NTNU
Norwegian University of
Science and Technology

Decision Support for Predictive Maintenance of Exposed Aquaculture Structures

Niklas Bae Pedersen
Fredrik Lindahl Roppestad

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Agnar Aamodt, IDI
Co-supervisor: Helge Langseth, IDI
Gunnar Senneset (SINTEF), IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In cooperation with SINTEF Fisheries and Aquaculture, our project is within EXPOSED SFI. We are under the technical Area 2 - Monitoring and operational decision support. Our main task is to study the combined use of data-driven machine learning (ML) and case-based reasoning (CBR) for enhanced data analysis and active decision support in fish farming, especially for the production structures.

In the specialisation project *Decision Support System for Exposed Aquaculture Operations*, there was proposed an architecture for a decision support system for predictive maintenance of exposed aquaculture structures. In this MSc thesis, the task is to study the method of implementing a system based on this architecture.

To achieve this goal, a new data analysis must be done. This analysis consists of pre-processing and exploratory analysis, as well as modelling and machine learning. As the previously proposed architecture is based on some assumptions and sensors not available in this MSc thesis, it is expected that revising the architecture is needed.

The architecture is based on principles from Structural Health Monitoring, Case-Based Reasoning and Machine Learning. The development of a prototype system consist of creating the different elements that constitutes the system, as well as implementation and evaluation.

Abstract

It is a global challenge to produce enough healthy food to a growing world population. By moving industrial fish farming to exposed locations, the farmers can possibly satisfy the dietary requirements of the future by expanding the production. The environment at exposed locations is rough, and conditions like harsh wind, large waves and strong currents are present. On the positive side, there are better water flow and distribution of waste. Exposed locations are also further away from natural salmons, which might reduce negative environmental effects.

The production plants for fish farming are designed to be flexible and adaptive to waves and sea currents. Monitoring of the health and condition of these structures will be more important regarded reducing cost of operations and maintenance. This data can be combined with historical data and expert knowledge to support the operators decision about acting upon a possible problematic situation.

Our goal in this MSc thesis is to study the method of implementing a decision support system for predictive maintenance of exposed aquaculture structures, based on a previously proposed architecture from the specialisation project. The architecture is based on the fields of Machine Learning, Structural Health Monitoring and Case-Based Reasoning.

This thesis describes the work done in order to achieve our goals, which includes the work of researching the relevant domains, conducting a data analysis, creating models with the resulting data sets, revising the previously proposed architecture and implementing a prototypical decision support system. At last we conclude and discuss the future work.

The work with this thesis shows that our prototype of a decision support system is able to support an operator with advice about what to do, if a situation similar to previously experienced situations occur.

Sammendrag

Det er en global utfordring å produsere nok sunn mat til en verdenbefolkning i vekst. Ved å flytte industriell fiskeoppdrett til eksponerte områder, kan oppdretterne ekspandere produksjonen slik at kravene til fremtidens matmengder blir tilfredsstilt. Ved eksponerte områder er miljøet tøft, og fohold som kraftig vind, store bølger og sterk strøm kan forekomme. På den positive siden er det bedre vanngjennomstrømning og distribusjon av avfall. Eksponerte områder er i tillegg lengre unna villaks, hvilket kan redusere sannsynligheten for negative miljøpåvirkninger.

Anleggene brukt til fiskeoppdrett er fleksible og adaptive til bølger og havstrømmer. Overvåking av disse strukterens helse og tilstand vil bli viktigere med tanke på kostnadene ved operasjoner og vedlikehold. Kombinert med historisk data og ekspertkunnskap, kan denne dataen støtte operatørenes beslutninger når problematiske situasjoner oppstår.

Målet med denne masteroppgaven er å studere metoden for implementasjon av et beslutningsstøttesystem for prediktivt vedlikehold av eksponerte akvakulturelle strukturer, basert pÃě en tidligere foreslått arkitektur fra forprosjektet. Arkitekturen er basert pÃě områdene Machine Learning, Structural Health Monitoring og Case-Based Reasoning.

Oppgaven beskriver arbeidet gjort for å nå prosjektets mål, dette inkluderer forskning av relevante domener, gjennomføring av en dataanalyse, utforming av modeller med det resulterende datasettet, revidering av den tidligere foreslåtte arkitekturen og implmentasjon av et prototypisk beslutningsstøttesystem. Til slutt konkluderer vi og diskuterer fremtidig arbeid.

Arbeidet med denne oppgaven viser at vårt prototypiske beslutningsstøttesystem er kapabelt til Ãě rådgi operatøren, dersom en situasjon lignende til en tidligere opplevd situasjon skulle oppstå.

Preface

This MSc thesis is written at the Department of Computer and Information Science at Norwegian University of Science and Technology, in collaboration with SINTEF Fisheries and Aquaculture. The work started with a specialisation project within the Artificial Intelligence field in autumn 2015.

During the project, the goals and specifications were redefined due to the data available. As the data was delivered midway through the project period and was found insufficient for the original specification, we focused more on research within the domain, creating reasonable symptoms and cases, and revising the architecture. The need for an implemented prototype became less important, and we focused more on studying the method.

We would like to thank our main supervisor Agnar Aamodt at NTNU for expertise, guidance and discussions throughout the project period. We would also like to thank our co-supervisors Helge Langseth at NTNU and Gunnar Senneset at SINTEF FH for valuable inputs and feedback. Further we would like to thank Kerstin Bach at NTNU, BjÃÿrn Magnus Mathisen at NTNU/SINTEF, David Kristiansen at SINTEF FH and Martin FÃÿre at SINTEF FH for their contribution.

Contents

List of Tables

List of Figures

Listings

Chapter 1

Introduction

In this MSc thesis, we aim to study the combined use of data-driven machine learning (ML) and case-based reasoning (CBR) for enhanced data analysis and active decision support in fish farming. As the scope of developing a decision support system (DSS) for the complete production system is too wide, this project targets the floating ring of the cage. The structure is flexible and the sensor basis is limited but also rich, which make this domain more complex than others where structural health monitoring (SHM) techniques are usually applied. By combining ML, CBR and SHM, we have provided a modular system that identifies symptoms of possible errors on the floating ring, and further give an advice and an explanation to the operator based on previous experience.

In the specialisation project "Decision Support System for Exposed Aquaculture Operations", an architecture for a decision support system for predictive maintenance of exposed aquaculture structures was proposed. In this MSc thesis, we have done further research and data analysis based on the available data from a real world production plant, and implemented a prototype system based on a revised edition of the previously proposed architecture.

The decision support system is based on principles from Structural Health Monitoring, Case-Based Reasoning and Machine Learning. Implementing the system consists of developing the four main main components of the architecture: Data Acquisition, Data Interpretation, Case-Based Reasoning and GUI. The implementation also

includes analysing and preparing the sensor data, definition and development of symptoms recognised in the data and creation of a CBR module. Testing to get an overview of the CBR system to be implemented should be done by using an existing CBR tool like myCBR.

A thorough overview of the report is given in Section 1.3.

1.1 Goals

Our goal in this MSc thesis is to study the method of implementing a decision support system for predictive maintenance of exposed aquaculture structures, based on a previously proposed architecture from the specialisation project. The architecture is based on the fields of Machine Learning, Structural Health Monitoring and Case-Based Reasoning. The purposes of this system are:

- Prediction of plant condition
- Automating the processes of
  - Early notification
  - Acting on event
  - Predictive maintenance

To achieve this goal, a new data analysis must be done. This analysis consists of pre-processing and exploratory analysis, as well as modelling and machine learning. As the previously proposed architecture is based on some assumptions and sensors not available in this MSc thesis, it is expected that revising the architecture is needed.

1.2 Motivation

Our MSc thesis is done in cooperation with SINTEF Fisheries and Aquaculture, several departments from the Norwegian University of Science and Technology (NTNU), and external companies in a SFI (Centre for Research-based Innovation) called EXPOSED. The recently started SFI, that has a planned duration from 2015 to 2022, targets developing knowledge and technology for robust, safe and ef-

ficient fish farming at exposed locations.

The challenges are summarised on EXPOSED's web page:

> *"Significant parts of the Norwegian coast are today unavailable for industrial fish farming because of its remote locations and rough conditions like wind, wave and current. Technical innovations like autonomous, offshore constructions and vessels are required for maintaining production during all conditions and to enable more robust, safe and controlled operations."*

It is a global challenge to produce enough healthy food to a growing world population, and fish farming will play an important role in satisfying the dietary requirements of the future. The World Bank Group estimates that 62 % of all seafood to consumers originates from fish farming by 2030. Norway's salmon farming is an important history of success in the global aquaculture production. From the early beginning in the 1970s the Atlantic salmon industry has spread all over the Norwegian fjords and coast, and in 2013 it produced 1.3 million tons of fish with an export value of 39.8 billion NOK. The fish farming industry is now an important employer and a socioeconomic mainstay supporting district communities along the coast of Norway. Further development is possible, and Norway has the possibility of producing 5 million tons of fish per year by 2050 if important challenges regarding production and environment are solved. The Food and Agriculture Organisation of the United Nations acknowledges that technology development will play an important role in future growth of the industry.

Fish farming of salmon and trout started in sheltered coastal areas, but are now moving towards more exposed locations. This is driven by the need of more space and better production environment, and exposed farming locations can have good conditions for production as well as important environmental influences are reduced. This locations give more stable growing conditions and larger distribution of waist material because of constant water flow. In addition, production areas will be further away from wild salmon near the coast, which can contribute to reducing the negative environmental consequences caused by lice and escapes.

Fish farmers who gradually have started using more exposed locations report significant difficulties with maintaining a reliable production, because of the challenges mentioned earlier. Some places

have been abandoned because of the difficulties of doing the necessary operations in an efficient way, which is one of the basic demands for profitable and sustainable farming. The EXPOSED SFI will use Norway's strong position and knowledge within maritime sectors, like aquaculture and offshore, to develop technical innovations within four research areas:

- Area 1 - Autonomous systems and technologies for remote operations
- Area 2 - Monitoring and operational decision support
- Area 3 - Structures for exposed locations
- Area 4 - Vessel design for exposed operations

In addition, two research areas focus on central assumptions for sustainable production:

- Area 5 - Safety and risk management
- Area 6 - Fish behaviour and welfare

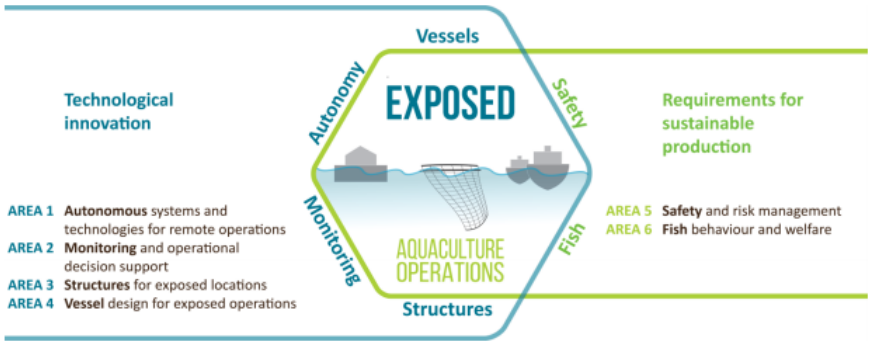

Figure 1.1: Research areas of the Exposed SFI [SINTEF, 2015].

In the context of the EXPOSED SFI, our project is within technical Area 2 - Monitoring and operational decision support. The motivation for our part of the project is to secure better fish welfare, reduced risk for the operational personnel, reduced environmental impact, and contribute to economic benefits by reducing operational costs and improving the quality of the end product [SINTEF, 2016].

The motivation for using Machine Learning, Case-Based Reason-

ing and Structural Health Monitoring stems from the challenges that lies beneath the mentioned motivation, as well as our goals.

Contrary to offshore oil and gas installations, the production plants for fish farming are designed to be flexible and adaptive to waves and sea currents. Monitoring the health and condition of these structures is of high complexity, and involves a combination of sensors measuring both the environment (sea current profiles, wave data) and structural behaviour (accelerations, deformations, strains). This data combined with predefined symptoms that later can be identified by data analysis and numerical models, can produce an estimate of the total system state of the fish farm. We can then make use of a CBR system that is using previous knowledge to make new decisions. The next chapter introduces these techniques, and shows some aspects of them.

1.3 Thesis Structure

In the next chapter we give a brief overview of the background for this MSc thesis, and relevant parts of the main technologies we aim to use in this project. Chapter 3 describes the related research, covering structural health monitoring, case-based reasoning and structures in the fish farming production. In Chapter 4 we describe the methodological approach and the main tools we have used to solve the tasks in the project. Chapter 5 covers the analysis of the data we have been given, and highlights the challenges with the available data. The modelling of the analysed data is described in Chapter 6, containing the creation of symptoms and cases. Chapter 7 contains a description of the architecture, before describing the implementation in Chapter 8. At last we evaluate our work in Chapter 9, before concluding and discussing the future work in Chapter 10.

Chapter 2

Background

This chapter presents the background for this MSc thesis and the Exposed SFI, as well as introducing the fields of Machine Learning, Case-Based Reasoning and Structural Health Monitoring. The first section describes the specialisation project done the autumn 2015, which is the predecessor to this thesis. The chapter also gives a brief introduction to the fish farming industry.

2.1 Specialisation Project

The specialistion project conducted during the autumn 2015 made the basis for the MSc thesis. The goal was to study the the fields of Structural Health Monitoring, Case-Based Reasoning and relevant civil engineering structures, and use the knowledge to to design an architecture for a decision support system within the fish farming domain. Further, we specyfied the guidelines for supplementing the system in this MSc thesis, and we did an analysis of the data available at that time.

The specialisation project was research based, and involved answering some reasearch questions we defined. Those were essential to answer for understanding the techniques we were going to use, and to get enough knowledge to propose an architecture. The RQs were based on previous knowledge and discussions with our supervisors, and was:

- How does Structural Health Monitoring methods and techniques work, and how are they implemented?
- How is the usage of Case-Based Reasoning and Structural Health Monitoring combined in other systems?
- What are the properties of a fish farming plant structure, and which should we focus on?

The answers to these questions were found by studying several research papers, as well as meeting experts from NTNU and SINTEF FH. The relevant results are shown later in this report, together with the proposed architecture.

2.2 The Aquaculture Industry

The Aquaculture industry has become a major worldwide industry and has history dating back to 6000 BC [Wikipedia, 2016a]. The idea to farm fish as we know it today originated in China around 2500 BC. An early technique used by the Chinese was to keep- and feed carp caught in small lakes formed by river floods. Today, China is the largest producer of seafood, while other parts of Asia, South America and Europe follow. There is six main kinds of aquaculture, include fish farming, shrimp farming, oyster farming, mariculture, algaculture (such as seaweed farming), and the cultivation of ornamental fish. Figure 2.1 shows a typical area with a lot of small fish farming plants in China.

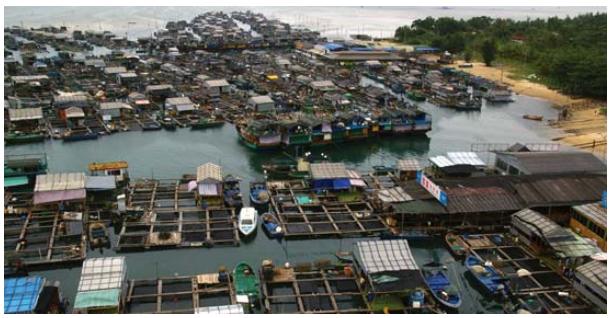

Figure 2.1: Fish farms near a seaside village on Hainan Island, China [Britannica, 2016].

Table 2.1: Aquaculture production (tons) by continents in 2010, 2012 and 2013.

	2010	**2012**	**2013**
Asia	52 440 372	58 955 770	62 546 664
Americas	2 581 206	2 977 959	3 068 755
Europe	2 543 978	2 876 726	2 781 125
Africa	1 286 441	1 485 387	1 615 608
Oceania	185 648	181 458	177 695
World	59 037 646	66 477 300	70 189 848

In the food-producing industry, aquaculture is probably the fastest growing sector in the world, and produces about 50% of all fish eaten today [FAO, 2016]. The total world production for the fisheries was 158 million tonnes in 2012, where 66.5 million tonnes came from aquaculture[1]. Table 2.1 illustrates the aquaculture production in the years 2010, 2012 and 2013 for all the continents (without Russia) with highest production and the total world production.

2.3 Fish Farming

Fish farming involves feeding fish commercially in tanks onshore or fish cages offshore along the coast. The different types of production facilities has advantages and disadvantages, including in terms of:

- cooling
- placement
- water quality
- possibilities of escape

The main process of fish farming is the same for both onshore and offshore, and consists mainly of breeding, feeding, taking care of fish and selling fish.

There are many fish species suitable for aquaculture, but in Norway mainly salmon is used for breeding. It takes approximately

[1] http://www.fao.org/3/a-i4899e.pdf

three years from salmon are eggs until they are ready for slaughter [Kvistad, 2016]. After hatching from the eggs, salmon develop into larvae. After between 10 and 16 months, the salmon become smolts and are ready to be deployed in the aquaculture facility. The salmon are taken up for slaughter when they weigh between 4 and 6 kg , and this takes about 14 to 22 months in the production plant. Figure 2.2 shows a wellboat retreiving salmon for slaughter on the Norwegian Aquaculture Center production site.

Figure 2.2: Wellboat retrieves salmon for slaughter at the ACE production plant [Wikipedia, 2016e].

A regular offshore production site consists of several production units. In the different production units there are fish in various size depending on which level the fish is in the breeding cycle. Under the breeding cycle, there are four main operations which must be performed, de-lousing, slaughtering, sorting and deployment.

2.4 Machine Learning

Machine learning is the study of computer algorithms that improve automatically through experience [Mitchell, 1997]. It is a subfield of computer science that is built on concepts from probability and statistics, information theory, philosophy, neurobiology and other fields.

> *"One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner."*
>
> *Tom M. Mitchell*

Depending on the type of feedback available, the learning tasks are broadly divided into three main categories [Russel and Norvig, 2010]:

- **Unsupervised learning** is when an agent[2] is not supplied with any explicit feedback. In these tasks the agent has to learn patterns or structures in the data on its own, and the most common task is clustering.
- **Reinforcement learning** is when rewards or punishments are given to the agent when it performs anything, as an indication of its performance. This can be a reward of points if the agent wins a game, or a punishment of no tips if a taxi agent does something wrong.
- **Supervised learning** is when an agent is presented with some examples, usually input-output pairs, and it learns a function or general rule that map inputs to outputs. This is typically when you have some training data and you want to classify new examples.

As we are going to work with sensor data and numerical models, our focus will be on supervised learning. As the training examples are determined by the sensor data from the production plant, the first step will be to analyse the data. Data analysis involves understanding and preparing the data, which we describe in section 2.1.1.

[2] An agent is something that acts. Computer agents are expected to operate autonomously, perceive their environment, persist over time, adapt to change, and create and pursue goals [Russel and Norvig, 2010].

Supervised learning is practically determining the nominal class[3] or numerical value[4] of unseen instances, based on the algorithm learned by training on the previous known instances. When solving a problem using supervised learning, the following steps must be performed [Wikipedia, 2016j].

- **1. Data analysis.** Determine the type of training examples, and gather a training set. These actions require data analysis, and domain knowledge to be performed optimally. Further should one determine how to represent the relevant features that the algorithm should learn. These steps are further explained in the next chapter.
- **2. Choose learning algorithm.** The choice of the learning algorithm is dependent on the complexity of the task, the feature representation and the capabilities of the algorithm itself. If the class is numerical and we predict a value, regression algorithms are preferable. When dealing with a (bi- or multi-)nominal class where the category is predicted, classification algorithms are used [Rohrer, 2016].
- **3. Optimisation and evaluation** When the algorithm is ready to be run on the training set, one should test it with different parameters and variants of the training set. The different types of algorithms often have several adjustable parameters that can greatly affect the performance. A separate validation set or a subset of the training set is often used when adjusting parameters. The latter method is called cross-validation and is widely used. The results should at last be evaluated by using a test set that is separate from the training set.

There are a lot of supervised learning algorithms to consider, depending on the above mentioned factors. Some often used ones are Support Vector Machine, Linear Regression, NaÃŕve Bayes, Decision Trees, K-Nearest Neighbour and Neural Networks. There is no single one that outperform the others on every supervised learning problem, as each has their strengths and weaknesses.

There are several issues to consider in supervised learning. Some

[3] text
[4] text

notable are:

- **The bias-variance dilemma** is when a learning algorithm is either biased or has high variance for a particular input x. The algorithm has bias if it trains on several different, but equally good training sets, and systematically classifies the output wrong. High variance means that the algorithm gives a different classification of an instance, based on which training set it has been trained on.

- **Overfitting** is the issue of trying to fit the data too carefully. This may happen if the model is too complex, and the consequence is that the learner is too sensitive to noise in the data.

- **Dimensionality of the input space** is a problem where the input feature vector have a high dimension, but the true function only depends on some particular features. The dimensions that does not influence the true function may create errors and confuse the learning algorithm, and cause a high variance.

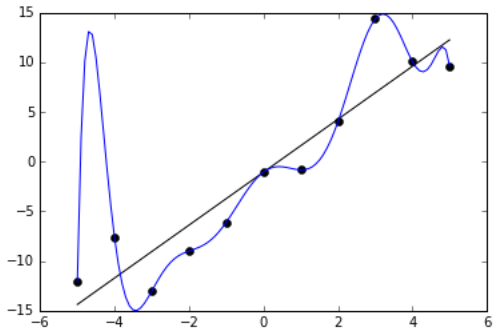

Figure 2.3: Noisy (roughly linear) data is fitted to both linear and polynomial functions. Although the polynomial function is a perfect fit, the linear version can be expected to generalise better. In other words, if the two functions were used to extrapolate the data beyond the fit data, the linear function would make better predictions [Wikipedia, 2016i].

Data Analysis

Data analysis is a term which means finding useful information in a set of values. This is a process which contains tasks like selecting, preprocessing, transforming and modelling data, and is a widely used and very important tool in many domains. As machine learning focuses on prediction based on known properties learned from the training data, data analysis focuses on the discovery of unknown properties in the data. One can look at data analysis as the process of obtaining raw data, and converting it to information useful for decision-making. This process is illustrated in Figure 2.4.

Data analysis is a broad term that is encompasses data mining, business intelligence, predictive analytics, data visualisation etc. [Wikipedia, 2015a]. Some important tasks on a data set, depending on the user interest are:

- Filtering
- Finding Extrema
- Sorting
- Clustering
- Correlating

To do effective and sound analysis, the analysts must distinguish fact from opinion, be aware of cognitive biases, and account for the innumeracy of the audience.

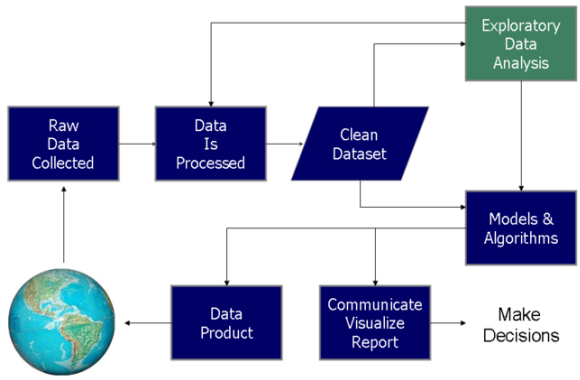

Figure 2.4: The data analysis process [Wikipedia, 2016c].

In the context of machine learning, data analysis tasks are often the first step when preparing the data sets for the learning algorithms. The steps and requirements for making these sets are:

- Correct data specification
- Data collection from different sources
- Data processing, exploitation and organising
- Data cleaning if the data is incomplete, has duplicates or contain errors
- Exploratory analysis, e.g by representing the average, median og extremum
- Mathematical modeling to identify variables like correlation or the mean square error of a regression

At last, we can build a data set for further analysis like visualisation, or to use in machine learning algorithms.

2.5 Case-Based Reasoning

Case-Based Reasoning is a problem solving approach where the system uses prior knowledge to solve new problems [Aamodt and Plaza, 1994]. Where many other machine learning approaches uses general knowledge about a problem domain, CBR uses specific knowledge about previous experienced situations to find solutions to new problems. This specific information is called a case, and is stored in a case-base. When a new problem is solved, CBR looks at which cases are similar to the problem case and tries to reuse the previous solution. The solution can then be adapted to the new problem, and then be retained in the case-base as a new solution. This makes CBR a lazy learner, which means that it learns incremental. The CBR approach is therefore well suited for learning to solve new problems.

CBR is regarded a subfield of machine learning, and works much like the human reasoning process. Take for instance help desk support, where an employee receives incoming calls. A new problem is solved by using the experience of the employee, or she has to get more knowledge. When the problem is solved, the employee now knows how to solve it. This process is very much the same in CBR, and the CBR

cycle by Aamodt and Plaza shows the four generalised steps:

- **Retrieve** cases similar to the problem case.
- **Reuse** the solution stored with these cases, and adapt it if needed to fit the new problem case.
- **Revise** the solution if necessary, after performing a real world test, a simulation or asking a domain expert.
- **Retain** the case with the new solution in the case base.

A visualisation of the CBR cycle, and how the four steps and the case-base are connected is shown in Figure 2.5.

As mentioned, CBR uses past experience stored as cases for solving new problems. These typically contains a problem description and a solution description. The input to a CBR system is typically a problem case, and a possible solution may exist in a similar case in the case base. The output is typically the most similar case(s), which contain a suggested solution.

To propose a solution to a problem case, the CBR system must be able to calculate the similarity between the input case and the case(s) to be retrieved. This can be done in different ways, and Richter et al. mentions seven types of similarity measures [Richter and Weber, 2013]:

- Counting similarities
- Metric similarities
- Transformation similarities
- Structure-oriented similarities
- Information-oriented similarities
- Relevance-oriented similarities
- Dynamic-oriented similarities

This shows that similarity can be measured in many ways. In more complex case structures, elements like weights on different features and a local-global approach could be useful. The latter means that objects are constructed by atomic parts that reflect the local measure, while looking at the whole objects reflect the global view.

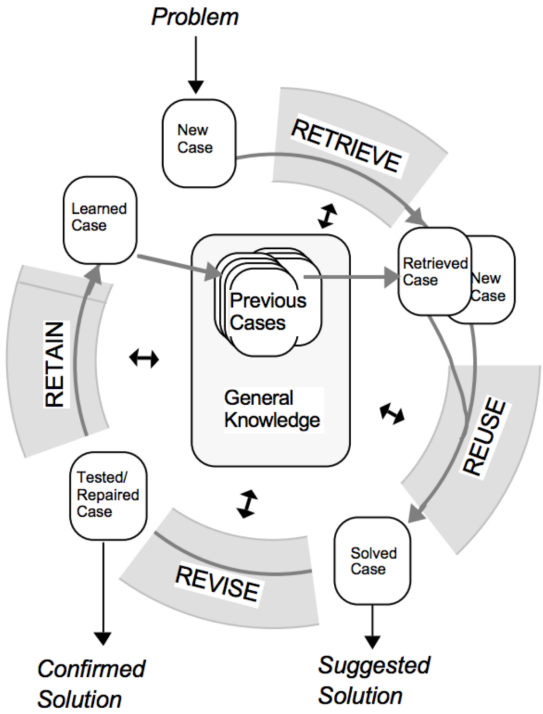

Figure 2.5: The CBR cycle by Aamodt [Aamodt and Plaza, 1994].

2.6 Decision Support

A decision support system (DSS) is a computer-based information system that supports humans in many types of decision making activities [Wikipedia, 2015b]. Typical applications for these systems are in planning, direct choices, management and operations, and they aim to make people make better decisions in complex problems.

Many types of computer systems can be used as a decision support system. Rule-based expert systems, neural networks and many other machine learning and knowledge based systems can be used in a decision support system. The simplified architecture for DSSs is the same, regardless of the inference engine and knowledge base, as shown in Figure 2.6. As computer systems can assist humans with drawing inferences when the knowledge base or the statistical data contains a

lot of information, the developers have to build them as useful, accurate and explanatory as possible. This is in most situations a difficult task.

Decision support systems have many different applications in several domains. In health care, DSSs have been used for diagnostics, drug advisory, alerts and reminders, therapy planning and more [Darlington, 2011]. In the business sector, where large amounts of corporate data are stored for analysis, decision support plays a key role in the tools that plan the strategic choices of the corporation [Beal,]. In the domain of structural health monitoring, systems have been developed for damage detection in bridges and lamp posts. We will describe SHM in section 2.4, as we aim to combine this technique with decision support.

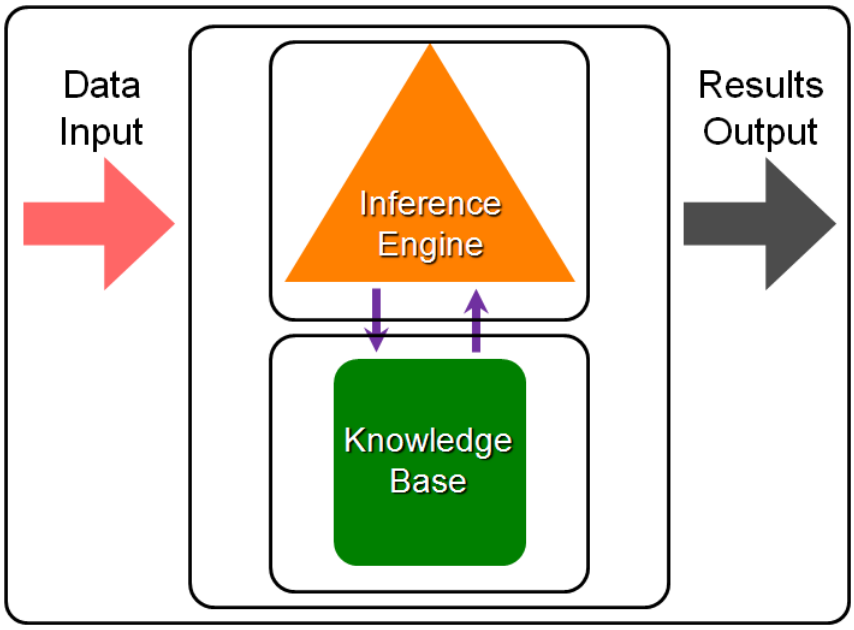

Figure 2.6: An illustration of a typical decision support system architecture [Bonney, 2011].

2.7 Structural Health Monitoring

Structural Health Monitoring is a process of dealing with the development and implementation of systems and techniques, where damage detection of structures is the essential part [AGH, 2008]. Different sensors are monitoring the states and features of a structure, and statistical analysis define its health.

As SHM has developed over the last three decades, researchers and engineers have boiled down the fundamentals of the field to a few fundamental axioms. The general principles defined by Worden et al. [Worden et al., 2007] are:

- **Axiom I:** All materials have flaws inherent or defects.
- **Axiom II:** The assessment of damage requires a comparison between two system states.
- **Axiom III:** Identifying the existence and location of damage can be done in an unsupervised learning mode, but identifying the type of damage present and the damage severity can generally only be done in a supervised learning mode.
- **Axiom IVa:** Sensors cannot measure damage. Feature extraction through signal processing and statistical classiiňĄcation is necessary to convert sensor data into damage information.
- **Axiom IVb:** Without intelligent feature extraction, the more sensitive a measurement is to damage, the more sensitive it is to changing operational and environmental condition.
- **Axiom V:** The length- and time-scales associated with damage initiation and evolution dictate the required properties of the SHM sensing system.
- **Axiom VI:** There is a trade-off between the sensitivity to damage of an algorithm and its noise rejection capability.
- **Axiom VII:** The size of damage that can be detected from changes in system dynamics is inversely proportional to the frequency range of excitation.

Typical applications for SHM are large structures like bridges and buildings, where sensors typically are accelerometers, strain gauges, temperature sensors etc. [Dascotte et al., 2013]. SHM has also been used offshore, both for vessels and platform structures [Ren et al., 2006b],

and we aim to examine the use of this technique in the fish farming domain.

Chapter 3

Related Research

The topics covered in our study of related research are based on the technologies and methods presented in the previous chapter. The purpose of the study is to acquire knowledge and information about the systems similar to what we aim to create, and the structures used in fish farming production.

Before conducting the research study in the specialisation project, we defined some overall research questions mentioned in Section 2.1. This chapter describes the results found by studying relevant literature and information gathered from meetings with domain experts.

3.1 SHM Methods & Techniques

The main inspection approach for large civil engineering structures that must be routinely inspected is manual visual inspection. In a study of the Nondestructive Evaluation Validation Center, the method of visual inspection of bridges has been shown quite subjective, because of factors like traffic, visual abilities, light intensity, inspector work load, complexity, degree of maintenance, and accessibility [Moore et al., 2000]. Visual inspection is also an expensive task, which in 2009 had a cost of approximately 10 000 EUR per 100 meters bridge [Wenzel, 2009]. SHM offers different methods and techniques that have been developed with the purpose to ease this work, increase safety and precision, and reduce the large costs. See Section 2.3 for

more information.

We have chosen to look at three different methods from real world implementations and research. These methods are based on either using measurements of vibrations and material properties, or the use of guided-waves. These two approaches can be seen as the main techniques within the field.

Guided-Wave

Guided-waves can be defined as stress waves forced to follow a path defined by the material boundary of a structure [Raghavan, 2007]. Guided-waves are generated by an actuator as a high frequency pulse signal. By measuring these signals, one can locate structural discontinuity, which typically indicate a damage in the structure, a structural feature or a boundary. When encountering these elements, the guided-wave signal scatters in all directions as seen in Figure 3.1.

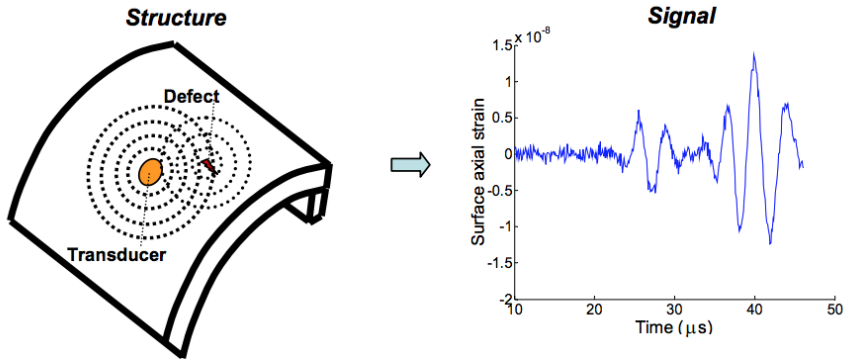

Figure 3.1: The first two steps in the Guided-Wave SHM technique [Raghavan, 2007].

To be able to distinguish between damage, structural features and boundaries, it is necessary to obtain prior information about the measured structure in its undamaged state. This is used as a reference in future measurements. By using signal processing algorithms on the sampled signal shown in Figure 3.1, relevant features are extracted as shown in Figure 3.2. At last, pattern recognition techniques are used to classify the damage and estimate its extent, in Figure 3.2 shown as

a neural network classifier. In the guided-wave approach, a threshold value decides whether the discovered damage actually is real or not. The threshold value is usually application dependent.

In civil engineering structures, the guided-wave approach has not been given as much attention as vibration and material property based methods. A reason is that structures like bridges and buildings are in general big and thick in size, in opposite to mechanical domains like aerospace, automotive and petrochemistry. However, the approach shows promising results when used on smaller scale civil engineering structures.

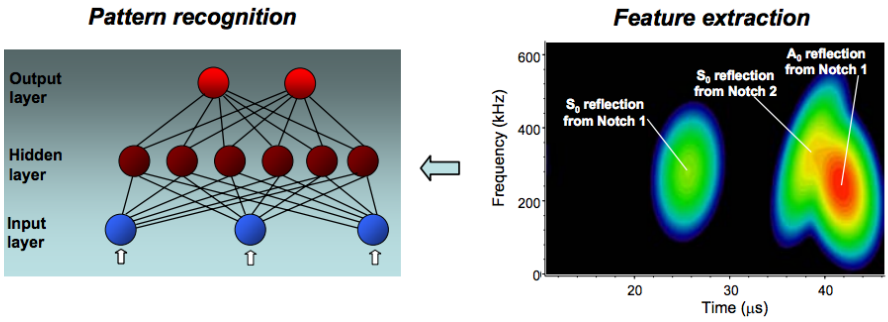

Figure 3.2: The last two steps in Guided-Wave SHM , feature extraction and pattern recognition [Raghavan, 2007].

Vibrational Characteristics

The researchers Caicedo and Dyke uses another approach with measurements of vibrational characteristics to detect, identify and quantify structural damage [Caicedo and Dyke, 2005]. They describe their work within SHM for flexible bridge structures such as cable-stayed bridges, and present evidence for successful damage detection in an experimental bridge model.

Cable-stayed bridges are complex since they are flexible, and bring challenges to SHM as the vibration frequencies are very low. An often used SHM method for flexible structures is based on changes in the dynamic properties of the structures. The technique is based on collecting and producing information from both the healthy and the damaged structure, before damage is identified, located and quanti-

fied by comparing the information from both data sets. The five main steps of the technique are:

- **Development of an identification model** which defines the elemental parameters, and further specifies where the sensors in use are located. This is a simplified model of the total system.
- **Placement of sensors** determined by the identification model. To measure the responses of structures, accelerometers are typically used.
- **Acquisition of data** means measuring a sufficient number of modes of the structure. The sampling frequency depends on the structure, as for instance structures with a low frequency of vibration need long records of data.
- **Modal identification** is done by using the eigensystem realisation algorithm[1] in combination with impulse hammer response data, to identify the modal parameters of the structure.
- **Parameter identification** is estimating values for the parameters that form the elemental stiffness matrices of the structure. Caicedo and Dyke use undamped natural frequencies and mode shapes found by a least-squares solution to the eigenvalue problem[2].

By comparing the identified parameters calculated in the last step for both damaged and healthy modes, one can decide whether there is a damage in the structure or not. The flow is illustrated in the overview in Figure 3.3.

Implementation of the proposed methodology first requires the development of an identification model. This is a construction of a finite element mesh, and special care must be taken when having elements whose stiffness cannot be isolated from the affection of nearby elements. Figure 3.4 shows a two floor structure and its corresponding identification model. In this structure, the Young's moduli parameter must be identified for every element. This is not possible if assuming that the floors are rigid and sensors are on each floor, as the numbered elements in the Figure 3.4(a) affect the stiffness of the floors. The two

[1]The ERA uses the principle of minimum realisation to obtain a state space representation of the system [Juang and Pappa, 1985], for more information see [Caicedo and Dyke, 2005].

[2]See [Caicedo and Dyke, 2005] for mathematical calculations.

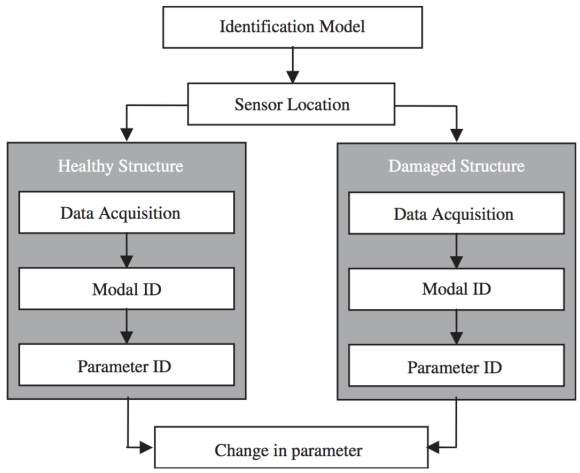

Figure 3.3: Overview of the structural health monitoring technique [Caicedo and Dyke, 2005].

degrees of freedom[3] identification model shown in Figure 3.4(b) will be a suitable model for the structure. The sensors should be placed according to the identification model, and they should measure all the active degrees of freedom of the model.

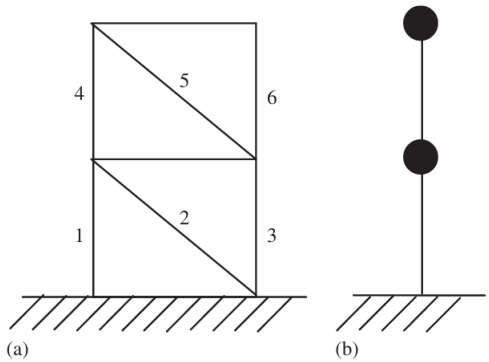

Figure 3.4: Two-floor structure (a) and corresponding identification model (b) [Caicedo and Dyke, 2005].

[3]The degrees of freedom of a system is the number of parameters of the system that may vary independently[Wikipedia, 2015c]

To identify the modal parameters of the structure, the ERA is used. To implement the ERA, the first step is to build the Hankel matrix. The next step is to perform a singular value decomposition of the Hankel matrix, before calculating two matrices, A and C, for the discrete-time state representation of the system. The natural frequencies correspond to the eigenvalues of A, and the mode shapes corresponding to the response measurement are C multiplied by the eigenvectors of A.

To estimate the parameters that form the elemental stiffness matrices of the structure, a least squares solution to the the eigenvalue problem is used. The natural frequencies and mode shapes are used to obtain the solution. This methodology can be used to identify different parameters like Young's moduli, moments of inertia, and other parameters that affect the stiffness matrix. The formulas for calculating the stiffness can be obtained in [Caicedo and Dyke, 2005]. The calculation must be done for identified parameters of the structure in both damaged and healthy states. The results are then compared to find damage in the structure.

An experimental test done by Caicedo and Dyke shows successful implementation of the described SHM methodology. The test was performed on a model of a cable-stayed bridge, designed and constructed to reproduce the complex, three dimensional behaviour of this kind of bridges. Two tests were performed on the healthy structure, and two on the damaged structure. When damage was induced, the tests shows variations of maximum 1.75 % in the identified natural frequencies, as shown in Table 3.1. Further the mode shape of vibration changed as much as 18.7 %, as shown in Figure 3.5.

	Healthy structure		Damaged structure	
Frequency number	Test 1	Test 2	Test 3	Test 4
1	6.29	6.29	6.24	6.24
2	15.23	15.22	15.16	15.16
3	16.90	16.90	16.67	16.72
4	21.90	21.91	21.56	21.57
5	27.32	27.33	26.92	27.03
6	30.10	30.10	29.57	29.58
7	39.01	39.00	38.61	38.87
8	41.61		41.24	

Table 3.1: Identified natural frequencies [Caicedo and Dyke, 2005].

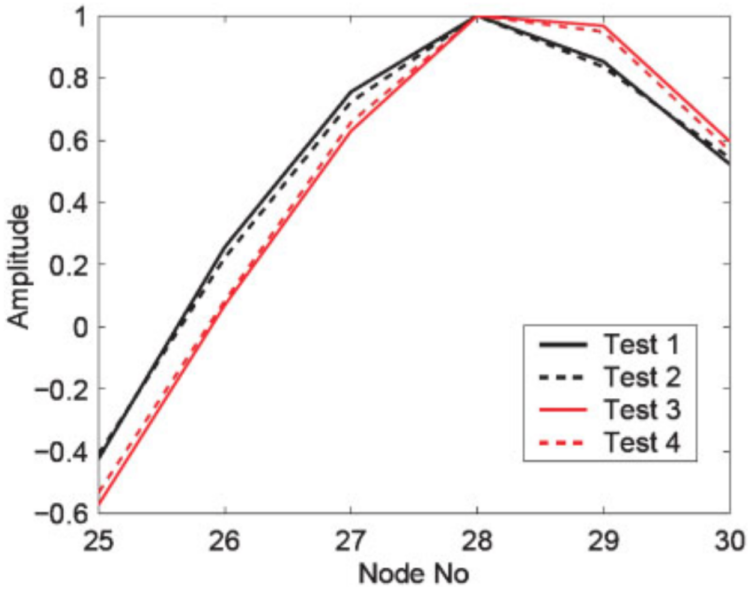

Figure 3.5: Identified mode shape [Caicedo and Dyke, 2005].

Strain Sensors

In a damage study of composite structures performed by Kamath et al., resistance strain gages and fiber Bragg grating sensors (FBG) were used [Kamath et al., 2010]. The application was different types of composite test boxes where bending loads were applied, and the strains were measured near and away from the bending points.

A fiber Bragg grating is a type of distributed Bragg reflector[4]. It is constructed in a short segment of optical fiber, that reflects particular wavelengths of lights and transmit all others [Wikipedia, 2015e]. The advantages of FBG sensors are that they are immune to electromagnetic interference, they can be multiplexed, and can easily be integrated in different structures. Further, they are well suited for load monitoring, and the usage as strain sensors are well known within SHM [Kamath et al., 2010, Ren et al., 2006a, Murawski et al., 2012].

The tests were done by comparing the measured strain on healthy and damaged structures, when applying various load strength on the

[4]A DBR is a reflector used in waveguides, like optical fibers [Wikipedia, 2015d].

structures. This was performed on four different composite testboxes of different construction, and Figure 3.6 shows the comparison of the strain gage and FBG data measured on a healthy case to the left and a damaged case to the right. As the testboxes were different in construction and damage, the results gave different graphs. All the results can be found at [Kamath et al., 2010]. Although, the conclusion of the study was that when the absolute strains were converted to relative strains by taking the healthy strains as the baseline, the use of FGB-based strain measurement systems makes a strong case.

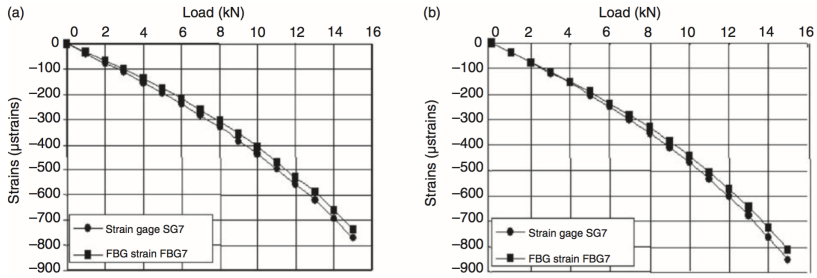

Figure 3.6: Comparison of FBG and strain gage data for a testbox at outer surface of top skin: (a) healthy case and (b) weakened structure case [Kamath et al., 2010].

A way to implement a SHM system by using FBG sensors is to calculate the modal macro-strain ratio (MMSR) [Serker and Wu, 2010]. This ratio represents the comparison of the MMS on a reference point, like a healthy structural point, with other points. The basic concept for using MMSR for damage detection, is that the calculated strain ratio is constant between two different positions on the strain sensor under the same amount of load. Mathematical formulas for the MMSR calculation is found on [Serker and Wu, 2010]. The methodology is based on four main steps, which are:

- Installation of the FBG sensors and the reference sensors.
- Data acquisition and calculation of MMS.
- Create reference model and the new model from the data set.
- Compare the measured model with the reference model.

In Figure 3.7, all the four main steps are visualised.

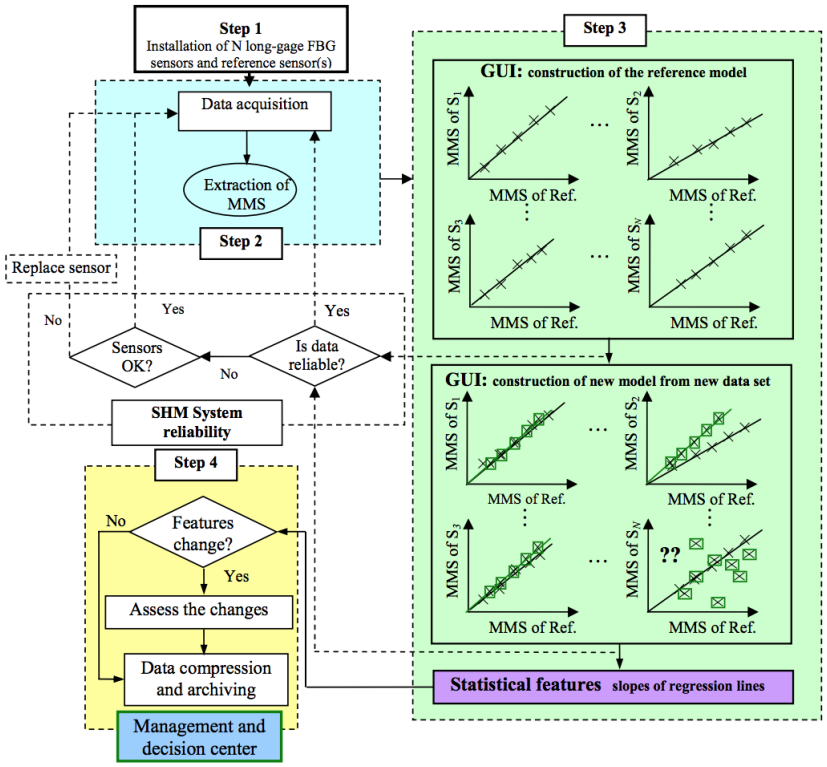

Figure 3.7: Flowchart by using the modal macro-strain ratio approach [Serker and Wu, 2010].

3.2 Combination of CBR & SHM

As mentioned earlier, CBR has the advantage of reusing past experience to solve new problems. This reflect the engineers like when they use experience when interpreting measured data. Together with the possibility to concentrate on known important aspect of certain problems, and avoiding repetition of failures, this is the most important advantages of CBR for SHM. Within the field of SHM, CBR can be used in different ways. There are many examples of the usage of Case-Based Reasoning in both periodic and permanent monitoring [Freudenthaler, 2011, Delgado, 2005, Gundersen et al., 2013].

In a typical CBR system for SHM, measured parameters of a structure are in the case. These parameters are in general processed data,

and depends on the sensors and SHM method used in the application. Further, the input case is compared with other cases in the case base, and the most similar are presented. The system may then assess the structure given in the new case, or give any kind of advice.

Combined with SHM, CBR supports the very complex and time consuming manual interpretation, and aims to help making better decisions. The (semi-) automation of the assessing process also decreases cost and subjectivity. This section describes the use of CBR as a decision support system in different applications.

Periodic Monitoring

In the context of using CBR in periodic monitoring, a new case typically consist of the parameters of a new measured structure. This kind of monitoring are either based on the comparison of similar measurements of similar structures, or the comparison of past measurements of the structure which shall be analysed [Stumptner et al., 2009].

An example of periodic monitoring is the CBR decision support system for semi-automated assessment of lamp posts, described by B. Freudenthaler [Freudenthaler, 2011]. He describes a system where cases consist of structural parameters of these posts in different conditions. Expert consultation indicates that the number of attributes describing the lamp posts should be set to 20, and the three main types were:

- **Geometric attributes** covering the geometric properties like height, radius etc.
- **Numerical attributes** like eigenfrequencies, vibration data etc. in both stimulated and natural environments.
- **Descriptive attributes** describing the optical inspection.

The classification of the posts was the same as the civil engineers use, ranging from A to F, where A means at least 12 year stable and F means immediate replace.

The case base contained 799 cases, and each case represented a single real world measurement of a certain lamp post. When a new case should be classified, the most similar case are used as a reference case to suggest the classification of the new case. It was also possi-

ble to use the k most similar cases, and classify the new case as the majority of the k nearest cases. It was possible to adjust the weights of the attributes if the experts find some more important than others. Regarding this, it is important that the subjectivity remains, so a predefined or computer-optimised weighting may be advised. In the finishing step of the classification, the user had the possibility to correct the suggested classification if the expert has further information. Finally, the case could be stored in the case base for future reuse, or be discarded.

The test of the system was done by using one, three, five, ten, 15, 20 nearest cases, as well as clustered centres for each class. Outlier detection was performed on the case base as well. The results showed in general that the more reference cases used for classification, the more accurate the result became. But, the averaged correctness per class was better if less reference cases were considered. In this case the clustered centres were by far best, which may be because of few cases in the case base for the classes E and F. The results are shown in Table 3.2. It is essential to remember that even if k-nearest reference cases gave the overall best result, the most critical cases in the classes E and F may not be classified correctly as these two classes only contained only 1.13 % of the cases.

Table 3.2: Lamp post classification results [Freudenthaler, 2011].

	K-ref. cases	Clustered centres
Total	85.86-89.05%	78.22-86.76%
Per class	43.31-49.22%	58.72-72.14%

The conclusion of the test shows that the assessment of the lamp posts by using CBR combined with SHM is successful. The time used was reduced from 100 - 130 working hours when assessing manually, to a few seconds of computation plus about one day of expert review.

Permanent Monitoring

When using CBR in permanent monitoring, the structures monitored are often at high risk. Simple systems of this kind may be constructed by alarms going of if reaching a predefined threshold value. Usually, permanent monitoring compares past measurements with new ones

of the current structure to be analysed, and not with other similar structures. When monitoring for instance a bridge real-time, the case base typically contains damaged or unhealthy states of the structure. New measurements then acts as new cases, and is compared against the case base [Delgado, 2005].

The DrillEdge real-time DSS for drilling oil wells uses CBR combined with pattern matching to identify symptoms of problems during the drilling operation. In this system a case is a concrete drilling situation that contains a collection of symptoms that previously lead to a problem, as well as a recommendation for how to handle a similar situation [Gundersen et al., 2013]. These cases are made from historic drilling data and by the best practice of the operators for how to handle the situations.

The system has to be able to learn from few examples, as there are relatively few examples of serious incidents or accidents in the history. The developers have chosen a two step approach containing:

- **Pattern matching agents** that identify the symptoms in the data by forming abstractions over the sensor data.
- **A CBR system** that identify whether the set of identified symptoms has caused problems in similar situations in the past.

The system contains many pattern matching agents, where each agent has the task of identifying one type of symptom predefined by domain experts. These agents are similar to expert systems that acquires knowledge from a data set. The data is simple, and based on a rudimentary model, instead of an advanced model. This means selecting only the most obvious influencing parameters, and ignore unimportant effects.

The main reason for using CBR as the situation identifier is that the system must provide explanatory support as well as an answer. The system does not make a prediction explicit, but it displays all the most similar cases, sorted by the prediction they imply. This is shown in Figure 3.8, where the cases are displayed on the radar GUI element. The most similar cases are closer to the centre, and when you click on the case it opens. The case has two parts, the description and the solution. The description is used by the computer for comparison of cases, while the solution is an experience transfer from one human to another. The role of the case is not

only to classify the situation, but also to provide experience and best practice. Figure 3.9 shows the DrillEdge CBR cycle. In this system cases are represented in tree structures and stored as XML files.

The degree of similarity between two cases is found by comparing the root nodes in the case trees. This similarity is calculated by the aggregation of recursively combined section similarities of the tree. Both domain specific and standard similarity measures are used.

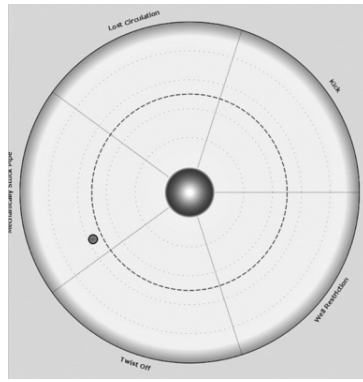

Figure 3.8: The radar in the DrillEdge GUI [Gundersen et al., 2013].

If the current situation is not covered by any cases in the case base, a new case is created by the drilling engineer. A new case is then reviewed by a group of domain experts. Together with revising the symptom recognition agents and satisfying the real time demand of the similarity comparison, the work and time used to define new cases were the three main challenges during the development of the system.

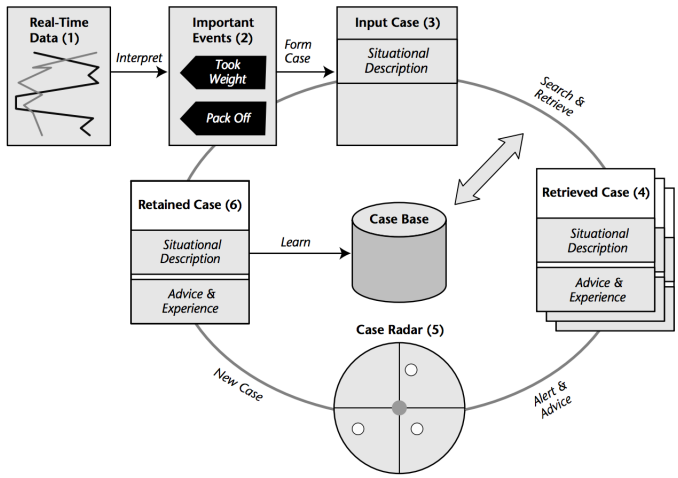

Figure 3.9: DrillEdge CBR cycle [Gundersen et al., 2013].

3.3 Properties of Fish Farming Plant Structures

To be able to develop a system for decision support regarding the health and maintenance of a fish farming plant, we have to understand the structures to work with. As we collaborate closely with SINTEF Fisheries and Aquaculture, we scheduled a meeting with David Kristiansen[5], head of Area 3 in the Exposed SFI, to get a lecture about the area.

Properties of a Fish Farming Plant

The industry status of modern fish farming at the largest and best sites, produces about 10 000 - 15 000 tonnes of Salomon per cycle. A farm site contains 10 - 12 cages, where one cage has a radius of 50 meters and contains maximum 200 000 fishes.

The fish farm consists of three main components. These are:

- Feed barge
- Mooring system
- Cage structure

The feed barge contains the feed storage unit, the operation room and a living quarter. Here the operators control and monitor the farm, and regulate the feeding of the fish.

Figure 3.10: Typical float collar [Aqualine, a].

The mooring system holds the cage structure in place, and different areas need different mooring system concepts. A broadly used system is the frame mooring, as we will describe later. Research tasks within these systems are looking at the non-linear effects of mooring load, classifying damaged conditions,

[5]PhD., Research Manager - Aquaculture structures at SINTEF Fisheries and Aquaculture.

and the effects of interventions or mooring operations.

The normally used cage structure contains a plastic floating collar, a net cage and a weight system. The plastic floating collar is designed to withstand the forces from the mooring. It is typically made of two flexible plastic pipes, mounted together by steel brackets as shown in Figure 3.10. The pipes may be filled with Styrofoam, which make the construction float even if the pipe gets damaged or punctured. The mooring is attached in brackets, specially made to reduce the load on the collar. Figure 3.11 shows a three-way bracket attachment used in a frame based mooring system.

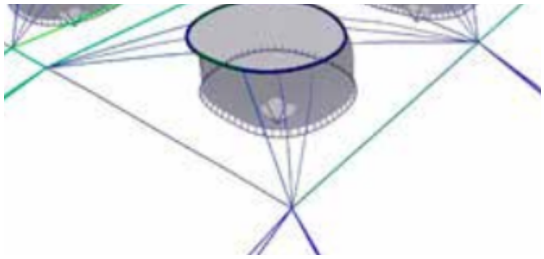

Figure 3.11: Typical mooring hub or "rooster foot" [Aqualine, b].

The net is attached to the floating collar with steel brackets, which secure that the load is evenly distributed. The net itself is a simple structure, but it is very robust with high quality ropes that tolerate harsh environments. A the bottom, the bottom ring is attached. This ring keeps the net in the correct formation. Figure 3.12 shows a visualisation of a complete cage from the leading Norwegian provider Aqualine AS.

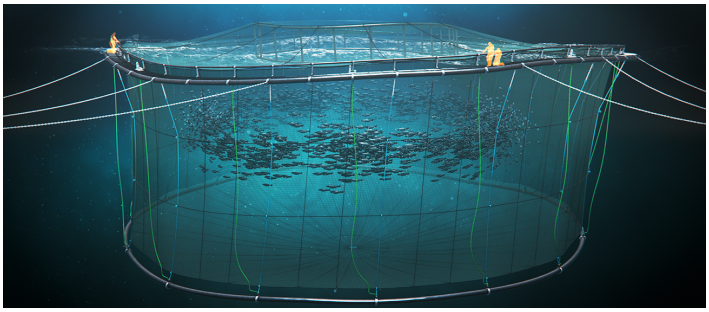

Figure 3.12: Visualisation of a complete cage [Aqualine, a].

Risk Factors

There are many risk factors for both human beings and the fish farming structure. By the specifications of this project, we focus on the structural risk factors only. The risks of structural damage by environmental stress and operational work are present at both exposed and sheltered plants, but the effects and outcome may be more severe in exposed locations [M.G. Sandberg, 2012].

The environmental impact on the plant is mainly wear and growth of unwanted sea life organisms. The latter has the consequence that the structure needs cleaning by operators, which we mention later. The structural wear is mainly a long-term process which make the equipment and main parts weaker over time. As the locations get more exposed, the sudden (and long-term) wear caused by more harsh weather and sea currency need more attention. An other impact is frost on the plastic collar, caused by low air-temperature and frosty winds [Utne et al., 2015]. This makes the structure less flexible and more vulnerable to damage, and must be removed.

Material damage may be caused by the plant itself, or by functional or supportive operations. Damage to the net and attachment points can be caused by materials rubbing and gnawing on each other, and is often a long time process. Operations on the other hand are direct interaction with the plant or single cages, and the most important are listed in Table 3.3. The ones that require vessels usually have a larger scope, and are the most critical regarding material damage. These are marked as bold in the table. Some real world material damage reported in incident reports are:

- Boat stuck in delousing tarpaulin and got dragged under water.
- Bottom ring chain rubbed a hole in the net.
- ”Rooster foot” damaged by boat propeller.
- Snapped bottom ring chain.

Additionally to environmental and operational impact, the plant may be affected by other human interactions. There are examples of boats running over the plant, and damage the structure [Okstad, 2014]. Boats travelling nearby the production area poses a threat for both the mooring system and the other structures, as they might collide and some create large waves.

Table 3.3: The most important operations.

Operation	*Frequency*	*Duration*	*Scope/Vessel*	*Critical Factors*
Retrieval of dead fish	Daily	Hours	Little, workboat	Weather permitting, manual labor by cage
Feeding	Daily	Hours	Little	Weather permitting, fish welfare, appetite
Refill of feed barge	Weekly	Hours	Moderate, feeding boat by fleet	Weather permitting, vessels at facilities, use of crane
Sampling	Weekly	Hours	Little, workboat	Weather permitting, access to fish, manual labor on cage
Washing of nets	Weekly	Hours	Moderate, service vessel, workboat	Weather permitting, manual labor, use of washing rigs, damage on net, vessels at facilities
Delousing with use of wellboat	Monthly	Hours	Large, wellboat/service vessel /workboats	Weather permitting, vessels at facilities, use of crane, fish welfare, chemical use
Washing of float collar	Monthly	Hours	Moderate, service vessel/ workboats	Weather permitting, vessels at facilities, use of crane, manual labor
Fish sorting	Every six months	Hours	Moderate, wellboat, workboat	Weather permitting, fish welfare, manual labor
Deployment of smolt from wellboat	Annual	Hours	Moderate, wellboat	Weather permitting, vessels at facilities
Change of net	Annual	Hours/ Days	Moderate, workboat	Weather permitting, use of crane, manual labor on cage, vessels at facilities
Delivery of fish for slaughter	Annual	Hours	Moderate, wellboat, workboats	Weather permitting, fish welfare, manual labor on cage, use of crane

Chapter 4

Methodological Approach

This chapter describes the main tools and software we have used in this MSc thesis, as well as the different methods used to achieve our results.

4.1 Tools

myCBR[1] is a tool for creating Case-Based Reasoning applications. The tool is open source and Java-based, and it can easily be implemented in your own code. The user can choose between using the GUI, the API or both.

Weka[2] is an application with a collection of machine learning algorithms for data mining tasks. Weka is open source and Java-based. All the different algorithms can be applied directly to a dataset from the GUI or from your own Java code. Weka contains different tools for pre-processing, classification, regression, clustering, visualisation and more. The datasets can be loaded from files or a SQL database.

Eclipse[3] is an open source IDE, which is run by the Eclipse Foundation. Eclipse is famous for its Java IDE, C/C++, JavaScript and

[1] http://www.mycbr-project.net/index.html
[2] http://www.cs.waikato.ac.nz/ml/weka/
[3] https://eclipse.org

PHP IDEs. The program is written in Java, and its primary usage is Java applications.

MATLAB[4] is an appication by MathWorks for math, graphics and programming. MATLAB is widely used among engineers and scientists, and has many applications. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more.

Microsoft Excel[5] is a spreadsheet made by Microsoft. The program has many features, including calculations, graphic tools, tables and more. The program is extendable as the user can download data analysis tools, programming tools and more.

FhSim[6] is a software for simulating and visualising marine operations and systems, such as fishery, aquaculture and offshore. The applications of FhSim are training simulations, vessel design, machinery design, surveillance and more. The simulation software is developed by SINTEF Fishery and Aquaculture, and it is flexible and effective. FhSim can be run on several operating systems and can be linked with e.g. MATLAB and Java, and it is implemented in C++. It also has the advantage that it can be run in parallel on both GPU and CPU, with or without 3D visualisation.

4.2 Analysing & Modelling the Data

To achieve our goal of implementing a prototypical decision support system, it is necessary to perform a data analysis and to model the data. By doing this we are able to extract the patterns and information that will be useful for the application. When conducting the analysis, we follow a given method. The first step will be to acquire the data before pre-processing it. The next step will be to perform an exploratory analysis by visualisations and mathematical calculations,

[4] http://se.mathworks.com/products/matlab/index.html?s$_t id = gn_l oc_d rop$

[5] https://products.office.com/nb-no/excel

[6] http://www.sintef.no/fiskeri-og-havbruk-as/programvare/fhsim—simulering-av-marine-operasjoner-og-systemer/

which is a crucial part of getting an overview of the data. This will be the basis for creating the clean data sets that will be used when modelling the data.

The modelling of the data consists of creating symptoms that can be recognised in the data, as well as creating the cases. The symptoms are models describing different conditions that the sensors are able to register. Parts of creating these models require machine learning, and we are using Weka as our primary data mining tool, together with Excel and MATLAB. The last part of the analysis and modelling will be to create the cases for the CBR system, which uses the symptoms as attributes for similarity assessment.

4.3 Implementation

The implementation of our system is Java-based, and developed in the Eclipse IDE. We are using GitHub for version control which eases both development and cooperation. The implementation is described further in Chapter 8 Implementation.

4.4 Evaluation

When the implementation of the system is done, it is important to evaluate its performance. In our application the performance of the artificial symptoms and cases will be the most important parts to evaluate.

Evaluating the system with empirical data is difficult in this project, as we do not have any data about previous incidents. We do not have any textual reports either, which means that we have use optional methods. When evaluating the symptoms and cases that we create, the method will be discussion. At last we will evaluate the available sensors, also by discussion. For one of the symptoms created by machine learning, we are testing and evaluating different machine learning algorithms as well. These tests and evaluation will be the basis for choosing the machine learning method to use in that symptom.

Chapter 5

Data Analysis

In this chapter we analyse the data we have been given, to better understand the basis for the system we shall develop. We further use the generated data sets to extract useful information that are used in the implemented system. We first give a recap from the data analysis performed in the specialisation project, before discussing the changes and moving on to the current available data.

5.1 Specialisation Project

5.1.1 Previous Results

The previous data sets was provided by our co-supervisor Gunnar Senneset at SINTEF FH, and was fragments of a wind and wave sensor, as well as two Attitude and Heading Reference Systems (AHRS). It consisted of approximately four consecutive days of measurements, chosen randomly. The wind and wave sensor was measuring different properties of the wind and wave, and outputs data once per hour. It measured continuously, but the data has been aggregated heavily and only some statistical values are used to represent the profiles. The most important outputs was the wind direction and the wind speed.

The AHRS consisted of a gyroscope and an accelerometer. Both measured in the X, Y and Z directions, and the sampling frequency

was 10Hz. The measured values of the gyroscope are the change in rotation around the three different axes in radians, while the accelerometer values was the change in the three directions in g[1]. An overview of the sensors is shown in figure 5.1, and we had data from AHRS 1 and 2 which was attached on the plastic collar and the separate wind and wave sensor shown at the bottom of the illustration.

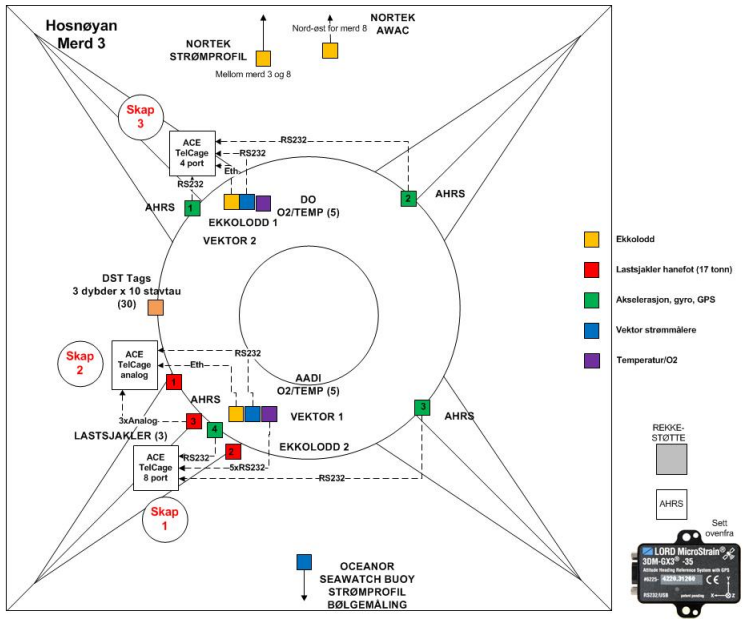

Figure 5.1: Overview of sensors at cage 3 on HosenÃÿyan production plant.

As the wind and wave data was highly correlated, we show the results from the wind speed in this recap. The correlation analysis between the wind speed and the max impact on the cage in the xy-plane indicates whether the two have an impact on each other. The correlation coefficient was calculated to 0.911 which indicated a high degree of linear dependency between the variables. Figure 5.2 shows the linear regression model of the relationship between the variables. The coefficient of determination r^2, which indicates how well the data fits the statistical model, is 0.83.

[1]The mean surface gravitational acceleration, 9.81 m/s^2

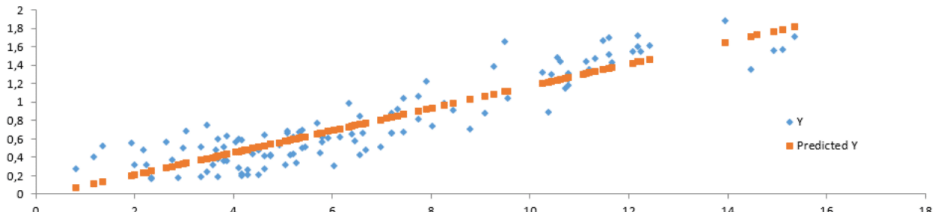

Figure 5.2: Linear regression model of the impact/wind speed data.

5.1.2 Challenges

During the work with the data analysis, we identified challenges with the data. We also felt that the data we had at that moment was not sufficient to develop a satisfying decision support system. This lead to making assumptions about the future sensors and available data, and how to represent cases. The main challenges were:

- **Data from healthy and unhealthy structural states** are needed to identify the structure's condition.
- **States, situations or operations to use as cases** are necessary to create a case base, and to use previous experience about similar incidents.
- **Expert knowledge and incidents reports** to support creation of cases, and provide threshold values, descriptions and solutions.
- **More sensors** to provide more useful information about the structural health of the cage.

The new data analysis will show whether it is possible to solve these challenges or if we have to find different solutions. A big difference from the specialisation project is that we no longer have access to AHRS sensors, as our data is from a different production plant. The new data and sensors are explained in the next section.

5.2 Data Acquisition

The available data provided by our co-supervisor Gunnar Senneset at SINTEF FH is fragments from a wind, wave and current sensor (ACE buoy), and three load shackles. For the rest of this report, the term environmental data refers to wave, wind and current data, weather data refers to wind and wave data, and shackle or strain data refers to the shackle data. We have approximately 47 consecutive days of measurements, from mid February to end of Mars 2016.

The ACE buoy is measuring different properties of the wind, wave and current, and outputs data once per hour. It measures continuously, but only some statistical values are used to represent the profiles. The outputs are:

- **resultTime** - The date and time of the measurement.
- **currentDirection** - Current direction in different depth layers, measured in degrees.
- **currentSpeed** - Current speed in different depth layers, measured in cm/s.
- **hm0** - Significant wave height in meters, which is the average height of the 1/3 highest measured waves within the last hour.
- **hmax** - Highest measured wave in meters, higher than a minimum threshold.
- **mdir** - Wave direction in degrees.
- **tp** - Average wave period in seconds.
- **WindDir** - Wind direction in degrees.
- **WindGust** - Maximum wind gust last hour in m/s.
- **WindSpeed** - Average wind speed last hour in m/s.

The load shackles are measuring the tensile loads between the cage and the mooring system. The shackles provides three unique measurements, as they are mounted between the attachment points of the rooster foot and the cage. The unit of the measurement is tonnes, and the sampling frequency is 1Hz. In addition, the load shackles stores information about the plant ID, sensor ID, unit, time stamp. An overview of the sensors is shown in figure 5.3, and we currently have data from load shackle 1, 2 and 3 attached on the plastic collar.

The separate wind, wave and current sensor are shown at the bottom of the illustration.

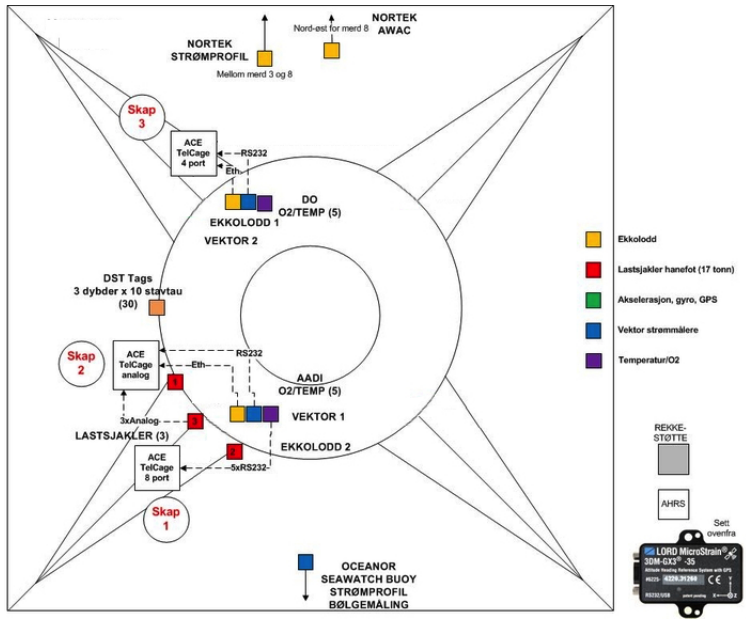

Figure 5.3: Overview of the sensors at cage 7 on Rataren production plant.

Raw data files

The raw data file of the ACE buoy contains data sampled every hour from from mid February to end of Mars 2016. A fragment is shown in Figure 5.4. The low output frequency limits the accuracy, but the measurement is representative as a statistical value for the weather within the given hour.

The sampling frequency of the shackle data is higher than for the Ace buoy, which make the detail level higher in this data set. This causes that we have to aggregate the entries with respect to the task to be done, which we come back to in the next chapters. The entries from all the available shackles are originally in the same csv-file, and a fragment is shown in Figure 5.5.

resultTime	currentDire	currentSpe	hm0	hmax	mdir	tp	WindDir	WindGust	WindSpeed
2016-03-13 14:00:00	203,906	3,516	0,195	0	202,5	8,984	220,781	5,127	4,16
2016-03-13 15:00:00	165,938	12,891	0,195	0	178,594	9,18	198,281	6,973	3,691
2016-03-13 16:00:00	191,25	8,203	0,156	0	194,063	8,691	211,641	5,4	4,688
2016-03-13 17:00:00	203,906	3,516	0,156	0	174,375	8,496	195,117	5,605	4,512
2016-03-13 18:00:00	39,375	5,859	0,156	0	175,781	8,594	225,352	9,844	7,793
2016-03-13 19:00:00	33,75	11,719	0,234	0	213,75	8,789	234,492	12,441	9,961
2016-03-13 20:00:00	57,656	18,75	0,313	0	196,875	2,051	241,875	14,424	11,25
2016-03-13 21:00:00	56,25	32,813	0,313	0	206,719	2,246	245,039	13,604	10,957
2016-03-13 22:00:00	50,625	28,125	0,352	0,469	198,281	2,637	244,336	14,287	11,25
2016-03-13 23:00:00	53,438	31,641	0,352	0,391	240,469	3,125	246,094	14,287	11,426
2016-03-14 00:00:00	50,625	26,953	0,352	0,391	240,469	2,246	252,422	13,672	10,43
2016-03-14 01:00:00	56,25	19,922	0,352	0,391	250,313	2,637	250,664	13,467	10,547
2016-03-14 02:00:00	60,469	17,578	0,391	0,469	223,594	2,344	246,797	11,963	9,375
2016-03-14 03:00:00	120,938	8,203	0,43	0,469	227,813	2,93	245,742	12,51	9,668
2016-03-14 04:00:00	126,563	11,719	0,391	0,469	230,625	2,246	242,93	13,262	10,254
2016-03-14 05:00:00	75,938	2,344	0,352	0,391	225	3,809	240,469	13,467	10,137

Figure 5.4: Original data set from the wind, wave and current sensor (ACE buoy).

R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.056763	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.049435	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.050767	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.053099	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.057096	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.043772	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.057762	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.050767	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.048435	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.054098	Tonne
R03 Cage Rataren 1	Load shackle 99873	16.02.2016 02:00	UTC	0.046437	Tonne

Figure 5.5: Original data set from the load shackles, but because of limited space for the illustration we have removed 19 currentDirection and currentSpeed columns.

5.3 Data Preparation

The preparation of the raw data is an essential step for the further analysis. If the preparation is done unsatisfactory according to our requirements for accuracy, usefulness and fidelity, the analysis results will turn out badly and so for the project.

5.3.1 Weather

The weather data consists of measurements from the ACE buoy, and gives information about the current at several depth layers, the wind and wave. The file structure is described above. The work with this data set is removing the unnecessary features, and aggregating the current speed and direction as one total value instead of one value per depth layer. These operations were done in Eclipse by running Java-code as it saves us a lot of time compared to Excel or similar tools. The reason for aggregating the current speed and direction measurements from the different depth layers, is to create a simplified attribute for this feature. The aggregated attribute gives the current measurement a similar representation as the wind and wave, as the latter do not have different heights or depths.

When aggregating the current measurements, we keep four of the total 20 depths which is representative of the current. These are the measurements at 5, 9, 13 and 17 meters. The final aggregated attribute of the speed is the average of these four depth layers, and the average direction is calculated by the Java code in Listing 5.1.

```
directionX = (Math.cos(depth5.get(id))
  + Math.cos(depth9.get(id))
  + Math.cos(depth13.get(id))
  + Math.cos(depth17.get(id))) / 4;
directionY = (Math.sin(depth5.get(id))
  + Math.sin(depth9.get(id))
  + Math.sin(depth13.get(id))
  + Math.sin(depth17.get(id))) / 4;

direction = Math.atan2(directionY, directionX);

if (direction < 0) {
  direction += 360;
}
```

Listing 5.1: Calculation of average current direction in Java.

In the Java code above we calculates the average direction by first calculating the average cosine as the X direction, and the average sine as the Y direction. At last we calculate the arctangent to the two average directions.

5.3.2 Shackles

The shackle data contains information of the tensile load on each shackle. For analysis and visualisation purposes we have split the complete data set into six new sets, one per shackle. This gives us a better opportunity to check the data for errors and to easier understand it. We could further compare each of the shackles to each other and the weather data. When done analysing, we create a new data set containing all the shackle information as a whole.

We have mostly used a frame from 28.03 to 26.04 in this project, as the measurements here was the most complete without a lot of missing or corrupt values. We could predict the missing or corrupt values, but as this window was nearly complete we have chosen to not construct any fake data.

When visualising each of the measurements from the different shackles, we noticed that some of the shackles have a lot of entries with tensile value of -0.05, as well as some had extremely low variations. The latter is shown in Figure 5.6. We discussed these entries in a meeting with Area 3 project leader David Kristiansen from SINTEF FH, where we learned that the reason for the -0.05 entries most probably was loose mooring. This is illustrated in Figure 5.7. The consequence of loose moorings is that the shackles will register low strains as -0.05, as the sensor is not calibrated for it. The total mooring system may loosen over time, either by environmental influence like strong current or sudden waves causing the anchor to slip, or by natural extension of the mooring. We will explain the consequences of a dynamic construction further in Section 6.1. The reason for the measured values with low variation is most probably an error on the sensor.

46	Load shackle 85129+2016-02-28 00:46:02+-4.078589
47	Load shackle 85129+2016-02-28 00:47:02+-4.078589
48	Load shackle 85129+2016-02-28 00:48:02+-4.0785216
49	Load shackle 85129+2016-02-28 00:49:02+-4.0784542
50	Load shackle 85129+2016-02-28 00:50:02+-4.078589

Figure 5.6: Segment of one of the data sets where a lot of the values have low variation and too high measured values. The tensile load is the last column.

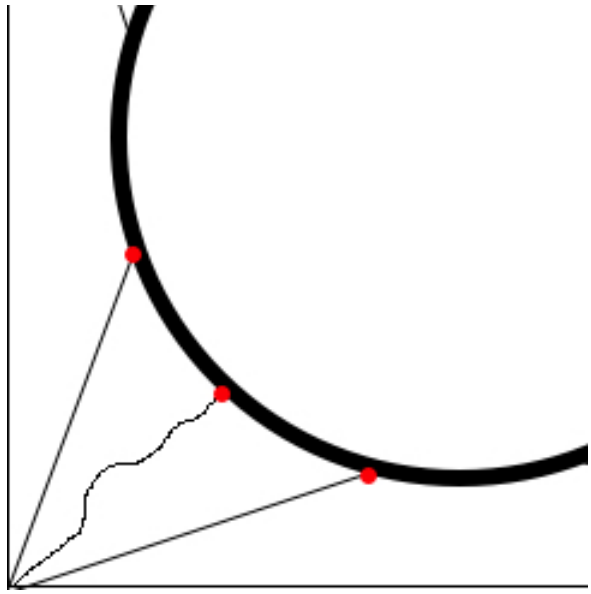

Figure 5.7: Illustration of one loose mooring.

The direct consequence for the shackles that contains a lot of entries with low variations is that we have decided to not include them in the further analysis and system implementation. This problem was specific for one of the rooster foots. The main reason for leaving them out is that we need data from all three shackles on a rooster foot to be able to say something about the structural health.

The operations done when preparing the remaining shackle data was done in order to remove possible noise and to represent the true load like in signal processing. Hence to the Shannon-Nyquist sampling theorem, we downsampled the measurements to 1/3 of the original frequency before smoothing the data [Smith, 2016]. From these 20 remaining values per minute we calculated the average of the five highest entries, making the new filtered attribute value per minute. By doing this we have removed the -0.05 values, and reduced the noise by averaging the highest values.

The advantage of keeping the values at a detail level of one entry per minute is that we have the opportunity to extract information about events within an hour. As the weather data is sampled every hour, we also aggregated the shackle data to one entry per hour for

comparing purposes. This aggregation was done by calculating the average of the above mention values per hour.

5.4 Visualisation and Correlation

The data sets produced in the pre-processing step above, are ready for visualisation and correlation analysis. As mentioned earlier, we have chosen a time frame from 28.2 to 26.3, and all further analysis are within this space.

Visualisations of the wave and wind data are shown in Figure 5.8 and 5.9, where the blue curve represents the wave and the orange curve represent the wind. The first figure shows the wave and wind speed, and the second shows the direction.

The next figures visualise the current and wave data, where Figure 5.10 shows the speed and Figure 5.11 shows the direction. In both figures the blue curve shows the current and the orange shows the wave.

Visualisations of the sensor data make it easier to interpret how the relationship between the wind, wave and current works, and roughly tell if some of them have an impact on each other. By studying the visualisations we can see that the wind and wave data have similarities along the curve, and most of the tops and bottoms follow each other in different speeds and directions. The current and wave visualisations on the other hand does not follow each other at the same level. This makes it reasonable to believe that the wind vs. wave are more related than the current vs. wave and the current vs. wind. These assumptions should be further investigated by doing a correlation analysis.

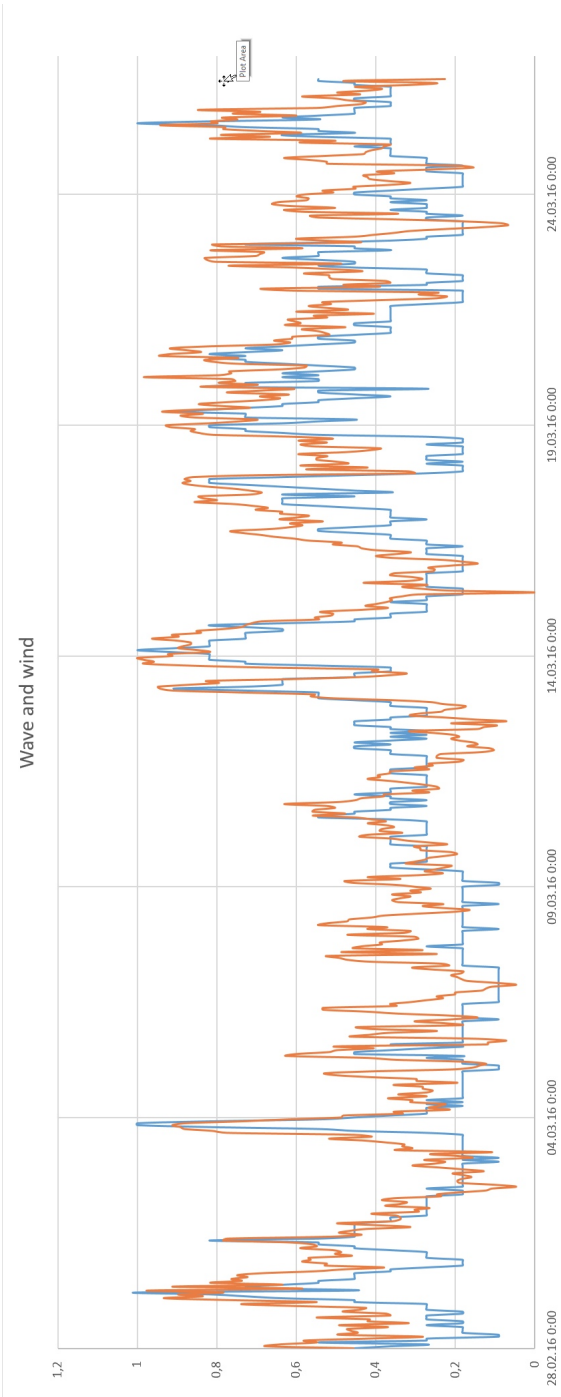

Figure 5.8: Visualisation of the wave height and wind speed.

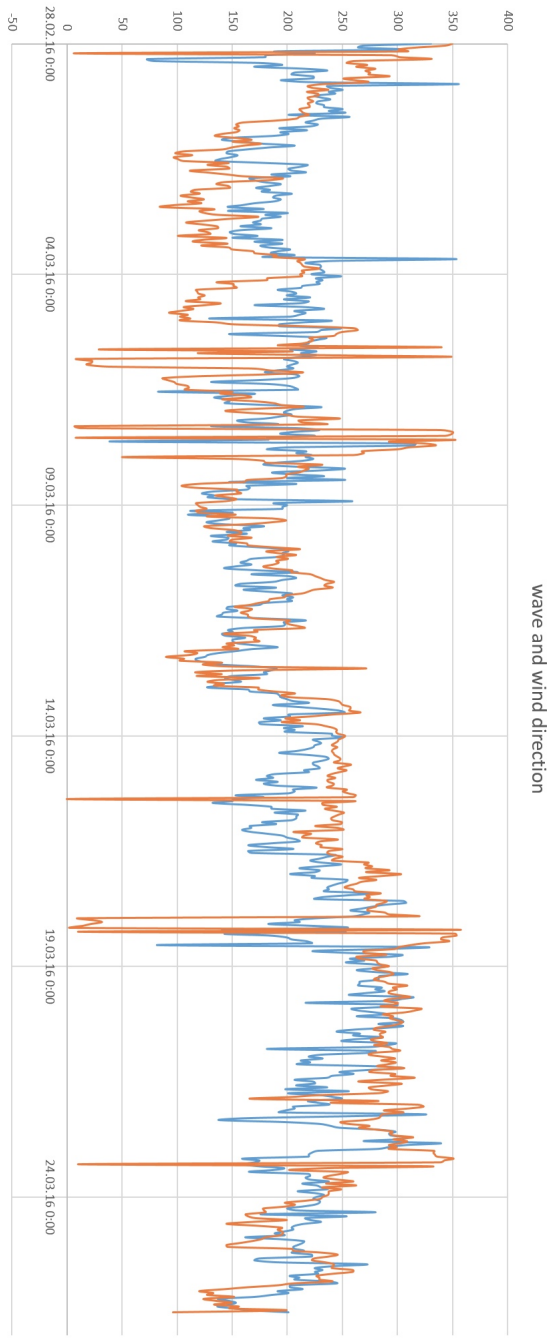

Figure 5.9: Visualisation of the wave and wind direction.

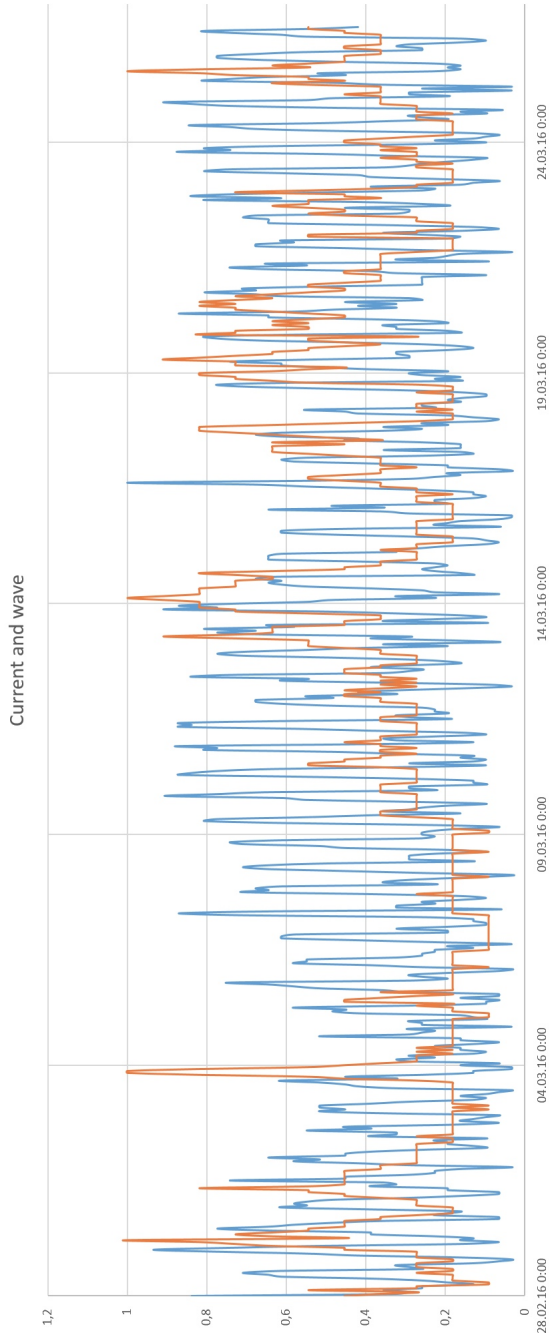

Figure 5.10: Visualisation of the current speed and wave height.

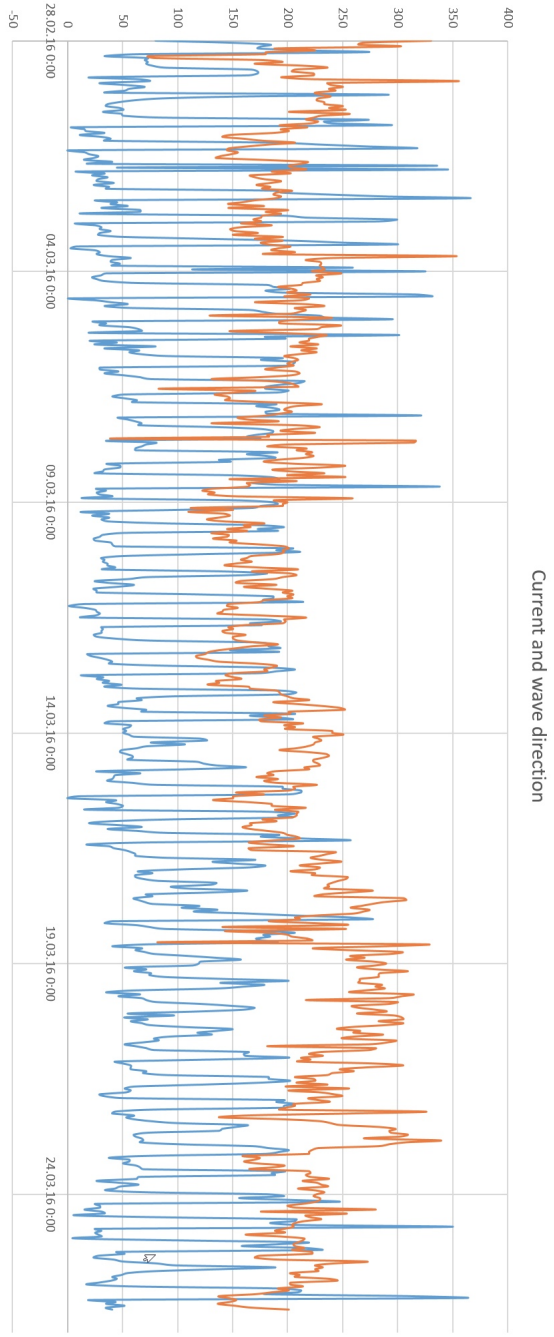

Figure 5.11: Visualisation of the current and wave direction.

Visualisations of the shackle data are shown in Figure 5.12, 5.13 and 5.14. The blue curve shows the strain per minute in the mentioned days, where the unit is tonnes. The first figure shows data from shackle 85130, the second shows shackle 99871 and the third shows shackle 99875.

The next figures visualise the wave and current data and shackle data in the same plot. In the plots shown in Figure 5.15, 5.16, 5.17 and 5.18 the data is normalised as the two set are not in the same scale. In all four figures the blue curve shows the environmental data and the orange shows the shackle data.

By studying the visualisations of the shackle data plots, we can see that the strain measurements varies in strength and that the last plot differ from the others. The reason for having different measurements on the three shackles are most probably that the mooring is tightened differently on each of the shackles as discussed in Section 5.3.2. This makes it difficult to rely on and make use of the measured value on each of the shackles, as all the measurements vary from time to time. A flexible structure like the cage and mooring system will also change over time by both wear and maintenance.

The lowest measured value of the shackle data is -0.05. This threshold means no registered strain. An exception is the shackle 99875, where we have measured values down to -1.12. This is because all the values are registered with an opposite sign. We have chosen to plot the original values in Figure 5.14 and to multiply the values with -1 in Figure 5.18 for visual purposes and the further analysis.

The visualisation of the wave data and shackle data together shows that there are few similar patterns in the two curves. It seems like the current data curve and the shackle data curve follow each other far better, as the tops and bottoms are at similar points on the x-axis that indicates the time of the measurements. It is therefore reasonable to believe that the current and the shackle data are more related, and that the wave and wind data is less related to the shackle data. These assumptions are the subject of the correlation analysis, and we should investigate further why or why not the different data is related. It is crucial to understand which environmental forces that has the most impact on the cage to be able to create symptoms, cases and especially for the usage of machine learning.

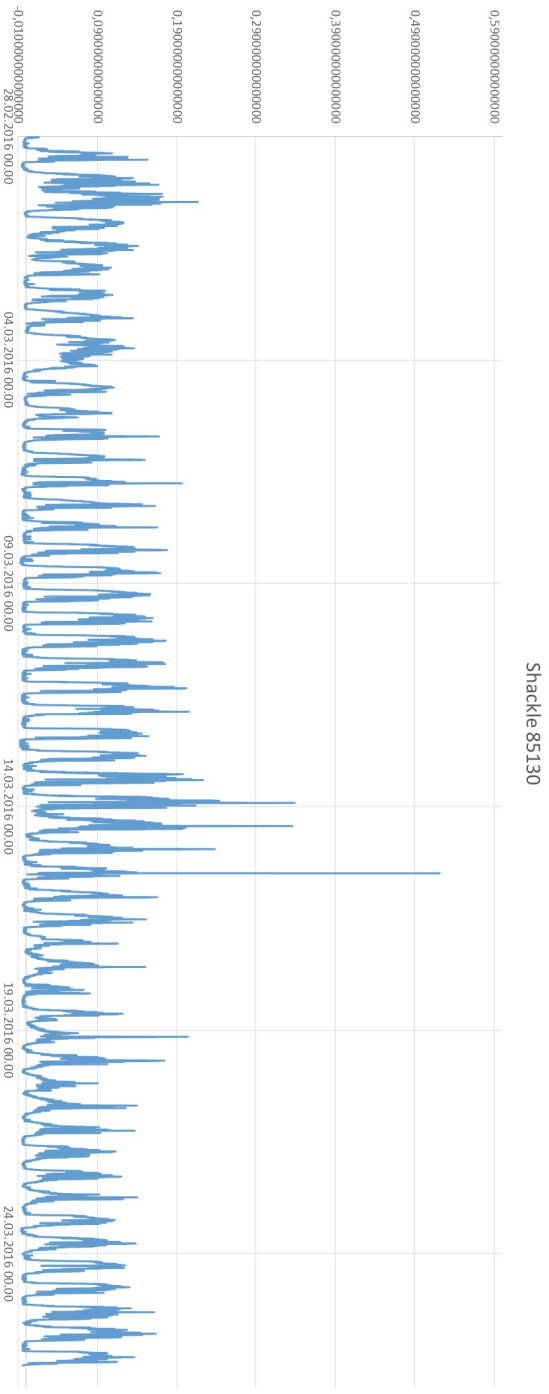

Figure 5.12: Visualisation of the strain measured on shackle 85130.

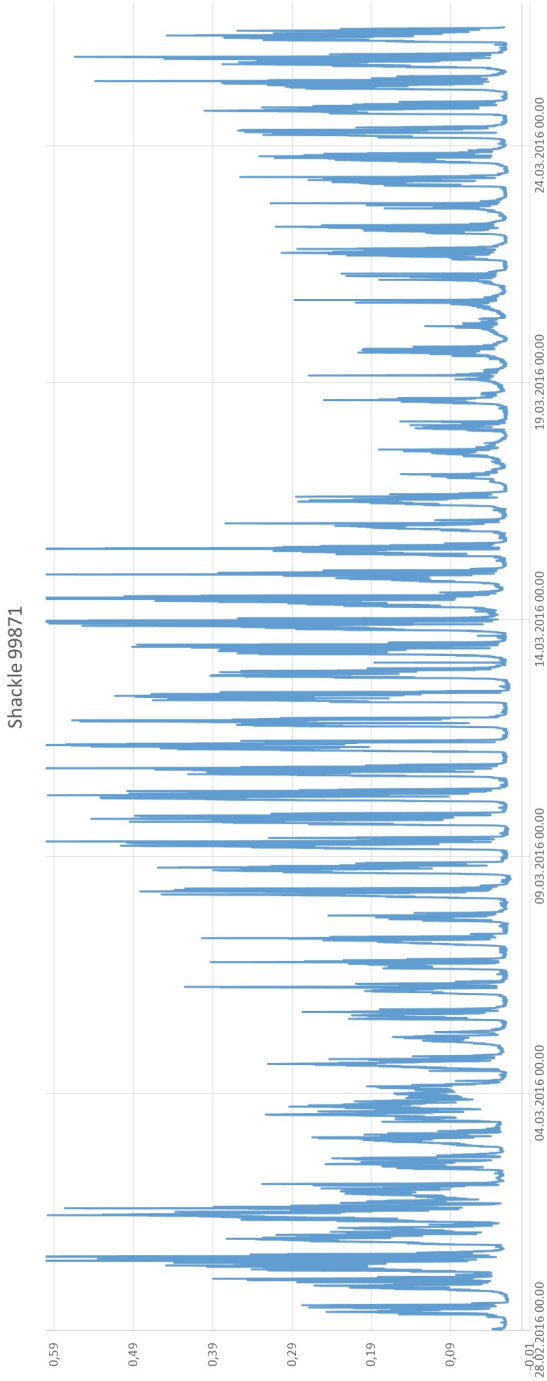

Figure 5.13: Visualisation of the strain measured on shackle 99871.

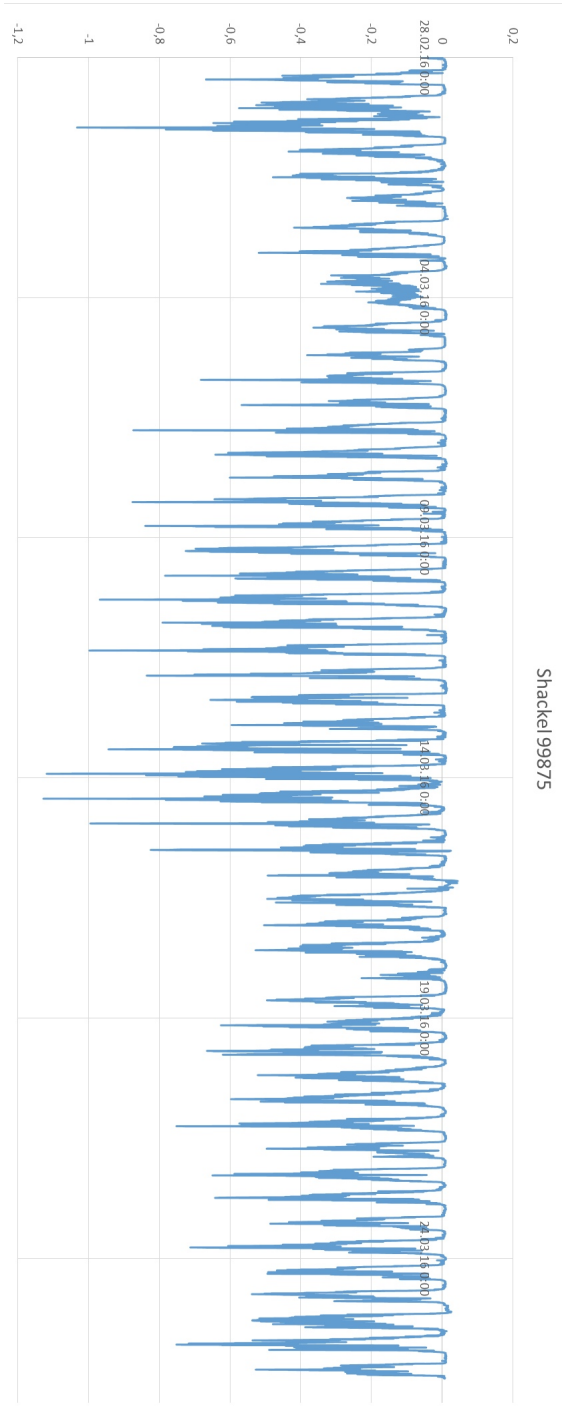

Figure 5.14: Visualisation of the strain measured on shackle 99875.

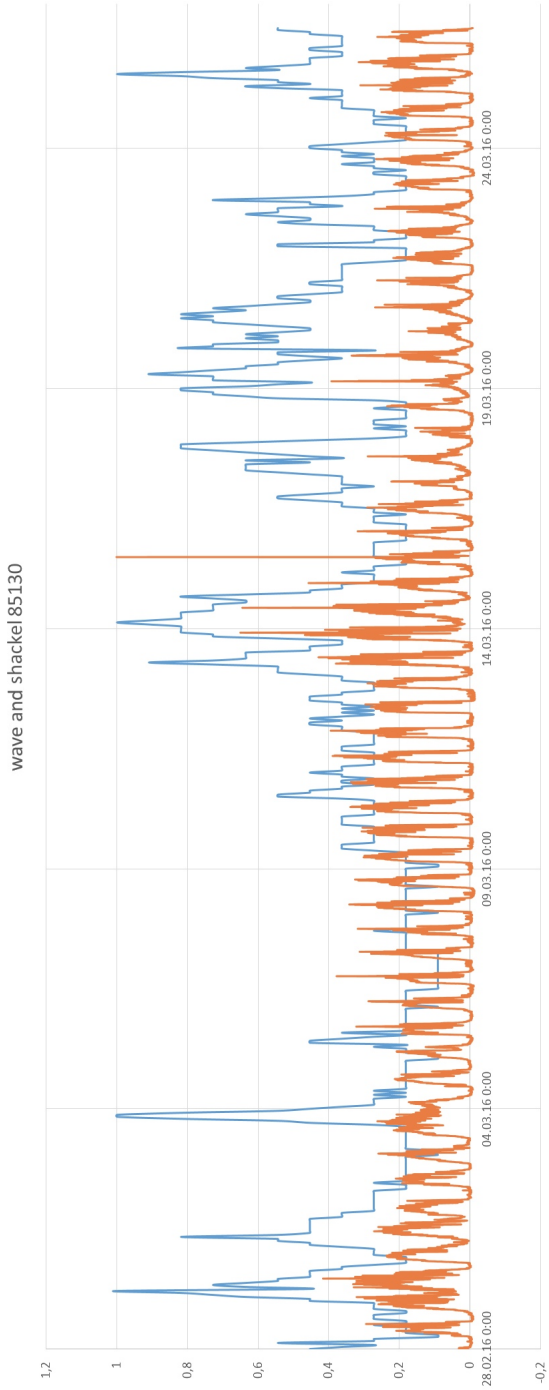

Figure 5.15: Visualisation of the wave and shackle 85130 data.

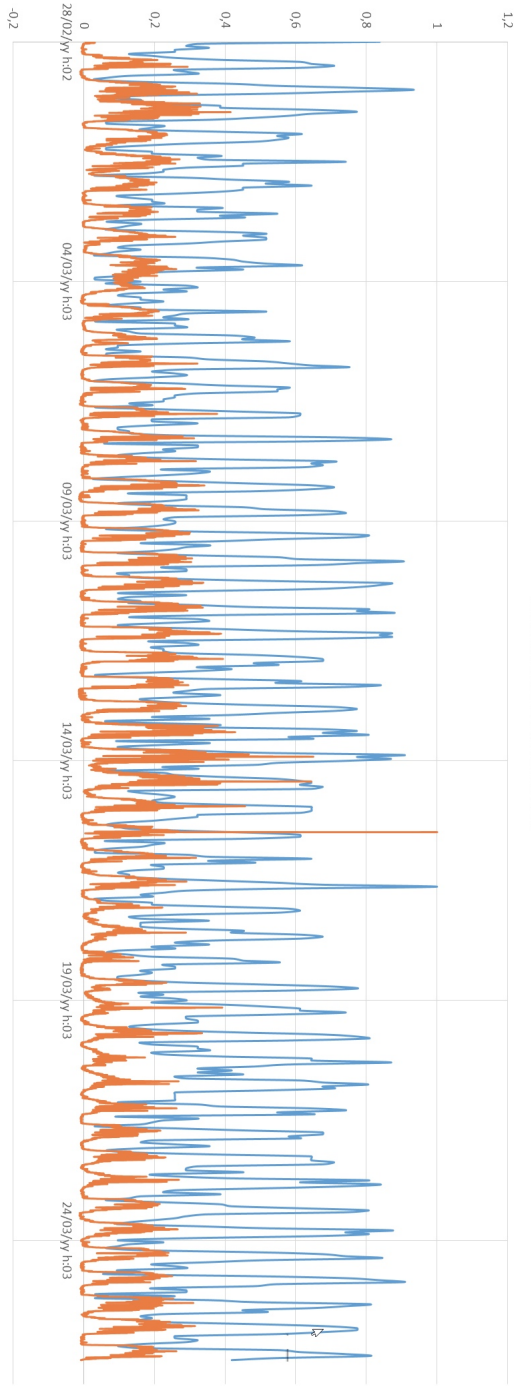

Figure 5.16: Visualisation of the current and shackle 85130 data.

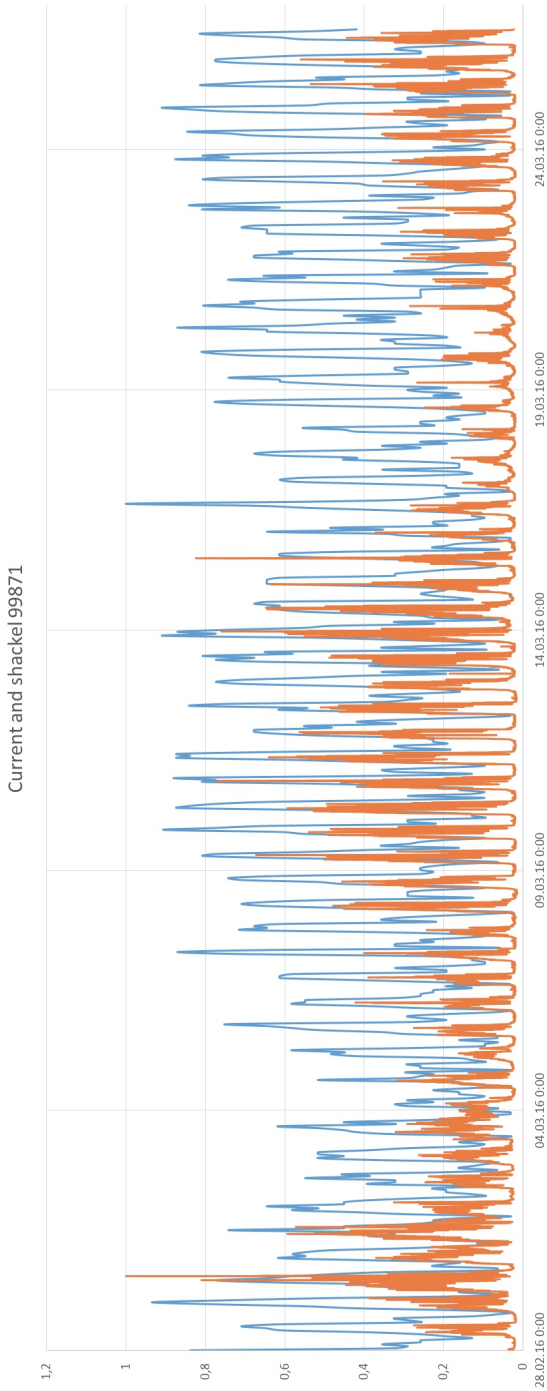

Figure 5.17: Visualisation of the current and shackle 99871 data.

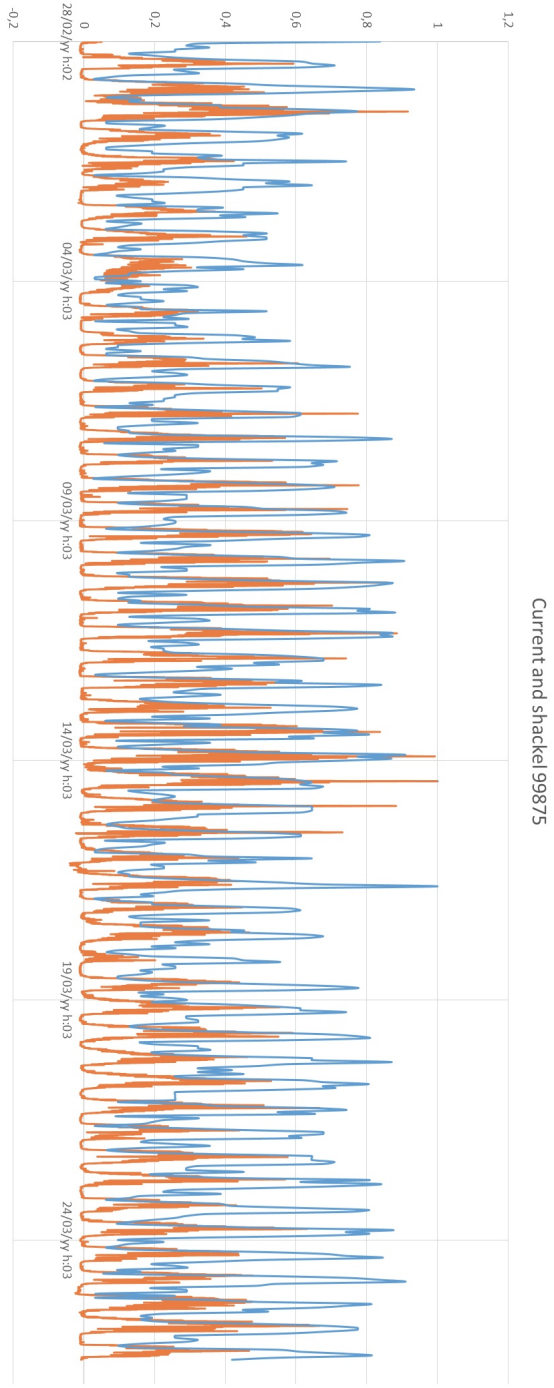

Figure 5.18: Visualisation of the current and shackle 99875 data.

To investigate the relationship between the different data, we calculate the correlation coefficient. The correlation between the data available are presented in Table 5.1. By looking at the visualisations of the data, we can see a slight phase shift between the shackle data and the environmental data. By displacing the data according to this phase shift, the correlation increases. These correlation coefficients are the ones showed in the table. As the correlation coefficient shows that the highest correlated data is the shackles and the current, it is a high degree of linear dependency between this data. We can by the numbers in the table say that the wind and wave have small or no direct impact on the sensors available at the moment.

We have previously shown in the specialisation project that the wind speed has an impact on the maximum impact on the accelerometers, but unfortunately we do not have these sensors available. This is explained in Section 5.1. A reason for not registering the wind in the shackle entries is the placement of the different sensors. The shackles are connected to the cage near or under the water, and the accelerometers are mounted at the top of the cage. It is therefore reasonable to say that the wind has a lot more affection on sensors above sea level, which are more exposed to this kind of impact. We have also learned in meetings with SINTEF that the accelerometers are much more sensitive to impact than the strain.

The table shows that the wind and wave are highly correlated and the wave/wind and current are not. This means that when trying to predicting the strain by machine learning, the data from the wind and waves should not be a part of the training data.

As the correlation between the current and the shackles are high, we should investigate why. As we can see in Figure 5.19 showing the current and shackle data from two days, the data have tops and bottoms at the same time. This is consistent for all the data available. By doing research on sea currents, we found that current is strongly related to the tide. Tides are driven by the gravitational force of the moon and sun [NOAA, 2015]. One can see the tide by the changing water level. The tide creates currents as it creates movement in the ocean. This current is called tidal currents and can be predicted as they change in a very regular pattern. Other factors that create currents are the wind and thermohaline circulation. The wind creates current that are near the surface, and the thermohaline circulation creates both deep and shallow currents. As the production plant used

in fish farming are massive and can be more than 25 meter deep, the tidal current are the most interesting. The surface current is interesting if we have more sensitive sensors, and the current created by thermohaline circulation does not influence the cage as it moves much slower.

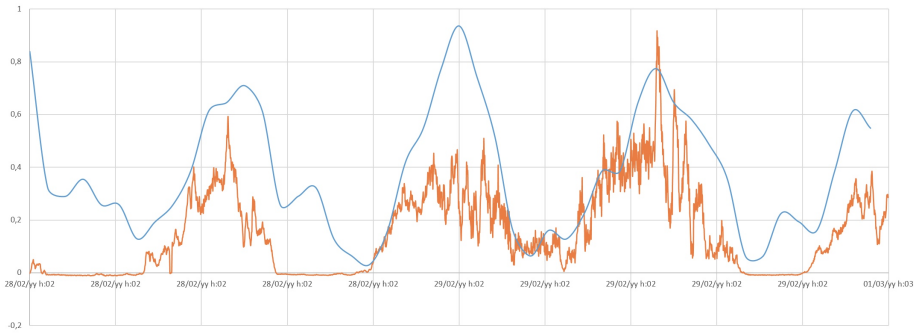

Figure 5.19: Visualisation of the current and shackle data from two days.

By analysing tide data from Rataren where the production plant is located, we see that the tide is highly correlated to the current data. Measured tide data from Kartverket is available online [2]. The correlation coefficient is 0.74, and Figure 5.20 shows a plot of the current data and tide data. Rataren is a location exposed to currents, which is shown in Figure 5.21.

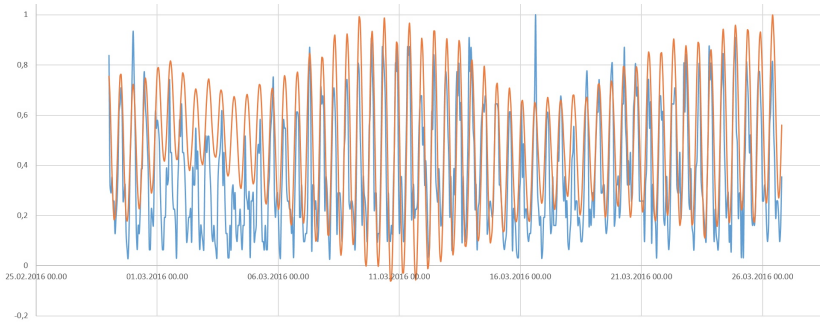

Figure 5.20: Visulalisation of the current and tide data.

[2] http://www.kartverket.no/sehavniva/sehavniva-lokasjonside/?cityid=221996city=Rataran

Table 5.1: Overview of the correlation coefficients, with the data shifted one hour.

	Shackle 85130	**Shackle 99871**	**Shackle 99875**	**Wind speed**	**Wind dir**	**Wave height**	**Wave dir**	**Current speed**	**Current dir**
Shackle 85130	1								
Shackle 99871	0.923	1							
Shackle 99875	0.915	0,806	1						
WindSpeed	0.065	-0.25	0.159	1					
WindDir	-0.072	-0.182	0.082	0.510	1				
WaveHeight	0.073	0.025	0.113	0.776	0.351	1			
WaveDir	-0.152	-0.275	-0.012	0.57	0.471	0.449	1		
CurrentSpeed	0.736	0.655	0.820	0.113	0.073	0.084	0.030	1	
CurrentDir	-0.659	-0.567	-0.600	-0.048	-0.053	-0.024	0.048	-0.543	1

Figure 5.21: Location of the Rataren, Tristeinen and Korsneset production plants [ACE, 2016].

The visualisations and correlation analysis show that we should focus on the current and shackle data when diagnosing the structural health of the cage. As the different shackles on the same rooster foot gives different strain values, we must focus on characteristics of the curve and the statistics. The wind and wave data can still be used to tell whether it is safe or not to do manual inspection, maintenance and other work on the platform, as the workers can not be on the platform in too rough conditions.

5.4.1 Challenges

When conducting the data analysis, we met some challenges with the available data. Some of them are similar to the ones that are mentioned in the start of this chapter, which we aimed to find solutions for in this analysis. The important challenges are:

- **Usage of FhSim to simulate data** and use information about a healthy and unhealthy state to identify the structural condition.
- **Limited time to work with the available data** as the sensors was up and running first in February, and we could not start working with the data before April. This limits the scope of

the analysis, modelling and implementation drastically as these tasks are dependent on the data.

- **The amount of, and the information in the usable, available data does not provide enough information** to create real symptoms and cases. These should optimally be based on real events and incidents. This implicates that the symptoms and cases that we create will be artificial, and based on knowledge gathered by meetings with experts and studies. The loss of the AHRS sensors was also a big drawback as these provided information that could be very useful.
- **Evaluating the system with empirical data can be difficult** due to lack of previous experience about similar incidents. We do not have any textual incident reports either.

We hoped to use FhSim to simulate an unhealthy state of the cage, and then compare the simulated strain with real shackle data. An indication of a structural damage could then be too high deviation between the simulated and measured strain. The principle is illustrated in Figure 5.22, where (a) represents a healthy heart rate, (b) an unhealthy one, and (c) represents the anomalies found by subtraction.

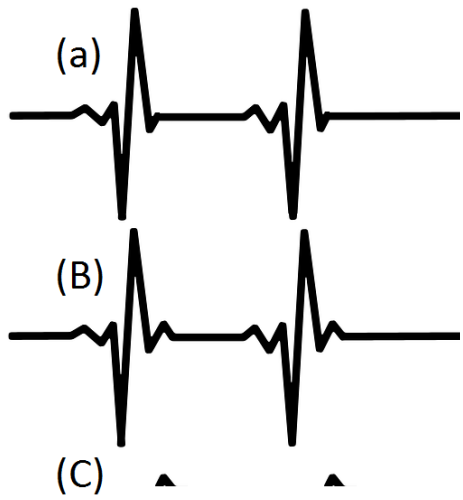

Figure 5.22: Heart rates and anomalies found by subtraction: (a) healthy heart rate, (b) unhealthy heart rate and (c) anomalies.

The problem with FhSim is that we can not simulate a specific sensor value, but rather the force in a given point on the cage. That force can be directed in any given direction. As our shackles only measure the strain in its own direction, which we know nothing about, we can not relate the two values. As we do not have any possibilities of simulating a value that we can register by any of the available sensors, we can not use the idea mentioned above.

FhSim is not implemented in our system, but we got some ideas from analysing the output from the simulation of a broken mooring. We manually removed one or more moorings, and analysed the results. This confirmed our assumption of that if one or more moorings are broken, the force on the remaining mooring was notably higher.

We thought of using the measured difference in force as a reference for the shackle data, but as mentioned we do not have any information of the direction of the shackle. A small variation in the direction would make a big difference in the simulated force and the measured value.

As the symptoms and cases that we create will be artificial, we are defining the threshold values, descriptions and solutions ourselves. This information will be based on knowledge gathered in meetings with experts and studies, as mentioned above. As the data was available to us for a limited time only, we had more time to do research within the domain. This makes us more capable of creating reasonable symptoms and cases.

As we do not have empirical data to use when evaluating the system, we have use optional methods. When evaluating the symptoms, cases and available sensors, the method will be discussion and machine learning. This is further described in Chapter 4 Methodological Approach.

Chapter 6

Modelling

In this chapter we are modelling the data that has been pre-processed and analysed in the previous chapter. The models we are creating are individual symptoms that represent some kind of state that can be recognised in the data. Further, we are forming the cases that are used in the CBR-system.

6.1 Symptoms

A symptom is a predefined signal recognised in a given data set. As mentioned, a symptom is a model of a state, and the signal is any kind of pattern in the data that describes the state. To be able to recognise these patterns, we need a data set with relevant attributes. These data sets are the results from the data analysis described in the previous chapter.

The symptoms we create uses sensor data as input, and the output is a descriptive value of the state the symptom represents. Inside each symptom there are performed calculations on the input before deciding the output based on thresholds.

As the system should be developed modular and extendable, we have defined some general mathematical calculations that can be used in every symptom. The point is that when creating a new symptom, we have a tool box of functions to assemble it. The following functions have been chosen with basis in the conducted analysis above:

- Calculation of maximum
- Calculation of minimum
- Calculation of average
- Calculation of slope

These functions can be calculated for every input, e.g. the current speed or wind direction. We can also specify the time length for the calculation, e.g. the max within the last two or ten hours.

A symptom may be signalised as present if the value calculated from the functions above reaches a threshold. The symptoms must generally be developed according to the domain and available data. An example of an implemented symptom is shown in Listing 6.1. The output of presented symptom is the status of all shackles on a rooster foot, which uses the shackle data as input. We calculate the maximum value of each shackle within a given time length, and the status is bad if one or more shackles have a maximum value less or like zero. The status is good if else.

```
shackleStatus(shackle1data, shackle2data, shackle3data, timeLength){
    shackle1Max = 0;
    shackle2Max = 0;
    shackle3Max = 0;

    shackle1Max = calculateMax(shackle1data, timeLength);
    shackle2Max = calculateMax(shackle2data, timeLength);
    shackle3Max = calculateMax(shackle3data, timeLength);

    shackleStatus = "good";

    if(shackle1Max <= 0 || shackle2Max <= 0 || shackle3Max <= 0){
      if(shackle1Max <= 0){
        shackleStatus = "bad";
      }
      if(shackle2Max <= 0){
        shackleStatus = "bad";
      }
      if(shackle3Max <= 0){
        shackleStatus = "bad";
      }
    }
    return shackleStatus;
  }
```

Listing 6.1: Example of the implementation of a symptom. Types and notations are removed for better readability.

We have focused on creating relevant symptoms for our domain. Based on the challenges and resulting data sets from the analysis, we have created six artificial symptoms. All of them have been created by discussing different states that could possibly occur. The thresh-

old values in the symptoms are chosen by discussing the levels with domain experts and analysing the minimum, maximum, median and mean values for the different data. As mentioned above, the symptoms are artificial which means that the threshold values do not necessary represent any real incidents. In Chapter 9 Evaluation we discuss this challenge further. In the first four symptoms we are using the functions mentioned above as the decision basis for the state. The fifth symptom does not use the functions, but instead we are using machine learning to classify the state. The sixth symptom is different than the others, as it is only present when the operator gives a signal. The following section shows the symptoms:

Symptoms with state based on functions:

- Shackle status
- Weather condition
- Deviant slope between the current and shackle data
- Current speed

Symptoms with state based on machine learning:

- Deviant data from past similar conditions

Other:

- Manual work

The next subsections describe each of the listed symptoms more specific, where we explain the mathematics and logic behind them.

6.1.1 Shackle Status

The shackle status symptom indicates whether there is a problem with one or more of the shackles. The input is the shackle measurements, and the output is a number which indicates how many shackles that are out of order. Within this symptom the maximum measured value of each of the shackles within a predefined time span are calculated. The time span in this symptom is typically the last one or two hours. An error is counted if the maximum value is less than zero. This means that the measured value is -0.05, which may indicate that the sensor is broken or that the mooring is loose or has snapped.

6.1.2 Weather Condition

The weather condition symptom indicates whether if it is safe or possible for the operator to perform manual inspection, maintenance and other work on the platform. The input is the wind speed and wave height as these factors affect the upper parts the most. The output is a textual description of how risky it is to stay on the platform, which is determined by threshold values for the mentioned inputs. Currently the weather condition is indicated as:

Extreme if

- wind speed is more than 12 m/s or
- wave height is higher than 1 m

Bad if

- wind speed is more than 7 m/s or
- wave height is higher than 0.5 m

Else good

This symptom uses the most recent measurements as the operator must know the present conditions.

6.1.3 Deviant Slope Between Current & Shackle Data

This symptom indicates whether the deviation in slope between the current data and shackle data is larger than a given threshold. The input is current speed together with the data from all three shackles. The time span for this symptom may vary between two and twelve hours, depending on how far back we want to calculate the trend. The slope of the input data is calculated, and the current slope is compared to all three shackle slopes. The output is a textual description displaying ”low” or ”high”. The latter is indicated if one or more of the calculated slopes deviates more than the given threshold. This threshold is set to 30% deviation in our implementation. This symptom indicates faulty sensors or mooring like the shackle status symptom, but it is calculated using different calculation functions. It is important to have redundant calculations to ensure that the most crucial parts of the platform are reliable.

6.1.4 Current Speed

The current speed symptom indicates whether the current rises to risky levels. The input is the current speed, and the output is a textual description of how dangerous the current level is at the moment. The current speed is currently indicated as:

Low if

- current speed is lower than 8 cm^3/s

Medium if

- current speed is lower than 20 cm^3/s

Else High

High currents may be problematic in many situations, and monitoring this data could be useful for the operator.

6.1.5 Deviant Data from Past Similar Conditions

The deviant sensor data symptom calculates the deviation between the sensor data and past similar conditions. The latter is based on machine learning, as we want to build a model to classify the shackle data based on the current speed and the current direction.

The model will be trained on previous measurements from the current and shackle data. When the model is built, its input will be recent measurements of the current speed and current direction, and the output will be a classification of the strain as either class 1, 2 or 3. The classes represent different strain strength, where class 1 represents low values, class 2 middle values and class 3 high values. The measured strain from the shackle data is converted to classes as well. At last, we compare the shackle data with the classifications from the machine learning model. The result from the comparison is the number of matching classes. We also count the number of instances classified too low and too high, according to the measured class. Depending on these numbers, the following textual description is given:

Good if

- the number is higher a threshold defined by the evaluation done by a test set when the machine learning model is trained.

Under if

- the number is lower than the threshold for good accuracy, and the majority of the instances are classified too low.

Over if

- the number is lower than the threshold for good accuracy, and the majority of the instances are classified too high.

The input in this symptom is the current data and the shackle data from only one shackle, and the output is the textual description given by the comparison mentioned above.

We have used Naïve Bayes as our machine learning algorithm to create the model. The tests and evaluation of different machine learning methods are given in Chapter 9 Evaluation.

6.1.6 Manual Work

The manual symptom indicates whether some personnel performs manual work on the production plant. This symptom is signalised by the operator, as he must push a button when someone conducts manual work. The output is a binary signal indicating whether manual work is performed at the moment or not.

6.1.7 Excluded Symptoms

During the data analysis process we discussed a lot of other symptoms. Some of them were interesting and desirable to create, but the challenges mentioned in the Section 5.4.1. In some cases the instrumentation was too limited, and in other cases the data analysis proved that our assumptions of a good symptom was invalid. Some of these symptoms was:

- Vulnerable weather directions

  - Discarded due to the fact that the current is created by cyclical tide. This is more important when assembling the plant.
- Wind speed/wave height and wind/wave direction
  - Discarded due to lack of accelerometers and low correlation with load shackle data.

6.2 Cases

When generating the cases we are using experience and knowledge gathered by studying the domain and discussions with domain experts. We have also used FhSim to verify some assumptions of what would happen if one or more moorings are removed or loosened. This means that we have to create artificial events and incidents as the cases. We should optimally have real incident reports and empirical data as the basis for the cases, but as these do not exist we have to use other methods. The following subsections describe the case structure and the five cases of possible and realistic situations, that uses the symptoms created above as the input.

6.2.1 Case Structure

A CBR system uses input cases to represent the current state of the system, and stored cases in the case base to represent the past. An input case consists of input attributes, and the stored cases consist of a case description and a case solution. In our system, the input attributes is the output from each of the symptoms. The case description describes the past cases, both by the previously present symptoms and a textual explanation of the details in the past event that the case represents. The case solution is the experience gathered from when the case was present in the past, used to solve the problem.

The point of the CBR system is to calculate the similarity between the input case and the cases in the case base. The similarity is a measure of how close the input case is to the different cases in the case base. The case with the highest score is the most similar one. The similarity can be calculated in many different and complex ways, and

similarity is often measured as a degree. Similarity is not the same as equality, which has a binary outcome. An example of similarity is that if a customer in a car shop wants a dark blue car, a light blue car is e.g. 50% similar and a red car is 0% similar. On the other hand, we can say that both the light blue and the red car is 0% equal. The advantage with using similarity is that it allows a lot more generality, and the cases does not necessary need to be equal before alarming the operator.

In the cases described next we mention the symptoms that we have chosen to be relevant for the case. All the symptoms are always included for the comparison of the cases, but only the ones that are mentioned have an impact on the similarity measure of the case comparison. As our system is simplified and prototypical, the others does not influence the similarity measure at all.

6.2.2 Risk of Manual Work

The case "Risk of Manual Work" is an artificial case describing a situation where the well-boat was interacting with the production plant and an operator was in danger because of to rough wind and wave conditions.

Relevant Symptoms

- Weather condition
- Manual work

Textual Description
The well-boat was interacting with the cage and requested human assistance. The wind and wave conditions was at a dangerous level, which made the operator to hurt him self when assisting the well-boat.

Similarity measure
The input case will be completely similar to this case if:

- Manual work indicates "yes"
- Weather condition indicates "bad" or "extreme"

Solution

- Wait for acceptable conditions before assisting.
- Wear safety line and clothing if assisting is unavoidable.

6.2.3 Slipping Anchor

The case ”Slipping Anchor” is an artificial case describing a situation where the anchor has slipped, which made the cage drift.

Relevant Symptoms

- current speedslope between the current and shackle data
- Shackle status

Textual Description
The calculated slope of the current data and the shackle data showed a deviation that indicates an error in the mooring system. The shackle status indicated that the mooring is not broken, which may indicate that the anchor has slipped. The last statement was confirmed by the operator when he did a visual inspection.

Similarity measure
The input case will be completely similar to this case if:

- current speedslope indicates ”high”
- Shackle status indicates ”good”

Solution

- Perform a visual inspection.
- Move anchor to correct position.

6.2.4 Loose Mooring Ropes

The case "Loose Mooring Ropes" is an artificial case describing a situation where the mooring ropes are loose, which caused an uneven distribution of the strain. The input is the state of the current speeddata from all three shackles connected to one rooster foot.

Relevant Symptoms

- current speeddata from from past similar conditions (3 shackles)
- Shackle status

Textual Description
The machine learning algorithm classified the strain as class 3 on all three shackles, while the measured strain was a lower class. The shackle status indicated that the mooring is not broken, which may indicate that the mooring ropes are loose. The last statement was confirmed by the operator when he did a visual inspection. This case implies that all three mooring ropes are loose.

Similarity measure
The input case will be completely similar to this case if:

- current speeddata indicates "over" on all shackles
- Shackle status indicates "good"

Solution

- Perform a visual inspection.
- Tighten the loose mooring ropes.

6.2.5 Difficult to Manoeuvre

The case ”Difficult to Manoeuvre” is an artificial case describing a situation where well-boat has difficulties with the manoeuvring when docking to the cage, which resulted in a damage on the cage.

Relevant Symptoms

- Weather conditions
- Current speed
- Manual work

Textual Description
The well-boat had problems with manoeuvring when docking to the cage because the environmental conditions was at a dangerous level. This made the boat crash into the cage structure, and damaged it severely.

Similarity measure
The input case will be completely similar to this case if:

- Weather condition indicates ”extreme”
- Current speed indicates ”high”
- Manual work indicates ”yes”

Solution

- Wait for acceptable conditions before docking.

The limits for acceptable environmental conditions are set by us.

6.2.6 Broken Mooring

The case ”Broken Mooring” is an artificial case generated with information from FhSim, which describe a situation where the mooring is broken. This caused an uneven distribution of the strain. The input is the state of the deviant data from all three shackles connected to one rooster foot.

Relevant Symptoms

- Shackle status
- Deviant data from from past similar conditions(3 shackles)

Textual Description
The machine learning algorithm classified the strain as a higher class than the measured strain on one of the shackles. Further, the machine learning algorithm classified the strain as a lower class than the measured strain on the other two shackles. The shackle status indicated that a mooring is broken, which was confirmed by the operator when he did a visual inspection. This case implies that one mooring rope is broken, while the to remaining are healthy.

Similarity measure
The input case will be completely similar to this case if:

- Shackle status indicates ”bad”
- Deviant data indicates ”over” on one shackle
- Deviant data indicates ”under” on two shackle

Solution

- Perform a visual inspection.
- Repair the broken mooring.

Chapter 7

Architecture

This chapter describes the conceptual architecture designed in the specialisation project, as well as the changes made to the actual implemented architecture that fits our data basis.

7.1 Conceptual Architecture

The conceptual architecture designed during the specialisation project period was based on the knowledge gathered in our research study and data analysis. The architecture is based on the principles of structural health monitoring, combined with machine learning and data analysis for data interpretation, and case-based reasoning as the inference engine. When working with the data available in this MSc thesis, we discovered that the assumptions made then was different to what we had to work with. We also discovered that we could not use FhSim the way we earlier thought.

Despite the fact that the basis of the architecture is somehow changed as the data basis is new, we can still make use of the main elements of the conceptual architecture. We proposed an architecture that consisted of four independent components, that also adaptable to a change of input data. The components were:

- **Data Acquisition**
- **Data Interpretation**

- **Case-Based Reasoning**
- **Graphical User Interface**

By designing the architecture very flexible to changes, it can practically adapt to many types of problems and domains.

> *"The general aspects of the proposed system architecture are to some degree independent of the mentioned assumptions. This means that even if the decision to be supported or the available data are changed, the architectural shell remains the same. This makes the system flexible and modular, and easier to handle if there are any changes."*
> *[Pedersen and Roppestad, 2015]*

We are using the same four components as presented above in the implemented architecture, but the tasks done in the different component is different compared to what we proposed previously.

The biggest change is that we previously included FhSim as a simulator that uses the environmental data in numerical models to simulate the environmental affections on the cage structure and mooring system. The output data was simulated cage monitoring data, which is similar to the data collected from the real world sensors. This was supposed to represent the healthy state of the monitored structures, but we encountered some major problems:

- The simulation was extremely slow, as it took between 5 to 16 hours to simulate 500 second, depending on the step length of the integration. The step length has an impact on the accuracy.
- The accuracy was too bad, as we could not recognise the real shackle measurements in the simulated data. The model was too advanced and required more sensors at the opposite side of the cage, or an accelerometer to be calculated properly.
  - As the real shackle data is monitored on a structure that wears down over time and is very dynamic, it is difficult to find statistical similarities in the simulated data based on a "perfect, undamaged condition" structure.
  - FhSim outputs the force in a given point on the cage, which is useful in many cases. But as we only know the strain between the rooster foot and the cage, we can not tell in

which direction the cage is moving.

We therefore decided to leave the simulation out in this component, and base the symptoms on data analysis, machine learning and mathematical modelling.

7.2 Implemented Architecture

The implemented architecture consists of the four components mentioned above. The process view is illustrated in Figure 7.1 with the arrows indicating the direction of the data flow.

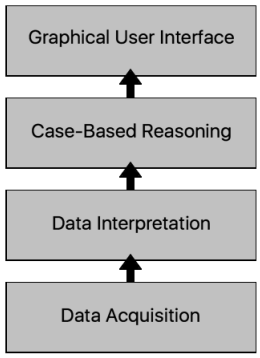

Figure 7.1: Process view of the architecture with arrows indicating the data flow direction.

The next sections describes the data flow and what happens in the four components in detail.

7.2.1 Data Acquisition

The lowest level of the architecture is data acquisition, where data is acquired from the sources. The sources are typically databases containing sensor data, acquired either automatically in real time or manually at later stages. This part of the architecture consists of a data reader and a data preparator, as shown in Figure 7.2.

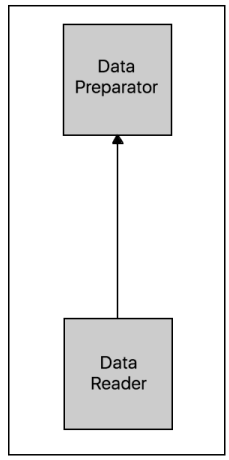

Figure 7.2: Overview of the elements in the data acquisition level, where the arrows indicating the data flow direction.

As the sensors are in a different project area than ours, we assume that the sensor data are pushed to a database. The data reader is responsible for collecting this data. There are mainly two different kinds of sensor data, which are the cage monitoring data (shackles) and the environmental data (wind, wave, current). The reader then pushes all the data.

When receiving the data from the reader, the preparator generates data sets ready for data analysis and pre-processing, and pushes these to the next level of the architecture: Data Interpretation.

7.2.2 Data Interpretation

In the data interpretation level, software agents make use of the data acquired from the acquisition level. An agent is the implemented version of the symptoms mentioned in the previous chapters. Additionally, this level contains a pre-processor and an organiser, as shown in Figure 7.3. It is in this component the data modelling is done.

In the pre-processing step operations on the input data sets are done, which results in data that more than one agent can make use of. This prevents redundant calculations in multiple agents and makes the data easier to handle.

The tasks of the agents are to recognise patterns, monitor thresholds and trends and to signalise if a symptom occurs. An agent is assigned to identify a single symptom only. The input to an agent is typically a predefined and preprocessed set of values, of a specific time period.

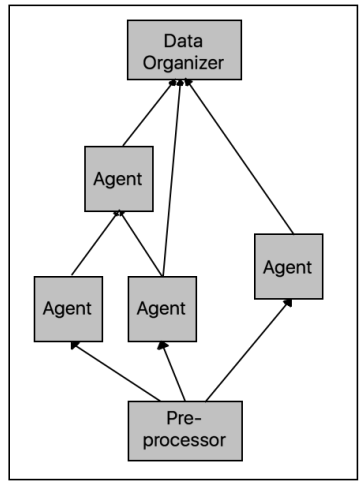

Figure 7.3: Overview of the elements in the data interpretation level, where the arrows indicating the data flow direction.

The output from an agent is an indication of the state that it is monitoring, often as a textual description. The agents are also responsible for handling the difficult aspect of time. By assigning this task to the agents, we ease the complexity of comparing dynamic cases in the CBR system as this is a struggle in CBR. An example is comparing the slope between two data sets for a given time period.

All the agents report their state to a data organiser, whose role is to have an overview of the current situation that are interpreted by the agents. The state of the agents could be binary or have a degree of how present a symptom are. The data organiser then pushes the gathered information to the next level of the architecture: Case-Based Reasoning.

7.2.3 CBR

The case-based reasoning engine receives the information generated in the data interpretation level, and uses this to recognise broader patterns focusing on the complete situation. The CBR engine compares the current situation with past situations to find out if the statuses given by the agents are similar to a past problematic situation, for then to notify and advice the operator on what to do. The past situations represent events and incidents the outcome was unwanted, which we now want to avoid. Figure 7.4 shows the crucial parts of this architecture level, which are the case, retrieval of cases, pushing similar cases to a graphical user interface, and retention of cases.

A case consists of a description of the situation and the solution of the problem. When used as the input case, it does not have the solution part. The case description therefore only contains information about the symptoms that are present or not. The description is then used by the CBR system to compare the input case with the cases in the case base, where the latter contains a solution. This phase is the retrieval step. The retrieved case(s) are determined by the degree of similarity between the input case and the retrieved one(s). The retrieved case(s) are then pushed to the graphical user interface for visualisation and decision support.

The case can be retained in the case base if the current situation does not fit any previous known ones. Retention of a case can also be done if the operator or domain expert finds it useful to store the experience about the current situation for later use. The new case should be properly reviewed by experts to assure quality and make sure that it fits to the CBR system.

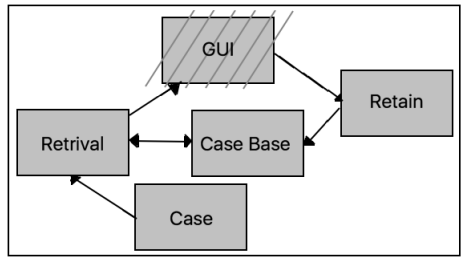

Figure 7.4: Overview of the elements in the CBR level, where the arrows indicating the data flow direction.

7.2.4 Graphical User Interface

The graphical user interface is the last and highest level of the system architecture. It is here that the results from the CBR level are visualised for the operator, so that he or she can act upon the current situation.

The GUI is tightly coupled with the CBR level of the architecture, and works as the interaction module for the operator. The output from the retrieval step are visualised in the GUI as the most similar case(s), and the operator should be able to investigate these. The operator should also be notified if the similarity between the current case and a previous case are higher than a threshold, which indicates that a problem could be close to occur. By being aware of that the current situation are close to any previous problem situations, the operator has the choice to act upon it or not. The cases provided to the operator contain experience and solutions, which can support his or hers decision.

Retaining the case should also be a possibility from the GUI. When retaining a new case, it should be sent to a system for revision and quality control before storing it in the case base.

Chapter 8

Implementation

This chapter describes the work done with the implementation of the decision support system. Through the project period we have met some challenges that has limited the need for a complete system. These challenges, our focus points of the implementation and a system walkthrough will be the subjects of the next sections.

8.1 Overview

At the beginning of this project, our goal was to implement a prototypical decision support system. As we met the challenges mentioned in Section 5.4.1 with the data basis, the need for an implemented prototype became less important. We have instead focused more on research within the domain, creating reasonable symptoms and cases, and proposing a proper architecture. Especially the late arrival of the data and the limited information about real incidents was big drawbacks, which made big changes for the results of the project. Because of these circumstances, we have abandoned some parts of the implementation, like making the data acquisition real-time, retaining new cases and making a GUI customised for the operators.

After receiving the data, we conducted a data analysis before creating the symptoms and cases. These elements are the most important for this kind of systems. The fact that our symptoms and cases are artificial, further substantiate that a complete implementation is not

the most important part of this project. The next sections describe the implementation of the four parts of the architecture, where the most important parts are the data interpretation and the CBR components. The first is where the symptoms are implemented as agents, and the latter is where the cases are compared. The last part of this chapter is a system walkthrough that provide useful information for understanding how the components are interacting, and the process from first input to last output of the system.

8.2 Data Acquisition

In the data acquisition component we deal with the file reading. The files that are used in the system is the resulting data sets from the data analysis as described in Chapter 5 Data Analysis. Listing 8.1 shows the code for reading the shackle data.

```
1  ArrayList<double[]> readShackles(String csvFile, int startTime) throws
       IOException {
2      double[] shackle1, shackle2, shackle3 = new double[120];
3
4      String line = "";
5      String cvsSplitBy = ",";
6      int lineCounter = 0;
7
8      BufferedReader br = new BufferedReader(new FileReader(csvFile));
9      while ((line = br.readLine()) != null) {
10
11       if(lineCounter >= startTime*60 && lineCounter < (startTime*60+
       shackle1.length)){
12         String[] data = line.split(cvsSplitBy);
13
14         shackle1[counter-startTime*60]= Double.parseDouble(data[0]);
15         shackle2[counter-startTime*60]= Double.parseDouble(data[1]);
16         shackle3[counter-startTime*60]= Double.parseDouble(data[2]);
17       }
18       lineCounter++;
19     }
20     ArrayList<double[]> shackles = new ArrayList();
21     shackles.add(shackle1);
22     shackles.add(shackle2);
23     shackles.add(shackle3);
24
25     return shackles;
26   }
```

Listing 8.1: Code for reading the shackle data.

The method readShackle stores the shackle measurements from the last two hours in three separate arrays, one per shackle. Every time the method is called we jump one hour and store the data from the new two hours. The method returns an arraylist containing the three

arrays to the Data Interpretation component. The method is similar for the environmental data.

8.3 Data Interpretation

The data interpretation component contains the implemented mathematical functions used within the symptoms, and the implemented symptoms themselves. As Section 6.1 describes the thoughts and logic behind all of the symptoms in depth, we only show the implementation of one of the symptoms in this chapter. The structure of the code implementation are similar in many of the symptoms, and the only difference is the mathematical function. The next section describe the mathematical function, before describing the implementation of a symptom.

8.3.1 Mathematical Functions

We have created a set of mathematical function that are used within the symptoms. These functions calculate the maximum, minimum, mean and slope for an arbitrary input. The different symptoms only use the functions they need. As the calculation of the maximum, minimum and mean is basic, we only show the implementation of the function that calculates the slope.

The calculation of the slope is shown in Listing 8.2. It is based on the following procedure:

1. Calculate the mean of the x values.
2. Calculate the mean of the y values.
3. Calculate the standard deviation sX of the x values.
4. Calculate the standard deviation sY of the y values.
5. Calculate the correlation r between X and Y.
6. Calculate the slope.

The comments in the listing are corresponding to the steps in the procedure above.

The input is e.g. data from one of the shackles, which is rep-

resented by yData. The xData is the corresponding x-value[1] as in xy-pairs, which is a continuous number series with length equal to the specified variable timeLength. The timeLength variable defines the time span that the slope are calculated within.

```
1  double calcSlope(double yData[], int timeLength) {
2    double avgX, r, sX, sY, slope, rX, rY = 0;
3    int counter = 1;
4
5    //step 1
6    double avgY = calcAvg(yData, timeLength);
7
8    //step 2
9    for (int i = 1; i <= timeLength; i++) {
10     avgX += i;
11   }
12   avgX /= timeLength;
13
14   //Calculations for sX, sY and r
15   for (int i = yData.length - 1; i >= yData.length - timeLength; i--) {
16     sX += Math.pow((counter - avgX), 2);
17     sY += Math.pow((yData[i] - avgY), 2);
18     rX = counter - avgX;
19     rY = yData[yData.length - counter] - avgY;
20     r += rX * rY;
21     counter++;
22   }
23
24   //step 3 and 4
25   sX = Math.sqrt(sX / (timeLength - 1));
26   sY = Math.sqrt(sY / (timeLength - 1));
27
28   //step 5
29   r = (1.0 / (timeLength - 1)) * (r / (sX * sY));
30
31   //step 6
32   slope = r * (sY / sX);
33   }
34
35   return slope;
36 }
```

Listing 8.2: Code for calculating the slope.

8.3.2 Symptoms

The symptoms we have created are described earlier in the report, and this subsection will give an insight of the implemented version of the symptom "Deviant Slope Between Current and Shackle Data". The method is similar in the other symptoms, just with other calculations and thresholds for the output. The general method for implementing a symptom is:

[1] x-min = x1 = 1. x-max = timeLength.

1. Define which input(s) that is relevant for the symptom.
2. Identify the necessary calculations, and initialise variables.
3. Perform the calculations.
4. Decide the output state by comparing the calculated value(s) with predefined thresholds.
5. Return the state.

Listing 8.3 shows the implementation of the symptom mentioned above.

```
1  public String slopeDeviation(double[] data, double[] shackle1, double[]
       shackle2, double[] shackle3, int timeLength){
2          //step 1 is the input to the method
3          //step 2
4      double dataSlope, shackleSlope1, shackleSlope2, shackleSlope3, diff1,
        diff2, diff3 = 0;
5      double threshold = 30;
6
7      String slopeDeviation = "low";
8
9      //step 3
10     dataSlope = calcSlope(data, timeLength);
11     shackleSlope1 = calcSlope(shackle1,timeLength);
12     shackleSlope2 = calcSlope(shackle2,timeLength);
13     shackleSlope3 = calcSlope(shackle3, timeLength);
14
15     diff1 = Math.abs(Math.abs(dataSlope-shackleSlope1)/dataSlope*100);
16     diff2 = Math.abs(Math.abs(dataSlope-shackleSlope2)/dataSlope*100);
17     diff3 = Math.abs(Math.abs(dataSlope-shackleSlope3)/dataSlope*100);
18
19     //step 4
20     if(diff1 > threshold){
21       slopeDeviation = "high";
22     }
23     if(diff2 > threshold){
24       slopeDeviation = "high";
25     }
26     if(diff3 > threshold){
27       slopeDeviation = "high";
28     }
29     //step 5
30     return slopeDeviation;
31   }
```

Listing 8.3: Code for the symptom ”Deviant Slope Between Current and Shackle Data”.

The comments in the listing are corresponding to the steps in the procedure above. The most important aspects from the listing is the threshold defined to 30% in step 2 and the calculation of the slopes and the difference between the data slope and all the shackle slopes in step 3. Step 4 compares the calculated difference with the threshold.

8.4 CBR

The CBR component is responsible for storing the cases in the case base, and comparing new cases with the stored ones. We originally wanted to implement myCBR as our CBR tool, but myCBR had too limited options regarding the similarity measure. This led to that we implemented our own CBR engine.

The first step of implementing the CBR component is to develop a case base. As we have few cases, we are not limited by the processing time when comparing input cases with the stored cases. We therefore created a simple case base structure. The case base is initialised as a two-dimensional string array, where the first dimension keeps track of the case ID and the second dimension describes the attributes, the case description and the case solution. A case is defined as Case[ID][attribute]. Listing 8.4 shows the initialising of the case base, and one of the cases. The comments describe the names of the attributes, and the rest of the fields as well. This representation makes it easy to extract the attributes for comparison with other cases and the description and solution when needed.

Index 0 to 7 in the second dimension are the symptom attributes, and the information in these fields are different for all the cases. If a symptom attribute is defined as unspecified (”u”), it means that the similarity of these attributes will not contribute to the global similarity. This will be described next.

```
1  public class CaseBase {
2      String[][] Case = new String [5][10];
3    void fillCaseBase(){
4      Case[1][0] = "u";    //currentSpeed
5      Case[1][1] = "u";    //deviantData85130
6      Case[1][2] = "u";    //deviantData99871
7      Case[1][3] = "u";    //deviantData99875
8      Case[1][4] = "u";    //manualWork
9      Case[1][5] = "good";//shackleStatus
10     Case[1][6] = "high";//slopeDeviation
11     Case[1][7] = "u";    //weatherCondition
12     //caseDescription
13     Case[1][8] = "The calculated slope of the current data ...";
14     //caseSolution
15     Case[1][9] = "Perform a visual inspection. Move anchor...";
16     //caseID
17     Case[1][10] = "1";
18     //caseTitle
19     Case[1][11] = "Slipping anchor";
20   }
21 }
```

Listing 8.4: Implementation of the case base.

To calculate the similarity between cases we use local similarity measures for the attribute to attribute comparison, and a global similarity measure. The latter makes it possible to choose which attributes that are relevant for the different cases, and to calculate the total similarity based on the local similarity.

Listing 8.5 shows the implementation of the global similarity. Inside this method, we are calculating the local similarity as shown at line 10 to line 12 for the "Current Speed" symptom attribute. The if-test is checking whether the attribute is unspecified for the case in the case base. If so, the local similarity does not contribute to the global similarity. The if-test is done for each of the symptom attributes. At last we calculate the global similarity which is a weighted sum of the relevant local similarities. The local similarities are calculated by comparing the symptom attribute in the input case with the cases in the case base. The result is often a degree of similarity, as mentioned in Section 6.2.1. An overview of the local similarities for all the possible states of each symptom attribute is given in the Tables 8.1-8.5. The query attribute is to the left of the tables, and the compared case attribute is on top. The Manual Work attribute is not included, as it is a binary value.

```
1  public ArrayList<Double> globalSim(String currentSpe, String DeviantData1,
       String DeviantData2, String DeviantData3, String manualWork, String
       shackleStatus, String slopeDev, String WeatherCon) {
2      double similarity = 0;
3
4      ArrayList<Double> similarities, result = new ArrayList();
5
6      double divSym = 0.0;
7
8      for(int i = 0; i< CB.CaseBase.length; i++){
9
10       if(!CB.CaseBase[i][0].equals("u")){
11         similarities.add(currentSpeSim(currentSpe,CB.CaseBase[i][0]));
12       }
13       .
14       .
15       .
16       if(!CB.CaseBase[i][7].equals("u")){
17         similarities.add(weatherConSim(WeatherCon,CB.CaseBase[i][7]));
18       }
19       divSym = 1.0/similarities.size();
20
21       for(Double value : similarities){
22         similarity += value*divSym;
23       }
24
25       result.add(similarity);
26       similarity = 0;
27       similarities.clear();
28     }
29     return result;
30   }
```

Listing 8.5: Implementation of the global similarity function.

Table 8.1: Local similarity for current speed.

	low	medium	high
low	1.0	0.0	0.0
medium	1.0	1.0	0.5
high	1.0	1.0	1.0

Table 8.2: Local similarity for deviant data.

	over	under	low
over	1.0	0.0	0.0
under	0.0	1.0	0.0
low	0.0	0.0	1.0

Table 8.3: Local similarity for shackle status.

	good	bad
good	1.0	0.0
bad	0.0	1.0

Table 8.4: Local similarity for slope deviation.

	low	high
low	1.0	0.0
high	0.0	1.0

Table 8.5: Local similarity for weather condition.

	extreme	bad	good
extreme	1.0	0.0	0.0
bad	0.5	1.0	1.0
good	0.0	0.0	1.0

8.5 Graphical User Interface

The graphical user interface has the ability of starting the decision support system, enable the manual work symptom, display cases with similarity within the threshold value defined by a slider and displaying information about the cases in the list. The GUI is primarily developed for testing purposes, and it is not design optimally for an operator. Even though it is simple, it is informative and contains the most important elements of a GUI for a decision support system. The GUI is shown in Figure 8.1.

The implementation of the GUI is done with the e(fx)clipse plug-in, and the information displayed are extracted from the CBR component of the system.

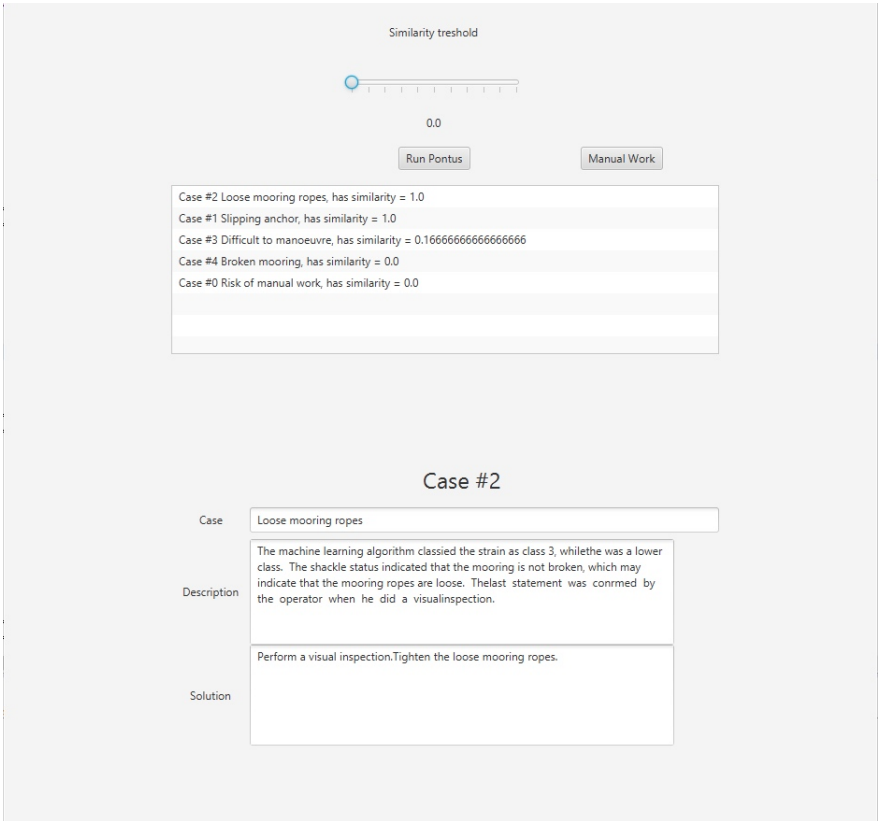

Figure 8.1: Screenshot of the GUI.

8.6 System Walkthrough

This system walkthrough describes the process of the system. This will hopefully give a deeper understanding of the components of the decision support system we have created. This chapter shows the steps in the system from first input to final output. The implemented prototype is inspired by the architecture, but as mentioned we do not analyse any raw data in the Data Interpretation component. The preprocessing was a part of the data analysis, and the resulting data sets are used as input.

8.6.1 Data Acquisition

The first step of the system is to acquire the data in the Data Acquisition component. This component reads the weather.csv and shackle.csv files, which is shown in top of Figure 8.2. As mentioned, the weather data contains measurements of the current, wind and wave, and the shackle data contains strain measurements from three shackles.

After the files are read into the system, the different measurements are stored in separate containers. In our prototype we have configured the system to store data from the last two hours in these containers. At every iteration, the containers are updated with data from a new hour, and the oldest is thrown. This is shown in the bottom of Figure 8.2. These containers constitutes the basis for the system, and are now ready for usage in the Data Interpretation component.

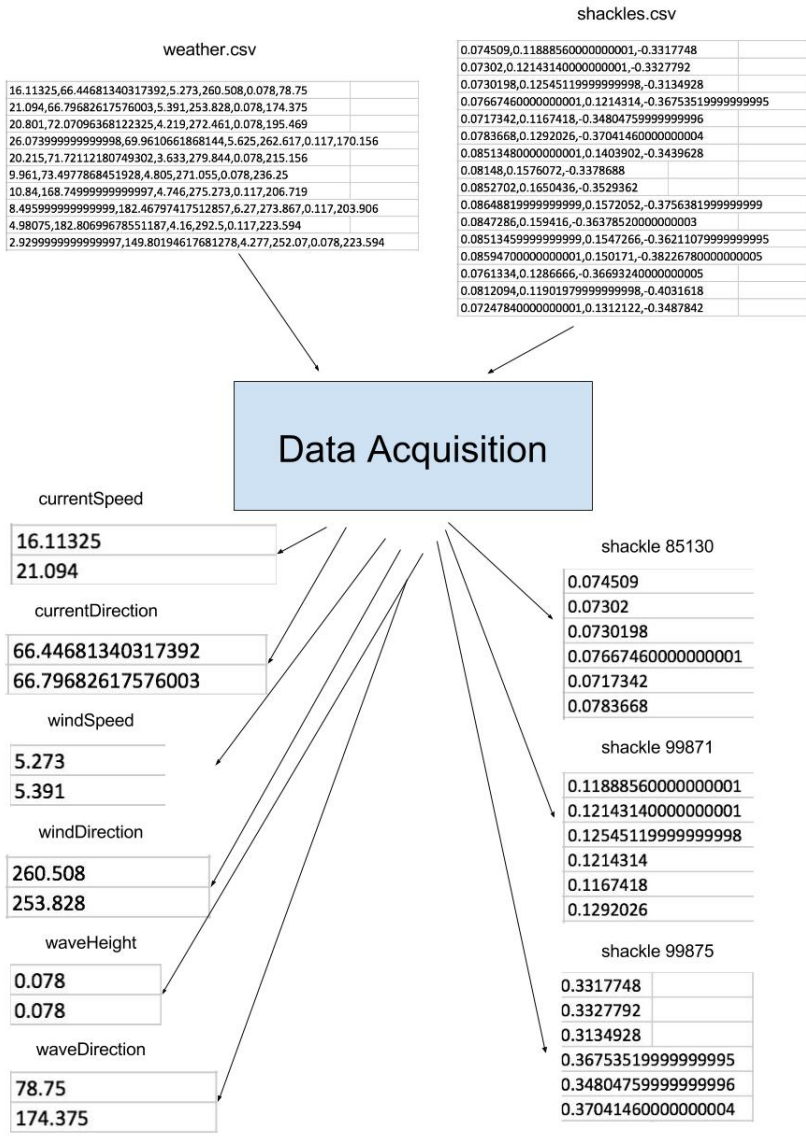

Figure 8.2: Overview of the Data Acquisition component.

8.6.2 Data Interpretation

The second step of the system is to use the data in the containers to describe the state of the defined symptoms. Figure 8.3 shows the containers and their relations to the different symptoms in the blue boxes. Some of the containers are related to a single symptom only, and others are related to more. At the bottom of the figure, the arrows points to the resulting state of each symptom. The states are described by a textual description, and is calculated from the values in the containers. This calculation is shown in Listing 8.2 and 8.3 in Section 8.3. The states are describing the current situation, and are the product of this component. The textual description from each symptom is then used as the input to the CBR component. The states are updated every iteration, as the data in the containers are updated.

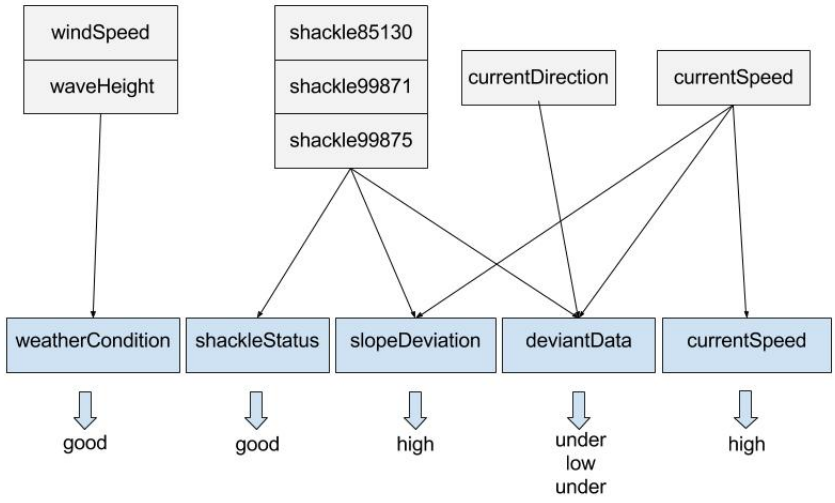

Figure 8.3: Overview of the Data Interpretation component.

8.6.3 Case-Based Reasoning

The third step of the system is to calculate how similar the query case is to any case in the case base. The query case represents the current situation. The textual descriptions from the Data Interpretation component is then used as the query for calculating the similarity between the cases. This is shown to the left in Figure 8.4, which shows the calculation of the similarity between the new case and Case #1 in the case base. As Case #1 is undefined ("u") for all symptoms except shackleStatus and slopeDeviation, the local similarity is only calculated for these two. As both the query and the case has "good" as shackleStatus and "high" on slopeDeviation, the local similarities for these symptoms are 1.0. The complete similarity tables are shown in Section 8.4. The global similarity is calculated in the bottom of the figure. It is calculated by giving each of the symptoms that are defined equal weights, and summing the weighted local similarities. Figure 8.4 shows that Case #1 has a global similarity of 1.0, and Figure 8.5 shows the same calculations for Case #4 that has a global similarity of 0.5. It is the global similarity that decides how similar two cases are. The result from the CBR component is a list of the similarity between all the cases in the case base and the query case.

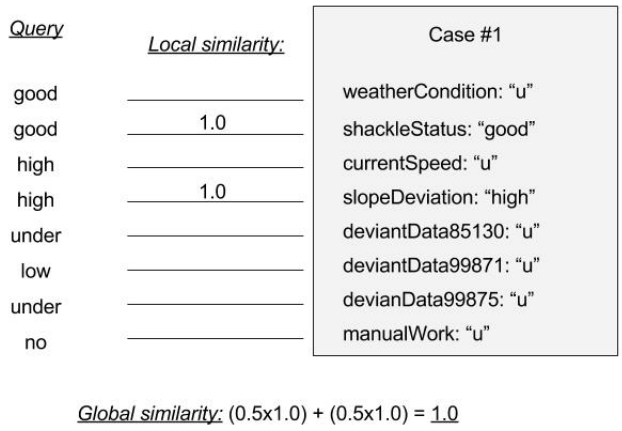

Figure 8.4: Query case and similarity to case #1.

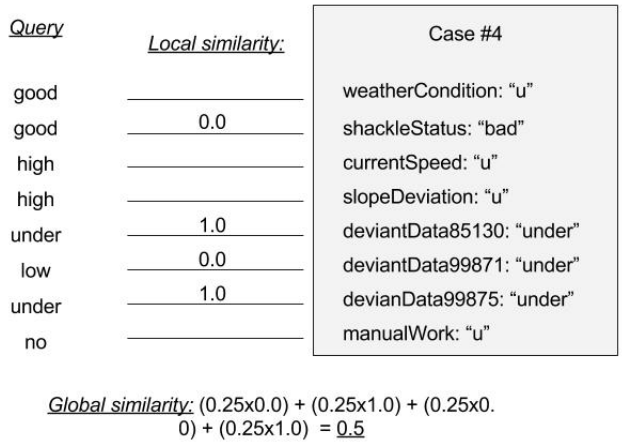

Figure 8.5: Query case and similarity to case #4.

8.6.4 Graphical User Interface

The final step of the system is to display the results from the CBR component in a graphical user interface. The GUI has some simple features, like choosing the minimum threshold of similarity for a case to be displayed, choosing whether manual work is in progress or not, and a button for starting the process.

The list provided from the CBR component contains the case ID and the associated similarity. It is then sorted by similarity, and the cases with similarity above the given threshold is displayed in the display list. The fields below the display list shows information about the most similar case to the user. The information displayed are the case ID, the case title, case description and case solution. This information are provided for all the cases in Chapter 6 Modelling. As the program runs, the fields are updated continuously as the initial information containers iterate.

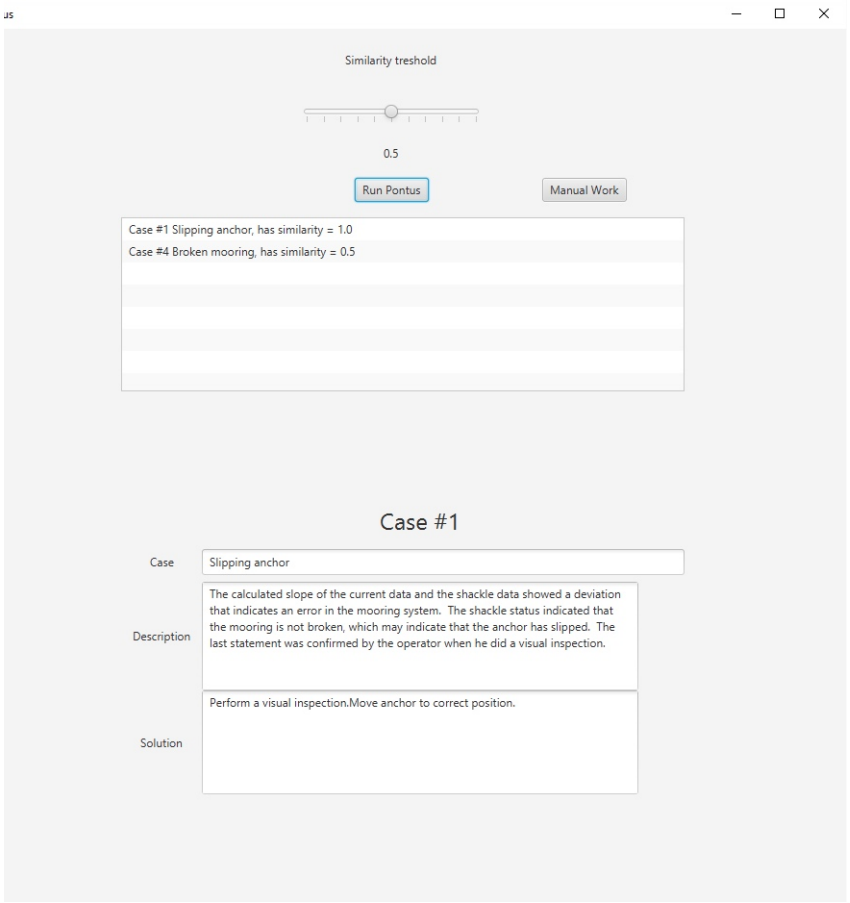

Figure 8.6: Screenshot of the GUI.

Chapter 9

Evaluation

This chapter contains the results of this project, as well as the evaluation of one of the symptoms created. As mentioned in Section 4.4 evaluating the implemented system is difficult due to lack of empirical data. Our main results are the symptoms and the cases, which implies that it is more meaningful to evaluate and discuss these elements. The methods of the evaluation will be performing machine learning tests, and an analytic discussion. We also find it useful to discuss different aspects we have worked with during the project period, like the sensors, challenges and the fact that the symptoms and cases are artificial.

9.1 Machine Learning Tests

We have used machine learning algorithms to predict the strain value based on the current speed and the current direction. We have used the current data as the visualisations and correlation analysis proved that the current speed and direction was most related to the strain. The point of conducting the following tests is to evaluate which machine learning algorithm that gives the best accuracy for the mentioned classification. The best algorithm is then used to build the model that we are using in the symptom "Deviant Data from Past Similar Conditions".

The training sets consists of 521 instances and the test sets consists

of 151 instances. We have chosen to evaluate six different learning algorithms on all three shackles at the one available rooster foot. The first three are regression algorithms, and depends on a numerical, continuous value to be predicted. Here we are using the shackle data with no modifications. The last three are classification algorithms, which depends on a nominal, discrete attribute as the class to be classified. This requires us to create separate classes of the numerical shackle data, and we have made three classes which represent low, medium and high values.

When evaluating the different machine learning algorithms we are testing, the following aspects will be the most important:

- The accuracy should be higher in the critical areas of the scale, which are the area where the strain values are highest.
- The learning algorithm should rather over estimate than under estimate, as higher values are the most dangerous for the cage.
- The general accuracy and running and building time should be within reasonable limits.

The following subsections display the results from the tests done in the Weka data mining program. We provide it with with one data set as the training set, and we use a separate test set for evaluation. Figure 9.1 shows the training set selection with visualisations of the instance distribution and statistical values of shackle 85130. Figure 9.2 shows the distribution of the classes when converting the shackle data to nominal. The information provided for shackle 85130 in the figures are representative for the three shackles on the same rooster foot that we have used in the data analysis. The test information, as well as the explanation is only showed for shackle 99875, but the results for all three shackles are presented and discussed later on.

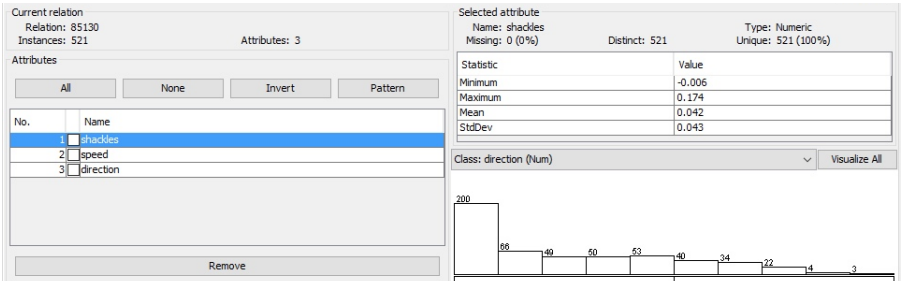

Figure 9.1: Visualisation of the training set with numerical values in Weka.

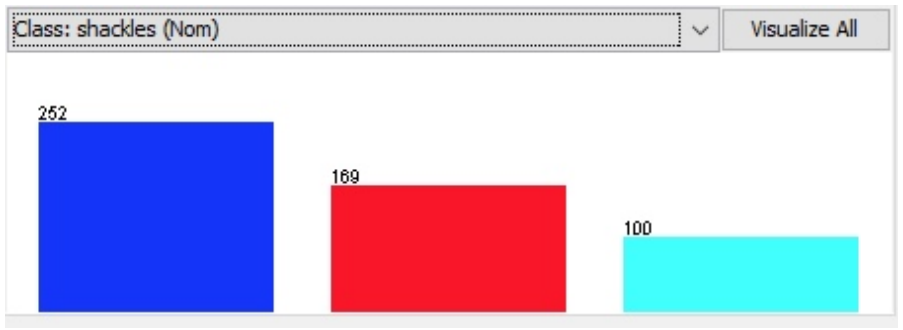

Figure 9.2: Visualisation of the distribution in the training set with nominal classes. The blue bar shows class 1 instances, the red class 2 and the cyan class 3.

9.1.1 Linear Regression

Linear regression is an approach for modelling the relationship between a dependent variable y and one or more explanatory variables X. Our variable y is the strain value, and the X is the current speed and direction [Wikipedia, 2016g]. When using linear regression for prediction the method is used to fit a predictive model to an observed data set of y and X values. The Listing 9.1 shows the input configurations used for the linear regression in Weka, and Listing 9.2 shows the results. We have not changed the parameters in this test, but some available choices are the attribute selection method, to eliminate co-linear attributes or not and the ridge.

```
1 === Run information ===
2 Scheme:weka.classifiers.functions.LinearRegression -S 0 -R
3 Relation:     99875
4 Instances:    521
5 Attributes:   3
6               shackles
7               speed
8               direction
9 Test mode:user supplied test set:     151instances
```

Listing 9.1: Linear regression run information.

The results show that the building time of the model is extremely fast, as the number of instances are relatively low. The model is also not very complex as we only have two dimensions. The evaluation on the test set shows highly correlated data, but the relative absolute error[1] is high.

```
1 Time taken to build model: 0 seconds
2 === Evaluation on test set === === Summary ===
3
4 Correlation coefficient                  0.8657
5 Mean absolute error                      0.0639
6 Root mean squared error                  0.0817
7 Relative absolute error                 50.0346 %
8 Root relative squared error             56.5576 %
9 Total Number of Instances               151
```

Listing 9.2: Linear regression evaluation summary.

[1]The RAE is the normalised mean absolute error in percent. The MAE is the measure of how close the predictions are to the actual values..

9.1.2 Artificial Neural Network

Artificial neural networks are models based on biological neural networks found in the brain [Wikipedia, 2016b]. It can be used with a large number of inputs, and the network consists of interconnected nodes in different layers. The layers are the input, the hidden layer(s) and the output. Listing 9.3 shows the configuration of our neural network model called Multilayer perceptron. A Multilayer perceptron consists of multiple layers of nodes in a directed graph. We have chosen to run the ANN with the learning rate parameter -L=0.1, training time -N=10000 and the rest as standard configured.

```
1 === Run information ===
2
3 Scheme:weka.classifiers.functions.MultilayerPerceptron -L
      0.1 -M 0.2 -N 10000 -V 0 -S 0 -E 20 -H a
4 Relation:     99875
5 Instances:    521
6 Attributes:   3
7               shackles
8               speed
9               direction
10 Test mode:user supplied test set:     151instances
```

Listing 9.3: Multilayer perceptron run information.

The results in Listing 9.4 show that the model is fast to build, because of the same reasons as mentioned for the linear regression. The correlation coefficient shows highly correlated data, and the accuracy is better than for linear regression.

```
1 Time taken to build model: 1.83 seconds
2
3 === Evaluation on test set ===
4 === Summary ===
5
6 Correlation coefficient                  0.8843
7 Mean absolute error                      0.0541
8 Root mean squared error                  0.0796
9 Relative absolute error                 42.3528 %
10 Root relative squared error            55.1073 %
11 Total Number of Instances              151
```

Listing 9.4: Multilayer perceptron evaluation summary.

9.1.3 Instance-Based Learner

Instance-based learning compares new problem instances with previously training instances to predict its value [Wikipedia, 2016f]. It does not perform explicit generalisation after training, which means that IBL is a kind of lazy learning. A common problem for learner that create hypotheses directly from the training instances is that all training data must be stored in memory and overfitting. In our case we have the advantage of few training instances because of our dynamic structure. We have chosen to use K* as our IBL, which uses an entropy-based distance function for similarity assessing. The run information of the K* is shown in Listing 9.5, with the standard parameter configuration.

```
1 === Run information ===
2
3 Scheme:weka.classifiers.lazy.KStar -B 20 -M a
4 Relation:     99875
5 Instances:    521
6 Attributes:   3
7               shackles
8               speed
9               direction
10 Test mode:user supplied test set:     151instances
```

Listing 9.5: Lazy K* run information.

The results in Listing 9.6 show that the model is extremely fast to build, and the results are encouraging.

```
1 Time taken to build model: 0 seconds
2
3 === Evaluation on test set ===
4 === Summary ===
5
6 Correlation coefficient                  0.9104
7 Mean absolute error                      0.045
8 Root mean squared error                  0.069
9 Relative absolute error                 35.2546 %
10 Root relative squared error             47.8062 %
11 Total Number of Instances               151
```

Listing 9.6: Lazy K* evaluation summary.

9.1.4 Support Vector Machine

As the first classification based learning algorithm with nominal class values, we have chosen a support vector machine. A model build by an SVM represents examples as point in space, mapped so that examples of each category are divided by a gap that is as wide as possible [Wikipedia, 2016k]. New examples are then mapped into the same space and classified to a category based on which side of the gap they fit in. Listing 9.7 shows the configuration of the SVM, where we are using a first degree polynomial kernel.

```
=== Run information ===

Scheme:weka.classifiers.functions.LibSVM -S 0 -K 1 -D 1 -G
    0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -seed
    1
Relation:     99875
Instances:    521
Attributes:   3
              shackles
              speed
              direction
Test mode:user supplied test set: size unknown (reading
    incrementally)
```

Listing 9.7: LibSVM run information.

The results in Listing 9.8 shows that the model is fast to build, and the classification is 83.4% accurate. The confusion matrix at the bottom shows the how the SVM has classified each of the test instances.

```
Time taken to build model: 1.8 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances         126       83.4437 %
Incorrectly Classified Instances        25       16.5563 %
Kappa statistic                          0.7075
Mean absolute error                      0.1104
Root mean squared error                  0.3322
Relative absolute error                 29.861  %
Root relative squared error             77.9517 %
Total Number of Instances               151

=== Confusion Matrix ===
  a  b  c   <-- classified as
 71  7  0 |  a = 1
  3 51 13 |  b = 2
  0  2  4 |  c = 3
```

Listing 9.8: LibSVM evaluation summary.

9.1.5 Bayesian Classification

A Naïve Bayes classifier is a probabilistic classifier based on applying Bayes Theorem with strong independence assumptions between the feature [Wikipedia, 2016h]. Naïve Bayes is widely used and studied, and it is highly scalable. Even though the algorithm has a naive design and oversimplified assumptions, it works well on many difficult real-world problems. When dealing with Naïve Bayes one should always remember that it believes that all features contribute to the classification independently, regardless any correlation between them. Listing 9.9 shows the configuration of the classifier with the standard parameters.

```
=== Run information ===

Scheme:weka.classifiers.bayes.NaiveBayes -K
Relation:     99875
Instances:    521
Attributes:   3
              shackles
              speed
              direction
Test mode:user supplied test set: size unknown (reading
    incrementally)
```

Listing 9.9: Naïve Bayes run information.

Listing 9.10 shows the results of the classification. The time to build the model is extremely good, and the results are good as well.

```
Time taken to build model: 0 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances         128       84.7682 %
Incorrectly Classified Instances        23       15.2318 %
Kappa statistic                          0.7334
Mean absolute error                      0.1687
Root mean squared error                  0.2876
Relative absolute error                 45.6417 %
Root relative squared error             67.485  %
Total Number of Instances              151

=== Confusion Matrix ===
  a  b  c   <-- classified as
 72  6  0 |  a = 1
  2 51 14 |  b = 2
  0  1  5 |  c = 3
```

Listing 9.10: Naïve Bayes evaluation summary.

9.1.6 Learning Tree

Decision trees are a predictive model to assess instances to classes. Each leaf represent a class and the branches are conjunctions of features which eventually lead to the classification [Wikipedia, 2016d]. Decision trees can be used both for continuous and discrete values. We have chosen J48 as our algorithm which is based on the popular C4.5 algorithm. Listing 9.11 shows the run information where we have changed the parameter -C, confidence factor used for pruning to 0.01.

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.01 -M 2
Relation:     99875
Instances:    521
Attributes:   3
              shackles
              speed
              direction
Test mode:user supplied test set: size unknown (reading
    incrementally)
```

Listing 9.11: J48 run information.

The results are shown in Listing 9.12, where we can see that the building time is very low and the accuracy is not as good as the other classification algorithms.

```
Time taken to build model: 0.01 seconds

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances         112       74.1722 %
Incorrectly Classified Instances        39       25.8278 %
Kappa statistic                          0.5622
Mean absolute error                      0.2135
Root mean squared error                  0.3225
Relative absolute error                 57.7704 %
Root relative squared error             75.6641 %
Total Number of Instances              151

=== Confusion Matrix ===
  a  b  c   <-- classified as
 72  6  0 |  a = 1
  8 36 23 |  b = 2
  0  2  4 |  c = 3
```

Listing 9.12: J48 evaluation summary.

9.2 Results

This section describes the results from the machine learning test presented above, and the remaining results from the project itself.

9.2.1 Machine Learning Results and Discussion

The tests above was done for shackle 85130 and 99871 as well, and we also tried to remove the direction feature to see if there was any improvement. The results from these tests are shown in Table 9.2 and 9.3. The information provided are the mean absolute error (MAE) and the relative absolute error (RAE) for the regression tests, and correctly classified instances (CCI) and correctly classified instances of class 3 for the classification tests. As the run and building times was satisfyingly low on all algorithms, we do not include them in the tables. To better understand the shackle values, we have provided the maximum, mean, the threshold value for class 3 and the class 3 percentage split. These values are presented in Table 9.1.

As we can see from Table 9.2, Multilayer perceptron and Lazy K* has the best overall performance. The latter has remarkable better performance on shackle 99875. The RAE is remarkably high on all test results. This happens because all shackle data sets have an overweight of entries with a value in the range -0.05 to 0.05 which drastically affects the mean. We should therefore analyse the visualisations of the regression, and account for the maximum and mean values shown in Table 9.1. Figure 9.3 shows the visualisation of the classifiers errors of the K* algorithm, where small crosses means a small error. As we can see, K* overestimates the middle values and underestimates the highest values. This is a bad sign, as we rather want to overestimate the highest values because these are the most dangerous entries for the production plant. In the figure, the orange crosses are underestimated, and the blue are overestimated.

When analysing the classification algorithms, it is important to see which of the instances that are correctly classified. As we have created the classes manually by using the 35-40% highest values[2], we

[2]Shackle 85130 and 99871 has 7 and 6 instances that has very high values, and we used a threshold of 38% and 40 % adjusted for them.

see it more important to classify this class more accurate. It is therefore important to look at the confusion matrix to get an indication of how much the algorithm overestimates or underestimates.

The results in Table 9.3 shows that Naïve Bayes and J48 have the best accuracy regarding correct classified instances, but Naïve Bayes performs better for the critical class 3. When studying the confusion matrices in Listing 9.10 and 9.12, we can see that J48 overestimates more in class 2 (23 vs. 14 instances classified as class 3) and underestimates more in class 3 (2 vs. 1 instance classified as class 2).

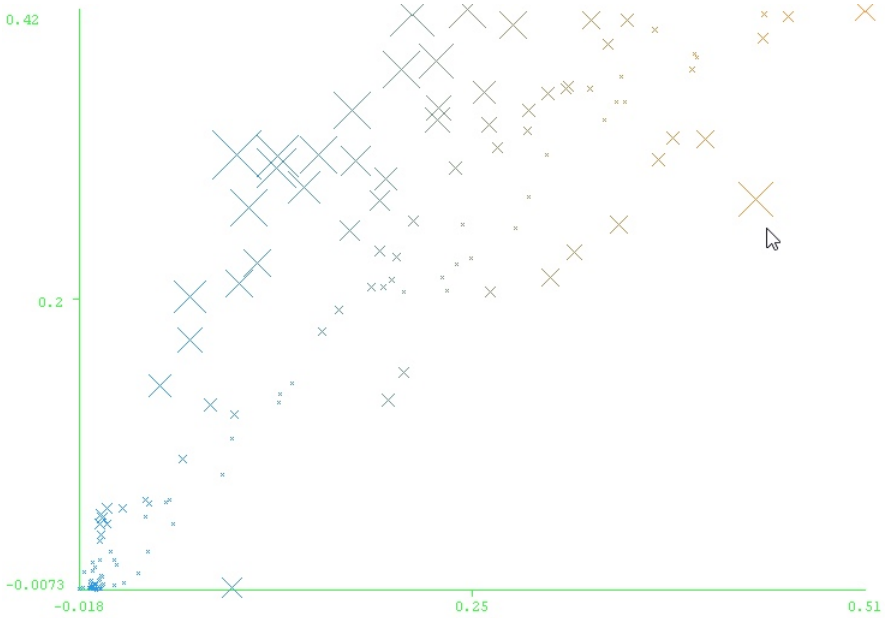

Figure 9.3: Visualisation of the classifier errors of the K* algorithm.

Table 9.1: Max, mean and class 3 threshold and percentage for all shackles.

	Max	**Mean**	**C3**	**C3%**
Shackle 85130	0.138	0.042	0.085	38%
Shackle 99871	0.337	0.099	0.225	40%
Shackle 99875	0.634	0.135	0.4	37%

Table 9.2: All results from the regression based algorithms.

	Shackle 88130		**Shackle 99871**		**Shackle 99875**		**Shackle 99875 (speed)**	
	MAE	*RAE*	*MAE*	*RAE*	*MAE*	*RAE*	*MAE*	*RAE*
Linear Reg.	0.0185	56.5%	0.044	66.9%	0.064	50.0%	0.066	51.1%
Multi-layer Percep.	0.0134	40.9%	0.032	48.2%	0.054	42.4%	0.059	45.5%
Lazy K*	0.014	42.6%	0.033	49.6%	0.045	35.3%	0.058	44.8%

Table 9.3: All results from the classification based algorithms.

	Shackle 88130		**Shackle 99871**		**Shackle 99875**		**Shackle 99875 (speed)**	
	CCI	*C3*	*CCI*	*Class3*	*CCI*	*C3*	*CCI*	*C3*
LibSVM	74%	79%	73.3%	64%	83.4%	67%	81.3%	60%
Naïve Bayes	72%	95%	72.6%	82%	84.8%	83%	81.3%	60%
J48	72.6%	89%	74%	82%	74.2%	67%	81.3%	60%

The results shows that for both regression based and classification based algorithms, the accuracy decreases when removing the direction. This indicates that the shackle data is related to current direction, as we assumed when discussing the tide effects on the current. Figure 9.4 shows the three classes and the distribution of the instances regarding the direction. It is a distinct top of class 1 instances around 180 degrees, which is where we believe that the tide is changing.

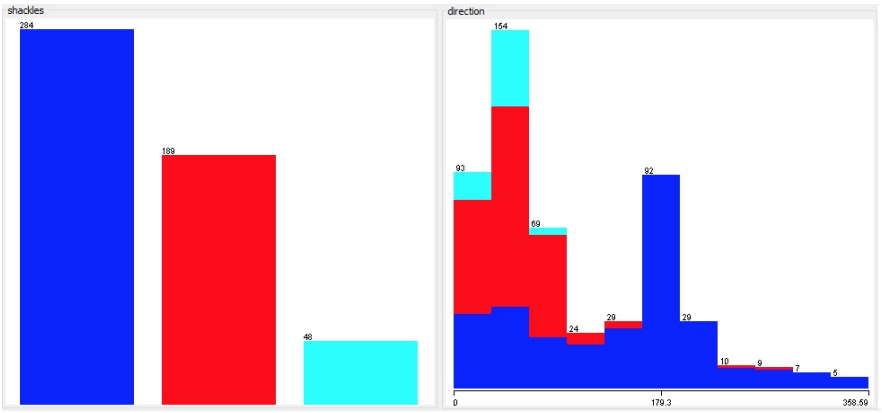

Figure 9.4: Three classes and and instance distribution regarding direction.

Because of the encouraging performance and since the class attribute is nominal, we chosen the Naïve Bayes as the algorithm to use in the ”Deviant Data from Past Similar Conditions” symptom. The advantage with a nominal class is that we can modify the classes depending on the individual shackle and regulate the critical strain level. The bad thing by manually assessing the classes is that the thresholds must be carefully chosen, most preferably by evidence in the data.

9.3 Project Results and Discussion

Because of the challenges discussed in Section 5.4.1, the implementation of the decision support system was limited. We chose to focus more on the method of symptom and case creation, rather than developing an empty shell of a program. This implies that the main results

are these symptoms and cases, and experience gathered through the project period.

Some of the things we have learned that it is important to pass on is that the load shackles on one rooster foot alone, does not provide much information that can be used to explain the structural state of the cage. As mentioned earlier in the report, these sensors can not register any impacts from the wind and wave. This is because the sensor is not sensitive enough for the impact the wind and wave have on the cage. This does not mean that the wind and wave have no impact on the cage at all. These weather effects have a larger impact on the parts of the production plant that is over the sea level and just over the crust.

An important factor for the impact on the cage is the tide. The data analysis shows that at the location of our platform Rataren, the tide is the force that decides the movement of the cage. We can see that the strain measurements follow the current measurements, and that the current is highly correlated with the tide. This is an important factor to consider when starting new production plants, and when attaching the moorings. As we had no options for monitoring the direction the strain was pulling in, we could not relate this direction to the current direction purely. We did see some correlation between the strain and the current direction, which means that this data is somehow dependent.

As we only had the sensors on the shackles on one rooster foot, the information these sensors provided was limited. We have earlier showed that the wind are correlated with the accelerometer data from the specialisation project. If the AHRS sensors was implemented on Rataren, we could combined this data with the shackle data. We could then try to identify the direction of the strain, which opens the possibility of comparing the sensor data with simulation data from FhSim. We could also do analysis to test if the -0.05 entries represent the cage moving ”back in place”. This assumption could also be confirmed or busted by having more strain sensors, at least on the rooster foot at the opposite side.

The symptoms and cases we have created are based on knowledge gathered in meetings with experts and research studies. We assume that the threshold values we have defined are reasonable for our artificial symptoms and cases, but we have no empirical data to verify it. This means that we can not evaluate the system regarding noise

in the data. It is also a problem that the symptoms and cases may not be realistic to look for at all, as we do not have any real incident reports to draw inspiration from. As the cage structure is dynamic and constantly in change, we can see that the measurements from the three available shackles are different. This may also occur if the moorings are tightened differently, which probably will occur over time as the cage goes through maintenance and natural wear. It is therefore important to not use too old data when creating models by machine learning, as the old data will be less representative as the time goes by.

The implemented system are working good according to the implementation of the cases and symptoms. When testing the similarity measure we have created, we run through the data file containing the measurements from 28.2 to 26.3 to see if the similarity is calculated correctly. By analysing this data manually, we know which instances that are matching with 100% similarity, and we know the id of those records. When testing the program and collecting the 100% similar instances, we see that the implemented program is able to catch all the entries it is supposed to.

Chapter 10

Conclusion & Future Work

10.1 Conclusion

In this MSc thesis we have developed a prototypical decision support system for predictive maintenance of exposed aquaculture structures, and early notification of possible structural damage. The system is based on a proposed architecture that consists of four modular components: Data Acquisition, Data Interpretation, Case-Based Reasoning and Graphical User Interface. When creating the proposed architecture, we draw inspirations from the fields of Structural Health Monitoring, machine learning and Case-Based Reasoning.

The most important parts of the decision support system is the symptoms and the cases, which are based on a data analysis and knowledge gathered by studying the domain of aquaculture structures. Knowledge from domain experts at SINTEF FH has also been invaluable as input when creating the symptoms and cases. The system uses CBR as the inference engine, which implies that the cases represent past situations or incidents. The current state or situation works as the input case, and the similarity between the input case and all past cases stored in the system is calculated. The user is then provided with a textual description and a solution from the most similar past case, if the similarity reaches a predefined threshold. If no cases are similar enough, it means that the current situation does not pose any threat that the system is aware of.

A major drawback during the project period was limited information it was possible to extract from the available data. This inferred that the symptoms and cases that optimally should be based on empirical data and real incidents, became artificial and based on our experience. We used the available data to create reasonable symptoms and cases, but the lack of incidents limited the possibilities of evaluating our work with real world data.

We can conclude with that the implemented decision support system is able to support an operator with advice about what to do, if a situation similar to previously experienced situations occur. This is verified by the tests performed on the system, which show that the performance regarding giving advice about the artificial cases are satisfying. It is reasonable to assume that when creating the symptoms and cases based on real world data, the threshold values and patterns we are looking for in our system must be revised drastically.

As we met several challenges with the available data, the need for a fully functional GUI became less important. The project was therefore more research based, and we focused on creating reasonable symptoms and cases, and proposing a proper architecture.

10.2 Future Work

The future work of this MSc thesis are mainly improving the decision support system and its necessary components. There are many challenges and problems to be solved in the different components of the architecture, and we will highlight some potential improvements that we find important for the future development.

Regarding the system, there are several parts, both minor and major, to improve. In the acquisition component, the main improvement will be to acquire information in real-time. It is expected that the future sensors store the registered data in a database, and the system must be capable of collecting and processing this data efficiently.

The interpretation component is considered the most important part of the system. In the prototype we provided the symptoms with preprocessed data sets, but the improved system must be capable of processing sensor data and model it to useful information automatically. It will be important to perform this process within reasonable

time. The current sensors sample once per hour, which means that the maximum processing time for the complete system must follow this upper limit. The system should optimally be as fast as possible if the future sensors sample at a higher frequency.

As the symptoms and cases are artificial in the prototype, new symptoms and cases based on real data should be created. This requires more sensors which provide a wider basis for describing the structural health of the production plant. By creating symptoms and cases based on real incidents that can be described by patterns in the data, these elements could be evaluated properly with empirical data. Together with an domain expert evaluation, this will eventually lead to testing the system in a real environment.

When more sensors are available with different types of data, a new analysis should be performed. This will reveal if there are any other relations of significance. After conducting the new analysis, one should consider:

- Which symptoms and cases could be created based on the available data?
- Should FhSim be considered?
- Could more advanced and accurate machine learning models be created?
- Are the previous challenges solved or still present?

It is also interesting to consider all the available data that is not primarily provided by the Exposed SFI, like the weather and tide forecasts. These forecasts could provide useful information for planning purposes, like when strong winds are predicted and when the tide is changing.

When developing the graphical user interface, the general aspects of a good GUI for an operator should be in focus. GUI designing for monitoring application are a field on its own, but it is important that the cases given by the CBR system occupies the attention of the operator. The operator should also be able to explore the cases, and receive all the relevant information. The development of the GUI should be in cooperation with the operators that are actually going to use it. Additionally, the operator should be able to retain a new case if desirable. Before retaining a new case, it should be revised by

domain experts.

When developing a complete system, it will be important that the code is written after good standards. The system should follow the modular structure defined by the architecture, and it should be easy to extend with new features and update with new symptoms and cases.

Bibliography

[Aamodt and Plaza, 1994] Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues,methodological variations, and system approaches. *AI Communications*, 7(1):39–59.

[ACE, 2016] ACE (2016). Anlegg âĂŞ geografisk plassering. URL: http://aceaqua.no/tristeinen-geografisk-plassering/.

[AGH, 2008] AGH (2008). Advanced lecture series on structural healthmonitoring. URL: http://www.gruppofrattura.it/pdf/eventi/2008/SHM

[Aqualine, a] Aqualine. aqualine - styrken teller. URL: http://aqualine.no/.

[Aqualine, b] Aqualine. Marine engineering. URL: http://aqualine.no/media/brosjyre/norsk/marine-engineering.pdf.

[Beal,] Beal, V. Bi - business intelligence. URL: http://www.webopedia.com/TERM/B/Business_Intelligence.html.

[Bonney, 2011] Bonney, W. (2011). Impacts and risks of adopting clinical decision support systems. *Efficient Decision Support Systems - Practice and Challenges in Biomedical Related Domain*, pages 21–30.

[Britannica, 2016] Britannica (2016). aquaculture. URL: http://kids.britannica.com/comptons/article-9272920/aquaculture.

[Caicedo and Dyke, 2005] Caicedo, J. M. and Dyke, S. J. (2005). Experimental validation of structural health monitoring for flexible bridge structures. *Structural Control and Health Monitoring*, 12(3-4):425–443.

[Darlington, 2011] Darlington, K. W. (2011). Designing for explanation in health care applications of expert systems. *SAGE Open*, 1(9).

[Dascotte et al., 2013] Dascotte, E., Strobbe, J., and Tygesen, U. (2013). Continuous stress monitoring of large structures. *5 th International Operational Modal Analysis Conference*.

[Delgado, 2005] Delgado, L. E. M. (2005). *A hybrid approach of knowledge-based reasoning for structural assessment*. PhD thesis, Universitat de Girona.

[FAO, 2016] FAO (2016). Aquaculture. URL: http://www.fao.org/fishery/aquaculture/en.

[Freudenthaler, 2011] Freudenthaler, B. (2011). Cbrshm - a case.based decision support system for semi-automated assessment of structures in terms of structural health monitoring. In Springer, editor, *Case-Based Reasoning Research and Development*, volume 6880 of *Lecture Notes in Artificial Intelligence*, pages 437–451.

[Gundersen et al., 2013] Gundersen, O. E., SÃÿrmo, F., Aamodt, A., and Skalle, P. (2013). A real-time decision support system for high cost oil-well drilling operations. *AI Magazine*, 34(1):21–32.

[Juang and Pappa, 1985] Juang, J.-N. and Pappa, R. S. (1985). An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627.

[Kamath et al., 2010] Kamath, G., Sundaram, R., Gupta, N., and Raho, M. (2010). Damage studies in composite structures for structural health monitoring using strain sensors. *Structural Health Monitoring*, 9(6):497–512.

[Kvistad, 2016] Kvistad, A. (2016). Norsk laks fra fjord til bord. URL: http://laks.no/lakseproduksjon/.

[M.G. Sandberg, 2012] M.G. Sandberg, AM. Lien, L. S. K. S. L. S. T. (2012). Erfaringer og analyser fra drift av oppdrettsanlegg pÃě eksponerte lokaliteter. Technical report, SINTEF.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

[Moore et al., 2000] Moore, M., Phares, B., Graybeal, B., Rolander, D., and Washer, G. (2000). Reliability of visual inspection for highway bridges. Technical Report 1, Federal Highway Administration Research and Technology.

[Murawski et al., 2012] Murawski, L., S.Opoka, K.Majewska, M.Mieloszyk, Ostachowicz, W., and Weintrit, A. (2012). Investigations of marine safety improvements by structural health monitoring systems. *TransNav*, 6(2):223–229.

[NOAA, 2015] NOAA (2015). What's the difference between a tide and a current? URL: http://oceanservice.noaa.gov/facts/tidescurrents.html.

[Okstad, 2014] Okstad, M. (2014). KjÃÿrte over merde med 200.000 laks - hevder han ikke merket det. URL: http://www.fosnafolket.no/nyheter/article9369199.ece.

[Pedersen and Roppestad, 2015] Pedersen, N. and Roppestad, F. (2015). Decision support system for exposed aquaculture operations. Technical report, Norwegian University of Science and Technology.

[Raghavan, 2007] Raghavan, A. (2007). *Guided-Wave Structural Health Monitoring*. PhD thesis, The University of Michigan.

[Ren et al., 2006a] Ren, L., Li, H.-N., Chou, J., Li, D.-S., and Sun, L. (2006a). Optical engineering. *Optical Engineering*, 45(8):084401–1 – 084401–9.

[Ren et al., 2006b] Ren, L., Li, H.-N., Zhou, J., Li, D.-S., and Sun, L. (2006b). Health monitoring system for offshore platform with fiber bragg grating sensors. *Optical Engineering*, 45(8).

[Richter and Weber, 2013] Richter, M. M. and Weber, R. O. (2013). *Case-based Reasoning, A textbook*. Springer.

[Rohrer, 2016] Rohrer, B. (2016). How to choose algorithms for microsoft azure machine learning. URL: https://azure.microsoft.com/en-gb/documentation/articles/machine-learning-algorithm-choice/.

[Russel and Norvig, 2010] Russel, S. and Norvig, P. (2010). *Artificial Intelligence: A modern Approach*. Prentice Hall, third edition.

[Serker and Wu, 2010] Serker, N. K. and Wu, Z. (2010). Structural health monitoring using static and dynamic strain data from long-gage distributed fbg sensor. In *Proceedings of the 2nd International Joint Conference of IABSE-JSCE on Advances in Bridge Engineering - II*, pages 519–526.

[SINTEF, 2015] SINTEF (2015). ForskningsomrÃěder. URL: http://exposedaquaculture.no/forskning.

[SINTEF, 2016] SINTEF (2016). OmrÃěde 2 âĂŞ overvÃěkning og beslutningsstÃÿtte. URL: http://exposedaquaculture.no/forskning/2-overvakning-og-beslutningsstotte/.

[Smith, 2016] Smith, S. W. (2016). The sampling theorem. URL: http://www.dspguide.com/ch3/2.htm.

[Stumptner et al., 2009] Stumptner, R., Freudenthaler, B., and KÃijng, J. (2009). On similarity in case-based reasoning for structural health monitoring. In *Computer Aided Systems Theory - EUROCAST 2009*, volume 5717 of *Lecture Notes in Computer Science*, pages 231–238. Springer.

[Utne et al., 2015] Utne, I., SchjÃÿlberg, I., and Holmen, I. (2015). Reducing risk in aquaculture by implementin autonomous systems and integrated operations. In *Safety and Reliability of Complex Engineered Systems: ESREL 2015*, pages 3661–3679.

[Wenzel, 2009] Wenzel, H. (2009). *Health Monitoring of Bridges*. Wiley.

[Wikipedia, 2015a] Wikipedia (2015a). Data analysis. URL: https://en.wikipedia.org/wiki/Data$_a nalysis$.

[Wikipedia, 2015b] Wikipedia (2015b). Decision support system. URL: https://en.wikipedia.org/wiki/Decision$_s upport_s ystem$.

[Wikipedia, 2015c] Wikipedia (2015c). Degrees of freedom. URL: https://en.wikipedia.org/wiki/Degrees$_o f_f reedom$.

[Wikipedia, 2015d] Wikipedia (2015d). Distributed bragg reflector. URL: https://en.wikipedia.org/wiki/Distributed$_B ragg_r eflector$.

[Wikipedia, 2015e] Wikipedia (2015e). Fiber bragg grating. URL: https://en.wikipedia.org/wiki/Fiber$_B ragg_g rating$.

[Wikipedia, 2016a] Wikipedia (2016a). Aquaculture. URL: https://en.wikipedia.org/wiki/Aquaculture.

[Wikipedia, 2016b] Wikipedia (2016b). Artificial neural network. URL: https://en.wikipedia.org/wiki/Artificial$_n$*eural*$_n$*etwork*.

[Wikipedia, 2016c] Wikipedia (2016c). Data science. URL: https://en.wikipedia.org/wiki/Data$_s$*cience*.

[Wikipedia, 2016d] Wikipedia (2016d). Decision tree learning. URL: https://en.wikipedia.org/wiki/Decision$_t$*ree*$_l$*earning*.

[Wikipedia, 2016e] Wikipedia (2016e). Fiskeoppdrett. URL: https://no.wikipedia.org/wiki/Fiskeoppdrett.

[Wikipedia, 2016f] Wikipedia (2016f). Instance-based learning. URL: https://en.wikipedia.org/wiki/Instance-based$_l$*earning*.

[Wikipedia, 2016g] Wikipedia (2016g). Linear regression. URL: https://en.wikipedia.org/wiki/Linear$_r$*egression*.

[Wikipedia, 2016h] Wikipedia (2016h). Naive bayes classifier. URL: https://en.wikipedia.org/wiki/Naive$_B$*ayes*$_c$*lassifier*.

[Wikipedia, 2016i] Wikipedia (2016i). Overfitting. URL: https://en.wikipedia.org/wiki/Overfitting.

[Wikipedia, 2016j] Wikipedia (2016j). Supervised learning. URL: https://en.wikipedia.org/wiki/Supervised$_l$*earning*.

[Wikipedia, 2016k] Wikipedia (2016k). Support vector machine. URL: https://en.wikipedia.org/wiki/Support$_v$*ector*$_m$*achine*.

[Worden et al., 2007] Worden, K., Farrar, C. R., Manson, G., and Park, G. (2007). The fundamental axioms of structural health monitoring. *Proceedings of the Royal Society A*, 463.