Article citation info:
GERDES M, GALAR D, SCHOLZ D. Decision trees and the effects of feature extraction parameters for robust sensor network design. Eksploatacja i Niezawodnosc – Maintenance and Reliability 2017; 19 (1): 31–42, http://dx.doi.org/10.17531/ein.2017.1.5.

Mike GERDES
Diego GALAR
Dieter SCHOLZ

DECISION TREES AND THE EFFECTS OF FEATURE EXTRACTION PARAMETERS FOR ROBUST SENSOR NETWORK DESIGN

WYKORZYSTANIE DRZEW DECYZYJNYCH ORAZ WPŁYWU PARAMETRÓW EKSTRAKCJI CECH DO PROJEKTOWANIA ODPORNYCH SIECI CZUJNIKÓW

Reliable sensors and information are required for reliable condition monitoring. Complex systems are commonly monitored by many sensors for health assessment and operation purposes. When one of the sensors fails, the current state of the system cannot be calculated in same reliable way or the information about the current state will not be complete. Condition monitoring can still be used with an incomplete state, but the results may not represent the true condition of the system. This is especially true if the failed sensor monitors an important system parameter. There are two possibilities to handle sensor failure. One is to make the monitoring more complex by enabling it to work better with incomplete data; the other is to introduce hard or software redundancy. Sensor reliability is a critical part of a system. Not all sensors can be made redundant because of space, cost or environmental constraints. Sensors delivering significant information about the system state need to be redundant, but an error of less important sensors is acceptable. This paper shows how to calculate the significance of the information that a sensor gives about a system by using signal processing and decision trees. It also shows how signal processing parameters influence the classification rate of a decision tree and, thus, the information. Decision trees are used to calculate and order the features based on the information gain of each feature. During the method validation, they are used for failure classification to show the influence of different features on the classification performance. The paper concludes by analysing the results of experiments showing how the method can classify different errors with a 75% probability and how different feature extraction options influence the information gain.

***Keywords**: decision trees, feature extraction, sensor optimization, sensor fusion, sensor selection.*

Niezawodne monitorowanie stanu wymaga niezawodności czujników i pochodzących z nich informacji. Systemy złożone są zazwyczaj monitorowane przez wiele czujników, co pozwala na ocenę stanu technicznego oraz aspektów eksploatacyjnych. Gdy jeden z czujników ulega uszkodzeniu, uniemożliwia to obliczenie bieżącego stanu systemu z dotychczasową niezawodnością lub uzyskanie kompletnych informacji o bieżącym stanie. Stan można co prawda monitorować nawet przy niekompletnych danych, ale wyniki takiego monitorowania mogą nie odpowiadać rzeczywistemu stanowi systemu. Sytuacja taka ma miejsce w szczególności, gdy uszkodzony czujnik jest odpowiedzialny za monitorowanie istotnego parametru systemu. Problem uszkodzenia czujnika można rozwiązywać na dwa sposoby. Pierwszy polega na zwiększeniu złożoności systemu, co umożliwia jego sprawniejsze działanie w sytuacji, gdy dane są niekompletne. Drugim sposobem jest wprowadzenie nadmiarowego sprzętu (hardware'u) lub oprogramowania. Niezawodność czujników stanowi krytyczny aspekt systemu. Oczywiście, ze względu na ograniczenia przestrzenne, ekonomiczne i środowiskowe nie wszystkie czujniki w systemie mogą być nadmiarowe. Redundancja powinna dotyczyć wszystkich czujników, które dostarczają istotnych informacji na temat stanu systemu, natomiast dopuszczalne są błędy mniej ważnych czujników. W niniejszej pracy pokazano jak obliczać istotność informacji o systemie dostarczanych przez poszczególne czujniki z wykorzystaniem metod przetwarzania sygnałów oraz drzew decyzyjnych. Zademonstrowano również w jaki sposób parametry przetwarzania sygnałów wpływają na poprawność klasyfikacji metodą drzewa decyzyjnego, a tym samym na poprawność dostarczanych informacji. Drzew decyzyjnych używa się do obliczania i porządkowania cech w oparciu o przyrost informacji charakteryzujący poszczególne cechy. Podczas weryfikacji zastosowanej metody, drzewa decyzyjne wykorzystano do klasyfikacji uszkodzeń celem przedstawienia wpływu różnych cech na dokładność klasyfikacji. Pracę kończy analiza wyników eksperymentów pokazujących w jaki sposób zastosowana metoda pozwala na klasyfikację różnych błędów z 75-procentowym prawdopodobieństwem oraz jak różne opcje ekstrakcji cech wpływają na przyrost informacji.

***Słowa kluczowe**: drzewa decyzyjne, ekstrakcja cech, optymalizacja czujników, fuzja czujników, dobór czujników.*

1. Introduction

This paper presents a condition monitoring system with sensor optimization capabilities to prevent unscheduled delays in the aircraft industry. Unscheduled delays cost airlines a great deal of money but can be prevented by condition monitoring [11]. The aim was to develop a simple condition monitoring system that can be understood by humans and modified by experts to incorporate knowledge that is not in the learning data set, using decision trees as the main tool. Decision trees satisfy the requirements and provide a ranking of data sources for condition monitoring.

The first section of the paper gives the motivation for developing a condition monitoring system with sensor optimization capabilities and explains the basic concepts of the proposed method. The second section explains the method in detail. Section three discusses the experiments validating it. The results of the validation experiments are given in section four. The paper concludes with a discussion of the results.

New and better monitoring approaches are required for condition monitoring, because systems are becoming more complex and more difficult to monitor [32]. Condition monitoring requires reliable sensors. To obtain enough sensing data, special attention should be given to optimizing sensor allocation to ensure system diagnosability, lower sensing cost and reduce time to diagnosis [37]. Sensors can be used to determine the system health of control systems; a failed sensor can lead to a loss of process control [18]. The information about a system is incomplete, if a sensor fails. Complex systems are often monitored by multiple sensors. An advantage of a multi sensor system is that a single failed sensor shows its effects in multiple sensors [18].This means the system condition is defined by all information from the sensors. However, the system's health status becomes uncertain when a sensor fails or sends wrong data. This could trigger incorrect maintenance, including maintenance on a part with no failure, as well as long maintenance times to find the correct fault or not noticing the fault at all.

The Safety Integrity Level (SIL) defines the probability that the system safety function for a Safety Instrumented System (SIS) can be executed. There are four SILs; level four is the level with the highest probability that an SIS can be performed. Sensor failure detection (sensor validation) is a critical part of the safety function of a system. When a failure is detected SIS is put into a safe state to avoid risk and damage to humans and machines [14, 13].

Redundancy is used to reduce the risk of model uncertainty [5]. One way to create sensor redundancy is hardware redundancy; another is analytical redundancy [5]. Analytical redundancy assumes multiple sensors deliver the same information, and, thus, a sensor fault can be compensated for. Hardware redundancy is not always possible, as it can be difficult to install multiple sensors because of physical or cost constraints [41, 22].

The proposed condition monitoring method uses a data driven model and uses machine learning methods to learn the model. Data driven modelling is a popular approach, especially as data harvesting is often cheaper than creating a physical model, offering cheap electronics, high computation power and advanced algorithms. Decision trees are used for machine learning because they create a comprehensive model, which can easily be modified and adapted. Decision trees are numerically stable, the learning is deterministic, and they are easy to test. The decision tree algorithm also sorts inputs of the model based on information gained. This latter feature is used for sensor optimization.

The novelty of this approach is that it presents a method for condition monitoring suitable for the very restricted aircraft environment. It combines decision trees with very stable and simple feature extraction methods. The method offers fast, testable and low footprint online condition monitoring for aircraft. The added sensor optimization allows the aircraft manufacturer to install redundant sensor hardware for the significant sensors, if software redundancy is not possible.

The inputs for the classifier are feature vectors (representing healthy and unhealthy states) and classifications of the vectors. The vectors for the supervised learning phase need to contain the classification of the data, because decision tree learning is supervised learning. These vectors represent the knowledge on which the classifier is based and used to classify new unknown samples.

Fig. 1. Basic Condition Monitoring Process [15]

The basic condition monitoring process is shown in Figure 1. It consists of three steps:

1. Data Acquisition: All data required for the monitoring are gathered, including data from multiple sources.
2. Data Processing: The collected data are processed and analysed. The goal is to create a meaningful collection of features for the decision making step. Operations include **signal processing** and **feature extraction**. The focus of the present research is on this step.
3. Maintenance Decision Making: The features are evaluated, and a decision is made based on this evaluation. The result can be a failure classification, a maintenance action or other relevant actions. Results are obtained by using a decision maker based on logic rules, pattern recognition, probability or some other method.

1.1. Civil Aerospace Software Development

Software development, documentation, testing, and certification in the civil aerospace industry are regulated by the DO-178B/C standard [27]. DO-187B/C defines how the software development process can be regulated to ensure safe software is written. More specifically, it defines a requirements-based development process with high and low level requirements. High level requirements concentrate on functionality, while low level requirements are often written in pseudo code or source code.

The most important step in the software development process is to define to which DAL (Design Assurance Level) the software belongs. There are five DALs; each is associated with a hazard/failure condition class defining how dangerous a software failure can be. The DALs are the following:

- **DAL A**: Catastrophic; normally with hull loss and multiple fatalities.
- **DAL B**: Hazardous; large reduction in functional capabilities and serious or fatal injury to a small number of passengers or crew.
- **DAL C**: Major; significant reduction of functional capabilities and physical distress or injuries for passengers or crew.
- **DAL D**: Minor; slight reduction in functional capabilities and physical discomfort for passengers.
- **DAL E**: No effect; no effect on operational capabilities and no inconvenience for passengers.

The software development objectives of a software developing agency are based on the DAL. DAL A requires 66 objectives, DAL B 65 objectives, DAL C 57 objectives, DAL D 28 objectives, and DAL E 0 objectives. The objectives are achieved by completing ten processes in the development of the software:

1. Software Planning Process.
2. Software Development Process.
3. Verification of Outputs of Software Requirements Process.
4. Verification of Outputs of Software Design Process.
5. Verification of Outputs of Software Coding & Integration Process.
6. Testing of Outputs of Integration Process.
7. Verification of Verification Process Results.
8. Software Configuration Management Process.
9. Software Quality Assurance Process.
10. Certification Liaison Process.

The most complex step besides coding for a software developer is testing the coded software. Based on the DAL, the testing needs to satisfy certain code coverages. For DAL D and E, for example, no code coverage is required; only the requirements need to be tested. DAL C adds statement coverage to the testing requirements. This means the tests need to address each line of code. No dead code is allowed. In addition to this, DAL B requires decision coverage; each possible path in the code must be taken. For DAL A, developers must

show that each variable for a decision in the code can influence the result (modified condition/decision coverage), as well as satisfying all other code coverages. All software testing needs to be done as black box testing. The tester cannot know the code but must work only with the compiled code, requirements and testing tools.

Robustness tests require border values of numerical values and of decisions to be tested and invalid or missing data to be identified. There can obviously be problems if algorithms use the wrong data type.

1.2. Feature Extraction

Feature extraction is the process of reducing the dimension of the initial input data to a feature set of a lower dimension containing most of the significant information of the original data [8]. Extraction is done to extract important features from noisy sensor data [19, 10] and to avoid having too many input features (especially for vibration data) in the classifier learning phase [19]. For these reasons, feature extraction is often a first and essential step for any classification [19]. Accordingly, it is part of the data processing step in the basic condition monitoring process (Figure 1).

Features are extracted from the time domain and the frequency domain (Fourier Transformation, Wavelet Transformation [10]). Basic features to extract are maximum, mean, minimum, peak, peak-too-peak interval etc. [15]. Complex feature extraction methods include principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) [39].

1.2.1. Time domain features

Time domain features can be direct features like the number of peaks, zero-crossings, mean amplitude, maximum amplitude, minimum amplitude or peak-too-peak interval [15, 23]. In addition, it is possible to analyse a signal using probabilistic methods like root mean square, variance, skewness or kurtosis to get features that represent the signal [17]. Other methods include using correlation, autocorrelation, Entropy, principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA)[39].

1.2.2. Frequency and Time-Frequency domain

The Fast Fourier Transformation (FFT) transforms a signal from the time domain into the frequency domain. FFT takes a time series and transforms it into a complex vector that represents the frequency power in frequency domain. The basis of the FFT algorithm is the discrete Fourier transformation (DFT), defined as shown in equation 1 with $xn \ldots x_{n-1}$ as complex numbers:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \ldots, N-1 \tag{1}$$

A FFT is performed in *O(N log N)* operations and can be calculated in real time due to the fact that it can be executed in parallel. It is a widely used and well established method [24, 5]. Recent researches use the discrete wavelet transformation (DWT) to represent time series in the frequency domain. The DWT represents the time series in a time-scale form [15] and is especially suited to represent non-stationary signals [19].

1.2. Decision Trees

Decision trees are a method from the area of artificial intelligence and are used for machine learning. They are often binary trees, where each node has an if-then-else function on an attribute of the sample data. The ID3 algorithm (Iterative Dichotomiser 3, published by J. Ross Quinlan in 1986, used to generate decision trees [25]) was the first algorithm to construct decision trees. ID3 had some problems and was improved. The improved version of ID3 is C4.5 [26]. It enhances the ID3 algorithm with the ability to handle both discrete and continues attributes, it can handle samples with missing attributes and supports pruning of the tree at the end of the algorithm (removing branches from the tree).

Decision trees are in the proposed method used to calculate and order the features based on the information gain of each feature. During the method validation they are used for failure classification to show the influence of different features on the classification performance.

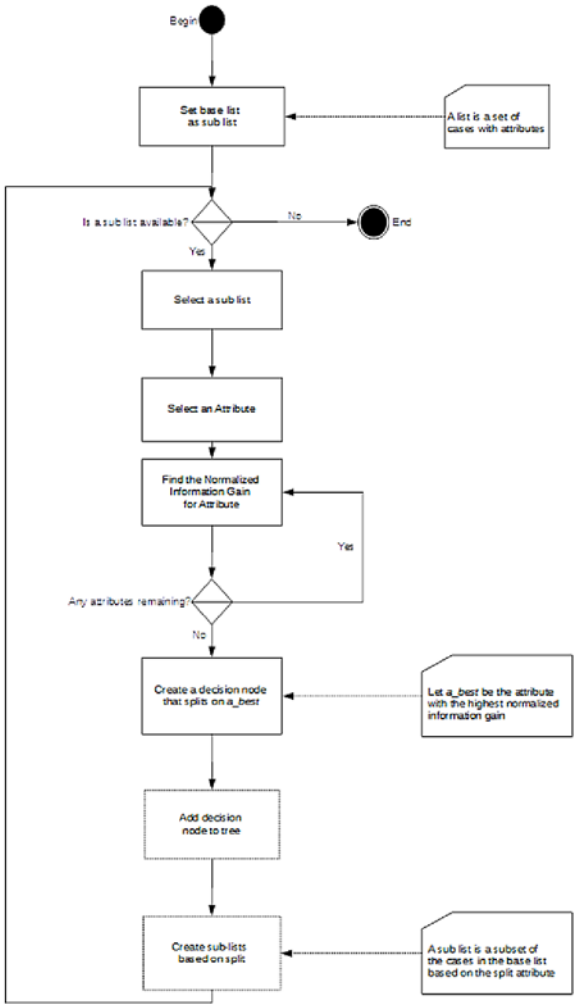

Fig. 2. Decision Tree Algorithm Flow Chart

The result of the algorithm is a binary decision tree, where the root of the tree is the attribute with the highest normalized information gain. Nodes in the following levels of the tree represent attributes with lower normalized information gain. If pure information gain is used for splitting, then classes with the most cases are favoured [26].

Information entropy is the knowledge that is contained in an answer depending on one's prior knowledge. The less is known, the more information is provided. In information theory information entropy is measured in bits. One bit of information entropy is enough to answer a yes/no question about which one has no data [29]. The information entropy is also called information and is calculated as shown below. $P(v_i)$ is the probability of the answer v_i:

$$I\left(P(v_i), \ldots, P(v_n)\right) = \sum_{i=1}^{n} - P(v_i) log_2 P(v_i) \quad (2)$$

The information gain from an attribute test is the difference between the total information entropy requirement (the amount of information entropy that was needed before the test) and the new information entropy requirement. *p* is the number of positive answers and *n* is the number of negative answers [29]:

$$Gain(X) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} \times I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right) \quad (3)$$

C4.5 uses the normalized information gain or the gain ratio. Split info is the information that is gained from choosing the attribute to split the samples:

$$Split\ Info\ (X) = - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} log_2 \left(\frac{p_i + n_i}{p+n}\right) \quad (4)$$

Gain ratio is the normalized information gain and is defined as shown in equation *5* [26]:

$$Gain\ Ratio\ (X) = \frac{Gain\ (X)}{Split\ Info\ (X)} \quad (5)$$

Pruning is the reduction of the depth of a decision tree. The tree gets better at classifying unknown samples, but might get worse at classifying the test samples. Normally pruning increases the overall classification accuracy, but too much pruning can increase the number of false classifications.

Decision trees are good for diagnostics in the context of condition monitoring. They classify data with low computation needs and the generated decision trees are highly comprehensible by humans. Another advantage of decision trees for condition monitoring is that they can be transformed into simple logical equations for each class that can be checked and modified by a human expert.

Decision trees are used to solve a large variety of problem e.g. tag speech parts [33], land cover mapping [9], text mining [1] or condition monitoring [36, 30, 31].

1.4. Basic Condition Monitoring Process Enhancements

Sensor optimization and sensor data fusion can be seen an enhancement of the basic condition monitoring process (Figure 1). Figure 3 shows how sensor optimization and sensor fusion can be embedded in the basic CM process.

Fig. 3. Enhanced condition monitoring process

Sensor optimization is the basis for the condition monitoring and is either performed before the monitoring process (sensor locations) or later to add new sensors [7] or to analyse the available sensor influences. Sensor fusion is done before the actual data processing to improve the performance of the data processing by improving the input from the sensors (removing redundant and low influence features).

1.5. Sensor Optimization

Often multiple sensors (sensor network) are used to give a more complete overview about the environment than a single sensor can give [40, 15]. This increases the diagnosis ability (failure detection and localization [5]) of a system and makes sensor optimization critical for failure diagnosis. The problem of designing a sensor network is to find a set of sensors so that costs, observability, reliability, estimation accuracy and flexibility are satisfied [16].

Sensor optimization shall help to design a sensor network that satisfies the requirements. It is a very wide topic and includes a number of different definitions. A few different meanings for sensor optimization are:

- Optimizing the position of sensors [35, 7].
- Optimizing the processing of sensor data [6].
- Optimizing the information gain of sensors.

Sensor optimization has the meaning of hardware redundancy optimization. Optimization is done by identifying significant sensors from a number of available sensors that give the most information about a system and thus increasing the information gain.

Goal of sensor optimization is to prevent unnecessary hardware redundancy and to improve the reliability of the condition monitoring system. This optimization can be supported by identifying redundant information in sensor data [5]. Traditional sensor optimization methods don't take into account the requirements for prognostic and health monitoring [34].

1.6. Multi-sensor Data Fusion

Having a network of different sensors that monitor a system leads to the problem of sensor data fusion. Multi-sensor data fusion covers the problem of combining sensor data from different sources into one consistent model. The main questions of sensor fusion are [2]:

- How to get accurate and reliable information from multiple and possible redundant sensors?
- How to fuse multi-sensor data with imprecise and conflicting data?

Techniques for sensor fusion can be grouped into these levels [15, 28, 3]:

- Data-level fusion (e.g. combining sensor data from same sensors directly [20]).
- Feature-level fusion (e.g. combining vectors and feature reduction techniques [28]).
- Decision-level fusion (e.g. vote schemes [28]).

Sensor data fusion is an important step for condition monitoring tasks. Most systems have more than one sensor and the sensor have different influences on the condition monitoring accuracy. Data for condition monitoring that needs to be fused is not only from sensors but can also be event and process data, which can deliver important information for the condition monitoring [15].

At the data-level fusion means the direct combination of sensor data; the data from sensors of the same kind is merged and fed into the condition monitoring system. The difficulty here is how to merge multiple sensors into one. Sensor fusion on the feature-level includes cleaning of sensor data and combining the sensor data after the features have been extracted and the dimensions reduced. Decision-level fusion can mean implementing a condition monitoring for each sensor separately and then use voting to decide on the system condition.

A condition monitoring system can use only one or multiple data fusion methods to detect the system conditions. This shows that the sensor fusion is a difficult problem that highly depends on the target system, and sensors. One solution would be to implement sensor fusion on all levels and then use a heuristic optimization like genetic algorithms, simulated annealing or hill climbing to get the bet sensor fusion methods for the given problem (data and system conditions).

2. Proposed methodology

Sensor selection and sensor fusion and consists of two steps. First a decision tree is build using feature extraction to increase the classification accuracy. The resulting decision tree may be analysed to generate a ranking of the involved sensors and features as the second step. The ranking then can be used to decide which sensors add significant information/features and which not. A sensor fusion may be performed at the feature-level. The calculated decision tree represents the feature fusion sorted by information gain. Sensor fusion on the feature-level does have the advantage that event data can also be added to the feature vector. Conventional methods use a fixed set of features to create feature vectors for the decision tree training and neglect the sensor fusion. Decision trees are used because they are good to use and implement in the aircraft environment. The task of the decision trees is merging of features from different sensor into one system health model and to use this model to classify the condition.

The focus is on hardware redundancy of sensors based on information gain to avoid having to install all sensors redundant and only focus on the sensors that give significant information for the failure detection and identification.

Goal of the method is to use information gain for ranking sensor importance and thus to have a measurement for sensor optimization. Feature extraction can increase the information gain and significance of different sensors.

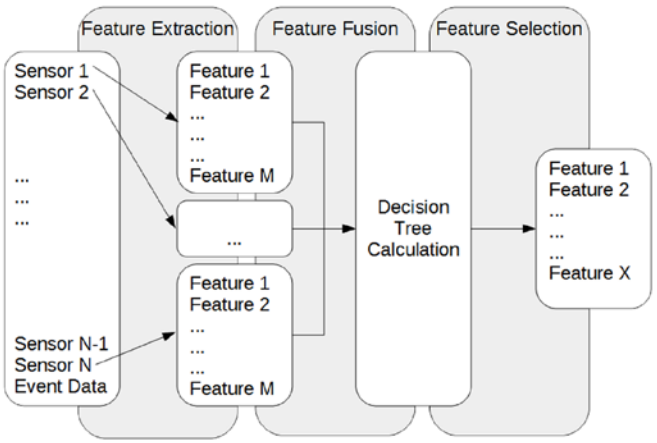

Fig. 4. Feature Selection Process

2.1. Feature Extraction and Sensor Fusion

The features extraction includes features from the time and the frequency domain. Time-frequency domain features were specifically not used. The reason was that the method shall only use basic methods thus Fast Fourier Transformation has been used. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations the can be different for each problem [21]. This makes elementary extraction operations also good for machine learning. The used operators are also fast to compute and can be used for online monitoring.

The data from the different sensors is not merged at the sensor level; instead it is merged on the feature extraction level. A feature set is calculated for each input from each sensor. These features are then merged into one feature input vector for the decision tree learning phase. No frequency features are calculated for signals that are nearly constant (Boolean switches, discrete system settings, and certain process parameter). The features extraction includes features from the time and the frequency domain. Time-frequency domain features were specifically not used. The reason was that the method shall only use basic methods thus Fast Fourier Transformation has been used. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations the can be different for each problem [21]. This makes elementary extraction operations also good for machine learning. The used operators are also fast to compute and can be used for online monitoring.

The data from the different sensors is not merged at the sensor level; instead it is merged on the feature extraction level. A feature set is calculated for each input from each sensor. These features are then merged into one feature input vector for the decision tree learning phase Figure 4. No frequency features are calculated for signals that are nearly constant (Boolean switches, discrete system settings, and certain process parameter).

2.2. Decision Tree Generation

This subsection describes the decision tree generation and signal processing. The decision tree is generated using algorithm C4.5. Algorithm C4.5 was used, because it is more advanced than the basic ID3 algorithm (accepts both continuous and discrete features, solves over-fitting problem by pruning handles, incomplete data points) and is available as an open source implementation J48. Input for the decision tree generation is a set of features that was extracted from sensor data. The feature extraction was controlled by different parameters. Table 1 shows the parameter list.

Table 1. Feature Extraction Parameter

Parameter	Possible Values	Default Value
Block Width	5/50/100/200	100
Noise Reduction Factor	0/1/2/5	1
Maximum Amplitude	Yes/No	Yes
Mean Amplitude	Yes/No	Yes
Maximum Power	Yes/No	Yes
Maximum Frequency	Yes/No	Yes
Mean Power	Yes/No	Yes
Number of Peaks	Yes/No	Yes
Peak Border	1/2/5	2
Global Maximum Amplitude	Yes/No	Yes
Global Mean Amplitude	Yes/No	Yes
Global Maximum Power	Yes/No	Yes
Global Mean Power	Yes/No	Yes
Global Number of Peaks	Yes/No	Yes
Confidence Factor	0.0001/0.001/0.01/0.1/1	0.001

The data types can range from Boolean data generated by switches or system conditions (event data) to high frequency data generated by sound and vibration data. Four more specific parameters will be explained in more detail below in this section.

2.2.1. Block Width

The block with defines how the frequency domain is partitioned to get features for each partition. A full transformation with the sampling frequency is done. After the fast Fourier transformation is done, the frequencies are partitioned up into blocks. The number of the frequencies that are grouped in one block is determined by the calculation parameter *Block Width*. If less then *Block Width* frequencies are available, then all frequencies are treated as one block. After partitioning all blocks are transformed back into the time domain, to get information about the behaviour of the block-signal over the time.

2.2.2. Noise Reduction Factor

Noise reduction is applied to the signal to remove random data from the samples in order to improve the feature detection of the undisturbed signal. The maximum frequency power is calculated and then every frequency signal that is below a defined fraction of the maximum frequency power is reduced to zero to remove noise from the sample. The exact fraction of the maximum frequency power for noise reduction is a parameter of the experiments *(Noise Reduction Factor)*. Noise reduction is done as shown in the Matlin Listing 1.

```
Y = fft(y);
x = mean(abs(Y)) *  NoiseReductionFactor ;
Y = Y. * ( abs (Y)>x );
```

Listing 1: Noise Reduction

2.2.3. Peak Border

The peak border is used for counting the number of frequencies that have a power above multitude of the mean power. The Matlab Listing 2 shows how the peaks are calculated. *peakBorder* is the parameter that can be varied and it defines, when a spike counts as a peak.

```
currPeakNum = 0;
for X = 1: blockWidth
  if ( Y_block (X) >= meanPower * peakBorder )
    peaks_block = peaks_block +1;
  end
end
```

Listing 2: Peak Calculation

The additional information is also calculated for the complete signal sample. Sensor Optimization

2.2.4. Confidence Factor

The confidence factor is a parameter of the software (WEKA [38]) that was used to create the decision trees and it defined how much tree pruning is done. A confidence factor of greater than 0.5 means that no pruning is done. The lower the confidence factor is the more pruning is done.

2.3. Sensor Optimization

The calculation of the *information gain* and learning of the decision tree is done by the *C4.5* algorithm that is used to construct decision trees for classification problems. Features are first extracted for each sensor signal then merged into one feature vector that is then the input for the C4.5 algorithm. The features are then sorted by the learning algorithm by the information gain. The feature with the highest information gain is placed at the root of the tree. Nodes with less information gain are placed at lower levels. For a binary decision tree this means that two nodes are in the second level of the tree, four nodes in the third level and so on. Each feature corresponds to a sensor.

The features in the decision tree are replaced with the sensor names for sensor ranking. If a sensor name appears on a level it is removed from all lower levels, so that for each sensor matches to one level in the decision tree. The sensors are now ranked by the decision tree level to which they are linked. It may happen that two sensors are at the same level.

3. VALIDATION

Two experiments were done to validate the concepts and ideas. The first experiment was done to show the effects of feature optimization and the second experiment was done to show how feature and sensor selection is done.

3.1. Feature Extraction Parameter Influence

To show the performance and concepts of the algorithm, a sensitivity analysis was performed by using different process parameters. Figure 5 shows the experiment process and how the results were generated. First samples are created and then sequentially are feature extraction parameters (see Table 1) modified. The influence of the modified parameter is measured by comparing the classification accuracy.

3.1.1. Data Sampling

The data for the experiments and the feature extraction was sampled with an autonomous box (Figure 6) that contained sensors and logic to save the data on a SD card. As a basis for the data collection a test rig was used. Vibration data with a sampling rate of 44 kHz of a simple PC fan (Figure 8) was collected. A PC fan was used to show the principals of the method. Data is saved in a raw wave format onto a SD card and then transferred onto a PC. In addition to the raw sensor data the condition of the component was saved. The fan is operated with standard speed, but three different conditions were sampled.

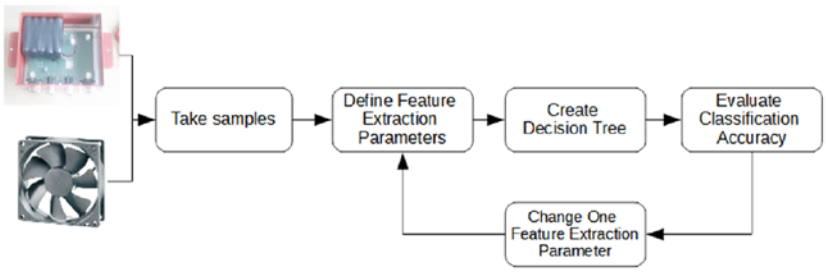

Fig. 5. Experiment Process Diagram

Data from the following conditions was collected:

- No additional weight.
- A very small weight (less than one gram) is added to one blade.
- A small coin (one Eurocent) is added to one blade.

Fig. 6. Data Recording Box

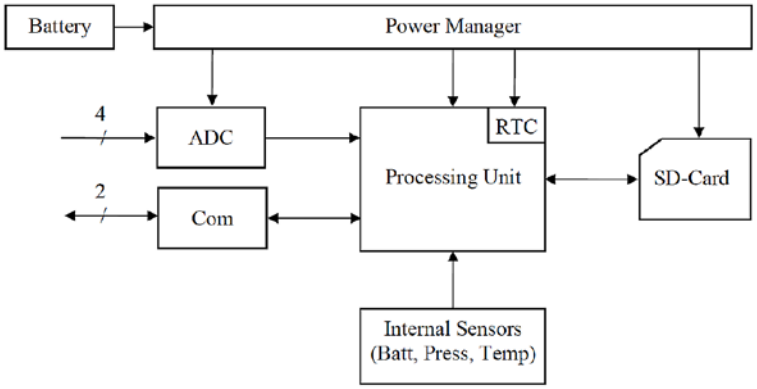

Fig. 7. Data Recording Box Architecture

Fig. 8. Used PC Fan

For each case 900 samples were collected. Every sample contains the vibration data of one second. Ten minutes passed between the individual samples. Samples were collected during office work hours and so a variety of noise is contained in the samples. In the experiment 900 "No weight" (no additional weight), 450 "Small weight" (a very small weight) and 450 "Big weight" (a small coin) samples were used. The decision tree of the J48 algorithm (an implementation of C4.5) in WEKA was validated with a 3-fold-crossvalidation (all samples are used for testing and training and the cross-validation process is repeated 3 times).

3.1.2. Calculating the Decision Tree

The decision tree is calculated with the open source Java software WEKA [38]. WEKA allows the user to test different algorithms and shows the classification errors that occurred. The correct data format is generated by using a Java program that transforms the output files from Matlab into input files for WEKA. For classification J48 is chosen, which is an implementation of the C4.5 decision tree algorithm, and a confidence factor of 0.0001. The confidence factor defines how much pruning is done to the resulting decision tree. The complete processed data is used as training data. After the generation of the decision tree the same data is used for testing the decision tree. In general the training and the testing data should not be the same, but in this case it is exactly what is wanted. The goal is not to classify new objects correctly, but to check how good the available data is classified and what part of the data gives us the most information about the system.

3.1.3. Experiment Parameters

Calculations with the same input data, but different parameter values, were performed to show the influence of the parameters on the results; Table 1 shows the available parameters with their possible values. All "Yes/No"-parameters are Boolean parameters, that toggle the calculation of that parameter during the processing. Default parameters are the values that are used, when the effect of a parameter onto the algorithm is tested. Only one value per test varies, while all other parameters keep their default value. The data processing with Matlab generates a number of different input sets for the J48 algorithm. For every input set a decision tree is generated and the influence of the modified parameter is then evaluated.

3.2. Sensor Optimization

The method was evaluated by using aircraft sensor data from the air conditioning system of an A320 aircraft. The aircraft was operated by ETIHAD Airways and was operating in the Middle East. The sensor data from the aircraft includes 589 flights over the duration of two years. Each sensor reading includes over 80 values consisting of continuous (numerical) and discrete data (Boolean). The data was sampled with a frequency of 1 Hz. Source of the data are different bus systems from the air conditioning system. Most data are temperature data and valve states. The sensor data includes:

Table 2. A320 sensor data description

Description	Bus	Type
Cabin Compartment Temperature Group 1	Zone Control	Numerical
Cabin Compartment Temperature Group 2	Zone Control	Numerical
Cabin Compartment Temperature Group 3	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 1	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 2	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 3	Zone Control	Numerical
Duct Overheat Warning Group 1	Zone Control	Boolean
Duct Overheat Warning Group 2	Zone Control	Boolean
Duct Overheat Warning Group 3	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 1	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 2	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 3	Zone Control	Boolean
Duct Temperature Group 1	Zone Control	Numerical
Duct Temperature Group 2	Zone Control	Numerical
Duct Temperature Group 3	Zone Control	Numerical
G + T Fan OFF	Zone Control	Boolean
Hot Air Switch Position ON	Zone Control	Boolean
Minimum Bleed Air Pressure Demand	Zone Control	Numerical

4. Result analysis

This section analyses the results of the data processing of the previous section. It begins by evaluating the experiments and their parameters and goes on to discuss the results of the best parameter configuration.

Figure 9 summarises the validation process. Results were created using the default parameter set; then, each parameter was varied based on its type. Boolean values were simply inverted but continuous val-

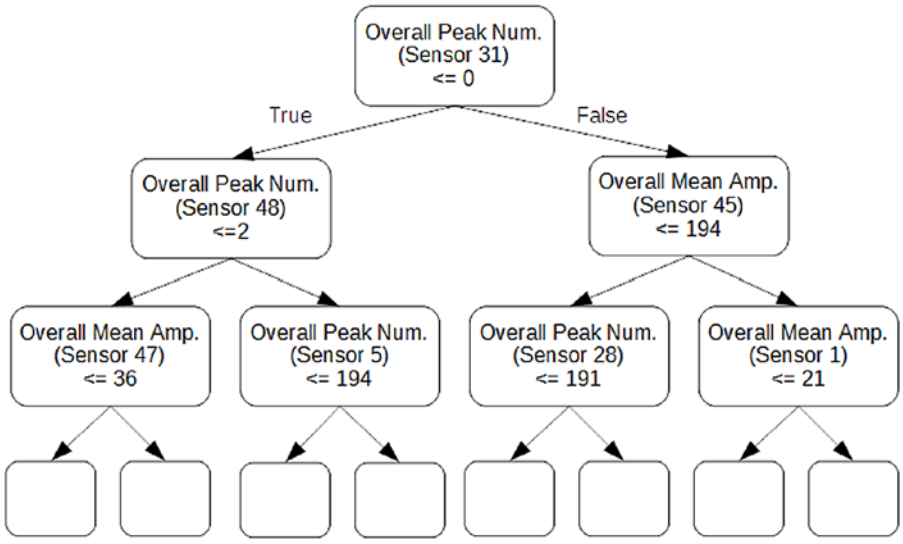

Fig. 10. Validation Process

ues were changed. A sensitivity analysis was performed after each parameter variation. After the parameter variation, a new decision tree with the same sensor data was created, and the change in the number of correctly classified samples was noted.

4.1. Parameter Evaluation

This section examines the results of the different input sets, based on the parameter variation. The influence of a parameter is judged by the number of correctly classified samples for every input set. Finding an optimal set of all parameters for the given samples, i.e., those giving the lowest overall false classification rate, is a complex problem. The complexity of the problem is so high that it is not possible to solve it in a fixed time; rather, heuristic methods have to be used. The results below are not the optimal parameter values; they only show the influence of the different parameter values on the classification accuracy and suggest the importance of optimizing the feature extraction parameters.

The first calculation was performed using the default parameters (see Table 1). The results are shown in Table 3. The numbers imply that about three quarters of the test cases are correctly classified. The error rate is quite high, but that is to be expected, because a non-optimal parameter set was selected as the default parameter set.

Table 3. Results for the Default Parameter Set

Correct Classified	False Classified
73.4 %	26.6 %

Table 4 splits the classification error into different classes. As the table shows, the majority of the samples are correctly classified. For samples with no additional weight and a big additional weight, the classification is very good, but samples with a small additional weight are often classified as samples with no additional weight. The results are still good, however, because the small attached weight is quite light, and sensing accuracy is not very high.

Table 4. Distribution of Wrongly Classified Samples

Sample Class	Classified as No	Classified as Small	Classified as Big
No	755	103	76
Small	175	218	57
Big	41	61	348

When only no additional weight and big additional weight samples are used, the number of wrongly classified samples dropps to 5 %. This is to be expected, because the features of both these classes have bigger differences than those found between the “Small” and “No” classes.

Table 6 shows the results achieved when the block width varies. The decreasing numbers imply that, at some point, an optimal block width can be reached and a minimum of false classified samples can be obtained. The error rate increases after the optimal point if the block width is too wide. The block width significantly affects how many features are calculated in total. Some features are calculated for each block. More features are calculated if the block width is low and, thus, the number of blocks is high.

Table 7 shows the experimental results for a varying noise reduction. The results indicate the accuracy of the classification can be improved by removing all frequencies with a power below the mean level. However, removing more frequencies with a high power can reduce the classification accuracy significantly because significant information about the signal is removed. This result also shows the noise frequency features have a significant influence on the accuracy of the classification.

Table 5. Results for the Default Parameter Set with no Small Weight Samples

Sample Class	Classified as No	Classified as Big
No	862	38
Big	60	390

Table 6. Results for Block Width

Block Width	False Classified
5	43.3%
50	27.4%
100	26.6%
200	24.3%

Table 7. Noise Reduction

Noise Reduction	False Classified
0	26.6%
1	24.2%
2	27.6%
5	42.6%

The calculation of the maximum amplitude can be turned on or off. Table 8 and Table 9 show the results. Results show the maximum amplitude does not have a big influence on the classification in this problem. This indicates a high resilience of the input data to noise, something relevant to entropy, or the information content in a message or information. The finding suggests the data samples contain a lot of information, and there is not much uncertainty in them [23]. This is even more interesting, because amplitude is the value recorded by the vibration sensors; it can be taken as an input without additional processing.

Table 8. Results for Maximum Amplitude per Block

Maximum Amplitude	False Classified
Yes	26.6%
No	26.5%

Table 9. Results for Global Maximum Amplitude

Global Maximum Amplitude	False Classified
Yes	26.6%
No	26.6%

Table 10 and Table 11 show the influence of the mean amplitude values. Again, the influence is quite small, similar to the influence of the maximum amplitude features. This is to be expected when the previous results are taken into account. The amplitudes are the only features based on the time domain data. This can indicate that the time domain features are not very significant for the classification as is often the case for rotary movements. More time domain features should be added to the feature extraction operations (like probabilistic moments) to give more information about the significance of the time domain signal.

Table 10. Results for Mean Amplitude per Block

Mean Amplitude	False Classified
Yes	26.6%
No	27.7%

Table 11. Results for Global Mean Amplitude

Global Mean Amplitude	False Classified
Yes	26.6%
No	26.6%

Table 12 and Table 13 show the results of the parameter variations for the maximum frequency power. Again, these features do not influence the result of the classification very much. It is interesting to note that the classification error is reduced if the block based maximum frequency power feature is turned off. This example clearly shows that having many features and features with little information gain can decrease the classification performance. It also highlights the importance of good feature selection.

Table 12. Results for Maximum Frequency Power per Block

Maximum Frequency Power	False Classified
Yes	26.5%
No	25.0%

Table 13. Results for Global Maximum Frequency

Maximum Frequency Power	False Classified
Yes	26.6%
No	26.6%

Table 14 and Table 15 show the results of the parameter variations for the frequency with the maximum power. The Hertz of the frequency with the highest power (local for each block or for the complete signal) does not influence the result in a significant way. This is to be expected because the maximum power also has little influence.

Table 14. Results for Frequency with Highest Power per Block

Frequency with Highest Power	False Classified
Yes	26.6%
No	26.3%

Table 15. Results for Global Frequency with Highest Power

Frequency with Highest Power	False Classified
Yes	26.6%
No	26.6%

Table 16 and Table 17 show the influence of the parameter variations for the mean frequency power. Mean frequency power is a big factor and can improve the classification by nearly 4 %. The global mean values give no information about the condition of the fan. This result is especially interesting, because the other frequency based features have little influence on the classification error. However similar to the maximum frequency power feature, the error rate decreases if this feature is not used.

Table 16. Results for Mean Frequency Power per Block

Mean Frequency Power	False Classified
Yes	26.6%
No	22.8%

Table 17. Results for Global Mean Frequency Power

Mean Frequency Power	False Classified
Yes	26.6%
No	26.6%

Table 18 and Table 19 show the influence of the number of peaks on the calculation. The number of peaks has an even bigger influence on the classification than the mean frequency power, and the false classification rate can be improved by nearly 5 %. To this point, this is the largest performance increase.

Table 18. Results for Number of Peaks per Block

Number of Peaks	False Classified
Yes	26.6%
No	21.8%

Table 19. Results for Global Number of Peaks

Number of Peaks	False Classified
Yes	26.6%
No	26.6%

The peak border (the value defining what a peak is) also influences the calculation, as shown in Table 20. Results for the peak border show no clear trend, but the numbers suggest an optimum exists. These results are interesting if we take into account how much the error rate improves when peaks per block are not calculated. Very few peaks are generated if the peak border is set to 5. This is quite similar to having no peaks at all.

The confidence factor determines how much the decision tree is pruned and has an influence on the classification accuracy. With less pruning, more samples are wrongly classified. Over-fitting is reduced

Table 20. Results for Peak Border

Peak Border	False Classified
1	24.3%
2	26.6%
5	22.3%

Table 21. Results for Confidence Factor

Confidence Factor	False Classified	Tree Size
1 (no pruning)	27.4%	275 Nodes
0.1	26.7%	225 Nodes
0.01	26.2%	185 Nodes
0.001	26.0%	163 Nodes
0.0001	26.6%	109 Nodes

when pruning is used. More pruning increases the generalisation ability of the decision tree, generally a good feature, but tree that is too small is not good. As in all other features, it is important to find the best value for the given classification problem.

It is interesting to note that the most significant feature seems to be the block based mean amplitude feature. The error rate increases for all other features if they are used. More experiments with different settings could ascertain how the different parameters are correlated, but finding the optimal parameter set can be really difficult. The best result (for the default parameter set and if only one parameter is modified) can be reached if the peak number is turned off. These results emphasise the importance of good feature selection and remind us of the difficulty of performing feature selection by hand. An automated feature selection is needed to find an optimal parameter set which improves the classification accuracy.

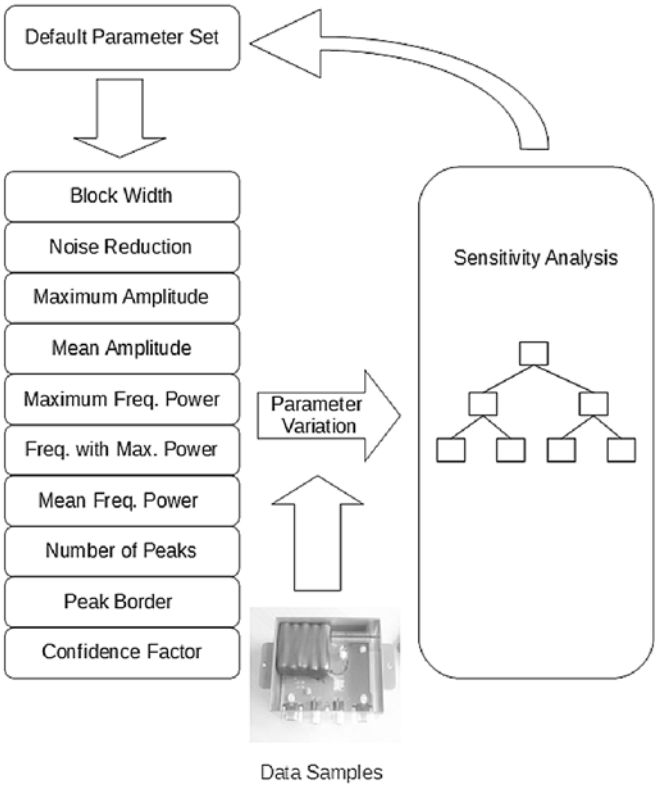

Fig. 9. Example Decision Tree

4.2. Sensor Optimization

This section shows the sensor optimization using the aircraft data with 80 sensors. Figure 10 contains a sample decision tree. The most important feature is the overall (global) number of peaks for the data source 31, followed by the overall (global) mean amplitude for sensor 45. Based on the decision tree, the sensors can be ranked as:

1. Sensor 31,
2. Sensor 48/Sensor 45,
3. Sensor 47/Sensor 5/Sensor 28/Sensor 1.

Sensor 31 is the most relevant sensor for the classification; sensors 48 and 45 are the second most significant ones. Redundancy, thus, applies to sensors 31, 48 and 45.

The decision tree also shows that the overall peak number and mean amplitude are the most relevant features. The significance of the amplitude is easily explained because the data contain switch and valve values which change slowly. The mean amplitude gives the classifier an indication of how often the switch is true and how often false. It is interesting to see that the peak number has more influence here than in the previous experiment.

5. Conclusions and Discussions

The method discussed here has been developed to handle a specific problem (classification of a small number of classes with simple features) in a specific domain (civil aircraft operation with online monitoring). Decision trees are a good solution, given these constraints. However, decision trees have limitations, and more powerful algorithms are available, if a similar problem needs to be solved outside the given constraints. More complex problems may need a different tool set. The methods used here are already well known and well researched, but their usage in this particular environment is novel. A previous paper [12] addressed the topic but evaluated the classification; it did not address sensor optimization and used an artificial experiment setup. This paper shows the result of sensor optimization by using real world data; it also explains the results and classification process in more detail than the earlier paper.

The architecture shown in Figure 4 is a good way to rank sensors by their significance for condition monitoring. The basic idea is to use a decision tree for feature ranking and feature fusion. Not all available features are used in the final decision tree thanks to tree pruning. As a result, fewer data are needed, and some sensors may not be used for the condition monitoring at all. It also improves the classification error rate and generalisation ability.

The validation experiment shows good failure classification can be performed with the proposed algorithms and methods. The feature extraction offers a modular system of elementary operations that can be used to extract features for a given problem.

The sensor optimization is best used for existing systems where reliability can be improved by additional hardware. The design is suitable when no online computation is available or data are logged but not evaluated but must be available in case of a failure for offline fault identification. The method can also be used for all systems with multiple sensors.

While the features improve the accuracy of the decision tree, it would be even better if more advanced feature extraction methods were used. Wavelet Package Transformation and KPAC are two suggestions. The method in this paper does not address how the best feature extraction parameter set is generated, but as the paper shows, this task is extremely important. Optimization algorithms are required. Future work could include using a genetic algorithm to search for the best parameter combination to classify a given dataset. The condition monitoring results can be used for trending and remaining useful life prediction.

References

1. Apte C, Damerau F, Weiss S.M. Text mining with decision trees and decision rules. In Proceedings of the Conference on Automated Learning and Discorery, Workshop 6: Learning from Text and the Web, 1998.
2. Basir O, Yuan X. Engine fault diagnosis based on multi-sensor information fusion using Dempster-Shafer evidence theory. Information Fusion 2007; 8 (4): 379-386, http://dx.doi.org/10.1016/j.inffus.2005.07.003.
3. Castanedo F. A review of data fusion techniques. The Scientific World Journal 2013, http://dx.doi.org/10.1155/2013/704504.
4. DIN EN 13306. Maintenance terminology. European Committee for Standardization, 2008.
5. Emami-Naeini A, Akhter M.M, Rock S. M. Effect of model uncertainty on failure detection: the threshold selector," IEEE Transactions on Automatic Control 1988; 33 (12): 1106-1115, http://dx.doi.org/10.1109/9.14432.
6. Farrar C.R, Park G, Puckett A.D, Flynn E.B, Mascarenas D.L, Todd M. D. Sensing and Sensor Optimization Issues for Structural Health Monitoring. 23rd Aerospace Testing Seminar, Manhatten Beach, 2006.
7. Fijany A, Vatan F. A unified and efficient algorithmic approach to model-based diagnosis and optimal sensor placement. Proceedings of 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS), 2005.
8. Fonollosa J, Vergara A, Huerta R. Algorithmic mitigation of sensor failure: Is sensor replacement really necessary?. Sensors and Actuators B: Chemical 2013; 183: 211-221, http://dx.doi.org/10.1016/j.snb.2013.03.034.
9. Friedl M.A, Brodley C.E. Decision tree classification of land cover from remotely sensed data. Remote sensing of environment 1997; 61 (3): 399-409, http://dx.doi.org/10.1016/S0034-4257(97)00049-7.
10. Fu T.-C. A review on time series data mining. Engineering Applications of Artificial Intelligence 2011; 24 (1): 164-181, http://dx.doi.org/10.1016/j.engappai.2010.09.007.
11. Gerdes, M., Scholz, D. & Randerath, B. Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration. 2nd International Workshop on Aircraft System Technologies, 2009.
12. Gerdes, M., Scholz, D. Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters. Deutscher Luft- und Raumfahrtkongress 2009.
13. International Electrotechnical Commission. IEC 61511: Functional Safety - Safety instrumented systems for the process industry sector. International Electrotechnical Commission, 2003.
14. International Electrotechnical Commission. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. International Electrotechnical Commission, 2010.
15. Jardine A.K, Lin D, Banjevic D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. Mechanical systems and signal processing 2006; 20 (7): 1483-1510, http://dx.doi.org/10.1016/j.ymssp.2005.09.012.
16. Kotecha P. R, Bhushan M, Gudi R. D. Design of robust, reliable sensor networks using constraint programming. Computers & Chemical Engineering 2008; 32 (9): 2030-2049, http://dx.doi.org/10.1016/j.compchemeng.2008.03.005.
17. Lambrou T, Kudumakis P, Speller R, Sandler M, Linney A. Classification of audio signals using statistical features on time and wavelet transform domains. Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998, http://dx.doi.org/10.1109/icassp.1998.679665.
18. Li, F Li F. Dynamic modeling, sensor placement design, and fault diagnosis of nuclear desalination systems. University of Tennessee 2011.
19. Lin J, Qu L. Feature extraction based on Morlet wavelet and its application for mechanical fault diagnosis. Journal of sound and vibration 2000; 234 (1): 135-148, http://dx.doi.org/10.1006/jsvi.2000.2864.
20. Lu Y, Michaels J.E. Feature extraction and sensor fusion for ultrasonic structural health monitoring under changing environmental conditions. IEEE Sensors Journal 2009; 9 (11): 1462-1471, http://dx.doi.org/10.1109/JSEN.2009.2019339.
21. Mierswa I, Morik K. Automatic feature extraction for classifying audio data. Machine Learning 2005; 58 (2-3): 127-149, http://dx.doi.org/10.1007/s10994-005-5824-7.
22. Novis A, Powrie H. PHM sensor implementation in the real world-a status report. 2006 IEEE Aerospace Conference, 2006, http://dx.doi.org/10.1109/AERO.2006.1656113.
23. Pascual D. G. Artificial Intelligence Tools: Decision Support Systems in Condition Monitoring and Diagnosis, CRC Press, 2015, http://dx.doi.org/10.1201/b18384.
24. Peng Z, Chu F, He Y. Vibration signal analysis and feature extraction based on reassigned wavelet scalogram. Journal of Sound and Vibration 2002; 253 (5): 1087-1100, http://dx.doi.org/10.1006/jsvi.2001.4085.
25. Quinlan J. R. Induction of Decision Trees. Machine Learning 1986; 81-106, http://dx.doi.org/10.1007/BF00116251.
26. Quinlan J. R. C4.5: Programs for Machine Learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc, 1993.
27. Radio Technical Commission for Aeronautics. DO 178C: Software considerations in airbone systems and equipment certification. Radio Technical Commission for Aeronautics, 2011.
28. Ross A, Jain A. Information fusion in biometrics. Pattern recognition letters 2003; 24 (13): 2115-2125, http://dx.doi.org/10.1016/S0167-8655(03)00079-5.
29. Russell S.J, Norvig P. Artificial Intelligence: A Modern Approach. Pearson Education, 2003.
30. Saimurugan M, Ramachandran K, Sugumaran V, Sakthivel N. Multi component fault diagnosis of rotational mechanical system based on decision tree and support vector machine. Expert Systems with Applications 2011; 38 (4): 3819-3826, http://dx.doi.org/10.1016/j.eswa.2010.09.042.
31. Sakthivel N, Sugumaran V, Nair B.B. Comparison of decision tree-fuzzy and rough set-fuzzy methods for fault categorization of mono-block centrifugal pump. Mechanical systems and signal processing 2010; 24 (6): 1887-1906, http://dx.doi.org/10.1016/j.ymssp.2010.01.008.
32. Scerbo M.W, Bailey N. R. Automation-induced complacency for monitoring highly reliable systems: the role of task complexity, system experience, and operator trust. Theoretical Issues in Ergonomics Science 2007; 321-348.
33. Schmid H. Probabilistic part-of-speech tagging using decision trees. Proceedings of the international conference on new methods in language processing, 1994.
34. Shuming Y, Jing Q, Guanjun L. Sensor optimization selection model based on testability constraint. Chinese Journal of Aeronautics 2012;

25 (2): 262-268, http://dx.doi.org/10.1016/S1000-9361(11)60386-5.
35. Smith M. J. Sensor Location & Optimization Tool Set. Chemical Biological Information Systems Conference, Albuquerque, 2005.
36. Sugumaran V, Ramachandran K. Automatic rule learning using decision tree for fuzzy classifier in fault diagnosis of roller bearing. Mechanical Systems and Signal Processing 2007; 21 (5): 2237-2247, http://dx.doi.org/10.1016/j.ymssp.2006.09.007.
37. Sun J.-W, Xi L.-F, Pan E.-S, Du S.-C, Xia T.-B. Design for diagnosability of multistation manufacturing systems based on sensor allocation optimization Computers in Industry 2009; 60 (7): 501-509, http://dx.doi.org/10.1016/j.compind.2009.02.001.
38. University of Waikato, WEKA, 2009.
39. Widodo A, Yang B.-S. Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors. Expert Systems with Applications 2007; 33 (1): 241-250, http://dx.doi.org/10.1016/j.eswa.2006.04.020.
40. Xiong N, Svensson P. Multi-sensor management for information fusion: issues and approaches. Information fusion 2002; 3 (2): 163-186, http://dx.doi.org/10.1016/S1566-2535(02)00055-6.
41. an W, Goebel K. F. Sensor validation and fusion for gas turbine vibration monitoring. AeroSense 2003.

Mike GERDES
Hamburg University of Applied Sciences
Aero - Aircraft Design and Systems Group
Berliner Tor 9, 20099 Hamburg, Germany
Luleå University of Technology
Division of Operation and Maintenance Engineering
SE-971 87 Luleå, Sweden

Diego GALAR
Luleå University of Technology
Division of Operation and Maintenance Engineering
SE-971 87 Luleå, Sweden
University of Skövde
Reliability and Maintenance Engineering
SE-54128, Skövde, Sweden

Dieter SCHOLZ
Hamburg University of Applied Sciences
Aero - Aircraft Design and Systems Group
Berliner Tor 9, 20099 Hamburg, Germany

Emails: mike.gerdes@philotech.de, diego.galar@ltu.se, info@profscholz.de