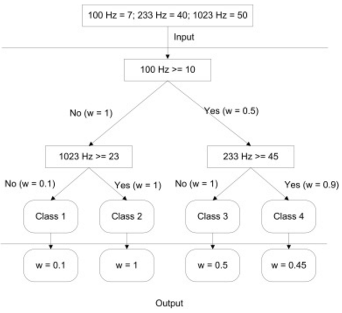

Predictive Health Monitoring for Aircraft Systems using Decision Trees

Mike Gerdes

Licentiate Thesis
Division of Fluid and Mechatronic Systems
Department of Management and Engineering
Linköping University

Linköping Studies in Science and Technology
Licentiate Thesis No. 1655

Predictive Health Monitoring for Aircraft Systems using Decision Trees

Mike Gerdes

Linköping University
INSTITUTE OF TECHNOLOGY

Division of Fluid and Mechatronic Systems
Department of Management and Engineering
Linköping University, SE–581 83 Linköping, Sweden

Linköping 2014

Predictive Health Monitoring for Aircraft Systems using Decision Trees

ISBN 978-91-7519-346-5
ISSN 0280-7971

Distributed by:
Division of Fluid and Mechatronic Systems
Department of Management and Engineering
Linköping University
SE-581 83 Linköping, Sweden

Printed in Sweden by LiU-Tryck, Linköping 2014.

Abstract

Unscheduled aircraft maintenance causes a lot problems and costs for aircraft operators. This is due to the fact that aircraft cause significant costs if flights have to be delayed or canceled and because spares are not always available at any place and sometimes have to be shipped across the world. Reducing the number of unscheduled maintenance is thus a great costs factor for aircraft operators. This thesis describes three methods for aircraft health monitoring and prediction; one method for system monitoring, one method for forecasting of time series and one method that combines the two other methods for one complete monitoring and prediction process. Together the three methods allow the forecasting of possible failures. The two base methods use decision trees for decision making in the processes and genetic optimization to improve the performance of the decision trees and to reduce the need for human interaction. Decision trees have the advantage that the generated code can be fast and easily processed, they can be altered by human experts without much work and they are readable by humans. The human readability and modification of the results is especially important to include special knowledge and to remove errors, which the automated code generation produced.

Keywords: Condition Monitoring, Condition Prediction, Failure Prediction, Decision Trees, Genetic Algorithm, Fuzzy Decision Tree Evaluation, System Monitoring, Aircraft Health Monitoring

Acknowledgment

I thank all persons who made this thesis possible. Professor Scholz and Bernhard Randerath for setting up the PAHMIR project (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) and supporting me during all the time during my work. I also thank professor Petter Krus for making it possible to be a PhD. student at the Linköpings Univesitet and for supporting me on my way. Finally I would like to thank Philotech GmbH that they made it possible for me to leave my job for a few years and return after my work at the HAW Hamburg.

The research for this thesis is sponsored by the government of Hamburg, Ministry for Economics and Labor (Behörde für Wirtschaft und Arbeit - BWA) as part of the Aviation Research Programme Hamburg (LuFo Hamburg).

Foreword

This thesis presents a concept for adaptable predictive aircraft health monitoring with decision trees. The project (PAHMIR - Preventive Aircraft Health Monitoring with Integrated Reconfiguration) that lead to this dissertation was started in 2008 as a cooperative project between Hamburg University of Applied sciences and Airbus Operations GmbH.

The organization of the dissertation is as followed:

Introduction The introduction chapter shall give the reader an understanding of the problem, the motivation for the research, the research background and the solution concept. In the begin the section describes the project in which the research was done and which gave the motivation for the research. This is followed by an explanation of the motivation and why research work was necessary. This is enhanced by a full description of the objectives of the research. The section closes with a review of the concepts that were applied to solve the problem and reach the objectives.

Theoretical Background The second section explains the theoretical background of the different concepts which are used for the concept. Those are: decision trees, heuristic optimization, signal analysis, condition monitoring and time series analysis. The order of the topics is based on the order how they are later used in the concept. The section closes with a summary.

Condition Monitoring The condition monitoring section contains the first part of the developed concept to solve the initial problems. It explains the process and how the methods that were presented in the previous chapter were applied to solve a part of the problem.

Condition Prediction Condition prediction is the second part of the concept and is explained in the fourth section. The section is in the same way structured like the previous section. The process of condition prediction is explained and it is shown how the different methods work together to get a prediction of the system condition.

Failure Prediction Failure prediction is the combination of condition monitoring and condition prediction to forecast when a failure will happen. This section describes how the two previous processes can be combined to provide a complete process for failure prediction.

Experiments The experiments section shows how feasible and usable the developed concepts really are. The section is divided up into the evaluation of the condition monitoring concepts and the evaluation of the condition prediction. The two concepts use different experimental setups for the evaluation.

Conclusions The conclusions section summarizes the results and shows were future work can still be done.

Papers

This subsection gives an overview about the papers that were published during the writing of this thesis and to which the thesis refers during various points in the text. The papers are ordered in the chronological ordered in which they were published.

Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration The first published paper analyses causes and costs for aircraft delays due to faults in the air conditioning system. The analysis includes an analysis of the duration of delays and how the duration and thus the costs can be reduced by using preventive maintenance. Additionally the paper discusses different method to reduce the duration for cabin reconfiguration. It was published during the International Workshop on Aircraft System Technologies 2009 (AST2009) in Hamburg[1].

Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters The second published paper concentrates on feature extraction and sensor optimization using decision trees. A decision tree is used for sorting the features based on information gain. Then the sensors that produce the important feature can be made redundant, while other can be neglected. The second paper was published in 2010 during the Deutschen Luft- und Raumfahrt Kongress (DGLR2009) in Aachen[2].

Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems Following the feature extraction and sensor optimization is a concept for automatically selecting signal analysis methods and parameters to generate features for the decision tree calculation. A number of different signal analysis methods are selected and then an optimization algorithm is used to select the best signal analysis methods to generate feature for a given decision problem.

The paper compares different optimization algorithms and compares the speed and accuracy of those using data from an Airbus testrig for air conditioning. This paper was published during the International Workshop on Aircraft System Technologies 2011 (AST2011) in Hamburg[3].

Fuzzy Condition Monitoring of Recirculation Fans and Filters Decision trees normally yield only hard discreet results. In this paper a method was published, which uses a normal decision tree and weights decisions taken to create a fuzzy result. The result gives the user a feedback on how likely other results are and what happens to the results if a parameter slightly changes. The paper was published in the CEAS Aeronautical Journal in 2011 and presented on the Deutschen Luft- und Raumfahrt Kongress (DGLR2011) in Bremen[4].

Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning The last published paper looks at the use of decision trees to predict future values in a time series. A decision tree is trained on past data and then uses features of a time series to decide, which extrapolation method for the next data points is the best for the current time series. The paper also shows some theoretical experiments and shows how the method can be improved for different problems. The last paper was published 2013 in the Expert Systems With Applications journal[5].

[1] Mike Gerdes, Dieter Scholz and Bernhard Randerath. ‘Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration’. In: *2nd International Workshop on Aircraft System Technologies, AST 2009 (TUHH, Hamburg, 26./27. März 2009)*. 2009.

[2] Mike Gerdes and Dieter Scholz. ‘Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters’. In: *Deutscher Luft- und Raumfahrtkongress 2009 : Tagungsband - Manuskripte (DLRK, Aachen, 08.-10. September 2009)*. 2009.

[3] Mike Gerdes and Dieter Scholz. ‘Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems’. In: *3nd International Workshop on Aircraft System Technologies, AST 2011 (TUHH, Hamburg, 31. März - 01. April 2011)*. 2011.

[4] Mike Gerdes and Dieter Scholz. ‘Fuzzy Condition Monitoring of Recirculation Fans and Filters’. In: *CEAS Aeronautical Journal* 2 (1–4 2011), pp. 81–87.

[5] Mike Gerdes. ‘Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning’. In: *Expert Systems With Applications* 40 (12 2013), pp. 5021–5026.

Contents

Appended papers

1 Introduction

This section gives the reader an overview about the motivation and concept discussed in this document.

1.1 Preventive Aircraft Health Monitoring for Integrated Reconfiguration (PAHMIR)

The PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) project was the research environment and base for this work. PAHMIR was a cooperative research project between Airbus Operations GmbH and Hamburg University of Applied Sciences (HAW Hamburg). The project was funded by the city of Hamburg and had a duration of 3.5 years. Start was January 2008 and end was June 2011. All research work was done during this period. Goal of PAHMIR was to analyze existing in-service aircraft maintenance data, develop a predictive aircraft health monitoring system and analyze how such a system might be integrated into a dynamic cabin concept. Concepts for condition monitoring, condition prediction and in door localization were developed and tested in experiments in the project.

The goals of PAHMIR are:

- Reduction of unscheduled maintenance
- Advanced failure prediction
- Condition monitoring
- Ability to better plan maintenance

- Improve cabin reconfiguration

1.2 Motivation

One goal of PAHMIR was to prevent and forecast failures. The main drivers for the development of a failure prediction concept are the costs of a delay of an aircraft departure or arrival. Delays can be caused by unscheduled maintenance between aircraft arrival and aircraft departure. Failure prediction shall allow the aircraft operator to repair or replace a system during scheduled maintenance, if the system is not yet broken but will be before the next scheduled maintenance. Figure 1.1 shows the handling of an aircraft fault without predictive health monitoring (failure prediction).

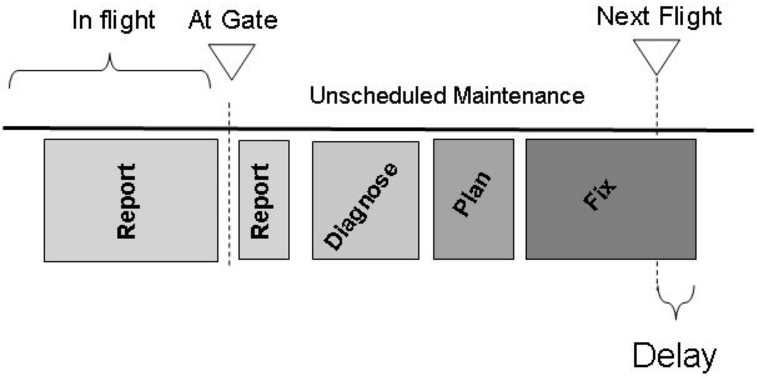

Figure 1.1 *Unscheduled maintenance without failure forecasting*

The maintenance case in Figure 1.1 is like this: a fault happens in flight. Sensors detect the fault and report the fault to the cockpit. The pilot/aircraft sends a maintenance request to the airport. A maintenance mechanic checks the aircraft, when it is on ground. The mechanic performs a fault search and a fault diagnosis. Spare parts are ordered and a repair plan is made after the fault has been identified. When the spare parts arrive, it is possible to do the repair. The aircraft is ready again after the repair. It is possible that the fault identification, diagnostics and spare parts management take too much time, so that the aircraft departure is delayed or even canceled. A delay causes significant costs for an aircraft operator.

In the ideal case (to which PAHMIR is a step) most faults that will occur are repaired during scheduled maintenance (Figure 1.2). It is still possible that a fault occurs. Sensors and fault detection systems identify and diagnose the fault, if a fault occurs during flight. The aircraft sends a fault report and diagnosis to the ground, where a maintenance mechanic gets the spare parts and prepares the repair. The fault is repair after the aircraft lands. Delays can be prevented by repairing future faults in the hanger. This reduces the number of unscheduled maintenance cases.

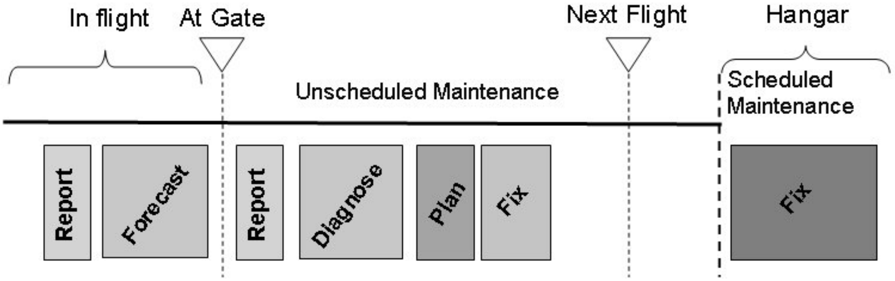

Figure 1.2 *Unscheduled maintenance with failure forecasting*

The costs of a delay can be quite large if the delay is long or the flight is canceled. [1] shows an analysis of the costs of a delay and what can be saved by forecasting faults and do repairs during scheduled maintenance.

1.3 Research Goal

It is possible to formulate the goal, requirements and constrains of a failure prediction concept with the given motivation and goals of the PAHMIR project. Important factors are the aircraft environment and generalization of the concept. This leads to the following goals for the concept:

- An adaptable system condition monitoring
- Simple and verifiable monitoring algorithms
- Failure prediction with 500 flight hours in advance
- Condition monitoring should be online and offline possible
- Condition monitoring and prediction needs to be usable for a changeable cabin layout

- Low number of needed sensors
- Low hardware profile
- Use new and existing sensors
- Preferable not only limited to the aircraft domain
- Low human interaction

Goal of the research is to develop a condition monitoring and forecasting concept that is usable in the aircraft environment, that can be used for different system and can be used by operators without much system knowledge and knowledge of the monitoring and prediction concept. It is critical that the developed concept can used for different system without much work to adapt it. This is because the system shall be easy to use by different developers for different aircraft systems. Also to be able to use the concept in the aircraft environment it is necessary that the used algorithms can easily be verified and understood by a human. This makes it easier to ensure that the system correctly monitors the system condition and also that it forecasts the condition correctly.

The given requirements lead to the concept that is given below. The concept is a software concept that can be embedded in different environments. The largest amount of computation takes place during the configuration of the failure prediction system and not during the operation. Most used computations and methods are fast to calculate and need not much hardware power or memory. The concept needs sensor input and a way to output the predictions.

1.4 Concept

The developed concept for predictive aircraft health monitoring is able to predict failures so that maintenance can be planned ahead. The developed concept is based on two different processes (condition monitoring and condition forecasting) that work together to create a complete concept. The two processes use decision trees (see Section 2.4). Decision trees are used to make decision in the concept and are the core of both processes. In the first process (condition monitoring) the task is to decide, which condition the sensor data represents and in the second process (condition prediction) the task is to decide how to predict data points. Both tasks are solved by the decision tree.

The core idea behind the concept is to use machine learning to create an expert system. A human expert is needed for configuring the starting parameters and linking sensor data to a system condition during the training. After the training the system is able to work without a human expert. The expert system is designed as a statistical system model to allow a high level of adaptability. A statistical system allows the user to use measurement data to create a system model without the need of full system knowledge. The two processes use parameter optimization to improve the performance of the decision trees and the overall performance. Optimization reduces the need for a human after the initial data and parameter configuration. All process parameters that may change can be changed until an optimal parameter set is found or until a number of different decision trees have been calculated.

The concept can be embedded in most hardware platforms and is system independent. The training of the decision algorithms can be done on any platform and the resulting code is based on simple "if-then-else" statements, which can also be implemented in most platforms. Digital signal processors (DSP) are especially suited for the condition monitoring, because they can calculate the signal processing very fast. With an optimal hardware architecture and a good implementation it is possible to use the condition monitoring and the condition prediction in real-time. Signal processing and prediction parameter approximation parameter calculation take most time.

1.4.1 Condition Monitoring

The condition monitoring concept uses sensor data to calculate the current condition of the system. This can be the system state (e.g. normal, error 1, error 2 ...) or the remaining life time. The concept does not rely on a special type of sensor or only sensor data from one source or kind. It is possible to use any kind of data or combination of data. The concept works best with sensor data which changes multiple times per second. If the data is not numerical, then the preprocessing can not be applied, but it is still possible to use all other parts of the process. This makes it possible to merge data from different sources and of different kinds into one system condition. An extra output of the concept is the similarity of the current sensor data to another system condition and not only to the class is was mapped to. The condition monitoring process is shown in Figure1.3.

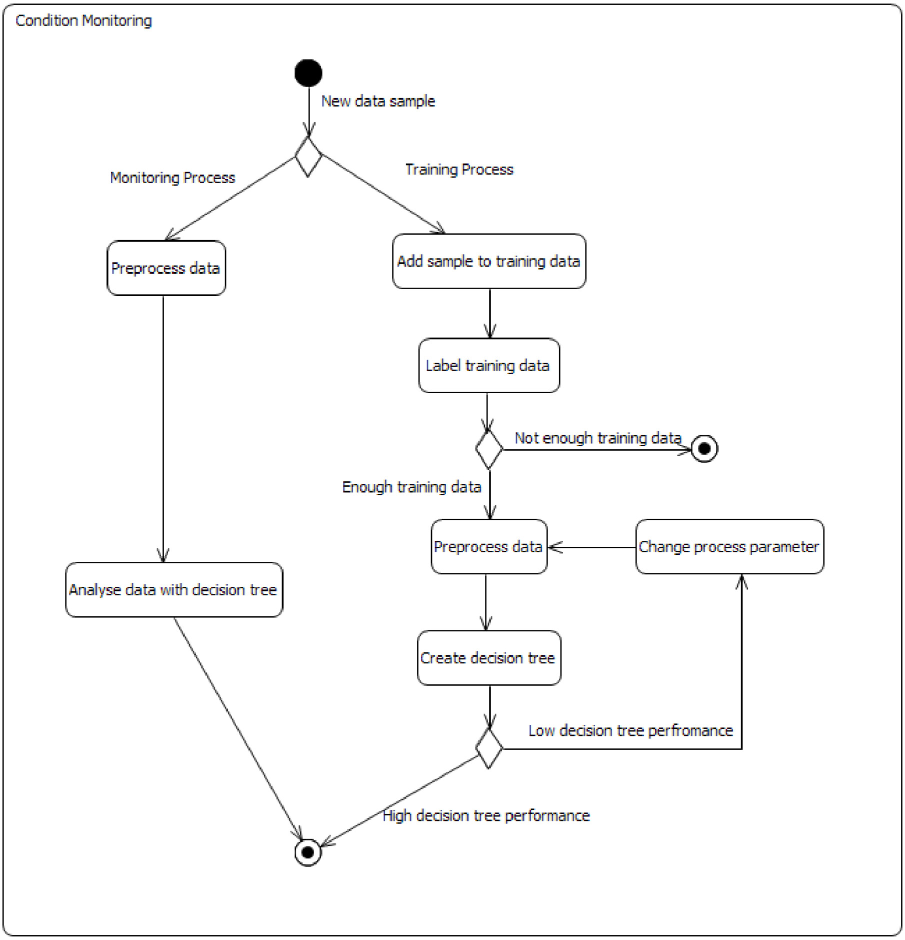

Figure 1.3 *Condition monitoring process*

Condition monitoring is based on a decision tree, which calculates a decision based on signal features. The decision tree needs to be trained with data samples. Preparation of training samples (signal feature extraction) is a complex task and controlled by parameters. The performance and adaptability are improved by an optimization process. Condition monitoring is a simple process compared to the training process. The following methods and technologies are used for the concept:

- Decision trees (Section 2.4)
- Signal analysis (Section 2.2)
- Optimization (Section 2.5)

Fuzzy decision tree evaluation is used and needed to provide also continuous results (percentage values) in addition to the discrete decision of the decision tree evaluation. The continuous results are the similarity of the data belonging to another class. The value of the failure case is used as an input to the condition prediction process, which needs a continuous value.

1.4.2 Condition Prediction

Condition prediction takes a time series (chronological ordered data points) and predicts future data points based on learned patterns/knowledge. It is possible to train the system to predict data points in the close future or in the far future. Prediction is done by calculating a suitable approximation based on learned experience. The condition prediction process is shown in Figure 1.4.

Training of the decision tree is the most complex task of the process like for condition monitoring. The time series, which shall be predicted needs to be prepared and features need to be extracted. Performance is also improved by an optimization process. While the process looks more complicate than the condition monitoring it is easy to compute and the steps are easy to understand by a human. The following methods and technologies are used for the concept:

- Decision trees (Section 2.4)
- Time series analysis (Section 2.3)
- Optimization (Section 2.5)

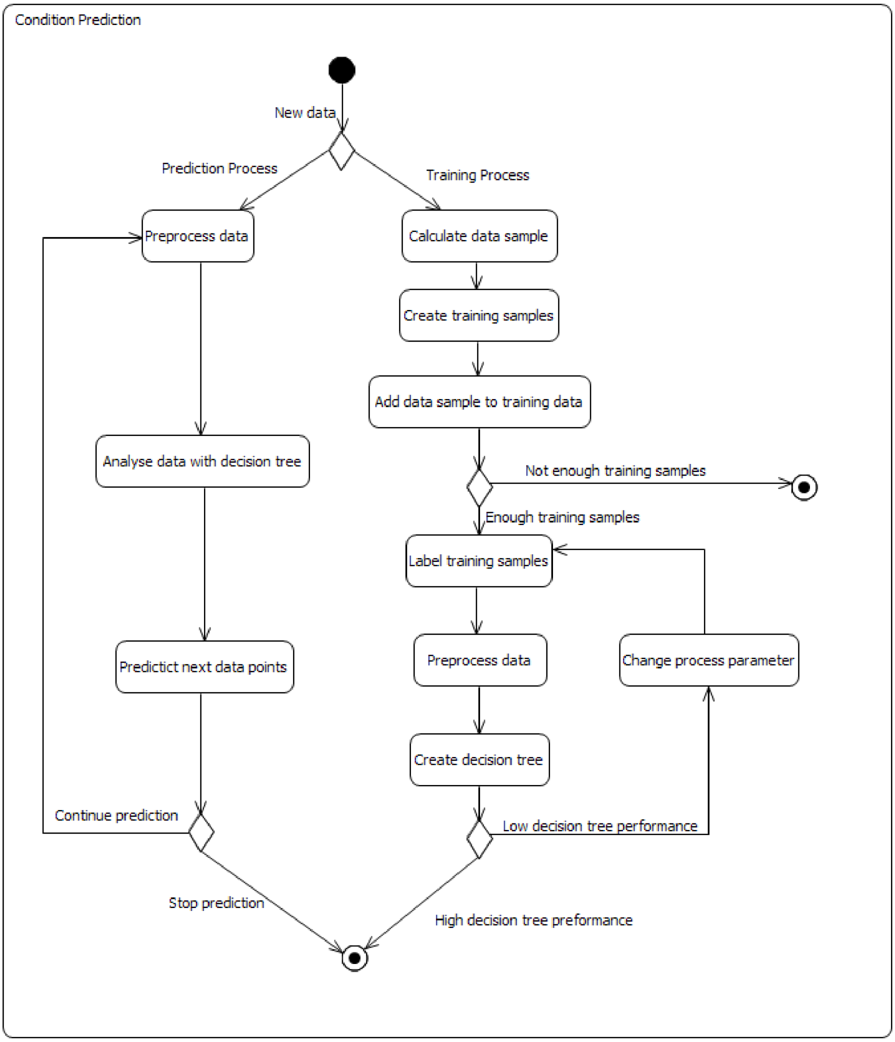

Figure 1.4 *Condition prediction process*

1.4.3 Failure Prediction

Failure Prediction combines the condition monitoring and the condition prediction processes into one compete process that can be used to forecast a failure. Base for the failure prediction is the condition monitoring which gives the process the current system state (if correctly trained). However for the failure prediction there is no direct need for the current system state. What is needed is the similarity of the current state to a failure state. This gives the user more info than just the system state, because the a sample might be from the border of a class. Fuzzy decision tree evaluation[4] allows it to take any decision tree and calculate how similar a sample is to another class. A side effect is that the fuzzy evaluation converts the discrete result of the decision tree classification into a continuous number, if you just want to know how similar a sample is to a specific class. For the failure prediction this class is the class of the failure that shall be predicted. Continuous numbers are needed as the input for the condition prediction process, which predicts the trend of the class based on past data.

- Condition Monitoring
- Fuzzy decision tree evaluation
- Condition Prediction

2

Theoretical Background

2.1 Conditon Monitoring

Condition monitoring is part of condition based maintenance[6]. Maintenance is the combination of all technical and associated administrative actions intended to retains an item in, or restore it to, a state in which it can perform its required function[7]. The goal is to prevent fatal damage for machine, human or environment, to prevent unexpected machine failure, condition based maintenance planning, safety of production and quality control[8]. Figure 2.1 shows a breakdown of different maintenance strategies.

Basically there are three different maintenance strategies [6][8]:

- **Run-to-break** is the most simple maintenance that is often used for systems that are cheap and where a damage does not cause other failures. The machine or system is used until it breaks. It is commonly used for consumer products[8].

- **Preventive Maintenance** is the most common maintenance method for industrial machines and systems. Maintenance is performed in fixed intervals. The intervals are often chose so that only 1-2 % of the machine will have a failure in that time[6].

- **Condition-Based Maintenance** is also called predictive maintenance. Maintenance is dynamical planned based on machine or

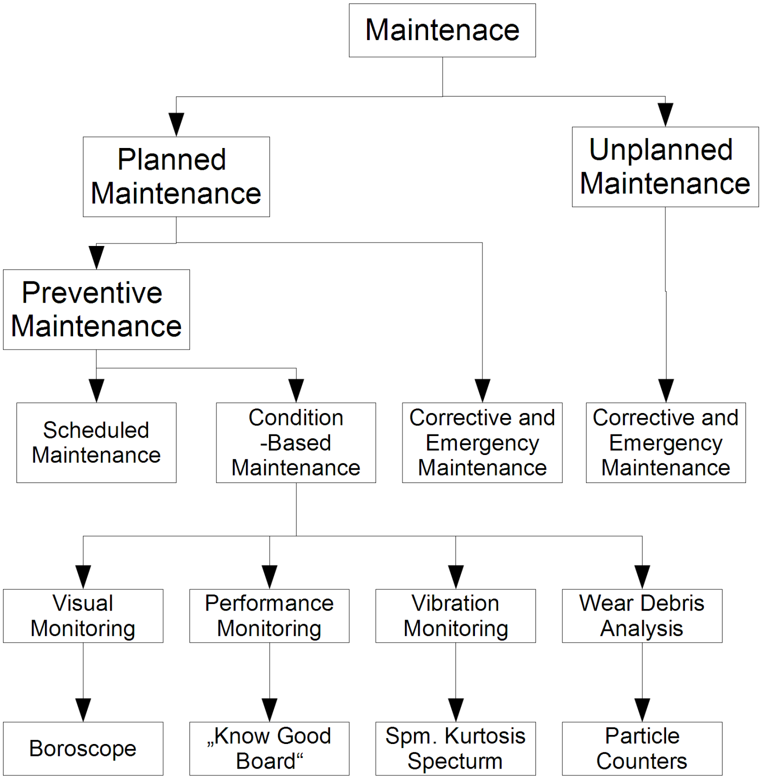

Figure 2.1 *Maintenance[9]*

system condition. Condition-Based Maintenance does have advantages compared to the other two strategies, but requires a reliable condition monitoring method [6]

Figure 2.2 shows a typical machine condition based monitoring case. First the machine goes into operation and then it is in normal operation. The machine is replaced short before a failure happens[8].

Condition monitoring can be performed in two strategies for monitoring[6][8]:

- **Permanent monitoring** is based on fix installed measurement systems. Often these systems need to be very complex to react correctly if a failure occurs. These systems are often used if a fast reaction is required after a failure. Permanent monitoring often shuts down a machine if a dangerous failure is detected[6].

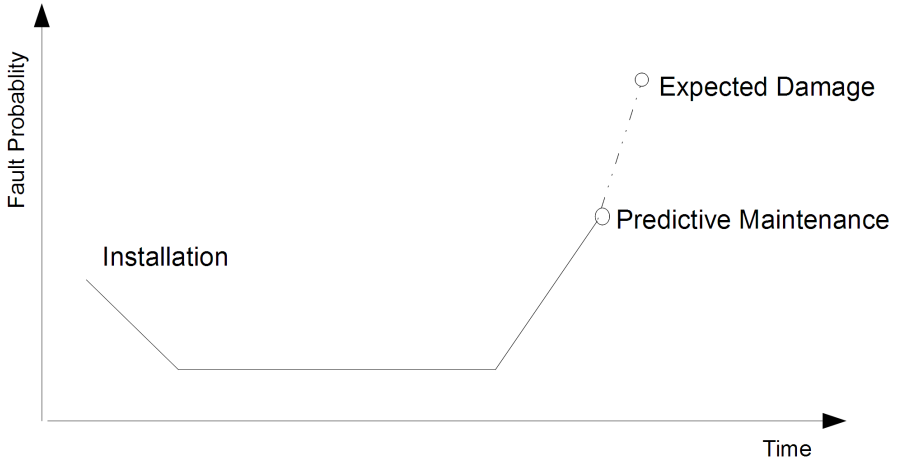

Figure 2.2 *Machine/system condition over time[8]*

- **Intermittent monitoring** is generally used to failure prediction and diagnosis. Measurements are taken or regular basis with a mobile device. Data evaluation is done at an external device. Intermittent monitoring is often used to give a long term advance warning[8].

Permanent monitoring is often more easy than intermittent monitoring, because fast reaction times are required. Intermittent monitoring can be more complex and can do more complex computations[6]. Permanent and intermittent monitoring can be combined using the same sensors and working in parallel. This allows the intermittent monitoring to be carried out more often (data is always available[6].

Different methods for condition monitoring are[6]:

- **Vibration analysis** measures the vibration of a machine or system and compares it to a given vibration signature. Vibrations can be linked to events in a machine based on their frequency. Therefore a vibration signal is often analyzed in the time domain and in the frequency domain. Vibration analysis is often used for condition monitoring[6][8].

- **Lubricant/Oil analysis** analyzes the quality of the fluid and if particles are in the fluid. Contaminants in lubrication oils and hydraulic fluids can lead to the failure of the machine/system. The physical condition of a fluid can be measured in viscosity, water content, acidity and basicity. For a condition monitoring strategy this means condition based oil change. It is also possible to detect wear debris of mechanical systems with a particle analysis[9].

- **Performance analysis** is an effective way of determining whether a machine is functioning correctly. Performance analysis monitors process parameters such as temperature, pressure, flow rate or processed items per hour[6].
- **Thermography** is used to detect hot spots in a system or a machine. At this time thermography is used principally in quasi-static situations.

Condition monitoring can be used for one sensor or for a complex system. Two approaches are used for monitoring a system: one-to-one and one-to-many[9]. In one-to-one monitoring a system parameter which is measured by a sensor is directly forwarded to a signal processing and a condition monitoring (see Figure 2.3) independent of the sub system that the parameter belongs to[9].

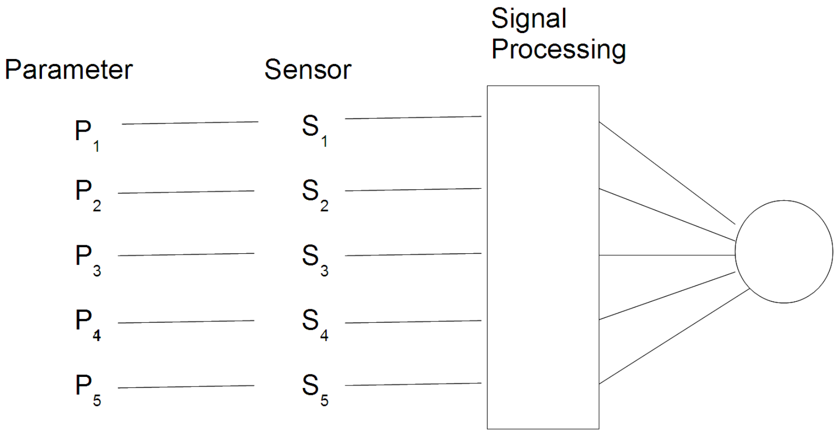

Figure 2.3 *One-to-one condition monitoring[9]*

In one-to many monitoring one sensor is used to give the condition monitoring information on more than one sub system (see Figure 2.4). One-to-many monitoring helps with failure location[9].

There are different methods for failure detection using condition monitoring. If only one sensor/parameter is evaluated then trend analysis

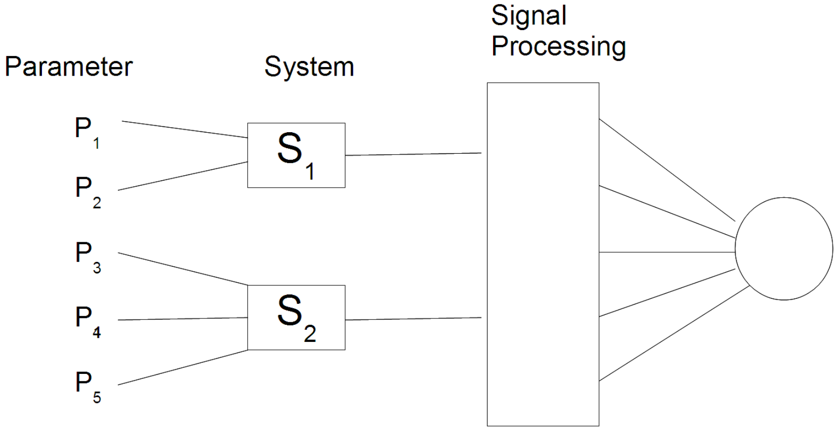

Figure 2.4 *One-to-many condition monitoring[9]*

or limits can be used[8]. Using a **limit** is to most simple method. The sensor signal is compared to a given limit. A failure has occurred if the sensor signal is greater than the given limit. A limit based failure detection can not be used to predict failure[8]. **Trend analysis** records a time series of the sensor signal. It can be assumed that the machine operates normal, if only small changes occur over time. A stronger change in the time series indicates the development of a failure. Trend analysis can be used for failure prediction[8].

If a system is monitored then a system model needs to be created (see Figure 2.5). The model is used to compare the actual system outputs to the theoretical outputs and a difference between both signals indicates an error or a upcoming error(Figure 2.6)[9]. A system can be modeled by a mathematical description through Laplace-based system models or through dynamic (statistical modeling)[9].

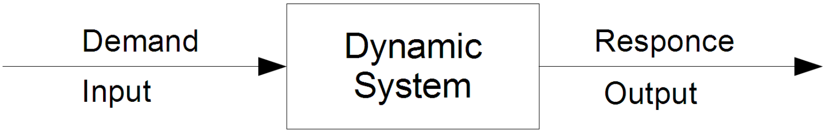

Figure 2.5 *System model[9]*

The **mathematical model** tries to describe the system in equations. A mathematical model can become quite complex but a complete definition of the system is often not needed[9]. **Laplace-based system**

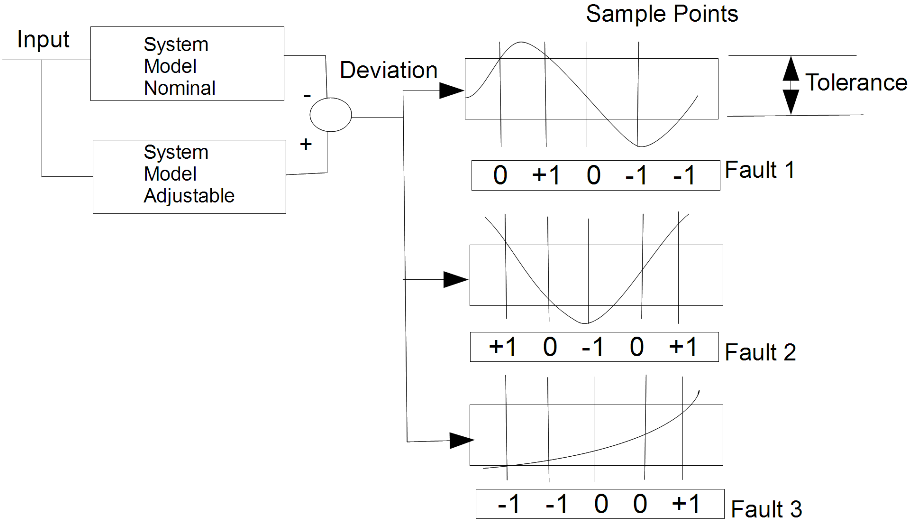

Figure 2.6 *Fault detection with a system model[9]*

models use the Laplace transformation to model a system with one or more building blocks (See Figure 2.7)[9]. System modeling and simulation tools like MATLAB Simulink, Modellica ... use Laplace like building blocks.

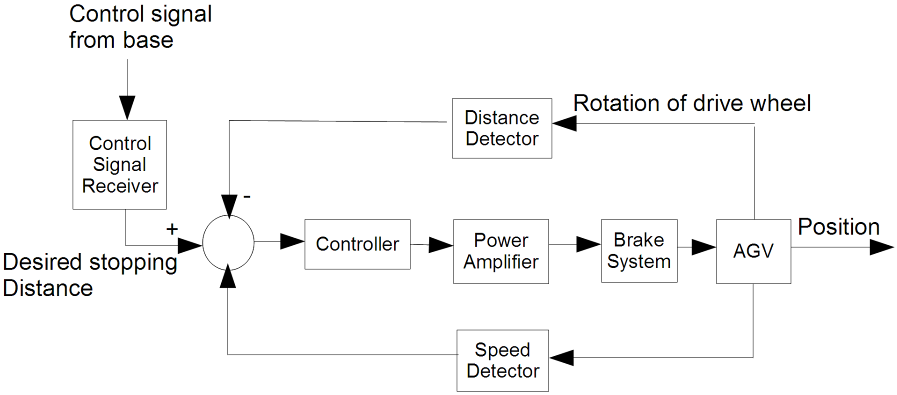

Figure 2.7 *Laplace based system model[9]*

Dynamic fingerprinting works without full knowledge of the system. The output for a given input is recorded and the collection of these records make the model[9]. **Outlier detection** uses different methods and techniques to detect an anomaly or a fault in sensor data. Often an outlier stands for a system fault[10].

Other methods for a system modeling and condition monitoring includes the use of neural networks and other machine learning techniques. Machine learning and pattern recognition is often used for condition monitoring and trending in complex systems[6][8]. In [11] is an example of such an approach shown. [12] uses a neural network for sensor fusion (one-to-many) while [13] an example of the one-to-many concept for distributed agents is. A real time monitoring with a neural network is shown in [14].

2.2 Signal Analysis

A signal is a vary quantity whose value can be measured and which conveys information[15]. Signals can represent sound, vibrations, color values, temperature ... There are two types of signals: analogue and digital. An analogue signal is a continuous signal and a digital signal does have a finite number of values. The process of transforming an analogue signal into a digital signal is called sampling. Sampling represents an analogue signal by a number of regular spaced measurements or samples[15]. Figure 2.8 shows the sampling of an analogue signal.

The number of regular spaced samples per second is the sampling rate and measured in Hz. A signal does have an amplitude and a phase. The amplitude is the sampling value and the phase measures the time delay this motion and another motion of the same speed[15]. Signals represented as above are in the time domain. It is possible to transform signals so, that they are represented in the frequency domain. In the frequency domain are the signal represented by cosine and sine function with different frequencies[15]. The process which converts the signal is called Fourier transform for analogue signals and discrete Fourier transform for digital signals. Equation 2.1 shows the discrete Fourier transform.

$$Z(f) = \sum_{k=0}^{N-1} z(k) e^{-2\pi j f k N} \tag{2.1}$$

$Z(f)$ is the Fourier coefficient at frequency f[15]. N is the total number of samples and k is the current sample. $z(k)$ is $x(k) + jy(k)$, where x and y are the amplitude and the phase of the signal. It is also possible

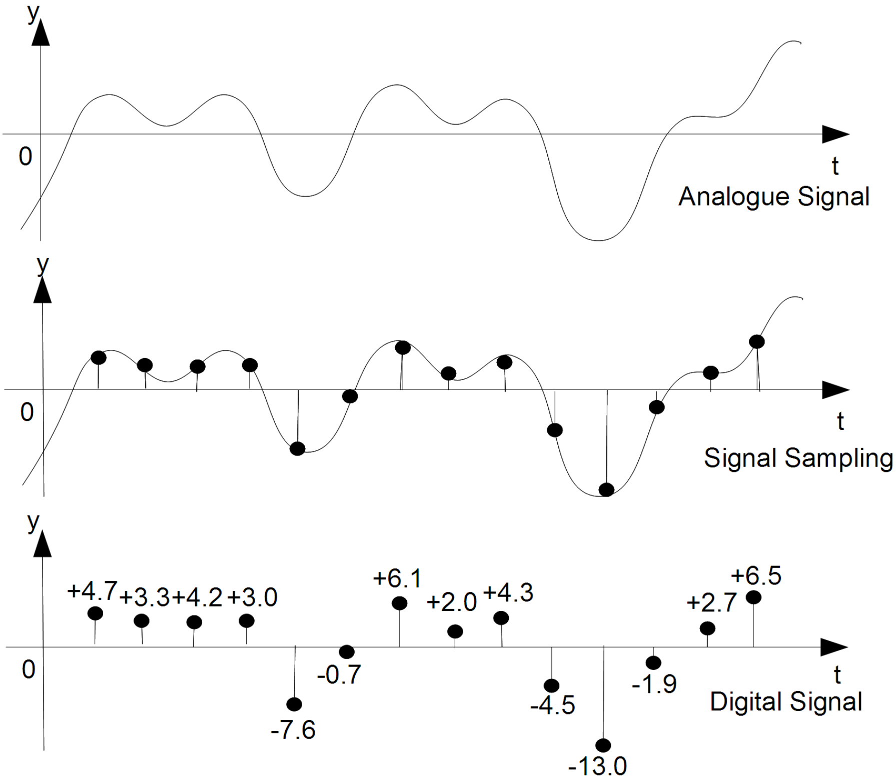

Figure 2.8 *Signal sampling[15]*

to reverse the transform by used Equation 2.2.

$$z(k) = \frac{1}{N}\sum_{f=0}^{N-1} Z(f)e^{2\pi jfkN} \tag{2.2}$$

It is also possible to treat the complex values as real values if the phase is unknown or zero. In this case of an only real valued signal only $N/2$ coefficients are independent. This is because $Z(N-f)$ and $Z(f)$ are the same if only the real part is considered. For practice this means that $2N$ samples are needed to get N Fourier coefficients. Figure 2.9 shows a real valued signal transformed into the frequency domain.

An algorithm to compute the discrete Fourier transform on a computer is called the fast Fourier transform (FFT). The FFT requires that N is a power of two[15].

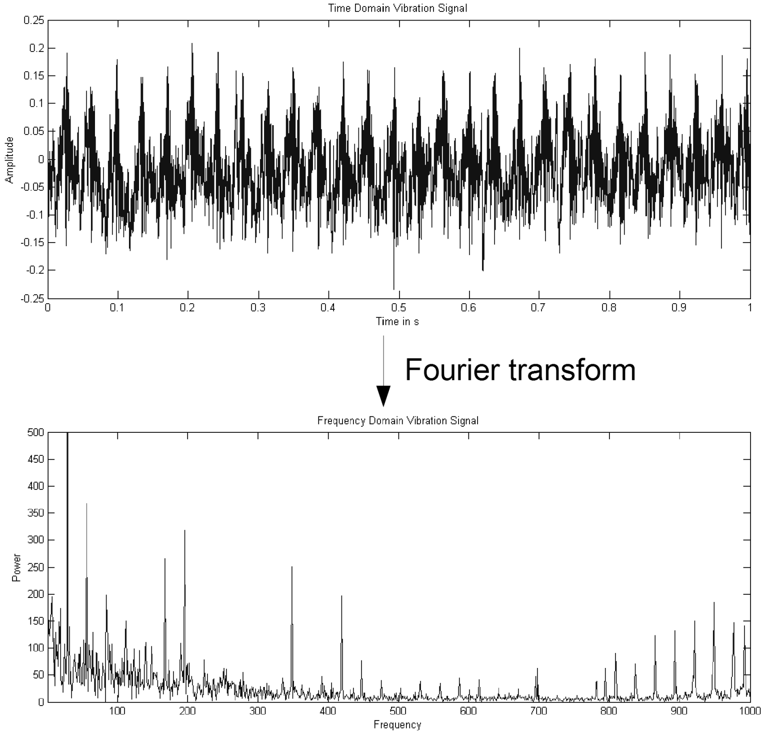

Figure 2.9 *Time domain to frequency domain*

A filter is a process which changes the shape of a signal[15]. Often filters change the signal in the frequency domain. Usual types of filters are low-pass, high-pass or band-pass filters. Low-pass filters keeps low frequency components of the signal and blocks high frequency components. A high-pass filter blocks low frequencies and keeps high frequencies. A band-pass filter blocks all but a given range of frequencies[15]. One way to apply a filter is to transform the time domain signal into the frequency domain, apply the filter and the transform the signal back into the time domain.

Band-pass filters can be used to extract frequency components from a signal into a new signal. If multiple band-pass filters are applied to a signal to extract different frequencies then the filter is called filter bank. The individual band-pass filters can either have the same size or the size can vary[16]. Figure 2.10 shows a filter bank with equal sized band-pass filters and Figure 2.11 shows a filter bank with band-pass filters of a different size.

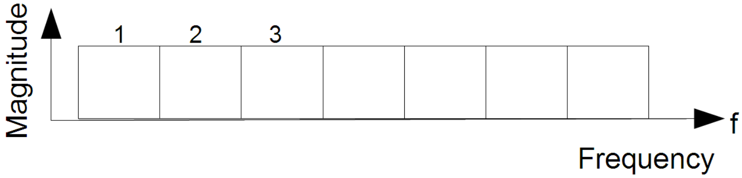

Figure 2.10 *Equal sized filter bank[16]*

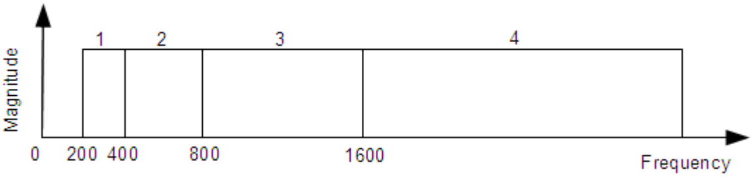

Figure 2.11 *Variable sized filter bank[16]*

2.3 Trend Series Analysis & Forecasting

A time series is a chronological sequence of observations on a particular variable[17]. This means that a time series is a number of data/time pairs that are ordered chronological. Figure 2.12 shows some time series. Time series analysis is done to discover historical patters, which can be used for forecasting[17]. Forecasting is defined as: Predictions of future events and conditions are called forecasts, and the act of making such a prediction is called forecasting[17]. Goal of forecasting is to reduce the risk of decision making[18].

Time series analysis and forecasting are used in many different areas from economic forecasting and logistic management to strategic management[17][19][18]. A time series is defined by[17][18]:

- Trend is the upward or downward movement of a time series over a period of time.

- Cycle refers to recurring up and down movements around trend levels.

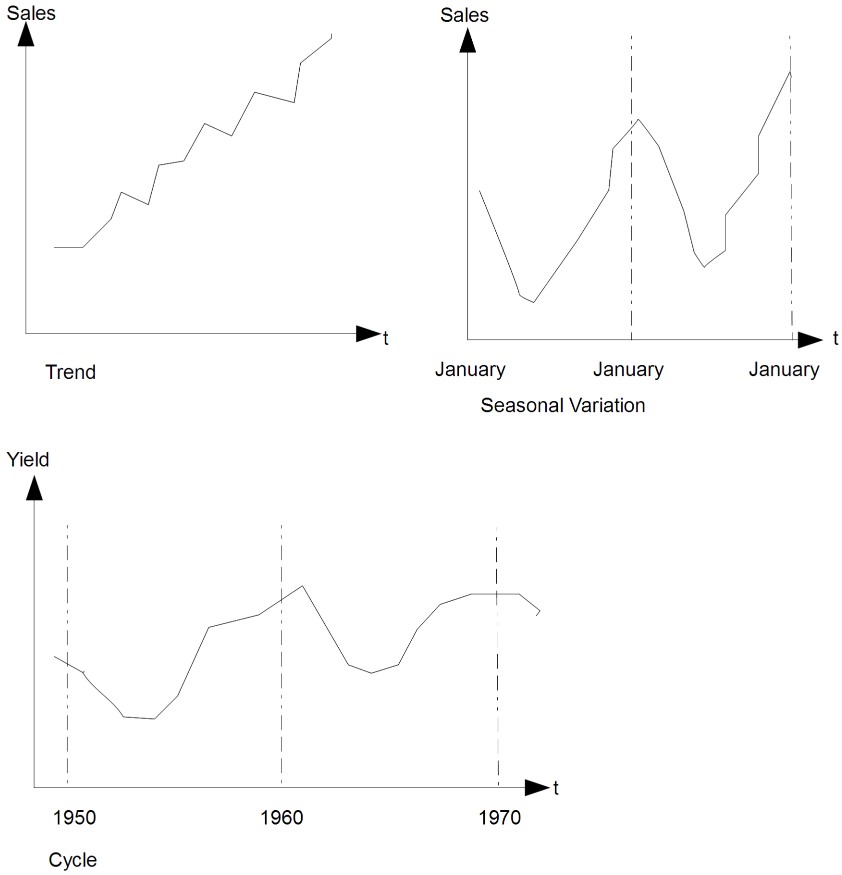

Figure 2.12 *Time series examples[17]*

- Seasonal variations are periodic patters that complete themselves in a calendar year.
- Irregular fluctuations are movements that follow no patters.

Time series can be split up in two categories: continuous and discrete. A continuous time series is recorded at all the time, while a discrete time series is recorded at given intervals (hourly, daily ...)[19]. Time series forecasting can be influenced by many factors like the availability of data, cost of analysis or management preferences[18]. Forecasting is defined by[18]:

- Forecasting period is the basic unit of time for which forecasts are made (hours, days, weeks ...).

- Forecasting horizon is the number of periods in the future covered by the forecast.
- Forecasting interval is the frequency in which forecasts are made.

Often the forecasting interval is the same as the forecasting period. This means that the forecasting is revised after each period[18]. Two types of forecasts can be made: expected value int he future and prediction interval[18][17]. The prediction interval is an interval in that has a stated chance of containing the future value. Usually two forecasting strategies are available: qualitative and quantitative methods[17][18]. Qualitative methods involve an expert while quantitative methods analysis the historical observations to predict the future. Bases for the forecasting is to develop a model of the historical data. The model can be based on a single time series (uni-variant model) or it can include multiple variables (causal model)[17][19][18]. Figure 2.13 shows a simple sample model of a time series.

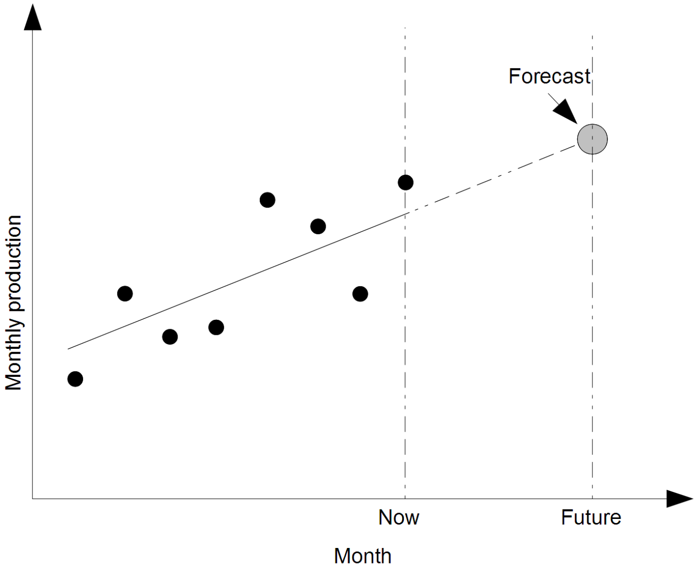

Figure 2.13 *A linear model of a time series[18]*

Different methods are available for quantitative forecasting[17][19][18]:

- Simple Linear Regression

- Multiple Regression
- Moving Average Model
- Exponential Smoothing
- Box-Jenkins

Each of the five methods will be explained in this section. Simple linear regression and multiple regression methods can be used to calculate a trend in a time series[17].

2.3.1 Simple Linear Regression

The most simple regression method is simple linear regression. Goal of the simple linear regression is to model the time series with a single straight line (Figure 2.13)[17][18]. The model does have two parameters: the slope and the y-intercept. The model can be written as:

$$y = b_0 + b_1 x + \epsilon \tag{2.3}$$

A usual method to estimate the two parameters b_0 and b_1 is to use least-squares[17][18]. Least-squares tries to find parameters for which the error sum of squares is the least. This means the sum of the squared error between the line and the point y_i The error sum can be written as:

$$\ell(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2 \tag{2.4}$$

The complete equation for the calculation of b_0 and b_1 is[17][18]:

$$b_1 = \frac{n \sum\limits_{i=1}^{n} x_i y_i - \left(\sum\limits_{i=1}^{n} x_i \right) \left(\sum\limits_{i=1}^{n} y_i \right)}{n \sum\limits_{i=1}^{n} x_i^2 - \left(\sum\limits_{i=1}^{n} x_i \right)^2} \tag{2.5}$$

and $b_0 = \bar{y} - b_1 \bar{x}$
where $\bar{y} = \frac{\sum\limits_{i=1}^{n} y_i}{n}$ and $\bar{x} = \frac{\sum\limits_{i=1}^{n} x_i}{n}$
The fitted simple linear regression model is:

$$\hat{y} = \hat{b_0} + \hat{b_1} z \tag{2.6}$$

2.3.2 Multiple Regression

Multiple regression is similar to simple linear regression, but the regression depends on more than one variable (see Equation 2.7)[17][18].

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + \epsilon \tag{2.7}$$

The variables $x_1, \ldots, x_n$ can be different functions of time like $x_1 = x^2$[18]. $x_1, \ldots, x_n$ may also be other time series like temperature and sales which may influence a time series. Equation 2.8 shows an example.

$$y = b_0 + b_1 w(t) + b_2 s(t) \tag{2.8}$$

where $w(t)$ is a function over time like the weight of a human over time and $s(t)$ is also a function over time like the salary. 2^{nd} order or higher order polynomial models can be used[17]. Figure 2.14 shows some 2^{nd} order functions.

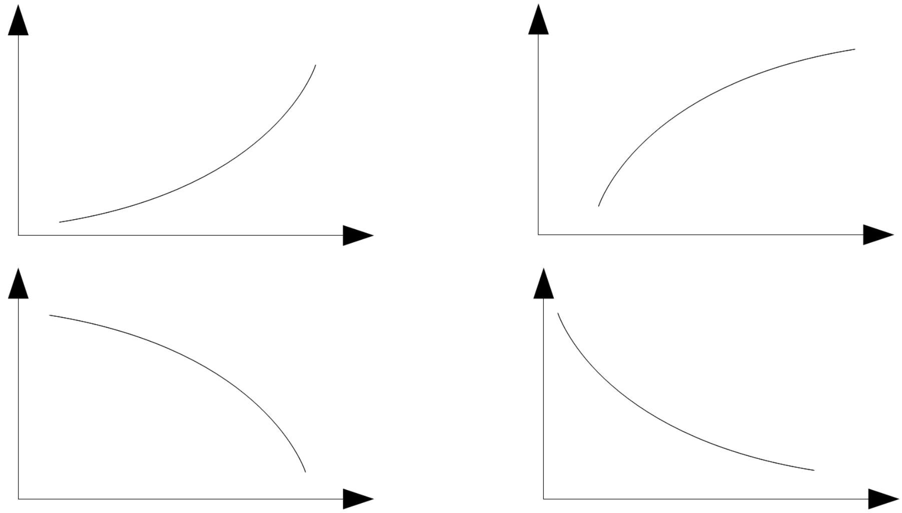

Figure 2.14 2^{nd} *order polynominal models* ($y = b_0 + b_1 x + b_2 x^2$)*[17]*

The general representation of p^{th} order polynomial model is:

$$y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots + b_p x^p + \epsilon \tag{2.9}$$

Multiple regression also uses the least-squares method to calculate the parameters $b_0, \ldots, b_n$. The least squares problem is often describes in a matrix form like in Equation 2.11[18]. The problem as a matrix is defined as[18]:

$$\hat{y} = \mathbb{Z}\hat{b} \tag{2.10}$$

$$\begin{pmatrix} 13 \\ 20 \\ 5 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 12 & 4 \\ 19 & 34 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \tag{2.11}$$

The normal equations can be used to solve the least-squares problem in a simple way (Equation 2.12)[18].

$$\hat{b} = (\mathbb{Z}'\mathbb{Z})^{-1}\mathbb{Z}'y \tag{2.12}$$

Normal equations is not the most stable method to solve the problem. The QR factorization solves the problem in a more stable way[20].

2.3.3 Moving Average Model

The moving average model can be seen as a more simple form of the simple linear regression model. Not the complete time series is evaluated, but only N points of the time series[18]. The moving average model can be seen as a way to reduce the noise in a time series. In the most simple case is the y_i the average (arithmetic mean) of the last N values[18]. The equation for the moving average model is:

The moving average model can also be used to forecast a trend by using the following equation[18].:

$$M_\tau = \frac{y_\tau + y_{\tau-1} + y_{\tau-2} + \cdots + y_{\tau-N+1}}{N} \tag{2.13}$$

The equation calculates a forecast of τ periods into the future[18].

$$\hat{y}_{T+\tau}(T) = 2M_T - M_T^{[2]} + \tau \left(\frac{2}{N-1} \right) (M_T - M_T^{[2]}) \tag{2.14}$$

where $M_T^{[2]}$ is a second-order statistic (moving average of the moving averages):

$$M_T^{[2]} = \frac{M_T + M_{T-1} + \cdots + M_{T-N+1}}{N} \tag{2.15}$$

2.3.4 Exponential Smoothing

Exponential smoothing is a method for smoothing similar to the moving average model. The difference is that the data points are weighted unequal. The most recent data point is weighted more then past data points[17][18]. The equation for a simple exponential smoothing is:

$$S_T = \alpha x_T + (1 - \alpha) S_{T-1} \tag{2.16}$$

S_T is a weighted average of all past observations. To define an exponential smoothing that is an n-period moving average α is set to[18]:

$$\alpha = \frac{2}{N+1} \tag{2.17}$$

The starting value S_0 can be gained by taking the average of certain number of past data points or to choose it[17][18]. The forecast for the time period $T+1$ is S_T[18]. A low value of α causes the forecast to weight the last value more and makes the forecast react faster to changes but also to noise. A low values of α lets the forecast react more slowly.

2.3.5 Box-Jenkins

The Box-Jenkins methodology was developed by Box and Jenkins in 1976. The methodology consists of a four step iterative procedure[17]:

1. Tentative identification: historical data are used to tentatively find an appropriate Box-Jenkins model.
2. Estimations: historical data are used to estimate the parameters of the tentatively identified model.
3. Diagnostic checking: various diagnostics are used to check the adequacy of the tentatively identified model and, if need be, to suggest an improved model, which is then regarded as a new tentatively identified model.
4. Forecasting: once a final model is obtained, it is used to forecast future time series values.

Box-Jenkins models are the autoregressive model, moving averages model, autoregressive-moving average model and autoregressive integrated moving average mode. **Autoregressive processes** use weighted past data to predict a future value. A white noise signal (fixed variance and mean zero) with a defined variance (that is the same for each period t) and is added to the past data[19]. The autoregressive process is defined as[18]:

$$x_t = \xi + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \epsilon_t \tag{2.18}$$

where ϵ_t is white noise, ξ is a constant and $\phi_1, \ldots, \phi_p$ are parameters (weights) of the model. The random shock ϵ_t describes the effect of all other factors than $x_{t-1}, \ldots, x_{t-p}$ on x_t[17]. An autoregressive process of the order p is call AR(p)[18]. Where p is the number of past data points. Autoregressive processes use the fact that the value of the time series are correlated[17]. Related to the autoregressive processes are the **moving average processes**. The moving average process is defined as[18]:

$$x_t = \mu + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \cdots - \theta_q \epsilon_{1-q} \tag{2.19}$$

where μ is the mean of the time series, $\epsilon_t, \ldots, \epsilon_{1-q}$ are random shocks and $\theta_1, \ldots, \theta_q$ are a finite set of weights. A moving average process of order q is called MA(q). The random shocks for the moving average process are also white-noise random shocks, this means that they have the mean zero, have a normal distribution and are defined by the variance. It is possible to combine the autoregressive process and the moving average process. The combined model is called **autoregressive-moving average (ARMA)** and does have two orders p,q or ARMA(p,q)[18]. ARMA models can only represent stationary time series[18]. A time series is stationary if the statistical properties (for example mean and variance) of the time series are essentially constant through time[17].

It is possible to convert a non-stationary process into a stationary process by calculating the differences between two successive values. The first differences of the time series values $y_1, y_2, \ldots, y_n$ are[17]:

$$z_t = y_t - y_{t-1} \tag{2.20}$$

where $t = 2, \ldots, n$

Taking only one difference is called first differences. If the first differences is also no stationary process then it is possible to take again the differences and get a second differencing. Figure 2.15 shows a second differing[18].

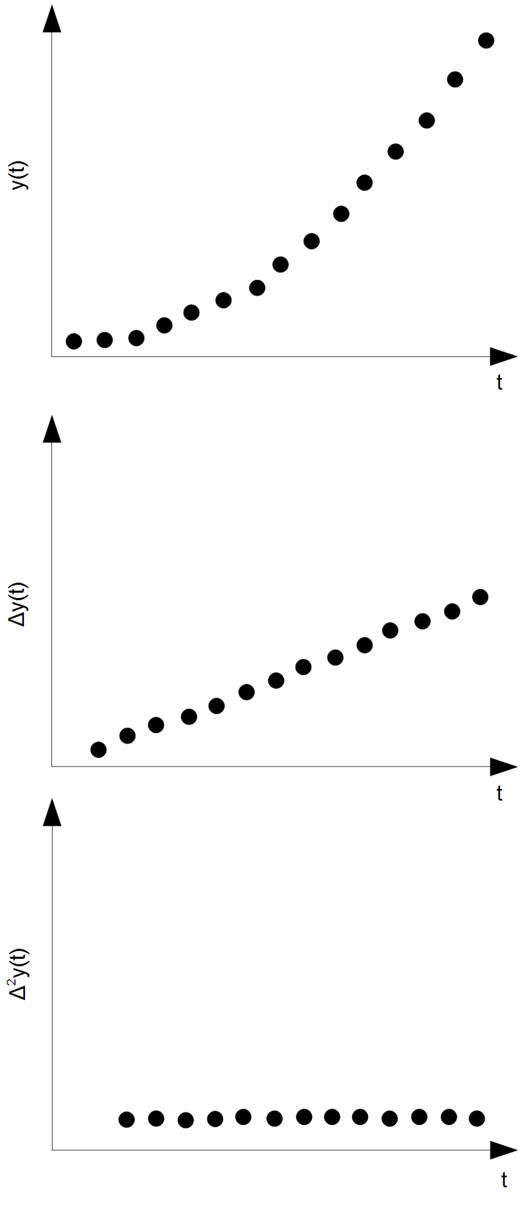

Figure 2.15 *Differencing a time series two times[18]*

Autoregressive integrated moving average (ARIMA) models use the d^{th} difference to model non-stationary time series. An ARIMA model does have an order of (p,d,q) where d is the d^{th} difference of the original series[18].

2.3.6 Other methods

There are many different approaches available for modeling and forecasting a time series. [21] uses an artificial neural network to forecast stock prices. [22] uses piecewise linear approximation to detect trend in a streaming time series. Bayesian and other probability methods are also used to model and forecast a time series[23][18].

2.4 Decision Trees

A decision tree is a tool from the artificial intelligence area. A decision tree is a tree which classifies instances by sorting them down from the root to some leaf node[24]. Each node in the tree specifies a test on an attribute and each branch from to node to another node or leaf correspondences to a test result[24]. An example decision tree is shown in Figure 2.16. The example decision tree classifies the weather if it is suitable to play tennis or not.

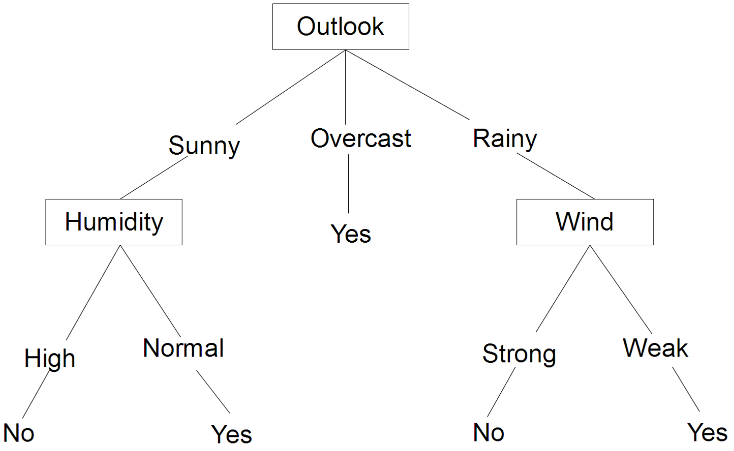

Figure 2.16 *Decision tree example[24]*

If the decision tree is used to learn a discrete valued function (like the example) it performs a classification. If the tree is used to learn a continuous function it performs a regression[25]. Any decision tree can be converted into a logical expression[25]. The example can be expressed as:

$$\begin{aligned} &(Outlook = sunny \wedge Humidity = normal) \\ &\vee (Outlook = overcast) \\ &\vee (Outlook = rain \wedge Wind = weak) \end{aligned} \tag{2.21}$$

An instance to test consist of attribute value pairs. Each instance is described by a fixed set of attributes (e.g. Outlook) and their values (e.g. Sunny). Decision tree learning is based on a number of provided samples which specify the problem. The set of examples is called training set. There are different algorithms to learn a decision tree. The basic decision tree learning algorithm works as followed[25].:

1. Create a new node
2. Split the examples based on the values of the best attribute for splitting.
3. Check for each value of the attribute:
   (a) If the remaining examples have a different classification, then choose the best attribute to split them and create a new child node.
   (b) If all remaining examples have the same classification then the tree is trained. It is possible to make a final classification. Create a leaf.
   (c) If there are no examples left, it mean that no such example has been observed.

There is an error in the training examples, if two or more examples have the same attribute values but different classifications. In this case it is possible to return the classification of the majority of the classifications or to report the probability for each classification[25]. A common a method for selecting the best attribute to split the examples is the ID3 and the C4.5 by Quinlain[24]. The idea of ID3 is to select a node based on the information gain. Information needs to be defined to define information gain and understand the concepts. 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[25]. The information entropy is also called information and is calculated as shown below in Equation 2.22. $P(v_i)$ is the probability of the answer v_i.

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) log_2 P(v_i) \tag{2.22}$$

The information gain from an attribute test (setting the value of a node in a tree, see Figure 2.16 for an example) 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[25].

$$\begin{aligned} Gain(X) = {} & I(\frac{p}{p+n}, \frac{n}{p+n}) \\ & - \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \cdot I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}) \end{aligned} \tag{2.23}$$

The performance of a decision tree can be tested with a number of test examples. Test examples are examples from the training data, which were not used for the learning. Performance of the decision tree depends on the number of correct classified examples.

A common problem for decision trees is over fitting if noise is in the training data or when the number of training examples is to small[24]. A model has a bad performance with testing data if it is over fitted. A simple method to remove over fitting is decision tree pruning. Pruning works by preventing recursive splitting on attributes that are not clearly relevant[25]. Pruning means to remove a sub-tree from the decision tree. Information gain can be used name irrelevant attributes. Another possibility to reduced over-fitting is cross-validation. In cross validation multiple decision tree are trained each with a different set oft raining and testing examples. The decision tree with the best performance is chosen. A K-fold-cross-validation means that k different decision tree are trained and each is tested with a different set $1/k$ of the examples[25].

Decision trees can be extended to handle the following cases[25]:

- Missing data: not all attribute values are known for all examples.

- Multivalued attributes: The usefulness of an attribute might be low i an attribute does have many different possible values (Name or credit card data).
- Continuous and integer-valued input attributes: numerical attributes often have an infinite number of possible values. A decision tree typically chooses a split point that separates the values into groups (e.g. Weight $<$ 160).
- Continuous-valued output attributes: the tree does have at the leafs a linear function rather then a single value (regression tree).

Another class of decision trees are fuzzy decision trees. Fuzzy decision trees are not based on crisp training data, but on fuzzy training data. [26][27][28] have examples of fuzzy decision tree training and uses of fuzzy decision trees.

2.5 Local search and optimization

Local search is a special area of search algorithms. In many cases the search algorithm does have a memory of the way to the solution. This means the algorithm knows which steps it took. Local search algorithms have no memory and know only the current state. It might be possible that they check a member of the search space twice. Local search algorithms do not search systematically[25]. Hill climbing search (greedy local search), simulated annealing or a genetic algorithm are all local search algorithms.

Local search algorithms can not only be used to find a goal but also for pure optimization problems. Local search algorithms work in a state space landscape (Figure 2.17). Each state does have a corresponding location and the elevation of the state/location is the value of the heuristic cost function. Goal is to find the state/location with the lowest elevation(costs). It is also possible to find the highest peak if the elevation is not the costs[25].

The hill-climbing algorithm is a simple loop that moves in the direction of the increases value. Hill climbing only evaluates the neighbor states and then choses the best of those. For this reason hill-climbing is sometimes called greedy local search. Hill-climbing can get stuck fast, because it makes no downhill moves and stays on a plateau or a local maxima[25].

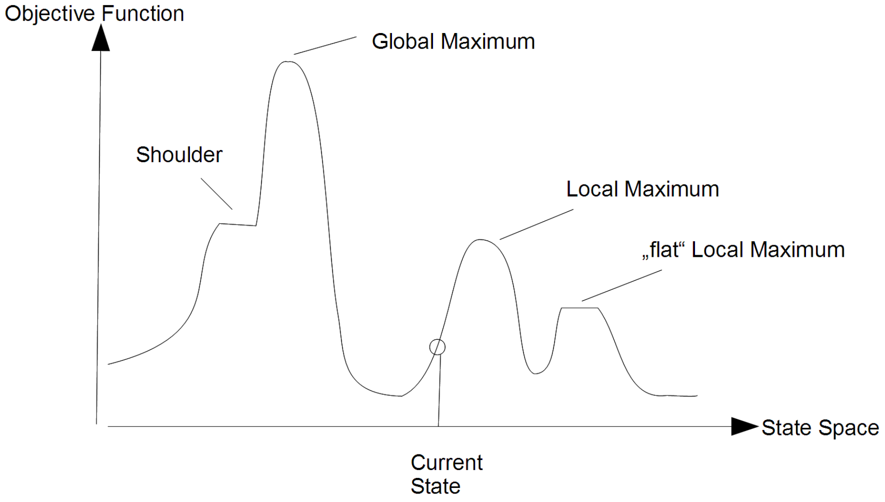

Figure 2.17 *Hill-climbing example[25]*

Simulated annealing is a hill-climbing algorithm that can move downwards. The algorithm is based on the annealing process in metallurgy. The metal gets into a fixed state as it cools down. The simulated annealing algorithm selects a random move and if it improves the the situation it is accented. If not then the move is accepted based on a probability value. The probability decrease exponential with the of the move. The probability also goes goes with each step[25].

A genetic algorithm not only keeps one state in memory but more than one. The states in memory are called population. Turing each step new states (individual) are calculated based on the current population. The first population is generated randomly. New individuals are calculated through crossover and mutation. In cross over two individuals are chosen from the population based on their fitness. Then two new individuals are created by taking a part of one parent and another part of the other parent. Thus the new individual is created by having a part of both parents. The second child is constructed out of the not selected parts of both parents. Mutation modifies each individual based on an independent probability. Figure 2.18 shows an example of a genetic algorithm. The children form the new population[25][24]. [29] use genetic algorithms to adapt approximation functions from old problems to new problems and [30] use genetic algorithms to select features for decision trees.

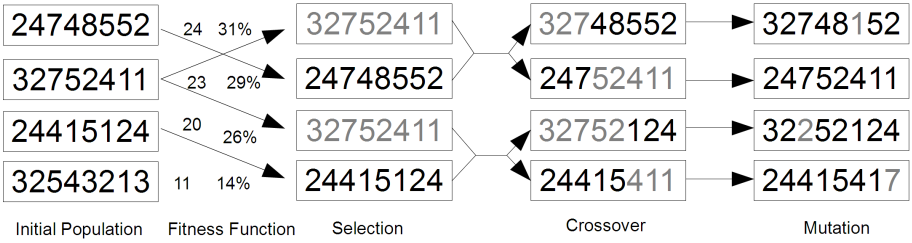

Figure 2.18 *Genetic algorithm example[25]*

2.6 Literature Review

This subsection takes an evaluation of different publications that had an influence on this work.

2.6.1 Look-Ahead Based Fuzzy Decision Tree Induction

Look-Ahead based Fuzzy decision tree induction is a new method to create a decision tree. The method does not use the greed top-down method, but uses a look-ahead method. A look ahead decision tree induction takes a node and calculates the split. Then each branch is again evaluated. Thus the algorithm is looking ahead. However Look-Ahead algorithms don't always produce the best results. The paper offers a method to increase the results of Look-Ahead algorithms and proves this with experimental results[28].

2.6.2 Fuzzy Decision Tree for Data Mining of Time Series Stock Market Databases

The paper proposes a method for a new fuzzy decision tree to classify stock trading time series. Example stock market data is used through the paper to show example of how to construct the fuzzy decision tree. However the experimental result section is very short[27].

2.6.3 Induction of Decision Trees

This paper describes the ID3 decision tree induction algorithm by Quinlan. ID3 decision trees can deal with incomplete or noisy data. Quinlan explains his ID3 algorithm and uses many examples to show how the algorithm works. The paper concludes with a conclusion. This paper was the bases for the decision tree algorithms that were used in this thesis and the experiments[31].

2.6.4 A complete fuzzy decision tree technique

This paper presents a method call "soft decision trees" (SDT). Soft decision trees output a numerical value instead of a crisp decision and share similarities with regression trees. Soft decision calculate the membership of an object to a class, similar like the fuzzy decision tree evaluation concept in this thesis. Soft decision trees however use fuzzy input values[26].

2.6.5 Intelligent stock trading system by turning point confirming and probabilistic reasoning

This paper talks about the application of probabilistic models for stock market systems. A Markov Network is used to detect turning points in the stock data based on different indicators. The paper shows the usability of the concept with experiments using real world data[23].

2.6.6 Continuous Trend-Based Classification of Streaming Time Series

Streaming time series are time series where new values arrive and a new trend has to be calculated every time. This is the typically the case for online monitoring. The paper tries to classify trends for streaming time series. An algorithm is given and the performance is analyzed using stock market data and Tropical Atmosphere Ocean data. Classifying streaming time series was also a problem in this thesis[22].

2.6.7 A Neural Stock Price Predictor using Quantitative Data

The paper describes the usage of a neural network to predict stock market prices. In addition to only calculating the predicition the method in this paper also calculates the accuracy of the forecast[21].

2.6.8 A Neural Network Approach to Condition Based Maintenance: Case Study of Airport Ground Transportation Vehicles

This paper describes a concept for monitoring different types of automated vehicle doors. A neural network is used to learn the characteristics of the different types of doors and to predict the condition. The concept was evaluated using real world data in an experiment. Three different types of neural network paradigms were evaluated for the used neural network. The paper closes with an interesting discussion about the learned lessons. The topic of the research and the approach for the condition monitoring were similar to the approach taken in this thesis[14].

2.6.9 A Multi-Agent Fault Detection System for Wind Turbine Defect Recognition and Diagnosis

Anomaly detection and data trending are used to detect faults in a distributed environment. The paper proposes an architecture that uses multiple agents to detect a system state. The application does have many similarities to the application in this thesis. Only the focus is on detecting and not on forecasting failures[13].

2.6.10 Sensor fusion of a railway bridge load test using neural networks

This publication uses neural networks to detect the condition of a bridge. An neural network is used to model the input and output relation of finite elements model. The bridge was monitored by multiple sensors. Data from those sensors was fused in the neural network using the data as input to the network. The results of the paper were that a neural network can be used to represent a finite elements model under certain constrains and thus can be used to monitor structures[12].

2.6.11 Automated Rule Extraction for Engine Vibration Analysis

Health monitoring and detection and classification of possible failures is the topic of this paper. The goal is to develop an automatic system for extracting rules for health monitoring using evolutionary programming. Neural networks and evolutionary programming are used to extract rules. With the rules a decision tree is then constructed[11].

2.6.12 A survey of outlier detection methodologies

This publication describes and evaluates different outlier detection methods. It starts with an introduction to the topic of outlier detection and defining the different types of outliers. The section is followed with a list and description of different methods beginning with statistical methods and parametric methods. The publication also includes the discussion of using neural networks and decision trees for outlier detection. Each method contains a note for what kind of outlier it is usable[10].

2.6.13 Using Genetic Algorithms for Adapting Approximation Functions

This short paper explains how genetic algorithms can be used to modify the parameters of an approximation problem or for selecting a new approximation function[29].

2.6.14 Decision tree classifier for network intrusion detection with GA-based feature selection

This paper proposes a method for feature selection for a decision tree based on a genetic algorithm. The decision tree is then used for intrusion detection of computer systems. The method used in this paper is very similar to the one used in this thesis however it is slightly differently used for a different application[30].

3 Condition Monitoring

The condition monitoring process consists of two sub-processes; a training process and a decision process. In the training process the decision tree for the given problem is created. The decision process uses the created decision tree to decide in which condition the system currently is. Goal of the decision monitoring process is to have an adaptable process that can be used to calculate the current condition of a monitored system. This is done with a decision tree trained on the problem. Figure 3.1 shows an activity diagram of the process.

The left side of the diagram shows the decision process and the right side shows the training process. When a new data sample is available it is checked, if it is a training sample (no decision tree yet created) or if it should be evaluated with the decision tree. The process is simple. If the data sample should be evaluated, the data is preprocessed based on parameters and the processed data is then evaluated with the decision tree. Output is a system condition. If the new data is training data then it is labeled/classified and added to the sample data base. The decision tree is trained when enough training samples are available. Samples are divided into training and testing samples. The training and testing samples are preprocessed based on initial selected parameters. The decision tree is calculated using any decision tree algorithm e.g. ID3 or C4.5 and the training samples.

The decision tree is tested with the testing sample set. If the performance of the decision tree is below a given limit, then the preprocessing parameters are changed, based on a heuristic optimization algorithm.

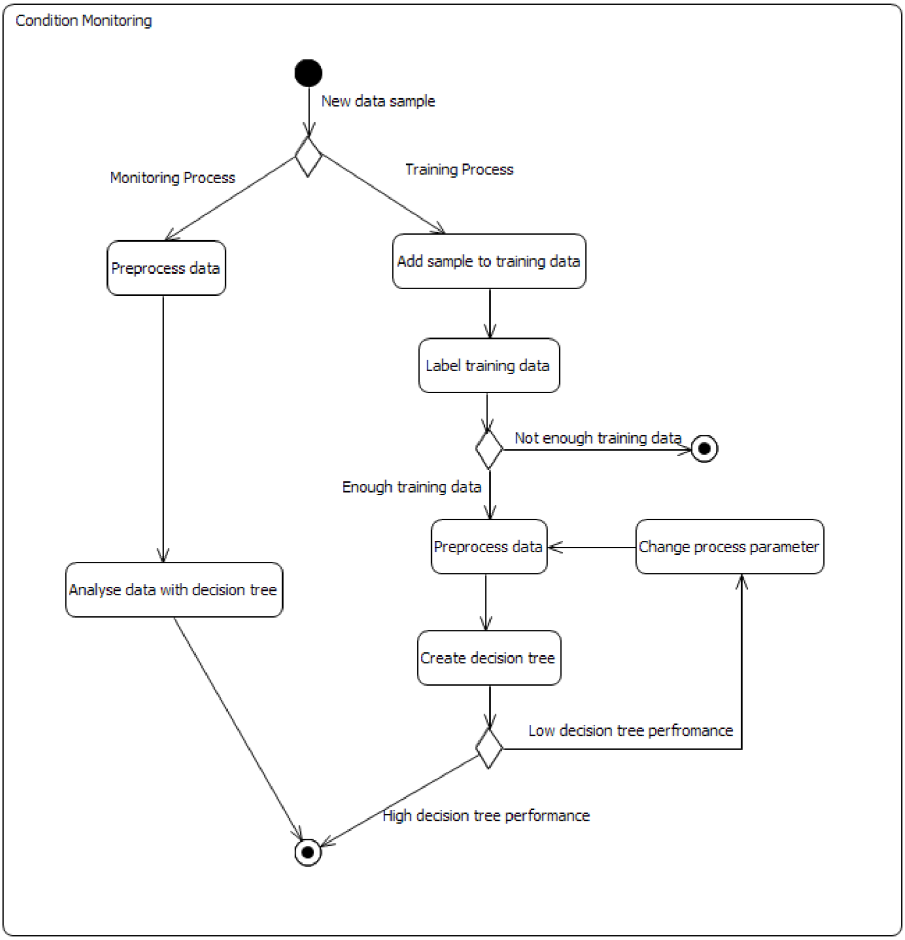

Figure 3.1 *Condition monitoring process*

This algorithm can be e.g. Greedy Search, Simulated Annealing or Genetic Algorithm. A new decision tree is calculated based on the modified parameters. The new decision is again tested and compared against a limit. This optimization process is repeated until a given number of decision trees have been calculated or until a decision tree has a better performance than the given limit. The decision tree and the data preprocessing parameters are saved.

Evaluation of a new data sample is a simple task. First the data is processed depending on the preprocessing parameters from the training process. After the data sample is preprocessed it is inputted into the decision tree. Evaluation can be done in two ways: usual decision tree induction or fuzzy decision tree evaluation[4]). Multiple decision trees with different preprocessing parameters can be calculated and used to improve the performance and reduce the noise sensitivity. If multiple decision trees are used then the condition that the majority of the decision trees select is taken.

3.1 Data Requirements

To use the proposed concept a few requirements need to be fulfilled.

Stable System A stable system behavior is needed to use the condition prediction. This means that the system should not change its condition frequently. Reason for this restriction is that the condition should not change during the sampling of samples. If the condition changes during the recording then the classifier cannot classify the current state correctly.

Continuous Sensor Values The condition monitoring was not developed with discreet values and events in mind. The system needs values which change over time. Most physical system fit these criteria unless their operation is triggered by a highly unpredictable external source. E.g. accelerating a car is a continuous time series, while starting the car or closing a door is not.

High Frequency Sensor Data The system works best if the sensor input changes frequently during sampling of a sample (e.g. sound, vibration, power consumption ...). Most data preprocessing steps are developed to work with a signal input with more than 1 Hz. But it is possible to work also with slow changing values like temperature, even if less information can be extracted from such data.

Periodical Data Sampling The concept relies on data samples which are collected during given intervals and not at discrete or random times. This requirement is less for the condition monitoring needed but it is required for the trend prediction to be able to make a forecast. No time information is saved in the recorded data.

Discreet Data Sampling Condition monitoring and condition prediction work with fixed sample lengths. This means that it is not possible to record continuous data samples without modifying the algorithm and splitting the continuous data sample into multiple one second samples.

No True Real Time Monitoring Calculation of a condition needs some time. The calculation of one condition may take as much as ten seconds, depending on the number of the used decision trees and the number of the sensors. If only one tree is used and only a few FFTs need to be calculated then it is possible to have a calculation time of less than one second. This was the usual case during the experiments. If the calculation time is less than one second then it is possible to calculate the condition while a new data sample is recorded.

Multiple Conditions Data samples of more than one condition are needed for learning patterns and classification. With the concept it is not possible to have a "One-Class" classifier. A One-Class classifier is a classifier, which detects incorrect states and conditions based only on the data of the usual operation. Condition monitoring always needs at least data of two different operation modes or conditions. It is possible to change the concept to find incorrect conditions, but this would require a significant change.

The following subsections detail each step of the process.

3.2 Training Process

The training process explained in this section calculates one optimized decision tree and a set of optimal preprocessing parameters. The process (except data sampling and labeling) needs to be executed multiple times if more than one decision tree needs to be calculated.

3.2.1 Data samples

The condition monitoring concept uses model based on statistical data to classify new data. Therefore a lot data samples are needed, before the "real" training process can be started. Multiple samples for each different class of the system are needed. How much training data is needed is not easy to say. It depends on the complexity of the system[3]. Getting enough data and useful data is a difficult process. For new systems it is possible to collect data during system testing and prototyping. Data collecting for older systems is possible if multiple systems are available and data recording can be performed in parallel.

System data can come from many different sources. These sources can be from internal sensors and/or external sensors. In the experiments (Section 6) external sensors were used. The sensor signals were recorded for one second each minute. Different time intervals are possible depending on the dynamic of the system. If the system changes fast then a higher sample frequency is needed. If the system does have a slow dynamic, then a lower sampling frequency can be used.

The proposed concept can work with any kind of input data; however it is assumed that a data sample is a discrete signal with more than one data point. A sample length of one second is enough for most system to extract information, longer sampling periods allow the calculation of frequencies smaller than 1 Hz. For most use cases it is enough to get one data sample every ten minutes during operation or cruise flight. A sensor sampling frequency of higher than 100 Hz is recommended. If a lower frequency is used, then the preprocessing needs to be adapted to that frequency. The signal source does not matter, it can be sound, vibration, temperature, power consumption, weight or magnetic flow data as long as it is a one-dimensional time series source (see Figure 3.2). If more than one data source is used or a data sample does have more than one dimension, then the preprocessing algorithm also needs to be adapted or the data needs to be transformed. The simplest way is to have one preprocessing step for each dimension of the data and then concatenate the preprocessed data before giving it to the pattern recognition. Each used data sample needs to be classified/labeled by a supervisor.

3.2.2 Data Labeling

An important part of the training process is to classify each data sample. The learning algorithm needs all training samples to have a number of features and a classification. The classification of the data samples should describe the condition of the system that the data sample stands for. Some possible classifications for sensor data samples can be:

Life time Each system does have a life time after which it needs to be replaced. Life time can be measured in operating hours. If life time

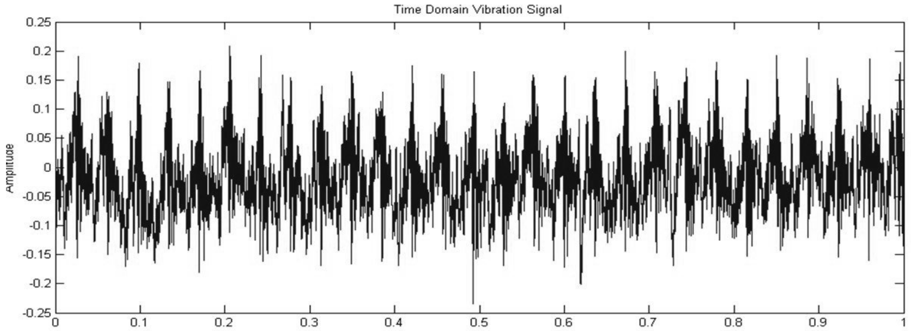

Figure 3.2 *A data sample*

should be used as a condition it is often useful to use the passed life time or the remaining life time of the system. Life time should be represented as a percentage value or in blocks to prevent too many different classes. More classes slow the training and make the system more sensible to noise (over-fitting)[31].

System mode System mode means if the system is in normal operation or if some failure occurred. This classification is useful to detect failures in a system.

A good classification can influence the performance of the condition monitoring significantly. Many or very specific classes may cause overfitting and makes the system sensible to noise.

3.2.3 Preprocessing

Signal analysis and machine learning are used to detect the condition of the system. For learning and classification the data samples need to be prepared[2]. The process depends on different parameters. Each of these parameters needs to be adapted to the data. In this concept the selection of the optimal parameters is performed by a genetic algorithm (Section 2.5). These parameters include:

- Signal transformation from the time domain into the frequency domain
- Noise reduction
- Grouping of frequencies

- Calculation of the maximum and mean frequency power of every frequency group
- Calculation of the number of peaks of all groups
- Transformation of the frequency groups back into the time domain
- Calculation of the maximum and mean amplitudes
- Calculation of the maximum and mean values of the complete signal

Noise and the amount of data are reduced and extra information is added during preprocessing to the data. First, the data is transformed into the frequency domain, where the noise is reduced. Then, frequencies are grouped. It is possible that the frequency span of the groups overlap each other. E.g. if the frequencies 1 to 50 belong to one group and have an overlap of 50 %, then the second group contains the frequencies from 26 to 75 and the third group contains the frequencies from 51 to 100. Mean and maximum powers are calculated for each frequency group, as well as the number of peaks. Then each group is transformed back into the time domain, where the mean and maximum amplitudes are calculated. The mean and maximum frequency power and mean and maximum amplitude of the complete signal is calculated as a last step. Table 3.1 shows the parameters of the preprocessing and the possible values. Figure 3.3 shows the preprocessing steps.

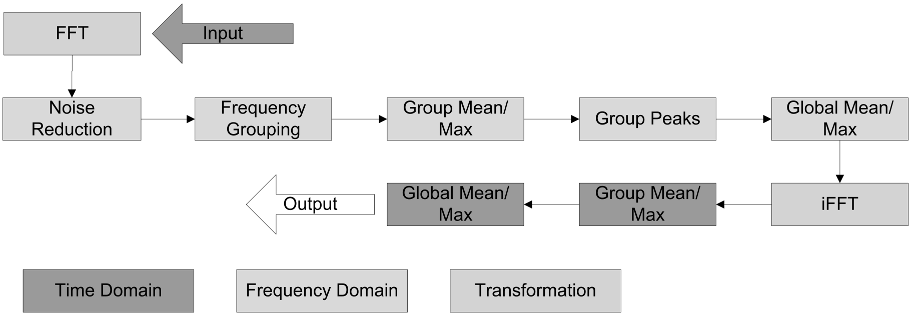

Figure 3.3 *Signal preprocessing*

- Fast Fourier Transformation
- Noise Reduction

- Frequency Grouping
- Calculate Group Mean/Max
- Calculate Group Peaks
- Calculate Global Min/Max
- Group Inverse Fast Fourier Transformation: Each frequency group is separately transformed back into the time domain. With this transformation it is possible to analyze the individual groups or frequencies in the time domain, without all other frequencies in the signal.
- Calculate Group Mean/Max
- Calculate Global Mean/Max
- Output: The output of the algorithm are the mean and maximum values of the frequency groups in the time and frequency domain, the number of peaks and the mean and maximum values of the complete signal in the time and frequency domain. This is much less data than pure signal data. The total number of the values depends on the width of the frequency groups (blocks).

Data samples can usually be divided up into two categories of data: high frequency data and low frequency data. Low frequency data is defined as data with a sampling frequency less than 1 kHz. High frequency data is any data with a higher sampling rate than 1 kHz.

The low frequency data will not be processed besides bringing the data into the correct data format for the algorithm. There is too little data to do frequency analysis and compression.

The high frequency data will be processed with the following steps: First the data is transformed into the frequency domain and then noise reduction is applied to the data, after that the frequency data is partitioned into small blocks and finally each block group is enhanced with extra information.

Fast Fourier Transformation and Grouping The fast Fourier transformation takes a number of time-domain samples and transforms them into the frequency domain. Basis of the FFT algorithm is the discrete Fourier transformation (see Section 2.2 for information on the Fourier transformation). A fast Fourier transformation is performed in

Table 3.1 *Preprocessing parameters*

Parameter Name	Possible Values	Unit
Block width	0–1000	Hz
Block overlap	0–50	%
Noise reduction	0–5	–
Calculate the mean amplitude for each block	boolean	–
Calculate the maximum amplitude for each block	boolean	–
Calculate the mean frequency power for each block	boolean	–
Calculate the maximum frequency power for each block	boolean	–
Calculation the number of peaks for each block	boolean	–
Minimum Value of a peak	0–5	–
Calculate the overall mean and maximum values	boolean	–

$O(NlogN)$ operations. A full transformation with the sampling frequency is done. After the fast Fourier transformation is done, the frequencies are divided up into blocks. A frequency group is called a "block". It is possible that frequency groups overlap each other. That means if a frequency group is from 1 to 100 and the overlap is 50 %, then the next frequency groups is from 51 to 150 and the following frequency group is from 101 to 200. If the overlap is 0 % then the first block is from 1 to 100, the second from 101 to 200 and the third from 201 to 300. The overlap is controlled by the *block overlap* parameter. The number of the frequencies that are grouped in one block is determined by the calculation parameter *Block Width*. If less than *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 behavior of the block-signal over the time. Figure 3.4 shows how a signal in the frequency domain is separated into blocks and how they are transformed back.

Noise Reduction Noise reduction is applied to the signal to remove random data from the samples to improve the feature detection of the

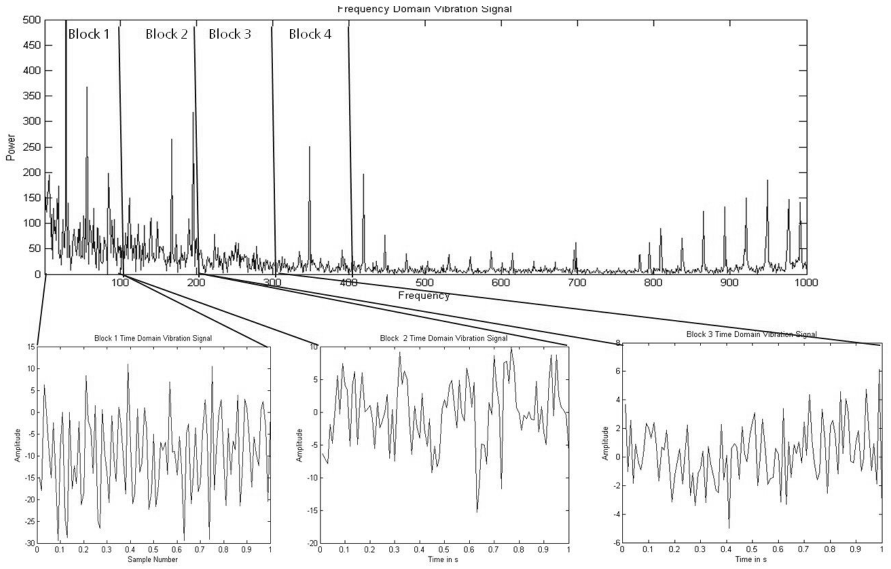

Figure 3.4 *Blocks and inverse Fourier transformation*

undisturbed signal. The maximum frequency power is calculated and then each frequency signal that is below a given 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*).

Additional Information and Data Compression Each block of the sampled data is enhanced with extra information. This information is added to give the following algorithm more information about the signal in the time and the frequency domain. The added information is for the time domain:

- The maximum amplitude of each block
- The mean amplitude of each block
- The maximum amplitude of the complete signal
- The mean amplitude of the complete signal

In the frequency domain the following information is added:

- The mean frequency power of each block

- The maximum frequency power of each block
- The frequency with the highest power of each block
- The number of peaks that are higher than a given magnitude of the mean frequency power
- The mean frequency power of the complete signal
- The maximum frequency power of the complete signal

The extra information is also calculated for the complete signal sample. Experiments showed that the added information is more useful for the algorithm than the raw data. This allows compressing the data. For example the information of 100 frequencies is reduced down to four attributes (maximum and mean power, the frequency with he maximum power and the number of peaks). Almost the same result is achieved in the time domain. Instead of calculating the amplitude for each frequency in the time domain, only two attributes (maximum and mean amplitude) are calculated for 100 frequencies.

$$\begin{aligned}
Freq\_Info &= 4 \cdot \frac{Frequencies}{BlockWidth} && (3.1)\\
Time\_Info &= 2 \cdot \frac{Frequencies}{BlockWidth} && (3.2)\\
Total\_Info &= Freq\_Info && (3.3)\\
&\quad + Time\_Info \\
&= 6 \cdot \frac{Frequencies}{BlockWidth} \\
Normal\_Info &= 2 \cdot Frequencies && (3.4)\\
Compression &= \frac{Total\_Info}{Normal\_Info} && (3.5)\\
&= \frac{3}{BlockWidth}
\end{aligned}$$

The needed data is reduced to 3 % if $BlockWidth = 100$ and $Frequencies = 11000$.

3.2.4 Calculate Decision Tree

The sensor data samples are converted to training samples in the preprocessing step. All training samples now have a number of features and a class. Decision tree calculation uses any of the available algorithms (ID3, C4.5, Random Forests, CART, ...).After the decision tree is calculated it needs to be tested and evaluated. If the performance of the decision tree is below a limit depending on the needed accuracy then it is possible to try to improve the performance by modifying the preprocessing parameters.

3.2.5 Optimize Decision Tree

The performance of a decision tree can be improved by modifying the preprocessing process[2]. Processing option can be turned on or off and parameters can be changed. It might be unfeasible to calculate the optimum parameter set depending on the number of the options and their possible combination. If one decision tree calculation takes a long time and if the solution space is large, then it is not possible to test possible combination. Instead a heuristic optimization approach is needed. Greedy Search, Simulated Annealing and Genetic Algorithm are the most common heuristic optimization methods. Some methods may be more useful than others depending on the problem. The genetic algorithm does have the advantage that it can be executed in parallel which reduces the overall calculation time.

A decision tree is calculated with each new generated preprocessing parameter set. The optimization continues until a given number of decision trees have been calculated or until a decision tree has a better performance than the limit[3].

3.3 Monitoring Process

The monitoring process classifies new sensor data samples. For this task the new sensor data samples are preprocessed with the same parameter set, which was used to calculate the decision tree. The decision tree then evaluates the data sample and classifies it. Output of the process is a classification that is system condition that the sensor data sample stands for.

3.4 Summary

This section showed a process for monitoring the condition of a system. The process needs no knowledge about the system beside classified sensor data samples. The process can adapt itself to different system with signal processing and heuristic optimization. When the decision tree was calculated it can be used to classify new sensor data samples.

4 Condition Prediction

The condition prediction process decides on the best prediction method for the current time series and then to predict a certain number of future data points. The process is divided into two sub-processes. One process for problem training and the other sub-process for prediction (Figure 4.1).

The overall process is very similar to that of condition prediction. Training also uses an optimization loop. But the individual process steps are different. In the training process all training samples are classified by the process and not by a human operator. The human operator only defines the maximum number of past data points and how far into the future the process shall predict the time series. The prediction process contains a loop that calculates multiple predictions.

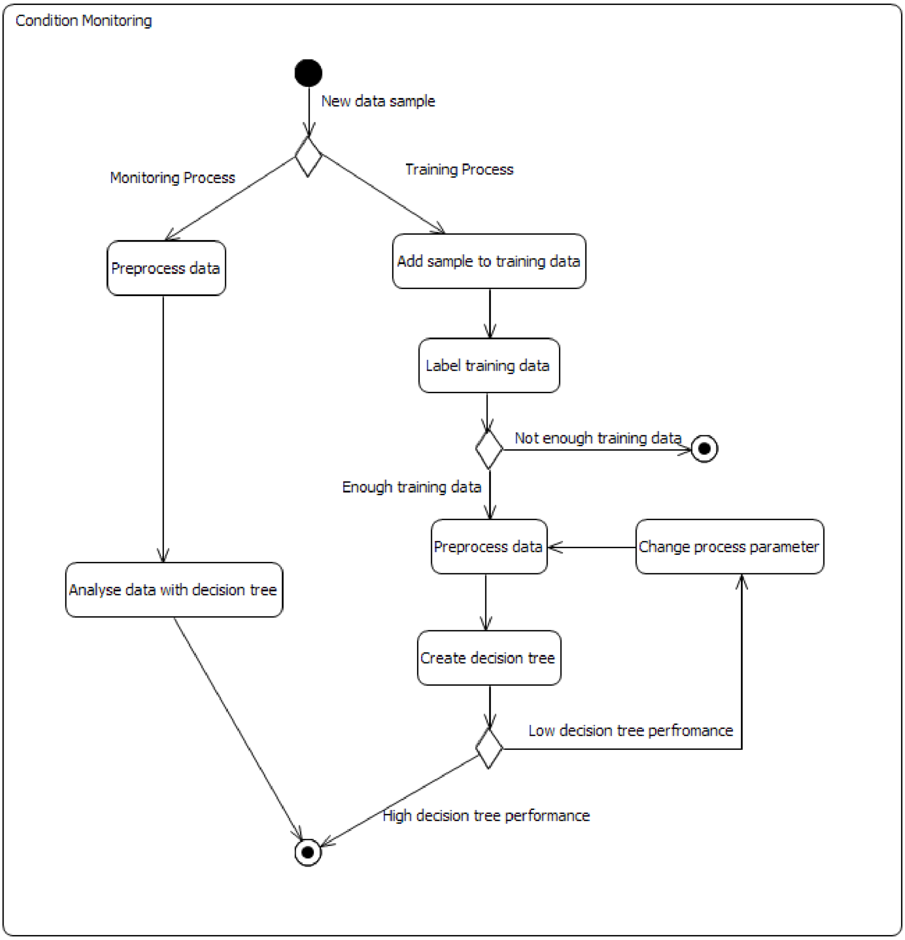

Figure 4.1 *Condition prediction process*

4.1 Training Process

The training process does have five fundamental steps: data sampling, data classification, data preprocessing, decision tree calculation and prediction testing.

4.1.1 Data Samples

Generation of time series data samples is controlled by prediction constrains. Time series data samples can be created in a static and in a dynamic way. The basic time series data sample generation process is shown in Figure 4.2.

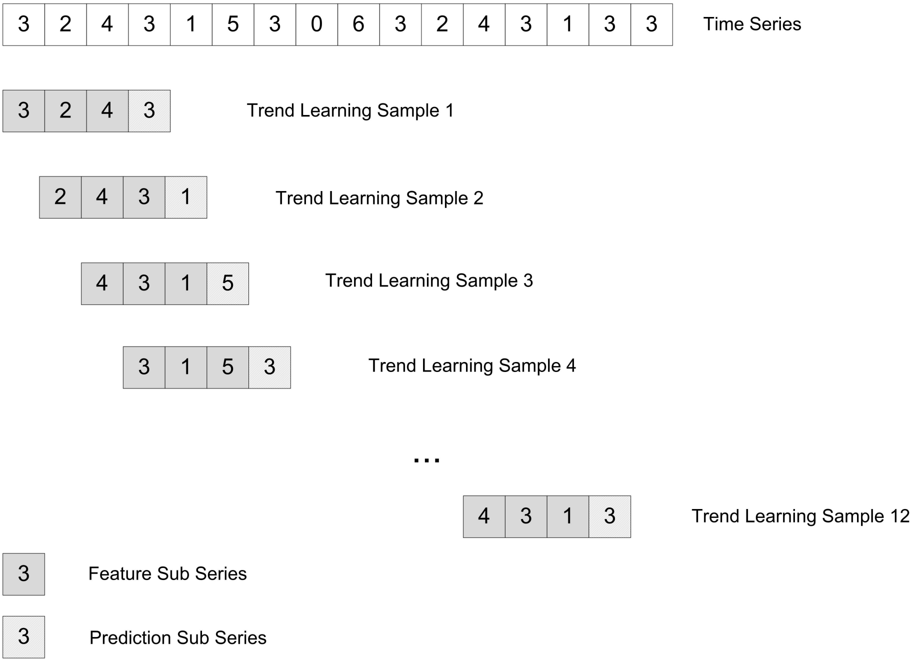

Figure 4.2 *Time series sample generation*

Multiple time series data sample are generated from one or more time series. A time series data sample is generated by moving a window over the time series. All data points in the window form a time series data sample. The window is shifted by one or more data points after a sample is taken. The number of data points the window is shifted depends on how many training data samples are needed. The **static window** size is:

$$w = d_p + d_f \tag{4.1}$$

where w is the window size, d_p is the number of past data points and d_f is the prediction horizon. It is possible to create and mix time series data samples from different time series for the training, if multiple time series from one problem are available.

A **dynamic window** is also possible. In this case the window grows with each step. The window size starts with only a few data points, but grows with each step. This is normally the case when the training data shall represent the time series as it grows and shall include all past data points. A dynamic window can only be used if the possible features (Section 4.1.3) do not include any features that are dependent on the number of data points.

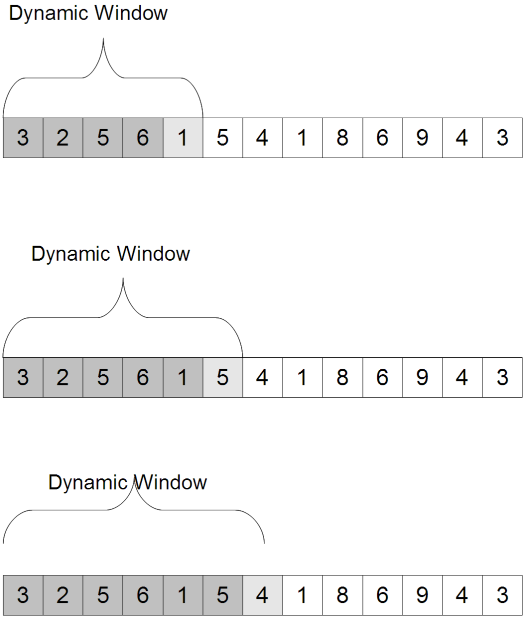

Figure 4.3 *Dynamic window*

A static window can include the complete time series. In this case just the separation between past data points and future data points for the training data is changed to create samples.

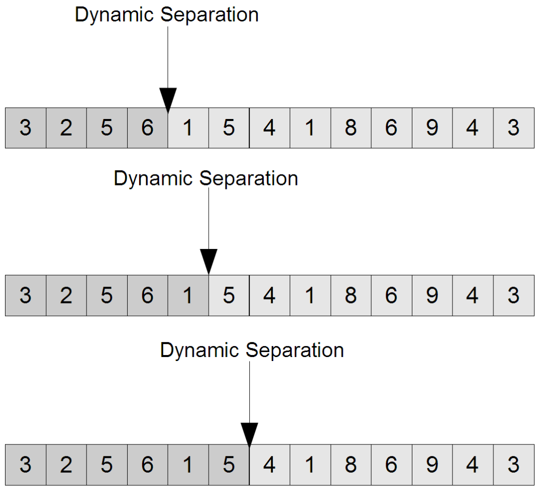

Figure 4.4 *Dynamic time series separation*

The data classification step needs the past data and the future data points of a training sample. Data preprocessing needs only the past data points.

4.1.2 Data Classification

Data classification is used to calculate the best prediction method for the current time series sample. The calculation is done by testing which of the available approximation methods has the least approximation mean square error for the approximated future data points. A constraint is that the approximation/extrapolation can only be calculated for past data points of the training sample and cannot use the data points that are marked as future data points. The reason for this constraint is that the decision tree later will also only have those data points available. This means an approximation/extrapolation is only calculated for the constrained time series sample (only past data points), but the mean square error for the future data points needs to be low. The following methods can be used for the prediction of data points:

- Linear Regression
- Multiple Regression

- Moving Average
- Exponential Smoothing
- Autoregressive Integrated Moving Average (ARIMA)

4.1.3 Data Preprocessing

Data preprocessing transforms a time series data sample into a training data sample by calculating the time series features. Different features for each sample are calculated. Those features plus the classification, which is calculated in a previous step, then form the training data sample. Which features are calculated and how they are calculated depends on the preprocessing parameters. This process step is similar to the preprocessing in the condition monitoring process. The following features are possible:

- Maximum value
- Mean value
- Minimum value
- Gradient

Controlled is the process by the following variable parameters:

- Maximum number of past data points
- Usage of maximum value
- Usage of mean value
- Usage of minimum value
- Usage of gradient
- Usage of other time series if available.

It is possible to use other features and parameters that are not listed if they can be applied to time series. Preprocessing is only applied to the data points that are marked as past data points. The data points are the same that were used for calculating the classification.

4.1.4 Decision Tree Calculation

A decision tree can be calculated after the training data have been calculated. Decision tree calculation can be done with any available algorithm for decision tree creation. The result is a decision tree that decides which method shall be used to predict data points.

Testing the decision tree for prediction is more complex than for condition monitoring. Standard methods cannot be used, because the time series prediction is the goal and not the decision making. The decision tree is tested by calculating the prediction for the original time series that were used to create the time series data samples. For this step the prediction process is executed multiple times. The prediction is calculated for every possible starting point of the original time series. For each prediction the following values are calculated:

- Maximum squared prediction error
- Mean squared prediction error
- Minimum squared prediction error
- Confidence range for a maximum prediction error of 10 %
- Confidence range for a maximum prediction error of 5 %
- Confidence range for a maximum prediction error of 1 %

Confidence range is the forecasting horizon, where the maximum prediction error is below a defined limit. Confidence range is measured as a fraction of the forecasting horizon that should have been predicted. E.g. a forecasting horizon of 10 data points out of a trained prediction horizon of 100 data points would be a confidence range of 0.1. The measurement of the overall performance is:

$$
\begin{aligned}
p_{pred} = 6 - \frac{w_0}{1 + err_{max}} + \frac{w_1}{1 + err_{mean}} + \frac{w_2}{1 + err_{min}} \\
+ w_3 cr_{10} + w_4 cr_5 + w_5 cr_1
\end{aligned}
\tag{4.2}
$$

where $w_0, \dots, w_5$ are weights between 0 and 1. err_{max}, err_{min} and err_{mean} are the calculated prediction errors. cr_{10}, cr_5 and cr_1 are the confidence ranges. p_{pred} is the prediction performance value. A lower value indicates a better prediction performance.

4.1.5 Optimization

If the performance of the prediction is lower than a limit, an optimization loop is started. The optimization loop works exactly like the optimization loop for the condition monitoring process. A heuristic optimization is used to modify the parameters for the data classification and the data preprocessing. The parameter for the maximum past data points may be not increased past the maximum past data points limit. The number of the future data points to be predicted may not be changed.

4.2 Prediction Process

The time series prediction is an iterative process based on three steps. First is the given time series preprocessed to extract the time series features. In the second step is the decision tree evaluated to select the best prediction method for the given time series. In the next step one or more data points (based on parameters) are calculated. Predicted data points are added to the time series. The two steps are repeated with the new time series containing the new predicted data points.

4.2.1 Data Preprocessing

The first step is to preprocess the data and calculate the features of the time series that shall be predicted. Data preprocessing depends and the windowing of the training data. If a static window and static data separation was used then the preprocessing should use the same number of past data points that was used in the training. If a dynamic window or dynamic data separation was used then this needs also to be considered when choosing which past data points are used. Data preprocessing uses the same parameters that were used for the training of the final decision tree.

4.2.2 Calculate Prediction Method

The best prediction method can be chosen with the decision tree, when the features have been calculated. The decision tree is evaluated using the default decision tree evaluation rules.

4.2.3 Predict data point(s)

Data points are predicted using the selected prediction method and the time series data sample. The prediction method can be used to predict one or more new data points. Normally a number of future data points shall be predicted that is the same as in the training.

Newly predicted data points are added to the time series data sample. If the prediction horizon is not yet reached the process is executed again using the new data points as past data points.

4.3 Summary

The condition prediction process is more complex than the condition monitoring complex. But if the parameters are set then the process works automatically and creates a prediction method for the current problem. The prediction can is optimized for a certain prediction horizon. It is possible to calculate different decision trees and preprocessing parameters to have a short term forecasting and a long term forecasting. Both methods give different information to the user.

5 Failure Prediction

Failure prediction is the process to predict a failure in the future based on current and past data. The process takes condition monitoring data from monitored system. Data is enhanced with additional information by fuzzy decision tree evaluation[4] and then used for condition prediction to forecast failures in the future. Failure prediction can be considered as a sample application of the condition monitoring and the condition prediction process. Fuzzy Decision Tree Evaluation is the link between the two processes and thus will be explained in the following subsection, followed by the process description itself.

5.1 Fuzzy Decision Tree Evaluation

Fuzzy decision tree evaluation takes a normal decision tree and evaluates all possible path and calculates the similarity of the input data to each class, where the correct class has a similarity of 1 or 100 %. The evaluation of all paths is done by assigning each decision a weight based on the boolean decision. The "true" decision is is given a weight of one and the "false" decision gets a value lower than one and higher than zero. The value of the "false" decision is calculated based on the distance of the data to the "true" border (decision split). Different methods to calculate the distance can be applied based on the problem. During the evaluation the value for each path is calculated by taking the sum of the weights of the path and then divide the sum by the depth of the path (taking the average of the path values). This results in a value for each leaf. It is possible for one class to have multiple leafs, in this case the largest value of all leafs for one class is used as the result for the class. The advantage of this evaluation is that the decision tree creation algorithm does not to be changed and it can be applied to any decision tree instead of the default decision tree evaluation.

5.2 Goal Setup

Goal of the failure prediction is to predict a failure. The prediction process takes a time series and predicts the future of the time series. Fuzzy decision tree evaluation returns multiple results for each data sample. If the data sample come in a chronological order with the same time between each sample, then we get multiple time series, one for each possible class. The prediction process can only predict one time series at a time. The user of the failure prediction needs to decide what class he wants to predict. A drawback of the fuzzy decision tree evaluation is that the class of the current sample always has a result of 100 %. This means that the it is not possible to use the condition prediction for the no-failure state, if there is only one, because then the time series will have multiple 100 % values in a row, which makes the prediction as it is implemented impossible. The algorithm does not "know" at which position in time it is. That means a class needs to be monitored that is not the no-failure class. If there is only one failure class then that class is selected for the prediction, otherwise one prediction for each failure class has to be made. Each predictor has to be individually trained, which increases the training time significantly.

5.3 Training Process

The training process for the condition monitoring part of the failure prediction process is the same as for the default condition prediction. The next step is to use the training condition monitoring tree to create a time series for the prediction process. For this are the samples ordered by time and are processed by the fuzzy decision tree evaluation using the trained tree. This results in multiple parallel time series. One or more time series are now selected based on the selected goal setup. The image below shows the training process for the failure prediction.

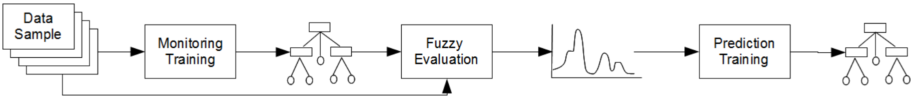

Figure 5.1 *Failure Prediction Training Process*

5.4 Prediction Process

The prediction process for failure prediction is similar to that of condition monitoring and condition prediction. First the current condition of the system is evaluated based on a data sample. Next the fuzzy decision tree evaluation is applied and the output is the input for the condition prediction together with past data. Here it is possible to perform multiple predictions for different target conditions if needed or wanted. For the prediction it is needed to keep a time series for each target condition in memory. The image below shows how the prediction process looks like.

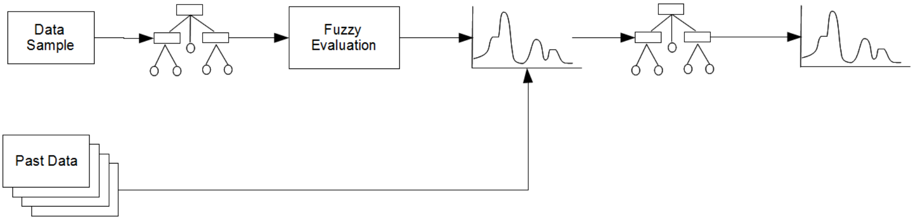

Figure 5.2 *Failure Prediction Process*

5.5 Summary

This section described a method how to combine the previously shown two methods. The combined method can be used for a complete failure prediction. Link between the two methods is the fuzzy decision tree evaluation, which allows the transformation of discrete decision results into continuous data. The prediction process takes the data adds it to data in a database and predicts a time series for one or more different target system states.

6 Experiments

Experiments were done to evaluate the condition monitoring and prediction concept. Different experiments for each of the two processes were done. A test rig which was built together with Airbus Operation GmbH was used for the condition monitoring experiments. Matlab was used for condition prediction, because it was not possible to use the test rig to generate a time series that would represent the reality.

Both experiments try to simulate the filter clogging of the high pressure air filters of the air conditioning of the A340-600 aircraft. This system was chosen, because it has no active parts and is completely passive and thus difficult to monitor. But the system is connected to fans and air flows through the filters.

6.1 Test Rig

The test rig was built by Airbus Operations GmbH during the PAHMIR project to provide a testing environment. The test rig is built into an aircraft hull and consists mostly of aircraft parts. The goal is to reassemble the real environment as closely as it is possible and needed. The test rig was constructed from the following parts:

- HP Recirculation Fan of the A340-600
- HP Air Filter of the A340-600
- Ducting
- Electronic Vibration Measurement Box

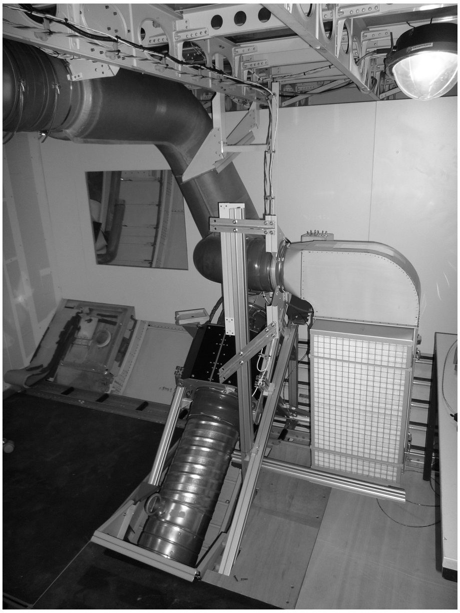

Figure 6.1 *Test rig*

The Electronic Vibration Measurement Box (EVB) is used to record sensor data. The box is equipped with two vibration sensors and two microphones. One of each sensor type is attached to the fan and to the filter housing. The EVB (Figure 6.2) is a hardware developed in PAHMIR. In total ten boxes were produced for different tasks. Design goal was to have a box which can record and store different sensor date for a long time (8 weeks). During the project the EVB was used to record data from different experiments and at the ground test rig. The EVB was also used on an Airbus Operations test flight in Toulouse to record data [32][33].

Parts breakdown:

- Enclosure
- Autonomic electronic board
- 8 AA batteries
- SD card (16 GB) are stored in the enclosure.
- 4 sensors (2 microphones and 2 acceleration sensors)

Dimensions:
Sensor: 2200mm (cable)
Enclosure: 105mm x 55mm x 190mm

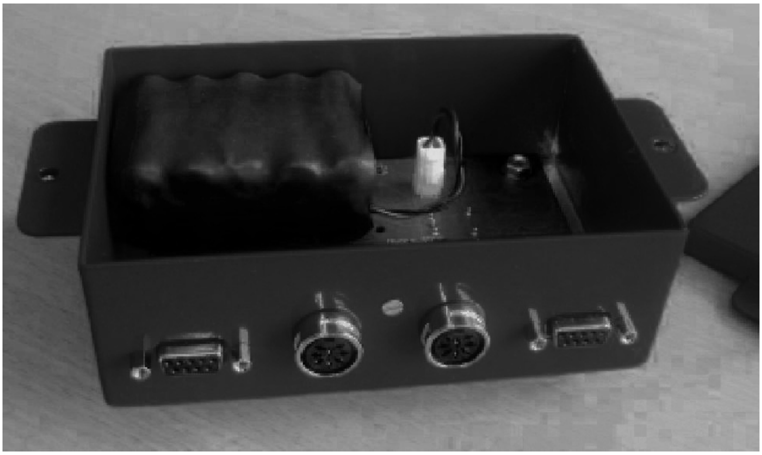

Figure 6.2 *Open EVB*

The autonomic box and the SD – card are stored in the enclosure. The battery pack that powers the autonomic sensor box is equipped with overcharge and short circuit protection. It consists of eight Panasonic LR6AD AA primary batteries, which comply with IEC 60086. The EVB contains two internal sensors in addition to the external attachable sensors: a temperature and a pressure sensor. With the pressure sensor it is possible to detect if the aircraft is in cruse flight or not. With the temperature sensor it is possible to detect the temperature setting of the environment, specifically, if the air conditioning is turned to low or high. Up to four external sensors can be attached to the EVB. Sensor data is recorded as a four channel wave file.

The EVB is recorded with a simple configuration file, which is stored on the SD-card. Configuration options are:

- Sampling frequency (default: 48000 Hz)
- Number of thousands of samples that will be recorded each time (default: 48000)
- Number of seconds the device will sleep in between 2 recordings (default: 600 seconds)

- Now much time the sensors take to stabilize at power on (default: 50 milliseconds)
- Gain for each channel (default: 1)

6.2 Condition Monitoring

The experiments in this section shall show the ability of the concept to detect clogging of air filters by using the processes. The condition monitoring experiment simulate the clogging of an air filter with dust. In addition experiments for evaluating the preprocessing, optimization and fuzzy decision tree evaluation were performed. The preprocessing experiments are shown in [2], the optimization experiments are shown in [3] and the fuzzy decision tree experiments are shown in [4].

6.2.1 Setup

Data for the experiments were collected at the ground test rig. Fan and filter vibration and sound data were recorded. During the data collection each filter was polluted with dust (MIL-Spec (quartz) dust). The dust was added in 25 gram steps going from 25 gram up to 200 gram per filter for a total of eight possible classes. The classifier was trained by applying the optimization process with genetic optimization and calculated 10 generations each with 20 members. The starting population is a random parameter list. 15 data samples were used for every class (pollution grade). To test if the classification accuracy can be increase the experiments were performed with a single tree classifier and with a classifier build out of three different decision trees using the decision that was selected by the majority of the classifiers.

6.2.2 Results

In the experiment the goal was to detect how much dust is in the filters. The complete data set was used for the experiment. For test data the training data was arranged by increasing weight. The classification should detect a nine step function. Table 6.1 shows the number of the samples per class. Figure 6.3 shows that the calculation resembles a step function, just a few classes are classified wrongly. With three classifiers (same training data, different parameters) the number of wrong classified classes drops even more (Figure 6.4). Figure 6.5 shows the results of the optimization. The red line is the average fitness and the green line is the best fitness. It clearly shows that more generation would have given a better performance. The average and maximum fitness of the population steadily increased.

Table 6.1 *Number of detected classes for one and three decision trees*

Dust	1 tree	3 trees
25 gram	15	15
50 gram	15	15
75 gram	15	15
100 gram	13	13
125 gram	13	17
150 gram	14	14
175 gram	16	16
200 gram	19	15

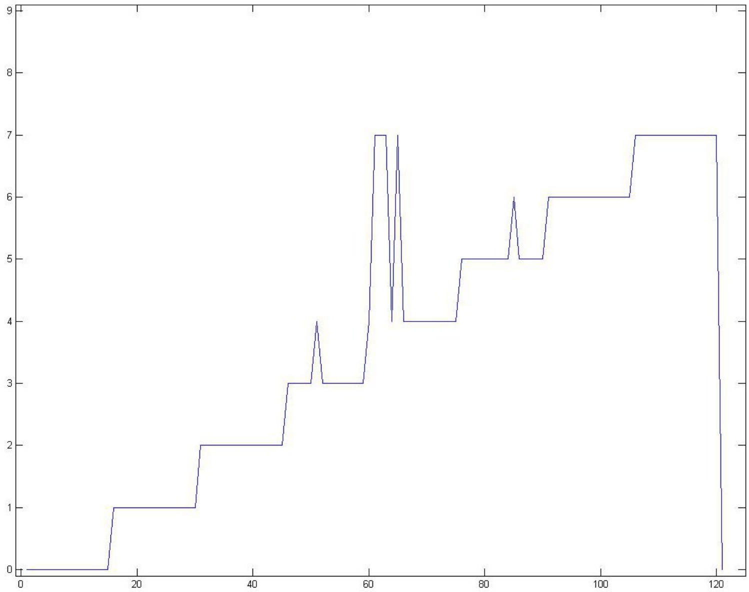

Figure 6.3 *Classifications as a time series with one classifier*

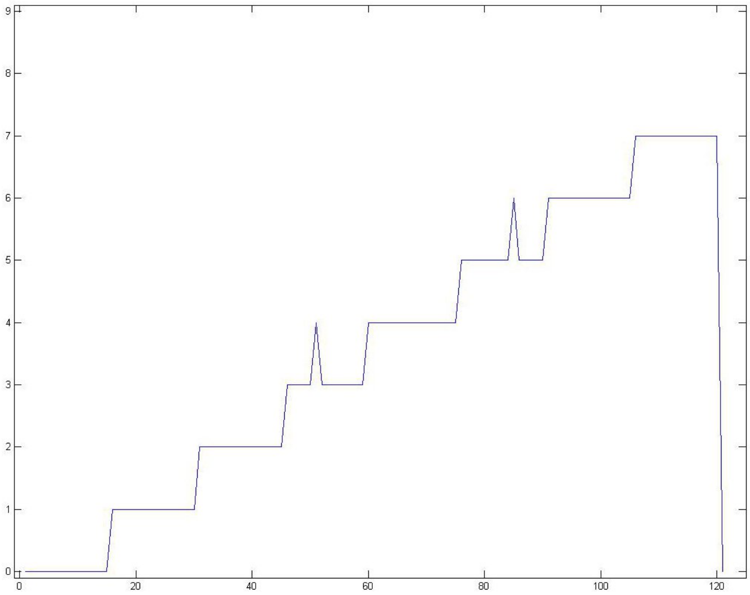

Figure 6.4 *Classifications as a time series with three classifiers*

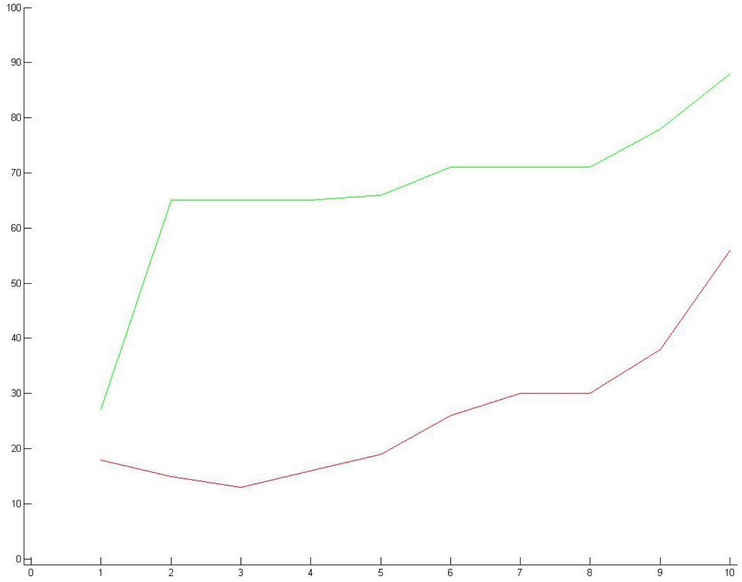

Figure 6.5 *Genetic optimization performance*

6.3 Condition Prediction

The goal is to predict a typical system health condition function[8]. The function should be followed as close as possible. The experiment and results are shown in[5]. [34] shows how the concept can be used for long horizon forecasts. [34] also evaluates time series models like moving average, exponential smoothing, ARMA and ARIMA.

6.4 Summary

This section showed that the developed concepts work and that it is possible to make a reliable condition monitoring and condition prediction. Condition monitoring was tested on a test system developed by Airbus Operations GmbH. Condition prediction was tested for with a short forecasting horizon and a general system condition function.

7 Conclusions

The document showed that it is possible to monitor and predict the condition of a system with the developed concepts. The developed concept combine the use of decision trees and a genetic algorithm with signal analysis and approximation methods to create a system that can learn and adapt to various problems. This does have the advantage that the concept can be easily understood by a human operator and that it can be applied to different monitoring problems. The uses of a decision tree together with the optimization algorithm during the training of the system condition monitoring enables to process to use only the signal analysis methods that give the most useful information to solve a given problem. During the system condition monitoring the decision tree uses the data from the signal processing to classify the signal data and detect the current system condition. The advantages for the forecasting are similar. The decision tree and the optimization are used to select different forecasting methods based on past experience. This does have the advantage that the forecasting method can be switched dynamically during the prediction to enable the process to react to events and to handle non-linearities in the observed data.

Experiments showed that training and using multiple classifiers for the same problem and then take the class which the majority selected improves the accuracy of the classification. For fuzzy decision tree evaluation this means that the similarity value of the other classes needs to be averaged over the decision trees that decided for the same class.

The processes work mostly autonomous and do not require much human interaction. Only for collecting and labeling sample data are humans needed. The algorithms can work alone after the data is setup.

Due to the simplicity of the concepts it is possible to change the algorithms and include new functions. Different decision tree algorithms and time series models can be used.

Future improvements would include improving the preprocessing for the condition monitoring, better labeling of samples for condition prediction and combination of short and long horizon forecasts. The concepts can not replace another maintenance method on the fly. Instead the system can be operated in parallel to the currently used maintenance method. Sensors collect data and they are correlated with maintenance action and system/machine age. After some time the new maintenance concept can replace the old one.

Bibliography

[6] Robert Bond Randall. *Vibration-Based Condition Monitoring. Industrial, Aerospace, and Automotive Applications*. John Wiley & Sons, Ldt, 2011.

[7] British Standards Institution. *Glossary of terms used in terotechnology*. British Standards Institution, London, 1993.

[8] Josef Kolerus and Johann Wassermann. *Zustandsüberwachung von Maschinen: Das Lehr- und Arbeitsbuch für den Praktiker*. Expert-Verlag, 2011.

[9] John H. Williams, Alan Davies and Drake Paul R. *Condition-based Maintenance and Machine Diagnostics*. Chapman & Hall, 1994.

[10] Victoria J. Hodge and Jim Austin. 'A survey of outlier detection methodologies'. In: *Artificial Intelligence Review* 22 (2004), p. 2004.

[11] Tom Brotherton, George Chadderdon and Paul Grabill. 'Automated Rule Extraction for Engine Vibration Analysis'. In: *Proceedings of the 1999 IEEE Aerospace Conference*. 1999.

[12] Sh. Ataei et al. 'Sensor fusion of a railway bridge load test using neural networks'. In: *Expert Syst. Appl.* 29 (3 2005), pp. 678–683.

[13] A.S. Zaher and S.D.J. McArthur. 'A Multi-Agent Fault Detection System for Wind Turbine Defect Recognition and Diagnosis'. In: *2007 IEEE Lausanne Powertech*. 2007.

[14] Alice E. Smith, David W. Coit and Yun-chia Liang. *A Neural Network Approach to Condition Based Maintenance: Case Study of Airport Ground Transportation Vehicles.*

[15] Mark Owen. *Practical Signal Processing*. Cambridge University Press, 2007.

[16] Lawrence Rabiner and Biing H. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[17] Douglas C. Montgomery, Lynwood A. Johnson and John S. Gardiner. *Forecasting & Time Series Analysis.* McGraw-Hill, Inc, 1990.

[18] Bruce L. Bowerman and Richard T. O'Connell. *Forecasting and Time Series. An Applied Approach.* Duxbury Press, 1993.

[19] C.W.J. Granger and Paul Newbold. *Forecasting Economic Time Series.* Academic Press, 1977.

[20] Hans R. Schwarz and Norbert Köckler. *Numerische Mathematik.* Teuber, 2004.

[21] Animesh Chaturvedi and Samanvaya Chandra. 'A Neural Stock Price Predictor using Quantitative Data'. In: *iiWAS 2004 - The sixth International Conference on Information Integration- and Web-based Applications Services, 2004, Jakarta, Indonesia.* Vol. 183. Austrian Computer Society, 2004.

[22] Maria Kontaki, Apostolos N. Papadopoulos and Yannis Manolopoulos. 'Continuous Trend-Based Classification of Streaming Time Series.' In: *ADBIS.* 2005, pp. 294–308.

[23] Depei Bao and Zehong Yang. 'Intelligent stock trading system by turning point confirming and probabilistic reasoning'. In: *Expert Systems with Applications* 34.1 (2008), pp. 620–627.

[24] Tom M. Mitchell. *Machine Learning.* The McGraw-Hill Companies, Inc., 1997.

[25] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[26] Cristina Olaru and Louis Wehenkel. 'A complete fuzzy decision tree technique'. In: *Fuzzy Sets Syst.* 138 (2 2003), pp. 221–254.

[27] Mohd Noor Md Sap and Rashid Hafeez Khokhar. 'Fuzzy Decision Tree for Data Mining of Time Series Stock Market Databases'. In: *Methods of Microarray Data Analysis IV.* 2004.

[28] Ming Dong et al. 'Look-Ahead Based Fuzzy Decision Tree Induction'. In: *IEEE-FS* 9 (2001), pp. 461–468.

[29] Marin Golub and Andrea Budin Posavec. ‘Using Genetic Algorithms for Adapting Approximation Functions’. In: *Proceedings of the 19th International Conference on Information Technology Interfaces ITI ’97* (1997).

[30] Gary Stein et al. ‘Decision tree classifier for network intrusion detection with GA-based feature selection’. In: *Proceedings of the 43rd annual Southeast regional conference - Volume 2*. ACM-SE 43. Kennesaw, Georgia: ACM, 2005, pp. 136–141. ISBN: 1-59593-059-0. DOI: 10.1145/1167253.1167288.

[31] J. Ross Quinlan. ‘Induction of Decision Trees’. In: *Mach. Learn* (1986), pp. 81–106.

[32] Mike Gerdes. ‘PAHMIR Final Project Report’. 2009.

[33] Loretta Grieshaber. *Erfassung und Synchronisation von Systemschwingungen bei unterschiedlichen Flugphasen*. 2009.

[34] Adam Kret. *Statistische Analyse und Vorhersage von Zeitreihen zur vorausschauenden Wartung von Flugzeugkomponenten*. 2011.

Paper I

Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration

Mike Gerdes[1], Dieter Scholz[1] and Bernhard Randerath[2]

1 Aircraft Design and Systems Group - Aero
Hamburg University of Applied Sciences
Berliner Tor 9, 20099 Hamburg, Germany
2 Maintainability, Reliability & Testability
Airbus Deutschland GmbH
Kreetslag 10, 21129 Hamburg, Germany

This paper has been published as:
M. Gerdes, D. Scholz and B.üRanderath, “Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration”, in *2nd International Workshop on Aircraft System Technologies*, AST 2009, TUHH, Hamburg, 26./27. März 2009.

Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration

Mike Gerdes, Dieter Scholz and Bernhard Randerath

Abstract

With respect to future trends in cabin design, this paper takes a close look at two problem areas that produce delays and reduce the time an aircraft is in the air. The first is related to unscheduled maintenance and the second is related to necessary cabin reconfigurations. Each of the two problem areas are analyzed for time consumption and costs. The research project PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration, a cooperation project between Airbus Germany and Hamburg University of Applied Sciences) ads "intelligence" to aircraft components and novel technologies to the aircraft system. These measures will be able to reduce time consumption and costs for unscheduled maintenance caused by the air conditioning system in the order of 8 %. These measures will be able to reduce time consumption for cabin reconfiguration by about 36+%. The upper price limit for economic intelligent quick fasteners seems to be around 500 USD.

Keywords: Aircraft Delays, Cost Reduction, A340, Condition Monitoring, Reconfiguration

1 Introduction

Unscheduled Maintenance Unscheduled maintenance is caused by equipment that shows an unexpected fault during flight. The aircraft continues to fly safely due to its built-in redundancy, but the equipment (generally) needs to be fixed before the next take off. If it is not possible to fix the equipment during turn around time, the flight will be delayed until the fault is eliminated. By definition, delays are incurred when an aircraft is prevented from off-block to its destination for an interval of 15 min or more. To avoid unscheduled maintenance and possible delays, components are replaced after some time. In this way, maintenance is scheduled (hopefully) before the component shows a fault. In order not to waste life time of a component, a trend analysis of component parameters should help to predict the time of component failure and dynamically schedule a maintenance action shortly before the assumed failure is about to occur.

Reconfiguration The task of an aircraft cabin is to transport the majority of the pay load of a passenger air-craft. The cabin accommodates the passengers in a safe and comfortable way together with their hand luggage. The baggage, on the other hand, is stored in the cargo compartment. In a cabin mainly seats and monuments cover the available cabin floor space. Monuments are cabin units like galleys and lavatories. Cabin reconfiguration means a change in the type or number of seats, monuments or other cabin components. Reconfiguration includes work tasks like removal, modification, installation and testing.

PAHMIR The PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) project investigates new technologies: a) for maintenance, based on trend analysis and b) for cabin components important for the reconfiguration process. By adding "intelligence" to components and novel technologies to the system, delays / time and costs can be saved.

Aircraft, System and Database The Airbus A340-600 was selected for this study, because it is quite new (entry into service in 2002) and but already enough experience and data is present. By November 2008, 84 air-crafts of the Airbus A340-600 were in service [1].Focus in PAHMIR is on the air conditioning system (ATA Chapter 21). The air conditioning system was chosen, because it is flight critical, monitored [2], consists of a combination of mechanical and electrical systems and is one of the cabin systems that cause most delays [3]. A lot of reliable information is available in the database of the Airbus In-Service Report (ISR) [4].

2 Unscheduled Maintenance

Unscheduled maintenance is caused by equipment that shows an unexpected fault during flight. This fault needs to be fixed before the next scheduled flight – according to MEL (Minimum Equipment List). If it is not possible to fix the equipment during turn around time, the flight will be delayed until the fault is fixed. In extreme cases flights could get canceled and passengers may have to be redirected or compensated. Figure

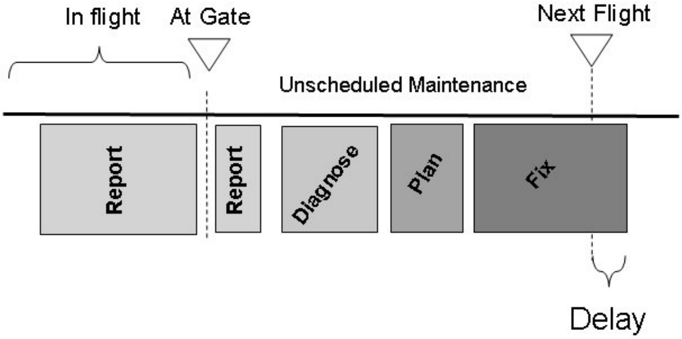

Figure 1 *Sequence of events during unscheduled maintenance leading to delays*

1 shows a typical sequence of events leading to unscheduled maintenance and delay.

2.1 Delay Analysis

By definition, delays are incurred when an aircraft is prevented from off-block to its destination for an interval of 15 min or more [5]. Delays can originate from traffic, passengers, weather and the aircraft. A delay causes additional operating costs. Theses costs are crew-related, ramp-related, aircraft-related and passenger-related (hotel and meal, re-booking and re-routing, luggage complaints and revenue losses) [6]. Delay costs can be calculated from [7], [8] or [9]. The magnitude of the delay costs caused by the air conditioning system is calculated, in order to show how important it is to reduce delays. Delay costs caused by the air conditioning system are calculated base on [9] with updated economic data from [10]. According to these sources, costs of a delay are assumed to be a linear function of delay time. The base value for delay costs is given in USD/min. The average delay costs (without network effects) are about 47 USD/min. With network effects this value increases to 78 USD/min. Based on [4], it is possible to calculate an average delay time caused by the air conditioning system of 90 min. Multiplying this value with the average delay costs (47 USD/min) yields delay costs of 4230 USD for a 90-minute delay. In 50 % of the studied cases however, the delay is less then 50 min (see Figure 2). In these cases, the average delay costs are below 2350 USD. These costs are quite substantial, so any efforts to reduce delays are highly welcome.

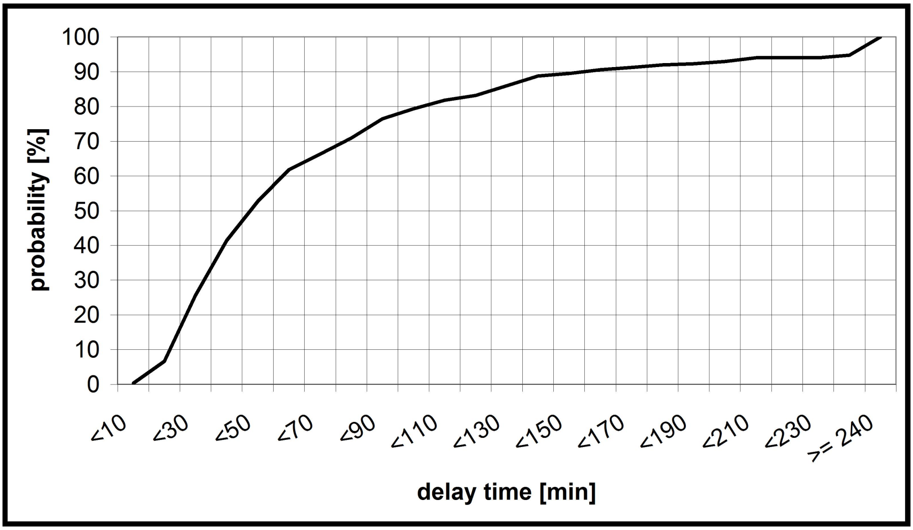

Figure 2 *Cumulative probability of delay time of the air conditioning system of the Airbus A340-600 based on data from [4]*

2.2 Reduction of Delays and Costs

To avoid unscheduled maintenance, different maintenance concepts have been developed. In preventive maintenance, components are replaced after a given period of time. In this way, maintenance is scheduled hopefully before the component shows a fault. In condition monitoring and trend analysis dynamic intervals are used.

Preventive Maintenance Preventive maintenance is the standard method for reducing unscheduled maintenance. Aircraft components are inspected after given time intervals. The intervals depend on the component type and can vary from airline to airline. Reducing the time interval can increase the need for spare parts; increasing the interval increases the risk of unscheduled maintenance [11]. Looking at preventive maintenance in more detail, three types can be identified [12]: **Hard-Time (HT)**: Scheduled removal of a component before some specified maxi-mum permissible age limit.

On-Condition (OC): Scheduled inspections, tests, or measurements to determine whether an item is in, and will remain in, a satisfactory condition until the next scheduled inspection, test, or measurement.

Task Oriented Reliability-Centered Maintenance (RCM): Tasks are selected in a hierarchy of difficulty and cost, from lowest to highest. Each task must also pass the applicability and effectiveness criteria. Depending on the consequence of failure (safety, operational, economic, hidden safety and hidden non-safety) a single or combination of tasks is selected.

Condition Monitoring Condition monitoring constantly measures and analyses relevant mechanical and electrical component parameters during operation. Those parameters are selected for monitoring that allow determining the condition and failure state. The need for maintenance of the component is only indicated, if parameters show a predefined degradation of the component [11].

Trend Analysis Trend analysis is an extension to condition monitoring. The analysis algorithm does not only look at recorded parameters at a single moment in time, but rather takes the full parameter history into account. The need for maintenance of the component is only indicated, if the data trend of parameters shows a degradation of the component. Based on the parameter time history, the analysis algorithm also allows giving a forecast of the remaining lifetime of the component [11].

Cost Reduction Those mechanical faults of fans and valves were selected from the ISR database [4] that could have been prevented by using trend analysis based on vibration measurements. Analysis showed that 20 % of all faults in the air conditioning system could have been prevented. Counting their delay time with 0 min gives an average delay of 83 min, which is a time reduction of 7 min or 8 % compared to the original A340-600 value of 90 min. This also means a reduction of the average costs of delays caused by the air conditioning system by 329 USD or 8 %. Calculating the average delay of the remaining incidents shows that the average delay increases to 102 min. This effect is caused by the fact that most faults that were potentially prevented caused delays less than 90 min. Figure 3 shows the cumulative probability with respect to delay time of the air conditioning system of the Airbus A340-600 assuming trend analysis has already been applied.

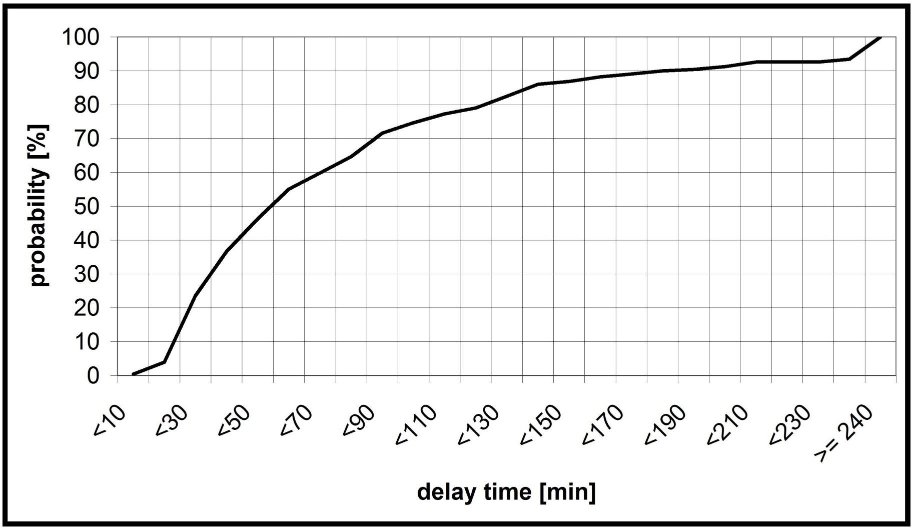

Figure 3 *Cumulative probability of delay time of the air conditioning system of the Airbus A340-600 assuming trend analysis would have been applied*

3 Cabin Reconfiguration

Aircraft cabins have various classes. First class (F/C), business class (B/C) and Yankee class (Y/C) are the basic classes. The Yankee class is also called economy class. The classes can be differentiated by their level of comfort. Most important for the level of comfort is seating space. In a cabin mainly seats and monuments cover the available cabin floor space. Monuments are cabin units like galleys or lavatories. Cabin reconfiguration means a change in the type or number of seats and monuments. Many other reconfiguration tasks are possible. Two forms of cabin reconfiguration can be differentiated: In class resizing, one cabin class is enlarged whereas another class is reduced in size. In monument shifting, monuments are assigned a different location in the cabin. The focus of this study is class resizing. Cabin reconfiguration can be divided into five subtasks [13] are: Removal: Mechanically and electrically installed components are removed from the cabin. The number and type of components that need to be removed depend on the reconfiguration scenario. Modification: Modification includes all tasks that are needed to prepare the new installation area. This includes cleaning, change of wiring, oxygen and system routing. Installation: Installation includes all tasks that are needed for installation of the new cabin layout. During installation all mechanical and electrical components need to be installed and configured for the new cabin layout. Testing: Testing includes all procedures and tasks to ensure the new cabin layout is sound. The new layout has to be tested to confirm that all components are installed correctly and that the correct components are installed. Both software and hardware is tested during this step. Testing time does not depend on the number of installed components, because testing is done in parallel. Tests are done for the complete reconfigured cabin. Troubleshooting: Troubleshooting tasks need only to be performed, if tests showed a failure during testing. Removal and installation tasks can become part of troubleshooting. Cabin components that are affected by reconfiguration include: seats, Passenger Service Units (PSU), stowage's, class dividers, carpets, lavatories, galleys and panels. This is not a complete list of all affected components and not all cabin components are affected by reconfiguration, e.g. if two rows of business class (B/C) seats shall be removed and three rows of Yankee class (Y/C) seats shall be installed, then only the affected seat rows, PSU, ceiling panels and class dividers need to be modified. Quick cabin reconfiguration is demanded more and more by airlines to be able to adapt an aircraft to different flight routes. The goal is to be able to do class resizing during turn around and monument shifting during one night.

3.1 Process Time Analysis

Worked example: A very common task in reconfiguration is to remove Y/C seats and replace them with B/C seat or vice versa. For the analysis of the process, the example reconfiguration scenario, consisting of the removal of 12 B/C seats, replacement by 36 Y/C seats. The time is given as a percentage of the total reconfiguration process time. For the study, the time of individual reconfiguration subtasks in the class resizing process was analyzed. Improved hardware and software solutions for these subtasks help to reduce testing time. More complex reconfigurations, like those involving lavatories and galleys, are presently done in a hangar. In the close future it will still be unlikely to see these big reconfigurations taking part during turn around on the apron.

The relative process time for the above specified example reconfiguration scenario is presented in Table 1. The process time was taken from

Process	Workers	Time
Removal		**24 %**
Seat rail cover removal	3	1.5 %
B/C seat de-installation from seat rail	6	7.5 %
B/C seat de-installation	6	7.5 %
B/C seat removal	6	7.5 %
Modification		**12.4 %**
Cleaning of seat rails	6	3.7 %
Secondary structure	6	5.0 %
Oxygen	2	3.7 %
Installation		**34.0 %**
Y/C seat transportation into aircraft	8	4.5 %
Y/C seat positioning into seat rail	8	4.5 %
Y/C seat installation (mechanical)	8	4.5 %
Seat rail cover installation	6	1.0 %
IFE (In-Flight Entertainment) connection	1	7.5 %
PSU connection	3	11 %
Y/C seat installation (electrical)	3	1.0 %
Testing		**29.6 %**
Test of the passenger call system		3.0 %
Test of the passenger Supply Channel (PSC)		7.5 %
Illumination test		5.6 %
In-seat power supply test		3.0 %
Test of the IFE seat component		3.0 %
Test of seats		7.5 %

Table 1 *Relative process time for the example reconfiguration scenario*

[13] with selected measurement results explained in [14]. Table 1 shows that about 30 % of the reconfiguration time is taken up by testing. Also 4.5 % of the time is used to position the seat into the seat rail and about 13.5 % of the time is used for testing operation of the seat. It is also interesting to note that mechanical seat installation and removal takes up 12 % of the reconfiguration time. Hence 60 % of the time is taken up by tasks that can be facilitated by process improvements as proposed in the next section.

3.2 Reduction of Process Time and Costs

Component Abilities The PAHMIR project wants to reduce the reconfiguration time by adding "intelligence" to components [15] and novel technologies to the system. The intelligence includes the following component abilities: **Component Self Identification** allows a component to know what kind of component it is and how it can communicate with a configuration management in the aircraft cabin. By giving each component an identity, about 30 % of the software installation time can be saved. The software that was normally installed during the reconfiguration is now on-board the component and does not have to be installed on the aircraft.

Component Location Detection enables a component to know where it is installed in the cabin. A component can detect its location at any time, when the location detection system is turned on. With location detection it is possible to reduce the time needed for seat positioning into seat rail and later for troubleshooting. It also helps to reduce configuration time together with identification. First experience was gained with seat positioning by location detection and showed a process time reduction of 40 %.

Component Self Testing allows a component to use the installed testing equipment to start and run self tests as soon as it has power. Only Passenger Supply Channel (PSC) tests and seat tests are reduced by self testing during reconfiguration. The time saving is achieved by self testing in parallel and does not depend on a finished installation of the cabin. Measurements showed that the time needed to perform a single seat and PSC self test is only about 20 % of the complete testing time. A testing time reduction by 80 % means that the time to test a seat falls below the installation time of one seat ($0.2 \cdot 7.5\ \% < 4.5\ \%$). It follows that if a seat test is finished before another seat is installed

Process	Old Time	New Time	Saving
Removal	**24.00 %**	**17.25 %**	**6.75 %**
Seat rail cover removal	1.50 %	1.50 %	0.00 %
B/C seat de-installation from seat rail	7.50 %	0.75 %	6.75 %
B/C seat de-installation	7.5 %	7.50 %	0.00 %
B/C seat removal	7.5 %	7.50 %	0.00 %

Table 2 *Time saving for the removal in an example reconfiguration scenario*

and at the end of the seat installation only one seat is remaining to be tested.

With **Quick Fastening** it is possible to reduce the reconfiguration time significantly, because mechanical unlocking the locking mechanisms takes a significant amount of the time. Experiments showed that with a quick fastening technology, e.g. TZ Intevia fasteners [16], it is possible to reduce the time for fastening and loosening by 90 %.

Time Saving Potentials Taking the above component abilities and apply them to the above example reconfiguration scenario, where 12 B/C seats were exchanged against 36 Y/C seats, results in consider-able time savings.

Time Saving for Removal: For the removal process the following time saving is possible due to the recommended technologies. For removal it is possible to save 7,0 % of the total removal time and about 30 % of the removal time (Table 2).

Time Saving for Modification: With intelligent and quick fastening it is possible to reduce the time for modification of the secondary structures. For modification it is possible to save 4,5 % of the total reconfiguration time and about 30 % of the modification time (Table 3).

Time Saving for Installation: For the installation process the following time saving is possible due to the recommended technologies. For installation it is possible to save 8 % of the total reconfiguration time and about 24 % of the installation time (Table 4).

Time Saving for Testing: Time saving for testing is mainly achieved through parallel self-testing. For the testing process the following time saving is possible mainly due to self-testing. For testing it is possible to save 17 % of the total reconfiguration time and about 56 %

Process	Old Time	New Time	Saving
Modification	**12.40 %**	**7.90 %**	**4.50 %**
Cleaning of seat rails	3.70 %	3.70 %	0.00 %
Secondary structure	5.00 %	0.50 %	4.50 %
Oxygen	3.70 %	3.70 %	0.00 %

Table 3 *Time saving for the modification in an example reconfiguration scenario*

Process	Old Time	New Time	Saving
Installation	**34.00 %**	**25.90 %**	**8.10 %**
Y/C seat transportation into aircrafts	4.50 %	4.50 %	0.00 %
Y/C seat positioning into seat rail	4.50 %	2.70 %	1.80 %
Y/C seat installation (mechanical)	4.50 %	0.45 %	4.05 %
Seat rail cover installation	1.00 %	1.00 %	0.00 %
IFE connection	7.50 %	5.25 %	2.25 %
PSU connection	11.00 %	11.00 %	0.00 %
Y/C seat installation (electrical)	1.00 %	1.00 %	0.00 %

Table 4 *Time saving for the installation in an example reconfiguration scenario*

of the testing time (Table 5).

Total Time Saving: Combining the tables of the above example give the total time saving of the example reconfiguration scenario. Table 6 shows the results. Reconfiguration time can be reduced by 36 % in this example. The reduction can vary depending on the reconfiguration scenario (different duration, complexity and processes), but in this common scenario a significant process improvement was shown.

Time Saving for Troubleshooting: To show the possibility of time saving for troubleshooting, another example is selected. It is assumed that a seat with a defect IFE unit was installed. Two workers are involved in this troubleshooting task. Table 7 shows the tasks that need to be done and the possible time savings that amount in total to 20 %.

Cost Saving Potentials The cost savings for the given example reconfiguration scenario, where 12 B/C seats were exchanged against 36 Y/C seats, are calculated in this subsection. First the budget, which the air-

Process	Old Time	New Time	Saving
Testing	**29.60 %**	**12.80 %**	**16.80 %**
Test of the passenger call system	3.00 %	3.00 %	0.00 %
Test of the PSC	7.50 %	1.50 %	6.00 %
Illumination test	5.60 %	5.60 %	0.00 %
In-seat power supply test	3.00 %	0.60 %	2.40 %
Test of the IFE seat component	3.00 %	0.60 %	2.40 %
Test of seats	7.50 %	1.50 %	6.00 %

Table 5 *Time saving for the testing in an example reconfiguration scenario*

Process	Old Time	New Time	Saving
Total	100.00 %	63.85 %	36.15 %
Removal	24.00 %	17.25 %	6.75 %
Modification	12.40 %	7.90 %	4.50 %
Installation	34.00 %	25.90 %	8.10 %
Testring	29.60 %	12.80 %	16.80 %

Table 6 *Total time saving in an example reconfiguration scenario*

line will save by the reduced reconfiguration time, is calculated. Then the maximum price of the new component that enables to save time and money is calculated. For this study it is assumed that the costs of ownership (COO) for the aircraft are 10 USD/min as calculated by [9] and [10] and maintenance related burdened labor costs are 10 USD/min [17]. So the total costs for the reconfiguration example are 20 USD/min. The reconfiguration time for the given example is assumed to be eight hours (480 min) with current (2008) technologies and enough trained personal on the job [18]. For the new process, a time consumption of 306.5 min (480 min $\cdot$ 0.635 = 306.5 min) is calculated. Taking the assumed costs and applying them to the current reconfiguration (480 min $\cdot$ 20 USD/min = 9600 USD) and the modified reconfiguration process (306.5 min $\cdot$ 20 USD/min = 6130 USD) gives the resulting savings as shown in Table 8.

Process	Old Time	New Time	Saving
Troubleshooting	**100.00 %**	**80.50 %**	**19.50 %**
Fault Analysis	7.50 %	7.50 %	7.50 %
Removal	26.00 %	19.25 %	6.75 %
Seat rail cover removal	3.50 %	3.50 %	0.00 %
Y/C seat de-installation from seat rail	7.50 %	0.75 %	6.75 %
Y/C seat de-installation	7.50 %	7.50 %	0.00 %
Y/C seat removal	7.50 %	7.50 %	0.00 %
Installation	66.50 %	53.75 %	12.75 %
Y/C seat transportation into aircraft	7.50 %	7.50 %	0.00 %
Y/C seat positioning into seat rail	7.50 %	4.50 %	3.00 %
Y/C seat installation (mechanical)	7.50 %	0.75 %	6.75 %
Seat rail cover installation	2.50 %	2.50 %	0.00 %
IFE connection	9.00 %	6.00 %	3.00 %
PSU connection	30.00 %	30.00 %	0.00 %
Y/C seat installation (electrical)	2.50 %	2.50 %	0.00 %

Table 7 *Time saving for the troubleshooting of a seat related IFE fault*

	Old Process	New Process	Saving
Costs	9600 USD	6130 USD	3470 USD

Table 8 *Reconfiguration costs*

Upper Price Limit for Intelligent Quick Fasteners Within the PAHMIR project it is planned to develop intelligent quick fasteners, which include nearly all of the technologies that are needed to achieve the proposed cost saving. These fasteners are more expensive then currently used fasteners and thus their costs have to be compared against the cost saving for reconfiguration. According to [18] a typical reconfiguration frequency is twice a year. Applying this frequency to the total life of an aircraft (30 years) gives 60 reconfigurations during the aircraft lifetime. This means at least 60 reconfigurations are needed to reach the break-even point. In Table 8 savings of 3470 USD are shown. The basis for the calculation is the two-class cabin layout of the A340-600 (36 B/C seats at 40 inch pitch and 383 Y/C seats at 32 inch pitch). Every seat row (consisting of two seats) needs to be fixed with two intelligent quick fasteners. This means 419 fasteners have to be installed. Based on the above assumption, the following break-even point is calculated:

$$0 = 3470\,USD \cdot 60 - 419 \cdot x \Rightarrow x = 497\,USD \tag{1}$$

Equation (1) shows that the break-even point is reached within 60 reconfigurations, if the upper price limit for one fastener is below 497 USD. The example cabin reconfiguration is a simple reconfiguration scenario and much more complex reconfigurations (e.g. including monument reconfiguration) are planned by airlines. In these scenarios the absolute cost saving is much higher.

4 Sumary

This study shows that trend analysis applied to components of the air conditioning system can most probably reduce delays from unscheduled maintenance by about 8 %. An example calculation shows that time consumption can be reduced due to improved reconfiguration concepts by about 36 %. The upper price limit for economic intelligent quick fasteners seems to be around 500 USD. The PAHMIR project tries to integrate technologies, which are necessary to gain these advantages, into a single component. This will be reached by combining trend analysis with location detection, component self testing, component identification and quick fastening technologies. The major aircraft manufacturers (Airbus and Boeing) as well as airlines show an interest in quick cabin reconfiguration and trend analysis to reduce maintenance and reconfiguration times and thus to save costs and gain a higher aircraft usage.

References

[1] Airbus Operations. *Orders and Deliveries.* 2008. URL: `http://www.cs.waikato.ac.nz/ml/weka/` (visited on 2008).

[2] Kirsten Weber. *Filter Clogging Indication of Recirculation Air Filters.* 2008.

[3] Airbus Operations. 'A330/A340 Reliability Report'. 2008.

[4] Airbus Operations. 'In-Service Reports database'. 2008.

[5] Federal Aviation Administration. *Aviation System Indicators - 1994 and 1995 Annual Report.* 1995.

[6] J. Poubeau. 'Methodology for Analysis of Operational Interruption Costs'. In: *FAST* 26 (2002).

[7] D. Scholz. 'Betriebskostenschätzung von Flugzeugsystemen als Beitrag zur Enwurfsoptimierung'. In: *Deutscher Luft- und Raumfahrtkongress*. 1995.

[8] D. Scholz. *DOCsys - A Method to Evaluate Aircraft Systems*. 1998.

[9] A. Cook, G. Tanner and S. Anderson. *Evaluating the true cost to airlines of one minute of airborne or ground delay*. 2004.

[10] Eurocontrol. *Cost of Delays - Notes*. 2006.

[11] Josef Kolerus and Johann Wassermann. *Zustandsüberwachung von Maschinen: Das Lehr- und Arbeitsbuch für den Praktiker*. Expert-Verlag, 2011.

[12] Air Transport Association of America. *MSG-3: Operator/Manufacturer Scheduled Maintenance Development*. 2007.

[13] Airbus Operations. 'Cabin Reconfiguration Concept A350XWB Issue 2.0'. 2008.

[14] Airbus Operations. 'Maintenance Task Analysis'. 2008.

[15] Mike Gerdes. 'Technical Note: PAHMIR Project Description'. 2008.

[16] TZ Intevia. *Aeropace Applications*. 2008.

[17] Institut du Transport Aérien. *Costs of Air Transport Delay in Europe*. 2000.

[18] Airbus Operations. 'MTTC reconfiguration topics'. 2007.

Paper II

Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters

Mike Gerdes[1] and Dieter Scholz[1]

[1] Aircraft Design and Systems Group - Aero
Hamburg University of Applied Sciences
Berliner Tor 9, 20099 Hamburg, Germany

This paper has been published as:
M. Gerdes and D. Scholz, “Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters”, in *Deutscher Luft- und Raumfahrtkongress 2009 : Tagungsband - Manuskripte*, DLRK, Aachen, 08.-10. September 2009.

Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters

Mike Gerdes and Dieter Scholz

Abstract

Complex systems are commonly monitored by many sensors. When one of the sensors fails, the current state of the system can not be calculated or the information about the current state is not complete. For that reason sensor failures are one of the main error sources of a system. Thus sensors that deliver significant information about the system state need to be redundant. This paper shows how to calculate the significance of the information that a sensor gives about a system by using modern signal processing and artificial intelligence technologies. It also shows how significant features can be extracted, evaluated from a set of data samples, how difficult it is to find an optimal parameter and sensor set and that it is possible to reduce the size of needed data by 97%. The paper concludes analyzing the results of experiments that show that the methods can classify different errors with a 75% probability.

Keywords: Feature Extraction, Decision Trees, Air Conditioning, A340

1 Introduction

New and better monitoring approaches are required for condition monitoring, because systems become complexer. This is especially true for aircraft systems, where a high availability is required. The air conditioning system is interesting to monitor, because it not only supplies the passengers with fresh and tempered air; it also cools most of the electronic systems in the aircraft. Recirculation fans and filters are used for removing and cleaning a part of the used air from the cabin and direct the used air into the air mixing chamber, where it is mixed with fresh air and returned into the cabin. The recirculation filters can get clogged and the fans can get broken. If this happens the comfort of the passengers is reduced, because clogged filters smell and a broken fan causes smoke. It is difficult to predict, when a filter gets clogged, because the clogging of the filter depends on the usage of the aircraft. If a filter gets clogged then the fans might get broken faster and thus cause more costs and damage. Monitoring both components can save a lot of trouble. The recirculation fan and filter system consists of active (fans) and passive (filter) parts and is one of the few systems in the aircraft that uses condition monitoring.

Condition monitoring requires reliable sensors. If a sensor fails the information about the system is incomplete and thus it is recommended to install significant sensors redundantly. One problem of current condition monitoring technologies is, that it is difficult to calculate which sensors give significant data about the condition of a system and thus have to be installed redundantly. Sensor data needs to be processed to gain more information, than just time domain values. Frequency data and information about patterns in the sensor data can be useful to gain information about the condition of a system.

2 Condition Monitoring of Recirculation Fans and Filters

Today only a limited number of methods for monitoring the condition of recirculation fans and filters is used on aircraft. Older aircraft monitor only very rudimentary the air conditioning system, while newer aircraft sometimes use trend monitoring for the air conditioning. Most of the technologies that are used for fan and filter monitoring have disadvantages that make them not suited for complex condition monitoring. Some of the most common technologies, for recirculation fan and filter monitoring, will be explained below.

Pressure Switch: On many aircraft pressure switches are used to monitor recirculation fans or filters. These switches generate a failure message, if the pressure reaches a defined threshold [1].

Differential Pressure Sensor: Newer aircraft use differential pressure sensor to monitor filters and fans. The pressure difference is used as an indicator for the clogging of the filter. To get a clogging level and a forecast, measured values are compared against a curve, which relates the pressure difference to clogging. A filter clogging can be detected up to 100 hours in advance [1].

Vibration Sensor: Boeing uses on the 767 and 747 [2] vibration sensing systems to monitor fans. These systems are mounted on top of a fan and shut the fan off, if a certain vibration intensity is exceed. This concept tries to prevent serious damage to the air conditioning subsystems.

All presented monitoring concepts use the raw sensor data without additional signal processing and can only perform simple monitoring tasks. They operate without significant knowledge of the system and often only differentiate between a no fault and faulty state. To be able to get more knowledge about the system, more and different sensors have to be applied and the sensor data needs to be processed further.

3 Background

The methods that are presented and used in this paper are from the field of information theory and artificial intelligence. This section will show the basic principles that are used for the concept in Section 4.

3.1 Information Gain

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 [3]. The information entropy is also called information and is calculated as shown below in equation 1. $P(v_i)$ is the probability of the answer v_i.

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) log_2 P(v_i) \tag{1}$$

The information gain from an attribute test (setting the value of a node in a tree, see Figure 3.2 for an example) 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 [3].

$$Gain(X) = I(\frac{p}{p+n}, \frac{n}{p+n}) - \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} \cdot I(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}) \tag{2}$$

3.2 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 [4]) was the first algorithm to construct decision trees. ID3 had some problems and was improved. The improved version of ID3 is C4.5 [5]. 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). The algorithm uses the concept of information gain to choose attributes from the data and build a decision tree. The algorithm in pseudo code is:

```
1. Check for base cases
2. For each attribute a
    (a) Find the normalized information gain from splitting on a (see below)
3. Let a_best be the attribute with the highest normalized information gain
4. Create a decision node that splits on a_best
5. Recurse on the sub-lists obtained by splitting on a_best, and add those nodes as children of node
```

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 favored [5]. 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.

$$Split\ Info(X) = -\sum_{i=1}^{n} \frac{p_i + n_i}{p + n} log_2(\frac{p_i + n_i}{p + n}) \tag{3}$$

Gain ratio is the normalized information gain and is defined as shown in equation 4 [5].

$$Gain\ Ratio(X) = \frac{Gain(X)}{Split\ Info(X)} \tag{4}$$

A resulting decision tree (for the decision if you can play outside) may look like the tree in figure 3.2.

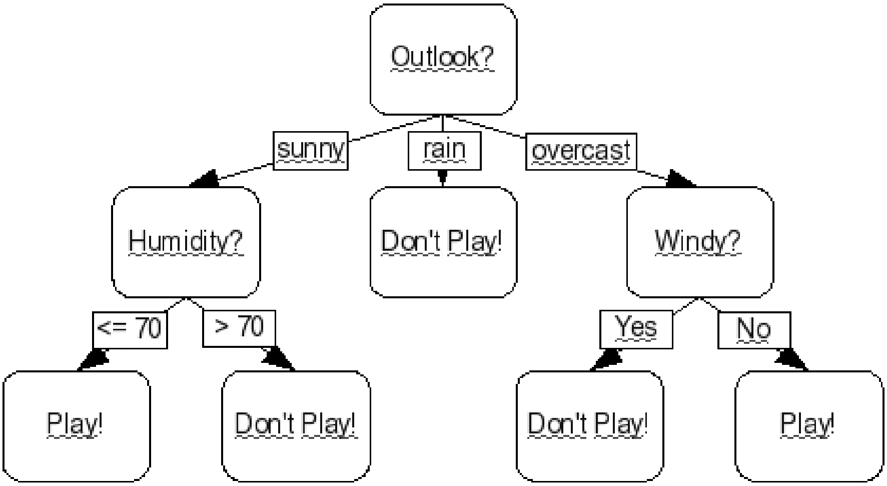

Figure 1: Example Decision Tree

3.3 Feature Extraction

The standard application of decision trees is pattern recognition (classification), learning and decision making. The tree is used to make a decision for new data, based on old knowledge. Either a sample is classified based on prior analyzed data or a decision is made based on prior ’experience’.

Using a decision tree for feature extraction works a bit different. The goal is not to classify new data, instead the goal is to build a tree, classify given data and find the attributes with the highest information entropy. During testing, some attributes can be disabled or deleted and their influence on the condition monitoring can be tested.

4 Sensor Optimization by Feature Extraction for PAHMIR

This section shows how the concepts of the preceding sections are applied for high pressure recirculation fan and filter monitoring within the PAHMIR project (Preventive Aircraft Health Monitoring for Integrated Reconfiguration [6]). Sensor optimization 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 [7].
- Optimizing the processing of sensor data [8].
- Optimizing the information gain of sensors.

In this paper sensor optimization has the meaning of sensor optimization, where identifying significant sensors in a number of available sensors that give the most information about a system, and thus increasing the information gain, is the goal. Signal processing methods are also counting as sensors in this context (e.g. frequency information and average values). Optimization in this paper is finding the sensors with the highest information gain of the data. The data can be raw sensor data, like amplitudes or processed data like frequency data. The calculation of the *information gain* and the sorting of the data is done by the *C4.5* algorithm that is used to construct decision trees for classification problems. Data samples are first processed, then the C4.5 algorithm is applied and finally the data is analyzed. Raw data samples are normally processed before they are used as an input. This means that the data is transformed into the correct input format for algorithms, it is enhanced with additional information and it is even possible to compress the data as it will be shown. Processing is done with Matlab and does not use any special toolboxes.

Data samples can generally be divided up into two categories of data: high frequency data and low frequency data. Low frequency data is defined as data with a sampling frequency less then 20 Hz. High frequency data is any data with a higher sampling rate then 20 Hz.

The low frequency data will not be processed beside bringing the data into the correct data format for the algorithm. There is too little data to perform significant frequency analysis and compression.

The high frequency data will be processed with the following steps: First the data is transformed into the frequency domain, then noise reduction is applied to the data, after that the frequency data is partitioned into small blocks (the size of the blocks depend on the data see Section 6.1)and finally every block is enhanced with additional information.

4.1 Fast Fourier Transformation and Partitioning

The fast Fourier transformation takes a number of time-domain samples and transforms them into the frequency domain. Basis of the FFT algorithm is the discrete Fourier transformation, which is defined as shown in equation 5 with $x_n, \ldots, X_{N-1}$ as complex numbers.

$$X_k = \sum_{n=o}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \ldots, N-1 \tag{5}$$

A fast Fourier transformation is performed in O(N log N) operations [9]. The fast Fourier transformation (FFT) is done with the Matlab function fft. A full transformation with the sampling frequency is done. After the fast Fourier transformation is done, the frequencies are divided 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 behavior of the block-signal over the time.

4.2 Noise Reduction

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 Matlab code 1.

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

Listing 1: Noise Reduction

4.3 Additional Information and Data Compression

Every block of the sampled data is enhanced with additional information. This information is added to give the following algorithm more information about the signal in the time and the frequency domain. The added information is for the time domain:

- The maximum amplitude of each block
- The mean amplitude of each block

In the frequency domain the following information is added:

- The mean frequency power of each block
- The maximum frequency power of each block
- The frequency with the highest power of each block
- The number of peaks that are higher then a defined magnitude of the mean frequency power

The Matlab code 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. Experiments showed that the added information is more useful for the algorithm, then the actual raw data. This allows to compress the data. The information of 100 frequencies (100 frequencies are used to simpler convert the result into percent values) is thus reduced down to four attributes (maximum and mean power, the frequency with he maximum power and the number of peaks). A similar result is achieved in the time domain. Instead of calculating the amplitude for each frequency in the time domain, only two attributes (maximum and mean amplitude) are calculated for 100 frequencies.

$$
\begin{aligned}
Freq\_Info &= 4 \cdot \frac{Frequencies}{BlockWidth} && (6)\\
Time\_Info &= 2 \cdot \frac{Frequencies}{BlockWidth} && (7)\\
Total\_Info &= Freq\_Info && (8)\\
&\quad + Time\_Info \\
&= 6 \cdot \frac{Frequencies}{BlockWidth} \\
Normal\_Info &= 2 \cdot Frequencies && (9)\\
Compression &= \frac{Total\_Info}{Normal\_Info} && (10)\\
&= \frac{3}{BlockWidth}
\end{aligned}
$$

Thus the needed data is reduced to 3 % if $BlockWidth = 100$ and $Frequencies = 11000$. The results of the analysis of the data with and without ($BlockWidth = 1$) compression will be shown in the section 'Result Analysis'.

5 Experiment

To show the performance and concepts of the algorithm, experiments were performed with different parameters. The data for the experiments and the feature extraction was sampled with an autonomous box, 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 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. Data from the following conditions was collected:

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

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-cross-validation (all samples are used for testing and training and the cross-validation process is repeated 3 times).

5.1 Calculating the Decision Tree

The decision tree is calculated with the open source Java software WEKA [10]. 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.

5.2 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.

6 Result Analysis

This section analyzes the results of the data processing of the previous section. First the different experiments with the different parameters are analyzed and evaluated in the next step the results of the best parameter configuration are taken and analyzed more closely.

6.1 Parameter Evaluation

This section shows the processing results of the different input sets, based on the parameter variation. The influence of a parameter is judge by the number of correct classified samples for every input set. Finding an optimal set of all parameters for the given samples, that give the lowest overall false classification rate, is a very complex problem. The complexity of the problem is so high that it is not possible to solve the problem in a fixed time, instead heuristic methods have to be used to find an optimal solution.

Default Parameters

The first calculation was performed using the default parameters (see Table 1). The following results in Table 2 were gained.

The numbers imply that about three quarter of the test cases are correctly classified. When looking in more detail at the classification distribution Table 3 results.

Parameters	Possible Values	Default Value
Block Width (see 4.1)	5/50/100/200	100
Noise Reduction Factor (see 4.2)	0/1/2/5	0
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 Maximum Frequency	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	0.0001

Table 1 *Processing Parameters*

Correct Classified	False Classified
73.4 %	26.6 %

Table 2 *Results for the Default Parameter Set*

Sample Class	Classified as No	Classified as Small	Classified as Big
No	755	103	42
Small	175	218	57
Big	41	61	348

Table 3 *Distribution of Wrongly Classified Samples*

Correct Classified	False Classified
92.7 %	7.3 %

Table 4 *Results for the Default Parameter Set with no Additional small Weight Samples*

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

Table 5 *Distribution of Wrongly Classified Samples with no Small Weight Samples*

The majority of the samples was correctly classified. For samples with no additional weight and a big additional weight the classification was very good, while samples with a small additional weight were often classified as samples with no additional weight. The results are still good, because the small attached weight was really light and sensing accuracy was not very high. When only no additional weight and big additional weight samples are used then the number of wrongly classified samples goes down to 5 %.

Block Width

Table 6 shows the results, when the block width varies.

The decreasing numbers imply that at some point an optimal block width can be reached, at which a minimum of false classified samples is obtained.

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

Table 6 *Results for Block Width*

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

Table 7 *Noise Reduction*

Maximum Amplitude	False Classified
Yes	26.6 %
No	26.5 %

Table 8 *Results for Maximum Amplitude per Block*

Noise Reduction

Table 7 shows the experimental results for a varying noise reduction.

Maximum Amplitude

The calculation of the maximum amplitude can be turned on or off. The Tables 8 and Table 9 show the results.

These results show that the maximum amplitude does not have a big influence on the classification in this problem. This is even more interesting to notice, because amplitude is the value that the vibration sensors record and that can be taken as an input without additional processing.

Mean Amplitude

The Tables 10 and 11 show the influence of the mean amplitude values.

The mean amplitude seams to have little influence on the accuracy.

Global Maximum Amplitude	False Classified
Yes	26.6 %
No	26.6 %

Table 9 *Results for Global Maximum Amplitude*

Mean Amplitude	False Classified
Yes	26.6 %
No	27.7 %

Table 10 *Results for Mean Amplitude per Block*

Global Mean Amplitude	False Classified
Yes	26.6111 %
No	26.6111 %

Table 11 *Results for Global Mean Amplitude*

Maximum Frequency Power

Table 12 and Table 13 show the results of the parameter variations for the maximum frequency power.

The maximum power seams to have a small influence on the classification.

Maximum Frequency Power	False Classified
Yes	26.6 %
No	25.0 %

Table 12 *Results for Maximum Frequency Power per block*

Maximum Frequency Power	False Classified
Yes	26.6 %
No	26.6 %

Table 13 *Results for Global Maximum Frequency Power*

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

Table 14 *Results for Frequency with Highest Power per Block*

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

Table 15 *Results for Global Frequency with Highest Power*

Frequency with Highest Power

Table 14 and Table 15 show the results of the parameter variations for the frequency with the maximum power.

The maximum power seams to have a small influence on the classification.

Mean Frequency Power

Table 12 and Table 13 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.

Mean Frequency Power	False Classified
Yes	26.6 %
No	22.8 %

Table 16 *Results for Mean Frequency Power per Block*

Mean Frequency Power	False Classified
Yes	26.6 %
No	26.6 %

Table 17 *Results for Global Mean Frequency Power*

Number of Peaks	False Classified
Yes	26.6 %
No	21.8 %

Table 18 *Results for Number of Peaks per Block*

Number of Peaks

Table 18 and Table 19 show the influence of the number of peaks for the calculation.

The number of peaks does have an even bigger influence on the classification then the mean frequency power and the false classification rate can be improved by nearly 5 %.

Peak Border

The peak border (the value that defines what is a peak) also influences the calculation. Table 20 shows this.

Results for the peak border show no clear trend, but the numbers suggest that an optimum exists.

Number of Peaks	False Classified
Yes	26.6 %
No	26.6 %

Table 19 *Results for Global Number of Peaks*

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

Table 20 *Results for Peak Border*

Peak Border	False Classified	Tree Size
1	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

Table 21 *Results for Confidence Factor*

Confidence Factor

The confidence factor determines how much the decision tree is pruned and also has an influence on the classification accuracy.

The results of the variation of the confidence factor shows the problem of over-fitting. The less pruning is performed the more samples are wrongly classified. Over-fitting is reduced when pruning is used.

6.2 Sensor Optimization

To be able to define the set of sensors that give the most information about the fan condition a decision tree has to be generated and evaluated. To find the attributes, which contain the most information about the system, a decision tree with a low false classification rate is needed.

Feature Extraction

To extract significant attributes from the decision tree a parameter set is needed that generates a tree with a low false classification rate. However it is not possible to take a parameter set of only the best attribute settings that were calculated in the experiments, because the parameters

Correct Classified	False Classified
73.0 %	27.0 %

Table 22 *Results for the Modified Parameter Set*

Sample Class	Classified as No	Classified as Small	Classified as Big
No	741	104	55
Small	163	225	62
Big	63	39	348

Table 23 *Distribution of Wrongly Classified Samples*

influence each other in a highly complex way. For example, if we deactivate the calculation of the mean frequency power and the number of peaks (two values which improved the false classification rate if turned off) then we get the classification results in Table 22 and Table 23.

Table 22 shows that the false classification rate gets worse and indicates that both values are highly connected. Thus to find an optimal set of parameters special algorithms are needed, as it was mentioned in Section 6.1. Using the default parameters, which represent a good parameter set, and the sample data results in a decision tree that is partly as shown below.

```
Block109MeanPower <= 0.01
|   Block112PeakNum <= 11
|   |   Block106PeakNum <= 8
|   |   |   ...
|   |   Block106PeakNum > 8: small
|   Block112PeakNum > 11
|       Block123MeanPower <= 0.007: no
|       Block123MeanPower > 0.007: small
Block109MeanPower > 0.01
    Block110PeakNum <= 1
    |   Block11PeakNum <= 3
    |   |   ...
    |   Block11PeakNum > 3
    |        ...
    Block110PeakNum > 1
        Block111MeanPower <= 0.009
        |   ...
        Block111MeanPower > 0.009
            ...
```

Listing 3: Part of the Decision Tree

It is evident that the most important feature in the tree is the mean frequency power of block 109. Followed by other attributes like the number of peaks. The tree contains no nodes of the “global” attributes (e.g. Global Mean Power). That means that they give nearly no information for the classification problem. The block peak number and the block mean power are the most significant features or attributes for the given problem and the samples. It is interesting to notice here that the correct classification rate increases, when one of both features is not used. This indicates the complex problem of finding the optimum parameter set for a given problem.

Sensor Selection

Based on the feature extraction in the previous subsection it is possible to decide, which sensors and processing methods need to be considered for monitoring the system. For the experiments only two “sensors” need to be considered to gain a fairly good classification.

Application for the Recirculation Fans and Filters

The experiment shows good results. To validate the method and experiments two more complex tests are planned to be performed. A test on aircraft will be performed that collects real world operation samples. These samples of a well functioning aircraft will contain no information about failure cases, however they will contain much noise from the operational environment of the aircraft. The additional noise is needed, because the algorithm will later also operate in the real world and therefore need to be well adapted to work with noise. Because it is difficult to collect information about failure cases in a test on aircraft a second test will be performed. This second test will be constructed and installed in a hangar, where recirculation fans and filters will be monitored and different failure cases will be simulated. This test set up allows gathering of samples in different failure conditions with low noise. Information processing (noise reduction, FFT) is well supported by state of the art digital signal processors and it is possible to evaluate the information nearly in real time.

7 Summary and Recommandation

The results of the experiment show that a fairly good failure classification can be made with the recommended algorithms and methods. While the classification is not perfect, it allows to decide, which features and sensors contain the most information about a system. The experiment used only simple PC fans, but even with those simple devices it was possible to evaluate the method and detect what weight was attached to the fans.

References

[1] Kirsten Weber. *Filter Clogging Indication of Recirculation Air Filters.* 2008.

[2] Inc. Inflight Warning Systems. 2009. URL: `http://www.iwsus.net` (visited on 01/07/2009).

[3] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[4] J. Ross Quinlan. 'Induction of Decision Trees'. In: *Mach. Learn* (1986), pp. 81–106.

[5] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1-55860-238-0.

[6] Barbara Stumpp. *Noch mehr Komfort für zukünftige Fluggäste*. 2009.

[7] Michael J. Smith. 'Sensor Location & Optimization Tool Set'. In: *Chemical Biological Information Systems Conference*. Albuquerque, 2005.

[8] Charles R. Farrar et al. 'Sensing and Sensor Optimization Issues for Structural Health Monitoring'. In: *23rd Aerospace Testing Seminar*. Manhatten Beach, 2006.

[9] C. Sidney Burrus. *Fast Fourier Transforms*. 2008. URL: `http://cnx.org/content/col10550/1.18`.

[10] Machine Learning Group at the University of Waikato. 2009. URL: `http://www.cs.waikato.ac.nz/ml/weka/` (visited on 01/07/2009).

Paper III

Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems

Mike Gerdes[1] and Dieter Scholz[1]

[1] Aircraft Design and Systems Group - Aero
Hamburg University of Applied Sciences
Berliner Tor 9, 20099 Hamburg, Germany

This paper has been published as:
M. Gerdes and D. Scholz, “Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems”, in *3nd International Workshop on Aircraft System Technologies*, AST 2011, TUHH, Hamburg, 31. März - 01. April 2011.

Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems

Mike Gerdes and Dieter Scholz

Abstract

In the PAHMIR (Preventive Aircraft and Health Monitoring) project pattern recognition and signal analysis is used to support and simplify the monitoring of complex aircraft systems. The parameters of the signal analysis need to be chosen specifically for the monitored system to get the best pattern recognition accuracy. An optimization process was developed that uses global heuristic search and optimization to find a good parameter set for the signal analysis. The computed parameters deliver slightly (one to three percent) better results than the ones found by hand. In addition it is shown that not a full set of data samples is needed. Genetic optimization showed the best performance.

Keywords: Genetic Algorithm, Optimization, Feature Extration, Signal Analysis, Global Heuristic Search

1 Introduction

An aircraft consists of many complex systems, which together define the state of the aircraft. Many systems are difficult to monitor or they give only little information to the aircraft maintenance systems. In [1] it was analyzed how much money could be saved, if a faulty system is replaced before it fails. Faults leading to a delay, can be prevented. It is either possible to replace the system on a fixed interval like it is commonly done in aircraft maintenance or to monitor the condition of the system and predict when the system will fail. The condition monitoring approach needs a deep understanding of the system and its behavior. Ideally, a computer should be able to compute the condition of a system to reduce the amount of humans involved in the process. One approach is to let the monitoring system learn the behavior of a monitored system. Future faults can be extrapolated based on the condition of a system in the present and the past.

In the PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) project systems are monitored by watching the inputs and outputs of a system and then forecast an error based on these values. Focus of the monitoring is on sound and vibration. The monitoring data is preprocessed and then automatically analyzed by a pattern recognition algorithm. Pattern recognition is used to find patterns in the data that represent a certain condition of the system. In [2] signal analysis and machine learning are used to detect the condition of an experimental setup. The concept is half automated and does need much fine tuning by hand. This is because the process depends on several different parameters. Each of these parameters need to be adapted to the data. Goal of this paper is to automate the selection of an optimal parameter set for the preprocessing to generate data, which yields the best results when it is analyzed by the pattern recognition algorithm. These parameters include:

- Signal transformation from the time domain into the frequency domain
- Noise reduction
- Grouping of frequencies
- Calculation of the maximum and mean frequency power of every frequency group
- Calculation of the number of peaks of every group
- Transformation of the frequency groups back into the time domain
- Calculation of the maximum and mean amplitudes
- Calculation of the maximum and mean values of the complete signal

The process for the condition monitoring in PAHMIR is shown in Figure 1. Input data is recorded and then is sent into a preprocessing step, where noise is removed and additional information is extracted from the data. The data is sent to a pattern recognition algorithm that tries to categorize the data after it is prepared. The performance of the pattern recognition algorithm is influenced by the number of available samples to learn from, the data preprocessing and the algorithm itself.

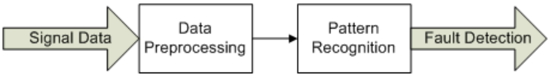

Figure 1 *The PAHMIR condition monitoring process*

1.1 Signal Data

The proposed concept can work with any kind of input data, however it is assumed that the data is a discrete signal with more than one data point. A sampling frequency of higher than 1 kHz is required. If a lower frequency is used, then the preprocessing needs to be adapted to that frequency. The signal source does not matter, it can be sound, vibration, temperature power consumption, weight or magnetic flow data as long as it is a one dimensional time series source. If more than one data source is used or a data sample does have more than one dimension, then the preprocessing algorithm also needs to be adapted or the data needs to be transformed. The simplest way is to have one preprocessing step for every dimension of the data and then concatenate the preprocessed data before giving it to the pattern recognition.

1.2 Preprocessing

Noise and the amount of data are reduced and additional information is added during preprocessing to the data. First, the data is transformed into the frequency domain, where the noise is reduced. Then, frequencies are grouped. It is possible that the frequency span of the groups overlap each other. E.g. if the frequencies 1 to 50 belong to one group and have an overlap of 50 %, then the second group contains the frequencies from 26 to 75 and the third group contains the frequencies from 51 to 100. Mean and maximum power are calculated for every frequency group, as well as the number of peaks. Then each group is transformed back into the time domain, where the mean and maximum amplitudes are calculated. The mean and maximum frequency power and mean and maximum amplitude of the complete signal is calculated as a last step. Table 1 shows the parameters of the preprocessing and the possible values.

1.3 Pattern Recognition Training

Pattern recognition belongs to the area of artificial intelligence. It is used to find patterns in data that allows the algorithm to categorize that data. First the algorithm has to "learn" or find the patterns in the data and construct a function or algorithm that represents that data. After that, new data samples can use the function or algorithm to categorize the new data based on the experience of the old data. The

Parameter Name	Possible Values	Unit
Block width	0–1000	Hz
Block overlap	0–50	%
Noise reduction	0–5	–
Calculate the mean amplitude for each block	true or false	–
Calculate the maximum amplitude for each block	true or false	–
Calculate the mean frequency power for each block	true or false	–
Calculate the maximum frequency power for each block	true or false	–
Calculation the number of peaks for each block	true or false	–
Minimum Value of a peak	0–5	–
Calculate the overall mean and maximum values	true or false	–

Table 1 *Preprocessing parameters*

original concept of PAHMIR uses Decision Trees [2]. In this paper two other algorithms (Bayesian Network and Support Vector Machine) are compared with Decision Trees (see Section 2.1).

2 Methods

The basic concept of the PAHMIR condition monitoring process was modified in two ways. First different pattern recognition algorithms were tested and second an optimization process was developed to find a good parameter set for the data preprocessing.

2.1 Pattern Recognition

The influence of different pattern recognition algorithms was analyzed in experiments. Only algorithms that can evaluate a data sample fast, can be understood and checked by a human and have low hardware requirements can be used for the pattern recognition in PAHMIR. The restrictions for algorithms are given by the aircraft environment. Three algorithms were chosen for comparison: Decision Tree, Bayesian Network and Support Vector Machine.

Decision Trees

Decision Trees are very easy to understand. They are unidirectional trees. The leafs are the categorizes of the data, while the nodes are if-then decisions. Decision trees use the concept of information gain for learning and finding the attribute in the data that divides the data so that the most information is gained. [3] explains decision trees in further detail.

Bayesian Network

Bayesian networks are trees like decision trees. A Bayesian network does not use information gain to construct the tree, instead Bayesian networks use conditional probability. Conditional probability means the probability that an attribute has a certain value, if the value of another related attribute is known. Nodes of the network contain the probabilities of choosing the next node based on knowledge of the stare of some other nodes. Bayesian networks are explains in further detail in [3] .

Support Vector Machines

Support Vector Machines or SVMs are a complete numerical approach. The goal is to separate the data samples by a linear function into classes. If it is not possible to separate the data into classes with a "straight line" then the data set is transformed into a higher dimensional space until it is possible to separate the data samples. A special kind of function is used to perform the transformation. The data is transformed by weighted dot products. The weight vector is called support vector. [3] gives more details on SVMs.

2.2 Optimization Concept

The preprocessing and the corresponding parameters influence the accuracy of the pattern recognition strongly. Tests of the learning function showed that the solution space is not linear and does have many local minima. Such a problem is difficult to optimize with traditional methods, because the solution space is very large. An automated parameter configuration is needed to find an optimal parameter set that will improve the performance of the pattern recognition. Heuristic search methods, which search for a minimum or maximum, can help to find a good solution to the optimization problem. The goal of the optimization is to maximize the percentage of the correctly classified data samples.

An optimization step is included in the process shown in Figure 1. The optimization takes part during the learning step, which is needed before any data can be evaluated. Figure 2 shows the modified PAHMIR condition monitoring process. First a sample data set is preprocessed with a random parameter set. The preprocessed data is then fed into a pattern recognition algorithm that searches for patterns. The resulting algorithm is then tested and yields an accuracy percentage. Then the optimization loop is entered and a new parameter set is chosen, based on an optimization algorithm (Greedy Search, Simulated Annealing or Genetic Algorithm). After the new parameter set is chosen the loop starts again. The output is a parameter set, which is used for the data preprocessing in the PAHMIR condition monitoring process (Figure 1). All three algorithms are adapted to the problem and start from a given data base of parameter sets. The parameter range of the mutations (for Simulated Annealing and the Genetic Algorithm) is also adapted and all algorithms work with the same in and output to be able to chain them.

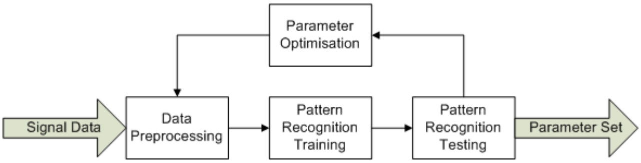

Figure 2 *PAHMIR condition monitoring process with optimization*

Traditionally the pattern recognition algorithms are optimized by modifying the underlying algorithm. This optimization concept doesn't touch the optimization algorithm. It optimizes the input data so that a high accuracy is gained. As a side effect the chosen parameters show which signal processing steps are important and which are not needed for a successful classification.

Greedy Search

Greedy search is the simplest optimization algorithm of the three that were analyzed. Greedy search starts at a random starting point in the parameter space and compares the value of that point with the value of a random neighbor. If the neighbor performs better, then the neighbor is chosen, otherwise the current parameter set stays at the starting point. In [3] is a more detailed explanation of the Greedy Search algorithm.

In this paper the greedy search algorithm is implemented without any modifications. A neighbor is defined by a block size of 1 Hz to 50 Hz and a block overlay that varied by up to 10 % of the starting point. All other values can vary by one point and have a 50 % probability of changing. Greedy Search stops, if the best value does not change for 30 steps.

Simulated Annealing

Simulated annealing is a more complex variant of greedy search. In simulated annealing the algorithm may choose a worse neighbor depending on a probability, which decreases with the difference between the performance of the neighbor and the number of performed steps. With simulated annealing it is possible to move away from a local maximum and not to get struck. There is a more detailed explanation of the Simulated Annealing algorithm in [3].

A new neighbor is found by varying the block size up to 500 Hz and the block overlay up to 20 %. The range of both values decreases with the number of performed function evaluation. All other values vary by one point and have a probability to do so of 50 %. The function that controls when a worse neighbor is selected is a linear function. A neighbor performance value may be up to 20 % worse than the current point. This value decreases linearly over time until it reaches 0. Simulated Annealing stops, if 480 optimization steps are executed. The best value is returned and not the current value.

Genetic Algorithm

The genetic algorithm is a more complex simulated annealing. It evaluates different points in parallel, chooses the best and creates new variations of those by combining and changing the parameter sets. With the genetic algorithm it is possible to search a wider area in the problem space and a higher chance to find the global maximum. The algorithm performs multiple evaluations in parallel thus it is slower than the other two algorithms. In [3] is a more detailed explanation of the Genetic Algorithm.

The genetic algorithm used the same mutations as the Simulated Annealing algorithm. Reduction of the mutation variance over time was used to force a better convergence. New children were created by taking the block width of one member, the block overlay of another, one part of the other remaining parameters from a third parent and the rest parameters form a fourth parent. The first third of the population was left unchanged, the rest of the population can mutate. Genetic evolution stopped, if 20 generations with 24 members have been evaluated.

3 Experiments

Four different experiment setups were used to evaluate the concept, optimization algorithms and pattern recognition algorithms. First the influence of different input data sizes, then the influence of choosing different starting points was tested. Third different optimization algorithms (Greedy Search, Simulated Annealing and Genetic Algorithm) and combinations of them were tested with the sample data. Then different pattern recognition algorithms (Decision Trees, Bayesian Networks and SVMs) were tested.

Data which was used for the experiments was generated in a test rig (see Figure 3). The test rig simulates a part of the air recirculation system of an A340-600 aircraft. Valves control the two inlets and the outlet. For the experiment data different valve conditions are chosen and then the vibration of the recirculation fan was recorded. One sample was recorded every 10 seconds. Every condition was monitored for 4 minutes, which results in 24 samples per condition. In total 25 conditions were recorded. All experiments used a 10-fold cross-validation to check the performance of the calculated pattern recognition.

3.1 Sample Data

In the first experiment the influence of the number of samples on the calculation time and the pattern recognition accuracy was evaluated. The optimization and learning process was tested five times with an increasing number of samples (5, 10, 15, 20, 24) per class. 24 samples was the maximum possible number of samples per class (all recorded

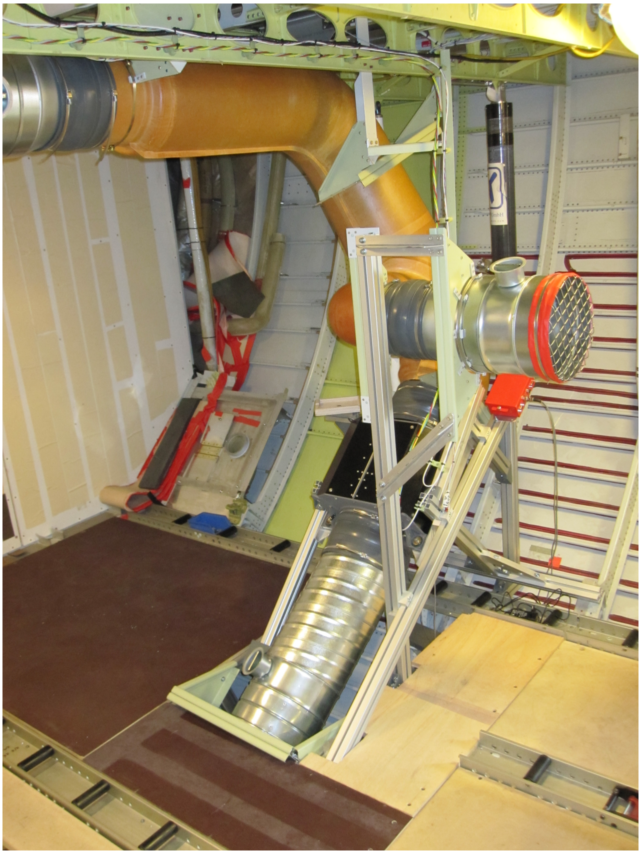

Figure 3 *Test rig for recording experiment data*

samples). The best solution that was found with the reduced sample set was used to classify the full sample set. This was done to show if it was possible to train the algorithm with a reduced data set and gain the full classification accuracy. The genetic algorithm setup was used (see subsection 2.2).

3.2 Starting Points

An experiment was performed to show the effect of starting at different points. Genetic algorithm was used with 20 samples per class. The algorithm was evaluated ten times with different randomly selected starting populations.

3.3 Optimization Algorithm

The three different optimization algorithms (Greedy Search, Simulated Annealing, and Genetic Algorithm) were tested alone with different parameter sets. Then Simulated Annealing and Genetic Algorithms were chained so that one produced starting points for the other one. The idea was to use one algorithm to find a good starting point and that the following algorithm can use that point to perform even better, than it would normally alone. The single algorithm experiments and the chained experiments used the same number of function evaluations to be comparable. All algorithms started at the same starting point. The genetic algorithm generated additional random starting points up to the needed population size.

- Greedy Search
- Simulated Annealing
- Genetic Algorithm
- Simulated Annealing and Genetic Algorithm
- Genetic Algorithm and Simulated Annealing

3.4 Pattern Recognition

In this experiment the performance of the three algorithms for pattern recognition (Decision Tree, Bayesian Network and Support Vector Machine) were compared when they use a Genetic Algorithm. The run time of the algorithms was measured beside the percentage of correctly classified samples .

4 Results

This section shows the results of the experiments as they were described in the previous section.

4.1 Number of Samples

The pattern recognition accuracy between two sample data bases with a different number of samples varies a lot (Table 2). As it is visible the accuracy of the smaller sample base and the full sample base are very similar. The difference is only for the 5 data sample base significantly. This observation can be used to reduce the training time a lot, if only

Data Samples per Class	Correctly Classified	With 24 Samples	Calculation Time
5	90 %	96 %	1166 s
10	96 %	97 %	2767 s
15	97 %	96 %	3572 s
20	98 %	96 %	6182 s
24	98 %	98 %	6700 s

Table 2 *Evaluation of different data sample sizes*

half of the available data samples are taken. Another advantage of this approach is that the other half of the data samples can be used to verify the classification results as a testing data set.

4.2 Starting Points

The pattern recognition accuracy depends on the selected starting points (Table 3). Calculation times depend on the selected parameters and vary. A maximum accuracy of 99.4 % was reached in the experiments. This value is higher than the value in Table 2 where the best value was 98 %. The random selected starting points and the randomness of the optimization algorithm are the cause of this effect. With a larger population and more generations it is possible to reduce that effect and get a better convergence. Still all starting points reached a very good accuracy.

4.3 Optimization Algorithm

The selection of the algorithm greatly influences the performance of the optimization as this section shows. Greedy Search and Simulated Annealing are more random and converge slower.

No Optimization

A calculation without an optimization step was done to be able to judge and evaluate the results of the different optimization algorithms. 24 random parameter sets were generated, evaluated and the best parameter set was selected. This resulted in an accuracy of 97.5 % and took 517 s.

Starting Population Number	Correctly Classified Samples	Calculation Time
1	98.6 %	3967 s
2	98.1 %	5596 s
3	98.3 %	5653 s
4	98.6 %	4643 s
5	99.4 %	4492 s
6	98.8 %	4352 s
7	98.6 %	4403 s
8	98.6 %	4638 s
9	98.9 %	4850 s
10	98.9 %	4568 s

Table 3 *Evaluation of the influence of different starting points*

Greedy Search

The best result of the Greedy Search algorithm was 97.7 % and the calculation time was only 1250 s. This is as expected. Greedy Search is a really fast algorithm but it also can get stuck in a local maximum easily. To get better results the algorithm needs to be executed more than one time, which negates the speed advantage.

Simulated Annealing

Simulated Annealing had about the same speed as the Genetic Algorithm of about 5605 s. This is unsurprisingly due to the fact that both algorithms evaluated the function 480 times. Simulated Annealing achieved an accuracy of 97.7 %, which is similar to the Greedy Search algorithm and a bit worse than the Genetic Algorithm. The problem space contains many local maxima and is very huge. Simulated Annealing does not get trapped in a local maximum as fast as Greedy Search, but can also fall in that trap if the problem space contains very many local maxima.

Genetic Algorithm

The Genetic Algorithm had the highest accuracy with 98 %. It needed 5418 s to finish. The Genetic Algorithm delivers the similar results as

Pattern Recognition Algorithm	Correctly Classified Samples	Calculation Time
Decision Trees	94.4 %	1381 s
SVM	99.4 %	12312 s
Bayesian Network	99.4 %	2791 s

Table 4 *Evaluation of different pattern recognition algorithms with optimization*

simulated annealing. It searches at multiple places at once and then chooses the best ones to continue. However none of the experiments that were performed delivered the global maximum.

Simulated Annealing and Genetic Algorithm

Using Simulated Annealing to create a base population works quite well, however the results are not better than using the Genetic Algorithm alone (98.6 %) and the calculation time was twice as long.

Genetic Algorithm and Simulated Annealing

The idea to use the best parameter set of the Genetic Algorithm as a starting point for Simulated Annealing works also well and results in an accuracy of 98.3 %. The calculation time again was twice as long as for a single algorithm.

4.4 Pattern Recognition

Table 4 shows accuracy of the different pattern recognition algorithms. To be able to use the Bayesian Network algorithm all values need to be discredited (an attribute can only have a pre-defined value). If numerical values are used, then Weka needs to discretize the attribute values automatically, which results in an "No memory left" error. To limit the amount of needed memory and make the calculation feasible the maximum number of blocks was limited to 15, 10 data samples per class and the bandwidth of the input data was only 7.5 kHz (half the bandwidth of the original data samples).

It is visible in Table 4 that the SVM performs the best and Decision Trees perform worst. The Bayesian Network algorithm works well be-

Pattern Recognition Algorithm	Correctly Classified Samples	Calculation Time
Decision Trees	91.7 %	71 s
SVM	98.9 %	1860 s
Bayesian Network	98.9 %	193 s

Table 5 *Evaluation of different pattern recognition algorithms without optimization*

cause of the reduced amount of attributes, however the Decision Tree algorithm seems to suffer and performs weak.

In Table 5 are the three different algorithms tested with the same parameter set and without optimization. It is visible that SVM delivers again the best results. There is a minimal optimization included in the calculation. 24 random parameter sets were generated (the same as the starting parameter set for Table 4) and then the parameter set with the best performance was used.

While Bayesian Networks deliver good results, they are not the best. The Table 5 also shows that the calculation time depends on the number of the blocks. If that number is restricted, then all algorithms perform significantly faster. The optimization process doesn't give a significant improvement in this setup. This is due to the fact of the solution space was much smaller and that the random starting points where good.

5 Conclusions

The results show that an optimization can increase the performance of the signal analysis and pattern recognition. However the increase is less than 5 %. Still it is possible to push the accuracy up to 99.4 %. Even the short searches with a small population showed a good performance compared to choosing parameters by hand, which is nearly equal to chose a random parameter set. One goal of the project to reduce the amount of expert knowledge to use the system is achieved. The concept works well if a significant number of data samples are available.

Another advantage of the concept is that it can be parallelized without much work, if a genetic algorithm is used. The members of the population can be spread over the available processor. With parallelization it is possible to reduce the computation time a lot and a much larger space can be searched in the same time.

References

[1] Mike Gerdes, Dieter Scholz and Bernhard Randerath. ‘Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration’. In: *2nd International Workshop on Aircraft System Technologies, AST 2009 (TUHH, Hamburg, 26./27. März 2009)*. 2009.

[2] Mike Gerdes and Dieter Scholz. ‘Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters’. In: *Deutscher Luft- und Raumfahrtkongress 2009 : Tagungsband - Manuskripte (DLRK, Aachen, 08.-10. September 2009)*. 2009.

[3] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

Paper IV

Fuzzy Condition Monitoring of Recirculation Fans and Filters

Mike Gerdes[1] and Dieter Scholz[1]

[1] Aircraft Design and Systems Group - Aero
Hamburg University of Applied Sciences
Berliner Tor 9, 20099 Hamburg, Germany

This paper has been published as:
M. Gerdes and D. Scholz, “Fuzzy Condition Monitoring of Recirculation Fans and Filters”, in *CEAS Aeronautical Journal*, December 2011, Volume 2, Issue 1-4, pp 81-87.

Fuzzy Condition Monitoring of Recirculation Fans and Filters

Mike Gerdes and Dieter Scholz

Abstract

A reliable condition monitoring is needed to be able to predict faults. Pattern recognition technologies are often used for finding patterns in complex systems. Condition monitoring can also benefit from pattern recognition. Many pattern recognition technologies however only output the classification of the data sample but don't output any information about classes that are also very similar to the input data. This paper presents a concept for pattern recognition that output similarity values for decision trees. The concept can be used on top of any normal decision tree algorithms and is independent of the learning algorithm. Performed experiments showed that the concept is very reliable and it also works with decision tree forests to increase the classification accuracy.

Keywords: Decision Trees, Fuzzy Decision Tree Evaluation, Air Conditioning, A340

1 Introduction

Aircraft systems can be very complex and difficult to monitor. A good condition monitoring does not only depend on good sensors and a good model but also interpreting of the numbers. Interpreting sensor data however is not a trivial task. Classification and condition monitoring as it is shown in this paper also reduces the amount of information that needs to be monitored. Often an expert is needed to interpret the data and make a meaningful "classification".

With pattern recognition and classification it is possible to replace the human expert by a computer algorithm. Pattern recognition and classification are a typical application of artificial intelligence technologies. With pattern recognition is possible to find complex patterns that are hidden in the sensor data of a system. Classification maps input data to a class based on learned or given pattern. In case of system monitoring the class can be a failure or condition of a system.

Most classifiers map input data to one output class (Figure 1; the input is mapped to "class 1"). However for monitoring systems and reducing

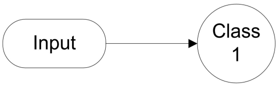

Figure 1 *Common classification mapping of one input to one class*

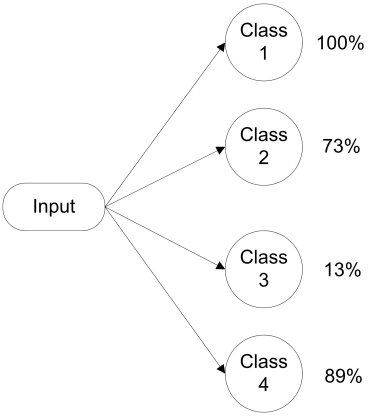

Figure 2 *Classification mapping of one input to one class and output of similarity*

NFF failures it is useful to know the probability of an input to belong to all possible classes, instead of only the most likely class (Figure 2; the input is still mapped to "class 1", but the input also matches the pattern of "class 4" in 89 % of the criteria for "class 4). Artificial Neural Networks (ANN) can output the similarity of an input to all classes if one output node is available for every class. But ANNs have their own disadvantages compared to other methods.

In the project PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) decision tree classifier are used to interpret system sensor data and classify the system conditions. Decision trees are a simple and fast classifier with feature extraction, learning and a high robustness. Decision trees are a method from the area of artificial intelligence and are used for decision making and classification. They are often binary trees, where each node has an if-then-else function on an attribute of the sample data. Advantage of decision trees is the simple structure, the fast calculation and the inherent feature extraction. The ID3 algorithm (Iterative Dichotomiser 3, published by J. Ross Quinlan in 1986, used to generate decision trees [1]) was the first algorithm to construct decision trees. ID3 had some problems and was improved.

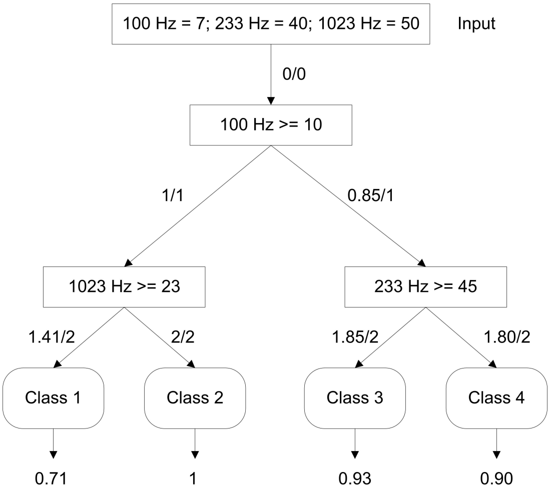

Figure 3 *Fuzzy Decision Tree Evaluation*

The improved version of ID3 is C4.5 [2]. 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). The algorithm to build a decision tree uses the concept of information gain to choose attributes from the data and build the tree. Output of a decision tree is only the most likely class for one data sample.

To get more information out of the classification the decision tree evaluation algorithm was modified to output the probabilities of all classes. This modification was done without changing the learning algorithm for decision trees and can be used with any binary decision tree. An example is shown in Figure 3. The figure shows a decision tree for deciding what class a sample belongs to. Input is one frequency power observation (100 Hz, 233 Hz and 1023 Hz). Each node in the tree is related to an attribute. The path from one node to another is labeled with the taken decisions. Weights are based on the distance of the attribute value from the sample value in the observation.

The highest calculated value (between 0 and 1) for every possible decision is outputted at the end of the evaluation. Thus we gain a measure for how close an input data sample is to other classes. In case of the example the evaluation of the decision tree classifies the sample as an example of Class 2 (weight = 1). The sample is also in similar to

Class 3(weight = 0.5) and 4(weight = 0.45).

Advantage of this approach is that the similarity of a data sample to different conditions can be calculated and that the decision tree generation algorithm doesn't need to be changed. It should be noted that the concept is designed in such a way that the characteristics of one attribute (mean, minimum, maximum ...) doesn't need to be known. In addition the input data for learning and classification doesn't have to be modified in any way to fit the new algorithm.

Carrying these ideas further, this paper now shows how a fuzzy evaluation of decision trees using numerical attributes can be used to gain more information about a system than just the current condition.

2 Concept

This section shows the concept that was developed for a fuzzy evaluation of decision trees. A decision tree is generated traditionally, but uses a fuzzy evaluation for the evaluation of an input sample:

1. A decision tree is generated with C4.5 based on labeled data samples.
2. The decision tree is evaluated using the fuzzy evaluation concept.

The fuzzy evaluation calculates an output for every leaf. Goal is to output one value for every leaf. Every output value of a leaf is between 0 and 1 to show the similarity of the sample to every class (see Figure 3 for an example). Similarity is a function of the distance of an input sample to other classes but its own. The similarity is calculated based on a weighting function of the decisions taken (node evaluations) to classify the sample. For condition monitoring and maintenance the similarity can indicate possible other faults and conditions of a system.

The proposed fuzzy evaluation works as follows:

1. Every path between two nodes/leafs has two labels: PathDepth and PathWeight.
2. Choose the root node
3. PathDepth and PathWeight are 0 for the root node
4. Evaluate the node condition

5. Calculate path labels:
   (a) If the evaluation of the condition is true then label the "True" path with PathDepth +1 and PathWeight + 1.
   (b) If the evaluation of the condition is false then label the "False" path with PathDepth +1 and PathWeight + 1.
   (c) Label the other path to sub-nodes with PathDepth +1 and PathWeight + $nodeweight(AV, SV)$. See Equation 1 and 2
6. Choose a new node, with a labelled path to its parent.
7. Use the path labels for PathDepth and PathWeight.
8. If the node is no leaf then continue at 4.
9. If the node is a leaf then return $\frac{PathWeigth}{PathDepth}$ and the leaf label and continue with another node.

It is assumed that the tree is a binary tree with numerical attribute values. Every returned value for a leaf is between 0 and 1. The weight for each node is calculated as shown in Equation 1. Equation 2 limits the function values. Nodeweight can be between 0 and 1. AV (attribute value) is the value of the sample for the condition of the current node. SV (split value) is the value of the node, which marks the border of the condition e.g. the split value of “$x <= 17$” is 17.

$$nodeweight(AV, SV) = 1 - \frac{|AV - SV|}{2SV} \tag{1}$$

$$\begin{aligned} If\ nodeweight(AV, SV) < 0 \\ \Rightarrow nodeweight(AV, SV) = 0 \end{aligned} \tag{2}$$

It is possible to use a different weighting function. The requirement is that the weight for each node needs to be between 0 and 1 for the algorithm. If the weight is higher then the result at a leaf can be higher than 1. The given weighting function was chosen for different reasons. One reason was that a black box approach for used for the monitored system and it is unknown what input values of samples mean. It can not be assumed that the learning data samples for the algorithm contain the max or min values or even represent a mean. These circumstances make it difficult to use an absolute values. The only values that are available without adding additional information or calculation to the algorithm is the split value of a node. Also the correct classified class needs to be 1. Choosing $2 \cdot SV$ as the limit where the weight will be 0 was an arbitrary choice which is based on some tests with input data. It is possible to use a higher or lower value.

The concept was designed with only numeric values in mind. However it is possible to use Boolean and discrete values as well with a small modification of the process.

Nodes with Boolean attributes If Boolean attributes are used instead of numerical values then the weight is assumed to be 0.

Nodes with discrete values If a node does have more than two children (one for every possible value of the attribute) then only the path linked to the evaluation of the node condition is weighted with 1. For every other unlabeled path to a child the weight needs to be calculated separately.

2.1 Fuzzy Decision Tree Evaluation Example

Below is an example. Each path from one node to another is labeled with PathWeigth and PathDepth in the form of PathWeigth/PathDepth e.g. 2/5. In Figure 4 an input data set is given. The input contains the power of the frequencies at 100 Hz, 233 Hz and 1023 Hz. At the first node the 100 Hz value is checked if it is larger or equal than 10. The power is not larger than 10 (it is 7) so right path which is the *false* path get +1/+1. For the other path a 1 is added to the PathDepth and a 0.85 (the similarity) to the PathWeigth.

In the next step the 1023 Hz node is evaluated. The input data does have a power at 1023 Hz which is higher than 23, so the *true* path gets +1/+1 for a total of 2/2 (+1/+1 to the 1/1 from the parent path). The other path gets a +0.41/+1 for a total of 1.41/2.

The same process is done for the left hand node. Evaluating the node gives a +1/+1 to the *false* path and a +0.95/+1 to the other path.

In the last step (Figure 7) the weight of the leafs and the classes are calculated. The input data is classified as class 2, it does have a similarity of 0.71 to class 1, 0.93 to class 3 and 0.9 to class 4. A similarity of 0.93 means that the values don't have to move much to switch the classification result to class 3.

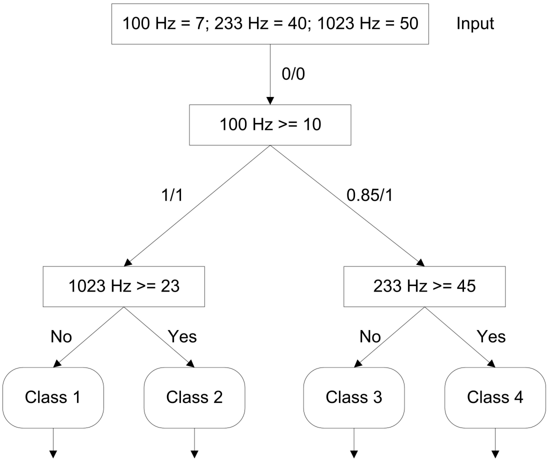

Figure 4 *Sample fuzzy evaluation step 1*

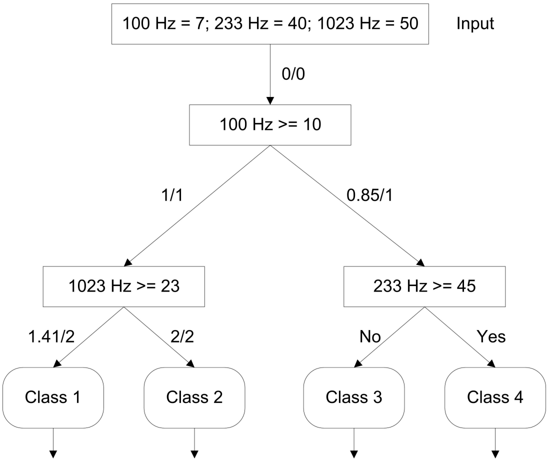

Figure 5 *Sample fuzzy evaluation step 2*

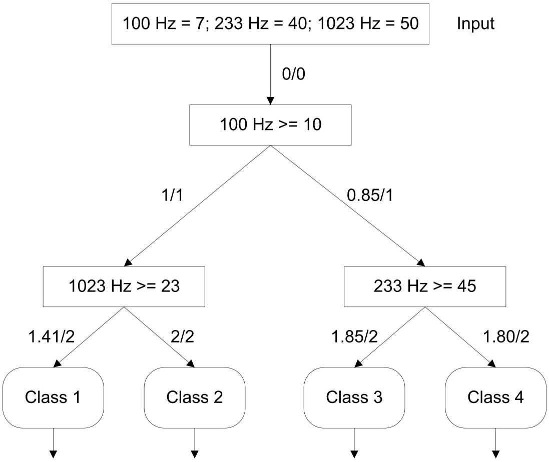

Figure 6 *Sample fuzzy evaluation step 3*

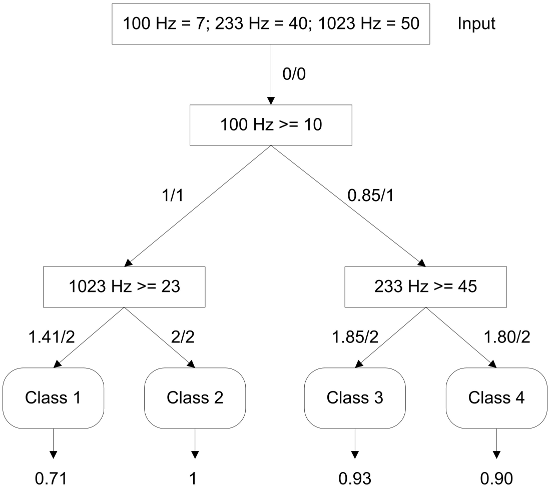

Figure 7 *Sample fuzzy evaluation step 4*

2.2 Classification Accuracy

The accuracy of a classification is defined as the number of correct classifications (CC) in relation to the number of all classifications (TS), where TS is the number of correct classification plus the number of wrong classifications (see 3).

$$Classification\ Accuracy = \frac{CC}{TS} \tag{3}$$

Classification Accuracy is a number between 0 and 1, which can be transformed into a percentage value.

The accuracy of the decision tree evaluation is not influence by the fuzzy decision tree evaluation. This is because of the fact that the "correct" paths are weighted with "1" while all other paths are weighted with a number lower than 1. The "correct" path will always have the highest weight. However it is possible that the classification accuracy of fuzzy decision tree evaluation is slightly lower than for standard decision tree evaluation because of the limits of numerical computations. If the weight of a node is very close to 1 than it is possible that due to rounding errors it is counted as a 1 instead of a value that is lower than 1. This case happened in some experiments and was more frequent if a 32 bit Java floating point data type was used instead of 64 bit Java floating point data type.

2.3 Fuzzy Decision Tree Forest Evaluation

A decision tree forest can also be evaluated using the proposed concept. Each decision tree in the forest is evaluated separately using fuzzy decision tree evaluation. If all trees are evaluated then the weights (similarities) for a class from all trees are added together and are divided by the number of trees. Basically taking the average of a class over all trees in the forest. This is done for all available classes.

3 Experiments

Experiments were performed to ensure the performance of the concept. Goal of the experiments was to evaluate the fuzzy decision tree evaluation for a single decision tree and for a decision tree forest.

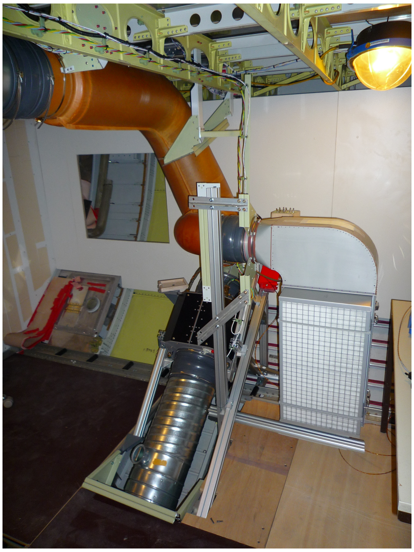

Figure 8 *Test Rig at Airbus Operations GmbH*

0/0	0/15	0/30	0/45	15/0	15/15
15/30	15/45	30/0	30/15	30/30	30/45
45/0	45/15	45/30	45/45	60/0	60/15
60/30	60/45	75/0	75/15	75/30	75/45
90/0	90/15	90/30	90/45		

Table 1 *Experiment Conditions (valve1/valve2)*

3.1 Database

Data for the experiments was recorded on a test rig. The test rig resembles a part of the air conditioning system of the A340-600. Focus of the test rig is the HP recirculation fan and filter. 29 different conditions were recorded. The test rig is equipped with two valves, one valve at each end of the two tubes. It is possible to set the valves to "open", "15°", "30°", "45°", "60°", "75°" and "closed". 20 samples of one second duration were recorded for every condition of the test setup.

Valve1 is the inlet valve and *Valve2* is the outlet valve. The following conditions were recorded:

In addition to these 28 conditions an "not running" condition was recorded. In total 580 samples were recorded.

3.2 Setup

The Java software Weka was used to perform the experiments. J48 (C.45) was used to built the decision tree. A signal analysis was done before those data samples were inputted to Weka. The signal analysis includes noise reduction, Fast Fourier Transformation, mean/max calculation ... see [3] for details. A random parameter set for the signal analysis was generated. 20 random samples of very class were selected for the decision tree generation.

3.3 Single Fuzzy Decision Tree Evaluation

The experiments themselves were very simple. A decision tree was calculated using the data samples and the signal analysis parameters. The decision tree was then evaluated with fuzzy decision tree evaluation. For comparison four neighboring classes were compared. To be able to evaluate the node weighting function (Equation 1), four other weighting functions were also tested (Equation 4, Equation 5, Equation 6 and Equation 7) with the four neighboring classes. The whole experiment was tested with 5 different data samples. All data samples belong to the "15/0" class.

$$nodeweight(AV, SV) = 1 - \frac{|AV - SV|}{0.1 \cdot SV} \tag{4}$$

$$nodeweight(AV, SV) = 1 - \frac{|AV - SV|}{10SV} \tag{5}$$

$$nodeweight(AV, SV) = exp(-|AV - SV|) \tag{6}$$

$$nodeweight(AV, SV) = \exp\left(-\frac{|AV - SV|}{SV}\right) \tag{7}$$

Each equation replaces Equation 1. Equation 2 is left unchanged. For each equation the same random parameter set and decision tree was used.

3.4 Decision Tree Forest

A decision tree forest with three trees was created for the experiments with a decision tree forest. Each tree does have a classification accuracy of a least 85 %. Every tree uses a different signal processing parameter set, which generates different learning input and data input for the evaluation. During the experiments the classification accuracy compared to a single decision tree with a classification accuracy of 85 % was evaluated. The fuzzy evaluated results for the same five data samples and neighboring classes as in the experiments for the single decision tree were also evaluated.

4 Results

This section shows the results of the experiments. First the results for a single fuzzy decision tree evaluation are shown and second the results for a fuzzy decision tree forest evaluation are shown.

4.1 Single Fuzzy Decision Tree Evaluation

Table 2 shows the averaged results for five fuzzy decision tree evaluations with data samples of class "15/0". The numbers show that the correct class is classified in the right way. For the other classes is the similarity to the "15/0" class not big, but visible. A reason for the small similarity between the different classes for all four data samples is that the input data is very similar to each other and contains a huge amount of noise. It is possible to increase the variance of the similarity if another equation for weighting is used. The results of the experiment confirm this. When Equation 4 was used the variance was the highest.

This was to be expected, because more paths in the tree were weighted with 0 instead of a number larger than 0. In accordance Equation 5 does have the lowest variance in the similarity.

Class	Eq. 1	Eq. 4	Eq. 5	Eq. 6	Eq. 7
0/0	0.994	0.890	0.999	0.990	0.989
15/0	1.000	1.000	1.000	1.000	1.000
15/15	0.994	0.898	0.999	0.989	0.990
30/0	0.996	0.920	0.999	0.875	0.990

Table 2 *Averaged evaluation results*

Class	Eq. 1	Eq. 4	Eq. 5	Eq. 6	Eq. 7
0/0	0.996	0.918	0.999	0.989	0.992
15/0	1.000	1.000	1.000	1.000	1.000
15/15	0.993	0.874	0.999	0.970	0.984
30/0	0.998	0.966	0.999	0.848	0.997

Table 3 *Averaged fuzzy decision tree forest evaluation*

4.2 Fuzzy Decision Tree Forest Evaluation

This section shows how the results improve, when a decision tree forest is used.

Classification Accuracy

Using a forest with 3 decision trees increased the classification accuracy from 85 % per single tree to 95.4 % for the complete forest. This is a significant classification accuracy improvement which comes at the cost of three times the calculation time. However in the example application of air filter monitoring the time is not a critical factor.

Fuzzy Decision Tree Evaluation

Comparing the results of the same four classes that were evaluated for a single decision tree we get Table 3.

The results are similar to a single decision tree with fuzzy evaluation, but the average similarity and the overall classification accuracy is higher.

5 Conclusions

The concept for fuzzy evaluation of decision trees, which is shown in this paper can achieve the desired performance and delivers good results. It is possible to calculate a similarity measurement for classes in a decision tree without changing the algorithm to create the tree. Fuzzy classification plus decision trees are a powerful tool for condition monitoring especially in the aircraft environment. The fuzzy decision tree evaluation is easily understood, fast and small which makes it good for use in environments characterized by high safety requirements. With additional optimization of the weighting equation it is possible to increase the variance of the fuzzyfication.

References

[1] J. Ross Quinlan. 'Induction of Decision Trees'. In: *Mach. Learn* (1986), pp. 81–106.

[2] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1-55860-238-0.

[3] Mike Gerdes and Dieter Scholz. 'Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters'. In: *Deutscher Luft- und Raumfahrtkongress 2009 : Tagungsband - Manuskripte (DLRK, Aachen, 08.-10. September 2009)*. 2009.

Paper V

Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning

Mike Gerdes[1]

[1] Philotech GmbH
Brauereiweg 4, 21614 Buxtehude, Germany

This paper has been published as:
M. Gerdes, “Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning”, in *Expert Systems with Applications*, Volume 40, Issue 12, 15 September 2013, Pages 5021–5026.

Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning

Mike Gerdes

Abstract

Unscheduled maintenance of aircraft can cause significant costs. The machine needs to be repaired before it can operate again. Thus it is desirable to have concepts and methods to prevent unscheduled maintenance. This paper proposes a method for forecasting the condition of aircraft air conditioning system based on observed past data. Forecasting is done in a point by point way, by iterating the algorithm. The proposed method uses decision trees to find and learn patterns in past data and use these patterns to select the best forecasting method to forecast future data points. Forecasting a data point is based on selecting the best applicable approximation method. The selection is done by calculating different features/attributes of the time series and then evaluating the decision tree. A genetic algorithm is used to find the best feature set for the given problem to increase the forecasting performance. The experiments show a good forecasting ability even when the function is disturbed by noise.

Keywords: Decision Trees, Prediction, Approximation

1 Introduction

Unscheduled maintenance costs are a significant cost factor in aircraft operation[1]. Aircraft operators have significant increased costs, if an aircraft departure is delayed or canceled. One source of unscheduled maintenance is when a part of an aircraft needs to be replaced or repaired before the scheduled replacement time. The aircraft air conditioning and filtering system is such a system. The air filters clog faster or slower depending on the environment conditions where the aircraft is mainly operating. Filters clog faster in a moister environment and slower in a dry environment. A clogged filter system may not only cause a delay but also passenger discomfort. Currently the air condition system is monitored by using pressure sensors to detect changes in the air pressure. Forecasts are done by comparing the measurement against an empirical curve, which relates the pressure difference to the probability of clogging and a forecast for the mean time to clogging [2].

This paper describes a method to use machine learning to forecast the condition of a system. The method uses a decision tree to decide what the best method to forecast a future data point is and a genetic algorithm to adapt the decision tree to the current problem to improve the performance. The decision of the best forecasting method is based on learned patterns from past data. Motivation for this approach was to have a simple method to forecast the condition of the air conditioning system, which can adapt itself to different time series based on the operation of the aircraft and handle the influence of events on time series. The method should be easy to understand by an operator and should be able to adapt itself to different problems without much need for human interaction/expert. A time series of the system condition may be constant or linear for a long time, but suddenly an event happens and the time series changes significantly. Forecasting such a time series is difficult. The use of a decision tree enables the proposed method to use the best available forecasting method based on learned experience to adapt better to the new condition. An advantage of this method is, that it can use currently existing sensors and forecasting concepts to calculate the forecast. A genetic algorithm is used to increase the performance of the forecasting by searching for the optimal features set which generates the best decision tree for the given problem.

1.1 Time Series

A time series is a chronological sequence of observations on a particular variable[3]. This can be the production of a company, the DOW, a temperature, a pressure difference or a system condition. The history of a system condition can be seen as a single or multi dimensional time series. If the condition of a system is represented only by a single variable then the resulting time series is a one dimensional time series. If the condition is represented by two or more different variables then the resulting time series is a multidimensional time series. Prediction of future events and conditions is called forecast, the act of making such a prediction is called forecasting[3]. Common methods for time series forecasting are:

- Simple linear regression[4]

- Polynomial Regression[3]
- Multi Regression[4]
- Moving average[4]
- Exponential smoothing[4]
- Autoregressive integrated moving average (ARIMA)[4]

Marin Goluband Andrea Budin Posavec propose the use of genetic algorithms for adapting the approximation functions for forecasting[5], Animesh Chaturvedi and Samanvaya Chandra use quantitative data and a neural network to forecast financial time series[6].

1.2 Decision Trees

Decision trees are a method from the area of artificial intelligence and are used for machine learning[7]. They are often binary trees, where each node has an if-then-else function on an attribute of the sample data. More complex versions with more than two branches use a switch function. Training of the tree can be done with the ID3 algorithm (Iterative Dichotomiser 3, published by J. Ross Quinlan in 1986[8]). ID3 was the first algorithm to construct decision trees. ID3 had some problems and was improved. The improved version of ID3 is C4.5[9]. There are other algorithms to construct a decision trees available, including random trees. Decision trees are easy to understand and a decision/-classification can be calculated fast. A sample decision tree can be seen in Fig. 1.

1.3 Genetic Algorithms

Genetic algorithms belong to the class of heuristic local search algorithms. They evaluate multiple valid solutions (population), choose (selection) the best (fitness) solutions and create new variations of those by combining (crossover) and changing (mutation) the solution. The new set of solutions is now evaluated and the best ones are selected again and combined and changed. Each iteration of these steps is called generation. The search is finished when the algorithm calculated a certain number of generations or when an abort criteria is reached[7].

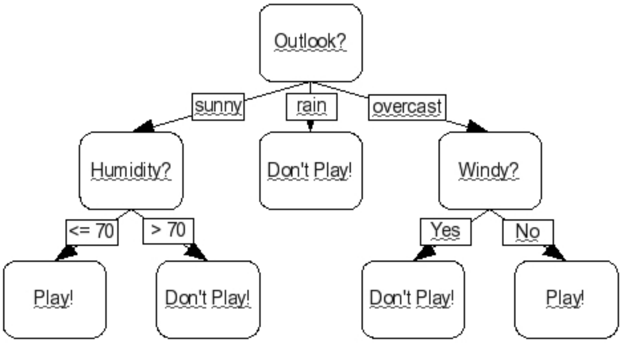

Figure 1 *A simple decision tree*

2 Method

The method proposed in this paper for time series forecasting is based on decision trees. The inputs to a decision tree are time series characteristics (e.g. maximum value, gradient) and the output is an approximation function/method for forecasting based on training data. Forecasting quality is increased by using a genetic algorithm[7] for optimizing the process parameters. Optimization of the process parameters allows the process also to adapt itself to different problems without human interaction. Training enables the forecasting process to use experience of past data to predict the future data points in a much more reliable way then without training. This is obtained due to the fact that the process can learn when to use a different forecasting function then the obvious, because of irregularities in the time series. These irregularities can be triggered by the occurrence of certain events, that change the future data points of the time series significantly e.g. switching from a simple linear behavior to an exponential behavior.

The process is divided into two parts, one part for training of the algorithm and optimizing the decision tree and one part for forecasting the time series. In the training part training samples are created, time series features are calculated, a forecasting method is selected and the decision tree is generated. Data points are forecasted, after the decision tree is generated and the process parameters are optimized. Each iteration of the forecasting process calculates a single future data point. With multiple iterations it is possible to calculate more data points. Variations of the default process can calculate multiple data points and are shown later in this section.

2.1 Training Process

The learning process takes much more time than the forecasting process and is only executed when new training samples are available and during the initial training. Goal of the training process is to find an ideal set of features of the times series that give the most information for finding the optimal extrapolation algorithm. Input to the training process is a data set with different features and the best extrapolation algorithm for the time series that the features represent. The learning process does have seven steps. The first four steps are executed only once to generate the input for the last three steps. All steps except the last one (process parameter optimization) are iterated multiple times to generate a random base population of decision trees for the parameter optimization with a genetic algorithm (last step).

Samples Decision trees and most other concepts from artificial intelligence need many data samples for learning and finding patterns. For a time series this means, that a time series should not be to short and/or that multiple time series are available. Sample time series should include all relevant conditions and events. The algorithm can only learn from past data and thus it cannot predict events that were not in the sample data.

Process Parameters The training process is controlled by multiple parameters. These parameters control how samples and time series characteristics are calculated. Process parameters are:

- Window size [numeric]. This parameter defines how many data points each data sample contains. These data points include past data and the data points to forecast. This can be a fixed or a varying number, which is different for each data sample.
- Window shift [numeric]. This parameter defines by how many data points the sampling window should be shifted to generate a new data sample from the time series. Window shift and window size define how many training samples are generated.
- Forecast horizon [numeric]. The parameter controls how large the forecasting horizon is. This means for how many future data points

the forecasting method should be calculated. The forecasting horizon can be from one data point up to all remaining data points in the time series. In the remaining document is a forecasting horizon of one data point used. Forecasting horizon cannot be larger than window size.

- Short Term Gradient calculation [Boolean]. This parameter defines, if the gradient for the data points in the current window shall be calculated.

- Long Gradient calculation [Boolean]. This parameter defines, if the gradient for the complete time series until the last data points shall be calculated.

- Mean value calculation [Boolean]. This parameter defines if the mean of the sample data points shall be calculated.

- Maximum value calculation [Boolean]. This parameter defines if the maximum of the sample data points shall be calculated.

- Minimum value calculation [Boolean]. This parameter defines if the minimum of the sample data points shall be calculated.

- Dimensions used for classification [Boolean list]. This parameter defines which dimensions shall also be used for characteristics calculation.

- Zero Crossing[Boolean]. This parameter decides if the number of zero crossings in the current window shall be calculated.

Other parameters may also be used. Including long term trends and short term trends is also possible by calculating the parameters for the current window and for the complete time series. The optimization algorithm then decides what long term and what short term parameters deliver the best results.

Time Series Features In this step the training samples are generated and time series features are calculated, based on the process parameters. First the time series is split into multiple smaller time series. The length of these smaller time series is defined by the window size. Data sample generation is done by shifting a n-data point window over the sample time series. See Fig. 2 for an example with a one point window shift and a window size of three. Other features may be used, depending on the problem and the need.

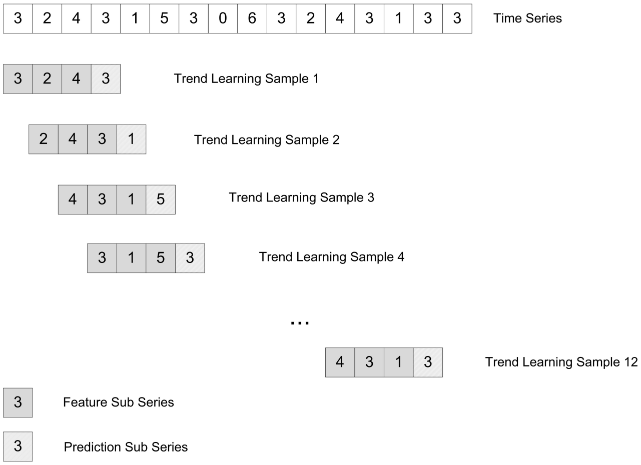

Figure 2 *Generation of trend learning samples with a sliding window*

Training Samples Next the decision tree output for the training time series needs to be calculated. For each possible approximation function/-method from a given list is the forecasting for the forecasting horizon calculated. Next is the squared forecasting error calculated. The best fitting approximation function/method (with the least forecasting error) defines the "class" of the data sample. The following example is a list of possible simple approximation functions (simple linear regression, polynomial regression and multi regression), each approximation function can contain up to four parameters (a, b, c and d):

$$
\begin{aligned}
&f(x) = a \cdot x + br \\
&f(x) = a \cdot x^2 + b \cdot x + c \\
&f(x) = a \cdot x^3 + b \cdot x^2 + c \cdot x + d \\
&f(x) = a \cdot e^{b \cdot (x+c)} + d \\
&f(x) = a \cdot log(b \cdot (x + c)) + d \\
&f(x) = a \cdot sin(b \cdot (x + c)) + d \\
&f(x) = a \cdot cos(b \cdot (x + c)) + d \\
&f(x) = 1 - a \cdot e^{b*(x+c)} \\
&f(x) = a \cdot b^{x+c} \\
&f(x) = a \cdot (b \cdot (x + c))^2 + d
\end{aligned}
\tag{1}
$$

The given list was used in the later experiments. In addition to the previous list of simple functions it is possible to use other forecasting methods. Moving average, auto regressive process, moving average process, ARMA, ARIMA, exponential smoothing ... can also be used as a function. [10] did some experiments for using complex functions to forecast the complete future time series instead of only one future data point.

Decision Tree Training The decision tree is trained after all samples have been created using any decision tree learning algorithm like ID3, C4.5, random trees ...

Performance After the decision tree has been calculated, it can be evaluated and tested on the sample time series. The training data samples are applied to the forecasting process to calculate the rest of the sample time series from each training sample. Next the performance is calculated by calculating the maximum squared error of the forecasting and use this value as the fitness of the decision tree. Different approaches for calculating the fitness of the decision tree like the maximum confidence range are possible. The confidence range describes how many data points in a row can be forecasted until the error (either relative or absolute) is greater than a given limit.

Process Parameter Optimization If the best performance of the generated decision trees is below a given limit, then a genetic algorithm is used to generate new valid solutions. New decision trees are generated until one the decision trees does have a performance above the given limit or until a certain number of generations have been calculated. Input to the genetic algorithm are the process parameters and the fitness is measured based on the performance of the decision tree. This does have the advantage that the genetic algorithm adapts the process parameters for the decision tree to the current problem by selecting the time series features which give the best performance. In addition this also reduces the need for a human expert to set the process parameters.

2.2 Forecasting Process

The forecasting process is an iterative process. During each interaction a new data point is forecasted. The forecasting process is simple and consist of the following steps:

Time Series Features The first step in the forecasting process is to calculate time series features based on the optimized process parameters and given past data points.

Decision Tree Evaluation Next step is to evaluate the time series features by using the generated decision tree. The result is an approximation function.

Next Data Point(s) Prediction The next step is to use the approximation function to extrapolate the next data point of the time series.

Add Result to Time Series The new calculated data point is added at the end of the given time series. All such calculated data points are the forecasting.

Iterate If more than one point shall be forecasted then the forecasting process is repeated with the new added data point as part of the past data points. Each iteration step calculates a new data point.

2.3 Method Summary

The method consist of two processes: one training process and one forecasting process. Both processes use the same given set of methods to forecast the next data point for a given time series. Input to both processes are short (5-20 data points) time series snippets. The time series snippets contain one additional point for the training. In the training process is a decision tree calculated that is than used for the forecasting. The decision tree was calculated using process parameters that were optimized by a genetic algorithm. The forecasting is done point by point. The decision tree is used to select the best forecasting method for the current given time series snippet.

2.4 Process Modifications

The method can be modified in different ways based on the problem at hand. One modification is to not only use the past data points plus one future point for calculating the approximation function, but to use more than one future data point to account for a long term trend (greater forecast horizon[4]). The forecasting would still calculate only one future data point, but the long term trend is considered for generating the decision tree.

Another modification would be to iterate the training process only once and then use the calculated forecasting method to forecast all remaining future data points of the time series. This modification works well when the training data not only includes one future data point, but multiple future data points up to all remaining data points of the sample time series. [10] shows this modification.

A regression tree[11] can be used, if a future data point shall be directly forecasted without a approximation function. This reduces the needed processing power, but the process needs to iterate to forecast multiple data points.

3 Experiments

To validate the process three experiments have been performed. All experiments used a time series that is typical for the change of a system condition over time[12]. The first experiment was about forecasting the time series without any noise. In the second experiment noise was added to the test samples, while the training samples were without noise. For the third experiment noise was added to training and testing samples to get a more realistic use case. The forecasting of future data points was started at three different points in the time series: 1/4, 2/4 and

3/4 of the function. The time series was forecasted until the end of the sample time series and the number of past data points were ten with a forecasting horizon of one.

The time series for the experiment was a one dimensional time series that contained 120 data points. The first 80 data points were calculated via $f(x) = \frac{0.1}{80} \cdot x$ the data points between 81 and 120 were calculated by the following equation $f(x) = (\frac{x-81}{120-81})^2 + 0.1$. Fig. 3 shows the time series.

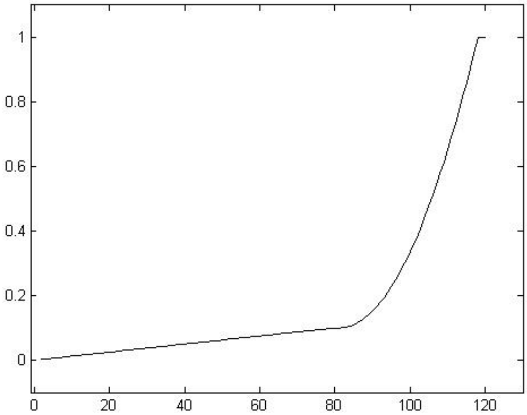

Figure 3 *Plot of the experiment function*

The quality of the forecasting was measured by six values. For every forecasting the maximum, mean and minimum forecasting error was calculated and the confidence range for an absolute error of 0.01, 0.05 and 0.1 was also calculated. The used genetic algorithm for the experiments had population of 50 members and 20 generations. New members were generated by dividing the process parameters into three parts and generating a child from three different parents. The mutation rate for each process parameter was 10 %.

3.1 Forecasting Without Noise

The first experiment was about forecasting the time series by using the method as it was presented. There was no noise in either training or forecasting data. In Fig. 4 the results of the forecasting are plotted.

The black line are the input points, the red line is the sample time series to predict, while the green line is the calculated forecasting, starting at data point 30, 60 or 90. The image shall show the advantage of using a decision tree. The image shows on the right side the forecasting using the best prediction method without using a decision tree to select the method. This is the same method that is used to generate training samples in the training process. On the left side is the same input but the proposed process is used to forecast the next data point. It is visible that the advantage of the decision tree comes, when the function suddenly changes (data point 80) 0-degree polynomial to a second-degree polynomial function. Here is the decision tree forecasting method much more accurate than the simple method that uses only the best fitting method.

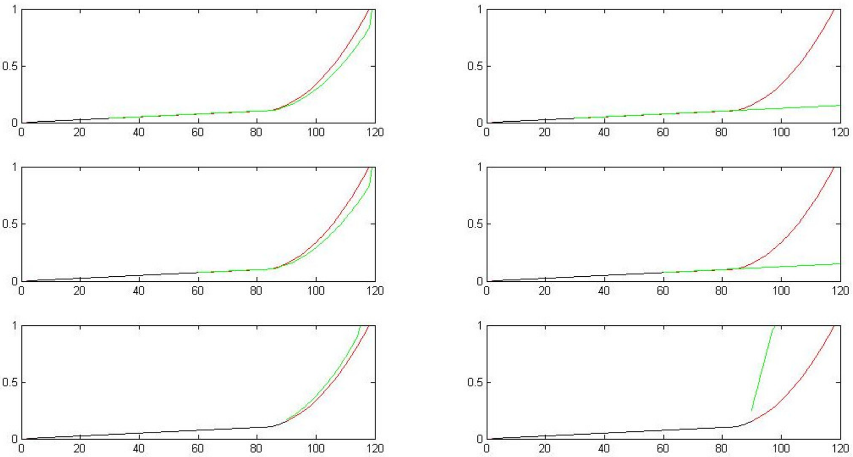

Figure 4 *Plot with different starting points and forecast*

Fig. 4 shows that the algorithm was quite good at following the time series. The maximum error was low and the algorithm was able to predict the time series nearly perfect. The preconditions for this forecasting were ideal (no noise) and the time series was pretty simple. The advantage of using a decision tree is clearly visible.

3.2 Forecasting with noisy test samples

In the second experiment the training samples were not changed, but the testing samples were modified by noise. A random number between

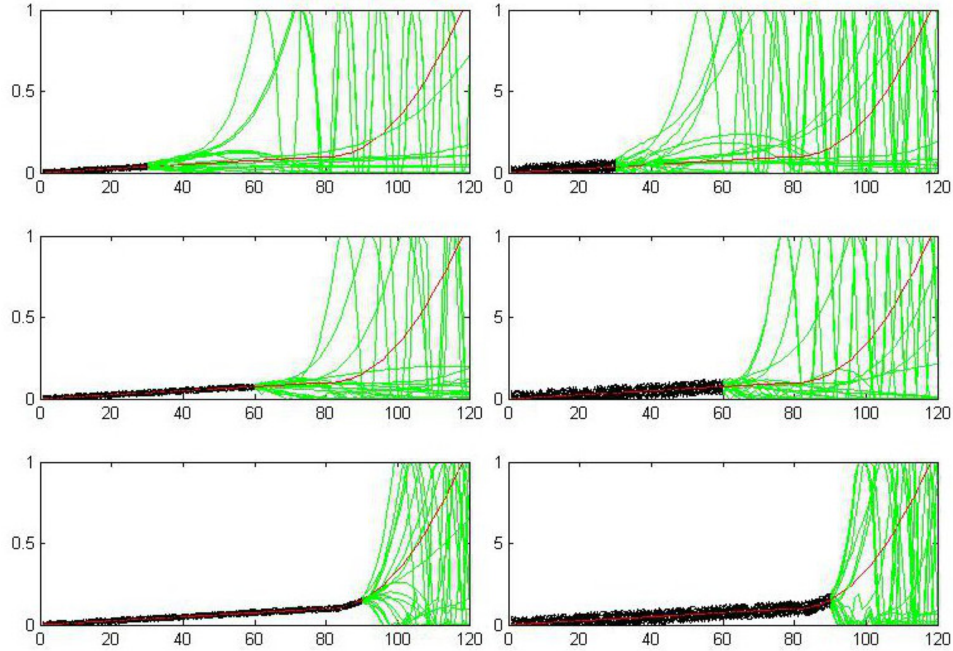

Figure 5 *Plot with different starting points, noisy test data and noise (left: 2 % and right: 5 %)*

-0.02 and 0.02 (2 % noise) or for the second test -0.05 to 0.05 (5 % noise) was added to each data point. Goal of the experiment was to check how well the method can work with noisy input data. Results are shown in Fig. 5. The test was repeated 20 times to show the forecasting results for different noisy input data.

Oscillations are caused by the used functions for the predictions like sin and polynomial functions. The forecasting accuracy was lower than without noise. If the starting point of the forecasting was at data point 60 then the results were especially bad. The algorithm was unsure at which position in the time series it was. The reason for this was the high impact of the noise on the time series features due to the relative small number of past data points (10). This let the decision tree make the wrong decision.

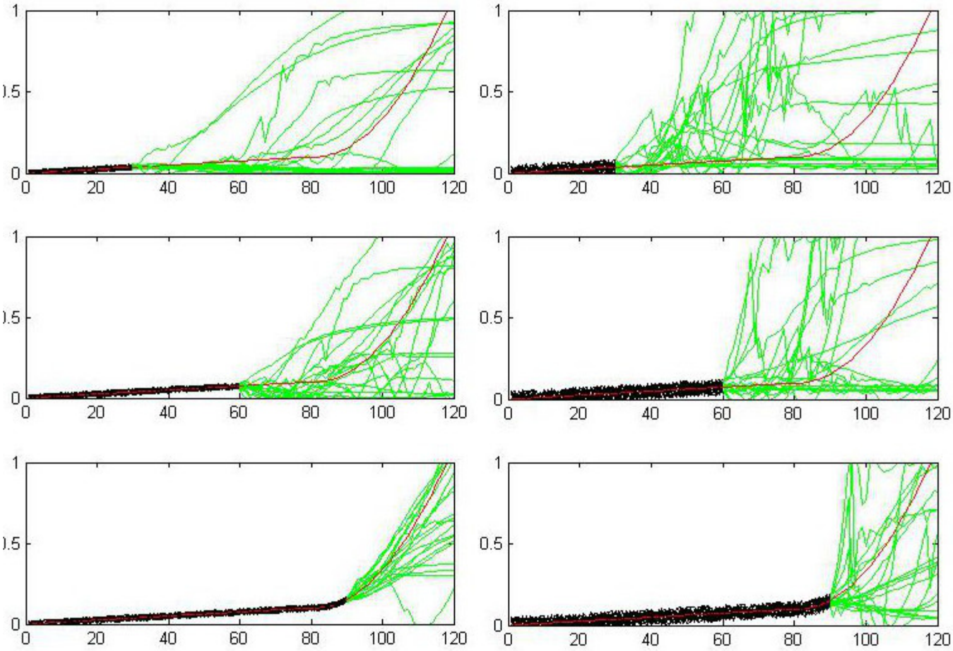

Figure 6 *Plot with different starting points, noisy training and testing data and noise (left: 2 % and right: 5 %)*

3.3 Forecasting noisy training and test samples

In this experiment the training data and testing data were noisy. Two tests with different noise levels were calculated. For each test three noisy training time series were used to generate training samples for decision tree calculation. Each data point in the time series was modified by a random number. This number was white noise between -0.02 and 0.02 (2 % noise) for the first test, -0.05 to 0.05 (5 % noise) in the second test. The results are shown in Fig. 6. The test was repeated 20 times to show the forecasting results for different noisy input data.

In the picture (Fig. 6) it is visible, that the accuracy of the forecasting is not so well, but this was to be expected. It is worth to note that only three noisy training time series were used. Still the predicted time series are similar to the original curve. The inaccuracy is created by the problem that the algorithm was not sure at which position it was and that the approximation only had ten noise points to calculate the next data point. With more training data and more past data points the algorithm will be able to perform better. With noise training and testing data the algorithm used different functions for the approximation and thus created no oscillations.

4 Conclusion

Experiments showed that the method is well suited for time series forecasting tasks. The advantage of using a decision tree is clearly visible and the performance of the forecasting is significantly improved. With more advanced forecasting methods it is possible to increase the performance. The method is able to adapt to different problems and the performance can be increased by using problem specific approximation functions/methods and process parameter optimization. A single future data point is forecasted (by offering a good approximation function) and then the process is iterated until a desired number of data points were forecasted. It is easily possible to train the algorithm to use an approximation function to forecast more than one data point if the problem desires this (long term forecasting). Forecasting quality of the method increases with the number of available past data points, more available features and more test and training samples especially when the training data is very noisy. The method can be enhanced by different concepts to have an increased quality of the forecast.

References

[1] Mike Gerdes, Dieter Scholz and Bernhard Randerath. 'Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration'. In: *2nd International Workshop on Aircraft System Technologies, AST 2009 (TUHH, Hamburg, 26./27. März 2009)*. 2009.

[2] Kirsten Weber. *Filter Clogging Indication of Recirculation Air Filters*. 2008.

[3] Bruce L. Bowerman and Richard T. O'Connell. *Forecasting and Time Series. An Applied Approach*. Duxbury Press, 1993.

[4] Douglas C. Montgomery, Lynwood A. Johnson and John S. Gardiner. *Forecasting & Time Series Analysis*. McGraw-Hill, Inc, 1990.

[5] Marin Golub and Andrea Budin Posavec. 'Using Genetic Algorithms for Adapting Approximation Functions'. In: *Proceedings of the 19th International Conference on Information Technology Interfaces ITI '97* (1997).

[6] Animesh Chaturvedi and Samanvaya Chandra. 'A Neural Stock Price Predictor using Quantitative Data'. In: *iiWAS 2004 - The sixth International Conference on Information Integration-and Web-based Applications Services, 2004, Jakarta, Indonesia.* Vol. 183. Austrian Computer Society, 2004.

[7] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, 2003.

[8] J. R. Quinlan. 'Induction of Decision Trees'. In: *Mach. Learn* (1986), pp. 81–106.

[9] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. ISBN: 1-55860-238-0.

[10] Adam Kret. *Statistische Analyse und Vorhersage von Zeitreihen zur vorausschauenden Wartung von Flugzeugkomponenten.* 2011.

[11] L. Breiman et al. *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks, 1984.

[12] Josef Kolerus and Johann Wassermann. *Zustandsüberwachung von Maschinen: Das Lehr- und Arbeitsbuch für den Praktiker.* Expert-Verlag, 2011.