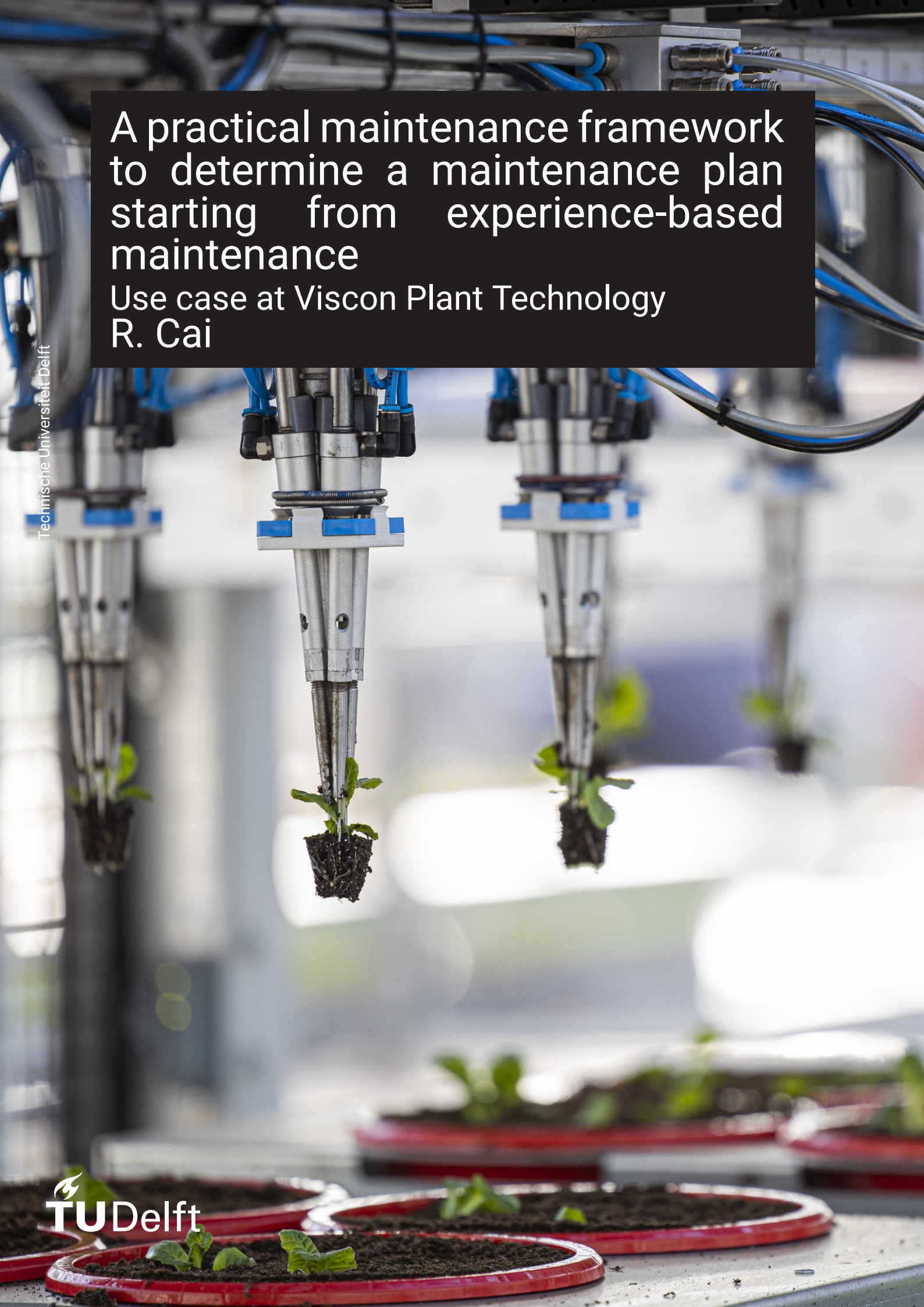
A practical maintenance framework to determine a maintenance plan starting from experience-based maintenance

Use case at Viscon Plant Technology

R. Cai

Technische Universiteit Delft



TU Delft

A practical maintenance framework to determine a maintenance plan starting from experience-based maintenance

by

Ruili Cai

Master thesis

In partial fulfilment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical, Maritime and Materials Engineering of Delft University of Technology
to be defended publicly on

Student number:	4582233	
MSc track:	Multi-Machine Engineering	
Report number:	2022.MME.8754	
Project duration:	February 12, 2022 – February 9, 2023	
Supervisor:	Ir. W. Bos,	Supervisor TU Delft
Thesis committee:	Dr. Ir. D. L. Schott,	Tu Delft committee Chair, 3ME
	Ir. P.J. Jonker,	Company Supervisor Viscon Group
	Ir. A. Shah,	Company Supervisor Viscon Group

TU Delft

Preface

This report is the result of my thesis project in the field of Mechanical Engineering at Technical University of Delft. This report presents the graduation project which has been conducted at Viscon Plant Technology B.V. in 's-Gravendeel. Viscon Plant Technology B.V. is a business unit (subsidiary) of Viscon Group. The Viscon Group provides automation solutions for industrial automation with a total of 13 business units. Viscon Plant Technology B.V. is the oldest business unit of the group and provides greenhouse and nursery automation.

It would not have been possible to complete this thesis project without the support of other people.

First of all, I would like to thank Wouter van den Bos and Dingena Schott from the university who guided me during my project and provided me with feedback on my work.

Secondly, I also want to express my gratitude to all the employees at Viscon Group. Especially my company supervisors: Hans Jonker and Aadesh Shah. Thank you for all the time you spent in me and for all the discussions and conversations we had. Furthermore, I want to thank all the employees at Viscon Group for making time for me and always being open to questions and discussions.

Finally, I want to thank everyone who supported me during my thesis project. First of all, I would like to thank my parents and my girlfriend Emi Huang for their continued support. Then, I would like to thank all my friends who gave me advice and cared about my thesis project. Finally, I want to thank my friend and supervisor Michel Pan. Thank you for all the time you spend. Your feedback and lessons have greatly improved the quality of my report.

Once more to everyone, thank you for your help! It has been a great journey.

Abstract

Maintenance of complex machines or assemblies is an essential part of machine-building companies. Besides building and designing machines, they are responsible for replacing components and restoring the machines to maintain good operational conditions. An optimal maintenance plan ensures that customers spend less on maintenance and remain satisfied. Also, for the machine builders, an optimal maintenance plan shows efficient use of the service mechanics, trust in the machines and the quality of the knowledge. This research aims to create a maintenance framework that leads to a suitable maintenance plan starting from experience-based maintenance. The framework proposes steps in which initial data points are collected, calculates the maintenance intervals and estimates values for a failure distribution to obtain knowledge of components, while it is still manageable (practical) for the service department. Eventually, this paper shows the benefits of the proposed maintenance framework relative to maintenance based on experiences like reactive maintenance. A simulation model has been developed, showing the KPIs of the framework and results from the framework. In addition to the simulations, this project provides guidelines for the service department to implement the framework.

Summary

This research aims to design a practical maintenance framework for the service department of machine-building companies. Machine-building companies like Viscon Plant Technology (VPT) execute maintenance based on their experience. In their current situation, few or incomplete data points are collected. To become the market leader in their relevant sector, VPT wants to improve their maintenance plan and this research provides the first steps towards a data-oriented company.

VPT and possibly other machine-building companies execute maintenance based on 'break and fix' or 'check and replace'. 'Break and fix' also called reactive maintenance is maintenance after a component has failed. In contrast, 'check and replace' also called scheduled-based maintenance, is a form of preventive maintenance determined based on maintenance intervals. The service department mainly determines these intervals from experience. During that maintenance, the condition of components is checked through a visual inspection before replacement. Usually, maintenance will be executed prematurely or after the component's failure because of the uncertainty of failure. Both the frequency of maintenance and the failure of components have a high impact on the total maintenance costs. Therefore, we investigate how the service department can determine an appropriate maintenance plan starting from experienced-based maintenance plans.

Firstly, we dived into the literature for the available maintenance policy and concepts. Several maintenance policies and concepts were found, discussed and evaluated. The maintenance policies that were feasible from experienced-based maintenance plans were scheduled-based maintenance, hour-based and usage-based maintenance because of the low-level data readiness and investments, while still considered as an improvement in knowledge about the component failure and reduced maintenance costs. Therefore, these were the maintenance policies to focus on in this project.

Afterwards, we needed to define the KPIs to determine the usefulness of this project. For appropriate maintenance, we need to determine the accuracy of the maintenance interval based on the total maintenance costs. Furthermore, we also need to determine the accuracy of the knowledge gained of the component's failure behaviour.

We developed a maintenance framework based on the PDCA approach to determine an optimal maintenance interval and whether a given maintenance interval from the previous section is suitable. The PDCA approach is an excellent structure which is iterative. Collecting data from each maintenance cycle, the KPIs are becoming accurate, which leads to an accurate determination of maintenance intervals and optimal maintenance costs.

The proposed maintenance framework starts with the planning phase of the PDCA cycle. The first planning phase is the component selection. Due to the many components in a machine, we need to select the most important ones with specific failure mechanisms or based on expert knowledge. After that, the service department needs to gather the most basic data points, which can already be requested from resources. Data such as service life, corresponding failure mechanisms and relevant costs are general data given in technical data sheets from the Original Equipment Manufacturer (OEM). Lastly, the planning phase also consists of collecting the productivity data of machines. The productivity data translates the technical data into the service life of a machine in running hours.

After the planning phase, the doing phase is conducted by a Maintenance optimisation model. The Maintenance optimisation model is based on the Block replacement model. We extended the Block replacement model by adding a variable for peak season intervals and an algorithm for grouping components. In addition, the Maintenance optimisation model needs additional failure distribution from components. From the literature, the Weibull distribution should be the most common distribution for modelling failure times. The Weibull parameter can be divided into the shape and scale parameters. A shape parameter tells about the scatteredness of failure of time. A high shape parameter means relatively uniform failure intervals. The scale parameter is the time at which 63.2% of the same component has failed. Since the OEM does not provide these data, the service department should start with an initial value. Our proposal is to start with a high shape parameter value and the scale parameter is determined based on the service life. By keeping the shape parameter constant, the model tries to optimise the scale parameter first.

After calculating the maintenance interval from the Maintenance optimisation model, the checking phase is the actual maintenance execution by the service mechanics. The service mechanics execute maintenance by performing a visual inspection of the machine and replacement for the component which reached its maintenance interval or has been failed. The only added task of the service mechanic is the data collection of the replaced and failed component. The failed component is labelled in failure time and the replaced component without failure will be labelled as suspension time.

After the checking phase, the service department has to estimate the Weibull parameters in the adjustment phase. The estimation method starts with the estimation of only the scale parameter. The scale parameter increases if zero failures have occurred and after the first two failures, the scale parameter of a component decreases. Components with three or more than three failures will be estimated by the Maximum likelihood estimation (MLE). The MLE estimates both shape and scale parameters.

The shape and the scale parameters are the most important values for the service department. The shape parameter gives insights about the failure scatteredness over time, while the scale parameter gives insights into the characteristics of service life in which 62.3 % of the same type of component will fail at that specific hour. Both insights allow for determining an appropriate maintenance plan for the service department.

Due to time restrictions, we implemented the proposed maintenance framework through a simulation. We developed a simulation model to compare the outcomes of the estimates of the Weibull parameters with true given values. The simulation model consists of the inputs from the planning phase, the Maintenance optimisation model from the doing phase, the generated lifetime based on given true values, sorting them in failure and suspension time from the checking phase and the estimation method from the adjustment phase. After the adjustment phase, it starts over again from the Maintenance optimisation model. In between, multiple data sets are recorded for verification and the total maintenance costs over machine years. The simulation model can be converted into a separate Maintenance optimisation model and estimation method as calculation tools for the service department.

A use case is performed for the implementation of the proposed maintenance framework. The planning phase data was collected for a system from a machine of VPT. After that, we added the input to the simulation model. The simulation model calculates the maintenance intervals and the total maintenance costs and estimates the Weibull parameters for each maintenance cycle over the course of 200 machine years.

The use case resulted in most components being accurately estimated after three failures. It also shows that components with relative random failure intervals equal to or lower than a shape parameter of two should not be scheduled as a constant maintenance interval. Those components are scheduled for either a short maintenance interval, in which maintenance is most likely executed prematurely or a long maintenance interval, in which failures occur frequently. Also, the total maintenance costs over the course of 200 machine years show that the maintenance costs of the proposed maintenance framework can be higher than the reactive maintenance. Components with a shape parameter equal to or lower than two have high deviations due to random failures. On the other hand, components higher than a shape parameter of three remain in a high shape parameter. It means less significant difference as a high shape parameter. Therefore, the advice is to use the proposed maintenance framework till a component reaches a shape parameter of two. Then the service department can investigate further to those components that may need other maintenance policies.

Two hundred machine years seems long. Many machines could be changed due to innovative technologies. However, two hundred machine years is not equal to two hundred years. Usually, a couple of machines are used simultaneously and multiple components are assembled in a machine which 200 machine years are reached quicker.

Undoubtedly, the cost for the customer would be reduced, which they remain satisfied, the machine-building company builds trust and reputation and the service department would gain knowledge about the component's failure behaviour. It provides additional knowledge to be more efficient for the machines and service mechanics. Reduced maintenance costs deliver more customers to buy a service contract in which the company controls more machines that speed up the data collection for different productivity and situations to keep using the proposed maintenance framework.

List of definitions

Term	Definition
Accelerated life test	The process of testing a system or material to its failure in a short amount of time.
Age replacement model	A maintenance model to calculate the time of maintenance of a component based on its failure time or preventive maintenance time.
Aggressive maintenance	A term of continuously improving and redesigning components.
Analytical hierarchy Process	A technique for the Multi criteria decision making which considers qualitative and quantitative elements.
Analytical network process	A technique for the Multi criteria decision making which considers qualitative and quantitative elements and adds the internal relation between the elements.
Block replacement model	A maintenance model to calculate the time of maintenance of a component based on the optimal time interval.
Clustering	The combination of components maintaining on the same moment.
Condition-based monitoring	A part of preventive maintenance which is executed after a certain control limit is reached, based on the state of a part.
Data points	Different measurement points to determine the state of a component.
Downtime	All the time a machine is unavailable for production.
Failure mechanism	A process that leads to a component failure.
Framework	A structure intended to serve as a support or guide for the building of something that expands the structure into something useful (Lutkevich, 2020).
Industry 4.0	The latest part of the growing trend towards the technology of automation and data exchange.
Length of interval	The interval of maintenance execution time.
Maintenance concept	The process towards maintenance policies.
Maintenance cycle	The interval between two consecutive maintenance actions.
Maintenance policy	The maintenance method (either RM, PM or PdM) with corresponding cost per year and maintenance time on component level.
Maximum likelihood estimation	A method to estimate parameters of an assumed probability distribution.
Measuring attractiveness by a categorical-based evaluation technique	A technique for the Multi criteria decision making which attempts to avoid the difficulty felt by decision-makers with their preference of critical aspects.
Median rank regression	A specific method for Weibull parameters estimation.
Monte Carlo simulation	A simulation with a range of possible outcomes and probabilities.
Multi-criteria decision-making	The analysis of various available choices in a situation.
On-site service mechanics	The service mechanics or technicians at the customer company which can execute maintenance.
Overall Equipment Effectiveness	The standard to calculate machine effectiveness and manufacturing productivity.
Peak season	The time interval when machines are used continuously.
Piecewise function	A function defined by multiple sub-functions, where each sub-function applies to a different interval.
Predictive maintenance	A company environment which is continuously able to predict when maintenance should be executed through continuous monitoring.
Preventive maintenance	Maintenance method executed before the actual failure of a component.
Reactive maintenance	Maintenance execution after a failure of a component to restore it to its required performance level.
Reliability	The probability of performing consistently well of a component.
Reliability Centered Maintenance	A maintenance concept which determine the most suitable maintenance policy.

Term	Definition
SAE JA101	A standard for Reliability Centered Maintenance (RCM).
Scale parameter	A Weibull parameter which represents the characteristic life. It is defined as the time at which 63.2% of a component has failed.
Scheduled based maintenance	A preventive maintenance which is executed based on a service contract to execute components simultaneously.
Service life	Mostly reffered to B10 value. The life of a component in which 10 procent of the same type of component has failed.
Setup costs	Fixed costs per maintenance visit due to traveling and visit preparation.
Shape parameter	A Weibull parameter which represents the behavior when a component is likely to fail.
Spare parts	A replacement component that is kept in an inventory till the old component is replaced
Suspension data	The failure data which is censored due to maintenance or uncommon failures causes
Technique for order of preference by similarity to Ideal Solution	A technique for the MCDM which the Euclidean distance approach is used
Total productive maintenance	A maintenance concept with the involvement of machines, equipment's, employees and supporting processes to improve the quality of system.
Uptime	All the production time plus idle time.
Use-based maintenance	A preventive maintenance method which is executed after a certain usage count or age, disregarding the actual state of the component.

List of abbreviations

Abbreviations	Definition
AHP	Analytical Hierarchy Process
ALT	Accelerated life test
ANP	Analytical Network Process
ARM	Age replacement model
B10	Bearing life
CBM	Condition-based monitoring
FCGR	Fatigue crack growth rate
FME(C)A	Failure Mode Effect (& Criticality) Analysis
FTA	Fault Tree Analysis
LoI	Length of interval
MCDM	Multi criteria decision making
MLE	Maximum likelihood estimation
MRR	Median rank regression
OEE	Overall Equipment Effectiveness
OEM	Original Equipment Manufacturer
PDCA	Plan Do Check Act
PM	Preventive maintenance
RCM	Reliability Centered Maintenance
RM	Reactive maintenance
RP	Renewal Process
SAE JA101	Society of Automotive Engineers
SBM	Scheduled-based maintenance
TEC	Total Expected Costs
TPM	Total Productive Maintenance
UM	Use-based maintenance
VPT	Viscon Plant Technology

List of symbols

Variable	Definition
a	Crack length [m]
A	Fatigue crack growth rate (FCGR)
a_b	Acceleration [m]
b	Belt width [mm]
B_n	Actual bearing force of bearing n [N]
C	Basic dynamic load rating [N]
C_{comp}	Component cost
C_{lost}	Lost of production costs per hour
C_mh	Manhour cost per hour
C_{pen}	Penalty costs of RM
C_{PM}	Total PM cost
C_{RM}	Total RM cost
C_s	Setup costs
$\frac{da}{dN}$	Crack growth rate [$\frac{m}{cycle}$]
e	Exponential constant
$f(t)$	Probability density function;
F_{ab}	Acceleration force [N]
f_e	Fundamental frequency [Hz]
F_H	Vertical lifting force [N]
F_R	Resisting force of friction [N]
F_{TV}	Pretension forces per belt side [N]
F_U	Peripheral force [N
F_{wdyn}	Dynamic pulley load [N]
g	Gravity acceleration
$H(t)$	The expected number of failures in interval
k	wear coefficient or specific wear rate [$\frac{mm^3}{N \cdot m}$]
l	Belt length
L_{10}	Basic rating life (at 90% reliability) [millions of revolutions]
L_{10h}	Basic rating life (at 90% reliability) [operating hours]
L_{10m}	SKF rating life (at 90% reliability) [million revolutions]
ln	Natural logarithm
l_T	Span length corresponding to the frequency [m]
L_{travel}	Average distance per cycle for pick and place [mm]
m	Total mass [kg]
m_{belt}	Belt mass [kg]
m_{gr}	Crack growth rate exponent
m_{spez}	Specific belt mass [kg/mm belt width and per m belt length]
n	Rotational speed [r/min]
P	Equivalent dynamic bearing load [N]
p	Exponent of the life equation
Q	Wear rate [$\frac{mm^3}{mm}$]
r	Radius
$R(t)$	Reliability
t	unit [usually in time]
Tm	Optimal time of maintenance
T_{repair}	Repair time

Variable	Definition
T_{res}	Visit response time
$t_{service}$	service life [hours]
$U_1, U_2, ...$	Life cycle fraction under the conditions1,2, ... $[U_1 + U_2 + ... + U_n]$
W	Normal force [N]
y	Crack geometry factor
$Z_i(t)$	total maintenance cost per time unit of component i
α	Reduced factor of lost production outside peak season
β	Shape parameter
η	Scale parameter
σ	Stress
σ_{max}	Maximum stress
σ_{mean}	Mean stress
σ_{min}	Minimal stress
$\Delta\sigma$	The change in stress
μ	Friction coefficient

List of Tables

List of Figures

Contents

1

Introduction

This research is the result of a graduation project conducted at Viscon Plant Technology B.V. (VPT) from 's-Gravendeel, The Netherlands, in order to be awarded a master's degree in mechanical engineering with the track Multi-machine engineering at Delft University of Technology.

The horticulture sector is one of the fastest-growing industries worldwide (Bhatt et al., 2022). One of the reasons is the numerous changes in the horticulture sector. It is facing challenges such as a lack of skilled labours, climate changes and increased plant quantities. Consequently, there is a high demand for automation in the horticulture sector. Viscon Plant Technology (VPT) is one of those companies focusing on the automation process in the horticulture sector. They provide many kinds of machinery that automate the process of growing, harvesting or picking up plants and relocating from one pot to another. Most people will probably consider designing top-quality machines to become the market leader in the automated process in the horticulture sector. Characteristics of top-quality machines are, for example, high productivity, quality components or economic use. However, an important aspect that is often undervalued is the maintenance of machines. No matter how well machines are designed, machines degrade over time. Then it is important to replace them in time before machines fail. Once failed, the automation process cannot continue, and customers lose their sales. That is why it is crucial to have an appropriate maintenance plan for the service department of the machine-building company that gives customers trust in the machine and they are more likely to follow the maintenance plan of the service department.

For years, the service department of VPT can be characterised as a "break and fix" organisation. Maintenance starts when a customer requires it because something is wrong with the machine. The "break and fix" occurs unexpectedly. Usually, it is an unprepared event that must be fixed rapidly.

Another commonly used policy is the "check and replace", but for VPT, it is still in its infancy. Maintenance is executed when machines reach a specific time interval, in which they check components and replace them if it is necessary. It may lead to more benefits than the break and fix since soon-to-failure components are identified before failure. However, the "check and replace" can lead to unnecessary checks and can be conducted prematurely. This means that a replacement may have been executed at an interval too early or too late. Both "break and fix" and "check and replace" are mainly based on experience and the gut feeling of the service mechanics.

Therefore, besides innovating machines, VPT wants to be able to evaluate when maintenance needs to be executed. They want to determine an appropriate maintenance plan by collecting the right data to gain insights into the component failure behaviour and apply a maintenance plan accordingly.

This research focuses on the maintenance gap between the current situation and the ultimate goal of VPT. VPT wants to transform into a data-orientated company by collecting different data sets that predict or replace failures right before the failure. This project proposes a "guideline" for machines that produce certain products or results. The guideline helps the service department to identify the data sets. The knowledge gained from data will open up new insights into improving "the check and replace", or it will result in an alternative maintenance plan because it may show that the check and replace is not suitable. A convincing structure or guideline must be created. Together with the literature and the service department of VPT, a design to determine a suitable maintenance plan for machine-building companies starting from experience-based maintenance is developed. This project uses a simulation model and a transplanting machine of VPT as a use case to show the results of the design of this project.

1.1. Research design

The purpose of this section is to describe the project. First, the aim of the study and the scope of the project are discussed. It is followed by the relevance for VPT. Eventually, the goals and research questions are described.

1.1.1. Aim of the study and scope

This research sets out to investigate an appropriate maintenance plan from experienced-based data sets and the insights of these data sets for providing an alternative maintenance plan for machine-building companies. In this case, this project helps VPT to give them insights for determining a maintenance plan. The current "break and fix" and the "check and replace" have solely been executed based on experience. Therefore, poor information is collected about the current situation and the service department has little indication about the performance of their maintenance plan.

1.1.2. Problem statement

Currently, the maintenance plan at VPT is to wait till a call from customers or schedule an inspection prematurely. Based on experience and observation on the machine, maintenance will be executed. This could lead to an uncertainty of premature maintenance and history could repeat itself by occurring unexpected failures. These two types of maintenance affect negative maintenance costs for customers. These two types of costs are as follows:

- Setup costs: These are fixed costs for each maintenance execution. Usually, service mechanics have to travel to the customers and make any administration and preparation costs before executing any maintenance. In addition, machines are sold worldwide, which is an expensive cost factor if service mechanics have to travel. Therefore, if the frequency of maintenance increases due to a premature "check and replace" maintenance plan, the total maintenance costs increase rapidly.
- Downtime costs: These are the costs of not operating the machines due to a failed component. Every hour of downtime may cause an enormous profit reduction for the customers.

In the current situation, none of the costs, the failure behaviour of components, or any other data are well documented. This means that little data is used for determining a maintenance plan. The maintenance decision-making is solely based on the experience and gut feeling of the service department and their service mechanics. Any improvements in maintenance remain unknown. The reason for remaining unknown improvements on maintenance is because of the following activities:

1. **The company has barely done any research to obtain insights on the improvements in maintenance plans**: The current maintenance policy in VPT and possibly other machine-building companies are reactive, and some are scheduled maintenance based on expert knowledge or experiences. The general interpretation of improvement in maintenance in VPT is the prevention of reactive maintenance. As long as service department replaces the components in time, that is already a sign of good maintenance. However, each maintenance can be executed prematurely. Therefore, the indication of the amount of reactive maintenance does express the whole performance of a maintenance plan.

   Some research needs to be spent on maintenance. Research can show how the improvement of maintenance can be conducted and what needs to be done to determine a maintenance plan. Nevertheless, adopting a maintenance plan from research takes time to implement and convince the service department and customers. Furthermore, it requires additional scientific knowledge. Therefore, little attention is paid to the research on improving maintenance plans from the literature.

2. **There is no data collection**: The company does not collect any data. A visual inspection is performed before the service mechanics know which components must be replaced. Components are observed superficially. Each service mechanic writes down in their way what they interpret during the visual inspection and replace it if they believe it is necessary. No structured qualitative and quantitative data points are collected that can be used to know the failure behaviour of the components and determine a maintenance plan accordingly.

   In addition, the service department needs to figure out which data sets need to be collected and how they can be collected. Data gives the knowledge and evidence that events have occurred in the past.

When data is structured and collected, the service department can compare and analyse the same qualitative data points. Quantitative data points can be used to improve maintenance decisions. Therefore, comprehensive data collection could improve maintenance policy and save service costs and high machine reliability knowledge.

3. **The current situation is practical**: All maintenance decisions are based on experience and knowledge. It makes maintenance practical because little effort is required. Therefore, the flow of a maintenance plan hardly has any structure. A structure is necessary to improve a maintenance plan consistently. A framework or any similar tools provide steps or guidelines in a structured manner in which a maintenance plan is determined structurally. In addition, it can act as a tool to generate an overview that follows into results and alternatives.

These three challenges that lead to unknown improvements in the maintenance plan are addressed in this project. VPT and other similar machine-building companies can use this project to gain knowledge of determining a maintenance plan while still being practical. A maintenance plan that is continuously improved can achieve significant benefits in the future. This project investigates the design of collecting data and giving insights for maintenance plans starting from experience-based data while still being practical. Hence, this study is intended for all companies designing, building, and selling automated production machines. We use a machine from VPT as a use case to show the usefulness of the design of this project with a simulation model. Obtaining insights into the maintenance plans should lead to a better maintenance determination which reduces costs, increases customer relationships, and increases knowledge about the component's failure behaviour and time efficiency for the service department significantly.

1.1.3. Scientific contribution

The research gap covers the gap between academic research and practicality for the service department of machine-building companies. The scientific contribution is the design and choice of a guideline for the service department to supply increased knowledge and steps towards determining a suitable maintenance plan for machine-building companies. The guidance provides the research on the initial data collection. It uses these data to show results of reactive (break and fix) and preventive (check and replace) maintenance and the failure behaviour of components for the service department. This project contributes to companies like VPT that are unfamiliar with collecting data to gain increased knowledge for determining a maintenance plan. Therefore, the project gives insights into the practical usefulness of the designed choice of this project based on the combination of the practical point of view and literature research.

1.1.4. Research goal and questions

The main goal of this project for VPT is to take the first steps towards a data-orientated company. Tackling the three challenges as discussed in the previous section translates into the following goal of this project: Introduce a design that provides data collection and the usefulness of these data that can lead to increased knowledge between maintenance and components' failure for the service department of machine-building companies while still being practical for the service department. The service department can determine a suitable maintenance plan with the increased knowledge. Together, the proposed design in this project contributes to which data to collect to get insights and helps VPT service department in the following aspects:

1. Optimise the maintenance interval from the collected data;
2. Or obtain insights that show that the current maintenance plan is not appropriate. Therefore, some components need an alternative maintenance plan.

The main research question towards the research goal is as follows: **How can the service department of a machine-building company determine a maintenance plan starting from experience-based maintenance?**

We distinguish four sub-questions in order to answer the main research question.

1. What are the available maintenance policies and concepts in the literature that contribute to a maintenance plan?
2. What are the Key Performance Indicators to validate the usefulness of the proposed methodology?
3. What maintenance strategy is proposed?

   (a) What are the available data?
   (b) What data need to be collected during maintenance?
   (c) How can the maintenance schedule be optimised?
   (d) How will this maintenance flow determine a maintenance plan?

4. What are the benefits of the maintenance methodology relative to the initial maintenance plan?

   The result of the design of this project may be used as a proof of concept. Analysis and simulation models will be done in Python, which is a software used for mathematical computation.

1.1.5. Project approach

The project approach includes the steps of the process that are taken in order to answer the research questions. Therefore, the project approach steps are listed in chronological order and each research question is answered sequentially.

1. Literature review; **Chapter 2** provides an overview of the maintenance policy and concepts in the literature. This information will give insights, techniques and definitions about maintenance for this project.
2. Methodology; **Chapter 3** presents the design and choice of a structured maintenance methodology. Based on the literature review and the research design, we evaluate which maintenance policies, concepts and techniques are useful in determining a suitable maintenance interval.
3. Proposed methodology framework; **Chapter 4** presents the proposed maintenance framework in depth. The maintenance framework is based on the proposed methodology and is divided into multiple modules. Each module is described.
4. Simulation model; **Chapter 5** describes the simulation model that is based on the proposed maintenance framework. The input, process and the output of the simulation model are explained.
5. Use case; **Chapter 6** presents a use case of a machine from VPT and the simulation model is implemented. The results of the use case on the simulation model are shown.
6. Discussion; **Chapter 7** discusses the results from the use case and the proposed maintenance framework.
7. Conclusion and recommendations; **Chapter 8** draws general conclusions based on answering the research questions. Furthermore, limitations, future research and recommendations are described after the conclusion.

2

Literature review

This chapter consists of two parts: It translates the current situation of VPT to the academic terminology on maintenance plans and shows the available methods and techniques of the maintenance field in the literature.

Over the past decades, substantial research has been done on maintenance policies, concepts, and other maintenance-related definitions. These maintenance-related terminologies have become more general, while it also becomes complicated to implement them in practice (Narayan, 2005 and Smith and Mobley, 2003). Papers, such as Ding and Kamaruddin, 2014 and Fraser et al., 2015 review various maintenance policies, strategies, selection problems or concepts on different machines, systems or components.

In this project, we discuss what is already available and search for the concepts, techniques and tools in the literature to determine a maintenance plan. We are defining the maintenance policy first to have a common ground regarding the essential terminology. A maintenance strategy is mainly referred to as maintenance policy, and maintenance management involves managing different maintenance-related people (Narayan, 2005). Maintenance concepts are steps or instructions to determine the maintenance policy. Ultimately, a maintenance policy is chosen for each component or machine.

We divide maintenance into two categories; The maintenance that is executed based on the maintenance policy and the steps towards determining the maintenance policy as maintenance concepts. Thus, the maintenance concepts can include steps, techniques, tools and guidance to determine a maintenance policy. A maintenance policy provides the time to execute maintenance. With both knowing maintenance policy and concept, the first research question can be answered: **What are the available maintenance policies and concepts in the literature that contribute to a maintenance plan?**

2.1. Maintenance policies

2.1.1. Reactive maintenance

Reactive maintenance (RM), also known as corrective or firefighting maintenance, is executed when a component has failed or obviously almost failing. A component has been used over or on its maximum lifetime. Maintenance will be executed to replace or repair the (almost) failed component and optionally inspect other components for damages. The failure of a component can occur unexpectedly, causing unplanned and significant downtime because customers react when failure has become noticeable. Therefore, the maintenance preparation will take place after the failure, increasing the downtime. Thus, the customer will make an appointment for maintenance and wait until the machine is fixed to an operational state. It is the most primitive maintenance policy, which does not necessarily mean that this policy is the worst. If no attention or analysis is given to the machine, the maintenance policy will be automatically categorised as RM. However, if some analysis is conducted and the unplanned breakdown is minor, RM can save costs. RM would be a fine choice.

2.1.2. Preventive maintenance

Preventive maintenance (PM) aims at reducing the failure risk or performance deterioration of the component by replacing or repairing components before failure. The time for maintenance can be determined by reaching specific indications such as running hours, meters or counts (use-based maintenance) or a prede-

termined interval (scheduled-based maintenance) or a specific limit on a component's condition (condition-based maintenance). So PM branches into different indicators for the maintenance intervals.

PM reduces the chances of an unexpected failure, high costs and consequences than RM by having maintenance before a failure. PM could benefit from catching the problems before a breakdown. However, maintaining the components too early provides unnecessary high costs due to over-maintaining components. Each preventive-based maintenance is explained in the following section.

Scheduled-based maintenance

Scheduled-based maintenance (SBM) is maintenance based on a time interval. The time interval determining is an agreement between the service department and the customer. The intervals are usually chosen either by the customers because there is an opportunity to execute maintenance due to a low operating period or by the service department because they believe it is necessary. During maintenance, service mechanics perform a visual inspection of the entire machine. The replacement or repair of components is performed after the visual inspection if it is necessary.

Components in a defective state are considered to be replaced or repaired, and components in a good state will be used again until the next maintenance interval. The indication time of SBM does not indicate the failures of components. That is why it is necessary to thoroughly investigate each component to determine whether a component has to be replaced.

Use-based maintenance

Use-based maintenance (UM) is the maintenance of components based on intervals of reaching certain hours, meters or counts for each component. Hours, meters or counts can be interpreted differently. Some machines may have only two modes: on and off mode. The hour that is on is proportional to the travelled meters. However, when multiple modes or capacity levels exist in a machine, the usage of components will change over the travelled units (for example, distance, rotations, flow or energy) on the machine while the time remains constant at a different usage. Therefore, UM can be split up further into usage-based and hour-based maintenance. Its specific unit, which is primarily the running hours of a component, is mainly given by the component's original equipment manufacturer (OEM). The OEM defines the limits of components through experience, historical data from experiments and calculations. In addition, OEM provides manuals or prescriptions for each component.

Components that are necessary for many machine-building companies to operate smoothly, such as bearings, motors, and gears, are purchased from an OEM company. The prescribed failures and failure conditions can differ for specific machines due to different environments or usage. Therefore, Tracking and documenting the parameters of its specific unit on the machine makes the limits of its specific units even more tightly related to failures.

Condition based monitoring

Condition-based monitoring (CBM) is based on the condition of components. Components are constantly observed by monitoring the actual health. CBM is the opportunity to keep track of the condition of a component. Customers can proactively respond and schedule maintenance by reaching a specific condition limit to replace the soon-to-fail component. Diagnosis and decision-making are based on the condition of the components. Maintenance can be planned even more tightly and specifically, creating an almost maximum use of the monitored component. The condition of a component can be evaluated by, for example, its position, geometry, vibration, and temperature deviations (Moubray, 1999 and W. Wang, 2006). Maintenance must be executed for a component if it exceeds a specific limit. Monitoring of some parameters can be continuously monitored with sensors or discrete by visual inspections of measure assessment which can be less accurate for small details or deviations. Therefore, CBM is usually referred to as implementing sensors to monitor continuously for reaching a specific limit.

2.1.3. Predictive maintenance

Predictive maintenance (PdM) uses a large sample size of multiple data sets and points of component conditions as the input for setting up a prediction model. Instead of responding on one indication as CBM, PdM set up a prognosis to predict when a failure will occur. Calculations for the prediction model show how components' conditions will behave in the future. In addition to a prediction model, prognostic approaches for PdM can be subdivided into four categories: a data-driven model, a physical model, a knowledge-based model and a hybrid model (Wan et al., 2018). The decision-making of PdM is based on real-time diagnosis detection of impending failures and prognosis of future performance.

PdM is considered the most advanced and intensive maintenance policy because it requires large amounts of data collection, data points, an accurate prediction model, sensor devices and a change in company culture. The cost of the machines and the time increases to predict an accurate failure.

2.2. Maintenance practicality

A practical application in the maintenance field is still a considerable challenge in finding a balance between a theoretical approach and its practicality (Ding and Kamaruddin, 2014; Fraser et al., 2015; Ingemarsdotter et al., 2021). In 2019, it was estimated that around 25% used Internet of Things connections with its component. However, some companies have struggled to recoup their investments. Therefore, most companies are conservative in their maintenance plans because of the difficulty of implementing advanced maintenance plans.

The paper of Ding and Kamaruddin, 2014 used a degree of certainty of the available information in the company to determine which maintenance concept is preferable. The degree of certainty of the available information was divided into certainty, risk and uncertainty categories. A certainty category reveals a simple way of determining optimal maintenance policy. However, the requirement of a complete data set is a critical disadvantage. The risk category mainly involves complex algebraic calculations for literature research purposes. Usually, it neglects the application of a practical point of view. The service mechanics from the service department who do not have strong mathematical theoretical background will have difficulties understanding the algebraic terms. They become less confident with the models. Lastly, the uncertainty category shows a higher degree of flexibility and practicability compared with the risk category. Although this kind of uncertainty concept tends to be practically based on logical thinking approaches. Therefore, it is facing difficulties in collecting data. It utilises experience and expert knowledge, which is not always adequately documented. Through the review of Ding and Kamaruddin, 2014, there is still a big gap between the academic and industrial applications of maintenance concepts and their maintenance policy.

The paper of Fraser et al., 2015 reviewed maintenance models to identify empirical evidence. The paper has found 37 maintenance models in which policy and concepts are umbrella words. From the 37 maintenance models, most of which were very similar, six had practical evidence. These models consist of the following maintenance policies or concepts:

- Condition-based Maintenance
- Preventive maintenance
- Reactive maintenance
- Reliability Centered Maintenance
- Total Production Maintenance
- Other models within the previous maintenance policies and concepts.

Other papers reviewed models within a maintenance concept. For example, the paper of de Jonge and Jakobsons, 2018 reviewed different mathematical models for different situations within maintenance. The paper of Patil et al., 2021 reviewed different techniques for solving Multi-criteria decision-making (MCDM). Other papers review maintenance management in which the network or infrastructure in the company is taken into account (Fernández and Márquez, 2014 and Márquez, 2010). Most reviewed models are lacking in practicality. Therefore, huge changes and investments must be made within the company or the connection to the service department becomes missing.

2.3. Maintenance policy comparison

In this section, we compare the most common maintenance policies found in the literature. Before selecting a maintenance policy for this project, we are comparing the maintenance policies. Each maintenance policy exists because each maintenance policies have its own benefits. We will compare these benefits and limits to only the maintenance policies that can be applied to machine-building companies. For comparing the maintenance policies, we used a multi-criteria decision analysis based on the following criteria:

- Data readiness: The selection of a maintenance policy depends on the data readiness of the company. Without the required type of data, some maintenance policies will not be feasible to perform.

Table 2.1: A rank order comparison of different maintenance policies from favourable (1) to least favourable (6). *for companies similar to VPT

Maintenance policy	Data readiness	Investment	Practicality	Maintenance cost*
Reactive maintenance(RM)	1	1	1	6
Scheduled based maintenance (SBM)	2	2	2	5
Hour based maintenance (HBM)	3	3	3	4
Usage based maintenance (UBM)	4	4	4	3
Condition based monitoring (CBM)	5	5	5	2
Predictive maintenance (PdM)	6	6	6	1

- Investment: The maintenance policy selection also depends on the research invested in the components. For this, there must already be some knowledge of which type of data, sensors and systems are required in order to be able to perform a maintenance policy.
- Practicality: Maintenance policies differ in the practicality of performing a maintenance policy. Some require multiple actions to know when maintenance is necessary and others do not need any requirements or difficulty for maintenance.
- Maintenance Cost: Eventually, a maintenance policy pays back in reduced costs and losses over time. The return in costs and losses of each maintenance policy will be different because each maintenance policy estimates maintenance intervals in its own manner.

Each maintenance policy has its benefits, and table 2.1 shows the relative benefits of each maintenance policy. The values are numbered on a scale from one to six. A value of one means a favourable situation, while six is the least favourable situation per category. The red-marked boxes on the table are boxes to avoid. Due to an experience-based maintenance plan, the service department do not have any data ready yet know which components need to be invested in bringing CBM and PdM as feasible options. Conversely, a maintenance plan solely on RM can lead to the highest maintenance costs, which may be the least preferable choice for customers.

2.4. Maintenance concept comparison

In this section, we compare the maintenance concepts found in the literature with the research design. The design of improving a maintenance decision can be developed from a combination of different maintenance concepts. Different aspects within maintenance concepts can be considered that fulfil the research design. A brief description is given in Table 2.2.

Table 2.2: Maintenance concepts and its method

Concepts	Method to determine a maintenance policy
Total Production Maintenance (TPM)	Total employee involvement and improvement of organisation.
Reliability Centered Maintenance (RCM)	Failure identification, detection and prevention.
Three stage funnel-based	Failure frequency and breakdown. High breakdown with low failure frequency is nominated for predictive.
Multi-Criteria Decision Making (MCDM)	Creating criteria with a score per result on the alternatives.
Mathematical and Simulation models	Creating possible future outcomes with algorithm.
Industry 4.0	Starting from reactive with no data to predictive with available technology and many data points.

We examine the following aspects based on research questions and the problem statement that need to be in line with the concepts from the literature:

1. First of all, the service department needs to know which data needs to be collected. They need data to determine a suitable maintenance policy. Data ensures that experience or knowledge will be less relevant. Operators, service mechanics and the service department can understand why a decision is made based on data.

2. Secondly, the service department needs a guideline for how these data points can be collected. Some literature papers or concepts describe only an approach to a result but not how these data points are collected. This needs to be explained in detail to be sure these data points are feasible to collect.

3. The maintenance concept provides the process of the collected data to improve maintenance decisions. The collected data should link with the maintenance decision. The service department should know which maintenance plan is suitable. The maintenance concept results in a maintenance policy that benefits both the service department and customers.

4. A practical guideline is necessary for the whole process of improving a maintenance decision for the service department. A guideline provides the necessities for the aspect mentioned above. Usually, literature studies assume that the essentials are already given. However, to become practical, we must provide all the tasks for the service department. Eventually, they are the ones that decide when and which maintenance plan will be executed.

5. The framework needs to improve continuously. The framework starts with high uncertainty due to low sample sizes. However, the framework gains knowledge from maintenance and consistently provides better results. Eventually, the maintenance cycle should stabilise with quality data sample size, leading to more accurate results.

We will assess these concepts to reach if they solve one of these aspects. The number point of aspects corresponds to the numbers within the maintenance concepts.

- TPM

    1. TPM does not explicitly mention the data sets. The data set differs from each application in which a general data set is not given. In addition, this concept focuses the organisation's involvement on the responsibility of explicitly providing the data points.
    2. The tools and techniques, such as collecting data, have not been appropriately explained (Mishra et al., 2021). Most of the data collection has already been given.
    3. From the data collection which was already given, they used different methods to come up with the implementation of a new design for each specific component. The solution aims at the specific new design for weak components.
    4. TPM has proposed a 12-step implementation methodology. It is total employee involvement. It is not only the responsibility of the service department and service mechanics. Huge company cultural changes have to be accomplished for TPM, which is a time-consuming process to implement TPM. This makes it less practical.
    5. TPM provides continuous improvement to increase overall equipment effectiveness.

- RCM

    1. The data sets that need to be collected are mostly given from a worksheet. Usually, it is in the form of occurrence, severity, failure mode and failure visibility.
    2. Data points as occurrence, failure modes, severity and criticality must be collected, and a value of the Risk Priority number is the outcome of the guideline. However, the collection of these data sets is subjective and based on the perception of the failure of the components.
    3. The tools for using data points come from an FMEA or similar analysis. The worksheet is filled without a measurable quantitative result. From the analysis, the seven questions can be answered subjectively and a maintenance policy is chosen.

4. A guideline for RCM consists of evaluating the components based on the SAE JA1011 standard. The result is the choice of maintenance policy. The service department has to answer the seven main questions of RCM as input for their outcome. In addition, the service mechanics write down the tasks for the different employees to execute.
5. RCM provides continuous improvement from the feedback of the maintenance execution.

- Three-stage funnel-based approach

1. The data points are the failure occurrence and the severity of each component.
2. The tool for collecting data is not entirely clear. The data parameters like failure frequency and severity are not discussed on how they should be collected. Thus, these data points were already given.
3. The maintenance policy is determined by the intensity of failure occurrence and severity of each component. The intensity of both is plotted in a graph with four quadrants.
   (a) upper right: Redesign component.
   (b) upper left: Fix failures with spare parts;
   (c) lower left: Regular SBM;
   (d) lower right: Potential PdM/CBM.
4. A guideline is explained in detail for determining a PdM and CBM. The funnel is to filter out components. Eventually, the components that are left are the possible PdM/CBM candidates. The explanation of other maintenance policies was missing.
5. Three-stage funnel-based does not necessarily have a continuous improvement. This concept is mainly the reduction and identification of the components for PdM. Although, the three-stage funnel-based can be reused every time with more data sets.

- MCDM

1. The data points mainly consist of subjective criteria in the form of a score number. MCDM is a black box with a maintenance policy as a result.
2. The data collection tool is a combination of subjective criteria and given data about the failures of the components.
3. The data points are used for one of the MCDM methods. It results in one of the maintenance policies.
4. A guideline for different techniques in the MCDM is explained for determining a maintenance policy. However, the tasks to perform the determined maintenance policy is missing.
5. MCDM is not a continuous improvement. However, the MCDM can be fine-tuned for each score per criteria from which different results can be obtained. This means that results can be biased.

- Mathematical/simulation model

1. Usually, the data points are given. However, each mathematical or simulation model describes its data points.
2. The tool for collecting data is not always given. Usually, this data is already known. This makes it challenging to find out how data was obtained.
3. The exact PM, CBM and PdM execution are based on analytical calculations or simulations. There are different mathematical models of when maintenance should be performed for different Key Performance Indicators (KPIs). Usually, the models minimise in the form of costs or maximise reliability or availability.
4. A guideline for mathematical models and simulations is given. Generally, in the models and simulations, a black box is made, and the input of the black box is the responsibility of the service department. However, the models mainly show results. However, the practical side for the service department is mostly missing.
5. Mathematical models can be continuously improved when the data becomes complete or more reliable.

- Maintenance/Industry 4.0

  1. The road to maintenance/industry 4.0 is a global concept in which different paths can achieve it. So the data points that must be collected is missing.
  2. This concept does not discuss what data should be collected. Therefore, it is unknown how the data should be collected.
  3. The maintenance policy is given. However, the maintenance policy is not further elaborated.
  4. A general guideline is explained for the "upgrade" to an advanced maintenance policy. Industry 4.0 is a generalised concept. The inputs and outputs for the service department and service mechanics are not explained in detail because reaching the maintenance policies can be taken with different paths.
  5. It is a general improvement with the shift from RM to PdM.

A summary of maintenance concepts is listed in Table 2.3. For each concept, it is shown whether the aspects are incorporated with the concepts. If the box is marked green, the corresponding concept aligns with the aspect. The table shows that each concept has some aspects that are incorporated into the concept. However, none fits the goal of this project perfectly. Concepts such as Three stage funnel-based, MCDM and mathematical models usually have a specific purpose while neglecting one of the aspects. Others, such as TPM and RCM have a comprehensive package that allow for various goals to be achieved. However, at the cost of time, major business changes and objectivities have to be implemented in the company.

Table 2.3: Comparison of different maintenance concepts from the literature to the aspects

Maintenance concepts	Data points	Data collection	Data execution	Guidelines	Improve continuously	Concerns
Total Production Maintenance (TPM) (Nakajima and Bodek, 1988)				yes	yes	Huge cultural change in the company
Reliability Centered Maintenance (RCM) (Moubray, 1999)	yes		yes	yes	yes	Subjective and relative time-consuming
Three stage funnel-based (Tiddens, 2018)	yes		yes			Focus on PdM
Multi Criteria Decision Making (MCDM) (M. Shafiee, 2015)	yes	yes	yes			Decisions are based on subjectivity
Mathematical/simulation models (Ding and Kamaruddin, 2014)	yes		yes		yes	Can vary to complex mathematical model
From Industry 1.0 to Industry 4.0 (Poór et al., 2019)					yes	Global concept

2.5. Outcome literature review

In this chapter, a literature review of maintenance policies, concepts and other maintenance-related topics has been conducted in order to answer the first sub-question. Maintenance is a broad terminology which involves much more than just maintenance policies and concepts. Many different models and techniques have been developed in the literature for different situations and purposes. Figure 2.1 shows the maintenance policies mentioned in this chapter and a comparison is shown in Table 2.1.

For this project, we mainly focus on the preventive maintenance of SBM, HBM and UBM. The investment and the data readiness are relatively low, while it is still practical. The data collected based on the mentioned maintenance policy will also be the first step towards a data-oriented company.

In addition, we have also compared maintenance concepts with the aspects of our research design. The most well-known concepts were explained in this project. The concept of maintenance policy selection is determined in different ways. It was from a quantitative approach, such as mathematical models, to a qualitative approach, such as RCM. Also, the concepts could be short-termed such as three-stage funnel-based or a long-term approach, such as TPM or maintenance/industry 4.0. Table 2.2 shows a summary of the maintenance concepts with their determination method.

Considering maintenance/industry 1.0 to 4.0, the current maintenance plan of VPT belongs in the RM and SBM. Both are indeed primitive maintenance policies in which no data points have to be collected. Use-based maintenance is the next maintenance policy towards maintenance 4.0, which starts collecting data. Use-based uses time as data points. Time is a clear data point that is relatively easy to record, whereas CBM or PdM needs an additional measuring system to gather more complex data points continuously.

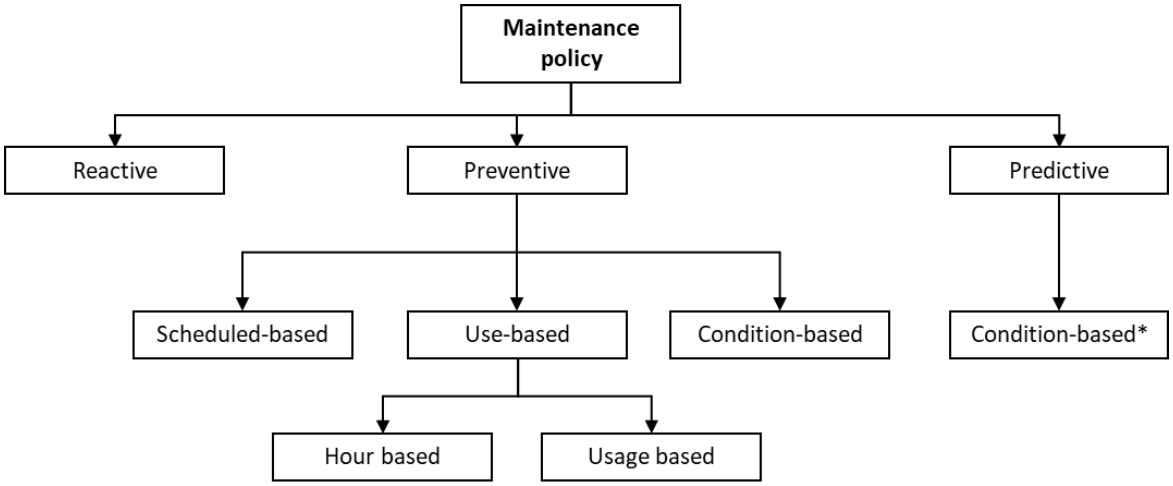


Figure 2.1: All maintenance policies based on different literature reviews. *Additional systems or tools are required.

Altogether, each concept has some valuable tool and technique to solve one of the research sub-questions 1.1. However, none provides the whole package. With this knowledge in our minds, we move on to the development of a maintenance concept that are in line with the research design which improves the mainenance decision.

3

Methodology

In the remaining chapters, we will elaborate on the methodology for this project. Recall from Chapters 1.1 and 2 that the project includes practicality for the service department and the second research sub-question will be answered: **What are the Key Performers Indicators to validate the usefulness of the proposed methodology?**

From the literature study, maintenance is a versatile terminology. That is why assumptions have been made to focus on the purpose of this project. The following assumptions are used:

- Maintenance from the service department is only executed for possible replacements of components. Maintenance to clean, lubricate components or make the filters dust free is the customers' responsibility. Therefore, it is out of scope. When manuals describe that customers should clean daily or weekly, it is assumed that they follow the manuals to execute minor maintenance.
- We assume that there are two kinds of service mechanics. Service technicians on-site are employees from the customer company and service mechanics off-site are employees from the machine builders company. We mainly focus on the off-site service mechanics due to not knowing the customer's decisions.
- Maintenance is executed by VPT service department due to components or errors that the customer itself cannot fix. On-site service technicians can fix easy-to-replace or repair components. They can buy spare parts from the machine builders company to execute maintenance by themselves. Therefore, they are responsible for which maintenance policy they apply or when maintenance is executed.
- This project was set up on the side of the machine builders. The project will only provide choices and reasons for the machine builders. Each customer can differ in behaviour and wishes towards the machines, such as the number of spare parts and components the customers are maintaining. The choice of a customer made in this project is an assumption.
- During maintenance, the failed or soon-to-fail component will be only replaced by a new identical component. Therefore, we assume that the replaced component is as good as new one.
- This project does not focus on the techniques of data collection in detail. As long as the data is stored in a structured way, each data collection method should be acceptable. However, it is advisable to digitise the data collection since it is convenient and requires a low motivation threshold for the service department. However, some investment time and cost should be taken into account. Some data collection methods could be:
  - Pen and papers;
  - Excel;
  - Computerised maintenance management system (CMMS);
  - Other monitoring, collection systems or databases.

To determine a suitable maintenance plan for the service department, we need a framework that supports VPT and gives them a structure to follow. The definition of a general framework is "a structure intended to serve as a support or guide for the building of something that expands the structure into something useful" (Lutkevich, 2020). A maintenance framework is a structured way of collecting and using the data to provide knowledge for determining a maintenance plan.

The methodology of the proposed maintenance framework utilises an iterative cycle of planning, doing, checking and adjusting (PDCA approach) (Waeyenbergh and Pintelon, 2009; de Jonge and Jakobsons, 2018 and Nakajima and Bodek, 1988). The PDCA cycle in the next Chapter can include all the aspects of the research design and it is simultaneously structured in a flow of planning, doing, checking and adjusting. It is a proper flow of preparing data, using the data to schedule maintenance time intervals and checking the maintenance results to obtain knowledge for the next suitable maintenance plans.

From the planning phase, the service department has to bring up all the necessary inputs for preventive maintenance. After identifying the data, the maintenance framework uses the data to schedule maintenance time intervals in the doing phase. This is scheduled maintenance is followed and the service mechanics will check the actual maintenance with the maintenance plan. New data points will be collected to give new insights about components' failure behaviour and support maintenance plan decisions for the following cycles. Figure 3.1 shows a visual methodology of the PDCA cycle, which starts from the upper left corner.

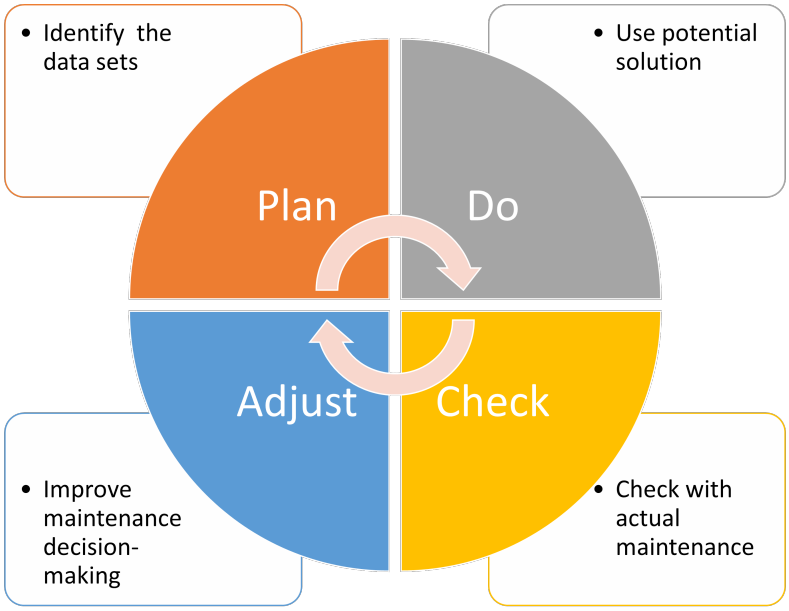


Figure 3.1: The methodology of the PDCA cycle.

The PDCA approach is a structured cycle with a workflow that the service department can follow. The PDCA cycle is a method which it can be implementable even when data are scarce. Scarce data has high uncertainty in which PDCA is an appropriate method to improve the maintenance decision-making along the PDCA cycles (de Jonge and Jakobsons, 2018).

Each phase of the PDCA cycle can be divided into multiple modules because it sets small and understandable steps. Each module of the PDCA cycle can be based on one of the tools or techniques in a maintenance concept. Therefore, the proposed maintenance framework combines different techniques from the maintenance concepts to improve maintenance decisions with the flow of the PDCA cycle.

For the Key Performance Indicator(KPI), the doing and the adjusting phases are the phases of finding the usefulness of the proposed maintenance framework. For the doing phase, we must find the optimal maintenance interval based on an indicator from preventive maintenance. If the proposed maintenance framework can estimate an accurate maintenance interval, it shows that the proposed maintenance framework performs well. Therefore, the accuracy of the estimates of the maintenance interval is one of the KPIs. Secondly, the adjusting phase is the phase of adjusting the failure behaviour of components value that represents the correct failure behaviour of components. If the proposed maintenance framework can estimate the failure behaviour of components accurately, it will also show that the proposed maintenance framework performs well. Thus the two KPIs are the accuracy of estimating maintenance intervals and the failure behaviour of components.

4

Proposed maintenance framework

Now that we have given an overview of maintenance policies and concepts and explained the methodology, we are able to introduce the proposed maintenance framework formally. In this chapter, we will explain the steps concerning the PDCA cycle.

This chapter continues to answer the third research sub-question: **What maintenance strategy is proposed?** This will be answered as well as the corresponding sub-questions: **What are the available data?**, **What data need to be collected during maintenance?**, **How can the maintenance schedule be optimised?** and **How will this maintenance flow determine a maintenance plan?**

4.1. Module one: component selection

The first phase of the PDCA cycle is to identify the data sets. We must determine which machine, its systems, and its components we need to gather data sets. Machines are becoming complex. They consist of hundreds or even thousands of individual types of components. Collecting data and paying attention to all the components in detail during maintenance is time-consuming and costly. Service mechanics would lose motivation and the customers would not like the idea of going through each machine component due to high maintenance costs. Therefore, these time and cost restrictions lead us to the first module that identifies the components which the service department has to select for the maintenance framework.

The selection of the components consists of two parts: It is determined based on failure mechanisms or experts' experience. We will start with components and their failure mechanism and then to components based on experts' experience afterwards.

Every material, component and system in machines will fail after a certain amount of time. Each material has its rate of deterioration, and many factors affect this rate. As a result, each component in a machine has a unique failure time due to its characteristics, environmental effects, and loads applied to the components. Sachs and Neville, 2007 and Tinga, 2013 have established the possible failure mechanism. The failure mechanisms have been added to this project in Appendix B. Figure B.5 shows an overview of the common failure mechanisms of a component. The failure mechanisms branch even further to the possible failure causes of the component.

4.1.1. Component selection based on failure mechanism

Each component always suffers from material degradation. One of the reasons for material degradation is ageing in any environment. Other possible failure mechanisms can occur for all components. These could be, for example, corrosion, human errors and overload. It is a combination of systematic, random, or sporadic mistakes in which these failures may appear. Both ageing, corrosion, human errors and overload can be hard to notice at the beginning because they cannot be directly observed during the machine's operation. Usually, it is observable when damage to components becomes obvious and even then, it is often a combination of several failure mechanisms.

On the other hand, fatigue, wear and tear, or electrical failures occur only to specific components. These components can be observed directly because these particular components are moving in cycles with varying applied loads, have relative motion between a contacting surface or are supplied with electricity. Therefore, these specific components suffer, besides from ageing also from wear and tear, fatigue or electrical failure.

Furthermore, when looking at machine-building companies, their machines are typically used for continuous process for many hours in sequence. The components in the machines move mainly in high accelerations and slowdowns repeatedly. The products in the machines move in the machines to be processed by a robot arm, gripper or any other processing tool that needs to accelerate to the product and stop instantly at the product back and forth. Therefore, we must focus on components with these distinct failure mechanisms. These failure mechanisms can be identified by answering the following questions:

- Does the component drive other components? (related to electrical components)
- Does the component have relative motion between the contacting surface and itself? (related to wear and tear components)
- Does the component use varying load? (related to fatigue components)

These questions could be answered by observing the machine during operation. A yes to one of the questions means that the component should be selected for the framework.

4.1.2. Component selection based on experts' experience

The second method is based on experts' experiences and knowledge related to the machine. Experts include the operators and customers who observe the machine during operation and the engineers and service mechanics responsible for its design and maintenance. They experience failure of components through the following methods (M. Shafiee, 2015):

- Experiments/clinical trials;
- Observing and recording the events(e.g., counting the number of component failures at specified times);
- Obtaining relevant data sheets, formulas, Original Equipment Manufacturer(OEM) knowledge;
- Administering surveys with closed-ended questions(e.g., face-to-face, interviews, questionnaires, brainstorming).

The components that the experts recommend must also be selected in this framework. They experience which components are critical, i.e., frequently fails, high breakdown time, or any other undesirable factors. Usually, experts do not necessarily have lists of critical components. They mention a couple of components that need attention. Figure 4.1 shows the framework of module one.

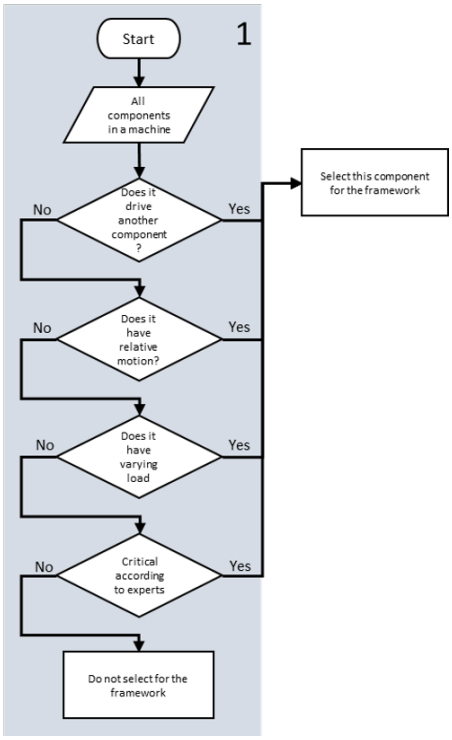


Figure 4.1: Framework of module one

4.2. Module two: gather available data sets

After selecting the components from module one, we searched for available data sets. Since data have yet to be collected during maintenance, the service department has to gather data sets somewhere else initially.

During the design phase of machines, the choice of components is determined based on what the component must be able to withstand in terms of force, pressure and size. In other words, the components are chosen based on the design engineers' requirements to become suitable components for the machine. Usually, these components are purchased from an OEM or a supplier. However, there may also be custom-made components. We start explaining the OEM-purchased components and then the custom-made components.

4.2.1. OEM purchased components

Components purchased from an OEM or a supplier should have data available related to their failure. They should have done the required amount of testing and can therefore estimate the general component's service life and know several failure causes. Thus, the service department can request the service life, failure causes and possible other failure-related data in the form of data sheets. These given service life and failure causes are the initial data sets to determine an appropriate maintenance plan.

Another way of gathering data is by calculating the service life. Some formulas for service life are widely available on the internet or in books. For example, the service life for rotational bearings is given by SKF (SKF, n.d.).

4.2.2. Custom-made components

It is challenging to determine when and which custom-made components will fail in new machines without data from the OEM. Data on custom-made components may be unknown because engineers fabricate them themselves without testing them for a long period. These are often materials in which the machine builder processes the material into a component. Engineers make calculations in order to determine the type of material, the thickness and the geometry to fulfil its function. These specifications are applied to the material to become a component for a machine, but the service life remains unknown due to specific or unique geometry. Therefore, data is simply not available. In the next module, the component for which no data is available will be inspected during each maintenance, and the data will be collected during failure or bad condition. The component will be replaced at its failure or when it behaves noticeably strangely.

Accelerated life test (ALT) has been mentioned in the literature to collect data (Kobbacy and Murthy, 2008 and Jardine and Tsang, 2013). Hundreds of machines were tested at their highest capacity in order to obtain failure data more quickly. However, the ALT application mainly applies to mass-produced machines in which thousands of them are made and sold quickly. Testing a few hundred compared to the production of thousands of machines is a small amount of the total machines. However, when the company sells a few dozen machines, it is inevitable to test a large number of machines. Every machine built is valuable for them. The selling price of a machine skyrockets by applying ALT because the lost machines have to be earned back. Therefore, ALT is an unattractive method and, in practice, is not the method to obtain failure data. The most apparent method is to let the customers use the machines until they fail or become soon-to-fail components, in which it is observable that a component deteriorated significantly.

Obviously, when resources are available from, for example, historical data, experience or any other resources that can estimate the service life and failure causes of custom-made components, the service department can use these values as an initial start. With absolutely no data available, these components have to start initially and the service department obtains knowledge of the failure behaviour during its operation and visual inspections through maintenance.

4.2.3. Fitting a suitable probability distribution

As mentioned before, the service life refers to the B10 life. B10 life is a metric for estimating unit (usually time) to failure and a statistical value for life expectancy (isixsigma, n.d. and Abernethy, 2006). Specifically, B10 life is the quantitative unit in which 10% of the same type of component is expected to fail. Usually, the B10 life is specific to a mechanical load lower than the maximum allowable load.

B10 life is the time in a probability distribution when the component has 90 % reliability. So there is a 10% risk that the component can fail before reaching the service life. Depending on the OEM company, service life can differ from the B10 value. For this, checking the meaning of the service life at the OEM is essential. For example, sometimes service life is given in B1, which has 99 % reliability on the service life. To know the entire degradation of the reliability of each component over time, we must use a suitable probability distribution. Keep in mind that the service life requested from OEM is not always given in time. Sometimes, it could be in a more specific unit from usage-based maintenance, such as travelled distance or amount of rotations. Reaching this specific unit depends mainly on the productivity of the machine and can then be translated in time. Productivity data will be explained in the next section.

A typical distribution to model failure time is the Weibull distribution. It has been found that the Weibull distribution provides a good description for many types of components of failure time data (de Jonge and Jakobsons, 2018). The Weibull distribution is a continuous probability distribution that is versatile. Many other distributions are included in the Weibull distribution, either exactly or approximately. This includes the normal, exponential, Rayleigh and Poisson distributions. Unlike the normal distribution, which has a peak on the average and is symmetric on both halves, Weibull can also model left- and right-skewed data depending on the shape parameter. Components with failure types such as fatigue and wear and tear have an increased probability of failure over time, causing the failure data to have a skewed distribution (Albrice, 2013). The reliability of Weibull is calculated with formula 4.1 (Abernethy, 2006):

$$R(t) = e^{-\left(\frac{t}{\eta}\right)^{\beta}} \tag{4.1}$$

Where

- t = unit of service life(usually in time);
- η = scale parameter of Weibull distribution;
- β = shape parameter of Weibull distribution;
- $R(t)$ = Reliability

Studies on maintenance typically assume that the shape and scale parameters are known due to their actual data collection. However, sufficient failure data is required to accurately estimate shape and scale parameters and their probability distribution. This poses a problem as limited failure events are available in practice since they start from experience-based maintenance. Low-quality data acquired implies that these assumed probability distributions are dubious or even unavailable. A limited sample size of the data points means that

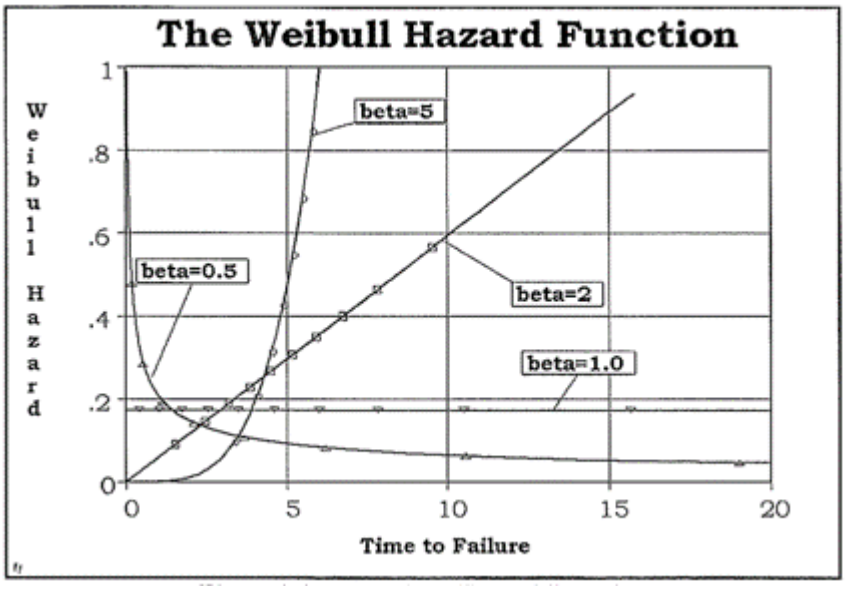


Figure 4.2: Weibull hazard function(Abernethy, 2006)

these data points cannot achieve accurate estimates for maintenance. A commonly used approach by most reliability engineers is to assume a particular distribution and evaluate the parameters by expert judgment. Other methods to determine parameter uncertainties are techniques such as Bayesian, Bootstrap technique, maximum likelihood and median rank progression(Abernethy, 2006; de Jonge and Jakobsons, 2018). Some of these techniques will be explained after the check phase of the PDCA cycle.

Weibull experts generally agree that the shape parameter's value (β) tends to be reasonably constant for specific or generic failure types (de Jonge and Jakobsons, 2018). However, it is much more complicated to estimate the value of the scale parameter. Examples of failure types for which the shape parameter value is predictable are given by (Abernethy, 2006; GE digital solutions, n.d.; P.E., 2021). During each maintenance cycle, the shape parameter will change iterative according to the failure and deviations from the actual failure data. To understand the parameters of the Weibull distribution, we have to know the effect of different shape parameters (β) and scale parameters (η) values. Figure 4.2 shows the effect of the shape parameter value on the hazard function. The hazard rate function is the probability density function (PDF) ratio to the reliability function. An increased hazard rate function means more components are failing with respect to the probability of the failure.

The shape parameter values can be divided into four different categories:

- $\beta < 1$
  - The expected failure has a decreased failure rate with time. Therefore, the reliability increases over time. Usually, both electronic and mechanical components can initially have high failure rates. The manufacturers test the machine before delivery to the customer. Thus, components with a β lower than one are unlikely to occur when the machine is at the customer.
- $\beta = 1$
  - The expected failure cause occurrences are independent of time. An old component is as good as a new component. Usually, this failure cause is due to human, overload or maintenance errors or failures by nature.
- $1 < \beta < 4$
  - This category has many mechanical failure types, and β is predictable for generic failure modes. Great examples that belong in this category are wear and tear components. A relative motion between contacting surfaces removes particles of the component over time, in which slack becomes significant, and the chance of failure increases over time. Therefore, A β greater than 1 implies increased deterioration over time.

- $\beta > 4$
  - β higher than 4 has a negligible probability of failure before replacement. Therefore high β have a safe period before failure within which the probability of failure is negligible. β to infinity implies perfect design, quality data and production.

The shape parameter describes the scatteredness of failure over time. A high shape parameter implies perfect design because the deviation of failures is low, such that each failure is approximately at the same time, which becomes predictable. Material degradation, such as ageing, is the common failure mechanism for which an increased probability of failure over time is valid (Moubray, 1999; Nowlan and Heap, 1978). Other failure mechanisms in the proposed maintenance framework, such as wear and tear and fatigue, have an increased probability of failure since they remove particles and initiate cracks on the component over time, respectively. Lower shape parameters make it a challenge to estimate the scale parameter properly. This is due to the scattering of the failure. Scattered failures ensure that a failure can fail extremely early and extremely late.

The second Weibull parameter is the scale parameter (η). It is also called the characteristic life and typically represents the time to failure in the Weibull analysis. It is defined as the time at which 63.2 % of the components have failed (Galar and Kumar, 2017 and Pasha et al., 2006). Increasing the scale parameter value will stretch out the probability density function(PDF) while holding the shape parameter and the service life constant. However, the peak of the PDF curve will be lower.

For this framework, we have to find the parameters based solely on the given data sheets of OEM and some expert knowledge. The proposal of the initial parameters is to set the shape parameter (β) on a high value. This means that the failure is less scattered over time which is better for predicting failure. Since it is better to predict failure, the components will be estimated as late as possible at the time of failure and the service department will rely entirely on the given service life of the OEM. It is possible to translate the scale parameter from the service life to the scale parameter using the reliability formula(eq: 4.2).

$$\eta = \left(\frac{t_{service}}{-\ln\left(R(t_{service})\right)^{\frac{1}{\beta}}} \right) \tag{4.2}$$

Where

- $t_{service}$ = the service life
- $R(t_{service})$ = reliability of the service life which is 0.9
- η = scale parameter
- β = shape parameter

For visualisation of the effect of the Weibull parameters, figure 4.3 shows components with different shape parameters and a constant scale parameter. The graph shows 100 failure times of the characteristic shape parameter. The components with high shape parameters fail close to the time of the scale parameter which has relatively uniform failure intervals, while low shape parameters are more scattered in time and have relatively random failures. Especially a shape parameter lower than one often fails early. Figure 4.4 shows the different scale parameters while the shape parameter is constant. Increasing the scale parameter scales up the failure time to the scale parameter. In other words, the failures stretch towards the scale parameter.

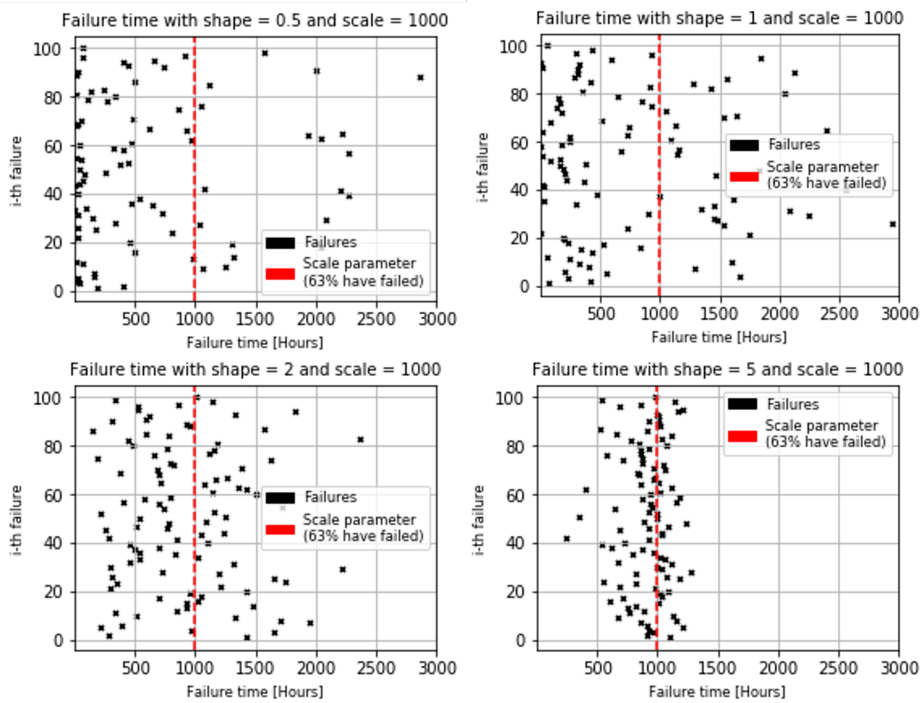


Figure 4.3: Failure behaviour of different shape parameter

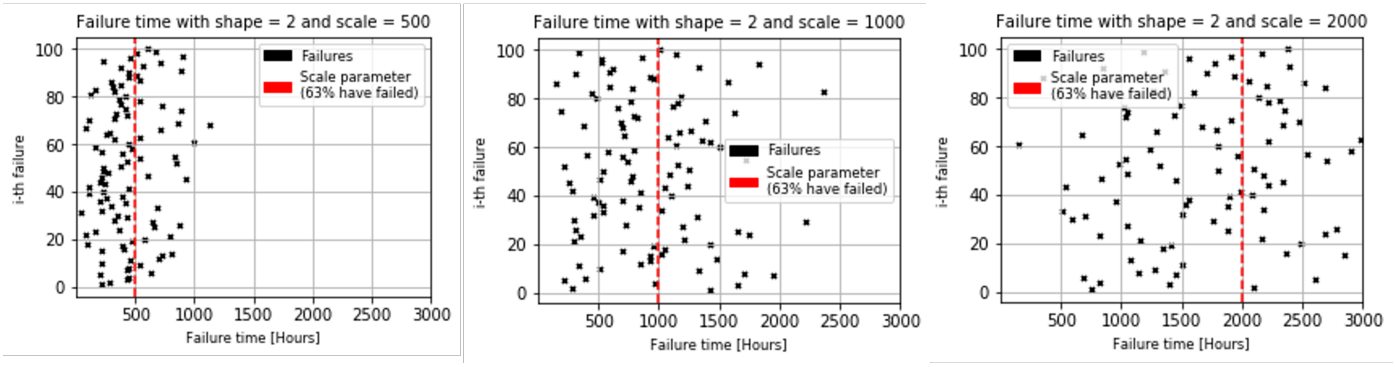


Figure 4.4: Failure behaviour of different scale parameter

Therefore, components with available service life have a corresponding Weibull distribution. We have found the data sets which are available and are derived from the OEM. These data sets include:

- Service life (always be sure it refers to the B10 value);
- Failure causes;
- Shape parameter of Weibull distribution;
- Scale parameter of Weibull distribution.

A decision tree for module two is shown in Figure 4.5.

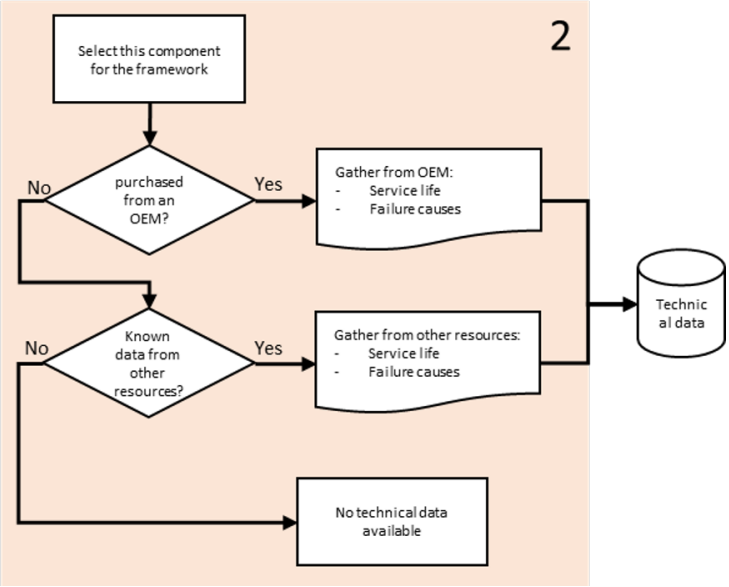


Figure 4.5: Framework of module Two

4.3. Module three: gather productivity data set

After gathering the available data set of each component, the service department needs additional data from the customer: the productivity data. This is due to the fact that component degradation depends highly on the productivity of the machine. Heavy use of the machine affects the components differently than low use (Tinga et al., 2020). The variation in productivity is partly taken into account. The effectiveness achieved with this method depends largely on the selected productivity parameter's relevance. The number of travelled distances would be more relevant than calendar time unless machines are not operating and ageing leads to failure.

Instead of assigning the service mechanics to collect all specific data for each component, we derive these specific data from a general counter of the whole machine. It saves time for the service mechanics and they can do their job of executing maintenance instead of collecting data. The most basic productivity data to start with is the running time. It is only the time that the machine is operating. Different customers have different operating times, leading to different running hours for the same year.

The second productivity data is the amount of production or average capacity. Usually, each component has its specific amount of usage to produce one or a group of products. Suppose a machine produces one product. A bearing has to rotate 4.7 times. The total amount of rotation can be established by multiplying the total amount of production. Basically, all components can be converted into the service life of (running) hours by linking the amount of production and the running hours,

In addition to the productivity data specific to each customer, the costs are also personalised. Machines can be used for different purposes, resulting in different production losses. Typically, high-quality products lose more than low-quality products in which customers with high-quality products are most likely risk averse because of high losses during downtime.

A second specific cost is the customer location. Depending on the location of the customers, it contains higher setup costs. The setup cost is the fixed cost included before maintenance is executed. It includes the travel expense the service mechanics must take.

Customers use their machines using their specific running time, the amount of production within the running time and costs. Thus, improving the maintenance decision is based both on the knowledge of components (modules one and two) and awareness of each customer's productivity. The following data sets will be collected from module three:

- Running hours;
- Amount of production;
- Customer specific costs:
  - Setup costs;
  - Production losses.

Module three is a short workflow during operation. The workflow is shown in Figure 4.6

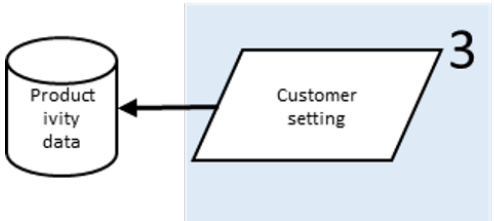


Figure 4.6: Framework of module three

4.4. Module four: the Maintenance optimisation model

Module four is the doing phase of the PDCA cycle. It concerns the analysis of the data gathered from modules one to three. When many data points are collected from multiple maintenance cycles, module four becomes more valuable in finding a suitable maintenance interval or knowing there is no suitable maintenance interval. In chapter 3, it was mentioned that an improved maintenance plan can either improve the SBM to

minimalise the amount of maintenance and still maintain before actual failure or consider an alternative maintenance plan. In this doing phase, we are trying to improve the SBM. Since the service department has gathered quantities such as service life, productivity and costs from previous modules, we were searching for a model, technique or tool to improve maintenance intervals. A general maintenance model is required, which applies to different components.

In order to find an appropriate mathematical model, researchers like de Jonge and Scarf, 2020; Ding and Kamaruddin, 2014; Zhu, 2015 have written extensive reviews in their papers on maintenance optimisation, both per component and as a whole. In this section, the choices are made for this project. Figure 4.7 shows the fourth module of the maintenance framework.

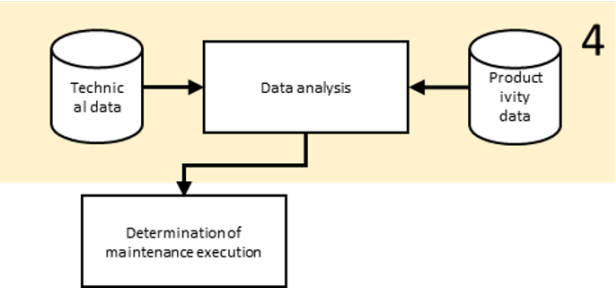


Figure 4.7: Framework of module four

The first category to make is that of the deterioration state process. A two-state deterioration process consists of a good and failed state, and a three-state deterioration process consists of a good, defective and failed state. In the actual situation, most selected components have a defective state in which the component can still operate but with less accuracy or a lower capacity. So, the defective state exists and is observable by slack, noise or volume degradation. Nevertheless, for this project, we consider a two-state deterioration process. As mentioned before, the conditions of components are not continuously monitored or measured because of investment costs of sensors and monitoring systems or practical issues. The component will be replaced if any observable defective state occurs. For example, if slack was noticeable during a visual inspection. The service mechanic will not measure the value of slack. The component will be replaced regardless of the defective state because it has already become obvious. Thus, a third-state deterioration process will not be used because there is hardly any data available on the defective state of components during maintenance and visual inspections. Furthermore, the OEM does not define any defective limits before components are replaced and do not measure the defective values. Without these data, it cannot be determined when the defective state will last before it fails. In addition, usually, customers prefer PM over RM due to the high downtime costs. Therefore, the component will be replaced at an observable defective state without considering the absolute value.

Lastly, components in a defective state could affect the surrounding or connected components, which leads to a higher deterioration rate of surrounding components. For this reason, the components should not remain in the defective state for too long. Therefore, a two-state deterioration process model is chosen in which its failure and noticeable behaviour difference are defined as a failure state. The proposed deterioration state process is shown in Figure 4.8 where the vertical area left of the vertical red line is defined as a good state and the area right of the vertical line is considered a failed state.

For returning a component to a good state, we assume that components are non-repairable. The selected components have mainly a failure mechanism that is non-repairable. Wear and tear is an example of a phenomenon where small particles become detached from the components. Usually, these tiny particles cannot be attached back to the components by repair. Therefore, the old component must be replaced by a new identical component. In addition, this project aims to stay as close as possible to what the company of Viscon Group performs. They replace almost any failed component for the sake of quality service to customers. This means that the models should expect replaced components that are as good as new. The cycle of deterioration can then start over again.

For dependencies, economic dependence is virtually always present. Clustering multiple components in one maintenance execution will have a positive business case due to a single setup cost. Service mechanics must prepare and travel only once to replace multiple components. Researcher as Zhu, 2015 considers economic dependence to be the most valuable to consider in the mathematical model.

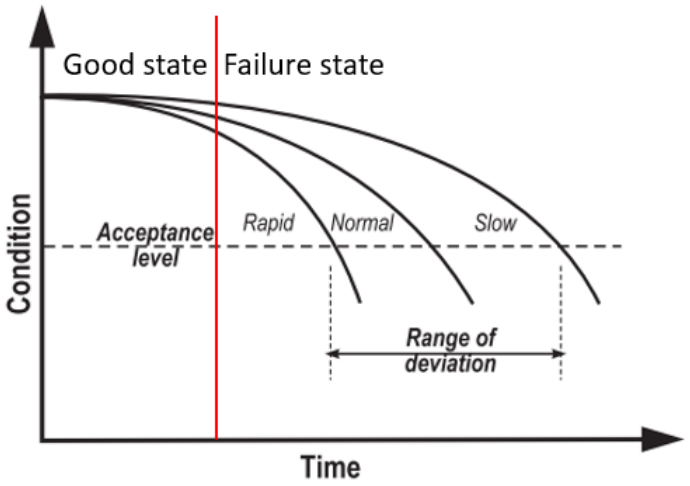


Figure 4.8: Proposed deterioration state process of different deterioration rate (Narayan, 2005).

Thus, we focus on models that use a two-state deterioration process, assume component replacement and consider economic dependency. Module four aims to find the optimal time for maintenance execution considering these three factors. For each component, it is possible to determine its optimal time. After finding these, we consider a variable that consists of a time interval of no operation. In the time interval when machines are not operating, maintenance costs are lower than when machines are operating. This variable has been added because most transplanting machines in VPT only operate during peak season. Outside the peak season, these machines are not operating at all. For economic dependence, we are considering a maintenance optimisation for multiple components with positive economic benefits and its practicability of having fewer maintenance moments.

4.4.1. Optimisation on component level

After determining the categories, a model can be created. When we are looking for a total replacement model in which only components are replaced during maintenance, the two widely used techniques for component replacement are the age replacement model (ARM) and the Block replacement model (BRM) because of their versatility (Márquez, 2010; Savits, 1988). In the case of an age replacement model, maintenance is executed when a component reaches its optimal time based on reliability **or** when it has failed. In both cases, the maintenance cycle starts over again. Scheduling a large maintenance interval means the reliability becomes low, and the chance of RM increases.

In the case of a Block replacement model, maintenance is executed based on an optimal time interval **with** the failure time of components lower than the optimal time of replacement. The cycle of maintenance execution does not start over again. Late planned maintenance means that the number of expected failures increases, which increases maintenance costs. Both models are visualised in Figure 4.9.

The age replacement model was proven to be globally optimal for a given planned replacement cost, suggesting that the age replacement model is preferable to the Block replacement model (Berg and Epstein, 1978).

However, each component has its age limits, meaning each component should be replaced at a different time. For the age replacement model, the maintenance cycle starts over in which the maintenance interval of a component shifts to an earlier moment which could be an unfavourable maintenance interval, and it ruins grouping multiple components in maintenance. In contrast with the Block replacement model, the maintenance cycle starts over after the calculated maintenance intervals.

Therefore, the Block replacement model is considered over the age replacement model. By grouping components, the service mechanics only have to execute maintenance once. If the optimal maintenance interval is chosen so there is little chance of failure, the components would almost always be replaced at the same interval. While the optimal age of components can differ significantly, a failure causes a component to become out of sync with the other components in the cluster.

Consider a case where the optimal maintenance interval is the same for both the age and Block replace-

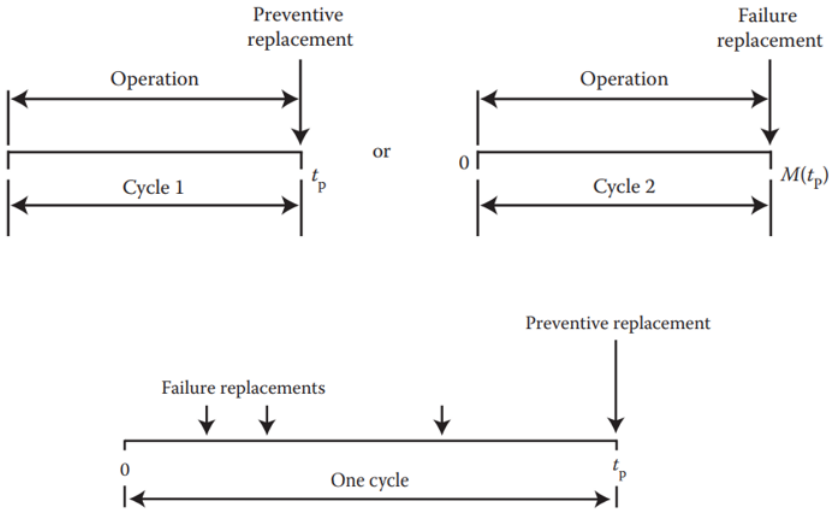


Figure 4.9: up: Age replacement model. Down: Block replacement model

ment models. In the age model, if a component fails and is replaced, the maintenance time starts over from the incident. In the block model, the maintenance time does not start over after the component replacement, resulting in a second replacement when the original time for maintenance is reached.

In module five, we consider the possibility of failure at a different time. In practice, perfectness is almost impossible and a component can fail before SBM.

The Block replacement model shows the cost of the replacement of each component. To reduce the number of failures, a preventive replacement can be scheduled to occur at specific time intervals. However, a balance between failures and the frequency of preventive replacement is required. The Block replacement model is one in which preventive replacements occur at fixed times. The model determines the optimal interval between the preventive replacements to minimise the total expected cost of replacing the component per unit of time. The formula for component costs per unit of time in the Block replacement model is as follows:

$$\text{Component's maintenance costs per time unit} = \frac{\text{Total expected costs}(TEC)}{\text{Length of interval}(LoI)}$$

The Total expected costs (TEC) include the costs of PM and losses during RM with the increased chance of failure over time.

The Length of Interval (LoI) contains the interval of maintenance execution.

The lowest component's maintenance costs per time unit of component i consist of a PM cost and the expected number of RM divided by the maintenance execution time. The formula for the Block replacement model becomes equation 4.3:

$$Z_i(Tm) = \frac{C_{PM} + H(Tm) \cdot C_{RM}}{Tm} \tag{4.3}$$

Where

- $Z_i(Tm)$ = Lowest maintenance cost per time unit of component i;
- C_{PM} = Total cost if PM is applied;
- $H(Tm)$ = The expected number of failures in the interval $(0, Tm)$;
- C_{RM} = Total cost if RM is applied;
- Tm = Maintenance interval.

The preventive and reactive costs can be further divided into different types of costs and losses. The type of costs and the values depend on the company and its machine. The most likely costs in reactive and preventive consist of equations 4.4 and 4.5, respectively (Hoogendoorn, 2020):

$$C_{RM} = C_s + (C_{lost} + C_{mh}) \cdot T_{repair} + C_{comp} + C_{lost} \cdot T_{res} + C_{pen} \tag{4.4}$$

and

$$C_{PM} = C_s + (C_{lost} + C_{mh}) \cdot T_{repair} + C_{comp} \tag{4.5}$$

Where

- C_s = Setup costs
- C_{lost} = Lost of production costs per hour
- C_{mh} = Manhour cost per hour
- T_{repair} = Repair time
- T_{res} = Visit response time
- C_{comp} = Component cost
- C_{pen} = Penalty costs of RM, other components damage and costs for calling customer service.

There are a few approaches to determine the expected number for RM (Jardine and Tsang, 2013). The discrete approach assumes adding the expected number of RM at each time interval with the assumption of H(0) = 0. Therefore, the formula for the expected number of failures is shown in equation 4.6:

$$H(Tm) = \sum_{i=0}^{Tm-1} (1 + H(Tm - i - 1) \cdot \int_{i}^{i+1} f(t)\, dt) \tag{4.6}$$

Where

- $f(t)$ = probability density function;
- H(Tm) = Expected number of RM
- Tm = Time of maintenance

Thus, the Block replacement model depends on two variables. The first variable is the difference between PM and RM over time. Over time, the chance of failure of components varies, with the risk of having RM costs. Usually, maintenance should be executed as soon as possible because PM is cheaper than RM.

The second variable is the time that maintenance is executed. By postponing maintenance, the frequency of maintenance decreases. So the costs per unit of time are decreasing over time. Combining these two variables result in a balance in which the cost per time unit is the lowest somewhere in between, which is the best time to execute maintenance.

Furthermore, components with a low shape parameter ($\beta < 2$) have scattered failure times. Due to much spread of failure time, the Block replacement model will find an optimal maintenance execution time in which the expected failure is relatively high. RM occurs more often. Papers of Tam et al., 2006; H. . Wang, 2002 pointed out that for a Maintenance optimisation model (MOM), minimising cost did not necessarily imply maximising reliability and to achieve the most suitable maintenance execution time, it needs to consider both cost and reliability. This insight is eventually up to the service department and customers' behaviour. Some customers are taking risks and others are not.

Many extensions and modifications of replacement Block replacement models have been developed (Archibald and Dekker, 1996; Arts and Basten, 2018; de Jonge and Scarf, 2020; Li and Zhang, 2015; van Elderen, 2016; Vilarinho et al., 2017; H. Wang et al., 2017). A significant number of these models take uncertainty in the lifetime distribution into account. The effect of adding uncertainty would be interesting because model uncertainty is likely to exist in practice. However, this project considers the standard Block replacement model to apply to different machines.

4.4.2. Consideration of peak seasons

We added a third variable in the Block replacement model considering peak seasons. The third variable is the time in which the machine does not operate. These are periods when no production has to be made, mainly because of no resources to produce. Like crops, there are harvesting seasons and the machine does not have to operate the rest of the time. These are intervals where the costs are even lower than the preventive costs (C_{PM}). This makes outside peak season a favourable moment to execute maintenance because there are no or hardly any production losses. This variable adds the following piecewise function 4.7:

$$C_{PM} = \begin{cases} (C_{lost} + C_{mh}) \cdot T_{repair} + C_{comp} & \text{if } Tm \text{ is during peak season} \\ (\alpha \cdot C_{lost} + C_{mh}) \cdot T_{repair} + C_{comp} & \text{if } Tm \text{ is not during peak season} \end{cases} \tag{4.7}$$

where

- α = reduced factor of lost production outside peak season

Note that the time interval outside peak season is modelled as one moment: at the end of the peak season. We assume that the machine does not deteriorate when the machine does not operate. By applying this assumption, no time intervals need to be created because the reliability remains constant. The assumption can be tested by future research in which components are accurately tested for reduction of reliability during machine downtime or outside the peak season.

4.4.3. The Maintenance optimisation model (MOM)

Besides the addition of peak season in the model, an optimal SBM and economic benefits can be realised by clustering components in maintenance intervals. This minimises costs and also lowers the frequency of maintenance. Before components are clustered, we use the previous Block replacement model without the setup costs to determine the optimal maintenance interval. With the optimal maintenance interval for each component, we use the lowest maintenance interval of a component as the maintenance cycle. All other components with an optimal time smaller than two times this interval are included for the clustering of components. Components with a maintenance time greater than the interval fall into the interval's subsequent cycles. For example, the earliest optimal maintenance of component A is in 1000 hours. Therefore, the interval of component A is at every 1000 hours. All components with an optimal time between 1000 and 2000 are considered to be clustered in the interval of 1000 and 2000 hours. So if component B optimal maintenance interval is in every 1300 hours. The MOM includes component B in the iteration of component A. The consideration depends on the addition of the setup cost. Maintenance on the same interval needs only one setup cost while having separate intervals, it needs for each separate interval a setup cost. After determining the first clustering of component A, it will be replaced again without affecting the frequency of maintenance of a component higher than the interval. Thus, the clustering of components relies on the interval of the earliest optimal maintenance time from the Block replacement model without setup costs. The setup costs are considered during its clustering. We add an additional variable for the formula for grouping components.

In addition, we added constraints to the MOM. The MOM algorithm will search for the cheapest maintenance intervals execution. However, this could suffer the reliability of the component. In other words, grouping maintenance could be at a later moment than a component's optimal time, which increases the expected failure of a component Shafiee and Finkelstein, 2015. The expected failure is a critical factor influencing the chance of RM. As given from the OEMs service life of B10 value, we model the maintenance intervals by not exceeding the optimal maintenance interval of the Block replacement model.

The algorithm of MOM

The summary of the previous section has been written in a python program for clustering components with the MOM. The MOM uses an iterative procedure to optimise the time of maintenance on the machine level, which exists of six steps of an algorithm:

1. First, the Block replacement model is used for each component. The Block replacement model includes the peak season length with reduced costs after peak season. In addition, the Block replacement model excludes the setup cost of each component. The setup costs will be added in step three when the algorithm considers clustering.

2. After using the Block replacement model of step one, the optimal maintenance time can be found based on the lowest cost per time unit. Having the optimal maintenance interval of each component, the algorithm divides maintenance time into ranges. The range is based on the time interval of the first component that needs maintenance. It is divided into ranges because components outside a range have other maintenance opportunities closer to their optimal maintenance interval.

3. Then, components in the first range are clustered. The optimal time interval of the components is optimised in an iterative way based on the value of component cost per time unit with the range from step one and the amount of setup cost per time unit. The MOM will optimise the first range.

4. The process of step three is repeated for the following ranges with additional consideration of previous maintenance schedules. The setup cost becomes zero on the maintenance times of the common multiple of the previous maintenance schedule.

Ultimately, the sum of the maintenance interval(Tm) for which the lowest cost rate of the machine $Z_{machine}$ $(Tm_1, Tm_2, ..., Tm_n)$ is the optimal maintenance intervals. The corresponding time interval of individual components, together with the optimal cost rate, form the optimal maintenance plan for the machine.

The visualisation of the MOM algorithm is shown in Figure 4.10.

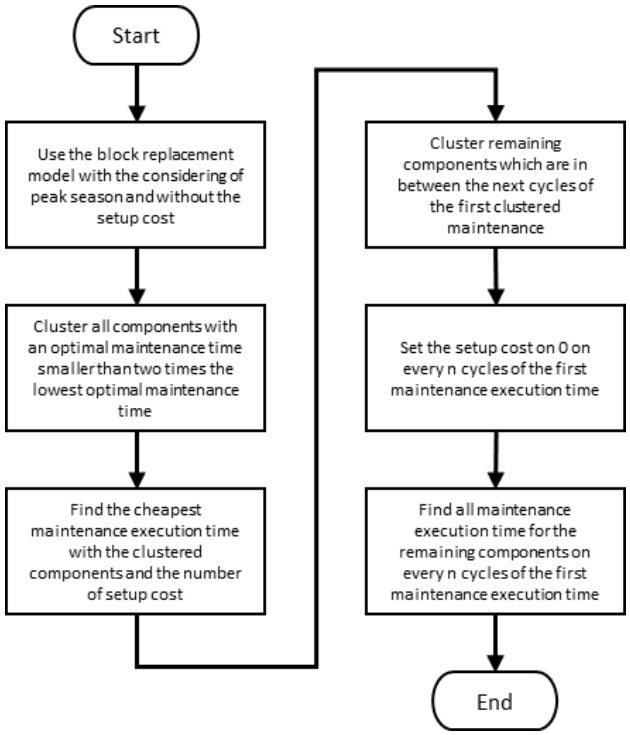


Figure 4.10: The clustering algorithm

Thus, the MOM is used to determine the maintenance interval for each component by adding the fourth variable, which is the number of setup costs. Therefore, the algorithm gives the results of when maintenance should be executed and which components should be replaced. Maintenance is executed following the results from the MOM.

4.5. Module five: maintenance execution

After determining the maintenance interval for each component from MOM, maintenance will be executed. The check phase will be started. During maintenance, the service mechanics go to the customer and its corresponding machine. They record the machine's running time and replace the components determined by MOM.

In addition, a visual inspection is held for the possible failure of other components which are not selected for this framework. Components will be recorded and taken into account for the next maintenance cycle if

it needs a replacement. The service mechanics must follow the schedule and replace components according to the MOM by recording the indication. Furthermore, the service mechanics inspect the remaining components for additional replacement and information about the other components. The only additional task for the service mechanics is reading and recording the machine's running time. The running time can be translated into the corresponding indication for each component. Therefore, the service mechanics do not have to bother about recording multiple individual indications. Their tasks are what they are meant to do. Furthermore, a visual inspection continues and the components are replaced as necessary. As a result, the practicality of the service mechanics remains unchanged.

Components that have failed during reactive maintenance are labelled as failure data and the failure times are recorded. Components that have been replaced during maintenance and are still working are labelled as suspensions (also called censored) data and recorded as suspension time.

4.6. Module six: estimation model

Module six starts with the adjustment phase of the PDCA approach. Maintenance is a dynamic plan. In this module, we discuss the outcome of the actual situation after maintenance. After maintenance, the service department recorded the maintenance data as failure and suspension time. These data points are used to estimate new Weibull parameters, which leads to improved maintenance intervals and eventually, to optimal maintenance costs. Several methods are available to estimate the Weibull parameters from failure and suspension time data (Abernethy, 2006; Lu and Wang, 2008). The commonly used methods include:

- Median rank regression(MRR)
  - MRR linearises the Weibull data and performs simple linear regression on the transformed data. The analysis estimates the parameters for the Weibull distribution through linear regression.
- Maximum likelihood estimation(MLE)
  - MLE maximises the Weibull data's likelihood function to the distribution parameters.
- Bayesian approach
  - Bayesian approach takes into account all observed and unobserved parameters in a statistical model.

These methods have been used in different situations and can be found in various papers. Several papers have compared these methods (Abernethy, 2006; Ducros and Pamphile, 2018; Genschel and Meeker, 2010; Olteanu and Freeman, 2010; Pham, 2010; Zhang et al., 2019).

MRR is generally a better estimation when a few failures and suspension data are available. However, MLE provides better parameter estimation when more failure data is available (Olteanu and Freeman, 2010). Nonetheless, both MLE and MRR do not produce stable estimates for extremely small numbers of failures or large numbers of suspension time data. Therefore, it becomes hard to pick the best method, especially for these few numbers of failures and high suspensions data.

Furthermore, MLE can easily adapt right, left and interval suspension times. MRR methodologies are only developed for the right suspended data and need to be adapted to handle more complicated data sets. Eventually, many factors seem to contribute to which method is better at estimating Weibull parameters. It includes the number of failure data, suspension data, and initial parameter values.

The Bayesian approach is an alternative approach of estimating Weibull parameters. However, Bayesian methods are very computationally intensive (Ducros and Pamphile, 2018). They require complex integration or the need for Markov chain Monte Carlo algorithms to be helpful. Due to time constraints, the Bayesian approach was not considered.

In conclusion, to keep it practical, we recommend the MLE method, which is the most widely available and used method to estimate Weibull parameters. MLE estimates better when more failure data is available and considers different types of suspension data. However, it needs at least ten failure time data available to be robust and accurate(Genschel and Meeker, 2010; Olteanu and Freeman, 2010). Ten failures are on the high side for machines, or it takes time to obtain at least ten failures.

In our case, we assumed a predictable failure behaviour of a component since we started with a high shape parameter for each component. The estimation method adjusts only the scale parameter till the first couple of failures. An algorithm is created for the estimation model. It is based on the total amount of failures.

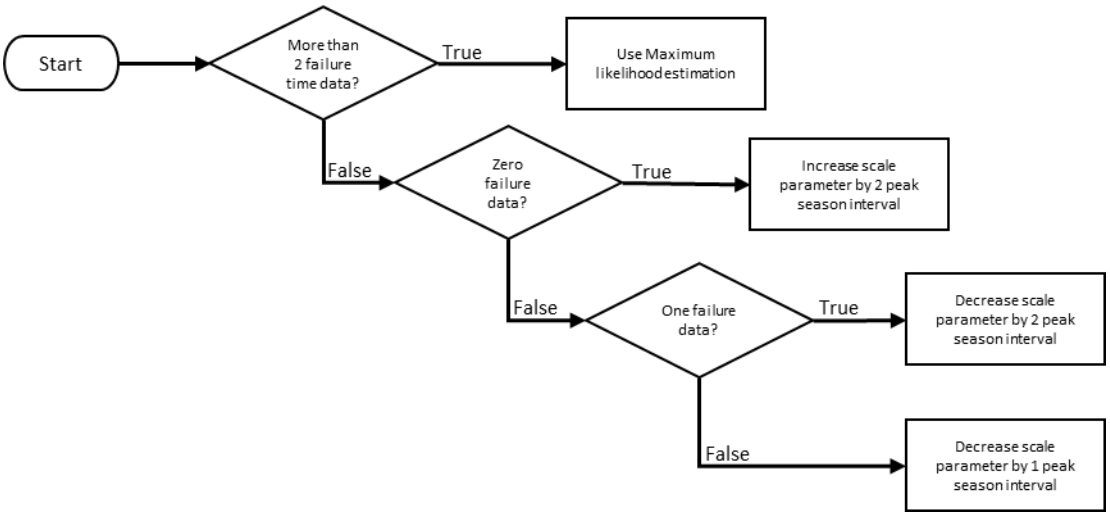


Figure 4.11: New Weilbull parameter estimation algorithm

The algorithm counts the number of failures. When a component fails, we assume that the predictability (or scatteredned) of failure over time remains the same, but the time when it fails shifts to an earlier time. That means we first only shift the scale parameter to a lower value when the component fails and increase the value when it has not failed. After three failures, the proposed maintenance framework uses the MLE method to estimate the shape and scale parameter values. Therefore the algorithm is as follows:

- We started with a shape parameter of five for which the actual shape parameter is unknown.
- For the maintenance cycles with less than three failures, the algorithm only shifts the scale parameters. It increases when no failure has occurred and decreases after the first two failures. The scale parameter will remain constant when no failures occur after the first two failures
- After three or more failures, the framework uses the MLE method to estimate both shape and scale parameters.

4.6.1. Decision after the proposed maintenance framework consider alternative maintenance plan

After estimating the Weibull parameter, the service department determines a maintenance plan. It is to keep using the proposed maintenance framework and calculate the new maintenance interval or the decision to choose an alternative maintenance plan. This is the second adjustment phase of the PDCA cycle. In this module, we must evaluate if the Maintenance optimisation model is suitable for each component.

The variables that give the most insights about component failure are the Weibull parameters. These are also the main variables that need to be estimated. Again, the shape parameters give insights about the scatteredness of failure over time and the scale parameter gives insight into the time of failure for 62.3% of the same type of component in the same conditions.

Components with a low shape or scale parameter may not be suitable for SBM in calculating maintenance intervals. It can result in high total maintenance costs or expected failures. A low shape parameter makes it challenging to determine a constant maintenance interval. The failure times are relatively random in that a short maintenance interval may be replaced prematurely and a long maintenance interval will occur multiple failures. Therefore, these components with low shape parameter characteristics may not be suitable for scheduling maintenance intervals.

This leads to added costs of maintenance and losses for customers. For a high expected failure on the maintenance time interval, it becomes less relevant to execute SBM, while the chance of failure before scheduled maintenance is unacceptable. In other words, SBM is not always suitable for each component, even when the Maintenance optimisation model is applied.

The Maintenance optimisation model shows the optimal maintenance interval with specific costs and the expected failure of a component at that specific time interval. The expected failures imply the failure

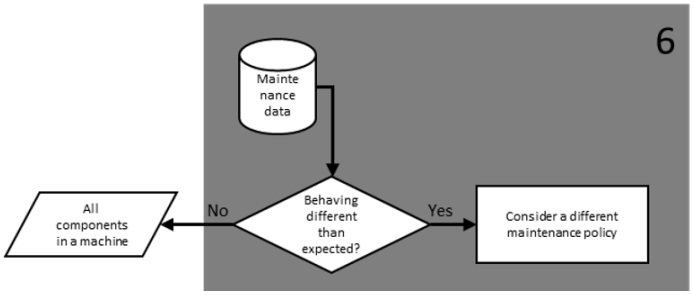


Figure 4.12: Framework of module six

behaviour of the components. The expected failure increases over time, but the increasement of the expected failure per time unit differs per component. As a result, the optimal maintenance time intervals can be on an interval where the expected failure is relatively high. For example, if there is an expected failure of 0.5 (50%) of a component on the optimal maintenance interval, the components have failed half of the time before the maintenance time interval so that a high amount of reactive maintenance will still be present. This implies that scheduled maintenance can be irrelevant to apply. Therefore these components need an alternative maintenance plan.

Lastly, there are exceptional cases in which components are not closely related to the running time or productivity of the machine. Some plastic or electrical components could be, for example, sensitive to temperature. An operating environment with a relatively high temperature degrades faster to failure than an environment with a low temperature. As a result, a customer could fail much more often when it has a relatively high temperature than customers when it is cold. Different data sets need to be collected for these components to estimate the failure behaviour and maintenance plan better. That is why the failure cause is an essential data set to record. An unknown failure causes that occurs more often needs to be investigated and to why the unknown failure occurs.

The three main possible alternative maintenance plans are discussed and each of the following maintenance plans will be explained in the next section:

- Consider CBM. The service department needs to consider an additional condition measurement which is close related to its failure;
- Redesign the component or machine to become viable for the MOM from module four;
- Use components till it has failed and store these components as spare parts. Optionally, on-site service mechanics can replace them.

4.6.2. Evaluation of Condition-based monitoring

Some components continue to fail randomly at the calculated running time. This may mean that failure is less affected by time. Other factors may play a role that is closely related to the failure behaviour. These failure causes should become recognisable after many maintenance cycles. Then it may be preferable to add a system to monitor the condition of the closely related failure. CBM would be the maintenance policy to realise the maintenance plan. The idea of CBM is the continuous monitoring of the condition of a component by using sensors. The condition measurement is continuously gathered or shown and used to assess the current condition. These components need to be further analysed on technical and economic feasibility to ensure that CBM is the most suitable maintenance policy (Tiddens, 2018).

4.6.3. Evaluation of redesign

For the redesign, a component is replaced with a component of a different quality, size, or strength capacity. A redesigned component would reduce the uncertainty so that the expected failure becomes low for the optimal maintenance interval. Then the proposed maintenance framework may be applicable again.

4.6.4. Evaluation of training on-site technicians

If we compare the costs of both RM and PM, we can make the cost factors visible. Recall equation 4.4, 4.5 and 4.7. The cost factors used in both maintenance are the cost of components (C_{comp}) and the cost of repair

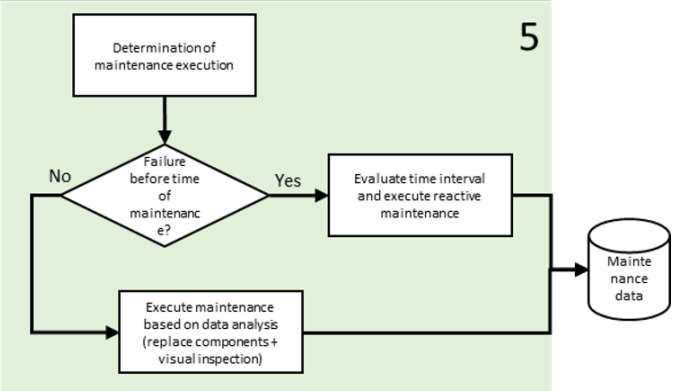


Figure 4.13: Framework of module six

$(C_{mh} \cdot T_{repair})$. When these cost factors become dominant, the RM and PM will differ little from each other. As a result, RM maintenance may be conducted as a maintenance plan. This also applies if the other cost factors of RM are insignificant since the costs of PM are more expensive the more often PM is committed early.

Figure 4.13 shows the framework of module six.

4.7. Overview of the proposed maintenance framework

In this chapter, we proposed a maintenance framework that starts from the planning phase to the adjusting phase of the PDCA cycle methodology. We proposed a Maintenance optimisation model to calculate the optimal maintenance interval with the consideration of the peak season intervals and grouping components in the same maintenance intervals. The calculated maintenance intervals by the Maintenance optimisation model are used for the maintenance execution. During the maintenance execution, the components that have been replaced are recorded as failure or suspension time. These recorded times are used to estimate Weibull parameters accurately. The outcome of the proposed maintenance framework is the shape and scale parameters of the Weibull parameters and the maintenance time intervals. When the Weibull parameters become accurate, it gives the service department insights into determining a maintenance plan. The shape tells about the scatteredness of failure over time. A highly scattered failure over time which is a low shape parameter means the component has relatively random failure intervals. A constant maintenance interval provided by the Maintenance optimisation model could be inappropriate. The scale parameter tells about the time when 62.3% of the same type of components will fail. Both give insights into using the maintenance time interval from the Maintenance optimisation model or considering an alternative maintenance plan. The alternative maintenance plans are discussed briefly in module six. However, the implementation of alternative maintenance plans needs to be further investigated. The investigation of an alternative maintenance plan is for each component different. Each component may have unique solutions with different variables. The components in which new maintenance plans must be implemented are further filtered. We improve the maintenance plan and act accordingly from the data collected from the previous maintenance framework cycle. The complete schematic proposed maintenance framework is shown in 4.14.

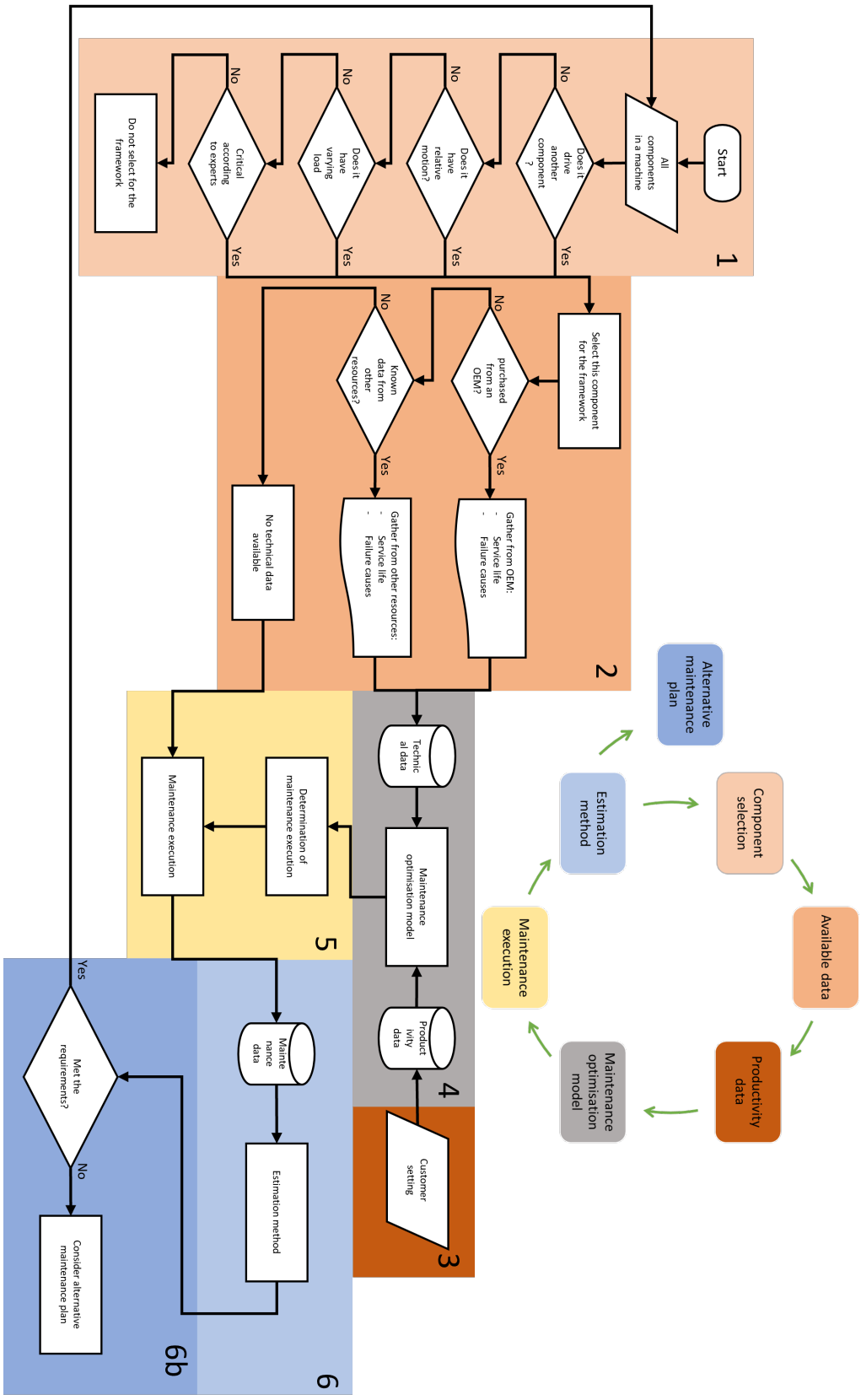


Figure 4.14: Proposed maintenance framework

5

Simulation model

Due to the time constraint of this project, we implemented and used a simulation model to obtain results and KPIs. It also helped to develop a tool for the Maintenance optimisation model and the estimation method for the service department. That tool is nothing more than a manual input instead of automated generated values. So a simulation model is developed to provide the last research question of this project: **What are the benefits of the maintenance methodology relative to the initial maintenance plan?**

To facilitate that VPT can put the proposed maintenance framework into practice and show the benefits of the proposed maintenance framework, a simulation model is developed in Python program. The simulation model is the repeat process of modules four, five and six with the inputs of modules one, two and three and some additional inputs. The simulation procedure is shown in Figure 5.1 and will be similar to the python code written for the service department. The general working of this simulation is as follows:

1. First, the inputs are inserted in the simulation model. The inputs consist of the selected components with all related maintenance costs, Weibull parameters and the productivity data. These were the inputs collected from modules one, two and three. The simulation model has an additional input: the true Weibull parameters of each component and the total machine years. In reality, the true Weibull parameters are unknown initially. However, due to its simulation, we must insert the true Weibull parameters to simulate the failure lifetimes of components.

2. Then, the simulation model starts at virtual time = 0 on the timeline. The Maintenance optimisation model is used to calculate the maintenance interval of each component. The Maintenance optimisation model optimises the cost per running time in an iterative way. The optimal maintenance interval for a component is calculated based on the cost values from all other components.

3. Then, based on the values of the true Weibull parameters of each component, a set of arbitrary duration of lifetimes are generated for each component over the course of given total machine years. This step represents module four.

4. These arbitrary lifetimes are rounded to integer numbers and will be sorted in failure and suspension data. The lifetime becomes suspension data when the generated data is higher than the determined maintenance interval. The suspension data becomes the time of the maintenance interval. The lifetime becomes failure data when it is lower than the determined maintenance interval. The failure data is the time of the generated lifetime. This step represents a simple version of module five with no visual inspection.

5. New Weibull parameters are estimated based on failure and suspension data. The estimation method is based on the number of failures. This step represents module six.

6. Lastly, the virtual time is added by the first maintenance interval and the Maintenance optimisation model is repeated for the new Weibull estimates from the previous step. Components that did not have maintenance on the first maintenance interval will be reduced over virtual time. Step two to step six is repeated till the given total machine years.

7. Meanwhile, data such as maintenance intervals, Weibull estimates, failures and suspension data are saved in an excel file for verification and as data for the calculation of the total maintenance costs. Eventually, the total maintenance costs over the years can be calculated. The stored data is also relevant to the KPIs of the proposed maintenance framework.

5.1. The input of the simulation model

First, the simulation model's input is the data collected from the planning phase of the proposed maintenance framework. These inputs are as follows:

- Amount of components
- Related to maintenance costs
  - Component costs per component;
  - Repair time per component;
  - Penalty costs per component;
  - Setup costs;
  - Capacity;
  - Manhour cost;
- Component's failure behaviour
  - Service life per component;
  - Initial shape parameter per component;
- Productivity data
  - Peak season interval;
  - Hours per week;
  - Life span of a machine;

In addition, the following variables are only in the simulation part:

- Total machine hours or the total amount of maintenance cycle;
- Amount of generated duration of lifetime.

5.2. The Output of the simulation model

The outputs are used as the new input for the next maintenance cycles as well as the data collection. The data collection of each maintenance cycle will be saved in an excel file. The excel files are collected in order to find the median and average over the course of the total machine years. The output, as well as the new input of the next maintenance cycles, are the followings:

- Maintenance interval per component
- Failure and suspension time per component. This indicates the number of failures and suspensions as well.
- Weibull parameter estimates
- Maintenance costs over different maintenance methods. The maintenance methods include the total maintenance costs of reactive maintenance, perfect scenario maintenance, service contract maintenance and the proposed maintenance framework.

5.3. Implementation and verification of the simulation model

We implemented the simulation model in Python. The code can be found in Appendix G and H for the simulation and maintenance tools, respectively. To verify the implemented simulation model, we use the numerical example of the use case in the next Chapter. The result of the verification can be found in Appendix F. We can conclude that the model is implemented correctly based on the verification and the heuristic approach.

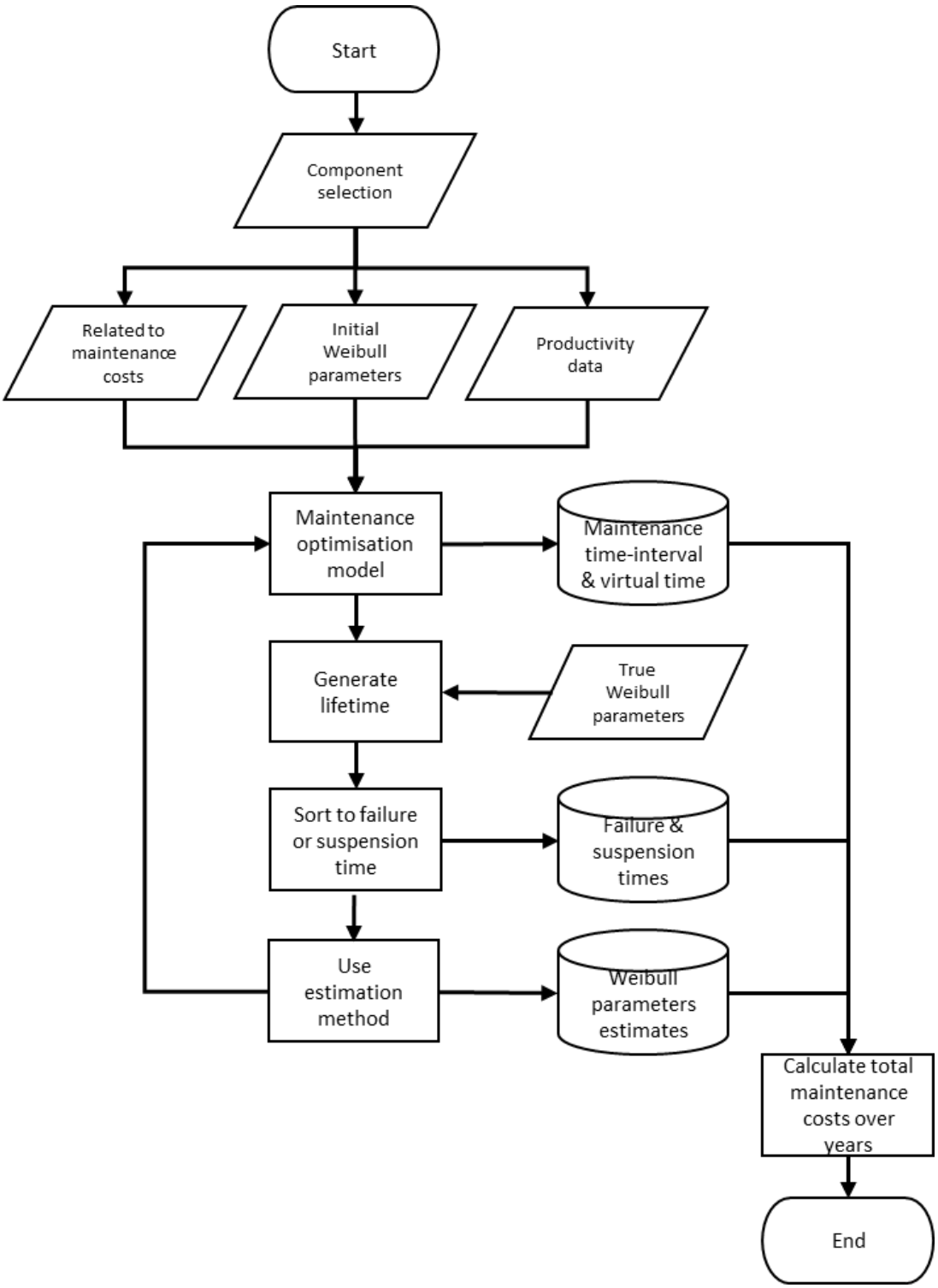


Figure 5.1: A visualisation of the simulation procedure.

6

Use case

In this chapter, we perform a use case using the proposed maintenance framework for the components of the Autostix machine from VPT. Based on the simulated results of this use case, we can show results from the proposed maintenance framework. The results of the proposed maintenance framework help the service department to select the most suitable maintenance plan. From the results, we can then discuss the usefulness of the proposed maintenance framework. In this chapter, we will provide results that will answer the last research question: **What are the benefits of the maintenance methodology relative to the initial maintenance plan?**

6.1. Autostix machine

In total, 30 Autostix machines are sold. They are mainly sold in the United States. Usually, the machines are used for a double shift on the highest mode for 15 hours daily and in our case, most machines run only during peak seasons for about ten weeks. Due to time restrictions of requesting the technical data of each component and the simulation time of a framework cycle, we will focus only on one system of Autostix with the most common productivity mentioned above. The system of Autostix, which we are focusing on, consists of a gripper system subdivided between the gripper cart and the gantry system. The gripper beam consists of different components. Mainly the gripper system has many moving components. Figure 6.1 shows the Autostix machine and its focused subsystem.

6.2. Component selection

As Chapter 4 mentioned, module one starts with component selection. The gripper beam consists of more than 100 components. Some components are the same and endure the same applied load. These same types of components are categorised as the same component. Other components that are the same but endure different applied loads are categorised as different components. For example, bearing blocks are used multiple times in a machine. It indicates the same components but can have different applied loads due to the location of the component in the machine. The different applied loads of the same components will be indicated in the brackets of table 6.1.

The service department selects components from the system by following the decision tree of module one, shown in Figure 4.1. The selected components are shown in Table 6.1 and these can be implemented in the database to automate the process.

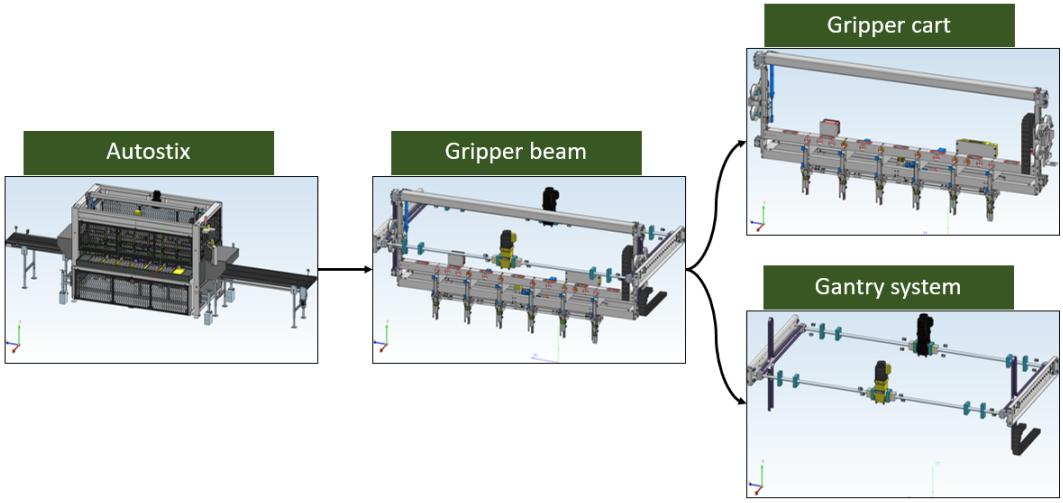


Figure 6.1: Autostix machine zoomed in its focused system and subsystem level

Table 6.1: Selected components through module one

System category	Component description	Selected based on
Grippers	Gripper profile	Wear & tear
Grippers	Pricker plate	Wear & tear
Grippers	Gear and gear rack	Wear & tear
Grippers	Roller cassette and rail (on the grippers)	Wear & tear
Grippers	Gearwheel and gear rack	Wear & tear
Grippers	Igus flanged bush	Wear & tear
Gantry system	Roller cassette and rail (vertical on the gantry system)	Wear & tear
Gantry system	Belt pulley idler	Fatigue
Gantry system	Bearing 6004-2RS	Fatigue
Gantry system	Roller cassette and rail (horizontal on the gantry system)	Fatigue
Gantry system	Gantry drive pulley	Fatigue
Gantry system	Bearing block UCFL205(1)	Fatigue
Gantry system	Bearing block UCFL205(2)	Fatigue
Gantry system	Bearing block UCFL205(3)	Fatigue
Gantry system	Gantry belt	Fatigue
Grippers	Radial Gripper	Electrical component
Grippers	Compact Cylinder	Electrical componentt
Grippers	Cylinder DHZ-20-80-PPV-A	Electrical component
Gantry system	AC Motor with Motorreductor	Electrical component

6.3. Available data gathering

All data from the OEM of the selected components (module two) and productivity data (module three) are collected. This can be done simply by requesting the technical data from OEM and the machine's control panel or customer. The service life from OEM and productivity data is shown in Figure 6.2 and 6.3, respectively. Figure 6.2 shows that the service life is expressed in different units. Combining the productivity data, the service life can be translated into the specific running hours of the customer. A cycle of a component and machine makes a specific pattern. Knowing the pattern, the service department can convert specific cycles into running hours.

Table 6.2: Components and its technical data. * same type of component, however different load.

Component description	Service life prescribed from OEM
Gripper profile	Unknown
Pricker plate	Unknown
Gear and gear rack	140 million cycles
Roller cassette and rail (g)	10 million meters
Gearwheel and gear rack	100 million cycles
Igus flanged bush	120 million meters
Roller cassette and rail (v)	20 million meters
Belt pulley idler	30 million rotations
Bearing 6004-2RS	35 million rotations
Roller cassette and rail (h)	6 million meters
Gantry drive pulley	84 million rotations
Bearing block UCFL205 (1) *	210 billion rotations
Bearing block UCFL205 (2) *	476 million rotations
Bearing block UCFL205 (3) *	59.5 million rotation
Gantry belt	4 years
Radial Gripper	12 million cycles
Compact Cylinder	480 kilometers
Cylinder DHZ-20-80-PPV-A	660 kilometers
AC Motor with Motorreductor	10 years

Table 6.3: Productivity data example of machine A

Month	Production	Capacity (cycle per hour)	Capacity of 1 gripper (cycle per gripper per hour)	Capacity per day	Days	Hours per day	Hours
1 to 2	0	0	0	0	0	0	0
2 to 3	2400000	8000	1333.33	120000	20	8	150
3 to 4	2400000	8000	1333.33	120000	20	8	150
4 to 5	1200000	8000	1333.33	120000	10	15	150
5 to 6	0	0	0	0	0	0	0
6 to 7	”	”	”	”	”	”	”
7 to 8	”	”	”	”	”	”	”
8 to 9	”	”	”	”	”	”	”
9 to 10	”	”	”	”	”	”	”
10 to 11	”	”	”	”	”	”	”
11 to 12	”	”	”	”	”	”	”
12 to 13	”	”	”	”	”	”	”
Total per year	**6000000**				**50**		**750**

The expected lifetime of Autostix machines is assumed to be 20 years. From Table 6.2 some components have a service life that is longer than the lifespan of 20 years. Therefore, these components do not need maintenance for the coming 20 years and are removed from the component selection. In the end, ten components remain to be vulnerable. These remaining ten components with their characteristics and assumed Weibull

Table 6.4: Components from service life to service life in running hours

Component number	Component description	Service life [$runninghours$]	Shape parameter (β)	Scale parameter (η)
1	Belt pulley idler	8177	5	12825
2	Bearing 6004-2RS	9290	5	14571
3	Roller cassette and rail (h)	5741	5	9004
4	Gantry drive pulley	11279	5	17690
5	Bearing block UCFL205 (3)	8121	5	12737
6	Gantry belt	5700	5	8940
7	Radial Gripper	6000	5	9411
8	Compact Cylinder	3750	5	5882
9	Cylinder DHZ-20-80-PPV-A	3047	5	4779
10	AC Motor with Motorreductor	13000	5	18821

parameters will be focused on and are shown in Figure 6.4.

6.4. Simulation model implementation

After gathering all data sets, the following phases are implemented in the simulation model. We use the simulation model from Chapter 5. The simulation is the process of calculating the maintenance interval, generating a duration of failure lifetime and estimating new Weibull parameters. The necessary data for the simulation model is sorted and shown in Table 6.5. This table shows both components, customer and simulation-specific data points.

For this use case, we used a time span of a total of 200 machine years and after that, the simulation model starts over again till 100 simulations are conducted. It may seem unrealistic for a machine to run for 200 machine years, but if several machines are operating and each machine contains multiples of the same components under the same conditions, the total machine years can be reached significantly faster in actual years. For example, if a component has an average failure time of two years, 100 total machine years can be achieved in two years with 25 machines with two components in a machine. In addition, a machine often presents the same type of components with the same applied force. Therefore, the 100 machine years can even be reached in less than 100 years. Note that if 25 machines have operated for a year, it does not mean 25 machine years have been reached if the components did not fail. The machine years are only added when the component has failed or been replaced. Furthermore, for convenience, we use the component numbers as references for the components' names.

6.5. Results

In this section, we show the proposed maintenance framework's results and the KPIs from the simulation model. The main KPI is the accurate estimates of the component failure behaviour and the accurate estimates of the maintenance intervals. The optimal maintenance interval will lead to optimal maintenance costs for some specific components. In addition, the estimation of Weibull parameters should estimate as close as to the true given parameters. Without accurate estimates, applying the Maintenance optimisation model makes no sense. It will lead to inaccurate maintenance time intervals. Eventually, it results in the wrong maintenance costs.

The challenging part is the fact that true Weibull parameters of components are unknown in reality. The duration of the lifetimes depends on true Weibull parameters. We inserted a given true Weibull parameters between one and five, shown in Table 6.5. A shape parameter higher than five will be less significant than any higher shape parameter value. A component lower than one means that these components are burn-in components, and the chance of failure decreases over time. Those components are less relevant since the machine is tested and operated for the first 50 hours. These burn-in components should already be identified in the first 50 hours before sending the machine to the customers. For the scale parameter value, these numbers are randomly inserted. Three possible possibilities are used. A scale parameter that is higher, lower or almost equal to the initial scale parameter can have a different effect on the estimation method. The determination of the initial scale parameter can be uncertain. Therefore, we inserted scale parameters that consist of those three different possibilities.

Table 6.5: Data inputs for the simulation model

Component specific data					
Component number	**Component costs [€]**	**Repair time** $[hours]$	**Penalty cost on failure** [€]	**Initial shape parameter** (β_0)	**Initial scale parameter** (η_0)
1	3128.08	3	22250	5	23100
2	14.5	2	21850	5	4920
3	1167.36	3	22750	5	14600
4	395.60	2	22250	5	9570
5	22.76	2	22050	5	4920
6	580	3	26250	5	14600
7	2848.04	1.5	22050	5	22670
8	539.16	1	21850	5	18020
9	1125.36	1.5	21750	5	22170
10	2270	1	23250	5	23530

Customer specific Data	
Setup cost	€1000
Capacity	1333 per hour
Lost per production	€0.1 per cutting · capacity + €150 per hours for the operators
Man hours	€100 per hour
Average running hours per day	15 hours
Average peak season weeks	10 weeks
Machine lifespan	20 years

Simulation specific input		
Total machine years	200 (=150000 hours)	
Maximum amount of simulation	100	
Amount of generated lifetime per component	500	
True Weibull parameters		
Component number	**True shape parameter** (β)	**True scale parameter** (η)
1	2	10000
2	1.3	22000
3	1.5	6000
4	4	15000
5	3	12000
6	5	7380
7	2	8000
8	2.5	7000
9	5	3500
10	2.5	14000

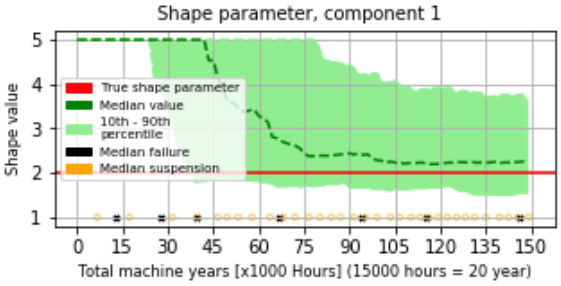


Figure 6.2: Shape parameter value over the course of 200 machine years (component 1)

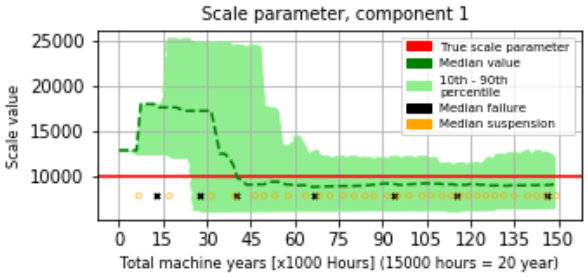


Figure 6.3: Scale parameter value over the course of 200 machine years (component 1)

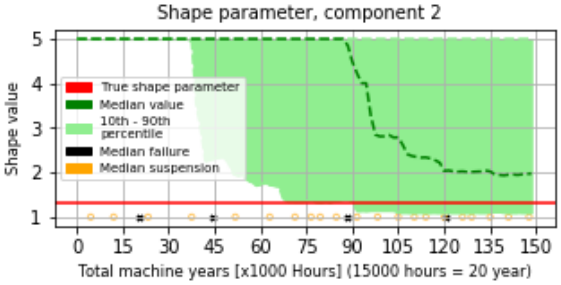


Figure 6.4: Shape parameter value over the course of 200 machine years (component 2)

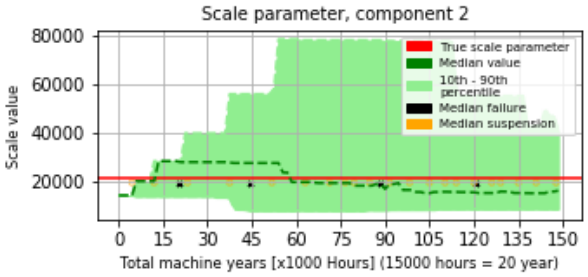


Figure 6.5: Scale parameter value over the course of 200 machine years (component 2)

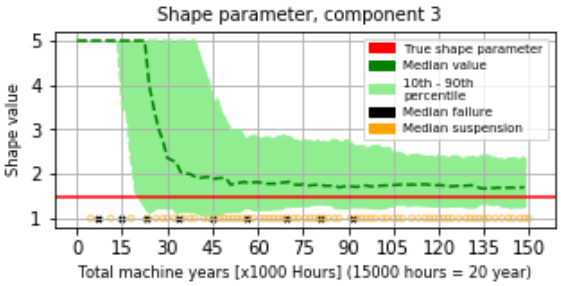


Figure 6.6: Shape parameter value over the course of 200 machine years (component 3)

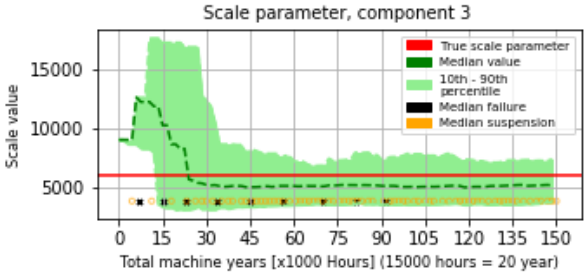


Figure 6.7: Scale parameter value over the course of 200 machine years (component 3)

6.5.1. Results on the estimation method

We start first with the estimation method. The estimates will become the inputs of the Maintenance optimisation model. So accurate estimates will lead to the optimal maintenance interval and the maintenance interval to the optimal maintenance costs over time. For the simulation model, the duration of lifetimes is generated. A distinction between suspension and failure data is made from the generated lifetimes. This is followed by the estimation method. In this use case, we have ten different components. All ten components converge within 200 machine years. The tenth, median and ninetieth percentile of the shape and scale parameters over the 200 machine years are shown from Figure 6.2 to Figure 6.21. The true shape and scale parameters are shown in the red lines. The green line is the median (50th percentile) from 100 simulations and the green area is the area of the tenth to the ninetieth percentile from 100 simulations. The median line is always between the area of the tenth to the ninetieth percentile.

6.5.2. Results on the Maintenance optimisation model

From Weibull parameters estimates, the Maintenance optimisation model calculates maintenance intervals. Figures 6.22 till 6.31 show the total maintenance costs over 200 machine years compared with the reactive maintenance (yellow line) and perfect scenario maintenance (red line). The perfect scenario maintenance is the maintenance execution in the interval before it fails in the next year. Therefore, this maintenance

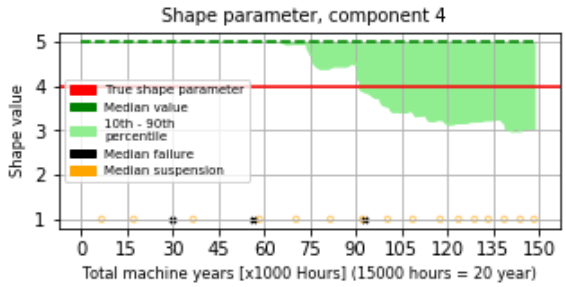


Figure 6.8: Shape parameter value over the course of 200 machine years (component 4)

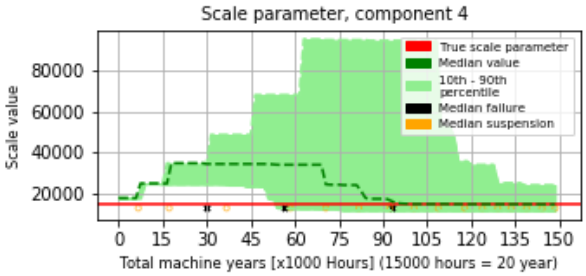


Figure 6.9: Scale parameter value over the course of 200 machine years (component 4)

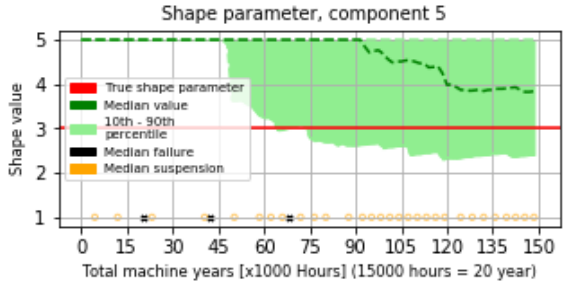


Figure 6.10: Shape parameter value over the course of 200 machine years (component 5)

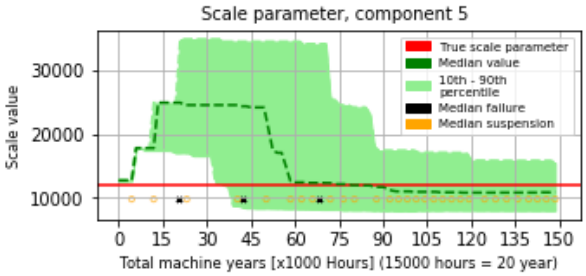


Figure 6.11: Scale parameter value over the course of 200 machine years (component 5)

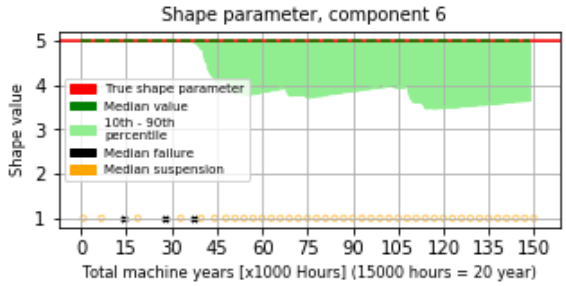


Figure 6.12: Shape parameter value over the course of 200 machine years (component 6)

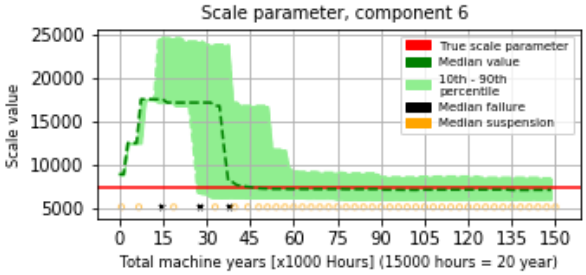


Figure 6.13: Scale parameter value over the course of 200 machine years (component 6)

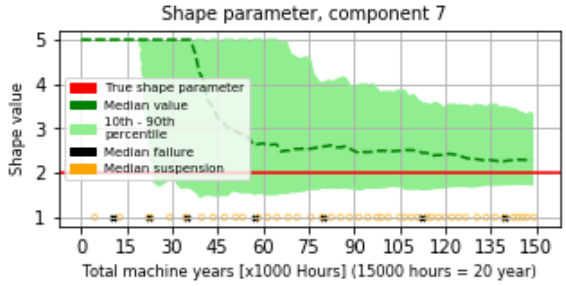


Figure 6.14: Shape parameter value over the course of 200 machine years (component 7)

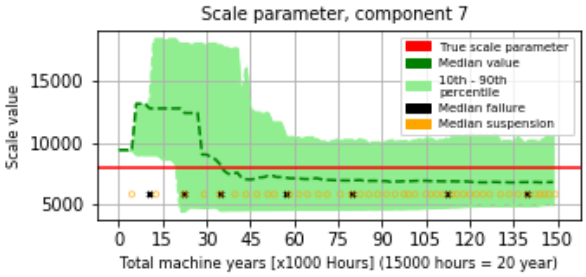


Figure 6.15: Scale parameter value over the course of 200 machine years (component 7)

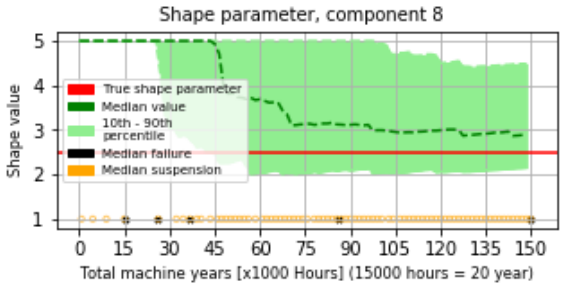


Figure 6.16: Shape parameter value over the course of 200 machine years (component 8)

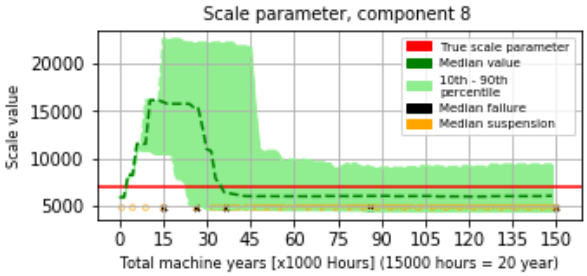


Figure 6.17: Scale parameter value over the course of 200 machine years (component 8)

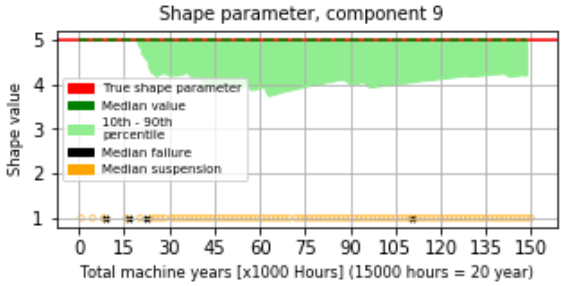


Figure 6.18: Shape parameter value over the course of 200 machine years (component 9)

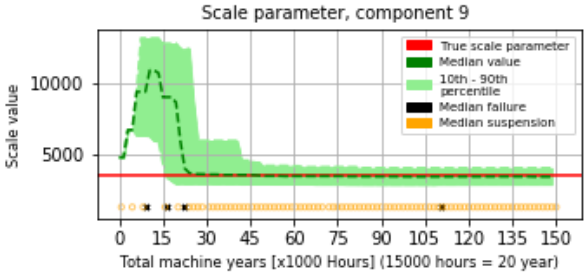


Figure 6.19: Scale parameter value over the course of 200 machine years (component 9)

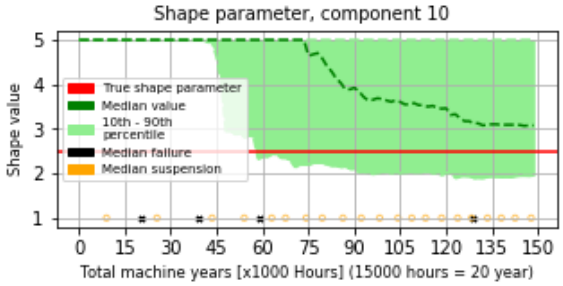


Figure 6.20: Shape parameter value over the course of 200 machine years (component 10)

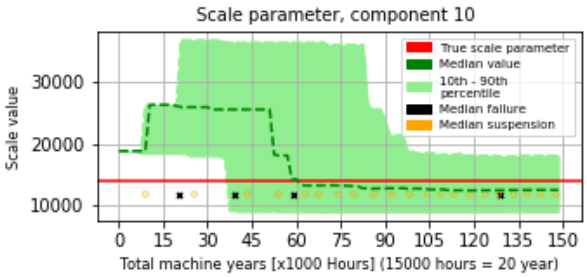


Figure 6.21: Scale parameter value over the course of 200 machine years (component 10)

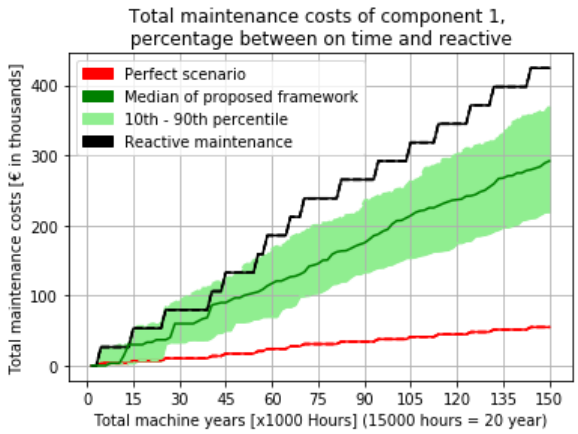


Figure 6.22: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

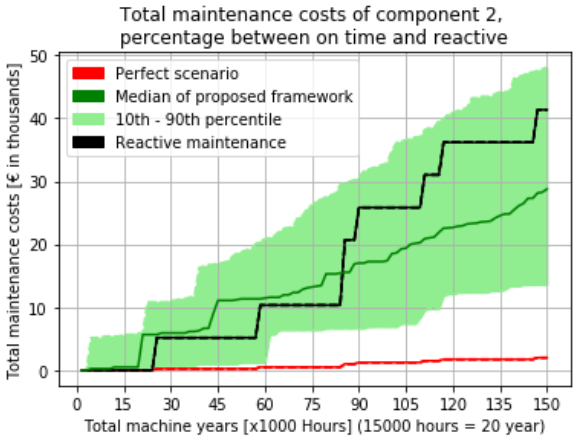


Figure 6.23: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

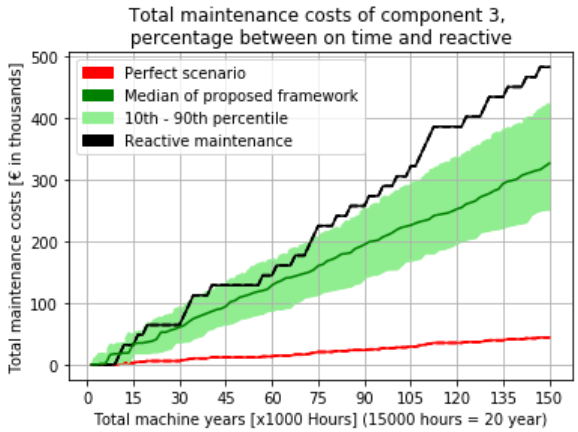


Figure 6.24: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

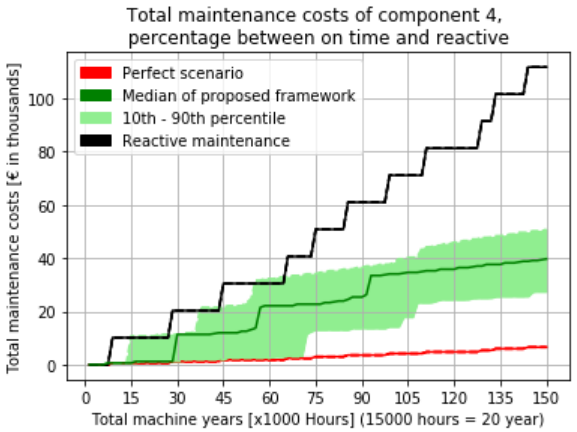


Figure 6.25: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

optimisation should be the most optimal maintenance interval since it is dynamic and is always just before the generated lifetime of each component. However, in reality, we can never predict future failures with full certainty. Furthermore, these graphs also show the tenth, median and ninetieth percentile of the simulation as the green lines and area, respectively.

6.6. Result overview

For the overview, we first need to show the maintenance intervals when we use the true Weibull parameters. Collecting failure and suspension data makes the estimates more accurate to the true Weibull parameters. To show that the Maintenance optimisation model has the most economic benefits, we compared the total maintenance costs over the machine years with different maintenance plans. Two maintenance plans are based on the current maintenance plan. These are the 'break and fix' and the 'check and replace' maintenance plans. The 'break and fix' is reactive maintenance. Components are only replaced after they have failed. The 'check and replace' is a yearly check and based on their experience, they replace it if they think it is necessary. Due to their experience, they can estimate maintenance execution similarly to the Block replacement model. However, since we do not have real data, we expect the replacement to be randomly uncertain between a stepsize early or later.

In addition, the Maintenance optimisation model is based on the Block replacement model. Therefore, we also compare the Maintenance optimisation model with the Block replacement model. Figure 6.32 shows the total maintenance costs over the course of 50 machine years with the four different maintenance plans and the maintenance interval of each component. The Block replacement model shows similar costs as the

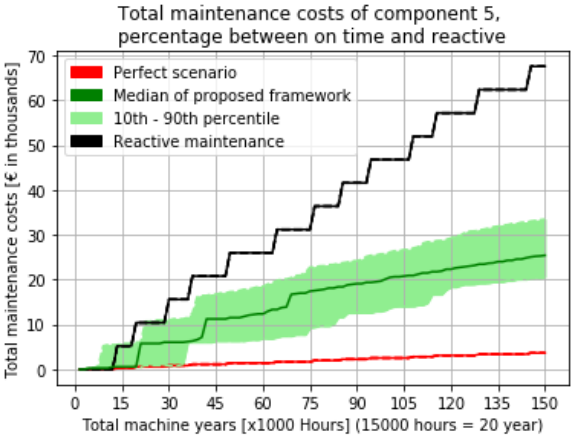


Figure 6.26: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

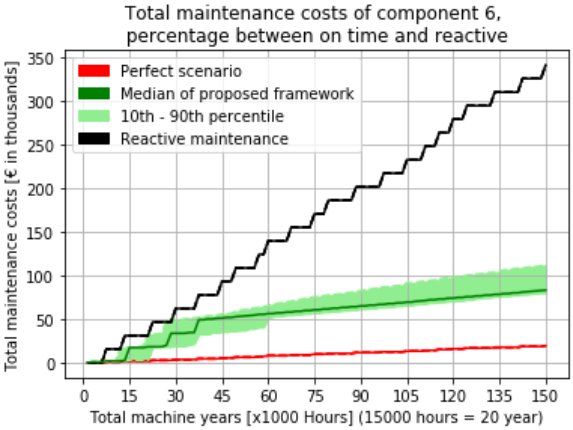


Figure 6.27: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

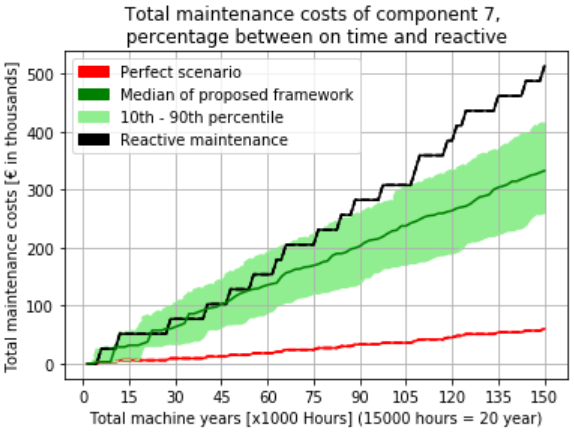


Figure 6.28: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

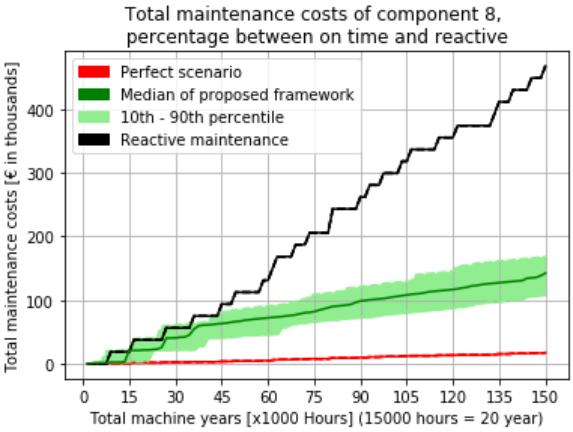


Figure 6.29: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

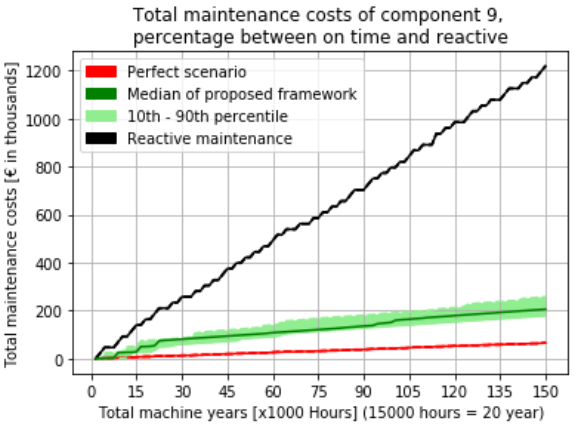


Figure 6.30: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance

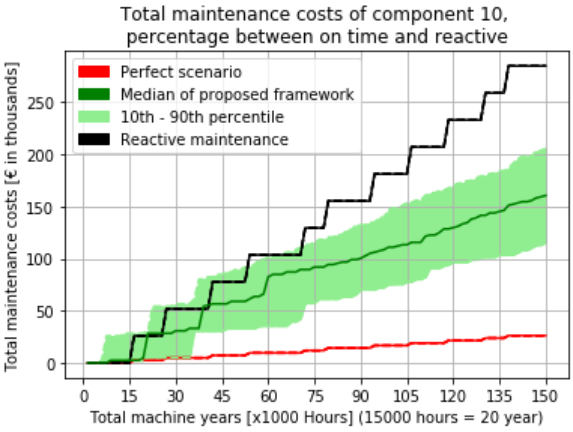
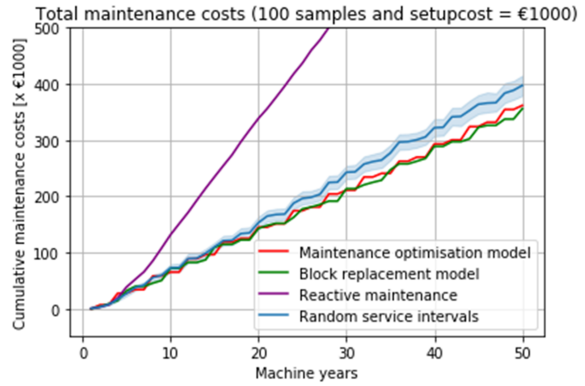
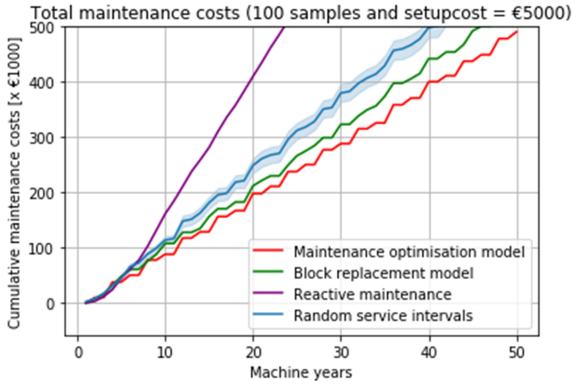


Figure 6.31: Total maintenance costs over 200 years compared to reactive maintenance and perfectly on time maintenance



Maintenance plan	1	2	3	4	5	6	7	8	9	10
MOM[years]	4	6	2	6	4	4	4	2	2	6
BRM[years]	5	7	3	7	5	4	4	2	2	6

Figure 6.32: Total maintenance costs over 200 machine years compared to reactive maintenance and perfect scenario maintenance



Maintenance plan	1	2	3	4	5	6	7	8	9	10
MOM[years]	4	4	2	6	4	4	4	2	2	6
BRM[years]	5	5	2	7	4	4	4	2	2	6

Figure 6.33: Total maintenance costs over 200 machine years compared to reactive maintenance and perfect scenario maintenance

Maintenance optimisation model. Both have more economic benefits than the 'break and fix' (reactive maintenance) and the 'check and replace' (random service intervals). The random service interval is randomly close to the Block replacement model determined. Therefore, the confidence interval of the random service intervals is shown as well.

Increasing the setup costs, it becomes clear that the Maintenance optimisation model is more beneficial than the block replacement model. The increased setup costs with the maintenance of each component are shown in Figure 6.33.

Lastly, Table 6.6 shows an overview of each component's initial, estimated and true variables. The values of these variables are all calculated from the simulation model. In addition, the accuracy of the shape and scale parameters is provided by the minimum between true value and estimated value divided by the maximum between true value and estimated value (Kędzia, n.d.).

Both Figure 6.22 till 6.31 and Table 6.6 show for which component it is suitable to use maintenance intervals. Although some components show to be unsuitable for scheduling an optimal maintenance interval, in totality, it shows benefits over the reactive maintenance and random service intervals in Figure 6.32 and 6.33.

Table 6.6: An overview of the use case with the components, Weibull parameters and their maintenance interval.

Component number	Scattered Failures			Time of 63% failures [Running hours]			Individual maintenance interval [years]			Stabilisation [Running hours]		Accuracy[%]		Median amount of failures three failures
	β_0	$\hat{\beta}$	β	η_0	$\hat{\eta}$	η	Tm_0	$\hat{Tm}$	Tm	$x_{\hat{\beta}}$	$x_{\hat{\eta}}$	$\beta_{\pm}$	$\eta_{\pm}$	
1	5	2.7	2	12800	9800	10000	9	5	5	67500	40500	73.7	97.6	4
2	5	2.2	1.3	15000	18500	22000	8	5	7	115500	82500	59.6	84.3	1
3	5	2	1.5	9000	5600	6000	6	2	3	34500	24000	74.9	94	11
4	5	5	4	17500	16100	15000	10	9	7	91500	91500	80	93	0
5	5	5	3	14100	12400	12000	8	7	5	69000	69000	60	97.1	0
6	5	5	5	10300	8200	7380	6	5	4	37500	37500	100	90.3	0
7	5	2.6	2	9000	8300	8000	6	4	4	57000	34500	76.3	96.5	4
8	5	3.1	2.5	6000	6300	7000	3	2	2	82500	37500	79.5	90.5	2
9	5	5	5	5000	4000	3500	3	2	2	112500	22500	100	87.2	1
10	5	3	2.5	19000	14200	14000	12	7	6	126000	60000	84.3	98.4	1

7

Discussion

In this chapter, we discuss and interpret the process and results of this project. We identify the importance of the proposed maintenance framework. This research aims to design a practical maintenance framework for the service department of machine-building companies. Maintenance has been solely executed based on experience. The proposed maintenance framework supports companies such as VPT with the first steps towards a data-oriented company and helps them to determine a maintenance plan. It is a maintenance framework in which the service department can collect data and determine whether preventive maintenance is suitable for each selected component. The proposed maintenance framework is based on the PDCA cycle, which consists of four phases of a maintenance process: planning, doing, checking and adjusting.

7.1. Planning phase: component selection, available data and productivity data (modules one, two and three)

The proposed maintenance framework starts with the planning phase. It contains selecting components, gathering available data from resources and collecting the productivity data from the machine. The main purpose of the planning phase is to collect available data and prepare it for the maintenance process. A database is a way to store this available data so that it can be retrieved quickly. A separate tab in the databases can be created for the selected components with their OEM data, costs factor, maintenance history, and current and past Weibull parameters value. This makes it somewhat practical and convenient to find.

Since we showed a use case in Chapter 6, the practicality of the planning phase was not proved with certainty. However, we expect the planning phase to be practical due to the iterative maintenance framework. The iterative maintenance framework gives room for improvements, missing data and adjustments. The effort to gather this data is straightforward since it can be selected and should be found quickly. Missing data, components or roughly estimated productivity data will still lead to a usable maintenance framework. Therefore, the maintenance framework is robust for incomplete data and gives the space to add and remove data in the planning phase, keeping the planning phase practical.

7.2. Doing phase: Maintenance optimisation model (module four)

After the planning phase, the service department can insert the data into the Maintenance optimisation model. The Maintenance optimisation model is based on the Block replacement model. The Maintenance optimisation model has an additional algorithm that considers all the selected components.

In Chapter 6, we showed that the Maintenance optimisation model is at least as good as the Block replacement model. As the setup costs increase, the Maintenance optimisation model becomes evident to be more beneficial. However, the maintenance intervals are the same when the first maintenance interval is small. Every other component can be scheduled on one of the maintenance intervals of the first component to be maintained. Both models will indicate the same maintenance intervals. In the case of a large first maintenance interval, the Maintenance optimisation model will not schedule every component in one of the multiplications of the first maintenance intervals. The model always makes a trade-off of a new setup cost on a different maintenance interval and no setup cost on the same maintenance interval with a higher maintenance cost. When a new setup cost, which is another maintenance interval than the first one, turns out to be

cheaper than the existing one, the component will be scheduled on a new maintenance interval. Thus, the Maintenance optimisation model is great for scheduling maintenance intervals considering multiple components.

The Maintenance optimisation model can also be used by the service department. The code is shown in Appendix H. The model is disassembled from the simulation model. The input must be inserted manually for the service department and the model calculates it for one maintenance cycle.

7.3. Checking phase: maintenance execution (module five)

After determining the maintenance intervals, the service mechanics must execute maintenance which is the checking phase. Executing maintenance remains similar to the current situation. The service mechanics perform a visual inspection. This ensures that other components that are not selected in the framework are considered and may need to be added in the next framework. In addition, the components are replaced according to the Maintenance optimisation model and the running time with its possible failure causes are saved in a database. The main concern is that everything the service mechanics or the customers have done should be recorded. This is an important mention to the customers and the service mechanics. Small adjustments and replacements already provide information for the estimation method in the next phase of the maintenance framework.

7.4. Adjusting phase: estimation method and maintenance plan determination (module six)

The adjusting phase consists of the estimation method and the decision on a maintenance plan. In Chapter 6, the Weibull estimates are shown over the course of 200 machine years in Figure 6.2 till 6.21. The key findings will be discussed from these graphs.

When we are only looking at the median of the estimated Weibull parameters, most components showed convergence close to the true Weibull parameters after three failures. The scale parameters (η) converge right after the three failures, while shape parameters (β) need some more time to stabilise. The Weibull parameters are stabilised when the median remains more or less constant over time. A slow stabilisation of the shape parameter is probably that 50% of the component did not fail. Therefore, 50% still has a shape parameter of five and the drop of the value is delayed than the scale parameter. The scale parameter increases and decreases continuously after suspension and failure, respectively. Therefore, the scale parameter converges much faster.

7.5. Results from the simulation

In Chapter 6, the simulation model has been applied for the use case. It resulted in the Weibull parameter estimates, maintenance costs over the course of 200 machine years.

Components with a true shape parameter higher than two (component 4, 5, 6, 8, 9, 10) needed only three failures to schedule appropriate maintenance intervals. It led to zero or a few failures later on. This means that the maintenance interval is correct or the maintenance is executed too early. However, since those components converge close to the true Weibull parameters after three failures, the maintenance interval is scheduled on time. That is also showed in maintenance costs graphs. The maintenance costs graphs show that these components are relative closer to the perfect scenario maintenance than the reactive maintenance. Also the slope of the framework is also closer to the slope of the perfect scenario maintenance.

The remaining components (component 1,2,3,7) led to multiple failures even when some components were estimated close to the true Weibull parameters. Components 2 and 3 have similar shape parameters, while the scale parameters are completely different. A low scale parameter shows a maintenance interval too early and a high scale parameter shows a maintenance interval too late. However, we must also consider the difference in each component's reactive and preventive costs. Component 2 has a lower reactive cost compared to component 3. Both factors of a relatively lower reactive cost and a high scale parameter discourage scheduling a maintenance interval.

In addition, the convergence of some components is after 100 machine years. While that may seem unreachable when a few machines are running. Nevertheless, when we translate that into the number of failures, we mention that it only needs three failures which sounds more plausible.

Recall from Chapter 3 that the KPIs for the proposed maintenance framework are the accuracy of the estimated Weibull parameters and the maintenance interval. The maintenance interval can only be accurately

determined when the Weibull parameters are accurate and when the Maintenance optimisation model is verified to be optimal. When we are looking at the accuracy of estimating the Weibull parameters, which are shown in Table 6.6, the scale parameter is generally better estimated than the shape parameter. The shape parameter is more difficult to predict since generated lifetime does not accurately respond to the true Weibull parameter. In Appendix G, the simulation model verification showed that the generated lifetimes are not accurate when the true shape parameter and scale parameter are again calculated with the generated lifetimes. The consequence of low accurate shape parameters on the actual maintenance costs and in reality, has to be investigated yet. However, the Weibull parameters of components are always shifting a bit (Abernethy, 2006; de Jonge and Jakobsons, 2018).

If we observe component 2, it is a component with a low shape and relatively high scale parameters. That means that 63.2 percent of the failures are in long intervals, which means that it takes time and is a challenge to estimate correctly. In addition, observing component 5, which also has a lower accuracy in the shape parameter, is indifferent using a shape parameter of three or five. Of course, both observations need more experiments to prove explicitly.

All in all, even if we observe the maintenance costs over the years from Figure 6.22 till 6.31, we see that the components higher than 2 are reasonably parallel with the perfect scenario maintenance at least after three failures. This means that components with a shape parameter equal to or lower than two are unsuitable for calculating the maintenance intervals. The shape parameter gives the service department insight into the scatteredness of failure over time. The scale parameter gives the service department insight into the lifetime in which 63.2 % have failed. In other words, it gives insight into when most components are failing. Therefore, we advise an alternative maintenance plan once a component reaches a shape parameter equal to or below two. The proposed maintenance framework schedules a low maintenance interval, which means that components are most of the time replaced too early. Condition-based monitoring might be suitable. Observing the condition of a component, instead of the time or usage, might give the appropriate indication to execute maintenance. In addition, the proposed maintenance framework could also schedule a high maintenance interval. Reactive or train in-site service mechanics could be an appropriate maintenance plan. A high maintenance interval increases the chances of failure. If the on-site service mechanics can fix it right after the failure, they can reduce huge losses in production.

This is the advice if we only observe the proposed maintenance framework. Ultimately, it is the preference of the customers. Some customers prefer to execute the maintenance early to have a high certainty that their machines will not fail and other customers do not need that certainty and will save costs if they stick to this proposed maintenance framework.

8

Conclusion and recommendations

8.1. Conclusion

In this chapter, we discuss and answer the research questions defined in section 1.1 and we state the most important conclusions. The goal of this research was to introduce a new maintenance framework for the service department of a machine-building company like VPT. The proposed maintenance framework supports the service department in determining a suitable maintenance plan. The maintenance framework helps them to collect data and guidelines towards their decisions. Eventually, we answer the research questions. We formulate the main research questions as: **How can the service department of a machine-building company determine a maintenance plan starting from experience-based maintenance?**

The first research question is the following: What are the available maintenance policies and concepts in the literature that contribute to a maintenance plan? We find that, currently, components are either reactive or preventively maintained based on experience. Given from the literature, those two are the most primitive ones. In the literature, we found preventive maintenance based on different indications in which a component needs to reach a predetermined value of that indication before it will be maintained. This was the most feasible one since it started from experienced-based maintenance. Keeping track of these indications will help the service department determine its optimal predetermined value or give insights into the component's failure behaviour which preventive maintenance based on an indication is inappropriate.

The second research question is: What are the Key Performance Indicators to validate the usefulness of the proposed methodology? In Chapter 3, we described a PDCA approach which is a great approach starting from an experienced-based maintenance towards a structured, iterative data collection approach. For this approach, the doing and adjusting phases are the most important results to be accurate. The first KPI is the accuracy of the maintenance interval which can be translated into the maintenance costs over time. The accurate maintenance interval of each component will bring the maintenance costs as low as possible. Furthermore, the second KPI is the accuracy of the failure behaviour of a component. Accurate estimates of the failure behaviour of components give the service department insight into when components will fail and the characteristics of the scatteredness over time of a component. So random failure intervals give the service department the knowledge of finding an alternative maintenance plan. Therefore, the failure behaviour has to be accurate to determine a maintenance plan.

The third research question is: What maintenance strategy is proposed? In Chapter 4, we described different modules derived from the PDCA approach. The planning phase consists of the components selection, gathering available data and collecting data from the machine or customer. After selecting the components based on failure mechanism and expert knowledge, the service department starts with gathering technical data from OEM. The OEM provides some initial values and knowledge to start with. In addition, the service department must collect data from the machine to know its productivity and running hours. From there on, we used a maintenance optimisation model to schedule the maintenance interval of each component. The maintenance optimisation model is based on the Block replacement model. However, the proposed model also considers other components, while the block replacement model only schedules it as an independent component. After determining the maintenance interval, maintenance can be executed by the service mechanics and new data as failure and suspension time provide new estimates of the failure behaviour of components by using the proposed estimation method. It shifts its scale parameter at the low amount of failure

and after three failures, it uses the Maximum likelihood estimation. New estimated values can be used for the next maintenance cycle and the cycle will be repeated again starting from the planning phase.

The fourth research question is: What are the benefits of the maintenance methodology to the initial maintenance plan? Due to the time restriction of this project, the proposed maintenance framework was not implemented in reality. Therefore, in Chapter 5, we developed a simulation model that provides the benefits of the proposed maintenance framework compared to the initial experience-based maintenance plan. The input of the simulation is the planning phase of the proposed maintenance framework and some additional duration for the process of the simulation model. After that, it simulates modules four, five and six. The output is data collection that the service department should also collect. New maintenance intervals, estimates and total maintenance costs can be determined with the collected data.

In Chapter 6, we described a use case to show the results of the proposed maintenance framework and will also be compared with other maintenance plans. There were several benefits of using the proposed maintenance framework. The first benefit of the proposed maintenance framework is the fact that most of the maintenance based on the framework is less expensive than maintenance based on experience. Secondly, the data collection gives already insights about the component's failure behaviour and scheduling of a constant maintenance interval. For experienced-based maintenance, data is incomplete or even unknown. So the total maintenance cost is a random variable that although they can guess it correctly about the maintenance interval, but the expectation is that most components are maintained either too early or too late.

All taken together, we can answer the main research question mentioned before. First, the service department can use the proposed maintenance framework to determine the optimal maintenance interval for each component. It can be used till the shape parameter from the Weibull distribution becomes two or even lower. This is the point at which failures occurrence are more scattered over time and will occur more frequently with the determined maintenance interval or the maintenance interval is scheduled too early. In other words, scheduling a maintenance interval for components with a shape parameter of two or lower may be inappropriate. The Weibull parameters and the maintenance optimisation model which are both in the process of the proposed maintenance framework, support the service department of a machine-building company in determining the optimal maintenance interval or the need for an alternative maintenance plan.

8.2. Limitations and future research

A number of limitations and future research directions can be identified from our project. First, the main focus of this project was to reduce maintenance costs over time based on component failure behaviour and schedule an optimal maintenance interval. In Chapter 6, we describe the hypothetical reactive and random service contracts case. However, in reality, no actual data has been found. So it becomes uncertain if those hypothetical reactive and random service contract costs are accurate to the actual situation.

Second, We did not consider all estimation methods. Methods such as the Bayesian approach and bootstrap need heavy computational models to work properly. These may lead to a faster approach towards the true Weibull parameters.

Third, we did not expand the proposed maintenance framework to alternative maintenance plans. This is the first research within VPT connecting maintenance and data, which is the reason we kept it a single preventive maintenance plan. Interesting future research would be to investigate how the maintenance framework can provide multiple maintenance policies for different components and combine them into one maintenance plan. For example, in this project, we determined that some components are unsuitable for a predefined maintenance interval. Therefore, some components may need condition-based maintenance. When considering components for CBM and others for SBM, a maintenance plan has to consider both maintenance policies. We expect that a multi-maintenance policy in a maintenance plan leads to even more efficient maintenance, such that the total maintenance costs can be reduced further.

Fourth, we did not consider validation research of the proposed maintenance framework. Due to the lifespan of this project, we could not implement the proposed maintenance framework, gather all the technical data from the OEM and wait several years to collect data and obtain any results. Therefore, we only made a simulation model to get hypothetical outcomes of the hypothetical inputs. It becomes interesting if the proposed maintenance framework is implemented in a real case study or experiment.

fifth, The input of the simulation keeps the same over the multiple simulations. No additional components are added or removed in the simulation model, which could be false in reality. These are all uncertain factors that can be changed in reality. However, there is an option for the service department to add, remove and take into account other maintenance schedules.

Sixth, this project consists of many variables to fill in. Since the use case is all hypothetical values, we need multiple tests to find the correlations of all variables. A small change in variables, such as the number of components, different cost factors, Weibull parameters, with or without peak season interval can all result differently. Therefore, we need more experiments with the proposed maintenance framework to come up with proves and other key findings.

Lastly, we did not include environmental aspects in the proposed maintenance framework since environmental aspects are the most challenging part of component degradation. It is a research topic on its own to experiment with different environmental aspects, such as temperature, humidity, and pH level. When the service department thinks this is necessary, the proposed maintenance framework must consider these environmental aspects.

8.3. Recommendations

The practical recommendations for Machine-building companies such as VPT are listed in this section.

Investigate additional lists in the database for the collected data of the proposed maintenance framework.

In order to implement and use the proposed maintenance framework in practice, a couple of new lists must be added to the database. Currently, there are two issues. First as explained in section 5 and 5, we developed a maintenance tool for the service department to calculate the optimal maintenance interval and estimate the Weibull parameters of each component. The outcomes of both maintenance tools should be connected to the database. This should be easier to get overviews out of the database without a lot of manual data pre-processing.

Set up a dashboard for the data collection.

In Chapter 6, we describe the maintenance interval based on average productivity data. However, in order to get insights about the actual productivity and running hours a simple dashboard needs to be developed on which both the service department and the customers can observe the machines day to day running hours and productivity.

Gather the technical data as early as possible.

When a new machine is designed or new components have to be selected for the proposed maintenance, try to request the technical data and save it in the database as soon as possible. This ensures that much effort is not required when selecting components for the proposed maintenance framework. In addition, if the technical data of components is waiting too long, some components will be forgotten and it will take extra effort to find the correct technical data.

Do not forget the visual inspection for all components during maintenance.

Besides replacement from the proposed maintenance framework, the maintenance event is also intended to observe other components for possible replacement. A visual inspection does not have to be thoroughly performed for all components, but a test and observation of components can give already some indication of the condition of the components. If the components are in bad shape, these findings must be reported to the customer. They have to decide whether they also want to have it replaced to avoid reactive maintenance.

Bibliography

Abernethy, R. (2006). *The New Weibull Handbook* (5th ed.). R.B. Abernethy.

Albrice, D. (2013). Wear Out Failure Pattern. https://www.assetinsights.net/Glossary/G_Wear-out_Failure_Pattern.html

Archibald, & Dekker. (1996). Modified block-replacement for multiple-component systems. *Transactions on reliability*, *45*, 75–83. https://ieeexplore-ieee-org.tudelft.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=488920

Arts, J., & Basten, R. (2018). Design of multi-component periodic maintenance programs with single-component models. *IISE Transactions*, *50*(7), 606–615. https://doi.org/10.1080/24725854.2018.1437301

Berg, M., & Epstein, B. (1978). Comparison of Age, Block, and Failure Replacement Policies. *IEEE Transactions on Reliability*, *R*(1), 25–29. https://doi.org/10.1109/tr.1978.5220230

Bhatt, Bhatt, Ballabh, Prakash, Baloni, & Majumdar. (2022). Automation in horticulture: The future of orchards. *Riverpublishers*. https://www.riverpublishers.com/pdf/ebook/chapter/RP_9788770227667C30.pdf

Bimba. (2012). Cylinder Life Expectancy. https://airinc.net/wp-content/uploads/2014/08/Original-Line-Cylinder_Life_Expectancy-1.pdf

Bloch, H., & Geitner, F. (2012). *Machinery Failure Analysis and Troubleshooting*. Elsevier Gezondheidszorg.

Comsol. (2016, March 15). *Material fatigue definition*. https://www.comsol.com/multiphysics/material-fatigue#:%7E:text=Material%20fatigue%20is%20a%20phenomenon,behind%20failures%20of%20mechanical%20structures.

de Jonge, B., & Jakobsons, E. (2018). Optimizing block-based maintenance under random machine usage. *European Journal of Operational Research*, *265*(2), 703–709. https://doi.org/10.1016/j.ejor.2017.07.051

de Jonge, B., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*(3), 805–824. https://doi.org/10.1016/j.ejor.2019.09.047

Ding, S.-H., & Kamaruddin, S. (2014). Maintenance policy optimization—literature review and directions. *The International Journal of Advanced Manufacturing Technology*, *76*(5-8), 1263–1283. https://doi.org/10.1007/s00170-014-6341-2

Ducros, F. ., & Pamphile, P. . (2018). Bayesian estimation of weibull mixture in heavily censored data setting. *Reliability Engineering & System Safety*, *180*, 453–462. https://doi.org/10.1016/j.ress.2018.08.008

Elatech. (n.d.). *Elatech Drive calculation* (tech. rep.). Elatech. https://www.steminbreitbach.com/doc/24029-Elatech_drive-calculation-D-E-0316.pdf

Fernández, J. . G. . F. ., & Márquez, C. . A. . (2014, April 13). *Maintenance management in network utilities: Framework and practical implementation (springer series in reliability engineering)* (2012th ed.). Springer.

Fraser, K., Hvolby, H.-H., & Tseng, T.-L. ((2015). Maintenance management models: a study of the published literature to identify empirical evidence. *International Journal of Quality & Reliability Management*, *32*(6), 635–664. https://doi.org/10.1108/ijqrm-11-2013-0185

Galar, D., & Kumar, U. (2017). *eMaintenance*. Elsevier Gezondheidszorg.

GE digital solutions. (n.d.). Distribution Types | Predix APM | GE Digital. https://www.ge.com/digital/documentation/predix-apm/latest/reliability-analytics-distribution-types.html

Genschel, U. ., & Meeker, W. Q. (2010). A comparison of maximum likelihood and median-rank regression for weibull estimation. *Quality Engineering*, *22*(4), 236–255. https://doi.org/10.1080/08982112.2010.503447

Helmenstine, T. (2021). Calculate Percent Error. https://sciencenotes.org/calculate-percent-error/

Hibbeler, R. (2016). *Engineering Mechanics* (14th Global edition). Pearson Education Limited.

Hoogendoorn, D. (2020). *Condition based maintenance with event and usage data at Canon Production Printing* (tech. rep.). Series Master Theses Operation Management; Logistics. https://pure.tue.nl/ws/portalfiles/portal/151787566/Master_Thesis_Dirk_Hoogendoorn.pdf

Ingemarsdotter, E., Kambanou, M. L., Jamsin, E., Sakao, T., & Balkenende, R. (2021). Challenges and solutions in condition-based maintenance implementation - A multiple case study. *Journal of Cleaner Production*, *296*, 126420. https://doi.org/10.1016/j.jclepro.2021.126420

isixsigma. (n.d.). *B10 life*. https://www.isixsigma.com/dictionary/b10-life/

Janssen, M., Zuidema, J., & Wanhill, R. (2004). *Fracture Mechanics, Second Edition* (2nd ed.). Taylor & Francis.

Jardine, A., & Tsang, A. (2013). *Maintenance, Replacement, and Reliability: Theory and Applications, Second Edition (Mechanical Engineering)* (2nd ed.). CRC Press.

Jiménez, A., & Bermúdez, M. (2011). Friction and wear. *Materials Science*, 33–63.

Kędzia, Ł. (n.d.). Calculating forecast accuracy and precision. https://best-excel-tutorial.com/59-tips-and-tricks/530-calculating-forecast-accuracy-and-precision

Kobbacy, K. A. H., & Murthy, P. (2008). *Complex System Maintenance Handbook*. Springer Publishing.

Li, X., & Zhang, C. (2015). Delayed Age Replacement Policy with Uncertain Lifetime. *Mathematical Problems in Engineering*, *2015*, 1–6. https://doi.org/10.1155/2015/528726

Lu, & Wang. (2008). Weibull data analysis with few or no failures. *Recent Advances in Reliability and Quality in Design*, 201–210. https://doi.org/10.1007/978-1-84800-113-8

Lutkevich, B. (2020). framework. https://www.techtarget.com/whatis/definition/framework#:%5C%7E:text=In%5C%20general%5C%2C%5C%20a%5C%20framework%5C%20is,the%5C%20structure%5C%20into%5C%20something%5C%20useful

Márquez, C. . A. . (2010). *The Maintenance Management Framework: Models and Methods for Complex Systems Maintenance (Springer Series in Reliability Engineering)* (Softcover reprint of hardcover 1st ed. 2007). Springer.

Meng, H., & Ludema, K. (1995). Wear models and predictive equations: their form and content. *Wear*, *181*, 443–457. https://doi.org/10.1016/0043-1648(95)90158-2

Mishra, R. P., Gupta, G. ., & Sharma, A. . (2021). Development of a model for total productive maintenance barriers to enhance the life cycle of productive equipment. *Procedia CIRP*, *98*, 241–246. https://doi.org/10.1016/j.procir.2021.01.037

Moncmanová, A. (2007). Environmental factors that influence the deterioration of materials. *Environmental Deterioration of Materials*, 1–25. https://doi.org/10.2495/978-1-84564-032-3/01

Moubray, J. (1999). *Reliability-Centred Maintenance* (2nd). Elsevier.

Nakajima, S., & Bodek, N. (1988). *Introduction to TPM: Total Productive Maintenance (Preventative Maintenance Series) (English and Japanese Edition)* (Eleventh Printing). Productivity Pr.

Narayan, V. (2005). *Effective Maintenance Management*. Industrial Press Inc.

Nipun, B. . (2015, October 14). *Difference between yield strength and tensile strength*. Retrieved October 3, 2022, from https://pediaa.com/difference-between-yield-strength-and-tensile-strength/

Nowlan, F. . S. ., & Heap. (1978). *Reliability-centered maintenance*. Amsterdam University Press.

Olteanu, D. ., & Freeman, L. . (2010). The evaluation of median-rank regression and maximum likelihood estimation techniques for a two-parameter weibull distribution. *Quality Engineering*, *22*(4), 256–272. https://doi.org/10.1080/08982112.2010.505219

Pasha, G., Khan, M., & Pasha, A. (2006). Empirical analysis of the Weibull distribution for failure data. *Journal of Statistics*, *13*(1), 33–45. https://www.researchgate.net/publication/265074902_Empirical_Analysis_of_The_Weibull_Distribution_for_Failure_Data

Patil, A., Soni, G., Prakash, A., & Karwasra, K. (2021). Maintenance strategy selection: a comprehensive review of current paradigms and solution approaches. *International Journal of Quality & Reliability Management*, *39*(3), 675–703. https://doi.org/10.1108/ijqrm-04-2021-0105

P.E., S. N. (2021). *Practical Plant Failure Analysis: A Guide to Understanding Machinery Deterioration and Improving Equipment Reliability, Second Edition* (2nd ed.). CRC Press.

Peeters, J., Basten, R., & Tinga, T. (2018). Improving failure analysis efficiency by combining FTA and FMEA in a recursive manner. *Reliability Engineering & System Safety*, *172*, 36–44. https://doi.org/10.1016/j.ress.2017.11.024

Perneder, R., & Osborne, I. (2012). Handbook Timing Belts. *Springer-Verlag Berlin Heidelberg 2012*. https://doi.org/10.1007/978-3-642-17755-2

Pham, H. . (2010, October 28). *Recent advances in reliability and quality in design (springer series in reliability engineering)* (Softcover reprint of hardcover 1st ed. 2008). Springer.

Poór, E., Ženíšek, S., & Basl, A. (2019). Historical Overview of Maintenance Management Strategies: Development from Breakdown Maintenance to Predictive Maintenance in Accordance with Four Industrial Revolutions. *IEOM Society International*. http://ieomsociety.org/pilsen2019/papers/135.pdf

Sachs, N., & Neville, W. (2007). *Practical Plant Failure Analysis* (2nd edition). Taylor & Francis.

Savits, T. H. (1988). A cost relationship between age and block replacement policies. *Journal of Applied Probability*, *25*(04), 789–796. https://doi.org/10.1017/s0021900200041589

Shafiee, & Finkelstein. (2015). An optimal age-based group maintenance policy for multi-unit degrading systems. *Reliability Engineering & System Safety*, *134*, 230–238. https://doi.org/10.1016/j.ress.2014.09.016

Shafiee, M. (2015). Maintenance strategy selection problem: an MCDM overview. *Journal of Quality in Maintenance Engineering*, *21*(4), 378–402. https://doi.org/10.1108/jqme-09-2013-0063

SKF. (n.d.). Bearing rating life. https://www.skf.com/sg/products/rolling-bearings/principles-of-rolling-bearing-selection/bearing-selection-process/bearing-size/size-selection-based-on-rating-life/bearing-rating-life

Smith, R., & Mobley, K. (2003). *Industrial Machinery Repair* (first edition). Butterworth-Heinemann.

Tam, A. S. B., Chan, W. M., & Price, J. W. H. (2006). Optimal maintenance intervals for a multi-component system. *Production Planning & Control*, *17*(8), 769–779. https://doi.org/10.1080/09537280600834452

Tiddens, W. (2018). Setting sail towards predictive maintenance : developing tools to conquer difficulties in the implementation of maintenance analytics. *Tools4LCM project*. https://doi.org/10.3990/1.9789036546034

Tinga, T. (2012). *Mechanism Based Failure Analysis*. Nederlandse Defensie Academie.

Tinga, T. (2013). *Principles of Loads and Failure Mechanisms*. Springer Publishing.

Tinga, T., Wubben, F., Tiddens, W., Wortmann, H., & Gaalman, G. (2020). Dynamic maintenance based on functional usage profiles. *Journal of Quality in Maintenance Engineering*, *27*(1), 21–42. https://doi.org/10.1108/jqme-01-2019-0002

van Elderen, C. (2016). *Multi-item maintenance optimization based on failure, usage and condition information* (tech. rep.). Series Master Thesis Operations Management; Logistics. https://pure.tue.nl/ws/portalfiles/portal/46944329/855089-1.pdf

Vilarinho, S., Lopes, I., & Oliveira, J. A. (2017). Preventive Maintenance Decisions through Maintenance Optimization Models: A Case Study. *Procedia Manufacturing*, *11*, 1170–1177. https://doi.org/10.1016/j.promfg.2017.07.241

Waeyenbergh, G., & Pintelon, L. (2009). CIBOCOF: A framework for industrial maintenance concept development. *International Journal of Production Economics*, *121*(2), 633–640. https://doi.org/10.1016/j.ijpe.2006.10.012

Wan, A., Gu, F., Chen, J., Zheng, L., Hall, P., Ji, Y., & Gu, X. (2018). Prognostics of gas turbine: A condition-based maintenance approach based on multi-environmental time similarity. *Mechanical Systems and Signal Processing*, *109*, 150–165. https://doi.org/10.1016/j.ymssp.2018.02.027

Wang, H. . (2002). A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research*, *139*(3), 469–489. https://doi.org/10.1016/s0377-2217(01)00197-7

Wang, H., Wang, W., & Peng, R. (2017). A two-phase inspection model for a single component system with three-stage degradation. *Reliability Engineering & System Safety*, *158*, 31–40. https://doi.org/10.1016/j.ress.2016.10.005

Wang, W. (2006). *Condition Monitoring and Control for Intelligent Manufacturing* (2006 ed.). Springer Publishing.

Zhang, L. ., Jin, G. ., & You, Y. . (2019). Reliability assessment for very few failure data and weibull distribution. *Mathematical Problems in Engineering*, *2019*, 1–9. https://doi.org/10.1155/2019/8947905

Zhu. (2015). *Technische universiteit eindhoven* (No. 978-90-386-3777-8). Technische Universiteit Eindhoven. https://research.tue.nl/en/publications/maintenance-optimization-for-multi-component-systems-under-condit

A

Appendix A: Scientific Research Paper

A practical maintenance framework to determine a maintenance plan starting from experience based maintenance plan

R. Cai, P.H. Jonker, A. Shah and W. van den Bos

Abstract— **Maintenance of complex machines or assemblies is an essential part of machine-building companies. Besides building and designing machines, they are responsible for replacing components and restoring the machines to maintain good operational conditions. An optimal maintenance plan ensures that customers spend less on maintenance and remain satisfied. Also, for the machine builders, an optimal maintenance plan shows efficient use of the service mechanics, trust in the machines and the quality of the knowledge. This research aims to create a maintenance framework that leads to a suitable maintenance plan starting from experience-based maintenance. The framework proposes steps in which initial data points are collected, calculates the maintenance intervals and estimates values for a failure distribution to obtain knowledge of components while it is still manageable (practical) for the service department. Eventually, this paper shows the benefits of the proposed maintenance framework relative to maintenance based on experiences like reactive maintenance. A simulation model has been developed, showing the KPIs of the framework and results from the framework. In addition to the simulations, this project provides guidelines for the service department to implement the framework.**

Keywords—Maintenance plans, Maintenance optimisation model, Practical maintenance framework, lack of data

I. Introduction

The horticulture sector is one of the fastest-growing industries worldwide [Bhatt et al.,]. One of the reasons is the numerous changes in the horticulture sector. It is facing challenges such as a lack of skilled labourers, climate changes and increased plant quantities. Consequently, there is a high demand for automation in the horticulture sector. Viscon Plant Technology (VPT) is one of those companies focusing on the automation process in the horticulture sector. They provide many kinds of machinery that automate the process of growing, harvesting or picking up plants and relocating from one pot to another. Although the machinery is constantly improving and becoming more innovative, maintenance is also an essential part that helps companies become the market leader in the sector. The main challenge is that a maintenance plan is undervalued since the maintenance plans of the company are still based on experience. It needs to keep up the pace of this fastest-growing industry sector.

For years, maintenance in the horticulture sector can be characterised as ”break and fix”; Maintenance starts when a customer requires it because something is wrong with the machine. A failure occurs unexpectedly, which takes some time to respond to and fix. Usually, it is an unprepared event that must be fixed rapidly due to production losses. Another commonly used maintenance plan is the ”check and replace”; Maintenance is executed when machines reach a specific time interval, in which service mechanics check the machine before a component has failed and replace it if it is necessary. It may lead to benefits since failures occur less frequently. However, the ”check and replace” can lead to unnecessary checks and maintenance can be executed prematurely. The leading causes of keeping experienced-based maintenance are little research has been conducted, there is no data collection and experienced-based maintenance is practical.

Several maintenance plans have been developed and generalised in the last three decades. Usually, it requires some data collection. Data collection ensures that maintenance can be executed based on various indications from past data. Several papers have shown how specific data types, indicators, or systems can be used to estimate when maintenance needs to be executed.

Some simple indications, such as (running) hours or amount of cycles, may already improve the maintenance interval relative to experience-based maintenance. These values are recorded when failure or maintenance occurs. From the recorded data, a limit value can be determined to execute maintenance. Therefore, a failure can be prevented by executing maintenance on one of those indications [Tinga et al., 2020]. These are primarily preventive maintenance. An upcoming hot topic of research in the field of maintenance is condition-based and predictive maintenance. Several papers show how and which methods and indications can be used to show possible improvements in maintenance costs and insights on the component's failure behaviour [de Jonge and Jakobsons, 2018], [Ding and Kamaruddin, 2014], [Fraser et al., 2015].

These hot maintenance plans are advanced methods and could greatly reduce maintenance costs and significantly increase knowledge of the failure behaviour. However, these maintenance plans are often only used on a small scale and experimentally [Zonta et al.,], [Tiddens et al., 2018].

Condition-based and predictive maintenance are not always obvious to implement because of the large investments in time and costs and a large trade-off with practicality [Tinga et al., 2020], [Tiddens, 2018].

An investment in condition-based or predictive maintenance starting from experience-based maintenance would not be recommended since it is unknown which component gained significant benefits versus other maintenance plans. It is especially a huge investment when the current maintenance plan is completely based on experience without collecting data.

This paper will focus on the determination of a maintenance plan for the service department of a machine-building company that uses an experience-based maintenance plan. Therefore, the main research question of this analysis is:

- How can the service department of a machine-building company determine a maintenance plan starting from an experience-based maintenance plan?

This main research question is broken down into sub-questions that are stated as follows:

- What are the available maintenance policies and concepts in the literature that contribute to a maintenance plan?
- What are the Key Performance Indicators to validate the usefulness of the proposed methodology?
- What maintenance strategy is proposed?
    1) What are the available scarce data?
    2) What data need to be collected during maintenance?

3) How can the maintenance schedule be optimised?
4) How will this maintenance flow determine a maintenance plan?

- What are the benefits of the maintenance methodology relative to the initial maintenance plan?

These sub-questions are investigated and answered respectively in the sections of this research paper. The remainder of this paper is organised as follows. In section II, we describe terms commonly used in the maintenance field. In section III, the methodology towards the results of determining a maintenance plan is explained. We formulate the proposed maintenance framework in section IV. A use case of the proposed maintenance framework on a production-line machine of Viscon Plant Technology (VPT) is presented in section V. Finally, we end with conclusions and directions for future research in section VI.

II. Maintenance review

In this section, we describe common terms used in the field of maintenance. Over the past decades, substantial research has been done on maintenance policies, concepts, and other maintenance-related definitions. These maintenance-related terminologies have become more general [Narayan, 2005], [Smith and Mobley, 2003]. Others review various maintenance policies, strategies, selection problems or concepts on different machines, systems or components [Ding and Kamaruddin, 2014], [Fraser et al., 2015]. The first research question can be answered: **What are the available maintenance policies and concepts in the literature that contribute to maintenance improvement?**

A. Maintenance policies

As described before, three main maintenance policies are generally defined in the literature. It consists of reactive, preventive and predictive maintenance. Usually, maintenance is executed based on one or a combination of indicators. Such indicators with its maintenance policy could be the following:

- Machine cannot start $->$ reactive (RM)
- Absolute time $->$ scheduled based (SBM)
- Running time $->$ hour based (HBM)
- Amount of cycles $->$ usage based (UBM)
- Travelled distance $->$ usage based (UBM)
- Vibration $->$ condition based (CBM)
- Combination of multiple indicators $->$ predictive (PdM)

B. Other maintenance plans consideration

The maintenance policies described in the previous subsections are primarily long-term maintenance plans. While maintenance plans such as design out maintenance (DOM) can be executed as a one-time event. This is when the current design does not meet the requirements. Then it can be considered to redesign a component, system or machine. All maintenance policies are shown in Figure 1

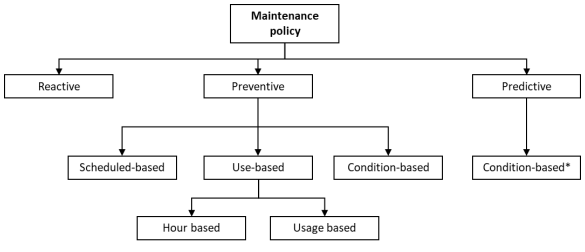


Fig. 1. Maintenance policies

C. Maintenance concepts

In addition, maintenance concepts describe the entire step-by-step plan for determining a maintenance policy or the entire structure behind a maintenance policy. Reliability Centered Maintenance (RCM) and Total Production Maintenance (TPM) are more practical concepts that have been applied in many companies [Moubray, 1999], [Nakajima and Bodek, 1988]. These concepts needed huge investment and cultural changes in the company. Other concepts, such as multi-criteria decision-making (MCDM) and three-stage funnel-based, are mainly academic concepts applied in various papers [de Jonge and Jakobsons, 2018], [Tiddens, 2018], [Patil et al., 2021], [Shafiee, 2015], [Ding and Kamaruddin, 2014]. These concepts mainly contain mathematical models or algorithms.

D. Literature outcome

Each maintenance policy has its own benefits. We compared the maintenance policies based on four categories. we ordered the policies for each category:

- Data readiness: This is the relative amount of different data types that the company should have already collected.
- Practicality: This is the service department's relative difficulty in implementing the maintenance policy.
- investment: This is the relative amount of investment to implement the maintenance policy. This could be for example, sensors, systems and training courses.
- Maintenance Cost: This is the return on costs and losses of each maintenance policy. This criterion is the total cost for the customers.

Due to an experience-based maintenance plan, they do not have any data ready. Data on the condition need additional sensors to evaluate the condition. On the other hand, a maintenance plan solely on RM can lead to the highest maintenance costs, which is the least preferable choice for customers. Therefore, this paper will mainly focus on the preventive maintenance of SBM, HBM and UBM. The investment and the data readiness are relatively low, while it is still practical. Although concepts have useful methods, tools and steps, no concept can be implemented without any data, cultural change within the company or major investments. Therefore, in this paper, we develop a new maintenance framework.

III. Methodology

In the remaining chapters, we will elaborate on the methodology. Recall from the introduction that there is a need for a practical maintenance framework as a guideline for the initial start of data collection. From the previous section, Preventive maintenance based on time, running time and usage is the most practical data to collect first. These collected data can then be used either to:

- Optimise preventive maintenance by using the most optimal indication and limit;
- Consider an alternative maintenance plan.

We will answer the second sub question: What are the Key Performance Indicators to validate the usefulness of the proposed methodology? To optimise a maintenance interval, we need to provide a guideline or a tool that provides the maintenance interval of each component. The maintenance interval should be as close to the failure time. Therefore, the methodology should estimate the maintenance interval as accurately as possible. Furthermore, the estimates of the failure behaviour should be accurate as well. Providing an accurate

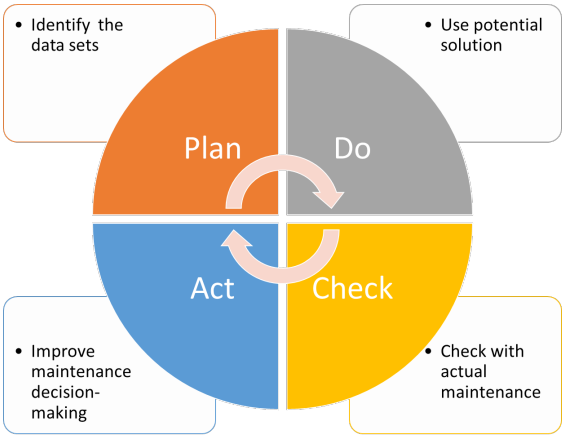


Fig. 2. The flow of a PDCA cycle

failure behaviour of a component gives the service department reliable insights. The insights might lead to an alternative maintenance plan for components with still relatively random failure intervals.

The methodology of the proposed maintenance framework utilises an iterative cycle of planning, doing, checking and adjusting (PDCA approach) [Waeyenbergh and Pintelon, 2009], [de Jonge and Jakobsons, 2018], [Nakajima and Bodek, 1988]. The PDCA cycle in the next section can include all the aspects of the research design, and it is simultaneously structured in a flow of planning, doing, checking and adjusting. It is a suitable and a practical flow of preparing data, using the data to schedule maintenance intervals and checking the maintenance results to obtain knowledge for the next suitable maintenance cycle. A visual methodology of the PDCA cycle is shown in Figure 2, which starts from the upper left corner.

The proposed maintenance framework should ultimately lead to lower costs in maintenance and losses and it should lead to insights into the failure behaviour of components. This means that the component's failure behaviour estimates must be accurate to interpret the values properly.

IV. Proposed Maintenance Framework

In this section, we describe the proposed maintenance framework and continue to answer the third research sub-question: **What maintenance strategy is proposed?** This will be answered as well as the corresponding sub-questions.

A. Module one: component selection

The first phase of the PDCA cycle is to identify the data sets. To identify the data sets, we must determine which machine, its systems, and its components we need to collect data.

Every material, component and system in machines will fail after a certain amount of time. Each material has its rate of deterioration, and many factors affect this rate. As a result, each component in a machine has a unique failure time due to its characteristics, environmental effects, and loads applied to the components. Literature papers have established possible failure mechanisms [Sachs and Neville, 2007], [Tinga, 2013]. The failure mechanisms branch even further to the possible failure causes of the component. When looking at machines in the horticulture sector, these machines are typically used continuously for many hours in sequence. The components in the machines move mainly in accelerations and slowdowns repeatedly. The products in the machines are moved and processed by a robot arm, gripper or any other processing tool that moves and stops a product instantly at the right position in the machine back and forth. Therefore, we are focusing on components with these characteristic failure mechanisms. these failure mechanisms can be identified by answering the following questions:

- Does the component drive other components? (related to electrical components)
- Does the component have relative motion between the contacting surface and itself? (related to wear and tear components)
- Does the component use varying load? (related to fatigue components)

The second method is based on experts' experiences and knowledge related to the machine. Experts include the operators and customers who observe the machine during operation and the engineers and service mechanics responsible for its design and maintenance. The components that the experts recommend must also be selected in this framework. They experience which components are critical, i.e., frequently fails, high breakdown time, or any other undesirable factors [Shafiee, 2015].

B. Module two: gather available data sets

After selecting the components from module one, we searched for available data sets. Since no data have been collected during maintenance, the service department has to initially gather data sets somewhere else.

During the design phase of machines, the choice of components is determined based on what the component must be able to withstand in terms of force, pressure and size. In other words, the components are chosen based on the design engineers' requirements to become suitable components for the machine. Usually, these components are purchased from an Original Equipment Manufacturer (OEM) or a supplier.

Components purchased from either OEM or supplier should have technical data sheets related to their failure behaviour. They should have done the required testing and experiments. Thus, the service department can request the service life, failure causes and possible other failure-related data. The service life is usually given as a B10 value. The B10 value is the time at which 10% of the same type of components have failed. These given service life and failure causes are the initial data sets to determine a maintenance interval.

However, there may also be custom-made components in a machine. It is challenging to determine when and how custom-made components will fail without data from the OEM. Data on custom-made components may be unknown because engineers fabricate them themselves without testing them for a long period. These components are assembled in the machine and will be inspected during each maintenance interval to evaluate the condition.

C. Module three: gather productivity data set

After gathering the available data set of each component, the service department needs additional data from the customer: the productivity data. This is due to the fact that component degradation depends highly on the productivity of the machine. Heavy use of the machine affects the components differently than low use [Tinga et al., 2020].

Instead of assigning the service mechanics to collect all specific data for each component, we derive these specific data from a general counter of the whole machine. It saves time for the service mechanics and they can do their job of executing maintenance instead of collecting data. The most basic productivity data to start with is the running hours.

It is only the hours that the machine is operating. Different customers have different operating times, leading to different running hours for the same year.

D. Module four: the Maintenance optimisation model(MOM)

Module four is the doing phase of the PDCA cycle. It concerns the analysis of the data gathered from modules one to three. When many data points are collected from multiple maintenance cycles, module four becomes more valuable in finding a suitable maintenance interval.

Our model assumptions:

- All failure events are statistically independent;
- Each maintenance is perfect maintenance;
- The failure distribution is a Weibull distribution;

The proposed MOM is based on the Block replacement model (BRM). Maintenance is executed based on an optimal time interval by considering all the failure events in between the maintenance interval. A short maintenance interval means the frequency of maintenance increases and a long maintenance interval means the chance of failure increases. The BRM will find the optimum based on the cost per unit of time. The formula for component costs per unit of time in the Block replacement model is as follows:

$$\text{Component costs per time unit} = \frac{\text{Total expected costs}(TEC)}{\text{Length of interval}(LoI)}$$

The Total expected costs (TEC) include Preventive and Reactive costs. The Length of Interval (LoI) contains the interval of maintenance. The lowest total maintenance costs per time unit of component *i* consist of a preventive cost and the expected number of reactive cost divided by the maintenance interval. The formula for the Block replacement model becomes equation 1:

$$Z_i(Tm) = \frac{C_{PM} + H(Tm) \cdot C_{RM}}{Tm} \tag{1}$$

Where

- $Z_i(Tm)$ = Lowest total maintenance cost per time unit of component i;
- C_{PM} = Total cost if PM is applied;
- $H(Tm)$ = The expected number of failures in the interval $(0, Tm)$;
- C_{RM} = Total cost if RM is applied;
- Tm = Time of maintenance.

The preventive and reactive costs can be divided into different costs and losses factors. The type of costs and the values depend on the company and its machine[Hoogendoorn, 2020], [van Elderen, 2016]. The failure distribution of each component is based on a Weibull distribution. Due to unknown Weibull parameters, we propose to start with a low scatteredness of failure over time (shape parameter = 5) and the failure time can be converted from the service life gathered from the OEM and the reliability formula of the Weibull distribution (see eq 2).

$$R(t) = e^{-}\left(\frac{t}{\eta}\right)^{\beta} \tag{2}$$

Where

- t = unit of service life(usually in time);
- η = scale parameter of Weibull distribution;
- β = shape parameter of Weibull distribution;
- $R(t)$ = Reliability

In addition, the proposed MOM uses an iterative algorithm to optimise and find the optimal maintenance interval for each component by considering grouping multiple components on the same maintenance interval. The algorithm consists of the following four steps:

1) First, the BRM is used for each component. However, for the MOM, it excludes the setup cost of each component. The setup costs will be added in step three when it considers the grouping of components.
2) Just like the BRM, the optimal maintenance interval can be found based on the lowest cost per unit time. Having the optimal maintenance interval of each component, the algorithm divides maintenance into ranges. The range is based on the maintenance interval of the first component that needs maintenance.
3) Then, the MOM iterates components in the first range considering the frequency of setup costs. Grouping components on the same interval need one setup cost and adding each maintenance interval needs a setup cost. The MOM finds the lowest costs per time unit
4) The process of step three is repeated for the following ranges with additional consideration of previous maintenance intervals. The setup cost becomes zero on the maintenance intervals that have already been determined. Ultimately, the model finds all component's maintenance intervals (Tm).

E. Module five: maintenance check

The check phase starts after the maintenance intervals are determined from the previous module. The service mechanics must adopt the schedule and replace components according to the MOM and recording the running hours as failure time or suspension time if the component did not fail. Furthermore, the service mechanics inspect the remaining components for additional replacement and information about the other components.

F. Module Six: determination of a maintenance plan

After the check phase, the service department can come up with a conclusion for determining a maintenance plan. As mentioned before, the result of the proposed maintenance framework is either to optimise the maintenance intervals or to consider an alternative maintenance plan. First, the service mechanics collects data points from each maintenance. The data points must be analysed to estimate new Weibull parameters. The data points collected from maintenance are used by the Maximum Likelihood Estimation (MLE) to estimate Weibull parameters. Since almost all estimation methods need many data points to obtain accurate Weibull parameters, MLE is the most widely available, well-known and robust. We recommend at least three failures to estimate with MLE accurately. Before three failures, we increase or decrease the scale parameter which implies increasing or decreasing the service life and so also the maintenance interval.

Lastly, when the Weibull parameters are roughly accurate after three failures, the service department can use the Weibull parameter as insight into whether the maintenance intervals from the MOM are appropriate. The scale parameter gives the failure time in which 63.2 % of the component has failed and the shape parameter gives insights into the failure scatteredness over time. An estimated low shape parameter (shape $<= 2$) means that components fail at random (running) time. This means that the maintenance intervals based on running hours might not be a great maintenance indicator before failure time.

Therefore, an alternative maintenance plan would be more suitable.

V. Use case

In this section, we perform a use case using the proposed maintenance framework for the components of a machine from VPT. Based on the simulated results of this use case, we show the results of the proposed maintenance framework. In this section, we will answer the last research question: **What are the benefits of the maintenance methodology relative to the initial maintenance plan?**

A. Autostix machine

The machine that will be used is the Autostix machine. The machine automatically picks up cuttings and puts them in a tray or pots. Usually, the machines are used for a double shift on the highest mode for 15 hours daily and in our case, most machines run only during peak seasons for about ten weeks. Due to time restrictions of requesting the technical data of each component and the simulation time of a framework cycle, we will focus only on the gripper system of Autostix with the most common productivity mentioned above.

B. Planning phase

After gathering all data sets from modules one to three. The necessary data for the optimisation model is sorted and shown in Table I. The table shows both components and customer-specific data points.

C. Remaining phases with simulation

We developed a simulation model for the remaining phases to calculate the optimal maintenance interval by the MOM, generate failure lifetimes based on the given input and estimate new Weibull parameters. It is the combination of modules four, five, and six. The inputs of Table I are inserted into the Python program.

Furthermore, the simulation model repeatedly does the process after estimating new Weibull parameters for multiple machine years to show possible outcomes of the proposed maintenance framework.

D. Results

From Table I, ten components have been selected to demonstrate the approach presented in this section. The failure and productivity data have been obtained from the OEM and customer, respectively. Table I shows all the inputs and gathered data from modules one to three.

As result, Figure 3 show three different graphs of the best and the worst component for MOM, respectively. The first two graphs of Figure 3 show the Weibull parameters estimates compared with the true Weibull parameters and failures over the course of 200 machine years. The simulation was conducted over 100 times and the 10th, 50th (median) and 90th percentile are shown. The third graph compares the total maintenance costs with reactive and perfect scenario maintenance costs. The perfect scenario is the maintenance on the largest maintenance interval before failure.

The results of each component are shown in Table II. It shows the initial, estimate and true variable of shape, scale and maintenance interval of each component. The estimated and stabilisation years are after three failures of the total machine years.

TABLE I

Data inputs for the simulation model

Component specific data

Component number	Component costs [€]	Repair time $[hours]$	Penalty cost on failure [€]	Initial shape parameter (β_0)	Initial scale parameter (η_0)
1	3128.08	3	22250	5	23100
2	14.5	2	21850	5	4920
3	1167.36	3	22750	5	14600
4	395.60	2	22250	5	9570
5	22.76	2	22050	5	4920
6	580	3	26250	5	14600
7	2848.04	1.5	22050	5	22670
8	539.16	1	21850	5	18020
9	1125.36	1.5	21750	5	22170
10	2270	1	23250	5	23530

Customer specific Data

Setup cost	€1000
Capacity	1333 per hour
Lost per production	€0.1 per cutting · capacity + €150 per hours for the operators
Man hours	€100 per hour
Average running hours per day	15 hours
Average peak season weeks	10 weeks
Machine lifespan	20 years

Simulation specific input

Total maintenance years	100 (=75000 hours)
Amount of simulation	100
Amount of generated lifetime per component	500

True Weibull parameters

Component number	True shape parameter (β)	True scale parameter (η)
1	2	10000
2	1.3	22000
3	1.5	6000
4	4	15000
5	3	12000
6	5	7380
7	2	8000
8	2.5	7000
9	5	3500
10	2.5	14000

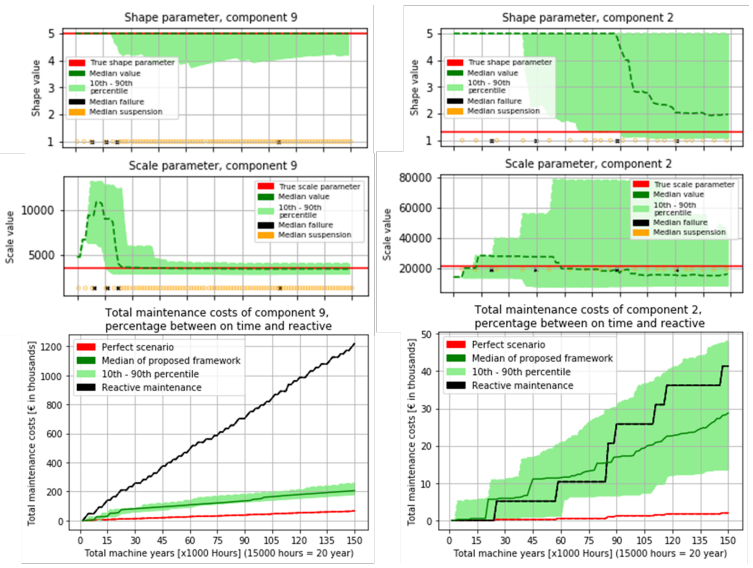


Fig. 3. All characteristics of component 9 and 2, their true values, maintenance time and costs over the course of 200 machine years

TABLE II

AN OVERVIEW OF THE USE CASE WITH THE COMPONENTS, WEIBULL PARAMETERS AND THEIR MAINTENANCE INTERVAL.

Component number	Scattered Failures			Time of 63% failures [Running hours]			Individual maintenance interval [years]			Stabilisation [Running hours]		Accuracy[%]		Median amount of failures three failures
	β_0	$\hat{\beta}$	β	η_0	$\hat{\eta}$	η	Tm_0	$\hat{Tm}$	Tm	$x_{\hat{\beta}}$	$x_{\hat{\eta}}$	$\beta_{\pm}$	$\eta_{\pm}$	
1	5	2.7	2	12800	9800	10000	9	5	5	67500	40500	73.7	97.6	4
2	5	2.2	1.3	15000	18500	22000	8	5	7	115500	82500	59.6	84.3	1
3	5	2	1.5	9000	5600	6000	6	2	3	34500	24000	74.9	94	11
4	5	5	4	17500	16100	15000	10	9	7	91500	91500	80	93	0
5	5	5	3	14100	12400	12000	8	7	5	69000	69000	60	97.1	0
6	5	5	5	10300	8200	7380	6	5	4	37500	37500	100	90.3	0
7	5	2.6	2	9000	8300	8000	6	4	4	57000	34500	76.3	96.5	4
8	5	3.1	2.5	6000	6300	7000	3	2	2	82500	37500	79.5	90.5	2
9	5	5	5	5000	4000	3500	3	2	2	112500	22500	100	87.2	1
10	5	3	2.5	19000	14200	14000	12	7	6	126000	60000	84.3	98.4	1

VI. DISCUSSION

In this paper, we developed a maintenance framework. A maintenance framework in which the service department can schedule the optimal maintenance interval and determine whether maintenance intervals are suitable for each selected component.

The proposed maintenance framework is based on PDCA cycle. It starts with the planning phase. The planning phase is expected to be practical since missing components, data and roughly estimated productivity will still lead to a usable maintenance plan. Therefore, the maintenance framework is robust for incomplete data and gives the space to add and remove data points in the planning phase, keeping the planning phase practical.

The Maintenance optimisation model (MOM) is based on the Block replacement model (BRM). However, it adds the consideration of grouping multiple components. Scheduling multiple components on the same maintenance intervals reduces maintenance frequency, leading to reduced maintenance costs and the convenience of maintaining multiple components simultaneously.

As a result of the use case based on simulation, Figure 3 started with a shape parameter of five and the scale parameter was the only adjustment one before three failures. After three failures, the MLE method estimates close to the true Weibull parameters. Especially, components with relatively uniform failure intervals (Shape > 2) had low deviations in the simulations. A low shape parameter with a high scale parameter becomes hard to estimate due to randomness in a large range of the scale parameter. In addition, a low shape parameter will either schedule a short or long maintenance interval. The right column of Figure 3 shows that it may seem excellent since the Weibull parameters were close to the true values and there was only one failure after the three failures. However, when we look at the maintenance costs graph, it shows cost close to reactive costs. This combines a long maintenance interval at the beginning and a short one after three failures. Both choices are worst for components with relative random failure intervals. The MLE is the most available one to estimate Weibull parameters. Heavy computational approaches could estimate faster and more accurate than the MLE.

Furthermore, a shape parameter of three became indifferent to any higher shape parameter. Components four and five remain an estimated shape parameter of five. It does not have a significant impact on a shape parameter higher than three. This means that a shape parameter higher than three has a uniform failure interval of five and the MOM is suitable.

Lastly, despite an accuracy of 80-90%, the maintenance interval are mostly accurate. We can conclude that the maintenance intervals are close to the actual maintenance intervals.

All in all, even if we observe the maintenance costs over the machine years from Table II, we notice that the components higher than two are reasonably parallel with perfect scenario maintenance at least after three failures. This means that components with a shape parameter equal to or lower than two are unsuitable for this preventive maintenance plan. The shape parameter gives the service department insight into the scatteredness of failure over time. The scale parameter gives the service department insight into the lifetime in which 63.2% have failed. In other words, it indicates components failure time. Therefore, we advise an alternative maintenance plan once a component reaches a shape parameter equal to or below two. For other components, we advise to keep using the maintenance framework since the maintenance intervals are close to the perfect scenario maintenance. Eventually, it is up to the service department and the customers if they want to switch to an alternative maintenance plan or to keep using the proposed maintenance framework. In addition, the proposed maintenance framework is open to be expanded to multiple alternative plans or add additional factors as environmental aspects.

VII. CONCLUSION

This paper focuses on developing a maintenance framework for the service department to determine a maintenance plan by either using the maintenance intervals provided by the maintenance framework or it shows the need for alternative maintenance plans. By creating a maintenance framework, we plan the data sets by selecting components and gathering available and productivity data. A MOM is developed based by the BRM with additional consideration of grouping components. After determining the maintenance intervals by MOM and executing maintenance, the service department can estimate the failure distribution of each component. The Weibull distribution was chosen for this project. Based on a use case and the simulation, we observed that most components are estimated close to the true values after three failures. Components with random failure intervals will have low shape parameter estimates, while relatively uniform failure intervals will have relatively high shape parameter estimates. The characteristic lifetime of a component will be expressed in the scale parameter. Both shape and scale parameters will be estimated and it will give the service department insights into the component's failure behaviour. These indicators determine the choice of using the proposed maintenance framework or using looking for an alternative maintenance plan which also answers the main question of this project. Looking from the results of the use case in this project, it is advisable to use an alternative maintenance plan if components reach a shape parameter of two or lower. For other components, they can still keep using the proposed maintenance framework since it provides an optimal maintenance interval based on total costs.

REFERENCES

[Bhatt et al.,] Bhatt, S., Bhatt, A., Ballabh, J., Prakash, S., D., B., and Majumdar, S.

[de Jonge and Jakobsons, 2018] de Jonge, B. and Jakobsons, E. (2018). Optimizing block-based maintenance under random machine usage. *European Journal of Operational Research*, 265(2):703–709.

[Ding and Kamaruddin, 2014] Ding, S.-H. and Kamaruddin, S. (2014). Maintenance policy optimization—literature review and directions. *The International Journal of Advanced Manufacturing Technology*, 76(5-8):1263–1283.

[Fraser et al., 2015] Fraser, K., Hvolby, H.-H., and Tseng, T.-L. B. (2015). Maintenance management models: a study of the published literature to identify empirical evidence. *International Journal of Quality & Reliability Management*, 32(6):635–664.

[Hoogendoorn, 2020] Hoogendoorn, D. (2020). Condition based maintenance with event and usage data at Canon Production Printing. Technical report.

[Moubray, 1999] Moubray, J. (1999). *Reliability-Centred Maintenance*. Elsevier, 2nd edition.

[Nakajima and Bodek, 1988] Nakajima, S. and Bodek, N. (1988). *Introduction to TPM: Total Productive Maintenance (Preventative Maintenance Series) (English and Japanese Edition)*. Productivity Pr, eleventh printing edition.

[Narayan, 2005] Narayan, V. (2005). *Effective Maintenance Management*. Industrial Press Inc., Amsterdam, Netherlands.

[Patil et al., 2021] Patil, A., Soni, G., Prakash, A., and Karwasra, K. (2021). Maintenance strategy selection: a comprehensive review of current paradigms and solution approaches. *International Journal of Quality & Reliability Management*, 39(3):675–703.

[Sachs and Neville, 2007] Sachs, N. and Neville, W. (2007). *Practical Plant Failure Analysis*. Taylor & Francis, Abingdon, United Kingdom, 2nd edition edition.

[Shafiee, 2015] Shafiee, M. (2015). Maintenance strategy selection problem: an MCDM overview. *Journal of Quality in Maintenance Engineering*, 21(4):378–402.

[Smith and Mobley, 2003] Smith, R. and Mobley, K. (2003). *Industrial Machinery Repair*. Butterworth-Heinemann, first edition edition.

[Tiddens, 2018] Tiddens, W. (2018). Setting sail towards predictive maintenance : developing tools to conquer difficulties in the implementation of maintenance analytics. *Tools4LCM project*.

[Tiddens et al., 2018] Tiddens, W. W., Braaksma, A., and Tinga, T. (2018). Selecting Suitable Candidates for Predictive Maintenance. *International Journal of Prognostics and Health Management*, 9(1).

[Tinga, 2013] Tinga, T. (2013). *Principles of Loads and Failure Mechanisms*. Springer Publishing, New York, United States.

[Tinga et al., 2020] Tinga, T., Wubben, F., Tiddens, W., Wortmann, H., and Gaalman, G. (2020). Dynamic maintenance based on functional usage profiles. *Journal of Quality in Maintenance Engineering*, 27(1):21–42.

[van Elderen, 2016] van Elderen, C. (2016). Multi-item maintenance optimization based on failure, usage and condition information. Technical report.

[Waeyenbergh and Pintelon, 2009] Waeyenbergh, G. and Pintelon, L. (2009). CIBOCOF: A framework for industrial maintenance concept development. *International Journal of Production Economics*, 121(2):633–640.

[Zonta et al.,] Zonta, T., da Costa, C., da Rosa, Rand de Lima, M. J., da Trindade, E., and Li, G. P. date = 2020-12, d. . j. j. . C. l. . N. p. . . p. . E. t. . P. u. . h. v. . .

B

Appendix B: Failure mechanisms

Although designers and manufacturers have been evaluated and tested many times, the properties of a material deteriorate over time, which is inevitable. The rate of deterioration depends on many factors and each component will fail eventually. This chapter explains the failure mechanism. The book Sachs and Neville, 2007 and Tinga, 2013 gives a detailed explanation of the failure of machines, systems and components. This chapter provides a summary of that book supported by other papers.

B.1. Basic failure

Before moving on to the primary failure mechanism, it is necessary to understand the basic failure mechanism. Each material has different properties, which causes a material to break at a different times. Stress causes the components to bend and break, but each material reacts differently.

First, every material is somehow elastic. When a force is applied to a material, the material will stretch like a spring. The amount of strain is proportional to the force applied to the yield strength. When the applied stress is lower than the yield strength, the material will bounce back to its original geometry when the force is released. When the stress is higher than the yield strength, permanent deformation occurs. The material is plastically deformed. Ultimately, a rupture would occur if the tension exceeds the tensile strength. A visualisation is shown in figure B.1.

Another property of a material is its toughness of a material. A material that has the ability to absorb sudden energy is tougher than materials that cannot absorb such impact. Therefore, the material that absorbs much energy has higher toughness than materials that shatter, like glasses.

The choice of a material or component is almost always determined and tested during the design phase. Engineers ensure that all components remain in the elastic range, also known as the young modulus of a

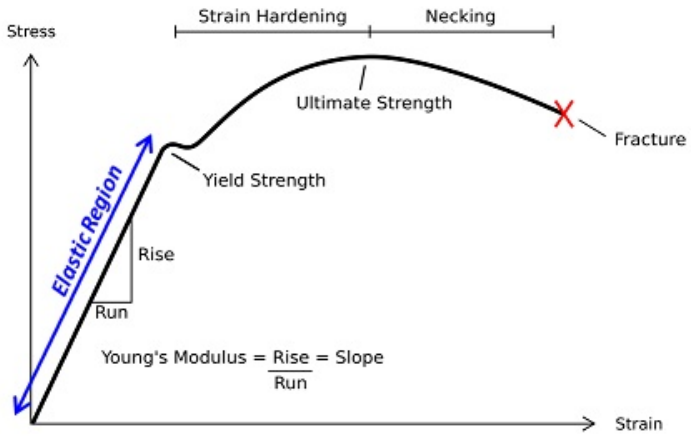


Figure B.1: Stress strain curve (Nipun, 2015)

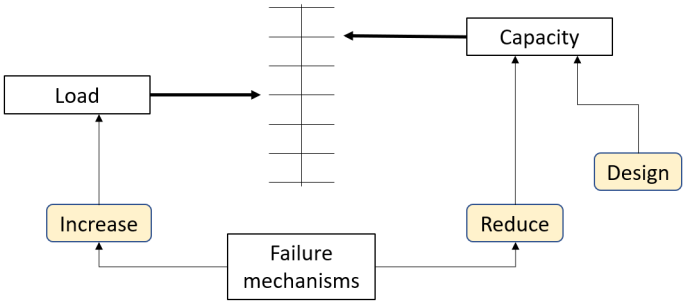


Figure B.2: Schematic representation of the relation between forces and capacity. A modification of Tinga, 2013

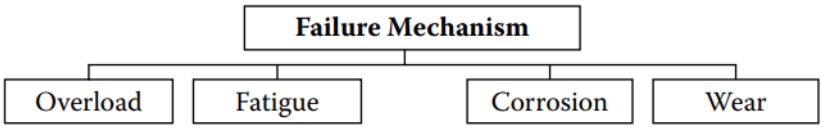


Figure B.3: The four primary physical failure mechanisms according to Sachs and Neville, 2007 and Bloch and Geitner, 2012

material, during the operation. In addition, safety and service factors are multiplied over the calculation to ensure that the material remains far below the yield strength to prevent failure from random external forces, and plastic deformation will be prevented from external forces. Therefore, the capacity is calculated and determined during design, and the actual load applied should be lower than the capacity.

While it means that the design meets the requirements of the applied forces of the machine, it does not mean that the machine is prevented from failing. Material ageing and other failure mechanism cause the properties to deteriorate the yield strength. Therefore, the capacity of the material reduces over time. A random external force or operating force could then be too high, and a failure of the material occurs. The factors of the failure mechanisms will be explained and Figure B.2 shows a schematic representation.

B.2. Human errors

First of all, humans could be the root error during designing, manufacturing, maintenance and operations. Every person does it his way, and every person can make random, systematic or sporadic mistakes that accelerate the rate of deterioration of materials (Narayan, 2005). A good statistic or documentation on the frequency of human error in an industry is generally hard to find (Sachs and Neville, 2007). Trying to understand the reasons for these errors is an endless task and different per company or machine. Monitoring human error is almost impossible due to human unpredictability and unique factors of what humans can do wrong. Nevertheless, people's mistakes could lead to accelerated deterioration of failures.

According to the book of Sachs and Neville, 2007, detailed failure analysis was done on 131 failure analyses, and from these 131 analyses, there was a combination of four primary physical failure mechanisms. Figure B.3 shows the ultimate failures. This figure shows the primary failure mechanisms, most of which have multiple roots and factors, such as human error or environmental effects. For example, Human error causes an overload failure of the machine, or an increased temperature causes more energy dissipation, which accelerates the rate of fatigue and wear and tear.

B.3. Overload failure

Overload failure almost always occurs instantaneously. Sudden high stress on a material breaks into pieces or plastically deforms. This concerns whether the material is brittle or ductile. A brittle material breaks instantaneously, and a ductile material deforms plastically first until the surface becomes too narrow that it breaks. That depends on the material and its specific properties. However, the temperature can cause the properties of the materials to change their ductility and brittleness. The properties of ductility or brittle metals change as the temperature increases. In the range between -75 degrees and 250 degrees Celsius, the yield strength

and tensile strength will not change that much. However, the toughness can deteriorate by more than ten times, turning a ductile metal into a more brittle metal.

During the operation of the machine, the operating cycle of a machine remains more or less the same. Therefore, it is implausible that a significant overload will occur, causing a component to overload failure. However, overload failure could occur during the machine's first start of peak season. Due to the non-operation period, components have been exposed to the environment, such as temperature, radiation, humidity, and dust. As a result, components have deteriorated during the non-operation period, and properties such as brittleness and young modulus may have deteriorated. Therefore, during the start of the operation period, machines need to provide extra force due to the brittleness of a component. With the change of geometry of material, overload failure occurs.

B.4. Fatigue

Fatigue causes more mechanical failure than any other mechanism. Although overload failure occurs during one high-stress application, fatigue failure is generally a lower force cycle than yield strength. Fatigue needs many stress cycles to realise a failure. Fatigue is described as failure due to a cyclic load of components (Comsol, 2016). Fatigue occurs even when the peak of the cyclic load is far below the static material strength. During many cycles, a crack is initiated on a microscopic level. The crack grows for each cycle until it reaches a critical length. Eventually, the component breaks because the component can not withstand the peak load.

Fatigue can be a function of different variables, such as stress, strain or energy dissipation. A load cycle is defined as the duration of one cycle of a machine. In an ideal situation, it can be measured straightforwardly from one peak to the next peak. A straightforward cycle is shown in fig B.4, where stress is the fatigue-controlling state variable. The most critical parameter for fatigue damage is the stress amplitude. Under the influence of a non-constant external load, the state in the component also varies with time.

Fatigue testing can be time-consuming, as a single test can run for many cycles before fatigue can be observed. Each of the same components is inhomogeneous on a micromechanical level, which results in scattering of failure. Therefore, some machines can fail much earlier than others, and it is unreliable to review components only during time intervals.

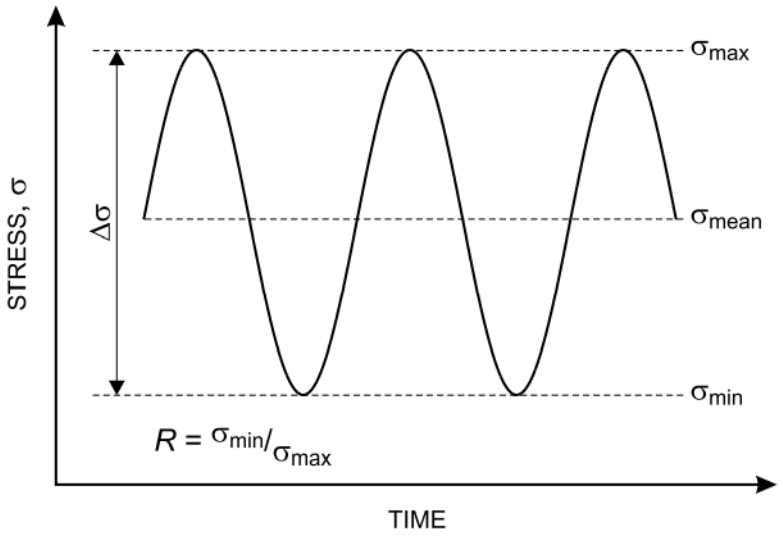


Figure B.4: A load cycle example from Janssen et al., 2004

B.5. Corrosion

Of the four failure mechanisms, corrosion is probably the most expensive failure. The leading cause of corrosion is the result of electrical currents and electrochemical reactions. Together with temperature, exposure time, moisture, liquid conductivity, pH, oxygen availability, solution contaminants, flow velocity, surface condition, and other variables, they affect the corrosion rates. Therefore, the reactions are highly complex and depend on many variables. Because of all these variables and their interactions, it is complicated to perform experiments to reproduce the actual field corrosion rates accurately. The effects of the corrosion rates and the different types of corrosion can be found in the book Sachs and Neville, 2007 and Tinga, 2013.

B.6. Wear & Tear

Wear is the undesired removal of solid surfaces due to the relative motion between the surface in contact with another surface (Jiménez and Bermúdez, 2011). The relative motion causes losses of the volume from both surfaces in moving contact. On a microscope scale, smooth surfaces are never truly smooth. Any surfaces have sharp, rough and rugged projections (see fig). The roughness on the surface is called asperities. By a relative motion between two surfaces in contact, the tips of the softer asperities could plastic deform, and particles of the tip will break from the surface, which will release as wear debris particles. Surfaces continue to be damaged by prolonged motion on the contact points. Eventually, the surface damage can be visualised by more significant volume losses.

Despite the importance of wear on machines, developing a universal model to describe the physical mechanisms is challenging. The process of asperities is complex because surfaces on a microscopic level look different and have all kind of different shapes. As a result, every contact point and surface roughness are different. In addition, wear processes can be classified into different types. Generally, there are three primary types of wear on solid surfaces; sliding/adhesive wear, abrasive wear and chemical or corrosive wear.

Adhesive wear is the relative motion between two surfaces. Removing parts are either permanently or temporarily attached to the other surface. Having two identical materials slide over each other, both surfaces will become roughened. The asperities on both surfaces remove and stick to the other surface. Both surfaces deteriorate and wear will occur. By having two different materials that slide over each other, the particles of the softer material will be attached to the harder material. In this instant, wear will be largely confined to the softer surface.

Abrasive wear is the removal or displacement of the asperities by the relative motion of two hard surfaces if adhesion is not dominant. This occurs when the asperities have the geometry of any cutting shape. The sharp surface abrades the harder surface, or plastics deform the asperities from the harder surface.

Erosive wear is the loss of asperities due to relative motion between a surface and particles in a fluid medium. By shooting particles against the surface, the surface will be damaged and the asperities will deform and be removed from the surface. The angle of incidence of the particles on the surface determines the degree of wear.

A simple and frequently used model for wear is the Archard wear equation. This model derives an equation for the wear rate as the volume of a surface removed per unit sliding distance, assuming that the wear volume is proportional to sliding distance. The second assumption of this model is that wear only occurs by the contact points of the asperities and the assumption that the shape of an asperity is a halved sphere. Therefore, asperities are the sum of all individual asperities in contact with an area surface of

$$\frac{2}{3} \cdot \pi \cdot r^3$$

Thirdly, this model assumes the sliding spherical asperities to deform fully plastically in contact. Therefore, the Archard equation for one sliding spherical asperity is expressed as:

$$Q = k \cdot W$$

Where

- Q is the wear rate, volume loss per sliding distance $[\frac{mm^3}{d}]$
- k is the wear coefficient or specific wear rate which is the volume per asperity of sliding distance divided by the hardness of the material $[\frac{mm^3}{N \cdot m}]$

$$\frac{\frac{2}{3} \cdot \pi \cdot r^3}{2 \cdot r} / H$$

- W is the normal force [N]

Although Archard's equation was developed for adhesive wear, it is widely used for modelling other types of wear processes. A paper (Meng and Ludema, 1995) has identified many equations for different types of wear. These equations will not be further elaborated on in this report because they have all been measured and applied on a microscopic scale of micrometres which is too small and detailed for the machine's components. Nevertheless, the Archard equation shows that more contact points of the asperities will touch each other by applying a high normal force on the two surfaces. This means that more wear processes occur because more contact points are in touch by a greater normal force of the surfaces (see Archard equation).

In addition to the Archard model, which is based on normal force and distance travelled, the cause of wear failure depends on the consideration of many factors. Various other elements influence whether and how fast wear takes place. By operating machinery exposed to harsh environmental conditions, stress and fatigue, the machinery influences additional wear behaviour. Over time, cracks will initiate, or the geometry of the material has deformed.
The crack becomes significant, and the surfaces are no longer perfectly aligned. It becomes looser between the materials. Misalignment of the materials can cause more force, and shock loads will arise, accelerating the wear behaviour. Ultimately, the component will fail in a fracture.

A complicating factor affecting wear and tear was considered a fracture phenomenon caused by repeated loading, defined as fatigue. Fatigue failure is a progressive and localised process. Eventually, the crack's growth becomes unstable, and a fracture occurs. With the cyclic stress range, the crack front will move into the material to the crack propagation law. In the direction of 'a', the rate of propagation is given by:

$$\frac{da}{dN} = A(\Delta\sigma \cdot \sqrt{a} \cdot y)^{m}_{gr}$$

Where

- $\frac{da}{dN}$ is the crack growth rate [m/cycle];
- a is the crack length [m];
- $\Delta\sigma$ is the nominal stress range (normal stress);
- A is the fatigue crack growth rate (FCGR) material constant;
- m_{gr} is the crack growth rate exponent;
- y is the crack geometry factor depending on the crack shape, orientation and surface boundary dimensions.

The stress intensity range (ΔK), which equals $\Delta\sigma \cdot \sqrt{a} \cdot y$, A and m are obtained through experimentation and measurement.

Components affected by wear and wear often have lubricants to reduce wear rate. The first concept to understand lubricants is the lambda ratio. The lambda ratio is the thickness of the lubricant film and is a comparative measurement of the separation of two surfaces and their relative roughness. As the separation between two surfaces increases, lambda increases and the wear rate decreases. Therefore, it is possible to monitor the lubricant to determine the condition of the wear rate.

B.7. Electrical failures

Electronic components and devices have a major impact on the operating of machines. Compared to mechanical systems, the failure of electronic components are rather complex Tinga, 2013. A electronic component or device consists of a large number of individual components. Therefore, it is hard to measure or estimate the loading of each component.

The major electric failure mechanism is due to an overload of electric current that leads to overheating. Consequently, individual components in electronic devices are melting, evaporating or cracking. An overload of electric current could cause by a short circuit. A short circuit is a connection between a low resistance with two conductors. Short circuits result in excessive current in the power source. Short circuits can occur due to many factors, causing the machine to fail.

In addition, the electrical components fail when the expected output is not equal to the actual output, causing the component not to function correctly. Due to shifts or improper calibrations, the output gives

Table B.1: Tools for determining the root cause failures from Narayan, 2005

Tool(Abbreviation)	Description
Fault Tree Analysis (FTA)	Graphical representation of the relationship between the causes of failure and system failure
Fishbone (Ishikawa) Analysis	Helps identify proximate causes
Stair-step Analysis	A sequence of 'Why' questions, beginning with the failure consequence
Failure Mode & Effects (and criticality) Analysis (FME(C)A)	A table with the failure modes with the causes and its frequency, severity and detection
Pareto and degrader analysis	Based on the 20% of the failures are responsible for 80% of the maintenance costs, or total downtime
Root cause analysis	Only addressing the root cause of a problem

different values than the actual conditions. Although electric components could still operate, undesired values could cause an unplanned breakdown, in which desired values are necessary to let the machine operates normally. Therefore, maintenance is needed to repair the electric components.

Due to the complexity of electrical failure, which consists of many more factors that lead to failure, electronic parts are often considered a random process that cannot be predicted to any failures. Therefore, the statistical approach based on past experiences is generally expressed as the mean time between failures (MTBF) with large uncertainties. The drawback can be circumvented when physical models for the relevant failure mechanisms are applied to predict the service lifetime of each component on the electronic parts at given operating conditions. However, the large number of components makes this approach quite infeasible for electronic parts.

B.8. Software failures

Lastly, software failure for controlling the machine can occur during operation. Software is a general term, and there can be unique failures per software program. Such failures require IT experts, to fix the possible failures. The software is not thoroughly inspected during scheduled maintenance unless specific problems occur due to a notification or reaction. For software failure, IT experts are called to solve such software problems. The service mechanics do not have the expertise to solve the failure. Therefore, software failure is mainly based on reactive maintenance and software failures will not be discussed further in this report.

B.9. Failure analysis tools

Multiple specific factors cause the components to fail into the four primary physical failures, material ageing, human errors and electrical failures. Other failure mechanisms may still exist for specific components. To better understand the failures and factors, a so-called failure analysis can be conducted to dig deeper into particular factors that affect the rate of deterioration of the components. There are different methods for applying a failure analysis. Table B.1 shows the possible tools for determining root causes or failures per component. A failure analysis will be more straightforward if the history of the component and a machine, from design to operation, is documented. Therefore, data needs to be collected from maintenance reports, failure observations and daily reports to better understand the failures and possible factors that were present.

B.9.1. Failure Mode & Effects Analysis (FMEA)

FMEA is a systematic method to map systems or machines' failures, effects, and causes Tinga, 2012 Peeters et al., 2018. The FMEA is often carried out with experts from a diverse team of people with various backgrounds. Understanding the material properties, designs, software, operations and maintenance records increases the probability that all possible failures are identified and the effects are appropriately estimated. By adding a criticality analysis, FMEA is extended to an FMECA. The criticality of each failure is quantified by the risk priority number (RPN), which consists of occurrence, severity and detectability values. The results of both FMEA and FMECA are recorded in a table.

B.9.2. Fault tree analysis (FTA)

FTA is an alternative method to investigate failure behaviour Tinga, 2012 Peeters et al., 2018. The fault tree analysis (FTA) was employed to establish potential root causes of failures on critical components. It uses

logic gates and events to show the state of components related to the state of the system. FTA is a visual failure analysis tool to outline the causes of undesired results and shows the underlying system levels. FTA uses a top-down approach, starting at the top and branching downwards to review each underlying system level. The main event starts at the top and draws branches representing the main event's causes. Each branch is further divided into next level branches that represent its causes.

B.10. Conclusion on failure mechanisms

In general, there are four main mechanical failures, human errors and material ageing. In addition, depending on the components, electrical and specific errors could occur. Different effects and root causes result in the failure mechanism's deterioration rate. Figure B.5 gives a clear visualisation of the failure as mentioned earlier mechanisms and the possible factors which affect the rate of deterioration failure mechanisms. Using the possible methods from table B.1 makes it possible to find factors as the root causes. This allows experts to prevent or minimise these root causes, reducing the rate of deterioration and making a component failure more predictable or extending the lifetime. In addition, figure B.5 gives an idea of the possible related variables to the failure mechanism. The parameters can be monitored and evaluated by experts during maintenance to determine the condition of components.

Deciding how deeply to dive into the failure analysis depends on the possible consequences, safety and cost. The factors can again be subdivided into possible causes and further subdivided into the ultimate root causes. Collecting maintenance reports, failure observations, and documentation makes analysis more accessible and realistic to find the root causes. The frequency and impact of a failure are more likely to be accurate.

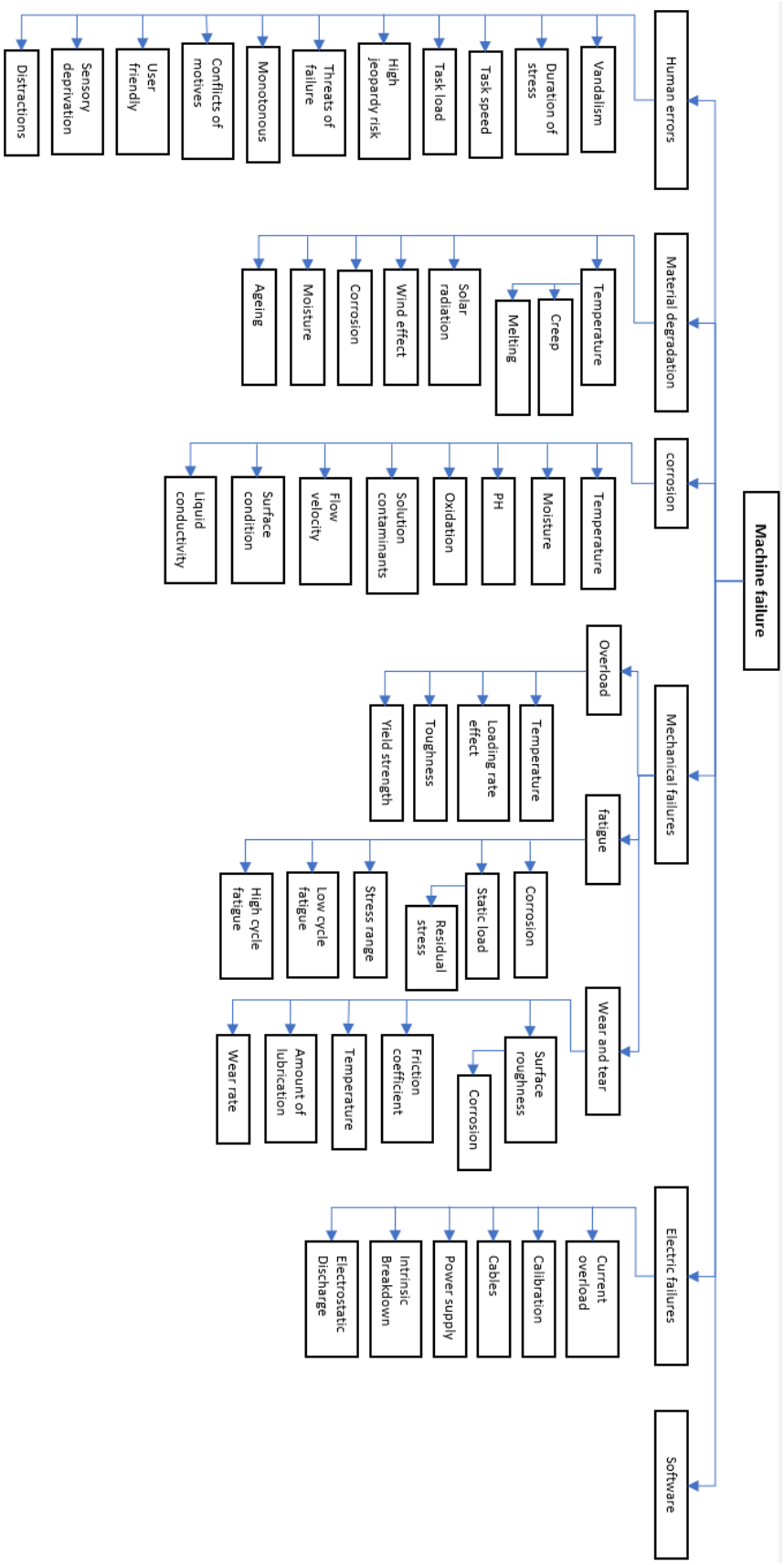


Figure B.5: Failure mechanisms and its factors which lead to a failure. Source: in combination of Sachs and Neville, 2007, Moncmanová, 2007 and Narayan, 2005.

C

Appendix C: Autostix Function

Appendix C elaborates on the usage of the machine. First of all, the Autostix machine is depicted. It follows the focus on the gripper beam and its function.

C.1. Autostix function diagram

The function diagram of Autostix machine is shown in Figure C.1.

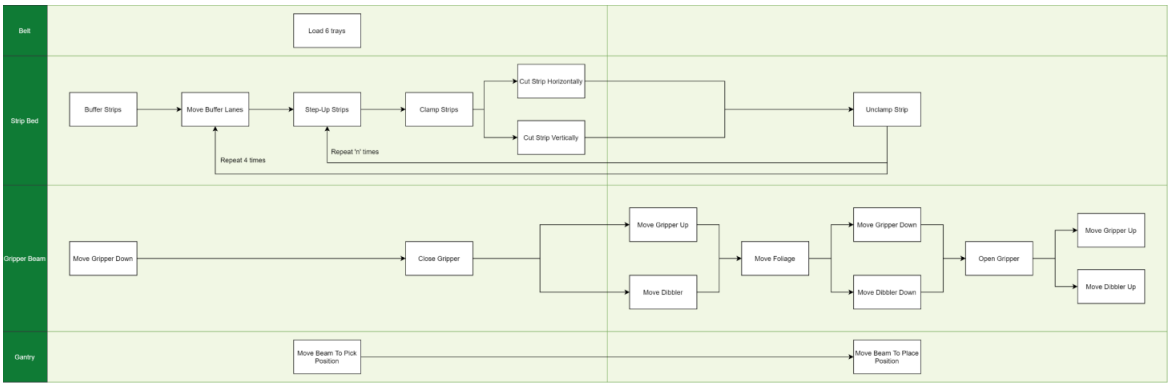


Figure C.1: Autostix Function Diagram made by VTP

C.2. Gantry system

The following system is the processing part of the Autostix machine. The machine has a gripper beam which consists of the gantry system and grippers. The gripper beam is the most crucial system of the Autostix machine. The gantry system moves to transplant plants or cuttings from the input system to the output system.

A gantry system, also called X–Y table consists of a two-axis motion control system that allows grippers to move in two directions in a single plane. Every position on the XZ plane can be reached through the interaction of both axes. Each Model machine has a gantry system that allows moving the plants from the input to the output of the machine. The gantry system uses two servo motors to control the XZ coordinates. It accelerates and decelerates continuously in short periods of time to move back and forth from the input to the output. Two servo motors are rotating for the movement of the gantry system. A servo motor on both sides of the machine is coupled to two shafts with bearings. The shafts transmit the rotation of the servo motors to the drive pulley. Two drive pulleys rotate an open-end gantry belt to move the gantry along the XZ plane. The open-end gantry belt is guided by four idler belt pulleys, one idler pulley with teeth at the top of the gripper beam, and linear bearings for horizontal and vertical movements. Therefore, combined with the direction of rotation of the drive pulleys and the components mentioned above, the gantry can be lifted and moved horizontally simultaneously. The gantry system consists of the most moving components, and the components are all connected to reach an accurate position and smooth the movements.

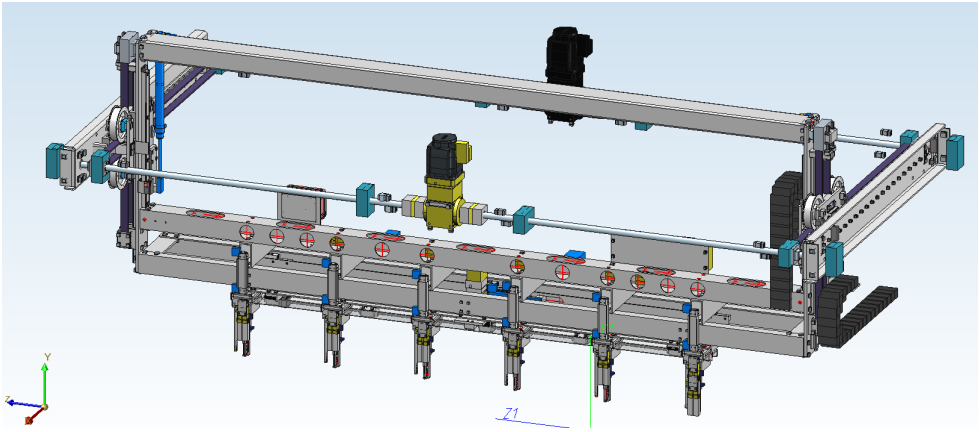


Figure C.2: Gripper beam

The grippers are attached to the gripper beam to grab and release cuttings from the input to the output. Different grippers have been designed, but they have the same purpose: to grab, hold and release a plant or a cell of the strips. The model machine consists of two L profiles that push against each other to grab cells from the input system by a pneumatic cylinder. In addition, the grippers contain a dibbler knife to prick in the soil of each pot of trays. The grippers release the cell on the pricked spot of each pot or tray. Due to different tray configurations, the distance between the dibbler knife and grippers is movable through a servomotor to linear bearings and rack and pinion because the grippers release the cuttings in the first row of pots. In contrast, dibbler knives prick the next row of pots. Therefore, grippers do not have to wait till all pots or a tray have been pricked. The distance between grippers and dibbler knives is the distance between pots next to each other on a tray. Figure C.2 shows the system of the gripper beam.

The gantry system mainly consists of multiple of two identical components. However, these components depend on each component. This means that a failure on one of the components affects the whole gantry system in a breakdown. In contrast, the grippers function independently from each other.

C.2.1. Gripper beam movement

Due to two different combinations of each motor direction, the gantry system can move in four different directions. Movements of the gantry and the rotational directions of the motors are shown in figure C.3.

The movement profile of the gantry system is shown in Figure C.4.

The total average distance per cycle for pick and place no side shifts:

$L_{travel} = 2 \cdot (15 + \frac{1}{4} \cdot 2 \cdot \pi \cdot 15 + 500.5 + \frac{1}{4} \cdot 2 \cdot \pi \cdot 15 + 125.5) = 1376,25mm$

The speed and acceleration profile was not given. However, the maximum speed and acceleration were given by the motor. Figure C.5 shows some technical data about the maximum velocity and acceleration

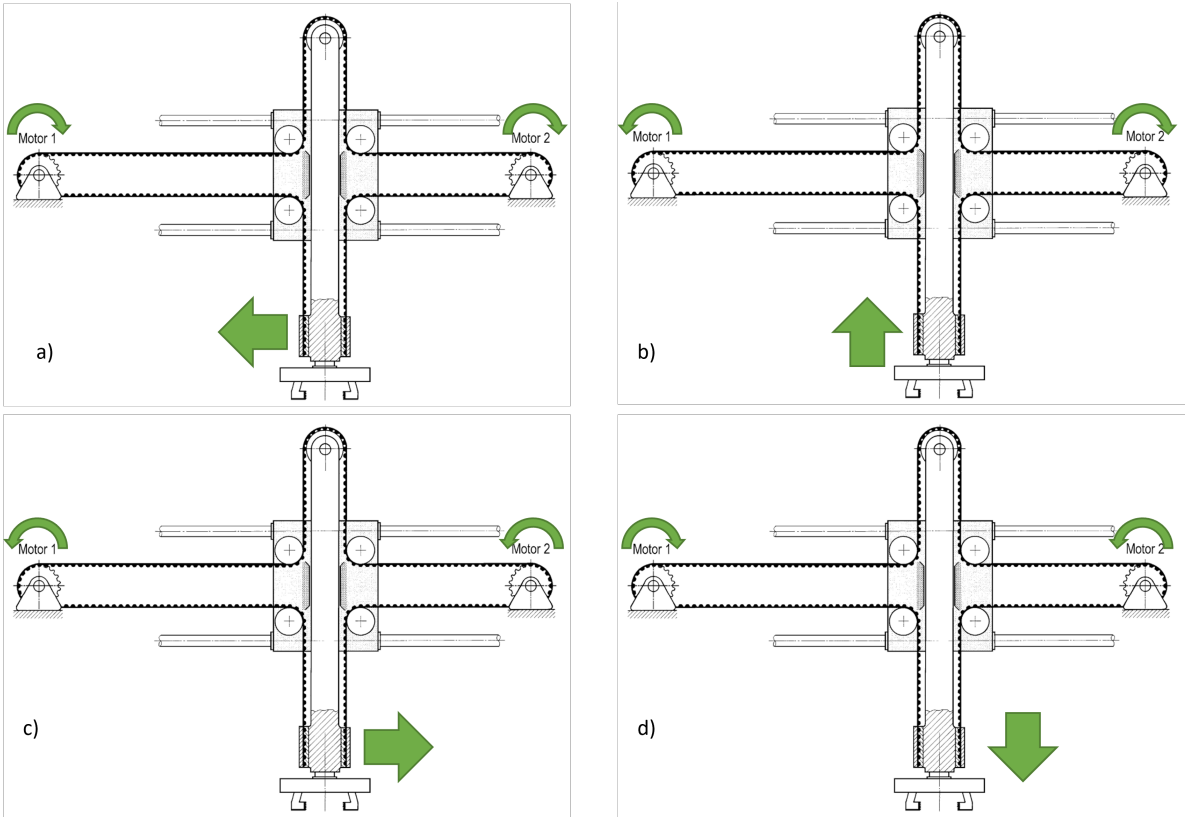


Figure C.3: Movement of the gantry system. a) The gantry moves to motor 1. b) The gantry moves upwards. c) The gantry moves to motor 2. d) The gantry moves downwards.

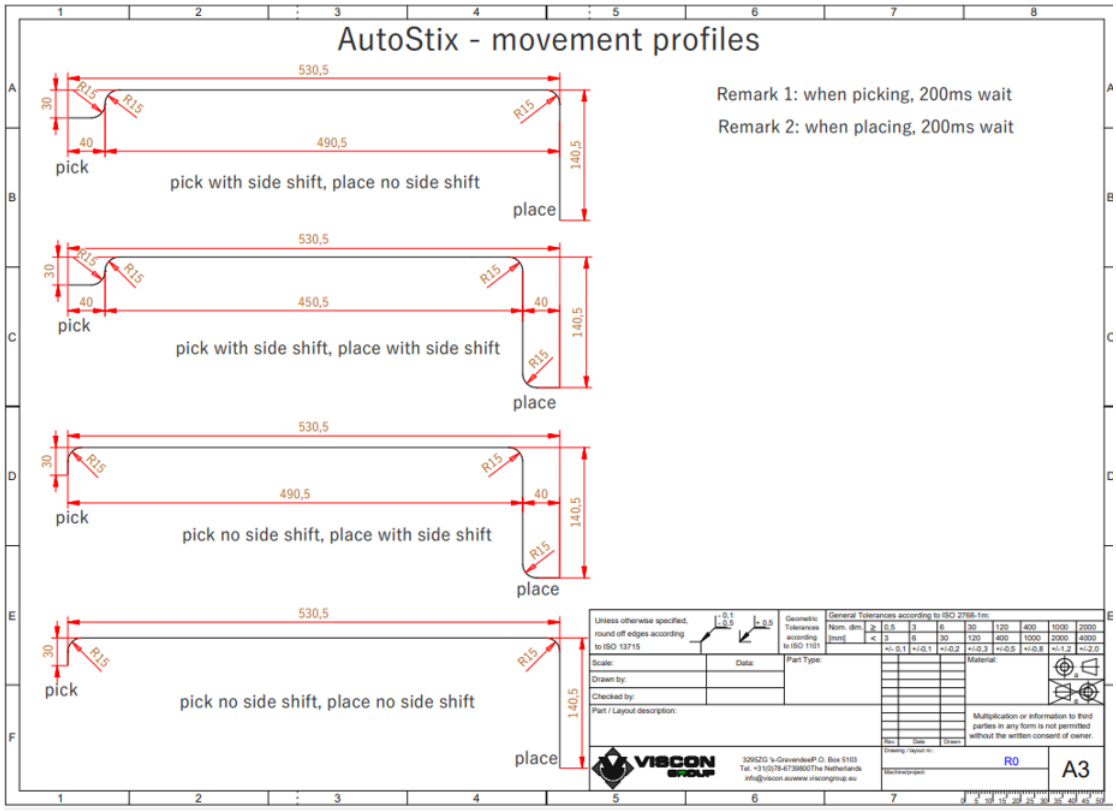


Figure C.4: Movement profile of Autostix. We used the fourth movement profile with no side shifts in the use case situation.

C.3. Technical aspects of the gantry system

The usage data is given in Figure C.5

Mode	Percentage of the maximum capacity	Capacity per hour	Capacity per gripper per hour	Capacity per gripper per sec	Time per cycle [sec]	Waiting time [sec]	Total time duration per cycle[sec]
0	0	0	0	0	0	0	0
1	30%	3000	500	0,138888889	7,2	0,4	6,8
2	50%	5000	833,3333333	0,231481481	4,32	0,4	3,92
3	80%	8000	1333,333333	0,37037037	2,7	0,4	2,3
4	100%	10000	1666,666667	0,462962963	2,16	0,4	1,76

Mode	Avg RPM without waiting time	Max velocity[m/s]	Max acceleration [m/s^2]	Avg RPM with waiting time	Waiting per minute
0	0	0	0	0	0
1	114,7058824	0,33	5,19	108,3333333	3,333333333
2	198,9795918	0,55	8,65	180,5555556	5,555555556
3	339,1304348	0,88	13,84	288,8888889	8,888888889
4	443,1818182	1,1	17,3	361,1111111	11,11111111

Figure C.5: Technical aspects of the gantry system

The input and calculation data of the belt is given in Figure C.6 and C.7.

Drive data		Input	
Mass of load	Mload	120	kg
Mass of counterweight	mc	0	kg
Acceleration	a	17,33	m/s²
Deceleration	b	17,33	m/s²
Friction	μ	0	
Belt pitch	p	LL 8MR ST	mm
Number of belts	n	2	
Pulley size	z	24	
Shaft diameter	ds	25	mm
Pulley material		ST	
Required Service factor	SF	1,60	
Center distance	cd	2171,5	mm
Free Span Length ****	SL	750	mm
Angle of elevation	ß	90	Degree

Figure C.6: Belt specification

Calculation data		
Highest tooth load incl. pulley- and belt mass *	1644	N
Allowable Working Tension at width factor 1	1568	N
Required width factor	1,05	
Required width factor incl. Sf	1,68	
Required width	31,5	mm
Next Standard width	30	mm
Pulley pitch diameter dp	61,12	mm
Pulley mass	1,07	kg
Reduced pulley mass	0,63	kg
Belt mass	1,01	kg

The smaller standard belt width will be chosen if the required belt width is slightly above the smaller standard belt width.

Figure C.7: Drive data

D

Appendix D: Technical data collection

For this project, the use case of the gripper beam from the Autostix machine was used. This appendix shows how the data related to the service life was obtained. However, it is less accurate due to the usage and environmental changes. The service life for components in the gripper beam has been calculated and collected. The data required for the service life is given below. The service life of components is either calculated with formulas, experiments or given by the component's OEM.

D.1. Based on experiments

Components based on experiments are components that are not purchased or ready-made from an OEM. These components are designed and built for specific machines. In addition, these are the components that are less commonly implemented in industrial machinery. Therefore, OEMs do not have any data or experience for those components. Examples are the gripper profile and pricker plate. These components are tested by the company of the machine builders themselves. The components are tested until they fail. From experiments, people who tested it can tell something about it.

D.2. Based on experts

Expert-based components are components whose data is provided by the OEM of the component. They have their tools and experiences as to how they got this data. The OEM provides the data sheets of technical and maintenance information. However, the distributions of the failure are usually unknown.

Another alternative is based on experts. Usually, they need additional information before they can determine the service life. For example, the gantry belts were purchased by a company with a calculation tool that calculates the service life. The necessary input is often required to be able to calculate the service life. Therefore, the engineers must be well aware of the forces, velocity and accelerations or any additional conditions applied to the components. Some examples are shown in Figure D.1 and references (Bimba, 2012).

Size Calculation for Aluminium Roller Guides

We will be pleased to calculate the size of aluminium roller guide you require!

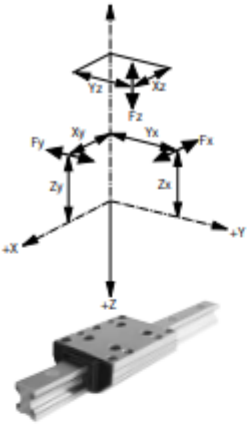
Copy this page, enter the requested technical data and send this page to your local technical adviser or to one of the contact addresses on the last page of this brochure.

Company	
Phone	
Telefax	
email	
Branch	
Department	
Name	
Date	

Order No.	

Technical data	
Stroke [mm]	
Mounting position	vertical ☐ horizontal ☐ angle ☐
Speed [m/s]	
Acceleration [m/s²]	
Lifetime (desired) L [km]	
Carrying length A [mm] *	
Carrying width B [mm] **	

* The distance from centre to centre of two cassettes / pairs of roller shoes on a rail
** With multi-track arrangement distance of the rails



Sketch

Loads, Forces, Lever arms	
F_x [N]	=
Y_x [mm]	=
Z_x [mm]	=
F_y [N]	=
X_y [mm]	=
Z_y [mm]	=
F_z [N]	=
X_z [mm]	=
Y_z [mm]	=

Environment: (Dirt. Humidity ...)

Figure D.1: An example of inputs for linear bearings in which the OEM can calculate the service life.

D.3. Based on calculations

Components based on calculations are components for which the formula has been given and generally applied for the calculations of the service life of the components. As an example, we use bearings. Bearings have well-known formulas to calculate the service life SKF, n.d. The bearing block is used as an example, and its calculations are described below.

The service life of the bearing can be calculated according to bearing rate life SKF, n.d.
$L_{10} = (\frac{C}{P})^p$
or
$L_{10h} = (\frac{10^6}{60 \cdot n}) \cdot L_{10}$
Where

- L_{10} = basic rating life (at 90% reliability) [millions of revolutions]
- L_{10h} = basic rating life (at 90% reliability) [operating hours]
- C = basic dynamic load rating [N]
- P = equivalent dynamic bearing load [N]
- n = rotational speed [r/min]
- p = exponent of the life equation [= 3 for ball bearings] [= 10/3 for roller bearings]

For example: Calculate the expected service life of the ball bearings (B_1, B_2, B_3) of UFCL 205 for the following situation with forces applied (see Figure D.2). This figure shows the drive from the motor on the left to the gantry belt with a shaft belt. The weight of the pulley is negligible low.

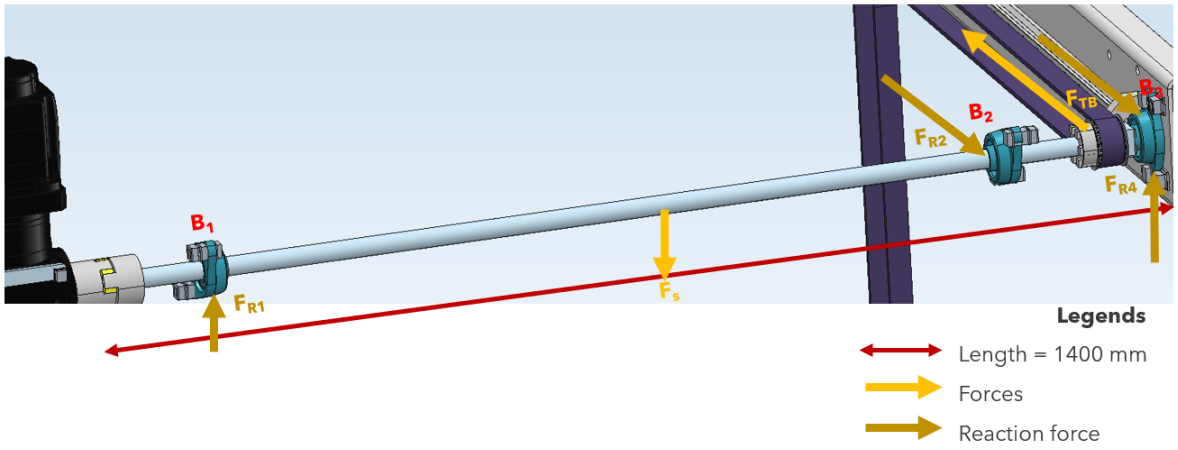


Figure D.2: Visual example of the bearing blocks on the shaft

For the dynamic load of the bearings due to the belt, a free body diagram has made and is shown in Figure D.3

The pulley load under dynamic conditions of the belt can be calculated by the following formula (Elatech, n.d.):
$F_{wdyn} = 2 \cdot F_{TV} + F_U$
in which
$F_U = F_{ab} + F_H + F_R = m \cdot a_b + m \cdot g + m \cdot g \cdot \mu$ F_U can be filled in from the given data of Figure C.7. By taking the worst-case scenario, the answer is $F_U = 120 \cdot 17.3 + 120 \cdot 9.81 + 120 \cdot 9.81 \cdot 0 = 3253.2N$
Where

- F_{wdyn} = Dynamic pulley load [N]
- F_{TV} = Pretension forces per belt side [N]
- F_U = Peripheral force [N]

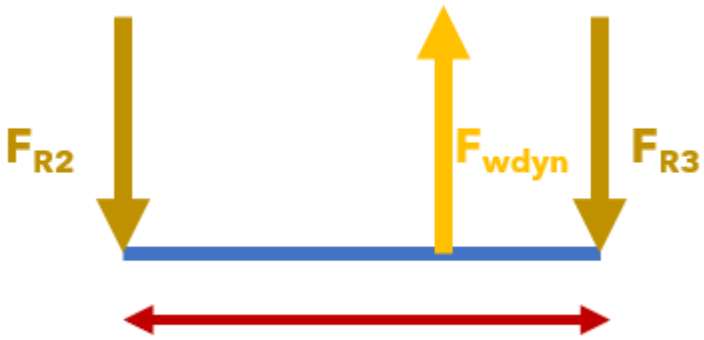


Figure D.3: Free body diagram of the forces due to toothed belt

- F_{ab} = Acceleration force [N]
- F_H = Vertical lifting force [N]
- F_R = Resisting force of friction [N]
- m = Total mass [kg]
- a_b = Acceleration [$\frac{m}{s^2}$]
- g = Gravity acceleration = 9.81 [$\frac{m}{s^2}$]
- μ = Friction coefficient

And for F_{TV} (Perneder and Osborne, 2012):
$f_e = \sqrt{\frac{F_{TV}}{4 \cdot m_{spez} \cdot b \cdot l_T^2}}$

- f_e = Fundamental frequency [Hz]
- F_{TV} = Pretension forces per belt side [N]
- m_{spez} = Specific belt mass [kg/mm belt width and per m belt length]
- b = Belt width [mm]
- l_T = Span length corresponding to the frequency [m]

The minimal fundamental frequency is given from the supplier and the frequency should be minimal 60Hz. Furthermore, the chosen timing belt from Figure C.6 and Figure C.7 consists of:

- Belt width (b) = 30 mm
- Belt mass (m_{belt}) = 1.01 kg
- Belt length (l) = 4.535 m
- Span length (l_T) at which the frequency is tested = 0.75 m

Therefore, the specific belt mass:
$m_{spez} = \frac{m}{l \cdot b} = 0.00742 kg/mm$
The span load can be calculated by:
$F_{TV} = 4 \cdot m_{spez} \cdot b \cdot l_T^2 \cdot f_e^2 = 1804N$
The maximum dynamic force on the pulley is:
$F_{wdyn} = 2 \cdot F_{TV} + F_U = 2 \cdot 1804 + 3253.2 = 6861.2N$

The dynamic load of $B_1, B_2 and B_3$ are only the peaks dynamic load. These bearings are continuously changing loads during one cycle, each different load level can be accumulated, and the load spectrum reduced to a histogram plotting constant-load blocks (SKF, n.d.). Each interval in a cycle should characterise a given percentage or time fraction during operation. Heavy and normal loads consume bearing life faster than light loads. Therefore, it is essential to have peak load well represented in the load diagram, even if the occurrence of these loads has a relatively short duration.

The bearing load and operating conditions can be averaged within each interval to a representative, constant value. The number of operating hours or revolutions expected from each interval within a cycle, showing the life fraction required by that particular load condition, should also be included. Under variable operating conditions, bearing life can be rated using D.4. This calculation method is well suited for application conditions of varying load levels and speed with available time fractions SKF, n.d.

$$L_{10m} = \frac{1}{\frac{U_1}{L_{10m1}} + \frac{U_2}{L_{10m2}} + \frac{U_3}{L_{10m3}} + \dots}$$

Figure D.4: Formula dynamic load with variable operating condition SKF, n.d.

Where

- L_{10m} = SKF rating life (at 90 % reliability) [million revolutions]
- $L_{10m1}, L_{10m2}, \dots$ = SKF rating lives (at 90 % reliability) under constant conditions of each interval per cycle [million revolutions]
- $U_1, U_2, \dots$= life cycle fraction under the conditions 1,2, ... $.U_1 + U_2 + \dots + U_n = 1$

A schematic sketch of the force of the bearing load with the time and its interval time per cycle profile is shown in Figure D.5. This shows that the peak dynamic load is only at intervals U5 till U6. Suppose the calculations are done in the same manner as the maximum dynamic load for the remaining intervals $(U_1, U_2, ..., U_n)$. In that case, the average cycle and service life of the bearings can be calculated. For this application, the numbers are filled in:

$$L_{10m} = \frac{1}{\frac{0.0926}{3608} + \frac{0.0463}{6300} + \frac{0.278}{5700} + \frac{0.116}{4300} + \frac{0.0926}{3608} + \frac{0.083}{6861} + \frac{0.255}{6400} + \frac{0.037}{3608}} = 5378.3N$$

The ratio at which the B_2 and B_3 towards the pulley is 1:2, in which B_2 is twice as far
Therefore the bearings have the following dynamic load:
$F_{R2} = \frac{1}{3} \cdot 5378.3 = 1792.8N$
$F_{R3} = \frac{2}{3} \cdot 5378.3 = 3585.5N$
In addition, the shaft gravity is held by B_1 and B_3. A free body diagram is shown in Figure D.6
Due to static load the sum of the force and moment is zero (Hibbeler, 2016)
$\Sigma F = 0 => F_{shaft} - F_{R1} - F_{R4} = 0$
$\Sigma M_{R4} = 0 => M_{R4} = F_{shaft} \cdot r_{shaft} - F_{R1} \cdot r_{R1} = 0$

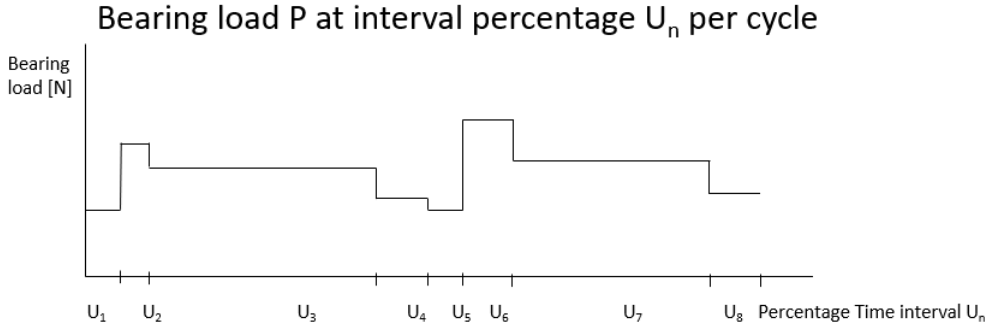


Figure D.5: Schematic sketch of the bearing load P and interval U_n per cycle

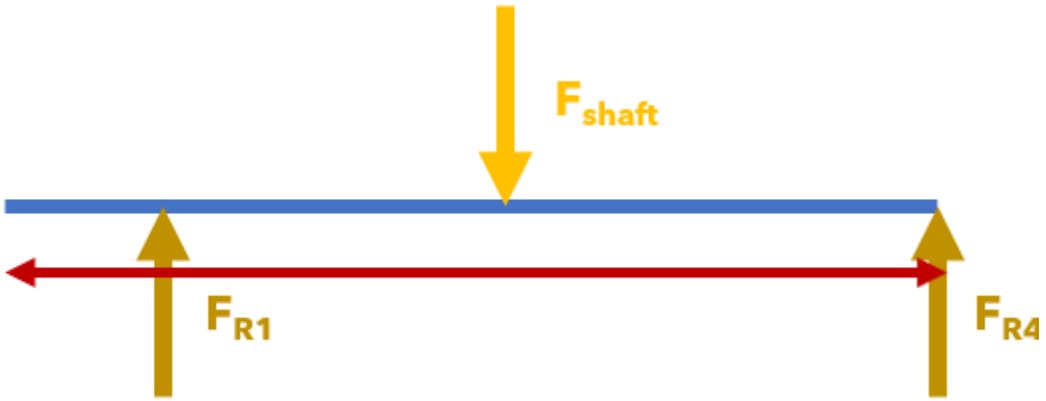


Figure D.6: Free body diagram of the forces due to the weight of the shaft

Fill in, and the answer is
$M_{R4} = 5.3 \cdot 9.81 \cdot 700 - F_{R1} \cdot 1280 = 0$
$F_{R1} = \frac{5.3 \cdot 9.81 \cdot 700}{1280} = 23.56N$
$F_{R4} = 52 - 23.56 = 28.43N$
Therefore, the dynamic load on the bearing is as follows:

- $B_1 = 23.56N$
- $B_2 = 1792.8N$
- $B_3 = \sqrt{3585.5^2 + 28.43^2} = 3585.6N$

The service life of bearings are
$L_{10,B_1} = (\frac{C}{P})^p = (\frac{14000}{23.56})^3 = 2.1 \cdot 10^6$million revolutions
$L_{10,B_2} = (\frac{C}{P})^p = (\frac{14000}{1792.8})^3 = 476.2$ million revolutions
$L_{10,B_3} = (\frac{C}{P})^p = (\frac{14000}{3585.6})^3 = 59.5$ million revolutions
These service lives are converted in hours or days when the operating capacity is known. These numbers are mentioned in the use case in Chapter 6.

E

Appendix E: Verification process of the maintenance optimisation model

In this appendix, the verification process of the maintenance optimisation model used in the proposed framework is shown. The verification of models is a fundamental part of modelling. This must be conducted in order to know that the proposed framework with its developed models can be used. For the verification, the focus lies on the following question:' Is the model right?' In other words, is the model that has been implemented according to the specifications?

First, the model must run without errors after the implementation. Hereafter the verification checks that have been conducted will be explained for each model. Below every verification check, the expected and actual results will be compared. These will be indicated in the verification checklist. The quantitative results are shown after the check. The maintenance optimisation model is used for the simulation and the service department programs.

E.1. Verification on the Block replacement model

In this section, we show a complete mathematical model of the Block replacement model. The Maintenance optimisation model of the proposed maintenance framework is based on the Block replacement model.The main difference is that the Maintenance optimisation model uses the setup cost as a variable factor and adds an algorithm to the Block replacement model.

First of all, the process of the Block replacement model in the implementation of the Maintenance optimisation model has to be verified. Based on the knowledge of the Weibull parameter, we know the Block replacement model schedules a high shape parameter later than a low shape parameter with the same scale parameter.

The cost difference in reactive maintenance and preventive maintenance will also cause different maintenance intervals. A large difference in reactive and preventive maintenance costs will lead to a small maintenance interval since the reactive cost is large. A small difference in costs will then lead to a large maintenance interval.

For the verification, we are taking four different components. The characteristics of the components are shown in Table E.2

The expected result between components one and two will be that component one has a smaller maintenance interval since a failure of component one is more expensive than component two. Also, comparing components three and four, component three will have a smaller maintenance interval.

Comparing the shape parameters, since component one and two has low shape parameter, the difference in the maintenance intervals is more than the difference between high shape parameters. A high shape parameter is more certain. Therefore, The change of maintenance intervals will be less than a low shape parameter.

The optimal cost per hour depends mainly by the total costs of reactive and preventive maintenance and the shape parameter. Comparing components one and two, component two will have the lower optimal costs per hour since the reactive cost is lower on component two than one. This also applies to components three and four. Furthermore, components three and four have a higher shape parameter than one and two. A

Table E.1: Verification of the Block replacement model.

Input						
Component number	Shape parameter	Scale parameter [hours]	Reactive cost [€]	Preventive cost [€]	Total hours	step size [hours]
1	2	5000	1000	10000	8000	15
2	2	5000	1000	2000	8000	15
3	10	5000	1000	10000	8000	15
4	10	5000	1000	2000	8000	15

Output		
Component number	Maintenance interval [Hours]	Optimal cost per hour [€/hour]
1	2700	1.971
2	15000	1.340
3	3450	0.773
4	3675	0.726

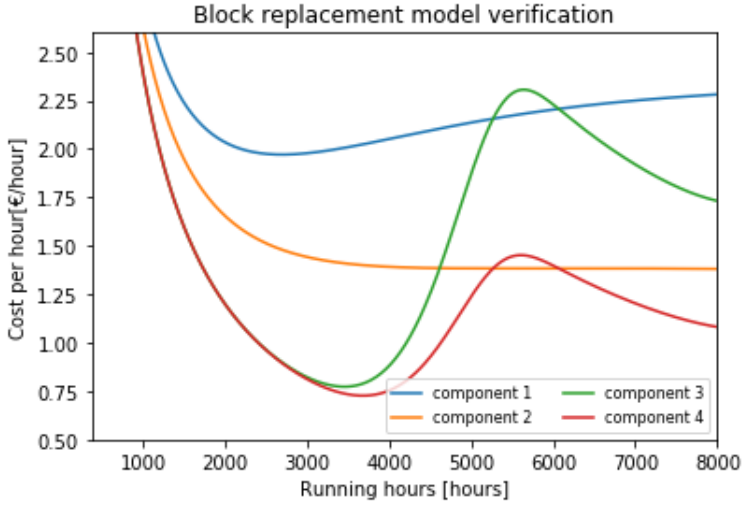


Figure E.1: Visualisation of the four components based on Block replacement model

high shape parameter gives a low scatteredness of failure, so it because more predictable and the chance for preventive maintenance will be higher. Therefore, component three and four will have a lower optimal cost per hour than components one and two.

Compared to the expected result, the results show expected values. Also, Figure E.1 shows no sign of errors. Comparing the four components, the maintenance interval order, its difference and its optimal costs are as expected. Therefore, we assume the BRM to be implemented successfully.

E.2. Verification on peak season machines

After implementing the BRM, we added an algorithm for the Maintenance optimisation model. For this verification, we used the same value of the Block replacement model (see Figure E.2. Changing the peak season interval resulted on peaks after the peak season interval. Figures E.2 showed different peak season intervals and it shows that the lines remain as FigureE.1 except the peaks. Therefore, the peak season intervals are correctly implemented.

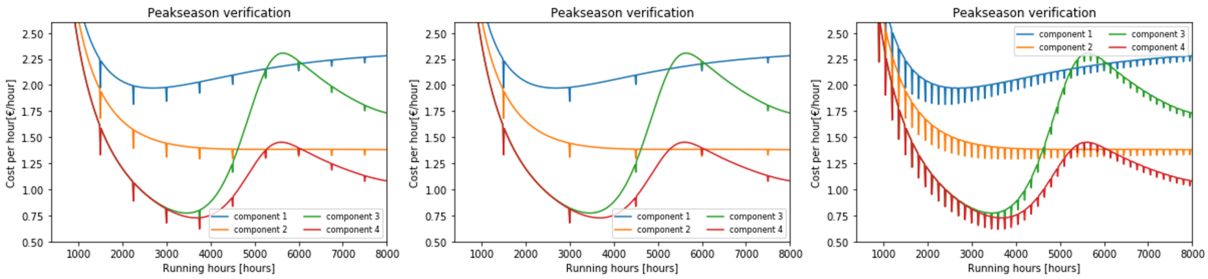


Figure E.2: The Block replacement model with peak season intervals. Peak season intervals: left = 750 hours, middle = 1500 hours and right = 75 hours

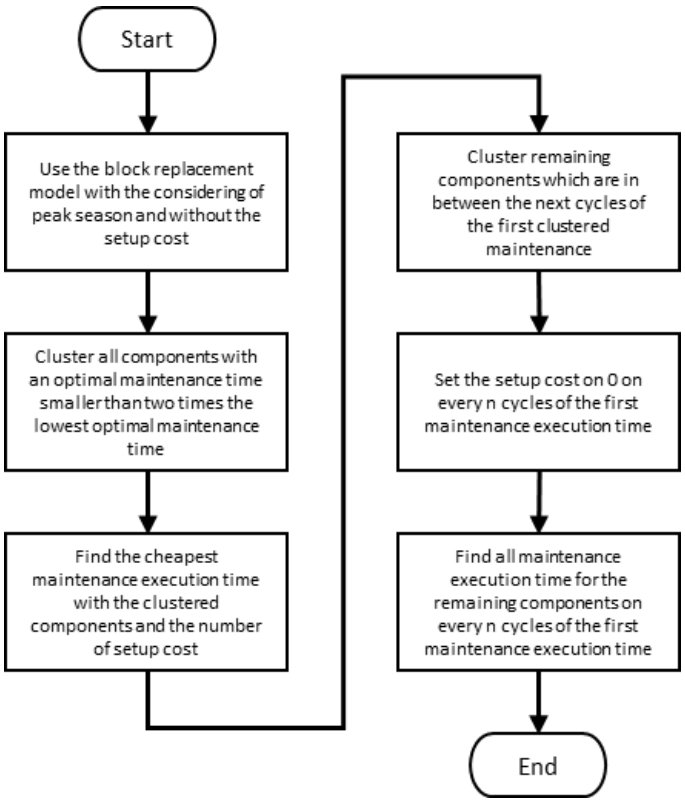


Figure E.3: MOM algorithm

E.3. Verification on the Maintenance optimisation model

The process of the MOM algorithm is shown again in Figure E.3. For the verification of the MOM, we used the same example as given in Section E.1, which is shown on the input of Table E.2.

Hereafter, we made a list of changing variables and checked if the results led to the expectation. The results are shown in Table E.3. The changes on the table are in chronological order, as the list of all verification.
A list of all verification:

1. **Changed description:** Increase the reactive maintenance costs of component one to 50000.
   Expectation: The difference between preventive and reactive maintenance will increase. It might result in a shorter maintenance interval and an increased cost per hour for component one.
   Result from the model: As expected, a relatively higher reactive maintenance cost resulted in a shorter maintenance interval.

2. **Changed description:** Increase scale parameter of component two to 8000.
   Expectation: the scale parameter is related to the characteristic life of a component. It is the time at which 63.2% of the components will fail. Therefore, it might result in a longer maintenance interval

Table E.2: Verification of the Block replacement model.

Input				
Component number	Shape parameter	Scale parameter [hours]	Reactive cost [€]	Preventive cost [€]
1	2	5000	1000	10000
2	2	5000	1000	2000
3	10	5000	1000	10000
4	10	5000	1000	2000
Total hours			8000	
Step size [hours]			15	
Peak season interval [hours]			750	
Setup costs [€]			1000	

Output		
Component number	Maintenance interval [Hours]	Optimal cost per hour [€/hour]
1	1500	1.307
2	1500	1.016
3	3000	0.355
4	3000	0.345
Setup costs on each interval		0.666

and a decreased cost per hour for component two.
Result from the model: As expected, a relatively higher scale parameter resulted in a longer maintenance interval

3. **Changed description:** Decrease shape parameter of component Three to two.
Expectation: The shape parameter that has a distinct effect on the scatteredness of failures in time.in failure time. Decreasing the shape parameter will increase the scatteredness. The chance of failure becomes less predictable. Therefore, it might result in a shorter maintenance interval and an increased cost per hour for component three.
Result from the model: As expected, a relatively low shape parameter resulted in a shorter maintenance interval.

4. **Changed description:** Increase setup cost to 100.
Expectation: A low setup cost becomes less incentive to group multiple components. The amount of maintenance execution becomes less expensive due to the decreased setup cost. Therefore, grouping multiple components on one maintenance interval will be less likely. The maintenance intervals will be closer to their actual maintenance intervals (See Figure E.1).
Result from the model: As expected, component two got its own maintenance interval due to a low setup cost. Therefore, it is checked successfully.

5. **Changed description:** Increase peak season interval to 2000 hours
Expectation: The peak season interval provides reduced maintenance costs after the peak season interval. Increasing the peak season interval will change the maintenance intervals for all components. The costs per hour will be different but should be close to the cost of the initial peak season interval.
Result from the model: As expected, the maintenance intervals and the cost per hour are cost to the initial peak season interval. Therefore, it is checked successfully

Table E.3: Overview of the verification list of the Maintenance optimisation model

Description number	Changed input	Changed output maintenance interval [hours]	Changed output optimal cost per hour [€]	As expected?
Originally		1500/1500/3000/3000	1.307/1.016/0.355/0.345	
1	50000	750/2250/3000/3000	2.851/0.949/0.355/0.345	yes
2	8000	1500/3000/3000/3000	1.307/0.602/0.355/0.345	yes
3	2	1500/1500/1500/3000	1.307/1.016/1.307/0.345	yes
4	100	1500/2250/3000/3000	1.254/0.874/0.354/0.345	yes
5	2000	2000/2000/2000/4000	1.333/0.955/0.501/0.403	yes

F

Appendix F: Verification simulation model

In this appendix, the verification of the simulation model is shown. The simulation model is a combination of different tasks repeatedly. The essential task of the simulation model is the connection of taking the correct output value for the following tasks.

For this verification, we gathered all related data and task connections from each maintenance cycle. For this test, we used the example of the use case. The following data points are saved in excel:

- Maintenance cycle number;
- Shape parameter (beta);
- Scale parameter (eta);
- Optimal individual maintenance interval (from BRM);
- Optimal maintenance interval (from MOM);
- Amount of failure;
- Amount of suspension;
- Virtual time;
- Failure times;
- Suspension times;
- Samples of generated lifetime.

With the following data points, we can point out if the output is correctly connected with the next input. Based on the use case,we used the same input. The table is shown again in Table F.1

F.1. Generated lifetime

To verify the simulation model, we start with the generated value. After that, we follow chronologically the actions that must be performed after generating the value. The generated value is the lifetime based on the true Weibull parameter given as an input. We used the Maximum likelihood estimation for calculating 200 generated lifetimes back to the given true Weibull parameters. Table F.2 shows the input and output of the generated lifetimes. The calculation of error percentage is given in Equation F.1 (Helmenstine, 2021).

$$percentageerror = \frac{|experimentalvalue - truevalue|}{truevalue} \cdot 100 \tag{F.1}$$

Table F.1: Data inputs for the simulation model

Component specific data					
Component number	**Component costs [€]**	**Repair time** $[hours]$	**Penalty cost on failure** [€]	**Initial shape parameter** (β_0)	**Initial scale parameter** (η_0)
1	3128.08	3	22250	5	23100
2	14.5	2	21850	5	4920
3	1167.36	3	22750	5	14600
4	395.60	2	22250	5	9570
5	22.76	2	22050	5	4920
6	580	3	26250	5	14600
7	2848.04	1.5	22050	5	22670
8	539.16	1	21850	5	18020
9	1125.36	1.5	21750	5	22170
10	2270	1	23250	5	23530

Customer specific Data	
Setup cost	€1000
Capacity	1333 per hour
Lost per production	€0.1 per cutting · capacity + €150 per hours for the operators
Man hours	€100 per hour
Average running hours per day	15 hours
Average peak season weeks	10 weeks
Machine lifespan	20 years

Simulation specific input		
Total maintenance years	100 (=75000 hours)	
Amount of simulation	100	
Amount of generated lifetime per component	500	
True Weibull parameters		
Component number	**True shape parameter** (β)	**True scale parameter** (η)
1	2	10000
2	1.3	22000
3	1.5	6000
4	4	15000
5	3	12000
6	5	7380
7	2	8000
8	2.5	7000
9	5	3500
10	2.5	14000

Table F.2: Input and output of the generated lifetimes of each component

Input			Output			
Component number	**True shape parameter**	**True scale parameter**	**Estimated shape parameter**	**Estimated scale parameter**	**Shape error percentage**	**Scale error percentage**
1	2	10000	1.816	9749	9.2	2.5
2	1.3	22000	1.308	22462	0.6	2.1
3	1.5	6000	1.623	6062	8.2	1.0
4	4	15000	3.741	14884	6.5	0.8
5	3	12000	3.118	11929	3.9	0.6
6	5	7380	5.117	7397	2.3	0.2
7	2	8000	1.994	7984	0.3	0.2
8	2.5	7000	2.292	6898	8.3	1.46
9	5	3500	5.56	3639	11.2	4.0
10	2.5	14000	2.277	13663	8.9	2.4

F.2. Sorting lifetime to suspension and failure time

After generating 200 lifetimes for each component, each maintenance cycle will take a sample of each component. A general virtual time of the machine and the local virtual time of each component are created to sort samples of a lifetime in failure or suspension time. The general and local virtual times are added for each maintenance cycle by the upcoming maintenance interval. However, the local virtual time becomes zero when the component has been replaced. If the first sample of lifetimes is lower than the local virtual time of that component, the lifetime becomes a failure time. When the local virtual time reaches the maintenance interval of that component, it becomes suspension time. To verify this action, an example of the first 20 maintenance cycles of a simulation for the use case is shown in F.1 and F.2.

It shows that in the first maintenance cycle, component six, eight and nine are in the first maintenance interval of 30 weeks (2250 hours). Observing the first lifetime sample of component six, eight and nine in Figure F.2, component six and eight have both lifetimes higher than 2250 hours and component nine is lower than the maintenance interval. Therefore components will be failed before the maintenance interval and are sorted as failure time. Component six and eight will be maintained before failure. Therefore, the service department would not know the real failure time. Instead of recording the lifetime value. The local virtual time is recorded as suspension time instead. In the second maintenance cycle. Component nine has been maintained again. The suspension time is the local virtual time of component nine. Since it was already maintained on the first maintenance cycle, the virtual local time started over again. Observing the 20 maintenance cycles, sorting failure and suspension time have been successfully implemented.

F.3. Estimation method

For the estimation method, the proposed maintenance framework uses two methods: Adjusting the scale parameter and the maximum likelihood estimation. After a suspension with zero failures, it adds up to 1.4 times the initial scale parameter. After the first failure, the scale parameter decreases a half peak season interval. Back to Figure F.1, the first maintenance cycle shows a suspension time for component six and eight and a failure time for component nine. In the second maintenance cycle, component six and eight have an increased scale parameter by a multiplication of 1.4, while component nine is decreased by a half peak season interval (375 hours).

Furthermore, in the eleventh maintenance cycle, component three has failed thrice. In the next maintenance cycle, the maximum likelihood estimation is used to estimate new Weibull parameters. The twelfth maintenance cycle shows that the shape parameter has been estimated as well and the scale parameter has been completely different than before. Observing all maintenance cycles, the estimation method is successfully implemented.

F.4. Maintenance optimisation model

Lastly, the Maintenance optimisation model will be used to calculate the maintenance interval for each component. The MOM has been verified in Appendix E. However, the connection of using the right input in the simulation model must be verified. Observing the optimal individual columns, the individual week increases

Cycle	Beta	Eta	Optimal individual [weeks]	Maintenance interval [weeks]	Amount of Failures	Amount of Suspensions	Virtual time	Failures	Suspensions
1	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[12825.0, 14571.0, 9004.0, 17690.0, 12737.0, 8940.0, 9411.0, 5882.0, 4779.0, 18821.0]	[90, 80, 60, 100, 70, 50, 60, 30, 30, 120]	[90, 60, 60, 90, 60, 30, 60, 30, 30, 120]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]	[0, 0, 0, 0, 0, 1, 0, 1, 0, 0]	2250	[[], [], [], [], [], [], [], [], [], [2250]]	[[], [], [], [], [], [], [2250], [], [], []]
2	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[12825.0, 14571.0, 9004.0, 17690.0, 12737.0, 12516.0, 9411.0, 8234.8, 4404.0, 18821.0]	[90, 80, 60, 100, 70, 70, 60, 40, 20, 120]	[60, 40, 20, 60, 40, 60, 20, 40, 20, 90]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]	[0, 0, 1, 0, 0, 1, 1, 1, 1, 0]	3750	[[], [], [], [], [], [], [], [], [], [3750]]	[[], [], [3750], [], [], [], [3750], [], [1500], []]
3	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[12825.0, 14571.0, 12605.6, 17690.0, 12737.0, 12516.0, 13175.4, 8234.8, 4404.0, 18821.0]	[90, 80, 80, 100, 70, 70, 90, 40, 20, 120]	[40, 20, 80, 40, 20, 40, 80, 20, 20, 70]	[0, 0, 1, 0, 0, 0, 0, 0, 1, 0]	[0, 1, 1, 0, 1, 1, 1, 2, 2, 0]	5250	[[], [], [597], [], [], [], [], [], [], []]	[[], [5250], [], [], [5250], [], [], [3000], [1500], []]
4	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[12825.0, 20399.4, 12230.6, 17690.0, 17831.8, 12516.0, 13175.4, 11528.72, 4404.0, 18821.0]	[90, 110, 80, 100, 100, 70, 90, 60, 20, 120]	[20, 100, 80, 20, 100, 20, 60, 60, 20, 50]	[0, 0, 1, 1, 0, 0, 0, 0, 1, 0]	[1, 1, 1, 0, 1, 2, 1, 2, 3, 0]	6750	[[], [], [], [5762], [], [], [], [], [], []]	[[6750], [], [], [], [], [4500], [], [], [1500], []]
5	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 20399.4, 12230.6, 17315.0, 17831.8, 17522.4, 13175.4, 11528.72, 4404.0, 18821.0]	[120, 110, 80, 100, 100, 100, 90, 60, 20, 120]	[120, 80, 60, 100, 80, 100, 40, 40, 20, 20]	[0, 0, 1, 1, 0, 0, 0, 0, 1, 0]	[1, 1, 1, 0, 1, 2, 1, 2, 4, 1]	8250	[[], [], [], [], [], [], [], [], [], []]	[[], [], [], [], [], [], [], [], [1500], [8250]]
6	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 20399.4, 12230.6, 17315.0, 17831.8, 17522.4, 13175.4, 11528.72, 4404.0, 26349.4]	[120, 110, 80, 100, 100, 100, 90, 60, 20, 160]	[100, 60, 40, 80, 60, 80, 20, 20, 20, 160]	[0, 0, 1, 1, 0, 0, 1, 0, 1, 0]	[1, 1, 1, 0, 1, 2, 1, 3, 5, 1]	9750	[[], [], [], [], [], [], [], [], [], [5227]]	[[], [], [], [], [], [], [], [], [4500], [1500], []]
7	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 20399.4, 12230.6, 17315.0, 17831.8, 17522.4, 12800.4, 16140.21, 4404.0, 26349.4]	[120, 110, 80, 100, 100, 100, 80, 80, 20, 160]	[80, 40, 20, 60, 40, 60, 80, 80, 20, 140]	[0, 0, 1, 1, 0, 1, 1, 0, 1, 0]	[1, 1, 2, 0, 1, 2, 1, 3, 6, 1]	11250	[[], [], [], [], [], [], [], [], [4333], []]	[[], [], [6000], [], [], [], [], [], [1500], []]
8	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 20399.4, 12230.6, 17315.0, 17831.8, 17147.4, 12800.4, 16140.21, 4404.0, 26349.4]	[120, 110, 80, 100, 100, 100, 80, 80, 20, 160]	[60, 20, 80, 40, 20, 100, 60, 60, 20, 120]	[0, 0, 1, 1, 0, 1, 1, 0, 1, 0]	[1, 2, 2, 0, 2, 2, 1, 3, 7, 1]	12750	[[], [], [], [], [], [], [], [], [], []]	[[], [7500], [], [], [7500], [], [], [], [1500], []]
9	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 28559.16, 12230.6, 17315.0, 24964.52, 17147.4, 12800.4, 16140.21, 4404.0, 26349.4]	[120, 150, 80, 100, 140, 100, 80, 80, 20, 160]	[40, 140, 60, 20, 140, 80, 40, 40, 20, 100]	[0, 0, 2, 1, 0, 1, 1, 0, 1, 0]	[1, 2, 2, 1, 2, 2, 1, 3, 8, 1]	14250	[[], [], [2486], [], [], [], [], [], [], []]	[[], [], [], [7500], [], [], [], [], [1500], []]
10	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[17955.0, 28559.16, 11855.6, 17315.0, 24964.52, 17147.4, 12800.4, 16140.21, 4404.0, 26349.4]	[120, 150, 70, 100, 140, 100, 80, 80, 20, 160]	[20, 120, 60, 100, 120, 60, 20, 20, 20, 80]	[0, 0, 2, 1, 0, 1, 1, 0, 1, 0]	[2, 2, 2, 1, 2, 2, 2, 4, 9, 1]	15750	[[], [], [], [], [], [], [], [], [], []]	[[9000], [], [], [], [], [], [6000], [6000], [1500], []]
11	[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 28559.16, 11855.6, 17315.0, 24964.52, 17147.4, 12800.4, 22596.29, 4404.0, 26349.4]	[170, 150, 70, 100, 140, 100, 80, 120, 20, 160]	[170, 100, 40, 80, 100, 40, 80, 120, 20, 60]	[0, 0, 3, 1, 0, 1, 1, 0, 1, 0]	[2, 2, 2, 1, 2, 2, 2, 4, 10, 1]	17250	[[], [], [2518], [], [], [], [], [], [], []]	[[], [], [], [], [], [], [], [], [1500], []]
12	[5, 5, 1.12, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 28559.16, 4948.3, 17315.0, 24964.52, 17147.4, 12800.4, 22596.29, 4404.0, 26349.4]	[170, 150, 110, 100, 140, 100, 80, 120, 20, 160]	[150, 80, 100, 60, 80, 20, 60, 100, 20, 40]	[0, 0, 3, 1, 0, 2, 1, 1, 1, 0]	[2, 2, 2, 1, 2, 2, 2, 4, 11, 1]	18750	[[], [], [], [], [], [6409], [], [2170], [], []]	[[], [], [], [], [], [], [], [], [1500], []]
13	[5, 5, 1.12, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 28559.16, 4948.3, 17315.0, 24964.52, 16772.4, 12800.4, 22221.29, 4404.0, 26349.4]	[170, 150, 110, 100, 140, 90, 80, 120, 20, 160]	[120, 60, 90, 40, 60, 90, 40, 120, 20, 20]	[0, 0, 3, 1, 0, 2, 1, 1, 1, 1]	[2, 2, 2, 1, 2, 2, 2, 4, 12, 1]	20250	[[], [], [], [], [], [], [], [], [], [11266]]	[[], [], [], [], [], [], [], [], [], []]
14	[5, 5, 1.12, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 28559.16, 4948.3, 17315.0, 24964.52, 16772.4, 12800.4, 22221.29, 4404.0, 25974.4]	[170, 150, 110, 100, 140, 90, 80, 120, 20, 160]	[100, 40, 70, 20, 40, 70, 20, 100, 20, 160]	[0, 0, 3, 1, 0, 2, 1, 1, 1, 1]	[2, 2, 2, 2, 2, 2, 3, 4, 13, 1]	21750	[[], [], [], [], [], [], [], [], [], []]	[[], [], [], [7500], [], [], [6000], [], [1500], []]
15	[5, 5, 1.12, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 28559.16, 4948.3, 17315.0, 24964.52, 16772.4, 12800.4, 22221.29, 4404.0, 25974.4]	[170, 150, 110, 100, 140, 90, 80, 120, 20, 160]	[80, 20, 50, 100, 20, 50, 80, 80, 20, 140]	[0, 0, 3, 1, 0, 2, 1, 2, 1, 1]	[2, 3, 2, 2, 3, 2, 3, 4, 14, 1]	23250	[[], [], [], [], [], [], [], [], [], [3575]]	[[], [], [10500], [], [], [10500], [], [], [1500], []]
16	[5, 5, 1.12, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 39982.82, 4948.3, 17315.0, 34950.33, 16772.4, 12800.4, 21846.29, 4404.0, 25974.4]	[170, 210, 110, 100, 190, 90, 80, 110, 20, 160]	[60, 200, 20, 80, 190, 20, 60, 110, 20, 120]	[0, 0, 3, 1, 0, 3, 1, 2, 1, 1]	[2, 3, 3, 2, 3, 2, 3, 4, 15, 1]	24750	[[], [], [], [], [], [5733], [], [], [], []]	[[], [], [7500], [], [], [], [], [], [1500], []]
17	[5, 5, 1.1, 5, 5, 5, 5, 5, 5, 5]	[25137.0, 39982.82, 7912.55, 17315.0, 34950.33, 5895.49, 12800.4, 21846.29, 4404.0, 25974.4]	[170, 210, 400, 100, 190, 30, 80, 110, 20, 160]	[40, 190, 200, 60, 170, 20, 40, 90, 20, 100]	[1, 0, 3, 1, 0, 3, 1, 2, 1, 1]	[2, 3, 3, 2, 3, 3, 3, 4, 16, 1]	26250	[[9708], [], [], [], [], [], [], [], [], []]	[[], [], [], [], [], [], [1500], [], [], [1500], [], [], [1500], []]
18	[5, 5, 1.1, 5, 5, 5, 5, 5, 5, 5]	[24762.0, 39982.82, 7912.55, 17315.0, 34950.33, 5895.5, 12800.4, 21846.29, 4404.0, 25974.4]	[170, 210, 400, 100, 190, 30, 80, 110, 20, 160]	[170, 170, 200, 40, 150, 20, 20, 70, 20, 80]	[1, 0, 4, 1, 0, 3, 1, 3, 1, 1]	[2, 3, 3, 2, 3, 4, 4, 4, 17, 1]	27750	[[], [], [2250], [], [], [], [], [], [], [3839]]	[[], [], [], [], [], [], [1500], [], [], [6000], [], [], [1500], []]
19	[5, 5, 1.1, 5, 5, 5, 5, 2.64, 5, 5]	[24762.0, 39982.82, 6171.88, 17315.0, 34950.33, 5895.5, 12800.4, 5455.37, 4404.0, 25974.4]	[170, 210, 400, 100, 190, 30, 80, 20, 20, 160]	[150, 150, 200, 20, 130, 20, 80, 20, 20, 60]	[2, 0, 4, 1, 0, 3, 1, 3, 1, 1]	[2, 3, 3, 3, 3, 5, 4, 5, 18, 1]	29250	[[1907], [], [], [], [], [], [], [], [], []]	[[], [], [], [7500], [], [1500], [], [1500], [1500], []]
20	[5, 5, 1.1, 5, 5, 5, 5, 2.69, 5, 5]	[24387.0, 39982.82, 6171.88, 17315.0, 34950.33, 5895.51, 12800.4, 5457.53, 4404.0, 25974.4]	[160, 210, 400, 100, 190, 30, 80, 20, 20, 160]	[160, 130, 200, 100, 100, 20, 60, 20, 20, 40]	[2, 0, 4, 1, 0, 3, 2, 3, 1, 1]	[2, 3, 3, 3, 3, 6, 4, 6, 19, 1]	30750	[[], [], [], [], [], [], [], [], [], [2769]]	[[], [], [], [], [], [], [1500], [], [1500], [1500], []]

Figure F.1: A simulation example of the data point of the first 20 maintenance cycles.

lifetime sample	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Comp 1	11850,	9816,	9708,	1907,	12542,	14804,	7594,	2125,	2345,	4423,	10191,	15893,	5516,	4766,	14763,	13801,	6366,	10675,	15079,	3592,
Comp 2	30769,	36023,	11909,	26496,	4525,	10451,	24843,	67553,	11882,	4953,	10814,	39663,	3420,	27184,	2580,	11112,	87616,	41754,	31026,	8150,
Comp 3	7883,	597,	6810,	2486,	2518,	8150,	2250,	3177,	2296,	4716,	8110,	5822,	6872,	1094,	1505,	1698,	3923,	4040,	3309,	13582,
Comp 4	5762,	10466,	11311,	15692,	9200,	14316,	17785,	18536,	12592,	11965,	12354,	16976,	6695,	16915,	18478,	18543,	13840,	14481,	13822,	16788,
Comp 5	10320,	14128,	10824,	7867,	4491,	10745,	18909,	13466,	3251,	14180,	14259,	6827,	14196,	12907,	12349,	6504,	11161,	7738,	9860,	15080,
Comp 6	5478,	8225,	4333,	6409,	5733,	5905,	7179,	6445,	6492,	6155,	9506,	6822,	5793,	8795,	6015,	7037,	6454,	5064,	9591,	4983,
Comp 7	10316,	5227,	8094,	13442,	7288,	2769,	772,	13085,	5146,	3849,	2440,	5639,	1526,	15817,	5008,	9876,	7379,	4000,	6866,	3189,
Comp 8	7842,	3472,	6336,	11967,	2170,	3575,	3839,	10922,	2631,	6841,	3849,	4887,	3239,	12342,	5378,	7266,	3384,	6625,	1067,	10113,
Comp 9	1837,	2359,	2868,	3607,	3589,	4119,	3727,	2507,	3734,	3839,	3600,	2111,	4775,	3828,	2609,	3117,	3904,	3791,	3653,	4409,
Comp 10	11214,	11266,	13044,	12094,	4707,	9845,	19553,	17980,	9201,	7125,	9086,	9758,	14260,	7394,	10230,	6671,	25346,	8843,	6154,	17028,

Figure F.2: An example of the generated lifetime of each component

when the scale parameter increases and decreases when the scale parameter decreases as well. The maintenance interval decreases over time because the local virtual time increases over time. The total weeks of the maintenance interval remain constant approximately. Component with no failures and suspensions remains the same optimal individual weeks, while it decreases in the maintenance interval. Therefore, the maintenance interval is implemented successfully.

G

Appendix G: Simulation model code

```
import numpy as np
import itertools as iter
import matplotlib.pyplot as plt
import math as math
import scipy.integrate as integrate
import pandas as pd
import time
from datetime import datetime
from pyprobs import Probability as pr
from functools import reduce
from statistics import mean
from scipy.stats import weibull_min
from scipy import stats
from predictr import Analysis
from tqdm import tqdm
from itertools import compress
import winsound
from collections import deque



# ============================================================================
#                           START (INPUT)
# ============================================================================
#True Weibull parameter that generates failures
trueBeta = [2, 1.3, 1.5, 4, 3, 5, 2, 2.5, 5, 2.5    ]
trueEta = [10000, 22000, 6000, 15000, 12000, 7380, 8000, 7000, 3500, 14000 ]



#Cost input
compc = [3128, 50.5, 1167, 395.60, 80.76, 580, 2848, 539, 1125.36, 2270 ]
                # Cost of each comp [euro]
Trepair = [3, 2, 3, 2,  2, 3, 1.5, 1.5, 1.5, 1 ]
                                        # Repair time [hours]
Penalty = [1000, 100, 1000, 500, 100, 1000, 1000, 600, 500, 2000   ]
                    # Penalty costs [euro]
```

```
setup = 1000
    # Setupcost for each maintenance [euro]
capacity = 1333
    # Cycles per hour of the machine [production per production ]
lost = 0.1 * capacity + 150
    # Lost of production per cycle [euro]
responseDay = [5, 1, 3, 2, 1, 3, 5, 4, 5, 5]
    # Response day for each comp
Tresponse = [i*15 for i in responseDay]
    # Maintenance to response [hours (days * hours)]
manhour = 100
    # Manhour cost per hour [euro per hours]

# Components failure behaviour
servicelife = [ 8177 ,9290 ,5741, 11279, 8121, 5700, 6000, 3750, 3047, 12000]
    # Service life [hours]
betaInitial = [    5,    5,    5,    5,    5,    5,    5,    5,    5,    5   ]
    # Initial shape parameter
etaInitial = []
    # Calculated scale parameter per comp
for i in range(len(trueBeta)):
    etaInitial.append(round(servicelife[i]/ (-np.log(0.9)) ** (1/betaInitial[i])))




# ================================================================================
#                       (Mathematical) FUNCTIONS
# ================================================================================
def f(x, p, beta, scale):
    return x*weibull_min.pdf(x, beta[p], scale = scale[p])                    #
        expected failure Weibull distribution

def h(x, p, beta, scale):
    return weibull_min.pdf(x, beta[p], scale = scale[p])                      #
        weibull distribution (pdf)

def cdf(x, p, beta, scale):
    return weibull_min.cdf(x, beta[p], scale = scale[p])                      #
        weibull distribution (cdf)


def GenerateWeibull(sampleSize, beta, eta):                                   #
    Generate Data based on True Weibull parameters
    n = sampleSize      # number of samples
    k = beta      # shape
    lam = eta    # scale
    data = weibull_min.rvs(k, loc=0, scale=lam, size=n)
    return data

def EstimateParams(failures, suspensions, method = "mle"):                    #MLE
    (or MRR) method
    if len(failures) < 2:
        return ()

    weibullAnalysis = Analysis(df=failures, ds=suspensions, show=False)
```

```
    if method == "mrr":
        weibullAnalysis.mrr()
    elif method == "mle":
        weibullAnalysis.mle()
    else:
        raise ValueError(f"method argument must be either ‘mle‘ or ‘mrr‘, provided
            method was ‘{method}‘")
    return weibullAnalysis.beta, weibullAnalysis.eta


#Block replacement model
def CalculateInitialCosts(compc, Trepair, Penalty, setup, lost, Tresponse, manhour,
    betaInitial , etaInitial, inputBeta=None, inputEta=None ):

    if not inputBeta or not inputEta:
        newBeta = betaInitial
        newEta = etaInitial
    else:
        newBeta = [i if i else j for i,j in zip(inputBeta, betaInitial)]
        newEta = [i if i else j for i,j in zip(inputEta, etaInitial)]

# ============================================================================
#  Customisable Block replacement model
# ============================================================================
    start = 0.0000001                                                        #
        Start Time (dont use 0. It gives some warnings)
    stepsize = 75                                                            #
        Hours per week
    amountsteps = 401                                                        #
        Amount of weeks to be considered -1
    end = (amountsteps-1)*stepsize                                           #
        Stepsize * amountsteps
    Hours = np.linspace(start,end,amountsteps)                               #
        Timeline
    counter = 0
    psInterval = 10                                                          #
        Peak season interval [weeks]
    psIntervalHours = psInterval * stepsize                                  #
        Peak season interval [Hours]
    year = int((amountsteps - 1) /psInterval)                                #
        Total years


    Total_indi = []                                                          #
        Expected cost per component per hour (result of BRM)
    totalComponentCosts = []                                                 #
        Expected cost per component without setup costs per hour (results for MCM)
    EXP_Failure = []                                                         #
        Expected Failure per component per hour (result of BRM)
    overviewCost_rm = []                                                     #
        Reactive costs per component [euro]
    overviewCost_pmPS = []                                                   #
        Preventive costs per component [euro]

    # Calculation of the five variables mentioned above
    for i in range(len(newBeta)):
```

```
        Cost_pm = compc[i] + ((manhour + lost) * Trepair[i])
        Cost_rm = compc[i] + ((manhour + lost) * Trepair[i])  + (lost * Tresponse[i
            ]) + Penalty[i]
        Cost_pmPS = compc[i] + manhour * Trepair[i]
        overviewCost_rm.append(Cost_rm)
        overviewCost_pmPS.append(Cost_pmPS)
        array_integrals = []
        H = []
        EXP_Failure.append(H)

        for j in Hours:
            integral_j = integrate.quad(lambda x: h(x, i, newBeta, newEta) , j, j+
                stepsize)
            array_integrals.append(integral_j[0])
            H_pseudo = []
            array_steps = np.linspace(0, j, int( j / stepsize + 1) )
            steps_counter = len(array_steps) - 1
            counter = 0
            if j != 0:
                for L in range(steps_counter):
                    H_term_i = (1 + H[steps_counter - counter - 1]) *
                        array_integrals[counter]
                    H_pseudo.append(H_term_i)
                    counter = counter + 1
                H.append(sum(H_pseudo))

        TECC_indi =   Cost_pm  + setup + ((Cost_rm + setup) * np.array(H))
        TECC_gr = Cost_pm + ((Cost_rm + setup) * np.array(H))
        for y in range(0, year + 1):
            TECC_indi[psInterval*y] = (Cost_pm + setup - (lost * Trepair[i])) + ((
                Cost_rm + setup) * np.array(H[psInterval*y]))
            TECC_gr[psInterval*y] = (Cost_pm  - (lost * Trepair[i])) + ((Cost_rm +
                setup) * np.array(H[psInterval*y]))

        Total_indi.append(np.asarray([x/y if y != 0 else math.inf for x, y in zip(
            TECC_indi, Hours)]))
        totalComponentCosts.append(np.asarray([x/y if y != 0 else math.inf for x, y
            in zip(TECC_gr, Hours)]))


# ================================================================================
#     Graph of Block Replacement Model (UNCOMMENT THIS IF YOU WANT TO SEE THE GRAPH)
# ================================================================================
    #     plt.plot(Hours, Total_indi[i])
    #     plt.grid()
    #     plt.title("Block replacement model with peak season model")
    #     plt.xlabel("Running hours [hours]")
    #     plt.ylabel("Cost per hour[   /hour]")
    #     plt.ylim(ymax = 4, ymin = 0)
    #     plt.xlim(xmax = end, xmin = 375)
    #     plt.legend(["Belt pulley idler",
    #             "Bearing 6004-2RS",
    #             "Roller cassette and rail (h)",
    #             "Gantry drive pulley",
    #             "Bearing block UCFL205 (3)",
    #               "Gantry belt",
```

```
    #             "Radial Gripper",
    #             "Compact Cylinder",
    #             "Cylinder DHZ-20-80-PPV-A",
    #             "AC Motor with Motorreductor"
    #             ], ncol =2, prop={'size': 8})
    # plt.show()


# ============================================================================
#     Optimisation calculation for the BLOCK REPLACEMENT MODEL
# ============================================================================
    setupCosts = []                                         # Setup cost per
        eachhour
    individualOptimalValue = []                             # Maintenance interval
        Block replacement model
    groupOptimalValue = []                                  # Maintenance interval
        without setup costs
    nineFive=[]                                             # Maintenance interval
        with reliability > 95% (Option)
    for i in Hours:
        setupCosts.append(setup/i)

    # Calculation of the four variables mentioned above
    for k in range(len(betaInitial)):
        timeOfMain = np.argmin(Total_indi[k])
        ninefiveR = [i for i,j in zip (totalComponentCosts[k], EXP_Failure[k]) if j
            <0.05]
        timeOfMain1 = np.argmin(totalComponentCosts[k])
        timeOfMain2 =  np.argmin(ninefiveR)
        individualOptimalValue.append(timeOfMain)
        groupOptimalValue.append(timeOfMain1)
        nineFive.append(timeOfMain2)

    # print(f"Optimal individual time is on week {individualOptimalValue}")
    # print(f"Optimal group time is on week {groupOptimalValue}")
    # print(f"95% group time is on week {nineFive}")
    return groupOptimalValue, overviewCost_pmPS, overviewCost_rm, setupCosts,
        totalComponentCosts, setup, psIntervalHours, amountsteps, psInterval, end,
        EXP_Failure



# ============================================================================
# Maintenance optimisation model (algorithm)
# ============================================================================
def GetCostsInfo(totalComponentCosts, setupCosts, individualOptimalValue, psInterval
    , amountsteps, EXP_Failure, pastTime):

    # Get the data of specific component
    if None in individualOptimalValue:
        indicesNone = [x is not None for x in individualOptimalValue]
        totalComponentCosts = [x for x,y in zip(totalComponentCosts, indicesNone) if
             y == True]
        EXP_Failure = [x for x,y in zip(EXP_Failure, indicesNone) if y == True]
        pastTime = [x for x,y in zip(pastTime, indicesNone) if y == True]
    trueIndividualOptimalValue = [x for x in individualOptimalValue if x != None]
```

```
# print(trueIndividualOptimalValue)
maxLife = 20                                                    #
    machine years
if max(trueIndividualOptimalValue) <= maxLife *psInterval:
    maxIndiOptimalValue = max(trueIndividualOptimalValue)
else:
    maxIndiOptimalValue = maxLife*psInterval

if min(trueIndividualOptimalValue) <= 1 *psInterval:
    minIndiOptimalValue = 1*psInterval
elif min(trueIndividualOptimalValue) <= maxLife *psInterval:
    minIndiOptimalValue = min(trueIndividualOptimalValue)
else:
    minIndiOptimalValue = maxLife*psInterval

for i in range(len(totalComponentCosts)):
    filteredCost = deque(totalComponentCosts[i])
    for j in range(pastTime[i]):
        filteredCost.popleft()
    totalComponentCosts[i] = list(filteredCost)

# print(f'{trueIndividualOptimalValue} \n {maxIndiOptimalValue} \n {
    minIndiOptimalValue}')

quickSimulation = []
for i in range(len(totalComponentCosts)):
    quickSimulation.append(totalComponentCosts[i][math.floor(minIndiOptimalValue
        /psInterval)*psInterval:maxIndiOptimalValue+1:psInterval])
# print(quickSimulation)
numberOfComponents, numberOfTimesteps = np.shape(quickSimulation)


totalSteps = numberOfTimesteps ** numberOfComponents
indices1 = range(math.floor(minIndiOptimalValue/psInterval)*psInterval,
    maxIndiOptimalValue+1 , psInterval )

newTic = time.time()
indicesCombinations = iter.product(indices1, repeat=numberOfComponents)
costs = []
timeSteps = []
for indices in tqdm(indicesCombinations, total=totalSteps, position=0, leave=
    True):
        base = sum(totalComponentCosts[component][time] for component, time in
            enumerate(indices))
        setup = (sum([setupCosts[time] for time in set(indices)]))
        costs.append(base + setup)
        timeSteps.append(indices)

# newToc = time.time()
#print("New time duration:", newToc-newTic)

#Get the lowest costs (optimal)
timeStepsFiltered = [i for i, item in enumerate(timeSteps) if all(x <= y for x,y
     in zip(list(item), trueIndividualOptimalValue))]
```

```
    costsFiltered = [costs[i] for i in timeStepsFiltered]
    minimumCosts = min(costsFiltered )
    indexMinimum = costsFiltered.index(minimumCosts)
    t_value = timeSteps[timeStepsFiltered[indexMinimum]]

    t_valueStorage = []
    counter = 0
    for x in individualOptimalValue:
        if x is not None:
            t_valueStorage.append(t_value[counter])
            counter = counter + 1
        else:
            t_valueStorage.append(0)
    t_value = tuple(t_valueStorage)

    # print(f"Minimum costs are {minimumCosts} at index {indexMinimum}, with
        threshold week values of {t_value}")

    # df = pd.DataFrame()
    # df["Indices"] = timeSteps
    # df["Total"] = costs

    return t_value

# Cost of framework
def FrameworkCost(failureEachYear, suspensionEachYear, initialCostRM, initialCostPM,
     pastTime, prevReactiveCost = 0, prevPreventiveCost = 0):
    lengthRM = [len(i) for i in failureEachYear]
    cumReactiveCost = [i*j for i,j in zip(initialCostRM, lengthRM)]
    if prevReactiveCost == 0:
        totalCumReactiveCost = cumReactiveCost
    else:
        totalCumReactiveCost = [i+j for i,j in zip(cumReactiveCost, prevReactiveCost
            )]

    lengthPM = [len(x) if x!= [None] else 0 for x  in suspensionEachYear]
    cumPreventiveCost = [i*j for i,j in zip(initialCostPM, lengthPM)]
    if prevReactiveCost == 0:
        totalCumPreventiveCost = cumPreventiveCost
    else:
        totalCumPreventiveCost = [i+j for i,j in zip(cumPreventiveCost,
            prevPreventiveCost)]

    totalCost = sum(totalCumPreventiveCost) + sum(totalCumReactiveCost)

    df_costOutput = pd.DataFrame([[max(pastTime), totalCumPreventiveCost,
        totalCumReactiveCost, sum(totalCumPreventiveCost), sum(totalCumReactiveCost),
         totalCost]],
                                 columns = ["Time", "Preventive", "Reactive", "Sum
                                     preventive", "Sum reactive", "Totalcost"])
    return df_costOutput, totalCumReactiveCost, totalCumPreventiveCost

#From week to hours
def FindThreshold(t_value):
    threshold = [75 * x for x in t_value]
```

```
    return threshold

# reactive, perfect scenario and random interval costs
def OtherScenarios(years, timeCalculation, data, initialCostPM, initialCostRM,
    initialSetup):

    perfectAmount = []
    perfectCost = []
    tooLateCost = []
    realCost = []
    sortMaintenance = []
    servicelife = [ 3750 ,3000 ,3000, 3000, 2250, 3000, 4500, 1500, 1500, 7500]
    for i in range(len(trueBeta)):
        perfectAmountSub = []
        perfectAmount.append(perfectAmountSub)
        sortMaintenanceSub = []
        sortMaintenance.append(sortMaintenanceSub)

        sortMain = [item if item < servicelife[i] else servicelife[i] for item in
            data[i]]
        for j in range(0,len(data[i])):
            if sum(perfectAmount[i]) + data[i][j] <= timeCalculation:
                perfectAmountSub.append(data[i][j])
            else:
                break
        for j in range(0,len(data[i])):
            if sum(sortMaintenance[i]) + sortMain[j] <=timeCalculation:
                sortMaintenanceSub.append(sortMain[j])
            else:
                break


        lenMain = sortMaintenance[i].count(servicelife[i])
        realCostSub1 = lenMain* initialCostPM[i]
        lenFail = len(sortMaintenance[i]) - sortMaintenance[i].count(servicelife[i])
        realCostSub2 = lenFail *initialCostRM[i]
        realCost.append(realCostSub1+realCostSub2)
        perfectCostSub = len(perfectAmount[i]) *initialCostPM[i]
        tooLateCostSub = len(perfectAmount[i]) *initialCostRM[i]
        perfectCost.append(perfectCostSub)
        tooLateCost.append(tooLateCostSub)

    realTotalPerf = round(years * initialSetup + sum(perfectCost))
    tooLateTotal = round(years * initialSetup + sum(tooLateCost))
    realCostTotal = round(years * initialSetup + sum(realCost))



    return realTotalPerf, tooLateTotal, realCostTotal, perfectCost, tooLateCost,
        realCost


def SummaryCost(initialCostPM, initialCostRM, initialSetup, data, limitYear):

    print("SUMMARY")
    realTotal =[]
```

```
    tooLateTotal = []
    realCostTotal = []
    perfectComp =[]
    lateComp = []
    contractComp = []

    for i, j in zip(range(1500, int(limitYear)+1, 1500), range(2,int(limitYear/750)
        +1,2)):
        timeCalculation = i
        years = j
        realTotalSub, tooLateTotalSub, realCostTotalSub, perfectCost, tooLateCost,
            realCost = OtherScenarios(years, timeCalculation, data, initialCostPM,
            initialCostRM, initialSetup)
        realTotal.append(realTotalSub)
        tooLateTotal.append(tooLateTotalSub)
        realCostTotal.append(realCostTotalSub)
        perfectComp.append(perfectCost)
        lateComp.append(tooLateCost)
        contractComp.append(realCost)

    return  realTotal, tooLateTotal, realCostTotal, perfectComp, lateComp,
        contractComp

# ============================================================================
# Maintenance cycle of i+2 YEAR
# ============================================================================
def GetNextCycle(betaEstimates, etaEstimates, betaInitial, etaInitial, failures,
    suspensions,
                data, failureTF, pastTime, cumReactiveCost, cumPreventiveCost,
                    mainCycle):
    #All beta and eta estimates
    totalBetaEstimates = [[i] + j for i, j in zip(betaInitial, betaEstimates)]
    totalEtaEstimates = [[i] + j for i, j in zip(etaInitial, etaEstimates )]
    nextBetaEstimates = [x[-1] for x in totalBetaEstimates ]
    nextEtaEstimates = [x[-1] for x in totalEtaEstimates ]
    print(f"your new beta = {[round(i,3) for i in nextBetaEstimates]} ")##
    print(f"new eta = {[round(i,3) for i in nextEtaEstimates]}")##

    groupOptimalValue,  nextCostPM, nextCostRM, setupCosts, totalComponentCosts,
        initialSetup, psIntervalHours, amountsteps, psInterval, end, EXP_Failure =
        CalculateInitialCosts(compc, Trepair, Penalty, setup, lost, Tresponse,
        manhour, betaInitial , etaInitial, inputBeta=nextBetaEstimates, inputEta=
        nextEtaEstimates)

    # Separate components in a range < 2*min
    pastTimeWeekSub = [max(pastTime)-i for i in pastTime]
    pastTimeWeek = [int(i/75) for i in pastTimeWeekSub]
    reducedOptimal = [i-j for i,j in zip(groupOptimalValue, pastTimeWeek)]
    reducedOptimal = [10 if i < 10 else i for i in reducedOptimal]
    # print(reducedOptimal)
    mini = min(reducedOptimal)
    maxi = max(reducedOptimal)
    nCycles = range(1, math.floor(maxi/mini)+1)
    ranges = [range(mini*n, mini*(n+1)) for n in nCycles]

    separate = []
```

```
    for cRange in ranges:
        temp = []
        for x in reducedOptimal:
            if x in cRange:
                temp.append(x)
            else:
                temp.append(None)
        separate.append(temp)
    maintenanceAny = None
    if maintenanceAny != None:
        setupCosts[maintenanceAny::maintenanceAny] = [0 for x in setupCosts[
            maintenanceAny::maintenanceAny]]
    nextTotalThreshold = []
    thresholdCheck = False
    for x in separate:
        if all(i is None for i in x):
            continue
        elif thresholdCheck == True:
            for comp in range(len(nextTotalThreshold[0])):
                if 0 < nextTotalThreshold[0][comp] < 10:
                    nextTotalThreshold[0][comp] = 10
            for j in range(1, math.floor((amountsteps-1)/min(x for x in
                nextTotalThreshold[0] if x != 0))+1):
                setupCosts[j*min(x for x in nextTotalThreshold[0] if x != 0)] = 0
        nextT_value = GetCostsInfo(totalComponentCosts, setupCosts, x, psInterval,
            amountsteps, EXP_Failure, pastTimeWeek)
        nextTotalThreshold.append(list(nextT_value))
        thresholdCheck = True
    # print(pastTimeWeekSub)

    threshold = [sum(x) for x in zip(*nextTotalThreshold)]
    cumThresholdWeek = [i + j for i,j in zip(threshold, pastTimeWeek)]

    nextProb = []

    for i in range(len(betaInitial)):
        if EXP_Failure[i][cumThresholdWeek[i]] <= 0.1:
            nextProb.append(f"{round(EXP_Failure[i][cumThresholdWeek[i]],3)}")
        else:
            nextProb.append(f"{round(EXP_Failure[i][cumThresholdWeek[i]],3)}")



    nextThreshold =  FindThreshold(threshold)
    cumThresholdHour= [i + j for i,j in zip(nextThreshold, pastTimeWeekSub)]


    print(f" you start with threshold = {nextThreshold}")##
    print(f" Virtual time = {max(pastTime)}")
    print("READY FOR NEXT MAINTENANCE")


    # ============================================================================
    # Failure/suspension Simulation
    # ============================================================================
```

```
minimumThreshold = [j if j == min(nextThreshold) else math.inf for j in
    nextThreshold]
maintenanceOrder = [len(i+j) if len(i+j) < 450 else len(i+j) - 450 for i,j in
    zip(failures, suspensions)]
current = [i+min(nextThreshold) for i in pastTimeWeekSub]
print(f'current = {current}')
failureData = [i[l] if i[l] < j  else j for i,j,l,m in zip(data, current,
    maintenanceOrder, minimumThreshold)]
nextFailures =  [[i] if i < j  else [] for i,j in zip(failureData, current)]
pastTime1 = [i + j if k != [] else i for i,j,k in zip(pastTime,current,
    nextFailures)]

failureData2 = [i if j != math.inf and k == [] else math.inf for i,j,k in zip(
    failureData, minimumThreshold, nextFailures)]
failureData3 = [15000 if j >= 15000 and k == [] else i for i,j,k in zip(
    failureData2,current, nextFailures) ]
pastTime = [i + k if j != math.inf else i for i,j,k in zip(pastTime1,
    failureData3, current)]
print(f'pastTime = {pastTime}')

# print(f'cumThreshold = {cumThresholdHour} \n {failureData3}')
nextSuspensions = [[i] if j != math.inf and i == j and k == [] else [] for i,j,k
     in zip(cumThresholdHour, failureData3, nextFailures)]
nextSuspensions = [[j] if j >= 15000 and j != math.inf else i for i,j in zip(
    nextSuspensions, failureData3) ]

# Append failure and suspensions data for every component
failures = [i + j for i, j in zip(failures, nextFailures)]
suspensions = [i + j for i, j in zip(suspensions, nextSuspensions)]
# print(f"amount failures = {[len(i) for i in failures]}, {suspensions} ")
# print(f"{[[round(val) for val in sublst] for sublst in failures]}")

# Print and save part of results

df_cycle = pd.DataFrame([[mainCycle+2, [round(i[-1],2) for i in betaEstimates],
    [round(i[-1],2) for i in etaEstimates], groupOptimalValue, threshold, [len(i)
     for i in failures], [len(i) for i in suspensions], max(pastTime),
    nextFailures, nextSuspensions]],
                              columns = ["Cycle", "Beta", "Eta", "Optimal
                                  individual", "Threshold", "Amount of Failures
                                  ","Amount of Suspensions", "Virtual time", "
                                  Failures", "Suspensions"])

df_nextCostOutput, nextCumReactiveCost, nextCumPreventiveCost = FrameworkCost(
    nextFailures, nextSuspensions, nextCostRM, nextCostPM, pastTime,
    cumReactiveCost, cumPreventiveCost)

# ============================================================================
#    New parameter estimation
# ============================================================================

# Estimate beta and eta params for every component

betas = [[]] * len(trueBeta)
etas = [[]] * len(trueBeta)
```

```
    for compIndex in range(0, len(trueBeta)):
        if len(failures[compIndex]) > 2:
            if [EstimateParams(failures[compIndex], suspensions[compIndex])[0]] <
                [1.1]:
                betas[compIndex] = [1.1]
                etas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                    compIndex])[1]]
            elif [EstimateParams(failures[compIndex], suspensions[compIndex])[0]] >
                [5]:
                betas[compIndex] = [5]
                etas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                    compIndex])[1]]
            else:
                betas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                    compIndex])[0]]
                etas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                    compIndex])[1]]

        elif len(nextSuspensions[compIndex]) == 0 and len(nextFailures[compIndex])
            == 0:
              betas[compIndex] = [nextBetaEstimates[compIndex] ]
              etas[compIndex] = [nextEtaEstimates[compIndex] ]

        elif len(failures[compIndex]) == 0 :
            betas[compIndex] = [nextBetaEstimates[compIndex] ]
            etas[compIndex] = [nextEtaEstimates[compIndex] *1.4]

        elif len(failures[compIndex]) == 1 and failureTF[compIndex] == True:
            betas[compIndex] =  [nextBetaEstimates[compIndex] ]
            etas[compIndex] = [nextEtaEstimates[compIndex] -0.5*psIntervalHours]
            failureTF[compIndex] = False

        elif len(failures[compIndex]) == 2 and failureTF[compIndex] == False:
            betas[compIndex] =  [nextBetaEstimates[compIndex] ]
            etas[compIndex] = [nextEtaEstimates[compIndex] - 0.5*psIntervalHours]
            failureTF[compIndex] = True

        else:
            betas[compIndex] = [nextBetaEstimates[compIndex] ]
            etas[compIndex] = [nextEtaEstimates[compIndex] ]

    betaEstimates = [i + j for i, j in zip(betaEstimates, betas)]
    etaEstimates = [i + j for i, j in zip(etaEstimates, etas)]

    return betaEstimates, etaEstimates, betaInitial, etaInitial, failures,
        suspensions, data, failureTF, pastTime, nextCumReactiveCost,
        nextCumPreventiveCost, df_nextCostOutput, df_cycle, nextProb



def main(trueBeta, trueEta, betaInitial, etaInitial, order, failures = [[]] * len(
    trueBeta), suspensions = [[]] * len(trueBeta)):
    beta = trueBeta
    eta = trueEta

    #All lists to be saved and checked
```

```
initialList = [[]] * len(beta)
betaEstimates = initialList
etaEstimates = initialList
data = initialList
optimalProb = []
pastTime = [0] *len(beta)

# ============================================================================
#     First year:
# ============================================================================

print("CYCLE 1") ##
### The Calculation to determine time of maintenance execution ###
# Calculation on costs  with the "CalculateInitialCosts" function
groupOptimalValue, initialCostPM, initialCostRM, setupCosts, totalComponentCosts
    , initialSetup, psIntervalHours, amountsteps, psInterval, end, EXP_Failure =
    CalculateInitialCosts(compc, Trepair, Penalty, setup, lost, Tresponse,
    manhour, betaInitial, etaInitial)

# Separate components in separate cycles based on their optimal week to replace
mini = min(groupOptimalValue)                    # range of a cycle is length of
    minimal week for replacement
maxi = max(groupOptimalValue)
nCycles = range(1, math.floor(maxi/mini)+1)      # Amount of cycle based on total
     length/ minimal week
ranges = [range(mini*n, mini*(n+1)) for n in nCycles] #ranges of cycle at each
    cycle

#Taking components within each separate cycle
separate = []
for cRange in ranges:
    temp = []
    for x in groupOptimalValue:
        if x in cRange:
            temp.append(x)
        else:
            temp.append(None)
    separate.append(temp)

#calculate threshold maintenance for each cycle
totalThreshold = []
thresholdCheck = False
maintenanceAny = None
if maintenanceAny != None:
    setupCosts[maintenanceAny::maintenanceAny] = [0 for x in setupCosts[
        maintenanceAny::maintenanceAny]]

for x in separate:
    if all(i is None for i in x):
        continue
    elif thresholdCheck == True:
        for j in range(1, math.floor((amountsteps-1)/min(x for x in
            totalThreshold[0] if x != 0))+1):
            setupCosts[j*min(x for x in totalThreshold[0] if x != 0)] = 0
    initialT_value = GetCostsInfo(totalComponentCosts, setupCosts, x, psInterval
        , amountsteps, EXP_Failure, pastTime)
```

```
        totalThreshold.append(list(initialT_value))
        thresholdCheck = True
    threshold = [sum(x) for x in zip(*totalThreshold)]  #grouping back all threshold
         in a list
    initialThreshold = FindThreshold(threshold)          #threshold from week to
        hours

    optimalProb = []
    for i in range(len(betaInitial)):
        if EXP_Failure[i][threshold[i]] <= 0.1:
            optimalProb.append(f"{round(EXP_Failure[i][threshold[i]],3)}")
        else:
            optimalProb.append(f"{round(EXP_Failure[i][threshold[i]],3)}")
    df_prob = pd.DataFrame([optimalProb])


    # print(f" you start with beta = {betaInitial}") ##
    # print(f" eta = {etaInitial} \n")##
    print(f" you start with threshold = {initialThreshold} ")##
    print("READY FOR MAINTENANCE")


    # ================================================================================
    # Failure/suspension Simulation
    # ================================================================================

    #generate and process failure samples
    firstYearData = list(map(GenerateWeibull, [500] * len(beta), beta, eta))
    listDataGenerate = [i.tolist() for i in firstYearData]
    data = [[round(i) for i in sublist ] for sublist in listDataGenerate]
    dataAlternative = [[psIntervalHours*math.floor(i/psIntervalHours) for i in
        sublist] for sublist in listDataGenerate]

    # Threshold process
    minimumThreshold = [j if j == min(initialThreshold) else math.inf for j in
        initialThreshold]
    failureData = [i[0] if i[0] < min(minimumThreshold) else j for i,j in zip(data,
        minimumThreshold)]
    pastTime = [i + min(initialThreshold) if j != math.inf else i for i,j in zip(
        pastTime, failureData)]

    #filter in failure and suspension
    firstFailures = [[j] if j < i else [] for i,j in zip(initialThreshold,
        failureData)]
    firstSuspensions = [[i] if j != math.inf and i == min(minimumThreshold) and k ==
         [] else [] for i,j,k in zip(initialThreshold, failureData, firstFailures)]

    # Append failure and suspensions data for every component
    failures = [i + j for i, j in zip(failures, firstFailures)]
        #total failures
    suspensions = [i + j for i, j in zip(suspensions, firstSuspensions)]
        #total suspensions

    #Print and save part of result
    Prob = []
```

```
for i in range(len(betaInitial)):
    if EXP_Failure[i][threshold[i]] <= 0.1:
        Prob.append(f"{round(EXP_Failure[i][threshold[i]],3)}")
    else:
        Prob.append(f"{round(EXP_Failure[i][threshold[i]],3)}")

print(f"amount failures = {[len(i) for i in failures]} \n")
# print(f"{[[round(val) for val in sublst] for sublst in failures]}")
df_output = pd.DataFrame([[1, betaInitial, etaInitial, groupOptimalValue,
    threshold, [len(i) for i in failures], [len(i) for i in suspensions], max(
    pastTime), firstFailures, firstSuspensions]],
                         columns = ["Cycle", "Beta", "Eta", "Optimal individual
                             ", "Threshold", "Amount of Failures", "Amount of
                             Suspensions","Virtual time", "Failures","Suspensions
                             "])

df_costOutput, cumReactiveCost, cumPreventiveCost = FrameworkCost(firstFailures,
     firstSuspensions, initialCostRM, initialCostPM, pastTime)
df_probCycle = pd.DataFrame([Prob])

# ================================================================================
#    New parameter estimation
# ================================================================================

betas = [[]] * len(beta)
etas = [[]] * len(beta)
failureTF = [True] *len(beta)

for compIndex in range(0, len(beta)):
    if len(failures[compIndex]) > 2:
        if [1] <= [EstimateParams(failures[compIndex], suspensions[compIndex])
            [0]] <= [5]:
            betas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                compIndex])[0]]
            etas[compIndex] = [EstimateParams(failures[compIndex], suspensions[
                compIndex])[1]]
        elif [EstimateParams(failures[compIndex], suspensions[compIndex])[0]] >
            [5]:
            betas[compIndex] = [5]
            etas[compIndex] = [etaInitial[compIndex]]
        else:
            betas[compIndex] = [1.1]
            etas[compIndex] = [etaInitial[compIndex] ]
    elif len(firstSuspensions[compIndex]) == 0 and len(firstFailures[compIndex])
         == 0:
          betas[compIndex] = [betaInitial[compIndex] ]
          etas[compIndex] = [etaInitial[compIndex] ]
    elif len(failures[compIndex]) == 0 :
        betas[compIndex] = [betaInitial[compIndex] ]
        etas[compIndex] = [etaInitial[compIndex] *1.4]
    elif len(failures[compIndex]) == 1 and failureTF[compIndex] == True:
        betas[compIndex] = [betaInitial[compIndex]  ]
        etas[compIndex] = [etaInitial[compIndex]  -0.5* psIntervalHours ]
        failureTF[compIndex] = False
    else:
        betas[compIndex] = [betaInitial[compIndex]  ]
```

```
            etas[compIndex] = [etaInitial[compIndex] ]


    # new betas and etas
    print(f" from MLE calculation beta = {betas} and eta = {etas}")
    betaEstimates = [i + j for i, j in zip(betaEstimates, betas)]
    etaEstimates = [i + j for i, j in zip(etaEstimates, etas)]

    # ================================================================================
    #     From second year on:
    # ================================================================================

    for i in range(0, 200):
        print(f" CYCLE {i+2}")
        betaEstimates, etaEstimates,  betaInitial, etaInitial, failures, suspensions
            , data, failureTF,  pastTime,  cumReactiveCost, cumPreventiveCost,
            df_nextCostOutput, df_cycle, nextProb = GetNextCycle(betaEstimates,
                    etaEstimates, betaInitial, etaInitial, failures, suspensions
                        , data, failureTF, pastTime, cumReactiveCost,
                        cumPreventiveCost, i)
        #DATA STORAGE
        df_nextprob = pd.DataFrame([nextProb])
        df_output = pd.concat([df_output, df_cycle])
        df_probCycle = pd.concat([df_probCycle, df_nextprob])
        df_costOutput = pd.concat([df_costOutput, df_nextCostOutput])
        limitYear = 150000
        if max(pastTime) >= limitYear:
            break

# ================================================================================
# Data collection
# ================================================================================

    df_failSus = pd.DataFrame([[failures, suspensions]],columns = ["failures", "
        suspensions"])
    df_trueWeibull = pd.DataFrame([[trueBeta, trueEta]],columns = ["failures", "
        suspensions"])
    df_beginValue = pd.DataFrame([[[0],[0]]], columns = ["failures", "suspensions"])
    df_data = pd.DataFrame([[]])
    for i in range(len(trueBeta)):
        df_dataSub = pd.DataFrame([[data[i]]])
        df_data = pd.concat([df_data, df_dataSub])
    df_Weibull = pd.concat([df_trueWeibull, df_beginValue, df_failSus, df_data])

    #resultscalculate a period of costs

    realTotal, tooLateTotal, contractCostTotal, perfectComp, lateComp, contractComp
        = SummaryCost(initialCostPM, initialCostRM, initialSetup, dataAlternative,
        limitYear)
    # df_frameworkCost = pd.DataFrame([allTotal, frameworkReactiveComp,
        frameworkMaintenanceComp])
    df_perfectCost = pd.DataFrame([realTotal, perfectComp])
    df_tooLateCost = pd.DataFrame([tooLateTotal, lateComp])
    df_contractCost = pd.DataFrame([contractCostTotal, contractComp])
```

```
df_totalCost = pd.concat([df_perfectCost, df_tooLateCost, df_contractCost])
current_time = datetime.now().strftime("%y%m%d %H%M%S")


if order[1] == 0:
    df_costOutput.to_csv("results/Simu4/comMLE15highEC9_" + str(order[0]) +"_"+
        current_time + " A CostTest version7.0.csv", index=False)
    df_totalCost.to_csv("results/Simu4/comMLE15highEC9_" + str(order[0]) +"_"+
        current_time + " A summarycost version7.0.csv", index=False)
    df_output.to_csv("results/Simu4/comMLE15highEC9_" + str(order[0]) +"_"+
        current_time + " A process version7.0.csv", index=False)
    df_Weibull.to_csv("results/Simu4/comMLE15highEC9_" + str(order[0]) +"_"+
        current_time + " A failure version7.0.csv", index=False)
    df_probCycle.to_csv("results/Simu4/comMLE15highEC9_" + str(order[0]) +"_"+
        current_time + " A newexpectedfailure version7.0.csv", index=False)
elif order[1] == 1:
    df_costOutput.to_csv("results/Simu4/comMLE15perfEC9_" + str(order[0]) +"_"+
        current_time + " A CostTest version7.0.csv", index=False)
    df_totalCost.to_csv("results/Simu4/comMLE15perfEC9_" + str(order[0]) +"_"+
        current_time + " A summarycost version7.0.csv", index=False)
    df_output.to_csv("results/Simu4/comMLE15perfEC9_" + str(order[0]) +"_"+
        current_time + " A process version7.0.csv", index=False)
    df_Weibull.to_csv("results/Simu4/comMLE15perfEC9_" + str(order[0])+ "_"+
        current_time + " A failure version7.0.csv", index=False)
    df_probCycle.to_csv("results/Simu4/comMLE15perfEC9_" + str(order[0])+ "_" +
        current_time + " A newexpectedfailure version7.0.csv", index=False)
elif order[1] == 2:
    df_costOutput.to_csv("results/Simu4/comMLE15lowEC9_" + str(order[0]) +"_"+
        current_time + " A CostTest version7.0.csv", index=False)
    df_totalCost.to_csv("results/Simu4/comMLE15lowEC9_" + str(order[0]) +"_"+
        current_time + " A summarycost version7.0.csv", index=False)
    df_output.to_csv("results/Simu4/comMLE15lowEC9_" + str(order[0]) +"_"+
        current_time + " A process version7.0.csv", index=False)
    df_Weibull.to_csv("results/Simu4/comMLE15lowEC9_" + str(order[0])+ "_"+
        current_time + " A failure version7.0.csv", index=False)
    df_probCycle.to_csv("results/Simu4/comMLE15lowEC9_" + str(order[0])+ "_" +
        current_time + " A newexpectedfailure version7.0.csv", index=False)

else:
    print(f'ERRORRRRRRRRRRRRRRRRRRRRRR betainitial = {betaInitial}')

trueT = [0] *len(trueBeta)
groupTrueOptimalValue, initialCostPM, initialCostRM, setupCosts,
    totalComponentCosts, initialSetup, psIntervalHours, amountsteps, psInterval,
    end, EXPTrue_Failure = CalculateInitialCosts(compc, Trepair, Penalty, setup,
    lost, Tresponse, manhour, trueBeta, trueEta)

# Separate components in separate cycles based on their optimal week to replace
mini = min(groupTrueOptimalValue)                    # range of a cycle is length
     of minimal week for replacement
maxi = max(groupTrueOptimalValue)
nCycles = range(1, math.floor(maxi/mini)+1)      # Amount of cycle based on total
     length/ minimal week
ranges = [range(mini*n, mini*(n+1)) for n in nCycles] #ranges of cycle at each
    cycle
```

```
    #Taking components within each separate cycle
    separate = []
    for cRange in ranges:
        temp = []
        for x in groupTrueOptimalValue:
            if x in cRange:
                temp.append(x)
            else:
                temp.append(None)
        separate.append(temp)

    #calculate threshold maintenance for each cycle
    totalThreshold = []
    thresholdCheck = False
    maintenanceAny = None
    if maintenanceAny != None:
        setupCosts[maintenanceAny::maintenanceAny] = [0 for x in setupCosts[
            maintenanceAny::maintenanceAny]]

    for x in separate:
        if all(i is None for i in x):
            continue
        elif thresholdCheck == True:
            for j in range(1, math.floor((amountsteps-1)/min(x for x in
                totalThreshold[0] if x != 0))+1):
                setupCosts[j*min(x for x in totalThreshold[0] if x != 0)] = 0
        initialT_value = GetCostsInfo(totalComponentCosts, setupCosts, x, psInterval
            , amountsteps, EXPTrue_Failure , trueT)
        totalThreshold.append(list(initialT_value))
        thresholdCheck = True
    threshold = [sum(x) for x in zip(*totalThreshold)]  #grouping back all threshold
         in a list
    trueThreshold = FindThreshold(threshold)          #threshold from week to hours
    trueProb = []
    for i in range(len(betaInitial)):
        trueProb.append(f"{round(EXPTrue_Failure[i][threshold[i]],3)}")

    return failures, suspensions, betaEstimates, etaEstimates

# ============================================================================
# Additional simulation after the data collection (OPTIONAL)
# ============================================================================

def DatafromMain(trueBeta, trueEta, betaInitial, etaInitial, order, machines):
    totalFailures = [[]] * len(trueBeta)
    totalSuspensions = [[]] * len(trueBeta)
    betaNextEst = [[]] * len(trueBeta)
    etaNextEst = [[]] * len(trueBeta)
    for i in range(machines):
        failures, suspensions, betaEstimates, etaEstimates = main(trueBeta, trueEta,
             betaInitial, etaInitial, order)
        totalFailures= [i+j for i,j in zip(totalFailures, failures)]
        totalSuspensions = [i+j for i,j in zip(totalSuspensions, suspensions)]
        betaNextEst= [i+j for i,j in zip(betaNextEst, betaEstimates)]
        etaNextEst = [i+j for i,j in zip(etaNextEst, etaEstimates)]
```

```
# betas = [[]] * len(trueBeta)
# etas = [[]] * len(trueBeta)
# for compIndex in range(len(trueBeta)):
#     if len(totalFailures[compIndex]) > 3:
#         beta, eta = EstimateParams(totalFailures[compIndex], totalSuspensions[
    compIndex], method = 'mle')
#         if beta > 4 :
#             betas[compIndex] = 4
#             etas[compIndex] = eta
#         elif beta < 1:
#             betas[compIndex] = 1
#             etas[compIndex] = eta
#         else:
#             betas[compIndex] = beta
#             etas[compIndex] = eta
#     else:
#         betas[compIndex] = np.mean(betaNextEst[compIndex])
#         etas[compIndex] = np.mean(etaNextEst[compIndex])

# #year 40 - 60
# endFailures2 = [[]] * len(trueBeta)
# endSuspensions2 = [[]] * len(trueBeta)
# betaNextEst2 = [[]] * len(trueBeta)
# etaNextEst2 = [[]] * len(trueBeta)
# order[1] = 0
# for i in range(machines):
#     endFailures, endSuspensions, betaEstimates, etaEstimates  = main(trueBeta,
     trueEta, betas, etas, order, totalFailures, totalSuspensions)

#     endFailures2 = [i+j for i,j in zip(endFailures2, endFailures)]
#     endSuspensions2 = [i+j for i,j in zip(endSuspensions2, endSuspensions)]
#     betaNextEst2= [i+j for i,j in zip(betaNextEst2, betaEstimates)]
#     etaNextEst2 = [i+j for i,j in zip(etaNextEst2, etaEstimates)]

# betas2 = [[]] * len(trueBeta)
# etas2 = [[]] * len(trueBeta)
# for compIndex in range(len(trueBeta)):
#     if len(totalFailures[compIndex]) > 3:
#         beta, eta = EstimateParams(endFailures2[compIndex], endSuspensions2[
    compIndex], method = 'mle')
#         if beta > 4 :
#             betas2[compIndex] = 4
#             etas2[compIndex] = eta
#         elif beta < 1:
#             betas2[compIndex] = 1
#             etas2[compIndex] = eta
#         else:
#             betas2[compIndex] = beta
#             etas2[compIndex] = eta
#     else:
#         betas2[compIndex] = np.mean(betaNextEst2[compIndex])
#         etas2[compIndex] = np.mean(etaNextEst2[compIndex])

# order[1] = 3
# for i in range(machines):
```

```
    #      endFailures3, endSuspensions3, betaEstimates3, etaEstimates3  = main(
        trueBeta, trueEta, betas2, etas2, order, endFailures2, endSuspensions2)


    return "FINISH"


for i in range(5):
    order = [22, 0]
    machines = 1
    print(DatafromMain(trueBeta, trueEta, betaInitial, etaInitial, order, machines))



# ========================================================================
# multiple simulation for different Weibull parameters (OPTIONAL)
# ========================================================================


servicelife2 = [ 6677 ,7790 ,4241, 9779, 6621, 4200, 4500, 2250, 700, 10500]
servicelife2 = [i+1500 for i in servicelife2]
betaInitial2 = [    5,   5,    5,    5,    5,    5,    5,    5,    5,    5   ]
etaInitial2 = []      #Calculated value of scale parameter of each comp
for i in range(len(trueBeta)):
    etaInitial2.append(round(servicelife2[i]/ (-np.log(0.9)) ** (1/betaInitial2[i]))
        )

for i in range(20):
    order = [22, 1]
    machines = 1
    print(DatafromMain(trueBeta, trueEta, betaInitial2, etaInitial2, order, machines
        ))



servicelife3 = [ 6677 ,7790 ,4241, 9779, 6621, 4200, 4500, 2250, 100, 10500]
servicelife3 = [i+1500 for i in servicelife3]
betaInitial3 = [    5,   5,    5,    5,    5,    5,    5,    5,    5,    5   ]
etaInitial3 = []      #Calculated value of scale parameter of each comp
for i in range(len(trueBeta)):
    etaInitial3.append(round(servicelife3[i]/ (-np.log(0.9)) ** (1/betaInitial3[i]))
        )

for i in range(20):
    order = [22, 2]
    machines = 1
    print(DatafromMain(trueBeta, trueEta, betaInitial3, etaInitial3, order, machines
        ))


# duration = 400
# freq =440
# winsound.Beep(freq,duration)
```

H

Maintenance framework code

```
import numpy as np
import itertools as iter
import matplotlib.pyplot as plt
import math as math
import scipy.integrate as integrate
import pandas as pd
import time
from pyprobs import Probability as pr
from functools import reduce
from statistics import mean
from scipy.stats import weibull_min
from scipy import stats
from predictr import Analysis
from tqdm import tqdm
from itertools import compress

# ========================================================================
#                          START/INPUT
# ========================================================================

componentName = ["Belt pulley idler", "Bearing 6004-2RS", "Roller cassette and rail
    (h)",
                 "Gantry drive pulley", "Bearing block UCFL205 (3)", "Gantry belt",
                 "Radial Gripper", "Compact Cylinder", "Cylinder DHZ-20-80-PPV-A",
                 "AC Motor with Motorreductor"]
                                                        #Name of components

compc = [3128, 14.5, 1167, 395.60, 22.76, 580, 2848, 539, 1125.36, 2270 ]
                  #cost of each comp [euro]
Trepair = [3, 2, 3, 2,  2, 3, 1.5, 1.5, 1.5, 1 ]
                                          #repair time [hours]
Penalty = [1000, 600, 1500, 1000, 800, 5000, 800, 600, 500, 2000  ]
                       #Unforseen cost after a component failure


setup = 1000                                        #Setup cost for each maintenance
     [euro]
capacity = 1333                                     #Cycles per hour of the machine
```

```
lost = 0.1 * capacity + 150                          #Lost of production per cycle [
    euro]
Tresponse = 5 *15                                    #Maintenance to response [hours
    (days * hours)]
manhour = 100                                        #Manhour cost per hour [euro per
     hours]

servicelife = [ 6677 ,7790 ,4241, 9779, 6621, 4200, 4500, 2250, 1547, 10500]
              #Service life of each component
betaInitial = [2, 2, 1.5, 2, 2, 2.5, 2, 2, 2, 2]
                                        #Shape parameter of each component
etaNext = [0,0,0,0,0,0,0,0,0,0]        #If known, scale parameter of each component



etaInitial = [None]*len(servicelife)
                                                    #Scale parameter calculation

for i in range(len(etaInitial)):
    if etaNext[i] == 0:
        etaInitial[i] = round(servicelife[i]/ (-np.log(0.9)) ** (1/betaInitial[i]))
    else:
        etaInitial[i] = etaNext[i]

year = 0
stepsize = 75                                        #Running hours per week
psInterval = 10                                      #Amount of weeks of
    operating per year
maintenanceAny = None                                #If mainentance has to
    be executed anyway[weeks], else type = None

# ============================================================================
# Different functions
# ============================================================================

# Mathematical functions
def f(x, p, beta, scale):                                          # weibull
     distribution
    return x*weibull_min.pdf(x, beta[p], scale = scale[p])

def h(x, p, beta, scale):
    return weibull_min.pdf(x, beta[p], scale = scale[p])

def cdf(x, p, beta, scale):
    return weibull_min.cdf(x, beta[p], scale = scale[p])

# Procedural functions
def GenerateWeibull(sampleSize, beta, eta):
    n = sampleSize      # number of samples
    k = beta      # shape
    lam = eta    # scale
    data = weibull_min.rvs(k, loc=0, scale=lam, size=n)
    return data

#parameter estimation function
def EstimateParams(failures, suspensions, method = "mle"):
```

```
    if len(failures) < 2:
        return ()

    weibullAnalysis = Analysis(df=failures, ds=suspensions, bcm='c4', show=True)

    if method == "mrr":
        weibullAnalysis.mrr()
    elif method == "mle":
        weibullAnalysis.mle()
    else:
        raise ValueError(f"method argument must be either `mle` or `mrr`, provided
            method was `{method}`")
    return weibullAnalysis.beta, weibullAnalysis.eta, weibullAnalysis.beta_c4,
        weibullAnalysis.eta_c4

#block replacement model + peak season model
def CalculateInitialCosts(compc, Trepair, Penalty, setup, lost, Tresponse, manhour,
    betaInitial , etaInitial, stepsize, psInterval, inputBeta=None, inputEta=None ):

    if not inputBeta or not inputEta:
        newBeta = betaInitial
        newEta = etaInitial
    else:
        newBeta = [i if i else j for i,j in zip(inputBeta, betaInitial)]
        newEta = [i if i else j for i,j in zip(inputEta, etaInitial)]


    start = 0.0000001

    amountsteps = 201
    end = (amountsteps-1)*stepsize
                                                #peak season interval
    psIntervalHours = psInterval * stepsize

    year = int((amountsteps - 1) /psInterval)

    counter = 0
    Hours = np.linspace(start,end,amountsteps)                  #timeline
    totalComponentCosts = []                                                #ECC/ECL
        per component per hour
    EXP_Failure = []
    Total_indi = []
    overviewCost_pm = []
    overviewCost_rm =[]
    overviewCost_pmPS = []

    for i in range(len(newBeta)):
        Cost_pm = compc[i] + ((manhour + lost) * Trepair[i])
        Cost_rm = compc[i] + ((manhour + lost) * Trepair[i])  + (lost * Tresponse) +
             Penalty[i]
        Cost_pmPS = compc[i] + manhour * Trepair[i]
        overviewCost_pm.append(Cost_pm)
        overviewCost_rm.append(Cost_rm)
        overviewCost_pmPS.append(Cost_pmPS)
```

```
        array_integrals = []
        H = []
        EXP_Failure.append(H)

        for j in Hours:
            integral_j = integrate.quad(lambda x: h(x, i, newBeta, newEta) , j, j+
                stepsize)
            array_integrals.append(integral_j[0])
            H_pseudo = []
            array_steps = np.linspace(0, j, int( j / stepsize + 1) )
            steps_counter = len(array_steps) - 1
            counter = 0
            if j != 0:
                for L in range(steps_counter):
                    H_term_i = (1 + H[steps_counter - counter - 1]) *
                        array_integrals[counter]
                    H_pseudo.append(H_term_i)
                    counter = counter + 1
                H.append(sum(H_pseudo))

        TECC_indi =    Cost_pm  + setup + ((Cost_rm + setup) * np.array(H))
        TECC_gr = Cost_pm + ((Cost_rm + setup) * np.array(H))
        for y in range(0, year + 1):
            TECC_indi[psInterval*y] = (Cost_pm + setup - (lost * Trepair[i])) + ((
                Cost_rm + setup) * np.array(H[psInterval*y]))
            TECC_gr[psInterval*y] = (Cost_pm  - (lost * Trepair[i])) + ((Cost_rm +
                setup) * np.array(H[psInterval*y]))


        Total_indi.append(np.asarray([x/y if y != 0 else math.inf for x, y in zip(
            TECC_indi, Hours)]))
        totalComponentCosts.append(np.asarray([x/y if y != 0 else math.inf for x, y
            in zip(TECC_gr, Hours)]))


        plt.plot(Hours, Total_indi[i])
        plt.grid()
        plt.title("Individual peak season model")
        plt.xlabel("Running hours [hours]")
        plt.ylabel("Cost per hour[    /hour]")
        plt.ylim(ymax = 4, ymin = 0)
        plt.xlim(xmax = end, xmin = 375)
        plt.legend(componentName, ncol =2, prop={'size': 8})
    plt.show()

    # for k in range(len(etaInitial)):
    #     print(f" your failure chance for component {k+1} is {EXP_Failure[k][np.
        argmin(totalComponentCosts[k])]} ")

    setupCosts = []                                     #Setup cost per hour
    individualOptimalValue = []
    groupOptimalValue = []
    for i in Hours:
        setupCosts.append(setup/i)                      #setupcost loop

    #Check for optimal time for inidividual Components
```

```
    for k in range(len(betaInitial)):
        timeOfIndi = np.argmin(Total_indi[k])
        timeOfGroup = np.argmin(totalComponentCosts[k])
        individualOptimalValue.append(timeOfIndi)
        groupOptimalValue.append(timeOfGroup)
    print(f"Optimal individual time is on week {individualOptimalValue}")
    print(f"Optimal group time is on week {groupOptimalValue}")
    return groupOptimalValue, overviewCost_pmPS, overviewCost_rm, setupCosts,
        totalComponentCosts, setup, psIntervalHours, amountsteps, overviewCost_pm,
        end, EXP_Failure

#Clustering model
def GetCostsInfo(totalComponentCosts, setupCosts, individualOptimalValue, psInterval
    , amountsteps, EXP_Failure):



    # Get the data of specific component
    if None in individualOptimalValue:
        indicesNone = [x is not None for x in individualOptimalValue]
        totalComponentCosts = [x for x,y in zip(totalComponentCosts, indicesNone) if
            y == True]
        EXP_Failure = [x for x,y in zip(EXP_Failure, indicesNone) if y == True]

    trueIndividualOptimalValue = [x for x in individualOptimalValue if x != None]

    # print(trueIndividualOptimalValue)
    maxLife = 20                                                    #
        machine years
    if max(trueIndividualOptimalValue) <= maxLife *psInterval:
        maxIndiOptimalValue = max(trueIndividualOptimalValue)
    else:
        maxIndiOptimalValue = maxLife*psInterval

    if min(trueIndividualOptimalValue) <= 1 *psInterval:
        minIndiOptimalValue = 1*psInterval
    elif min(trueIndividualOptimalValue) <= maxLife *psInterval:
        minIndiOptimalValue = min(trueIndividualOptimalValue)
    else:
        minIndiOptimalValue = maxLife*psInterval



    # print(f'{trueIndividualOptimalValue} \n {maxIndiOptimalValue} \n {
        minIndiOptimalValue}')

    quickSimulation = []
    for i in range(len(totalComponentCosts)):
        quickSimulation.append(totalComponentCosts[i][math.floor(minIndiOptimalValue
            /psInterval)*psInterval:maxIndiOptimalValue+1:psInterval])
    # print(quickSimulation)
    numberOfComponents, numberOfTimesteps = np.shape(quickSimulation)


    totalSteps = numberOfTimesteps ** numberOfComponents
    indices1 = range(math.floor(minIndiOptimalValue/psInterval)*psInterval,
```

```
        maxIndiOptimalValue+1 , psInterval )

    newTic = time.time()
    indicesCombinations = iter.product(indices1, repeat=numberOfComponents)
    costs = []
    timeSteps = []
    for indices in tqdm(indicesCombinations, total=totalSteps, position=0, leave=
        True):
            base = sum(totalComponentCosts[component][time] for component, time in
                enumerate(indices))
            setup = (sum([setupCosts[time] for time in set(indices)]))
            costs.append(base + setup)
            timeSteps.append(indices)

    # newToc = time.time()
    #print("New time duration:", newToc-newTic)

    #Get the lowest costs (optimal)
    timeStepsFiltered = [i for i, item in enumerate(timeSteps) if all(x <= y for x,y
         in zip(list(item), trueIndividualOptimalValue))]

    costsFiltered = [costs[i] for i in timeStepsFiltered]
    minimumCosts = min(costsFiltered )
    indexMinimum = costsFiltered.index(minimumCosts)
    t_value = timeSteps[timeStepsFiltered[indexMinimum]]

    t_valueStorage = []
    counter = 0
    for x in individualOptimalValue:
        if x is not None:
            t_valueStorage.append(t_value[counter])
            counter = counter + 1
        else:
            t_valueStorage.append(0)
    t_value = tuple(t_valueStorage)

    # print(f"Minimum costs are {minimumCosts} at index {indexMinimum}, with
        threshold week values of {t_value}")


    return t_value

#Hours to weeks conversion
def FindThreshold(t_value, inHours = 75):
    threshold = [inHours * x for x in t_value]

    return threshold

# ======================================================================
#                                 MAIN FUNCTIONS
# ======================================================================
# ======================================================================
# Calculate maintenance execution time for each component
# ======================================================================
def Reliability(betaInitial, etaInitial, thresholdHours):
    reliabilityLimit = []
```

```
    for i in range(len(betaInitial)):
        reliabilityLimitSub = 1 - math.exp(-(thresholdHours[i]/etaInitial[i])**
            betaInitial[i])
        reliabilityLimit.append(reliabilityLimitSub)
    return reliabilityLimit

def Prepare(betaInitial, etaInitial, maintenanceAny, year):

    beta = betaInitial
    eta = etaInitial
    #All lists to be saved and checke

    ### The Calculation to determine time of maintenance execution ###
    # Calculation on costs  with the "CalculateInitialCosts" function

    groupOptimalValue, initialCostPM, initialCostRM, setupCosts, totalComponentCosts
        , initialSetup, psIntervalHours, amountsteps, overviewCost_pm, end,
        EXP_Failure = CalculateInitialCosts(compc, Trepair, Penalty, setup, lost,
        Tresponse, manhour, beta, eta, stepsize, psInterval)

    pastTimeWeek = [year *psInterval] *len(betaInitial)
    reducedOptimal = [i-j if i-j > 0 else i for i,j in zip(groupOptimalValue,
        pastTimeWeek)]


    # Separate components in separate cycles based on their optimal week to replace
    mini = min(reducedOptimal)                    # range of a cycle is length of
        minimal week for replacement
    maxi = max(reducedOptimal)
    nCycles = range(1, math.floor(maxi/mini)+1)      # Amount of cycle based on total
         length/ minimal week
    ranges = [range(mini*n, mini*(n+1)) for n in nCycles] #ranges of cycle at each
        cycle

    #Taking components within each separate cycle
    separate = []
    for cRange in ranges:
        temp = []
        for x in reducedOptimal:
            if x in cRange:
                temp.append(x)
            else:
                temp.append(None)
        separate.append(temp)


    #calculate threshold maintenance for each cycle
    totalThreshold = []
    thresholdCheck = False

    if maintenanceAny != None:
        setupCosts[maintenanceAny::maintenanceAny] = [0 for x in setupCosts[
            maintenanceAny::maintenanceAny]]

    for x in separate:
        if all(i is None for i in x):
```

```
            continue
        elif thresholdCheck == True:
            for j in range(1, math.floor((amountsteps-1)/min(x for x in
                totalThreshold[0] if x != 0))+1):
                setupCosts[j*min(x for x in totalThreshold[0] if x != 0)] = 0
        initialT_value = GetCostsInfo(totalComponentCosts, setupCosts, x, psInterval
            , amountsteps, EXP_Failure)
        totalThreshold.append(list(initialT_value))
        thresholdCheck = True
    threshold = [sum(x) for x in zip(*totalThreshold)]  #grouping back all threshold
         in a list
    thresholdHour = FindThreshold(threshold)          #threshold from week to hours



    for i in range(len(betaInitial)):
        if EXP_Failure[i][threshold[i]] <= 0.1:
            print(f" Expected failure for comp {i} = {round(EXP_Failure[i][threshold
                [i]],3)}, OK" )
        else:
            print(f" Expected failure for comp {i} = {round(EXP_Failure[i][threshold
                [i]],3)}, Warning!" )

    print(f" you start with shape = {betaInitial}") ##
    print(f" scale = {[round(i) for i in etaInitial]} \n")##
    print(f" you start with threshold = {thresholdHour} \n which is in week {
        threshold}")##
    link = [list(i) for i in zip(threshold, componentName)]
    link.sort()

    print(f" Therefore replacement for week {[i for i in link]}" )

    return "READY FOR MAINTENANCE"

# ========================================================================
# Input of failures and suspensions
# ========================================================================
    #Failure time data per component
totalFailures = [[4975], [],
                 [], [],
                 [], [],
                 [], [],
                 [], [] ]

     #Suspension time data per component
totalSuspensions = [[6000], [1500,1500,1500,1500,1500,1500,1500,1500],
                    [3000,3000], [4500, 4500],
                    [1500,1500,1500,1500,1500,1500], [1500, 1500, 1500, 1500,1500,
                        1500],
                    [3000,3000], [1500,1500,1500,1500, 1500, 1500],
                    [1500, 1500,1500,1500,1500,1500], []]

    #Amount of new failures
newFailures = [[0],[0],[0],[0], [0], [0], [0], [0], [0], [0]]
    #Amount of new suspensions
newSuspensions = [[0],[2],[0], [2], [2], [2], [0], [2], [2], [0]]
```

```
# ============================================================================
# New parameter estimation
# ============================================================================

def FailureAnalysis(totalFailures, totalSuspensions, newFailures, newSuspensions,
    stepsize, psInterval):
    initialList = [[]] * len(betaInitial)
    betaEstimates = initialList
    etaEstimates = initialList

    betaEtaMLE = list(map(EstimateParams, totalFailures, totalSuspensions, ["mle"] *
        len(betaInitial)))

    betas = [[]] * len(betaInitial)
    etas = [[]] * len(betaInitial)


    for compIndex in range(0, len(betaInitial)):
        if len(totalFailures[compIndex]) > 2:
            if [EstimateParams(totalFailures[compIndex], totalSuspensions[compIndex
                ])[0]] < [1.1]:
                betas[compIndex] = [1.1]
                etas[compIndex] = [EstimateParams(totalFailures[compIndex],
                    totalSuspensions[compIndex])[1]]
            elif [EstimateParams(totalFailures[compIndex], totalSuspensions[
                compIndex])[0]] > [5]:
                betas[compIndex] = [5]
                etas[compIndex] = [EstimateParams(totalFailures[compIndex],
                    totalSuspensions[compIndex])[1]]
            else:
                betas[compIndex] = [EstimateParams(totalFailures[compIndex],
                    totalSuspensions[compIndex])[0]]
                etas[compIndex] = [EstimateParams(totalFailures[compIndex],
                    totalSuspensions[compIndex])[1]]

        elif len(newSuspensions[compIndex]) == 0 and len(newFailures[compIndex]) ==
            0:
              betas[compIndex] = [betaInitial[compIndex] ]
              etas[compIndex] = [etaInitial[compIndex] ]

        elif len(totalFailures[compIndex]) == 0 :
            betas[compIndex] = [betaInitial[compIndex] ]
            etas[compIndex] = [etaInitial[compIndex] *1.4]

        elif len(totalFailures[compIndex]) == 1:
            betas[compIndex] =  [betaInitial[compIndex] ]
            etas[compIndex] = [etaInitial[compIndex] -0.5*(stepsize*psInterval)]

        elif len(totalFailures[compIndex]) == 2:
            betas[compIndex] =  [betaInitial[compIndex] ]
            etas[compIndex] = [etaInitial[compIndex] - 0.5*(stepsize*psInterval)]

        else:
            betas[compIndex] = [betaInitial[compIndex] ]
            etas[compIndex] = [etaInitial[compIndex] ]
```

```
    #new betas and etas
    betaEstimates = [i + j for i, j in zip(betaEstimates, betas)]
    etaEstimates = [i + j for i, j in zip(etaEstimates, etas)]
    realBeta = [i for sublist in betaEstimates for i in sublist]
    realEta = [i for sublist in etaEstimates for i in sublist]
    print(f"your new beta = {realBeta} \n and new eta = {realEta}")


    return "FINISH"


# ================================================================================
# BEFORE MAINTENANCE, PRINT PREPARE FUNCTION
# AFTER MAINTENANCE FILL IN THE NEW FAILURES AND SUSPENSIONS
# THEN PRINT FAILUREANALYSIS FUNCTION
# ================================================================================

# Maintenance optimisation model
print(Prepare(betaInitial, etaInitial, maintenanceAny, year))
# OR use Estimation method
print(FailureAnalysis(totalFailures, totalSuspensions, newFailures, newSuspensions,
    stepsize, psInterval))
```