-based or non-deterministic, such as the stochastic methods for FP and RUL estimation [11, 12]. The focus of these contributions generally lies in control systems research, with few focusing on scalability and data management, the relevant topics for application in digital twins.

For scaling up SHM methods, ontologies are a vital technology to expressively model system graphs for powerful system queries. Ontologies towards SHM are currently being researched in engineering but generally at small scales such as in a review [13], in prognostics and health management (PHM) solutions such as maritime engine use-cases [14], building information management (BIM) [15], process engineering [16], and soil simulations [17].

Previous work created a taxonomy to address SHM as shown in Figure 1. The taxonomy was developed to integrate fault categories into a concise model for simple diagnosis of heterogeneous sensor measurements. The method was validated on the aircraft sensor data generated by the In-flight Systems and Technology Airborne Research (ISTAR) aircraft, a modified Falcon 2000LX business jet operated by the German Aerospace Center (DLR). The SHM taxonomy classifies faults into three levels of complexity to increase the diagnostic ability.

In level 1, the availability of sensors is checked. Sensors could be faulty and not delivering data. In level 2, a signal analysis for the single sensor time series is performed to detect unexpected data types, outliers, and to quantify the noise. A proposed method for noise outlier detection was to isolate the signal using a short-time Fourier transform (STFT), retrieve the noise by subtracting this signal from the sensor measurements, and quantify the amplitude of this noise. The amplitude was then checked against expected limits. In level 3, redundant and physically related systems are cross-checked for faults by employing methods from the domain of Global Navigation Satellite System (GNSS) for redundant systems and parity equations for physically related parameters. Deficits show in the detail of interactions in this method as relations between systems can at most be modeled hierarchically and not in higher-degree graph-based modeling. [18]

Some of the challenges encountered by our previous work laid in the generation of ontologies that remained a largely manual process. To alleviate this issue, some contributions used natural language processing (NLP) by artificial intelligence (AI) methods. By employing AI in ontology generation, the

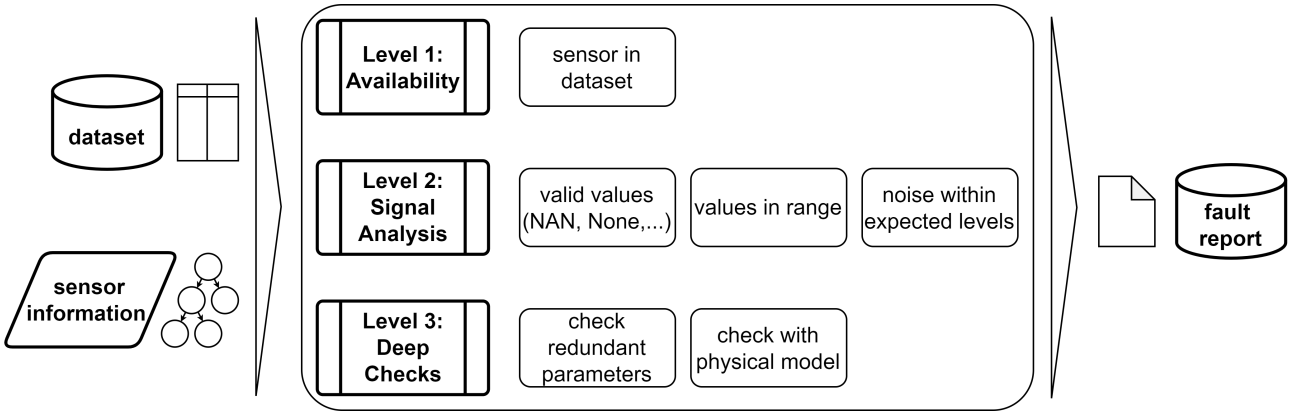


Figure 1 – The SHM process ingests sensor data as well as system and sensor metadata about the examined system. The process is split into three levels that examine availability, the sensor signals by themselves, and redundant as well as physically-related sensors. Results are stored in a fault report. [18]

reliability of the ontology decreases. [19, 20] To increase reliability in ontology generation, this work examines an ontology mapping for static metadata sources. Using this method, metadata structures are mapped to an ontology which can then be dynamically populated from metadata sources.

3. Method

The focus of this paper is addressing the deficits of the previous work, improving the links between data sources by employing ontology methods. To achieve this, an application ontology is proposed that describes static semantics for data sources. This semantic mapping is then employed to automatically generate an on-demand ontology to efficiently query over linked data sources using ontology queries. The specific architecture for the ontology-based SHM is based on a harmonization of three metadata sources into a single ontology, as shown in Figure 2. This figure shows the SHM method (see Figure 1) embedded into a metadata mapping process. The ontologies are used to bundle the relevant source data into a single ontology. The source metadata represented in the figure are the specific tabular sensor metadata, the hierarchical system information, and the hierarchical model information. To reason over all this metadata the ontology is then queried in the SHM method and retrieves data sources dynamically to detect configured fault detection methods and then retrieve sensor metadata. This sensor data can then be used to get time series data from a remote database. After performing the fault detection in the time series data, results are returned and stored in the time series database where the sensor data is also stored. Using these results, a web-based dashboard allows the user to visualize the SHM monitoring.

Three specific data sources are present in the ISTAR aircraft use case. First, general static sensor metadata is available in a tabular Excel format. This tabular data contains the data acquisition configuration information of the experiment such as sensor limits set by the experimentator and tags which sensors are redundant to each other such as *barometric pressure*.

Second, Data Acquisition System (DAQ) information is represented in a hierarchical format by usage of JSON or XML files. DAQ information is more dynamic than the general sensor metadata and comprises the latest configuration file of the experimental system setup. The contents of the DAQ information are updates in expected parameters and updated parameter attributes such as sampling rates.

Third, model information in a hierarchical format is present, defining which checks are performed and referencing equations for physical relations between sensors. Sensor and parameter references are stored in metadata from aforementioned sources and used by the SHM checks. To start the analysis of sensors, the SHM runs per flight and checks the sensors using methods developed in previous work as shown in Figure 1. To visualize the results, either the SHM output file can be directly read or visualized with the SHM dashboard. What also can be visualized is the ontology. This ontology can not only be visualized in the Protégé Ontology Editing Software [21] but also queried efficiently with

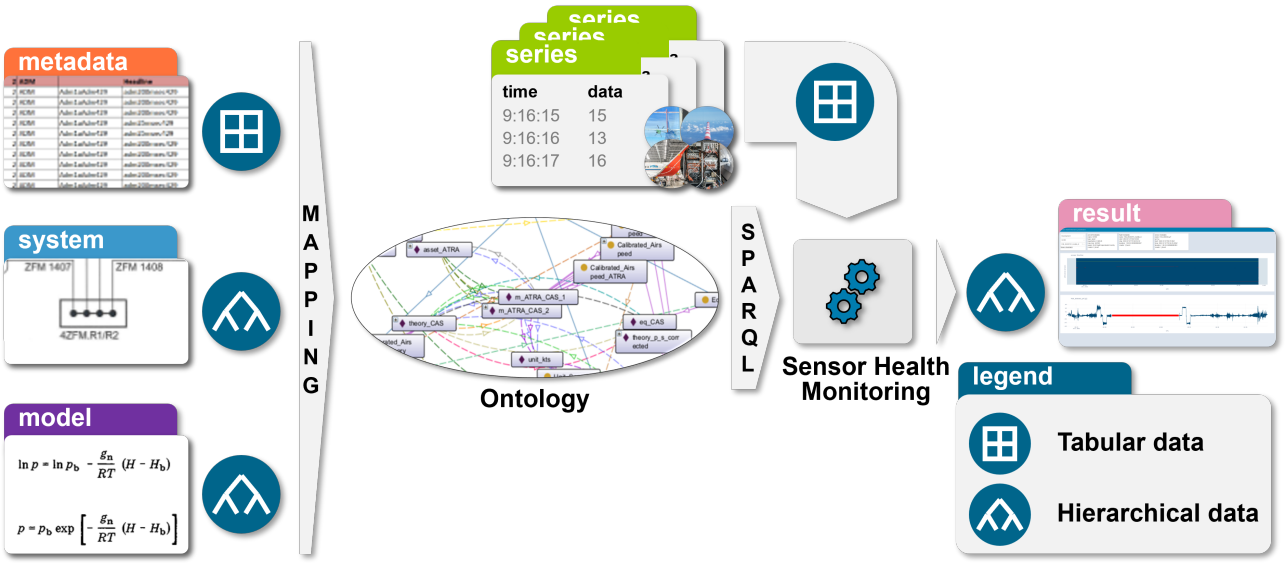


Figure 2 – OSHM is based on a collective aircraft ontology and sensor time series. The SHM inputs are general sensor metadata in tabular format, system metadata in a hierarchical format, and physical model data in hierarchical format. These inputs are converted into ontologies which the SHM can use as input. After processing, the SHM outputs the results in a hierarchical data format that can be visualized in a dashboard.

the triple query language SPARQL. In the following, the proposed low-level ontology is visualized with its class structure in Figure 3.

This class structure represents the following data sources from Figure 2 with the following structure: Time series database origin with asset, collection and time series representation. The sensor information as a merged place of static and dynamic sensor metadata. Finally, the SHM levels, checks and results. For the ISTAR aircraft, the ontology model is detailed in the following.

On the right, the structure of the time series database is represented. The asset is the top-level item which represents the specific system that is measured. In our instance, this represents the ISTAR. Below, collections inherit from the asset. Collections represent single experiments and measurements that generally possess defined start- and endpoints such as flight experiment recordings for the ISTAR. Within these collections, time series represent a single sensor datum with a tuple of time and value. In the ontology, the time series class stores the length of its time series, the sensor from which the metadata can be pulled and the database ID with which it can be queried in the database API. In the ISTAR use case, this can be the *barometric altitude 1* sensor value for the left-sitting pilot

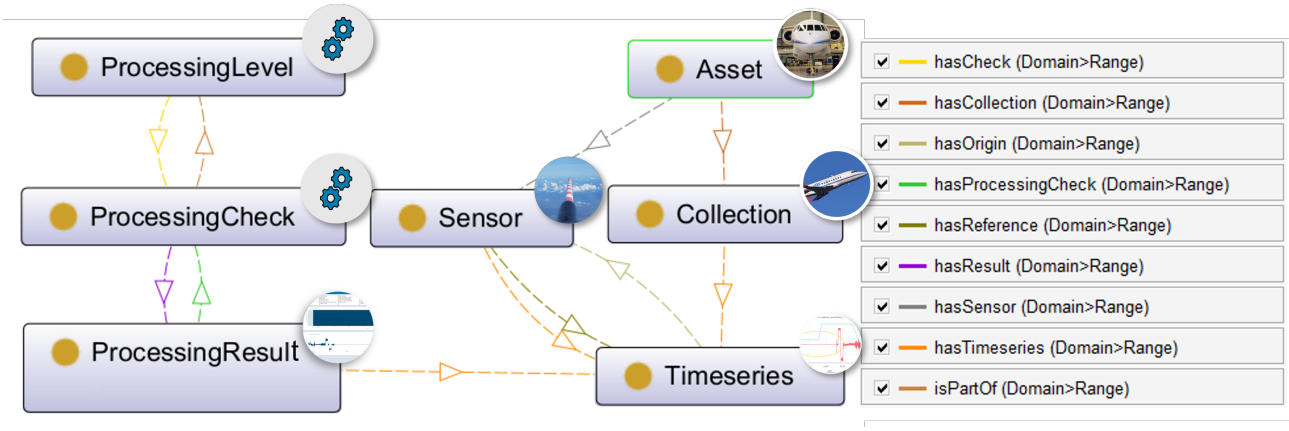


Figure 3 – OSHM classes separate into asset, collection, time series and sensor description as well as SHM information regarding processing level, check and result. A legend on relationships between classes is shown on the right.

in command. The time series database connects to the sensor class. This sensor class represents parameter metadata and represents joint Excel and XML data in the ontology class. Its attributes contain the parameter name that maps onto the time series database and the *hasUnit* attribute that stores the unit in a plain string.
In the leftmost column, the asset-specific SHM configuration is stored. The processing level forms the top level of the ontology. It is followed by the processing checks that define the single checks such as the frequency, limit or noise check. These checks need four data points: error tag, minimum, and maximum limits, as well as a transformation function. The error tag, that stores the errortag that will later represent the user interface to describe the detected fault. In the attributes *hasMin* and *hasMax*, the previously defined sensor attributes are stored. For the boundary check, this is *hasPhysicalMin* for the *hasMin* value and *hasPhysicalMax* for *hasMax*. The transformation function maps to a python function that transforms the time series into another scalar value. For the noise this can transform an accelerometer value into a noise value that can be checked against limits for fault detection. For the last ontology class, the processing result attribute stores detected faults for each check and sensor. The attributes of the results store the processing check information, as well as the tuples of the timestamps and values that were checked against limits and detected as faulty.
The presented approach solves the problem of past time-intensive ontology generation. By introducing a semantic mapping to existing static data sources, we can enrich and link existing data dynamically.

4. Implementation

In this section, we present the implementation of the OSHM framework including the ontology mapping and querying process. The mapping flow of the ontology with the instance population of methods, sensors and time series is described in the following and illustrated in Figure 4. In the upper box, the significant python operations are shown while the lower box shows the ontology operations in the python library owlready2. The method starts by receiving the collection ID for the flight experiment, creating a new ontology description in OWL format by using the python library owlready2. The next step consists of loading the class structure as proposed in Figure 3. This class structure consists of the class nodes and is predefined in the python owlready2 class description. The class attributes are attributes that are either data types or other class nodes and the connecting edges. To start populating the ontology, the collection's parent is fetched as the asset is required for populating the ontology and also for retrieving the SHM checks. After retrieving the checks, they are written to the ontology. Subsequently, the sensor metadata is retrieved. These descriptions are first loaded from an asset-specific Excel file. Upon going through an asset-custom mapping to the ontology attributes as defined in Figure 3, the sensor attributes are ready to be added to the ontology. The sensor metadata is stored by the identifier of sensor parameter ID. For development purposes, the current model

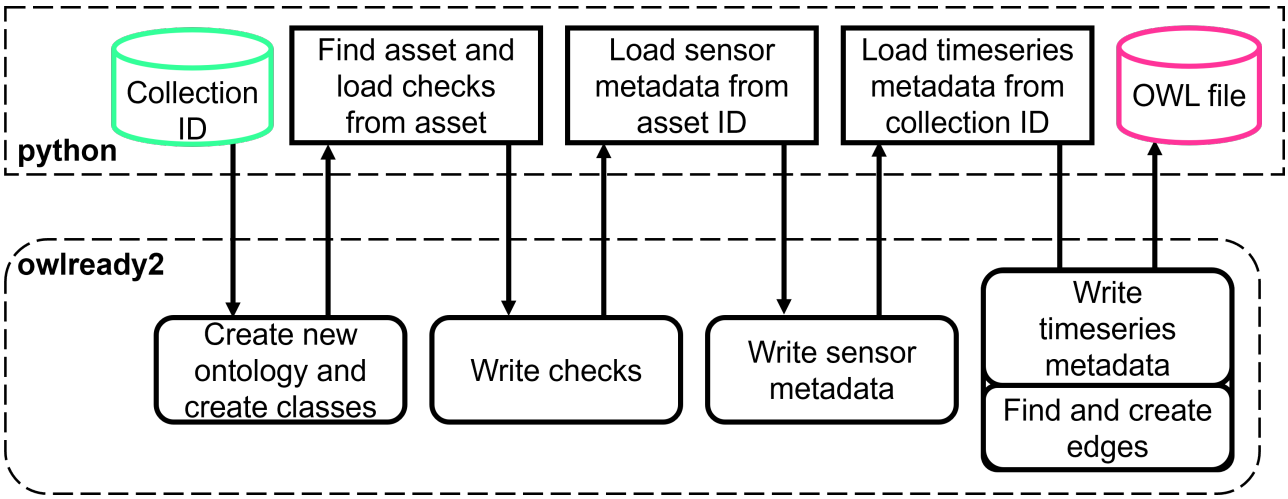


Figure 4 – The process on ontology mapping as introduced in Figure 2 is illustrated from input in green to output in red. Steps are associated to either python or ontology operation.

references ontologies as strings. In the sensor class, the attributes are mapped from the sensor metadata source (Excel file). The attributes consist of a free text parameter *hasDescription* attribute and the *hasDataType* attribute that defines if the parameter outputs str, int, float or other values. The *hasFrequency* attribute references the expected sensor sampling rate, a parameter checked in the SHM checks. In the *hasPhysicalMin* and *hasPhysicalMax* attributes limits for the SHM checks are stored that perform level 2 boundary checks. In the extended limit attributes *hasAmplitudeMin* and *hasAmplitudeMax*, limits for a custom level 2 check are stored that estimates the sensor amplitude for noise plausibility checks. The *hasTag* parameter stores level 3 redundancy tags through which sensor redundancies and physical relations are assigned. Finally, the attribute *hasTimeseries* maps onto the time series parameter by use of the time series and sensor name. When mapped, this attribute can be queried via SPARQL to enable deep queries into sensor data as shown in Figure 5.

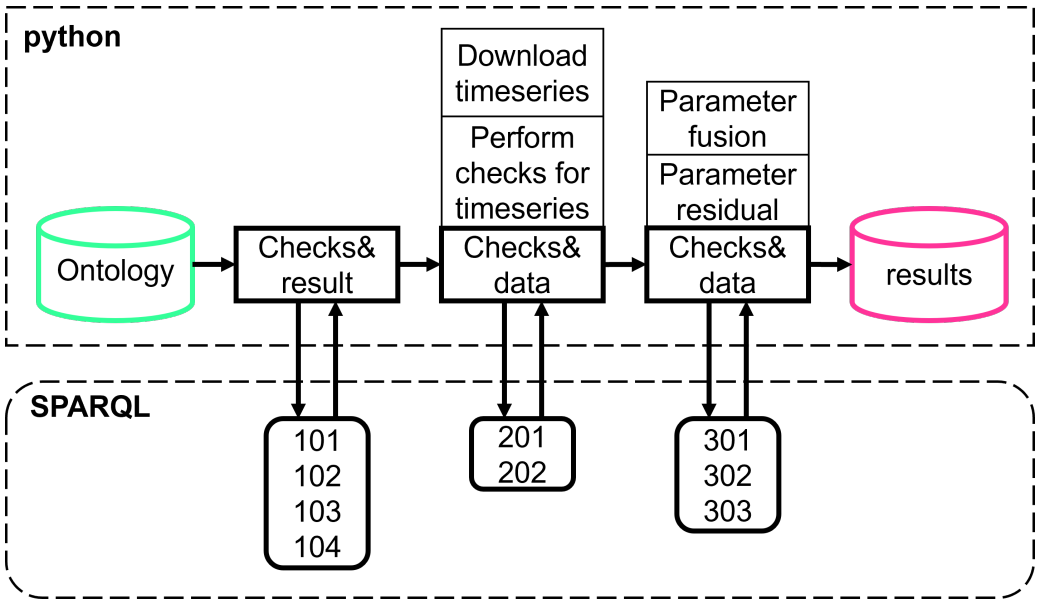


Figure 5 – The process on ontology querying as introduced in Figure 2 is illustrated from input in green to output in red. Steps are associated to either python or SPARQL operation. SPARQL codes are detailed in Table 1.

The time series are added once the sensor metadata is written. The method starts by searching for the time series associated to the given collection. Upon retrieval, all time series are written to the ontology. After ingesting all data, edges are created from the existing data. This includes firstly connecting time series to their sensor metadata if available. This connection is achieved by comparing name strings. Next, some sensors have referenced parameters such as the barometric altitude that needs a reference pressure to be converted from pressure altitude. This reference parameter can also be found by mapping by name and is also created. The connection of other edges such as the linking of processing results is performed after the SHM has run and operates analogously. After the ontology building has taken place, the ontology is stored in an OWL file.
Upon having created the ontology we face the challenge of walking the graph. Using SPARQL, we can assert triples and query the graph efficiently. These queries allow to reason over a graph and extract information effectively. The query language is based on triples to query relationships between objects using subject predicate and object notation. The return value of SPARQL outputs a table. In Figure 6, an exemplary query is shown detailing the constraints that ISTAR sensors need to be queried. The information we need to find all sensors is the attribute that the sensors have towards the ISTAR and the ISTAR itself. We further define a variable to fill in all nodes that fulfill the set criteria. Finally, we close the triple with a full stop. In the following, a sample query is shown:

$$:istar \ :hasSensor \ ?sensor \ .$$

When integrated into a full SPARQL query, some components are required to input more information to the reasoning algorithm. First, we need to integrate the ontologies that are relevant for the query by

using the keyword "PREFIX", followed by a synonym that needs to end with a colon such as "shm:". Since we only reason over one ontology, we can omit the string before the colon for brevity and assign the colon to the OSHM ontology. The ontology source is then defined within the angle brackets. Having defined the imports, the return values are defined by the "SELECT" keyword. Within the "WHERE" keyword the previously introduced triples are embedded and can be extended depending on the constraints needed.

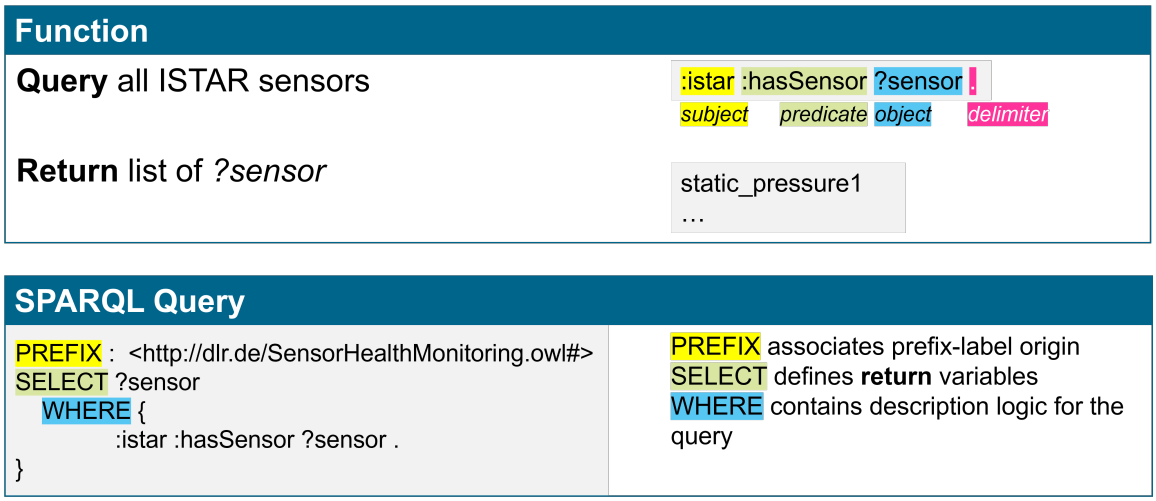


Figure 6 – Description of the SPARQL query syntax. In the Function box, the simplified inputs and outputs of a single SPARQL query are shown. Below, a functional SPARQL query is shown.

The SPARQL queries are started for each of the SHM levels. In level 1, the checks are queried from the ontology by acquiring checks and fault descriptions connected with the ontology node *level 1*. Once acquired, the checks are performed by finding all time series that have a corresponding sensor. As correlating entities have been assigned in the previous mapping step, the query for time series with available sensor information lies in searching for time series that have an origin of a sensor node which in turn is of the type sensor. Once all these pairs of time series and sensor have been found, the remainder are quantified by getting the orphaned time series without sensor information. As SPARQL allows filter functions we can use the function "FILTER NOT EXISTS" to search for time series that do explicitly not have a sensor origin node linked. This same filter function is also used to find orphaned sensors by keeping the filter logic but searching for sensors instead of time series.
For level 2, the process requires more steps. We start similarly to level 1 and get level 2 checks. Instead of check names and error tags, we also get the corresponding sensor attribute names for minimum and maximum limit as well as the name of a possible signal transformation function for sampling check and noise approximation for this work's case. For the processing, checks are performed per variable to reduce data traffic for sensor data download. Within the processing loop, a time series is downloaded once a corresponding sensor and at least one check is found. We then query the numerical limits by getting the sensor attributes referenced in the check description as minimum and maximum limits. With this information, we download the time series by its database ID, and process it according to the given transformation function to detect faults by comparing the resulting signal to the given limits.
The level 3 method detects the given SHM algorithms. For the previously developed parity equations algorithm, we query the error tag and transformation function. Within the parity equation, we query parameters for a given tag such as *barometric altitude*. The SPARQL query searches for any item of the type sensor with a linked time series that has a known tag as a tag string. The sensor name, database ID and tag name are then returned. In addition to the tags, some parameters require further information such as referenced parameters like the *reference pressure* to derive the barometric pressure from the pressure altitude. As we input the referenced parameters in the mapping stage previously, the query consists of querying for all reference parameter names and IDs for a given time series database ID. It was attempted to represent the information for the barometric altitude equation

in sympy and store it within ontologies in a single source of truth. This was considered not feasible as the equation needs to be converted for each time tuple. As the equation has components in the exponent, the processing begins stretching for a long amount of time if the equation needs to be solved for the exponent. For a barometric parameter with a regular sampling of 20Hz over one hour this easily becomes about 72.000 time data points. A prospective solution could be a compilation step in which the equation is compiled into a format considering all unknown parameters. In case of the full barometric ISA compliant equation, [22] the formula has steps for atmospheric temperature gradients that can be modeled and modularized in python relatively effectively. Stepping towards sympy, the development effort rises as legibility decreases.
In addition to employing owlready2 for inputting and querying the ontology, the ontology software Protégé is used in this work to examine, query, debug, and visualize the automatically generated ontologies. [21]

5. Case Study

We use the ISTAR to validate the developed method. The ISTAR is a Dassault Falcon 2000 LX aircraft, operated by the DLR at its Brunswick site to examine flight characteristics and enable research various aircraft as a research object. The ISTAR is an aircraft going through various temporary and permanent modifications to its airframe and sensor instrumentation. The aircraft possesses a permanent as well as temporary sensor instrumentation. [23]
The permanent instrumentation is shown in Figure 7 and consists of three main sensing modules as well as a DAQ. The DAQ is the central system in which all sensor signals are recorded which is shown as Measuring Unit (MU) in the figure. The remaining three boxes in the figure represent the sensing modules. These sensing modules are the ASCB data bus, the IMAR positioning unit, and the experimental sensors. The ASCB data bus is the main aircraft data bus, providing up to 46.000 bus parameters of which about 232 relevant aircraft parameters are permanently recorded by the DLR's DAQ. The IMAR is a secondary Inertial Measurement Unit (IMU) and GNSS receiver with sub-meter accuracy that provides 66 parameters for position and acceleration. The aircraft's aeroelastics sensors amount to 105 permanently installed aeroelastics sensors illustrated by the red dots in the figure. These aeroelastics sensors are placed at structurally relevant points and include strain gauges for measuring structure response to ground and air travel as well as accelerometers to measure vibrations and quantify forces. In addition to the permanent instrumentation, a semi-

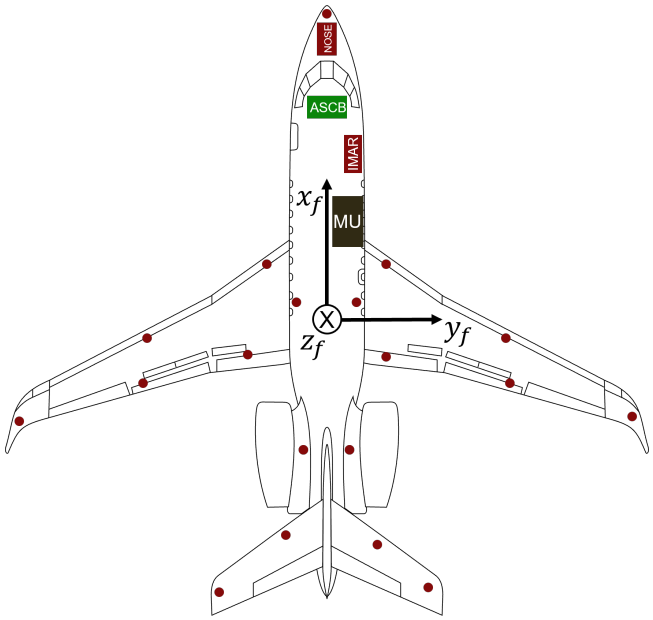


Figure 7 – The DLR's ISTAR sensor instrumentation. Red dots show sensor positions of strain gauges and accelerometers. The main, permanent measurement systems are referenced as the four boxes of NOSE, ASCB, IMAR and Measurement Unit (MU). [24]

permanent addition is the removable noseboom that can be installed based on the flight requirements. Next to this instrumentation, the ISTAR has the option to install measurement racks inside the aircraft if an experiment requires additional computation power or instruments.
The information about sensor placements lies distributed among an Excel file that contains static sensor and parameter information, DAQ information that is updated for each measurement and finally the time series metadata generated in the recordings. First, the Excel file describes fundamental properties of the measured parameters such as sampling frequency, and unit as well as parsing information and the component and bus in which the sensor is installed. The intention of this data source is to contain all sensor metadata that remains more or less static. Second, the DAQ configuration contains all information related to the current measurement, including the latest sampling rate, unit and also source data. Third, the time series contain metadata themselves regarding their measurements which should fundamentally be similar to the DAQ information but has been shown to act discordingly.

6. Results

The results towards the populated ontology and the SPARQL queries are shown and runtimes are compared.

6.1 Ontology Population

The generated ontology consists of a total of about 1873 individual nodes. Based on the given ISTAR flight we get 457 time series and 1403 sensors. In addition, we can see three SHM levels as well as 8 individual check methods. Regarding edges, the aircraft and asset each have one connection. The SHM levels have 25 edges and the SHM checks have 70 edges. With a stark increase, the time series nodes follow with 5322 edges, stemming from the amount of attributes and connections for each time series. The most connections are apparent in the sensors, with 16908 edges representing the various attributes stored in the sensor individuals.

6.2 SPARQL

Table 1 categorizes the SPARQL queries as detailed by [25]. The table columns comprise an ID referencing the SHM level, the different topics, the question representing the query and the number of results. In level 1, there are 4 queries. In query 101, check information about the present checks is retrieved from the ontology, including the check ID and the error tag that will be affixed to sensors for diagnosis and contains the fault description. For comparing the set of sensors and time series we utilize three queries to determine the overlap (102) and the two remaining excluded sets for both sensors (103) and time series (104).
For level 2, we start similarly by retrieving checks in query 201. The return values consist of check ID, error tag and also the names of sensor limit attributes that point to the numerical boundary values that are necessary for level 2 checks. Query 201 returns the label of the transformation function that can point to e.g. the noise transformation function that receives the sensor data, transforms it into a quantification of noise and allows this quantification to then be checked against limits. In query 202,

Table 1 – An illustration of SPARQL queries with their respective IDs, topic and reference question.

ID	topic	reference question	$n_{results}$
101	SHM	What checks exist in level 1?	3
102	Check	For which time series does sensor information exist?	403
103	Check	For which sensor does no time series exist?	1000
104	Check	For which time series does no sensor exist?	54
201	SHM	What checks exist in level 2?	3
202	SHM	What are the attributes for the given level 2 check?	586
301	SHM	Which checks exist in level 3?	1
302	sensor data	What parameters exist for tag x?	78
303	sensor data	What referenced parameters does parameter x have?	6

attributes can be queried by their parents by inserting the two custom strings into the SPARQL query. For the limit checks, sensor specific numerical limit for the boundary checks can be retrieved.
For the final level 3 check, we start the query similarly to the checks before by retrieving the defined checks (301). As level 3 fuses redundant values and transforms parameters we assign via tag strings. For the checks we have known tags for which we want to acquire sensors so we query the ontology for sensor IDs, time series IDs and their tags (302). Finally, with the last query 303 there is a need to find sensor parameters with referenced parameters. This reference is assigned for parameters such as the barometric altitude which requires a reference pressure set by the pilot to convert from a pressure altitude. The ontology can be queried for all reference parameters that belong to a given parameter.
A query used in the OSHM framework is detailed in the following:

```
1    SELECT  ?sensorId ?reference ?referenceId
2    WHERE {
3        ?sensorId a :Sensor ;
4                   :hasReference ?reference .
5        ?reference a :Sensor .
6        ?referencets :hasOrigin ?reference .
7        ?referencets :hasTwinstashId ?referenceId .
8    }
```

Query 303 to retrieve referenced parameters from a sensor's tag.

303: **What referenced parameters does a given parameter possess?** The answer consists of the original sensor ID (barometric altitude), the reference sensor's name ($baro_correction_1$) and the referenced sensor's time series database ID. For querying, the sensors are identified by asserting that the variable *?sensorId* gets all entities of the type sensor. Note: The "a" between *?sensorId* and :Sensor references the Ressource Description Framework (RDF)'s rdf:type and thus describes if something is type of something else. In the second line, the previously defined sensor is defined further by introducing the constraint that the sensor needs to have a referenced sensor. In line 3, the newly acquired reference sensor is defined further by introducing the constraint that it, too, needs to be of type sensor. The following line retrieves the sensor's connected time series that is required for the calculation and promptly also retrieves the referenced time series data base ID.

6.3 Runtime

The runtime in the OSHM version (commit 13df4ae1) was 4 minutes and 51 seconds versus a runtime of 7 minutes and 10 seconds in the previous SHM version without ontologies (commit a9c37923). The processing was performed on a laptop with an intel i7-1365U processor with 32GB of RAM and 100Mbit/s of network connection speed.

7. Discussion

This work focused on combining ontoloiges with SHM to build a metadata-driven OSHM method.
While the initial mapping showed great promises of consolidating different metadata, challenges remained at mapping data of larger amounts and intricate structures. Especially mapping the SHM results remained a challenge due to level 2 and 3 checks outputting time series by storing time stamps with fault detection result. The output format thus remained in the original JSON format and will be considered in the future to be translated into ontologies as querying with SPARQL shows promise for querying for parameter, fault or time window. Level 3 hierarchical system information posed challenges to convert the python dictionary variable depth recursive parsing method to ontologies. In addition, the static model-based simulation information remained in python such as the barometric equation. Last, units were not converted to semantic formats such as NASA's QUDT [26] due to lack of time.
The querying using SPARQL showed to be superior when it comes to addressing multiple linked hierarchical data sources compared to the methods used in the previous SHM designs. While the time to query metadata remained insignificant compared to the time series processing, SPARQL increased clarity and reduced loops in the software. The overall SHM processing time decreased. This can be attributed to the previous inefficient design that downloaded time series for each parameter and

check. In the ontology-based SHM, each parameter is still processed individually but downloaded time series are reused for other checks which reduced processing time by a third.
Outsourcing time series processing to python proved viable. Especially the interaction with querying the ontology for light-weight metadata in conjunction with processing-intense time series data acquisition and processing showed great promise. Solutions for data assertions in ontologies such as semantic web rule language (SWRL) were not considered due to the amount of data which ran at about 4GB per flight and 50.000 to 5.000.000 data tuples per sensor. Model information was left out such as the physical mapping. To represent mathematical information legibly for humans and optimized for computers in the same place is a known challenge and has not been sufficiently solved such as evident (see MathML duality of equations).

8. Conclusion

This paper presented an ontology-based Sensor Health Monitoring framework for metadata-driven fault detection and classification in aircraft sensor systems. The approach integrates heterogeneous metadata sources, including tabular sensor descriptions, hierarchical system configurations, and model information, into a unified semantic representation using ontologies. Validated on DLR's ISTAR flight-test data, the framework demonstrated that ontology-based querying can support system-level sensor health analysis while improving traceability, interpretability, and processing efficiency. The results show that ontologies are well suited to linking engineering metadata with operational time series data and to supporting deterministic reasoning for Predictive Maintenance applications. At the same time, complex numerical computations and mathematical models are better handled in dedicated processing environments, suggesting a hybrid architecture that combines ontology-based reasoning with conventional data processing. Future work will extend the ontology with additional aircraft system knowledge, semantic unit representations, and richer model descriptions. In addition, AI-based methods can be used to analyze sensor status separately and integrate the resulting assessments into the ontology for system-level reasoning.

9. Contact Author Email Address

Colin Klein, M.Sc.
German Aerospace Center (DLR)
colin.klein@dlr.de

10. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Vik Loxton, Emma; Krishnan. Aircraft mro 2.0:the digital revolution. McKinsey Company, 2024.

[2] Lukas Jilke. Investigation of degradation modeling for aircraft structures: A systematic literature review. AIAA, 2023.

[3] Geraldine Cros. Airline maintenance cost executive commentary. 2025.

[4] Seyyedabdolhojjat Moghadasnian. Ai-powered predictive maintenance in aviation operations. April 2025.

[5] J. C. D. Silva, A. Saxena, E. Balaban, and K. Goebel. A knowledge-based system approach for sensor fault modeling, detection and mitigation. *Expert Systems with Applications*, 39(12):10977–10989, 2012.

[6] Sebastian Haufe, Michael Bäßler, Christina Pätzold, Matheus Tchorzewski, Hendrik Meyer, Emy Arts, and Alexander Kamtsiuris. Digital twins storage and application service hub (twinstash). In *Deutscher Luft- und Raumfahrt Kongress 2022*, August 2023.

[7] Anna Sofia Lippolis, Mohammad Javad Saeedizade, Robin Keskisärkkä, Sara Zuppiroli, Miguel Ceriani, Aldo Gangemi, Eva Blomqvist, and Andrea Giovanni Nuzzolese. Ontology generation using large language models, 2025.

[8] Colin Klein, Florian Raddatz, Gerko Wende, and Zamira Daw. Sensor health monitoring based on ontologies. a systematic literature review. In *DLRK 2025*, September 2025.

[9] Vipin Mathew, Somnath Sengupta, Mayurika Chatterjee, and H. Srikanth Kamath. Sensor health monitoring using simple data driven approaches. pages 32–38, 2016.

[10] Tsechun Chen and Kaveri Mahapatra. Pmu data quality and sensor health monitoring. In *2024 IEEE POWER & ENERGY SOCIETY GENERAL MEETING, PESGM 2024*, IEEE Power and Energy Society General Meeting PESGM, 345 E 47TH ST, NEW YORK, NY 10017 USA, 2024. IEEE Computer Society.

[11] S. Ullas, B. Uma Maheswari, Seshaiah Ponnekant, and Thaluru M.Mohan Kumar. A three stage attention enabled stacked deep cnn-bilstm (asdcbnet) model for end-to-end monitoring of wastewater treatment plant. *Applied Water Science*, 15(8), July 2025.

[12] Aditya Tulsyan, Chris Garvin, and Cenk Undey. Condition-based sensor-health monitoring and maintenance in biomanufacturing. *IFAC PAPERSONLINE*, 53(2):177–182, 2020.

[13] C. Franciosi, Y. Eslami, M. Lezoche, and A. Voisin. Ontologies for prognostics and health management of production systems: overview and research challenges. *JOURNAL OF INTELLIGENT MANUFACTURING*, 36(4):2223–2253, 2025.

[14] G. Medina-Oliva, A. Voisin, M. Monnin, N. Kosayyer, and J.-B. Léger. A case study for fleet-wide semantic based predictive diagnostic. pages 1751–1760. shers, 2014.

[15] Henrik Dibowski, Ondrej Holub, and Jiri Rojicek. Knowledge-based fault propagation in building automation systems. In *2016 International Conference on Systems Informatics, Modelling and Simulation (SIMS)*, pages 124–132, ACS Global Laboratories, Honeywell, Prague, Czech Republic, June 2017. IEEE.

[16] Ville et al. Kukkonen. An ontology to support flow system descriptions from design to operation of buildings. *AUTOMATION IN CONSTRUCTION*, 134(WOS:000740339100005):104067, February 2022.

[17] Howard Beck, Kelly Morgan, Yunchul Jung, Sabine Grunwald, Ho-young Kwon, and Jin Wu. Ontology-based simulation in agricultural systems modeling. *Agricultural Systems*, 103(7):463–477, September 2010.

[18] Colin Klein, Christina Pätzold, and Matthias Bodenbenner. Fair sensor health monitoring of flight test data. Master's thesis, RWTH Aachen University, Juni 2023.

[19] B.; Wang S.; Pan J.; Huang Y.; Jiao X. Meng, X.; Jing. Fault knowledge graph construction and platform development for aircraft phm. *Sensors*, 24(231), 2024.

[20] Shuo; Lin Jia-Rui; Li Sun-Wei; Xiao Ya-Qi Hu, Zhen-Zhong; Leng. Knowledge extraction and discovery based on bim: A critical review and future directions. *ARCHIVES OF COMPUTATIONAL METHODS IN ENGINEERING*, 29(WOS:000639061200001):335–356, 2022.

[21] Mark A. Musen. The protégé project: a look back and a look forward. *AI Matters*, 1(4):4–12, 2015.

[22] DIN ISO. Standardatmosphäre DIN ISO 2533.pdf, 1978.

[23] Martin Gestwa, Dirk Leißling, and Matthew James Bruce. Der dlr-flugversuchsträger istar - status quo und quo vadis. In *Deutscher Luft- und Raumfahrtkongress 2023*, September 2023.

[24] Abdelwaheb El Houimel. Machine learning based fault detection in aircraft sensors, 2026.

[25] Jan Martin Keil, Tobias Köhler, and Sirko Schindler. Communication in design-for-circularity: Requirements to a knowledge graph. In *Proceedings of the The 2nd International Workshop on Knowledge Graphs for Sustainability (KG4S 2024)*, volume 3753 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2024.

[26] FAIRsharing Team. Fairsharing record for: Quantities, units, dimensions and types.