AALBORG UNIVERSITY

STUDENT REPORT

P1 Project

Software

Simulation of Overall Equipment Effectiveness of Manufacturing Processes

A Program that Solves a Problem

Authors:
Andreas Løvig Borg
Benjamin Veje Smolt
Lasse Ryge Andersen
Lukas Juel Jacobsen
Marius Ihlen Gardshodn
Mikkel Hagerup Dolmer

Supervisor:
Mathias Ruggard Pedersen

December 18, 2020

The project is written in Overleaf v2 and compiled with pdfLaTeX.

Dept. of Computer Science
Strandvejen 12-14
DK-9000 Aalborg
https://www.cs.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Simulation of Overall Equipment Effectiveness of Manufacturing Processes

Theme:
A Program that Solves a Problem

Project Period:
Autumn semester 2020

Project Group:
A325b

Participant(s):
Andreas Løvig Borg
Benjamin Veje Smolt
Lasse Ryge Andersen
Lukas Juel Jacobsen
Marius Ihlen Gardshodn
Mikkel Hagerup Dolmer

Supervisor(s):
Mathias Ruggard Pedersen

Copies: 1

Numbered Pages: 52

Date of Completion:
December 18, 2020

Abstract:

Manufacturing is a vast industry that is an extensive part of the global economy. Through manufacturing, a significant amount of waste is generated in terms of physical waste. Moreover, non-physical waste is also generated, which is waste that causes a time overrun or cost overrun.
In order to reduce the different types of waste, the effectiveness of the manufacturing processes can be increased. Principles such as Lean manufacturing and OEE are concerned with the effectiveness of manufacturing.
The purpose of this project is to develop a software solution that is capable of simulating the OEE of manufacturing processes in order to test organisational changes, thus reducing waste. To perform this simulation, a probabilistic model utilising Monte Carlo simulation and inverse transform sampling is developed. The software solution is successful in simulating the OEE of manufacturing processes.

Preface

We have written this project to obtain a better understanding of the factors that determine whether a manufacturing system is successful or not. Additionally, we want to implement these factors in a software program that can be applied to model and simulate different types of manufacturing systems. We accomplished this by expanding our knowledge of lean tools, probability theory, sampling methods, and algorithms.

We would like to thank our supervisor Mathias Ruggaard Pedersen for advising us during this project.

Contents

1 | Introduction

Every product is something that initially started out as one or sometimes thousands of raw materials. These products have all been manufactured. Manufactured meaning something made from raw materials by hand or machinery [1].

In Figure 1.1 the global manufacturing value added as percent of GDP (gross domestic product) is illustrated. Manufacturing value added of an economy is the estimate of net output of the manufacturing sector obtained by adding up outputs and subtracting intermediate inputs. In the last 18 years the global manufacturing value added as percent of GDP has been fairly steady between 15.2% and 17.4%. Hence, manufacturing is an essential part of the global economy.

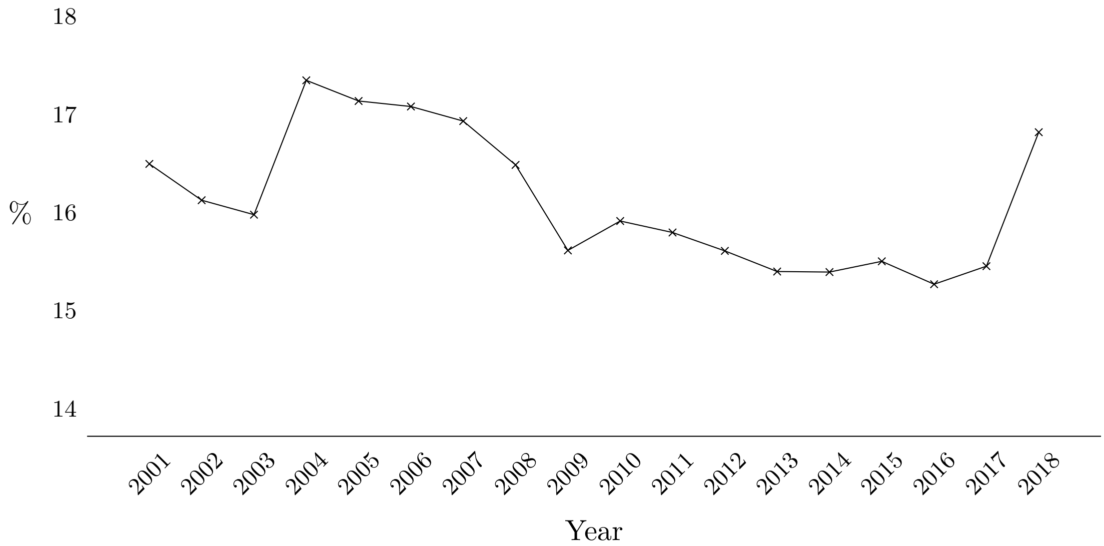

Figure 1.1: Global manufacturing value added as percent of GDP [2].

Since manufacturing is such an extensive part of the global economy, it is obviously a vast industry. Furthermore, manufacturing does not come without unnecessary expenditures and waste. For instance, in manufacturing significant amounts of physical waste is generated during the making of goods. In Figure 1.2 it is illustrated that the generation of waste from manufacturing constitutes 21.1% of the total waste generation from industries in EU. Specifically, the generation of waste in manufacturing includes the waste generated from manufacturing of foods, textiles, wood, paper, coke, chemicals, metals, electronics, transport equipment, and other machinery.

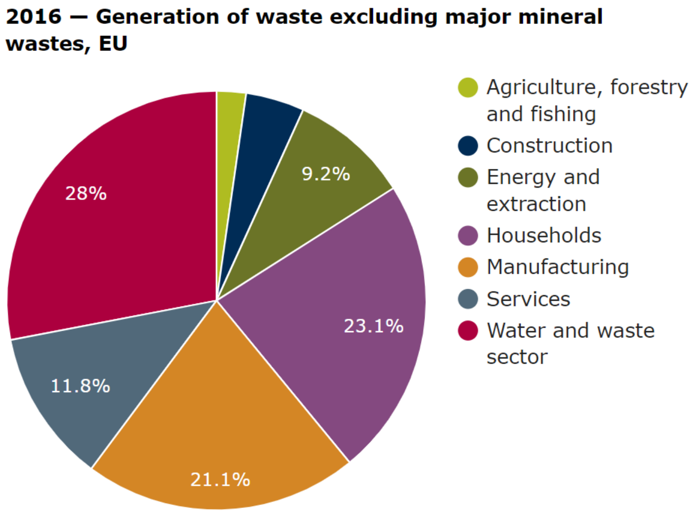

Figure 1.2: Waste generation from industries in EU [3].

Physical waste is not the only form of waste emerging in manufacturing systems. Whilst physical waste is easily measured, other forms of waste in manufacturing are equally important to consider when on the subject of manufacturing. These include human labour and the wear and tear of machines, as well as concepts such as time-efficiency and inventory management. The majority of these different types of waste are related to the effectiveness of the manufacturing.

In order to understand how these different types of waste are expressed in manufacturing, as well as how to model manufacturing systems with the intent of exposing whereby and to what degree the different types of waste are generated, it is important to understand the different types of manufacturing systems and how the different types of waste are categorised.

1.1 Manufacturing

The following section is based on [4, Ch. 1].

Manufacturing is the action of producing a product from mostly raw materials, adding value in the process. This is done through use of human labour as well as machines and equipment, that carry out a broad range of actions in a predetermined order to produce the product. This leads to the rather broad definition:

"[Manufacturing is] the making of products from raw materials using various processes, equipment, operations and manpower according to a detailed plan that is cost-effective and generates income through sales."

The manufacturing of a product is often described in a manufacturing system, which is just a series of processes, machines, etc. set together in such a way that they manufacture a product.

However, these broad definitions do not say a lot about how manufacturing is carried out in the real world. Since there is a variety of products being manufactured, many different manufacturing systems are used to describe different techniques and approaches to manufacturing. This leads to two basic categories of manufacturing systems, namely:

- Continuous process manufacturing.
- Discrete parts manufacturing.

1.1.1 Continuous Process Manufacturing

Continuous process manufacturing deals with continuous processes such as those found in the making of petroleum, steel, or sugar, where the product physically flows. This is sometimes confused with flow production, another term for mass production, which just refers to the fact that the assembly line is always running, and not the state of the product. Production in continuous process manufacturing often involves the use of chemicals in various stages of the production and might also involve mechanical means, all aiding in what is basically mixing the product following a recipe. However, it is important to note that no discrete product is made during processing, as the product is ever changing until finally being complete, and instead the outcome of this type of manufacturing system is often measured in volume or weight. Continuous process manufacturing often results in specialised equipment operating 24 hours a day to make the exact same product, which makes this type of manufacturing system highly specialised and thus not very flexible.

1.1.2 Discrete Parts Manufacturing

Discrete parts manufacturing deals with countable objects such as cars, toys, furniture, and the likes. These products all have the property in common of being countable. The production of discrete parts, as opposed to continuous, also allows for customisation of specific products to a certain degree as well as the ability to order anywhere from one to many millions of the product at a time, instead of always having continuous flow.

Discrete parts manufacturing is often further broken down into systems as shown in Figure 1.3. This figure also includes continuous manufacturing, which is not a type of discrete manufacturing. These different systems mainly describe the relationship between quantity and variety of the product, where larger quantities lead to less variety and the other way around.

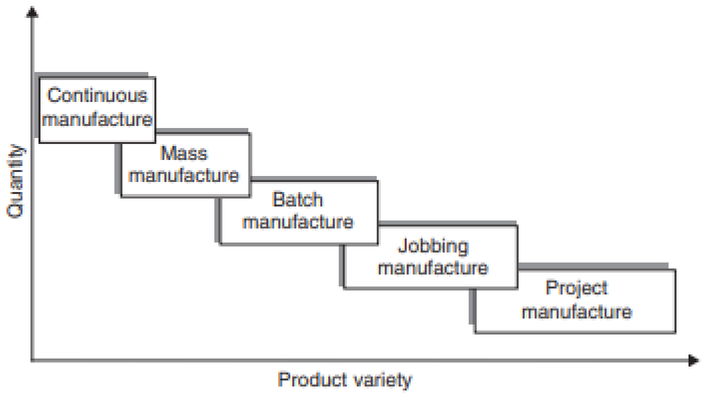

Figure 1.3: Quantity vs. product variety in the different production systems [4, Figure 1.9].

As seen in Figure 1.3, some of the different types of discrete manufacturing are as follows:

- Job manufacture.
- Batch manufacture.
- Mass manufacture.

Further explanation of these specific discrete manufacturing systems will be provided as well as insight in what this means for the workforce and tools needed to manufacture products using these systems.

1.1.3 Mass Manufacturing

Mass manufacturing is all about producing high rates of a specific product. To do this, specialised equipment and processes are used, which in turn means that the workforce has a lower skill level due to this specialised nature of the machines used in mass manufacturing. Machines in mass manufacturing are often arranged in a specific sequence to carry out their exact function one after another. To accomplish this, conveyor belts are often used to carry individual devices through the different machines in a predetermined sequence. This results in a sharp contrast to job manufacture, where an individual person can craft the entire product himself.

Mass manufacturing instead focuses on specialising individual processes in the making of the product and distributing these processes to different machines. This is much alike batch manufacturing, only on a larger scale with larger quantities and less variety. Mass manufacture is used to produce products which normally require a more steady output flow than batch manufacture, which is why it is also referred to as flow manufacturing.

1.1.4 Batch Manufacturing

Job manufacturing and batch manufacturing have quite a few similarities and are therefore often confused with one another. Normally batch manufacturing is a production of medium size lots. These lots are approximately 5-1000 units and sometimes even more. The difference between job manufacturing and batch manufacturing is not the number of components nor the number of lots, however, it is how the manufacturing itself is conducted. With batch manufacturing similar items are produced together, and each batch of components goes through one stage of the manufacturing process before going into the next stage. An example of batch manufacturing could be in a bakery where buns are prepared together, baked together, and they always stay together throughout the process.

1.1.5 Job Manufacturing

Job manufacturing is, as seen in Figure 1.3, a manufacturing system that focuses on variety and customisability as opposed to quantity. This means that lot-sizes are small, but the finished products are often unique. Manufacturing a lot of unique products requires machinery and tools that are non-specific, so they may be used for different purposes when producing different products. However, this requires the workforce to be highly skilled, since they must fulfill different assignments depending on the specific variety of a product being manufactured. The same person might also be the one to make an entire individual product.

1.1.6 Project Manufacturing

The characteristics of project manufacturing are the layout of the work. The product remains in the same position through the whole process, whereas in other manufacturing processes it typically physically moves through the different phases of the manufacturing process. The reason for the product being in the same place is usually because of its size and weight. These types of products usually have a low production rate. The workers, tools, and equipment used to produce these products are gathered around the product. Subparts of the product might be manufactured outside of the manufacturing process but are used as components in the process. The workers working on the product are usually highly able, and the handling of materials often requires a certain skill level as well. Examples of this manufacturing process are ships, buildings, bridges etc. As seen in Figure 1.3, there is a lot of variety in the products rather than quantity.

1.2 Manufacturing Effectiveness

One downside of the manufacturing industry is all the physical waste that is generated when raw materials are transformed into consumer goods. Some types of physical waste include various metals, oils, and chemicals, which may be hazardous and consequently dangerous. [5] However, waste within the manufacturing industry is not limited to only physical waste. Non-physical waste also exists and normally occurs during the manufacturing process. Non-physical waste is often referred to as anything that causes a time overrun or cost overrun. The awareness of the problems that lie within non-physical waste has led to the creation of different methodologies, which are targeted towards streamlining manufacturing processes and reducing waste. One of the most commonly heard of methodology within the bounds of manufacturing systems is Lean Manufacturing.

1.2.1 Lean Manufacturing

The term Lean Manufacturing refers to the application of Lean practices, principles, and tools to the development and manufacture of physical products. For many people, the term Lean manufacturing is synonymous with waste removal. But the ultimate goal of practicing Lean manufacturing is not simply to eliminate waste, instead it is to sustainably deliver value to the customer. To achieve that goal, Lean manufacturing describes waste as anything that requires an investment of time, money, or talent that does not create value for the customer [6].

One of the first people credited with introducing Lean practices at the workplace is the founder of Ford Motor Company, Henry Ford. Ford streamlined the process of manufacturing the Model T car by arranging workers, machines, parts, and tools in a continuous system. However, it was not until the 1930s that Toyota came up with the modern concept of Lean Manufacturing, when they invented the Toyota Production System (TPS). Toyota initiated the idea of "manufacturing to order" instead of "manufacturing to fill warehouses", because they realised products piled in warehouses without buyers were no more than just wastage. It made financial sense to base production targets on actual sales. This style of manufacturing eventually became known as Just-in-time (JIT) manufacturing [7].

TPS explicitly defines seven types of waste, which continues to be relevant in Lean manufacturing today. They are often referred to as TIMWOOD [8]:

- **Transport**
  Transportation of a product does not add any value to the product and is not a part of the manufacturing process, but it is a crucial part of the delivery of a product. Hence, it can not be separated from the process.

- **Inventory**
  When a company has bought too many materials or have too many finished goods lying around, then these items are piling up in the storage and taking up too much unnecessary space.

- **Motion**
  Unnecessary movement is movement of humans and/or machines, that is inefficient. For example, grabbing a heavy object off the floor, instead of doing it from an appropriate height. Lifting it from an appropriate height would put less strain on the person and make the activity faster and more effective.

- **Waiting**
  Waiting occurs when everything in the manufacturing process is not working optimally at the same time. This could occur as a result of damaged machines, inefficient manufacturing methods, or insufficient amounts of materials.

- **Overprocessing**
  Overprocessing is the usage of, for instance, unsuitable techniques, unfitting tools, or doing processes that are not requested by the customer. All of the above are unnecessary and will cost time and money.

- **Overproduction**
  If a company supplies more product than demanded by a customer, then it will result in an overuse of materials, energy, and human work.

- **Defects**
  When a defect occurs, it has to be accounted for. This is done either by reimbursement to the client, reparation of the defect, or replacement through overproduction. Hence, it results in a waste of time, money, and materials.

Elimination or reduction of the seven types of waste mentioned above is crucial for any manufacturing company, and if successfully done, it would lead to a more time and cost efficient production.

A variety of tools have been created to reduce waste in Lean Manufacturing, these are called Lean tools [9]. Some of the most popular tools are 5S, the 8-Step Problem Solving Process, and OEE. While many of these tools primarily focus on eliminating or reducing waste, the focus of OEE lies primarily on effectiveness. However, a consequence of increasing the effectiveness of a manufacturing process often results in a reduction of waste.

1.2.2 Overall Equipment Effectiveness

This secion is based on [10] unless otherwise stated.

Overall Equipment Effectiveness, referred to as OEE, has been well known in the manufacturing industry since the 1980s, and the concept itself originated from Japan [11].

OEE is a metric used by manufacturing companies to identify the effectiveness of their production for the purpose of improving their production. OEE is based on the three most significant factors for manufacturing losses. These are availability, performance, and quality. OEE is based on calculations of these factors and is a number from 0% to 100%, where an OEE of 100% would be described as a perfect production.

Availability is the measure of how much time the manufacturing equipment is available. This measure considers all events that stop the planned production. These events are, for instance, equipment failures, material shortages, unplanned and planned stops. The availability is calculated as the ratio

$$\text{Availability} = \frac{Run\,time}{Planned\,production\,time}, \tag{1.1}$$

where $Run\,time = Planned\,production\,time - Stop\,time$.

Performance is the measure of the ability of the manufacturing to operate at the ideal possible speed. Events that could inhibit the performance are, for instance, poor quality of raw materials, wrong temperatures, or wear and tear. The performance is calculated as

$$\text{Performance} = \frac{Ideal\,cycle\,time \cdot Total\,count}{Run\,time}, \tag{1.2}$$

where $Ideal\,cycle\,time$ is the ideal time to manufacture one piece and $Total\,count$ is the total amount of manufactured pieces.

Quality is the measure of how many manufactured pieces that meet the quality standard. Pieces that do not meet the quality standard includes pieces that are reworked, downgraded, or defects. The quality is calculated as the ratio

$$\text{Quality} = \frac{Good\,count}{Total\,count}. \tag{1.3}$$

Finally, the OEE, which takes into account the availability, performance, and quality resulting in an overall measure of the truly productive manufacturing time, can be calculated as

$$\text{OEE} = \text{Availability} \cdot \text{Performance} \cdot \text{Quality}.$$

By substituting in Equation (1.1), Equation (1.2), and Equation (1.3), the OEE can be calculated as

$$\text{OEE} = \frac{Ideal\,cycle\,time \cdot Good\,count}{Planned\,production\,time}.$$

1.2.3 Implementation of OEE

There are many companies such as Vorne Industries, Amper [12], Sistemas OEE [13], and OAL [14] that sell both hardware and software solutions to improve manufacturing effectiveness. All these companies use OEE as a tool to monitor and improve manufacturing lines with their own solutions.

Vorne Industries, for instance, is a company that specialises in OEE calculations for the purpose of improving the efficiency of manufacturing companies. Vorne has accomplished this by developing a tool known as XL Productivity Appliance. This tool can be applied to any manufacturing system to track production and monitor lost time. The collected data can be accessed immediately from reports that display real-time values such as OEE and down time. The data is visualised with live dashboards that are customised by the client [15].

Based on the continuance of different companies whose focus lies on improving manufacturing effectiveness, it can be concluded that keeping track of manufacturing numbers is essential for a company. However, having the ability to forecast future OEE results and other manufacturing numbers would allow a company to make manufacturing changes without having to look at real-time manufacturing numbers. This can be accomplished by the use of simulation.

1.3 Simulation

"Imitation of a situation or process." such is the definition of simulation [16]. Simulation can be used for many things and in many different ways. The applications range from computer simulations to real life simulations and virtual simulations [17].

Real life simulations are, for example, drills in the military where instead of sending the soldiers directly into a battlefield without any prior experience, an environment is simulated to resemble a battlefield. By simulating different scenarios and environments, experience can be gained without any direct danger or a need to be at the exact location. This is also why environments that resemble the conditions of space are created for astronauts, since it is very expensive to send humans into space.

Virtual simulations however are virtual environments generated from a computer to resemble an environment which can be directly interacted with. This is what is used in games, flight simulators, and virtual reality. For example, it was discovered that using a simulated firing range with dry-fire laser-based pistols instead of a real firing range and live weaponry had no effect on, if the new cadets could pass the pistol course-of-fire [18].

Computer simulations are used for experimenting and testing, this is where either a real situation or a hypothetical one is simulated with specific parameters, which can be altered and then changes will occur according to the implemented mathematical model, which is used for the simulation. Computer simulations are also used in companies to test a manufacturing system before creating it, to estimate and examine how effective the system performs before implementing it [19].

1.3.1 Simulation of Manufacturing Systems

Manufacturing simulation is the process of using a computer model to understand and improve a real manufacturing system. Simulation technology allows manufacturing companies to analyse and experiment with their processes in a virtual setting, reducing the time and cost of physical testing.

Simulating a manufacturing system ensures that every facet is being carefully considered and optimised. In addition to this, it is also an inexpensive and risk-free way to put a facility to the test, ensuring that production goals and quality standards are being met at the lowest possible cost. Simulation also offers a quick and efficient way to adjust parameters and simulate again, unlike forecasting that is based on spreadsheet-based analysis that requires substantially more manual inputs [20].

In order to create a simulation of a manufacturing system, knowledge of the specific domain that the simulation is imitating is required. Without this knowledge the simulation will be inaccurate. For example, in order to create a quality simulation of a manufacturing system, a person with domain knowledge in manufacturing is needed [21].

2 | Problem Statement

Throughout time, the concept of manufacturing has been under constant development, which has led to the creation of different manufacturing methods. Furthermore, different methodologies have been developed to reduce waste within manufacturing, one of the most prominent being Lean manufacturing. Lean builds on the principles of limiting any activity that requires time, money, or talent that does not add value to the customer. A useful metric based on Lean principles is OEE. By computing the OEE it is possible to measure the effectiveness of a manufacturing process. Furthermore, by implementing the OEE metric in a simulation of manufacturing processes, it is possible to adjust the parameters of the mathematical model used in the simulation, which allows for analysis of different organisational changes. This leads to the following problem statement:

How is it possible to develop a software solution capable of simulating the OEE of manufacturing processes in order to test organisational changes, thus reducing waste?

3 | System Description

In line with the problem statement, the purpose of this project is to design a software solution that is able to simulate the OEE of manufacturing processes in order to test organisational changes. In this chapter, the software solution based on the aforementioned problem statement will be established.

3.1 Delimitation of the Software Solution

In Section 1.1 a description of different manufacturing types was given. Developing a program that can be implemented on all of these manufacturing types (continuous, mass, batch, job, and project), would require a comprehensive method that takes many different parameters into account. Therefore, the software solution will be narrowed down to mainly target batch and mass manufacturing, since both manufacturing types are similar in many aspects. For instance, mass manufacturing and batch manufacturing are low in product variety. Additionally, both types of manufacturing are high in quantity. Batch manufacturing and mass manufacturing are chosen for this program, because the principles of Lean and implementations of OEE can be applied to both manufacturing types.

Additionally, the simulation in the software solution is delimited to simulating manufacturing models consisting of independent, successive manufacturing processes, contrary to simulating more complex manufacturing models consisting of interdependent processes.

3.2 Visual Representation of System

In order to envision the software solution, a series of flowcharts are presented. The purpose of these flowcharts is to give a visual representation of the software solution. First off, some common symbols need to be clarified in regards to what they represent:

- The oval, Figure 3.1, is the symbol for start/stop. It represents the start or end of the program.

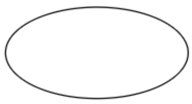

Figure 3.1: Oval.

- The rectangle, Figure 3.2, represents the call of a function.

Figure 3.2: Rectangle.

- The diamond, Figure 3.3, is the symbol for decisions. This symbol is used to check if a certain condition is true or false.

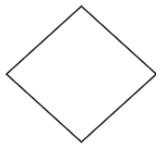

Figure 3.3: Diamond.

- The parallelogram, Figure 3.4, is the symbol for input/output. It represents information which the system reads as input or sends as output.

Figure 3.4: Parallelogram.

- The circle, Figure 3.5, symbolises the connection between two flowcharts. The character inside the circle indicates which connection to go to.

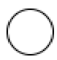

Figure 3.5: Circle.

- The arrow, Figure 3.6, is used to indicate the order that the different functions are called.

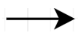

Figure 3.6: Arrow.

In Figure 3.7 a representation of the start of the program is illustrated. When the program is executed, the program starts and shows the main menu. The user navigates the program by typing certain numbers. The purpose of the function getch is to circumvent the need to press enter after a number is typed. The user has three options to choose from in the main menu:

- If "1" is typed, the program will run the function modelMenu, which is shown in Figure 3.9.
- If "2" is typed, a manual will be displayed.
- If "9" is typed, the program will shut down.

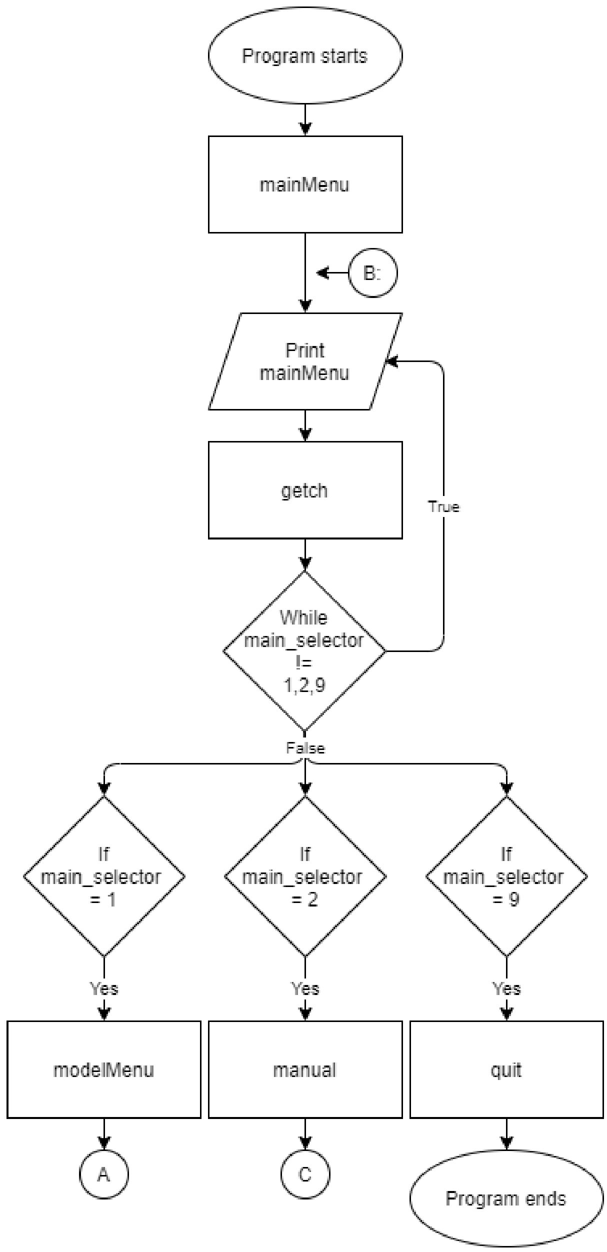

Figure 3.7: Program start and main menu.

If the user has chosen the manual, then the printManual function will be called, which prints the manual in the terminal. This part of the program is illustrated in Figure 3.8. From the manual, the user has the option to return to the main menu by typing "1".

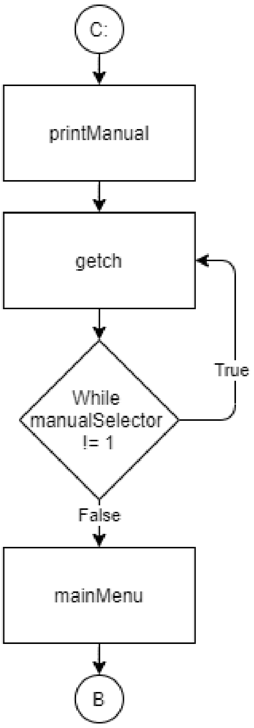

Figure 3.8: Manual.

In Figure 3.9 the modelMenu function, that is called when chosen from the main menu, is illustrated. The mode menu allows the user to create a visual model of a manufacturing system, by entering the amount of manufacturing processes the system contains. Furthermore, the user has the option to go back to the main menu or to quit the program. This is accomplished by changing the value of the variable called modelSelector, which gives the user three options:

- If "1" is typed, the user is prompted to enter the amount of manufacturing processes the system contains, and the system will create and display a model in the terminal.
- If "8" is typed, the user is brought back to the main menu.
- If "9" is typed, the program will shut down.

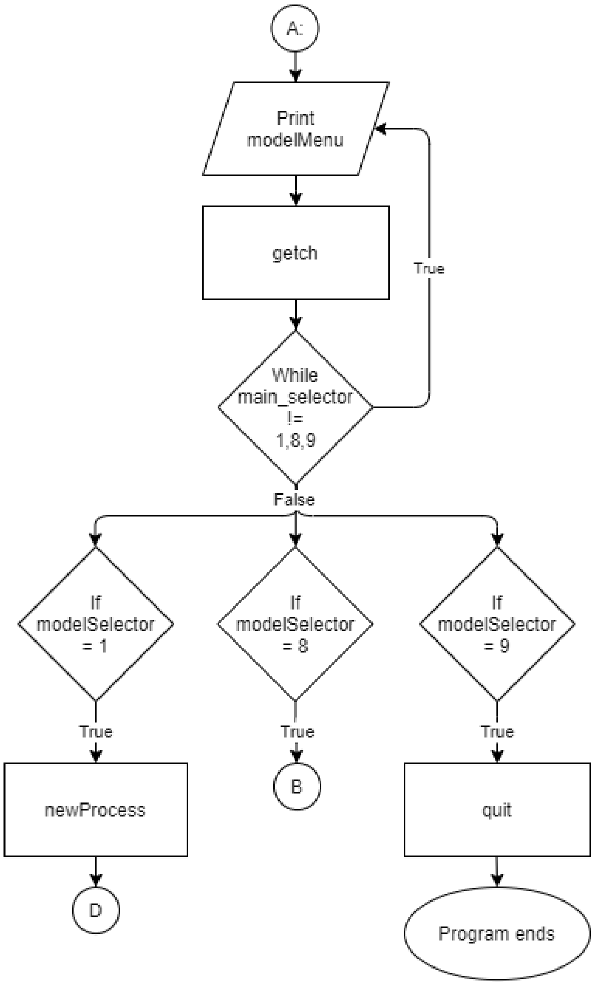

Figure 3.9: Model menu.

As mentioned before, the modelled system will be displayed on the screen after the user has chosen the desired amount of processes. The visual model will be created by the function newProcess.

After the amount of processes has been chosen and the model has been displayed in the terminal, the user has two options to choose from:

- If "1" is typed, the user is brought back to the model menu.
- If "2" is typed, the user goes forward to the next menu, which is the data menu.

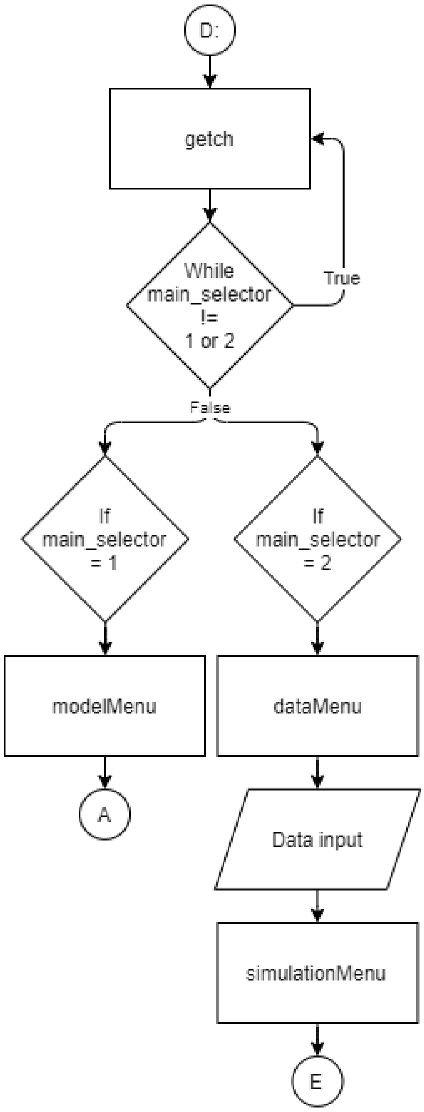

Figure 3.10: Data menu.

The data menu is seen in Figure 3.10. When the user has entered the desired amount of processes, they will be prompted for the required data in the data menu. Upon doing so the simulation menu will be called. Here the menu will be printed as seen in Figure 3.11. The options in the simulation menu are as follows:

- If "1" is typed, the program will run the simulation and print the results.
- If "8" is typed, the program will go back to the data menu.
- If "9" is typed, the program will shut down.

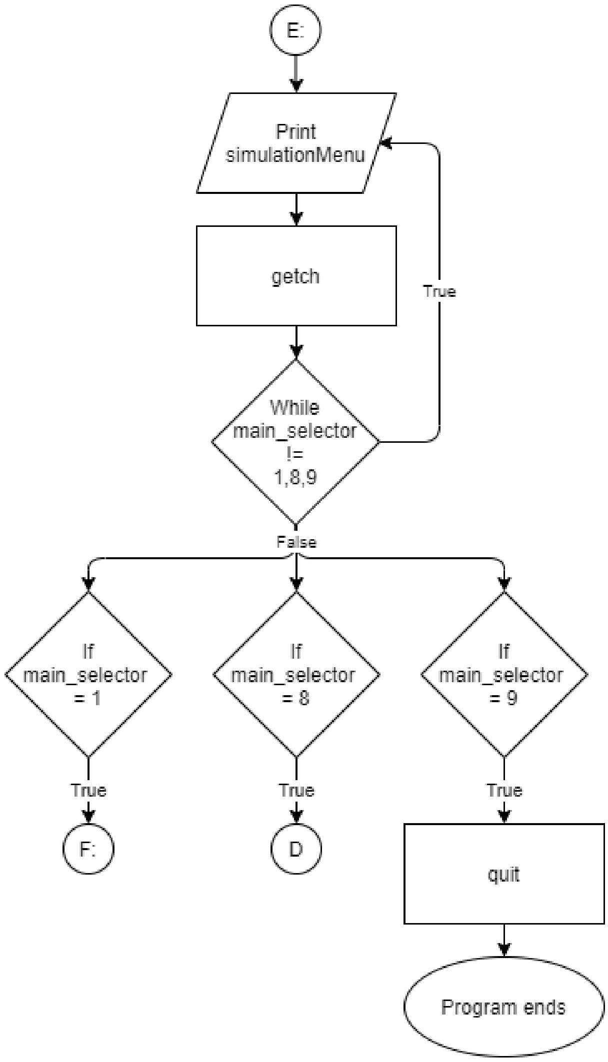

Figure 3.11: Simulation menu.

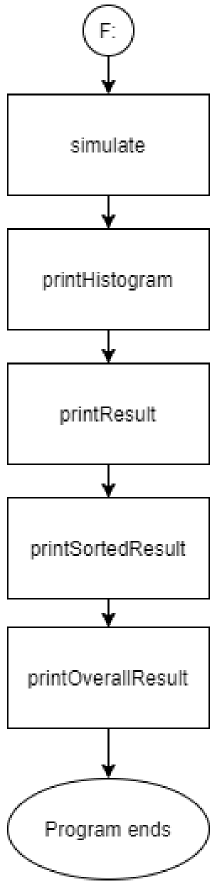

Figure 3.12: Running of simulation and printing of results.

If the user decides to run the simulation and print the results, the part of the program shown in Figure 3.12 is executed. To illustrate the results of the simulation, a table is generated including the results of each process. These results include the OEE, availability, performance, and quality. Furthermore, another table will be created, which also includes the OEE, availability, performance, and quality for each process. However, in this table the results are sorted using a sorting algorithm. More specifically, the results are sorted based on the OEE to illustrate which part of the manufacturing model is the most ineffective.

Lastly, the values for OEE, availability, performance, and quality for each process are used to calculate the overall values for OEE, availability, performance, and quality for the entire manufacturing system. The overall values will also be displayed.

3.3 System Requirements

In order to be able to develop the software solution that is described with flowcharts in Section 3.2, some more specific requirements for the system are needed. These system requirements are laid out in functional and non-functional requirements.

A functional requirement is a description of what the system must or must not do. If these requirements are not met, then the system will not function correctly. A non-functional requirement defines how the system should do it. Therefore, the non-functional requirements do not affect the basic functionality of the system [22].

3.3.1 Functional Requirements

In this subsection the functional requirements for the software solution are listed.

1. Include a simple user interface for navigating the program.
2. Display a manual from a text file.
3. Create and display a manufacturing model consisting of independent, successive manufacturing processes based on user inputs.
4. Assign a probability distribution with parameters to each process in the manufacturing model.
5. Simulate the modelled manufacturing system.
6. Compute the availability, performance, quality, and OEE of each process and the entire model.
7. Sort the results and display them in a table.
8. Exit while not receiving data inputs from the user, printing, or simulating.

3.3.2 Non-functional Requirements

The non-functional requirements for the software solution are listed below.

9. Navigate the program with input from a keyboard without the necessity to press the enter key.
10. Perform the simulation at least 100,000 times.

4 | Probability Theory

This chapter is based on [23].

Randomness is what happens in a situation where it is not possible to predict the outcome with certainty. The area of mathematics concerned with randomness is probability theory.

The purpose of this chapter is to get a basic understanding of probability theory. This theory will be used in the development of the software solution of this project. More specifically, it will be utilised to create a model for the simulation of manufacturing processes, where certain probability distributions are needed to simulate the outcome in terms of defective pieces and unplanned stops.

4.1 Sample Spaces and Events

Probability theory is used to describe different situations where randomness is present. Such situations will be referred to as random experiments. The result of a random experiment will be referred to as an outcome. For any given random experiment there is a set of possible outcomes. The following is a definition of the aforementioned set.

Definition 4.1 (Sample Space)
The set of all possible outcomes in a random experiment is called the sample space, denoted S.

[23, p. 3]

Example 4.1 Consider the random experiment where a die is rolled and the number is observed. In this case the possible outcomes are the numbers 1 through 6, hence the sample space is

$$S = \{1,2,3,4,5,6\}.$$

$\triangle$

The sample space is the set of all possible outcomes. However, when computing probabilities it is often desired to compute the probability of a single outcome or groups of outcomes. The subset of a single outcome or a group of outcomes is defined below.

Definition 4.2 (Event)
A subset of S, $A \subseteq S$, is called an event.

[23, p. 5]

Example 4.2 Consider again the random experiment where a die is rolled and the number is observed. Two events that are possible in this case are to get an even outcome and roll at least 5. The two subsets of the sample space that the two aforementioned events respectively create are

$$A = \{2,4,6\} \quad \text{and} \quad B = \{5,6\}.$$

Alternatively, the two sets can be written with a verbal description as

$$A = \{\text{even outcome}\} \quad \text{and} \quad B = \{\text{at least } 5\}.$$

△

4.2 The Axioms of Probability

In the previous section the basis needed to describe random experiments in terms of sample spaces, outcomes, and events are presented. In this section the basis needed to be able to compute probabilities are presented. This basis is a definition of probability in terms of a real-valued function, which satisfies three properties referred to as the axioms of probability.

Definition 4.3 (Axioms of Probability)
A probability measure is a function P, which assigns to each event A a number $P(A)$ satisfying

1. $0 \leq P(A) \leq 1$
2. $P(S) = 1$
3. If $A_1,A_2,\ldots$ is a sequence of pairwise disjoint events, that is, if $i \neq j$, then $A_i \cap A_j = \emptyset$, then

$$P(\bigcup_{k=1}^{\infty} A_k) = \sum_{k=1}^{\infty} P(A_k)$$

[23, p. 7]

As seen in Definition 4.3 there are three different axioms of probability. The first axiom is that the probability of any event is at least 0 and at most 1, which implies that the probability of any event is a nonnegative number. The second axiom is that the probability of the entire sample space is 1, which implies that that the sample space encompasses every possible outcome of a random experiment. The third and last axiom is that if some events are disjoint, then the probability of their union equals the probability of the summation of the probability of each event.

4.3 Discrete Random Variables

A convenient notation, that is utilised to denote different outcomes of specific events, is a random variable X. The value of the quantity X is thus not known before an experiment, but it becomes known after.

Definition 4.4 (Random Variable)
A real-valued random variable X is a function from the sample space S to $\mathbb{R}$ ($X : S \to \mathbb{R}$).

The definition of the random variable allows for the notation of an event as $\{X = x_k\}$ and, therefore, the notation of a probability measure as a function of an event as $P(X = x_k)$, where x_k is some outcome in the range of X.

Example 4.3 Consider once more the random experiment where a die is rolled, but let this time X denote the number. Then the range of X is $\{1,2,3,4,5,6\}$ and the associated probabilities are

$$P(X = 1) = P(X = 2) = P(X = 3) = P(X = 4) = P(X = 5) = P(X = 6) = \frac{1}{6}.$$

$\triangle$

Random variables are primarily distinguished between as random variables with countable range and random variables with uncountable range.

Definition 4.5 (Discrete Random Variable)
If the range of X is countable, then X is called a discrete random variable.

[23, p. 78]

When computing probabilities $P(X = x_k)$ for various values x_k in the range of X, it is convenient to view $P(X = x_k)$ as a function, which leads to the following definition.

Definition 4.6 (Probability Mass function)
Let X be a discrete random variable with range $\{x_1, x_2, \dots\}$ (finite or countably infinite). The function

$$p(x_k) = P(X = x_k), \quad k = 1,2,\dots$$

is called the probability mass function (pmf) of X.

[23, p. 78]

Example 4.4 Consider the random experiment of flipping a coin three times and let X denote the number of heads. Then the range of X is $\{0,1,2,3\}$ and the corresponding values of the pmf are

$$p(0) = \frac{1}{8}, p(1) = \frac{3}{8}, p(2) = \frac{3}{8}, p(3) = \frac{1}{8}.$$

$\triangle$

So far only events of the type $\{X = x_k\}$ have been considered. Another type of event is the type $\{X \leq x_k\}$, which is the event that X is less than or equal to some outcome x_k. The probability of such an event is also defined as a function.

Definition 4.7 (Cumulative Distribution Function)
Let X be any random variable. The function

$$F(x) = P(X \leq x), \quad x \in \mathbb{R}$$

is called the cumulative distribution function (cdf) of X.

[23, p. 80]

Example 4.5 Consider the aforementioned random experiment again, which is flipping a coin three times and letting X denote the number of heads. The range of X is $\{0,1,2,3\}$, and the corresponding values of the cdf are as follows.

$$F(0) = P(X \leq 0) = p(0) = \frac{1}{8},$$

since the only way to be less than or equal to 0 is to be 0. The next value $F(1) = P(X \leq 1)$ is equal to

$$p(0) + p(1) = \frac{1}{8} + \frac{3}{8} = \frac{1}{2},$$

since the events are disjoint, which then allows, by the third axiom of probability, a summation of the probability of each event. The values $F(2) = \frac{7}{8}$ and $F(3) = 1$ are computed likewise. △

Some properties of the cdf that apply for any random variable are described in the following theorem.

Theorem 4.8
Let X be any random variable with cdf F. Then

1. $P(a < X \leq b) = F(b) - F(a), \quad a \leq b$
2. $P(X > x) = 1 - F(x). \quad x \in \mathbb{R}$

[23, p. 84]

4.4 Continuous Random Variables

As mentioned before, random variables are distinguished by their range. A discrete random variable has a countable range, whereas a continuous random variable has an uncountable range. However, this is not sufficient to define a continuous random variable.

Definition 4.9 (Continuous Random Variable)
If the cdf F is a continuous function, then X is said to be a continuous random variable.

[23, p. 83]

The cdf is defined identically for discrete and continuous random variables. For discrete random variables the pmf is also defined, which is a function that assigns probabilities for values in the range of a discrete random variable. The continuous analogy to this function is the probability density function.

Definition 4.10 (Probability Density Function)
The function $f(x) = F'(x)$ is called the probability density function (pdf) of X.

[23, p. 84]

In Figure 4.1 the relation between a pdf and the corresponding cdf is illustrated. In accordance with Definition 4.10 the pdf follows the slope of the cdf.

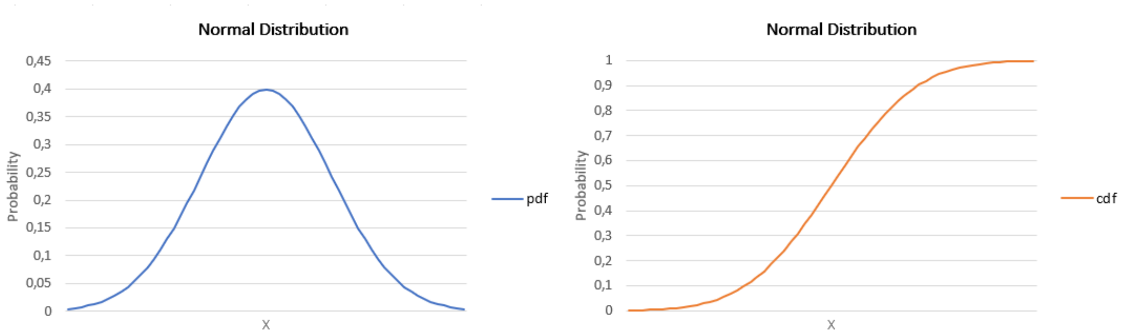

Figure 4.1: A pdf and the corresponding cdf.

4.5 Probability Distributions

A probability distribution is a function that gives the probability of the different possible outcomes of a random experiment or event. Some important terms needed in order to describe some different probability distributions are now presented.

Definition 4.11 (Expected Value)
Let X be a continuous random variable with pdf f. The expected value of X is defined as

$$E[X] = \int_{-\infty}^{\infty} x f(x) dx.$$

[23, p. 98]

The expected value is a generalisation of a weighted average. The expected value is sometimes denoted by μ. The integral limits are $-\infty$ and ∞, however, when applying the definition in practice, the limits are determined by the range of X. It is necessary to choose limits such that $f(x)$ is positive between the limits and not outside of the limits.

The expected value gives information on where X is on average. To get more information about a probability distribution, a measure for the variability of the random variable is beneficial. This variability of a random variable is called the variance.

Definition 4.12 (Variance)
Let X be a random variable with expected value μ. The variance of X is defined as

$$Var[X] = E[(X - \mu)^2].$$

[23, p. 105]

The variance is sometimes denoted as σ^2.

Since the values of $Var[X]$ are squared, they do not express the same unit of measure as the data. Thus, the following definition is often used.

Definition 4.13 (Standard Deviation)
Let X be a random variable with variance $\sigma^2 = Var[X]$. The standard deviation of X is then defined as $\sigma = \sqrt{Var[X]}$.

[23, p. 105]

A probability distribution, where the probability of each outcome is equivalent, is called a uniform distribution. The pdf is in this case constant on an interval.

Definition 4.14 (Uniform Distribution)
If the pdf of X is

$$f(x) = \frac{1}{b-a}, \quad a \leq x \leq b$$

then X is said to have a uniform distribution on $[a,b]$, written $X \sim \text{unif}[a,b]$.

[23, p. 91]

In Figure 4.2 the pdf of a uniform probability distribution is illustrated.

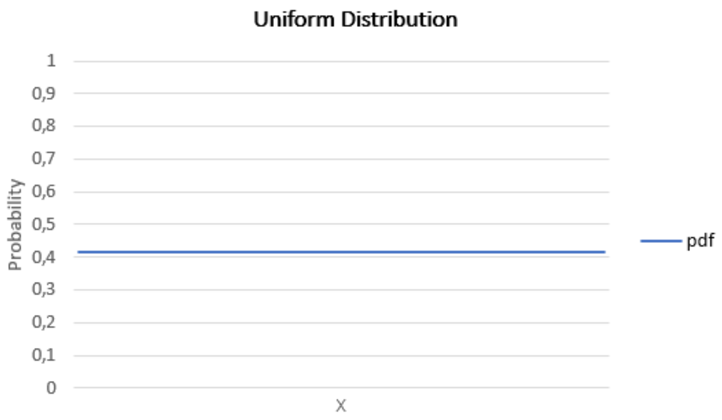

Figure 4.2: The pdf of a uniform probability distribution.

The following probability distribution is a distribution that pertains to a huge variety of situations. For instance, it is often applied in models of situations where there are measurement errors due to randomness or noise [24] or used to describe the behaviour of stock market prices [25]. Furthermore, it can, for instance, be used to describe data from pharmaceutical manufacturing processes [26].

Definition 4.15 (Normal Distribution)
If X has the pdf

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}$$

it is said to have a normal distribution with parameters μ and σ^2, written $X \sim N[\mu,\sigma^2]$.

[23, p. 127]

In Figure 4.1 the pdf of a normal distribution is illustrated.
The pdf of the normal distribution is symmetric around the expected value μ.

Theorem 4.16
If $X \sim N[\mu,\sigma^2]$, then $E[X] = \mu$ and $Var[X] = \sigma^2$.

[23, p. 128]

Note that the second parameter, in the definition of the normal distribution, is the variance. In some other cases the normal distribution is defined with the standard deviation as the second parameter.

Another probability distribution is the exponential distribution. This type of probability distribution is applicable for modelling processes that involve time interval between events. This could, for instance, be the time interval between two manufacturing processes [27].

Definition 4.17 (Exponential Distribution)
If the pdf of X is

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

then X is said to have an exponential distribution with parameter $\lambda > 0$, written $X \sim \exp(\lambda)$.

[23, p. 124]

In Figure 4.3 the pdf of an exponential probability distribution is illustrated.

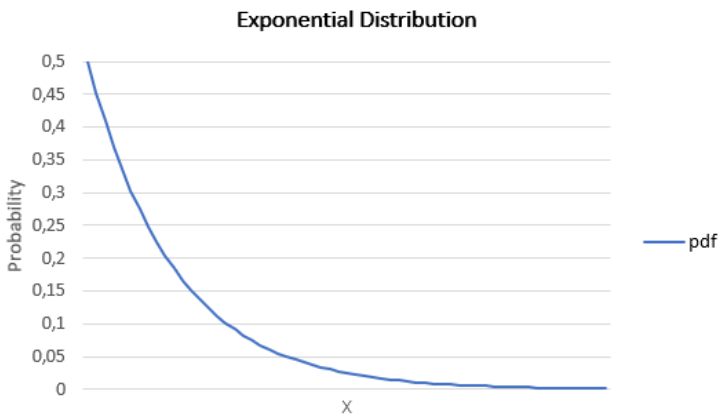

Figure 4.3: The pdf of an exponential probability distribution.

5 | Algorithms

An algorithm is a finite sequence of instructions, which are implementable in a computer, needed to perform calculations or to solve a specific type of problem.

This chapter describes some different algorithms that are needed in order to develop the software solution for this project. These algorithms will be used in the software solution for sorting data, simulating manufacturing processes, and sampling from probability distributions.

5.1 Complexity of Algorithms

This section is based on [28, Ch. 3].

When working with algorithms it is important to be able to assess the computational complexity of the algorithms. The computational complexity is utilised to measure the computer memory and processing time required by the algorithms to solve specific problems of a particular size. Hence, when the computational complexity of different algorithms are obtained, they can be compared in order to determine which is the most efficient.

In order to assess the computational complexity of algorithms, an estimation of the number of operations used by the algorithms is needed. The amount of operations used by an algorithm can be expressed in terms of a function. Big-O notation is used to estimate the amount of operations an algorithm uses with regards to the size of the input.

Definition 5.1 (Big-O Notation)
Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $O(g(x))$ if there are constants C and k such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > k$.

[28, p. 217]

Big-O notation is used to describe the growth of a function. Particularly, when some function $f(x)$ is $O(g(x))$, the function $f(x)$ is asymptotically bounded by the function $g(x)$. Thus, an upper bound is not ensured for all values of x, but only in the limit.

Example 5.1 Consider the function $f(x) = 2x^2 + 3x + 2$ and assume that it is desired to determine whether $f(x)$ is $O(x^2)$. In order for this to be true, Definition 5.1 has to be satisfied. Hence, there has to exist constants C and k such that

$$|f(x) = 2x^2 + 3x + 2| \leq C|x^2|, \quad \text{for } x > k.$$

It is observed that when $x > 1$ then $1 < x^2$ and $x < x^2$. Furthermore, taking the absolute value can be omitted, since both functions are positive when $x > 1$. Thus, it follows that

$$2x^2 + 3x + 2 \leq 2x^2 + 3x^2 + 2x^2 = 7x^2 \quad \text{for } x > 1.$$

Consequently, the constants can be chosen as $C = 7$ and $k = 1$. $\triangle$

If a lower bound is desired on some function $f(x)$, then big-Omega notation can be used.

Definition 5.2 (Big-Omega Notation)
Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Omega(g(x))$ if there are constants C and k with C positive such that

$$|f(x)| \geq C|g(x)|$$

whenever $x > k$.

[28, p. 227]

The constants C and k in Definition 5.1 are referred to as witnesses to the relationship $f(x)$ is $O(g(x))$. The constants C and k in Definition 5.2 are likewise referred to as witnesses to the relationship $f(x)$ is $\Omega(g(x))$.

In order to establish that some function $f(x)$ is $O(g(x))$, it is only necessary to find one pair of witnesses C and k such that $|f(x)| \leq C|g(x)|$ whenever $x > k$. Likewise, to establish that some function $f(x)$ is $\Omega(g(x))$, it is only necessary to find one pair of witnesses $C > 0$ and k such that $|f(x)| \geq C|g(x)|$ whenever $x > k$.

In order to show this, assume, without loss of generality, that C and k are one pair of witnesses to the relationship $f(x)$ is $O(g(x))$. Then, any pair C_1 and k_1, where $C < C_1$ and $k < k_1$, is also a pair of witnesses, since $|f(x)| \leq C|g(x)| \leq C_1|g(x)|$ whenever $x > k_1 > k$. Thus, only one pair of witnesses is needed.

If it is desired to obtain both a lower bound and an upper bound on the size of some function $f(x)$ in respect to a reference function $g(x)$, then big-Theta notation can be applied.

Definition 5.3 (Big-Theta Notation)
Let f and g be functions from the set of integers or the set of real numbers to the set of real numbers. We say that $f(x)$ is $\Theta(g(x))$ if $f(x)$ is $O(g(x))$ and $f(x)$ is $\Omega(g(x))$. When $f(x)$ is $\Theta(g(x))$, we say that f is big-Theta of $g(x)$, that $f(x)$ is of order $g(x)$, and that $f(x)$ and $g(x)$ are of the same order.

[28, p. 227]

Example 5.2 Consider again the function $f(x) = 2x^2 + 3x + 2$, which previously was determined to be $O(x^2)$. Assume now that it is desired to determine whether $f(x)$ is also $\Omega(g(x^2))$. In order to determine this, $f(x)$ has to satisfy Definition 5.2. Hence, there has to exist a pair of witnesses such that

$$|2x^2 + 3x + 2| \geq C|x^2|, \quad \text{for } x > k \text{ and } C > 0.$$

It is obvious to see that by choosing $C = 1$ then the equation is satisfied for all positive real numbers x, thus it suffices to choose $k = 0$.

It has now been determined that $f(x)$ is $O(x^2)$ and $f(x)$ is $\Omega(g(x^2))$, thus by Definition 5.3, $f(x)$ is $\Theta(g(x^2))$. △

5.2 Sorting Algorithms

When working with data, sorting is often applied. Techniques that sort data are known as sorting algorithms [29, p. 134].

A sorting algorithm is an algorithm that puts elements of a list in a certain order. The order is determined by some criterion. This could, for instance, be alphabetical or numerical order. Within computer science sorting algorithms are thus often used to produce more understandable output. To give an example, the grades from a class of students can be sorted from the highest grade to the lowest grade [30].

Throughout time, a lot of different sorting algorithms have been developed. Some of the different sorting algorithms are listed below:

- Selection sort
- Bubble sort
- Insertion sort
- Quicksort
- Heapsort
- Mergesort

Some of the algorithms have different time complexities. Therefore, some of the algorithms sort faster than others due to a lower complexity. In Table 5.1 a comparison of the different time complexities is shown.

Algorithm	Best-case	Average-case	Worst-case
Selection sort	$\Omega(n^2)$	$\Theta(n^2)$	$O(n^2)$
Bubble sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Insertion sort	$\Omega(n)$	$\Theta(n^2)$	$O(n^2)$
Quicksort	$\Omega(n \log_2(n))$	$\Theta(n \log_2(n))$	$O(n^2)$
Heapsort	$\Omega(n \log_2(n))$	$\Theta(n \log_2(n))$	$O(n \log_2(n))$
Mergesort	$\Omega(n \log_2(n))$	$\Theta(n \log_2(n))$	$O(n \log_2(n))$

Table 5.1: Algorithm and time complexity [31].

As shown in Table 5.1, quicksort is one of the most efficient sorting algorithms. The quicksort algorithm is implemented in the qsort function, which is defined in the standard library in C.

5.2.1 Qsort

The qsort function is designed to sort an array of elements of any data type. The qsort function takes in four parameters in order to sort an array [32]:

- The array that needs to be sorted.
- The total amount of elements in the array.
- The size in bytes of each element in the array.
- A compare function.

The compare function is a function that compares two elements, which determines the order that the array is sorted. In order to understand the quicksort algorithm that qsort implements, the algorithm is illustrated in Figure 5.1. The figure shows how the quick sort algorithm sorts after a pivot is selected. In this example, the pivot is always chosen as the rightmost element. However, the algorithm reaches a lower time complexity when an element close to the median is selected as the pivot.

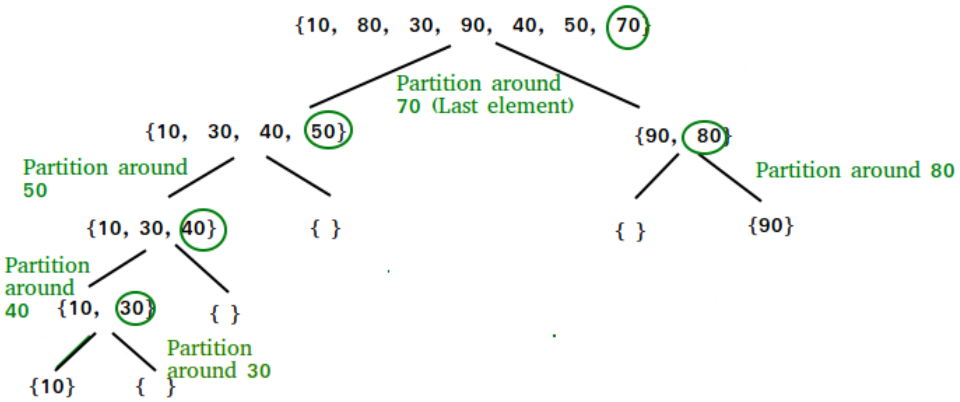

Figure 5.1: Quicksort algorithm graphical [33].

5.3 Monte Carlo Simulation

This section is based on [34, p. 507-508].

Monte Carlo simulation is an algorithm where an experiment is conducted N times and the outcome of each experiment is random. Suppose a certain event occurs M times. Then an estimate of the probability of M occurring can be found as M/N.

Assume that the purpose of an experiment is to count how many times a six is rolled, let this number be denoted by M. If the die is rolled nine times and the outcome, where a six is rolled, occurs three times, then $M = 3$. According to this experiment the probability of rolling a six is estimated to be $M/N = 1/3$. However, the actual probability of rolling a six is $1/6$. This discrepancy occurs due to the limited amount of simulations performed.

If the amount of simulations performed is increased, then the estimation of the probability will become more accurate. This is known as the law of large numbers. According to the law, the average of a random experiment approaches the expected value as $N \to \infty$.

5.4 Inverse Transform Sampling

This section is based on [35, Ch. 19.3].

Inverse transform sampling is a method for generating random samples from any probability distribution given its cdf. In order to derive the method of inverse transform sampling, some preliminary theory of functions is needed.

5.4.1 Preliminary Theory of Functions

Functions are also sometimes called transformations. Inverse transform sampling is feasible due to some different properties of the functions that are involved in the method. These different properties are presented below.

The first property is injection. An injective function maps distinct elements of its domain to distinct elements of its codomain.

Definition 5.4 (Injective)
A function f is said to be one-to-one, or an injection, if and only if $f(a) = f(b)$ implies that $a = b$ for all a and b in the domain of f. A function is said to be injective if it is one-to-one.

[28, p. 150]

Another class of functions, that is important for deriving the inverse transform sampling method, is monotone functions.

Definition 5.5 (Monotone Function)
A function $f : A \to \mathbb{R}$, where $A \subseteq \mathbb{R}$, is increasing, if for $x_1, x_2 \in A$ applies that

$$x_1 < x_2 \Rightarrow f(x_1) \leq f(x_2),$$

and strictly increasing if

$$x_1 < x_2 \Rightarrow f(x_1) < f(x_2).$$

Decreasing ($\geq$) and strictly decreasing ($>$) are defined analogously. A function is said to be monotone if it is increasing or decreasing. And a function is said to be strictly monotone if it is strictly increasing or strictly decreasing.

A class of functions, that maps distinct elements of its domain to distinct elements of its codomain, is strictly monotone functions. Hence, strictly monotone functions are injective.

Injective functions have a certain property, which is presented in the following definition.

Definition 5.6 (Inverse Function)
Let f be an injective function from the set A to the set B. The inverse function of f is the function that assigns to an element b belonging to B the unique element a in A such that $f(a) = b$. The inverse function of f is denoted by f^{-1}. Hence, $f^{-1}(b) = a$ when $f(a) = b$.

[28, p. 150]

5.4.2 The Method

The inverse transform sampling method is used to generate random samples from any probability distribution given its cdf. Specifically, the method is used to generate values of a random variable X, whose probability distribution can be described by the cdf $F(x)$, from $U \sim$ unif[0,1]. The advantage of generating numbers from a uniform distribution is, that it can be accomplished in practice with a random number generator. However, in this case the numbers will not be completely random, but pseudorandom.

In order to perform this method, a transformation $T : [0,1] \to \mathbb{R}$, such that $T(U) = X$, is desired. By Definition 4.7 the cdf of X is

$$F(x) = P(X \leq x).$$

Applying that $T(U) = X$ yields

$$P(X \leq x) = P(T(U) \leq x).$$

The transformation T has to be strictly monotone, since the transformation then is injective, and then there exists an inverse transformation of T, which allows for

$$P(T(U) \leq x) = P(U \leq T^{-1}(x)).$$

Since $P(U \leq y) = y$ when $U \sim$ unif[0,1], then

$$P(U \leq T^{-1}(x)) = T^{-1}(x).$$

Thus, the cdf $F(x)$ is the inverse function of T. Therefore,

$$T(u) = F^{-1}(u), \quad u \in [0,1],$$

which means it is possible to generate values of X from $F^{-1}(U)$.

Example 5.3 Assume that it is desired to apply the inverse transform sampling method to generate random samples from the exponential distribution

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

The first step is to obtain the cdf of the exponential distribution. By Definition 4.10 the cdf is obtained by integrating the pdf. The cdf is therefore

$$F(x) = \int_0^x \lambda e^{-\lambda x} = 1 - e^{-\lambda x}.$$

Given the cdf $F(x)$, it is desired to generate values of a random variable X, whose probability distribution is described by the cdf $F(x)$, from $U \sim$ unif[0,1]. Thus, the transformation $T(u) = F^{-1}(u)$ is now needed. The inverse cdf F^{-1} is derived as

$$\begin{aligned} u &= 1 - e^{-\lambda x} \\ 1 - u &= e^{-\lambda x} \\ ln(1-u) &= -\lambda x \\ X &= \frac{-ln(1-u)}{\lambda}. \end{aligned}$$

Hence, it is now possible to generate values of X from $F^{-1}(U) = \frac{-ln(1-u)}{\lambda}$. △

6 | Development of Solution

In this chapter the development of the software solution is detailed. The solution is created based on the system description in Chapter 3. Furthermore, it is developed for the purpose of satisfying the system requirements that are described in Section 3.3.

This chapter is structured in such a way, that each of the following sections corresponds to a functional system requirement. The non-functional requirements are also included in the following sections.

All figures, that are referred to in this chapter, are situated in Appendix A.

The proposed solution is a software program written in ANSI C.

6.1 User Interface

When the program is executed, the main menu is displayed in the terminal. The main menu consists of three options, as showed in Figure A.1. The user selects one of the three options by using the number keys on the keyboard. Which key to press is indicated to the left of the option.

To maintain consistency throughout the program, the "Go back" and "Quit program" options are always accessed through the same keys in all the menus. Additionally, the "Go back" and "Quit program" keys are located far away from the other menu keys to prevent misclicks that would unintentionally quit the program or go back.

Furthermore, the user does not need to press the enter key after selecting an option in a menu. This is achieved by adding the getch function shown in Listing 6.1. Getch disables buffering to prevent data from being stored temporarily. Afterwards, a char is read-in and the old terminal settings are restored.

```
char getch(void){
    char ch;
    initTermios();
    ch = getchar();
    resetTermios();
    return ch;
}
```

Listing 6.1: Reading input.

In Listing 6.2 the function that displays the main menu is shown. As seen from line 4 through 12, the main menu is printed using an iterative control structure namely a do while loop. The iterative control structure ensures that the menu is printed until one of the three options is chosen. When one of the three options is chosen it is saved in a variable called main_selector initialised in line 2. Then a switch statement is run from line 14 through 31. The switch statement evaluates main_selector and the case, that is run, is determined by the value previously assigned to main_selector. The other three menus of the program are structured in a similar way.

```
1  int mainMenu(void){
2      int main_selector = 0;
3
4      do{
5          system("clear");
6          printf(ANSI_UNDERLINED_PRE"Main Menu"ANSI_UNDERLINED_POST"\
               n\n");
7          printf("1. Model system \n"
8                 "2. Manual\n\n"
9                 "9. Quit program\n");
10
11         main_selector = getch();
12     }while(main_selector != ASCII_one && main_selector != ASCII_two
           && main_selector != ASCII_nine);
13
14     switch(main_selector){
15         case ASCII_one:
16             modelMenu();
17             break;
18         case ASCII_two:
19             system("clear");
20             manual();
21             printf("\n\n1. Return to Main Menu");
22             do{
23                 main_selector = getch();
24                 if(main_selector == ASCII_one)
25                     return mainMenu();
26             }while(main_selector != ASCII_one);
27             break;
28         case ASCII_nine:
29             quit();
30             break;
31     }
32
33     return EXIT_SUCCESS;
34 }
```

Listing 6.2: Main menu

Modifications like these allows the user to easily navigate through the program. Therefore, Requirement 1 is met. Furthermore, Requirement 9 is also fulfilled, since it is not necessary to press the enter key after each input.

6.2 Manual

When "Manual" is chosen from the main menu, the manual will be displayed. The manual includes a description of how the program functions and specifies what data that needs to be obtained before using the program. The manual can be seen in Figure A.2 and Figure A.3.

Requirement 2 is met with the following code shown in Listing 6.3. More specifically, the file is being opened and read on line 7. The file is then printed when line 15 is initiated.

```c
1  void manual(void){
2      FILE *file_pointer;
3      char c;
4
5      printf(ANSI_UNDERLINED_PRE"Manual"ANSI_UNDERLINED_POST"\n\n");
6
7      file_pointer = fopen("manual.txt", "r");
8      if(file_pointer == NULL){
9          printf("Can not open file \n");
10         exit(EXIT_FAILURE);
11     }
12
13     c = fgetc(file_pointer);
14
15     while(c != EOF){
16         printf ("%c", c);
17         c = fgetc(file_pointer);
18     }
19     fclose(file_pointer);
20 }
```

Listing 6.3: Displaying the manual from a text file.

6.3 Manufacturing Model

If "Model system" is selected in the main menu, the user is brought to the model menu. From here, the user can open "Model manufacturing system", as shown in Figure A.4. In "Model manufacturing system" the program prompts the user to enter the amount of manufacturing processes the manufacturing system consists of, as shown in Figure A.5.

After entering the amount of manufacturing processes the program will display a manufacturing model of manufacturing processes, which is showed in Figure A.6.

In Listing 6.4, the code for displaying the manufacturing model is shown. The function newProcess takes a pointer *amount_of_processes as a parameter. This pointer stores the memory address of the variable amount_of_processes. An iterative control structure is then utilised to print the model. The for loop from line 8 through 17 iterates the same amount of times as the value of the function parameter. A square of asterisks is printed for each manufacturing process. The number in the middle indicates which manufacturing process the square represents.

```c
void newProcess(int *amount_of_processes){
    int i;

    printf("Enter the amount of manufacturing processes: ");
    scanf(" %d", amount_of_processes);
    printf("\n");

    for(i = 1; i <= *amount_of_processes; i++){
        printf("* * * * *\n");
        printf("*       *\n");
        printf("*%4d   *\n", i);
        printf("*       *\n");
        printf("* * * * *\n");

        if(i != *amount_of_processes)
            printf("    |\n");
    }

    printf("\n1. Return to model menu\n");
    printf("2. Finish model");
}
```

Listing 6.4: Printing of a manufacturing model.

After the manufacturing system has been modelled, the user is given two options. The user can either "Return to model menu" or "Finish model". If the user returns to the model menu, the program will initiate the model menu. From here the user can model a new manufacturing system if the first one was not desirable. If finish model is chosen, the user will be prompted to input data for each manufacturing process, as shown in Figure A.7 and Figure A.8.

As shown in Listing 6.4 the program is capable of creating and displaying a manufacturing model consisting of independent, successive manufacturing processes based on user inputs. Therefore, Requirement 3 is met.

6.4 Probability Distribution

After entering the total count of products for the entire manufacturing system, the program will prompt for some values for each process. These values are the planned production time, the ideal cycle time, and the values that are needed to form the desired probability distributions for defects and unplanned stops. By Definition 4.15 to form a normal distribution, the value of the mean and the value of the standard deviation are needed. By Definition 4.17 to form a exponential distribution, the value of lambda is needed. The prompts are illustrated in Figure A.8.

This part of the program is achieved by using sets of printf and scanf functions. The values assigned to each manufacturing process are saved in a struct. This ensures that the values are never out of scope.

Requirement 4 is fulfilled since it is possible to assign a probability distribution with parameters to each process in the manufacturing model.

6.4.1 Inverse Transform Sampling

Listing 6.5 shows the section of the program that performs inverse transform sampling. This method is described in Section 5.4. In order to perform inverse transform sampling the inverse cdf of each probability distribution is needed.

The inverse cdf of the normal distribution does not have a closed-form expression [36]. Therefore, an approximation of the function is used. The function that generates the approximation is called r8_normal_01_cdf_inverse and is found in the C library called ASA241 [37].

The function inv_cdf_normal from line 1 through 3, performs inverse transform sampling of a single sample on an approximation of the inverse cdf of the standard normal distribution. The approximation is then scaled to any normal distribution, by multiplying the standard deviation and adding the mean.

The function inv_cdf_exponential from line 5 through 7 performs inverse transform sampling of a single sample on the inverse cdf of the exponential distribution.

The function sample from line 9 through 11 generates a sample in the interval [0,1].

```
1  double inv_cdf_normal(double mean, double std_deviation, double sample){
2      return r8_normal_01_cdf_inverse(sample) * std_deviation + mean;
3  }
4
5  double inv_cdf_exponential(double lambda, double sample){
6      return -(log(sample)) / lambda;
7  }
8
9  double sample(void){
10     return (double)rand() / (double)RAND_MAX;
11 }
```

Listing 6.5: Inverse transform sampling.

6.5 Simulation

The simulation menu is shown in Figure A.9. The user is brought here after assigning values to each process. From here the user has the option to "Run Simulation" based on the modelled manufacturing system. The user also has the option "Quit program" or "Go back" if any data was entered incorrectly.

Listing 6.6 displays the code that simulates all manufacturing processes by generating data for defects and unplanned stops. This data follows either a normal distribution or an exponential distribution that is generated based on the input of the user. The amount of simulations is determined by the constant NUM_SIM.

In accordance with Section 5.3, if the amount of simulations is increased, then the estimation of a probability approaches a more accurate result. Thus, for accurate simulations, the constant is chosen to have a value of 100,000.

Since the simulation is performed at least 100,000 times, Requirement 10 is fulfilled.

```c
double simulate(process processes[], int *amount_of_processes){
    int i, j;

    srand(time(NULL));

    for(i = 0; i < *amount_of_processes; i++){
        if(processes[i].mean_defects != -1)
            for(j = 0; j < NUM_SIM; j++)
                processes[i].defectsArr[j] = inv_cdf_normal(
                    processes[i].mean_defects, processes[i].
                    std_deviation_defects, sample());
        if(processes[i].lambda_defects != -1)
            for(j = 0; j < NUM_SIM; j++)
                processes[i].defectsArr[j] = inv_cdf_exponential(
                    processes[i].lambda_defects, sample());
    }

    for(i = 0; i < *amount_of_processes; i++){
        if(processes[i].mean_US != -1)
            for(j = 0; j < NUM_SIM; j++)
                processes[i].stopsArr[j] = inv_cdf_normal(processes
                    [i].mean_US, processes[i].std_deviation_US,
                    sample());
        if(processes[i].lambda_US != -1)
            for(j = 0; j < NUM_SIM; j++)
                processes[i].stopsArr[j] = inv_cdf_exponential(
                    processes[i].lambda_US, sample());
    }

    return EXIT_SUCCESS;
}
```

Listing 6.6: Simulation of defects and unplanned stops.

Requirement 5 has been met since the modelled manufacturing system can be simulated.

6.6 Computations

Followed by the simulation is the computation of the OEE, availability, performance, and quality for each process as described in Subsection 1.2.2. In Listing 6.7 is the code needed to perform these calculations illustrated.

```c
double defects(process process){
    int i;
    double defects_total;

    for(i = 0; i < NUM_SIM; i++)
        defects_total += process.defectsArr[i];

    return defects_total / NUM_SIM;
}

double stops(process process){
    int i;
    double US_total = 0;
```

```

    for(i = 0; i < NUM_SIM; i++)
        US_total += process.stopsArr[i];

    return US_total / NUM_SIM;
}

double calculateAvailability(double run_time, process process){
    return run_time / process.planned_production_time;
}

double calculatePerformance(double run_time, process process, int total_count){
    return process.ideal_cycle_time * total_count / run_time;
}

double calculateQuality(double good_count, int total_count){
    return good_count / total_count;
}

double calculateOEE(double availability, double performace, double quality){
    return availability * performace * quality;
}
```

Listing 6.7: Computation of the OEE, availability, performance, and quality of a process.

Consequently, it is possible to compute the OEE, availability, performance, and quality for each process and the entire model, therefore, Requirement 6 is fulfilled.

6.7 Results

The results of the simulation are printed as the last procedure of the program. An example of the results of a simulation is shown in Figure A.10 and Figure A.11.

The program displays histograms for defects and unplanned stops for each process. The histograms clearly illustrate, which distribution type is chosen. Followed by this, the program displays the OEE, availability, performance, and quality for each manufacturing process in a table sorted from first to last process and in another table sorted from lowest to highest OEE.

The data is sorted with the qsort function. In Subsection 5.2.1 it is described how the qsort function implements the quicksort algorithm, and how the quicksort algorithm functions. In Listing 6.8 the comparator function that is applied in qsort can be seen from line 1 through 10. The function compares two elements to determine the order. Then in line 15 the qsort function is called. The function orders the elements of the array called indexArr based on the output of the comparator function.

```c
int comparator(const void *element1, const void *element2){
    double *tp1 = (double*)element1, *tp2 = (double*)element2;

    if(*tp1 < *tp2)
        return -1;
    else if(*tp1 > *tp2)
        return 1;
    else
        return 0;
}

void printSortedResult(int amount_of_processes, process processes[], int total_count){
    double *indexArr;
    ⋮
    qsort(indexArr, amount_of_processes, sizeof(double), comparator);
    ⋮
}
```

Listing 6.8: Compare function and qsort

Lastly, the overall OEE, availability, performance, and quality for the entire manufacturing system are displayed. Therefore, Requirement 7 is met.

In Listing 6.9 the code that displays the overall results is shown. The procedure called printOverallResult calculates the average OEE, availability, performance, and quality of all processes in the model. This is conducted from line 5 through 15. Then the results are printed in a table. This is conducted from line 17 through 22.

```c
1  void printOverallResult(int amount_of_processes, process processes[], int total_count){
2      int i;
3      double OEE, availability_total = 0, performance_total = 0, quality_total = 0, availability_mean, performance_mean, quality_mean;
4
5      for(i = 0; i < amount_of_processes; i++){
6          availability_total += calculateAvailability(processes[i].planned_production_time - stops(processes[i]), processes[i]);
7          performance_total += calculatePerformance(processes[i].planned_production_time - stops(processes[i]), processes[i], total_count);
8          quality_total += calculateQuality(total_count - defects(processes[i]), total_count);
9      }
10
11     availability_mean = availability_total / amount_of_processes;
12     performance_mean = performance_total / amount_of_processes;
13     quality_mean = quality_total / amount_of_processes;
14
15     OEE = calculateOEE(availability_mean, performance_mean, quality_mean);
16
17     printf("\n___________________________________\n");
```

```
    printf(ANSI_UNDERLINED_PRE "|            Total OEE            |" ANSI_UNDERLINED_POST "\n");
    printf(ANSI_UNDERLINED_PRE "| OEE             |   %12.3f   |" ANSI_UNDERLINED_POST "\n", OEE);
    printf(ANSI_UNDERLINED_PRE "| Availability    |   %12.3f   |" ANSI_UNDERLINED_POST "\n", availability_mean);
    printf(ANSI_UNDERLINED_PRE "| Performance     |   %12.3f   |" ANSI_UNDERLINED_POST "\n", performance_mean);
    printf(ANSI_UNDERLINED_PRE "| Quality         |   %12.3f   |" ANSI_UNDERLINED_POST "\n", quality_mean);
}
```

Listing 6.9: Displaying results.

6.8 Exiting the Program

The function quit, that is illustrated in Listing 6.10, can be accessed at any point of the program, except when simulating and generating data. Hence, Requirement 8 is fulfilled.

When the function quit is called the terminal is cleared and the program terminates.

```
int quit(void){
    system("clear");
    printf("The program has shut down.\n");
    exit(EXIT_SUCCESS);
}
```

Listing 6.10: Exiting the program.

7 | Evaluation of Solution

In this chapter the limitations of the different functions of the software solution will be evaluated. Subsequently, it will be discussed whether the software solution is viable.

In the data menu, when pressing the wrong keys, the key pressed will be printed in the terminal and will not be removed as in the other menus. This small detail has a great impact on the visual aspect. Therefore, it would be beneficial if a function was developed, that would be capable of copying the current screen and printing it each time a wrong input was entered. Another solution would be to operate the program in another interface than the terminal.

In the data menu, the user is able to insert values for planned production time, ideal cycle time, lambda, and so forth. However, there are not any boundaries on the values the user can enter except the restriction of the data types themselves. If boundaries would be implemented for the input, then it would ensure, that the results of the computations would be realistic.

One of the intentions with the program was, that the user could return to the previous menu at any part of the program. However, when the user has chosen the option to finish the manufacturing model, then the program does not allow the user to return to the previous menu. Thus, the only way to adjust the amount of manufacturing processes is to quit the program and execute it again. Therefore, an option that would always allow returning to the previous menu, would permit the user to model a new manufacturing system. This option would be more convenient for the user.

The manufacturing models, histograms, and tables are all designed with characters. As a result of this, the design is simple. A more visually pleasing design could have been accomplished by including a graphics library that is able to print more advanced figures.

When the user has entered all the necessary data, and the program has run the simulation and printed the results on the screen, there is no option to store the results. If the user wants to store the results, it has to be done manually. The better solution would be for the program to have the option to store the results in a text file after the results have been printed.

If it was possible to store the different input values, such that the user had the ability to create presets, then the user would not have to reenter the input values every time a simulation of the same process is desired. This is useful if the user wants to change a single piece of the data, but keep the rest unchanged.

The program is only able to create manufacturing models, which consist of independent, successive processes. Furthermore, the manufacturing systems can only be modelled based on the normal distribution and the exponential distribution. If the program was able to simulate more complexly composed manufacturing models including additional probability distributions, then it would be applicable to more types of manufacture and thus, it would be more commercially viable, since more companies would be able to use the program.

Lastly, automatic data insertion could be implemented by installing hardware on the machinery in the manufacturing processes, that would be able to collect and store the necessary data to calculate and simulate the OEE. By doing this the OEE could be monitored in real time and be stored so it can be used for simulations.

Based on the aforementioned limitations, it is evident that the functionality of the program is far from excellent. However, the program does fulfill the stated system requirements, which are based on creating a simple and functional program as well as a solution that solves the problem. Thus, the software solution is able to solve the problem, which is to simulate the OEE of manufacturing processes to test organisational changes, thus reducing waste, but only to a small degree. However, if the improvements to the limitations mentioned above were implemented in the software solution, such that the software solution would be applicable to more types of manufacture and able to model more complexly composed manufacturing models, then it would solve the problem to a higher degree. Hence, it would be more commercially viable.

8 | Conclusion

In this project the aim was to develop a software solution, which computes and simulates the OEE of manufacturing processes in order to test organisational changes, thus reducing waste.

In order to develop the software solution some system requirements were established. These system requirements were composed of functional and non-functional requirements for the purpose of ensuring that the developed solution functions as intended.

The developed software solution was successful in fulfilling the system requirements. Specifically, to meet the requirement of simulating the OEE of manufacturing processes, a probabilistic model was developed utilising Monte Carlo simulation and inverse transform sampling. By applying this model in the software solution, it was possible to simulate the OEE of manufacturing processes in order to test organisational changes.

Bibliography

[1] *Manufacture | Definition of Manufacture by Merriam-Webster*. URL: https://www.merriam-webster.com/dictionary/manufacture.

[2] The World Bank. *Manufacturing, value added (% of GDP)*. 2018. URL: https://data.worldbank.org/indicator/NV.IND.MANF.ZS?end=2019&start=2001.

[3] European Environment Agency. *Generation of waste excluding major mineral wastes, EU*. 2019. URL: https://www.eea.europa.eu/data-and-maps/daviz/generation-of-waste-excluding-major-3#tab-dashboard-01.

[4] Peter Scallan. *Process Planning: The Design/Manufacture Interface (1 Introduction to manufacturing)*. Elsevier, 2003. URL: https://booksite.elsevier.com/samplechapters/9780750651295/9780750651295.PDF.

[5] Emgrisa. Types of waste: Classification, October 2014. URL = https://www.emgrisa.es/en/publications/types-of-waste/.

[6] Rachaelle Lynn. *What is Lean Manufacturing?* Planview. URL: https://www.planview.com/resources/guide/what-is-lean-manufacturing/.

[7] EKU. *An Introduction to Lean Management*. EKU. URL: https://safetymanagement.eku.edu/blog/an-introduction-to-lean-management/.

[8] *The Seven Wastes / 7 Mudas*. leanmanufacturingtools.org. URL: http://leanmanufacturingtools.org/77/the-seven-wastes-7-mudas/.

[9] P. Arunagiri and A. Gnanavelbabu. *Identification of High Impact Lean Production Tools in Automobile Industries using Weighted Average Method*. 2014. URL: https://core.ac.uk/download/pdf/82525423.pdf.

[10] Vorne. *OEE Measures Improvements in Productivity | Lean Production*. URL: https://www.leanproduction.com/oee.html.

[11] History of oee and tpm: 50 years strong improvement | oee coach. URL: https://oee.coach/oee-academy/oee-and-continuous-improvement/history-of-oee-and-tpm/.

[12] Amper | oee tracking and machine monitoring. URL = https://www.amper.xyz/.

[13] Sistemas oee. URL = https://www.sistemasoee.com/.

[14] Oal group - food processing, robotics & automation food manufacturing solutions. URL = https://www.oalgroup.com/.

[15] Vorne industries. URL = https://www.vorne.com/.

[16] Oxford. Uk dictionary. URL = https://www.lexico.com/definition/simulation.

[17] Bharath Srinivasan. Words of advice: teaching enzyme kinetics, sep 2020. URL = https://febs.onlinelibrary.wiley.com/doi/10.1111/febs.15537.

[18] Greg Kratzig. Simulated pistol training: The future of law enforcement training?, mar 2013. URL = https://dartrange.com/wp-content/uploads/2018/11/INT-POLICE-TRAINING-JOURNAL.pdf.

[19] Benny Tjahjono Ornella Benedettini. Towards an improved tool to facilitate simulation modelling of complex manufacturing systems, 2009. URL = https://link.springer.com/article/10.1007/s00170-008-1686-z.

[20] FlexSim. *What is Factory Simulation*. FlexSim. URL: https://www.flexsim.com/factory-simulation/.

[21] Chris Kolmar. Domain knowledge: What is it and examples, sep 2020. URL = https://www.zippia.com/advice/domain-knowledge/.

[22] Functional vs non-functional requirements: The definitive guide - qra corp, 2020. URL = https://qracorp.com/functional-vs-non-functional-requirements/.

[23] Peter Olofsson and Mikael Andersson. *Probability, Statistics, and Stochastic Processes*. John Wiley Sons, Inc., 2012.

[24] Thomas C. O'Haver. Signals and noise, October 2019. URL = https://terpconnect.umd.edu/~toh/spectrum/SignalsAndNoise.html.

[25] Gordon Scott. Normal distribution, March 2020. URL = https://www.investopedia.com/terms/n/normaldistribution.asp.

[26] Kim Erland Vukovinsky and Lori B. Pfahler. The role of normal data distribution in pharmaceutical development and manufacturing, October 2014. URL = https://www.pharmtech.com/view/role-normal-data-distribution-pharmaceutical-development-and-manufacturing.

[27] Arne Thesen. Ie 642 simulation of manufacturing systems, 1999. URL = http://homepages.cae.wisc.edu/~ie642/content/Techniques/exponential.htm.

[28] Kenneth H. Rosen. *Discrete Mathematics and Its Applications*. McGraw-Hill Education, 2019.

[29] Luca Mueller, John Massaron. *Algorithms*. For Dummies, 2017.

[30] Jeri R. Hanly and Elliot B. Koffman. *Problem Solving and Program Design in C*. Pearson Higher Education Professional Group, 2015.

[31] Time complexities of all sorting algorithms, 2020. URL = https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/.

[32] Bytellect LLC. *Sorting in C with the Qsort Function*. Bytellect LLC, 2017. URL: https://www.bytellect.com/resources/sorting-in-c-with-the-qsort-function-bytellect001.pdf.

[33] Quicksort, 2020. URL = `https://www.geeksforgeeks.org/quick-sort/`.

[34] HP Langtangen. *a primer on scientific programming with python*. Springer, 2016.

[35] Masashi Sugiyama. *Introduction Statistical Machine Learning*. Elsevier - Morgan Kaufmann, 2016.

[36] Allen B. Downey. *Think Stats: Probability and Statistics for Programmers*. Green Tea Press, 2011. URL = `http://greenteapress.com/thinkstats/`.

[37] Michael Wichura. Inverse of normal cumulative density function (cdf), May 2019. URL = `https://people.sc.fsu.edu/~jburkardt/c_src/asa241/asa241.html`.

Appendices

A | The Program

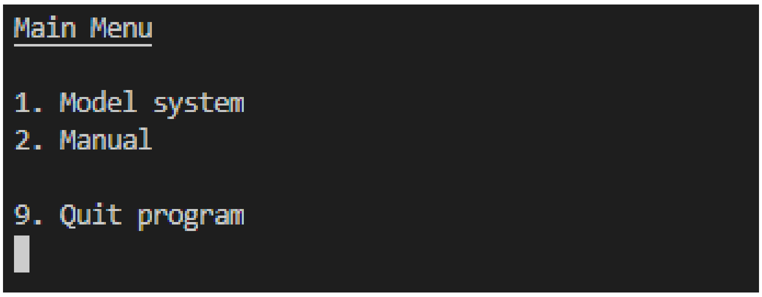

Figure A.1: Main menu.

```
Manual

The manual will give an explanation of how the program functions and how to use it. You can direct yourself back to
 the manual while using the program, if clarification is needed.

The purpose of this program is to model a manufacturing system and to run a simulation based on the modelled system
. After the simulation is done, the results will be shown.

Prior to using the program, the user needs to collect data for the following:
- Total count of products for the entire manufacturing system.
- Planned production time for each process.
- Ideal cycle time for each process.
- Whether data for defects and unplanned stops follows a normal distribution or an exponential distribution.
    - A normal distribution needs values for mean and standard deviation.
    - An exponential distribution needs a value for lambda.

The program consists of three different menus. An explanation of each menu is given below:

1. The main menu consists of three different choices:
    - If "Model System" is selected, the user is brought to the model menu.
    - If "Manual" is selected, the user is brought to the manual.
    - If "Quit program" is selected, the program quits.

2. The model menu consists of three different choices:
    - If "Model manufacturing system" is selected, the user is prompted to enter the amount of manufacturing proces
ses the manufacturing system contains.
      Followed by this, the manufacturing system will be illustrated in the terminal. The user now has the option t
o "Finish model" if he wants to proceed or "Return to model menu" if he wants to go back.
    - If "Go back" is selected, the user is brought back to the main menu.
    - If "Quit program" is selected, the program quits.

After selecting "Finish model", the user is prompted to enter the data mentioned earlier and is afterwards brought
to the last menu.

3. The simulation menu consists of three different options:
    - If "Run simulation" is selected, the simulation will run and the results are shown in the terminal.
    - If "Go back" is selected, the user is prompted to re-enter the data.
    - If "Quit" is selected, the program quits.

The results consists of:
    - Histograms that illustrate the distribution of unplanned stops and defects for each process. One "x" represen
ts 500 occurences.
    - A table that shows OEE, availability, performance, and quality for each process
    - A table that shows OEE, availability, performance, and quality for each process, sorted from lowest OEE to hi
ghest OEE.
    - A table that shows OEE, availability, performance, and quality for the entire manufacturing system.

An example of a result is shown below:
```

Figure A.2: First part of the manual.

```
Distribution of Defects for Process 1

[ 1][    28.31,    34.49] (0.026 %)
[ 2][    34.49,    40.66] (0.152 %)
[ 3][    40.66,    46.84]x (0.808 %)
[ 4][    46.84,    53.02]xxxxxxx (3.419 %)
[ 5][    53.02,    59.20]xxxxxxxxxxxxxxxxxxx (9.541 %)
[ 6][    59.20,    65.38]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (18.205 %)
[ 7][    65.38,    71.55]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (24.037 %)
[ 8][    71.55,    77.73]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (21.893 %)
[ 9][    77.73,    83.91]xxxxxxxxxxxxxxxxxxxxxxxxxxxx (13.666 %)
[10][    83.91,    90.09]xxxxxxxxxxxx (6.009 %)
[11][    90.09,    96.27]xxx (1.813 %)
[12][    96.27,   102.44]x (0.362 %)
[13][   102.44,   108.62] (0.061 %)
[14][   108.62,   114.80] (0.006 %)
[15][   114.80,   120.98] (0.000 %)


Distribution of Unplanned Stops for Process 1

[ 1][    45.23,    49.80] (0.005 %)
[ 2][    49.80,    54.38] (0.053 %)
[ 3][    54.38,    58.95]x (0.363 %)
[ 4][    58.95,    63.53]xxx (1.509 %)
[ 5][    63.53,    68.10]xxxxxxxxxx (4.826 %)
[ 6][    68.10,    72.68]xxxxxxxxxxxxxxxxxxxxxx (11.083 %)
[ 7][    72.68,    77.25]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (18.474 %)
[ 8][    77.25,    81.83]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (22.551 %)
[ 9][    81.83,    86.40]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (19.910 %)
[10][    86.40,    90.98]xxxxxxxxxxxxxxxxxxxxxxxxxx (12.693 %)
[11][    90.98,    95.55]xxxxxxxxxxx (5.906 %)
[12][    95.55,   100.13]xxxx (2.025 %)
[13][   100.13,   104.70]xx (0.503 %)
[14][   104.70,   109.28] (0.084 %)
[15][   109.28,   113.85] (0.013 %)


|                     OEE for Each Process                      |
| Process |  OEE  | Availability | Performance | Quality |
|       1 | 0.491 |        0.886 |       0.645 |   0.860 |


|           Total OEE            |
| OEE          |           0.491 |
| Availability |           0.886 |
| Performance  |           0.645 |
| Quality      |           0.860 |


1. Return to Main Menu
```

Figure A.3: Second part of the manual.

```
Model Menu

1. Model manufacturing system

8. Go back
9. Quit program
```

Figure A.4: Model menu.

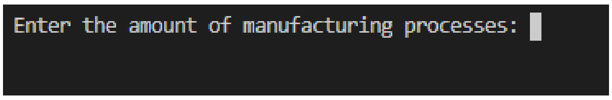

Figure A.5: Step one of modelling a manufacturing system.

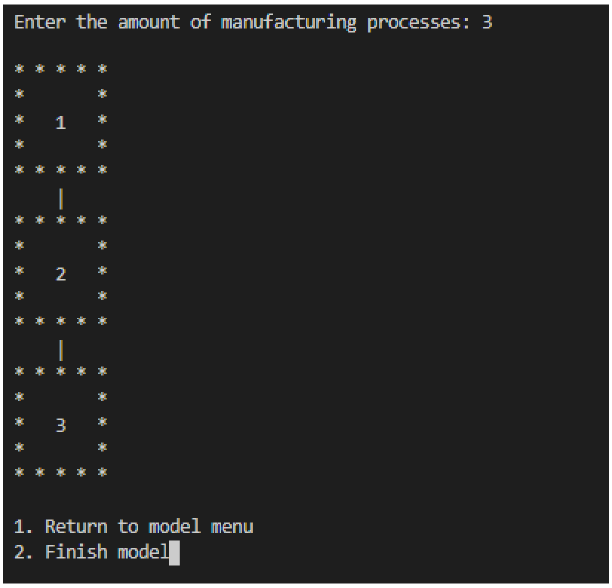

Figure A.6: Step two of modelling a manufacturing system.

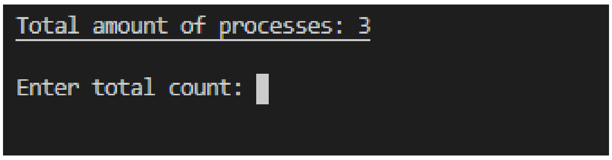

Figure A.7: Step three of modelling a manufacturing system.

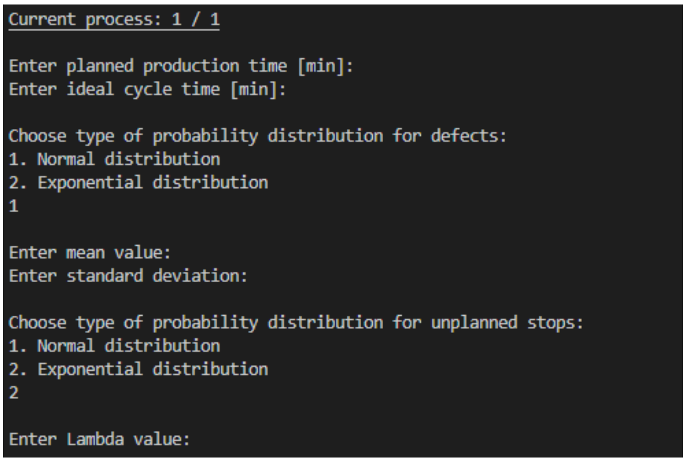

Figure A.8: Step four of modelling a manufacturing system.

```
Simulation

1. Run simulation

8. Go back
9. Quit program
```

Figure A.9: Simulation menu.

```
Distribution of Defects for Process 1

[ 1][   68.33,   79.07] (0.015 %)
[ 2][   79.07,   89.82] (0.101 %)
[ 3][   89.82,  100.56]x (0.551 %)
[ 4][  100.56,  111.30]xxxx (1.967 %)
[ 5][  111.30,  122.04]xxxxxxxxxxx (5.442 %)
[ 6][  122.04,  132.78]xxxxxxxxxxxxxxxxxxxxxx (11.233 %)
[ 7][  132.78,  143.52]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (17.658 %)
[ 8][  143.52,  154.27]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (21.231 %)
[ 9][  154.27,  165.01]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (18.991 %)
[10][  165.01,  175.75]xxxxxxxxxxxxxxxxxxxxxxxxx (12.898 %)
[11][  175.75,  186.49]xxxxxxxxxxxxx (6.508 %)
[12][  186.49,  197.23]xxxxx (2.501 %)
[13][  197.23,  207.97]xx (0.735 %)
[14][  207.97,  218.71] (0.144 %)
[15][  218.71,  229.46] (0.023 %)


Distribution of Unplanned Stops for Process 1

[ 1][    0.00,   16.39]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (55.971 %)
[ 2][   16.39,   32.79]xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx (24.520 %)
[ 3][   32.79,   49.18]xxxxxxxxxxxxxxxxxxxxx (10.799 %)
[ 4][   49.18,   65.57]xxxxxxxxxx (4.822 %)
[ 5][   65.57,   81.97]xxxx (2.237 %)
[ 6][   81.97,   98.36]xx (0.936 %)
[ 7][   98.36,  114.75]x (0.391 %)
[ 8][  114.75,  131.15] (0.174 %)
[ 9][  131.15,  147.54]x (0.091 %)
[10][  147.54,  163.93] (0.035 %)
[11][  163.93,  180.33] (0.012 %)
[12][  180.33,  196.72] (0.006 %)
[13][  196.72,  213.11] (0.003 %)
[14][  213.11,  229.51] (0.000 %)
[15][  229.51,  245.90] (0.001 %)
```

Figure A.10: Example of the histograms of results.

OEE for Each Process				
Process	OEE	Availability	Performance	Quality
1	0.583	0.867	0.769	0.875
2	0.583	0.914	0.750	0.850
3	0.280	0.980	0.571	0.500
4	0.187	0.975	0.256	0.750

OEE for Each Process (lowest to highest)				
Process	OEE	Availability	Performance	Quality
4	0.187	0.975	0.256	0.750
3	0.280	0.980	0.571	0.500
2	0.583	0.914	0.750	0.850
1	0.583	0.867	0.769	0.875

Figure A.11: Example of the tables of results.