DOCTORAL THESIS

Health Monitoring for Aircraft Systems using Decision Trees and Genetic Evolution

Mike Gerdes

Operation and Maintenance Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Health Monitoring for Aircraft Systems using Decision Trees and Genetic Evolution

Mike Gerdes

Principal supervisor:
Prof. Diego Galar (Luleå University of Technology)

Co-supervisors:
Prof. Uday Kumar (Luleå University of Technology)
Prof. Dieter Scholz (HAW Hamburg)

LULEÅ TEKNISKA UNIVERSITET

Division of Operation and Maintenance Engineering
Luleå University of Technology
Luleå, Sweden

DOI:
https://doi.org/10.15488/9213

URN:
https://nbn-resolving.org/urn:nbn:se:ltu:diva-76703
https://nbn-resolving.org/urn:nbn:de:gbv:18302-aero2019-12-20.012

Printed by Luleå University of Technology, Graphic Production 2019

ISSN 1402-1544
ISBN 978-97-7790-500-4 (print)
ISBN 978-91-7790-501-1 (pdf)

Luleå 2019

www.ltu.se

TABLE OF CONTENTS

Abstract 10

Acknowledgments 11

Appended Papers 12

List of Figures 15

1 Introduction 21

1.1 Preventive Aircraft Health Monitoring for Integrated Reconfiguration (PAHMIR) 21

1.2 Problem Description 21

1.3 Purpose and Objectives 23

1.4 Research Questions 23

1.5 Research Methodology 24

1.5.1 Research Strategy 24

1.5.2 Validation and Verification 25

1.6 Scope and Limitations 25

1.7 Authorship of Appended Papers 26

1.8 Outline of Thesis 27

1.9 Concept 28

1.9.1 Condition Monitoring 28

1.9.2 Condition Prediction 29

1.9.3 Remaining Useful Life Prediction 30

2 Background 31

2.1 Aircraft Maintenance 31

2.1.1 Failure Classes 34

2.1.2 Failure Probability 35

2.1.3 Unscheduled Maintenance 35
2.1.4 Maintenance Program Development 36
2.1.5 Preventive Maintenance 37
2.1.6 Condition-Based Maintenance 38
2.2 Condition Monitoring 40
2.2.1 Diagnosis 42
2.2.2 Prognosis 44
2.3 Signal Analysis 44
2.4 Feature Extraction 47
2.4.1 Time Domain Features 48
2.4.2 Frequency and Time-Frequency Domain Features 49
2.5 Data Fusion 50
2.6 Decision Trees 51
2.7 Local Search and Optimization 54
2.8 Trend Series Analysis and Prediction 56
2.8.1 Simple Linear Regression 59
2.8.2 Multiple Regression 60
2.8.3 Simple Moving Average Model 62
2.8.4 Exponential Smoothing 62
2.8.5 Box-Jenkins 63
2.8.6 Other Methods 65
3 Proposed Concept 67
3.1 Training Process 68
3.1.1 System Data 68
3.1.2 Classification Training 71
3.1.3 Prediction Training 77
3.2 Monitoring and Interactive Prediction Process 83

3.2.1 System Data 83
3.2.2 Condition Classification 83
3.2.3 Iterative Prediction 84
3.3 Summary 85
4 Validation 86
4.1 Condition Monitoring: Test Rig Validation 86
4.1.1 Test Rig 86
4.1.2 Validation 89
4.2 Condition Prediction: Validation with Generated Data Validation 92
4.2.1 Setup 92
4.2.2 Results 94
4.3 Condition Monitoring and Prediction: In-Service Aircraft Validation 96
4.3.1 Data 96
4.3.2 Data Analysis 102
4.3.3 Extrapolative Prediction Method 105
4.3.4 Extrapolative Prediction Results 110
4.4 Summary 111
5 Conclusions 113
5.1 Research Questions 114
5.2 Further Research 115
6 Paper 1: Effects of Condition-Based Maintenance on Costs Caused by Unscheduled Maintenance of Aircraft 116
6.1 Introduction 116
6.1.1 Aircraft Maintenance 116
6.1.2 Scheduled Maintenance 119
6.1.3 Maintenance Program Development 119
6.1.4 Unscheduled maintenance 121

6.1.5 Preventive Maintenance 121
6.1.6 Condition-Based Maintenance 122
6.2 Maintenance Costs 123
6.2.1 Estimation of Maintenance Global Cost 125
6.2.2 Downtime Cost and Failure Cost 126
6.2.3 Evaluation of the failure cost 127
6.3 Delays in Aviation 128
6.3.1 Delay Causes 128
6.3.2 Costs of Delays 129
6.4 Unscheduled Maintenance Causes 130
6.5 Aircraft System and Database 131
6.6 Empirical Study: AC System of A340 131
6.6.1 Delay Analysis 131
6.6.2 Integrating Condition-Based Maintenance into Preventive Maintenance 133
6.6.3 Strategy of Overinvestment 134
6.6.4 Reduction of Delays and Costs using CBM 135
6.6.5 Influence of CBM on Aircraft Costs 140
6.7 Discussion and Conclusion 140
7 Paper 2: Decision Trees and the Effects of Feature Extraction Parameters for Robust Sensor Network Design 142
7.1 Introduction 142
7.1.1 Civil Aerospace Software Development 144
7.1.2 Feature Extraction 145
7.1.3 Decision Trees 146
7.1.4 Basic Condition Monitoring Process Enhancements 149
7.1.5 Sensor Optimization 149
7.1.6 Multi-Sensor Data Fusion 150

7.2 Proposed Methodology 151
7.2.1 Feature Extraction and Sensor Fusion 152
7.2.2 Decision Tree Generation 152
7.2.3 Sensor Optimization 154
7.3 Validation 155
7.3.1 Feature Extraction Parameter Influence 155
7.3.2 Sensor Optimization 158
7.4 Result Analysis 159
7.4.1 Parameter Evaluation 159
7.4.2 Sensor Optimization 164
7.5 Conclusions and Discussion 165
8 Paper 3: Automated Parameter Optimization for Feature Extraction for Condition Monitoring 167
8.1 Introduction 167
8.2 Background 168
8.2.1 Condition Monitoring 168
8.2.2 Feature Extraction 170
8.2.3 Pattern Recognition 173
8.2.4 Optimization 176
8.3 Proposed Method 177
8.3.1 Training Process 179
8.3.2 Optimization Loop 181
8.4 Validation 182
8.4.1 Experiment 1: Sample Data 185
8.4.2 Experiment 2: Random Seeds 186
8.4.3 Experiment 3: Optimization Algorithm 186
8.4.4 Experiment 4: Pattern Recognition 186

8.5 Results and Discussion 186
8.5.1 Experiment 1: Number of Samples 187
8.5.2 Experiment 2: Different Seeds 187
8.5.3 Experiment 3: Optimization Algorithm 188
8.5.4 Experiment 4: Pattern Recognition 189
8.6 Conclusions 190
9 Paper 4: Fuzzy Condition Monitoring of Recirculation Fans and Filters 191
9.1 Introduction 191
9.2 Decision Trees 193
9.2.1 Fuzzy Decision Trees 196
9.2.2 Concept 198
9.2.3 Fuzzy Decision Tree Inference Example 201
9.2.4 Fuzzy Decision Tree Forest Inference 203
9.3 Validation 203
9.3.1 Setup 203
9.3.2 Results 207
9.4 Conclusion and Discussion 209
10 Paper 5: Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning 210
10.1 Introduction 210
10.1.1 Time Series 210
10.1.2 Decision Trees 211
10.1.3 Genetic Algorithms 212
10.2 Method 212
10.2.1 Training Process 213
10.2.2 Forecasting Process 216
10.2.3 Method Summary 217

10.2.4 Process Modifications 217
10.3 Experiments 218
10.3.1 Forecasting Without Noise 219
10.3.2 Forecasting with Noisy Test Samples 220
10.3.3 Forecasting Noisy Training and Test Samples 220
10.4 Conclusion 221
11 Paper 6: Genetic Algorithms and Decision Trees for Condition Monitoring and Prognosis of A320 Aircraft Air Conditioning 223
11.1 Introduction 223
11.1.1 Condition Monitoring 224
11.1.2 Feature Extraction 226
11.2 Method 228
11.2.1 Training Process 229
11.2.2 Classification and Prediction 233
11.3 Validation 237
11.4 CONCLUSION 244
12 References 247

ABSTRACT

Reducing unscheduled maintenance is important for aircraft operators. There are significant costs if flights must be delayed or cancelled, for example, if spares are not available and have to be shipped across the world. This thesis describes three methods of aircraft health condition monitoring and prediction; one for system monitoring, one for forecasting and one combining the two other methods for a complete monitoring and prediction process. Together, the three methods allow organizations to forecast possible failures. The first two use decision trees for decision-making and genetic optimization to improve the performance of the decision trees and to reduce the need for human interaction. Decision trees have several advantages: the generated code is quickly and easily processed, it can be altered by human experts without much work, it is readable by humans, and it requires few resources for learning and evaluation. The readability and the ability to modify the results are especially important; special knowledge can be gained and errors produced by the automated code generation can be removed.

A large number of data sets is needed for meaningful predictions. This thesis uses two data sources: first, data from existing aircraft sensors, and second, sound and vibration data from additionally installed sensors. It draws on methods from the field of big data and machine learning to analyse and prepare the data sets for the prediction process.

Keywords: Condition Monitoring, Remaining Useful Life Prediction, Decision Tree, Genetic Algorithm, Fuzzy Decision Tree Evaluation, System Monitoring, Aircraft Health Monitoring, Feature Extraction, Feature Selection, Data Driven, Health Prognostic, Knowledge Based System, Supervised Learning, Data-Driven Predictive Health Monitoring, Health Indicators, Machine Learning, Big Data, Pattern Recognition

ACKNOWLEDGMENTS

I thank all those who made this thesis possible. Professor Scholz and Bernhard Randerath set up the PAHMIR project (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) and supported me during my work. I also thank Professor Petter Krus for making it possible to be a PhD student at Linköpings Univesitet and for supporting me on my way to my Licentiate. I thank Professor Diego Galar for helping me to move from my Licentiate to the PhD and giving me many helpful tips. Elisabeth Thompson did a really great job of checking this thesis and fixing my errors. Finally, I would like to thank Philotech GmbH for making it possible for me to leave my job for a few years and return after my work at HAW Hamburg.

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

APPENDED PAPERS

The first published paper represented research on and an analysis of the benefits of predictive health monitoring by looking at the costs of unscheduled maintenance. The second paper developed a basic method for health monitoring using decision trees, sound/vibration signal and feature extraction. One drawback of the proposed method was that the feature extraction had a significant influence on the performance of the condition monitoring. So the next step (paper 3) was to improve the feature extraction by automatically selecting an optimal set of parameters using a genetic algorithm. Another drawback was that the condition monitoring returned only discrete values for the current condition and no continuous values or information about neighbouring states. Accordingly, the method was modified to return more information by using a post-fuzzification approach (paper 4). In the next step, the method was enhanced to allow the forecasting of the system health. To do this, time series data using the fuzzy condition results were created, then approximated and extrapolated into the future. Artificial test data were used for this step (paper 5). Finally, the method was modified and tested with real-world aircraft data to validate it (paper 6).

Paper 1: Effects of Condition-Based Maintenance on Costs Caused by Unscheduled Maintenance of Aircraft (Gerdes, et al., 2016)

This paper analyses the effects of condition-based maintenance based on unscheduled maintenance delays caused by ATA chapter 21 (air conditioning). The goal is to show the introduction of condition monitoring in aircraft systems. The research used the Airbus In-Service database to analyse the delay causes and lengths and to check whether they were easily detectable via condition monitoring. These results were combined with delay costs. Analysis showed that about 80% of the maintenance actions causing departure delays can be prevented when additional sensors are introduced. With already existing sensors, it is possible to avoid about 20% of the delay-causing maintenance actions.

Paper 2: Decision trees and the effects of feature extraction parameters for robust sensor network design (Gerdes, et al., 2017)

Reliable sensors and information are required for reliable condition monitoring. Complex systems are commonly monitored by many sensors for both health assessment and operation purposes. When one of the sensors fails, the current state of the system cannot be calculated reliably as the information about the current state will not be complete. This paper shows how to calculate the

significance of the information that a sensor gives about a system by using signal processing and decision trees. It also shows how signal processing parameters influence the classification rate of a decision tree and, thus, the information. The paper uses decision trees to calculate and order the features based on the information gain of each feature. During the method validation, they are used for failure classification to show the influence of different features on the classification performance.

The paper concludes by analysing the results of experiments; it finds the method can classify errors with a 75% probability and different feature extraction options influence the information gain.

Paper 3: Automated Parameter Optimization for Feature Extraction for Condition Monitoring (Gerdes, et al., 2016)

Pattern recognition and signal analysis can be used to support and simplify the monitoring of complex aircraft systems, but information must be extracted from the gathered data in a proper way. The parameters of the signal analysis need to be chosen specifically for the monitored system to get the best pattern recognition accuracy. The paper develops an optimization process to find a good set of parameters for signal analysis using a global heuristic search and optimization. The computed parameters deliver slightly better results (one to three percent) than manual analysis. In addition, a full set of data samples is not needed. Thus, genetic optimization has the best performance.

Paper 4: Fuzzy Condition Monitoring of Recirculation Fans and Filters (Gerdes & Galar, 2016)

Pattern recognition technologies are often used to find patterns in complex systems. Condition monitoring can also benefit from pattern recognition. However, many pattern recognition technologies only output the classification of the data sample; they do not output any information about classes that are very like the input vector. This paper presents a concept for pattern recognition that outputs similarity values for decision trees. Experiments confirm that the method works and shows good classification results. Different fuzzy functions are evaluated to show how the method can be adapted to different problems. The concept can be used on top of any normal decision tree algorithm and is independent of the learning algorithm. The goal is to determine the probabilities of a sample belonging to each class. Experiments show the concept is reliable and works with decision tree forests (discussed in the paper) to increase the classification accuracy. Overall, the concept has the

same classification accuracy as a normal decision tree, but it offers the user more information about how certain the classification is.

Paper 5: Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning (Gerdes, 2013)

This paper proposes a method for forecasting the condition of an 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 then uses 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 noise disturbs the function.

Paper 6: Genetic Algorithms and Decision Trees for Condition Monitoring and Prognosis of A320 Aircraft Air Conditioning (Gerdes, et al., 2017)

The paper shows condition monitoring can be introduced into most systems by adopting a data-driven approach and using existing data sources. The goal is to forecast the remaining useful life (RUL) of a system based on various sensor inputs. Decision trees are used to learn the characteristics of a system. The data for the decision tree training and classification are processed by a generic parametric signal analysis. To obtain the best classification results for the decision tree, a genetic algorithm optimizes the parameters. A forest of three different decision trees with different signal analysis parameters is used as classifier. The proposed method is validated with data from an A320 aircraft from ETIHAD Airways. Validation shows condition monitoring can classify the sample data into ten predetermined categories, representing the remaining useful life (RUL) in 10 percent steps. This is used to predict the RUL. There are 350 false classifications out of 850 samples. Noise reduction reduces the outliers to nearly zero, making it possible to correctly predict condition. It is also possible to use the classification output to detect a maintenance action in the validation data.

LIST OF FIGURES

Figure 1: Unscheduled maintenance without failure forecasting 22

Figure 2: Unscheduled maintenance with failure forecasting 22

Figure 3: Maintenance (Williams, et al., 1994) 31

Figure 4: Machine/system condition over time (Kolerus & Wassermann, 2011) 32

Figure 5: MRBR process 36

Figure 6: One-to-one condition monitoring (Williams, et al., 1994) 41

Figure 7: One-to-many condition monitoring (Williams, et al., 1994) 42

Figure 8: System model (Williams, et al., 1994) 42

Figure 9: Fault detection with a system model (Williams, et al., 1994) 43

Figure 10: Laplace-based system model (Williams, et al., 1994) 43

Figure 11: Main model categories for prediction of remaining useful life (Sikorska, et al., 2011) 44

Figure 12: Signal sampling (Owen, 2007) 45

Figure 13: Time domain to frequency domain 46

Figure 14: Equal sized filter bank (Rabiner & Juang, 1993) 47

Figure 15: Variable sized filter bank (Rabiner & Juang, 1993) 47

Figure 16: Example of a decision tree (Mitchell, 1997) 51

Figure 17: Example of hill climbing (Russell & Norvig, 2003) 55

Figure 18: Example of genetic algorithm 56

Figure 19: Examples of time series data analysis (Montgomery, et al., 1990) 57

Figure 20: Linear model of time series data (Bowerman & O'Connell, 1993) 59

Figure 21: **2*nd*** order polynomial models (Montgomery, et al., 1990) 61

Figure 22: Differencing a time series two times (Bowerman & O'Connell, 1993) 65

Figure 23: Training process 67

Figure 24: Monitoring and iterative prediction process 68

Figure 25: Data sample 69

Figure 26: Signal preprocessing 72

Figure 27: Blocks and inverse Fourier transform 74

Figure 28: Time series sample generation 78

Figure 29: Dynamic window 79

Figure 30: Dynamic time series separation 80

Figure 31: Test rig 87

Figure 32: Open EVB 88

Figure 33: Classifications as a time series with one decision tree 90

Figure 34: Classifications as a time series with three decision trees 91

Figure 35: Genetic optimization performance 91

Figure 36: Test function for validation with generated data 92

Figure 37: Prediction results with no noise 94

Figure 38: Prediction results with noisy test samples 95

Figure 39: Prediction results with noisy data samples 95

Figure 40: A320 AeroBytes data description 1 101

Figure 41: A320 AeroBytes data description 2 102

Figure 42: 90% RUL similarities 105

Figure 43: Remaining useful life prediction 106

Figure 44: Alternative classification training process 106

Figure 45: Alternative classification monitoring and prediction process 107

Figure 46: Classification time series with noise/wrong classifications 108

Figure 47: Classification time series with applied noise reduction 108

Figure 48: Remaining useful life prediction 109

Figure 49: Start of different system health conditions 111

Figure 50: MRBR process 120

Figure 51: Effectiveness-cost relationship 124

Figure 52: Causes of departure delays in 2014 in Europe (Eurocontrol, 2015) 128

Figure 53: Sequence of events during unscheduled maintenance leading to delays (Sachon & Patè-Cornell 2000) 130

Figure 54: Delay length distribution (Airbus SAS 2008a) 132

Figure 55: Cumulative probability of delay (Airbus SAS 2008a) 132

Figure 56: Delay length distribution of non-preventable faults 136

Figure 57: Cumulative delay probability of non-preventable faults 136

Figure 58: Delay length distribution of preventable faults 137

Figure 59: Cumulative delay probability of preventable faults 137

Figure 60: Delay length distribution of realistically non-preventable faults 138

Figure 61: Cumulative delay probability of realistically non-preventable faults 138

Figure 62: Delay length distribution of realistically preventable faults 139

Figure 63: Cumulative delay probability of realistically preventable faults 139

Figure 64: Basic condition monitoring process (Jardine, et al., 2006) 143

Figure 65: Decision tree algorithm flow chart 148

Figure 66: Enhanced condition monitoring process 149

Figure 67: Feature selection process 152

Figure 68: Experiment process diagram 155

Figure 69: Data recording box 156

Figure 70: Data recording box architecture 156

Figure 71: Used PC fan 157

Figure 72: Validation process 159

Figure 73: Example of a decision tree 165

Figure 74: Condition monitoring process without optimization 170

Figure 75: One-against-all SVM (Pascual, 2015) 174

Figure 76: Multi-class SVM 175

Figure 77: Genetic algorithm example 177

Figure 78: Condition monitoring process with parameter optimization 178

Figure 79: Data recording box 184

Figure 80: Data recording box architecture 184

Figure 81: Airbus test rig for data recording 185

Figure 82: Common classification mapping of one input vector to one class 191

Figure 83: Classification mapping of one input vector to one class and output of similarity 192

Figure 84: Decision tree algorithm flow chart 194

Figure 85: Sample fuzzy member functions for speed....197

Figure 86: Nodeweight for "false" decision....200

Figure 87: Fuzzy decision tree inference example 202

Figure 88: Test 4ig at Airbus 0perations GmbH 204

Figure 89: Simple decision tree 212

Figure 90: Generation of trend learning samples with a sliding window 215

Figure 91: Plot of the experiment function 218

Figure 92: Plot with different starting points and forecast....219

Figure 93: Plot with different starting points, noisy training data and noise....220

Figure 94: Plot with different starting points, noisy training and testing data and noise 221

Figure 95: Classification training process....229

Figure 96: Classification and prediction process 233

Figure 97: Condition monitoring with multiple trees (Zaher & McArthur, 2007)....234

Figure 98: Classification time series with noise/wrong classifications 235

Figure 99: Classification time series with applied noise reduction 236

Figure 100: Remaining useful life prediction 237

Figure 101: Second most likely class 242

Figure 102: Start of different system health conditions....243

1 INTRODUCTION

This section gives an overview of the derivation of this work from the project on preventive aircraft monitoring for integrated reconfiguration.

1.1 Preventive Aircraft Health Monitoring for Integrated Reconfiguration (PAHMIR)

The PAHMIR (Preventive Aircraft Health Monitoring for Integrated Reconfiguration) project provided the research environment and basis for the dissertation. 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 lasted 3.5 years, from January 2008 to June 2011. Most of the research was done during this period. In 2016, real-world aircraft data were used to verify and refine the developed methods. The goals of PAHMIR were to analyse existing in-service aircraft maintenance data, develop a predictive aircraft health monitoring system and analyse how such a system might be integrated into a dynamic cabin concept. Concepts of condition monitoring, condition prediction and indoor localization were developed and tested.

The goals of PAHMIR were to:

- Reduce unscheduled maintenance
- Perform advanced failure prediction
- Perform condition monitoring
- Better plan maintenance
- Improve cabin reconfiguration

1.2 Problem Description

One goal of PAHMIR was to forecast and prevent failures. The main driver of the development of a failure prediction concept was the cost of delay of an aircraft departure or arrival. Unscheduled maintenance can cause delays. Failure prediction should 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 shows the handling of an aircraft fault without predictive health monitoring (i.e., failure prediction).

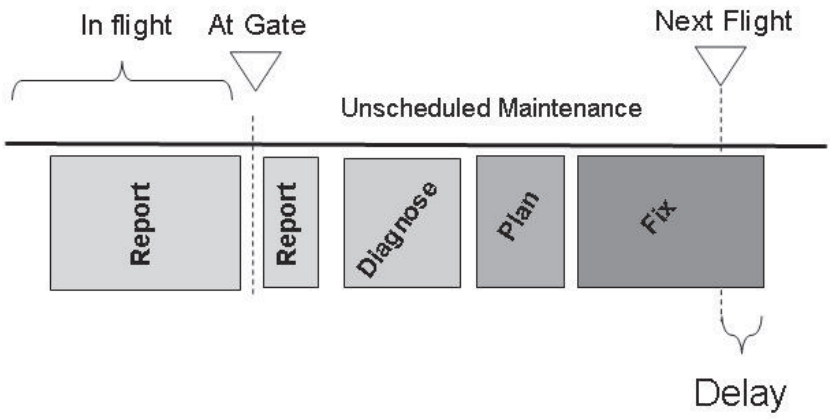

Figure 1: Unscheduled maintenance without failure forecasting

The maintenance case in Figure 1 is the following: 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 the 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. If the fault identification, diagnostics and spare parts management take too much time, the aircraft departure is delayed or even cancelled. A delay causes significant costs for an aircraft operator.

In the ideal case, most faults are repaired during scheduled maintenance (Figure 2). However, a fault still may occur. Delays of the type mentioned above can be prevented by repairing future faults in the hanger; this reduces the number of unscheduled maintenance cases.

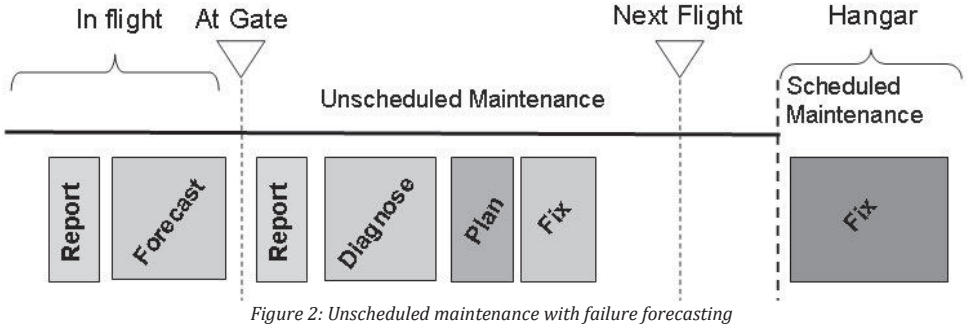

Figure 2: Unscheduled maintenance with failure forecasting

The costs of a delay can be quite large if the delay is long or the flight is cancelled. Gerdes, Scholz, Galar and Randerath (Gerdes, et al., 2009) (Gerdes, et al., 2016) have analysed the costs of a delay and determined what can be saved by forecasting faults and making repairs during scheduled maintenance.

1.3 Purpose and Objectives

Following the PAHMIR project, the goals of the dissertation can be formulated as:

- Create adaptable system condition monitoring
- Find simple and verifiable monitoring algorithms

Perform failure prediction 500 flight hours in advance

- Perform condition monitoring both online and offline
- Make condition monitoring and prediction usable in a changeable cabin layout
- Decrease the number of needed sensors
- Lower the hardware profile
- Use new and existing sensors
- Extend research beyond the aircraft domain
- Decrease the required human interaction

Briefly stated, the goals are to develop a condition monitoring and forecasting concept that is usable in the aircraft environment, that can be used for different systems and can be used by operators without much system knowledge or knowledge of the monitoring and prediction concept. The system should be easy to use in different aircraft systems. In addition, the algorithms should be easily verified and understood to ensure the system correctly monitors and forecasts the system condition.

The concept following from these goals is software that can be embedded in different environments. In this concept, most computation takes place during the configuration of the failure prediction system, not during operation. Most anticipated computations and methods are fast and need little hardware power or memory. The concept needs sensor input and a way to output the predictions.

1.4 Research Questions

The following research questions (RQs) are answered in this work:

RQ1: Is it possible to predict a failure of the air conditioning system 400 flight hours before a failure occurs so that it can be prevented during scheduled maintenance? This is the core research question that motivated this project and set the scope and limitations.

RQ2: Is it possible, with established signal processing and pattern recognition methods, to monitor and predict the health of aircraft fans and filters in real time? This second question was derived from the first question and expanded it. The goal of this question is to limit the available methods and to ensure that the developed method can be run on embedded and integrated devices.

RQ3: Is it possible to monitor the aircraft fans and filters by using existing sensors or by monitoring using only sound and vibration data? The installation and aircraft certification of new sensors and devices is an expensive and complex process. Therefore, it would be of considerable interest if the condition monitoring of an aircraft system could be done without installing new sensors. Existing sensors monitor systems indirectly by process data, error messages and interaction with other systems.

The six appended papers address the three RQs as shown in Table 1.

Table 1: Relationship between RQs and appended papers

RQ	Paper 1	Paper 2	Paper 3	Paper 4	Paper 5	Paper 6
RQ 1	X				X	X
RQ 2		X	X	X	X	X
RQ 3			X	X		X

1.5 Research Methodology

This section explains the research methodology and strategy. The first subsection focuses on the research approach; the second subsection explains the validation strategy. Application of the research methodology appears in later sections.

1.5.1 Research Strategy

The dissertation research used an iterative and experimental approach based on the iterative development and rapid prototyping of software development. This approach was selected because the result of the research was intended to be software using available tools to handle a problem in a new domain by combining those tools and using the strength of computer powered machine learning. Before the software development, however, a cost-benefit analysis verified the validity of the research goals.

The idea of iterative development and software prototyping is to create fast and functional software that does not necessarily contain all required features. Only the most important features

are implemented and tested. In later iterations, more required features are added to the prototype until the software is complete.

The advantage of this approach is that fundamental errors can be detected at an early stage and corrected to ensure the basic features are working before more advanced features are added. For this research, in the first iteration, only one part of the method was implemented and tested. Each iteration relates to a published paper (papers 2, 3, 4, 5). The first iteration contained only decision tree learning for feature selection/extraction. The second compared different learning and optimization methods to improve the feature selection/extraction. The third added fuzzy evaluation of the classification results. The fourth iteration added forecasting to the software.

Using this approach reduced the risk of research going in the wrong direction and time being wasted on methods that did not add to the research goal.

1.5.2 Validation and Verification

The initial plan was to use test rig and real-world data to validate and verify the prototypes, but because of regulations and time issues, it was not possible to get real-world data from an aircraft until the end of the project. This meant only test rig data were used to validate the method. This proved problematic, because the method had to be reworked once real-world data were available (Gerdes, et al., 2017).

Each prototype was validated and verified using different kinds of test data. The first prototype was tested with basic data that were not related to the problem domain. Instead, these data were specifically created to validate the features of the first prototype. The second and third prototypes were tested with data from a test rig that simulated parts of the problem domain, but not all parts. The fourth prototype was tested with the full test rig and data that were like those in the problem domain. For the final validation, real-world data were used to ensure the method performed well and yielded the desired results.

1.6 Scope and Limitations

The research explores how established simple pattern recognition and signal processing methods can be used to predict system health. The proposed approach uses established methods like Fast Fourier Transformation, decision trees and genetic evolution. The limitations of this research are the following:

- The method was developed with the aviation environment in mind. Thus, it was tested with aircraft data. The method has not been tested in other domains, so its applicability is unknown.
- Simple methods were chosen to allow real time processing of sensor data within a low power processor. Originally, it was planned to embed the data evaluation in electronic fasteners. These devices have little processing power and small memory. However, after the start of the project, the partnership company was sold, and the new owner ended the cooperation. But real time on-aircraft monitoring (with low power processors) is still needed to reduce the data that are transmitted to the ground.
- The method focused on decision trees as the main method for pattern recognition. There are better and more powerful algorithms available. Decision trees were selected because they can easily be modified by a human operator and are easy to understand and test/verify and deterministic. This is important for software in the civil aviation environment.
- The method was developed to complement the existing aircraft maintenance policies, not to replace them. Current aircraft maintenance policies state that a part needs to be exchanged before it breaks after a certain usage time. Thus, no monitored device will ever show observable failure. The planned replacement of a component was therefore considered as the end of life.

1.7 Authorship of Appended Papers

Table 2 summarises the contribution of each author to the appended papers. Contribution is divided into the following tasks:

1. Study conception and design
2. Data collection
3. Data analysis and interpretation
4. Manuscript drafting
5. Manuscript critical revision

Table 2: Contribution of each author to the appended papers

Author	Paper I	Paper II	Paper III	Paper IV	Paper V	Paper VI
Mike Gerdes	X	X	X	X	X	X
Diego Galar	X	X	X	X		X
Dieter Scholz	X	X	X			X

1.8 Outline of Thesis

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

Introduction
The introduction explains the problem, the motivation for the research, the research background, and the concept of the solution. It begins by describing the project which motivated the research. This is followed by an explanation of the motivation for and necessity of the research. The introduction also gives a full description of the research objectives. It closes with a review of the concepts applied to solve the problem and reach the objectives.

Background
The second section explains the theoretical background of the concepts used: aircraft maintenance, condition monitoring, signal analysis, feature extraction, data fusion, decision trees, heuristic search and time series analysis. The order of the topics is based on the order they appear in the text. The section closes with a summary.

Proposed Concept
This section discusses, in detail, the method used for condition monitoring and prediction. First, the training process is explained, then the monitoring and prediction process. The training process is divided into input data, classification training and prediction training; the monitoring process is divided into classification and prediction. The prediction is based on an interactive approach, where one data point after another is calculated. Validation attempts using real-world aircraft data, however, showed the method does not work very well with noisy data. Thus, a new prediction method was developed. The reworked method is shown in section 4.3.

Validation
The experiments described in this section show how feasible and usable the developed concepts really are. The section is divided into the evaluation of the condition monitoring concept and the evaluation of the condition prediction. The experimental setup was different for each, so evaluation was also different. Condition monitoring was validated on a test rig at Airbus; the prediction of condition was validated using computer generated data and real-world data from an in-service aircraft. The validation attempts with the real-world data showed the concept needs to be modified to accommodate these data.

Conclusion
The concluding section summarizes the results, answers the research questions and suggests future work and improvements.

1.9 Concept

The developed concept can predict failures so that maintenance can be planned. It is based on two processes (condition monitoring and condition forecasting) that work together to create a complete concept. Decision trees are used to make decisions in the concept and are at the core of both processes. In the first process (condition monitoring), the task is to decide which condition the sensor data represent, and in the second process (condition prediction), the task is to decide how to predict data points. Both tasks are solved by decision trees.

The core idea behind the concept is to use machine learning to create an expert system that adapts to different aircraft types through machine learning. A human expert is needed to configure the starting parameters and link sensor data to a system condition during the training. After the training, the system can work without a human expert. The 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 needing 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 human input 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 several 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 condition monitoring, because they can calculate the signal processing very quickly. With optimal hardware architecture and good implementation, it is possible to perform condition monitoring and condition prediction in real time. Signal processing and prediction parameter approximation parameter calculations take more time.

1.9.1 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 lifetime. The concept does not rely on a special type of sensor or 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 change multiple times per second. If the data are not numerical, preprocessing cannot be

applied, but it is still possible to use all other parts of the process, making 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 sensor data to other system conditions beyond the class they were mapped onto.

Condition monitoring is based on a decision tree which makes a decision based on signal features. The decision tree needs to be trained with data samples. Condition monitoring is a simple process compared to training. Preparation of training samples (signal feature extraction) is a complex task controlled by parameters. If it can be optimized, both performance and adaptability will be improved. To sum up, the following methods and technologies are used for the concept:

- Decision trees
- Signal analysis
- Optimization

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

1.9.2 Condition Prediction

Condition prediction requires time series data (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. Decision trees are used to decide what the best approximation method is for the current condition time series. Health conditions sometimes change quite quickly, so a prediction method needs to be able to detect indications for a fast change and change its prediction method based on the new information.

As in condition monitoring, training the decision tree is the most complex task of the process. The time series data need to be prepared and features need to be extracted. Again, an optimization process will improve performance. While the process looks more complicated than condition monitoring, it is easy to compute, and the steps are easy to understand. The following methods and technologies are used for the concept:

- Decision trees
- Time series analysis

- Optimization

1.9.3 Remaining Useful Life Prediction

Remaining useful life prediction combines the condition monitoring and the condition prediction processes into one complete process that can be used to forecast a failure. The condition monitoring gives the current system state (if correctly trained). However, there is no direct need to know the current system state for failure prediction. What the user needs to know is the similarity of the current state to a failed state. Fuzzy decision tree evaluation (Gerdes & Scholz, 2011) (Gerdes & Galar, 2016) can 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; this can be useful if the user wants to know how similar a sample is to a specific class. To predict RUL, the following methods and technologies are required:

- Condition monitoring
- Fuzzy decision tree evaluation
- Condition prediction

2 BACKGROUND

This chapter gives some background details of the methods, ideas and techniques used in this thesis. It starts with an overview of aircraft maintenance and condition monitoring to shed light on the idea and constraints of the proposed method. This is followed by a description of various aspects of the method and its development.

2.1 Aircraft Maintenance

Maintenance is the combination of all technical and associated administrative actions intended to retain an item in, or restore it to, a state in which it can perform its required function (Institution, 1993). The goal is to prevent fatal damage to machines, humans or the environment, to prevent unexpected machine failure, to allow condition based maintenance planning and to ensure safety of production and quality control (Kolerus & Wassermann, 2011). Figure 3 shows a breakdown of the various maintenance strategies.

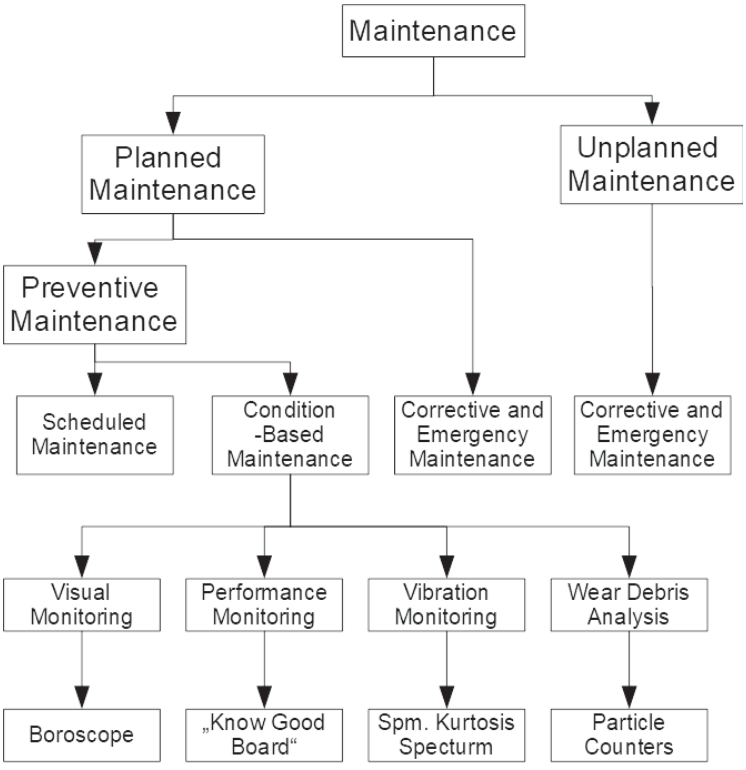

Figure 3: Maintenance (Williams, et al., 1994)

Basically there are three different maintenance strategies (Randall, 2011) (Kolerus & Wassermann, 2011):

- **Run-to-break** is the simplest maintenance: the machine or system is used until it breaks. This method is often used for systems that are cheap, especially when one failure does not cause other failures. It is commonly used for consumer products (Kolerus & Wassermann, 2011).
- **Preventive maintenance** is the most common maintenance method for industrial machines and systems. Maintenance is performed in fixed intervals. The intervals are often chosen so that only 1-2 percent of the machines will have a failure in that time (Randall, 2011).
- **Condition-based maintenance** is also called predictive maintenance. Maintenance is dynamically planned based on the machine or system condition. Condition-based maintenance has advantages over the other two strategies, but requires a reliable condition monitoring method (Randall, 2011).

Figure 4 shows a typical machine condition-based monitoring case. First, the machine goes into operation and is in normal operation. It is replaced shortly before a failure happens (Kolerus & Wassermann, 2011).

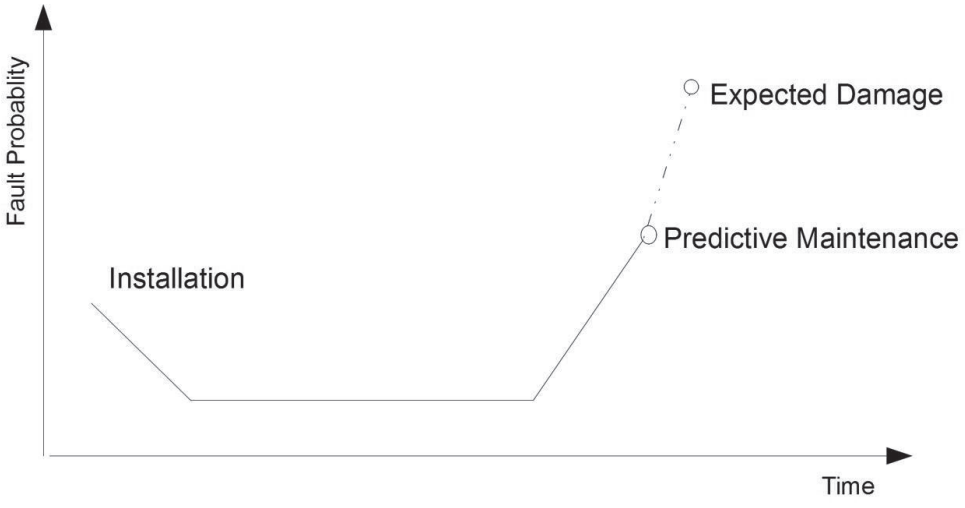

Figure 4: Machine/system condition over time (Kolerus & Wassermann, 2011)

Aircraft maintenance is based on reliability centred maintenance (RCM). The goal is to have maximum safety and reliability with minimized costs. Tasks are selected in a hierarchy of

difficulty and cost, from lowest to highest. Each task must also pass applicability and effectiveness criteria. Depending on the consequence of failure (safety, operational, economic, hidden safety and hidden non-safety) a single task or a combination of tasks is selected (Nowlan & Heap, 1978).

Reliability is the probability that an item will perform its intended function for a specified interval under stated conditions (US Department of Defense, 1998).

In the aircraft industry, the Maintenance Steering Group (MSG) has developed different maintenance concepts. The most recent is MSG-3 (Federal Aviation Administration, 2012). The focus of MSG-3 is the effect of a failure on aircraft operation (Nowlan & Heap, 1978) (Air Transport Association of America, 2007). For each item that affects airworthiness, a specific maintenance task is described (task oriented maintenance). MSG-3 can use condition-based maintenance or predetermined maintenance to achieve its goals. Most airlines and manufacturers use predetermined maintenance, as preventive maintenance with scheduled maintenance times provides both economic benefits and reliability (Kiyak, 2012).

The core concept of MSG-3 is Failure Mode and Effect Analysis (FMEA). With FMEA it is possible to determine which maintenance actions need to be performed during planned maintenance. This includes taking the probability and effects of a failure into account and planning the maintenance during system development. The FMEA uses a top-down approach, with analysis starting at the highest system level. A lot of detailed analysis is not needed, because most maintenance tasks are found at higher levels.

The FMEA process includes the following steps (Society of Automotive Engineers, 2001):

Identify Relevant Functions. In this step, all functions of a system are identified. See Table 3 for an example of a function.

Identify Functional Failures. The next step is to define the functional failure of a function. A function can have multiple failure modes. See Table 3 for an example.

Identify Failure Effects. The failure is classified by its effect using the process shown in Table 3

Identify Failure Probability. The probability of a failure is calculated based on experience or in-service data.

Select Maintenance Tasks. It is possible to define maintenance actions to prevent a failure, when the causes of a failure are defined. This step also includes determining the maintenance intervals, combining maintenance tasks and removing duplicate tasks.

Function	Functional Failure	Failure Mode
Provide redundant capability of informing crew of fire in each of four specific areas (right hand fan, left hand fan, core upper case, core lower case) in case of fire.	Loss of redundancy to detect fire in the designated engine fire zone.	Engine fire detector failure.
	Give false fire warning.	Engine fire detector failure.
Alerts crew of detection loop failure.	Does not alert crew of detection loop failure.	Engine fire detector failure. MAU Failure.

Table 3: Example of functional failure analysis - engine fire detection system (European Aviation Safety Agency, 2005)

2.1.1 Failure Classes

Failures are divided into five classes to determine the effect of a failure on the aircraft. A criterion for the classification is the severity of the failure for aircraft safety. Table 4 shows how failures are classified.

Table 4: Failure class criteria

<table>
<tr><td colspan="5">Is the occurrence of a functional failure evident to the operating crew during the performance of normal duties?</td></tr>
<tr><td colspan="3">Yes</td><td colspan="2">No</td></tr>
<tr><td colspan="3">Does the functional failure or secondary damage resulting from the functional failure have a direct adverse effect on operating safety?</td><td colspan="2">Does the combination of a hidden functional failure and one additional failure of a system related or backup function have an adverse effect on operating safety?</td></tr>
<tr><td>Yes</td><td colspan="2">No</td><td>Yes</td><td>No</td></tr>
<tr><td></td><td colspan="2">Does the functional failure have a direct adverse effect on operating capability?</td><td></td><td></td></tr>
<tr><td></td><td>Yes</td><td>No</td><td></td><td></td></tr>
<tr><td>Safety</td><td>Operational</td><td>Economic</td><td>Safety</td><td>Non Safety</td></tr>
<tr><td colspan="3">Evident</td><td colspan="2">Hidden</td></tr>
</table>

This results in the following failure classes (Air Transport Association of America, 2007):

Evident Safety. This must be approached with the understanding that a task is required to assure safe operation. If this is not the case, a redesign is required.

Evident Operational. A task is desirable if it reduces the risk of failure to an acceptable level.

Evident Economic. A task is desirable if the cost of the task is less than the cost of repair.

Hidden Safety. A task is required to assure availability and to avoid the adverse effect on safety of multiple failures. If this is not the case, a redesign is required.

Hidden Non-Safety. A task may be desirable to assure the availability necessary to avoid the economic effects of multiple failures.

2.1.2 Failure Probability

Ideally, in-service data are used to evaluate the risk of system failure based on the failure of different parts. However, during development, no in-service data are normally available. This means that during development, assumptions need to be made based on similar parts, tests, simulations or experience. Later, when in-service data are available, they can be used to update the failure probability.

Failure class and failure probability define the criticality of the failure. Criticality is used to plan the maintenance action.

2.1.3 Unscheduled Maintenance

Unscheduled maintenance is maintenance that needs to be done outside the defined periodic intervals because an unexpected failure occurs. The aircraft continues to fly safely because of 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 turnaround time, the flight will be delayed until the fault is eliminated. Depending on the failure, it is possible that the aircraft will need to stay on ground until the failure is fixed. The decision for the aircraft to stay on the ground (AoG - Aircraft on Ground) is based on the Minimum Equipment List (MEL) (International Civil Aviation Organization, 2015) (Civil Aviation Regulations Directorate, 2006). The MEL is based on the Master Minimum Equipment List (MMEL) (International Civil Aviation Organization, 2015) (Civil Aviation Regulations Directorate, 2006), a list accepted by national airworthiness authorities, but the MEL is an operator defined list that is stricter than the MMEL. If a faulty part is listed in the MEL, the aircraft is not allowed to operate until the failure is fixed.

Depending on the flight schedule of the aircraft, a departure delay may occur because of the maintenance operation. The flight may even have to be cancelled. Delays and cancellations are very expensive for an airline (Cook, et al., 2004) and should be avoided if possible.

2.1.4 Maintenance Program Development

Developing a plan for scheduled maintenance based on the MSG-3 method is complex. An Industry Steering Committee (ISC) consisting of authorities, aircraft operators and manufacturers is created. These actors form groups (MSG Working Groups (MWGs)) which meet and decide on the frequency and scope of needed maintenance actions (see Figure 5). First, MSG-3 analysis is performed based on aircraft data. Then, a Maintenance Review Board Report (MRBR) proposal is created and must be accepted. The MRBR contains the minimum scheduled tasking/interval requirements for a newly FAA type-certificated (TC) or derivative aircraft and its engines. The accepted MRBR is used by the manufacturer to create a Maintenance Planning Document (MPD) (Federal Aviation Administration, 2012) (Federal Aviation Administration, 1994) (European Aviation Safety Agency, 2008).

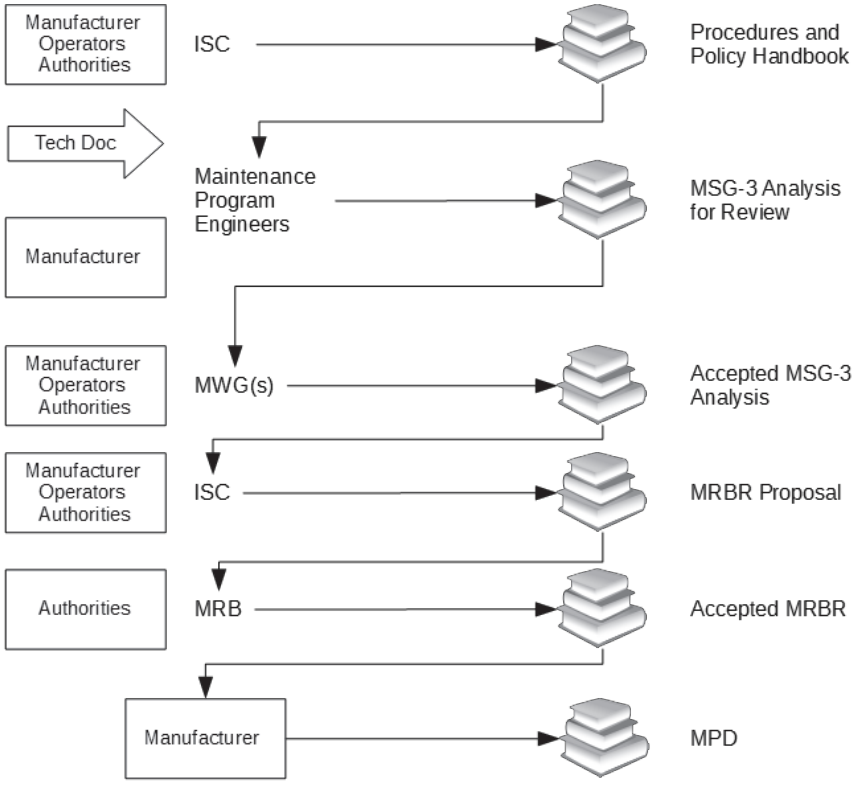

Figure 5: MRBR process

Revisions requiring formal approval are subject to the same consideration as initial approval (Federal Aviation Administration, 1978). Consequently, changing the maintenance plan is a

difficult process, and changing maintenance intervals requires a good reason. One problem is that there are no in-service data for a new aircraft, so maintenance intervals are created based on estimations. Operator in-service data are needed to adapt the maintenance intervals. Despite the difficulty, this is an important step; Ali and McLoughlin (2012) show the extent of costs savings when maintenance intervals are optimized.

2.1.5 Preventive Maintenance

Preventive maintenance (PM) is the standard method for reducing unscheduled maintenance. In the aircraft industry, aircraft components are inspected at 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 (Kolerus & Wassermann, 2011).

There are three types of preventive maintenance (Air Transport Association of America, 2007) (Nowlan & Heap, 1978) (Civil Aviation Authority, 1995) (Federal Aviation Administration, 1978):

Hard-Time (HT): Scheduled removal of a component before some specified maximum 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.

No Maintenance: This approach assumes a component can be used until it breaks and is then replaced. In MSG-2, this maintenance process is called “condition monitoring”. This maintenance process is not preventive but corrective (reactive); it is used for certain components in aircraft maintenance.

Periodic maintenance for aircraft is organised in five different classes of checks. Each check is performed at a different interval and becomes increasingly complex with the size of the interval. The intervals can vary depending on the aircraft type and aircraft operation (Air Transport Association of America, 2007).

2.1.5.1 Pre-/Post Flight Check

The most performed maintenance check is the pre-/post flight check. It is done daily. The pilot often does this check by walking around the aircraft and checking its general state.

2.1.5.2 A-Check

A-checks can be performed overnight in a hangar and are done every two months. An A-check examines all technical systems required for aircraft operation.

2.1.5.3 C-Check

The C-check is a major aircraft check, with the aircraft taken out of operation to be inspected. C-checks occur every two years and take about two weeks. The aircraft structure is inspected and all systems are tested.

2.1.5.4 IL-Check

The IL-check is done every four years and includes detailed checking and maintenance of systems and structures.

2.1.5.5 D-Check

This check is done every ten years and takes about one month. During this check, nearly the whole aircraft is dissembled and checked. Sometimes the paint is removed to check the structure. An aircraft has two to three D-checks during its lifetime.

2.1.6 Condition-Based Maintenance

Condition-based maintenance (CBM) is based on condition monitoring and aims at preforming maintenance based on the system condition and its trend. CBM can be used to realize RCM (Niu & Pecht, 2009).

Condition monitoring constantly measures and analyses relevant mechanical and electrical component parameters during operation. The parameters selected for monitoring allow determination of the condition and failure state. The need for maintenance of a component is only indicated if parameters show a predefined degradation (Kolerus & Wassermann, 2011).

The difference between CBM and preventive on-condition maintenance is that OC checks a system at defined intervals while condition monitoring continuously monitors the system.

Condition monitoring is used in a wide field of application, including rotary machines (gear boxes, gas and wind turbines, bearings etc. (Mahamad, et al., 2010) (Saravanan & Ramachandran, 2009) (Sugumaran & Ramachandran, 2011) (Tian & Zuo, 2010) (Zhao, et al., 2009), plants and structures (bridges, pipelines etc. (Goode, et al., 2000)). Vibration data are often used to perform the condition monitoring (Ebersbach & Peng, 2008).

The condition of the system is defined by setting limits on certain values based on experience (Mobley, 2002) or on a mathematical or data-driven model (Kolerus & Wassermann, 2011) (Williams, et al., 1994). Machine learning techniques, e.g., decision trees (Sugumaran & Ramachandran, 2007) (Sugumaran & Ramachandran, 2011) (Tran, et al., 2009), vector support machines (Pham, et al., 2012) (Sugumaran, et al., 2007) (Widodo & Yang, 2007) and neural networks (Chen, et al., 2012) (Mahamad, et al., 2010) (Tian, 2012), are often used to map the features of the input signal to a condition.

Another option is to use a mathematical model, feed the sensor input to the model, calculate the output and check how the output of the theoretical model deviates from the real system. This approach can also be used for fault isolation and identification of failures in addition to prognosis (Wang, et al., 2008) (Williams, et al., 1994) (Kolerus & Wassermann, 2011) (Jardine, et al., 2006).

Data-driven models use past data to create models with stochastic or machine learning algorithms (Pecht, 2008) (Garcia, et al., 2006) (Jardine, et al., 2006). These models require many data samples that represent different conditions of the system. Data-driven models require less human input than mathematical models; model validation and testing can be performed almost automatically.

Trend analysis is a method to achieve CBM. The analysis algorithm looks at recorded parameters at a single moment in time, but takes the full parameter history into account. The need for maintenance of a component is only indicated if the trend of a parameter shows degradation. Based on the parameter time history, the analysis algorithm can forecast the remaining lifetime of the component (Kolerus & Wassermann, 2011). A variety of methods are suitable for predicting future values. ARMA, ARIMA, artificial neural networks, sequential Monte Carlo and Markov models are used to predict values for a complex time series (Chen, et al., 2011) (Caesarendra, et al., 2010) (Pham & Yang, 2010) (Tian, et al., 2010). The output of the prediction is normally an estimated time to failure (ETTF) and a confidence interval (Sikorska, et al., 2011). The confidence interval defines the reliability of a prediction (Schruben, 1983) (Sikorska, et al., 2011) and can be calculated using a standard time series.

Implementing CBM is both difficult and costly. Many systems have barriers to its use. These barriers include (among others) (Stecki, et al., 2014):

- Inability to predict accurately and reliably the remaining useful life of a machine (*prognostics)*
- Inability to continually monitor a machine *(sensing)*
- Inability of maintenance systems to learn and identify impending failures and recommend

what action should be taken (*reasoning*).

- Initiation of CBM without full knowledge of how the system can fail
- Focusing of CBM research on specific techniques (better mousetrap syndrome)

2.2 Condition Monitoring

There are two strategies of monitoring (Randall, 2011) (Kolerus & Wassermann, 2011):

- **Permanent monitoring** is based on fixed, installed measurement systems. These systems often need to be very complex to react correctly if a failure occurs. They are used if a fast reaction is required after a failure. Permanent monitoring frequently shuts down a machine if a dangerous failure is detected (Randall, 2011).
- **Intermittent monitoring** is generally used for failure prediction and diagnosis. Measurements are taken on a regular basis with a mobile device. Data evaluation is done with an external device. Intermittent monitoring is often used to give long-term warnings (Kolerus & Wassermann, 2011).

Permanent monitoring is a better choice than intermittent monitoring when fast reaction times are required, but intermittent monitoring can do more complex computations (Randall, 2011). Permanent and intermittent monitoring can be combined using the same sensors and working in parallel. This allows intermittent monitoring to be carried out more often (hence, data are always available (Randall, 2011).

Methods of condition monitoring include the following (Randall, 2011):

- **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 analysed in the time domain and in the frequency domain. Vibration analysis is frequently used for condition monitoring (Randall, 2011) (Kolerus & Wassermann, 2011).
- **Lubricant/oil analysis** analyses the quality of the fluid and determines whether particles are in it. 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 of mechanical systems with a particle analysis (Williams, et al., 1994).

- **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 (Randall, 2011).
- **Thermography** is used to detect hot spots in a system or a machine. It is principally used in quasi-static situations.

Condition monitoring can be one-to-one or one-to-many (Williams, et al., 1994). In one-to-one monitoring, a system parameter measured by a sensor is directly forwarded for signal processing and condition monitoring (see Figure 6) independent of the sub-system to which the parameter belongs (Williams, et al., 1994).

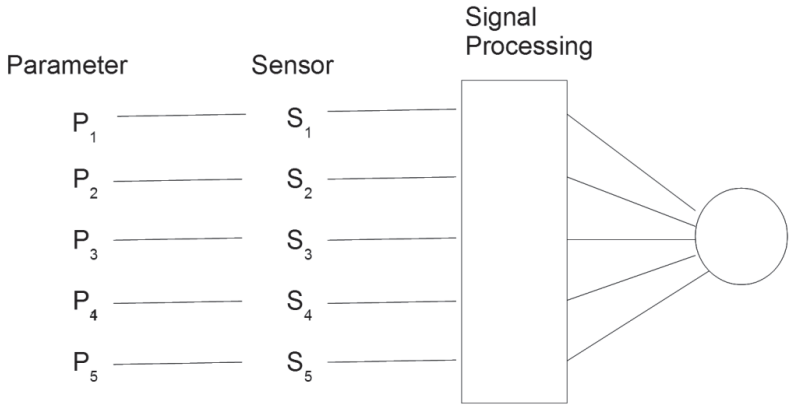

Figure 6: One-to-one condition monitoring (Williams, et al., 1994)

In one-to-many monitoring, one sensor is used to give condition information on more than one parameter (see Figure 7). This type of monitoring helps with failure location (Williams, et al., 1994).

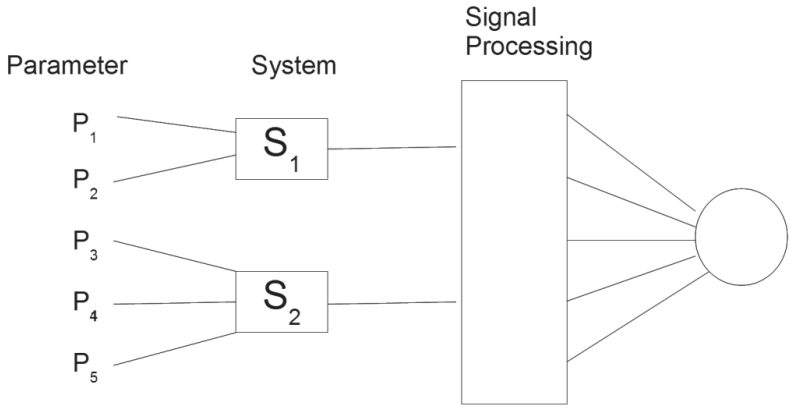

Figure 7: One-to-many condition monitoring (Williams, et al., 1994)

2.2.1 Diagnosis

There are several different methods for failure diagnosis using condition monitoring. If only one parameter is evaluated, trend analysis or limits can be used (Kolerus & Wassermann, 2011). Using a **limit** is the simplest method. The sensor signal is compared to a given limit; a failure has occurred if the signal is greater than the given limit. However, limit-based failure detection cannot be used to predict failure (Kolerus & Wassermann, 2011). **Trend analysis** records time series data of the sensor signal. It can be assumed that the machine is operating normally if only small changes occur over time. A stronger change in the time series indicates the development of a failure. Obviously, then, trend analysis can be used for failure prediction (Kolerus & Wassermann, 2011).

If a system is monitored, a system model needs to be created (see Figure 8). The model is used to compare the actual system outputs to the theoretical outputs. A difference indicates an error or an upcoming error (Figure 9) (Williams, et al., 1994). A system can be modelled by a mathematical description using Laplace-based system models or by dynamic (statistical) modelling (Williams, et al., 1994).

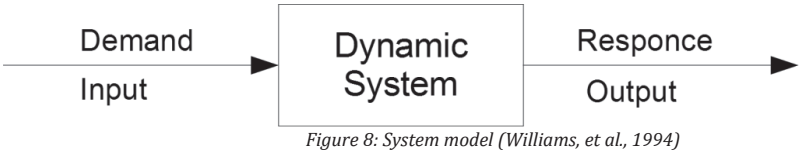

Figure 8: System model (Williams, et al., 1994)

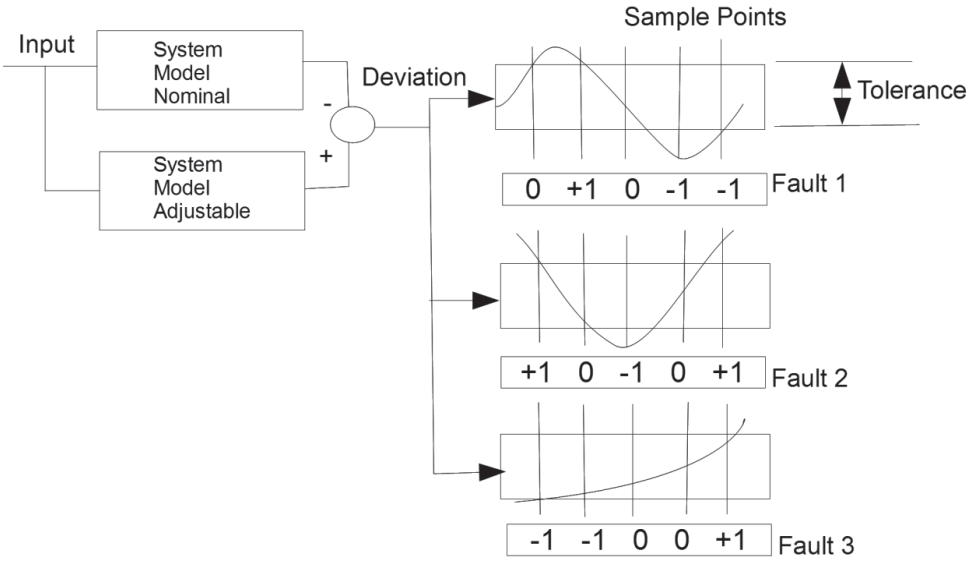

Figure 9: Fault detection with a system model (Williams, et al., 1994)

The **mathematical model** tries to describe the system in equations. It can become quite complex but a complete definition of the system is often not needed (Williams, et al., 1994). **Laplace-based system models** use the Laplace transformation to model a system with one or more building blocks (see Figure 10) (Williams, et al., 1994). System modelling and simulation tools like MATLAB Simulink, Modellica etc. use Laplace-like building blocks.

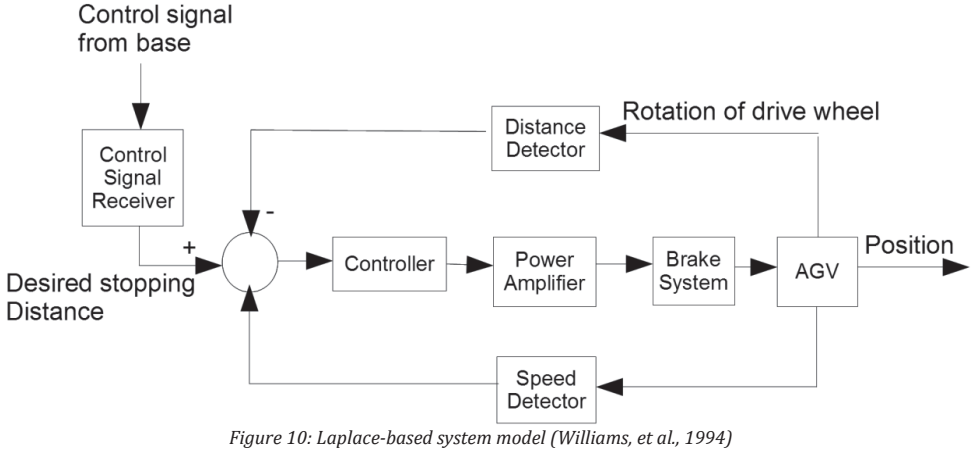

Figure 10: Laplace-based system model (Williams, et al., 1994)

Dynamic fingerprinting works without full knowledge of the system. The output for a given input is recorded, and the collection of the output makes the model (Williams, et al., 1994). **Outlier detection** uses various methods and techniques to detect an anomaly or a fault in sensor data. An outlier often indicates a system fault (Hodge & Austin, 2004).

Other methods for system modelling and condition monitoring include the use of neural networks and other machine learning techniques. Machine learning and pattern recognition are used for condition monitoring and trending in complex systems (Randall, 2011) (Kolerus & Wassermann, 2011). Brotherton et al. (Brotherton, et al., 1999) give an example of such an approach. Ataei et al. (Ataei, et al., 2005) use a neural network for sensor fusion (one-to-many), and Zaher and McAurthur (Zaher & McArthur, 2007) give an example of the one-to-many concept for distributed agents. Real-time monitoring with a neural network is shown in Smith et al. (Smith, et al., 2003).

2.2.2 Prognosis

Prognosis is the art of predicting the remaining useful life (RUL) of a system. Prognosis is related to diagnosis and depends on it. Diagnosis is the method of identifying and quantifying damage that has occurred (Sikorska, et al., 2011). Prognosis requires information from the diagnosis to forecast the future. However, in general, the degradation process cannot be directly observed or measured. It can only be investigated indirectly through the time series of features extracted from available process measurements (Yan, et al., 2010). This creates two major challenges for prognosis (Yan, et al., 2010):

1. How to design an appropriate degradation indicator (see Detection/Diagnosis)
2. How to establish a prediction model to estimate failure times

RUL Prediction methods can be classified according to the approach they use (Figure 11).

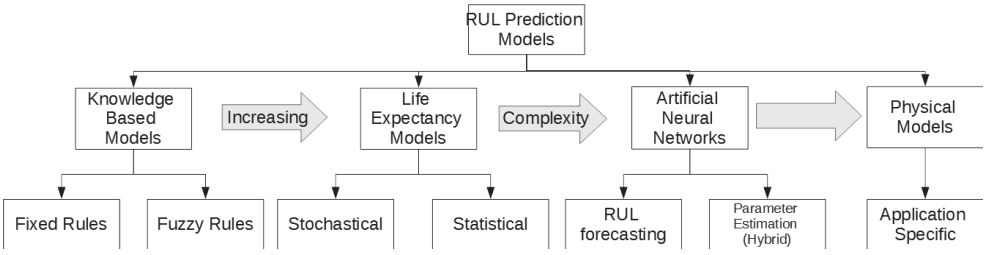

Figure 11: Main model categories for prediction of remaining useful life (Sikorska, et al., 2011)

2.3 Signal Analysis

A signal is an entity whose value can be measured and which conveys information (Owen, 2007). Signals can represent sound, vibrations, colour values, temperature etc. There are two types of signals: analogue and digital. An analogue signal is continuous, and a digital signal has a finite

number of values. The process of transforming an analogue signal into a digital signal is called sampling. Sampling represents an analogue signal as several regularly spaced measurements or samples (Owen, 2007). Figure 12 shows the sampling of an analogue signal.

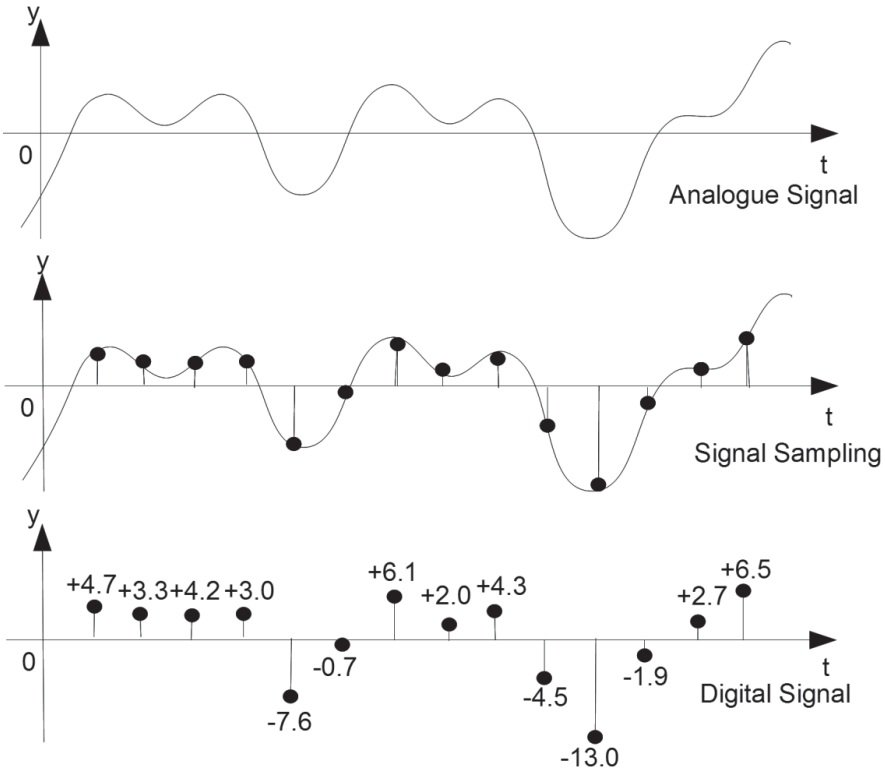

Figure 12: Signal sampling (Owen, 2007)

The number of regular spaced samples per second is the sampling rate measured in Hz. A signal has an amplitude and a phase. The amplitude is the sampling value, and the phase is the time delay between this motion and another motion of the same speed (Owen, 2007). Signals represented as in Figure 12 are in the time domain. It is possible to transform time signals so that they are represented in the frequency domain. In the frequency domain, the signal is represented by cosine and sine functions with different frequencies (Owen, 2007). The process converting the signal is called Fourier transform for analogue signals and discrete Fourier transform for digital signals. Equation (1 shows the discrete Fourier transform.

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

where *Z(f)* is the Fourier coefficient at frequency *f* (Owen, 2007), 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, respectively. It is possible to reverse the transform using Equation (2.

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

It is also possible to treat the complex values as real values if the phase is unknown or zero. In the case of a real value signal, only $\frac{1}{N}$ coefficients are independent because *Z(N-f)* and *Z(f)* are the same if only the real part is considered. In practice, this means *2N* samples are needed to get *N* Fourier coefficients. Figure 13 shows a real value signal transformed into the frequency domain.

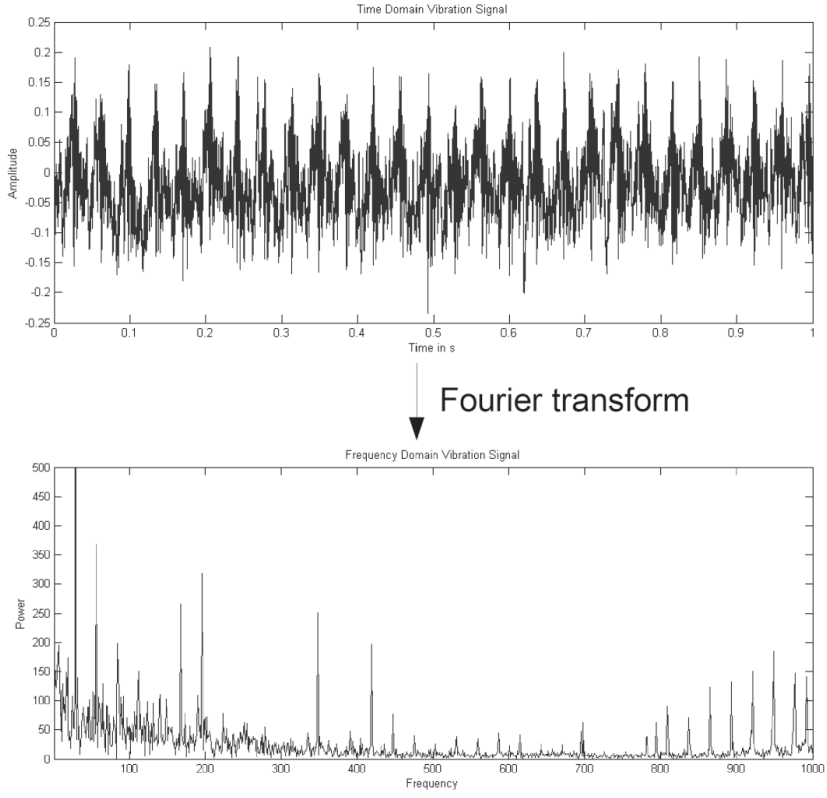

Figure 13: Time domain to frequency domain

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 (Owen, 2007).

A filter is a process which changes the shape of a signal (Owen, 2007), often in the frequency domain. Usual types of filters are low-pass, high-pass or band-pass filters. Low-pass filters keep low frequency components of the signal and block high frequency ones. A high-pass filter blocks low frequencies and keeps high ones. A band-pass filter blocks all but a given range of frequencies (Owen, 2007). One way to apply a filter is to transform the time domain signal into the frequency domain, apply the filter and 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, the filter is called a filter bank. The individual band-pass filters can have the same or different sizes (Rabiner & Juang, 1993). Figure 14 shows a filter bank with equal sized band-pass filters, and Figure 15 shows a filter bank with band-pass filters of a different size.

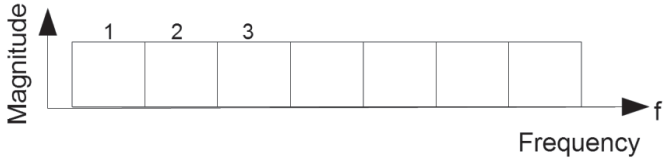

Figure 14: Equal sized filter bank (Rabiner & Juang, 1993)

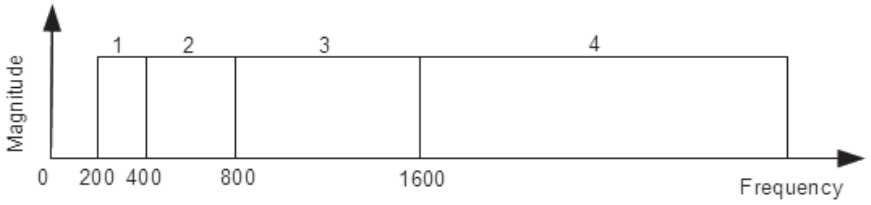

Figure 15: Variable sized filter bank (Rabiner & Juang, 1993)

2.4 Feature Extraction

Feature extraction is the process of reducing the dimension of the initial input data to a feature set of a lower dimension that contains most of the significant information of the original data (Fonollosa, et al., 2013). This is done to extract features from noisy sensor data (Lin & Qu, 2000); (Fu, 2011) and to avoid the problems caused by having too many input features (especially for vibration data) in the classifier learning phase (Yen & Lin, 2000). For these reasons, feature extraction is often a first and essential step for any classification (Yen & Lin, 2000).

Methods for feature extraction include extracting features from the time domain and the frequency domain (Fourier transformation, wavelet transformation (Fu, 2011)) and clustering them, if necessary. Basic features can be maximum, mean, minimum, peak, peak-to-peak interval

etc. (Jardine, et al., 2006). Complex feature extraction methods include principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) (Widodo & Yang, 2007). Other feature extraction methods are: t-test, correlation matrix, stepwise regression and factor analysis (FA) (Tsai, 2009). A comparison of feature extraction methods is found in Arauzo-Azofra et al. (Arauzo-Azofra, et al., 2011).

Clustering is needed if the data samples from which the features are extracted have no information about what the data represent (Li & Elbestawi, 1996). In such cases, clustering methods can be applied to group the data into classes.

Selecting relevant features for classifiers is important for a variety of reasons, including generalization of performance, computational efficiency and feature interpretability (Nguyen & De la Torre, 2010). Using all available features can result in over fitting and bad predictions, but it is not possible to look at each feature alone because many features are intercorrelated (Meiri & Zahavi, 2006). Noise, irrelevant features or redundant features complicate the selection of features even more. Thus, features are often selected using methods taken from pattern recognition or heuristic optimization or a combination. Sugumaran et al. (2007) show how different technologies can be combined for a single goal; they use a decision tree for feature selection and a proximal support vector machine for classification. Widodo and Yang (2007) combine ICA/PCA plus SVM for feature extraction and classification. A combination of particle swarm optimization (PSO) and SVM is used for feature extraction and process parameter optimization by Huang and Dun (2008). Many algorithms combine genetic algorithms (GAs) with a pattern recognition method like decision trees (DTs), SVM or artificial neural networks (ANNs). In these combinations, the GA is used to optimize the process parameters (Samanta, et al., 2003) (Huang & Wang, 2006) or to perform feature extraction and pattern recognition for classification (Samanta, 2004) (Saxena & Saad, 2007) (Jack & Nandi, 2002) (Samanta, 2004) Another popular approach is simulated annealing (SA) plus pattern recognition (Lin, et al., 2008) (Lin, et al., 2008).

2.4.1 Time Domain Features

Time domain features can be direct features like the number of peaks, zero-crossings, mean amplitude, maximum amplitude, minimum amplitude or peak-to-peak intervals (Jardine, et al., 2006) (Pascual, 2015). In addition, it is possible to analyse a signal using probabilistic methods, like root mean square, variance, skewness or kurtosis, to get features that represent the signal (Lambrou, et al., 1998). Mean square, variance, skewness and kurtosis are the four standardized moments of the probability distribution (see Table 5). Others are correlation, autocorrelation, entropy, PCA, ICA and KPCA (Widodo & Yang, 2007).

Moment	Degree	Definition
Mean	1	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}\right)}$
Variance	2	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}\right)^2}$
Skewness	3	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}\right)^3}$
Kurtosis	4	$\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}\right)^4}$

Table 5: Standardized probability distribution moments

2.4.2 Frequency and Time-Frequency Domain Features

The Fast Fourier Transform (FFT) transforms a signal from the time domain into the frequency domain. FFT takes a time series and transforms it into a complex vector that represents the frequency power in the frequency domain. The basis of the FFT algorithm is the discrete Fourier transform (DFT), defined as shown in Equation (3, where $x_n \ldots x_{n-1}$ are complex numbers.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k\frac{n}{N}} \quad k = 0, \ldots, N-1 \tag{3}$$

An FFT is performed in *O(N log N)* operations (Ohm & Lüke, 2010) and can be calculated in real time because it can be executed in parallel. It is a widely used and well established method (Peng, et al., 2002) (Fu, 2011). Recent research uses the discrete wavelet transform (DWT) to represent time series data in the frequency domain. The DWT represents the time series in a time-scale form (Jardine, et al., 2006) and is especially suited to represent non-stationary signals (Lin & Qu, 2000).

Existing failure diagnosis is mostly focused on the frequency domain, e.g. using Fourier transform or wavelet transform. In the early stages of failure development, damage is not significant, and a defect signal is masked by the noise in the acquired signal. The periodicity of the signal is not significant. Therefore, spectral analysis may not be effective. When the periodicity is significant, however, also using time domain features is recommended because normal and defect signals differ in their statistical characteristics in the time domain. Combining time domain features with those from other domains can improve the diagnosis accuracy.

2.5 Data Fusion

Having a network of different sensors monitoring a system leads to the need for sensor data fusion. Multi-sensor data fusion requires combining sensor data from different sources into one consistent model, but this can be difficult. The main problems of sensor fusion are (Basir & Yuan, 2007):

- How to get accurate and reliable information from multiple and possibly redundant sensors
- How to fuse multi-sensor data when data are imprecise and conflicting

Techniques for sensor fusion can be grouped into three levels (Jardine, et al., 2006), (Ross & Jain, 2003), (Castanedo, 2013):

- Data-level fusion, e.g., combining sensor data from the same sensors directly (Lu & Michaels, 2009)
- Feature-level fusion, e.g., combining vectors and feature reduction techniques (Ross & Jain, 2003)
- Decision-level fusion, e.g., using vote schemes (Ross & Jain, 2003)

Sensor data fusion is an important step in condition monitoring. Most systems have more than one sensor, and the sensors have different influences on condition monitoring accuracy. Condition monitoring data that require fusion come from sensors but they can also be event and process data, and these have important information for condition monitoring (Jardine, et al., 2006). Data-level fusion requires the direct combination of sensor data; the data from sensors of the same kind are merged and fed into the condition monitoring system. The difficulty is knowing how to merge multiple sensors into one. Sensor fusion at the feature level includes cleaning sensor data and combining the data after the features have been extracted and the dimensions reduced. Decision-level fusion can mean implementing condition monitoring for each sensor separately and using voting to decide on the system condition.

A condition monitoring system can use one or multiple data fusion methods to detect system condition. Sensor fusion depends on the target system and sensors. This makes it difficult to select a method. One solution is to implement sensor fusion on all levels and use a heuristic optimization like genetic algorithms, simulated annealing or hill climbing to get the best sensor fusion method for a given problem (data and system conditions).

2.6 Decision Trees

A decision tree is a tool of artificial intelligence. It classifies instances by sorting them from the root to a leaf node higher in the tree (Mitchell, 1997). Each node specifies a test on an attribute, and each branch from one node to another node or leaf corresponds to a test result (Mitchell, 1997). A sample decision tree is shown in Figure 16. This decision tree classifies the weather – is it suitable to play tennis or not?

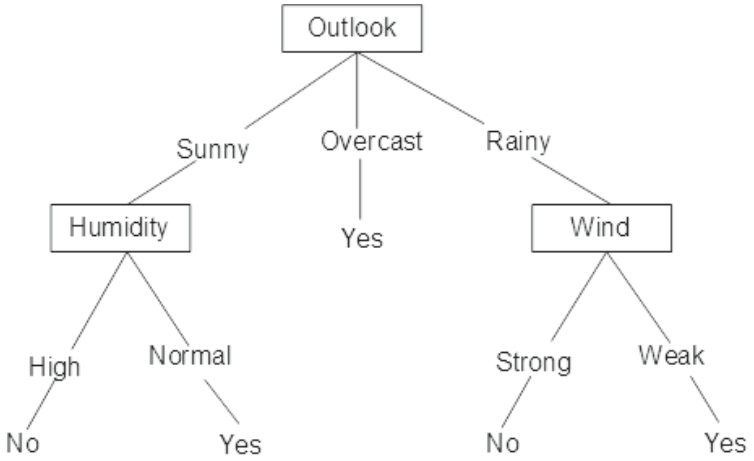

Figure 16: Example of a decision tree (Mitchell, 1997)

If the decision tree is used to learn a discrete value function (like the example), it performs a classification. If the tree is used to learn a continuous function, it performs a regression (Russell & Norvig, 2003). Any decision tree can be converted into a logical expression (Russell & Norvig, 2003). The example in Figure 16 can be expressed as:

$$Play = (Outlook = sunny \wedge Humidity = normal) \vee (Outlook = overcast) \vee (Outlook = rain \wedge Wind = weak) \quad (4)$$

Attribute value pairs represent an instance that might be tested. 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 several samples which specify the problem. The set of samples is called a training set. Several algorithms can be used to learn a decision tree. The basic decision tree learning algorithm works as follows (Russell & Norvig, 2003):

1. Create a new node.

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

There is an error in the training samples if two or more have the same attribute values but different classifications. In this case, it is possible to return the classification of most of the classifications or to report the probability of each classification (Russell & Norvig, 2003).

A common method for selecting the best attribute to split the samples is the ID3 (Mitchell, 1997). The idea of ID3 is to select a node based on the information gain. Information needs to be defined before we can define information gain and understand the concepts. Information entropy is the knowledge contained in an answer depending on prior knowledge. The less that 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 we have no data (Russell & Norvig, 2003). Information entropy is also called information and is calculated as shown in Equation (5:

$$I\left(P(v_i), \dots, P(v_n)\right) = \sum_{i=1}^{n} - P(v_i) log_2 P(v_i) \tag{5}$$

where $P(v_i)$ is the probability of the answer v_i.

The information gain from an attribute test (setting the value of a node in a tree; see Figure 16 for an example) is the difference between the total information entropy requirement (the amount of information entropy needed before the test) and the new information entropy requirement and is expressed in Equation 6:

$$Gain\ (X) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} \times I\ \left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right) \tag{6}$$

where p is the number of positive answers, and n is the number of negative answers (Russell & Norvig, 2003).

The performance of a decision tree can be tested with test samples from those training data not used for the learning. Performance of the decision tree depends on the number of correct classified samples.

A common problem for decision trees is over-fitting if there is noise in the training data or the number of training examples is too small (Mitchell, 1997). A model performs poorly with testing data if it is over-fitted. A simple method to remove over-fitting is decision tree pruning. Pruning means removing a sub-tree from the decision tree. It works by preventing recursive splitting on attributes that are not clearly relevant (Russell & Norvig, 2003). Another way to reduce over-fitting is cross-validation. In cross-validation, multiple decision trees are trained, each with a different set of training and testing samples. The decision tree with the best performance is chosen. A K-fold-cross-validation means k different decision trees are trained, and each is tested with a different set $\frac{1}{k}$ of samples (Russell & Norvig, 2003).

Decision trees can be extended to handle the following cases (Russell & Norvig, 2003):

- Missing data: not all attribute values are known for all samples.
- Multivalued attributes: the usefulness of an attribute might be low if an attribute has many different possible values (e.g., 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: at the leaves, the tree has a linear function rather than a single value (regression tree).

A second method for selecting the best attribute to split the samples is the C4.5 algorithm. It addresses some of the problems of the ID3 algorithm; for example, it accepts both continuous and discrete features and solves the over-fitting problem by pruning and handling incomplete data points. C4.5 uses the normalized information gain or the gain ratio. Split information (*Split Info*) is the information gained from choosing the attribute to split the samples. It is expressed in Equation 7 as:

$$Split\ Info\ (X) = -\sum_{i=1}^{n} \frac{p_i + n_i}{p + n}\ log_2\left(\frac{p_i + n_i}{p + n}\right) \tag{7}$$

The gain ratio is the normalized information gain and is defined as shown in Equation (*8* (Quinlan, 1993).

$$Gain\ Ratio\ (X) = \frac{Gain\ (X)}{Split\ Info\ (X)} \qquad (8)$$

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

Another class of decision trees is the fuzzy decision tree. As the name suggests, fuzzy decision trees are not based on crisp training data, but on fuzzy training data. Several researchers give examples of fuzzy decision tree training and suggest uses of fuzzy decision trees (Olaru & Wehenkel, 2003) (Sap & Khokhar, 2004) (Dong, et al., 2001).

2.7 Local Search and Optimization

Local search is a special area of search algorithms. In many cases, the search algorithm has 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. Therefore, they might check a member of the search space twice. Local search algorithms do not search systematically (Russell & Norvig, 2003). They include hill climbing search (greedy local search), simulated annealing and genetic algorithm.

Local search algorithms can also be used to solve pure optimization problems. They work in a state space landscape (Figure 17). Each state has a corresponding location, and the elevation of the state/location is the value of the heuristic cost function. The goal is to find the state/location with the lowest elevation (costs) (Russell & Norvig, 2003).

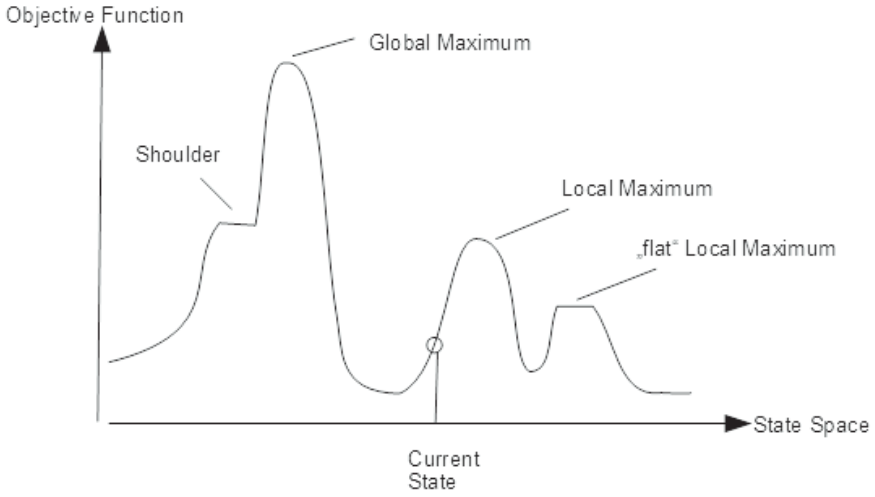

Figure 17: Example of hill climbing (Russell & Norvig, 2003)

The hill-climbing algorithm is a simple loop that moves in the direction of the increased value. Hill climbing evaluates the neighbour states and choses the best. For this reason, hill-climbing is sometimes called greedy local search. Hill-climbing can get stuck, because it makes no downhill moves and stays on a plateau or a local maximum (Russell & Norvig, 2003).

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 situation, it is accepted. If not, the move is accepted based on a probability value. The probability decreases exponentially with the move. It also decreases with each step (Russell & Norvig, 2003).

A genetic algorithm keeps one or more than one state in memory. The states in memory are called the population. During each step, new states (individual) are calculated based on the current population. The first population is generated randomly. New individuals are calculated through cross-over and mutation. In cross-over, two individuals are chosen from the population based on their fitness. Then two new individuals are created by taking part of one parent and part of the other parent. The first new individual has parts of both parents. The second one is constructed out of the not selected parts of both parents. Mutation modifies each individual based on an independent probability. Figure 18 shows an example of a genetic algorithm. The new individuals or children form a new population. Several authors (Russell & Norvig, 2003) (Mitchell, 1997). (Golub & Posavec, 1997) use genetic algorithms to adapt approximation functions from old problems to new problems; Stein et al. (Stein, et al., 2005) use genetic algorithms to select features for decision trees.

Figure 18: Example of genetic algorithm

2.8 Trend Series Analysis and Prediction

A time series is a chronological sequence of observations on a particular variable (Montgomery, et al., 1990). This means time series data are a number of data/time pairs ordered chronologically to show some time series. Time series analysis is done to discover historical patterns, which can be used for forecasting (Montgomery, et al., 1990). Predictions of future events and conditions are called forecasts, and the act of making such a prediction is called forecasting (Montgomery, et al., 1990). The goal of forecasting is to reduce the risk of decision making (Montgomery, et al., 1990).

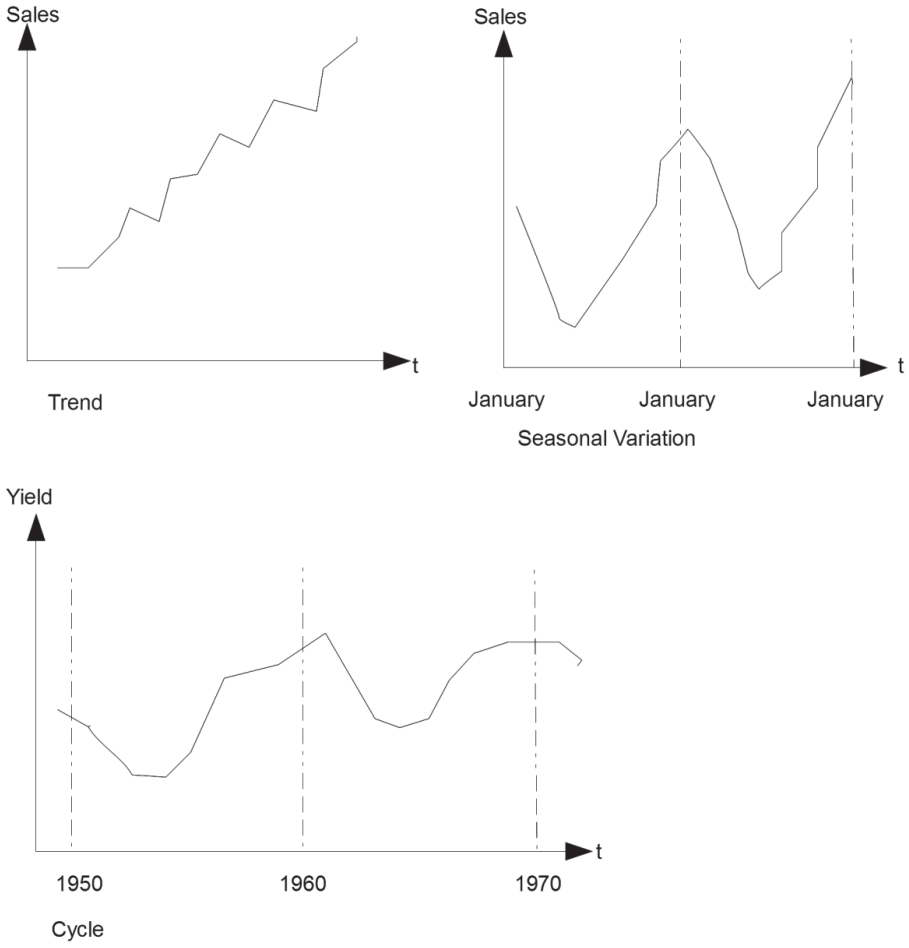

Figure 19: Examples of time series data analysis (Montgomery, et al., 1990)

Time series analysis and forecasting are used in many different areas, from economic forecasting and logistics management to strategic management (Montgomery, et al., 1990) (Granger & Newbold, 1977) (Bowerman & O'Connell, 1993). The following aspects are part of time series analysis (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993):

- Trend is the upward or downward movement of a time series over a period.
- Cycle refers to recurring up and down movements around trend levels.
- Seasonal variations are periodic patterns that complete themselves in a calendar year.
- Irregular fluctuations are movements that follow no pattern.

Time series data can be split into two categories: continuous and discrete. Continuous time series data are recorded at all times, while discrete time series data are recorded at given intervals

(hourly, daily etc.) (Granger & Newbold, 1977). Time series forecasting can be influenced by many factors, including the availability of data, cost of analysis or management preferences (Bowerman & O'Connell, 1993). The various elements of forecasting are defined by Bowerman and O'Connell as the following (Bowerman & O'Connell, 1993):

- Forecasting period is the basic unit of time for which forecasts are made (hours, days, weeks etc.).
- Forecasting horizon is the number of periods in the future covered by the forecast.
- Forecasting interval is the frequency with which forecasts are made.

The forecasting interval is frequently the same as the forecasting period, so the forecasting is revised after each period (Bowerman & O'Connell, 1993). There are two types of forecasts: expected value in the future and prediction interval (Bowerman & O'Connell, 1993) (Montgomery, et al., 1990). The prediction interval is an interval with a stated chance of containing the future value.

Forecasting can use qualitative or quantitative methods (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). Qualitative methods involve an expert while quantitative ones analyse historical observations to predict the future. The model of the historical data can be based on a single time series (uni-variant model) or it can include multiple variables (causal model) (Montgomery, et al., 1990) (Granger & Newbold, 1977). Bowerman and O'Connell (Bowerman & O'Connell, 1993) give examples of simple time series models.

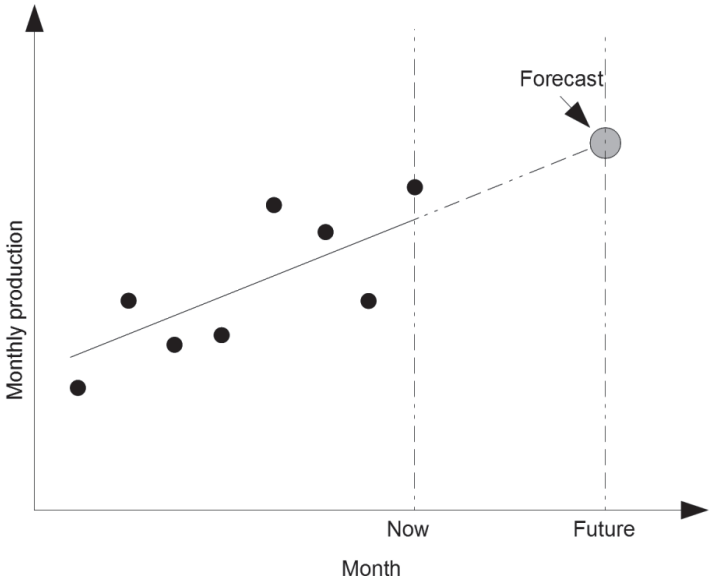

Figure 20: Linear model of time series data (Bowerman & O'Connell, 1993)

Several different methods can be used for quantitative forecasting (Montgomery, et al., 1990) (Granger & Newbold, 1977) (Bowerman & O'Connell, 1993):

- Simple linear regression
- Multiple regression
- Moving average model
- Exponential smoothing
- Box-Jenkins

Simple linear regression and multiple regression methods can be used to calculate a trend in time series data (Montgomery, et al., 1990). Each of the five methods is explained more fully in the following sub-sections.

2.8.1 Simple Linear Regression

The simplest regression method is simple linear regression. The goal of simple linear regression is to model the time series with a single straight line (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). The model has two parameters: the slope and the y-intercept. The model can be written as:

$$y = b_0 + b_1 x + \epsilon \tag{9}$$

A common method to estimate the two parameters b_0 and b_1 is to use least-squares (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). The method tries to find parameters for which the sum of the squared errors is the least. As shown in Equation 10, this means the sum of the squared errors between the line and the point y_i. The error sum can be written as:

$$l(b_0, b_1) = \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2 \tag{10}$$

The complete equation for the calculation of b_0 and b_1 is given in Equation 11 (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993):

$$b_1 = \frac{n \sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{11}$$

and $b_0 = \bar{y} - b_1 \bar{x}$

where $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$ and $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

The fitted simple linear regression model is:

$$\hat{y} = \hat{b}_0 + \hat{b}_1 z \tag{12}$$

2.8.2 Multiple Regression

Multiple regression is similar to simple linear regression but depends on more than one variable, as expressed in Equation (13 (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993).

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + \epsilon \tag{13}$$

The variables $x_1, x_2, \cdots, x_n$ can be different functions of time, like $x_1 = x^2$ (Bowerman & O'Connell, 1993). $x_1, x_2, \cdots, x_n$ may also be other time series information, like temperature or sales, both of which may influence the time series. Equation (14 shows an example.

$$y = b_0 + b_1 w(t) + b_2 s(t) \tag{14}$$

where *w(t)* is a function over time, like the weight of a human over time, and *s(t)* is a function over time, like salary. 2^{nd} order or higher order polynomial models can be used (Montgomery, et al., 1990). Figure 21 shows some 2^{nd} order functions.

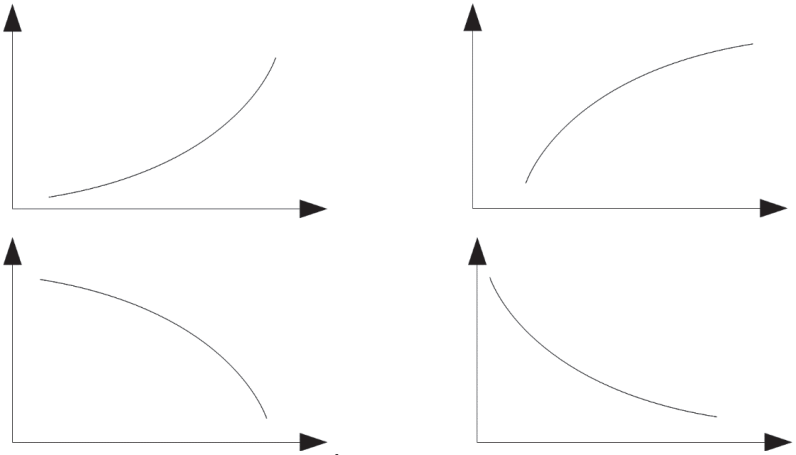

Figure 21: $\mathbf{2^{nd}}$ *order polynomial models (Montgomery, et al., 1990)*

The general representation of the p[th] order polynomial model is:

$$y = b_0 + b_1 x + b_2 x^2 + \cdots + b_p x^p + \epsilon \tag{15}$$

Multiple regression also uses the least-squares method to calculate the parameters $b_0, \cdots, b_n$. The least squares problem is often described in a matrix form as Equations 16 and 17 (Bowerman & O'Connell, 1993).

$$y = Z\hat{b} \tag{16}$$

$$\begin{pmatrix} 13 \\ 20 \\ 5 \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 12 & 4 \\ 19 & 34 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \end{pmatrix} \tag{17}$$

Normal equations can be used to solve the least-squares problem in a simple way (Equation (18) (Bowerman & O'Connell, 1993).

$$\hat{b} = (Z'Z)^{-1}Z'y \tag{18}$$

However, normal equations are not the most stable method to solve the problem. QR factorization solves the problem in a more stable way (Schwarz & Käckler, 2004).

2.8.3 Simple Moving Average Model

The simple moving average model is a simpler form of the simple linear regression model. The complete time series data are not evaluated, only *N* points of the time series (Bowerman & O'Connell, 1993). The simple moving average model is a way to reduce the noise in a time series. The simplest case, y_i, is the mean (arithmetic mean) of the last *N* values (Bowerman & O'Connell, 1993). The simple moving average model can be used to forecast a trend by using the following equation (Bowerman & O'Connell, 1993):

$$M_\tau = \frac{y_\tau + y_{\tau-1} + y_{\tau-2} + \cdots + y_{\tau-N+1}}{N} \tag{19}$$

Equation 20 calculates a forecast of τ periods into the future (Bowerman & O'Connell, 1993).

$$\hat{y}_{T+\tau} = 2M_T - M_T^{[2]} + \tau\left(\frac{2}{N-1}\right)\left(M_T - M_T^{[2]}\right) \tag{20}$$

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

$$M_T^{[2]} = \frac{M_T + M_{T-1} + \cdots + M_{T-N+1}}{N} \tag{21}$$

2.8.4 Exponential Smoothing

Exponential smoothing is a method for smoothing, much like the moving average model. The difference is that the data points are weighted unequally, with the most recent data point weighted more than past data points (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). The equation for simple exponential smoothing is:

$$S_T = \alpha x_T + (1-\alpha)S_{T-1} \tag{22}$$

where S_T is a weighted average of all past observations. To define an exponential smoothing that is an n-period, the moving average α is set as shown in Equation 23 (Bowerman & O'Connell, 1993):

$$\alpha = \frac{2}{N+1} \tag{23}$$

The starting value S_0 can be determined by taking the average of a certain number of past data points or by simply choosing a value (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). The forecast for the time period *T+1* is S_T (Bowerman & O'Connell, 1993). A low value of α causes the forecast to weight the last value more; this makes the forecast react faster to changes and to noise. A low value of α means the forecast will react more slowly.

2.8.5 Box-Jenkins

The Box-Jenkins methodology was developed by Box and Jenkins in 1976. The methodology consists of a four-step iterative procedure (Montgomery, et al., 1990):

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 include autoregressive models, moving average models, autoregressive-moving average models and autoregressive integrated moving average mode models. **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*) is added to the past data (Granger & Newbold, 1977). The autoregressive process is defined as (Bowerman & O'Connell, 1993):

$$x_t = \xi + \Phi_1 x_{t-1} + \Phi_2 x_{t-2} + \cdots + \Phi_p x_{t-p} + \epsilon_t \tag{24}$$

where $+\Phi_1 x_t$ is white noise, ξ is a constant and $\Phi_1, \cdots \cdots, \Phi_p$ are parameters (weights) of the model. The random shock ϵ_t describes the effect of all factors other than $x_{t-1}, \cdots, x_{t-p}$ on x_t (Montgomery, et al., 1990). An autoregressive process of the order p is called AR(p) (Bowerman & O'Connell, 1993), where p is the number of past data points. Autoregressive processes use the fact that the values of the time series data are correlated (Montgomery, et al., 1990). Related to the autoregressive processes are the **moving average processes**. The moving average process is defined as (Bowerman & O'Connell, 1993):

$$x_t = \mu + \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - \cdots - -\theta_q \epsilon_{t-q} \tag{25}$$

where μ is the mean of the time series, $\epsilon_t, \cdots, \epsilon_{t-q}$ are random shocks and $\theta_1, \cdots, \theta_q$ is 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 they have a mean of zero, a normal distribution and are defined by the variance. It is possible to combine the autoregressive process and the moving average process into an **autoregressive-moving average (ARMA)** model; the model has two orders, p,q or ARMA(p,q) (Bowerman & O'Connell, 1993). ARMA models can only represent stationary time series (Bowerman & O'Connell, 1993). A time series is stationary if the statistical properties (for example, mean and variance) of the time series are essentially constant through time (Montgomery, et al., 1990).

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 $y_1, \cdots, y_n$ are (Montgomery, et al., 1990):

$$z_t = y_t - y_{t-1} \tag{26}$$

where $t = 2, \cdots, n$

Taking only one difference is called the first difference. If the first difference has no stationary process, it is possible to redo the differences and get a second difference. Figure 22 shows a second difference (Bowerman & O'Connell, 1993).

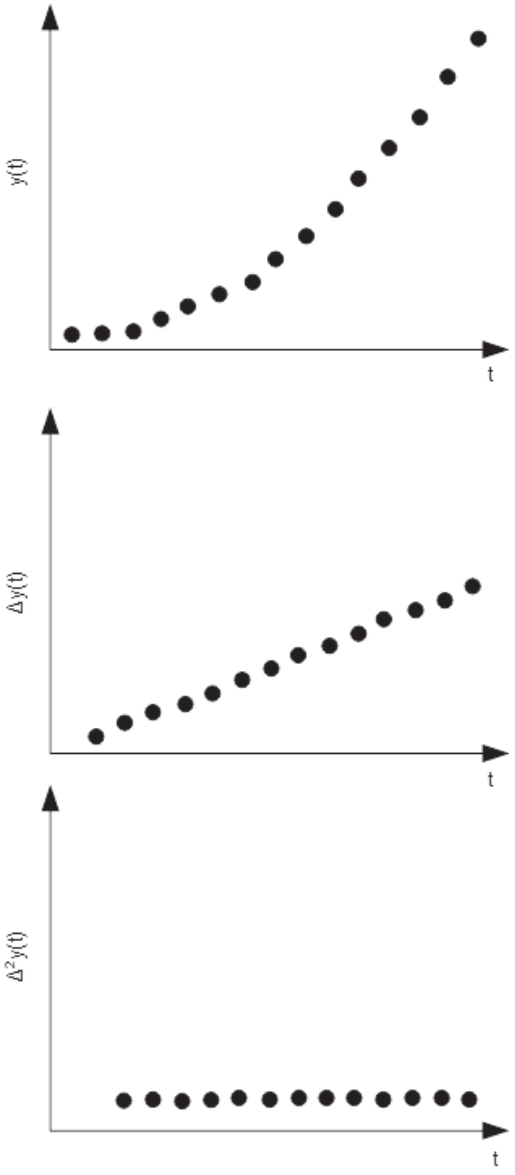

Figure 22: Differencing a time series two times (Bowerman & O'Connell, 1993)

Autoregressive integrated moving average (ARIMA) models use the d^{th} difference to model non-stationary time series data. An ARIMA model has an order of (p,d,q), where d is the d^{th} difference of the original series (Bowerman & O'Connell, 1993).

2.8.6 Other Methods

There are many different approaches to modelling and forecasting a time series. For example, Chaturvedi and Chandra (Chaturvedi & Chandra, 2004) use an artificial neural network to forecast

stock prices. Kontaki et al. (Kontaki, et al., 2005) use piecewise linear approximation to detect trend in a streaming time series. Others use Bayesian and other probability methods to model and forecast a time series (Bao & Yang, 2008) (Bowerman & O'Connell, 1993).

3 PROPOSED CONCEPT

The proposed concept for system health monitoring and prediction is presented in this section. The concept combines the previously introduced methods from Section 2 into a complete process that includes training, testing and operation. The process is separated into two parts. The first part is training and testing (Figure 23, Section 3.1) to adapt the method to the system to be monitored. Data samples from the system are needed for this step. The result is set of decision trees used to create a condition time series.

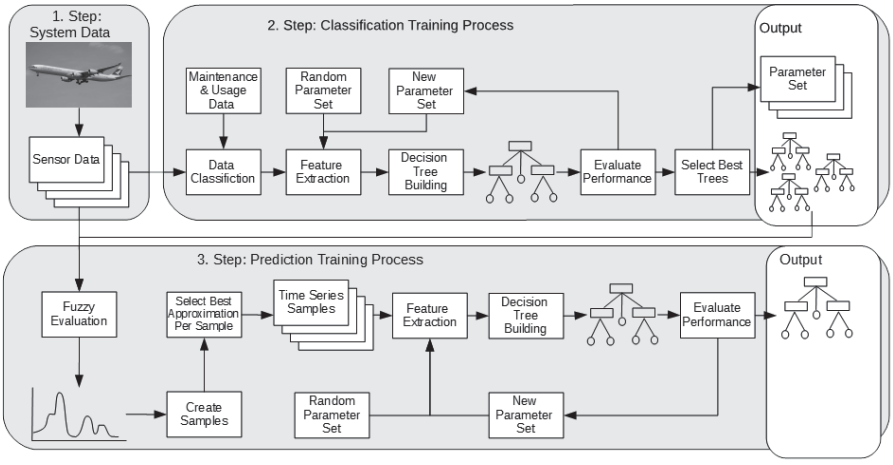

Figure 23: Training process

The second part is operation (Figure 24, Section 3.2), where new data samples are acquired, stored and evaluated by the system. The output is the current health condition (diagnosis) and a prediction of the future health condition (prognosis). This part creates a time series that represents the current system health and iteratively adds future data points based on a prediction.

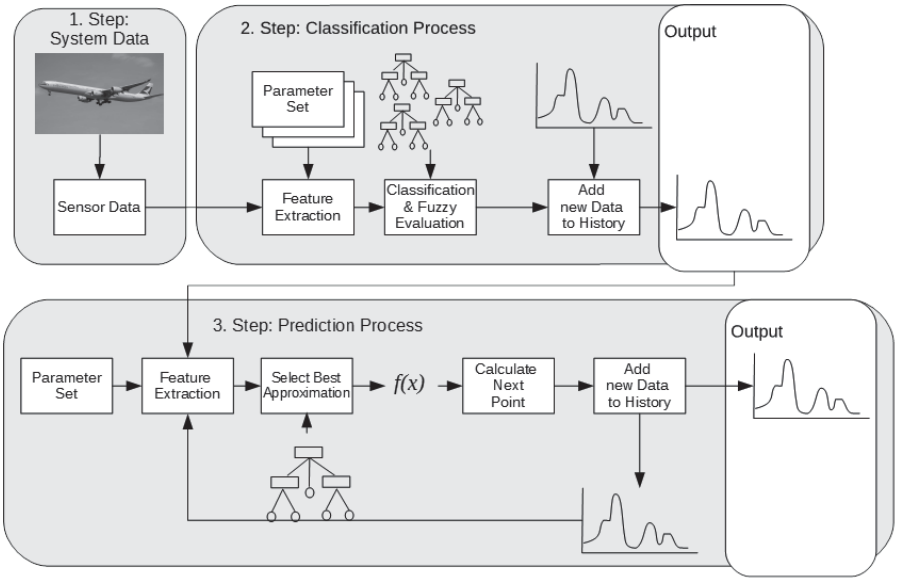

Figure 24: Monitoring and iterative prediction process

The method uses feature extraction and decision tree learning to detect patterns in sensor input and a time series. These patterns are then used to calculate the current health condition of the system and to select the best method for predicting future data points. The advantage of an iterative approach with feature extraction and switching of the extrapolation method is that the algorithm can react better to irregular results that are based on physical effects by switching the extrapolation method with the prediction based on past data.

3.1 Training Process

The training process has three steps. The first is the collection and sampling of training data. The second is the training and optimization of the current health condition. Output of this step is a system representation of a set of decision trees for calculating the current system's health based on a sensor data sample. The third step is the training of the prediction. The goal of the third step is to use the results of the first and second step to create a decision tree to predict the time series.

3.1.1 System Data

The condition monitoring concept uses models based on statistical data to classify new data. Therefore, many data samples are needed before the "real" training process can start. Multiple samples for each class of the system are needed. How much training data are needed is not easy to define. It depends on the complexity of the system (Gerdes & Scholz, 2011) (Gerdes & Galar, 2016). 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 can be recorded on all systems in parallel.

System data can come from both internal and external sensors. External sensors were used to validate the proposed method. The sensor signals were recorded for one second each minute. Different time intervals are possible, however, depending on the dynamics of the system. If the system changes quickly, a higher sample frequency is needed. If the system has slow dynamics, 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 systems to extract information; longer sampling periods allow the calculation of frequencies smaller than 1 Hz. In most 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, the preprocessing must 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 has more than one dimension, the preprocessing algorithm needs to be adapted or the data need to be transformed. The simplest solution is to have one preprocessing step for each dimension of the data and then concatenate the preprocessed data before giving them to pattern recognition. Each data sample needs to be classified/labelled by a supervisor.

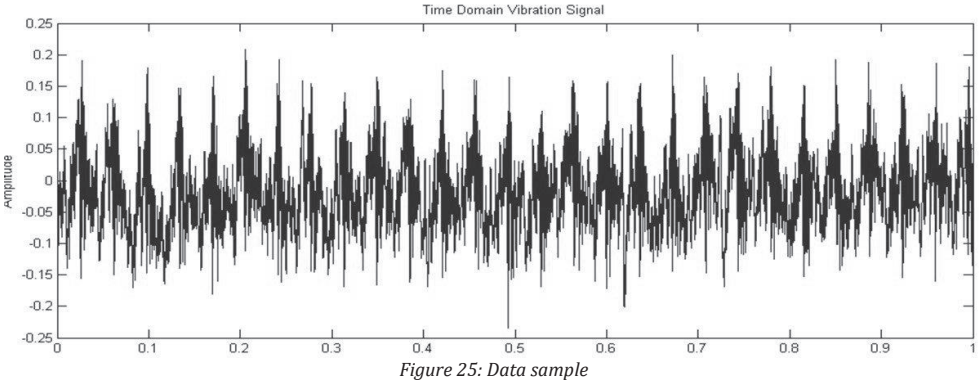

Figure 25: Data sample

To use the proposed concept, the following requirements must be met.

- **Stable System** behaviour is required to predict the condition. In other words, the system should not change its condition frequently. Specifically, the condition should not change during the sampling. If the condition changes at this point, the classifier cannot classify the current state correctly.
- Condition monitoring (**Continuous Sensor Values**) was not developed with discreet values and events in mind. The system needs values which change over time. Most physical systems fit these criteria unless their operation is triggered by a highly unpredictable external source. For example, accelerating a car is a continuous time series, while starting the car or closing a door is not.
- The system works best if the sensor input changes frequently (**High Frequency Sensor Data**) during sampling (e.g. sound, vibration, power consumption etc.). Most data preprocessing steps are developed to work with a signal input of more than 1Hz. But it is possible to work with slow changing values like temperature, even if less information can be extracted from such data.
- The concept relies on data samples collected during given intervals (**Periodical Data Sampling)**, not at discrete or random times. This is less important for condition monitoring but is required for condition prediction. No time information is saved in the recorded data.
- Condition monitoring and condition prediction work with fixed sample lengths. It is not possible to record continuous data samples without modifying the algorithm and splitting the continuous data sample into multiple one-second samples (**Discreet Data Sampling**).
- Calculation of a condition takes time (**No True Real Time Monitoring)**. The calculation of one condition may take as much as ten seconds, depending on the number of decision trees and the number of sensors. If only one tree is used and only a few FFTs need to be calculated, it is possible to have a calculation time of less than one second. In fact, this was the usual case during the experiments. If the calculation time is less than one second, it is possible to calculate the condition while a new data sample is recorded.
- Data samples of more than one condition are needed for learning patterns and classification (**Multiple Conditions**). With the proposed concept, it is not possible to have a "one-class" classifier. A one-class classifier detects incorrect states and conditions based only on the data of the usual operation. Condition monitoring always needs data of at least two different operation modes or conditions. It is possible to change the concept to find incorrect conditions, but this would require a significant change.

3.1.2 Classification Training

The obvious goal of 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 are in chronological order with the same time between each sample, a multiple time series is derived, 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 or she 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 it is not possible to use the condition prediction for the no-failure state, if there is only one, because the time series will have multiple 100 % values in a row, making the prediction impossible. The algorithm does not "know" at which position in time it is, so a class that is not the no-failure class needs to be monitored. If there is only one failure class, that class is selected for the prediction; otherwise, one prediction must be made for each failure class. Each predictor has to be individually trained, significantly increasing the training time.

3.1.2.1 Data Classification

An important part of the training process is to classify each data sample. The learning algorithm needs all training samples to determine the number of features and classification. The classification of the data samples should describe the condition of the system for which the data sample stands. Some possible classifications are given below.

- Each system has a **lifetime** after which it needs to be replaced. Lifetime can be measured in operating hours. If lifetime is used as a condition, it is often useful to use the past lifetime or the remaining useful lifetime (RUL) of the system. Lifetime 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 sensitive to noise (over-fitting) (Quinlan, 1986).
- **System mode** refers to normal operation or failure. This classification is useful to detect failures in a system.

Good classification can significantly influence the performance of the condition monitoring. Many or very specific classes may cause over-fitting and make the system sensitive to noise.

3.1.2.2 Feature Extraction

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 (Gerdes & Scholz, 2009) (Gerdes, et al., 2017). The process depends on different parameters, each of which is adapted to the data. In this

concept, the selection of the optimal parameters is performed by a genetic algorithm. The 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 to the data during preprocessing. First, the data are transformed into the frequency domain, and the noise is reduced. Then, frequencies are grouped. The frequency span of the groups may overlap. For example, 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 are the number of peaks. 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 are calculated as a last step. Table 6 shows the parameters of the preprocessing and the possible values.

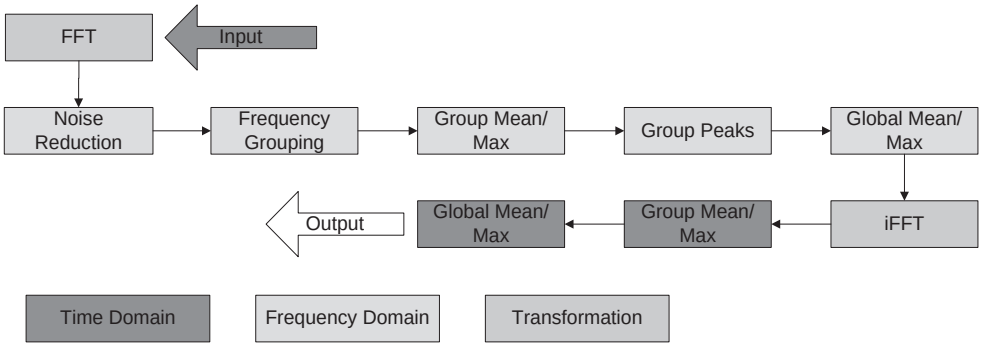

Figure 26: Signal preprocessing

Figure 26 shows the preprocessing steps. The steps are:

- Fast Fourier Transform

- Noise Reduction
- Frequency Grouping
- Group Mean/Max
- Group Peaks
- 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 analyse the individual groups or frequencies in the time domain, without all other frequencies in the signal.
- Group Mean/Max
- Global Mean/Max
- Output: The outputs 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. These are much less data than pure signal data. The total number of the values depends on the width of the frequency groups (blocks).

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

Table 6: Preprocessing parameters

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

The low frequency data were not processed in this research. There were too few data for frequency analysis and compression.

The high frequency data are processed by taking the following steps. First, the data are transformed into the frequency domain, and noise reduction is applied. Next, the frequency data are partitioned into small blocks. Finally, each block group is enhanced with extra information.

Fast Fourier Transform and Grouping

The fast Fourier transform (FFT) takes a number of time-domain samples and transforms them into the frequency domain. The basis of the FFT algorithm is the discrete Fourier. A fast Fourier transform is performed in $O(N\ log(N))$ operations, resulting in the full transformation of the sampling frequency. After the fast Fourier transform, the frequencies are divided into blocks. Note that a frequency group is called a "block". Frequency groups may overlap; this means if a frequency group is from 1 to 100 and the overlap is 50 %, the next frequency group is from 51 to 150 and the following frequency group is from 101 to 200. If the overlap is 0 %, 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 frequencies grouped in one block is determined by the calculation parameter *block width*. If not enough *block width* frequencies are available, all frequencies are treated as one block. After partitioning, all blocks are transformed back into the time domain to get information about the behaviour of the block-signal over the time. Figure 27 shows how a signal in the frequency domain is separated into blocks and how they are transformed back.

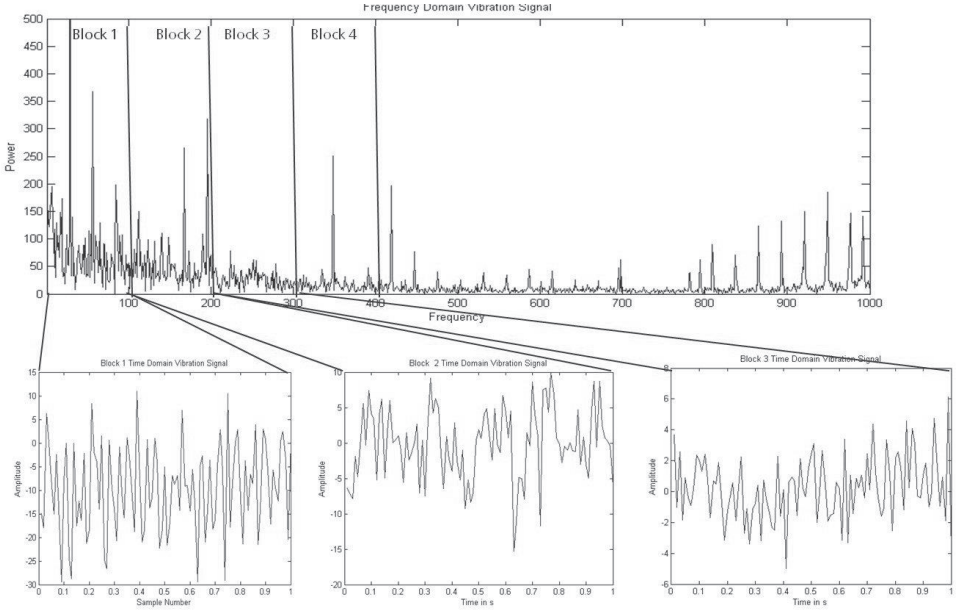

Figure 27: Blocks and inverse Fourier transform

Noise Reduction

Noise reduction is applied to the signal to remove random data from the samples to improve the feature detection of the undisturbed signal. The maximum frequency power is calculated and 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 in the experiments performed for this research (*noise reduction factor*).

Additional Information and Data Compression

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

- Maximum amplitude of each block
- Mean amplitude of each block
- Maximum amplitude of the complete signal
- Mean amplitude of the complete signal

In the frequency domain, the following information is added:

- Mean frequency power of each block
- Maximum frequency power of each block
- Frequency with the highest power of each block
- Number of peaks higher than a given magnitude of the mean frequency power
- Mean frequency power of the complete signal
- Maximum frequency power of the complete signal

The extra information is also calculated for the complete signal sample. Our experiments showed that the added information is more useful for the algorithm than the raw data as it allows data to be compressed. For example, the information for 100 frequencies can be reduced down to four attributes (maximum and mean power, the frequency with the 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.

$$Freq\ Info = 4\frac{Frequencies}{BlockWidth} \tag{27}$$

$$Time\ Info = 2\frac{Frequencies}{BlockWidth} \tag{28}$$

$$\begin{aligned} Total\ Info &= Freq\ Info + Time\ Info \\ &= 6\frac{Frequencies}{BlockWidth} \end{aligned} \tag{29}$$

$$Normal\ Info = 2Frequencies \tag{30}$$

$$Compression = \frac{Total\ Info}{Normal\ Info} = \frac{3}{BlockWidth} \tag{31}$$

The data required are reduced to 3 % if *BlockWidth* = 100 and *Frequencies* = 11000.

3.1.2.3 Decision Tree Building

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 etc.). 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 required accuracy, it is possible to try to improve the performance by modifying the preprocessing parameters.

3.1.2.4 Performance Evaluation

The performance of a decision tree can be improved by modifying the preprocessing process (Gerdes & Scholz, 2009) (Gerdes, et al., 2017). The 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, it is not possible to test all possible combinations. 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. As advantage of the genetic algorithm is that it can be executed in parallel, thus reducing the overall calculation time.

The number of correctly classified samples defines the fitness of the decision tree. This checking is done by classifying a test data set. The test data set can contain data samples used for building the decision tree, but it is preferable to keep the test set separate from the training set. A new parameter set for the feature extraction is created using the genetic algorithm, and a new input vector for the next iteration is created. A decision tree is calculated with each new generated preprocessing parameter set. The optimization continues until a given number of decision trees has been calculated or until a decision tree has a better performance than the limit (Gerdes & Scholz, 2011) (Gerdes & Galar, 2016).

3.1.2.5 Selecting Decision Trees

The best three performing decision trees from the training are used. Only three are selected to allow real time condition monitoring while increasing the accuracy in a noticeable way (Gerdes & Scholz, 2011) (Gerdes, et al., 2016).

3.1.3 Prediction Training

The prediction training step takes the decision trees and parameters from the condition training step and uses these together with the data samples to create a time series made from similarity values for a class. The first step is to create a time series using fuzzy evaluation and to average the result of all three decision trees. Next, data samples for training are extracted and labelled with the best approximation method; after this step, the time series features are extracted, and a decision tree is trained to give the best approximation method for a given data sample. The goal of the prediction is to predict when the system will fail (the RUL is zero). For this reason, it is recommended to try to predict the similarity curve for the 90% RUL time series.

3.1.3.1 Fuzzy Evaluation

Fuzzy decision tree evaluation takes a normal decision tree, evaluates all possible paths 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 given a weight of one, while 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 from the "true" border (decision split). The method selected to calculate the distance is based on the problem. During the evaluation, the values for each path are calculated by taking the sum of the weights of the path and dividing the sum by the depth of the path (taking the average of the path values). This results in a value for each leaf on the decision tree. It is possible for one class to have multiple leaves; in this case, the largest value of all leaves 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 can be applied to any decision tree.

3.1.3.2 Create Samples

The creation of samples for the prediction is controlled by prediction constraints. The samples created are parts of a time series created by using fuzzy evaluation and storing the results. First, a data sample is evaluated using fuzzy evaluation for all three selected decision trees. Then, the average similarity value for each class is taken and added to the similarity time series for the class.

A time series data sample consists of two parts. The first contains the past data from which the features are extracted and which are used as an input to calculate the approximation methods. The second contains additional data points used to select the best approximation method for this data sample. The extrapolation ability of the method is improved by using more data points for the approximation than for the later prediction. This way, the method will use extrapolation methods that are better suited for long term prediction.

Time series data samples can be created in a static or a dynamic way. The basic time series data sample generation process is shown in Figure 28.

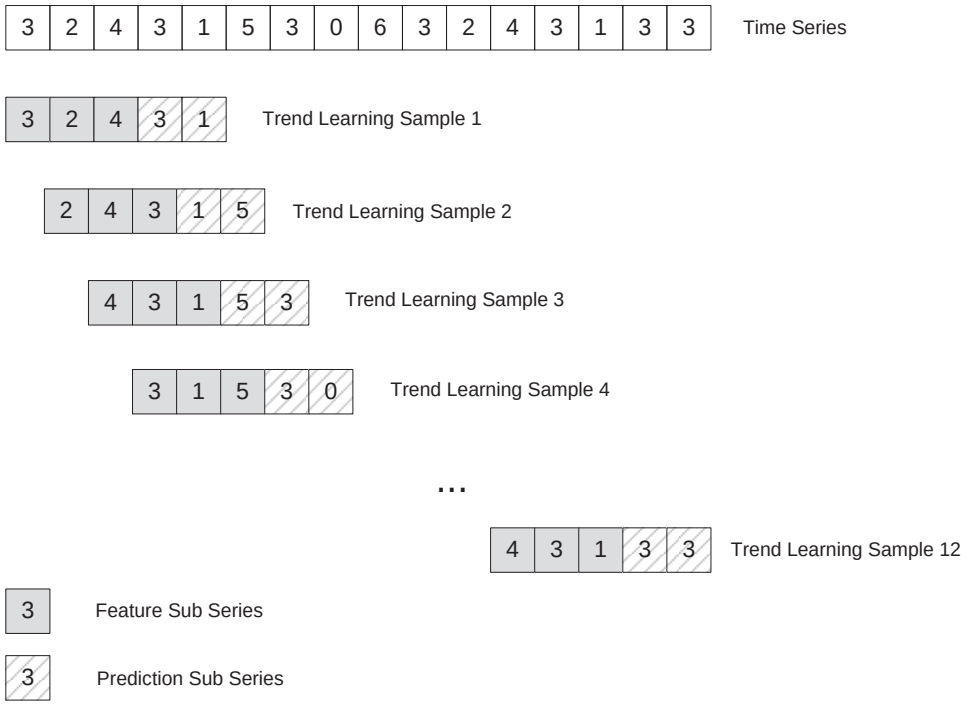

Figure 28: Time series sample generation

Multiple time series data samples 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:

$$w = d_p + d_f \tag{32}$$

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 data are available for a problem.

A **dynamic window** is also possible. In this case, the window size starts with only a few data points, but grows with each step. This is normally the case when the training data represent the time series data as they grow and also include all past data points. A dynamic window can only be used if no features are dependent on the number of data points.

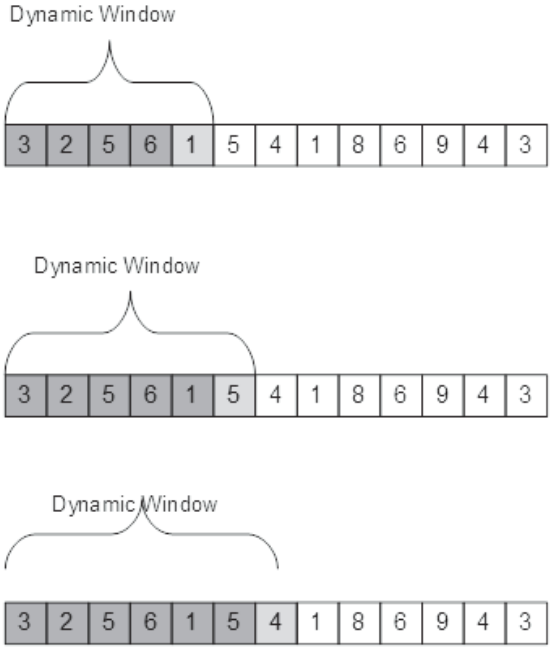

Figure 29: Dynamic window

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

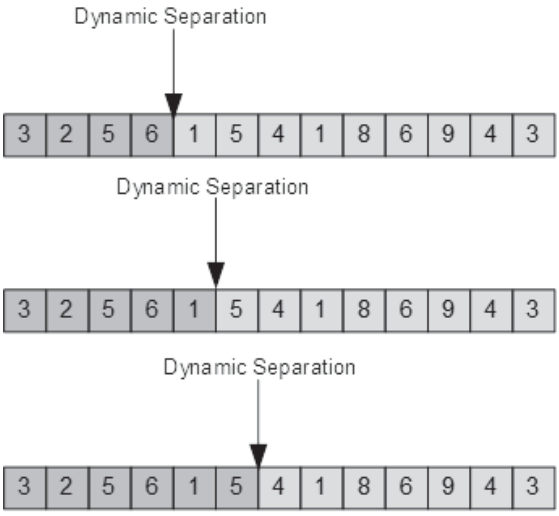

Figure 30: Dynamic time series separation

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.

3.1.3.3 Select Approximation Method

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 those data points that are marked as future data points because the decision tree will also be limited to those data points. 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 to predict data points:

- Linear regression
- Multiple regression
- Moving average
- Exponential smoothing
- Autoregressive integrated moving average (ARIMA)

Which similarity time series will be predicted needs to be decided during this step. It is possible to predict all time series, but it is recommended to predict only one. The most recommended option is to predict only the 90% remaining useful life (RUL) time series. An important consideration for maintenance decisions is knowing when the RUL is reached or is close, i.e., when data samples are classified as 90% RUL.

3.1.3.4 Feature Extraction

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, calculated in a previous step, form the training data sample. Which features are calculated and how they are calculated depends on the preprocessing parameters. This step is similar to preprocessing in the condition monitoring process. The following features are possible:

- Maximum value
- Mean value
- Minimum value
- Gradient

The process is controlled by the following variable parameters:

- Maximum number of past data points
- Use of maximum value
- Use of mean value
- Use of minimum value
- Use of gradient
- Use 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 data. Preprocessing is only applied to the data points marked as past data points. The data points are the same as those used to calculate the classification.

3.1.3.5 Decision Tree Building

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

Testing the decision tree for prediction is more complex than testing it for condition monitoring. Standard methods cannot be used, because the time series prediction is the goal, not decision making. The decision tree is tested by calculating the prediction for the original time series data 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 %

The 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. For example, 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:

$$P_{Pred} = 6 - \frac{w_0}{1 + err_{max}} + \frac{w_1}{1 + err_{mean}} + \frac{w_2}{1 + err_{min}} + w_3 cr_{10} + w_4 cr_5 + w_5 cr_1 \tag{33}$$

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

3.1.3.6 Performance Evaluation

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.

3.2 Monitoring and Interactive Prediction Process

The process for the actual monitoring of the system health and health prediction is divided into two processes; first, the classification process and second, the iterative prediction process. In order to make a prediction, the current system health condition needs to be classified.

The first step (system data) is taking a new sensor data sample; the sample needs to be compatible with the data samples used to train the system. The second step is the classification of a data sample using fuzzy evaluation and attaching the result to a time series. In the last step, the time series data are analysed and an iterative prediction is made based on their features. Each prediction iteration predicts only the next data point. That data point is added to the time series, and a new iteration is begun to predict the next data point.

3.2.1 System Data

The same sensors and data sources are used to create a sample with the same parameters as in the training process. Just one sample is taken; there is no history of past samples. However, past samples may be stored and added to the training set to improve the accuracy of the decision tree after a new training iteration.

3.2.2 Condition Classification

The classification of condition step uses the decision trees and feature extraction parameter sets from the training process. The parameter sets are used to create three different feature vectors for the three different decision trees. The decision trees evaluate these feature vectors and classify them using fuzzy decision tree evaluation. As a result, the sample is applicable to all ten health classes. The result is added to a time series that contains the classification of the past data samples.

3.2.2.1 Feature Extraction

The feature extraction is performed once for each decision tree using the parameter sets for each tree. Input for the feature extraction is the currently recorded sensor sample.

3.2.2.2 Classification

The previously generated feature vectors are now classified using fuzzy decision tree evaluation. This generates a set of similarities for each decision tree. The next step is the fusion of the classification results using a voting mechanism.

1. For each decision tree result select the class with 100% similarity, i.e., the class to which the sample was classified.

2. Decide which tree to use. The decision tree (DT) with the highest classification accuracy has the highest priority.
    a. Take the first DT if it and another DT have the same classification.
    b. Take the first DT if all DTs have different results.
    c. Take the second DT if it and the third have the same classification.
3. Select the 100% class of the previously selected DT for the classification of the time series.
4. Select the goal class of the previously selected DT to add it to the prediction of the time series.

3.2.2.3 Time Series

Two time series are generated by this process, as indicated in the previous step. The first is the classification time series, containing the actual states of the monitored system. The second is the prediction time series. These data contain the similarity values of the goal class (normally the 90% RUL class; see Section 3.1.3.3).

3.2.3 Iterative Prediction

The main goal of the proposed method is to predict the RUL. The prediction uses the training prediction decision tree and the fuzzy classification result of the condition monitoring. The process extracts features of the time series and uses them to decide which extrapolation method will be used to forecast the next data point. The new generated data point is added to the time series, and the process is repeated until the target number of future data points is calculated.

3.2.3.1 Feature Extraction

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

3.2.3.2 Select Extrapolation Method

The trained decision tree and feature vector are used to select the best extrapolation method for the time series which the features represent. The extrapolation method can only be selected from the set of previously trained approximation methods.

3.2.3.3 Calculate Next Data Point

The next future data point in the time series is calculated using the selected extrapolation method and the current data point of the time series. A possible modification is to calculate multiple data points instead of one. This depends on the goal of the prediction and is applicable when only one iteration is calculated.

3.2.3.4 Iterate

3.3 Summary

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

4 VALIDATION

The validation of the method required three steps. First, it was validated for condition monitoring using data from a test rig at Airbus; second, computer generated data were used to validate the condition prediction; third, real-world data from an aircraft were used to validate the complete method.

4.1 Condition Monitoring: Test Rig Validation

The test rig experiments were divided in two parts, one for each step of the process (monitoring and prognosis). A test rig 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 tried to simulate the filter clogging of the high-pressure air filters in the air conditioning of the A340-600 aircraft. This system was chosen because it has no active parts; it is completely passive and thus difficult to monitor. But the system is connected to fans and air flows through the filters.

4.1.1 Test Rig

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

- HP recirculation fan of the A340-600
- HP air filter of the A340-600
- Ducting
- Electronic vibration measurement box

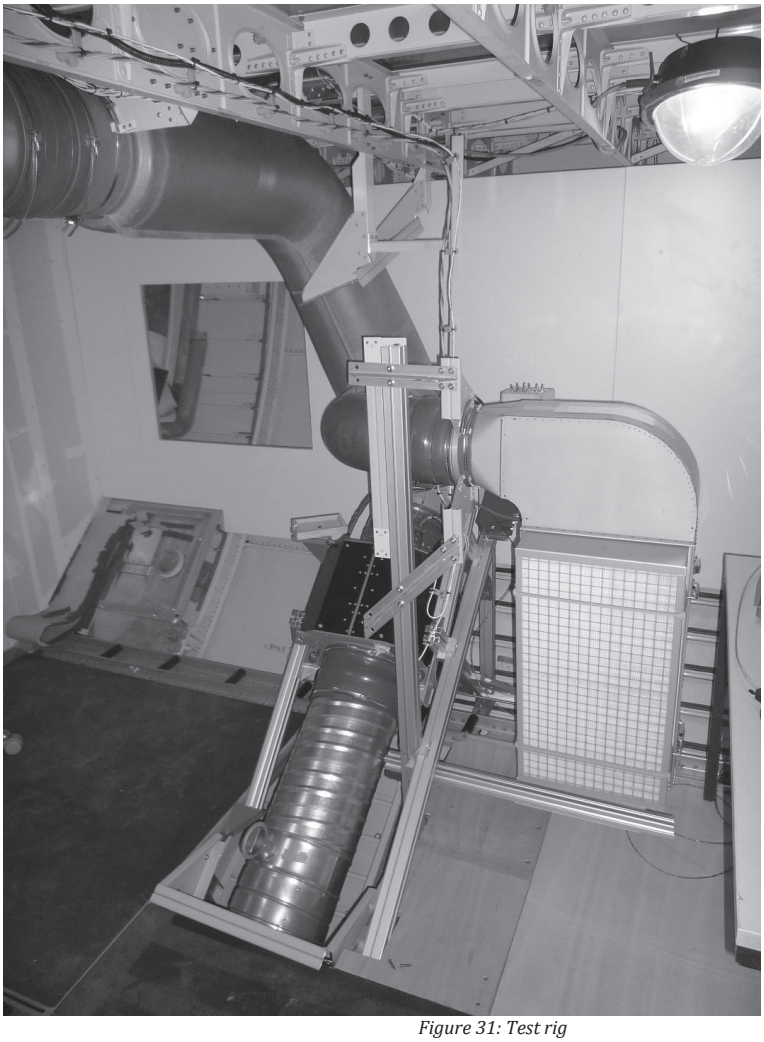

Figure 31: Test rig

An electronic vibration measurement box (EVB) equipped with two vibration sensors and two microphones was used to record sensor data. One of each sensor type was attached to the fan and to the filter housing. The EVB (Figure 32) is hardware developed for PAHMIR. Ten boxes were created for different tasks. The design goal was to have a box able to record and store different sensor data for a long time (8 weeks). During the project, the EVB was used to record data from different experiments and for the ground test rig. It was also used on an Airbus Operations test flight in Toulouse to record data (Gerdes, 2008) (Grieshaber, 2009).

Parts breakdown:

- Enclosure

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

Dimensions:

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

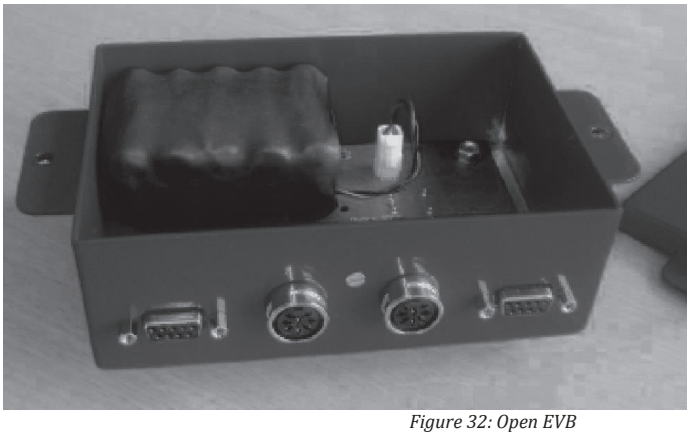

Figure 32: Open EVB

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 has 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 cruise 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 too low or high. Up to four external sensors can be attached to the EVB. Sensor data are recorded as a four-channel wave file.

The EVB has a simple configuration file 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 between two recordings (default: 600 seconds)
- How much time the sensors take to stabilize at power on (default: 50 milliseconds)
- Gain for each channel (default: 1)

4.1.2 Validation

The experiments discussed here show the ability of the concept to detect clogging of air filters. The condition monitoring experiment simulated the clogging of an air filter with dust. Other experiments tested the preprocessing, optimization and fuzzy decision tree evaluation. The parameter optimization experiments appear in (Gerdes & Scholz, 2009) (Gerdes, et al., 2017), the optimization experiments appear in (Gerdes & Scholz, 2011) (Gerdes, et al., 2016) and the fuzzy decision tree experiments are in (Gerdes & Scholz, 2011) (Gerdes & Galar, 2016).

4.1.2.1 Setup

Data for the experiments were collected at the ground test rig. Data included fan and filter vibration and sound data. 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 grams up to 200 grams per filter for a total of eight possible classes. The classifier was trained by applying the optimization process with genetic optimization, with 10 generations each for 20 members. The starting population was a random parameter list. Fifteen data samples were used for every class (pollution grade). To test if the classification accuracy could be increased, the experiments were performed with a single tree classifier and also with a classifier built out of three different decision trees using the decision selected by the majority of the classifiers.

4.1.2.2 Results

The goal was to detect how much dust was in the filters. The complete data set was used for the experiment. For test data, the training data were arranged by increasing weight. The classification should detect a nine-step function. Table 7: Number of classified samples per classes for one and three decision trees

shows the number of detected samples per class (15 samples per class were recorded). Figure 33 shows that the calculation resembles a step function, and only a few classes were classified wrongly. With three classifiers (same training data, different parameters), the number of wrong classified classes dropped even more (Figure 34).

Figure 35 shows the results of the optimization. The red line is the average fitness, and the green line is the best fitness. The figure clearly shows that more generation would have given better performance. The average and maximum fitness of the population steadily increased.

Dust	Classification results using 1 decision tree (samples per class)	Classification results using 3 decision trees (samples per class)
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

Table 7: Number of classified samples per classes for one and three decision trees

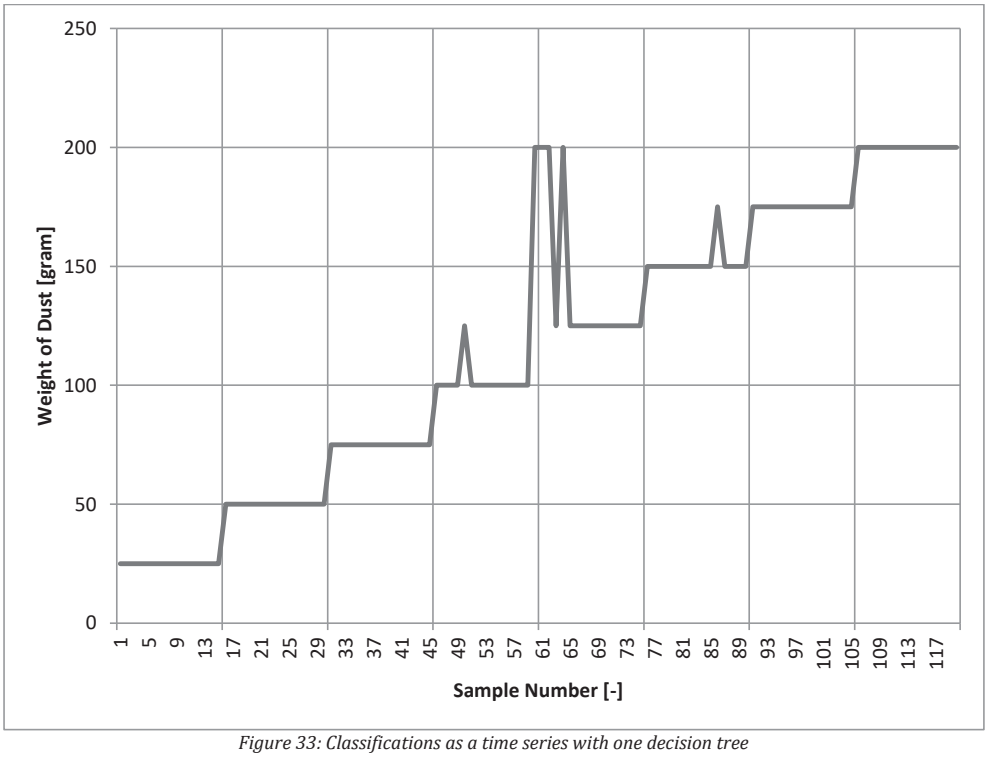

Figure 33: Classifications as a time series with one decision tree

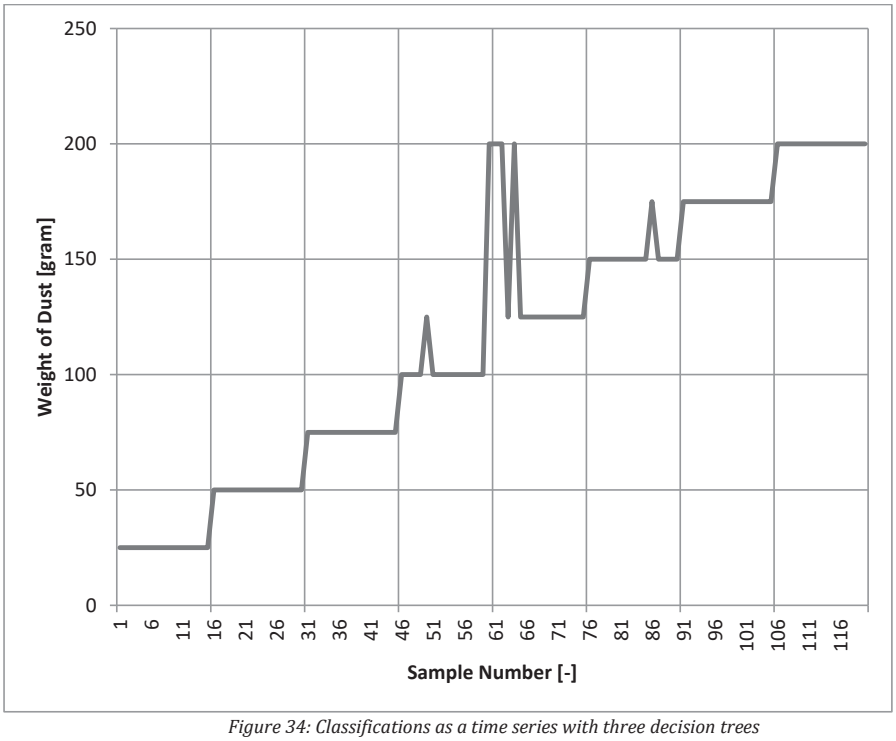

Figure 34: Classifications as a time series with three decision trees

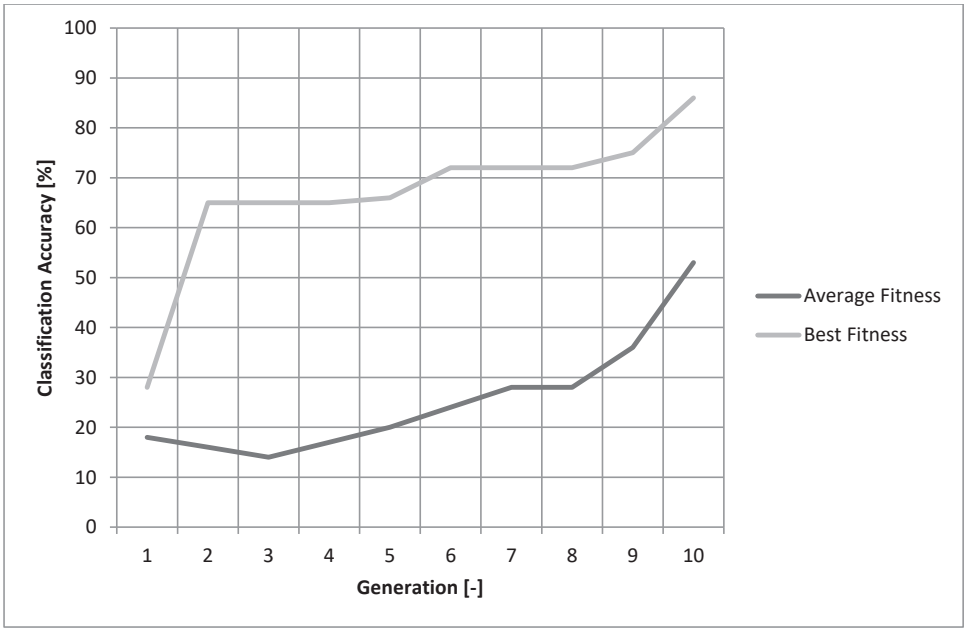

Figure 35: Genetic optimization performance

4.2 Condition Prediction: Validation with Generated Data Validation

The goal was to predict a typical system health condition function (Kolerus & Wassermann, 2011) and follow the function as closely as possible. The experiment used the condition prediction method presented earlier to predict the goal function. Details about the experiment and results are shown in (Gerdes, 2013). Kret (Kret, 2011) shows how the concept can be used for long horizon forecasts. Kret (Kret, 2011) also evaluates time series models like moving average, exponential smoothing, ARMA and ARIMA.

4.2.1 Setup

The experiment used Matlab to generate the data. Figure 36 shows the function used as the basis to generate data and which should be reproduced as closely as possible by the algorithm.

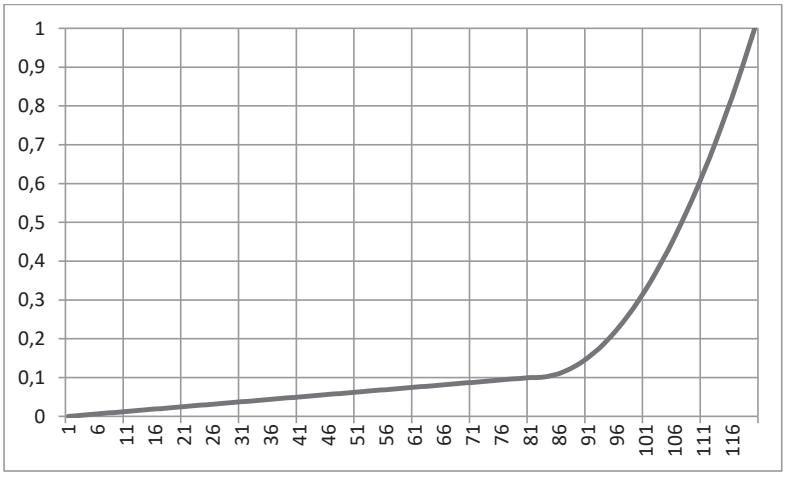

Figure 36: Test function for validation with generated data

The function should represent the typical health degradation of a component. First, there is a slow rise in the degradation and then a sudden rise until the system breaks. Equation (34 defines the function until data point 80; Equation (35 defines the remaining 40 data points.

$$f(x) = \frac{0.1}{80}x \tag{34}$$

$$f(x) = \left(\frac{x-81}{40}\right)^2 + 0.1 \tag{35}$$

The function was transformed into a set of features representing the function at a certain data point. The features were:

- Short term gradient calculation [Boolean]. This parameter defines whether the gradient for the data points in the current window should be calculated.
- Long gradient calculation [Boolean]. This parameter defines whether the gradient for the complete time series until the last data points should be calculated.
- Mean value calculation [Boolean]. This parameter defines if the mean of the sample data points should be calculated.
- Maximum value calculation [Boolean]. This parameter defines if the maximum of the sample data points should be calculated.
- Minimum value calculation [Boolean]. This parameter defines if the minimum of the sample data points should be calculated.
- Dimensions used for classification [Boolean list]. This parameter defines which dimensions should also be used to calculate characteristics.
- Zero Crossing [Boolean]. This parameter decides if the number of zero crossings in the current window should be calculated.

The "class" for each feature best approximated the function up to the data point for which the current feature set was generated. The following functions were available:

- $f(x) = ax + b$
- $f(x) = ax^2 + bx + c$
- $f(x) = ax^3 + bx^2 + cx + d$
- $f(x) = a\, e^{b(x+c)} + d$
- $f(x) = a \log\big(b(x + c)\big) + d$
- $f(x) = a \sin\big(b(x + c)\big) + d$
- $f(x) = a \cos\left(b(x + c)\right) + d$
- $f(x) = 1 - a\, e^{bx(x+c)}$
- $f(x) = ab^{x+c}$
- $f(x) = a(b(x + c))^2 + d$

The function was predicted based on three different starting points: 30, 60 and 90. Three different experiments were performed to show how well the algorithm could predict the function. The first experiment had no noise in the data; the second had noise in the range between -0.02 and 0.02 added to the function; the third added noise between -0.05 and 0.05. The second and third experiments were done once with noise-less training data and noisy test data and once with noisy training and test data.

4.2.2 Results

The results show (Figure 37, Figure 38 and Figure 39) that the method can predict the function quite well if the data are not disturbed by noise. If noise is added, the performance is worse. The method does not discern its position in time and starts to predict the "exponential part" of the goal function too early.

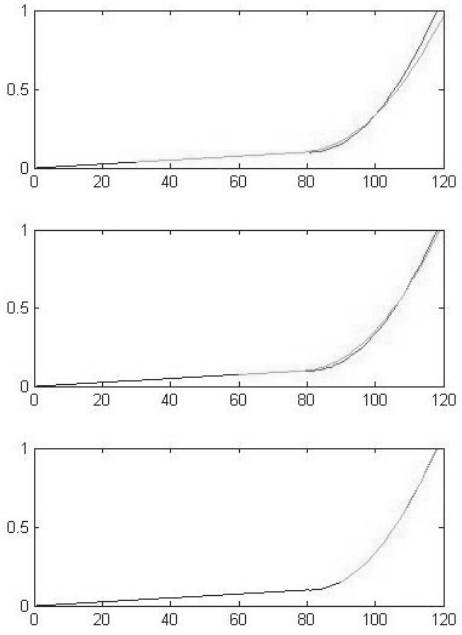

Figure 37: Prediction results with no noise

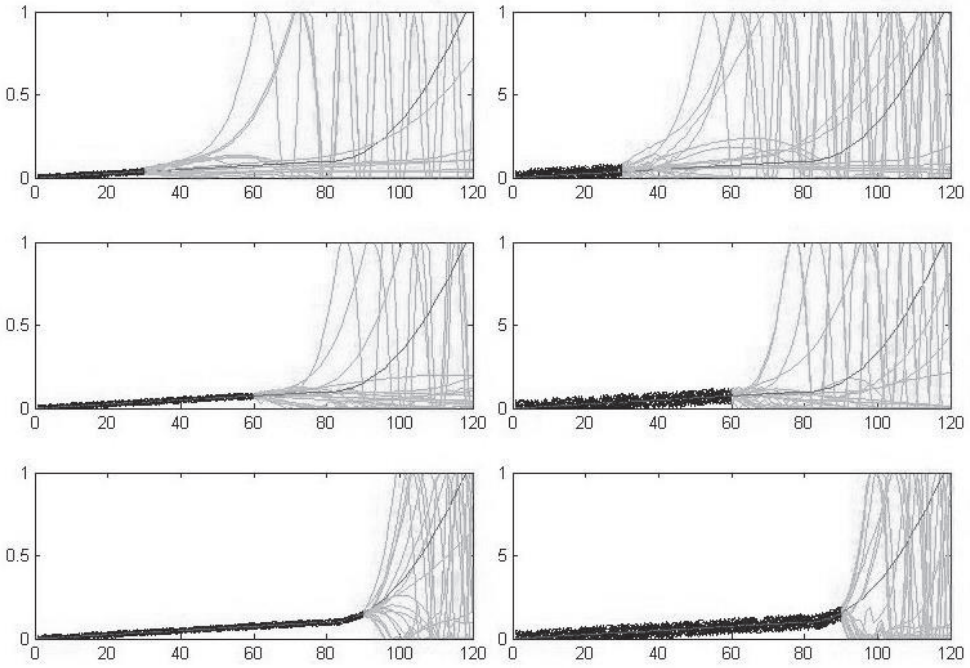

Figure 38: Prediction results with noisy test samples

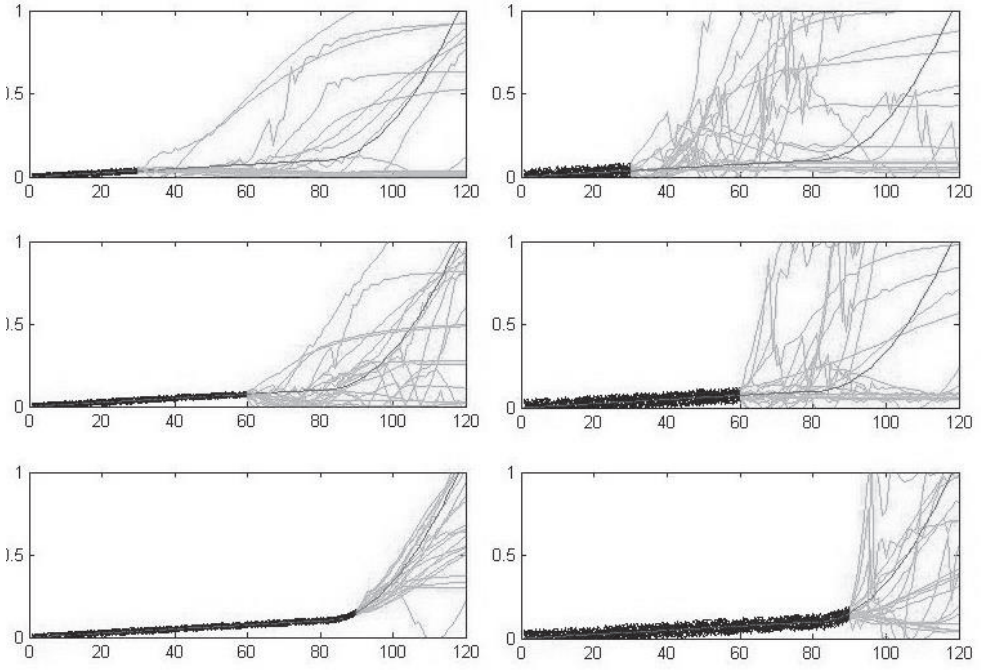

Figure 39: Prediction results with noisy data samples

4.3 Condition Monitoring and Prediction: In-Service Aircraft Validation

The method was evaluated using sensor data from the air conditioning system of an A320 aircraft operated by ETIHAD Airways in the Middle East. The sensor data included 589 flights over two years. No additional information was available – no data about maintenance, weather or number of passengers in the aircraft.

4.3.1 Data

Each sensor reading included over 80 values, consisting of continuous (numerical) and discrete data (Boolean) and sampled with a frequency of 1 Hz. Data came from bus systems in the air conditioning system and flight data system. Most were temperature data and valve state data. The sensor data are described in the following multi-page table (Table 8).

Description	Bus	Type
Total Air Temperature	Air Data Computer	Numerical
Date	Flight Data Interface Unit	Calendar Date
UTC Hours	Flight Data Interface Unit	Numerical
UTC Minutes	Flight Data Interface Unit	Numerical
UTC Seconds	Flight Data Interface Unit	Numerical
Pressure Altitude (1013 hPa)	Air Data Computer	Numerical
Present Position Lateral	Inertial Reference System	Numerical
Present Position Longitude	Inertial Reference System	Numerical
Cabin Compartment Temperature Group 1	Zone Control	Numerical
Cabin Compartment Temperature Group 2	Zone Control	Numerical
Cabin Compartment Temperature Group 3	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 1	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 2	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 3	Zone Control	Numerical
Duct Overheat Warning Group 1	Zone Control	Boolean
Duct Overheat Warning Group 2	Zone Control	Boolean
Duct Overheat Warning Group 3	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 1	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 2	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 3	Zone Control	Boolean
Duct Temperature Group 1	Zone Control	Numerical
Duct Temperature Group 2	Zone Control	Numerical
Duct Temperature Group 3	Zone Control	Numerical
G + T Fan OFF	Zone Control	Boolean
Hot Air Switch Position ON	Zone Control	Boolean
Minimum Bleed Air Pressure Demand	Zone Control	Numerical
Nacell Anti Ice - Engine 1	Zone Control	Boolean

Nacell Anti Ice - Engine 2	Zone Control	Boolean
Recirculation Fan Left Hand Fault	Zone Control	Boolean
Recirculation Fan Right Hand Fault	Zone Control	Boolean
Trim Air Pressure Regulation Valve Disagree	Zone Control	Boolean
Trim Air Pressure High	Zone Control	Boolean
Trim Air Pressure Regulation Valve Close	Zone Control	Boolean
Trim Air System Inoperational	Zone Control	Boolean
Zone Main Control Inoperational	Zone Control	Boolean
Zone Secondary Control Inoperational	Zone Control	Boolean
Abnormal Pressure - Engine 1	Bleed Monitoring Computer	Boolean
Abnormal Pressure - Engine 2	Bleed Monitoring Computer	Boolean
APU Bleed Push Button ON	Bleed Monitoring Computer	Boolean
APU Bleed Valve Fully Closed	Bleed Monitoring Computer	Boolean
APU Command	Bleed Monitoring Computer	Boolean
Bleed Push Button Position ON - Engine 1	Bleed Monitoring Computer	Boolean
Bleed Push Button Position ON - Engine 2	Bleed Monitoring Computer	Boolean
Cross Feed Valve Auto Closure	Bleed Monitoring Computer	Boolean
Cross Feed Valve Auto Opening	Bleed Monitoring Computer	Boolean
Cross Feed Valve Manual Closure	Bleed Monitoring Computer	Boolean
Cross Feed Valve Manual Opening	Bleed Monitoring Computer	Boolean
Cross Feed Valve Position Fully Closed	Bleed Monitoring Computer	Boolean
Cross Feed Valve Position Fully Open	Bleed Monitoring Computer	Boolean
Cross Feed Valve Selected Position Open	Bleed Monitoring Computer	Boolean
Cross Feed Valve Selected Position Shut	Bleed Monitoring Computer	Boolean

Cross Pressure Regulation Valve Command Close	Bleed Monitoring Computer	Boolean
Engine 1 Precooler Inlet Pressure	Bleed Monitoring Computer	Numerical
Engine 1 Precooler Outlet Temperature Bleed Monitoring Computer 1	Bleed Monitoring Computer	Numerical
Engine 1 Precooler Outlet Temperature Bleed Monitoring Computer 2	Bleed Monitoring Computer	Numerical
Engine 2 Precooler Inlet Pressure	Bleed Monitoring Computer	Numerical
Engine 2 Precooler Outlet Temperature Bleed Monitoring Computer 1	Bleed Monitoring Computer	Numerical
Engine 2 Precooler Outlet Temperature Bleed Monitoring Computer 2	Bleed Monitoring Computer	Numerical
Flap Actuator Valve Position Fully Closed - Engine 1	Bleed Monitoring Computer	Boolean
Flap Actuator Valve Position Fully Closed - Engine 2	Bleed Monitoring Computer	Boolean
Flap Actuator Valve Position Fully Open - Engine 1	Bleed Monitoring Computer	Boolean
Flap Actuator Valve Position Fully Open - Engine 2	Bleed Monitoring Computer	Boolean
High Pressure Valve Fault - Engine 1	Bleed Monitoring Computer	Boolean
High Pressure Valve Fault - Engine 2	Bleed Monitoring Computer	Boolean
High Pressure Valve Position Full Open - Engine 1	Bleed Monitoring Computer	Boolean
High Pressure Valve Position Full Open - Engine 2	Bleed Monitoring Computer	Boolean
High Pressure Valve Position Fully Closed - Engine 1	Bleed Monitoring Computer	Boolean
High Pressure Valve Position Fully Closed - Engine 2	Bleed Monitoring Computer	Boolean
Over Pressure Valve Position Full Open - Engine 1	Bleed Monitoring Computer	Boolean
Over Pressure Valve Position Full Open - Engine 2	Bleed Monitoring Computer	Boolean
Overheat - Engine 1	Bleed Monitoring Computer	Boolean

Overheat - Engine 2	Bleed Monitoring Computer	Boolean
Over Pressure - Engine 1	Bleed Monitoring Computer	Boolean
Over Pressure - Engine 2	Bleed Monitoring Computer	Boolean
Precooler In Pressure 10/60 - Engine 1	Bleed Monitoring Computer	Boolean
Precooler In Pressure 10/60 - Engine 2	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Auto Closure Control - Engine 1	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Auto Closure Control - Engine 2	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Low Regulation	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Position Full Open - Engine 1	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Position Full Open - Engine 2	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Position Fully Closed - Engine 1	Bleed Monitoring Computer	Boolean
Pressure Relief Valve Position Fully Closed - Engine 2	Bleed Monitoring Computer	Boolean
Start Valve Closed - Engine 1	Bleed Monitoring Computer	Boolean
Start Valve Closed - Engine 2	Bleed Monitoring Computer	Boolean
Transferred Pressure - Engine 1	Bleed Monitoring Computer	Numerical
Transferred Pressure - Engine 2	Bleed Monitoring Computer	Numerical
X Feed Command	Bleed Monitoring Computer	Boolean
Pack 1 Out Temperature	Pack Control	Numerical
Pack 2 Out Temperature	Pack Control	Numerical
Pack Flow 1	Pack Control	Numerical
Pack Flow 2	Pack Control	Numerical
Pack Water Extractor Temperature 1	Pack Control	Numerical

Pack Water Extractor Temperature 2	Pack Control	Numerical
Pack Outlet Temperature 1	Pack Control	Numerical
Pack Outlet Temperature 2	Pack Control	Numerical

Table 8: A320 sensor data description

Figures 40 and 41 show where the sensors are located in the aircraft.

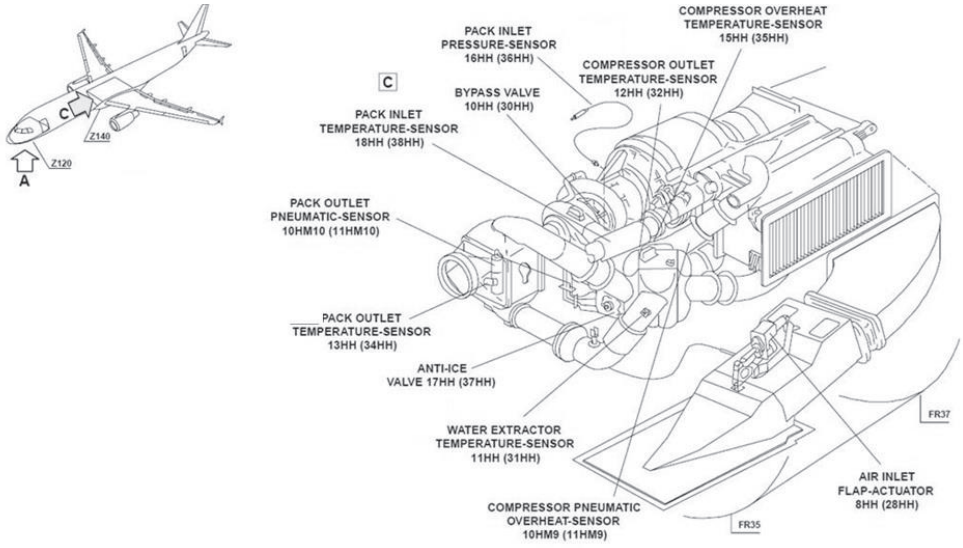

Figure 40: A320 AeroBytes data description 1

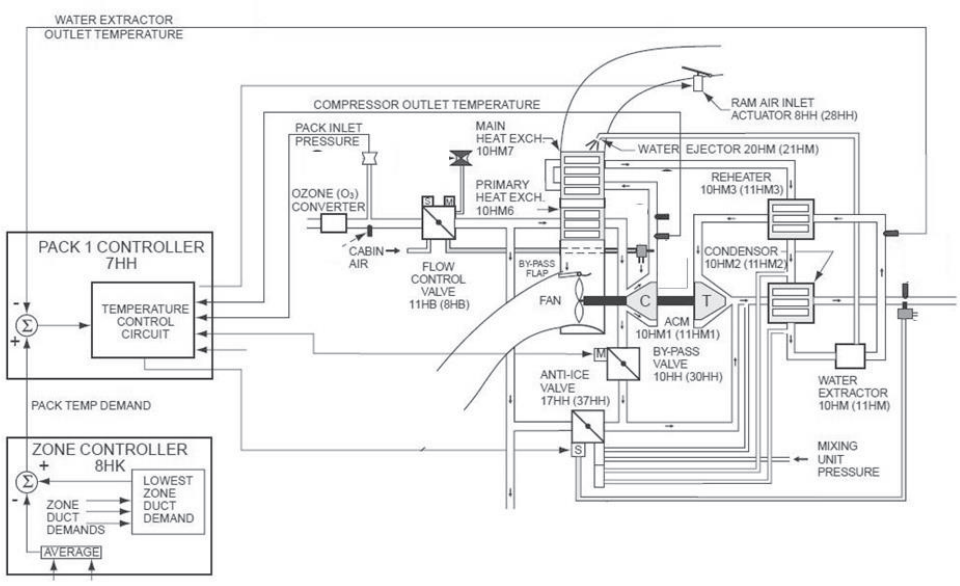

Figure 41: A320 AeroBytes data description 2

Data analysis showed a maintenance action after approximately 930 flight hours (FH) (Gerdes, et al., 2017). Samples for the decision tree generation were created by taking 1024 sensor data points after the aircraft reached a height of 30,000 ft. This equals 17 minutes of data in cruise flight and yields about 3400 data samples; 25% were used for training and 25% were used for testing. Samples were randomly taken from the flight data to avoid having the training data overlap with the test data and to ensure that the number of samples per class was not equally distributed. The data samples were divided into ten categories. The first category equalled all data samples with a time stamp lower than 10% of the RUL of the system, i.e., 930 FH. The maintenance action during the duration of the recorded data was assumed to be the end of the lifetime and the first data point is the beginning. We did this because no data above the whole life of the system are available. However, this should be sufficient to validate the concept. The last category contained samples with a timestamp between 90% and 100% of the RUL. All categories in between were split equally, each covering a 10% range of RUL.

4.3.2 Data Analysis

The results of applying the method to the real-world data are mixed. The condition monitoring works quite well if the system has collected some history, but the prognosis does not work well. As Figure 38, and Figure 39 show, the similarity of one class over time varies a great deal and is strongly influenced by noise. This is partly due to the data used. There are many discrete values

in the data, and the variations in the data are very small. Small changes can strongly affect the classification and similarity results.

Data analysis shows a maintenance action after approximately 840 flight hours (FH). The nature of the maintenance action is unknown, but an examination of the sample classifications shows that the data samples after 840 FH are like the ones at the beginning of the data recording. This indicates that the maintenance action reset the system condition and hints at a major maintenance action.

Table 9: Misclassification matrix without maintenance action shows the misclassification of samples if it is assumed that there was no maintenance action. This is also visible in Figure 42, Figure 43, Figure 44 and Figure 48, where the misclassifications are often the current class +/- “50”. The green marked entries are the correct classifications. On their left and right are misclassifications that are very similar. Parallel to the correct classifications are two other groups of misclassifications. The misclassifications in these groups indicate that the class "50" is like the class "10", class "60" is like class "20", class "70" is like class "30" … In other words, after class "40", the health condition is very like the beginning of the time series so something has reset the system health condition. Table 10 shows the misclassifications after it is assumed that at flight hour 840, the system health condition was reset. Now the misclassifications are neighbours of the correct classification; this is a good sign, because neighbouring classes should have similar features. This also shows that the assumption of the maintenance action at 840 was correct.

Classification/ Class	0	10	20	30	40	50	60	70	80	90
0	0	36	24	23	10	24	29	16	28	6
10	19	0	40	3	4	45	21	12	1	17
20	16	38	0	4	0	29	68	9	0	0
30	28	0	3	0	37	8	6	12	72	4
40	16	6	0	45	0	9	0	20	38	29
50	21	49	28	0	4	0	36	9	0	32
60	30	48	35	1	0	42	0	7	0	0
70	31	30	11	38	20	21	32	0	29	6
80	29	0	0	67	55	0	0	25	0	13
90	23	20	0	10	46	23	0	6	17	0

Table 9: Misclassification matrix without maintenance action

Classification/ Class	0	10	20	30	40	50	60	70	80	90
0	0	21	32	21	7	8	9	6	7	47
10	22	0	32	25	9	23	18	21	10	24
20	23	19	0	36	36	8	17	6	2	25
30	26	49	35	0	54	15	16	0	0	0
40	31	4	29	32	0	36	8	0	0	0
50	16	15	28	41	34	0	20	0	0	0
60	19	22	15	16	5	10	0	10	27	23
70	14	21	15	0	0	0	51	0	25	54
80	2	5	6	0	0	0	68	43	0	73
90	30	28	29	0	0	0	33	48	46	0

Table 10: Misclassification matrix with maintenance action

As mentioned above, the maintenance action during the duration of the recorded data is assumed to be the end of the lifetime, and the first data point is the beginning. We made this assumption because no data above the whole life of the system were available. However, this should be sufficient to validate the concept. The RUL of the system was set to 840 FH, because of the maintenance action around FH 840. All samples between 0 FH and 840 FH were labelled

accordingly into the 10 classes; the same was done for the samples between 840 FH and 1700 FH, thus simulating two total lifetimes of the system.

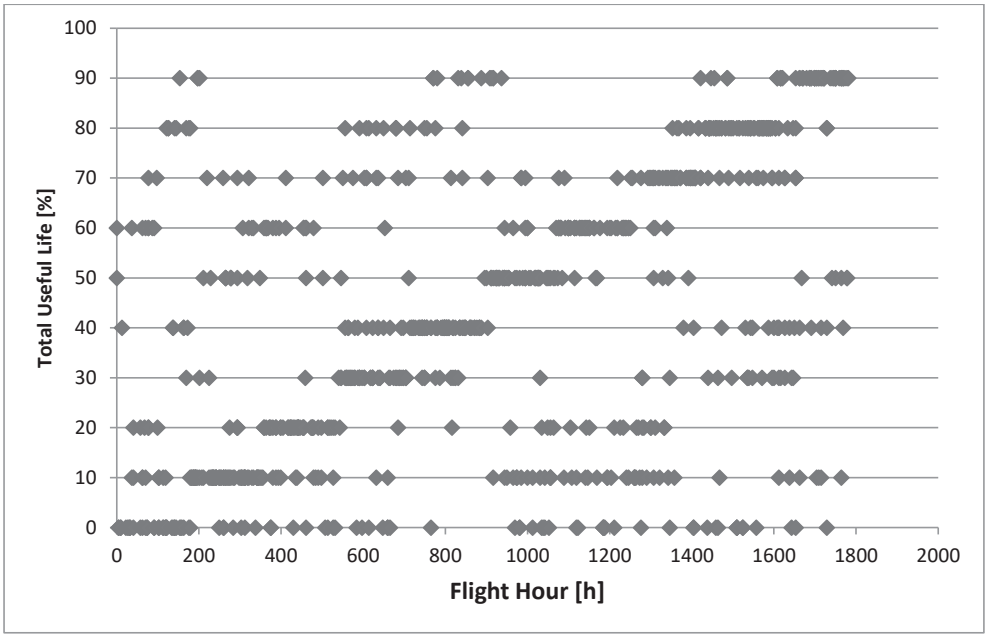

Figure 42: 90% RUL similarities

The similarity graphs also show the maintenance action around 840 FH. The classifications after 840FH seem to have a mirror "line" located four classes "below" the diagonal. The three sample graphs show a very different behaviour, but the curves before and after the maintenance action are similar, indicating a fixed correlation between the similarity curves and the RUL. However, the patterns in a single similarity curve are too weak and unsuited for the iterative prediction approach. This makes the prediction of the time series for the "90% RUL" class very unreliable using an iterative approach as initially proposed. A more complex prediction using more than one curve might solve the issue.

4.3.3 Extrapolative Prediction Method

To handle the high noise in the data, a different predication method was developed. This method does not use the similarity data; instead, it uses the switch from one RUL class to another. In this approach, the FH is noted when a class switch occurs; after two switches are recorded, it is possible to predict the RUL by using those two points for extrapolation (creating a linear model of the time series).

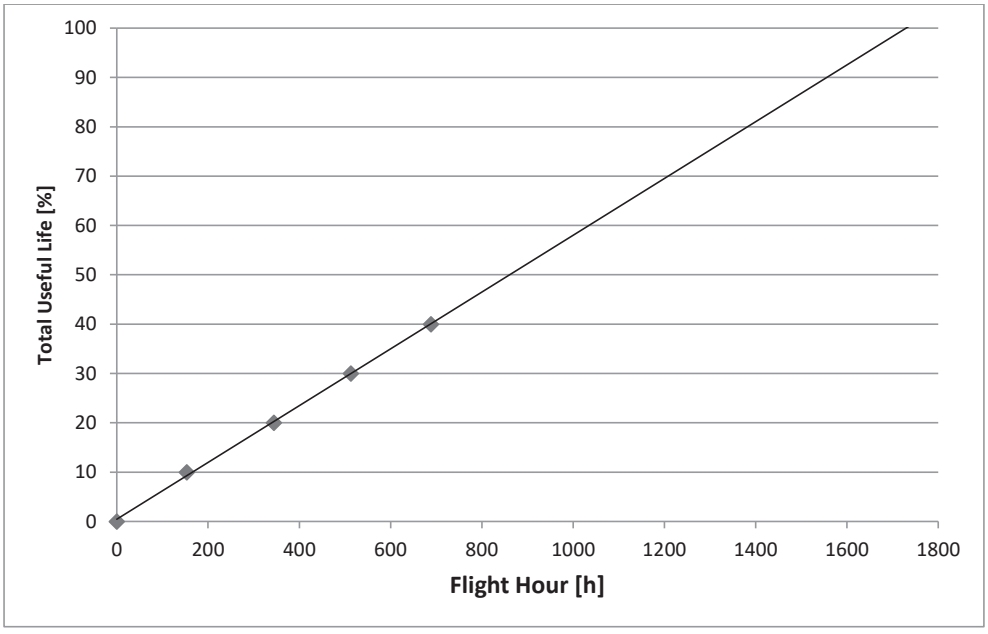

Figure 43: Remaining useful life prediction

Figure 44 shows the modified training process for this method. The major difference is that only the condition detection is trained; this time, the prediction method is not included.

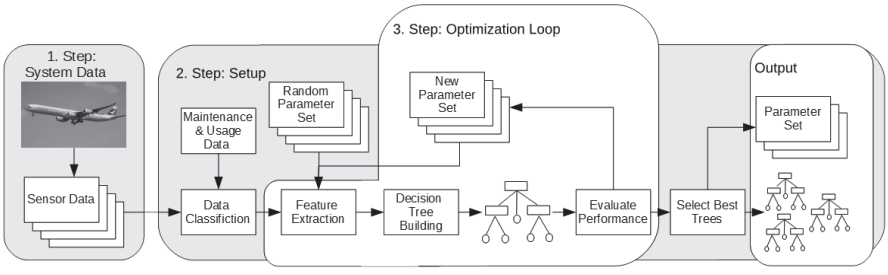

Figure 44: Alternative classification training process

The rest of the training process is unmodified. The alternative prediction process is completely new and shown in Figure 45.

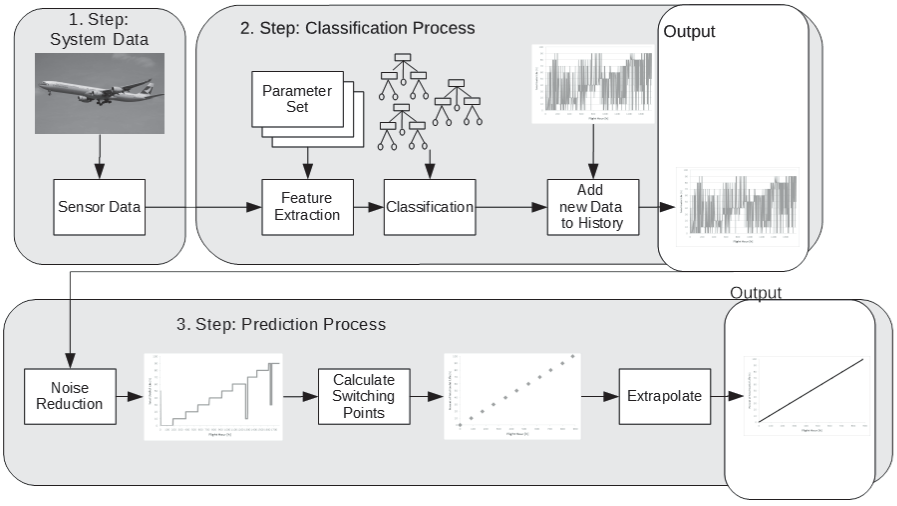

Figure 45: Alternative classification monitoring and prediction process

The process to predict the condition is based on the results of continuous condition monitoring and classification. Each classification adds a new data sample to the class time series. The process has three steps. The first is the recording of new sensor data; the second is classifying the sensor data and adding the classification result to a time series. A new data sample is taken and processed according to the feature extraction parameter set of each of the three selected trees. A voting process classifies the sample. If two or more trees classify a data sample as the same class, this class is taken. If all three trees get a different result, the result of the first tree is taken. The third step is marking when the system condition switches from one state to another and using these data points to extrapolate the data into the future.

The resulting time series data are subject to noise in form of wrong classifications (Figure 46). To reduce the noise, each data point is set to the class of the majority of its 20 preceding data samples (Figure 47) (this value may change depending on the noise in the time series). The noise can be even more reduced for the extrapolation if it is assumed that the health condition can only degrade and not improve unless a maintenance action is performed. A maintenance action restarts the monitoring process and “forgets” all previous classifications.

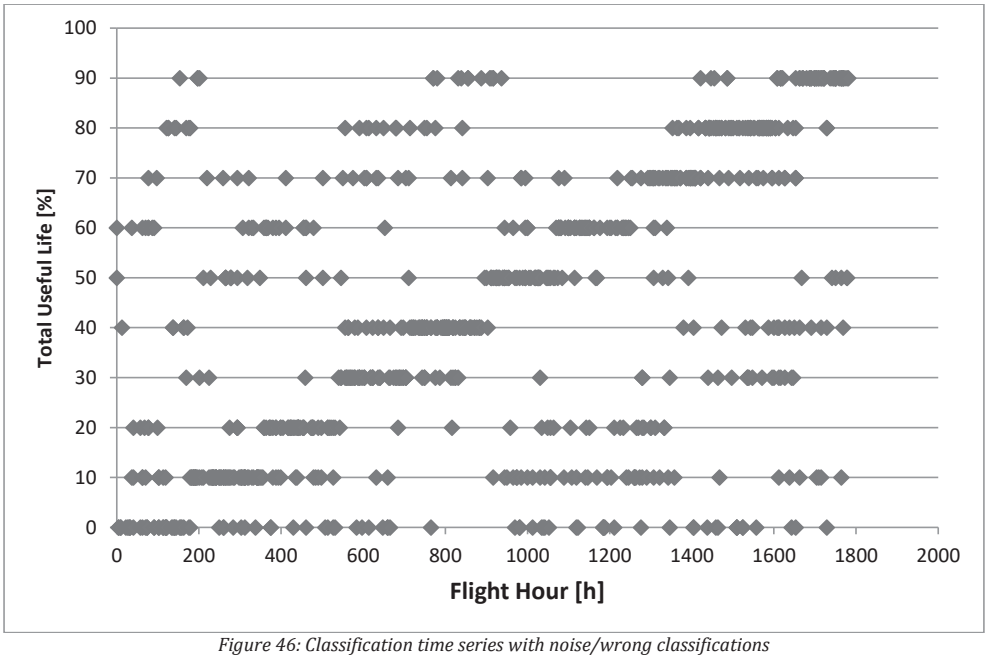

Figure 46: Classification time series with noise/wrong classifications

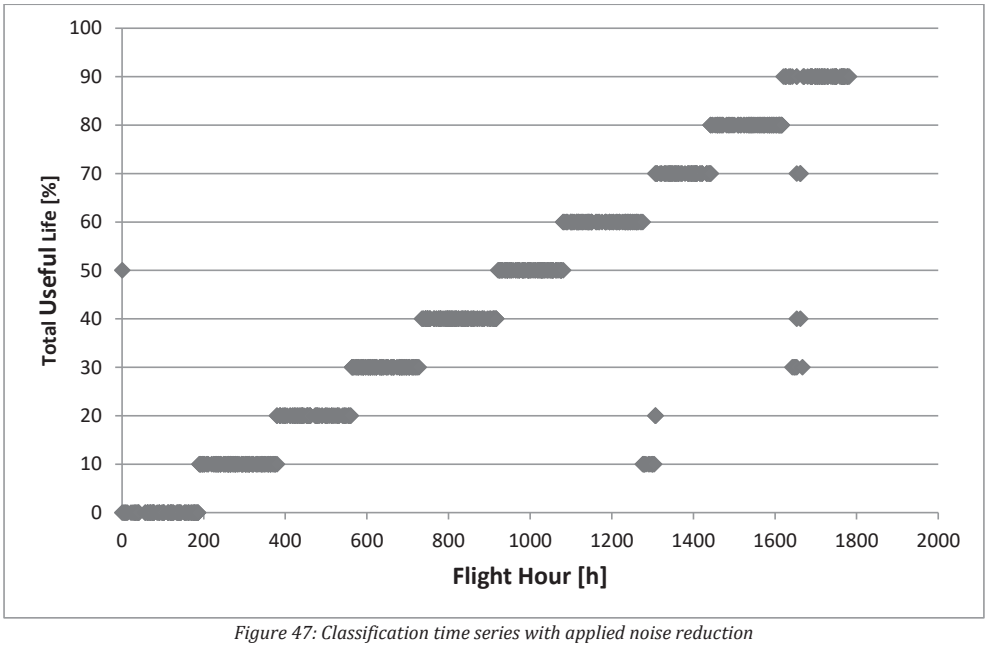

Figure 47: Classification time series with applied noise reduction

Condition prediction uses past condition monitoring data to predict the RUL of the monitored system. The prediction is done by detecting when the system health condition changes from one condition to another and marking these state changes on a FH time axis. Ideally, this plot should be a linear function with the health condition changes equally spacedFigure 100. Maintenance

action can alter the gradient of the function and/or introduce a discontinuity, so the prediction needs to be restarted from the maintenance action data point.

Depending on the usage of the system and the operation environment, the health condition changes may not be equally spaced. This indicates a change in the degradation and, thus, in the RUL. Prediction is possible when two or more state changes have been detected. Because the plot does not have more than 11 data points, it is possible to use a simple approximation method. The classification rules (the rules determining in which RUL class a sample belongs) are automatically generated by the samples used to train the decision tree. The threshold for an RUL of zero is determined by extrapolating the already classified samples (see Figure 48). This means the RUL is based on experience and not on physical models or aircraft regulations (e.g., maximum number of flight hours).

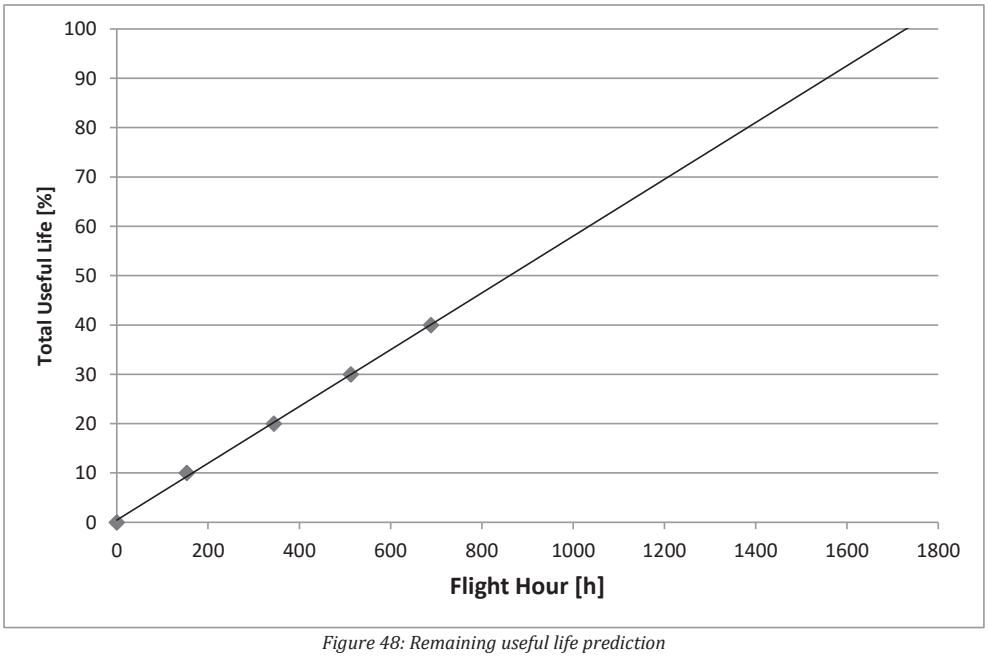

Figure 48: Remaining useful life prediction

4.3.4 Extrapolative Prediction Results

To test the performance of the new process, we created a training data set and a testing data set using the data set from Section 4.3.1. Each set (835 samples) randomly includes 25% of the available data samples (3340 samples). This introduces some noise in the distribution of the samples across different conditions. The process can classify the test set correctly with an error rate of 346 wrong samples from a total of 835 samples or 41%. This large error rate stems from the high noise in the data and the fact that data are not 100% correctly divided into two time series (two total lifetimes). It is difficult to reduce the noise at the source level, because direct data from the aircraft are being used, but it is possible to reduce the noise before the feature extraction step by applying noise reduction. Noise reduction needs to be applied carefully so that no significant features (especially for the Boolean data sources) are removed. For this reason, noise is not reduced for each channel; instead, noise reduction is performed on the results, because without maintenance, the RUL can only increase. The wrong classifications are spread over nine classes; correct classification is possible by taking the most common class over a range of 20 classifications (three flights or 10 FH) in the time series. Note that that the misclassifications are mostly misclassifications to neighbouring classes (seeTable 10) where samples close to the border of a class are wrongly classified as the neighbouring class.

The resulting condition time series is shown in Figure 49. The figure shows the points at which the current system condition switches are nearly equally spaced; even with many misclassifications in the data source, there is little deviation from the correct data points.

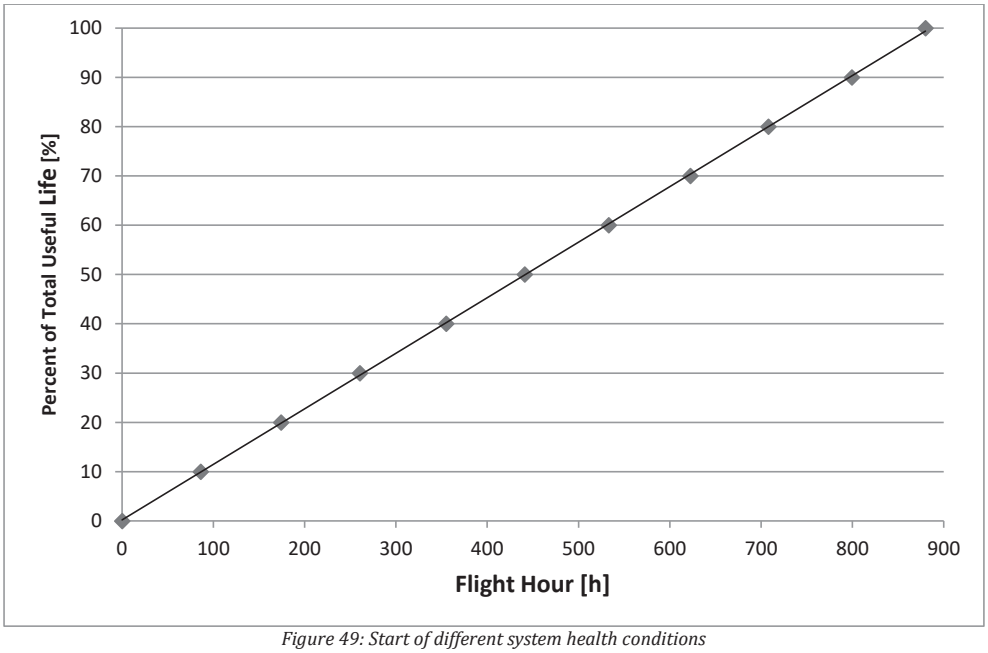

Figure 49: Start of different system health conditions

The results and the calculated "switching points" show that the process can handle the noise very well to make a prognosis. However, a drawback is the "lag" of the detected condition switches of 20 samples. This lag can be compensated with better noise reduction and the application of restrictions on the possible conditions (the system may only degrade unless a maintenance action is done) or a higher sampling rate (smaller data samples or overlapping data samples).

4.4 Summary

Testing showed the proposed concepts facilitate reliable condition monitoring and condition prediction. Condition monitoring was tested on a system developed by Airbus. Condition prediction was tested using both generated data and real-world data from an A320 aircraft. The validation with real-world data showed that the monitored system did not have a clear degradation pattern and was subject to noise. Condition monitoring worked for the real-world data, but prediction did not work as it should. Consequently, the concept had to be slightly reworked to employ a simpler and more noise resistant method of prediction. With the reworked concept, it is possible to reliably predict the remaining useful life of the system more than 400 flight hours in advance.

The concept had to be modified for RUL prognosis using real-world data. The iterative prediction process, as described in this thesis, does not work well with noisy data. Patterns are visible in the data after the fuzzy decision tree evaluation (e.g. the maintenance action is clearly visible and the

curves before and after the maintenance action are similar). The iterative approach does not work, because the curves of the similarity classes are based on a complex non-linear function.

5 CONCLUSIONS

It is possible to monitor and predict the health condition of a system with the proposed concept. The concept combines the use of decision trees, genetic algorithms, signal analysis and approximation methods to create a system that can learn and adapt to various problems. It is easily understood by a human operator and can be applied to different monitoring problems.

The use of a decision tree together with the optimization algorithm during the training of the system monitoring enables the process to use only the signal analysis methods that give the most useful information to solve a given problem. During the system 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 forecasting/prediction are similar. The decision tree and the optimization are used to select different forecasting methods based on experience. The iterative forecasting method can be switched dynamically during the prediction to enable the process to react to events and to handle non-linarites in the observed data. The alternative forecasting method proposed here is based on the fact that the decision tree uses crisp classes; it uses the change of one class to another to predict the remaining useful life. Both approaches have advantages and disadvantages. The iterative approach is well suited for short-term prediction or for data with low noise. The alternative approach needs more data points before the first prediction can be made, but it is much more robust against noise.

Experiments showed that training and using multiple classifiers for the same problem and then taking the class to which the majority belongs, improves the accuracy of the classification. For fuzzy decision tree forest evaluation, this means the similarity values of the other classes need to be averaged over the decision trees selected for the same class.

The processes are mostly autonomous and require little human interaction. Humans are only needed to collect and label data samples. The algorithms can work alone after the data are prepared and the algorithm is trained. Because of 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.

Another interesting result was that the process had to be changed to work with the real-world data from an aircraft. The process developed with lab data was not suitable for the characteristics of the real-world data. Therefore, two methods for forecasting are suggested here. First, an

iterative method calculates each future data point based on a past time series and, second, an extrapolative method uses the switching of one health condition to the next for prediction.

5.1 Research Questions

RQ1: Is it possible to predict a failure of the air conditioning system 400 flight hours before the failure occurs so that the failure can be prevented during scheduled maintenance?

The results show that it is indeed possible to predict failures in advance. The aircraft environment is extremely noisy because of the passengers, turbines, environmental conditions, speed and flight height. Even under these conditions, it is possible to predict the health of the system and when a failure will likely occur. However, the planned iterative approach does not work with the noise. A different method was developed to be more robust to noise. This was unexpected at the start of the research but proved to be an advantage. The modified method for prediction is simpler than the iterative prediction but, at the same time, more robust to noise. This comes at the cost of precision, but the prediction is still accurate enough.

RQ2: Is it possible, with established signal processing and pattern recognition methods, to adequately monitor and predict the health of aircraft fans and filters in real-time?

The proposed method for condition monitoring uses only basic signal processing methods like calculating the mean and maximum amplitude in the time and frequency domains. When used for pattern recognition and prediction, a genetic algorithm and decision trees proved powerful enough to make a reliable health prediction. The validation also showed these methods provide a very fast way to compute a variety of meaningful features without using much computation or electrical power. This combined with the simple decision tree evaluation allows health monitoring (not prediction) using small embedded devices which are not connected to the conventional aircraft health monitoring system to monitor sound and vibration (Internet of Things, IoT). The project also developed a sensor box that could last for six months without battery replacement while recording at a high rate. If the box were further developed, a much longer life span is possible.

RQ3: Is it possible to monitor the aircraft fans and filters using existing sensors or by monitoring only sound and vibration data?

The validation with real-world data showed it is possible to make predictions using only existing sensors and process data. But as mentioned, the prediction method needed to be modified to be more robust against noise. The experimental validation in a test rig showed it was possible to

detect the health condition of the vales, the fan and filter using only a vibration or sound sensor. Thus, it is possible to use a very simple method to retrofit existing parts to provide health information.

5.2 Further Research

Future research could include improving the preprocessing for the condition monitoring, better labelling of samples for condition prediction and using a combination of short and long horizon forecasts for the iterative method. Feature extraction could be improved by using more advanced feature extraction concepts for the time and frequency domain to get more meaningful features. The concepts cannot replace another maintenance method, but the system can be operated in parallel to the currently used one. Sensors collect data and they are correlated with maintenance actions and the system/machine age. Over time, the new maintenance concept can replace the old one.

Another improvement area would be to develop sensoring equipment that can be installed in/at unmonitored devices and connected to the aircraft network. The power supply of these devices could prove to be a problem, but with low power WLAN and clever design, it would be possible to create a small, long lifetime health monitoring device, which can be loaded with a trained decision tree monitoring algorithm.

6 PAPER 1: EFFECTS OF CONDITION-BASED MAINTENANCE ON COSTS CAUSED BY UNSCHEDULED MAINTENANCE OF AIRCRAFT

6.1 Introduction

This section introduces the concepts and specifics of the aircraft environment. It explains the goals of aircraft maintenance and the applicable regulations.

6.1.1 Aircraft Maintenance

Aircraft maintenance is based on Reliability Centred Maintenance. The goal is to have maximum safety and reliability with minimal costs. 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 task or a combination of tasks is selected (Nowlan & Heap, 1978)

Reliability is the probability that an item will perform its intended function for a specified interval under stated conditions (US Department of Defense, 1998).

The Maintenance Steering Group (MSG) was formed to develop maintenance concepts for aircraft. The most recent is MSG-3 (Federal Aviation Administration, 2012). The focus of MSG-3 is the effect of a failure on the aircraft operation (Nowlan & Heap, 1978) (Air Transport Association of America, 2007). For each item that affects the airworthiness, a specific maintenance task is described (task oriented maintenance). MSG-3 can use condition-based maintenance or predetermined maintenance to achieve its goals. Most airlines and manufacturers use predetermined (preventive) maintenance, as it provides both economic and reliability benefits (Kiyak, 2012).

The core concept of MSG-3 is Failure Mode and Effect Analysis (FMEA). With FMEA it is possible to determine which maintenance actions need to be performed during planned maintenance. This includes taking the probability and effects of a failure into account and planning the maintenance during system development. MEA uses a top-down approach, with analysis starting at the highest system level. A lot of detailed analysis is not needed, because most maintenance tasks are found at higher levels.

The FMEA process has the following steps (Society of Automotive Engineers, 2001):

Identify Relevant Functions: In this step, all functions of a system are identified. See Table 11 for an example of a function.

Identify Functional Failures: This step defines failures of a function. A function can have multiple failure modes. See Table 11 for an example.

Identify Failure Effects: The failure is classified using the process in Table 11.

Identify Failure Probability: The probability of a failure is calculated based on experience or in-service data.

Select Maintenance Tasks: It is possible to define maintenance actions to prevent a failure when the causes of a failure are defined. This step also includes determining the maintenance intervals, combining maintenance tasks, and removing duplicate ones.

Function	Functional Failure	Failure Mode
Provide redundant capability of informing crew of fire in each of the four specific areas (right hand Fan, left hand Fan, Core upper case, Core lower case).	Loss of redundancy to detect fire in the designated engine fire zone.	Engine fire detector failure.
	Provides false fire warning indication.	Engine fire detector failure.
Alerts crew of detection loop failure.	Does not alert crew detection loop failure.	Engine fire detector failure.
		MAU Failure.

Table 11: Example of functional failure analysis - engine fire detection system (European Aviation Safety Agency, 2005)

6.1.1.1 Failure Classes

Failures are divided into five classes depending on the effect of the failure on the aircraft. A criterion for the classification is the severity of the failure for aircraft safety. Table 12 shows how failures are classified.

<table>
<tr><td colspan="5">Is the occurrence of a functional failure evident to the operating crew during the performance of normal duties?</td></tr>
<tr><td colspan="3">Yes</td><td colspan="2">No</td></tr>
<tr><td colspan="3">Does the functional failure or secondary damage resulting from the functional failure have a direct adverse effect on operating safety?</td><td colspan="2">Does the combination of a hidden functional failure and one additional failure of a system related or backup function have an adverse effect on operating safety?</td></tr>
<tr><td>Yes</td><td colspan="2">No</td><td>Yes</td><td>No</td></tr>
<tr><td></td><td colspan="2">Does the functional failure have a direct adverse effect on operating capability?</td><td></td><td></td></tr>
<tr><td></td><td>Yes</td><td>No</td><td></td><td></td></tr>
<tr><td>Safety</td><td>Operational</td><td>Economic</td><td>Safety</td><td>Non-Safety</td></tr>
<tr><td colspan="3">Evident</td><td colspan="2">Hidden</td></tr>
</table>

Table 12: Failure class criteria

This results in the following failure classes (Air Transport Association of America, 2007):

Evident Safety: This must be approached with the understanding that a task is required to assure safe operation. If this is not the case, a re-design is required.

Evident Operational: A task is desirable if it reduces the risk of failure to an acceptable level.

Evident Economic: A task is desirable if the cost of the task is less than the cost of repair.

Hidden Safety: A task is required to assure the availability necessary to avoid the adverse effect on safety of multiple failures. If this is not the case, a redesign is required.

Hidden Non-Safety: A task may be desirable to assure the availability necessary to avoid the economic effects of multiple failures.

6.1.1.2 Failure Probability

Ideally, in-service data are used to evaluate the risk of a failure. However, during development, no in-service data are generally available. During development, assumptions need to be made based on similar parts, tests, simulations or experience. Later, when in-service data are available, they can be used to update the failure probability.

Failure class and failure probability define the criticality of the failure. The criticality is used to plan the maintenance action.

6.1.2 Scheduled Maintenance

Periodic maintenance actions are organised in five different classes of checks. Each check is performed at a different interval and gets more complex with the size of the interval. The given intervals can vary depending on the aircraft type and aircraft operation (Air Transport Association of America, 2007).

6.1.2.1 Pre-/Post Flight Check

The most performed maintenance check is the pre-/post flight check, and it is done daily. This check is often done by the pilot by walking around the aircraft and checking the general state of the aircraft.

6.1.2.2 A-Check

A-checks can be performed overnight in a hangar and are done every two months. During an A-check, all technical systems required for aircraft operation are checked.

6.1.2.3 C-Check

The C-check is a major aircraft check, where the aircraft is taken out of operation to be inspected. C-checks occur every two years and take about two weeks. The aircraft structure is inspected and all systems are tested.

6.1.2.4 IL-Check

The IL check is done every four years and includes the detailed checking and maintenance of systems and structure.

6.1.2.5 D-Check

This check is done every ten years and takes about one month. During this check, nearly the whole aircraft is dissembled and checked. Sometimes the paint is even removed to check the structure. An aircraft has two to three D-checks during its lifetime.

6.1.3 Maintenance Program Development

The process to develop a maintenance plan for scheduled maintenance based on the MSG-3 method is complex. An Industry Steering Committee (ISC) consisting of authorities, aircraft operators and the manufacturer is created. These actors, in turn, form groups (MSG Working Groups (MWGs)) which frequently meet and decide on the frequency and scope of needed maintenance actions (see Figure 50). First, the MSG-3 analysis is performed based on aircraft data. Then, a Maintenance Review Board Report (MRBR) proposal is created and accepted. The MRBR contains the minimum scheduled tasking/interval requirements for a newly FAA type-certificated

(TC) or derivative aircraft and its aircraft engines. The accepted MRBR is used by the manufacturer to create the Maintenance Planning Document (MPD) (Federal Aviation Administration, 2012) (Federal Aviation Administration, 1994) (European Aviation Safety Agency, 2008).

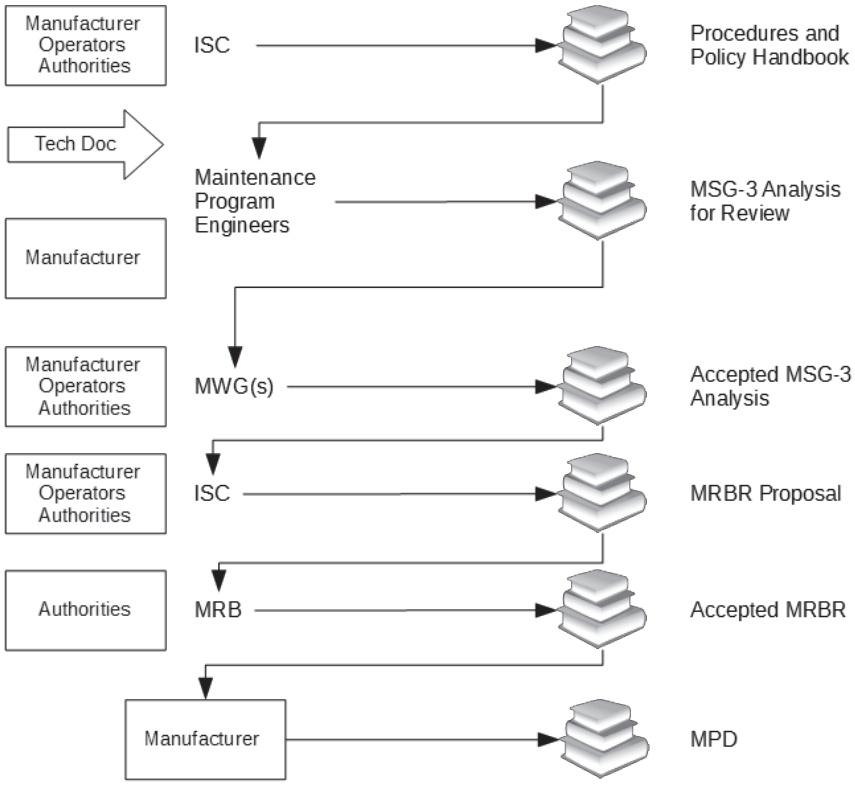

Figure 50: MRBR process

Revisions requiring formal approval are subject to the same consideration as initial approval (Federal Aviation Administration, 1978). Changing the maintenance plan is a difficult process, and there must be good reasons to change the maintenance intervals. European Aviation Safety Agency (2008) has defined a process to update maintenance intervals. This process is needed because initially no in-service data for a new aircraft are known, and the maintenance intervals are created based on estimations. Operator in-service data are needed to adapt the maintenance intervals.

Ali and McLoughlin (2012) show how much can be saved by optimizing the maintenance intervals.

6.1.4 Unscheduled maintenance

Unscheduled maintenance is done outside the defined periodic intervals because an unexpected failure has occurred. The aircraft continues to fly safely because of 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 turnaround time, the flight will be delayed until the fault is eliminated. Depending on the failure, the aircraft may have to stay on the ground until the failure is fixed. The decision to stay on the ground (AoG - Aircraft on Ground) depends on the Minimum Equipment List (MEL) (International Civil Aviation Organization, 2015) (Civil Aviation Regulations Directorate, 2006). The MEL is based on the Master Minimum Equipment List (MMEL) (International Civil Aviation Organization, 2015) (Civil Aviation Regulations Directorate, 2006), accepted by national airworthiness authorities. The MEL is an operator defined list that is stricter than the MMEL. If a faulty part is listed in the MEL, the aircraft is not allowed to operate until the failure is fixed.

Depending on the flight schedule of the aircraft, departure may be delayed because of the maintenance operation. Instead of a delay, the flight may need to be completely cancelled. Delays and cancellations are very expensive for an airline (Cook, et al., 2004) and should be avoided if possible.

6.1.5 Preventive Maintenance

Preventive maintenance (PM) is the standard method for reducing unscheduled maintenance. Aircraft components are inspected at given time intervals, depending on the component type and varying from airline to airline. Reducing the time interval can increase the need for spare parts, as increasing the interval increases the risk of unscheduled maintenance (Kolerus & Wassermann, 2011). Preventive maintenance can be divided into the following three types (Air Transport Association of America, 2007) (Nowlan & Heap, 1978) (Civil Aviation Authority, 1995) (Federal Aviation Administration, 1978):

Hard-Time (HT): Scheduled removal of a component before some specified maximum 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.

No Maintenance: This approach assumes the component can be used until it breaks, at which point it is replaced. This is not preventive maintenance but corrective (reactive) maintenance; nonetheless, it is used for certain components in aircraft maintenance.

6.1.6 Condition-Based Maintenance

Condition-based maintenance (CBM) is based on condition monitoring and aims at preforming maintenance based on the system condition and trend of the system condition. CBM can be used to realize reliability centred maintenance (RCM) (Niu & Pecht, 2009).

Condition monitoring constantly measures and analyses relevant parameters of mechanical and electrical components during operation. The parameters selected for monitoring allow determination of the condition and failure state. The need for maintenance of the component is only indicated if parameters show a predefined degradation (Kolerus & Wassermann, 2011).

The difference between CBM and preventive on-condition (OC) maintenance is that OC checks a system at defined intervals while condition monitoring continuously monitors the condition.

Condition monitoring is used in a wide field of applications, including rotary machines (gear boxes, gas and wind turbines, bearings etc. (Mahamad, et al., 2010) (Saravanan & Ramachandran, 2009) (Sugumaran & Ramachandran, 2011) (Tian & Zuo, 2010) (Zhao, et al., 2009), plants and structures (bridges, pipelines etc. (Goode, et al., 2000)). Often vibration data are used to perform the condition monitoring (Ebersbach & Peng, 2008).

The condition of the system is defined by setting limits on certain values based on experience (Mobley, 2002) or based on a mathematical or data-driven model (Kolerus & Wassermann, 2011) (Williams, et al., 1994). Machine learning techniques, such as decision trees (Sugumaran & Ramachandran, 2007) (Sugumaran & Ramachandran, 2011) (Tran, et al., 2009), vector support machines (Pham, et al., 2012) (Sugumaran, et al., 2007) (Widodo & Yang, 2007) and neural networks (Chen, et al., 2012) (Mahamad, et al., 2010) (Tian, 2012), are often used to map the features of the input signal to a condition.

Another option is to use a mathematical model, feed the sensor input into the model, calculate the output and check whether the output of the theoretical model deviates from the real system. This approach can also be used for fault isolation, identification of failures and prognosis (Wang, et al., 2008) (Williams, et al., 1994) (Kolerus & Wassermann, 2011) (Jardine, et al., 2006).

Data-driven models use past data to create models with stochastic or machine learning algorithms (Pecht, 2008) (Garcia, et al., 2006) (Jardine, et al., 2006). These models require many data samples that represent different conditions of the system. Data-driven models require less human intervention than mathematical models; model validation and testing can be performed almost automatically.

Trend analysis is a method to achieve CBM. The analysis algorithm doesn't just look at recorded parameters at a single moment in time; it also takes the full parameter history into account. The need for maintenance of a 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 a forecast of the remaining lifetime of the component (Kolerus & Wassermann, 2011). There are several different methods to predict future values. ARMA, ARIMA, artificial neural-networks, sequential Monte Carlo and Markov models are used for prediction for a complex time series (Chen, et al., 2011) (Caesarendra, et al., 2010) (Pham & Yang, 2010) (Tian, et al., 2010). Output of the prediction is normally an estimated time to failure (ETTF) and a confidence interval (Sikorska, et al., 2011). The confidence interval defines how reliable a prediction is (Schruben, 1983) (Sikorska, et al., 2011). It can be calculated using standard time series.

Implementing CBM is a difficult and costly task. Many barriers prevent the use of CBM on all systems. These barriers include (among others) (Stecki, et al., 2014):

- The inability to predict accurately and reliably the remaining useful life of a machine (*prognostics)*;
- The inability to continually monitor a machine *(sensing)*;
- The inability of maintenance systems to learn and identify impending failures and recommend what action should be taken (*reasoning*);
- The initiation of CBM programs without full knowledge of how the system can fail;
- The general focus of CBM research on specific techniques (better mousetrap symptom).

6.2 Maintenance Costs

Hard and soft costs are very common in maintenance. When defining a model of costs, it is necessary to select those that are easily measurable, from which soft indicators, representing an intangible aspect of a much more complex measurement, can be extracted. However, soft indicators, such as the cost of not having carried out training or the non-availability of condition monitoring equipment that could have detected an anomalous vibration, are not measurable using

a traditional collection of data. Therefore, it is necessary to look for more easily seen hard costs with the required information.

In the first step, the objectives of the system must be properly defined. The costs of a system's maintenance cannot be modelled when the inherent objectives of its design or the operational objectives for which it has been acquired are not known. Thus, for example, in the case of a spare or redundant centrifugal pump, the objective of the system is its condition of redundancy; its maintenance costs will be entirely different from an identical pump used in an area of high criticality due to the vastly different operating conditions.

Once the objectives have been identified, the next step is to select the equipment, if this it has not been already done, and to identify the alternate systems in which it is used. Finally, the optimal configuration for each system is determined, using a method of economic evaluation.

The criteria of evaluation at the time of selecting equipment must balance aspects of both life cycle cost and operational effectiveness. These criteria are shown in Figure 51.

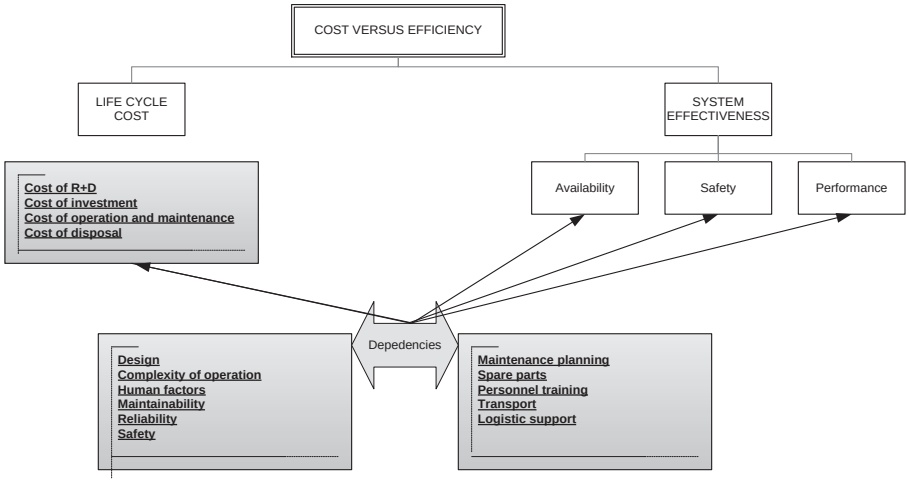

Figure 51: Effectiveness-cost relationship

A priori, it is less difficult to establish the cost criteria than the criteria of effectiveness using the manufacturer's data or drawing on similar experiences in similar equipment. This does not mean that the estimation of the cost is simple; it only means that the various classifications are normally better understood.

Maintenance costs are an inherent part of life cycle cost (LCC) and cannot be dissociated from the LCC concept. The cost of the maintainability of a system cannot be assessed if it is not considered from its conceptual design stage until its disposal. Therefore, a model of costs must cover the system's entire life cycle; it must include costs associated with research and development, engineering, design, production, operation, maintenance and disposal, as shown on the left side of Figure 51.

6.2.1 Estimation of Maintenance Global Cost

As explained by Komonen and Akatemia (Komonen & Akatemia, 1998), the costs of maintenance can be divided into two groups:

- Costs that appear in the operation of maintenance (administrative costs, cost of manpower, cost of material, cost of sub-contracting, cost of storage, cost of capital).
- Costs of loss of production due to shutdowns of production assets or reductions in the production rate, and loss of quality in the product due to equipment malfunctions.

This classification emphasizes the two main objectives of the maintenance function, both corresponding to the desired balance of effectiveness and efficiency:

- High availability of production assets;
- Low maintenance costs.

The global cost of maintenance C_g is the sum of four components (AFNOR, 1994):

- Cost of interventions (C_i);
- Cost of failures (C_f);
- Cost of storage (C_s);
- Cost of over-investment (C_{oi}).

$$C_g = C_i + C_f + C_s + C_{oi} \tag{36}$$

The objectives of all facets of the maintenance organization must be aligned so that an attempt to reduce one factor of the global cost will not produce an increase in another. The global cost can be calculated for a specific machine, group of machines or whole plants, but policies like RCM that rationalize the observance and application of maintenance only require the calculation of the costs of that equipment whose criticality or economic relevance affects the overall performance of the

entire system under consideration. For this reason, the equipment that most affects the global cost will receive more attention and be the subject of more detailed cost analysis.

There will be a problem if there is a mismatch between data and information, for example, overwhelming amounts of data yielding too little information. For this reason, the data contributed by a cost model to the set of financial indicators will only come from critical equipment or equipment consuming a high percentage of the allocated money (i.e. relevant in the maintenance budget).

6.2.2 Downtime Cost and Failure Cost

These costs correspond to the loss of profit because of a maintenance problem that has reduced the production rate. They are the result of the following:

- Preventive maintenance badly defined;
- Preventive maintenance badly executed;
- Corrective maintenance performed over an overly long period, badly executed, using bad or low quality spare parts.

It is important to highlight that the cost of failure of the asset corresponds to the loss of profit margin caused by a defect that brings unacceptable losses of production. The dilemma is whether the cost is attributable to the reasons cited above or to the following:

- Errors of use (misuse) that imply degradation of the asset;
- Environmental conditions outside normal working conditions specified by the asset manufacturer.

Such costs must be charged to the production, purchases or even engineering functions, not to maintenance. By slotting the costs of failure into the various functional areas, not just into maintenance, those responsible in each area can take corrective measures and, in some cases, assume full responsibility for the expenses. A single policy which puts all shutdowns and costs of failures under maintenance whatever the reason and no matter who is responsible should not be adopted.

For example, a maintenance failure should not be mistaken with a machine failure caused by buying unreliable equipment. Decisions to purchase or re-engineer equipment almost never depend on the maintenance department but are driven by productivity criteria, making it absurd to transfer that cost to maintenance.

Consider an organization with an engineering department which deploys projects or makes productive improvements using assets of low reliability, maintainability, safety, etc. without consulting or considering maintenance in any phases of any of the projects. The maintenance department is not involved at all in the decisions and therefore is not responsible for any of the resulting problems. However, as the general perception is that these costs should be attributed to maintenance, there is friction between departments about the imputation of the failure costs, even though these obviously result from making poor decisions at the outset.

6.2.3 Evaluation of the failure cost

The failure cost, C_f, can be calculated using the following formula:

C_f = unperceived income + extra production expenses - raw material not used.

The components of this cost are:

- Unperceived income: This factor will depend on the possibility of recovering lost production by rescheduling, working weekends etc. In cases of continuous production, however, there is no chance to recover; therefore, the production of that time slot and all incomes which could have been generated during the shutdown must necessarily be imputed to this first part of the equation.
- Extra production expenses: If it is possible to recover part of the production in other temporary slots, the following additional costs will be incurred:
    - Energy required for production;
    - Raw materials;
    - Expendable materials;
    - Services related to quality, purchases, maintenance, etc.
- Unused raw material: When it is not possible to recover production, the cost of the unused raw material will be subtracted from the failure cost. Although the raw material has not been used (unless it is a perishable product that must be thrown out), it will be consumed if the productive plan is recovered, possibly with some extra storage costs, transport cost or costs related to the degradation of the materials.

The most popular model used to calculate of the cost of failure when there are productive assets that totally or partially assume the tasks of the assets under maintenance is the Vorster method. It is used to calculate the maintenance costs and other financial indicators.

6.3 Delays in Aviation

Delays are incurred when an aircraft is prevented from leaving for its destination for an interval of 15 minutes or more (Federal Aviation Administration, 1995). Delays can originate from traffic, passengers, weather or aircraft. A delay causes additional operating costs, including crew-related, ramp-related, aircraft-related and passenger-related (hotel and meals, re-booking and re-routing, luggage complaints and revenue losses) costs (Poubeau, 2002). Delay costs can be calculated from Scholz (1995), Scholz (1998) or Cook et al. (2004).

The average departure delay in 2014 for European air traffic was 26 minutes (Eurocontrol, 2015). Most flights were not delayed, but delays longer than 30 minutes occurred in 8% of all departures (Eurocontrol, 2015) (US Department of Transportation, 2014). Flight cancellations represented between 1% and 2% of all scheduled departures (Eurocontrol, 2015) (US Department of Transportation, 2014).

6.3.1 Delay Causes

Delays can be reactionary or caused by the airline. Reactionary delays are caused by late arrival of aircraft, crew, passengers or load (Eurocontrol, 2015). These delays cannot be controlled because an external source causes them. Airline delays include aircraft and ramp handling, technical reasons, flight operation and crewing or passenger and baggage handling (Eurocontrol, 2015). Figure 52 shows the distribution of the delays in minutes per flight. Note that ATFCM stands for "Air Traffic Flow and Capacity Management"; government reasons for delay include security and immigration (Eurocontrol, 2015).

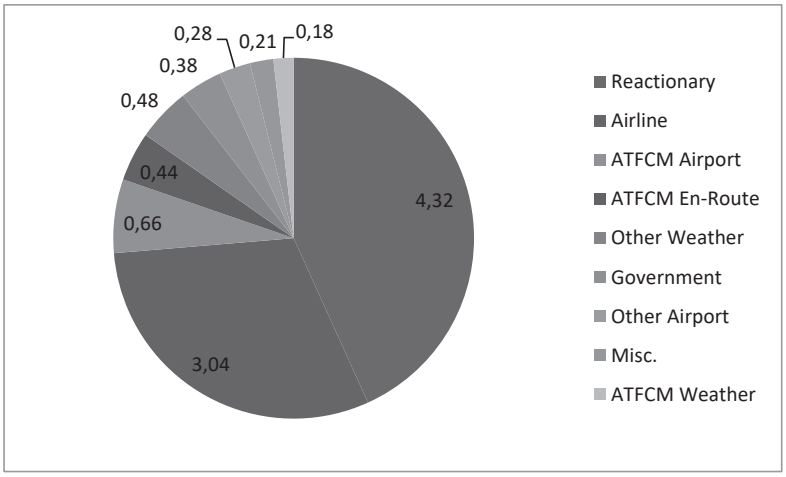

Figure 52: Causes of departure delays in 2014 in Europe (Eurocontrol, 2015)

Airline delays include delays caused by unscheduled maintenance. It is difficult to find data sources that show what percentage of the airline-caused delays are maintenance delays or which system caused the problem. However, Knotts (1999) shows that about 20% of the delays for the Boeing 747 are caused by technical problems, and the Civil Aviation Authority (2009) shows how aircraft maintenance is distributed over the aircraft systems. This gives an indication of the systems likely to cause maintenance delays.

A large problem is delays caused by No Fault Found (NFF) problems; these can represent a significant part of the unscheduled maintenance actions (International Air Transport Association, 2015). The NFF rate for the Generator Control Unit is about 71%.

Airline maintenance policy also affects the number and significance of delays. Rupp et al. (2006) explain how an airline policy influences the number of flight cancellations. Flights with fuller aircraft or on routes with higher competition are cancelled less often. It can be assumed that a similar effect can be observed for "normal" delays, because repairs can be deferred as per the Minimum Equipment List (MEL). Sachon and Patè-Cornell (2000) analyse how an airline maintenance strategy affects delays, cancellations and safety using a probabilistic risk analysis. The model shows a marginal trade-off between minimizing delays and maximizing safety for the Leading-Edge slat system of an aircraft. Models like this can help to adapt the maintenance policy to reduce delays at the management level; this includes qualification of maintenance personnel, timing of maintenance operations, number of deferrals allowed etc.

6.3.2 Costs of Delays

The costs of an aircraft delay can be calculated several different ways (Cook, et al., 2004). Cook et al. (2004) provide a method for calculating delay costs and their impact. One example is the calculation of the costs for a "network reactionary delay". The network effect is the effect of consecutive delays caused either by the aircraft incurring the initial delay or by other aircraft. These costs and effects can have a large influence on the overall number of delays (see Figure 54). The specific cost elements of a delay are:

- Fuel burn costs plus commentary on airborne delay
- Maintenance costs
- Flight and cabin crew salaries and expenses
- Handling agent penalties
- Airport charges
- Costs of passenger delay to airlines

The average delay costs (without network effects) are about 47 US$/min. With network effects, this value increases to 78 US$/min.

Cook and Tanner (2011) (see also Cook et al. 2004) divide the costs of a delay into strategic, tactical and reactionary costs. Strategic costs are accounted for in advance, e.g. buffer costs. Tactical costs are incurred on the day of operations and not accounted for in advance. Reactionary costs are caused by network effects. In Europe, for each minute of a primary delay, another 0.8 minutes of reactionary delay are caused. These values can vary significantly from airline to airline. Ferguson et al. (2013) show that the data and method from Cook et al. (2004) can be applied to the US airline industry.

Maintenance, repair and overhaul (MRO) companies want to reduce their costs to provide low cost maintenance services to airlines (Wagner & Fricke, 2006). For this, they need to know how many unscheduled maintenance events can be expected and how many man hours (MH) they need to be able to handle. Wagner and Fricke (2006) propose a method to estimate the number of needed MHs to handle unscheduled maintenance for a given fleet.

6.4 Unscheduled Maintenance Causes

Unscheduled maintenance is caused by equipment that shows an unexpected fault during flight. This fault needs to be fixed before the next scheduled flight as per MEL. If it is not possible to fix the equipment during turn-around time (due to a long repair time, missing spare parts or difficult failure identification), the flight will be delayed until the fault is fixed. In extreme cases, flights could be cancelled and passengers may have to be redirected or compensated. Figure 533 shows a typical sequence of events leading to unscheduled maintenance and delay (Sachon & Pate-Cornell, 2000).

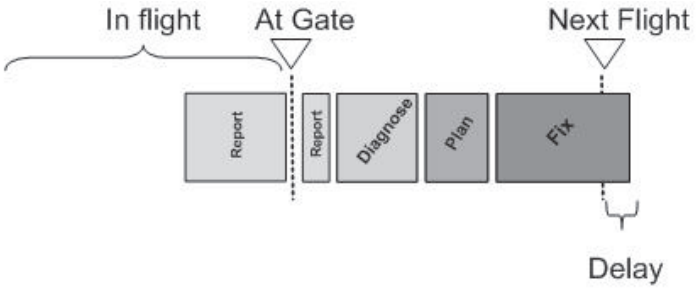

Figure 53: Sequence of events during unscheduled maintenance leading to delays (Sachon & Patè-Cornell 2000)

Mechanical/technical problems are not the only reason for unscheduled maintenance. Humans are also an error source. Failure symptoms are misinterpreted or not noticed or new failures are introduced during scheduled maintenance (Civil Aviation Authority, 2009). This may have an effect on the safety of the aircraft (Sachon & Pate-Cornell, 2000) or lead to an increased number of NFF (International Air Transport Association, 2015). Incorrect or incomplete maintenance comprises up to 61% of all maintenance actions. Most cases of incorrect or incomplete maintenance occur because parts are not correctly fitted or not set correctly (Civil Aviation Authority, 2009).

6.5 Aircraft System and Database

The Airbus A340-600 was selected for this study because it is quite new (entry into service in 2002) but there is already enough experience and data. By November 2008, 84 of the Airbus A340-600 aircraft were in service (Airbus SAS, 2008). The focus of the paper is on the air conditioning system (ATA (Air Transport Association) Chapter 21). The air conditioning system is suitable for analysis because it is flight critical, monitored (auxiliary power unit (APU), fans etc.) and consists of a combination of mechanical and electrical components. Reliable information on it is available in the database of the Airbus In-Service Report (ISR) (Airbus SAS, 2008).

6.6 Empirical Study: AC System of A340

6.6.1 Delay Analysis

The delay analysis is based on in-service data from Airbus (2008). The analysis is restricted to the air conditioning system to reduce the number of entries to check and to focus on the effects of one system on delays. Note that we did not consider flight cancellations. The delays caused by faults of the air conditioning represent about 6 % (Airbus SAS, 2008) of the total delays caused by unscheduled maintenance. Figure 544 shows the distribution of the lengths of delays caused by the air conditioning of an A340-600.

The delay costs caused by the air conditioning system are calculated to show the importance of reducing delays. Delay costs caused by the air conditioning system are calculated based on Cook et al. (2004) with updated economic data from Eurocontrol (2006). The costs of a delay are assumed to be a linear function of delay time. The base value for delay costs is given in US$/min. The average delay costs (without network effects) are about 47 US$/min. With network effects, this value increases to 78 US$/min. Based on Airbus SAS (2008), it is possible to calculate an average delay time caused by the air conditioning system of 90minutes. Multiplying this value by the average delay costs (47 US$/min) yields delay costs of 4230 US$ for a 90-minute delay. In 50

% of the studied cases, however, the delay is less than 50 minutes (see Figure 55). In these cases, the average delay costs are below 2350 US$. The total cost for 100 delays is about 432,990 US$. These costs are quite substantial, so efforts to reduce delays are welcome.

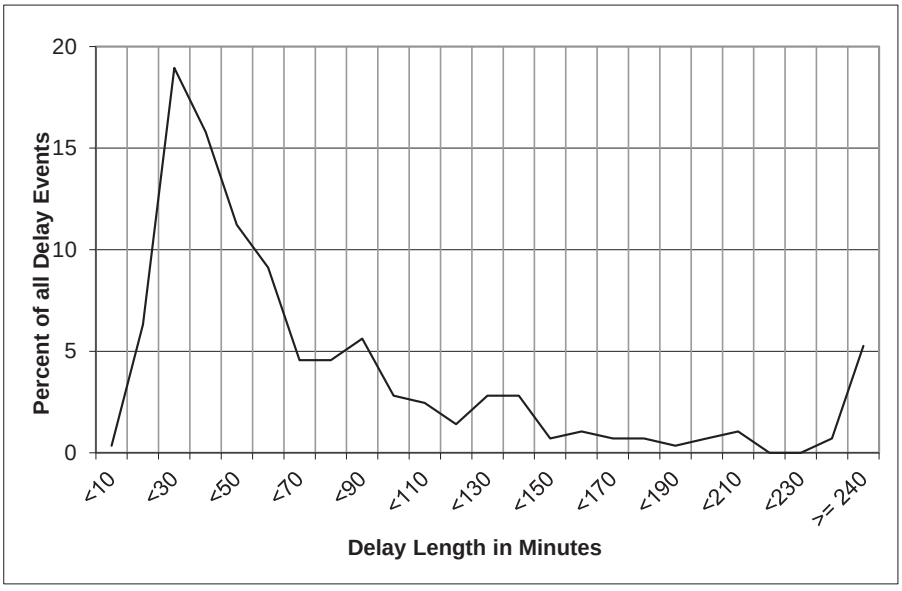

Figure 54: Delay length distribution (Airbus SAS 2008a)

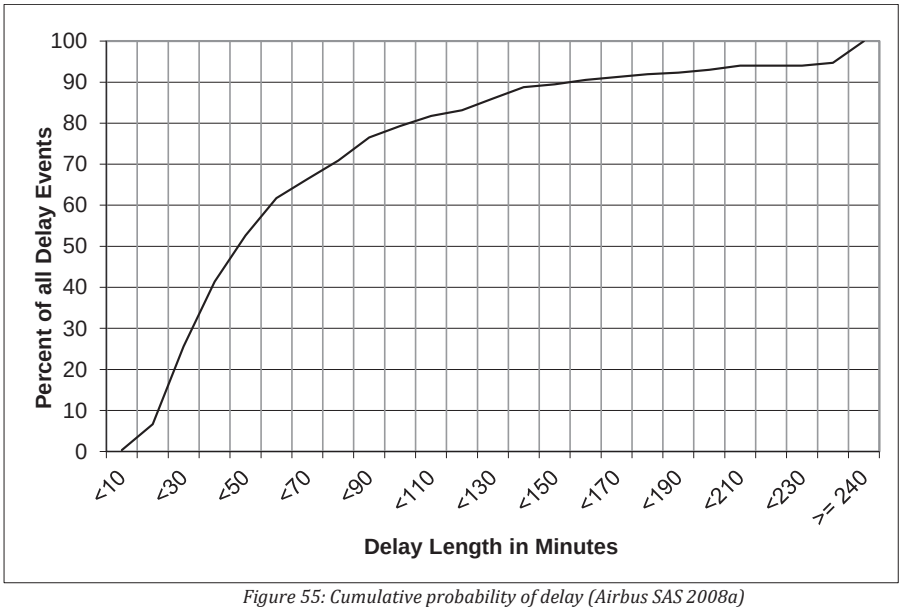

Figure 55: Cumulative probability of delay (Airbus SAS 2008a)

6.6.2 Integrating Condition-Based Maintenance into Preventive Maintenance

Several methods have been suggested to avoid unscheduled maintenance. In preventive maintenance, components are replaced after a given period or at scheduled intervals; in this method, maintenance is scheduled hopefully before the component shows a fault. Condition monitoring and trend analysis use dynamic intervals.

The default aircraft maintenance strategy is preventive maintenance (Muchiri, 2012). Preventive maintenance is well established and understood in the aviation industry. The maintenance intervals for systems are constantly updated and optimized by in-service data (Ahmadi, et al., 2010).

Condition-based maintenance uses variable maintenance intervals for maintenance planning based on the system condition and the condition trend, if available. Condition-based maintenance is difficult to implement in the aviation industry because it can cause more maintenance actions if wrongly implemented (Shin & Jun, 2015) (Sondalini, 2015), and this will cause more delays. The goal is not to increase the number of maintenance actions but to reduce them, as the following explains:

"For example, stadium lights burn out within a narrow period. If 10 percent of the lights have burned out, it may be accurately assumed that the rest will fail soon and should, most effectively, be replaced as a group rather than individually." (Mobley, 2002)

However, it is possible to use condition-based maintenance (CBM) to complement preventive maintenance (PM). Preventive maintenance would still be the major maintenance strategy, but condition-based maintenance can be used to plan maintenance actions required outside the maintenance intervals of preventive maintenance, for example, if a system is under stronger stress than planned. Reasons for this unplanned stress can be that an aircraft is mainly used in difficult environments or it has been exposed to other natural effects like heavy weather. Condition-based maintenance can also help to detect incorrect or incomplete scheduled maintenance. Muchiri (2012) analyses the two different strategies and finds that the difference between PM and CBM gets smaller for older aircraft because PM for older aircraft uses CBM principles.

Condition-based maintenance actions are performed to prevent a failure before it occurs. This allows the aircraft operator to place the maintenance action outside the regular flight traffic and avoid unscheduled maintenance.

Regular maintenance intervals are unchanged to ensure airworthiness and to perform preventive maintenance actions.

Hölzel et al. (2014) propose a method to use CBM to optimize scheduled maintenance planning and explain the benefits.

6.6.3 Strategy of Overinvestment

When designing a product, it is wise to select production equipment that minimizes the global cost, C_g, of maintenance during its service life. This equipment will require a higher initial investment to fulfil the same productivity requirements as cheaper equipment, but the costs of maintenance intervention and spare parts storage will be lower. As given previously, in Equation 36, the global cost can be expressed as:

$$C_g = C_i + C_f + C_s + C_{oi} \tag{37}$$

where C_i is the cost of investment, C_f is the cost of failures, C_s is the cost of storage and C_{oi} is the cost of over-investment.

To include over-investment in a global cost analysis, the initial price difference is amortized over the life of the equipment, making it possible to determine the extra investments required to minimize the other components of the cost.

A common problem in financial models of maintenance systems is that the original costs have been modified several times in successive applications of methodologies or technologies trying to reduce the global cost using what are called avoided costs.

In the indiscriminate implementation of policies of reduction of costs, three of the four parameters that constitute the global cost are affected:

- Costs of interventions (C_i): Normally these are reduced in frequency and in volume; most of the predictive technologies lead to less aggressive failures, with a reduction of corrective maintenance and an increase of preventive maintenance.
- Costs of failures (C_f): These costs are reduced in determined predictive policies where complete overhauls are replaced by small inspections performed without shutting down the process.
- Cost of over-sized investments (C_{oi}): Expensive equipment and planned inspections are the most noticeable items in this cost because the budget is increased.

The following equation shows the impact of the costs avoided with their double dimension, i.e. when a technology or concrete methodology implies an investment. The cost of intervention and failure will be reduced but the cost of over-investment will increase if the technique is not applicable to the company and does not result in a return on investment.

$$C_g = C_i + C_f + C_a + C_{si} + C_{av} \tag{38}$$

$$C_g = \left(C_i - C_{av\_i}\right) + \left(C_f - C_{av\_f}\right) + C_a + \left(C_{si} + C_{av\_si}\right) \tag{39}$$

6.6.4 Reduction of Delays and Costs using CBM

The study assumes that not all delays are caused by the air conditioning system (e.g. oil smell which originates from the engines); in addition, not all can be fixed by a reset action and some are not preventable. All other failures are preventable in theory (e.g. valve failures, leakages, fan faults, regulation faults etc.), by using condition monitoring and additional sensors (e.g. vibration monitoring). Data in the ISR database (Airbus SAS, 2008) show that about 80 % of all failures are preventable (see Figure 57).

When preventable failures are, in fact, prevented, this gives a new average delay of 40 minutes, a reduction of 50 minutes in the average delay shown in Figure 58, a 56 % improvement over the original A340-600 value of 90 minutes. This also means a reduction of the average costs of delays caused by the air conditioning system by 2350 US$ or 56 %. The total costs for 100 delays are reduced by 382,250 US$ to 41,740 US$ or by 90 %. Figure 58 and Figure 59 show the distribution and probability of the preventable delays.

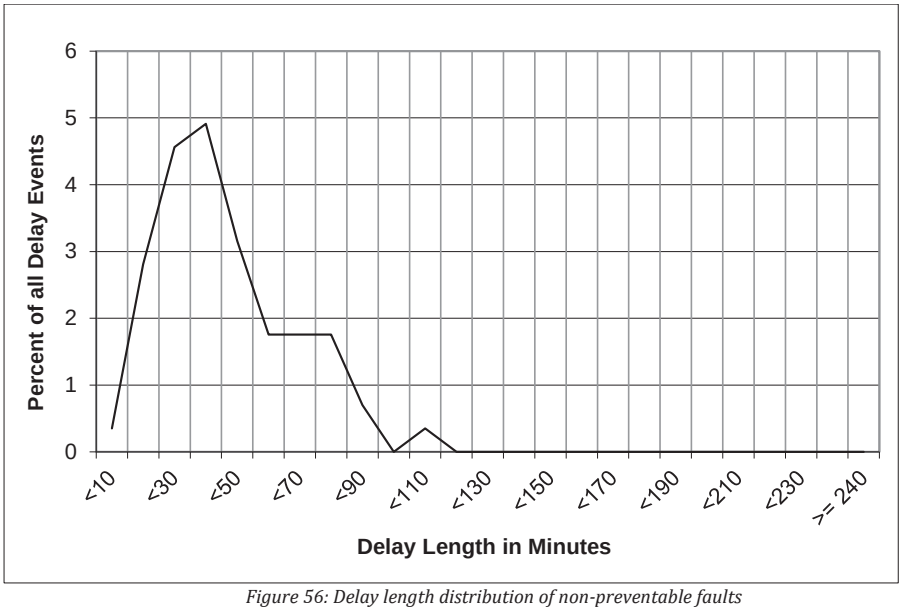

Figure 56: Delay length distribution of non-preventable faults

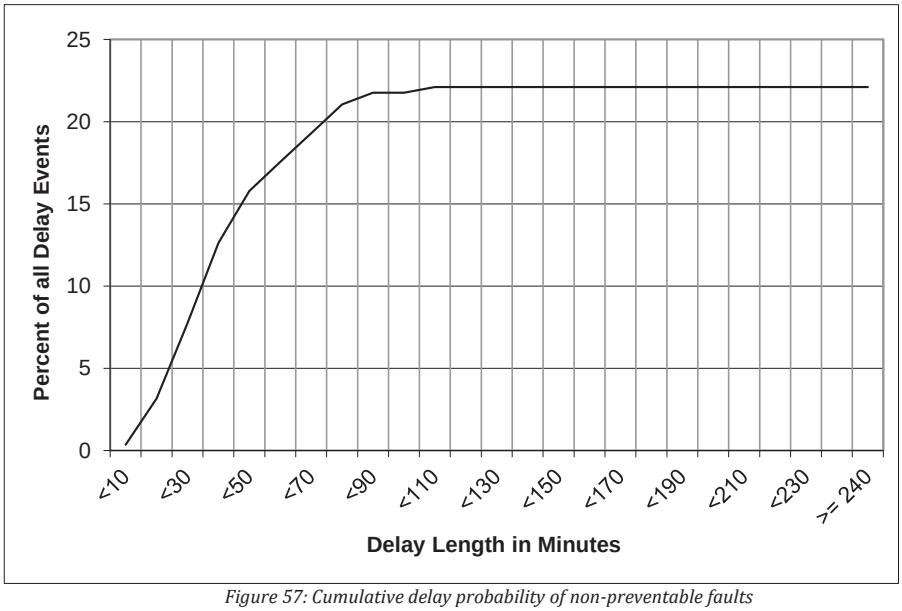

Figure 57: Cumulative delay probability of non-preventable faults

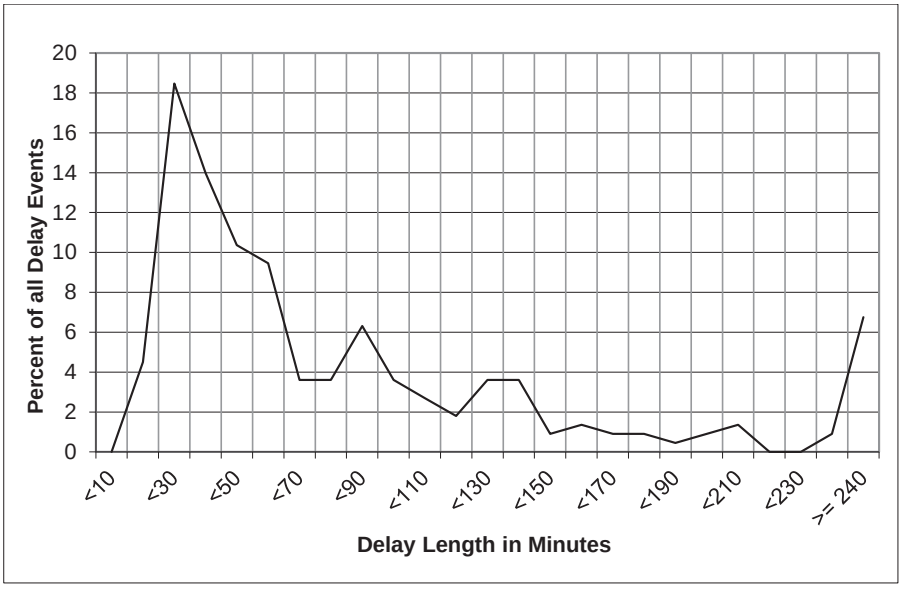

Figure 58: Delay length distribution of preventable faults

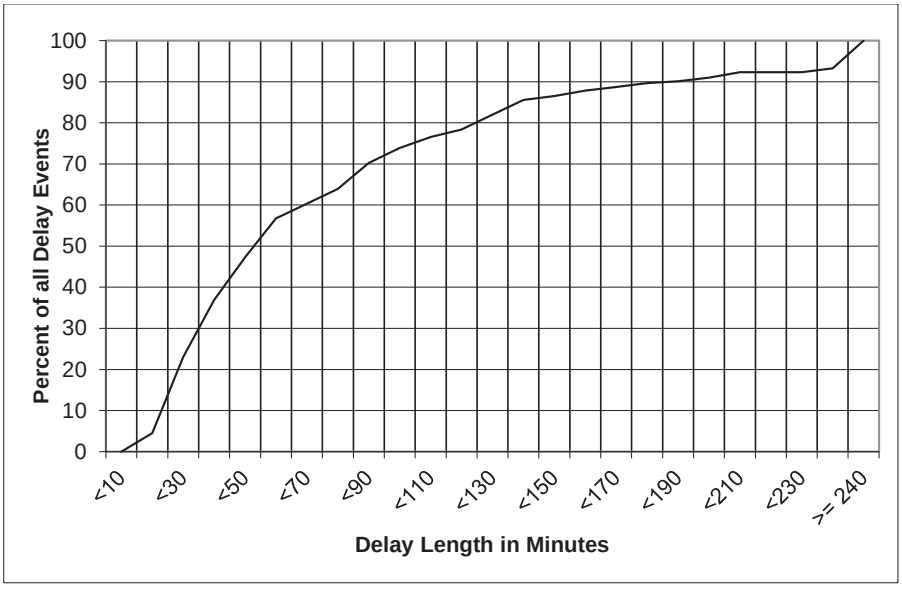

Figure 59: Cumulative delay probability of preventable faults

The above discussion assumes that reliable condition monitoring is available in all fault-causing systems. However, this is a very optimistic assumption and not likely realistic. The following figures show how the results differ when it is assumed that only faults in active system

components with integrated sensors (fan faults, pack faults etc.) can be prevented and other faults of other components (latches, connections, valves, sensors, leaks etc.) cannot be prevented.

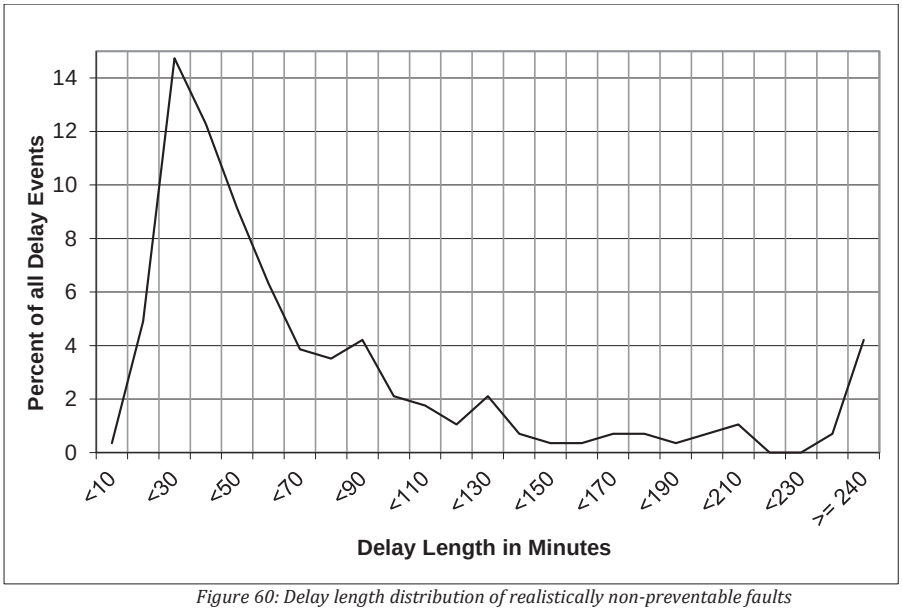

Figure 60: Delay length distribution of realistically non-preventable faults

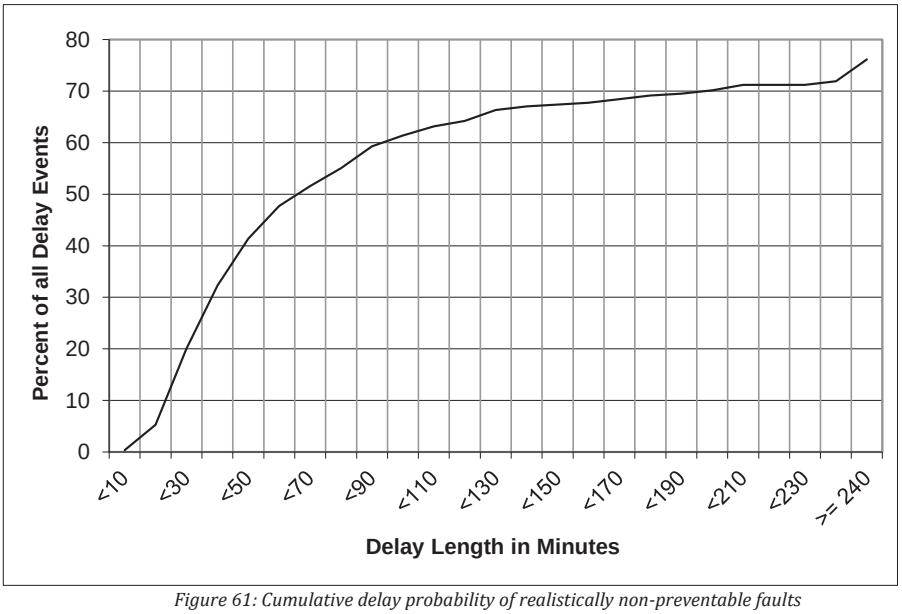

Figure 61: Cumulative delay probability of realistically non-preventable faults

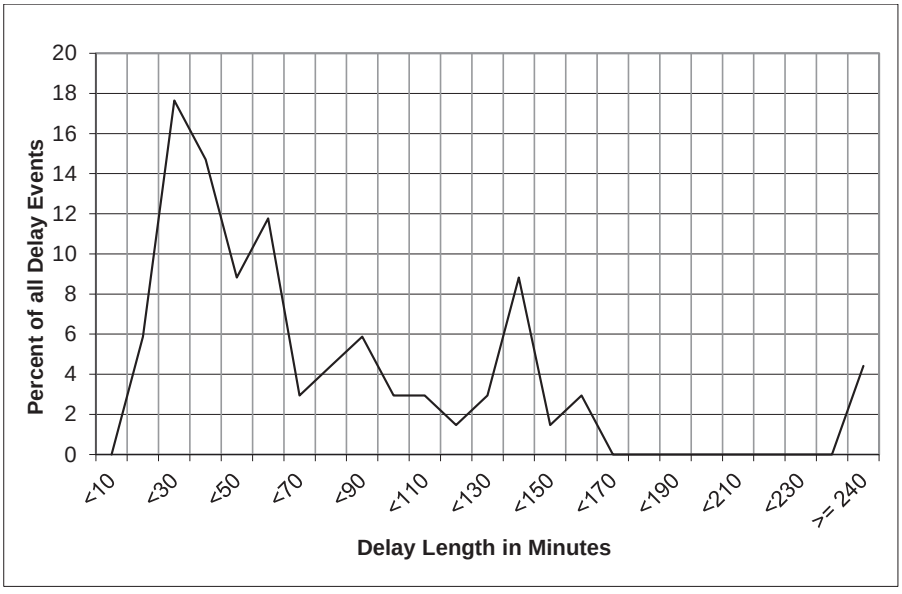

Figure 62: Delay length distribution of realistically preventable faults

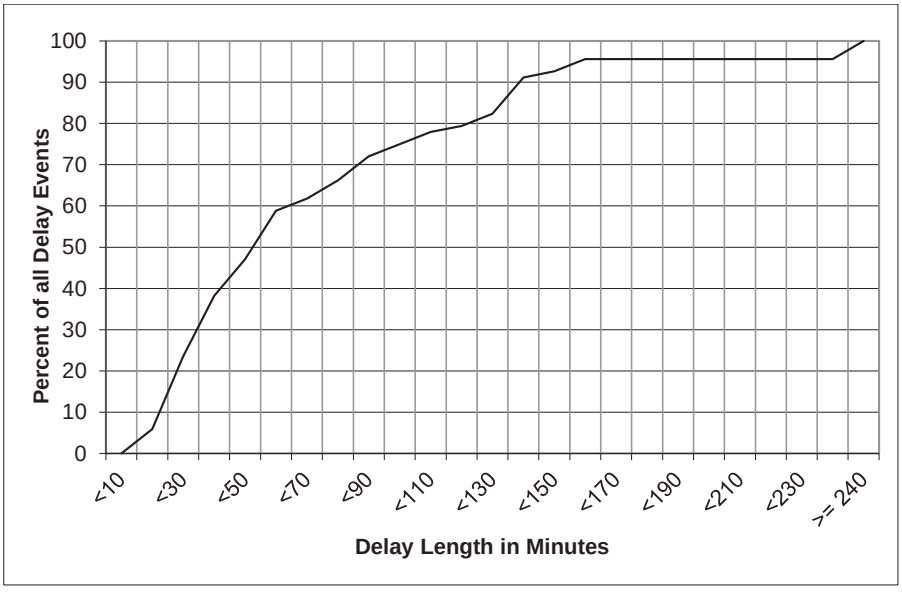

Figure 63: Cumulative delay probability of realistically preventable faults

Figure 62 shows that most preventable delays are shorter delays, and only a few longer ones can be prevented. However, some significant delays (longer than 170 minutes) can be prevented.

Figure 63 shows that it is realistically possible to prevent about 35 % of the delays, which is certainly a good goal to aim for. Figure 60 and Figure 61 show the distribution of the remaining delays. The average time of the remaining delays is about 88 minutes, and the average time of the prevented delays is about 97 minutes. The total costs for 100 delays are reduced by 108,891 US$ to 315,097 US$ or by 25 %. As only 6 % of unscheduled maintenance delays are caused by the air conditioning system, that system's total delays costs are reduced by 1.5 %.

Note: The cost saving calculations use 47 US$ per minute costs. All US$ values increase by 66 %.

6.6.5 Influence of CBM on Aircraft Costs

It is difficult to calculate the effect of CBM per flight hour, because only delays are analysed and delays occur on a "per flight" basis. The influence is stronger or weaker depending on the usage of the aircraft. In other words, the delay costs cannot be mapped onto the direct operating or direct maintenance costs. Instead, the effect is analysed on a "per flight" basis. Thus, the effects of CBM are based on the mission of the aircraft, when delays are considered.

CODA data for 2014 (Eurocontrol, 2015) show that the average delay per flight for all causes is 9.7 minutes. Technical issues cause 20% or 1.94 minutes of these delays (Knotts, 1999). This means that the air conditioning causes 0.12 minutes of delay per flight, as 6% of the technical delays are caused by the air conditioning system. In US$ this means the air conditioning costs 5.5 US$ (using the 47 US$) per flight.

CBM can save 1.1 US$ per flight (if the delays are reduced by 20%) for the air conditioning system based on the results of the analysis. This value seems low, but if it is multiplied by the number of flights per year, the effect is obvious. Eurocontrol (2014) forecast about 9,852,000 flights in Europe in 2015. In addition, this cost saving can be achieved with no additional hardware. Only new software needs to be installed to evaluate existing sensor data.

6.7 Discussion and Conclusion

This study shows that when condition-based maintenance and trend analysis are applied to components of the air conditioning system, the delays caused by unscheduled maintenance can be reduced by about 80 %. However, it is very unrealistic to assume that condition monitoring will be available for most aircraft parts. It is more realistic to assume that the existing sensors in aircraft systems will be used; thus, unscheduled maintenance delays will only be reduced by about 20 %. In addition, aircraft maintenance regulations are very conservative and restrictive. It would require a lot testing and verification to implement CBM in an aircraft environment and replace preventive maintenance, especially for critical systems. Using CBM to complement PM and to

gather data so that the preventive maintenance intervals can be optimized is possible, if scheduled PM maintenance actions are performed as required.

The major aircraft manufacturers (Airbus and Boeing) and the airlines flying their planes are interested in trend analysis to reduce maintenance, save costs and gain higher aircraft usage. However, significant work is needed to implement condition monitoring that makes useful predictions and does not cause an unnecessary work load. It would be optimal if existing sensors could be used for the condition monitoring, but this would reduce the number of possible detectable failures. Still it could possible to prevent delays of more than 30 minutes. Given the large number of delays due to factors the airline cannot control, it is worth putting more research and engineering effort into reducing unscheduled maintenance.

7 PAPER 2: DECISION TREES AND THE EFFECTS OF FEATURE EXTRACTION PARAMETERS FOR ROBUST SENSOR NETWORK DESIGN

7.1 Introduction

This paper proposes a condition monitoring system with sensor optimization capabilities to prevent unscheduled delays in the aircraft industry. Unscheduled delays cost airlines a great deal of money but can be prevented by condition monitoring (Gerdes, et al., 2009). The aim is to develop a simple condition monitoring system that can be understood by humans and modified by experts to incorporate knowledge that is not in the learning data set, using decision trees as the main tool. Decision trees satisfy the requirements and provide a ranking of data sources for condition monitoring.

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

New and better monitoring approaches are required for condition monitoring, because systems are becoming more complex and more difficult to monitor (Scerbo & Bailey, 2007). Condition monitoring requires reliable sensors. To obtain enough sensing data, special attention should be given to optimizing sensor allocation to ensure system diagnosability, lower sensing cost and reduced time to diagnosis (Sun, et al., 2009). Sensors can be used to determine the system health of control systems, but a failed sensor can lead to a loss of process control (Li, 2011) because the information about a system is incomplete if a sensor fails. Therefore, multiple sensors often monitor complex systems. An advantage of a multi-sensor system is that a single failed sensor shows its effects in multiple sensors (Li, 2011) and the system condition is defined by all information from the sensors. However, the system's health status becomes uncertain when a sensor fails or sends wrong data. This could trigger incorrect maintenance, including maintenance on a part with no failure, or long maintenance times to find the correct fault or not noticing the fault at all.

The Safety Integrity Level (SIL) defines the probability that the system safety function can be executed on a Safety Instrumented System (SIS). There are four SILs; level four is the level with the highest probability that the safety function can be performed. Sensor failure detection (sensor validation) is a critical part of the safety function. When a failure is detected, the SIS is put into a

safe state to avoid risk and damage to humans and machines (International Electrotechnical Commission, 2003) (International Electrotechnical Commission, 2010).

Redundancy is used to reduce the risk of model uncertainty (Emami-Naeini, et al., 1988). One way to create sensor redundancy is hardware redundancy; another is analytical redundancy (Emami-Naeini, et al., 1988). Analytical redundancy assumes multiple sensors deliver the same information, and, thus, a sensor fault can be compensated for. Hardware redundancy is not always possible, as it can be difficult to install multiple sensors because of physical or cost constraints (Novis & Powrie, 2006) (Yan & Goebel, 2003).

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

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

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

Figure 64: Basic condition monitoring process (Jardine, et al., 2006)

The basic condition monitoring process shown in Figure 644 has three steps:

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

7.1.1 Civil Aerospace Software Development

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

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

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

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

1. Software planning process
2. Software development process
3. Verification of outputs of software requirements process
4. Verification of outputs of software design process
5. Verification of outputs of software coding & integration process
6. Testing of outputs of integration process
7. Verification of verification process results
8. Software configuration management process
9. Software quality assurance process
10. Certification liaison process

The most complex step (besides coding) for a software developer is testing the coded software. Based on the DAL, the testing needs to satisfy certain code coverages. For DAL D and E, for example, no code coverage is required; only the requirements need to be tested. DAL C adds statement coverage to the testing requirements. This means the tests need to address each line of code. No dead code is allowed. In addition, DAL B requires decision coverage; each possible path in the code must be taken. For DAL A, developers must show that each variable for a decision in the code can influence the result (modified condition/decision coverage) and satisfy all other code coverages. All software testing needs to be done as black box testing. The tester cannot know the code but must work only with the compiled code, requirements and testing tools.

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

7.1.2 Feature Extraction

Feature extraction is the process of reducing the dimension of the initial input data to a feature set of a lower dimension containing most of the significant information of the original data (Friedl & Brodley, 1997). Extraction is done to extract important features from noisy sensor data (Fu, 2011) (Lin & Qu, 2000) and to avoid having too many input features (especially for vibration data) in the classifier learning phase (Lin & Qu, 2000). For these reasons, feature extraction is often a first and essential step for any classification (Lin & Qu, 2000). Accordingly, it is part of the data processing step in the basic condition monitoring process (Figure 64).

Features are extracted from the time domain and the frequency domain (Fourier transformation, wavelet transformation (Fu, 2011)). Basic features to extract are maximum, mean, minimum, peak, peak-too-peak interval etc. (Jardine, et al., 2006). Complex feature extraction methods include principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) (Widodo & Yang, 2007).

7.1.2.1 Time Domain Features

Time domain features can be direct features like the number of peaks, zero-crossings, mean amplitude, maximum amplitude, minimum amplitude or peak-to-peak interval (Jardine, et al., 2006) (Pascual, 2015). In addition, it is possible to analyse a signal using probabilistic methods like root mean square, variance, skewness or kurtosis to get features that represent the signal (Lambrou, et al., 1998). Other methods include using correlation, autocorrelation, entropy, principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) (Widodo & Yang, 2007).

7.1.2.2 Frequency and Time-Frequency domain

The Fast Fourier Transformation (FFT) transforms a signal from the time domain into the frequency domain. FFT takes a time series and transforms it into a complex vector that represents the frequency power in the frequency domain. The basis of the FFT algorithm is the discrete Fourier transformation (DFT), defined as shown in Equation (40) with $x_n \dots x_{n-1}$ as complex numbers.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \dots, N-1 \tag{40}$$

An FFT is performed in *O(N log N)* operations and can be calculated in real time because it can be executed in parallel. It is a widely used and well-established method (Emami-Naeini, et al., 1988) (Peng, et al., 2002). Recent research uses the discrete wavelet transformation (DWT) to represent time series in the frequency domain. The DWT represents time series in a time-scale form (Jardine, et al., 2006) and is especially suited to represent non-stationary signals (Lin & Qu, 2000).

7.1.3 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, (Quinlan, 1986)) was the first to construct decision trees. ID3 had some problems and was improved. The improved version, C4.5

(Quinlan, 1993), enhances the ID3 algorithm with the ability to handle both discrete and continuous attributes; it can handle samples with missing attributes and supports pruning of the tree at the end of the algorithm (removing branches from the tree).

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

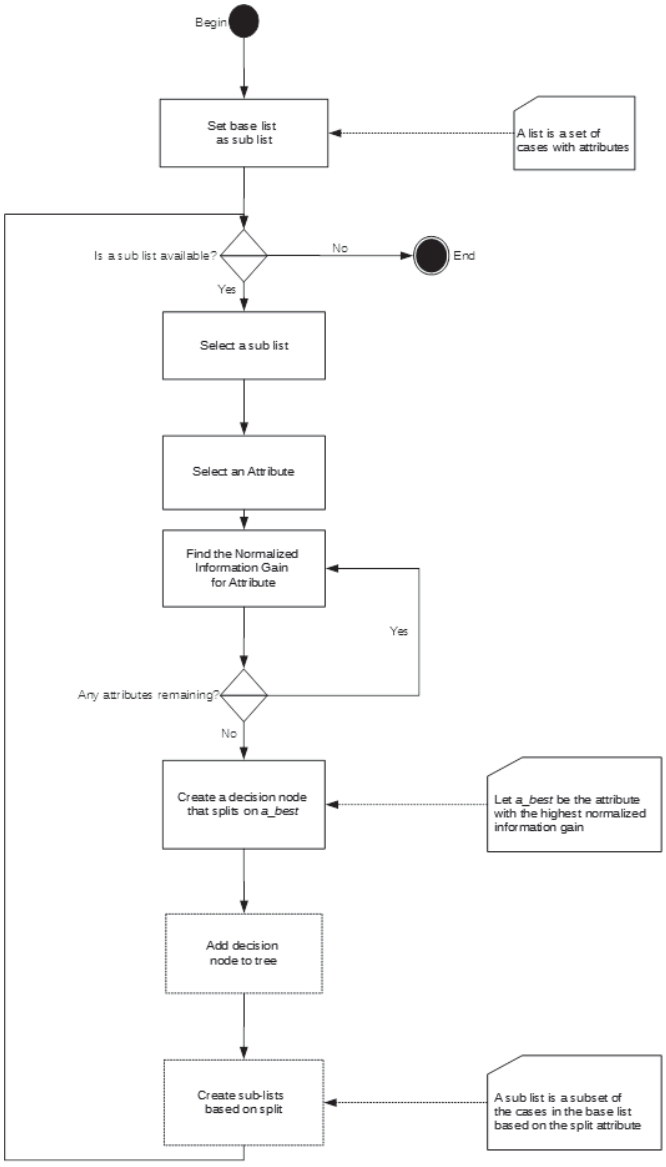

Figure 65: Decision tree algorithm flow chart

The result of the algorithm is a binary decision tree; 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, the classes with the most cases are favoured (Quinlan, 1993).

Information entropy is the knowledge contained in an answer depending on 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 (Russell & Norvig, 2003). Information entropy is calculated as shown below, where $P(v_i)$ is the probability of the answer v_i.

$$I\left(P(v_i), \dots, P(v_n)\right) = \sum_{i=1}^{n} - P(v_i) log_2 P(v_i) \qquad (41)$$

The information gain from an attribute test is the difference between the total information entropy requirement (the amount of information entropy needed before the test) and the new information entropy requirement, where p is the number of positive answers and n is the number of negative answers (Russell & Norvig, 2003).

$$Gain\ (X) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} \times I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right) \qquad (42)$$

Algorithm C4.5 uses the normalized information gain or the gain ratio. Split information (*Split Info*) shown in Equation 43 is the information gained from choosing the attribute to split the samples.

$$Split\ Info\ (X) = -\sum_{i=1}^{n} \frac{p_i + n_i}{p+n}\ log_2\left(\frac{p_i + n_i}{p+n}\right) \qquad (43)$$

Gain ratio is the normalized information gain and is defined as shown in Equation (44) (Quinlan, 1993).

$$Gain\ Ratio\ (X) = \frac{Gain\ (X)}{Split\ Info\ (X)} \qquad (44)$$

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

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

Decision trees are used to solve a large variety of problem, e.g. tag speech parts (Schmid, 1994), land cover mapping (Friedl & Brodley, 1997), text mining (Apte, et al., 1998) or condition monitoring (Saimurugan, et al., 2011) (Sakthivel, et al., 2010) (Sugumaran & Ramachandran, 2007).

7.1.4 Basic Condition Monitoring Process Enhancements

Sensor optimization and sensor data fusion are enhancements of the basic condition monitoring process (Figure 64). Figure 666 shows how sensor optimization and sensor fusion can be embedded in the basic CM process.

Figure 66: Enhanced condition monitoring process

Sensor optimization is the basis for condition monitoring; it is performed before the monitoring process (sensor locations) or new sensors can be added (Fijany & Vatan, 2005). Sensor fusion is done before the actual data processing to improve the performance of the data processing by improving the input from the sensors (removing redundant and low influence features).

7.1.5 Sensor Optimization

Often multiple sensors (sensor network) are used to give a more complete overview of the environment than a single sensor can give (Jardine, et al., 2006) (Xiong & Svensson, 2002). This

increases the diagnosis ability (failure detection and localization (Emami-Naeini, et al., 1988)) of a system and makes sensor optimization critical for failure diagnosis. The problem of designing a sensor network is to find a set of sensors so that costs, observability, reliability, estimation accuracy and flexibility are satisfied (Kotecha, et al., 2008).

Sensor optimization can help to design a sensor network that satisfies all requirements. Definitions of sensor optimization are:

- Optimizing the position of sensors (Smith, 2005) (Fijany & Vatan, 2005).
- Optimizing the processing of sensor data (Farrar, et al., 2006).
- Optimizing the information gain of sensors.

Sensor optimization also means hardware redundancy optimization by identifying significant sensors from several available sensors to determine which ones give the most information about a system and increase the information gain.

The goal of sensor optimization is to prevent unnecessary hardware redundancy and to improve the reliability of the condition monitoring system. This optimization can be supported by identifying redundant information in sensor data (Emami-Naeini, et al., 1988). Traditional sensor optimization methods do not consider the requirements for prognostic and health monitoring (Shuming, et al., 2012).

7.1.6 Multi-Sensor Data Fusion

Having a network of different sensors monitoring a system leads to the problem of sensor data fusion. Multi-sensor data fusion combines sensor data from different sources into one consistent model. The main questions of sensor fusion are (Basir & Yuan, 2007):

- How to get accurate and reliable information from multiple and possibly redundant sensors;
- How to fuse multi-sensor data with imprecise and conflicting data.

Techniques for sensor fusion can be grouped into three levels (Castanedo, 2013) (Jardine, et al., 2006) (Ross & Jain, 2003):

- Data-level fusion, e.g. combining sensor data from same sensors directly (Lu & Michaels, 2009);

- Feature-level fusion, e.g. combining vectors and feature reduction techniques (Ross & Jain, 2003);
- Decision-level fusion, e.g. vote schemes (Ross & Jain, 2003).

Sensor data fusion is an important part of condition monitoring. Most systems have more than one sensor, and the sensors have different influences on the condition monitoring accuracy. Data for condition monitoring that need to be fused are often from sensors but they can also be event and process data (Jardine, et al., 2006).

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

A condition monitoring system can use one or multiple data fusion methods to detect the system conditions. Sensor fusion is difficult, as it depends on the target system and the sensors. One solution is to implement sensor fusion on all levels and use a heuristic optimization like genetic algorithms, simulated annealing or hill climbing to get the best sensor fusion methods for the given problem (data and system conditions).

7.2 Proposed Methodology

A decision tree is built using feature extraction to increase the classification accuracy. The decision tree is analysed to generate a ranking of the sensors and features. This ranking is used to decide which sensors add significant information/features and which do not.

Sensor fusion may be performed at the feature level. The decision tree represents the feature fusion sorted according to information gain. If sensor fusion is on the feature level, event data can be added to the feature vector. Conventional methods use a fixed set of features to create feature vectors for the decision tree training and neglect sensor fusion.

Decision trees are applicable in the aircraft environment. Their task is to merge features from different sensors into one system health model and to use this model to classify the condition.

The focus is on hardware redundancy based on information gain; the goal is to avoid having redundant sensors and to focus on those giving significant information for failure detection and

identification. Information gain is used to rank sensor importance and measure sensor optimization. Feature extraction can increase the information gain and significance of different sensors.

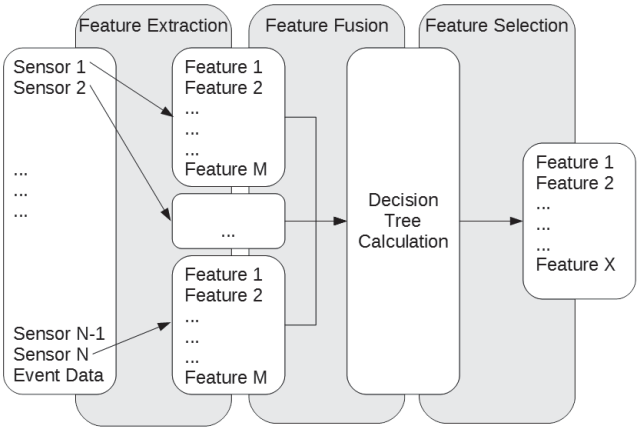

Figure 67: Feature selection process

7.2.1 Feature Extraction and Sensor Fusion

Feature extraction includes features from the time and the frequency domain. Time-frequency domain features are not used here; the method only uses basic methods so Fast Fourier Transformation (FFT) is selected. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations that can be different for each problem (Mierswa & Morik, 2005). This makes elementary extraction operations applicable for machine learning. The operations are also fast to compute and can be used for online monitoring.

The data from the different sensors are not merged at the sensor level but at the feature extraction level. A feature set is calculated for each input from each sensor. These features are merged into one feature input vector for the decision tree learning phase (Figure 67). No frequency features are calculated for signals that are nearly constant (Boolean switches, discrete system settings, certain process parameters).

7.2.2 Decision Tree Generation

The decision tree is generated using algorithm C4.5. Algorithm C4.5 is more advanced than the basic ID3 algorithm (accepts both continuous and discrete features, solves over-fitting problem

by pruning, handles incomplete data points) and is available as an open source implementation J48. Input for the decision tree generation is a set of features extracted from the sensor data. The parameters controlling the feature extraction are listed in Table 13.

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

Table 13: Feature extraction parameters

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

7.2.2.1 Block Width

The block width defines how the frequency domain is partitioned to get features for each partition. The sampling frequency is used for a full transformation. After the FFT, the frequencies are partitioned into blocks. The number of the frequencies grouped in a block is decided by the parameter *block width*. After partitioning, all blocks are transformed back into the time domain to get information about the behaviour of the block-signal over time.

7.2.2.2 Noise Reduction Factor

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

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

Listing 1: Noise Reduction

7.2.2.3 Peak Border

The peak border is used to count the number of frequencies with a power above multitude of the mean power. The MATLAB Listing 2 shows how the peaks are calculated. In the diagram, *peakBorder* is the parameter that can be varied; 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.

7.2.2.4 Confidence Factor

The confidence factor is a parameter of the software WEKA (Waikato, 2009) used to create the decision trees; it defines how much tree pruning is done. A confidence factor greater than 0.5 means no pruning is done. The lower the confidence factor, the more pruning is done.

7.2.3 Sensor Optimization

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

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

7.3 Validation

Two experiments were performed to validate the concepts and ideas. The first experiment shows the effects of feature optimization, and the second shows feature and sensor selection.

7.3.1 Feature Extraction Parameter Influence

To show the performance and concepts of the algorithm, a sensitivity analysis was performed using different process parameters. Figure 6868 shows the experimental process and the generation of results. The process is the following: samples are created and then feature extraction parameters (see Table 13) are modified. The influence of the modified parameters is measured by comparing the classification accuracy.

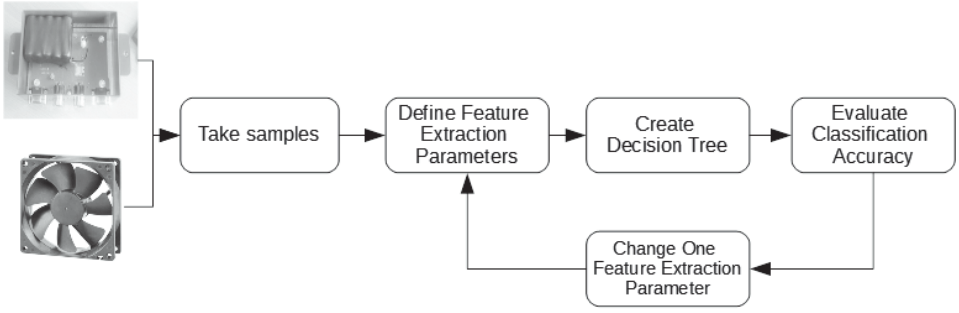

Figure 68: Experiment process diagram

7.3.1.1 Data Sampling

The data for the experiments and the feature extraction were sampled with an autonomous box (Figure 69) containing sensors and logic to save the data on a SD card. A test rig was used for the data collection. Vibration data with a sampling rate of 44 kHz were collected from a simple PC fan (Figure 71) to show the principles of the method. Data were saved in a raw wave format onto a SD card and transferred onto a PC. In addition to the raw sensor data, the condition of the component was saved. The fan is operated at standard speed, but three different conditions were sampled. Data were collected for the following conditions:

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

Figure 69: Data recording box

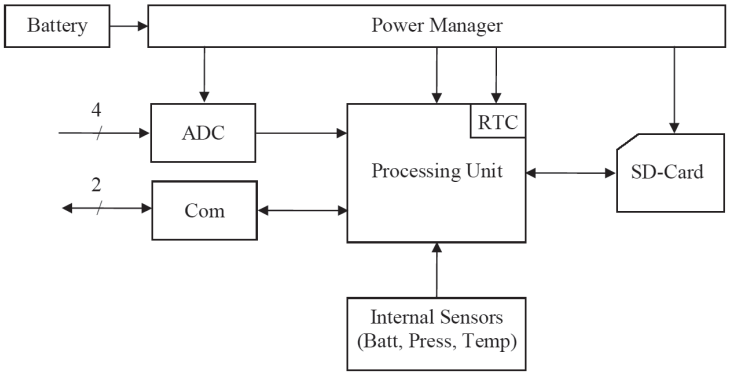

Figure 70: Data recording box architecture

Figure 71: Used PC fan

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

7.3.1.2 Calculating the Decision Tree

The decision tree was calculated with the open source Java software WEKA (Waikato, 2009). WEKA allows the user to test different algorithms and shows the classification errors. The correct data format is generated using a Java program that transforms the output files from MATLAB into input files for WEKA. J48 was chosen for classification; it is an implementation of the C4.5 decision tree algorithm and has a confidence factor of 0.0001. The confidence factor defines how much pruning is done to the resulting decision tree. The complete processed data were used as training data. After the generation of the decision tree, the same data were used to test the decision tree. In general, the training and the testing data should not be the same, but in this case, it was exactly what was required. The goal was not to classify new objects correctly, but to check how well the available data were classified and what part of the data gave the most information about the system.

7.3.1.3 Experiment Parameters

Calculations used the same input data but different parameter values to show the influence of the parameters on the results. Table 14 shows the available parameters with their values. All "Yes/No"-parameters are Boolean parameters; they toggle the calculation of that parameter

during processing. Default parameters are the values used when the effect of a parameter on the algorithm is tested. Only one value per test varies, while all other parameters keep their default value. The data processing with MATLAB generates several different input sets for the J48 algorithm. For every input set, a decision tree is generated, and the influence of the modified parameter is evaluated.

7.3.2 Sensor Optimization

The method was evaluated using aircraft sensor data from the air conditioning system of an A320 aircraft operated by ETIHAD Airways in the Middle East. The sensor data from the aircraft included 589 flights over two years. Each sensor reading included over 80 values consisting of continuous (numerical) and discrete data (Boolean). The data were sampled with a frequency of 1 Hz. The sources of the data were bus systems from the air conditioning system. Most data were temperature data and valve states. The sensor data are shown in Table 14.

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

Table 14: A320 sensor data description

7.4 Result Analysis

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

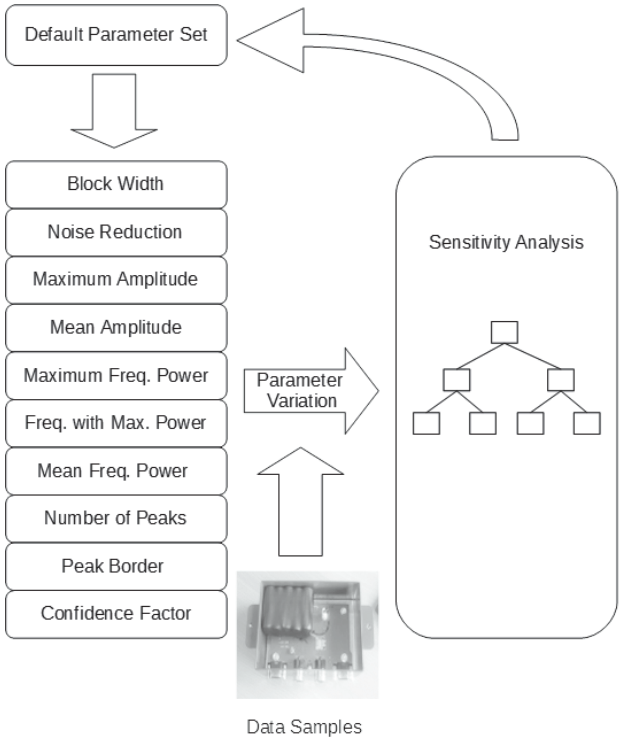

Figure 72: Validation process

Figure 722 summarises the validation process. The default parameter set was used to generate results; then, each parameter was varied based on its type. Boolean values were simply inverted, but continuous values were changed. A sensitivity analysis was performed after each parameter variation. After the parameter variation, a new decision tree with the same sensor data was created, and the change in the number of correctly classified samples was noted.

7.4.1 Parameter Evaluation

This section examines the results of the input sets, based on parameter variations. The influence of a parameter is judged by the number of correctly classified samples for every input set. Finding an optimal set of all parameters for the given samples, i.e., those giving the lowest overall false classification rate, is a complex problem. The problem is so complex that it cannot solved in a fixed

time, so heuristic methods must be used. The results below are not the optimal parameter values; they only show the influence of the different parameter values on the classification accuracy and suggest the importance of optimizing the feature extraction parameters.

The first calculation was performed using the default parameters. The results are shown in Table 15. The numbers imply that about three quarters of the test cases are correctly classified. The error rate is quite high, but that is to be expected, because a non-optimal parameter set was selected as the default parameter set.

Correct Classified	False Classified
73.4 %	26.6 %

Table 15: Results for default parameter set

Table 16 splits the classification error into different classes. As the table shows, most of the samples are correctly classified. For samples with no added weight and a big added weight, the classification is very good, but samples with a small added weight are often classified as samples with no added weight. The results are still good, however, because the small attached weight is quite light, and sensing accuracy is not very high.

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

Table 16: Distribution of wrongly classified samples

When only no added weight and big added weight samples are used, the number of wrongly classified samples drops to 5 %. This is to be expected, because there are bigger differences between them than between the small and no added weight classes.

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

Table 17: Results for default parameter set with no small weight samples

Table 18 shows the results achieved when the block width varies. The decreasing numbers imply that at some point, an optimal block width can be reached, and a minimum number of falsely classified samples can be obtained. The error rate increases after the optimal point if the block is too wide; more features are calculated if the block width is low.

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

Table 18: Results for block width

Table 19 shows the experimental results for a varying noise reduction. The results indicate the accuracy of the classification can be improved by removing all frequencies with a power below the mean level. However, removing more frequencies with a high power can reduce the classification accuracy significantly because significant information about the signal is removed. This result also shows the noise frequency features have a considerable influence on the accuracy of the classification.

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

Table 19: Noise reduction

The calculation of the maximum amplitude can be turned on or off. Table 20 and Table 21 show the results of each of these, respectively. Results show the maximum amplitude does not have a considerable influence on the classification in this problem. This indicates a high resilience of the input data to noise, something relevant to entropy. The finding suggests the data samples have a lot of information, with little uncertainty. This is even more interesting, because amplitude is the value recorded by the vibration sensors; it can be taken as an input without added processing.

Maximum Amplitude	False Classified
Yes	26.6%
No	26.5%

Table 20: Results for maximum amplitude per block

Global Maximum Amplitude	False Classified
Yes	26.6%
No	26.6%

Table 21: Results for global maximum amplitude

Table 22 and Table 23 show the influence of the mean amplitude values. Again, the influence is quite small. This is to be expected when the earlier results are taken into account. The amplitudes are the only features based on the time domain data. This can indicate that the time domain features are not very significant for the classification as is often the case for rotary movements. More time domain features should be added to the feature extraction operations (like probabilistic moments) to give more information about the significance of the time domain signal.

Mean Amplitude	False Classified
Yes	26.6%
No	27.7%

Table 22: Results for mean amplitude per block

Global Mean Amplitude	False Classified
Yes	26.6%
No	26.6%

Table 23: Results for global mean amplitude

Table 24 and Table 25 show the results of the parameter variations for the maximum frequency power. Again, these features do not influence the result of the classification very much. It is interesting to note that the classification error is reduced if the block-based maximum frequency power feature is turned off. This example clearly shows that having many features and features with little information gain can decrease the classification performance. It also highlights the importance of good feature selection.

Maximum Frequency Power	False Classified
Yes	26.5%
No	25.0%

Table 24: Results for maximum frequency power per block

Maximum Frequency Power	False Classified
Yes	26.6%
No	26.6%

Table 25: Results for global maximum frequency

Table 26 and Table 27 show the results of the parameter variations for the frequency with the maximum power. The Hertz of the frequency with the highest power (local for each block or for the complete signal) does not influence the result in a significant way. This is to be expected because the maximum power also has little influence.

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

Table 26: Results for frequency with highest power per block

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

Table 27: Results for global frequency with highest power

Table 28 and Table 29 show the influence of the parameter variations for the mean frequency power. Mean frequency power is a big factor and can improve the classification by nearly 4 %. The global mean values give no information about the condition of the fan. This result is especially interesting, because the other frequency based features have little influence on the classification error. However, as in the maximum frequency power feature, the error rate decreases if this feature is not used.

Mean Frequency Power	False Classified
Yes	26.6%
No	22.8%

Table 28: Results for mean frequency power per block

Mean Frequency Power	False Classified
Yes	26.6%
No	26.6%

Table 29: Results for global mean frequency power

Table 30 and Table 31 show the number of peaks has an even bigger influence on the classification than the mean frequency power, and the false classification rate can be improved by nearly 5 %. To this point, this is the largest performance increase.

Number of Peaks	False Classified
Yes	26.6%
No	21.8%

Table 30: Results for number of peaks per block

Number of Peaks	False Classified
Yes	26.6%
No	26.6%

Table 31: Results for global number of peaks

The peak border (the value defining what a peak is) also influences the calculation, as shown in Table 32. Results for the peak border show no clear trend, but the numbers suggest an optimum exists. These results are interesting if we take into account how much the error rate improves when peaks per block are not calculated. Very few peaks are generated if the peak border is set to 5. This is quite similar to having no peaks at all.

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

Table 32: Results for peak border

The confidence factor determines how much the decision tree is pruned and has an influence on the classification accuracy. With less pruning, more samples are wrongly classified. Over-fitting is reduced when pruning is used. More pruning increases the generalisation ability of the decision tree, generally a good feature, but a tree that is too small is not good. As in all other features, it is important to find the best value for the given classification problem.

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

Table 33: Results for confidence factor

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

7.4.2 Sensor Optimization

This section shows the sensor optimization using the aircraft data with 80 sensors. Figure 733 contains a sample decision tree. The most important feature is the overall (global) number of

peaks for sensor 31, followed by the overall (global) mean amplitude for sensor 45. Based on the decision tree, the sensors can be ranked as:

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

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

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

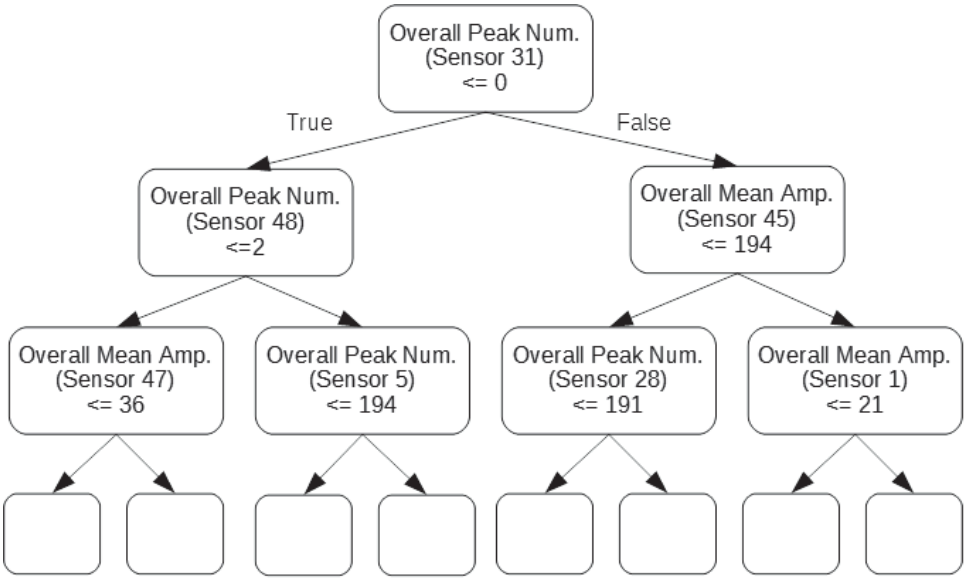

Figure 73: Example of a decision tree

7.5 Conclusions and Discussion

The method discussed here was developed to handle a specific problem (classification of a small number of classes with simple features) in a specific domain (civil aircraft operation with online monitoring). Decision trees are a good solution, given these constraints. However, decision trees

have limitations, and more powerful algorithms are available, if a similar problem needs to be solved outside the given constraints. More complex problems may need a different tool set. The methods used here are already well known and well researched, but their usage in this particular environment is novel. A previous paper (Gerdes & Scholz, 2009) addressed the topic but only evaluated the classification; it did not address sensor optimization, and it used an artificial experiment setup. This paper shows the result of sensor optimization using real-world data; it also explains the results and classification process in more detail than the earlier paper.

The architecture shown in Figure 67 is a good way to rank sensors by their significance for condition monitoring. The basic idea is to use a decision tree for feature ranking and feature fusion. Not all available features are used in the final decision tree thanks to tree pruning. As a result, fewer data are needed, and some sensors may not be used for the condition monitoring at all. It also improves the classification error rate and generalisation ability.

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

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

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

8 PAPER 3: AUTOMATED PARAMETER OPTIMIZATION FOR FEATURE EXTRACTION FOR CONDITION MONITORING

8.1 Introduction

An aircraft consists of many complex systems, which together define the state of the aircraft. Many systems are difficult to monitor, or they provide little information to the aircraft maintenance systems, but Gerdes et al. (2009) point to the potential savings if a faulty system can be found and replaced before it fails. Faults leading to a delay can be often prevented (in 20% of the cases without additional sensors and about 80% with additional sensors). For the air conditioning system of an aircraft, these faults include material weaknesses or broken valves. To prevent these faults, a new maintenance and monitoring strategy is needed. With condition based maintenance (CBM), it is possible to replace the system/component/part on a fixed interval. It is also possible 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 behaviour. Ideally, a computer should be able to compute the condition of a system to reduce the human input required.

CBM is based on condition monitoring and aims at preforming maintenance based on the system condition and trend of the system condition. The focus of CBM was traditionally on diagnosis, but with recent developments in condition monitoring, fault prognosis has become more important (Jardine, et al., 2006). Introducing new maintenance concepts, such as CBM, into the civil aircraft environment is difficult because of the focus on safety. A hybrid approach that uses the established maintenance plans as the first priority and condition based maintenance as the second priority might be a way to introduce condition monitoring (Phillips & Diston, 2011). Aerospace regulations also require that any decisions on maintenance, safety and flightworthiness must be auditable, and data patterns must relate to known information (Phillips & Diston, 2011). The military aircraft industry is less concerned with safety; thus, it has been traditionally easier to introduce new methods in military aircraft.

Condition monitoring relies on available sensor data and a system model of the system to be monitored. A model can be created based on physical properties (physical model), experience (knowledge-based model) or measured data (data-driven model) (Sikorska, et al., 2011). A model based on experience is defined by limits for certain sensor data; if the sensor data reach the limit, an action is triggered. The physical model compares the output of the model with the sensor data of the system; if there is a difference, an action is triggered. The data-driven model is generated

by analysing sensor data and generating rules for actions based on these rules. Artificial intelligence or statistics can be used to create a data-driven model (Si, et al., 2011). Data-driven models can be much simpler and smaller for complex systems than a physical model, because the system model is reduced to a few abstract rules instead of complex mathematical equations (Mosallam, 2014). However, they require a lot of historical data (Mosallam, 2014). Physical models are well suited for finding the root cause of an error because they constitute a "white box" approach instead of a data-driven "black box" approach.

Future faults can be extrapolated based on the condition of a system in the present and the past (Montgomery, et al., 1990) (Bowerman & O'Connell, 1993). Nui and Yang (2010) show a data-driven approach can be used for prognostics.

In Gerdes (2009), signal analysis and machine learning are used to detect the health condition of an experimental setup. The concept is half automated and needs fine tuning because the process depends on several different parameters. Each parameter needs to be adapted to the data.

The goal of this paper is to automate the selection of a good parameter set for the feature extraction to generate features which yield the best results when analysed by the pattern recognition algorithm, leading to better classification and facilitating condition monitoring. Another goal is to use automation to reduce the need for human configuration of the system; the data collected are so huge that the role of humans in data cleaning and preprocessing consumes too many resources (Jagadish, et al., 2014).

8.2 Background

8.2.1 Condition Monitoring

Condition monitoring is defined as (European Committee for Standardization, 2015):

"Activity, performed either manually or automatically, intended to measure at predetermined intervals the characteristics and parameters of the actual state of an item"

It is based on three steps (Jardine, et al., 2006):

1. **Data acquisition**: Collecting and storing data from physical assets. This includes event data and condition data. Event data are what happened and what the condition data represent.
2. **Data processing**: The first step of data processing is data cleaning followed by data

analysis. Data analysis includes transformation of data from the time domain into the frequency domain and feature extraction.

3. **Maintenance decision making**

Condition monitoring can either be continuous or periodic (Jardine, et al., 2006). Continuous monitoring is often done by installed sensors and automatically by machines. Periodic monitoring can be done by humans and can include checks at regular maintenance intervals.

Implementing condition monitoring is difficult and costly. Many barriers prevent the use of condition monitoring for a large number of systems. These barriers include (among others) (Stecki, et al., 2014):

- The inability to accurately and reliably predict the remaining useful life of a machine (*prognostics*);
- The inability to continually monitor a machine (*sensing*);
- The inability of maintenance systems to learn and identify impending failures and recommend what action should be taken (*reasoning*);
- The initiation of CBM programs without full knowledge of how the system can fail;
- The focus of research in CBM on specific techniques (better mousetrap symptom).

Diagnosis and prediction are two goals of condition monitoring (Jardine, et al., 2006). Diagnosis (posterior event) deals with the detection (Did something fail?), isolation (What failed?) and identification (Why did it fail?) of faults when they occur. It is defined as (European Committee for Standardization, 2001):

"Actions taken for fault recognition, fault localization and cause identification".

Prognostics predict future faults and determine how soon they will occur (Jardine, et al., 2006). There are two types of prediction: prediction of remaining time until a failure occurs and predicting the chance that a machine operates without a fault until the next scheduled maintenance (Jardine, et al., 2006).

Figure 74 shows the basic condition monitoring process with the manual feature selection used by Gerdes and Scholz (Gerdes & Scholz, 2009). It is the base process for the optimization proposed in this paper. Training input data are recorded and sent to a feature extraction step, where noise is removed and features are extracted from the data samples. The data are sent to a pattern

recognition algorithm that tries to categorize the data based on the extracted features. The performance of the pattern recognition algorithm is influenced by the number of available samples to learn from, the data features and the algorithm itself.

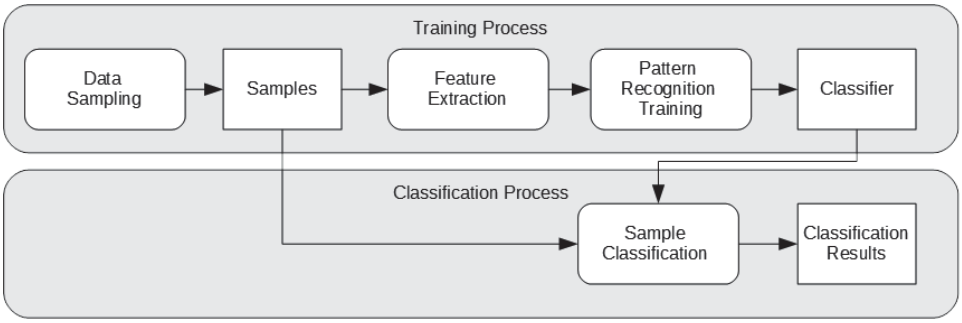

Figure 74: Condition monitoring process without optimization

8.2.2 Feature Extraction

Feature extraction is the process of reducing the dimension of the initial input data to a lower dimension while keeping most of the significant information of the original data (Fonollosa, et al., 2013), extracting features from noisy sensor data (Lin & Qu, 2000); (Fu, 2011) and avoiding problems with too many input features (especially for vibration data) for the classifier learning phase (Yen & Lin, 2000). For these reasons, feature extraction is often a first and essential step for any classification (Yen & Lin, 2000).

Methods include extracting features from the time domain and the frequency domain (Fourier transformation, wavelet transformation (Fu, 2011)) and clustering if necessary. Basic features can be maximum, mean, minimum, peak, peak-to-peak interval etc. (Jardine, et al., 2006). Complex feature extraction methods include principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) (Widodo & Yang, 2007). Other feature extraction methods are: t-test, correlation matrix, stepwise regression and factor analysis (FA) (Tsai, 2009). A comparison of the different feature extraction methods is found in Arauzo-Azofra et al. (Arauzo-Azofra, et al., 2011).

Clustering is needed if the data samples from which the features are extracted have no information about what the data represent (Li & Elbestawi, 1996). Clustering methods can be applied to group the data into different classes.

Selecting relevant features for classifiers is important for a variety of reasons, including generalization of performance, computational efficiency and feature interpretability (Nguyen &

De la Torre, 2010). Using all available features can result in over fitting and bad predictions, but it is not possible to look at each feature alone because many features are inter-correlated (Meiri & Zahavi, 2006). Noise, irrelevant features or redundant features complicate the selection of features even more. Thus, features are often selected using methods of pattern recognition or heuristic optimization or a combination of the two.

Sugumaran et al. (2007) show how different technologies can be combined for a single goal. A decision tree is used for feature selection and a proximal support vector machine (PSVM) for classification. Widodo and Yang (2007) combine ICA/PCA and SVM for feature extraction and classification. A combination of particle swarm optimization (PSO) and SVM is used for feature extraction and process parameter optimization by Huang and Dun (2008). Many algorithms combine genetic algorithms (GAs) with a pattern recognition method like decision trees (DTs), SVM or artificial neural networks (ANNs). In these combinations, the GA is used to optimize the process parameter (Samanta, et al., 2003) (Huang & Wang, 2006) or the GA is used for feature extraction and pattern recognition for classification (Samanta, 2004) (Saxena & Saad, 2007) (Jack & Nandi, 2002) (Samanta, 2004). Another popular approach is simulated annealing (SA) plus pattern recognition (Lin, et al., 2008)

Time domain features can be direct features like the number of peaks, zero-crossings, mean amplitude, maximum amplitude, minimum amplitude or peak-to-peak interval (Jardine, et al., 2006) (Pascual, 2015). In addition, it is possible to analyse a signal using probabilistic moments like root mean square, variance, skewness or kurtosis to get features that represent the signal (Lambrou, et al., 1998). Other methods include correlation, autocorrelation, entropy, PCA, ICA or KPCA (Widodo & Yang, 2007).

A few time domain features are listed in Table 34. Among these, kurtosis is an important and popular feature used in rolling element machines. Kurtosis defines the peakedness of the amplitude in the signal. When the amplitude follows a normal distribution, the kurtosis value is a fixed value 3. Beta parameters are the shape and scale parameters in the Beta distribution when assuming the amplitude of the signal follows a Beta distribution. The Beta distribution is a flexible distribution, and most signals can fit it. Since the parameters in the Beta distribution differ for normal and defect signals, these parameters have been used to diagnose failure (Heng & Nor, 1998). However, Heng et al. (1998) argue that the Beta method does not show significant advantages over kurtosis and the crest factor for rolling element bearings.

	Feature	Definition		Feature	Definition
1	Peak value	Pv=(1/2)[max(xi)-min(xi)]	6	Clearance factor	Clf = $\frac{Pv}{(\frac{1}{n}\sum_{i=1}^{n}\sqrt{\lvert x_i\rvert})^2}$
2	Root mean square	RMS = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i)^2}$	7	Impulse factor	Imf = $\frac{Pv}{\frac{1}{n}\sum_{i=1}^{n}\lvert x_i\rvert}$
3	Standard deviation	Std= $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2}$	8	Shape factor	Shf = $\frac{RMS}{\frac{1}{n}\sum_{i=1}^{n}\lvert x_i\rvert}$
4	Kurtosis value	Kv= $\frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^4}{RMS^4}$	9	Normal negative likelihood value	NNLV= $-\ln L$; $L=\prod_{i=1}^{N} f(x_i,u,\sigma)$
5	Crest factor	Crf = Pv/ RMS	10	Beta parameter	Parameters in beta function

Table 34: State of the art time domain features (Pascual, 2015)

Kurtosis, crest factor and impulse factor are non-dimensional magnitudes. Such features are independent of the magnitude of the signal power. Root mean square (RMS), peak value, standard deviation, and normal negative likelihood (NNL) value are fully dependant on the power of the signal. Some negative factors, such as the poor quality of the sensors or the location where they are mounted, influence the power and quality of the acquired signal. The main advantage of non-dimensional features is that they are immune from these nuisance factors. Nevertheless, RMS is an important feature in signal processing. It measures the power of the signal and can be used to normalize the signal; that is why some features are derived from RMS. Many other features have been used in the past, for example, Beta-kurtosis (Wang, et al., 2001), Weibull negative likelihood value (Abbasion, et al., 2007) (Sreejith, et al., 2008), kurtosis ratio (Vass, et al., 2008) etc. All have been discarded because they are useless, too complex for calculation or do not make contributions to existing features.

8.2.2.1 Frequency and Time-Frequency domain

The Fast Fourier Transformation (FFT) transforms a signal from the time domain into the frequency domain. FFT takes a time series and transforms it into a complex vector that represents the frequency power in the frequency domain. The basis of the FFT algorithm is the discrete Fourier transformation (DFT), defined in Equation 40, with $x_n \ldots x_{n-1}$ as complex numbers.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \ldots, N-1 \tag{45}$$

An FFT is performed in *O(N log N)* operations (Ohm & Lüke, 2010). It can be calculated in real time because it can be executed in parallel with other operations. It is a widely used and well established method (Peng, et al., 2002); (Fu, 2011). Recent research uses the discrete wavelet transformation (DWT) to represent time series in the frequency domain. The DWT represents the time series in a time-scale form (Jardine, et al., 2006) and is especially suited to represent non-stationary signals (Lin & Qu, 2000).

Existing failure diagnosis is mostly focused on the frequency domain, e.g. using Fourier transform or wavelet transform. In the early stage of failure development, damage is not significant, and the defect signal is masked by the noise in the acquired signal. The periodicity of the signal is not significant. Therefore, spectral analysis may not be effective. But the periodicity is significant, so using the time domain feature is recommended. Because normal and defect signals differ in their statistical characteristics in the time domain, the combined usage of time domain features with other domains can improve diagnosis accuracy

8.2.3 Pattern Recognition

The three pattern recognition methods used in this study, decision trees, Bayesian networks and support vector machines, are explained in this subsection. The study examined condition monitoring in the aircraft environment, a strict and static environment. Thus, only deterministic algorithms were evaluated; ANNs are not included in this section, even if they are widely used.

8.2.3.1 Decision Trees

Decision trees are very easy to understand. They are unidirectional trees. The leaves of the tree are the categories of the data, and the nodes/branches are if-then decisions. Decision trees use the concept of information gain for learning in order to find the attribute in the data that divides the

data so that the most information is gained. Russell and Norvig (2003) explain decision trees in further detail. Wang and Huang (Wang & Huang, 2009) compare decision trees with other methods.

8.2.3.2 Bayesian Networks

Bayesian networks are also trees, but instead of using information gain to construct the tree, they 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 state of some other nodes. Bayesian networks are explained in further detail in Russell and Norvig (2003).

8.2.3.3 Multi-Class Support Vector Machine (SVM)

SVM was initially developed to classify two classes of objects. One decision function must be found for such binary classifications, but there are many applications with more than two classes; see Figure 75. To accommodate the multi-class problem, one solution is to merge several SVMs. One-against-all multi-class SVM is the most common multi-class.

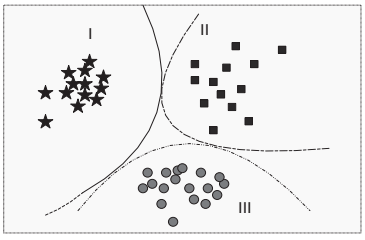

Figure 75: One-against-all SVM (Pascual, 2015)

Suppose there are k classes of patterns. Given l training data, $(x_1, y_1),...,(x_l, y_l)$ represents the input of SVM, a feature vector in this paper, and $y_i \in \{1,2,3,..n\}$ is the indicator of classes. The one-against-all method transforms the multi-class problem into n sub binary classification problems. The i^{th} sub binary classification problem labels the indicator of i^{th} data sets 1 and labels all the remaining data sets -1, or vice versa. The mathematical formula for this i^{th} classification is (Hsu & Lin, 2002):

$$\begin{aligned} \min\ & \frac{1}{2}(\omega^i)^T\omega^i + C\sum_{j=1}^{l}\xi_j^i(\omega^i)^T \\ & (\omega^i)^T\phi(x_j) + b^i \geq 1-\xi_j^i,\ if\ y_j = i \\ & (\omega^i)^T\phi(x_j) + b^i \leq -1+\xi_j^i,\ if\ y_j \neq i \\ & \xi_j^i \geq 0,\ j = 1,....,l \end{aligned} \tag{46}$$

After solving the above problem, we can obtain k decision functions:

$$\begin{aligned} & (\omega^1)^T\phi(x) + b^1 \\ & \ldots \\ & (\omega^k)^T\phi(x) + b^k \end{aligned} \tag{47}$$

Given an unknown input x, the predicted class of x is the class with largest decision function value, illustrated in Figure 76 and expressed as:

$$i = \arg\max((\omega^i)^T\phi(x) + b^i) \tag{48}$$

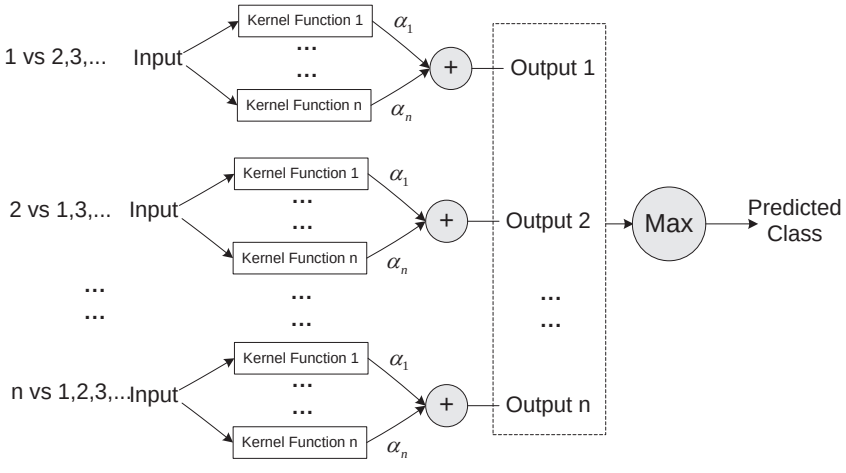

Figure 76: Multi-class SVM

SVM is a flexible classifier. When the kernel function is nonlinear, the decision function for failure detection is also nonlinear. The selection of the kernel function and the parameter in the kernel function influences the performance of failure diagnosis (Vapnik, 1998)

8.2.4 Optimization

The study used three different methods for the optimization concept: greedy search, simulated annealing and genetic algorithms.

8.2.4.1 Greedy Search

Greedy search is the simplest optimization algorithm of the three. It starts at a random point in the parameter space and compares the value of that point with the value of a random neighbour. If the neighbour performs better, the neighbour is chosen; otherwise, the current parameter set stays at the starting point. Russell and Norvig (2003) give a more detailed explanation of the greedy search algorithm.

8.2.4.2 Simulated Annealing

Simulated annealing is a more complex variant of greedy search. In simulated annealing, the algorithm may choose a worse neighbour depending on a probability, which decreases with the difference between the performance of the neighbour and the number of performed steps. With simulated annealing, it is possible to move away from a local maximum and avoid getting stuck. There is a more detailed explanation of the simulated annealing algorithm in Russell and Norvig (2003).

8.2.4.3 Genetic Algorithm

The genetic algorithm is a more complex simulated annealing. It evaluates different points in parallel, chooses the best and creates new variations 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 likelihood of finding the global maximum. The algorithm performs multiple evaluations in parallel; thus, it is slower than the other two algorithms. The algorithm has three steps (Russell & Norvig, 2003):

1. **Crossover**: The parameters of two random parents are combined to form a new individual.
2. **Mutation**: Each individual of a population can mutate with a certain probability.
3. **Fitness**: The fitness of each population member is calculated, and the algorithm is aborted if a certain fitness is reached; otherwise the steps are repeated.

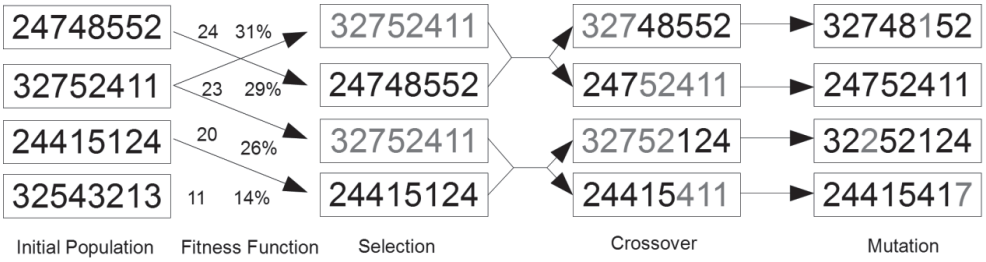

Figure 77: Genetic algorithm example

Figure 77 shows an example of a genetic algorithm. Genetic algorithms have been used for many different optimization problems (Golub & Posavec, 1997) (Jack & Nandi, 2000) (Stein, et al., 2005) and can be executed in parallel. Sörensens and Janssens (2003) and Carvalho and Freitas (2002) use GA to optimize the structure of a DT. Russell and Norvig (2003) give a more detailed explanation of the genetic algorithm.

8.3 Proposed Method

The proposed method is based on the basic optimization process as shown in Figure 74. An optimization loop for the pattern recognition training is added to enhance the performance. The goal of the optimization is to improve the feature extraction to improve the performance of the pattern recognition. Approaches traditionally extract features and then use these features for pattern recognition (Nguyen & De la Torre, 2010), but here, the feature extraction depends on the results of the pattern recognition and is not independent.

The feature extraction and the corresponding parameters strongly influence the accuracy of the pattern recognition. Previous experiments (Gerdes & Scholz, 2009) show that the solution space for an optimal parameter set is not linear and has 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 find a good solution to the optimization problem. The goal of the optimization is to maximize the percentage of the correctly classified data samples.

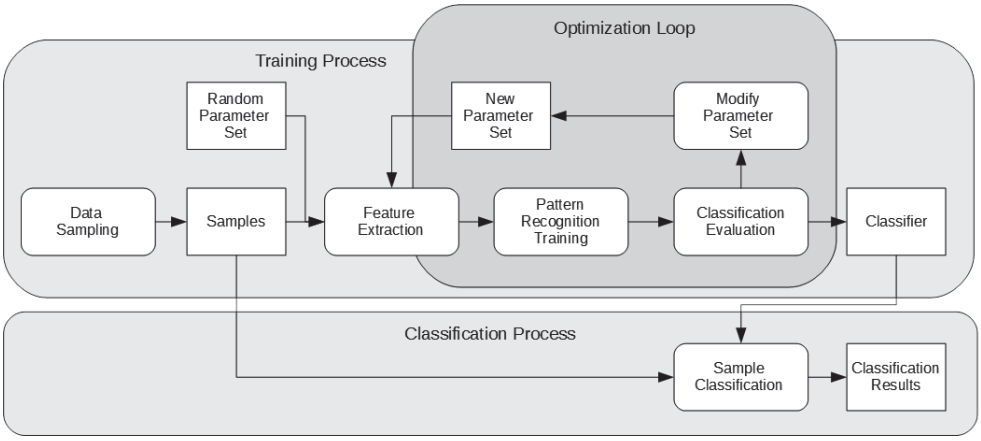

Figure 78: Condition monitoring process with parameter optimization

An optimization step is included in the process shown in Figure 78. The optimization takes place during the learning step and is needed before any data can be evaluated. The figure shows the modified condition monitoring process. First, a sample data set is processed with a random parameter set. Next, the features are fed into a pattern recognition algorithm that searches for patterns.

The resulting algorithm is tested and yields an accuracy percentage. At this point, 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 used for feature extraction in the condition monitoring process (Figure 78). All three algorithms are adapted to the problem and start from a given database 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 so they can be chained.

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.

8.3.1 Training Process

The basic training process consists of sample generation by data sampling and the extraction of features based on multiple process parameters. The final step is applying a pattern learning algorithm to classify the samples into classes. The details of each step are described below.

8.3.1.1 Data Sampling

The proposed concept can work with any kind of input data; however, it is assumed the data are discrete signal/time series with more than one data point. A sampling frequency of higher than 1 kHz is required. If a lower frequency is used, the feature extraction process (parameter, operators) 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 has more than one dimension, the feature extraction algorithm must be executed for each data source separately; features must be concatenated into one feature vector before being given to pattern recognition.

8.3.1.2 Feature Extraction

Noise is reduced, and a feature vector is created during the feature extraction step. First, the data are transformed into the frequency domain, where the noise is reduced. Then, frequencies are grouped. Mean and maximum power and the number of peaks are calculated for every frequency group. 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 are calculated as a last step. Table 35 shows the parameters of the feature extraction and the possible values.

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

Table 35: Feature extraction parameters

Block width defines how many frequencies are grouped in the frequency domain to form a block for detailed feature extraction. The **noise reduction factor** defines how much noise will be reduced. The noise reduction in this concept removes all frequencies wherein the power is below the *noise reduction factor* times *mean power.* Equation 49 shows how noise is defined for this method:

$$Frequency\ Power\ < Noise\ Reduction\ Factor \cdot Mean\ Power \tag{49}$$

Peak border controls what frequencies are defined as peaks. It is the opposite of the noise definition. Any frequency where power is greater than or equal to the *peak border* times the *mean power* is defined as a peak. Frequency power is defined as the following:

$$Frequency\ Power\ \geq Peak\ Border \cdot Mean\ Power \tag{50}$$

The **confidence factor** controls how much tree pruning is done and is a parameter of the J48 algorithm of the WEKA software (Waikato, 2009). A confidence factor greater than 0.5 means no pruning is done. The lower the confidence factor, the more pruning is done.

All other parameters are Boolean parameters; control is a given feature and may or may not be calculated. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations that can be different for each problem (Mierswa & Morik, 2005). This makes elementary extraction operations applicable to machine learning. The operations are fast to compute and can be used for online monitoring.

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

The features are determined by the parameters in Table 35. The values for the parameters are randomly generated or generated during the optimization using a search algorithm, as explained in the next section (see Figure 78).

8.3.1.3 Pattern Recognition Training

Pattern recognition belongs to the area of artificial intelligence. It is used to find patterns in data that allow the algorithm to categorize those data. First, the algorithm has to "learn" or find the patterns in the data and construct a function or algorithm that represents those data. New data samples can use the function or algorithm to categorize the new data based on the experience of the old data. This method uses supervised learning, where each data sample belongs to a known predefined class. Possible algorithms are decision trees, support vector machines or Bayesian networks. Artificial neural networks (ANNs) were not included in our experiments (described in the next section), because they are not deterministic pattern recognition algorithms. Deterministic learning is an important factor when the aircraft environment is considered.

8.3.2 Optimization Loop

The basic pattern learning process, as explained above, is improved with an optimization loop. It improves the classification accuracy by selecting an optimal process parameter set for the feature extraction step. The optimization can use several different algorithms to find the optimum. Local search methods are recommended because of the large size of the solution space.

8.3.2.1 Greedy Search

The greedy search algorithm can be implemented without any modifications. A neighbour is defined by a block size of 1 Hz to 50 Hz and a block overlay that varies 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.

8.3.2.2 Simulated Annealing

A new neighbour 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 evaluations. All other values vary by one point and have a probability of doing so of 50 %. A linear function controls when a worse neighbour is selected. A neighbour 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, not the current value.

8.3.2.3 Genetic Algorithm

The genetic algorithm uses the same mutations as the simulated annealing algorithm. Reduction of the mutation variance over time forces a better convergence. New children are 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 from a fourth parent. The first third of the population is left unchanged, but the rest of the population can mutate. Genetic evolution is stopped if 20 generations with 24 members have been evaluated.

8.4 **Validation**

The data used for the experiments were generated in a test rig (see Figure 81) by Airbus in Hamburg. The test rig simulated a part of the air recirculation system of an A340-600 aircraft. Valves controlled the airflow of the two inlets and the outlet on the bottom. Several different valve positions were chosen for the experiment (0° is fully open, and 90° is fully closed), and the vibration of the recirculation fan was recorded. One sample was recorded every ten seconds. The outlet valve was never fully closed to prevent permanent damage of the equipment. A total of 25 conditions were recorded (see Table 36).

Inlet Valve 1 Position	Inlet Valve 2 Position	Outlet Valve Position
0°	0°	0°
0°	0°	45°
0°	30°	0°
0°	45°	0°
0°	45°	45°
0°	60°	0°
0°	90°	0°
0°	90°	45°
45°	0°	0°
45°	0°	45°
45°	30°	0°
45°	45°	0°
45°	45°	45°
45°	60°	0°
45°	90°	0°
45°	90°	45°
60°	60°	0°
90°	0°	0°
90°	0°	45°
90°	30°	0°
90°	45°	0°
90°	45°	45°
90°	60°	0°
90°	90°	0°
90°	90°	45°

Table 36: Valve positions for the experiment

Every condition was monitored for four minutes, resulting in 24 samples per condition. A total of 600 data samples were recorded. The data were recorded using an autonomous recording box by Airbus (see Figure 79, Figure 80 and Figure 81). Two vibration sensors were attached to the test rig (see Figure 81). Vibration and sound data were sampled with a rate of 44 kHz. Data were saved in a raw wave format with two channels onto a SD card and then transferred onto a PC.

Figure 79: Data recording box

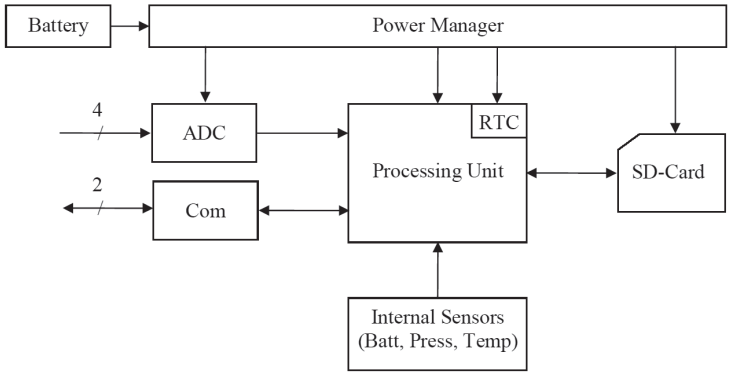

Figure 80: Data recording box architecture

All experiments used a 10-fold cross-validation to check the performance of the calculated pattern recognition. The software WEKA (Waikato, 2009) was used for the experiments.

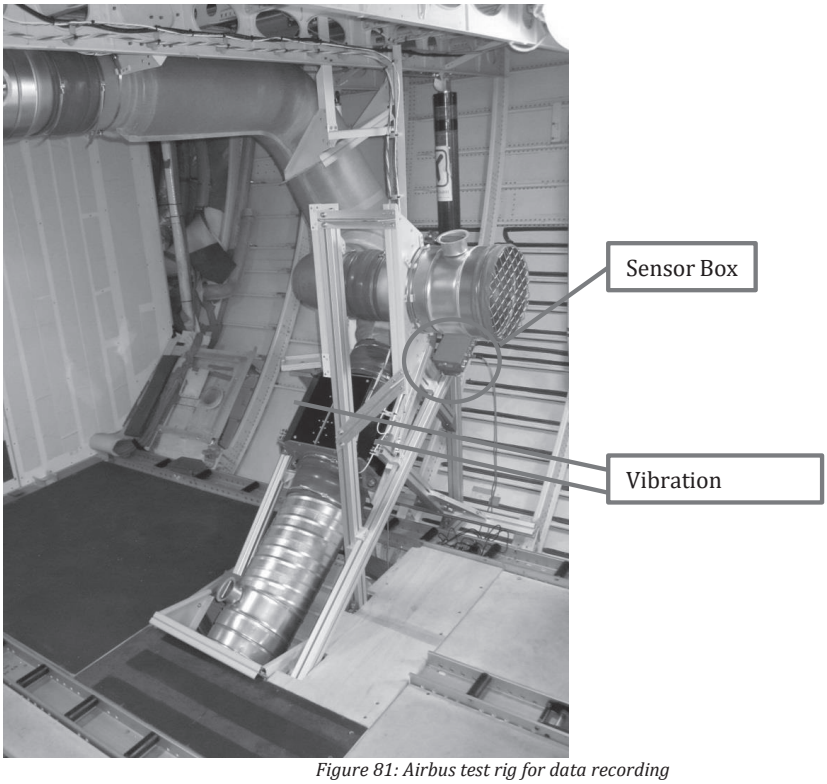

Figure 81: Airbus test rig for data recording

Four different experimental setups were used to evaluate the concept with the data from the test rig, optimization algorithms and pattern recognition algorithms. The same data set was used for all four experiments. Each experiment modified the optimization algorithm, the learning algorithm or the number of data sets for training.

Criteria for the earning algorithms:

- Greedy search stops if the best value does not change for 30 steps.
- Annealing stops if 480 optimization steps are executed.
- Genetic evolution stops if 20 generations with 24 individuals have been evaluated (for a total of 480 evaluated individuals).

8.4.1 Experiment 1: Sample Data

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

found with the reduced sample set was used to classify the full sample set to show if it was possible to train the algorithm with a reduced data set and still gain the full classification accuracy. The genetic algorithm and SVM were used.

8.4.2 Experiment 2: Random Seeds

The second experiment examined the effect of starting at different points. It used the genetic algorithm with 20 samples per class. The algorithm was evaluated ten times with different randomly selected starting populations.

8.4.3 Experiment 3: Optimization Algorithm

In the third experiment, the three optimization algorithms (greedy search, simulated annealing and genetic algorithm) were tested alone with different parameter sets. Then the simulated annealing and genetic algorithms were chained so that one produced starting points for the other. The idea was to use one algorithm to find a good starting point; the following algorithm could use that point to perform better than it normally would 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 required population size. To reiterate, the experiment used the following:

- Greedy search
- Simulated annealing
- Genetic algorithm
- Simulated annealing and genetic algorithm
- Genetic algorithm and simulated annealing

8.4.4 Experiment 4: Pattern Recognition

Experiment four compared the performance of the three algorithms for pattern recognition (decision tree, Bayesian network and support vector machine) when they used a genetic algorithm. The run time of the algorithms was measured against the percentage of correctly classified samples.

8.5 Results and Discussion

The results of the experiments are shown below. The calculations were done using an older Intel dual core processor. Note that they should not be viewed as absolute times; calculation time is just to compare the relative speed of the algorithms.

8.5.1 Experiment 1: Number of Samples

The pattern recognition accuracy of two sample databases with a different number of samples varied significantly (Table 37). As the table shows, the classification accuracy of the method with the smaller sample base and the full sample base was very similar. There was only a significant difference for the five sample databases. This means the data samples contain enough significant data so that only 10 samples are needed for training to get a good classification result. The training time can be reduced if only half the available data samples are taken. The other half can be used to verify the classification results as a testing data set. It is worth noting that this indicates a high resilience of the input data to noise and a few data samples are enough to ensure a good classification of the new data samples. The results also show that the algorithms are good at generalizing.

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

Table 37: Evaluation of different data sample sizes

8.5.2 Experiment 2: Different Seeds

The pattern recognition accuracy depended on the selected starting points (Table 38). Calculation times depended on the selected parameters, and they varied. The experiment reached a maximum accuracy of 99.4 %. This is higher than the value in Table 37 where the best value is 98 %. The random selected starting points and the randomness of the optimization algorithm caused this effect. With a larger population and more generations, it would be possible to reduce that effect and get a better convergence. Still, all starting points reached a very good accuracy.

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

Table 38: Evaluation of the influence of different starting points

8.5.3 Experiment 3: Optimization Algorithm

The selection of the algorithm greatly influenced the performance of the optimization.

8.5.3.1 No Optimization

The experiment performed a calculation without an optimization step to evaluate the results of the various optimization algorithms. Twenty-four random parameter sets were generated and evaluated, and the best parameter set was selected. This resulted in an accuracy of 97.5 % and took 517 seconds.

8.5.3.2 Greedy Search

The best result of the greedy search algorithm was 97.7 %, and the calculation time was only 1250 seconds. This was expected. Greedy search is a fast algorithm but it can easily get stuck in a local maximum. For better results, the algorithm needs to be executed more than once, but this negates the speed advantage.

8.5.3.3 Simulated Annealing

Simulated annealing had about the same speed as the genetic algorithm, i.e., about 5605 seconds. This is unsurprising, as both algorithms evaluated the function 480 times (the same number of iterations as for the genetic algorithm). Simulated annealing achieved an accuracy of 97.7 %, similar to the greedy search algorithm and a bit worse than the genetic algorithm. The problem space contained many local maxima and was huge. Simulated annealing did not get trapped in a local maximum as quickly as greedy search, but it could fall into that trap if the problem space has very many local maxima.

8.5.3.4 Genetic Algorithm

The genetic algorithm had the highest accuracy at 98 %. It needed 5418 seconds to finish. The genetic algorithm delivered results similar to those of simulated annealing. It searched at multiple places at once and chose the best ones to continue.

8.5.3.5 Simulated Annealing and Genetic Algorithm

Using simulated annealing to create a base population worked quite well, but the results were not better than using the genetic algorithm alone (98.6 %), and the calculation time was twice as long.

8.5.3.6 Genetic Algorithm and Simulated Annealing

The idea of using the best parameter set of the genetic algorithm as a starting point for simulated annealing also worked well and resulted in an accuracy of 98.3 %. The calculation time again was twice as long as for a single algorithm.

8.5.4 Experiment 4: Pattern Recognition

Table 39 shows the accuracy of the different pattern recognition algorithms with genetic algorithm optimization. To use the Bayesian network algorithm, all values need to be discretized (a feature can only have a pre-defined value). If numerical values are used, WEKA (Waikato, 2009) needs to discretize the feature values automatically, resulting in a "No memory left" error. To limit the amount of needed memory and make the calculation feasible, we limited the maximum number of blocks to 15; there were 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).

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

Table 39: Evaluation of different pattern recognition algorithms with optimization

Table 39 shows the SVM performed the best and decision trees performed the worst. The Bayesian network algorithm worked well because of the reduced number of features, but the decision tree algorithm seemed to suffer from the reduced number of features and performed weakly.

Table 40 shows the three algorithms tested with the same parameter set and without optimization. It is clear that SVM once again delivered the best results. There was a minimal optimization included in the calculation. Twenty-four random parameter sets were generated (the same as the starting parameter set for Table 39), and the parameter set with the best performance was used.

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

Table 40: Evaluation of different pattern recognition algorithms without optimization

While Bayesian networks delivered good results, they were not the best option. Table 40 also shows that the calculation time depended on the number of the blocks and, thus, the total number of features for the training. If that number was restricted, all algorithms performed significantly faster. The optimization process did not give a significant improvement in this setup; the solution space was much smaller, and the random starting points were good.

8.6 Conclusions

Adding an optimization loop to improve the feature extraction parameters and the classification accuracy showed good results. The classification accuracy of all tested learning algorithms improved simply by having a better feature set, with no changes required in the learning algorithms. The results show that an optimization can increase the performance of the signal analysis and pattern recognition. However, the increase is less than 5 %, largely because of the noise resilient input data. Still, it is possible to push the accuracy up to 99.4 %. Genetic algorithms performed well, even in the short searches with a small population. All algorithms showed good performance compared to choosing parameters by hand, which is nearly equal to choosing a random parameter set.

One goal of the research was to reduce the amount of expert knowledge required to use the system, and this goal was achieved. With automatic parameter optimization, no expert is needed to adept the feature extraction parameters for the problem. Instead, the algorithm adapts itself to the given data. 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 considerably reduce the computation time, and a much larger space can be searched in the same time.

There were a few limitations in the research. The results were based on the data from the Airbus test rig, and these do not represent real-world data. It is difficult to get real-world data for aircraft components because of safety restrictions, and it is really difficult to install measuring equipment in a non-test flight airplane. In addition, the number of tested algorithms was restricted because only deterministic pattern recognition methods were considered. However, it is possible to use different algorithms and methods for pattern recognition and optimization.

Future work will include testing the concepts on real-world data and using the method for condition monitoring and trending.

9 PAPER 4: FUZZY CONDITION MONITORING OF RECIRCULATION FANS AND FILTERS

9.1 Introduction

Complex systems can be difficult to monitor. Good condition monitoring depends on good sensors and a good model and on good interpretations of the data. However, interpreting sensor data is not a trivial task, and classification and condition monitoring can reduce the amount of information to be monitored. Often an expert is needed to interpret the data and make a meaningful "classification". Another problem with a “crisp” classification is that the user has no knowledge of the “stability” of the classification. Stability in this case means how fast the classification can change when the input data change. This is important information when working with sensors, because small sensor errors could cause a misclassification.

Classification is also essential to failure diagnosis. If a class represents a failure, it is useful for failure diagnosis to know which classes are similar to the current class, because these failure classes may also be responsible for the current visible failure effects. This

Basically, classification maps an input vector onto a class based on a learned or given pattern. For system monitoring, the class can be a failure or condition of a system.

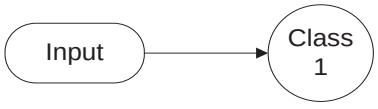

Figure 82: Common classification mapping of one input vector to one class

Most classifiers map an input vector to one output class (in Figure 8282, the input vector is mapped to "class 1"). However, to monitor systems and reduce NFF (no failure found) failures, knowing the probability of an input vector belonging to all possible classes, instead of only the most likely class (Figure 8383) is useful. The input vector is still mapped onto "class 1", but the input vector also matches the pattern of "class 4" in 89 % of the criteria for "class 4". Artificial neural networks (ANNs) can output the similarity of an input vector to other classes, if one output node is available for every class, but ANNs also have disadvantages.

There are several different methods for calculating the similarity of signals. Many are used in speech recognition (Rabiner & Juang, 1993).

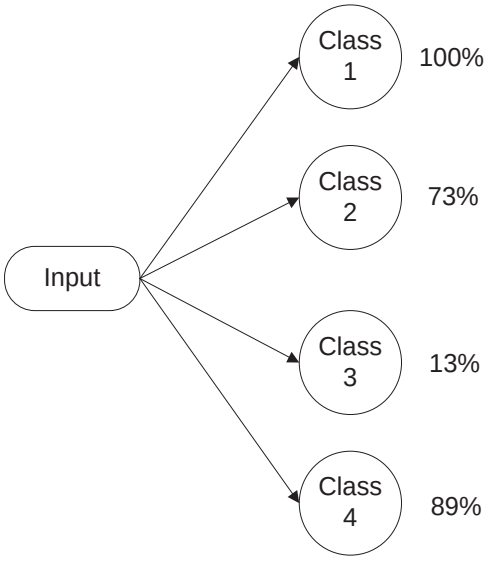

Figure 83: Classification mapping of one input vector to one class and output of similarity

Decision trees are simple and fast classifiers 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. They have a simple structure, fast calculation and inherent feature extraction. The ID3 algorithm (Iterative Dichotomiser 3) (Quinlan, 1986) was the first example of a decision tree, but it had some problems and was later improved. The improved version, C4.5 (Quinlan, 1993), enhances the ID3 algorithm with the ability to handle both discrete and continuous attributes. It can also handle samples with missing attributes and supports pruning of the tree at the end of the algorithm. The algorithm to build a decision tree uses the concept of information gain to choose attributes from the data and build the tree. The output of a decision tree is the most likely class for one data sample.

To get more information from classification, the decision tree inference algorithm has been modified to output the probabilities of all trained classes. This modification does not change the learning algorithm for decision trees, and it can be used with any binary decision tree.

In what follows, we show how a fuzzy inference of decision trees using numerical attributes can be used to gain more information about a system than just the current condition.

9.2 Decision Trees

As mentioned in the previous section, decision trees are a method from the area of artificial intelligence and are used for machine learning. They are often binary and each node has an if-then-else function on an attribute of the sample data. The ID3 algorithm (Quinlan, 1986), the first algorithm to construct decision trees, was improved in C4.5 (Quinlan, 1993). It has the ability to handle both discrete and continuous attributes; it can handle samples with missing attributes and supports pruning of the tree at the end of the algorithm.

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

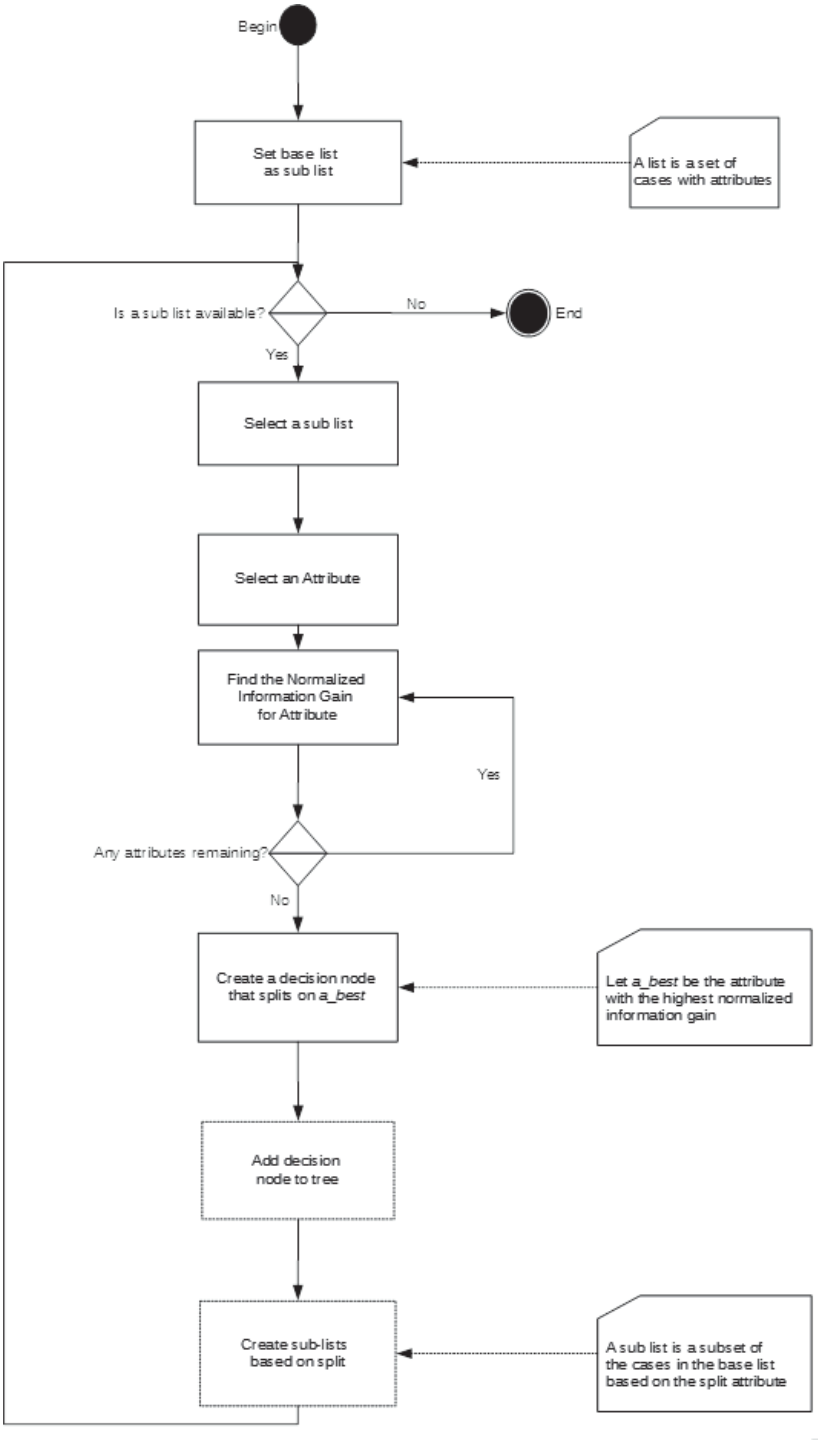

Figure 84: Decision tree algorithm flow chart

The result of the algorithm is a binary decision tree, where the root of the tree is the attribute with the highest normalized information gain. Nodes in the following levels of the tree represent attributes with lower normalized information gain. If pure information gain is used for splitting, the classes with the most cases are favoured (Quinlan, 1993).

Information entropy is the knowledge contained in an answer depending on 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 for which one has no data (Russell & Norvig, 2003). Information entropy is also called information and is calculated as shown below, where $P(v_i)$ is the probability of the answer v_i.

$$I\left(P(v_i), \dots, P(v_n)\right) = \sum_{i=1}^{n} - P(v_i) log_2 P(v_i) \qquad (51)$$

The information gain from an attribute test is the difference between the total information entropy requirement (the amount of information entropy needed before the test) and the new information entropy requirement, where *p* is the number of positive answers and *n* is the number of negative answers (Russell & Norvig, 2003).

$$Gain\ (X) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} \times I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right) \qquad (52)$$

Algorithm C4.5 uses the normalized information gain or the gain ratio. Split information (*Split info*) is the information gained from choosing the attribute to split the samples.

$$Split\ Info\ (X) = -\sum_{i=1}^{n} \frac{p_i + n_i}{p+n}\ log_2\left(\frac{p_i + n_i}{p+n}\right) \qquad (53)$$

Gain ratio is the normalized information gain and is defined as shown in Equation (54 (Quinlan, 1993).

$$Gain\ Ratio\ (X) = \frac{Gain\ (X)}{Split\ Info\ (X)} \qquad (54)$$

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

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

Decision trees are used to solve a large variety of problem, e.g., tag speech parts (Schmid, 1994), land cover mapping (Friedl & Brodley, 1997), text mining (Apte, et al., 1998) and condition monitoring (Sugumaran & Ramachandran, 2011) (Saimurugan, et al., 2011) (Sakthivel, et al., 2010).

9.2.1 Fuzzy Decision Trees

Decision trees can be evaluated and created using fuzzy rules and concepts. Most often, fuzzy attributes and values are used to create a fuzzy decision tree that operates on fuzzy sets. Fuzzy decision trees can be used to overcome some limitations of decision trees, e.g., where some of the available features are real- or multivalued, or a numerical decision is needed (Janikow, 1995). Small value changes can change the classification result (Quinlan, 1987). Wang, Zhai and Lu (Wang, et al., 2008) use fuzzy decision trees for database classification using rough sets. Yuan and Shaw suggest the following fuzzy decision tree induction (Yuan & Shaw, 1995):

1. **Fuzzifying the training data**. The data in the data set are converted into a fuzzy set using a membership function. Salary data, for example, can be converted into three groups, low, average and high. Each salary will have a value between 0 and 1, defining how well a class represents these data. Membership functions can come from mathematical, expert or statistical sources (Yuan & Shaw, 1995).

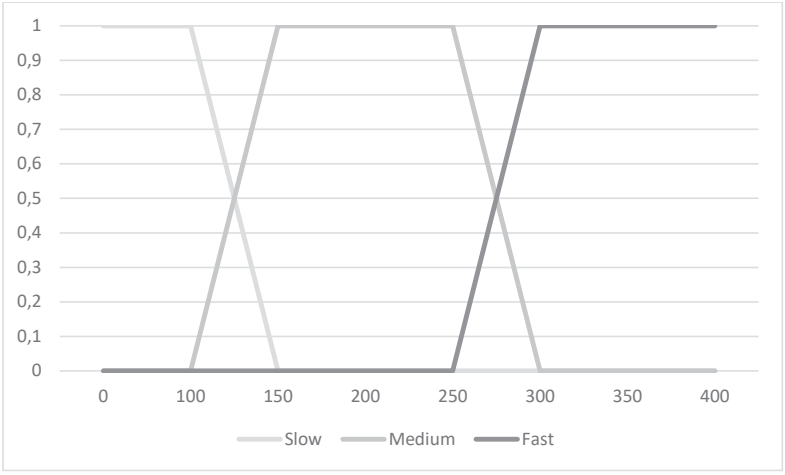

Figure 85: Sample fuzzy member functions for speed

2. **Inducing a fuzzy decision tree.**

   Step 1: Measure the classification ambiguity associated with each attribute and select the attribute with the smallest classification ambiguity as the root decision node.

   Step 2: Delete all empty branches of the decision node. For each nonempty branch of the decision node, calculate the truth level of classifying all objects within the branch into each class. If the truth level of classifying into one class is above a given threshold, terminate the branch as a leaf. Otherwise, investigate if an additional attribute will further partition the branch (i.e. generate more than one nonempty branch) and further reduce the classification ambiguity. If yes, select the attribute with smallest classification ambiguity as a new decision node from the branch. If not, terminate this branch as a leaf. At the leaf, all objects will be labelled to one class with the highest truth level.

   Step 3: Repeat step 2 for all newly generated decision nodes until no further growth is possible; the decision tree is then complete (Yuan & Shaw, 1995).

3. **Converting the decision tree into a set of rules**. Fuzzy decision trees can be converted into logical rules like crisp decision trees. Each path of the tree is converted into a single rule that represents the attribute decisions at each passed node until a leaf is reached.
4. **Applying fuzzy rules for classification**. Only one path/rule is evaluated if a crisp decision tree is evaluated. For a fuzzy decision tree evaluation, each path is evaluated, and for each path, a fuzzy result is calculated using the rules and fuzzy rules.

Janikow (1995) shows a slightly different method for fuzzy decision tree induction and explains how a fuzzy decision tree can be optimized to improve the classification accuracy.

The proposed method falls into the category of pre-fuzzification, where the data are fuzzified before the decision tree induction (Chiang & Hsu, 2002). Post-fuzzification is the fuzzification of the generated decision tree rules.

9.2.2 Concept

The proposed concept uses post-fuzzification to change the sample classification of an existing decision tree. One result per possible class is returned instead of one single classification result. The multiple results represent the similarity of the sample to each class. A decision tree is generated traditionally, but the process uses a fuzzy inference for the inference of an input vector:

1. Generation of feature vectors.
2. Decision tree generated with C4.5 based on labelled data samples.
3. Decision tree evaluated using the fuzzy inference concept.

The fuzzy inference calculates an output for every leaf of the tree. Every output value of a leaf is a value between 0 and 1 and represents the similarity of a sample to the class associated with the leaf. If multiple leaves are associated with the same class, then the leaf with the higher value is taken. The value is the similarity of the sample to the class. Similarity is a function of the distance of an input vector to a class. The similarity is calculated based on a weighting function of the decisions made (node inferences) to classify the sample. For condition monitoring and maintenance, the similarity can indicate possible other faults and conditions of a system. The class with the maximum similarity of 1 is still the same class that the C4.5 algorithm would generate. The proposed fuzzy inference works as follows:

1. Every path between two nodes or a node and a leaf has two labels: *PathDepth* and *PathWeight*.
2. Start at 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 test of the condition is true, label the *True* path as *PathDepth* +1 and *PathWeight* + 1.
   b) If the test of the condition is false, label the *False* path as *PathDepth* +1 and *PathWeight* + 1.
   c) Label the other path to child-nodes as *PathDepth* +1 and *PathWeight* + $nodeweight(AV, SV)$. See Equations (55 and (56.
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 not a leaf, continue from step 4.
9. If the node is a leaf, then return $\frac{PathWeigth}{PathDepth}$ and the leaf label and continue with another node.
10. If multiple leaves return values for a class, take the higher value.

The path from one node to another is labelled 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 returned at the end of the inference. Thus, all possible paths are evaluated, and we gain a measure of similarity of an input vector to all classes.

The advantages of this approach are that the similarity of a data sample to different conditions can be calculated and the decision tree generation algorithm does not need to be changed. It should be noted that the concept is designed so that the characteristics of one attribute (mean, minimum, maximum etc.) does not need to be known. In addition, the input vector for training and classification does not have to be modified in any way to fit the new algorithm.

It is assumed that the tree is a binary tree with numerical attribute values. Returned values of leaves are between 0 and 1. The weight for each node (*nodeweight*) is calculated as shown in Equation (55. Equation (55 limits the function values. In the equation, *nodeweight* can be between 0 and 1; AV (attribute value) is the value of the sample for the condition of the current node; and SV (split value) is the value of the node, which marks the border of the condition; e.g., the split value of "$x \leq 17$" is 17. Figure 866 shows the nodeweight if the decision is "false"; otherwise, the nodeweight is 1.

$$nodeweight(AV,SV) = \begin{cases} 0 & if\ nodeweight(AV,SV) < 0 \\ 1 - \frac{|AV-SV|}{2SV} & if\ 0 \leq\ nodeweight(AV,SV) \leq 1 \\ 1 & if\ nodeweight(AV,SV) > 1 \end{cases} \quad (55)$$

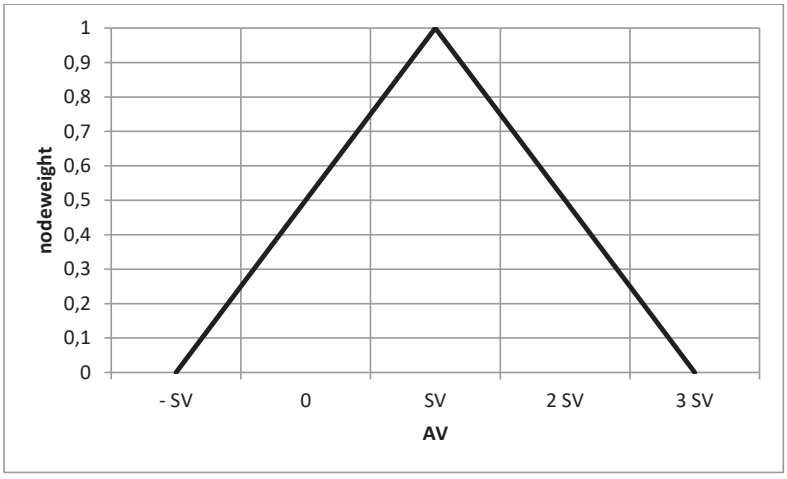

Figure 86: Nodeweight for "false" decision

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 than 1, the result at a leaf can be higher than 1.

This particular weighting function was chosen for several reasons. One is that a black box approach was used for the monitored system, so the meaning of the input values of vectors is unknown. It cannot be assumed that the training vectors for the algorithm contain the max or min values or even represent an average. These circumstances make it difficult to use absolute values. The only value that is available without adding additional information or calculations to the algorithm is the split value of a node. In addition, the correct classified class needs to have a value of "1" at the corresponding leaf. Choosing $2SV$ as the limit when the weight is 0 was an arbitrary choice based on tests with input vectors. It is possible to use a higher or lower value. If the maximum and minimum values for features are available, it is possible to use those as limits and use Equation (55. Otherwise, Equation 56 can be stated as:

$$nodeweight(AV, SV) = \begin{cases} 1 - \dfrac{SV - AV}{SV - min} \; if \; AV \leq SV \\ 1 - \dfrac{AV - SV}{max - SV} \; if \; AV > SV \end{cases} \tag{56}$$

A disadvantage of Equation (56 is that every decision returns a *PathWeight* of more than 0, and this can result in high similarity values. The concept was designed with only numeric values in mind, but it is possible to use Boolean or discrete values with a small modification of the process.

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

Nodes with discrete values: If a node has more than two children (one for every possible value of the attribute), only the path linked to the test of the node condition is weighted with 1. For every other unlabelled path to a child, the weight needs to be calculated separately.

9.2.3 Fuzzy Decision Tree Inference Example

An example of the process is shown in Figure 87. Each path from one node to another is labelled with *PathWeight* and *PathDepth* in the form of $\frac{PathWeigth}{PathDepth}$ e.g. $\frac{2}{5}$.

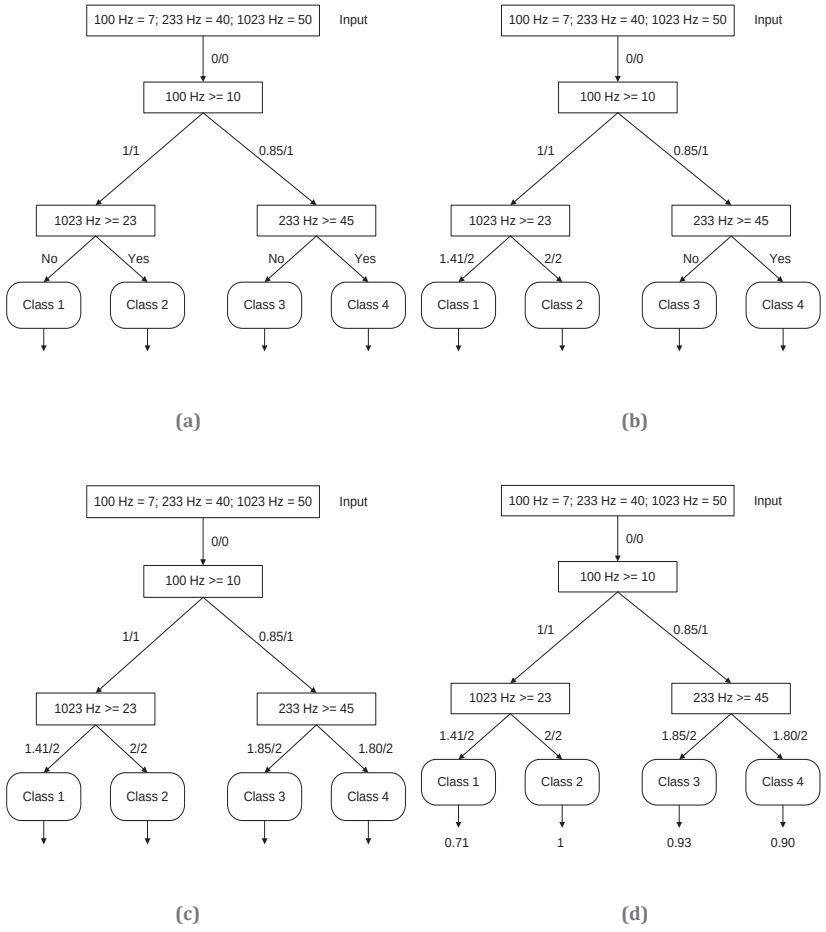

Figure 87: Fuzzy decision tree inference example

The input vector (values of the power spectrum of the transformed input signal) contains the power (energy per unit time) of the frequencies at 100Hz, 233Hz and 1023Hz. At the first node, the 100Hz value is checked to see if it is larger or equal than 10. The power is not larger than 10 (it is 7) so the right path, the *False* path, is assigned +1/+1. For the other path, a 1 is added to the *PathDepth* and 0.85 (the similarity) to the *PathWeight* (Figure 877 (a)). The 1023Hz node is evaluated in the next step. The input vector has a power at 1023Hz 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 receives a +0.41/+1 for a total of 1.41/2 (Figure 877 (b)). The process is repeated for the right-hand node (233Hz). Evaluating the node gives +1/+1 to the *False* path and +0.95/+1 to the other path (Figure 877 (c)). In the last step, the weight of the leaves and the classes are calculated. The input vector is classified as class 2. The similarity to class 1 is 0.71; to class it is 3 0.93 and to class

4 it is 0.9 (Figure 877 (d)). A similarity of 0.93 means the values do not have to move much to switch the classification result to class 3.

Only the attributes defining a certain class are evaluated during the inference. Attributes which the algorithm did not include in a decision path are not checked; e.g., after the root node, either 1023Hz or 233Hz is checked for classification of a class. So a class is either defined by 100Hz and 1023Hz or by 100Hz and 233Hz, but not by all three attributes. Which attributes are in a decision path and are checked is decided by the decision tree generation algorithm (in this case, C4.5). For more complex examples, an attribute can be checked multiple times (with different decision values) in a decision path; attributes which were neglected earlier are checked again. However, even in the simple example, all attributes are at least evaluated once, because the algorithm checks all paths. And the inference results are used to calculate the similarity values.

9.2.4 Fuzzy Decision Tree Forest Inference

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 inference. If all trees are evaluated, the results (similarities) for a class from all trees are added together and divided by the number of trees (taking the average of a class over all trees in the forest). This is done for all available classes.

9.3 Validation

Experiments were performed to test the performance of the concept. The goal of the experiments was to evaluate the fuzzy decision tree inference for a single decision tree and also for a decision tree forest.

9.3.1 Setup

This section details the setup of the validation. The validation was split into two parts. A single decision tree with fuzzy inference was analysed first, followed by a decision tree forest. Both used the data generated on a test rig at Airbus.

1.1.1. Data

Data for the experiments were recorded on a test rig. The test rig resembled a part of the air conditioning system of the A340-600. The focus of the test rig was the high pressure (HP) recirculation fan and filter. The test rig was equipped with two valves, one valve at the end of each of two open tubes. Table 41 shows the possible settings for both valves; 20 samples of one second duration were recorded for every condition of the test setup.

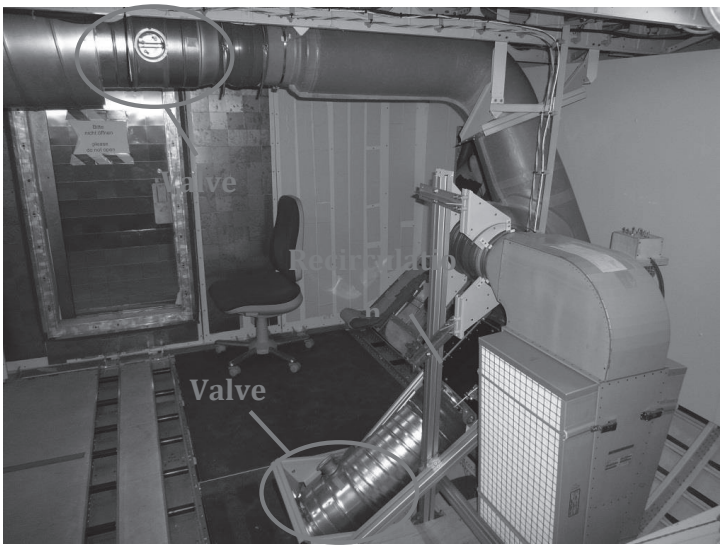

Figure 88: Test 4ig at Airbus 0perations GmbH

Valve 1	0	15	30	45	60	75	90
Valve 2	0	15	30	45	-	-	-

Table 41: Experiment conditions

Valve1 is the inlet valve, and *valve2* is the outlet valve. Twenty-eight conditions were recorded (Table 41). In addition, a *not running* condition was recorded. A total of 580 samples were recorded and labelled *valve1/valve2;* e.g. 15/0 means the test case valve 1 was closed by 15 degrees and valve 2 was fully open. Valve 2 was only closed up to 45° to prevent damage to the fan and tubes from overheating or overpressure.

The test rig also had a filter mounted connected to the fan inlet; there was always air flowing to the fan inlet even if valve 2 was completely closed. The fan, filter and the orange tube shown in Figure 88 are original aircraft parts. The positions in the rig were the same as in a real aircraft.

1.1.2. Feature Extraction

The Java software WEKA was used to perform the experiments. The decision tree was generated using the C4.5 algorithm; it is more advanced than the basic ID3 algorithm (accepts both continuous and discrete features, solves over-fitting problem by pruning handles, incomplete data points) and is available as an open source implementation J48. Input for the decision tree generation was a set of features extracted from sensor data. The parameters controlling the feature extraction are shown in Table 42.

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

Table 42: Feature extraction parameters

Block width defines how many frequencies are grouped in the frequency domain to form a block for detailed feature extraction.

The **noise reduction factor** defines how much noise will be reduced. The noise reduction in this concept removes all frequencies wherein the power is below the *noise reduction factor* times *mean power.*

Peak border controls what frequencies are defined as peaks. Any frequency whose power is greater than or equal to the *peak border* times the *mean power* is defined as a peak.

The **confidence factor** controls how much tree pruning is done and is a parameter of the J48 algorithm of the WEKA software (University of Waikato, 2009). A confidence factor of greater than 0.5 means no tree pruning is done. The lower the confidence factor, the more pruning is done.

All other parameters are Boolean parameters which control if a given feature is calculated or not. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations that can be different for each problem (Mierswa & Morik, 2005). This makes elementary extraction operations applicable to machine learning. The operations are fast to compute and can be used for online monitoring.

These parameters were used to generate a feature vector for each data sample. A genetic evolution algorithm was used to optimize the parameters to generate a decision tree with high classification accuracy. The three best decision trees were used to form a decision tree forest (Gerdes & Scholz, 2011). Twenty randomly ordered samples of every class were selected for the decision tree generation.

1.1.3. Single Fuzzy Decision Tree Inference

The experiments themselves were 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 inference. A test case "15/0" (explained previously) was used to compare the results; it was the correct class for all data samples. Five different node weighting functions were tested to evaluate the influence of the node weighting function. The average of all 20 fuzzy inferences of test case "15/0" was taken to reduce the influence of noise. The following equations were used in the inference:

- Equation (55 is the default node weighting function.
- Equation (57 is an example of a function where the weights are always close to 1 and 0.
- Equation (58 is a "flat" function with many values close to 1.
- Equation (56 is a function that always has a value $0 < x \leq 1$.
- Equation (59 is a function with more values that are 0; it scales with the value range.

$$nodeweight(AV, SV) = 1 - \frac{|AV - SV|}{0.01SV} \tag{57}$$

$$nodeweight(AV, SV) = 1 - \frac{|AV - SV|}{10SV} \tag{58}$$

$$\begin{aligned} if\ (AV\ \leq SV) \Rightarrow\ nodeweight(AV, SV) &= 1 - 5\frac{SV - AV}{SV - min} \\ if\ (AV > SV) \Rightarrow\ nodeweight(AV, SV) &= 1 - 5\frac{AV - SV}{max - SV} \end{aligned} \tag{59}$$

Each equation (set) replaced Equation (55. For each equation, the same parameter set and decision tree was used.

1.1.4. Decision Tree Forest

A decision tree forest with three trees was created for the experiments. Each tree had a classification accuracy of at least 90%. Each one used a different signal processing parameter set, generating different training vectors and test vectors for the inference. During the experiments, we evaluated the classification accuracy compared to a single decision tree with a classification accuracy of 95%. We also evaluated the fuzzy results for the same 20 data samples and neighbouring classes as in the experiments for the single decision tree.

9.3.2 Results

This section gives the results of the experiments, starting with the results for a single fuzzy decision tree inference and then moving to the results for a fuzzy decision tree forest inference.

1.1.5. Single Fuzzy Decision Tree Inference

Table 43 shows the averaged results for five fuzzy decision tree inferences with data samples of class "15/0". The numbers show that the correct class is classified. For comparison, three neighbouring classes (0/0, 15/15 and 30/0) are shown. They are very like class "15/0", but the variance in the results depends on the node weighting function. If a node weighting function is "flat" (i.e., the results are often higher than zero), the similarities are all close to 1 and have little variance (often between 0.9 and 1). If the function is "narrow" (i.e., zero results are produced), the variance is higher, and similarities can range from 0 to 1.

Class	Eq. (55	Eq. (57	Eq. (58	Eq. (56	Eq. (59
0/0	0.9585	0.5714	0.9917	0.8294	0.6462
15/0	1	1	1	1	1
15/15	0.9872	0.8357	0.9974	0.9209	0.8505
30/0	0.9996	0.9213	0.9999	0.8	0.8

Table 43: Averaged inference results

This effect occurs because *PathWeight* is often much higher than zero for "flat" functions. For the maximum "narrow" function, the similarity is based on the number of "true" decisions. This may be desirable for some applications, but it hides information that might be useful for condition monitoring. Small variations of an attribute are not represented in a very "narrow" function; they are filtered out. But the goal of condition monitoring is not only the similarity but also the ability to detect slight movements in the similarity to predict which "direction" a similarity is moving if one or more values is modified. Finding a fitting node weighting function depends on the problem and the goals of the application.

Table 44 shows a complete similarity matrix. The matrix contains the results of all leaves of the decision tree for Equation (57. The full similarity variance of the results is visible in the matrix. The table shows that the most similar classes do not always have to be neighbours; they can be farther away. Two of the neighbours are very similar, which is the desired result. The similarity of the other classes is mixed. The tendency is that the similarity is less if a class has a greater distance from "15/0".

Valve2/Valve1	0	15	30	45	60	75	90
0	0.5714	1	0.9213	0.5	0.65	0.7583	0.655
15	0.672	0.8357	0.6929	0.6667	0.6512	0.7278	0.4944
30	0.75	0.5	0.5	0.6	0.4444	0.5714	0.4286

45	0.65	0.75	0.5712	0.5667	0.25	0.5	0.25

Table 44: Similarity matrix

1.1.6. Fuzzy Decision Tree Forest Inference

This section shows how the results improved when a decision tree forest was used. The similarity values of the decision tree forest were calculated by taking the average similarity values of the single decision trees.

1.1.7. Classification Accuracy

The accuracy of a classification is defined as the number of correct classifications (CC) in relation to the total number of all classifications (TS), where TS is the number of correct classifications plus the number of wrong classifications (see Equation (60).

$$Classification\ Accuracy = \frac{CC}{TS} \tag{60}$$

Classification accuracy is a number between 0 and 1, which can be transformed into a percentage. The fuzzy decision tree inference does not influence the accuracy of the decision tree inferences because the *True* paths are weighted with "1" while all other paths are weighted with a positive number lower than 1 ($0 < x \leq 1$). The *True* path will always have the highest weight. However, the classification accuracy of fuzzy decision tree inference may be slightly lower than for standard decision tree inference because of the limits of numerical computations. If the weight of a node is very close to 1, it is possible that due to rounding errors, it may counted as a 1 instead of a value lower than 1. This happened in some of our experiments; it was more frequent if a 32-bit Java floating point data type was used instead of 64-bit Java floating point data type. Using a forest with three decision trees increased the classification accuracy from 95% per single tree to 99% for the complete forest. This is a significant classification accuracy improvement, but it comes at the cost of three times the calculation time. However, in the sample application of air filter monitoring, the time is not a critical factor.

1.1.8. Fuzzy Decision Tree Inference

By comparing the results of the four classes evaluated for a single decision tree, we get Table 45. The results are like a single decision tree with fuzzy inference, but the average similarity and the overall classification accuracy are higher. These results show that the fuzzy inference also works with a decision tree forest. But if one value changes in the input, the influence on the similarity is less. Decision tree forests with fuzzy inference are better at calculating the overall similarity of an input because different signal analysis steps are used, and different trees are evaluated. Thus, different features are checked and used to calculate the similarity result of a single decision tree.

Class	Eq. 5	Eq. 8	Eq. 9	Eq. 7	Eq. 10
0/0	0.9654	0.6905	0.9931	0.8725	0.7206
15/0	1	1	1	1	1
15/15	0.9815	0.8665	0.9963	0.9370	0.8736
30/0	0.9976	0.8528	0.9995	0.9286	0.9095

Table 45: Averaged fuzzy decision tree forest inference

Valve2/Valve1	0	15	30	45	60	75	90
0	0.6905	1	0.8528	0.5972	0.4833	0.5950	0.04718
15	0.6843	0.8665	0.7403	0.5637	0.4929	0.5537	0.4127
30	0.6630	0.5650	0.4061	0.6071	0.4648	0.6071	0.6111
45	0.6667	0.6533	0.4904	0.6389	0.4087	0.4401	0.4556

Table 46: Similarity matrix for the decision tree forest

9.4 Conclusion and Discussion

The proposed concept for fuzzy inference of decision trees can achieve the desired performance and deliver good results. It is possible to calculate a similarity measurement for classes in a decision tree without changing the algorithm used to create the tree. However, the design of the node weighting function is important. Paths should have a low *PathWeight* to get a meaningful similarity measurement. A "flat" node weight function is more sensitive to changes in the input values. On the downside, a "flat" function has a lower variance in the similarity values. "Narrow" node weighting functions have the opposite effect. Fuzzy classification plus decision trees creates a powerful tool for condition monitoring. The fuzzy decision tree inference is easily understood, fast and small, making it appropriate 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 fuzzification.

Another possible variation is to not limit the fuzzification to the split value and the “false” path, but to weight a part of the “true” path, creating a fuzzy border between the paths. The “true” path will always have a higher value than the “false” path, but it will only have the value 1 if the attribute value is far enough away from the split value (Chiang & Hsu, 2002).

10 Paper 5: Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning

10.1 Introduction

Unscheduled maintenance costs are a significant factor in aircraft operation (Gerdes, et al., 2009). Aircraft operators have significant increased costs if an aircraft departure is delayed or cancelled because of unscheduled maintenance. Unscheduled maintenance occurs 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 operating. Filters clog faster in a moist environment and slower in a dry environment. A clogged filter system may not only cause a delay but also cause passenger discomfort. An aircraft air conditioning system is monitored by pressure sensors that detect changes in the air pressure. Forecasts are done by relating the pressure difference to the probability of clogging and forecasting the mean time to clogging (Weber, 2008).

This paper describes a method to use machine learning to forecast the condition of a system. The method uses a decision tree to decide the best method to forecast a future data point and a genetic algorithm to adapt the decision tree to the current problem to improve performance. The decision about the best forecasting method is based on learned patterns from past data. The motivation for the approach was to create a simple method to forecast the condition of the air conditioning system, one able to adapt itself to different time series based on the operation of the aircraft and to handle the influence of events on time series data. 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/experts. Time series data 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 to the new condition. The method can use existing sensors and forecasting concepts for the forecast. It uses a genetic algorithm to improve the performance of the forecasting by searching for the optimal features set which, in turn, generates the best decision tree for the problem.

10.1.1 Time Series

A time series is a chronological sequence of observations on a particular variable (Bowerman & O'Connell, 1993). This can be the production of a company, a temperature, a pressure difference

or a system condition. The history of a system condition can be seen as a single or multidimensional time series. If the condition of a system is represented by a single variable, the resulting time series is one-dimensional. If the condition is represented by two or more variables, the resulting time series is multidimensional. A prediction of future events and conditions is called a forecast; the act of making such a prediction is called forecasting (Bowerman & O'Connell, 1993). Common methods for time series forecasting are:

- Simple linear regression (Montgomery, et al., 1990)
- Polynomial regression (Bowerman & O'Connell, 1993)
- Multi regression (Montgomery, et al., 1990)
- Moving average (Montgomery, et al., 1990)
- Exponential smoothing (Montgomery, et al., 1990)
- Autoregressive integrated moving average (ARIMA) (Montgomery, et al., 1990)

In addition, Golub and Posavec propose the use of genetic algorithms to adapt the approximation functions for forecasting (Golub & Posavec, 1997). Chaturvedi and Chandra use quantitative data and a neural network to forecast financial time series data (Chaturvedi & Chandra, 2004).

10.1.2 Decision Trees

Decision trees are a method from the area of artificial intelligence and are used for machine learning (Russell & Norvig, 2003). 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. The tree can be trained with the ID3 algorithm (Iterative Dichotomiser 3) (Quinlan, 1986). ID3 was the first algorithm to construct decision trees; the improved version of ID3 is C4.5 (Quinlan, 1993). There are other algorithms to construct a decision trees, including random trees. Decision trees are easy to understand, and a decision/classification can be calculated quickly. A sample decision tree is shown in Figure 899.

Figure 89: Simple decision tree

10.1.3 Genetic Algorithms

Genetic algorithms belong to the class of heuristic local search algorithms. They evaluate multiple valid solutions, choose the best ones and create new variations by combining and changing them. The new set of solutions is now evaluated; the best ones are selected, combined and changed. Each iteration of these steps is called a generation. The search is finished when the algorithm has calculated a certain number of generations or when an abort criterion is reached (Russell & Norvig, 2003).

10.2 **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. The quality of forecasting is increased by using a genetic algorithm (Russell & Norvig, 2003) to optimize the process parameters. This allows the process to adapt itself to different problems without human interaction. Training enables the forecasting process to use past data to predict the future data points in a much more reliable way than without training. The process can learn to use a different forecasting function when irregularities appear 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 behaviour to an exponential behaviour.

The process is divided into two parts, one for training the algorithm and optimizing the decision tree and the other for forecasting the time series. In former 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.

10.2.1 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. The goal of the training process is to find an ideal set of features of the times series that give the most information to find 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 has seven steps. The first four steps are executed only once to generate the input for the last three steps. All other 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. This means that a time series should not be too short and/or multiple time series are available. Sample time series data should include all relevant conditions and events. The algorithm can only learn from past data; 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 and 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, i.e., 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 what follows, we use a forecasting horizon of one data point. The forecasting horizon cannot be larger than the window size.

- Short term gradient calculation [Boolean]. This parameter determines whether the gradient for the data points in the current window should be calculated.
- Long gradient calculation [Boolean]. This parameter determines whether the gradient for the complete time series until the last data points should be calculated.
- Mean value calculation [Boolean]. This parameter defines if the mean of the sample data points should be calculated.
- Maximum value calculation [Boolean]. This parameter defines if the maximum of the sample data points should be calculated.
- Minimum value calculation [Boolean]. This parameter defines if the minimum of the sample data points should be calculated.
- Dimensions for classification [Boolean list]. This parameter defines which dimensions should be used for the calculation of characteristics.
- Zero crossing [Boolean]. This parameter decides if the number of zero crossings in the current window should be calculated.

Other parameters may be used. It is also possible to include long- and short-term trends 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. The time series data are 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 an n-data point window over the sample time series. See Figure 9090 for an example of a one-point window shift and a window size of three. Other features may be used, depending on the problem and the need.

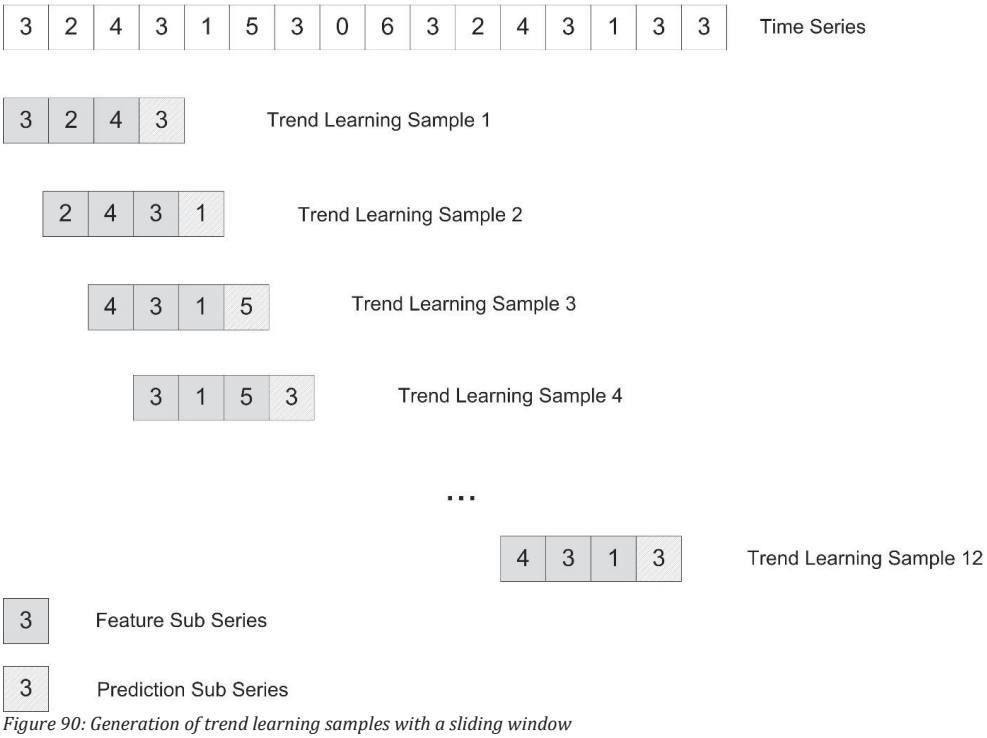

Figure 90: Generation of trend learning samples with a sliding window

Training Samples

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

$$f(x) = ax + b$$
$$f(x) = ax^2 + bx + c$$
$$f(x) = ax^3 + bx^2 + cx + d$$
$$f(x) = ae^{b(x+c)} + d$$
$$f(x) = a\ log\big(b(x+c)\big) + d$$
$$f(x) = a\sin\big(b(x+c)\big) + d$$
$$f(x) = a\cos\big(b(x+c)\big) + d$$
$$f(x) = 1 - ae^{b(x+c)}$$
$$f(x) = ab^{x+c}$$
$$f(x) = a(b(x+c))^2 + d$$

Note: this list was used in the 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, etc. can be used as functions. Kret (Kret, 2011) performed experiments 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 etc.

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. The performance is calculated by calculating the maximum squared error of the forecasting and using this value as the fitness of the decision tree. A number of different approaches can be used to calculate the fitness of the decision tree, such as the maximum confidence range. 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, a genetic algorithm is used to generate new valid solutions. New decision trees are generated until one of the decision trees performs above the given limit or until a certain number of generations has been calculated. Inputs to the genetic algorithm are the process parameters; the fitness is measured based on the performance of the decision tree. 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. This also reduces the need for a human expert to set the process parameters.

10.2.2 Forecasting Process

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

Calculate 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.

Evaluate Decision Tree

The second step is to evaluate the time series features by using the generated decision tree. The result is an approximation function.

Predict Next Data Point(s)

The third 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 is forecasted, the forecasting process is repeated with the new added data point as part of the past data points. Each iteration calculates a new data point.

10.2.3 Method Summary

The method consists of two processes: a training process and a forecasting process. Both processes use the same given set of methods to forecast the next data point for a given time series. Inputs 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, a decision tree is calculated and used for the forecasting. The decision tree is calculated using process parameters 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.

10.2.4 Process Modifications

The method can be modified depending on the problem at hand. One modification is to not only use the past data points plus one future point to calculate the approximation function, but also to use more than one future data point to account for a long term trend, i.e., a greater forecast horizon (Montgomery, et al., 1990). The forecasting will still calculate only one future data point, but the long-term trend is considered in the generation of the decision tree.

Another modification is 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 include multiple future data points up to all remaining data points of the sample time series (Kret, 2011).

A regression tree (Breiman, et al., 1984) can be used, if a future data point is directly forecasted without an approximation function. This reduces the required processing power, but the process needs to be iterated to forecast multiple data points.

10.3 Experiments

Three experiments were performed to validate the process. All experiments used a time series typical of the change of a system condition over time (Kolerus & Wassermann, 2011). The first experiment considered forecasting the time series without any noise. The second experiment added noise to the test samples, but the training samples were without noise. In 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; there were ten past data points with a forecasting horizon of one.

The time series for the experiment was a one-dimensional time series containing 120 data points. The first 80 data points were calculated via $f(x) = \frac{0.1}{80}x$; the data points between 81 and 120 were calculated via $f(x) = \frac{x-81}{120-81}^{2} + 0.1$. Figure 9191 shows the time series.

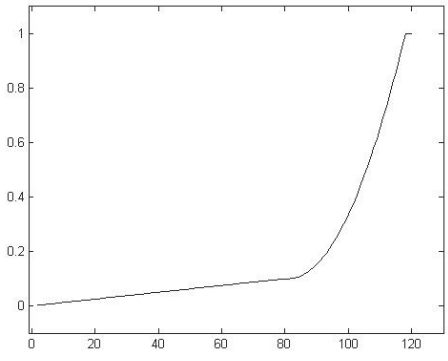

Figure 91: Plot of the experiment function

The quality of the forecasting was measured by six values. For every forecasting, the maximum, mean and minimum forecasting error were calculated; the confidence range for an absolute error

of 0.01, 0.05 and 0.1 was also calculated. The genetic algorithm for the experiments had a 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 %.

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. Figure 9292 plots the results of the forecasting. The black line is the input points, the red line is the sample time series to predict, and the green line is the calculated forecasting, starting at data point 30, 60 or 90. The image suggests the advantage of using a decision tree. The right side shows the forecasting using the best prediction method without using a decision tree to select the method. This method was also used to generate training samples in the training process. The left side has the same input, but the proposed process is used to forecast the next data point. The advantage of the decision tree becomes clear when the function suddenly changes (data point 80) from a 0-degree polynomial to a second-degree polynomial function. Here, using the decision tree forecasting method is much more accurate than using only the best fitting method.

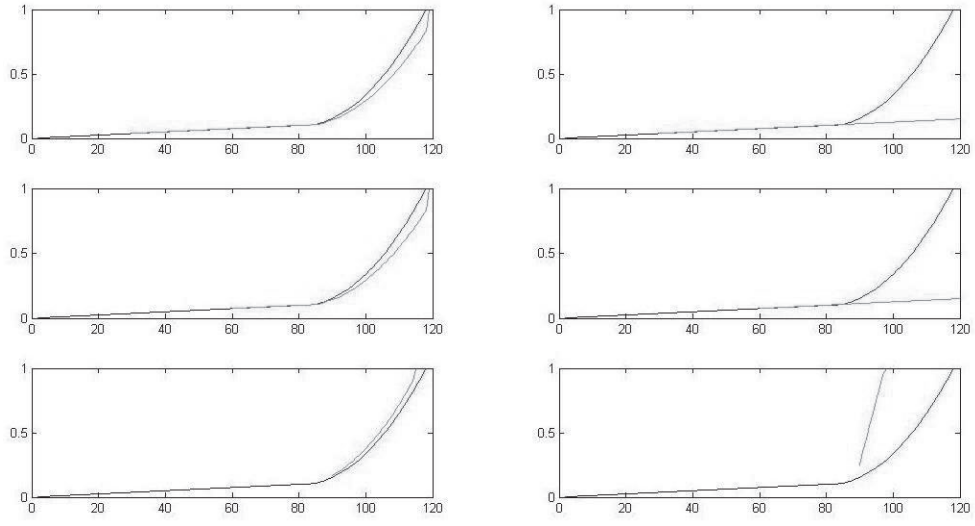

Figure 92: Plot with different starting points and forecast

Figure 9292 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 perfectly. The preconditions for this forecasting were ideal (no noise), and the time series was fairly simple. The advantage of using a decision tree is obvious.

10.3.2 Forecasting with Noisy Test Samples

In the second experiment, the training samples were not changed, but the testing samples were modified by noise. In the first test, a random number between -0.02 and 0.02 (2% noise) was added, and for the second test, -0.05 to 0.05 (5% noise) was added to each data point. The goal of the experiment was to see how well the method works with input noisy data. Results are shown in Figure 9393. The test was repeated 20 times to show the forecasting results for different noisy input data.

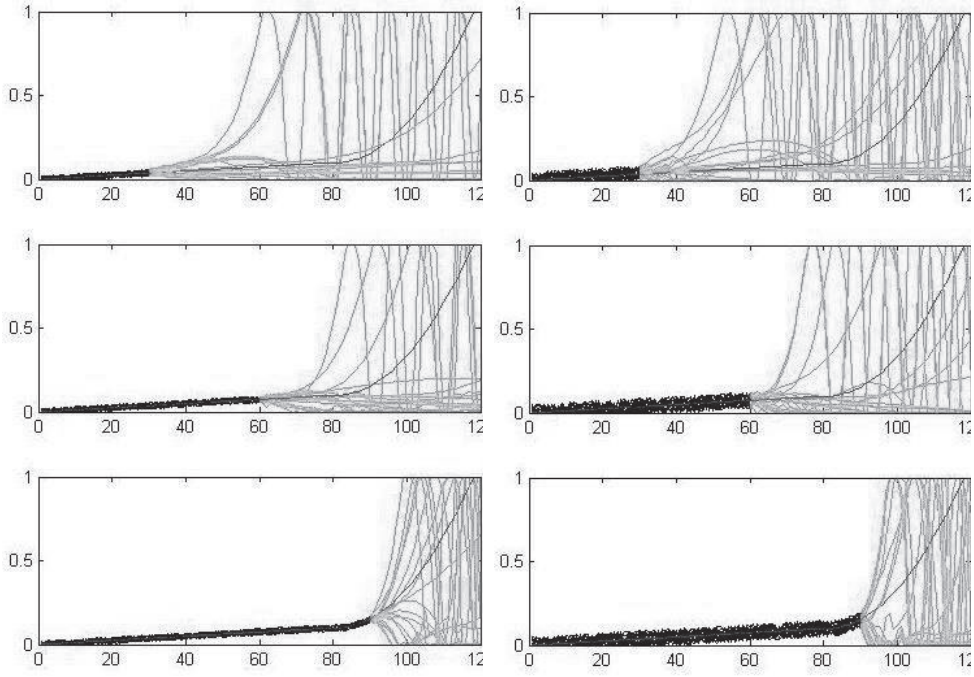

Figure 93: Plot with different starting points, noisy training data and noise

Oscillations were caused by functions used 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, the results were especially bad. The algorithm was unsure where it was in the time series. The high impact of the noise on the time series features because of the relatively small number of past data points (i.e., 10) caused the decision tree to make the wrong decision.

10.3.3 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 the 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 and 0.05 to 0.05 (5% noise) in the second test. The results are shown in Figure 9494. The test was repeated 20 times to show the forecasting results for different noisy input data.

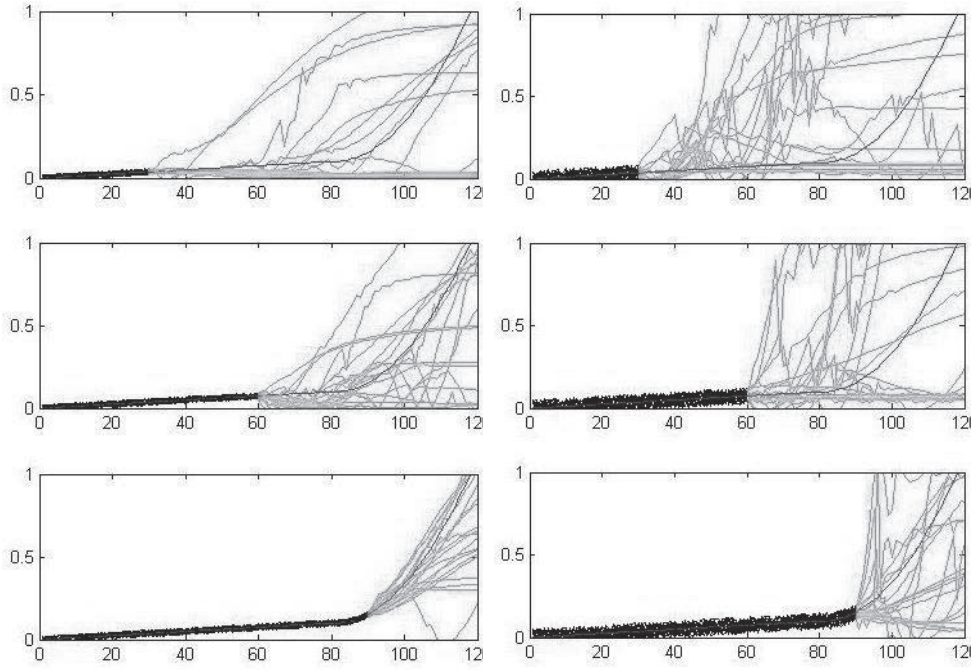

Figure 94: Plot with different starting points, noisy training and testing data and noise

In the figure, it is obvious that the accuracy of the forecasting is not as good, but this was to be expected. It is worth noting that only three noisy training time series were used. The predicted time series was similar to the original curve. The inaccuracy occurred because the algorithm was not sure of its position, and the approximation only had ten noise points to calculate the next data point. With more training data and more past data points, the algorithm would perform better. With noisy training and testing data, the algorithm used different functions for the approximation and thus created no oscillations.

10.4 **Conclusion**

The experiments showed that the method is well suited for time series forecasting tasks. The advantage of using a decision tree is clear, and the performance of the forecasting is significantly improved. With more advanced forecasting methods, it will be possible to improve the performance even further. The method is able to adapt to different problems, and the performance can be enhanced 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 the process is iterated until a desired number of data points are forecasted. It is easy to train the algorithm to use an approximation function to forecast more than one data point if the problem desires this (long-term forecasting). The forecasting quality of the method increases with more available past data points, more available features and more test and training samples, especially when the training data are very noisy. In other words, the method can be enhanced for better forecasting.

11 PAPER 6: GENETIC ALGORITHMS AND DECISION TREES FOR CONDITION MONITORING AND PROGNOSIS OF A320 AIRCRAFT AIR CONDITIONING

11.1 Introduction

Unscheduled maintenance can be very expensive for airlines (Institut du Transport Aerien, 2000) (Eurocontrol, 2006) (Cook, et al., 2004) and should be avoided. One way to prevent unscheduled maintenance is to predict a future failure and perform maintenance actions during line maintenance before the failure occurs. Condition monitoring can be used to detect the current system health status and to predict the future health status. Condition monitoring is widely used for, among others, rotary machines (gear boxes, gas and wind turbines, bearings), plants and structures (bridges, pipelines).

Vibration data are frequently used as the basis for condition monitoring. Goode, Moore and Roylance (2000) show how a basic life prediction method works using a simple system with alarm settings. An alarm is triggered if a vibration signal exceeds the alarm limit. Mahamad, Saon and Hiyama (2010) present a method to predict the remaining useful life of rotary machines using artificial neural networks. Using the vibration signal as input data, they show how RUL is important for condition based maintenance. Saravanan and Ramachandran [6] use wavelet features and decision trees for fault diagnosis. They use the same decision tree algorithm as we do in this paper but a different set of features. Sugumaran and Ramachandran (2011) use also the same decision tree algorithm, as well as the decision tree for feature selection, but they convert the resulting decision tree into a set of fuzzy rules to use for classification.

Condition monitoring is difficult to use in aircraft maintenance because certification issues create certain restrictions. Aircraft maintenance is based on Reliability Centred Maintenance (RCM). The goal is to have maximum safety and reliability with minimized costs. To meet this goal, the Maintenance Steering Group (MSG) has developed maintenance concepts specifically for aircraft. The most recent is MSG-3 (Federal Aviation Administration, 2012). Its focus is the effect of a failure on the aircraft operation (Nowlan & Heap, 1978) (Air Transport Association of America, 2007). For each item that effects airworthiness, it describes a specific maintenance task (task-oriented maintenance). MSG-3 can use condition based maintenance (CBM), defined as "*preventive maintenance which includes a combination of condition monitoring and/or inspection and/or testing, analysis and the ensuing maintenance actions*" (European Committee for Standardization, 2015) or predetermined maintenance (PM), defined as *"preventive maintenance*

carried out in accordance with established intervals of time or number of units of use but without previous condition investigation" (European Committee for Standardization, 2015) to achieve its goals. Predetermined maintenance is used by most airlines and manufacturers. Preventive maintenance with scheduled maintenance provides benefits not only for cost control but also for reliability (Kiyak, 2012). Condition based maintenance (CBM) is based on condition monitoring and aims at preforming maintenance based on the inspected system's condition and the trend of its condition. CBM can be used to realize RCM (Niu & Pecht, 2009) using condition monitoring.

Condition monitoring was introduced to EN 13306 in 2010 (European Committee for Standardization, 2010). Before 2010, only CBM and monitoring were included in EN 13306 (European Committee for Standardization, 2001). In other words, condition monitoring is a recent development; as it matures, however, it is slowly taking hold in a number of industries.

Aircraft maintenance procedures need to reflect the times and draw on current technologies. This paper suggests an approach to include modern technologies in aircraft maintenance by using a mixture of continuous and discrete input signals to predict the condition of an aircraft system. More specifically, it uses decision trees (Cernak, 2010) (Quinlan, 1986) to approximate a system model based on recorded aircraft data. The goal is to predict the remaining useful life (RUL) of the system by forecasting the system health condition. It uses a data-driven black box model (only the input of the system is known) based on sensor data over an extended period of time to create a system model represented by a decision tree forest. The novelty of the methodology is the use of an artificial intelligence method and a simple noise resistant prediction method in a restricted commercial aircraft area. The data used for the validation consist of recorded aircraft data available in most aircraft and contain a great deal of noise. The methods and operations were selected with certification and on-line monitoring in mind. The advantages of this approach are that it can be used on-ground or on-aircraft, and real-time monitoring of a single aircraft or a fleet is possible without the installation of additional hardware.

The structure of the paper is as follows. This section gives background information on the methods and technologies used to develop the proposed method. The next section explains the method in detail, and the following one validates the method with real world data from an in-service aircraft. The paper ends with a conclusion and offers some final comments.

11.1.1 Condition Monitoring

Condition monitoring is defined as (European Committee for Standardization, 2015):

"Activity, performed either manually or automatically, intended to measure at predetermined intervals the characteristics and parameters of the actual state of an item"

It is based on three steps (Jardine, et al., 2006):

1. **Data acquisition**: Collecting and storing data from physical assets. This includes event data and condition data. Event data are what happened and what the condition data represent.
2. **Data processing**: The first step of data processing is data cleaning followed by data analysis. Data analysis includes transformation of data from the time domain into the frequency domain and feature extraction.
3. **Maintenance decision-making**

Condition monitoring can either be continuous or periodic (Jardine, et al., 2006). Continuous monitoring (automatic) is often performed by installed sensors and automatically by machines. Periodic monitoring (manual or automatic) can be done by humans and can include checks at regular maintenance intervals. Implementing condition monitoring is a difficult and costly task.

Diagnosis and prediction are two goals of condition monitoring (Jardine, et al., 2006). Diagnosis (posterior event) deals with detection (Did something fail?), isolation (What failed?) and identification (Why did it fail?) of faults when they occur and is defined as (European Committee for Standardization, 2001):

"Actions taken for fault recognition, fault localization and cause identification".

Prognostic (prior event) deals with predicting future faults and how soon they will occur (Jardine, et al., 2006). There are two types of prediction: prediction of remaining time until a failure occurs and prediction of the chance a machine will operate without a fault until the next scheduled maintenance (Jardine, et al., 2006).

Condition prediction can be done based on sensor data or on system condition (Mobley, 2002). The analysis algorithm looks not just at recorded parameters at a single moment in time; it also takes the full parameter history into account. The need for maintenance of the component is indicated if the data trend of parameters points to a degradation of the component. Based on the parameter time history, the analysis algorithm allows a forecast of the remaining lifetime of the component (Kolerus & Wassermann, 2011). Analysis and prediction use a variety of methods to predict future values. Physics-based approaches for prediction include Kalman filter, sequential

Monte Carlo, Markov models and others. Data-driven approaches include artificial ARMA (autoregressive-moving average), ARIMA (autoregressive integrated moving average), neural-networks, Bayesian networks and others. All these methods can be used to predict values for a complex time series (Chen, et al., 2011) (Caesarendra, et al., 2010) (Pham & Yang, 2010) (Tian, et al., 2010). Output of the prediction is normally an estimated time to failure (ETTF) and a confidence interval (Sikorska, et al., 2011). The confidence interval defines how reliable a prediction is (Schruben, 1983) (Sikorska, et al., 2011) and can be calculated using standard time series.

The condition of the system is often defined by setting limits on certain values based on experience (knowledge based) (Mobley, 2002) or by creating a mathematical representation of the physical system model or a data-driven model (Kolerus & Wassermann, 2011) (Williams, et al., 1994) (Zhang, 2006). Other methods include machine learning techniques, such as decision trees (Sugumaran, et al., 2007) (Sugumaran & Ramachandran, 2007) (Tran, et al., 2009), vector support machines (Pham, et al., 2012) (Sugumaran, et al., 2007) (Widodo & Yang, 2007) and neural networks (Chen, et al., 2012) (Mahamad, et al., 2010) (Tian, 2012), to map the features of the input signal to a condition.

Another option is to use a physics-based model (representing the system's physical features, components and interactions), feeding the sensor input into the model, calculating the output and checking how the output of the theoretical model deviates from the real system. This approach can be used for fault isolation and fault identification of failures in addition to prognosis (Wang, et al., 2008) (Williams, et al., 1994) (Kolerus & Wassermann, 2011) (Jardine, et al., 2006).

Data-driven models use past data to create models with stochastically or machine learning algorithms (Pecht, 2008) (Garcia, et al., 2006) (Jardine, et al., 2006). These models require many data samples representing different conditions of the system. Data-driven models require less "manpower" than a mathematical model; model validation and testing can be performed almost automatically.

All three techniques are widely used and often are combined to perform more accurate diagnosis and prognosis.

11.1.2 Feature Extraction

Feature extraction is the process of reducing the dimension of the initial input data to a feature set of a lower dimension that contains most of the significant information of the original data (Fonollosa, et al., 2013). This is done to extract features from noisy sensor data (Lin & Qu, 2000);

(Fu, 2011) and to avoid problems caused by having too many input features (especially for vibration data) for the classifier learning phase (Yen & Lin, 2000). Feature extraction is often a first and essential step for any classification (Yen & Lin, 2000).

Methods of feature extraction include extracting features from the time domain and the frequency domain (Fourier transformation, wavelet transformation (Fu, 2011)) and clustering, if necessary. Basic features include maximum, mean, minimum, peak, peak-to-peak interval etc. (Jardine, et al., 2006). Complex feature extraction methods include principal component analysis (PCA), independent component analysis (ICA) and kernel principal component analysis (KPCA) (Widodo & Yang, 2007). Other feature extraction methods are: t-test, correlation matrix, stepwise regression and factor analysis (FA) (Tsai, 2009). A comparison of the various feature extraction methods appears in Arauzo-Azofra et al. (Arauzo-Azofra, et al., 2011).

Selecting relevant features for classifiers is important for a variety of reasons, such as generalization performance, computational efficiency and feature interpretability (Nguyen & De la Torre, 2010). Using all available features can result in over-fitting and bad predictions, but it is not possible to look at each feature alone because many features are inter-correlated (Meiri & Zahavi, 2006). Noise, irrelevant features or redundant features complicate the selection of features even more. Thus, features are often selected using methods from pattern recognition or heuristic optimization, or a combination. Sugumaran et al. (2007) show how different technologies can be combined for a single goal; they use a decision tree for feature selection and a proximal support vector machine for classification. Widodo and Yang (2007) combine ICA/PCA with SVM for feature extraction and classification. Many algorithms combine genetic algorithms (GA) with a pattern recognition method, like decision trees (DT), SVM or artificial neural networks (ANN). In these combinations, GA is used to optimize the process parameter (Samanta, et al., 2003) (Huang & Wang, 2006) or for feature extraction and the pattern recognition required for classification (Samanta, 2004) (Saxena & Saad, 2007) (Jack & Nandi, 2002) (Samanta, 2004)

11.1.2.1 Time Domain Features

Time domain features can be direct features, such as the number of peaks, zero-crossings, mean amplitude, maximum amplitude, minimum amplitude or peak-to-peak intervals (Jardine, et al., 2006) (Pascual, 2015). In addition, it is possible to analyse a signal using probabilistic moments like root mean square, variance, skewness or kurtosis to get features that represent the signal (Lambrou, et al., 1998). Other methods include using correlation, autocorrelation, entropy, PCA, ICA and KPCA (Widodo & Yang, 2007).

11.1.2.2 Frequency and Time-Frequency Domain

Fast Fourier transformation (FFT) transforms a signal from the time domain into the frequency domain. Specifically, FFT takes a time series and transforms it into a complex vector that represents the frequency power in frequency domain. The basis of the FFT algorithm is the discrete Fourier transformation (DFT), defined in Equation (61) with $x_n \dots x_{n-1}$ as complex numbers.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}} \quad k = 0, \dots, N-1 \tag{61}$$

FFT is performed in *O(N log N)* time (Ohm & Lüke, 2010) and can be calculated in real time because it can be executed in parallel. It is a widely used and well-established method (Peng, et al., 2002); (Fu, 2011). Recent research uses the discrete wavelet transformation (DWT) to represent time series in the frequency domain. The DWT represents the time series in a time-scale form (Jardine, et al., 2006) and is especially suited to represent non-stationary signals (Lin & Qu, 2000).

Failure diagnosis mostly focuses on the frequency domain, e.g. using Fourier transform or wavelet transform, but in the early stage of failure development, the damage is not significant and the defect signal is masked by the noise in the acquired signal. The periodicity of the signal is not significant. Although spectral analysis may not be effective, using the time domain feature is recommended. Normal and defect signals differ in their statistical characteristics in the time domain, so the combined use of time domain features and features of other domains can improve diagnosis accuracy.

11.2 **Method**

The method proposed in this paper for condition monitoring (CM) is based on decision trees (DTs). The inputs for the DTs are features (e.g. maximum value, mean values, frequency domain data; see Table 47) extracted from processed sensor data; the output is the current health condition in ten percent steps of the remaining useful life (RUL) (see Figure 98 and Figure 99). The health condition is used to forecast the RUL. The quality of the monitoring is increased by using a genetic algorithm (Russell & Norvig, 2003) to optimize the process parameters.

The method is divided into a training process and a classification process. The goal of the former is to create a set of optimal decision trees to classify the system condition. The goal of the latter is to classify a new sensor data sample in the current system condition and use the system condition history to make a prediction about the remaining RUL.

The novelty of the approach is the discretisation of the system's condition into ten categories and the use of the condition changes to create a time series for condition prediction (see Galar et al., 2012, for a similar approach using support vector machines).

The method adds the input from multiple mixed sensors (Boolean and continuous) to detect the system's health condition. The goal is to prevent unscheduled maintenance operations by estimating the RUL of the monitored system. The goal is not to predict the future time series of a sensor as well as possible. Nor will the method output what failures will happen; it will only show that a failure will happen at a certain point in time. Ideally, it will avoid the "better mousetrap symptom" (reinventing a common product with some improvements and higher costs) and will be applicable to more than one specific system. Jack and Nandi (2000) show that using a genetic algorithm to select features from a given set can significantly improve the accuracy of a classifier.

11.2.1 Training Process

The training process consists of three steps: system data acquisition, setup and optimization loop (see Figure 95). The first step is to record sensor data for the training process. The recorded data should contain all system conditions (from newly installed to broken) under different conditions (different environments, different aircraft). Any performed maintenance actions need to be recorded as well. Input data can be Boolean data (switches, valves etc.) or continuous data, like temperature and pressure data. Discrete input data are mapped to continuous data with the same frequency. All input data sources must have the same sampling frequency, but the source does not matter; data can be sound, vibration, temperature power consumption, weight or magnetic flow.

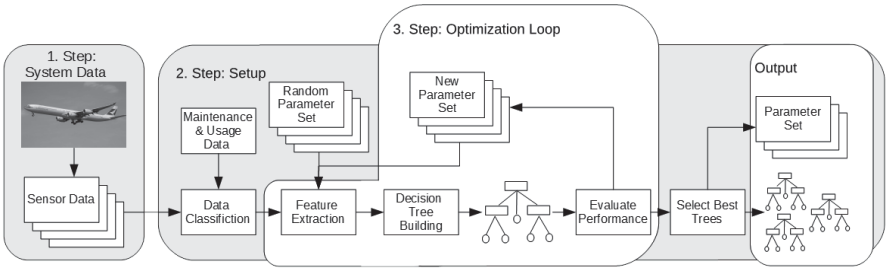

Figure 95: Classification training process

The second step is to prepare the data for the actual training process. First, the recorded data samples need to be correctly labelled and classified because the decision tree training is a supervised learning process. The data samples are labelled with the system condition, represented by the data in ten percent groups ($x < 10\%$ RUL; $10\% \leq x < 20\%$ RUL etc.). We predefined ten categories to give a good estimation of the remaining useful life and to ensure a

wide range of samples per category. Fewer classes will reduce the usefulness of the method for the operator, while more classes mean more misclassifications because more samples are likely be close to two classes.

Meta data, like maintenance actions, aircraft altitude and flight phase, need to be considered when labelling the data samples. The samples should present a stable system state, similar across different aircraft. In this case, we used the cruise flight phase; the aircraft is in a stable system condition with little stress and a condition that will not be changed for some time. This step also includes the generation of a random parameter set for the feature extraction as a starting point for the optimization loop.

The main work happens in the third step when multiple feature extraction steps are executed and multiple decision trees are built.

The task of the optimization loop is to modify the parameters for the feature extraction to improve the accuracy of the classification with a decision tree. A genetic algorithm with basic operations (crossover, mutation (Russell & Norvig, 2003)) is used to vary the parameters. The "fitness" of a parameter set is measured using the classification accuracy of the decision tree built using this parameter set. Genetic algorithms have been used for many optimization problems (Golub & Posavec, 1997) (Jack & Nandi, 2000) (Stein, et al., 2005) and can be executed in parallel. The first step is to create many different sets of training input vectors by applying feature extraction operations to the labelled data samples. These feature extraction operations are controlled by the previously created parameter sets. The second step of the optimization loop is to create a decision tree with the input vector representing the current feature set.

11.2.1.1 Feature Extraction

Having a set of elemental feature extraction operations controlled by a parameter means the operations can be executed in any order, and the feature extraction algorithm can be adapted for multiple problems (Mierswa & Morik, 2005) by using a genetic algorithm for optimization without human interaction. In Gerdes and Scholz (Gerdes & Scholz, 2011), signal analysis and machine learning are used to detect the condition of an experimental setup. The concept is almost automated and but requires minimal fine-tuning by hand because the process depends on several different parameters, each of which must be adapted to the data for optimal classification. These parameters include the following:

Parameter Name	Possible Values
Block width	0 – 1000 (Hz)
Use blocks	Boolean
Mean amplitude for each block	Boolean
Maximum amplitude for each block	Boolean
Mean frequency power for each block	Boolean
Maximum frequency power for each block	Boolean
Number of peaks for each block	Boolean
Minimum value of a peak	0 – 5
Overall mean and maximum values	Boolean
Confidence factor	0.00001 - 0.5

Table 47: Data processing parameters

Block width defines how many frequencies are grouped in the frequency domain to form a block for detailed feature extraction. **Noise reduction factor** defines how much noise will be reduced. The noise reduction in this concept removes all frequencies wherein power is below *noise reduction factor* times *mean power.* Equation 62 shows how noise is defined for this method.

$$Frequency\ Power\ < Noise\ Reduction\ Factor \cdot Mean\ Power \tag{62}$$

Minimum value of a peak controls what frequencies are defined as peaks. It is the opposite of noise. Any frequency where power is greater than or equal to the *peak border* times the *mean power* is defined as a peak, as shown in Equation (63.

$$Frequecy\ Power\ \geq Peak\ Border \cdot Mean\ Power \tag{63}$$

Confidence factor controls the extent of tree pruning and is a parameter of the J48 algorithm of the WEKA software (Waikato, 2009). A confidence factor greater than 0.5 means no pruning is done. The lower the confidence factor, the more pruning.

All other parameters are Boolean; they control whether a given feature is calculated or not. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations that can be different for each problem (Mierswa & Morik,

2005). This makes elementary extraction operations good for machine learning. The operations are also fast to compute and can be used for online monitoring. The data from the various sensors are not merged at the sensor level but at the feature extraction level. A feature set is calculated for each input from each sensor. These features are merged into one feature input vector for the decision tree learning phase. No frequency features are calculated for signals that are nearly constant (Boolean switches, discrete system settings, certain process parameters).

The selection of features is determined by the parameters in Table 47. The values for the parameters are randomly generated or generated during optimization using a search algorithm. We use these quite basic feature extraction operations to ensure that, even with simple hardware, a near real time monitoring is possible.

11.2.1.2 Building a Decision Tree

Decision trees are used in the area of artificial intelligence for decision making and machine learning. They are often binary; each node has an if-then-else function on an attribute of the sample data. The ID3 algorithm (Iterative Dichotomiser 3) (Quinlan, 1986) was the first algorithm to construct decision trees. ID3 had some problems and was improved in C4.5 (Quinlan, 1993). The revised algorithm has the ability to handle both discrete and continuous attributes, to handle samples with missing attributes and to support pruning of the tree. It uses the concept of information gain and information entropy to choose attributes from the data and build a decision tree. The result is a binary decision tree, whose root is the attribute with the highest normalized information gain. Nodes in subsequent levels of the tree represent attributes with lower normalized information gain. Decision trees are used to solve a large variety of problems, e.g. tag speech parts (Schmid, 1994), land cover mapping (Friedl & Brodley, 1997) and text mining (Apte, et al., 1998).

In our experiments, we used the C4.5 algorithm, specifically the open source implantation J48 from WEKA (Waikato, 2009) software. It supports continuous values and pruning and is very well understood.

The next step is to evaluate the fitness of the decision tree by checking how many samples are correctly classified. This checking is done by classifying a test data set. The test data set can contain data samples used to build the decision tree, but preferably, the test set should be disjunctive from the training set.

A new parameter set for the feature extraction is created using the genetic algorithm, and a new input vector for the next iteration is created. The optimization loop is executed until a given accuracy is achieved or a set number of iterations is performed.

11.2.2 Classification and Prediction

The process for predicting condition (Figure 96) is based on the results of continuous condition monitoring and classification. Each classification adds a new data sample to the class time series. The process has three steps: the first is recording new sensor data; the second is classifying the sensor data and adding the classification result to a time series; the third is marking when the system condition switches from one state to another and using these data points to extrapolate the data into the future.

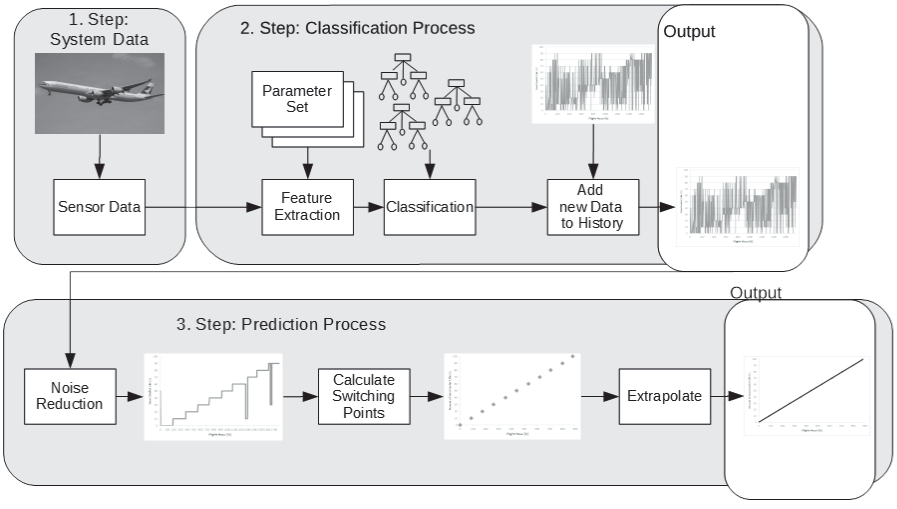

Figure 96: Classification and prediction process

Having a network of different sensors to monitor a system can cause sensor data fusion. Multi-sensor data fusion prevents the problem of combining sensor data from different sources into one consistent model, but the main questions of sensor fusion are (Basir & Yuan, 2007):

- How to get accurate and reliable information from multiple and possibly redundant sensors;
- How to fuse multi-sensor data with imprecise and conflicting data.

Techniques for sensor fusion can be grouped into the following levels (Jardine, et al., 2006) (Ross & Jain, 2003) (Castanedo, 2013):

- Data-level fusion, e.g. combining sensor data from same sensors directly (Lu & Michaels, 2009);
- Feature-level fusion, e.g. combining vectors and feature reduction techniques (Ross & Jain, 2003);
- Decision-level fusion, e.g. vote schemes (Ross & Jain, 2003).

The use of multiple decision trees has been shown to improve the overall accuracy and robustness of predictions (Zhang, 2006). The data of the decision trees are fused onto the decision level using a voting method (Ross & Jain, 2003). If the majority of the trees classifies a data sample as being in a certain class, this class is selected. If all trees get a different result, the result of the tree with the highest classification accuracy for the training data is used.

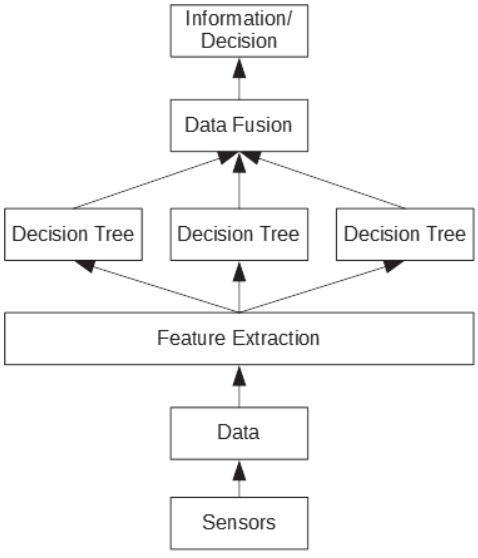

Figure 97: Condition monitoring with multiple trees (Zaher & McArthur, 2007)

The proposed method uses the three best performing decision trees. Only three are selected to allow real time condition monitoring while increasing the accuracy in a noticeable way (Gerdes & Scholz, 2011). A new data sample is taken and processed according to the feature extraction parameter set of each tree. The sample is classified using a voting process among the three

decision trees (Figure 97). If two or more trees classify a data sample as the same class, this class is selected. If all three trees get a different result, the result of the first tree is taken.

The resulting time series is subject to noise in form of erroneous classifications (Figure 98). If there are no errors, the curve should look like a set of stairs. To reduce the wrong classifications, each data point is set to the class of 20 of its neighbours (this value may change depending on the noise in the time series) (Figure 99). This means that for each data point in Figure 98, 20 neighbouring data points are taken; the current data point takes the value of the class with the most members. Each noise reduced data point is now added to a new vector. The new classification vector is shown in Figure 99.

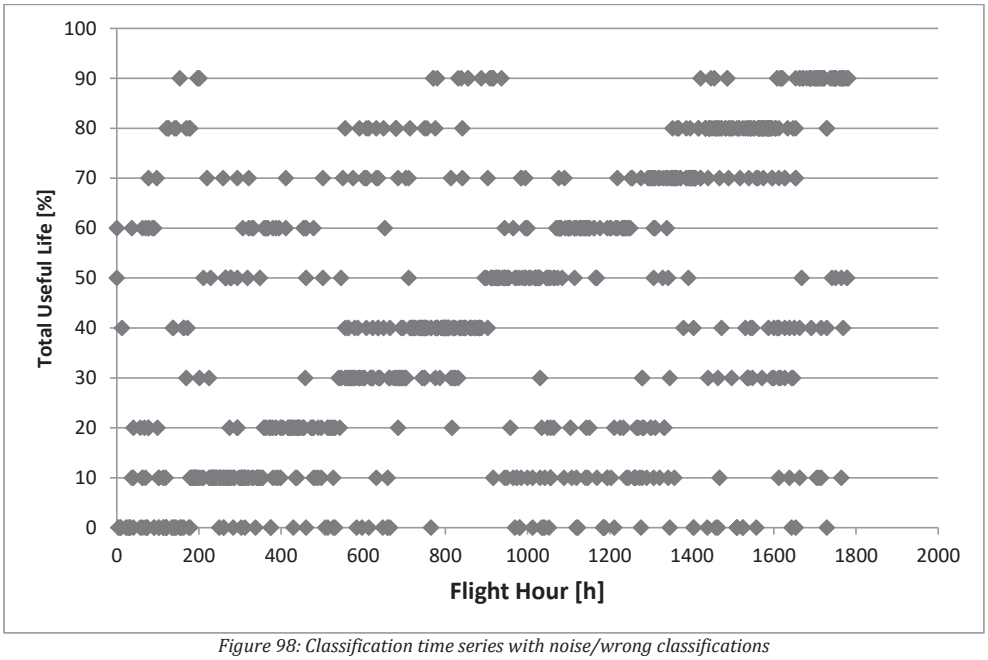

Figure 98: Classification time series with noise/wrong classifications

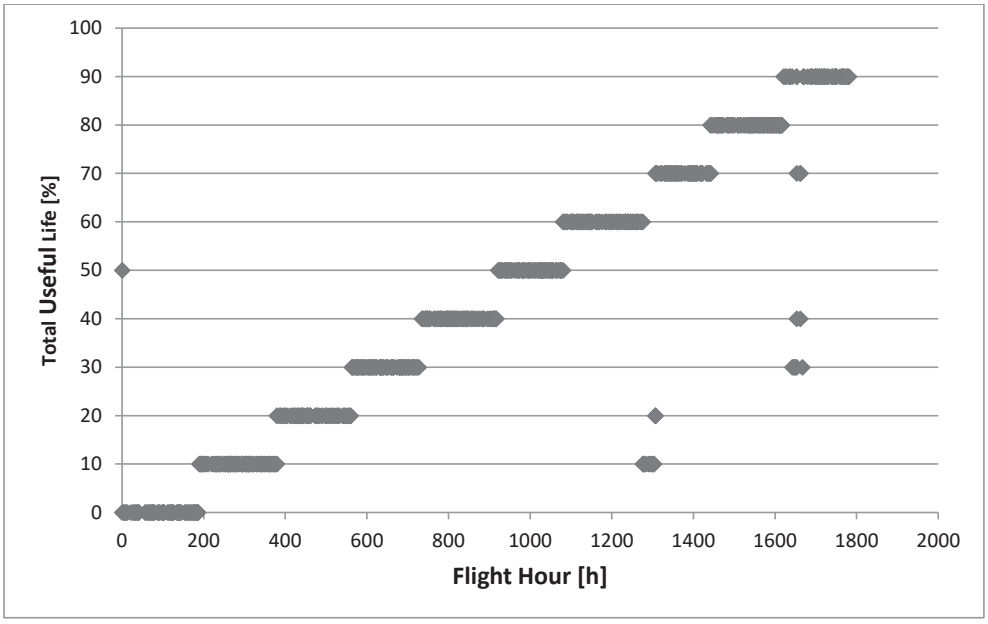

Figure 99: Classification time series with applied noise reduction

Condition prediction uses past condition monitoring data to predict the RUL of the monitored system by taking the first dot after each "health condition class jump" (jumps in Figure 99). Ideally, this plot should be a linear function, with the health condition changes equally spaced (see Figure 100). Maintenance action can alter the gradient of the function and/or introduce a discontinuity. In such cases, the prediction needs to be restarted from maintenance action data point.

Depending on the usage of the system and the operation environment, the health condition changes may not be equally spaced. Such spacing indicates a change in the degradation and, thus, in the RUL. Prediction is possible when two or more state changes have been detected, however. Because the plot does not have more than 11 data points, it is possible to use a simple approximation method. The classification rules (the rules determining in which RUL class a sample belongs) are automatically generated by the samples used to train the decision tree. The threshold for an RUL of zero is gained by extrapolating the already classified samples (see Figure 100). This means the RUL is based on experience, not on physical models or aircraft regulations (maximum number of flight hours).

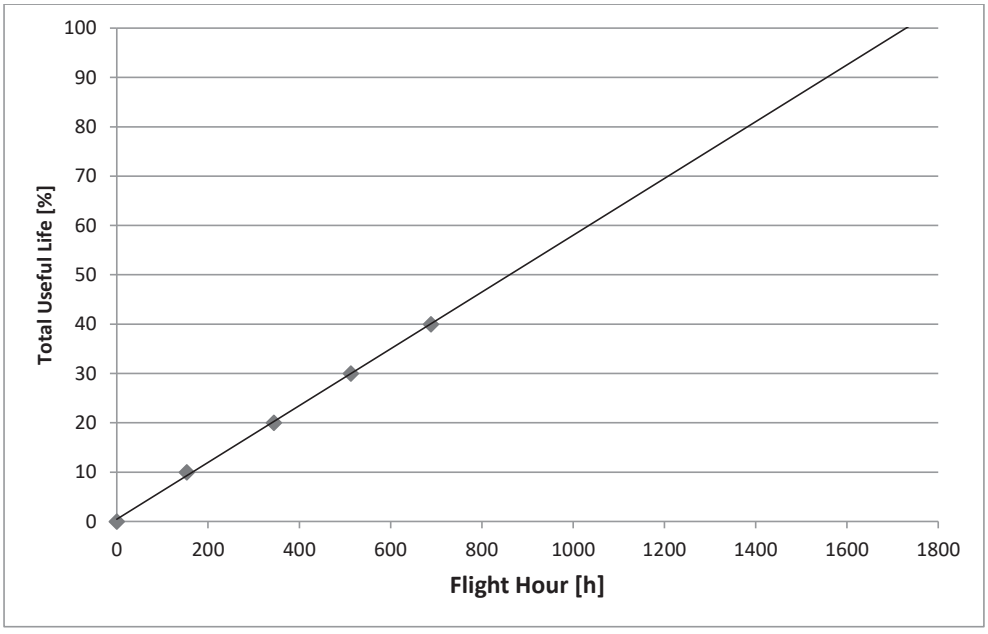

Figure 100: Remaining useful life prediction

11.3 **Validation**

We validated the proposed method using sensor data from the air conditioning system of an A320 aircraft operated by ETIHAD Airways in the Middle East. The sensor data were from 589 flights over two years. Each sensor reading included over 80 values, consisting of continuous and Boolean data. The data were sampled with a frequency of 1 Hz. Table 48 lists the sensor data.

Description	Bus	Type
Cabin Compartment Temperature Group 1	Zone Control	Numerical
Cabin Compartment Temperature Group 2	Zone Control	Numerical
Cabin Compartment Temperature Group 3	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 1	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 2	Zone Control	Numerical
Cabin Temperature Regulation Valve Position Group 3	Zone Control	Numerical
Duct Overheat Warning Group 1	Zone Control	Boolean
Duct Overheat Warning Group 2	Zone Control	Boolean
Duct Overheat Warning Group 3	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 1	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 2	Zone Control	Boolean
Duct Temperature 4 Times Limit Exceedance Group 3	Zone Control	Boolean
Duct Temperature Group 1	Zone Control	Numerical
Duct Temperature Group 2	Zone Control	Numerical
Duct Temperature Group 3	Zone Control	Numerical
G + T Fan OFF	Zone Control	Boolean
Hot Air Switch Position ON	Zone Control	Boolean
Minimum Bleed Air Pressure Demand	Zone Control	Numerical
Nacell Anti Ice - Engine 1	Zone Control	Boolean
Nacell Anti Ice - Engine 2	Zone Control	Boolean
Recirculation Fan Left Hand Fault	Zone Control	Boolean
Recirculation Fan Right Hand Fault	Zone Control	Boolean
Trim Air Pressure Regulation Valve Disagree	Zone Control	Boolean

Trim Air Pressure High	Zone Control	Boolean
Trim Air Pressure Regulation Valve Close	Zone Control	Boolean
Trim Air System Inoperational	Zone Control	Boolean
Zone Main Control Inoperational	Zone Control	Boolean
Zone Secondary Control Inoperational	Zone Control	Boolean

Table 48: A320 sensor data description

Description refers to the name of the sensor, bus indicates from which system data are taken (Air Data Computer, Flight Data Interface Unit, Inertial Reference System, Zone Control, Bleed Monitoring Computer, Pack Control) and type indicates if data are Boolean or continuous. Zone Control, Bleed Monitoring Computer and Pack Control buses are directly related to the air conditioning system. The Bleed Monitoring Computer bus monitors the input to the system (56 sensor values), the Pack Control bus has data directly related to the monitored system (8 sensor values) and the Zone Control bus contains sensor data that monitor the output of the system (28 sensor values). Data from the Air Data Computer, Flight Data Interface Unit and Inertial Reference System bus concern the position and environment of the aircraft (air temperature, time). As the table shows, the data contain failure states (e.g. Trim Air System Inoperational, Boolean value), switches (e.g. Hot Air Switch Position ON, Boolean value) and direct output data (e.g. Cabin Compartment Temperature Group 1, numerical value).

We generated samples for the decision tree by taking 1024 samples of sensor data after the aircraft has reached a height of 30,000 ft. This equals 17 minutes of data in cruise flight and yields about 3400 data samples.

Samples for the training and test set were randomly taken from the flight data with a 25% chance of preventing the training data from overlapping with the test data and ensuring the number of samples per class is not equally distributed. We divided the data samples into ten categories. The first included all data samples with a time stamp lower than 10% of the RUL of the system, and the last contained samples with a timestamp between 90% and 100% of the RUL. All categories in between were split equally, each covering a 10% range of RUL.

We fed data samples into the condition monitoring process and classified them, generating a time series with the classifications and a list with the sample numbers where the system's condition has changed (see Figure 49). We used a DT forest of three trees to classify the samples.

Data analysis shows a maintenance action after approximately 840 flight hours (FH). The nature of the maintenance action is unknown, but the sample classifications show the data samples after 840 FH are similar to those at the beginning of the data recording. This indicates the maintenance action reset the system condition and hints at a major maintenance action.

Table 49 shows the misclassification of samples if it is assumed there was no maintenance action. The green marked entries are the correct classifications. On the left and right are misclassifications that are very similar to the correct ones. Each entry in the table is a set of “X sample of class Y classified as class Z”. Class Y is the left-most column, and class Z is the top row of the table. X is the number where Y and Z intersect. For example, “19 samples of class 10 are classified as class 0”, or “49 samples of class 50 are classified as class 10”.

Many misclassifications are clustered. This is visible in Figure 98 and Figure 99, where the classified class is often the real class +/- “50” (RUL, class label). It is also apparent that two groups of misclassifications are parallel to the correct classifications. The misclassifications in these groups indicate class "50" is similar to class "10", class "60" is similar to class "20", class "70" is similar to class "30" etc. That means after class "40", the health condition is very similar to the beginning of the time series, suggesting something reset the system health condition.

Table 50 shows the misclassifications after it is assumed that at flight hour 840, the system health condition was reset. Now the misclassifications are neighbours of the correct classification; this is a good sign, because neighbouring classes should have similar features. It also shows that the assumption of the maintenance action at 840FH is correct.

Classification/ Class	0	10	20	30	40	50	60	70	80	90
0	0	36	24	23	10	24	29	16	28	6
10	19	0	40	3	4	45	21	12	1	17
20	16	38	0	4	0	29	68	9	0	0
30	28	0	3	0	37	8	6	12	72	4
40	16	6	0	45	0	9	0	20	38	29
50	21	49	28	0	4	0	36	9	0	32
60	30	48	35	1	0	42	0	7	0	0
70	31	30	11	38	20	21	32	0	29	6
80	29	0	0	67	55	0	0	25	0	13
90	23	20	0	10	46	23	0	6	17	0

Table 49: Misclassification matrix without maintenance action

Classification/ Class	0	10	20	30	40	50	60	70	80	90
0	0	21	32	21	7	8	9	6	7	47
10	22	0	32	25	9	23	18	21	10	24
20	23	19	0	36	36	8	17	6	2	25
30	26	49	35	0	54	15	16	0	0	0
40	31	4	29	32	0	36	8	0	0	0
50	16	15	28	41	34	0	20	0	0	0
60	19	22	15	16	5	10	0	10	27	23
70	14	21	15	0	0	0	51	0	25	54
80	2	5	6	0	0	0	68	43	0	73
90	30	28	29	0	0	0	33	48	46	0

Table 50: Misclassification matrix with maintenance action

This example suggests the condition monitoring process needs to be implemented into Computer Maintenance Management Systems (CMMSs), preventive maintenance (PM) scheduling, automatic work order generation, maintenance inventory control and data integrity (Van Horenbeek, et al., 2012) to get all needed information to make correct decisions. If the condition

monitoring system gets no feedback on performed maintenance actions, it may learn wrong classification rules or make incorrect classifications (Galar, et al., 2012).

With fuzzy decision tree evaluation (Gerdes & Scholz, 2011), it is also possible to see the second most likely classification of a data sample. Figure 101 shows the results if no maintenance action is considered. Together with Figure 98, this validates the results shown in Table 49 and the assumption of the occurrence of a maintenance action.

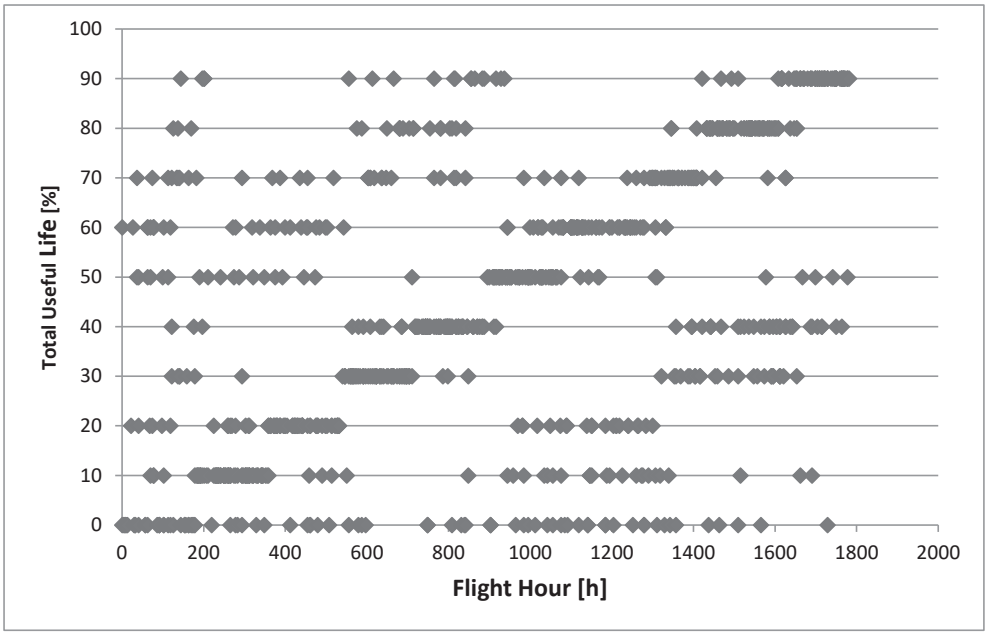

Figure 101: Second most likely class

The maintenance action during the duration of the recorded data was assumed to be the end of the lifetime, and the first data point the beginning, because no data above the whole life of the system were available. However, this should be sufficient to validate the concept. Recall that the RUL of the system was set to 840 FH, because of the maintenance action around FH 840. All samples between 0 FH and 840 FH were slotted into the 10 classes; the same was done for the samples between 840 FH and 1700 FH. This simulated two total lifetimes of the system.

The process was able to classify the randomly taken data sample correctly, with an error rate of 346 wrong samples from a total of 835 samples, or 41%. This large error rate was due to the high noise in the data and the lack of perfect division of the data into two time series (two total life times). It was difficult to reduce the noise at the source level, because we were using direct data

from the aircraft, but it was possible to reduce the noise before the feature extraction step by applying noise reduction. Noise reduction needs to be applied carefully so that no significant features (especially for the Boolean data sources) are removed. For this reason, we did not perform noise reduction for each channel; instead, we performed noise reduction on the results, because without maintenance, the RUL can only increase. The wrong classifications were spread over nine classes; this allowed correct classification because it indicated the most common class over a range of 20 classifications (three flights or 10 FH) in the time series. Note that the misclassifications were mostly caused by samples close to the border of a class and wrongly classified as the neighbouring class (see Table 50).

The resulting condition time series is shown in Figure 102. In the figure, the points at which the current system condition switches are nearly equally spaced, even with many misclassifications in the data source, and deviate very little from the correct data points.

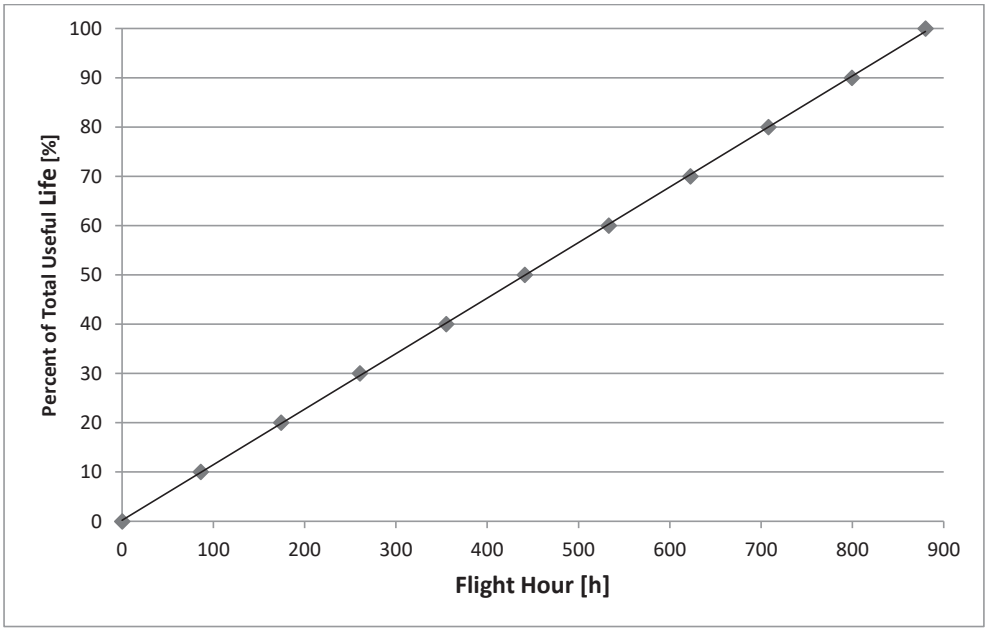

Figure 102: Start of different system health conditions

Table 51 shows how the error rate improved when more training samples were used for testing. The table shows the number of samples selected for training or testing (out of 3400 samples), the method of sample selection (random is self-explanatory; separation means training samples and testing samples don't mix), the error rate of the classification and a list of the detected condition changes. There is an overlap/intersection of training and testing data in all cases.

Number of Samples	Method	Error Rate	RUL -> FH
850 Training 850 Testing	Random	41%	72;163;264;70;434;532;620;702;808;840
850 Training 3400 Testing	Random	43%	85;39;259;341;429;4;169;688;196;840
1700 Training 1700 Testing	Random	23%	86;176;263;357;443;535;623;708;800;840
1700 Training 1700 Testing	Separation	40%	88;172;264;282;443;528;623;711;800;840
1700 Training 3400 Testing	Random	24%	72;176;266;357;441;537;623;711;802;840
3400 Training 3400 Testing	-	7%	88;174;264;355;441;532;620;708;800;840

Table 51: Different splitting of training and test data

Table 51 shows the classification error rate is about 40% for classifications where the training and testing data are separated or where is little overlap. The accuracy increases as soon as the overlap between training and test data increases. This behaviour is to be expected; it shows the algorithm and data are prone to overfitting, something common to many experiments. It also shows that beside a few outliers in the time series, all cases detect nearly the same condition changes in the system. The error rate of 40% looks very high, even with noise in the input data. However, the high error rate can be explained by the nature of the air conditioning system and the input data. The air conditioning system is a large distributed system constructed of active and passive subsystems. The subsystems are one “main” unit: the air cycle machine, multiple fans, filters, valves and tubes. The sensor data used for validation are the input and output data of the system; no sensor data representing the internal state are used. Finally, most of the sensor data are Boolean signals representing valves, switches or failure states.

11.4 Conclusion

Condition monitoring and prognostics are complex processes, difficult to establish in the aircraft environment. The method proposed in this paper can be used to classify a system’s health condition and forecast its remaining useful life. The novelty of this approach is that it offers a

simple method for condition monitoring and prediction that can be used in the restricted aircraft environment.

The method uses decision trees to generate simple logical clauses; these can be checked and certified with little difficulty. It can be used on-ground to monitor all aircraft that submit sensor data without the need to install additional sensors or hard- or software. This makes the method very interesting for fleet health monitoring applications. Finally, the relatively simple feature extraction, classification and prediction are handy when there is a need to monitor multiple aircraft with multiple systems in real time.

Classification and prediction are possible with this method, even with noisy real-world data and with no additional information about the system being monitored. It can be used for existing systems that already log events, process parameters and input/output data, but do not have a condition monitoring and prediction system. The method is generally applicable; it can use any available input data from any system because of the use of mixed data input channels (discrete and continuous data). One caveat is the need to combine CMMS and CM to avoid too many misclassifications caused by unknown maintenance actions in the training phase.

The validation process shows RUL can be forecast after 400 FH. This makes it possible to plan maintenance actions for aircraft in advance and avoid unscheduled maintenance. In fact, forecasting is possible after one condition change is detected (after about 90 FH), but if several conditions change, the forecasting accuracy improves.

Despite its efficacy, the method can be improved in future research. The accuracy of the individual decision trees and, thus, the overall accuracy can be improved by using advanced feature extraction methods, like wavelet package transformation, ICA, KPAC or PCA. Fusion of additional sources using noise reduction techniques and validation of the proposed method using flight data from multiple aircraft to perform sensitivity analysis at a system level should be considered in future research on aircraft prognostics in systems and at a fleet level. It would also be possible to use a hybrid model to generate a decision tree and an artificial neural network or other classification method for each set of parameters. The best method for the given parameter set could then be used in the final classification forest.

The method cannot be used to forecast a specific failure, nor can it be used for diagnosis, because the health condition of the whole system, i.e. prognosis at a system level, is detected and forecast.

The method could be improved with the use of cleaner and less noisy data. Nevertheless, the process is robust; it can compensate by applying noise reduction to the time series because of the uniform distribution of the noise. The cost of this compensation is a delay in the detection of the system's health.

12 REFERENCES

Abbasion, S., Rafsanjani Abbasi, A., Farshidianfar, A. & Irani, N., 2007. Rolling element bearings multi-fault classification based on the wavelet denoising and support vector machine. *Mechanical Systems and Signal Processing,* March.

AFNOR, 1994. *Recueil des normes françaises X 06.* s.l.:AFNOR.

Ahmadi, A., Söderholm, P. & Kumar, U., 2010. On aircraft scheduled maintenance program development. *Journal of Quality in Maintenance Engineering,* 16(3), pp. 229-255.

Air Transport Association of America, 2007. *MSG-3: Operator/Manufacturer Scheduled Maintenance Development,* s.l.: s.n.

Air Transport Association of America, 2007. *MSG-3: Operator/Manufacturer Scheduled Maintenance Development,* s.l.: s.n.

Airbus SAS, 2008. *In-Service Reports database,* s.l.: s.n.

Airbus SAS, 2008. *Orders and Deliveries.* s.l.:s.n.

Ali, K. M. & McLoughlin, B., 2012. *Benefits of Optimizing Maintenance Intervals.* s.l., s.n.

Apte, C. et al., 1998. *Text mining with decision trees and decision rules.* s.l., s.n.

Arauzo-Azofra, A., Aznarte, J. L. & Benítez, J. M., 2011. Empirical study of feature selection methods based on individual feature evaluation for classification problems. *Expert Systems with Applications,* 38(7), pp. 8170-8177.

Ataei, S., Aghakouchak, A. A., Marefat, M. S. & Mohammadzadeh, S., 2005. Sensor fusion of a railway bridge load test using neural networks. *Expert Syst. Appl.,* Band 29, pp. 678-683.

Bao, D. & Yang, Z., 2008. Intelligent stock trading system by turning point confirming and probabilistic reasoning. *Expert Systems with Applications,* 34(1), pp. 620-627.

Basir, O. & Yuan, X., 2007. Engine fault diagnosis based on multi-sensor information fusion using Dempster - Shafer evidence theory. *Information Fusion,* 8(4), pp. 379-386.

Bowerman, B. L. & O'Connell, R. T., 1993. *Forecasting and Time Series. An Applied Approach.* s.l.:Duxbury Press.

Breiman, L., Friedman, J., Olshen, R. & Stone, C., 1984. *Classification and Regression Trees.* Monterey, CA: Wadsworth and Brooks.

Brotherton, T., Chadderdon, G. & Grabill, P., 1999. *Automated Rule Extraction for Engine Vibration Analysis.* s.l., s.n.

Caesarendra, W., Niu, G. & Yang, B.-S., 2010. Machine condition prognosis based on sequential Monte Carlo method. *Expert Systems with Applications,* 37(3), pp. 2412-2420.

Carvalho, D. R. & Freitas, A. A., 2002. A genetic-algorithm for discovering small-disjunct rules in data mining. *Applied Soft Computing,* 2(2), pp. 75-88.

Castanedo, F., 2013. A review of data fusion techniques. *The Scientific World Journal,* Band 2013.

Cernak, M., 2010. A Comparison of Decision Tree Classifiers for Automatic Diagnosis of Speech Recognition Errors. *Computing and Informatics,* July, 29(3), pp. 489-501.

Cernak, M., 2010. A Comparison of Decision Tree Classifiers for Automatic Diagnosis of Speech Recognition Errors. *Computing and Informatics,* July, 29(3), pp. 489-501.

Chaturvedi, A. & Chandra, S., 2004. *A Neural Stock Price Predictor using Quantitative Data.* s.l., Austrian Computer Society.

Chen, C., Zhang, B. & Vachtsevanos, G., 2012. Prediction of machine health condition using neuro-fuzzy and Bayesian algorithms. *Instrumentation and Measurement, IEEE Transactions on,* 61(2), pp. 297-306.

Chen, C., Zhang, B., Vachtsevanos, G. & Orchard, M., 2011. Machine condition prediction based on adaptive neuro - fuzzy and high-order particle filtering. *Industrial Electronics, IEEE Transactions on,* 58(9), pp. 4353-4364.

Chiang, I.-J. & Hsu, J. Y.-j., 2002. Fuzzy Classification Trees for Data Analysis. *Fuzzy Sets Syst.,* #aug#, 130(1), pp. 87-99.

Civil Aviation Authority, 1995. *Condition monitored maintenance: An explanatory handbook,* s.l.: s.n.

Civil Aviation Authority, 2009. *Aircraft Maintenance Incident Analysis,* s.l.: s.n.

Civil Aviation Regulations Directorate, 2006. *Master Minimum Equipment List/Minimum Equipment List Policy and Procedures Manual,* s.l.: s.n.

Cook, A. J. & Tanner, G., 2011. *European airline delay cost reference values,* s.l.: s.n.

Cook, A. J., Tanner, G. & Anderson, S., 2004. Evaluating the true cost to airlines of one minute of airborne or ground delay: final report.

Cook, A., Tanner, G. & Anderson, S., 2004. *Evaluating the true cost to airlines of one minute of airborne or ground delay.* s.l.:Eurocontrol.

Dong, M., Member, S., Kothari, R. & Member, S., 2001. Look-Ahead Based Fuzzy Decision Tree Induction. *IEEE-FS,* Band 9, pp. 461-468.

Ebersbach, S. & Peng, Z., 2008. Expert system development for vibration analysis in machine condition monitoring. *Expert systems with applications,* 34(1), pp. 291-299.

Emami-Naeini, A., Akhter, M. M. & Rock, S. M., 1988. Effect of model uncertainty on failure detection: the threshold selector. *Automatic Control, IEEE Transactions on,* 33(12), pp. 1106-1115.

Eurocontrol, 2006. *Cost of Delays - Notes,* s.l.: s.n.

Eurocontrol, 2014. *Seven-Year Forecast - September 2014,* s.l.: s.n.

Eurocontrol, 2015. *CODA Annual Report 2014,* s.l.: s.n.

European Aviation Safety Agency, 2005. *Hidden Function of Safety/Emergency Systems or Equipment MSG-3 Category Selection - Embraer Example,* s.l.: s.n.

European Aviation Safety Agency, 2008. *Issue Paper 44 - MRB Check Interval Escalations,* s.l.: s.n.

European Committee for Standardization, 2001. EN 13306 - Maintenance terminology.

European Committee for Standardization, 2010. EN 13306 - Maintenance terminology.

European Committee for Standardization, 2015. EN 13306 - Maintenance terminology - draft.

Farrar, C. R. et al., 2006. *Sensing and Sensor Optimization Issues for Structural Health Monitoring.* Manhatten Beach, s.n.

Federal Aviation Administration, 1978. *AC 120-17A – Maintenance Control by Reliability Methods,* s.l.: s.n.

Federal Aviation Administration, 1994. *AC 25-19 – Certification Maintenance Requirements,* s.l.: s.n.

Federal Aviation Administration, 1995. *Aviation System Indicators - 1994 and 1995 Annual Report,* s.l.: s.n.

Federal Aviation Administration, 2012. *AC 121-22C - Maintenance Review Boards, Maintenance Type Boards, and OEM/TCH Recommended Maintenance Procedures,* s.l.: s.n.

Federal Aviation Administration, 2012. *AC 121-22C - Maintenance Review Boards, Maintenance Type Boards, and OEM/TCH Recommended Maintenance Procedures,* s.l.: s.n.

Ferguson, J., Kara, A. Q., Hoffman, K. & Sherry, L., 2013. Estimating domestic US airline cost of delay based on European model. *Transportation Research Part C: Emerging Technologies,* Band 33, pp. 311-323.

Fijany, A. & Vatan, F., 2005. *A unified and efficient algorithmic approach to model-based diagnosis and optimal sensor placement.* s.l., s.n.

Fonollosa, J., Vergara, A. & Huerta, R., 2013. Algorithmic mitigation of sensor failure: Is sensor replacement really necessary?. *Sensors and Actuators B: Chemical,* Band 183, pp. 211-221.

Fonollosa, J., Vergara, A. & Huerta, R., 2013. Algorithmic mitigation of sensor failure: Is sensor replacement really necessary?. *Sensors and Actuators B: Chemical,* Band 183, pp. 211-221.

Friedl, M. A. & Brodley, C. E., 1997. Decision tree classification of land cover from remotely sensed data. *Remote sensing of environment,* 61(3), pp. 399-409.

Fu, T.-c., 2011. A review on time series data mining. *Engineering Applications of Artificial Intelligence,* 24(1), pp. 164-181.

Galar, D., Gustafson, A., Tormos, B. & Berges, L., 2012. Maintenance Decision Making Based on Different Types of Data Fusion. *Eksploatacja i Niezawodnosc, Maintenance and Reliability,* 14(2), pp. 135-144.

Galar, D., GuSTAFSON, A., Tormos, B. & Berges, L., 2012. Maintenance Decision Making Based on Different Types of Data Fusion. *Eksploatacja i Niezawodnosc, Maintenance and Reliability,* 14(2), pp. 135-144.

Galar, D., GuSTAFSON, A., Tormos, B. & Berges, L., 2012. Maintenance Decision Making Based on Different Types of Data Fusion. *Eksploatacja i Niezawodnosc, Maintenance and Reliability,* 14(2), pp. 135-144.

Galar, D., GuSTAFSON, A., Tormos, B. & Berges, L., 2012. Maintenance Decision Making Based on Different Types of Data Fusion. *Eksploatacja i Niezawodnosc, Maintenance and Reliability,* 14(2), pp. 135-144.

Galar, D., Kumar, U., Lee, J. & Zhao, W., 2012. *Remaining useful life estimation using time trajectory tracking and support vector machines.* s.l., s.n., p. 012063.

Garcia, M. C., Sanz-Bobi, M. A. & del Pico, J., 2006. SIMAP: Intelligent System for Predictive Maintenance: Application to the health condition monitoring of a windturbine gearbox. *Computers in Industry,* 57(6), pp. 552-568.

Gerdes, M., 2008. *Technical Note: PAHMIR Project Description,* Hamurg University of Applied Sciences.

Gerdes, M., 2013. Decision Trees and Genetic Algorithms for Condition Monitoring Forecasting of Aircraft Air Conditioning. *Expert Systems With Applications,* Amsterdam: Elsevier, Volume 40, Issue 12, 15 September 2013, S. 5021–5026. - https://doi.org/10.1016/j.eswa.2013.03.025, download: http://PAHMIR.ProfScholz.de

Gerdes, M. & Galar, D., 2016. Fuzzy Condition Monitoring of Recirculation Fans and Filters. *International Journal of System Assurance Engineering and Management,* 7(4), p. 469 – 479. - http://doi.org/10.1007/s13198-016-0535-y, download: http://PAHMIR.ProfScholz.de

Gerdes, M., Galar, D. & Scholz, D., 2016. Automated Parameter Optimization for Feature Extraction for Condition Monitoring. In: International Measurement Confederation (Hrsg.): *14th IMEKO TC10 Workshop on Technical Diagnostics 2016 : New Perspectives in Measurements, Tools and Techniques for Systems Reliability, Maintainability and Safety*. Milan, Italy, 27-28 June 2016, S. 452-457. Budapest, Hungary, 2016. - http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-59611, download: http://PAHMIR.ProfScholz.de

Gerdes, M., Galar, D. & Scholz, D., 2016. Effects of Condition-Based Maintenance on Costs Caused by Unscheduled Maintenance of Aircraft. *Journal of Quality in Maintenance Engineering,* 22(4), pp. 394-417. http://doi.org/10.1108/JQME-12-2015-0062

Gerdes, M., Galar, D. & Scholz, D., 2017. Decision Trees and the Effects of Feature Extraction Parameters for Robust Sensor Network Design. *Eksploatacja i Niezawodnosc – Maintenance and Reliability,* 19(1), pp. 31-42. http://doi.org/10.17531/ein.2017.1.5

Gerdes, M., Galar, D. & Scholz, D., 2017. Genetic Algorithms and Decision Trees for Condition Monitoring and Prognosis of A320 Aircraft Air Conditioning. *Insight - Non-Destructive Testing and Condition Monitorin,* August, 59(8), pp. 424-433. http://dx.doi.org/10.1784/insi.2017.59.8.424

Gerdes, M. & Scholz, D., 2009. Feature Extraction and Sensor Optimization for Condition Monitoring of Recirculation Fans and Filters. In: DGLR: *Deutscher Luft- und Raumfahrtkongress 2009 : Tagungsband - Manuskripte* (DLRK, Aachen, 08.-10. September 2009). - ISBN: 978-3-932182-64-2. - Download: http://PAHMIR.ProfScholz.de

Gerdes, M. & Scholz, D., 2011. Fuzzy Condition Monitoring of Recirculation Fans and Filters. *CEAS Aeronautical Journal,* Springer, Vol. 2, No. 1-4, 2011, p. 81-87. -https://doi.org/10.1007/s13272-011-0021-9.

Gerdes, M. & Scholz, D., 2011. *P*arameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems*.* In: Estorf, O. von & Thielecke, F. (Ed.): *3rd International Workshop on Aircraft System Technologies, AST 2011* (TUHH, Hamburg, 31. März - 01. April 2011). Aachen : Shaker, 2011, pp. 229 - 340. - Download: http://PAHMIR.ProfScholz.de

Gerdes, M., Scholz, D. & Randerath, B., 2009. *Reducing Delays Caused by Unscheduled Maintenance and Cabin Reconfiguration.* In: Estorf, O. von & Thielecke, F. (Ed.): 2nd International Workshop on Aircraft System Technologies, AST 2009 (TUHH, Hamburg, 26./27. March 2009). Aachen : Shaker, 2009, pp. 109 - 119. - Download: http://PAHMIR.ProfScholz.de

Golub, M. & Posavec, A. B., 1997. Using Genetic Algorithms for Adapting Approximation Functions. Proceedings of the 19th International Conference on Information Technology Interfaces ITI '97 .

Goode, K., Moore, J. & Roylance, B., 2000. Plant machinery working life prediction method utilizing reliability and condition-monitoring data. Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering, 214(2), pp. 109-122.

Granger, C. & Newbold, P., 1977. Forecasting Economic Time Series. s.l.:Academic Press.

Grieshaber, L., 2009. Erfassung und Synchronisation von Systemschwingungen bei unterschiedlichen Flugphasen. s.l.:s.n.

Heng, R. & Nor, M., 1998. Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition. Applied Acoustics, 53(1-3), pp. 211-226.

Hodge, V. J. & Austin, J., 2004. A survey of outlier detection methodologies. *Artificial Intelligence Review,* Band 22, p. 2004.

Hölzel, N. B., Schilling, T. & Gollnick, V., 2014. *An Aircraft Lifecycle Approach for the Cost-Benefit Analysis of Prognostics and Condition-based Maintenance based on Discrete-Event Simulation.* s.l., s.n., pp. 442-457.

Hsu, C.-W. & Lin, C.-J., 2002. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks,* Mar, 13(2), pp. 415-425.

Huang, C.-L. & Dun, J.-F., 2008. A distributed PSO - SVM hybrid system with feature selection and parameter optimization. *Applied Soft Computing,* 8(4), pp. 1381-1391.

Huang, C.-L. & Wang, C.-J., 2006. A GA-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications,* 31(2), pp. 231-240.

Institut du Transport Aerien, 2000. *Costs of Air Transport Delay in Europe,* s.l.: s.n.

Institut du Transport Aerien, 2000. *Costs of Air Transport Delay in Europe,* s.l.: s.n.

Institution, B. S., 1993. *Glossary of terms used in terotechnology.* s.l.:British Standards Institution, London.

International Air Transport Association, 2015. Best Practices for Component Maintenance Cost Management – 1st Edition, s.l.: s.n.

International Civil Aviation Organization, 2015. Master Minimum Equipment List/Minimum Equipment List Policy and Procedures Manual, s.l.: s.n.

International Electrotechnical Commission, 2003. IEC 61511: Functional Safety - Safety instrumented systems for the process industry sector, s.l.: International Electrotechnical Commission.

International Electrotechnical Commission, 2010. IEC 61508: Functional safety of electrical/ electronic/programmable electronic safety-related systems, s.l.: International Electrotechnical Commission.

Jack, L. & Nandi, A., 2000. *Genetic algorithms for feature selection in machine condition monitoring with vibration signals.* s.l., s.n., pp. 205-212.

Jack, L. & Nandi, A., 2002. Fault detection using support vector machines and artificial neural networks, augmented by genetic algorithms. *Mechanical systems and signal processing,* 16(2), pp. 373-390.

Jagadish, H. V. et al., 2014. Big Data and Its Technical Challenges. *Communications of the ACM,* July, 57(7), pp. 86-94.

Janikow, C. Z., 1995. *A Genetic Algorithm for Optimizing Fuzzy Decision Trees.* San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., pp. 421-428.

Jardine, A. K., Lin, D. & Banjevic, D., 2006. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing,* 20(7), pp. 1483-1510.

Kiyak, E., 2012. The effects of aircraft preventive maintenance on reliability. *International Journal of Applied Mathematics and Informatics,* 6(1), pp. 9-16.

Knotts, R. M., 1999. Civil aircraft maintenance and support Fault diagnosis from a business perspective. *Journal of quality in maintenance engineering,* 5(4), pp. 335-348.

Kolerus, J. & Wassermann, J., 2011. *Zustandsüberwachung von Maschinen: Das Lehr- und Arbeitsbuch für den Praktiker.* s.l.:Expert-Verlag.

Komonen, K. & Akatemia, T. T., 1998. The structure and effectiveness of industrial maintenance. s.l.:Finnish Academy of Technology.

Kontaki, M., Papadopoulos, A. N. & Manolopoulos, Y., 2005. Continuous Trend-Based Classification of Streaming Time Series.. s.l., s.n., pp. 294-308.

Kotecha, P. R., Bhushan, M. & Gudi, R. D., 2008. Design of robust, reliable sensor networks using constraint programming. Computers & Chemical Engineering, 32(9), pp. 2030-2049.

Kret, A., 2011. Statistische Analyse und Vorhersage von Zeitreihen zur vorausschauenden Wartung von Flugzeugkomponenten. s.l.:s.n.

Lambrou, T. et al., 1998. *Classification of audio signals using statistical features on time and wavelet transform domains.* s.l., s.n., pp. 3621-3624.

Li, F., 2011. *Dynamic modeling, sensor placement design, and fault diagnosis of nuclear desalination systems,* s.l.: s.n.

Lin, J. & Qu, L., 2000. Feature extraction based on Morlet wavelet and its application for mechanical fault diagnosis. *Journal of sound and vibration,* 234(1), pp. 135-148.

Lin, S.-W., Lee, Z.-J., Chen, S.-C. & Tseng, T.-Y., 2008. Parameter determination of support vector machine and feature selection using simulated annealing approach. *Applied soft computing,* 8(4), pp. 1505-1512.

Lin, S.-W., Tseng, T.-Y., Chou, S.-Y. & Chen, S.-C., 2008. A simulated-annealing-based approach for simultaneous parameter optimization and feature selection of back-propagation networks. *Expert Systems with Applications,* 34(2), pp. 1491-1499.

Li, S. & Elbestawi, M., 1996. Fuzzy Clustering for Automated Tool Condition Monitoring in Machining. *Mechanical Systems and Signal Processing ,* 10(5), pp. 533-550.

Lu, Y. & Michaels, J. E., 2009. Feature extraction and sensor fusion for ultrasonic structural health monitoring under changing environmental conditions. *Sensors Journal, IEEE,* 9(11), pp. 1462-1471.

Mahamad, A. K., Saon, S. & Hiyama, T., 2010. Predicting remaining useful life of rotating machinery based artificial neural network. *Computers & Mathematics with Applications,* 60(4), pp. 1078-1087.

Mahamad, A. K., Saon, S. & Hiyama, T., 2010. Predicting remaining useful life of rotating machinery based artificial neural network. Computers & Mathematics with Applications, 60(4), pp. 1078-1087.

Meiri, R. & Zahavi, J., 2006. Using simulated annealing to optimize the feature selection problem in marketing applications. European Journal of Operational Research, 171(3), pp. 842-858.

Mierswa, I. & Morik, K., 2005. Automatic feature extraction for classifying audio data. Machine learning, 58(2-3), pp. 127-149.

Mitchell, T. M., 1997. Machine Learning. s.l.:The McGraw-Hill Companies, Inc.. Mobley, R. K., 2002. *An introduction to predictive maintenance.* s.l.:Butterworth-Heinemann.

Montgomery, D. C., Johnson, L. A. & Gardiner, J. S., 1990. *Forecasting & Time Series Analysis.* s.l.:McGraw-Hill, Inc.

Montgomery, D. C., Johnson, L. A. & Gardiner, J. S., 1990. *Forecasting & Time Series Analysis.* s.l.:McGraw-Hill, Inc.

Mosallam, A. a. M. K. a. Z. N., 2014. Data-driven prognostic method based on Bayesian approaches forÂ direct remaining useful life prediction. *Journal of Intelligent Manufacturing,* pp. 1-12.

Muchiri, A., 2012. Comparing Alternative Strategies of Aircraft Maintenance. *Sustainable Research and Innovation Proceedings,* Band 4.

Nguyen, M. H. & De la Torre, F., 2010. Optimal feature selection for support vector machines. *Pattern recognition,* 43(3), pp. 584-591.

Niu, G. & Pecht, M., 2009. *A framework for cost-effective and accurate maintenance combining CBM RCM and data fusion.* s.l., s.n., pp. 605-611.

Niu, G. & Yang, B.-S., 2010. Intelligent condition monitoring and prognostics system based on data-fusion strategy. *Expert Systems with Applications ,* 37(12), pp. 8831-8840.

Novis, A. & Powrie, H., 2006. *PHM sensor implementation in the real world-a status report.* s.l., s.n., pp. 1-9.

Nowlan, F. & Heap, H., 1978. *Reliability-centered Maintenance.* s.l.:Dolby Access Press.

Ohm, J. & Lüke, H., 2010. *Signalübertragung: Grundlagen der digitalen und analogen Nachrichtenübertragungssysteme.* s.l.:Springer Berlin Heidelberg.

Olaru, C. & Wehenkel, L., 2003. A complete fuzzy decision tree technique. Fuzzy Sets Syst., Band 138, pp. 221-254.

Owen, M., 2007. Practical Signal Processing. s.l.:Cambridge University Press.

Pascual, D. G., 2015. Artificial Intelligence Tools: Decision Support Systems in Condition Monitoring and DIagnosis. s.l.:Crc Press.

Pecht, M., 2008. *Prognostics and health management of electronics.* s.l.:Wiley Online Library.

Peng, Z., Chu, F. & He, Y., 2002. Vibration signal analysis and feature extraction based on reassigned wavelet scalogram. *Journal of Sound and Vibration,* 253(5), pp. 1087-1100.

Pham, H. T. & Yang, B.-S., 2010. Estimation and forecasting of machine health condition using ARMA/GARCH model. *Mechanical Systems and Signal Processing,* 24(2), pp. 546-558.

Pham, H. T., Yang, B.-S., Nguyen, T. T. & others, 2012. Machine performance degradation assessment and remaining useful life prediction using proportional hazard model and support vector machine. *Mechanical Systems and Signal Processing,* Band 32, pp. 320-330.

Phillips, P. & Diston, D., 2011. A knowledge driven approach to aerospace condition monitoring. *Knowledge-Based Systems* , 24(6), pp. 915-927.

Poubeau, J., 2002. Methodology for Analysis of Operational Interruption Costs. *FAST,* Issue 26.

Quinlan, J., 1987. Decision Trees as probabilistic Classifiers. In: P. Langley, Hrsg. *Proceedings of the Fourth International Workshop on MAchine Learning.* s.l.:Morgan Kaufmann, pp. 31-37.

Quinlan, J. R., 1986. Induction of Decision Trees. *Mach. Learn,* pp. 81-106.

Quinlan, J. R., 1993. *C4.5: Programs for Machine Learning.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Rabiner, L. & Juang, B. H., 1993. *Fundamentals of Speech Recognition.* s.l.:Prentice Hall.

Randall, R. B., 2011. *Vibration-Based Condition Monitoring. Industrial, Aerospace, and Automotive Applications.* s.l.:John Wiley & Sons, Ldt.

Ross, A. & Jain, A., 2003. Information fusion in biometrics. *Pattern recognition letters,* 24(13), pp. 2115-2125.

Rupp, N. G. & Holmes, G. M., 2006. An investigation into the determinants of flight cancellations. Economica, 73(292), pp. 749-783.

Russell, S. J. & Norvig, P., 2003. Artificial Intelligence: A Modern Approach. s.l.:Pearson Education.

Sachon, M. & Pate-Cornell, E., 2000. Delays and safety in airline maintenance. *Reliability Engineering & System Safety,* 67(3), pp. 301-309.

Saimurugan, M., Ramachandran, K., Sugumaran, V. & Sakthivel, N., 2011. Multi component fault diagnosis of rotational mechanical system based on decision tree and support vector machine. *Expert Systems with Applications,* 38(4), pp. 3819-3826.

Sakthivel, N., Sugumaran, V. & Nair, B. B., 2010. Comparison of decision tree-fuzzy and rough set-fuzzy methods for fault categorization of mono-block centrifugal pump. *Mechanical systems and signal processing,* 24(6), pp. 1887-1906.

Samanta, B., 2004. Artificial neural networks and genetic algorithms for gear fault detection. *Mechanical Systems and Signal Processing,* 18(5), pp. 1273-1282.

Samanta, B., 2004. Gear fault detection using artificial neural networks and support vector machines with genetic algorithms. *Mechanical Systems and Signal Processing,* 18(3), pp. 625-644.

Samanta, B., Al-Balushi, K. & Al-Araimi, S., 2003. Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection. *Engineering Applications of Artificial Intelligence,* 16(7), pp. 657-665.

Sap, M. N. M. & Khokhar, R. H., 2004. *Fuzzy Decision Tree for Data Mining of Time Series Stock Market Databases.* s.l., s.n.

Saravanan, N. & Ramachandran, K., 2009. Fault diagnosis of spur bevel gear box using discrete wavelet features and Decision Tree classification. *Expert Systems with Applications,* 36(5), pp. 9564-9573.

Saxena, A. & Saad, A., 2007. Evolving an artificial neural network classifier for condition monitoring of rotating mechanical systems. *Applied Soft Computing,* 7(1), pp. 441-454.

Scerbo, M. W. & Bailey, N. R., 2007. Automation-induced complacency for monitoring highly reliable systems: the role of task complexity, system experience, and operator trust. Theoretical Issues in Ergonomics Science, pp. 321-348.

Schmid, H., 1994. Probabilistic part-of-speech tagging using decision trees. s.l., s.n., pp. 44-49.

Scholz, D., 1995. *Betriebskostenschätzung von Flugzeugsystemen als Beitrag zur Enwurfsoptimierung*, (Deutscher Luft- und Raumfahrtkongress, Bonn, 26. - 29. September 1995). In: Bürgener, G. (Ed.): Jahrbuch 1995. Bonn : Deutsche Gesellschaft für Luft- und Raumfahrt, 1995, pp. 50 - 61. - Paper: DGLR-JT95-016, download: http://paper.ProfScholz.de

Scholz, D., 1998. *DOCsys - A Method to Evaluate Aircraft Systems.* In: Schmitt, D. (Ed.): Bewertung von Flugzeugen (Workshop: DGLR Fachausschuß S2 - Luftfahrtsysteme, München, 26./27. Oktober 1998). Bonn : Deutsche Gesellschaft für Luft- und Raumfahrt. - Download: http://paper.ProfScholz.de

Schruben, L., 1983. Confidence interval estimation using standardized time series. *Operations Research,* 31(6), pp. 1090-1108.

Schwarz, H. R. & Käckler, N., 2004. *Numerische Mathematik.* s.l.:Teuber.

Shin, J.-H. & Jun, H.-B., 2015. On condition based maintenance policy. *Journal of Computational Design and Engineering,* 2(2), pp. 119-127.

Shuming, Y., Jing, Q. & Guanjun, L., 2012. Sensor optimization selection model based on testability constraint. *Chinese Journal of Aeronautics,* 25(2), pp. 262-268.

Sikorska, J., Hodkiewicz, M. & Ma, L., 2011. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing,* 25(5), pp. 1803-1836.

Sikorska, J., Hodkiewicz, M. & Ma, L., 2011. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing ,* 25(5), pp. 1803-1836.

Si, X.-S., Wang, W., Hu, C.-H. & Zhou, D.-H., 2011. Remaining useful life estimation â€“ A review on the statistical data driven approaches. *European Journal of Operational Research ,* 213(1), pp. 1-14.

Smith, A. E., Coit, D. W. & chia Liang, Y., 2003. *A Neural Network Approach to Condition Based Maintenance: Case Study of Airport Ground Transportation Vehicles.* s.l.:s.n.

Smith, M. J., 2005. Sensor Location & Optimization Tool Set. Albuquerque, s.n.

Society of Automotive Engineers, 2001. ARP 5580: Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications, s.l.: s.n.

Sondalini, M., 2015. Don't Waste Your Time and Money with Condition Monitoring. s.l.:Lifetime Reliability Solutions.

Sörensen, K. & Janssens, G. K., 2003. Data mining with genetic algorithms on binary trees. *European Journal of Operational Research,* 151(2), pp. 253-264.

Sreejith, B., Verma, A. K. & Srividya, A., 2008. *Fault diagnosis of rolling element bearing using time-domain features and neural networks.* s.l., s.n., pp. 1-6.

Stecki, J. S., Rudov-Clark, S. & Stecki, C., 2014. *The rise and fall of CBM (Condition based Maintenance).* s.l., s.n., pp. 290-301.

Stein, G., Chen, B., Wu, A. S. & Hua, K. A., 2005. *Decision tree classifier for network intrusion detection with GA-based feature selection.* New York, NY, USA, ACM, pp. 136-141.

Sugumaran, V., Muralidharan, V. & Ramachandran, K., 2007. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing,* 21(2), pp. 930-942.

Sugumaran, V. & Ramachandran, K., 2007. Automatic rule learning using decision tree for fuzzy classifier in fault diagnosis of roller bearing. *Mechanical Systems and Signal Processing,* 21(5), pp. 2237-2247.

Sugumaran, V. & Ramachandran, K., 2011. Effect of number of features on classification of roller bearing faults using SVM and PSVM. *Expert Systems with Applications,* 38(4), pp. 4088-4096.

Sun, J.-w.et al., 2009. Design for diagnosability of multistation manufacturing systems based on sensor allocation optimization. *Computers in Industry,* 60(7), pp. 501-509.

Tian, Z., 2012. An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *Journal of Intelligent Manufacturing,* 23(2), pp. 227-237.

Tian, Z., Wong, L. & Safaei, N., 2010. A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. Mechanical Systems and Signal Processing, 24(5), pp. 1542-1555.

Tian, Z. & Zuo, M. J., 2010. Health condition prediction of gears using a recurrent neural network approach. Reliability, IEEE Transactions on, 59(4), pp. 700-705.

Tran, V. T., Yang, B.-S. & Tan, A. C. C., 2009. Multi-step ahead direct prediction for machine condition prognosis using regression trees and neuro-fuzzy systems. *Expert Systems With Applications,* July, 36(5), pp. 9378-9387.

Tsai, C.-F., 2009. Feature selection in bankruptcy prediction. *Knowledge-Based Systems,* 22(2), pp. 120-127.

University of Waikato, 2009. *WEKA.* s.l.:s.n.

US Department of Defense, 1998. *MIL-HDBK-338-1B, Electronic Reliability Design Handbook,* s.l.: s.n.

US Department of Transportation, 2014. *July 2015 Air Travel Consumer Report,* s.l.: s.n.

Van Horenbeek, A., Pintelon, L. & Galar, D., 2012. Integration of disparate data sources to perform maintenance prognosis and optimal decision making. *Insight-non-destructive testing and condition monitoring,* 54(8), pp. 440-445.

Vapnik, V. N., 1998. *Statistical learning theory.* 1 Hrsg. s.l.:Wiley.

Vass, J. et al., 2008. Avoidance of speckle noise in laser vibrometry by the use of kurtosis ratio: Application to mechanical fault diagnostics. *Mechanical Systems and Signal Processing,* 22(3), pp. 647-671.

Wagner, M. & Fricke, M., 2006. *Estimation of daily unscheduled line maintenance events in civil aviation.* s.l., s.n.

Waikato, U. o., 2009. *WEKA.* s.l.:s.n.

Wang, C.-M. & Huang, Y.-F., 2009. Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data. *Expert Systems with Applications,* 36(3), pp. 5900-5908.

Wang, T., Yu, J., Siegel, D. & Lee, J., 2008. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. s.l., s.n., pp. 1-6.

Wang, W. Q., Ismail, F. & Golnaraghi, M. F., 2001. Assessment of Gear Damage Monitoring Techniques using Vibration Measurements. *Mechanical Systems and Signal Processing,* Band 15, pp. 905-922.

Wang, X.-Z., Zhai, J.-H. & Lu, S.-X., 2008. Induction of Multiple Fuzzy Decision Trees Based on Rough Set Technique. *Inf. Sci.,* #aug#, 178(16), pp. 3188-3202.

Weber, K., 2008. *Filter Clogging Indication of Recirculation Air Filters.* s.l.:Airbus Operations GmbH.

Widodo, A. & Yang, B.-S., 2007. Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors. *Expert Systems with Applications,* 33(1), pp. 241-250.

Widodo, A. & Yang, B.-S., 2007. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing,* 21(6), pp. 2560-2574.

Williams, J. H., Davies, A. & R., D. P., 1994. *Condition-based Maintenance and Machine Diagnostics.* s.l.:Chapman & Hall.

Williams, J. H., Davies, A. & R., D. P., 1994. *Condition-based Maintenance and Machine Diagnostics.* s.l.:Chapman & Hall.

Xiong, N. & Svensson, P., 2002. Multi-sensor management for information fusion: issues and approaches. *Information fusion,* 3(2), pp. 163-186.

Yan, W. & Goebel, K. F., 2003. *Sensor validation and fusion for gas turbine vibration monitoring.* s.l., s.n., pp. 106-117.

Yen, G. G. & Lin, K.-C., 2000. Wavelet packet feature extraction for vibration monitoring. *Industrial Electronics, IEEE Transactions on,* 47(3), pp. 650-667.

Yuan, Y. & Shaw, M. J., 1995. Induction of Fuzzy Decision Trees. *Fuzzy Sets Syst.,* #jan#, 69(2), pp. 125-139.

Zaher, A. & McArthur, S., 2007. *A Multi-Agent Fault Detection System for Wind Turbine Defect Recognition and Diagnosis.* s.l., s.n.

Zhang, J., 2006. Improved on-line process fault diagnosis through information fusion in multiple neural networks. *Computers & chemical engineering,* 30(3), pp. 558-571.

Zhang, J., 2006. Improved on-line process fault diagnosis through information fusion in multiple neural networks. *Computers & chemical engineering,* 30(3), pp. 558-571.

Zhao, F., Chen, J., Guo, L. & Li, X., 2009. Neuro-fuzzy based condition prediction of bearing health. *Journal of Vibration and Control.*