Masterarbeit

Name:	Colin	Klein
Matr.-Nr.:	368642	

Thema: FAIR Sensor Health Monitoring of Flight Test Data

Betreuender Assistent:	M. Sc.	Matthias	Bodenbenner
Externe Betreuerin:	Dipl.Ing.	Christina	Pätzold

Aachen, den 09.06.2023

Diese Arbeit wurde vorgelegt am Werkzeugmaschinenlabor WZL, Lehrstuhl für Fertigungsmesstechnik und Qualitätsmanagement.

Aachen, 9. June 2023

Master Thesis

for B.Sc. Colin Klein
Matriculation number: 368642

Topic: FAIR Sensor Health Monitoring of Flight Test Data.

The DLR's ISTAR research aircraft is equipped with extensive permanent sensor instrumentation for the scientific investigation of aerophysical phenomena. The measurement data as well as a large part of the parameters on the aircraft's own data bus are continuously recorded by an additional Data Acquisitioning System (DAQ). To evaluate measurement data in order to gain knowledge, the form of data storage and the linkage of fault detections is vital to avoid erroneous conclusions. By honoring FAIR (Findable, Accessible, Interoperable, Reusable) principles, the detected faults andaAnomalies shall aptly be linked to the measurement data. In this work, a python application is developed to solve this problem by detecting faults, sensibly linking them to the measurement data and then visualizing the results to the user. To solve this task, data modeling techniques developed at the WZL are employed. In addition, data is checked for completeness, plausibility and correctness by using statistical methods as well as approaches from the field of Fault Mode and Effect Analysis (FMEA). Finally, the performance and trueness of the application-toolchain is tested against known errors and validated on a dataset of ISTAR Flight Test data.

Prof. Dr.-Ing. Robert Schmitt

I Contents

II Acronyms

Symbol	Unit	Description
p	Pa	Pressure at current altitude
p_0	Pa	Reference pressure
h	m	Current altitude
h_0	m	Reference altitude
$T(h_0)$	K	Temperature at reference altitude
a	K/km	ISA Temperature gradient
g	m/s^2	Gravity constant (9.80665)
R	$m^2/(K \cdot s^2)$	Ideal Gas Constant (287.05287)
σ	-	Standard Deviation
σ^2	-	Variance
μ	-	Statistical Mean
u	-	Process Input
y	-	Process Measurement
x	-	Process State
ρ	-	Pearson Coefficient
PMF	-	Probability Mass Function
SNR	-	Signal-to-Noise Ratio
CV	-	Coefficient of Variation

Abbreviation	Description
ADC	Analog Digital Converter
FX	Flight Experiments
FTI	Flight Test Instrumentation
DLR	Deutsches Zentrum für Luft- und Raumfahrt
RWTH	Rheinisch-Westfälische Technische Hochschule
SHM	Sensor Health Monitoring
FMEA	Failure Mode and Effects Analysis
GNSS	Global Navigation Satellite System
IoT	Internet of Things
IoP	Internet of Production
SOIL	SensOr Interfacing Language
ML	Machine Learning
DAQ	Data Acquisitioning System
ASCB-D	Avionics Standard Communications Bus, Version D
FAA	Federal Aviation Administration
DQI	Data Quality Indicator
SI	International System of Units

III List of Figures

IV List of Tables

1 Introduction

Experimental Research Data is essential to confirm or nullify scientific theories. Motivated by the drive for sustainability the idea of Research Data Reuse culminated in the 2016 FAIR Guiding Principles [WILK16]. These principles stand for Findable, Accessible, Interoperable and Reusable data signifying attributes that enable the sharing and colaboration of data. These principles then catalyze data reuse to facilitate Open Data Access and Big Data within the scientific field [HODS18]. In this work research data of the ISTAR (In-flight Systems & Technology Airborne Research) aircraft of the DLR (German Aerospace Center) will be examined to detect faults present in the data. This improves reusability and will notify future users of any known anomalies within the data in order to gain insights into the data and increase the data quality. In the context of the data producer and operator of the measuring system it is also vital to increase the fault detection rate and decrease detection time since sensor malfunctions are at best detected late. Commonly, a sensor- or system-failure may not be detected during a flight experiment but weeks after. The ensuing rescheduling within the aircraft's full timetable makes it even more economically important to implement a system to catch sensor faults and failures to avoid the need for an experiment to be repeated. In a worst case scenario a whole experiment would need to be replicated with the special aircraft configuration of custom sensor equipment needing to be recustomized and refitted which is a time- and funding intensive task. A Sensor Health Monitoring software would enable detecting errors imminently after the sensor data is extracted from the aircraft. Due to the lower reaction time, errors may be detected and diagnosed on the same day, allowing for insights into the data that are currently inaccessible.

In previous works, a database has been created and the Flight Test Data has been provisioned. This database and the infrastructure used in this work is the german aerospace centre's (DLR) digital twin project and the emerging Skystash platform [MEYE20; ARTS22]. The Skystash will be used to centralize data operations and enable distribution digitizing the data and digitalizing the convoluted internal processes. Next to data handling the metadata treatment is of great importance within this work's context to cleanly interface the different components that will be developed and to avoid high coupling of singular components. Fortunately, the Internet of Things (IoT) as well as the Internet of Production (IoP)[PENN19] and Digital Twin contexts provide a vast and rapidly expanding field of such methods. Notable works for metadata are the Asset Administration Shell (AAS) [BADE20] and the SensOr Interfacing Language (SOIL) [BODE21; BODE22] in IoP contexts that currently developed at the Chair of Production Metrology and Quality Management (MQ) at the Laboratory for Machine Tools and Production Engineering (WZL) of RWTH Aachen University (WZL-MQ).

Algorithms to detect faults can now be embedded into this landscape of data and metadata. A majority of these algorithms are developed for system control in the field of control theory and Failure Mode and Effects Analysis (FMEA) and thus are real-time applications that aim to maintain system control [ISER06]. Since the goal of this work is to quantify data quality these algorithms will have to be modified and can't be used out of the box. Furthermore, methods from the field of statistics and data statistics are examined for employability within this work [HAND17].

Within the scope of this work a prototype for a Sensor Health Monitoring (SHM) application is developed. First,

the data gets examined to generate sample malfunctions upon which the FMEA algorithms can be trained. The FMEA algorithms are then developed based on these test cases. The developed algorithms then get embedded into a data pipeline process using FAIR contexts. Within this pipeline, data gets generated, enriched, checked, reported on and finally visualized. A FMEA method prototype will be developed for a smaller subset of the actual sensors in order to deliver a holistic and heuristic approach that is scalable and adaptable.

2 State of the Art

Chapter 2 introduces the theoretic baselines for data acquisition, data metrics as well as standardized data formats in which to save metadata. Topics to consider when starting the Sensor Health Monitoring process are mainly that of providing a structured overview of the Sensor Metadata which in itself consists of many layers as a dynamically generated set of metadata is desired. This is motivated by the desire to represent changing Data Acquisitioning (DAQ) System configuration changes in the metadata system. Consideration is given to the SOIL data model and its' ability to handle the many demands that are expected of sensor data management. [BODE21] The second major part to consider is that of physical cross relations and deep checks which are an experimental mode of checking for inconsistencies among the data. Major research and implementation work goes into developing a dynamic model that is generated from the data and then checks back upon the data for possible discrepancies. This approach is chosen as it is estimated to be the most structured approach for a first prototype.

2.1 The ISTAR

The data itself is the essence of this work. The data used in this work is recorded in the ISTAR aircraft, a Falcon 2000LX. Since 2019 it is part of the DLR's research aicraft fleet (see Figure 2.1). The ISTAR possesses a large databus with over 40,000 parameters that are essential to transmit system info of singular aircraft components to the Flight Management Computer (FMC) and other recipients within the aircraft. These parameters consist of Primary aircraft data such as sensor measurements but a majority of the parameters consists of secondary derived data and status bits that are present in various SI-units and imperial aerospace units as well as data formats ranging from Boolean values to integers, floats and strings. The special part of the ISTAR's customization lies within its recording of aircraft bus parameters which allows to retroactively analyze flights. Facilitated by a DAQ that records the main aircraft bus, an additional inertial positioning unit, experimental aeroelastic sensors and the noseboom over 400 sensor parameters are recorded per flight.

Figure 2.1: The DLR's research aircraft fleet [DLR18] lined up for a photo shoot at the DLR Brunswick site

The ISTAR will be examined since its measurement data for Digital Twin applications needs to be certified in respect to its data quality. The aircraft system itself is tested and sensor errors are displayed to the pilots during the flight. However, there is no way to record these error messages and modify the fault detection algorithms to the custom experimental sensors. In addition, the DAQ hardware is certified to be safe within the aircraft (e.g. electromagnetic compatibility, avoiding spontaneous inflammation) to refrain from detrimenting flight conditions but it is not configured to also provide data reliably with a standard reliability of 10^{-6}. This further motivates the desire for an open SHM to further data knowledge and insights.

2.1.1 System overview

The ISTAR's data acquisitioning system (DAQ) is the hub to record and centralize all aircraft data and will be regarded as the focal element of this work. As previously mentioned the DAQ is not tested up-to aircraft standards. Because fault detection is shown in the cockpit during the flight but is not recorded in the DAQ it is necessary to implement such a fault detection. The data flow within the aircraft is complex and nested. The DAQ firstly taps into the main aircraft bus that is within the Avionics Standard Communications Bus, Version D (ASCB-D) as well as the aeroelastic sensors (strain gauges and accelerometers), the Nose-Boom (instrumentation to measure undisturbed airflow) and a highly precise additionally installed combined Inertial-Navigation-System (IMS) and Geodetical Navigation Satellite System (GNSS). As any other aircraft, the necessity of various reference systems is necessary. Generally, the geodetic, aerodynamic, airframe-fixed and track-fixed coordinates system are used [BROC11]. These compensate various effects and facilitate interpretation of various primary and secondary parameter values

2.1.2 Sensor to DAQ

In the following, an exemplary flow of information from a Pitot Tube is described, illuminating the various facets and trade-offs that need to be made along the way to the main aircraft bus and then the DAQ. In Figure 2.2 the exemplary data flow for a single sensor to the DAQ is shown. The Pitot Tube (PT) measuring a dynamic pressure is chosen for this exemplary case.

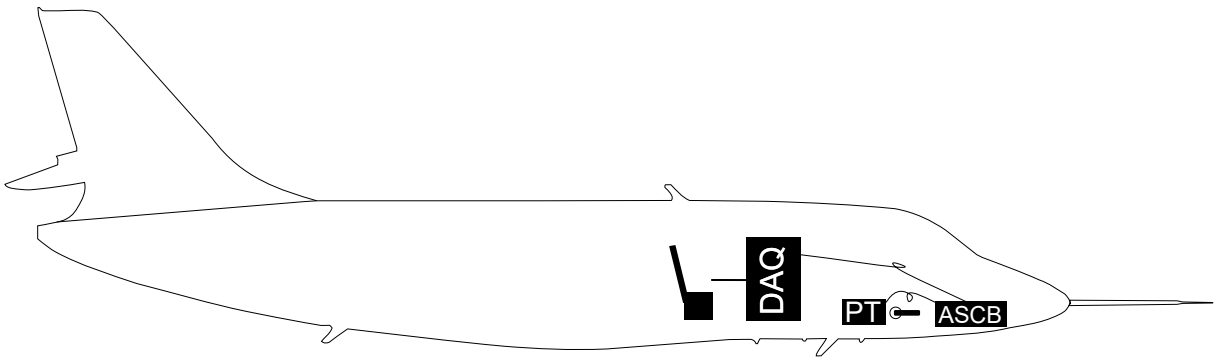


Figure 2.2: The ISTAR data flow with a Pitot Tube (PT) sending its data to the ASCB-D bus via the Air Data System (not shown) and then to the DAQ. Each entity of the flow applies its own sampling of data.

The Flight Test Data by the ISTAR gets recorded by a base DAQ system by IMC co. This base contains inputs from the main aircraft data bus, the experimental nose boom, experimental strain gauges and acceleration

sensors as well as an experimental Inertial Measuring Unit (IMU) combined with a GNSS-System. In addition, complementary DAQs are added to fulfill given experiment requirements. In regard to system complexity it is notable that the main aircraft bus data originates from various sources, ranging from air data that gets measured by a sensor working on an electric current which then gets transformed by an Analog Digital Converter (ADC) into a digital, discretized signal which then gets fed into the main aircraft bus system. Such a signal would then get transmitted to the IMC DAQ, making troubleshooting a multifaceted problem within this system due to the multiple points of failure.

2.1.3 State to Sensor

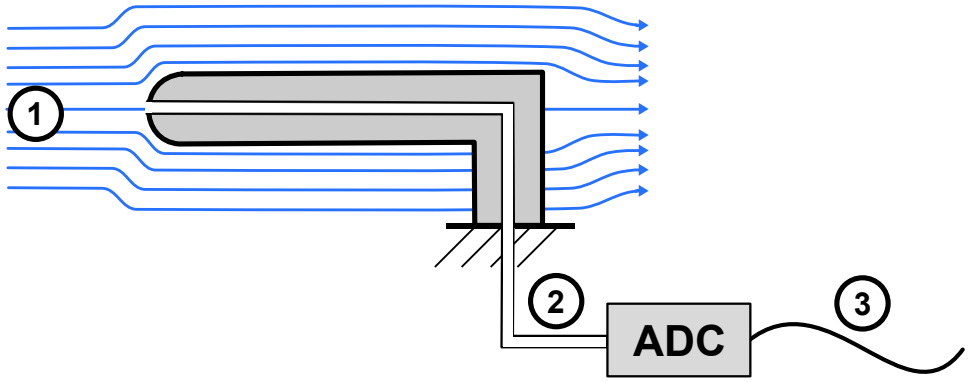
Within the following, some consideration is given to the inherent flaws lying in the actual measurement of state values. The Pitot Tube system from Figure 2.2 is examined in detail in Figure 2.3 showing the choices, experimenters need to make and the tradeoffs they inherit. An example is the sampling rate. Sampling at higher rates delivers much more precise results. It however also generates proportionally more data. It is then necessary to determine a sampling rate that is apt for each experiment. The Nyquist-Shannon sampling theorem is an aid in deciding the desired frequency since it establishes that the sampling frequency can only detect signals that are less than half its frequency as shown in Equation 2.1. [SMIT99]

$$f_{sampling} > 2 \cdot f_{max} \tag{2.1}$$

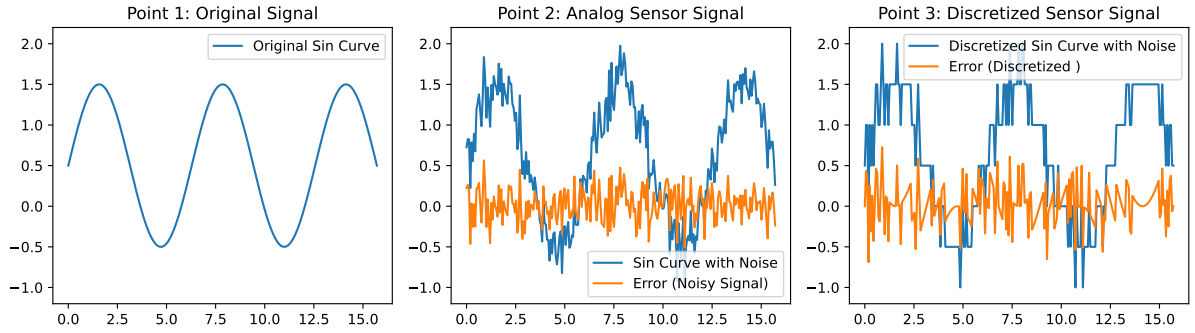
For context in this work, the ASCB-D has the tradeoff that it resamples all aircraft signals to 20Hz which in turn leads to some information loss with some signals being downsampled and some being oversampled. For most applications the Sampling of 20Hz is satisfactory although a resampling of any data inherently leads to information loss and is thus not desirable in most cases. This makes potentially noisy sensors output the same value twice in a row since the DAQ hasn't yet received the new sensor data and outputs the same value multiple times in a row. Even sensors with a high noise ratio such as the fine part of a GNSS latitude signal may sometimes output the same value twice in a row which is highly unlikely to happen on a statistical basis. Since this occurs quite often the question arises if the actual sampling may be lower or higher than the actual data. Perhaps it is then justified to resample some data to reduce oversampling in order to increase the data quality mitigating the changes made by the systems in and around sensor and DAQ.

For Illustration, an exemplary time-, and value-sampling of a Pitot Tube is shown.

The undisturbed system state at point 1 gets measured by a sensor and is transformed into the analog, measured signal at point 2 with the error being represented by the orange plot. An ADC is used in the next step of the signals journey and discretizes the time by sampling the signal to a given frequency and a discrete value given by a given resolution. Having been digitized at point 3 the signal error subsequently grows even further. Other error mechanisms that occur randomly (stochastic) include but are not limited to sensor drift as well as random noise. In the following, qualifiers and descriptors are introduced to quantify these qualities.



(a) A pitot tube setup for measuring dynamic pressure in order to measure aircraft velocity. Three spots are marked in which the state signal is skewed in a significant way.



(b) Original signal transformed into measured and discretized signal. The signal transformation at each step is shown together with the error occurring at each step

Figure 2.3: An illustration of random signal errors for a pitot tube. The Figure contains the undisturbed state variable (1), the unmodified measurement by the analog sensor (2) and the digitized, discretized sensor (3) values. In Figure 2.3b the modified signal is shown in blue with the error in orange.

2.2 Definitions

Motivating this section is the desire to understand the fundamental error mechanism. Definition for errors are adopted from Isermann [ISER06] defined in table 2.2, the de-facto standard of Fault Diagnosis descriptions within the automation domain.

2.2.1 Errors

To find and detect errors it is firstly important to understand them. Within Table 2.2 errors are described as the difference between measurement and actual value as seen in Figure 2.4. Sensor are used to measure physical states in the world. But how precise do they measure? No sensor is without error, an inevitable property accompanying every sensor application. In the following, sensor errors are classified into categories upon which an understanding about data quality can be built that then can describe the data quality of the ISTAR datasets.

Table 2.1: A general overview over the three different Error Types. Systemic errors are errors which are predictable and can be modeled mathematically. To compensate systemic errors a calibration process can be applied which removes the entire systemic error. The second error type is the random error which occurs constantly and behaves unpredictably. Random Errors are generally also understood as white noise. Lastly, the failure forms the final type of error type. It signifies through relatively rare occurrence but a then fatal failure of the data source. A solution for random errors is redundancy of sensors and the mixing of single sensor values. [HART22]

Error Type	Appearance	Occurrence	Compensation
Systemic Error	Deterministic	Permanent	Calibration
Random Error	Stochastic	Permanent	Filtering
Failure	Stochastic	Occasional	Mixing

Identifying sensor errors is the goal of this thesis. As described in Figure 2.4 the error can be imagined as a disturbance added upon the original sensor value. Which then results in a skewed sensor measurement.

In Table 2.1 sensor errors are divided into 3 subcategories which are classified by their appearance representing their manageability. Deterministic Errors can be easily compensated since they are reproduceable. Stochastic Errors however are non-deterministic. Figure 2.5 gives an overview over systemic errors which generally are compensated by using calibration mechanisms in practical applications. The sensors within this work are assumed to be calibrated so this property is solely mentioned for completeness. However, perhaps some new understandings may be gained after a full SHM routine has been implemented since accuracy of calibration specifications is only as accurate as previously specified by the operator.

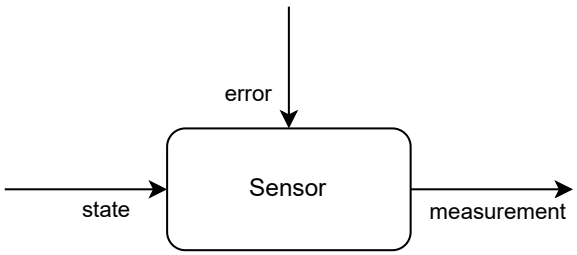


Figure 2.4: A sensor measurement is always the result of the actual true value plus a measuring error.

Errors that also occur continuously are random errors that are however unpredictable in their behavior. To compensate those errors, filtering can be applied by e.g., high- or lowpassfiltering. The final error type mentioned in Table 2.1 is sensor failure. This is characterised as a spontaneous event in which the sensor information stream is completely interrupted.

Now the question poses if errors can be pinpointed to their origin from where such errors arise. Generally, single sensors carry inherent errors such as clogged static ports (spontaneous hysteresis) or sensor drift (spontaneous bias) but errors also occur within the complex aircraft system architecture itself. An exemplary process step is the Analog to Digital Conversion. Sensor outputs are primarily analog but are then digitized in order to get

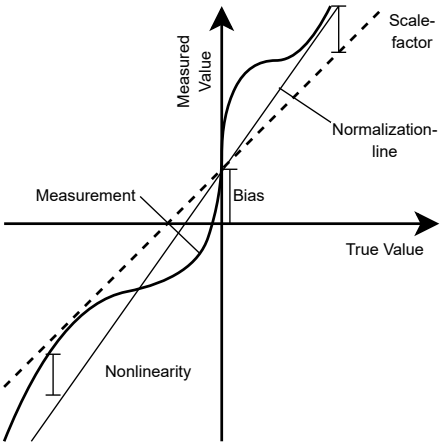


Figure 2.5: The types of systemic sensor errors introduced in Table 2.1 are shown by plotting the measured value over the true value. [HART22; DIN95]

transported by the aircraft data bus. This process is accomplished using an Analog Digital Converter (ADC), effectively eliminating information from the system. The full process is shown in Figure 2.3.

2.2.2 Data Quality

Foundation for this work lies within a common understanding of its results. Standardized qualifiers are found within control theory [ISER11] as well as GNSS applications [TEUN17]. Standardization is also found within the ISO-Database. However, when beginning to describe data quality within metadata, the issue of metadata quality itself also arises. In the following, a brief overview is given over descriptors and their qualities based on norms.

A general introduction to Semantics

Semantics is a broad term due to its common use. One definition that has found some acceptance is the definition by Metzlers Lexikon which relies on theories by Blackburn and Kutschera [SHOE87; KUTS75]. The following paragraph contains a small explanatory parenthesis to motivate the term Semantics and its parent term being the field of semiotics. Semiology (greek:σημεῖον,semeion:sign) describes the theory of signs and their usage as described in Figure 2.6. It is divided into the areas of semantics, syntactics and pragmatics in which syntactics are defined as the internal structure of signs within sign systems. In the next step pragmatics are defined as the theory of sign usage effectively thinking about how interaction with signs works. Finally, semantics define the relationship between signs and described objects. They work by allocating a structure or model to a predefined expression in the way a metadata model uses standardized descriptors to represent a meaning in the real world.

This gets developed further within ISO 8000-8: Data Quality [ISO15], concluding that syntactic qualities define the structure of the content language and semantic qualities define the structure of the content itself to allow verifying information. Whereas pragmatic qualities describe the use that can be extracted from the content

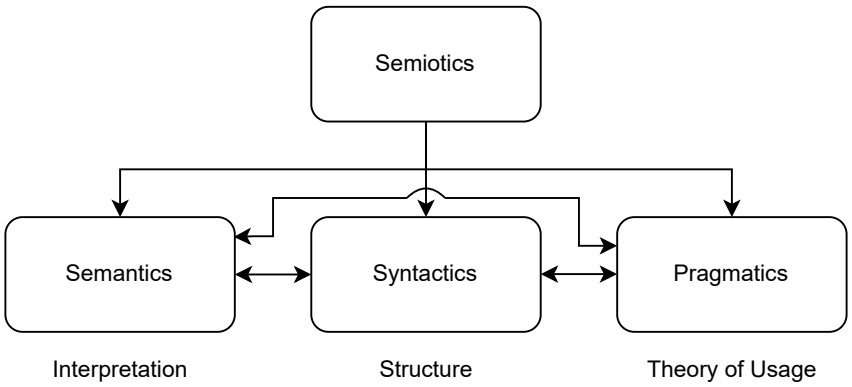


Figure 2.6: Semiology, according to Kutschera [KUTS75] and Shoemaker and Blackburn [SHOE87] contains three subfields of semantics, syntactics and pragmatics which form the cornerstones of common sign usage.

itself hence allowing validation.

Contextually, the metadata syntax of this work will be in a JSON-format. The used semantics will be standardized to use the SOIL (SensOr Interfacing Language) structure and pragmatics will be quantified by the validity of the actual metadata. Moving on from these fundamental concepts, the pragmatic data qualities will be defined in the following.

Metadata descriptors

Unambiguous terms are needed to provide clear and concise information within the sensor health monitoring data structure. This is especially important to create a common base of understanding. In the following, industry standards and definitions for data metrics are summarized.

Definitions based on a control theory background are shown in table 2.2. These have been agreed upon in discussion with VDI and VDE committees as well as the Reliability, Availability and Maintainability (RAM) dictionary. [ISER97; DIN77]

These terms are divided into the contextual categories of primarily defining signal and state descriptors. Next up are functions for fault processing followed by model attributes that are employed to compare them to the actual system. System properties can then be inferred by leveraging the previous descriptors for states, functions and models. Illustrating the fault detection process, Figure 2.7 shows the method generating a residual which in turn enables interpretation of said residual which may contain information to isolate and identify the error.

Further definitions are also employed by the Federal Aviation Administration (FAA) adding integrity as an attribute for GNSS systems that is a quantifier for trust that can be put into the system's measurements. Providing users with warnings when measurements are expected to be unreliable. [ADMI08, B.1.5, B.1.10]

Following the GNSS domain, the terms accuracy and precision are important measures for quantifying results. For representation, a dart board is illustrated in Figure 2.8. It is then important to distinguish that good accuracy is the result of good precision as well as a good trueness. A mathematical description for the precision could be the standard deviation and the deviation of mean and actual value as the trueness. [ISO97][SMIT99,

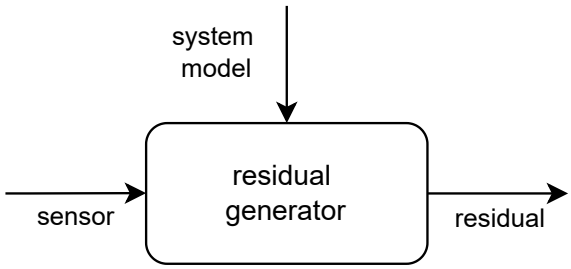


Figure 2.7: Comparing the sensor signal to a model-based value yields a residual which can be used to quantify the sensor error.

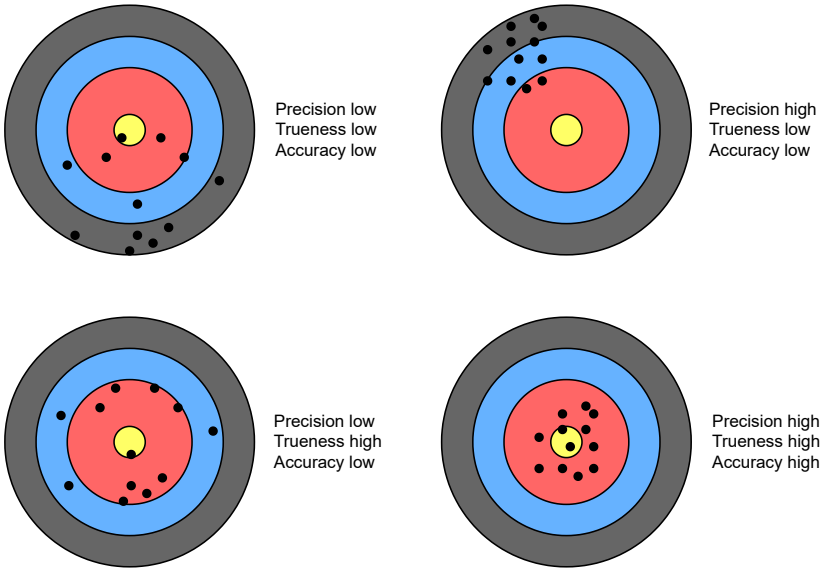


Figure 2.8: Dart boards displaying the Precision (as σ), Trueness (as μ) and Accuracy as the cumulative sensor quality [ISO97]

S.33ff.]

Table 2.2: Terminology for system properties [ISER11]

Descriptor	Description
States and Signals	
fault	unpermitted deviation of one subset of the system
failure	permanent interruption
malfunction	intermittent regularity
disturbance	unknown, uncontrolled input
perturbation	input, leading to temporary departure from steady state
error	deviation between measurement and true value
	$y_e = \bar{y} - y$
residual	deviation between measurements and model-based calculations
	$\hat{y} = \bar{y} - y_m$
Functions	
fault detection	determination fault presence
fault isolation	Determination fault properties: kind, location, time of detection
fault identification	determination of size and time-variant behavior of fault
fault diagnosis	includes fault detection, isolation, identification
monitoring	real-time determination of possible physical conditions and
	recognition, indication of behavioral anomalies
supervision	monitoring and taking actions to maintain operation during faults
protection	means by which potentially dangerous behaviors are suppressed if
	possible or how consequences are avoided
Models	
quantitative	describe system in quantitative mathematical terms
qualitative	describe system in causalities and if-then rules
diagnostic	link specific inputs (symptoms) to outputs (faults)
analytical redundancy	determine a quantity in an additional way by using a mathematical process model
System Properties	
reliability	ability to perform a function, measure $MTTF$, with λ as rate of failure per hour
safety	ability of a system not to cause danger to persons, equipment and environment
availability	$A = \frac{MTTF}{MTTF+MTTR}$
λ	rate of failure
μ	rate of repair
MTTF=$1/\lambda$	Mean time to failure
MTTR = $1/\mu$	mean time to repair

2.3 SHM Algorithms

Building now upon a solid foundation of descriptors, some previously implemented systems dealing with monitoring and supervision tasks are reviewed. The superset of these monitoring systems is generally described as Failure Mode and Effect Analysis (FMEA). In the following, some FMEA implementations are reviewed. Theoretic fundamentals for this works' actual implementation are then shown, checking sensor values against limits and then correlating sensor values with each other.

For the SHM process three distinct steps are proposed to filter out malfunctions by severity and to order the process flow. This entire flow is shown in Figure 2.9 displaying the proposed essential steps of the SHM. The core of the work is the data processing represented by the gear icon. A core problem of this work is finding a standardized way to transfer information between steps. In order to solve this a metadata model will have to be found that allows for standardized data transfer in order to allow reusability of this work.

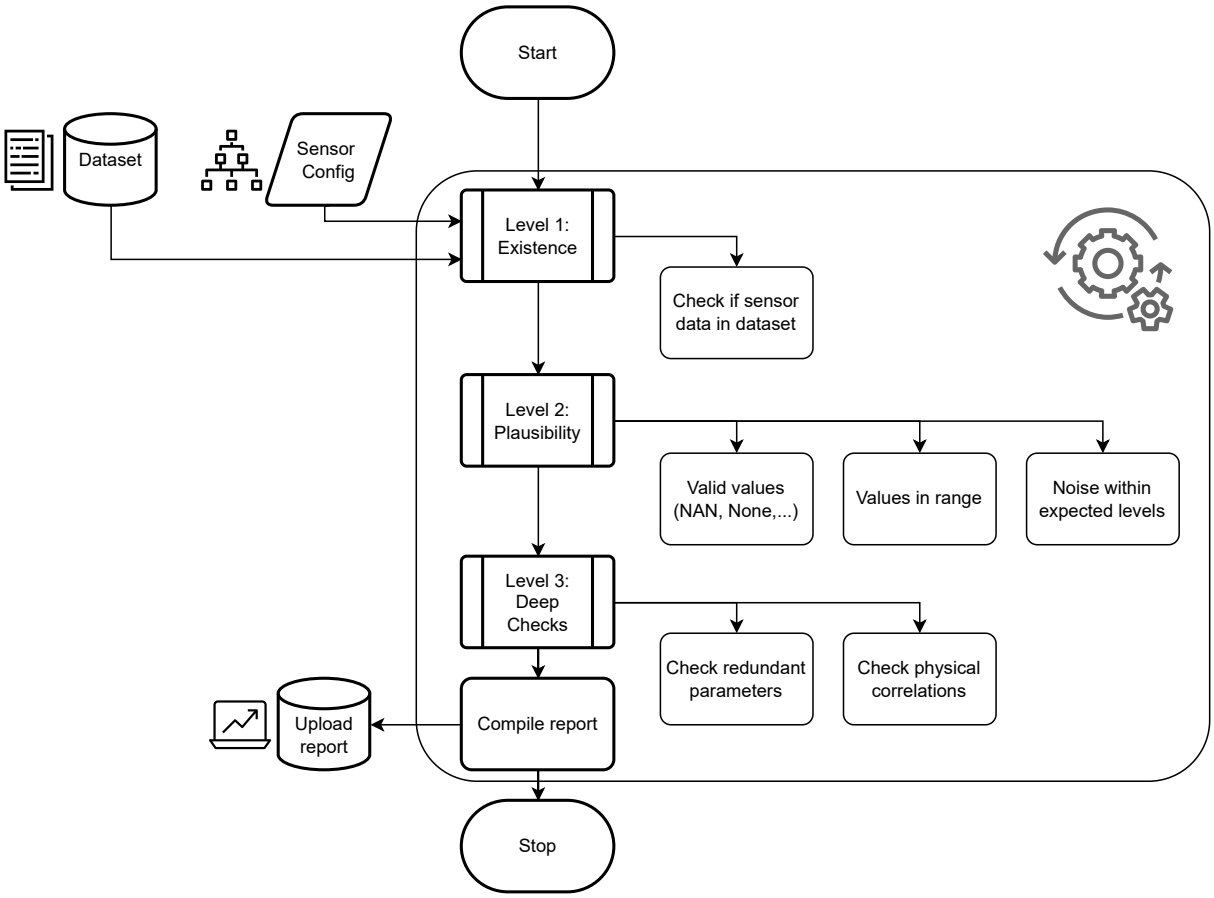


Figure 2.9: The data flow architecture proposed within this work. Starting with the configuration and dataset as fundamental process inputs feeding the data-processing, algorithmic step. Within the algorithmic step three distinct functions are proposed checking for existence of sensors in the dataset followed by the Plausibility step which checks sensor singularly without other sensors as context and finally the step of Deep Checks in which a correlation check is performed with a custom algorithm. The results from the three steps are then compiled into a data quality report which then can be visualized.

2.3.1 Introduction to FMEA

Motivated by the desire to warn operators of system failure the FMEA has a relatively simple goal. To detect failures and faults and alert the user to avoid further damage, costs and downtime. This naturally establishes it as one of the pillars of automation. [ISER06] Since automated systems are only as smart as their creator, methods have to be put into place assuring their high level of quality, guaranteeing smooth operation which in turn spawns and motivates the entire field of FMEA. In recent years, especially systems based on machine learning (ML) approaches have come into play due to widely available cheap computing power as well as increasingly optimized training algorithms. Due to the scope of this work this ML-approach however remains to be addressed at a later point.

In the following, state of the art non ML approaches are introduced preceded by a brief theoretic section on statistics for completeness. The FMEA methods then outlined feature the principal component analysis (PCA) used in live monitoring [XIAO06], Pseudo Transfer Functions (PTFs) in control theory [ALJA15], parity equations as well as grey box models using parameter tuning. [ISER06] In addition, some methods from the GNSS field are presented since much of GNSS research is public in contrast to most control algorithms for aircraft systems.

Statistical methods

In the following, fundamental statistical methods are shown to prepare the fundamentals for the proceeding SHM steps in Level 1, 2 and 3. Since the focus of this work lies in signal analysis, relevant methods from Smith [SMIT99] are proposed.

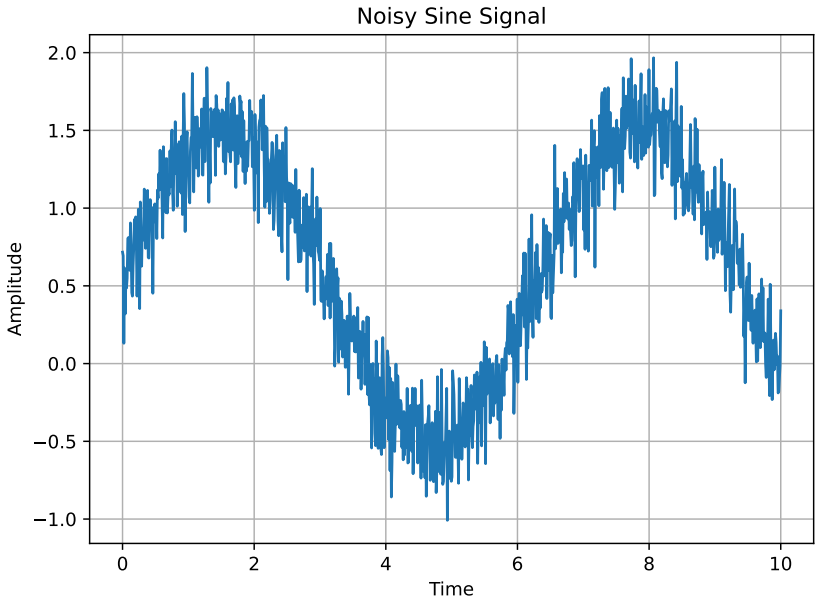


Figure 2.10: A sample sine signal with some white noise to illustrate the following statistical methods.

Signal properties are examined for a given signal in Figure 2.10

[SMIT99, S.13-17]

Time continuous mean

$$\mu = \frac{1}{T} \cdot \int_T x(t)dt \tag{2.2}$$

Time discrete mean:

$$\mu = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i \tag{2.3}$$

The variance σ^2 is a metric for the signal's behavior. It expresses the mean squared deviation from the mean.

Time

$$\sigma^2 = \frac{1}{T} \cdot \int_T [x(t) - \mu]^2 dt \tag{2.4}$$

$$\sigma^2 = \frac{1}{N} \cdot \sum_{i=0}^{N} [x_i - \mu]^2 \tag{2.5}$$

The standard deviation is derived from the variance. Its value gets square-rooted to better represent the power (magnitude of the amplitude).

Standard Deviation

$$\sigma = \sqrt{\sigma^2} \tag{2.6}$$

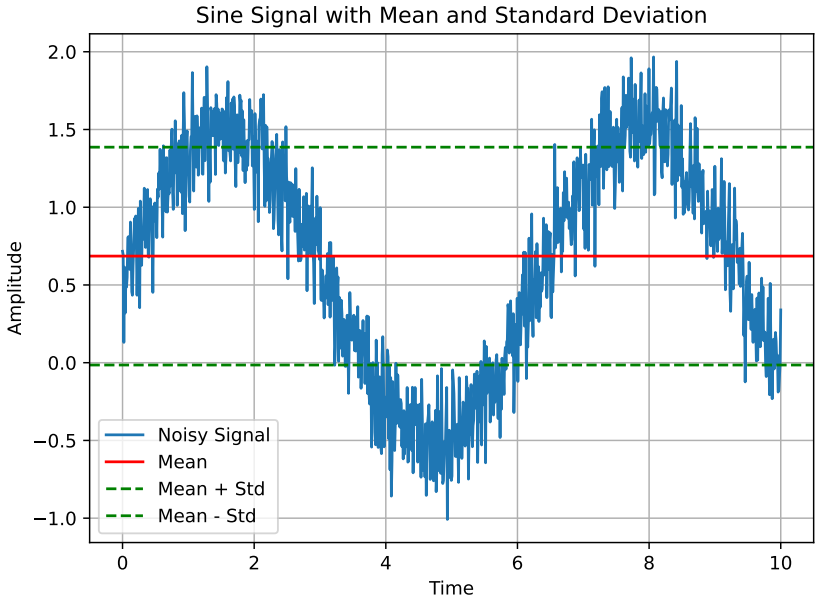


Figure 2.11: Signal with μ and σ borders for upper and lower limit

Mean and the standard deviation don't represent the desired metrics in some use cases. Rather more important is a comparison between the two. Hence, the Signal-to-Noise ratio (SNR) is used to compare and condense the mean and standard deviation by dividing the mean by the standard deviation.

$$SNR = \frac{\mu}{\sigma} \tag{2.7}$$

Another parameter is the coefficient of variation (CV) which is the standard deviation divided by the mean and multiplied by 100%. Effectively displaying the inverse of the SNR.

$$CV = \frac{\sigma}{\mu} 100\% \tag{2.8}$$

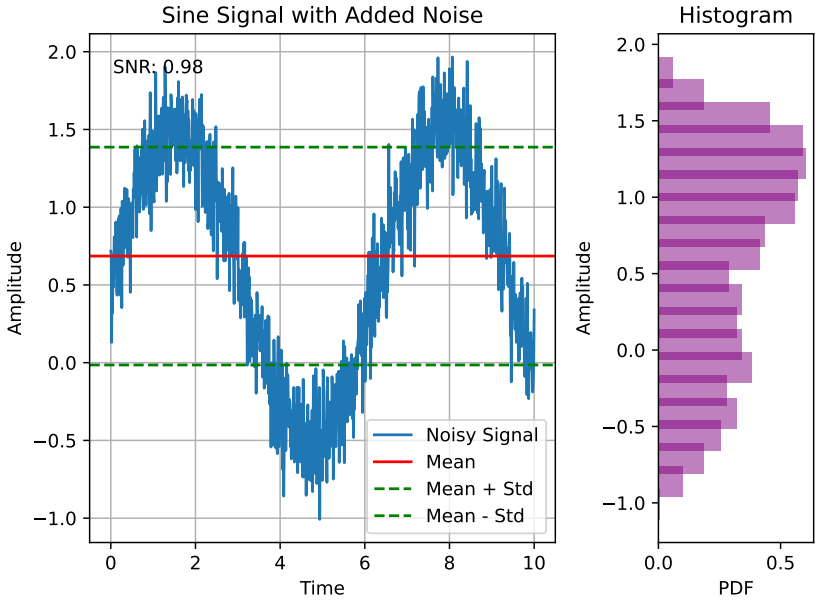


Figure 2.12: The Sine Signal with μ, σ, SNR and a Histogram giving more in-depth information on the signal 's behavior.

An arising problem based on the SNR and CV are however that they scale based on the mean value. Should the mean value lie at about 0 for e.g. a sensor of an aircraft control surface, the signal to noise ratio will be relatively high compared to an acceleration sensor in z axis with a constant offset of 1g. Practical example for mean and standard deviation in a given signal are overlaid in Figure 2.11 To evaluate a signal according to the quantities the next logical step is the probability mass function (PMF) 2.9. This part of signal evaluation is shown in the following Figure 2.12 by adding a histogram to the analysis which counts the signal points lying within discretized value ranges.

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.9}$$

Building upon these factors the covariance is introduced as a measure for comparing signals to each other. First the signal gets translated to have a mean value of 0 and then all values of the first series at a certain point t are multiplied with the points of the second series as displayed in the time continuous covariance function.

$$cov(t,x) = \frac{1}{T} \cdot \int_T [x(t) - \mu] \cdot [x(t-\tau) - \mu] dt$$

For the time-discrete covariance function follows henceforth:

$$cov(k,x) = \frac{1}{N} \cdot \sum_N [x_i - \mu] \cdot [x_{i-k} - \mu]$$

Based upon this the autocovariance for a series which compares a series' signal to itself can be represented as:

$$autocov(x) = cov(x, x)$$

Going further, the Pearson coefficient within a correlation matrix is reached by scaling the covariance matrix by the standard deviation: [SMIT99]

$$\rho_{x,y} = corr(x, y) = \frac{cov(x, y)}{\sigma_x \sigma_y}$$

With the limits of correlation being -1 for total inversely proportional correlation and 1 for total correlation. A total independence for $\rho_{x,y} = 0$ is not given since Pearson coefficient only detects linear correlations and not nonlinear behavior. Other correlation methods as rank-correlation (Spearman, Kendall) that detect change correlations are possible but are more complex in the implementation.

Summary and Motivation

Based upon this groundwork, methods for the three levels of the SHM are proposed. State of the Art solutions are discussed for each Level based on the common statistical methods proposed in the previous section.

2.3.2 Check 1: Existence

In the first Level of the SHM it is checked whether parameter data exists in the database. This is checked by comparing the DAQ-configuration to the actually measured data. Within step two, single sensor signals are examined by themselves, saving resources by iteratively working on each sensor. This approach also allows easy parallelization of tasks due to the encapsulation of each task. Step three allows the integration and correlation of different sensor signals. Allowing more complex FMEA systems.

Within the first step, the main challenge lies within the acquisition of sound configuration data. Errors may occur as early as the configuration step due to human error. Making it the first level of defense against simple mistakes.

2.3.3 Check 2: Plausibility

Within the plausibility check the single parameter is checked against limits that shall be preconfigured for each parameter. Various methods are examined. The primary focus is checking the value against limits to detect extreme values that are unrealistic. Within the next step, the series is transformed with the noise or other desired attributes being extracted. These attributes can then in turn be checked against other limits. For the noise, various known methods are examined to create a heuristic for noise approximation. Of course, noise approximation models range in complexity and detail from a simple moving window approximation into Fourier transforms (Fast-Fourier-Transform (FFT), Short-Time-Fourier-Transform(STFT), Wavelet Transform (WT) and into advanced works such as language-processing models [HEND08] which exceed the scope of this work and may be implemented at a later date.

These functions are evaluated into a ranking and displayed in Table 2.3. After brief evaluation, the Short-Time-Fourier Transform is considered due to its comparability among sensor signals. Allowing to compare movement activity by amplitude. The STFT algorithm is then implemented preliminary by using a short time window of

Table 2.3: Comparison of Level 2 Noise approximation algorithms (empty circle - low rating)

Noise algorithm:	Moving Average	FFT	STFT	WT
Implementation Effort	◔	◕	◓	●
Result Quality	◔	◓	◕	●
Real-Time Performance	●	◓	◕	◔

256 samples and then collapsing the resulting spectrogram by time. This results in an averaged amplitude value for each timestep that is detrended. Leading to a scalar value, upon which the noise of each value can be mapped.

2.3.4 Check 3: Parameter Correlation

Physically correlating and involving parameters into the anomaly detection of other parameters is the next logical step. It is primarily considered to use conventional non ML-approach using known correlations of parameters and perhaps allowing some grey box approaches for fine-tune limits and finding unknown correlation. A primary desire is to generate a model that is not hardcoded but easily expandable to satisfy various data-generators. A few non-ML FMEA approaches are presented in the following paragraphs to build a primary overview over the wide field of FMEA.

Principal Component Analysis

The Principal Component Analysis (PCA) was first introduced by Karl Pearson in 1901 ([PEAR01]) and has since become a popular method to analyze datasets and detect correlations. The PCA works by reducing the dataset dimensions using covariance of parameters relative to each other. This results in a method that needs minimal user input but delivers a condensed data overview. The only input needed is the number of dimensions into which data shall be condensed into, the so called Principal Components (PCs).

The functionality, applications as well as usability within the context of this work are discussed below.

Functionality

The PCA is a method that allows to reduce the information parameters of a dataset into a few principal components or dimensions. Generally, this is specified as the operation of multiplying the original dataset X_{Nxm} with the timesteps N and the measurement parameters m with the Permutation Matrix P_{mxr}. This results in the transformed principal components (PC) T_{Nxr} with r being number of reduced dimensions or also the number of Principal Components. This transformation is defined as:

$$T_{Nxr} = X_{Nxm} P_{mxr}$$

P being the permutation matrix that transform the original data into the principal components. The overall strategy to generate a PC is shown in Figure 2.13 as finding a function between two parameters x_1 and x_2 that maximizes the variance of data in a newly generated principal component. However, drawbacks arise using this approach since the variance optimization only happens linearly and is not able to function for nonlinear correlations similar to a standard Pearson-Coefficient Correlation matrix. [HAND17]

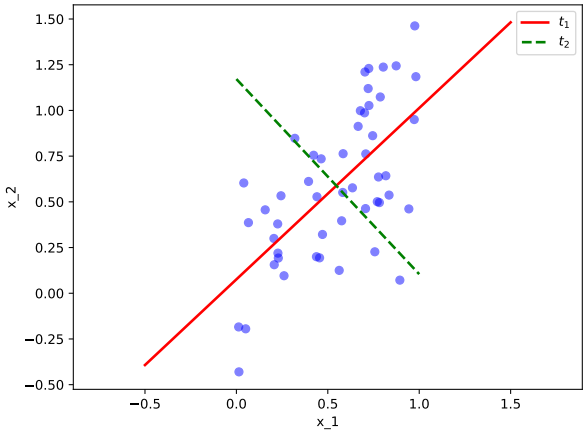


Figure 2.13: Parameters x_1 and x_2 get converted into principal components t_1 and t_2 by optimizing for minimal variance of blue dots along the red line.

PCs can be created in any direction. Where input is needed however is the number of Principal components that shall be kept. A cutoff value can be defaulted to but this may not be feasible for multidimensional systems such as the flight test data that is at hand.

Eliminating a principal component becomes easy if parameters show a direct correlation such as in Figure 2.13. The gatherings of t_2 then mostly consist of noise and can be safely discarded.

Utilization

This analysis method is especially helpful for datasets for which correlations are unknown such as biomedicine in which marker correlation needs to be identified. But even for aerospace contexts it may prove helpful for large quantities of data in which correlations are unknown e.g. for parameter identification as well as perhaps aeroelastic applications in which vast amounts of acceleration sensors and strain gauges are present and used to explain dynamic behavior of the aicraft's structure.

In these applications the PCA is a great tool to get a first overview over the data. For more refined analysis however, it lacks detail since it generally uses linear optimization as and may be prone to overfitting for redundant input parameters. In these cases, it may be sensible to use optimized PCA approaches or an entirely different analysis system such as ML-approaches.

The PCA can be defined as a condensation of a dataset into fewer parameters. These are then defined as principal components. These principal components can also be transformed back into the original parameters making the entire process reversible and allowing to create a data model. Based upon this model it is possible to then calculate residuals in order to possibly quantify the error. E.g. Isermann

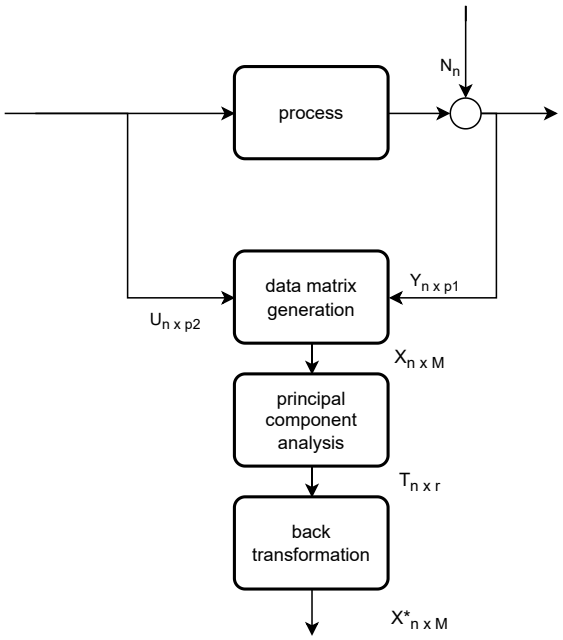


Figure 2.14: Process input U and Sensor Measurement Y get transformed into state X and fed into a PCA. N indexes timesteps and X^* is the predicted output.[ISER06, p.268]

[ISER06] presents a process for an autotuning PCA FMEA that can be used for real-time Fault Detection applications.

Assessment

The principal component analysis is based on correlation principles and allows reduction of datasets into independent components. Applications in an aerospace context include detection of previously unknown correlations between parameters and complex interactions in aeroelastics or otherwise large datasets. Usage however is associated with a certain effort since information needs to be preprocessed to a certain degree, reducing dimensions previous to processing in order to avoid feeding in redundant parameters.

Pseudo Transfer Functions

Novel developments in control theory port methods from previous applications in Structural Health Monitoring (StHM) whose original purpose is to analyze building and structure oscillative behavior such as frequency and amplitude to detect critical values. These StHM methods use multiple sensor data to detect anomalies in single sensors. In novel research these methods which were in the frequency domain have been transferred into the time domain, making them usable for general time series analysis. This process uses the Transmissibility function $\mathcal{T}$ using dynamically generated Markov-parameters for which a Pseudo Transfer Function (PTF) then needs to be found. Essentially spanning up a matrix of markov parameters similar to a correlation matrix. These Markov parameters then get tuned within the beginning of the measuring period after which a conistent model allowing to generate a residual to detect errors. [KHAL22a; KHAL22b; WOLT14]

Functionality

Following the recipe for a modeled aircraft based on sensor data y the parameters x and u of the aircraft

are attempted to be simulated. Aljanaideh and Bernstein [ALJA15] shows that this approach works for linking the angular velocities for the aircraft axes ω_x, ω_y, δ_{drift}(drift angle) with ω_z. The examined approaches include the transmissibility function $\mathcal{T}$ that models the system output without having to take the unknown system input into consideration.

Utilization

Since this is a relatively new development with no broad implementations yet existing. It is to note however, that the origin of this method in StHM might possess more implementations. Within multiple publications by the author this process is applied for various purposes such as the Aircraft Sensor Health Monitoring [ALJA15] and a newer addition including SHM for a Platoon of autonomous robots [KHAL22a].

Assessment

This approach seems especially suitable for real-time implementations and may also be used in postprocessing. It comes however with the penalty of not being usable straight out of the box since no python implementations exist so far. For future work, this may be implemented to evaluate its capability against other algorithms. It also remains to be seen how well this algorithm performs on large datasets and how much preprocessing effort needs to be undertaken.

Parity Equations

Parity Equations work by modelling the physical state of a system in normal working conditions. They are based on known physical relationships between sensor measurements and generate a residual. This residual is then continuously checked during operations and once it reaches or exceeds a previously defined threshold it notifies the system's operator. [ISER11]

Functionality

The Parity workflow separates into the two steps of calculating the physical state and generating a residual and then interpreting the residual.

Utilization

Parity Equations are employed in systems in which the physical correlations are well-known. They allow fusion of various parameters into system state variables to then generate residuals for single parameters to determine the parameter errors.

Assessment

Since the relations of the aircraft systems and behavior have historically been subject of extensive research the Parity equations lend themselves to be considered within this context. Extensive tuning is however integral part of Parity Equations since physical correlations as well as residual thresholds need to be tuned to avoid amassing false positives.

Grey Box models

Another method for system modelling lies in grey box models. The "grey" attribute arises from the property that these systems have baseline constraints that may be set and some degrees of freedom such as tuning parameters. These tuning parameters are then tuned against a set of previously defined training cases to satisfy the constraints. This represents an approach that works with limited system knowledge and implementations such as the PTF are implementations of this model. [ISER06]

FMEA Evaluation

An algorithm for evaluating data quality in Level 3 must now be found based on these state of the art approaches. Zhang and Jiang [ZHAN08] presents a general overview over the field of FMEA which allows to be split into the two main categories of model and data-based methods which are dependent on the system knowledge that shall be introduced into the FMEA. Both then carry two fields within of quantitative and qualitative methods which get applied for conditional modeling and less stringent data types. Within this endeavor it however

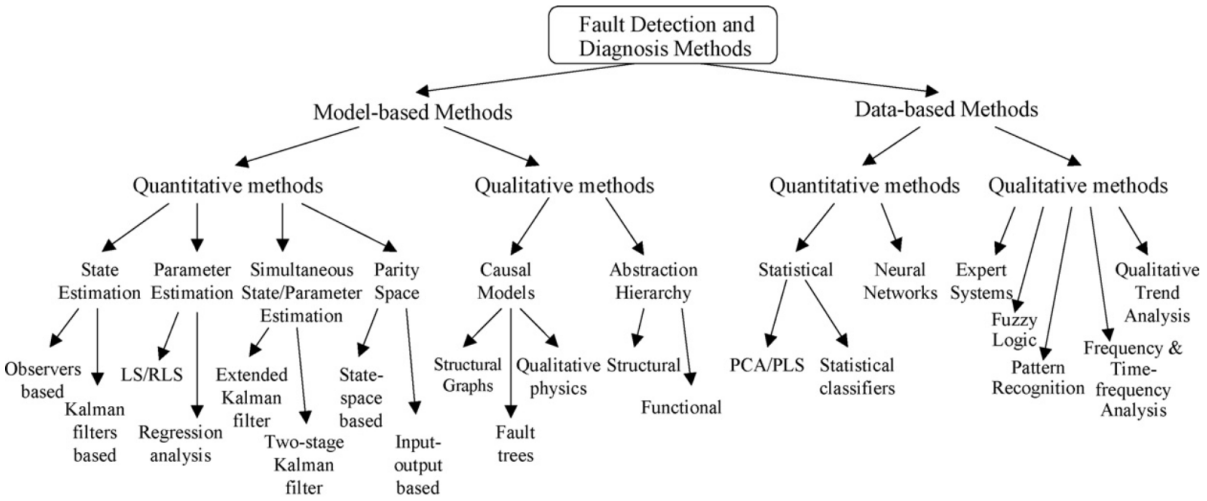


Figure 2.15: An overview over various kinds of FMEA satisfying a broad range of specific requirements that may be posed as system specifications. [ZHAN08] While clear distinctions must be made between quantitative and qualitative methods the main distinction lies in the amount of system knowledge provided to the method.

needs to be considered that this summary of FMEA methods barely scratches the surface. Far more FMEA variations exist than shown in Figure 2.15 and the effort within this work is limited. The goal is then to develop a primary heuristic that can deliver results whose primary purpose is to actually exist and not still be stuck in development. Additional important factors also are the FMEA-method's actual performance. Possible issues arise while comparing reliability. Especially in aerospace with multiple independent axes a false correlation may be detected due to circumstances. E.g. for a flight performing multiple consecutive landings it is possible for the Distance Measuring Equipment (DME) measuring distance to the airport to correlate with altitude and speed since all decrease and increase simultaneously. Simultaneously, employing no/low knowledge models enables rapid results eliminating the need for any degree of system knowledge.

Table 2.4: Comparing FMEA solutions by their respective usability

Noise algorithm:	Implementation Effort	System Knowledge	Reliability
Principal Component Analysis	◐	○	◓
Pseudo Transfer Functions	●	◓	◐
Grey Box Models	◐	◓	◓
Parity Equations	◔	●	●

In Table 2.4 the introduced FMEA-schemas are evaluated on their viability to use within this work. While PTF as well as other Grey-Box models exist, they need considerable effort within their primary implementation and then their respective training. Since the research and development of a novel, optimal FMEA is also not focal point of this work since it is rather more important to generate a holistic heuristic, solving first and foremost the basic problems associated with basic errors. Hence, Parity equations are chosen to be developed further within this work. Generating a dynamic model of the aircraft physical correlations that allows itself to be expanded for future use. It remains to be seen now, how well this work's architecture for integrating these FMEA may hold up in regard of testing these methods in future work.

2.3.5 Context within Aerospace Applications

The previously mentioned subset that will be examined within this work is the altitude of the aircraft. In the ISTAR the altitude can be determined by using the barometric altitude and the altitude determined by the Global Navigation Satellite System (GNSS). To implement a Parity Equation the system knowledge needed to create these models is introduced. In the following the exemplary implementation of a barometric system is presented followed by considerations needing to be made when examining the GNSS system and comparing it to the barometric system.

Barometric Altitude

For the Parity model, the barometric altitude is introduced to compare experimental pressure sensors with the aircraft's barometric altitude.

The International Standard Atmosphere (ISA) is the conventional method of measuring an aircraft's altitude based on air pressure and the industry standard. [ISO75] The method can be understood as a function of pressure and reference pressure (differing based on weather) which results in an altitude. The assumptions for each layer of the atmosphere are shown in table 2.5.

Based on these constants the general equation for pressure within an atmospheric layer with a nonzero temperature gradient is shown in equation 2.10. [ISO75]

$$\frac{p}{p_0} = \left(1 + \frac{a \cdot (h - h_0)}{T\,(h_0)}\right)^{\frac{-g}{a \cdot R}} \tag{2.10}$$

Table 2.5: Atmospheric Layers as defined in the International Standard Atmosphere [ISO75]

Atmospheric Layer	Geopotential Altitude [km]	Temperature T at bottom [K]	Temperature gradient a [K/km]
Troposphere	-2-0	301.15	-6.5
Troposphere	0-11	288.15	-6.5
Tropopause	11-20	216.65	0
Stratosphere	20-32	216.65	1
Stratosphere	32-47	228.65	2.8
Stratopause	47-51	270.65	0
Mesosphere	51-71	270.65	-2.8
Mesosphere	71-80	214.65	-2

p = pressure at current altitude [Pa]
p_0 = reference pressure [Pa]
h = current altitude [m]
h_0 = reference altitude [m]
$T(h_0)$ = temperature at reference altitude [K]
a = ISA Temperature gradient (see table 2.5) [K/km]
g = gravity constant $(9.80665)[m/s^2]$
R = Ideal Gas Constant $(287.05287)[m^2/(K \cdot s^2)]$

Transforming this formula for altitude resolves into equation 2.11

$$\rightarrow h = \left(\frac{p}{p_0}^{\frac{-a \cdot R}{g}} - 1 \right) \frac{T(h_0)}{a} + h_0 \tag{2.11}$$

Constant layer temperature

Since equation 2.10 does not represent the conditions for a constant temperature with $a = 0$ the original equation needs to adapted and it follows from ISO [ISO75]:

$$\frac{p}{p_0} = \exp\left(\frac{-g}{R \cdot T(h_0)} \cdot (h - h_o) \right) \tag{2.12}$$

The following equation 2.13 then resolves for the altitude based on pressure ratio within a layer with constant temperature.

$$h = ln(\frac{p}{p_0}) \cdot \frac{R \cdot T(h_0)}{-g} + h_0 \tag{2.13}$$

Logic Flow

The process to then actually calculate a barometric altitude is displayed in Figure 2.16. The input is p and p_0 with p being the measured pressure via the aircraft's static ports and p_0 being the reference pressure that is manually set by the pilot with the standard pressure being $p_{std} = 1013.25$hPa which varies depending to the weather.

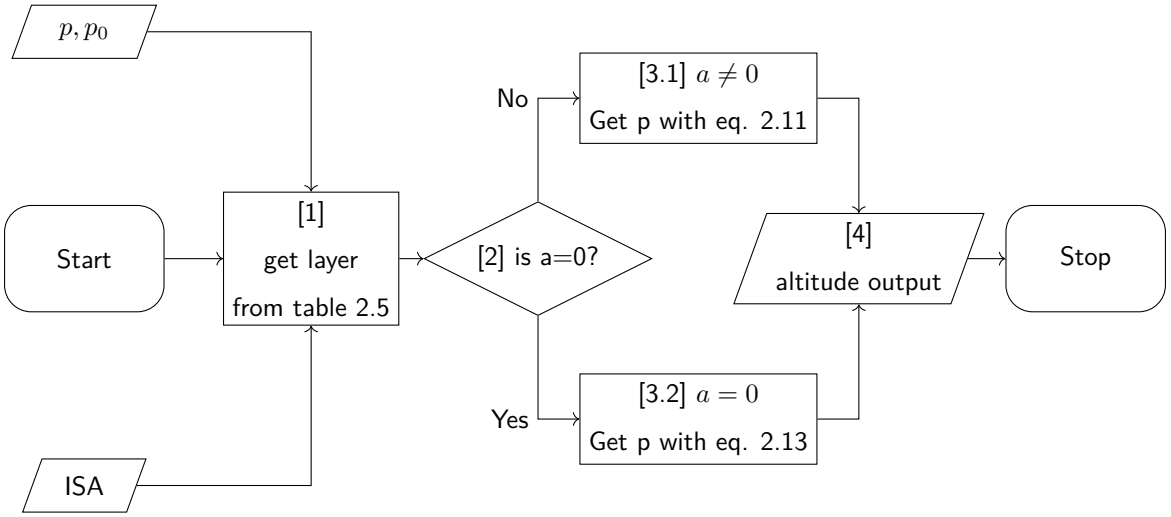


Figure 2.16: An algorithm to determine barometric altitude h based on p and p_0

In step 1 the current atmospheric layer gets found by matching the MSL-pressure ratio $\frac{p}{p_0}$ to precalculated boundary values of layers based on table 2.5. After determining the layer and the value for temperature coefficient a, the altitude can quickly be determined by inserting layer properties as well as the pressure ratio into their respective altitude equation (step 3.1 and 3.2 in flowchart 2.16). In the final step 4 the altitude is returned.

GNSS

In addition to the barometric altitude, the position of the ISTAR is also determined by the Global Navigation Satellite System (GNSS) in the aircraft. For reference, the aircraft altitude generally is defined as the displacement of the aircraft from Mean Sea Level (MSL). It is important to note that the earth can generally be described as an ellipsoid due to its rotational shape in a geodetic context resulting in a very close approximation of the earth's shape. However, varying density levels of the earth's crust cause the elevation and sea level to deviate from the ellipsoid shape. This results in a lopsided model that is modeled in the WGS84 ([SLAT98]) system. This is also the altitude that the GNSS system measures. And it will be the reference altitude for the following calculations.

Possible errors for each are:

Geodetic:	Inconvenient satellite placements,
	deflection of signals in the atmosphere and signal problems
Barometric Altitude:	Meteorological phenomena such as pressure shifts
	or also due to inappropriate reference altitude.

Going into the standardized World Geodetic System (WGS84, [TEUN17]) and the GNSS Altitude however exceeds the scope of this work. In the following, the satellite altitude above Mean Sea Level (MSL) is considered as the reference altitude. Fault detection algorithms within GNSS are described as Receiver Autonomous Integrity Monitoring (RAIM) and are tried and tested within GNSS implementations since high accuracy positioning is valuable for various applications, reaching precisions of up to a few centimeters. Explaining the full function of position calculation exceeds this works' scope. To summarize however, GNSS inputs form an overdefined system of equations which needs to be compensated within some algorithm to form one position based on multiple inputs. [TEUN17]

Summary

The fundamentals of potential algorithms to quantify system qualities have been presented in the past section. The statistical fundamentals have been briefly outlined followed by FMEA implementations from statistics in the shape of Principal Component Analysis (PCA) ranging to implementations from control theory. Advanced models such as machine learning models have not been presented but are generally similar to PCA since they also work without system knowledge and also extract features similar to Principal Components. In addition, the context within the ISTAR has been laid out, introducing the barometric altitude and GNSS systems as the systems that will be focused in the implementation of the Parity Equations.

The research within the general quality control field is generally very optimized with most of its research on analytical and traditional methods having been developed and implemented in the late 20th century [ISER11]. The new arising challenge faced within this work lies primarily in integrating these existing algorithms to allow evaluation of them. In the future such tool boxing approaches may allow quick and easy comparison of new experimental approaches by e.g. running new experimental ML-routines in such a toolbox to then validate them against existing tried and tested FMEA approaches within frameworks such as the ones developed within this work.

In addition to detecting faults it is also necessary for a SHM to display faults to the systems operator. This may happen in the shape of a display interface or a dashboard that generates fault and reliability ratings. Condensing information for operators is also a necessary part of SHM since suspicious occurrences may be frequent and raw information about events is less helpful than already preprocessed data. This is what is now presented in the upcoming sections, first dealing with the structure in which the metadata will be handled and discussing the origin and destination of data in the SHM context.

2.4 SHM Metadata Structures

Data handling as well as structuring the data is essential for fulfilling FAIR criteria in terms of the accessibility and reusability. Thus, necessitating a standardized data and metadata format. In the following, the FAIR principles and the motivation of data handling is presented and then its application in form of data format as well as the metadata format are introduced.

2.4.1 FAIR guiding principles

Standardized data handling and following of conventions in regard of data structures always directly clashes with the desire to optimize data structures, making them more efficient, dynamic as well as developing them further. Were data structures once used as purely SQL (tabular format) this has now expanded into a wide array of data structures that are tightly knit into the needs and requirements of their application. Starting from tree structures, going into graph-theory and furthering into unstructured paradigms, these approaches represent the tradeoff between standardization and accurate data representation. The following paragraph discusses the use of FAIR principles to satisfy the standardization aspect while also allowing flexibility within the system to represent the underlying physical systems. The full FAIR principles are shown in Figure 2.17 detailing the single steps associated with each of the four grand principles.[WILK16]

Findable	F1. (Meta)data are assigned a globally unique and persistent identifier
	F2. Data are described with rich metadata (defined by R1 below)
	F3. Metadata clearly and explicitly include the identifier of the data they describe
	F4. (Meta)data are registered or indexed in a searchable resource
Accessible	A1. (Meta)data are retrievable by their identifier using a standardised communications protocol
	A1.1 The protocol is open, free, and universally implementable
	A1.2 The protocol allows for an authentication and authorisation procedure, where necessary
	A2. Metadata are accessible, even when the data are no longer available
Interoperable	I1. (Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
	I2. (Meta)data use vocabularies that follow FAIR principles
	I3. (Meta)data include qualified references to other (meta)data
Reusable	R1. (Meta)data are richly described with a plurality of accurate and relevant attributes
	R1.1 (Meta)data are released with a clear and accessible data usage license
	R1.2 (Meta)data are associated with detailed provenance
	R1.3 (Meta)data meet domain-relevant community standards

Figure 2.17: The full FAIR guiding principles associated with their individual full description. [WILK16]

FAIR principles

Originating from a Dutch alliance of biotechnology institutes in 2015, the FAIR Guiding principles emerged in 2016 as a general best practice guide for research data, referencing best practices

FAIR principles have been published in 2016 by Wilkinson et al. [WILK16]. They set the foundation for a standardized and open data culture. Within these principles lie values like open access for data, findable and well

tagged datasets, interoperable data by using standardized formats and semantics which guarantee a reusability of data to generate a sustainable process for data usage. Effectively meaning that similar experiments do not have to be performed multiple times when well tagged and formatted data is freely available.

FAIR essentially acknowledges that data analytics have come so far that the data analysis has become very effortless. So effortless however, that a major part of data science consists of formatting and acquiring data. To circumvent this issue in the future and also learn from past data it is then vital to guarantee FAIR principles. This allows to leverage current technology to automate analysis and enable detection of previously undetected correlations within datasets.

SI-Units

Units, especially in aerospace contexts, are far from standardized. Most scientific users prefer the International System of Units (SI) due to its ease of use and lack of conversion factors [NEWE19]. However, the convention for Pilots and Flight Test Engineers remains imperial within units such as feet, Nautical Miles, knots and so on due to their prominent use in aviation. Naturally SI-units are chosen for the calculations in this work since calculations in SI-units are more efficient as well as less prone to errors. Within this work, the base 1 SI-units will be used (e.g. m, s, N, kg, Pa).

2.4.2 Methods

Within this work it is then necessary to use standardized FAIR frameworks for data and metadata exchange. Since the Skystash, a FAIR research data platform is already in development and in prototypical use within the DLR, the developed algorithm will be tested by working in conjunction with the Skystash. To exchange data, the SOIL (SensOr Interfacing Language) framework will be examined for defining metadata of the aircraft and the FMEA logic itself.

Skystash

The Skystash is a platform to distribute, analyze and visualize large flight data sets. It is currently in development within the DLR's digital twin project and aims to be a service platform for uploading and sharing the DLR's flight test data. Various requirements are posed from different stakeholders. And large data sets necessitate a platform that can handle these file sizes. Driven by the FAIR guiding principles the Skystash has unique identifiers for each data point and has a powerful search query embedded in its database structure. Such a database also enables data operations since it eases implementation efforts due to the size of data upwards of 2GB per flight, which rapdidly slows down computing on computers with less memory.

The Skystash internal architecture is presented in Figure 2.18. A nginx web server builds the centerpiece of the Skystash. It controls web traffic and accepts inputs via a python API as well as the website and redirects it to a keycloak service of the DLR for authentification. After authentification the Stash Service processes the request that has been transferred by nginx and forwards it to the appropriate service and checks it for any errors. The

Stash Service is generally the router for three different subsystems. Firstly a database in a mongoDB format which allows a hierarchical structuring of data and has a native powerful search query, improving Findability of data. Additional python appllications can be added to the backend for additional functionality such as upload or import modules for different data formats, allowing straightforward uploads for data. Thirdly, the Service Hub allows for Horizontal Scaling by controlling the docker containers used in the backend which encapsulate each subsystem of the Skystash. If nginx receives a request by the Web GUI it accesses the Web Gui provider to compile the web UI using an

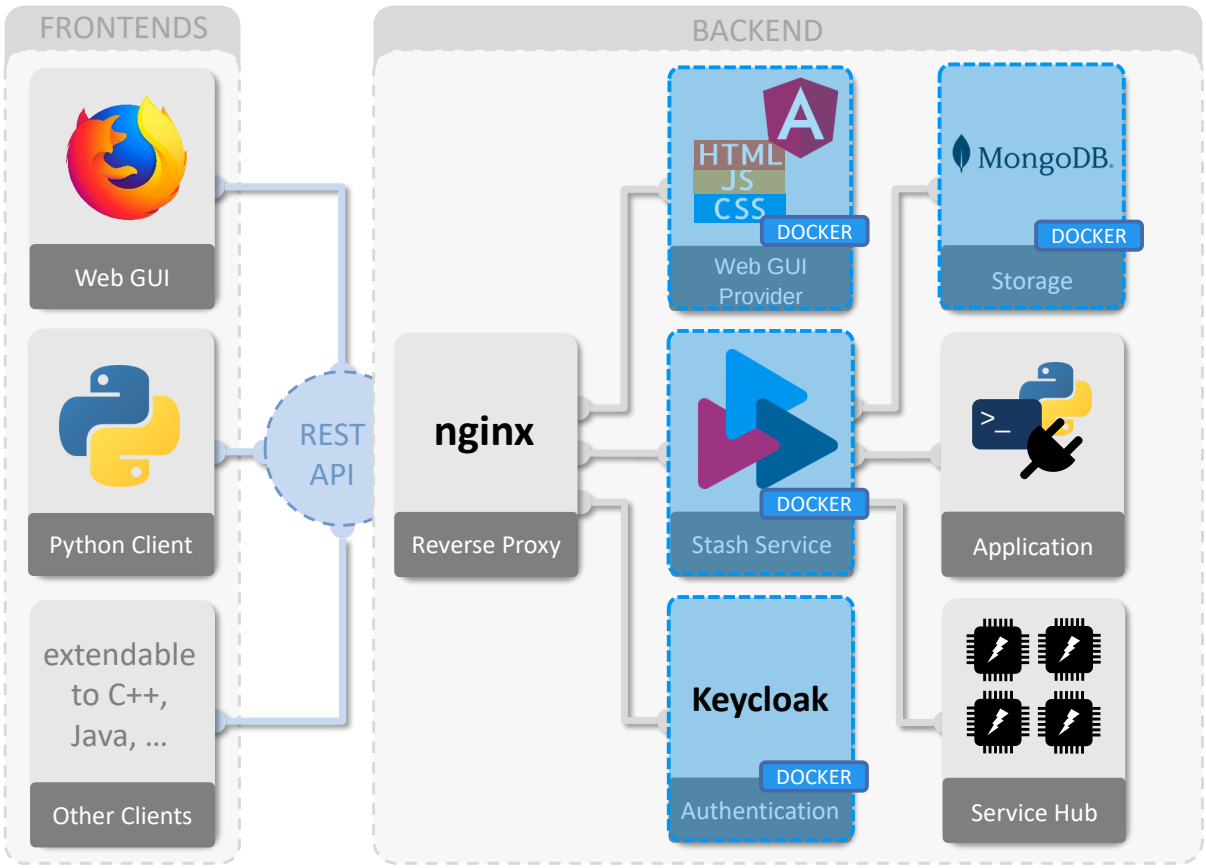


Figure 2.18: The Skystash architecture signifying the high coupling and low coherency by imlementing a REST API interface in order to spatially separate any interactions with the architecture. The backend of the Skystash uses multiple microservices for different aspects of the software by Routing through the *Stash Service* such as a MongoDB database for storage and python clients for applications. A Service Hub takes care of scaling the infrastructure as a whole. [ARTS22]

Flight test data needs to be perceived within the context of its circumstances to extract all its information. Hence the need for a digital twin arises. Within this context all necessary data shall be compounded to set a central starting point for data exploration and analysis. Figure 2.19 shows all data sources that will be gathered. For this works context this means that necessary aircraft configuration data such as location of sensors is available (metadata) next to the actual generated data from the aircraft systems (data). This for the first time allows contextualizing flight data in one place. When previously various metadata and configuration info had to be manually collected, this whole process aims to minimize manual steps within the whole process. [ARTS22]

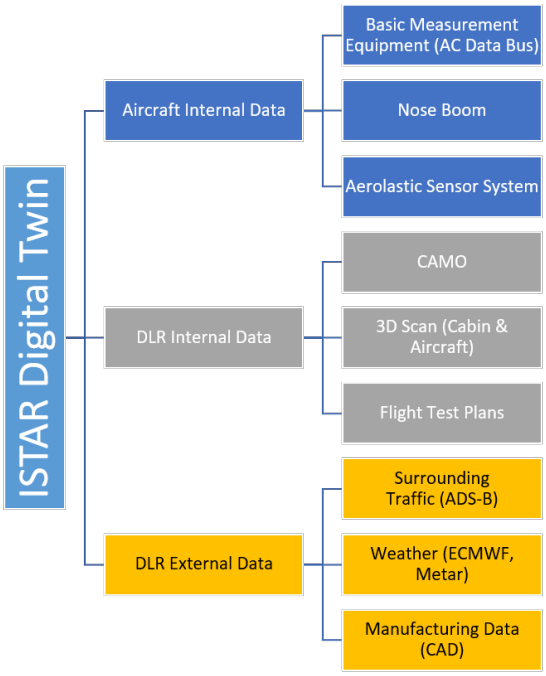


Figure 2.19: The different components of the ISTAR Digital Twin. The components not dealt with in this work include the *DLR Internal Data* as well as the *DLR External Data*. The components actually covered in this work include the *Aircraft Internal Data* as previously mentioned in Chapter 2.1 including the Basic Measurement Equipment as well as the aeroelastic sensor system and the data from the nose boom. [ARTS22]

Metadata exchange format

Aiming for a standardized machine-readable data format the SensOr Interfacing Language (SOIL) is examined. The aircraft data as shown in Figure 2.19 is varied in format and shape. For this work however, a data format is needed that is able to handle and represent the intricate details of the aircraft precisely. For the various sources, a metadata enrichment step might be practicable. Effectively, this would act as a metadata importer that fuses metadata and also translates it into SOIL.

Data Parsing

Another pillar of this work will be the format the original data is available in. The data the SHM will work on originates from the ISTAR and needs to uploaded without any modification to preserve originality and reduce the room for errors. This small section will deal with the handling of data from the aircraft until it reaches its destination in the Skystash.

Firstly, the Data is uploaded to the stash from the aircraft DAQ format. Its data is available in the proprietary IMC *.raw* formats. These are then parsed into the stash by converting them into timeseries for each parameter. The data structure used in the Skystash is represented in Figure 2.20. The folder structure's main path is the project folder which represents the project in which the flight was performed in and contains all the project's flights. Each flight then contains parameter measurements which represent the lowest level of objects in the architecture. All objects of project, flight or measurement type can also contain JSON-usertags as well as

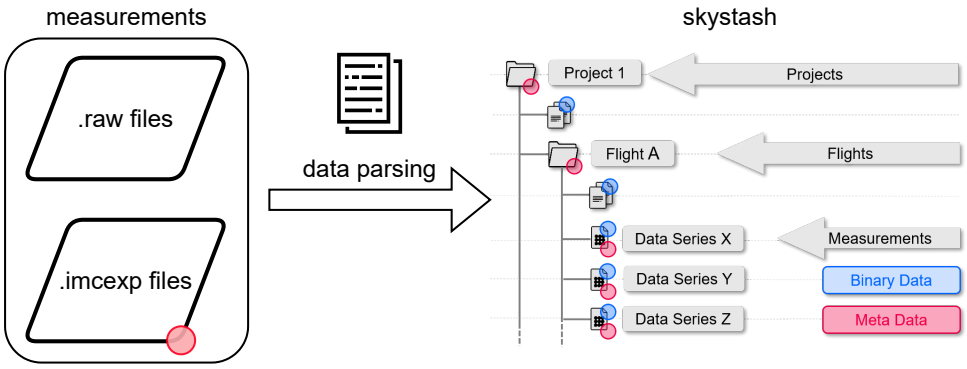


Figure 2.20: Pipeline preceding the data provisioning step shown in Figure 2.9 which consists of parsing the proprietary *.raw* (data) and *.imcexp* (metadata) files provided by the DAQ into a standardized format that is then available in the Skystash. The internal Skystash structure is also shown which consists of three distinct hierarchical levels for research projects which are structured into single flights (recording sessions) which then contain sensor measurements in time-series format. All metadata that exists next to the actual time-series can exist in a JSON-format (shown in red) as well as in an uninterpreted raw format (blue).

additional data in binary format such as pdf files.

To provide a full overview of the entire pipeline this step is mentioned. After a flight, the data is stored as a directory full of *.raw* and *.imcexp* files after a flight. Both are proprietary IMC formats with the *.raw* files containing the actual timeseries in an encoded format and the *.imcexp* files then contain the configuration settings for the specific flight or measurement.

The entire directory then gets parsed by a python script. Segmenting the binary structure of the raw files and extracting the timeseries for each parameter. The DAQ and aircraft sensor configuration for each flight gets exported from the imcexp file and uploaded to the Skystash without modification to preserve genuineness of the data as shown in Figure 2.20. Any Metadata in shape of a JSON format is represented with a red dot and any binary formats such as pdf files or other custom data are represented with a blue dot.

Parameters are saved under the flight object. Configuration data is assigned to the user tags of each parameter. The data is then parsed for each flight and inherent sensor. The ISTAR DAQ metadata is currently the only metadata in the Skystash. Since this metadata needs further information to make it understandable it will be enriched further within the Configuration step (See Figure 4.1).

Use of FAIR in the implementation

To now fulfill FAIR principles, the interfaces in the architecture are examined. The generated flight test data is available and metadata is already in a JSON-file format. To further specify and standardize metadata the metadata can be translated into a standardized format to further Interoperability. This enriched metadata then can be fed into the FMEA step. Exiting the FMEA-step, the Analysis results can also be defined within a SOIL/JSON format and then get opened by other applications to access data quality results and perform further analysis steps, increasing reusability.

By implementing standardized JSON-data structures for the metadata including dataset and report, interoperability as well as reusability are guaranteed. To increase findability, the Skystash architecture is used which generates a unique identifier for every dataset. [MEYE20]. By enriching metadata, findability as well as accessibility is guaranteed. And finally, converting to SI-units guarantees accessible, interoperable and reusable data.

2.5 SHM results configuration

Within the last step of the FMEA, the generated report results need to be analyzed which in turn might necessitate change of the FMEA parameters. To facilitate the process and lower the barrier of entry, the possibility of an interactive report respectively dashboard is examined with the ultimate goal to output a report that is clear, concise and standardized.

The requirements for such a report are firstly motivated by necessity. It is vital to show error detection, embed the report into the toolchain and find a format to communicate metadata as well as SHM findings. Such a report tool could then also be used in the development of new algorithms since it generates quick feedback and precise feedback, aiming to display various Indicators for data quality and FMEA quality.

2.6 Summary

In chapter 2 the fundamental definitions for system behavior from the field of control theory have been set forth as standardized descriptors of this work. The theoretical foundation has been created for this work by explaining methods from the field of FMEA and parity equations have been chosen for the further use in this work. Furthermore, the software-side of this work has been introduced. With metadata frameworks and the Skystash platform forming the infrastructure that this work rests on. The learnings from the metadata models within the SOIL infrastructure and the ontology field have been examined and will be implemented in the following. FAIR principles have been presented and have been contextualized within this work. Building upon these fundamentals the next chapter will present an in-depth problem description followed by Chapter 4 presenting the implementation details as well as the implemented methods in this work.

3 Problem Statement and Methods

This chapter serves as a brief, but in-depth summary, concisely detailing the problem. The approach to this work is then outlined by introducing the methodology used in this work.

3.1 Developing a Sensor Health Monitoring

Problems with large-scale measuring systems in comparison to smaller experimental setups is the sheer scale and inefficiency of manually checking for errors. Since errors in aviation contexts mostly have fault rates of $10^{-6}/h$ it allows for some errors during runtime even with large datasets of up to 5000 sensors. However, many sensors in this application are built-in experimentally and can be assumed to be at a fault rate of $1/h$.

This work then shall start with the Skystash architecture in mind. Firstly, it needs to be assured that the available data is sensibly imported into the Skystash. A second hurdle lies within the metadata treatment. Metadata needs to be parsed from the proprietary .imcexp format. Steps undertaken will then involve uploading the available data to the Skystash without modification, forming the first step of data provision. Metadata enrichment follows as the next step, enriching the DAQ metadata with additional sensor information, SHM information such as boundaries for Level 2 as well as Level 3 tags for mapping parameters onto categories. Within this step all this data is then taken and transformed into a standardized format (like the SOIL format) that enables a standardized interface to the data processing step. Data processing builds the next step. The previously standardized, enriched metadata is used to check Levels 1, 2 and 3 as presented in chapter 2. For Level 1 (Completeness) the challenge lies in accessing the configuration for expected sensor values. For Level 2 (Plausibility), statistical methods will be used to examine single sensors for plausible behavior such as noise and hard limits. In Level 3 (Correctness), the correlation of parameters has to be made to detect if sensors deviate strongly from other sensors measuring the same value. Also a way has to be found to implement physical relations. Building a visualization of the reported parameters is the next step. Here, a User Interface will be implemented to give a high information density, allowing users to gain an easy overview but also facilitating trouble shooting by giving a customizable depth of detail.

This work needs to be highly modular to allow parallelisation of tasks and also limit complexity of the codebase. Computing with large flight data sets is time-intensive and modularizing the logic as well as the data allows for easier fixes and better maintainability.

3.2 Methodology

Since SHM is a multifaceted problem that represents a grand undertaking with many moving parts and optimizable parameters, algorithms and systems that may be implemented, it is firstly considered necessary to develop an ecosystem and a process in which SHM algorithms may be tested, evaluated and optimized. To develop said ecosystem, the turbine model for development of Human-Machine Systems is featured (see Figure 3.1). Its process can be applied to this work without much change. Starting from Ideation stage that represents the beginning phase as well as the first Chapter of this work. Within Ideation phase the project orientation

takes place. Answering the question of which goals are considered viable within the given constraints of time and manpower as well as ressources such as knowledge and available tooling. After Ideation Phase the vertical movement representing creativity and innovational ideas is centered into a level-headed assessment of the problem containing goals and boundary conditions. Using these condensed ideas and values, the Research and Technology Assessment Phase is kicked off, represented by Chapter 2 of this work in which various technologies are examined for usage within the given problem context. Based upon a wide and thorough research feasible ideas and approaches can now be arranged within an infrastructure to solve the given problem. Fitting together various approaches from the literature like puzzle-pieces and condensing the ideas into a more refined concept, concluding literature review and representing a first draft. Using this first draft, the development cycle can now be started which can also be represented by the spiral model for development as presented by Boehm [BOEH86]. The challenge of this third stage is now to implement the draft given in the previous stage, this gets accomplished by bouncing from an exploration step in which the theoretical constructs from the previous steps are implemented into a review step in which the systems are rated upon usability. This also represents the procedure of chapter 4 in which the theory from chapter 2 is implemented and developed. After having then developed a prototype of this work the results for our test cases will be assessed (sensor malfunctions) in chapter 5.

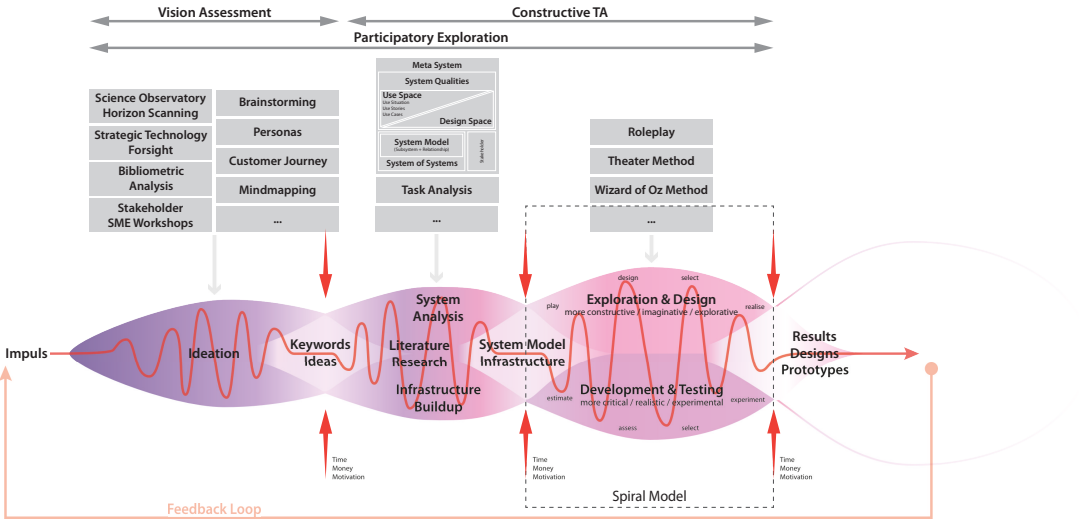


Figure 3.1: The innovation and exploration turbine in the context of this work as a nested feedback loop. [FLEM22]

4 Methods

The implementation efforts of this thesis are condensed into the following chapter. A holistic perspective on the data process is described in the following step motivated by the urgency that Sensor Health Monitoring needs to be perceived within its ecosystem. A SHM thus is directly reliant upon the data in the first place. It is then also reliant upon the configuration metadata since it will contain necessary information for processing the data further within the SHM. To then close the feedback loop it is also vital to generate a sensible data display that allows quick evaluation of the SHM since manual evaluation methods would slow the development process by orders of magnitude. Manual methods would not only cause slow-down but would also be more prone to errors than automated processes.[BARC11]

4.1 Introduction

The implementation can be detailed by Figure 4.1. After having uploaded the Dataset the relevant information on the sensors as well as the aircraft properties is gathered and condensed into a single configuration metadata-set represented by a JSON file. This JSON file is then appended to the Dataset within the Skystash architecture represented by the Online side that is accessed by the API. Processing the data now becomes a pure blackbox operation that is only fed by data received by the API returning its report containing notable occurrences via the API and storing all relevant information online. Since this software is developed for flight operations and an interactive visualization of the report data facilitates evaluation of the generated report data a User Interface within a dashboard application is developed. Additional benefits contrary to the generation of PDF reports are dynamic updates as well as interactivity and an agile update cycle considering the spontaneous emergence of bugs.

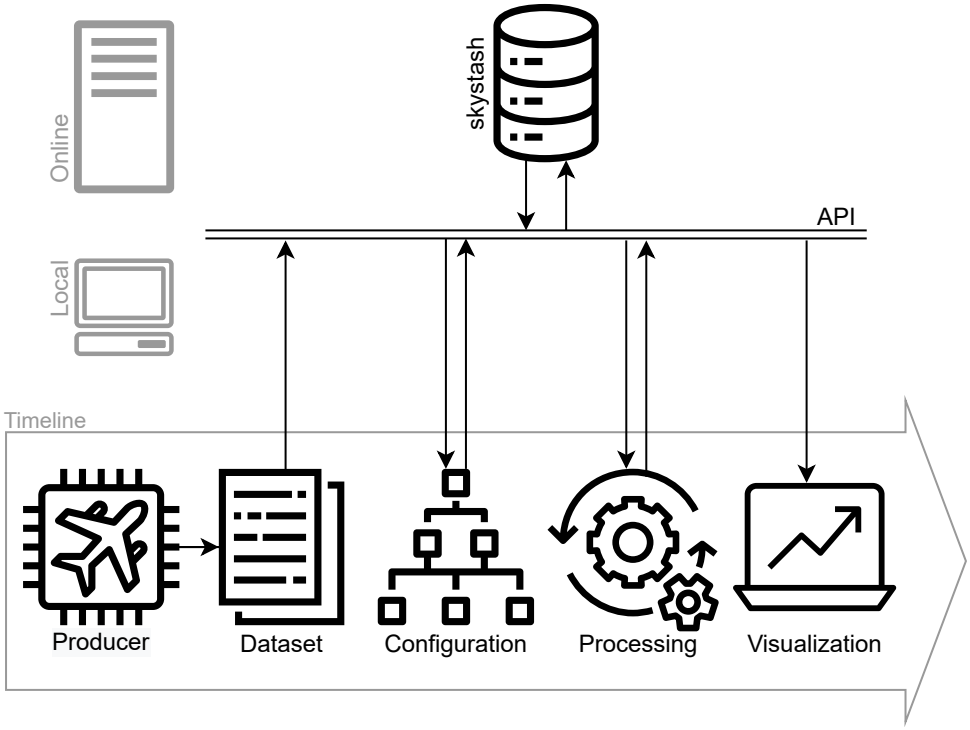


Figure 4.1: The data toolchain prospectively used for Sensor Health Monitoring results from transforming the flow from Figure 2.9 into single encapsuled functions that work by interacting with the Skystash. While the *Dataset* and *Visualization* step work by just writing or viewing the data the *Configuration* and *Processing* read the existing data and append new data.

4.2 Configuration-Metadata

The configuration from the previous step has now been uploaded to the Skystash. It describes the actual flight-dependent DAQ configuration and changes for every flight. It is, however, not complete. Since the DAQ only saves data relevant to itself, other vital metadata for users as well as the SHM are not present. In this section a method is implemented to enrich the DAQ metadata by transforming it into a common metadata format as well as appending useful additional metadata to each sensor and the flight. This permanent, generally unchanging metadata is first specified locally and contains values such as boundaries for Level 2 or tags for Level 3 to correlate parameters to each other. It is also specified for each parameter individually. It is then also possible to assign values such as frequency locally.

This necessitates a merging/fusion step of DAQ-configuration with the locally defined permanent metadata. This is especially useful since this locally defined metadata is not up-to-date and DAQ metada is not complete. Additionally, check configuration info such as checking limits and correlational information can be appended to the metadata structure in this step.

4.2.1 Requirements

This motivation culminates in the two main goals for the configuration step as shown in Figure 4.2.

1. Enrichment of Metadata
2. Conversion into a standardized format

These two steps guarantee a similarity across processes which is also vital should the SHM be expanded to other applications. An unstandardized metadata format would necessitate a large-scale adaption otherwise. The standardized format also supports the desire for FAIR principles since it allows to input data from various sources into the SHM data processing algorithm.

Additional attributes that are needed are aircraft specific information such as position of sensors within the aircraft coordinate system (COS), the actual position of the COS relative to the aircraft and its orientation as well. With such information a Level 3 algorithm can correlate different sensors using these additional correlations and enable additional corrections. For the SHM Level 2 physical limits as well as amplitudes for STFT are needed to give the algorithms boundaries to check the sensor values against. These boundaries are given within the permanent configuration as shown in Figure 4.2 and can then be used by the algorithms. For Level 3, only the parameter tag, defining its meaning in the application context such as *static pressure* and an eventual reference parameter is defined. The reference defines context or calibration and defines the *reference pressure* in the case of the *static pressure*. These tags then allow the Level 3 function to get all parameters containing the tag which then enables a method such as a Parity Equation to create a state model based on these parameters.

Generally, metadata exists in two forms. the actual flight-dependent DAQ configuration and the permanently valid metadata configuration that principally does not change. The actual configuration is different for each flight since it represents changes made by Flight Test Engineers to fit the DAQ-configuration the specific flight experiment. The permanent configuration however represents all additional info that may be relevant to a varied pool of users.

Once a merged data construct (as shown in Fig. 4.2) is reached it needs to be converted into a usable, standardized format. For this step, some candidates from Internet of Things (IoT) applications have already been examined in Bodenbenner, Montavon, and Schmitt [BODE22]. For the pipeline to the data processing the metadata can then be transformed into the SensOr Interfacing Language (SOIL), allowing a concise, specified format for this essential metadata.

4.2.2 Implementation

Required data then needs to be defined locally. For this, an already existing data-source is taken as a starting point. This data-source is in the shape of an excel file and will be used for development. In further steps it might be reasonable to migrate this data-source to a different format to guarantee true provenance for configuration information. However, this is reserved for a later point. It suffices for now and will be expanded for columns that can add attributes for each parameter such as the aforementioned Level 2 and 3 requirements and also additional data that may be of interest for users in the Skystash such as sensor position in the aircraft.

The metadata fusion is part of the feedback loop that is in-line with the data processing and feedback. This step is essential since it enables a quicker development since all steps displayed in the SHM-toolchain in Figure 4.1

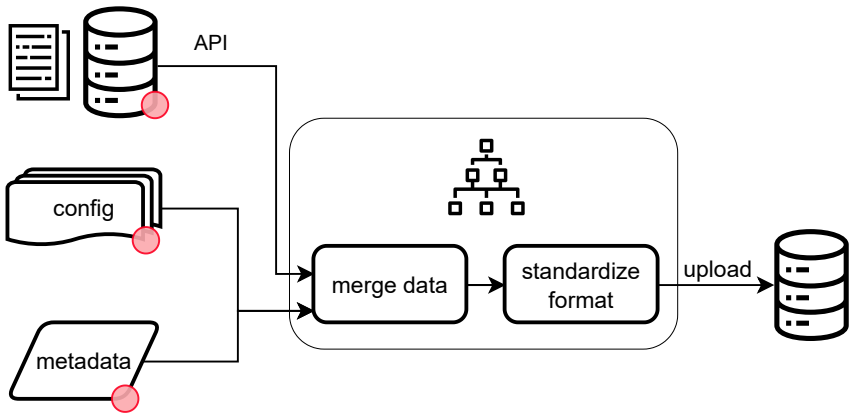


Figure 4.2: The *Configuration* metadata Pipeline as introduced in Figures 2.9 and 4.1. Metadata from the DAQ gets read from the Skystash via the API and additional local metadata and configuration gets appended containing e.g DLR Internal data as shown in Figure 2.19. The merging happens by prioritizing the DAQ metadata since it is assumed to contain the truest descriptors of data (see merging in Figure 4.3).

had to be performed previously if there had been any marginal changes. Compartmentalizing and modularizing these single steps allows to then accelerate development and testing by not having to parse data within each iteration.

Also, attributes that are generally transmitted within the DAQ configuration are sampling rates, data origins and other miscellaneous information related to the system. Not contained are information about general sensor description, position of sensors, overall setup of the aircraft sensor architecture and check limits for data values. To preserve the original DAQ infos as closely as possible the merging step as portrayed in Figure 4.3 is proposed in which conflicting values from DAQ and permanent configuration get overwritten by the DAQ. In this way, original DAQ info that is frequently changing such as the frequency gets preserved while still allowing additional data by the permanent local configuration to be used.

The metadata enrichment process is described in Figure 4.2 starting with data and metadata that is available in the Skystash and unmodified. This unmodified data then gets enriched with metadata for parameters that are not already described by the DAQ configuration. In this step other SHM attributes get assigned such as checking limits for Level 2 and variable tags for Level 3 defining the meaning of a parameter. Since the metadata uploaded to the stash in itself is not very expressive and cryptic it needs to be refined further into a usable, concise format.

As previously discussed in chapter 2.4.2, a common format for exchanging metadata is needed. Following Fusion, the data is now available in a JSON format that contains the previously defined attributes for each parameter. To increase standardization a further conversion into the SensOr Interfacing Language is considered. This could increase acceptance and standardization since it is starting to be increasingly used within IoT contexts. A proposed structure would contain single sensors as parent elements and components containing the appended attributes. [BODE21]

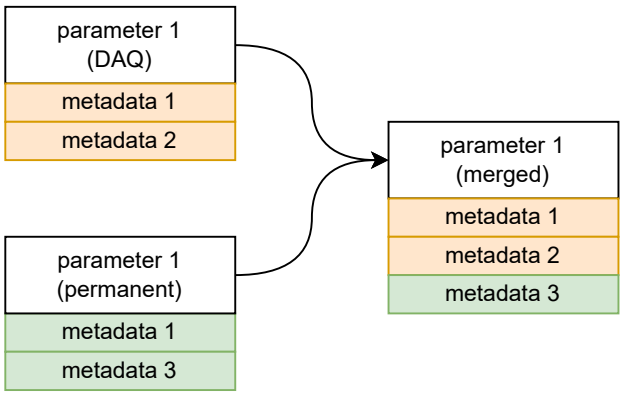


Figure 4.3: Priority Merging of Skystash data for an arbitrary Parameter 1 as shown in Figure 4.2. Using the actual-DAQ configuration other configuration information for same metadata parameters such as *sampling rate* gets overwritten since the DAQ is assumed to contain the truest and latest contents.

4.2.3 Summary

A central format for metadata has now been agreed upon. Especially data flows have been determined which strongly facilitates further development by structuring input and output operations. Primarily, the place to define further metadata attributes has been defined and the step to compile it within the configuration data merge step. Naturally, this will have to be changed further within further development. This method however condenses work into one single place. The incomplete metadata from the DAQ has been enriched with the locally available metadata such as additional knowledge about the aircraft, the sensors itself and its context. Also, necessary data for the SHM step has been added such as limits for checks as well as tagging parameters for physical correlation checks which has been fused into a single data structure. After compiling various inputs into a single metadata structure a common format has also been found. SOIL has been examined for usage and implementation is intended for future use.

4.3 Data Processing

Now starts the delve into the vital step of this work. Taking the inputs of data and metadata with the configuration needed and implementing sensible logic to reliably detect anomalies in the data. This will first be preceded by some considerations about software architecture and then delving into the specific methods for FMEA.

4.3.1 Software architecture considerations

Careful consideration needs to be given to the workflow of the Level 1 check to allow scalability and reduce manual interaction. To achieve this architecture manual steps are reduced as far as possible. However, some level of configuration must be implemented otherwise only relational sensor behavior could be detected. Meaning that sensor faults occurring temporarily within an experiment can be detected but permanent behavior is not noticed by a relative algorithm e.g. only detecting strong aberration. To satisfy this desire for information the preceding metadata configuration step in chapter 2.4.2 is implemented. A special focus lies on the modularity, modifiability, reusability and the ability for a quick and smallest possible feedback loop to be assured within this work.

4.3.2 Check 1 Implementation

For implementing the Level 1 checks, the measured data needs to be compared to the expected data. To gain insight into what the expected data is, the configuration file of the data acquisitioning system needs to get parsed as presented in chapter 2.4.2. The ISTAR DAQ configuration file provides the expected values while the Skystash provides the actual parameters that are available.

This DAQ flight-specific configuration file can be exported within the software *IMC Studio*. This reads *.imcexp* files and then allows export into a .xml format. Downsides are though that various manual steps are involved in this process and it is thus prone to errors. So, instinctually and naturally an attempt for automating it is launched. It turns out that the DAQ's imcexp format is in fact a .zip directory. The 7zip command line tool allows opening the configuration file since the configuration file's format is not fully complying to the zip standard making several python libraries fail during the process. This stems from an issue with the zip header and footer parts of the file that are not at expected places (effectively in the front and back). Once opened, the configuration file contains multiple files as well as an essential xml file that contains the needed sensor metadata. After some formatting these *.imcexp* can be read straight to generate similar info to the *IMC Studio* workflow.

The expected data then can be extracted from the *.imcexp* files and allows quick comparison against the actually generated data. This resolves the Level 1 implementation.

4.3.3 Check 2 Implementation

Within Level 2, the sensor timeseries get checked without contextualization of other sensor data. This gets accomplished by principally checking them against predefined limits as shown in Figure 4.4.

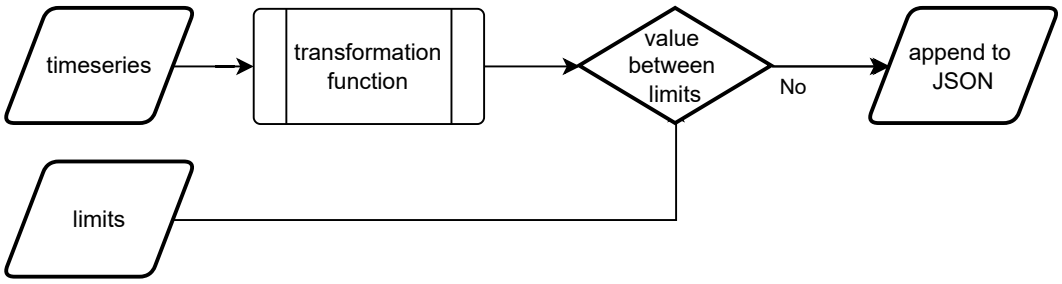


Figure 4.4: The checking Logic valid for every type of Level 2 function. Any timeseries, transformation function and desired limits serve as inputs. With the transformation function the data can be transformed in any desired way such as integrating, differentiating or modifying in any other way. After the transformation the data gets checked whether any data points are outside the limits and are appended to the report JSON if that is the case.

Use Cases for this are primarily the two implementations:

1. unmodified signal
2. noise amplitude by using STFT amplitude 2.3.3

Methods

The general workflow consists of getting the timeseries of the signal from the Skystash database, then transforming it in some respect and then checking it against limits.

The logic in this step can then generally be summarized in equation 4.1 with l being the limit that is checked against.

$$f_{report}(t) = \begin{cases} t : True & \text{for} \quad f_{value}(t) < l_{min} \vee f_{value}(t) > l_{max} \\ None & \text{for} \quad l_{min} < f_{value}(t) < l_{max} \end{cases} \tag{4.1}$$

The context $f_{value}(t) = g(f(t))$ is additionally used. Representing any transformation operation as the function g.

In the following, the three different types for Level 2 checking are introduced. Firstly, the topic of checking for invalid values, then the boundary check followed by the signal movement check which can be understood as the derivation of the signal.

Checking for nonexistent values is less straightforward than it seems within the Skystash. Since empty values are filtered out a method is needed to check for start and end time as well as the time-distance between singular data points. To detect *None* values to options can be considered:

1. check parameter start and stop time
2. check sampling rate

Table 4.1: Exemplary limits for checking parameters in Level 2 for the case without any transformation function.

Parameter	Minimum	Maximum
Barometric Altitude	-1000ft	40,000ft
Static Pressure	1Pa	120,000Pa

In the second step, the signal is checked upon crossing of boundaries. This means ranges such as defined in Table 4.1. Of course, this selection is biased and may not be accurate for most mission profiles like atmospheric research aircraft that cruise in altitudes of up to 45,000 ft MSL.

Movement check

The next category examining the sensor behavior can be classified into sensor movement being too low or sensor movement being too high. Movement being defined within this context as the difference of a new sensor value to the previous one.

Hence, an approach using a STFT is chosen in which the amplitude is averaged across all frequencies from its spectrum. This guarantees a generalized feature extraction, resulting in standardization for parameters. Based on this spectral analysis, the logarithmic order of the variable can be estimated within its dataset. Previously, the frequency is assumed via time difference of start and end time divided by $n_{datapoints}$. The STFT also implements preprocessing steps that would otherwise be necessary such as translating the signal by its average value as well as scaling it by its standard deviation. Would one be interested in examining a signal using functions from a statistical toolbox, a similar result may be achieved by performing the previous steps of averaging and scaling the signal and then examining the variance within a moving window of the signal, similar to the STFT. The size of the moving window for such an analysis is defaulted to 256 data points for each signal. Certain issues with methods of moving windows are however, that they are not very meaningful for the beginning and ending of datasets within which the window is not fully occupied. This may lead to spikes for the STFT, which is why it is found that the STFT is generally more powerful to examine insufficient sensor movement.

This can be described in equation 4.2 as:

$$g(f(t)) = \frac{1}{n} \sum_{m}^{n} F(m,t) \tag{4.2}$$

$f(t)$ = sensor data
$g(f(t))$ = t
$timestep$ =
m = frequency
n = amount of frequency points
$F(m,t) = \text{STFT}(f(t))$ = the STFT of f [SMIT99]

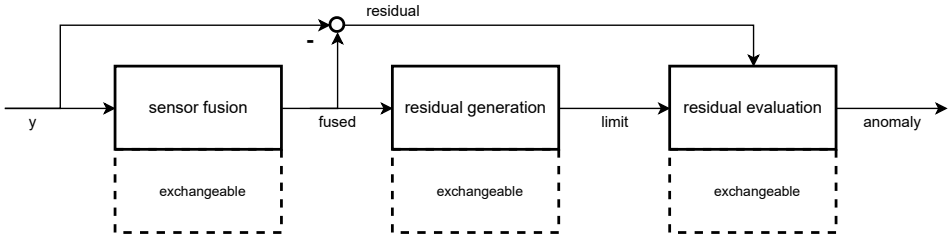


Figure 4.5: In Level 3 a Vector of y is merged in a sensor fusion step in order to generate a residual. Limits to evaluate the residual are generated in the next step followed by the residual evaluation determining if the value is anomalous.

4.3.4 Check 3 Implementation:

Aims of Level 3 Implementation are to model the aircraft's state in a reference system. Parameters are present in various reference systems as well as in redundancy. Within this section the issues arising from the merging from parameters will be the main topic, accompanied by how to implement the necessary data.

First, a common reference state will be defined for checks, then the Parity function 2.3.4 will be implemented and then the topic of how to deal with data input and output on a large scale will be discussed.

Finding a reference state

As discussed, many parameters are available that allow backtracking onto a single state variable of the aircraft. They are also available in various COS.

Aircraft position and state can be interpreted in various Coordinate Systems. It is also necessary to compare redundant parameters within these varied reference systems. Hence, the reduction of the various reference systems onto a single reference system is necessary to guarantee comparability.

Since the geodetic reference system is best described by the GNSS and Inertial Measuring Unit (IMU) it will be used in the following as a reference system for calculations. Additionally, moving aircraft coordinate systems like the aerodynamic and the along track Coordinate system may be derived using known transformation angles and velocities. [BROC11]

Dynamic configuration and correlation of parameters

The necessary configuration for the Parity Equations should then be expandable and dynamically adaptable. This means creating a dynamic structure that allows to represent the complex physical aircraft correlations. This includes creating hierarchical configuration information and an algorithm that reads this configuration and then downloads the sensors from the Skystash to include them into the essential Level 3 flow represented in Figure 4.5. The detected anomalies then get uploaded to the Skystash.

For the Level 3 configuration and information assignment a data structure similar to the SensOr Interfacing Language is proposed. An exemplary structure is shown in Figure 4.6.

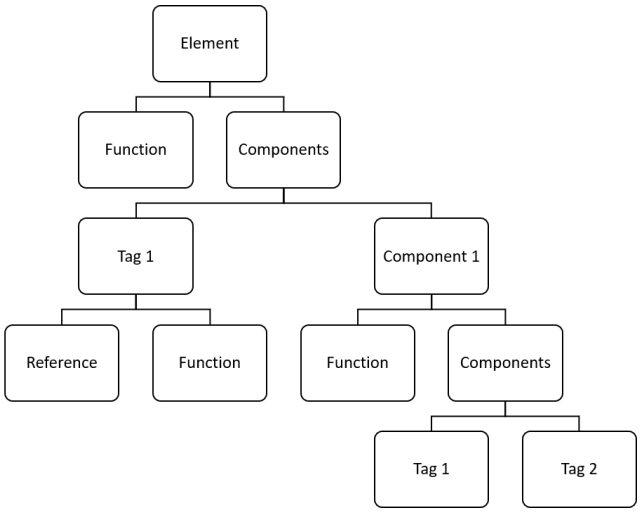


Figure 4.6: General structure of physical correlation setup for determining the exchangeable steps in Figure 4.5 is defined by the UML diagram in Figure 4.7. A parent *Element* contains multiple *Components* that are the independent system variables. These *Components* can then be either a *Tag* that references a specific parameter or an entire new *Component* that can then contain a *Function* to perform transformation on its subcomponents to adhere to the current stage of the configuration hierarchy.

This dynamic description for parameter correlations is required in order to implement an efficient and quick way to assign parameter correlations. It is then assumed that a tree structure is fit for this task since parameters can be correlated to other upstream parameters by using a single flow direction. A more detailed view in a Universal Markup Language (UML) graph is shown in Figure 4.7. The Class component generally forms the common element in the structure as seen in Figure 4.6. It is also notable that Tags as well as Elements inherit from the Components Class and are instances of it. Components possess the attributes of other *components*, their own *name*, the *merged parameter* which represents a single fused value for all parameters below and an attribute containing all lower lying parameters called *all parameters* which contains all parameters. It also contains two functions for firstly merging its components into the merged value and secondly transforming its parameters into a desired state dimension. E.g. converting Static Pressure into a barometric altitude. The current default of these functions consists of relatively rudimentary logic to not transform the parameter and to take the average of values to merge various parameters.

The inheriting entity *Tag* mainly differs from the *Component* in that it does not contain any subcomponents and contacts the Skystash to gather all variables from the database containing the tags that are represented by its name. An optional attribute is a reference or calibration value that is assigned in the permanent configuration database (See chapter: 2.4.2). The entity *Element* represents the parent node of the configuration tree. Its only difference to a regular *Component* is that it allows to keep multiple merged parameters in its view to e.g. allow altitude and speed to coexist and not get erroneously merged.

Using this configuration method, the Parity model can then be calculated using given methods to transform and fuse the data. This starts by parsing the configuration structure, downloading sensors for each Tag and fusing sensors y into a fused value as defined in the configuration and as shown in Figure 4.5.

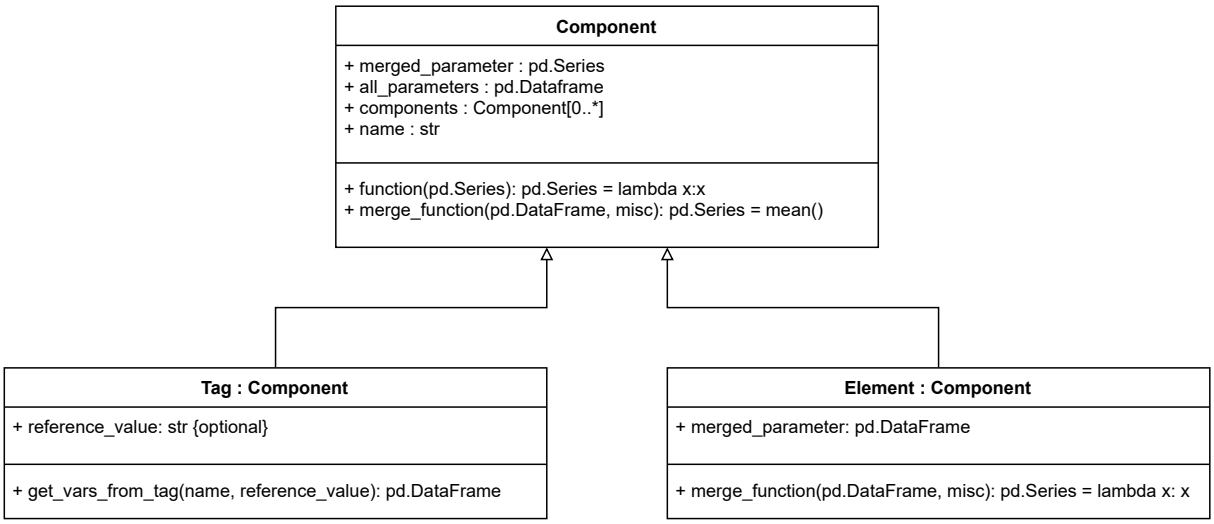


Figure 4.7: Description of the Level 3 JSON check configuration developed in this work based on SOIL [BODE21]. A general *Component* type can be extended by the types *Tag* for referencing parameter tags defined in Skystash parameter metadata or the *Component* can be extended by the *Element* which is the parent element of the configuration structure and thus can contain independent state variables (*Components*).

This system takes inspiration and inherits the configuration structure from the SensOr Interfacing Language (SOIL)[BODE21]. In this work's use-case the *Element* ISTAR research aircraft contains top-level components that are the independent state variables of the aircraft. An example that will be invested further in following sections is that of the *static pressure*. The workflow begins by defining the Tag *static pressure* in the permanent configuration and then configuring it into the Level 3-configuration structure. This tree structure is then parsed and calculates a value for each level of detail taking into account its lower lying neighbors. Some weighting also has to be considered since a number of redundant pressure sensor could skew results against a single GNSS parameter. Thus, a condensed parameter is calculated for each degree of freedom that is based on predefined algorithms and tags that can be freely defined within the configuration file that are then recursively parsed within the tree structure.

Limit Generation

Now the fused state value as well as the single, transformed sensor values are present. The next step consists of defining a limit for the resulting residual from combining the fused state with the single sensor value. Naturally, this could be defined manually by e.g. defining a limit of 0.5 meters for an altitude value. Since however automation is sought after, an automatic limit generation is determined as a requirement for dealing with these large data sets.

Statistical methods then are examined for automatically generating thresholds for the parameter values. A method is then implemented to detrend the signal and take the resulting noise as the allowed deviation. In practice this results in a rudimentary high-pass filter (HPF) that gets realized by using a moving window function combined with an averaging function with a window width of $n = 256$. After using this crude HPF to extract noise from the dataset the standard deviation of this noisy signal can be acquired and used to set the

limit for resulting residual evaluation.

Residual Interpretation

After generating the residual as well as the limits for the checks anomalous values get detected by checking whether they are outside the allowed range (see equation 4.1).

Using this generic approach primarily guarantees quick and easy operations. It remains yet to be seen, how well other methods might perform compared to it. A manually defined residual limit then might guarantee more custom boundaries and using a probability density function (PDF) implemented in works such as [SVÄR14] may also promise to deliver good results but nonetheless exceeds the scope of this work.

4.3.5 Report Structure

A standardized report structure is needed to store the results from the SHM but also allow reinterpretability of the detected anomalous values for future users. In the Skystash the metadata is appended within the flight hierarchy level and then also the parameter level to allow dynamic expansion of the data structure since in Level 3 always some kind of report data is generated. In Figure 4.8 the SHM-report for the flight level is shown. The structure is chosen for its clear distinction into the the SHM-levels and storing as much as necessary while also as little as possible as to not generate any bloat and boilerplate since the flights themselves are already richly described.

In Level 1 a list is appended for all missing parameters as previously introduced in Chapter 4.3.2. In Level 2

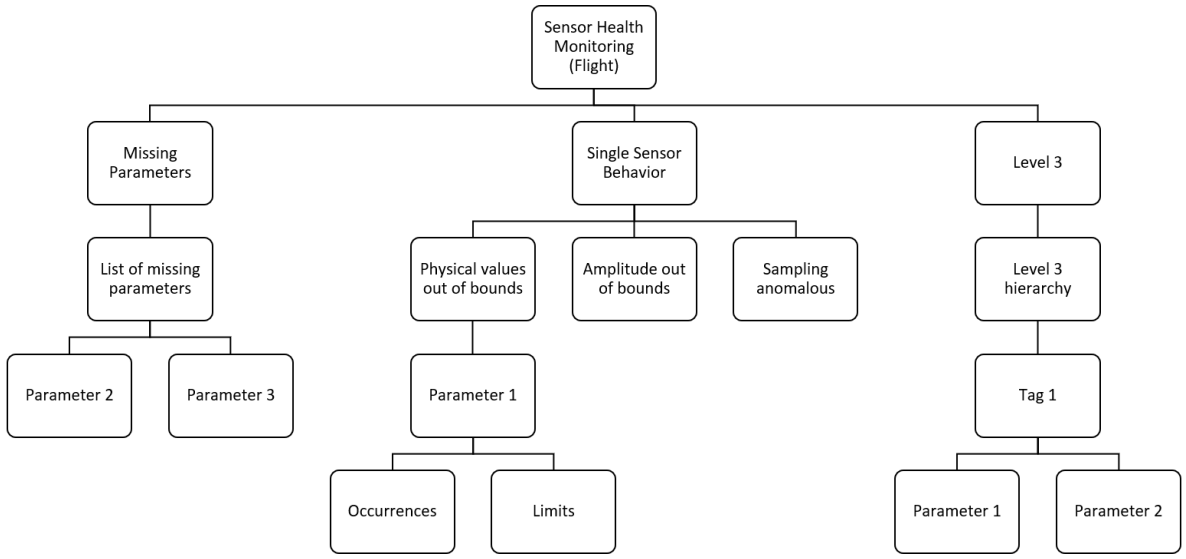


Figure 4.8: The Sensor Health Monitoring report on the flight level in the Skystash hierarchy. Level 1 is represented by the *Missing Parameters* box followed by a list of Missing parameters. For Level 2 each subroutine with their own transformation function possesses a sublevel such as the *Physical values out of bounds*. These sublevels are generally structured similarly with a parameter as the next step which contains occurrences with timestamps as well as the limits upon which it was tested. Level 3 is followed by the Parity Equation hierarchy upon which subtags are followed by the detected parameters in the flight set.

(see Chapter 4.3.3) all subroutines need to be taken into account since it should be easy to expand for more Level 2 functions in the future. Therefore, a substructure for each different kind of check is implemented in which each parameter is visible once it has been detected at least once. Upon detection the checking limits as well as the occurences with timestamps in UTC are appended. Next, Level 3, the previously introduced hierarchy in Figure 4.6 is appended to allow retraceability. The only difference is now that tags contain the parameters that have been detected in the flight set to contain these tags. The remaining information of Level

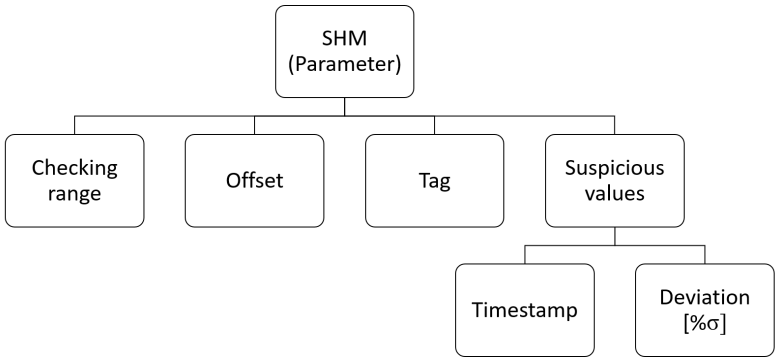


Figure 4.9: Level 3 SHM results get appended to single parameters which then display model parameters such as the Offset to the Fused value, the checking range in which they differed to the fused value and the Tag that was previously configured in the general metadata (see Figure 4.3). Any values that exceed the limits are then appended to the *Suspicious values* with timestamp and the ratio to the set limits.

3 is then shown in Figure 4.9 as the hierarchical structure appended to the parameter. The structure contains the detection information as well as the data used to check for errors such as the *Checking range*, *Offset* to the fused value and the parameter *Tag*. This structure is chosen to allow reusability and retraceability for future users to not just show detected values but also allow comprehension of why these values were detected.

4.4 Report Visualization

At this point, the data has been checked and a report within a JSON-format has been generated and appended to the Skystash flight. The missing parameters have been found, sensor values have been checked by themselves and additionally physical correlations and redundant sensors have been checked. However, FMEA algorithms generally need some tuning to become their most effective and analyzing the JSON data takes some time to grasp all peculiarities of the data. Facilitating and accelerating development, the STASHBOARD is introduced. It is an interactive data report that is developed for data analytics. And instead of a pdf type report it works more efficiently and rather follows the paradigm of a single source of truth. It allows for dynamic updates and thus does not represent a data source in itself but rather view on the source facilitating updates on the data. It allows to quickly gain an overview over the state of sensors and closes the feedback loop in the development cycle.

The flow of the Stashboard allows selection of any stash flight by using the Skystash API through dropdown menus. The Stashboard can then access the flight's metadata and represent it by using the previously developed data displays. This results in a flexible, scalable analytics application for Skystash data and SHM reports that gives a quick overview over the vast amounts of data and be used by flight test engineers to get an overview over data quality as well as developers to optimize algorithms. The first prototype of the developed application is shown in Figure 4.10.

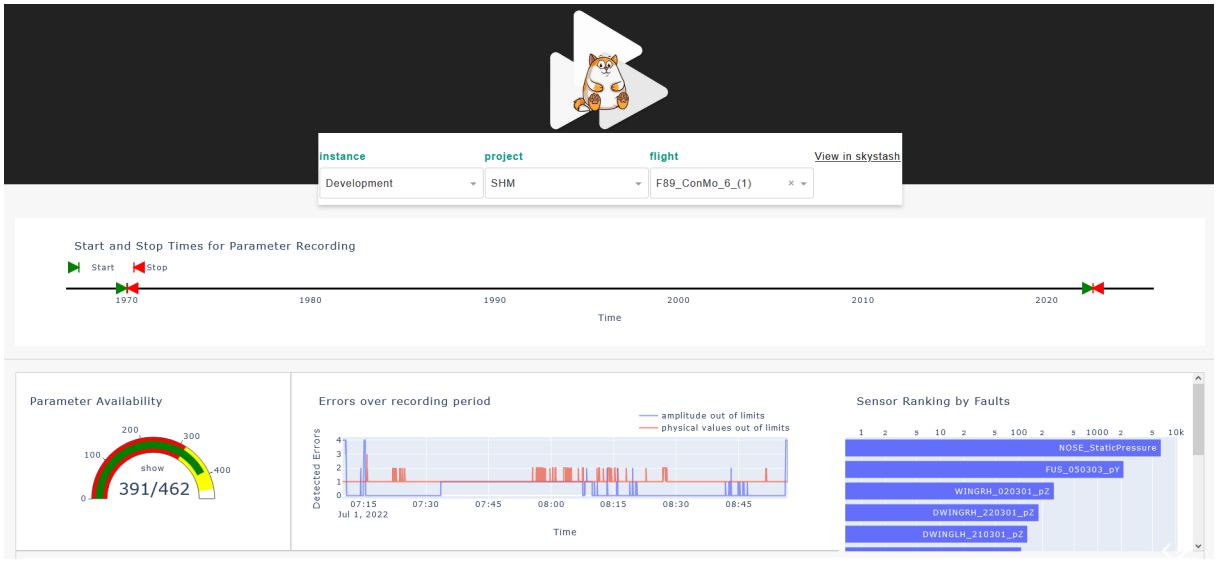


Figure 4.10: The Stashboard, a webbased analytic tool to quickly analyze data quality of an ISTAR flight. After the logo, the Digital Twin DigECAT mascot in front of the Skystash logo it contains three distinct drop down menus for selecting a specific flight, collection and instance of the Skystash. Below, the analytic plots visualize relevant Data Quality Indicators (DQI).

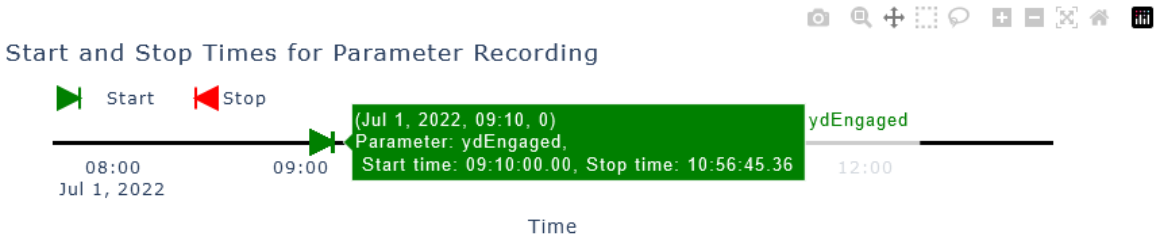
A main paradigm of this visualization is that it prioritizes qualitative over quantitative design. Meaning that it does not mention explicit details but allows to comprehend and retrace the qualitative FMEA results. It shall be a UX-centric design with the possibility of identifying outliers or oversensitive SHM configuration parameters. With Level 1 of the SHM giving a quick overview over Sensor Availability and Level 2 serving as

a simple check for any anomalies. Level 3 then serves as a more meaningful way to identify problems in the datasets by showing parameter relations and identifying deeper lying faults.

4.4.1 Timeline

Often, sensors may deliver erroneous or faulty values which get discarded along the workflow. These values are then plainly missing from the dataset. The Timeline attempts to solve this problem by generating a central overview over start and stop times of each sensor's measurement period. Complementing the timeline with the sampling rate check in Level 2 serves as a method to reliably detect missing sensor values. The Timeline's implementation is shown in Figure 4.11.

(a) The sensor timeline



(b) Interactive tooltip when hovering over a marker

Figure 4.11: The Sensor Timeline allows insights into the measurement periods of sensors by plotting start and end time of measurements to quickly identify outliers. Its rich tooltip enables a clean interface with detailed data handily available when hovering over a desired data point.

The sensor primarily delivers a clean view showing markers for each sensor measurement with start markers being green and stop markers being red in Figure 4.11a. Once additional information is desired related to an outlying sensor measurement the tooltip of a marker (see Figure 4.11b) shows parameter Name as well as the parameter's precise Start and Stop Time. Allowing further investigation by using the Skystash architecture.

4.4.2 Level 1: Parameter Availability

After monitoring, a list of missing parameters is generated. To show and detect this info quickly, a speed gauge style display is chosen to display this scalar value as shown in Figure 4.12. As speedometers are inherently performant and user-friendly for displaying scalar values they are chosen for displaying the number of available parameters. The red value is representing the percentage of the standard deviation ($\sigma = 68.27\%$) and the yellow area represents double the standard deviation ($2\sigma = 95.45\%$).

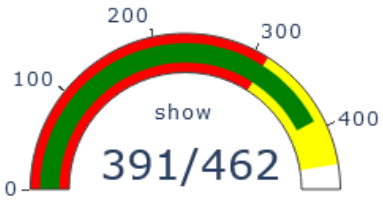
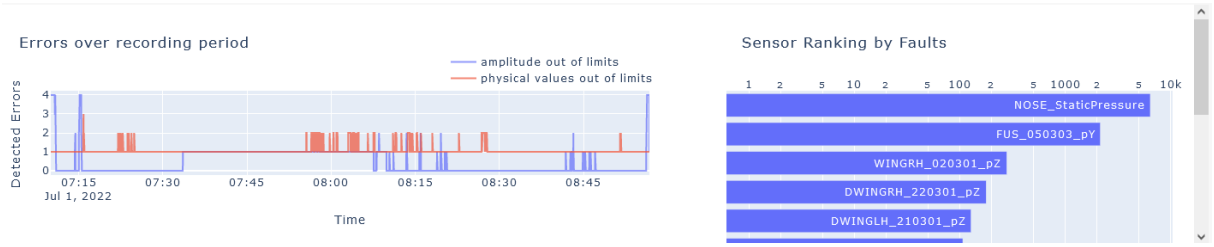


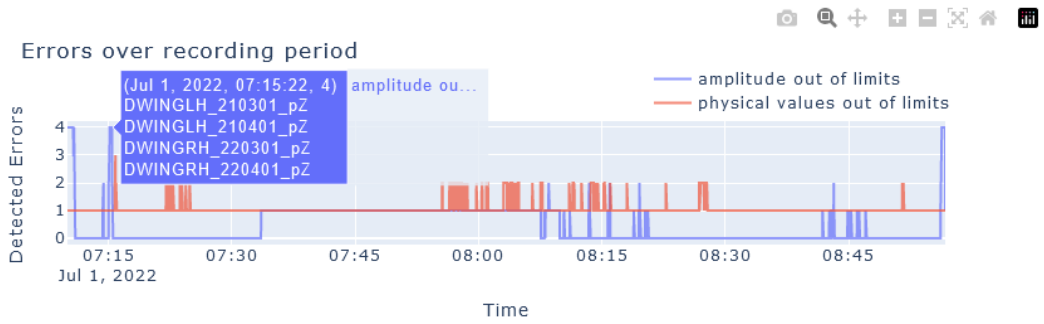
Figure 4.12: Displaying the available number of parameters in a gauge display. A fuller gauge signifies more available parameters. The levels for the red and yellow markings are set at the quality levels for 1σ and 2σ respectively.

4.4.3 Level 2: Singular Sensor Examination

To quickly get an overview of missing parameters a method to show cumulative errors over the flight is developed (see Figure 4.13).



(a) Detected cumulative Faults over time (left), sensors ranked by number of faults (right)



(b) Interactive tooltip showing anomalous sensors at timestep

Figure 4.13: The single sensor analysis in Level 2 shows cumulated faults over the measuring period and then ranks the sensors by their detected faults.

For each type of fault cumulative errors at a given timestep are added across sensors. Then the total number of errors is compounded for each sensor and displayed in a graph bar ranking. Specific points of interest on the graph can closer be examined by hovering over them as shown in Figure 4.13b. The tooltip then shows the exact time of anomaly as well as the detected errors. This visualization has the advantages that it not only shows when errors occured but also gives a quick analysis for the sensors which had anomalous behavior.

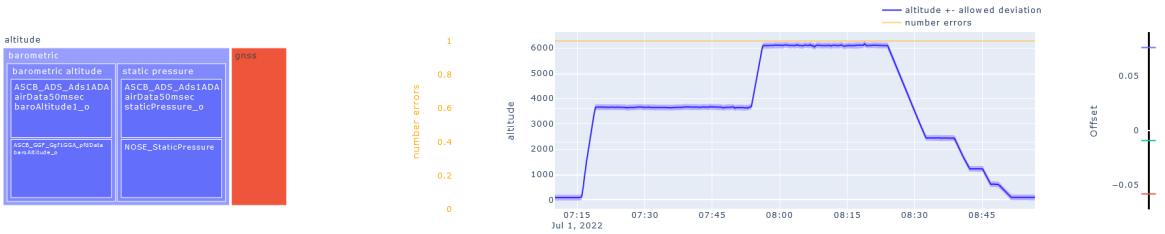


Figure 4.14: The developed Level 3 SHM visualization. On the left the hierarchical correlation of parameters the aircraft altitude is displayed. In the middle n_{errors} over time is displayed in yellow as well as the fused state value in blue with the allowed range in a lighter shade of blue. On the right the mean of the sensors is compared to quickly identify strong outliers.

4.4.4 Level 3: Examining Sensor Interactions

The Sensor Interaction and correlation is calculated for each sensor in Level 3. To then grasp the checked correlations and results at a glance a tree map is employed to display the hierarchical relations that is provided by the SHM-Level 3 configuration file (left in Figure 4.14).

Then, the fused state value associated with its upper and lower limits is displayed in the following window. Overlaid, the amount of detected sensor errors is shown, similar to Figure 4.13a. Finally, the average of the residual for each parameter is shown to quantify any occurring bias in the right window.

4.4.5 Implementation

To feed these displays a data pipeline is needed. The generated JSON-report file can be used to fulfill this job by feeding the flight SHM data to the Timeline, Level 1 and Level 2. For Level 3 the parameter SHM-metadata is needed as shown in Figure 4.15.

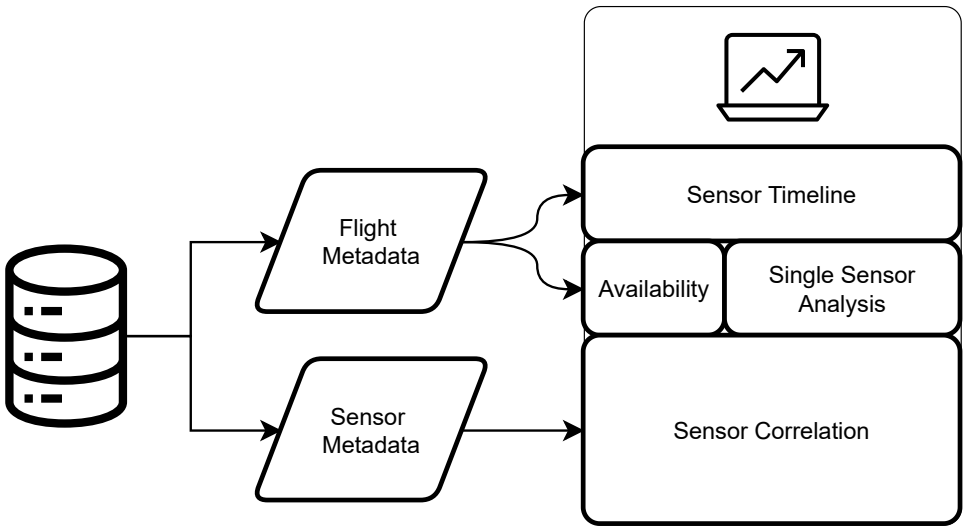


Figure 4.15: The stashboard's flow of data as introduced in Figure 2.9 and 4.1 is visualized by reading the JSON-metadata of the Skystash at flight and parameter-level (see Figure 4.3).

4.5 Summary

This chapter showed the methods used and the considerations made for the implementation of this SHM. Various methods and approaches were implemented to check for sensor faults. Next to the straight implementation, focus was put into developing future-proof interfaces to ease further implementation of coming methods of fault detection. This applies for Levels 1, 2 as well as 3 which in this work was only minimally implemented due to time constraints.

5 Results and Discussion

This chapter presents the findings of this work as well as the work that has been done and the challenges incurred and then discusses them in a wider context. The work regarding the configuration will first be evaluated upon its value and its design principles.

5.1 Metadata system

The Metadata plays a focal role within this work. It does not only just represent the data but is also vital for honoring the FAIR principles and opening this work up for further use. For completeness the parsing and upload will also be mentioned in this section since it all becomes part of the bigger picture. It also works up its way to achieve low coupling of software parts while maintaining high coherence of the metasystem behind, standardizing its single parts. Advertently achieving structured design as proposed by industry norms such as Stevens, Myers, and Constantine [STEV74]. For the data provision, a common ground has been found for the upload so that alterations to further processing parts of the cycle don't necessitate a full reupload of the original data. This common ground asserts that original data is uploaded once with the downside that the DAQ configuration may not be fully interpretable at this stage due not being self-describing metadata.

The following step of configuration solves this problem then by collecting and merging available metadata into a common scheme. This step works upon the DAQ metadata that is already uploaded to the Skystash and then fuses it with other metadata sources that are locally available to generate a holistically valid metadata set. Notable is that this step will certainly be part of rather constant modification due to SHM configurations that are also provided within this step. Also the metadata schema is far from complete and will be in need of constant expansion to compensate the complex dimensions of Flight Test Data as well as relevant test configuration data that also may be taken into consideration within this step 2.19. Since this step is far less computationally intensive than the parsing process this is considered as an acceptable tradeoff within this process.

5.1.1 Results

It has then be shown to be effective to separate metadata into the unmodified DAQ-data and a set of general metadata that provides all other necessary information. Be it for providing helpful details or vital SHM metadata. Fusing both sets of metadata then into a common set firstly elevates reliability of metadata by taking in account the DAQ-metadata. It then also guarantees interoperability and reusability by transforming metadata into a standardized metadata format that allows to be exchanged for the most effective format available since the transformation process takes marginal effort once the data is digitally available and compounded. It has then also been adhered to the process standard by not inventing another new standard but respecting existing standards to use an excel sheet for configuration purposes. Using this approach and keeping information centralized, a later migration of data will be much facilitated.

Regarding the algorithmic part, the configuration has been created within a JSON-format. Reducing coupling

while allowing a more dynamic expansion of the codebase. This makes it easier to implement changes and reduces the points of entry for making changes for creating new routines

5.1.2 Discussion

Adherence to the FAIR principles should be the main focus. Especially when handling metadata. The developed metadata handling can assert to have guaranteed the findability, interoperability and reusability within the metadata part of this work since the metadata clearly describes the data at hand (Findability), it then also clearly describes what attributes the data has and with which attributes the SHM algorithms will be run (Interoperability, Reusability) to then allow future users to reproduce and comprehend the full SHM process. In addition, the low coupling and high cohesion are central paradigms of this work.

The current form of the general configuration however poses some serious challenges and issues. It has the upside of being centralized but the downside of being in a not version-controlled environment. Not allowing backtracking and making search for faults very difficult. It also hinders development since experimental changes can only be made by copying to a local version that is then not version-controlled. It also does not allow dynamic changes to the metadata and in general does not adhere to the presented FAIR principles. However, once this issue has been fixed this work may bring more than marginal improvements to the world of data quality, allowing small changes of settings in the general configuration to trigger the SHM toolchain, updating data quality indices as you go. This metadata-workflow is also not limited to the ISTAR data and may be applied to other aircraft or even machines or systems as well since the area of sensors is a constantly growing field. Even considering that in the IoT and IoP [PENN19] the actual sensor values are hardly ever checked. Since this is a growing topic this work attempts to lay a solid foundation upon which future endeavors may build upon and flourish.

5.2 Algorithms

The core component of this work are the algorithms to actually detect the errors. These Algorithms have been implemented to catch the major recurring errors that are known to be within the datasets. Although most often errors occur in Level 1 and become critical due to oversight, other error types are also important to consider. The sheer amount of ISTAR sensors voids any ability to maintain an overview over the entire system.

For trial and erroring and embarking on the SHM journey, the experimentally installed sensors in the ISTAR serve as a significantly better benchmark for errors since they are less reliable than the inherent aircraft sensors and thus also produce more interesting errors. Four errors from accelerometers and strain gauges are shown in Figure 5.1. Ranging from interrupted transmission to odd offsets and plain overflow values.

5.2.1 Results

In the following, the outcomes of algorithms of Levels 1-3 are presented and henceforth discussed. The validation errors shown in Figure 5.1 are mostly solved since some strange errors make automatic detection difficult. Beginning with Level 1, all parameters that have not made it to the Skystash but have been expected

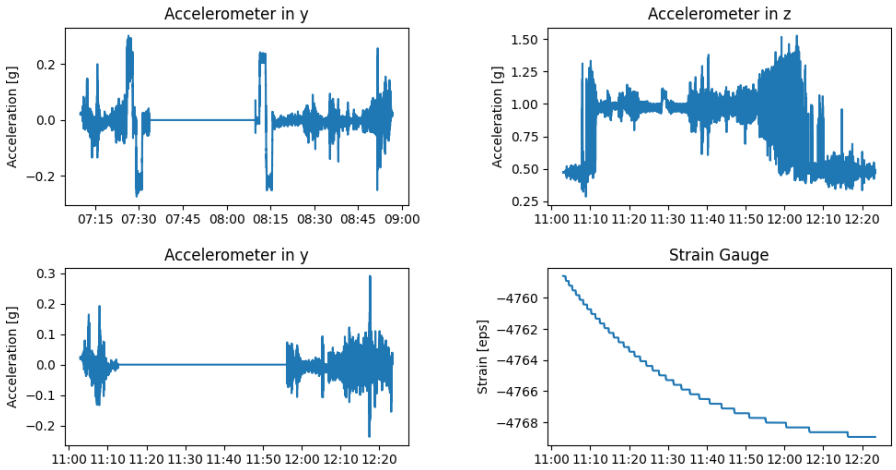


Figure 5.1: The exemplary errors upon which the SHM algorithms were trained are shown above. Three accelerometers in different flights were showing faulty behavior. The plots on the left show a sudden interruption of sensor data. While the plot on the top right shows an unusual offset of the accelerometer since the parameter value should average at one. On the bottom right the strain gauge sensor was expected to produce an output similar to the other sensors and contain some levels of noise. It is, however, at the hard limit.

to are filtered out. Acknowledging that Level 1 merely builds the first defense against errors this is a good start for the SHM, catching all errors that are plainly not available.

In Level 2 the most recurring faults are caught. It evaluates single sensor timeseries and possibly transforms them to then check against the limits defined in the configuration. In Figure 5.2 the basic spectrogram for the top left accelerometer in Figure 5.1 is given.

Based upon this spectrographic analysis outliers can be identified whose movement is either too high or too low. Using this spectrogram an averaged normed amplitude value can be calculated that represents the sensor movement. In Figure 5.3 as well as 5.4 the erroneous behaviors are clearly identified by falling outside the amplitude boundaries that have previously been defined in the general configuration. Additional faults such as the strain gauge forming the bottom right value in Figure 5.1 can quickly be identified since a regular value of such a sensor generally falls into the limits of $[-1000, +1000]$. This becomes obvious in Figure 5.5 when comparing to the limits in red.

Detecting smaller, constant offsets that are within limits such as the accelerometer in z direction in the top right plot in Figure 5.1 does not get filtered by Level 2 algorithms as shown in Figure 5.6.

To detect an offset in z direction it is more sensible to include other correlating z sensors or the rigid body model of the aircraft. Determining in acceleration in z direction within such a model should allow correlation of flight angles as well as other accelerometers in the Inertial Measurement Units (IMUs) as well as relational angles indicating turning of the aircraft and hence more acceleration in z-direction.

For now, however, only the altitude with the specialization of the barometric altitudes has been implemented. In Level 3 an automatic limit gets calculated based on the fused signal. Here, the fused signal gets detrended

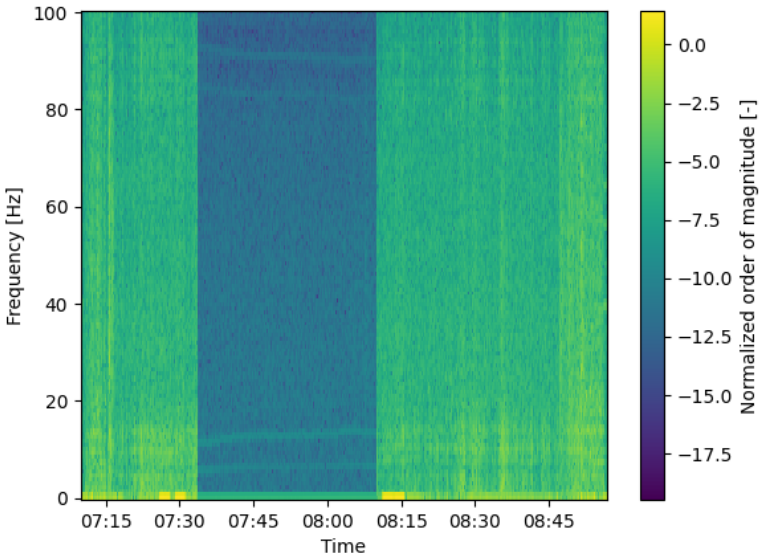


Figure 5.2: Spectrogram analysis of the erroneous accelerometer in Figure 5.1's top left. In this spectral analysis the color shows the amplitude of the frequency that is normalized to 1. The error is then quantifiable since hard limits can be defined for the signal's amplitude.

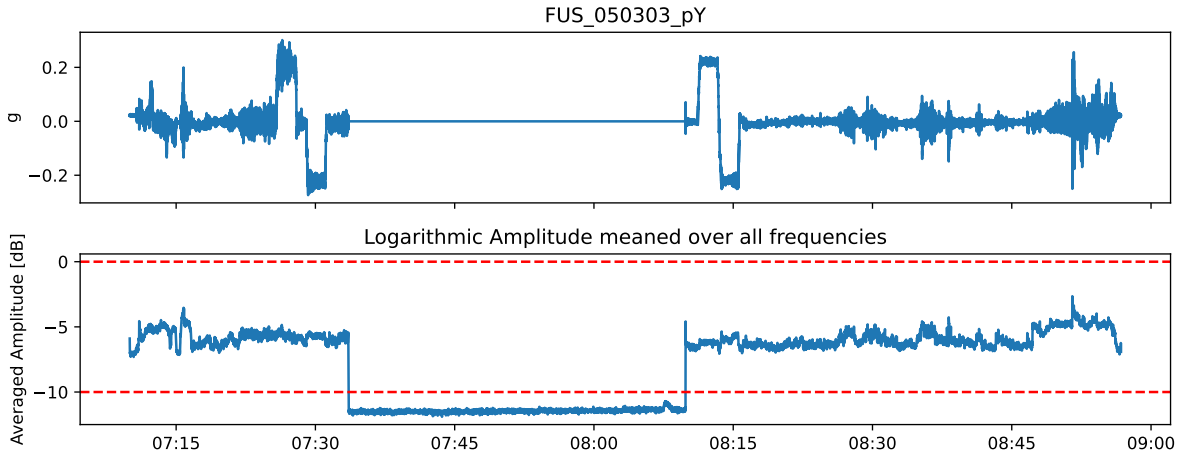


Figure 5.3: STFT Analysis for a faulty acceleration sensor (top left in Figure 5.1). Since the value could not be filtered by simply checking the limits a new method has been devised using the STFT. As the averaged amplitude exceeds the red limits as shown in the bottom figure the then detected anomalies can get recorded for the SHM.

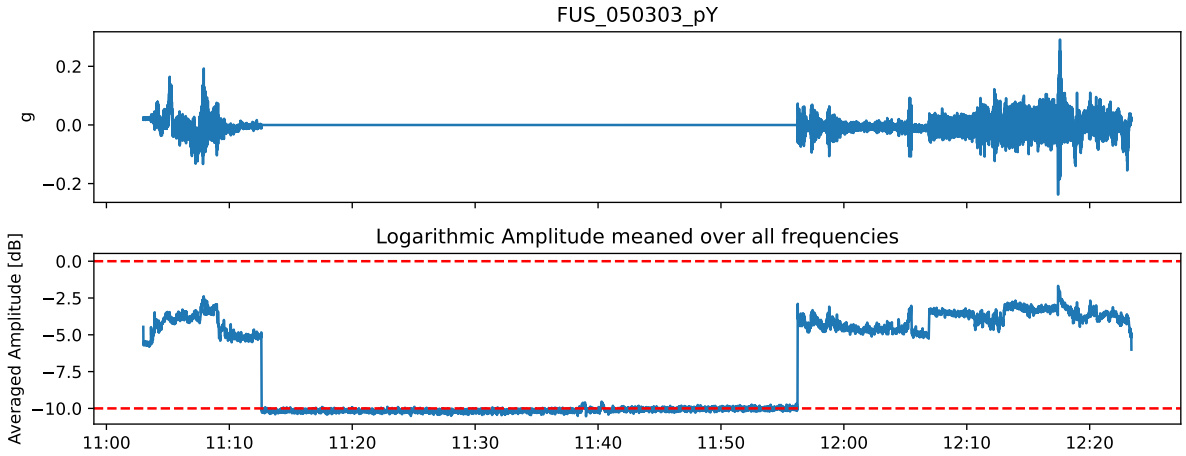


Figure 5.4: For the second parameter at the bottom left in Figure 5.1 the anomalous values can get detectd as well, proving the STFT approach to be successful on this limited validation set of sensor malfunctions.

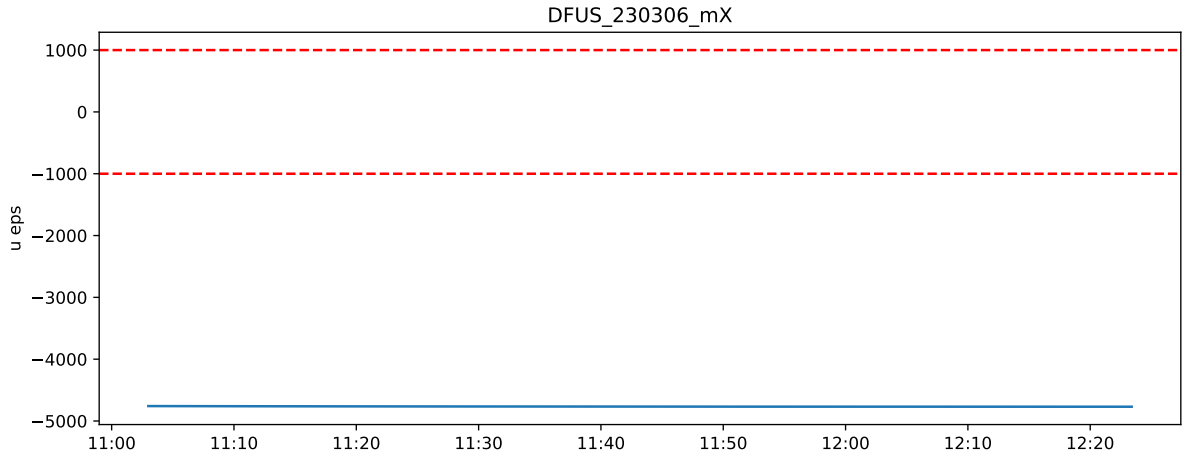


Figure 5.5: The malfunction of the strain gauge (bottom right in Figure 5.1) is shown. Since the average operating limits lie between -500 and +500 the conservatively chosen limit detects all erroneous values.

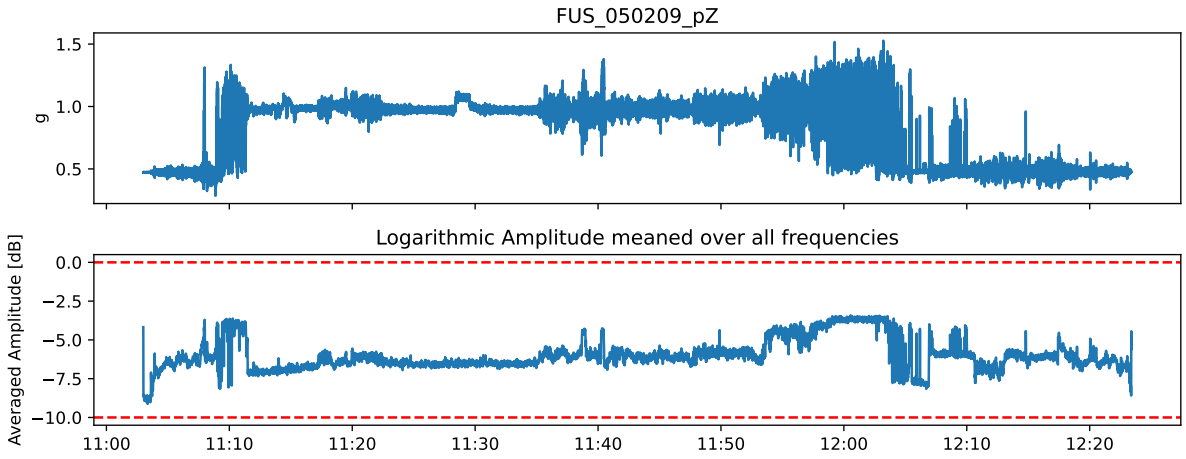


Figure 5.6: The analysis of the offset vertical accelerometer shows that this offset is not very easy to detect without any additional sensor-proprietary information. However, a Level 3 approach in further development may be able to detect such an offset by determining the fused rigid body acceleration of the ISTAR in z-direction and detecting that the acceleration values in this figure obviously strongly differ from other acceleration sensor in z-direction.

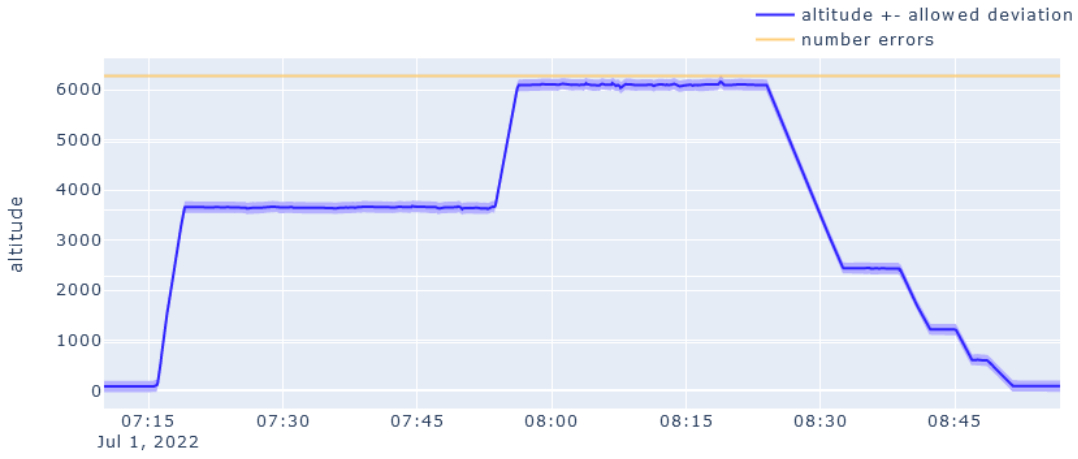


Figure 5.7: The Level 3 calculation for barometric altitudes with a the tolerance area shown as the transparent blue area are displayed. Since the allowed deviation is calculated automatically it is quite generous with about 100m. This is however far enough to determine that the static pressure measured by the nose-boom is out of bounds for the entire time of this recording session.

to remove any trends and the standard deviation of the signal gets used to measure expected deviations. The fused signal combined with the expected deviation is shown in Stashboard Figure 5.7.

Additionally, Level 3 serves as an additional layer of safety against errors. It also is possible to detect errors with a higher precision compared to the Level 2 algorithms.

Choosing check parameters

Automatically setting boundaries for Level 2 has proven to difficult due to the sheer difference of sensor behavior available. Hence, the automatically set limits for Level 2 were abandoned and replaced by hard limits that needed to be set in the general configuration. However, automatically determined boundaries were still employed and shown to be working for Level 3. Using the detrended signal has proven to deliver sensible limits as shown in the Stashboard in Figure 4.14.

However, setting limits for Level 2 still requires some trial and error to reliably work for sensors with the validation loop being driven by previously detected error types. It shows that while conservatively chosen ranges do not indicate many occurrences, tighter limits enable more detailed monitoring. Choosing a range still need some tuning since the limits at Figure 5.3 seem safe but for Figure 5.4 seem very close to detection. Thus, still needing further investigation to safely predict errors. Possibly implementing other type of sensor movement analyzes to deliver results that are more reliably and more tailored to various sensor.

Another interpretation of the sensitivity in analysis methods lies in too tight tolerances. These may detect every small aberration but generate too many occurrences when actual malfunctions are of interest and not every single perturbation. Due to limited time this algorithm could not be examined on its performance for other datasets which remains open for future work.

5.2.2 Discussion

The results from the algorithms speak for themselves. Although the entire configuration set is not complete and the system is in prototype stadium good results have been achieved. Most of the required errors and malfunctions shown in Figure 5.1 have been identified by the SHM developed in this work and even if these algorithms may not suffice more complex errors, the overall architecture makes integration of new algorithms for quick deployment easy. The results in Level 1 are primarily satisfying the application's needs. However, additional interfaces could be implemented displaying sensor subsets in a hierarchical view similar to Level 3 Stashboard implementation (see Figure 4.14). This could facilitate interpretation which subset of experiments is currently performed. E.g. whether the Nose boom of the ISTAR is presently used based on the given data and metadata.

Level 2 algorithms worked well on the posed problems and also detected complex errors such as the accelerometer faults that resulted in a very little amount of noise. Using Level 2 approaches enabled detection of most known errors, providing a substantial foundation for the proceeding work on ISTAR data. Implementing new functions is realized by using pandas series and dataframes for function in- and outputs and modifying the SHM JSON-configuration. The FMEA developed also paves the way for further advancements such as a

real-time implementation that may be used during experiments to detect possible sensor malfunctions based on processes that are computationally frugal and are able to run in real-time. Parity Equations enable correlation of parameter behavior even though they present a prototypical beginning for FMEA implementations in the SHM. Neglecting the limitations of Parity Equations they still provide value and truth for correlation making them indispensable in their function as white-box models. The Parity Equations developed in this work still has potential. It may be developed further to include a rigid body model of the aircraft, correlating various axes and states to finally achieve a minimal model error. Applications in other fields might also find this approach useful. Since the configuration has the only constraint of being hierarchical other physical relations such as in industrial applications can also be represented once these physical relations are mapped to python functions. Considering further possibilities for Level 3 FMEA various approaches could be assessed and implemented. These would be methods such as Kalman Filtering [LIE13], PCA [ISER06] and other analytical approaches [FREE13; PERH10]. Exceeding the conventional FMEAs presented in Figure 2.15, novel methods ranging from model driven designs such as the Pseudo Transfer Function approach [ALJA15] up to state of the art data-driven models for image processing such as GANomaly [AKCA18] could be implemented. These could be modified for use in research data using latent spaces to identify principal state parameters as well as identifying possibly new hidden correlations.

Within the algorithmic step it has been endeavored to comply with the Interoperable and Reusable criteria of the FAIR principles. Implementing this by using JSON-configurations in standardized formats when possible and decoupling the actual algorithmic from the data communication by using pandas as the interface for the python functions.

5.3 Visualization

An interactive frontend for the data quality report has been developed to enable accessibility of the SHM. Visualizing the data quality for an entire flight dataset at a single glance. Searchable metadata tags enable a dynamic search within the Skystash database and due to its web based architecture it can be deployed as a webservice to facilitate usage.

5.3.1 Results

As mentioned in 4.4, the Stashboard allows rapid SHM prototyping and quickly generates an overview over ISTAR flights' data quality, swiftly summarizing the large datasets. It interfaces with the Skystash over its python Application Programming Interface (API) and using the JSON-report data it is an extension module to provide a Human-Machine Interface (HMI) for the SHM.

The Timeline graph serves as a starting point to detect sensor failures that are associated with sensors outputting invalid values and thus having strange start- and end-times of their respective measurement. An advantage of this display is that it works out of the box on all Skystash flights since it accesses automatically generated Skystash data, not needing any processing and SHM.

The Level 1 implementation currently is available in the shape of a gauge, indicating the percentage of

available sensors. A helpful addition for later work could be the aforementioned visualization of sensor groups in hierarchical overview. Perhaps color mapping single sensors and sensor groups to their availability or other relevant properties.

For Level 2, any algorithms may be dynamically added to the existing configurations in the data processing step. Integrating this works without troubles since the JSON-report allows dynamic extensions on Level 2 attributes. In Level 3, it has been achieved to compare similar sensors that represent the same state variables by representing sensor relations within a tree-structure, the fused signal of all sensors with its automatically detected limit and overall sensor offsets to detect any strong biases of sensors.

5.3.2 Discussion

This dashboard implements a quick overview and architecturally attempts a low coupling high coherency to the Skystash and the other parts of the SHM-toolchain, having the Skystash API serve as the only interface needed to operate the Stashboard. Regarding FAIR guidelines, this step fulfills the Interoperability step, allowing any flight with a SHM-report to be loaded and displayed, lowering the barrier of entry for entering the world of data quality. Findability of data is then additionally guaranteed by providing a tool to check any dataset for its inherent data quality, facilitating data discovery by contributing a new method to clearly interpret data quality. For future expanses, new data quality visualizations and overall displays allow to be implemented in the dashboard structure. Also necessary to maintain overview would be aircraft landing page superset of the current dashboard. This method then may give a better overview over the entire aircraft lifespan and the reliability of its employed sensors and their failure rates. Additional implementations may add various analytic tools for the Flight Test Instrumentation Group to further analyze data such as the TerraWarn-Program, currently being developed at the FTI-group, analyzing flight performances for internal debriefing purposes. Other methods could be a Flight Data Player that allowed to replay the flight at any moment, generating data-quality analyzes for that specific point in time. Overall, this work provides a fundamental base for further Skystash analytic tooling in the Digital Twin project, honoring the FAIR guidelines.

5.4 Integration of the application

Summarizing the development, the integration into the Skystash ecosystem builds the foundation for a novel data space for Flight Test Data. Following Figure 4.1, all tools are dependent on the Skystash and serve as extensions for the existing architecture, expanding it to elevate data interpretability which is vital in the jumble of Research Data available. The programming standard of low coupling but high cohesion [STEV74] leverages the Skystash API and allows for reduction of technical debt in the future, making maintenance and expansions on single components less complex, reducing interactions. The overall evaluation of FAIR principles is shown in Figure 5.8. Starting from the upload to the Skystash. The genuine DAQ data and metadata is preserved. Metadata Enrichment then decodes the DAQ configuration and assigns extended information to the DAQ metadata and enhancing the knowledge further by also adding SHM-configurations. Within the following data processing step, all metadata for calculations is already available and the processing step can run without any interaction and also allowing backtracking based upon the enriched metadata that provides the data processing

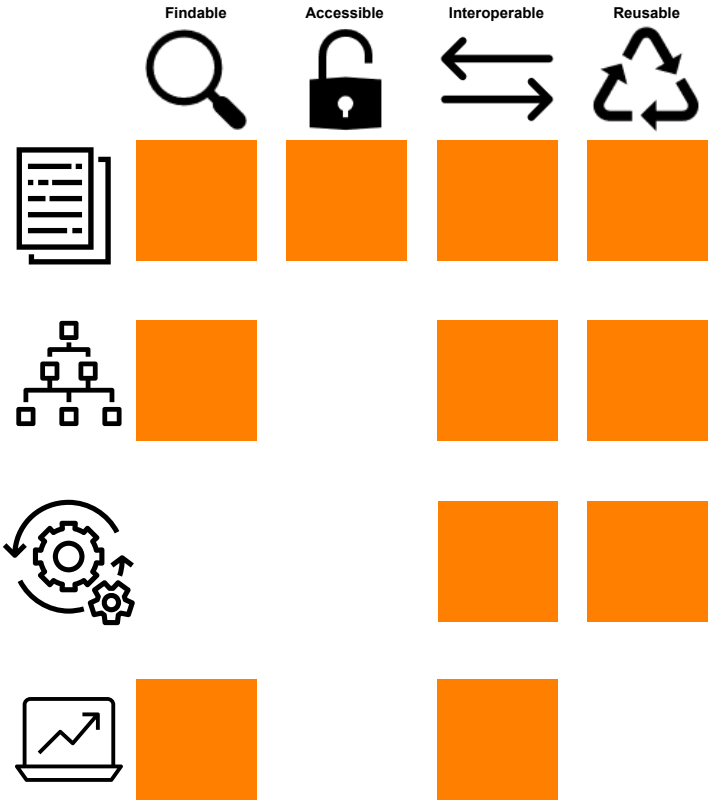


Figure 5.8: Correlating parts of this work by rating their adherence to the FAIR guiding principles. Shown are the icons describing the singular decoupled packages introduced in Figure 2.9 and 4.1 that work by interacting with the Skystash architecture (see Chapter 2.4.2)

configuration. Finally, the report is uploaded to the Skystash. It then serves as an indicator for the data quality which can then get interpreted by the data visualization tool which allows quick interpretation of the generated report.

5.5 Summary

The past chapter has shown the outcomes of this work and discussed tradeoffs as well as successes followed by opportunities for expansion which could be realized in future work. This work delivers a prototype for a Sensor Health Monitoring Infrastructure. It establishes the SHM-process as a holistic toolchain of singular methods to enable future expansion by design.

6 Conclusion

Development of an application to detect faults in Flight Test Data using conventional, non-ML methods is executed in this work. Firstly, the problem is segmented into three distinct sub-domains with the first being an availability check of sensor data by checking against the flight configuration and detecting whether sensors are available. The second step consists of checking single timeseries with algorithms that do not include other sensor data. Thirdly, sensor data is checked against each other to take into account redundant sensor data as well as physically correlating sensors. After introducing the general algorithms to detect faults, some attention is given to the software development side and the overall architecture in order to develop a design that fits into the Digital Twin project and interfaces cleanly with the Skystash while also honoring FAIR guiding princples. Within this part including configuration files and metadata management, a method is developed to separate the actually generated DAQ-data from additional, general metadata. This method will then allow to allocate various types of metadata as well as SHM-configuration into a single, standardized JSON format. This standardized shape of metadata can then also be used to configure the data processing step, making the kind of checks and algorithms retraceable. This essentially decouples the single steps of the entire process into four steps:

1. Upload of raw data and metadata
2. Metadata Merge
3. Data Processing and Fault Detection
4. Fault Mode Analysis

Using these four steps and having put an emphasis on the general architecture of SHM this work still achieves to catch many previously unknown faults and occurrences of interest while still employing relatively simple algorithms.

This approach and general implementation of novel architecture " cracks open a piñata filled with opportunity ".[ZACH23] Enabling algorithms for FMEA to be quickly deployed and evaluated in contrast to previous work-processes in which data accessibility has been a huge issue. At this point, the barrier of entry for other researchers is greatly lowered, facilitating implementation of novel FMEA algorithm and testing it on this new testbed to generate data quality indices while also being able to have access to test cases and other algorithms to compare and validate to. Since especially research data is highly customized this can be a possible way to deal with complex transformation functions as well as complex sensor/parameter correlations to quicker acquire an overview. Particularly the rapidly growing field of IoT may benefit from such an application that allows integration into a data space such as the Skystash to automatically evaluate sensors, infrastructures and FMEAs. Quickly generating value without having to implement novel infrastructure. Additionally, it lays the groundwork for a metadata-driven system monitoring which has the potential to be expanded to any system generating time-series sensordata.

References

[ADMI08] F. A. Administration. *Federal Radionavigation Plan*. DOT-VNTSC-RITA-08-02/DoD-4650.5. 2008.

[AKCA18] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon. "GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training". In: (2018). Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.1805.06725.

[ALJA15] K. F. Aljanaideh and D. S. Bernstein. "Aircraft Sensor Health Monitoring Based on Transmissibility Operators". en. In: *Journal of Guidance, Control, and Dynamics* 38.8 (Aug. 2015), pp. 1492–1495. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.G001125.

[ARTS22] E. Arts, M. Bäßler, S. Haufe, A. Kamtsiuris, H. Meyer, C. Pätzold, R. Schültzky, and M. Tchorzewski. "DIGITAL TWIN FOR RESEARCH AIRCRAFT". en. In: (2022).

[BADE20] S. Bader et al. *Details of the Asset Administration Shell : Part 1 - The exchange of information between partners in the value chain of Industrie 4.0; Version 2.0.1*. Tech. rep. Berlin: Federal Ministry for Economic Affairs and Energy (BMWi), 2020, 423 Seiten : Illustrationen, Diagramme.

[BARC11] K. A. Barchard and L. A. Pace. "Preventing human error: The impact of data entry methods on data accuracy and statistical results". en. In: *Computers in Human Behavior* 27.5 (Sept. 2011), pp. 1834–1839. ISSN: 07475632. DOI: 10.1016/j.chb.2011.04.004.

[BODE21] M. Bodenbenner, M. P. Sanders, B. Montavon, and R. H. Schmitt. "Domain-Specific Language for Sensors in the Internet of Production". en. In: *Production at the leading edge of technology*. Ed. by B.-A. Behrens, A. Brosius, W. Hintze, S. Ihlenfeldt, and J. P. Wulfsberg. Series Title: Lecture Notes in Production Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg, 2021, pp. 448–456. ISBN: 978-3-662-62137-0 978-3-662-62138-7. DOI: 10.1007/978-3-662-62138-7_45.

[BODE22] M. Bodenbenner, B. Montavon, and R. H. Schmitt. "Model-driven development of interoperable communication interfaces for FAIR sensor services". en. In: *Measurement: Sensors* 24 (Dec. 2022), p. 100442. ISSN: 26659174. DOI: 10.1016/j.measen.2022.100442.

[BOEH86] B. Boehm. "A spiral model of software development and enhancement". en. In: *ACM SIGSOFT Software Engineering Notes* 11.4 (Aug. 1986), pp. 14–24. ISSN: 0163-5948. DOI: 10.1145/12944.12948.

[BROC11] R. Brockhaus, W. Alles, and R. Luckner. *Flugregelung*. de. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN: 978-3-642-01442-0 978-3-642-01443-7. DOI: 10.1007/978-3-642-01443-7.

[DIN77] DIN. *Fault tree analysis; method and symbols*. DIN 25424. 1977.

[DIN95] DIN. *Fundamentals of metrology — Part 1: basic terminology*. DIN 1319-1. Jan. 1995.

[DLR18] DLR. *DLR-fleet*. 2018.

[FLEM22] F. O. Flemisch, M. Preutenborbeck, M. Baltzer, J. Wasser, C. Kehl, R. Grünwald, H.-M. Pastuszka, and A. Dahlmann. "Human Systems Exploration for Ideation and Innovation in Potentially Disruptive Defense and Security Systems". en. In: *Disruption, Ideation and Innovation for Defence and Security*. Ed. by G. Adlakha-Hutcheon and A. Masys. Series Title: Advanced Sciences and Technologies for Security Applications. Cham: Springer International Publishing, 2022, pp. 79–117. ISBN: 978-3-031-06635-1 978-3-031-06636-8. DOI: 10.1007/978-3-031-06636-8_5.

[FREE13] P. Freeman, P. Seiler, and G. J. Balas. "Air data system fault modeling and detection". en. In: *Control Engineering Practice* 21.10 (Oct. 2013), pp. 1290–1301. ISSN: 09670661. DOI: 10.1016/j.conengprac.2013.05.007.

[HAND17] A. Handl and T. Kuhlenkasper. *Multivariate Analysemethoden*. de. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017. ISBN: 978-3-662-54753-3 978-3-662-54754-0. DOI: 10.1007/978-3-662-54754-0.

[HART22] P. Hartmann and S. Seitz. *Flugführung: Navigation-Sensordatenfusion*. 2022.

[HEND08] R. Hendriks, J. Jensen, and R. Heusdens. "Noise Tracking Using DFT Domain Subspace Decompositions". en. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.3 (Mar. 2008), pp. 541–553. ISSN: 1558-7916. DOI: 10.1109/TASL.2007.914977.

[HODS18] S. Hodson, S. Jones, S. Collins, F. Genova, N. Harrower, D. Mietchen, R. Petrauskaité, and P. Wittenburg. "FAIR Data Action Plan: Interim recommendations and actions from the European Commission Expert Group on FAIR data". en. In: (June 2018). Publisher: Zenodo. DOI: 10.5281/ZENODO.1285290.

[ISER06] R. Isermann. *Fault-Diagnosis Systems*. en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN: 978-3-540-24112-6 978-3-540-30368-8. DOI: 10.1007/3-540-30368-5.

[ISER11] R. Isermann. *Fault-diagnosis applications: model-based condition monitoring: actuators, drives, machinery, plants, sensors, and fault-tolerant systems*. eng. Berlin Heidelberg: Springer, 2011. ISBN: 978-3-642-12766-3.

[ISER97] R. Isermann and P. Ball. *TRENDS IN THE APPLICATION OF MODEL-BASED FAULT DETECTION AND DIAGNOSIS OF TECHNICAL PROCESSES*. en. Tech. rep. 1997.

[ISO15] ISO. *Data quality Part 8: Information and data quality: Concepts and measuring*. ISO 8000-8:2015. Nov. 2015.

[ISO75] ISO. *Standard Atmosphere*. ISO 2533-1975. 1975.

[ISO97] ISO. *Accuracy (trueness and precision) of measurement methods and results - Part 1 : Generai principles and definitions*. ISO 5725-1 :1994. 1997.

[KHAL22a] A. Khalil, M. Al Janaideh, K. F. Aljanaideh, and D. Kundur. "Transmissibility-Based Health Monitoring of the Future Connected Autonomous Vehicles Networks". en. In: *IEEE Transactions on Vehicular Technology* 71.4 (Apr. 2022), pp. 3633–3647. ISSN: 0018-9545, 1939-9359. DOI: 10.1109/TVT.2022.3151326.

[KHAL22b] A. Khalil, K. F. Aljanaideh, and M. Al Janaideh. "Transmissibility-based Fault Detection in Systems with Unknown Time-Varying Parameters". en. In: *2022 American Control Conference (ACC)*. Atlanta, GA, USA: IEEE, June 2022, pp. 1947–1951. ISBN: 978-1-66545-196-3. DOI: 10.23919/ACC53348.2022.9867203.

[KUTS75] F. v. Kutschera. *Sprachphilosophie*. 2., völlig neu bearb. u. erw. Aufl. Uni-Taschenbücher ; 80. München: Fink, 1975. ISBN: 978-3-7705-1182-2.

[LIE13] F. A. P. Lie and D. Gebre-Egziabher. "Synthetic Air Data System". en. In: *Journal of Aircraft* 50.4 (July 2013), pp. 1234–1249. ISSN: 0021-8669, 1533-3868. DOI: 10.2514/1.C032177.

[MEYE20] H. Meyer, J. Zimdahl, A. Kamtsiuris, R. Meissner, F. Raddatz, S. Haufe, and M. Bäßler. "Development of a Digital Twin for Aviation Research". en. In: (2020). Artwork Size: 8 pages Medium: application/pdf Publisher: Deutsche Gesellschaft für Luft- und Raumfahrt - Lilienthal-Oberth e.V. Version Number: 1.0, 8 pages. DOI: 10.25967/530329.

[NEWE19] D. B. Newell and E. Tiesinga. *The international system of units (SI):: 2019 edition*. en. Tech. rep. NIST SP 330-2019. Gaithersburg, MD: National Institute of Standards and Technology, Aug. 2019, NIST SP 330–2019. DOI: 10.6028/NIST.SP.330-2019.

[PEAR01] K. Pearson. "LIII. *On lines and planes of closest fit to systems of points in space*". en. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1901), pp. 559–572. ISSN: 1941-5982, 1941-5990. DOI: 10.1080/14786440109462720.

[PENN19] J. Pennekamp et al. "Towards an Infrastructure Enabling the Internet of Production". en. In: *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*. Taipei, Taiwan: IEEE, May 2019, pp. 31–37. ISBN: 978-1-5386-8500-6. DOI: 10.1109/ICPHYS.2019.8780276.

[PERH10] M. G. Perhinschi, H. Moncayo, and J. Davis. "Integrated Framework for Artificial Immunity-Based Aircraft Failure Detection, Identification, and Evaluation". en. In: *Journal of Aircraft* 47.6 (Nov. 2010), pp. 1847–1859. ISSN: 0021-8669, 1533-3868. DOI: 10.2514/1.45718.

[SHOE87] S. Shoemaker and S. Blackburn. "Spreading the Word." In: *Noûs* 21.3 (Sept. 1987), p. 438. ISSN: 00294624. DOI: 10.2307/2215195.

[SLAT98] J. A. Slater and S. Malys. "WGS 84 — Past, Present and Future". In: *Advances in Positioning and Reference Frames*. Ed. by K.-P. Schwarz and F. K. Brunner. Vol. 118. Series Title: International Association of Geodesy Symposia. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 1–7. ISBN: 978-3-642-08425-6 978-3-662-03714-0. DOI: 10.1007/978-3-662-03714-0_1.

[SMIT99] S. W. Smith. *The scientist and engineer's guide to digital signal processing*. en. 2nd edition. OCLC: 493473234. San Diego (Calif.): California Technical Pub., 1999. ISBN: 978-0-9660176-7-0.

[STEV74] W. P. Stevens, G. J. Myers, and L. L. Constantine. "Structured design". In: *IBM Systems Journal* 13.2 (1974), pp. 115–139. ISSN: 0018-8670. DOI: 10.1147/sj.132.0115.

[SVÄR14] C. Svärd, M. Nyberg, E. Frisk, and M. Krysander. "Data-driven and adaptive statistical residual evaluation for fault detection with an automotive application". en. In: *Mechanical Systems and Signal Processing* 45.1 (Mar. 2014), pp. 170–192. ISSN: 08883270. DOI: 10.1016/j.ymssp.2013.11.002.

[TEUN17] P. J. Teunissen and O. Montenbruck, eds. *Springer Handbook of Global Navigation Satellite Systems*. en. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-42926-7 978-3-319-42928-1. DOI: 10.1007/978-3-319-42928-1.

[WILK16] M. D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". en. In: *Scientific Data* 3.1 (Mar. 2016), p. 160018. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.18.

[WOLT14] M. Wolter. "Anwendung von Pseudospektren in der Regelungstechnik". de. 2014.

[XIAO06] F. Xiao, S. Wang, and J. Zhang. "A diagnostic tool for online sensor health monitoring in air-conditioning systems". en. In: *Automation in Construction* 15.4 (July 2006), pp. 489–503. ISSN: 09265805. DOI: 10.1016/j.autcon.2005.06.001.

[ZACH23] Zacharias. *What is a positive version of the idiom "opening Pandora's Box?"* 2023.

[ZHAN08] Y. Zhang and J. Jiang. "Bibliographical review on reconfigurable fault-tolerant control systems". en. In: *Annual Reviews in Control* 32.2 (Dec. 2008), pp. 229–252. ISSN: 13675788. DOI: 10.1016/j.arcontrol.2008.03.008.