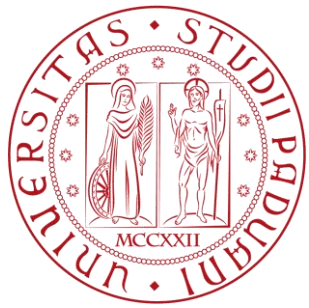


UNIVERSITÀ DEGLI STUDI DI PADOVA

DEPARTMENT OF MANAGEMENT AND ENGINEERING

Master's Degree in Management Engineering

Master Thesis

A SIMULATION MODELLING APPROACH TO IMPROVE THE OEE OF A BOTTLING LINE

Supervisor: Prof. Daria Battini
Co-supervisor: Prof. Ilenia Zennaro
Co-supervisor: Prof. Allan Larsen

Author: Pietro Zuccotto
Student ID: 1154985

ACADEMIC YEAR 2018 - 2019

To Nonno Rino and Nonna Luigina,
with all my love.

Acknowledgements

I would first like to thank my thesis advisor, prof. Daria Battini, and my supervisor during my experience at DTU, prof. Allan Larsen, for giving me the opportunity to develop this thesis. I would also like to acknowledge my co-supervisor prof. Ilenia Zennaro for her availability and her valuable support.

Many thanks go to my family: mum and dad, who have always believed in me and have encouraged me to reach my aims with courage and dedication, and my grandparents, my first supporters. I owe a very important debt to my brothers, Francesco and Lorenzo, who have proved to be close to me when it counts, even if I have stressed them with all my academic (and not) matters forever.

I would like to express my gratitude to Sofia, for being always there and for all the happiness showed towards my goals, and Forno, my right-hand man in all the decisions and thoughts, as well as friend of a thousand adventures.

The travel that brought me here has been intense and long, and it would have not been the same without the people who have accompanied me during it.
I want to thank my historic company from Isola, the reliable library's mates, and all the friends who showed their support and their affect towards me. Thanks to my friends from high-school: time goes, but the established bound remains always the same. My deepest appreciation is direct to my University's mates, from
"almost friends" they became like a second family. Besides that, their honesty and collaboration made this path more pleasant. Thanks to Daniele, Sara, Valentina, Alessandro, Giulia, Riccardo, Marco e Nicola.

A special mention is deserved by Guido, with whom I have shared difficulties, efforts but also fun and satisfaction related to this thesis project and to the whole Erasmus period. I would like to express my gratitude to him, as well as Alberto, Coco and Carlos for making the experience in Denmark unforgettable, to say the least.

Abstract

This dissertation presents a simulation approach to improve a production line affected by losses of efficiency. It starts with a theoretical overview of the topics of the project. Chapter 1 is about production effectiveness. It inspects the Total Productive Maintenance philosophy, its traditional evaluation measure, the Overall Equipment Effectiveness, and further developed measures that fit more with the case object of study. Chapter 2 deals with the job sequencing in multi-model lines, presenting either theoretical aspects on production scheduling and solving techniques. Chapters 3 and 4 concern the main concepts of simulation, its application in manufacturing and simulation modelling. A focus on the discrete-event based simulation models is carried out, just like a presentation of the simulation software AnyLogic. Chapter 5 regards a methodology for applying simulation to problem-solving, going deep into the explanation of its phases. Finally, the aspects above are gathered in the presentation of a case study about an automated bottling line whose bottleneck is studied, modelled and tested with the jointly use of a heuristic algorithm codified in Python and the simulation software AnyLogic. The purpose of the thesis project is to improve the efficiency performance of the line. Chapter 6 shows the problem and how the simulation model has been built while chapter 7 contains the experimental analysis, the tests and the economic analysis.

Contents

Introduction

Changes in the market and in the demand for goods and services that have taken place in the last decades have deeply influenced manufacturing systems. Increasing variety and differentiation due to factors, such as more customization, shorter product lifecycles and uncertainty in demand, need to go hand in hand with increased effectiveness in order to compete (Mourtzis et al., 2012). As personalization of products, mix variability, requirement of short time to market and risk of product obsolescence all increase, the need of continuous flow and JIT solutions forces industry constant improvements in terms of product quality, operation efficiency and production capacity utilization (Battini et al., 2006).

In a such a context, companies that belong to sectors characterized by high volumes and low margins, like the food and beverage, have invested in automated flow line manufacturing systems in order to guarantee a high efficiency in a mass production perspective. This means that there are several machines working in sequence, connected through various transport systems. Thus, modern companies must face a new objective at odds with cost reduction, that is the flexibility of the production plant that should manage the increasing variety of products and a rapid answer to customers' requests.

These aims can be achieved by means of a thorough control and measurement of system performances in order to find out the critical issues and improvement areas. A typical way to control companies' processes is the use of KPIs (Key Performance Indicator) that can give simple and instant insights on the performance of an activity, a process or an entire company's function. The main KPI utilized in the production context is OEE (Overall Equipment Effectiveness) (Nakajima, 1998), index able to gather much information on the performance of a system in terms of availability, performance and quality.

Implementing changes can be a difficult task for any organization, big or small. A tool that may help is simulation, considered as a key technology to support manufacturing in a fast, low cost and secure way. Simulation has gained importance in the past few years since it allows designers to imagine new systems and it enables them to both quantify and observe behaviours. Simulation can be used to study and compare alternative designs or to troubleshoot existing systems. Whit simulation models, how an existing system might perform if altered could be explored, or how a new system might behave before a modify is really applied, thus saving on costs and time (Hosseinpour et al., 2009).

The added value this dissertation wants to offer is a demonstration of simulation as a versatile, flexible and reliable tool in support to the changes that manufacturing system must face in order to remain competitive in the present ever-changing market. This thesis project carries on the application of different functions of simulation modelling applied to a case study. The case study regards an automated bottling line affected by failures, set-ups due to format changes and predictive maintenance, and product mix constraints. The purpose is to improve the OEE of the production line and maximize its throughput. The issues investigated concern the sequencing according to which the different formats of bottles are weekly produced (Format Sequencing Problem) and the buffer capacity of the bottleneck, considered not able to effectively decouple the operations of the work-stations between which it is located (Buffer Sizing Problem).

A discrete-event based simulation model is built with the software AnyLogic; hence, it is tested and validated. It is used as a benchmark to evaluate the suggested improvements regarding the optimal format sequencing respect to the starting situation. Afterwards, the length of the bottleneck's buffer is varied with a simulation experiment in order to study the impact that this modification has on the efficiency performances of the line and find out the optimal buffer size. The suggested scenario is evaluated also from an economic point of view.

Chapter 1

Effectiveness in a Flow Line

The changes of the market customization, shorter product lifecycles and uncertainty in demand highly affect the production systems characterized by high volumes and low margins. Among the sectors characterized by automated flow line manufacturing systems, it is possible to find the food and beverage. In addition to automation peculiarities, the food and beverage sector is nowadays characterized by the need for safety, quality and sustainability. These aspects co-locate and identify the company in the market. Production systems effectiveness continues to be the principal aim of each industry in order to be competitive and achieve success, but it is still deeply influenced by the previews market requests. In this context, *Total Productive Maintenance* (TPM) is a useful industrial tool to improve plant productivity and operation efficiency (Zennaro et al., 2018).

The core metric for measuring the success of TPM implementation program is the *Overall Equipment Effectiveness* (OEE) index. OEE combines three dimensions of effectiveness in one value: availability, performance rate and quality rate. OEE is the key measure to measure the performance of individual equipment. However, research studies have proved that when machines operate jointly in a manufacturing line, OEE alone is not enough to improve the performance of the entire system. Therefore, various changes and extensions to the original OEE figure have been carried out such as the Overall Line Effectiveness (OLE) and the Overall Equipment Effectiveness of a Manufacturing Line (OEEML).

1.1 Total Production Maintenance

1.1.1 Historical background

Efficiency theories have origins far in the late nineteenth century by means of the self-taught business management authority Harrington Emerson (1858-1931). According to contemporary, Emerson was inspired by the discipline evidenced in producing orchestral music, breeding horses and surveying railroad routes; he wanted to seek similar planning and control for manufacturing processes (Drury, 1918). So, when he decided to put his effort and his eclectic interests on manufacturing, he strove to determine product characteristics, costs compared to planned outcomes, and losses occurring in the use of raw materials, while planning, scheduling, and dispatching work through a large factory in order to bring efficiency. The results of his theoretical and hands-on manufacturing efforts became his "Twelve Principles of Efficiency", basis for all the further works.

Harrington Emerson, 1913 ***The Twelve Principles of Efficiency***

1 Clearly Defined Ideals
2 Common Sense
3 Competent Counsel
4 Discipline
5 The Fair Deal
6 Reliable, Immediate, and Accurate records
7 Despatching (sic)
8 Standards and Schedules
9 Standardized Conditions
10 Standardized Operations
11 Written Standard Practice Instructions
12 Efficiency Reward

Figure 1.1: *The Twelve Principles of Efficiency* by H. Emerson

These principles are reflected by present-day systems for manufacturing operations such as *Enterprise Resource Planning* (ERP) or *Quality Management Systems* (ISO 9000).

A further advancement on the effectiveness theories was given by Seiichi Nakajima, based on the experience of the practical application of maintenance best practice

in Japan between 1950 and 1970. Nakajima is the pioneer of the *Total Production Maintenance*, a methodology founded on three major concepts:

1. maximizing equipment effectiveness;
2. autonomous maintenance by operators;
3. small group activities.

TPM contributes effectively to improve the competitiveness and effectiveness of industries. In fact, it is a maintenance program which involves a newly defined concept for maintaining plants and equipment. TPM seeks to maximize equipment effectiveness throughout the life time of that equipment. It strives to maintain optimum equipment conditions in order to prevent unexpected break downs, speed losses, and quality defects arising from process activities.

1.1.2 The 5s and the eight pillars of TPM

The traditional approach to TPM was developed in the 1960s and consists of 5S as a foundation and eight supporting activities (sometimes referred to as pillars).

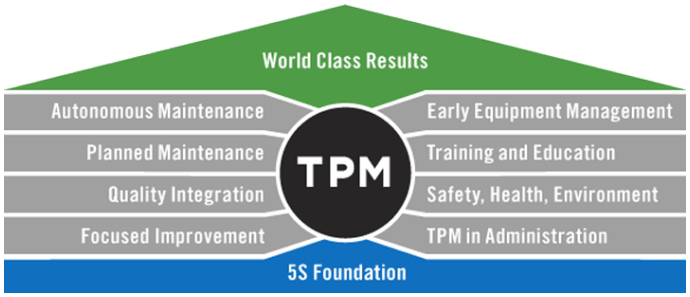


Figure 1.2: The TPM approach

The 5s Foundation

The 5*s* methodology gathers five steps into a systematic and repeatable method that aims to optimize the working standards and therefore the improvement of operative performances. It was born from the Japanese heritage which view was oriented towards the elimination of everything useless in terms of functioning for the activities (*muda*). It creates the foundation for the well-running equipment.

The term is inspired by the initials of the words that recap the five steps of the methodology:

1. Sort: eliminate anything that is not truly needed in the work area
2. Set in Order: organize the remaining items
3. Shine: clean and inspect the work area
4. Standardize: create standards for performing the above three activities
5. Sustain: ensure the standards are regularly applied.

The Eight Pillars

The eight pillars of TPM are mostly focused on proactive and preventative techniques for improving equipment reliability. They create a system for maximizing production effectiveness of any industry. The summary of eight pillars is given in the following table (Pandey, 2016).

TPM Pillar	*Description*	*Advantages*
Autonomous Maintenance	Hands operators of equipment responsible for carrying out basic maintenance of equipment	Operators feel responsible for their machines; equipment becomes more reliable
Planned Maintenance	Maintenance scheduled using the historical failure rate of equipment	Maintenance can be scheduled when production activities are few
Quality Maintenance	Quality ingrained in the equipment to reduce defects	Defect reduction & consequent profit improvement
Continuous improvement	Use of cross-functional teams for improvement activities	Improves problem-solving capabilities of the workers
Early Equipment Management	Design of new equipment using lesson learned from previous TPM activities	New equipment achieves full the potential in a shorter period
Education & Training	Bridging of the skills and knowledge gap through training of all workers	Employees gain the necessary skills to enable them to solve the problems within the organization
Health, Safety & Environment	Providing of an ideal working environment devoid of accidents and injuries	Elimination of harmful conditions & healthy workforce
TPM in the Office	Spread of the principles to administrative functions within an organization	Support functions understand the benefits of these improvements

Figure 1.3: Eight pillars of TPM

1.1.3 The Six Big Losses

Manufacturing processes are often influenced by disturbances. Such disturbances have been classified by Jonsson and Lesshammar (1999) as chronic and sporadic according to their frequency of occurrence. Chronic disturbances are usually small, hidden and complicated because they are the result of several concurrent causes. Sporadic disturbances are more obvious since they occur quickly and as large deviations from the normal state. Sporadic disturbances occur irregularly and their dramatic effects are often considered to lead to serious problems. However, research evidence suggests that it is the chronic disturbances that result in the low utilization of equipment and large costs because they occur repeatedly (Nord et al., 1997). Identification of chronic disturbances is only possible through comparison of performance with the theoretical capacity of the equipment.

Chronic and sporadic disturbances both have different negative impacts on the manufacturing process. They consume resources without adding any value to the final product. The generic losses which reduce the effectiveness of the equipment have been grouped and categorized as six big losses. In the technical literature, the six big losses are also an expression of the gap between the valuable operating time (VOT: fraction of the time in which an equipment works under optimal operating conditions) and loading time (LT: actual available time for operation, after removing all planned stops).

According to Nakajima (1998) the six big losses are:

1. Equipment failure/breakdown losses. They may be categorized as time losses when productivity is reduced, and quantity losses caused by defective products

2. Set-up/adjustment time losses result from downtime and defective products that occur when production of one item ends and the equipment is adjusted to meet the requirements of another item

3. Idling and minor stop losses occur when the production is interrupted by a temporary malfunction or when a machine is idling

4. Reduced speed losses refer to the difference between equipment design speed and actual operating speed

5. Reduced yield that occurs during the early stages of production from machine start up to stabilization

6. Quality defects and rework are losses in quality caused by malfunctioning production equipment

The first two big losses are known as downtime losses and are used to help calculate a true value for the availability of a machine. The third and fourth big losses are speed losses that determine the performance efficiency of a machine, i.e. the losses which occurs as a consequence operating at less than the optimum conditions. The final two losses are losses due to defects, the larger the number of defects the lower the quality rate of parts within the factory.

1.2 Measuring effectiveness of a flow line

The definition of metrics for measuring the productivity of manufacturing facilities has been an important field of research over the last decades. One of the first things to do in order to meet the requests of an ever-changing market is to analyse the efficiency metrics capable to assess how well equipment are exploited in comparison to their theoretical potential. Throughput, production rate and equipment utilization have been traditionally adopted as the standard way to assess the performance of manufacturing equipment, mainly because of their simplicity. Even so, these metrics lack in inclusiveness, because they measure only a part of the performances of a manufacturing system, while the effectiveness of a plant depends on the way it uses equipment, material, men and methods. For the above-mentioned reasons, a better choice to evaluate efficiency has been identified with the Overall Equipment Effectiveness (OEE). With the help of the OEE, productivity and economic benefit of a company can be well described. OEE was born as an index of performance evaluation of individual equipment in a production system, therefore in recent years various changes and extension to the original formula have been made in order to adapt it to evaluate the performance of an entire line (OLE) or of a system as a whole (OEEML).

1.2.1 Overall Equipment Effectiveness

The Overall Equipment Effectiveness was firstly proposed by Nakajima (1998) as the key metric to support TPM, and it is now a widely accepted way to monitor the actual performance of an equipment, in relation to its nominal capabilities under optimal operating conditions. OEE has many purposes, indeed it can be used as a "benchmark" to compare the initial performance of a manufacturing plant and its future values, thus quantifying the level of improvement made. Moreover, it can be used to measure the effectiveness of TPM and improve it in individual machines by reducing the concerned losses.

OEE is usually formulated as a function of a number of mutually exclusive compo-

nents (Hung et al., 2003), such as availability efficiency (A_{ef}), performance efficiency (P_{ef}) and quality efficiency (Q_{ef}). This formulation allows to break the performance of a manufacturing unit into three separate but measurable components:

$$OEE = Availability \cdot Performance \cdot Quality \tag{1.1}$$

Figure below summarizes the key elements and the fundamental influencing parameters of the OEE. Besides, it links the components with their category of loss.

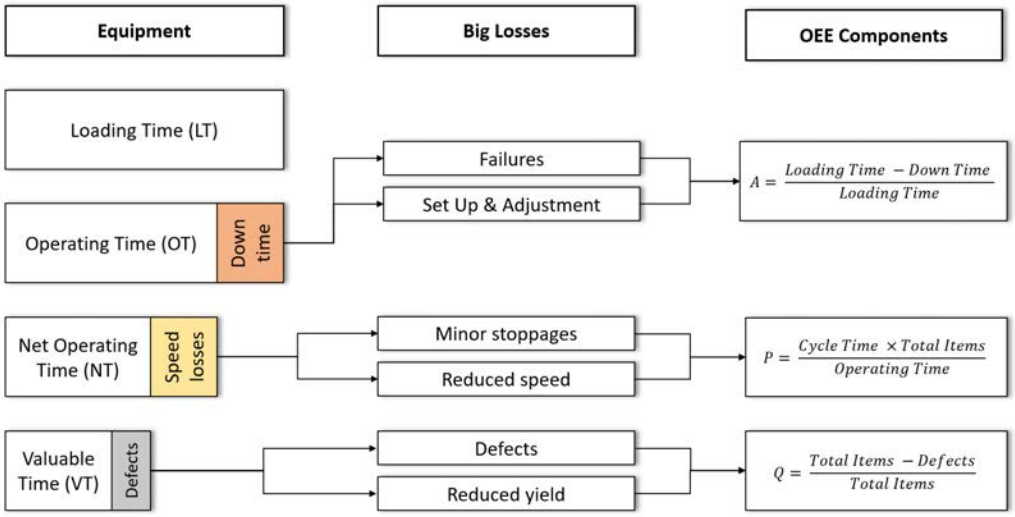


Figure 1.4: OEE and sources of loss to display the operational behaviour

As shown, the first two losses are categorized as downtime losses because they influence the true value of the availability of an equipment. In a similar manner, the third and the fourth entries are known as speed losses because they determine the performance efficiency of an equipment. The last two are known as defects losses because they are connected to defects, scraps and reworked items.

OEE can also be expressed as the ratio between what was manufactured and what could be ideally manufactured or, alternatively, as the fraction of time in which an equipment works at its full operating capacity. This concept can be formalized as follows:

$$\begin{aligned} OEE = \frac{Actual\,Output}{Reference\,Output} &= \frac{Cycle\,Time \cdot Valuable\,Operating\,Time}{Cycle\,Time \cdot Loading\,Time} \\ &= \frac{Valuable\,Operating\,Time}{Cycle\,Time} \end{aligned} \tag{1.2}$$

The three components of the OEE formulation are now explained.

Availability

The availability element of the OEE measure is concerned with the total stoppage time resulting from unscheduled downtime, process set-up and changeovers, and other unplanned stoppages. It is the ratio of actual operating time to the planned operating time and considers the theoretical production time against which unplanned downtime is highlighted.

$$Availability\,(\%) = \frac{Actual\,operating\,time\,(mins)}{Planned\,operating\,time\,(mins)} \cdot 100\% \tag{1.3}$$

where

- Planned operating time (mins) = Total shift time (mins) - Planned maintenance (mins)
- Actual operating time (mins) = Planned operating time (mins) - Unplanned maintenance (mins) - Minor stoppages (mins) - Setup changeover (mins).

Performance rate

Performance rate is the second element of the OEE calculation. It measures the ratio of the actual speed of the equipment to the ideal speed. Performance efficiency is achieved as the product of the operating speed rate and net operating rate. The operating speed rate of equipment is about the variance between the ideal speed and its actual operating speed.
The net operating rate measures the achievement of a stable processing speed over a given period.

The performance rate calculates the losses resulting from minor recorded stoppages, as well as those that go unrecorded on daily logs, such as small problems and adjustment losses.

$$Performance\,(\%) = Net\,operating\,rate \cdot Operating\,speed\,rate \cdot 100\,\% \tag{1.4}$$

where

- $Net\,operating\,rate = \frac{No.\,produced \cdot Actual\,cycle\,time}{Operation\,time}$
- $Operating\,speed\,rate = \frac{Theoretical\,cycle\,time}{Actual\,cycle\,time}$

Quality rate

Quality rate is the final element of the OEE calculation. It indicates the proportion of defective production to the total production volume.

$$Quality\,(\%) = \frac{Total\,no.\,produced - No.\,scrapped}{Total\,no.\,produced} \cdot 100\% \tag{1.5}$$

1.2.2 Limits of the traditional formula

Nowadays, OEE has been used in different industrial fields as the main efficiency metric. Even so, its application is not always straightforward; many drawbacks and difficulties have been found in several applications. The two main limits of the traditional formulation of OEE can be recapped as it follows:

1. Neither all the problems/inefficiencies a line is subject to can be classified in terms of the six big losses nor some problems can be directly tied to a specific equipment;

2. OEE measures the efficiency of a single equipment installed within a factory, whereas machines are usually not isolated, but operate jointly in a production line. Therefore, if the line is unbalanced, or if the manufacturing process is made of decoupled machines working in series or parallel, OEE alone is not enough.

In order to solve the first problem, Jeong and Phillips (2001) suggested an alternative losses classification scheme since the standard definition of OEE does not account for additional causes of losses such as preventive maintenance, off-shifts and holidays. Similarly, de Ron and Rooda (2005) noted that OEE takes into consideration losses, like blocking or starvation, that, being an effect of the whole productive system, cannot be directly attributed to a specific equipment. Thus, the authors proposed to exclude from OEE all the losses that are internal to the productive system but do not depend on the equipment itself.

A first operative approach to achieve a tentative evaluation of the efficiency of an entire line is presented by Robinson (2004). The author focuses on the aspect related to the pace of the line and what influences it. Since this element is determined by the constraining operation, both the availability and the performance rate of the bottleneck machine must be the same as that of the line. Also, quality defects upstream (US) from the constraining operation, affect the output of the line only if they result in the starvation of the bottleneck, whereas quality defects downstream (DS) from the bottleneck do affect the potential output of the line and should be counted against the quality rate. For these reasons the author suggested to evaluate the process OEE by means of equations (1.6) and (1.7), respectively:

$$Process\,OEE = A_{BN} \cdot P_{BN} \cdot Q_{Tot} \tag{1.6}$$

$$Q_{Tot} = \frac{TI_{BN} - DSD}{TI_{BN}} \tag{1.7}$$

where A_{BN} and P_{BN} are the availability and the performance rate of the bottleneck machine; TI_{BN} is the total number of items processed by the constraining operation; DSD is the total number of defects and reruns DS of the constraining operation.

1.2.3 Overall Line Effectiveness

Nachiappan and Anantharaman (2006) proposed the overall line effectiveness (OLE) as an alternative metric to evaluate the efficiency of a continuous product flow manufacturing system. As shown in equation (1.8), OLE is achieved as the product of two independent terms, namely the line availability (LA) and the line production quality performance (LPQPQ):

$$OLE = LA \cdot LPQP \tag{1.8}$$

This formula works under the hypothesis of no decouplers added between machines, so all the operations performed in a manufacturing line are strictly connected. Considering that, the operating time (OT) of the first machine will be the loading time (LT) of the second machine and, in analogy, the OT of the second machine will be the LT of the third machine and so on, proceeding downward in the line. This concept is shown in equation (1.9) where DT and PD stand for downtime and planned

downtime, respectively:

$$OT_i = (OT_{i-1} - PD_{i-1}) - DT_i \tag{1.9}$$

Thus, LA can be evaluated as the ratio of the OT of the last machine (i.e. the nth machine and the LT of the line, as stated by the following equation:

$$LA = \frac{OT_n}{LT} \tag{1.10}$$

Finally, as in the standard OEE definition, LPQP is defined as the ratio of the actual and the ideal productive rate of the line and is evaluated by applying equation (1.11)

$$LPQP = \frac{G_n \cdot CT_{BN}}{OT_l} \tag{1.11}$$

If applied to a continuous production line, OLE yields good results. In other cases, for example, when buffers or decouplers are displaced between machines, the hypothesis made to evaluate OT_i (of the generic ith equipment) do not apply. When there are buffers in the line, a DS machine can continue manufacturing even if the preceding machine is down and so, a straight application of OLE would underestimate the actual efficiency of the line. Furthermore, as shown in equation (1.8), both the terms used to calculate OLE (i.e. LA and LPQP) refer to the operating efficiency of the last machine. This is an additional disadvantage because by monitoring exclusively the last machine it is hard to identify the main criticalities and to detect the points of the line where they take place.

1.2.4 Overall Equipment Effectiveness of a Manufacturing Line

OEEML is a metric developed starting from the considerations on the limits of OLE formulation related to the hypothesis on the continuity of the flow. As already seen, when machines operate jointly in a production line, material flow, transportation, buffers and queues have a direct impact on equipment performance and vice versa. For this reason, in order to define a meaningful metric for the efficiency of the whole line, it is important to separate all the losses that can be directly ascribed to an equipment from the losses that are spread in the line (Braglia et al., 2008). These two losses type differ mainly for their dependency on the equipment; thus, as regards a manufacturing line it is efficient to distinguish between:

- Equipment dependent losses (EDL) such as defects or reduced yield;
- Equipment independent losses (EIL) such as blocking and starvation.

Any EDL can be eliminated repairing, improving or redesigning an equipment, while EIL can be eliminated acting directly on the production environment (i.e. plant layout, machine balancing, buffer sizing, etc.).

To evaluate the efficiency of a line, the early introduced additional modification to the traditional structure of losses must be considered. As noted by Jeong and Phillips (2001), the original definition of OEE is not appropriate for a production line because losses are subtracted starting from the LT, which does not include planned maintenance (PM) downtime. When PM is performed on a single machine it can reduce the availability of the line and so it must be accounted as an additional loss. Moreover, PM is intended to reduce machine failures and, if effective, it should lead toward a sensitive reduction of unplanned maintenance tasks. When time losses due to PM tasks are not detracted from the LT, such a positive balance between planned and unplanned maintenance will not be underlined by the OEE evaluation. According to the former considerations, the structure of losses shown below will be adopted as the operative framework in supporting the new formulation of the OEE.

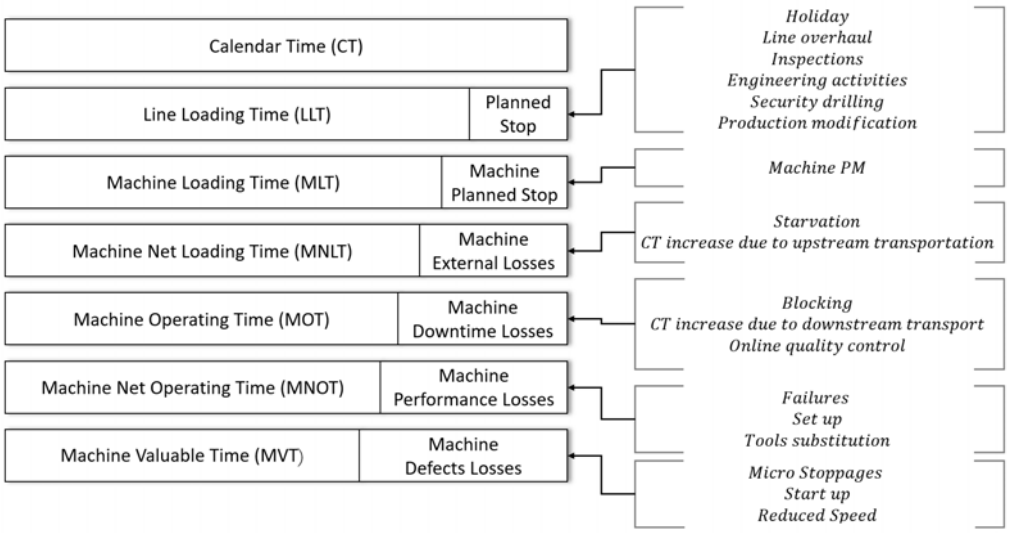


Figure 1.5: An alternative structure of losses

Referring to the alternative structure of losses shown in figure 1.5, OEE can now be modified as:

$$TOEE = \frac{MVT}{LLT} \tag{1.12}$$

Since this efficiency metric considers all the possible losses that may affect the performance of an equipment, it is referred to it as the Total Overall Equipment Effectiveness (TOEEE). Clearly, TOEE can also be expressed as the product of five independent factors as shown in equation (1.13):

$$\begin{aligned} TOEE = \frac{MLT}{LLT} \cdot \frac{MNLT}{MLT} \cdot \frac{MOT}{MNLT} \cdot \frac{MNOT}{MOT} \cdot \frac{MVT}{MNOT} \\ = A_{PM} \cdot A_{ext} \cdot OEE = A_{ext} \cdot OEEM \end{aligned} \tag{1.13}$$

where A_{PM} is the loss of availability due to predictive maintenance tasks; A_{ext} is the loss of availability due to the causes that are external to the machine (i.e. EIL); OEEM is the real machine efficiency indicator and it equals OEE times A_{PM}.

It is important to note that OEEM is a real machine efficiency indicator because it considers exclusively the events that are caused by the equipment itself and gives evidence of how much an equipment could be exploited if it was always fed and never blocked.

On the contrary, TOOE also includes the effects of the productive environment and gives evidence of the actual usage rate of a machine, which is considered as an integral part of a productive system.

Through the definition of TOEE, the evaluation of the OEEML is straightforward. Let CT_{BN} be the ideal cycle time of the bottleneck machine and O_{LM} the output released by the last machine (or operation) of the line. In accordance with equation (1.2), the following relation holds:

$$OEEML = \frac{Actual\,Output}{Reference\,Output} = \frac{O_{LM}}{LLT/CT_{BN}} \tag{1.14}$$

From the definition of machine valuable time (MVT), the output released by the last machine can also be expressed as the product of the ideal cycle time and the valuable time of the last machine of the line:

$$O_{LM} = CT_{LM} \cdot MVT_{LM} \tag{1.15}$$

Put into this form, OEEML can now be expressed as a function of the TOEE of the last machine. This is shown in the next equation:

$$OEEML = \frac{MVT_{LM}/CT_{LM}}{LLT/CT_{BN}} = \frac{CT_{BN}}{CT_{LM}} \cdot TOEE_{LM} \tag{1.16}$$

Through this formulation, the global efficiency of the plant is expressed starting from the last machine's one. However, this value regards only a certain amount of the losses linked to production; the other ones are considered by the ratio between the cycle times of the bottleneck and of the last machine.

$$\frac{CT_{BN}}{CT_{LM}} \tag{1.17}$$

In these terms, it is possible to determine a global performance index of the plant evaluating the production losses of the last machine of the line. In addition to it, the external factors that influence the performance within the system are considered by means of a correction factor.

Chapter 2

Job sequencing in a multi-model line

Sequencing and scheduling are decision-making issues that nowadays play a crucial role in the control field of the short-term period production planning. Sequencing concerns the planning of the order of the operations, or jobs, to be processed. Scheduling regards the allocation of (scarce) resources to tasks over given time periods. Their goal is to optimize one or more objectives. Usually, the optimization is about minimizing a certain time (or cost) related function. The development of these topics was driven by the increase of competitiveness and the necessity of companies to meet an always-growing demand in order to survive in the market place.

The study of the literature review on sequencing and scheduling problems in this thesis had started from the need of finding a solution to the job sequencing problem in a production system like an automated bottling line. An automated bottling line may be able to process different formats of bottles. Between the production of one format and the other, a changeover time to arrange the line for the next process may occur. For this reason, it belongs to the category of multi-model production lines. As regards the machine environment, an automated bottling line may be attributed to a flow shop, where a set of n jobs or tasks has to be processed on a set of m machines sequentially and with an identical operating order. Despite these considerations, also a study of the single-machine environment scheduling problems may be interesting in order to solve more complex configurations problems that involve sequence-dependent setup times. The objective is to find out the best sequence in order to minimize the total setup time; that is equal to minimize the makespan.

At first, an introduction to scheduling, sequencing and to their key notions to know for a better understanding of the matter is given. The second part starts with a literature review on scheduling problems. It is followed by a presentation of the computational complexity theory that characterizes and it is used to classify optimization problems. Then methods to solve this kind of problems, root of scheduling problems, are introduced. As regards scheduling problems, the single-machine environment is the heart of the chapter since it is the black box for solving more complex systems and a good link to the automated bottling line case when cycle times of the machine are equal. Therefore, some techniques to solve scheduling problems on a single-machine are presented. Then the flow shop is also introduced. The focus on both is towards the sequence-dependent setup times scheduling problems. Finally, it is presented the solving method developed throughout the project in order to solve the case study.

2.1 Production scheduling

Production concerns processes and methods used to transform tangible inputs (raw materials, semi-finished goods, sub-assemblies) and intangible inputs (ideas, information, knowledge) into goods or services. Resources are used in this process to create an output that is suitable for use or has exchange value. By a management point of view, critical issues but also cornerstones in production are production planning and production control.

- Production planning deals with basic concepts of what to produce, when to produce, how to produce, etc. Basically, it involves taking a long-term view.
- Production control looks to utilize different type of control techniques to achieve optimum performance out of the production system in a short-term productive period.

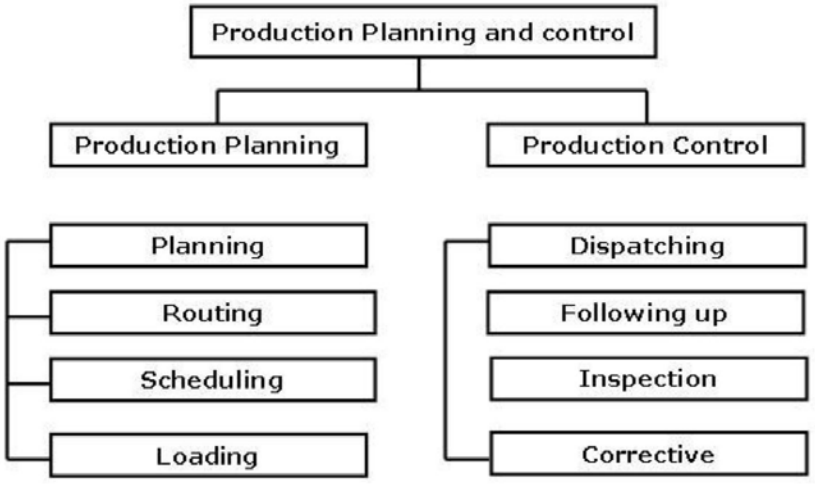


Figure 2.1: Production planning and control

One of the most important functions in the production control system is scheduling. Every production system should have a kind of production scheduling, no matter whether it is managed and organized traditionally or have a systematic and scientific approach to the planning. Given that, it is commonly known that an efficient scheduling plan guarantees a better usage of the resources, especially machines and

manpower, in order to achieve better production performances. Scheduling can be theoretically defined as a decision-making process, used on a regular basis in manufacturing and services industries, that deals with the allocation of resources (often simply called machines) to tasks (jobs) over given time periods. Its goal is to optimize one or more objectives (Pinedo, 2008).

Sequencing, scheduling and schedule are terms often used in this topic, but they blend in different meanings. A clarification is mandatory to move on with the discussion. *Sequencing* problem determines an appropriate order for the jobs to be processed within, e.g., the shortest possible time called makespan, used by Bard et al. (1992), Bolat (1994) and Lahmar et al. (2003). *Scheduling* is the process of arranging, controlling and optimizing work and workloads in a production process or manufacturing process. It is used to allocate plant and machinery resources, plan human resources, plan production processes, purchase materials or start and finish time for each order. Therefore, scheduling can bring productivity in a shop floor by providing a calendar for processing a set of jobs. It is nothing but scheduling various jobs on a set of resources (machines) such that certain performance measures are optimized. Since the sequencing problem also results in a schedule for the jobs on the stations, many authors use the term scheduling instead of sequencing. The work of Beaty (1992) highlights that the two problems are either intimately tied together or irrelevant to each other and many times are used interchangeably.
The work done in the scheduling phase ends up in a production *schedule*. Efficient production schedules can result in substantial improvements in productivity and cost reductions. Generating a feasible schedule that best meets management's objectives is a difficult task that manufacturing firms face every day (Ozgur et al., 1995).

The notation $\alpha|\beta|\gamma$

Scheduling problems are characterized by a proper three-fields notation $\alpha|\beta|\gamma$, where α stands for the system and the number of machines, β for the potential specific characteristics as set-ups, preemption, etc., and γ for the objective function. As regards the system, the possible structures are single-machine, parallel machines, flow shop and job shop, plus their derived structures; they will be denoted with

1, P, F and J respectively. The three fields of characterization of a scheduling problem that are machine environments, characteristics of the jobs and optimization objectives, will be presented in more detail later in this chapter.

2.1.1 Importance of scheduling in a manufacturing system

The current environment in manufacturing companies is characterized by massive competition faced by market and customers' requirement and expectations. The requests are rapidly increasing in terms of quality, cost and delivery time. Generally, the firm performance is built in two dimensions:

- Technological dimension
- Organizational dimension

The purpose of the technological dimension is to develop the inherent performance of marketed products in order to satisfy the requirement of quality and lower cost of the product. In this regard, it must be noted that the rapid technological growth for these products forced the companies to opt for mass production. This needs a flexible and progressive production system capable of adapting to market demand and needs quickly and efficiently.

An organizational dimension intends to performance improvements in terms of production cycle times, expected delivery date, inventory and work in process management, etc. Therefore, companies must have a powerful method and tools at their disposal for production planning and control. To achieve these goals, an organization normally implements several functions including scheduling with variety of products, processes and production levels, production planning, material and capacity planning, etc., for better coordination to increase productivity and minimize operation costs.

A production schedule detects the control over the release of jobs to the shops, ensure required raw materials are ordered in time and find strategies for resource conflicts. A production schedule can determine whether delivery promises can be met and identify the time period available for preventive maintenance. In a manufacturing

environment, all jobs or tasks are associated with a due date. These jobs must be processed on the machines in a given order or sequence. All these mentioned aspects build scheduling problems.

2.1.2 Characteristics of a scheduling problem

As seen in 2.1, term scheduling refers to a wide class of problem, often different in structure and complexity. From that issue, a proper notation to differentiate them is born.

β - Characteristics of a job

The scheduling problem is about the assignment of a resource to an activity to be done. Resources and activities are indicated as machines and tasks. Job is used to mean a collection of tasks technologically connected (e.g., three tasks necessary to produce the same object form a job). From now on, the letter m will represent the number of machines and n the number of jobs. Several information can be linked to a job:

- *Processing time* p_{ij}: time requested by the job j to the machine i to complete it.
- *Release date* r_j: timing (respect an initial time 0) before which it is not possible to start the job j.
- *Due date* d_j: timing (respect an initial time 0) within which the job j shall be finished. Usually, if the dates are not respected some costs occur like penalties, losses of trust from a client, etc.
- *Weight* w_j: relative importance of the job j respect the other jobs.

The aim of a scheduling problem is to find the best time utilization of the machine by the jobs that need to be done. This solution is called *schedule*. Instead, the *sequence* specifies only the order that the jobs must follow during the process, the schedule specifies also the starting time. Given a schedule *S*, *S(j)* will be the starting

time of a job j. It has to be stated that only the admissible schedules are interesting in being taken into consideration. They must respect all the implicit constraints of a scheduling problem. For example, the same job cannot be executed simultaneously by two machines, the same machine cannot work two jobs simultaneously, a job cannot be stopped (except in a case of a preemptive problem), that certain priorities must be respected. These specifications can be made clear when defining the scheduling problem. They may be:

- *Set-up time* s_{jk}: time necessary to reconfigure the machine that has worked the job j to make it work the following job k.
- *Preemption*: act of stopping a job to let the execution of a more urgent one be done.
- *Priority constrains*: they state that a certain job j is not allowed to start before a certain job k. Or conversely.
- *Blocking e no-wait*. If the buffer of the machine i is full, a job just finished on the machine i - *1* cannot be placed on the buffer of machine i. This situation causes a block of the machine i - *1*. In a *no-wait* situation, a job is not even allowed to wait on a machine. In a better case, it should be guaranteed that at a certain point a task is completed on a machine, the next machine might be free to process the job.

Purposes of a scheduling problem can be various and different. To state them in a formal way, it is also required to introduce some functions connected to the jobs in an admissible schedule.

- *Completing time* C_j: time at which the last task of the job j ends.
- *Lateness* L_j: difference between the completing time and the due date of the job j. A positive value states a delay, a negative one states an advance respect to the due date. The formulation is: $L_j = C_j$ - d_j.
- *Tardiness* T_j: it is equal to the lateness when the latter is positive, otherwise it is zero. That is, $T_j = \max(0, C_j - d_j)$.

α - Machine environments

In addition to the characteristics related to the jobs, there are the ones connected to the system. There is indeed a huge variety of service or production system architectures; this dissertation will introduce just many of them and it will focus mainly on the ones that depict the situation under analysis: an automated bottling line. With a *single machine*, all the jobs require the same resource to be done. In this case, every job consists of a single task. It is different the situation when m machines are in *parallel*. In the case of a *job shop* there are m machines but the jobs do not have an order to follow in the sequence of the process.
Finally, there is the *flow shop*. The system is made by m machines (work-stations) sequentially ordered. Every job must be executed by every machine progressively. That is, a job has to visit before the first machine, then the second machine, ..., and so on until the last machine m. It is often assumed that every machine has a FIFO type buffer that implies that the order that the jobs follow to be worked is the same for all the machines. The jobs cannot be overtaken. This configuration is named *permutation job shop*. A manufacturing automated flow line can be marked out as a permanent flow shop.

γ - Objective functions

It is now presented a list of some objective functions to be solved in a scheduling problem.

- *Maximum completing time* or *make-span* C_{max}. The makespan is equivalent to the completion time of the last job to leave the system. A minimum makespan usually implies a good utilization of the machine(s).
- *Maximum lateness* L_{max}. It measures the worst violation of the due dates.
- *Maximum tardiness* T_{max}. It is defined as $max(0, L_{max})$.
- *Weighted sum of completing times*. It gives an indication of the total holding or inventory cost incurred by the schedule.

2.1.3 The study of a scheduling problem

One of the first classification schemes for scheduling problems appeared in Conway, Maxwell and Miller (1967). Lawler, Lenstra and Rinnooy (1982), in their survey paper, modified and refined this scheme extensively. Herrmann, Lee and Snowdon (1993) made another round of extensions. For a survey of scheduling problems subject to availability constraints, see Lee (2004). For surveys on scheduling problems with non-regular objective functions, see Raghavachari (1988) and Baker and Scudder (1990). For a survey of scheduling problems with batch processing, see Potts and Kovalyov (2000). The complexity hierarchy of scheduling problems is motivated primarily by the works of Rinnooy Kan (1976), Lenstra (1977), Lageweg, Lawler, Lenstra and Rinnooy Kan (1981, 1982) and Lawler, Lenstra, Rinnooy Kan and Shmoys (1993).

The example cases from the literature that are going to be presented in this chapter are the one more relevant in the study of an automated flow line. A flow line may be properly the case of a flow shop, with m machines and series and an equal flow to be followed by all the n jobs that enter the system. Besides this, a rooted look at the literature has brought to infer that also a single machine environment, with proper assumptions, can be studied to schedule the products of a line with $m > 1$ machines. The single machine environment is very simple and a special case of all other environments. The passage can be easily legitimized mentioning the example of a given production line with $m > 1$ machines whit equal cycle times. The study of the case associating it to a flow shop may be either worthless and misleading because an optimal sequencing of the n jobs (or products) may be achieved just focusing on the first – single - machine of the line. Furthermore, it has not been found a proper algorithm for the flow shop scheduling problem with m machine with equal processing times $p_j = p$, since the resolution methods for flow shop, like the Johnson algorithm, always consider machines with different cycle times.

When an algorithm for one scheduling problem can be applied to another case there is a so-called *reduction* in the field. For example, $1||\sum C_i$ is a special case of $1||\sum w_i C_i$ and a procedure for the latter scheduling problem can, of course, also be

used for $1||\sum C_i$. In complexity terminology it is then said that $1||\sum C_i$ reduces to $1||\sum w_i C_i$. This is usually denoted by:

$$1||\sum C_i \propto 1||\sum w_i C_i$$

2.1.4 Scheduling problem in multi-model lines

In many industries, also belonging to the Food & Beverage sector, the choice of utilizing common resources to manufacture multiple products implies changeover and setup activities, representing costly disruptions to production processes. Therefore, setup reduction is an important feature of the continuous improvement program of any manufacturing, and even service, organization in general.
Setup time can be described as the time necessary to arrange the necessary resource (e.g., machines, people) to perform a task (e.g., job, operations). The figure below depicts a multi-model line with setup times between the processing of one job and the following one.

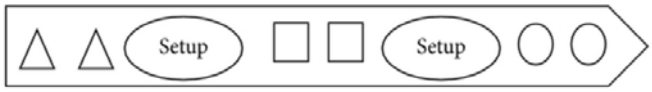


Figure 2.2: Multi-model production line

Setup time can be of two types: sequence-independent and sequence-dependent. If setup time depends exclusively on the task to be processed next, regardless of its preceding task, it is called sequence-independent. On the other side, setup time depends either on the task and its preceding task; it is called sequence-dependent setup time. They have been classified along this dimension by Allahverdi et al. (2008).
Scheduling problems with sequence dependent setup times can be found in various processing environments such as production, service and information processing systems. They can be shortened with SDST scheduling problems. One famous case of SDST problem regards a printing industry. Setup time is required to clean the machine and prepare it to print in a new colour, therefore it depends on the colour

of the current and immediately following jobs.
In a bottle industry, setup time may rely on the sizes and the shapes of the bottle. Similar situations arise in chemical, pharmaceutical, food processing, metal processing, paper industries, and many other areas.

The benefits of reducing setup times include:

- reduced expenses;
- increased production speed and output;
- reduced lead times;
- faster changeovers;
- increased competitiveness, personal and customer satisfaction;
- increased profitability;
- enabling lean manufacturing;
- broader range of lot sizes;
- lower inventory, total cost curve, minimum order sized;
- faster deliveries.

The importance and benefits of incorporating setup times in scheduling research have been investigated by many researches. See for instance Flynn (1987), Kogan and Levner (1998), Krajewski et al. (1987), Liu and Chang (2000), Trovinger and Bohn (2005).

2.2 Computational complexity theory

The objective functions presented in 2.1.2 are defined in relation to a key issue for a manufacturing firm: the maximization of productivity and therefore the minimization of completion times, the reduction of production costs and a better efficiency in materials management. During the years, many methodologies have been developed to reach the aimed solution. They can be implemented through algorithms codified in specific software. Before proceeding with an overview of these methods, it is interesting to present an important aspect related to optimization problems that is the complexity theory behind them. The aim of this theory is to determine whether a given kind of optimization problem is easy or not. This explanation helps to understand why an approach might be chosen rather than another. The elements used to evaluate the problem and to state its difficulty are:

- the computational time needed to solve the algorithm that identifies the problem;
- the amount of computational memory needed to find a solution.

When the computational time required can be described with a np function the algorithm is noted as polynomial. The algorithm is exponential if the computational time can be described through a $2n$ function. In this notation n is associated to the dimension of the input data and p is a certain constant value.
An algorithm is considered efficient when it is of polynomial type, given that polynomial type problems are faster than exponential ones.

Complexity classes

Introduced the basic concepts of the Computational complexity theory, it is possible to present the categories to which a given problem can be associated to.

- P class: problems whose solution can be reached in polynomial time.
- NP class: problems for which the precision of their solution can be verified in a polynomial time.

- coNP class: problems for which the inaccuracy of their solution can be verified in a polynomial time.
- NP-complete: kind of problems such that an NP problem can be transformed into an NP-complete problem in a polynomial time.
- NP-hard: problems that are at least as complicated as the most difficult NP problem, but they could be even more difficult than it.

It is necessary to precise that for a problem it is enough to verify the (non) accuracy of a single solution for that to enter into the classes NP or coNP, it is not mandatory to verify all the possible solutions. Moreover, certain problems present the characteristic to make verifiable either the accuracy and the non-accuracy of a solution in a polynomial time. This characteristic is associated with all the problems belonging to the class P, that is as a consequence a subset of NP. The figure below depicts the links between the different complexity classes of the problems.

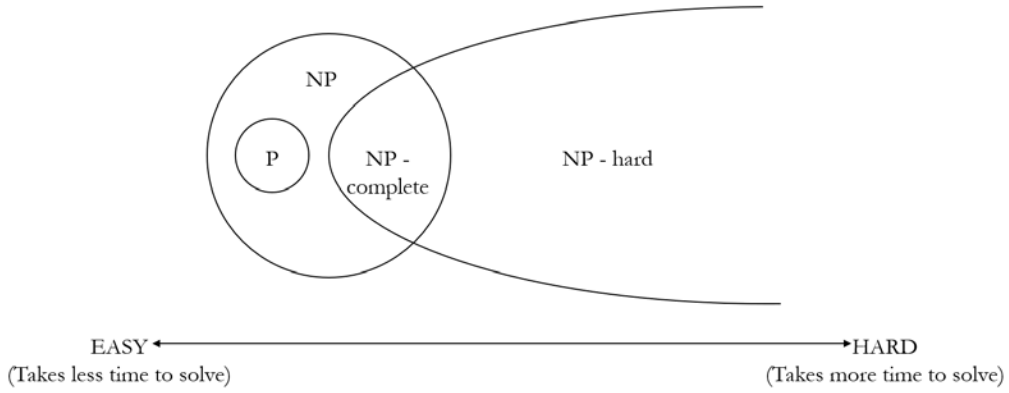


Figure 2.3: Representation of complexity classes

2.3 Solving techniques

After a presentation of the evolution that has regarded the short-period planning techniques, a brief overview of general purpose procedures is given. They are useful in dealing with scheduling problems in practice and they can be implemented with relative ease in industrial scheduling systems. Different kinds of techniques have been developed since scheduling problems had appeared. Like all the optimization problems, a scheduling problem can be solved with exact methods or heuristics methods based on its complexity. The two categories are different as regards the computational time needed to solve the problem and the goodness of the solution. Heuristics do not guarantee an optimal solution; they instead aim at finding reasonably good solutions in a relatively short time. Heuristics can be further categorized into two types: constructive and improvement.

2.3.1 Evolution of short-term planning methods

Production planning techniques that regard the short-term have started to be developed since the beginning of the seventies. In literature is present a suggested classification based on four eras for describing this evolutionary process (Caridi and Sianesi, 1999).

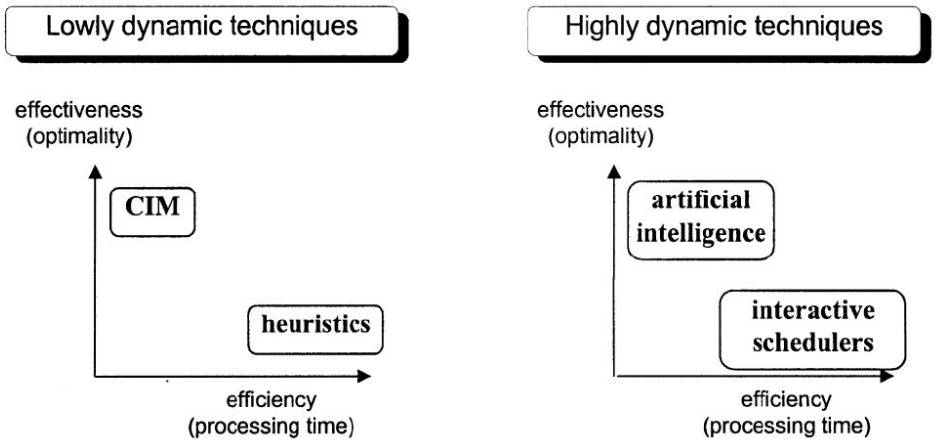


Figure 2.4: Evolutionary eras of production planning techniques

Optimization era

The so-called period regards the approaches aimed to optimize the productivity born between 1970 and 1980. They were characterized by Computer Integrated Manufacturing (CIM) based on a completely automated system. Nevertheless, this technique shows some problems that undermined its success:

1. It is required a long time to reach a good level of total automatizing, that is at odds with the need of the increasing speed of answer to the market in order to remain competitive.

2. A higher level of automating calls a higher standardization of the product. It is opposed to the desire to differentiate the production of many firms.

3. It is not that simple to translate decision processes and rules of a complex real system into analytic models.

As soon as these limitations have been completely understood, the pure optimization approach has started to be seen as something a bit utopian. Therefore, researchers started to develop techniques diametrically opposed to it.

Heuristics

Heuristic era started to develop in the eighties. It is born from the need to overcome the difficulties appeared with optimization methods when modelling reality. A better tool in decision-making processes was required. Basically, a heuristic model is the framework of the mental passages made by the planner during its decision process. The advantages of heuristic methods are:

- the logic model is closer to the physical one;

- the model is based on the experience of a single individual. This often implies that may be reached different objectives simultaneously.

The main lack of this approach is it to be static. That is, the rules and priorities of the firm might differ over time, but they are decided a priori when the scheduling system is designed. This limits the ability of the model to adapt rapidly towards the changes in the market.

Virtual Manufacturing Era

Dynamic systems started to arise the second half of the eighties with the era of Artificial Intelligence, denoted also as Virtual Manufacturing Era, until now. This epoch results to be the best answer in order to interpret the complex modern production systems. Different techniques were born during this period: Expert Systems, Neural networks, Genetic algorithms, Ant Colony Optimization, and so on. Some of these methods will be briefly presented later.

Interactive schedulers

Interactive schedulers were born at the beginning of 1990. The attempt was to exceed the problems that had emerged from the optimization approaches and heuristics. The new methods are planning systems that are easier to manage, where it is the scheduler that takes all the decision. The machine, with the implemented software, verifies the feasibility of what proposed by the scheduler or suggests a new general plan to use as a reference point. For that, this period is named *Interactive schedulers era*.

2.3.2 Exact optimization methods

Exact methods guarantee, at least theoretically speaking, to solve a Combinatorial Optimization (CO) problem in an exact way. That is, to find an admissible solution corresponding with the optimum of the objective function between all the admissible solutions. The application of exact methods is not always possible, mostly for two reasons:

- The complexity of the problem may cause a lot of computational time to solve the problem (e.g., NP-hard problems).
- Available time to solve the problem.

In spite of this, exact methods may be used to solve scheduling problems when the situation allows it. Therefore, here is given an overview of two exact optimization techniques such as branch-and-bound and dynamic programming.

Branch-and-Bound

Branch & Bound is a general-purpose technique to solve combinatorial optimization problems. It was at first proposed by Land and Doig in 1960 to solve integer linear programming problems.
It is based on the decomposition of the original problem in sub-problems that are easier to solve. Because of their method of trying all the possible solution until they found the optimal (or correct) one, Branch & Bound algorithms may also be classified as implicit enumeration algorithms.

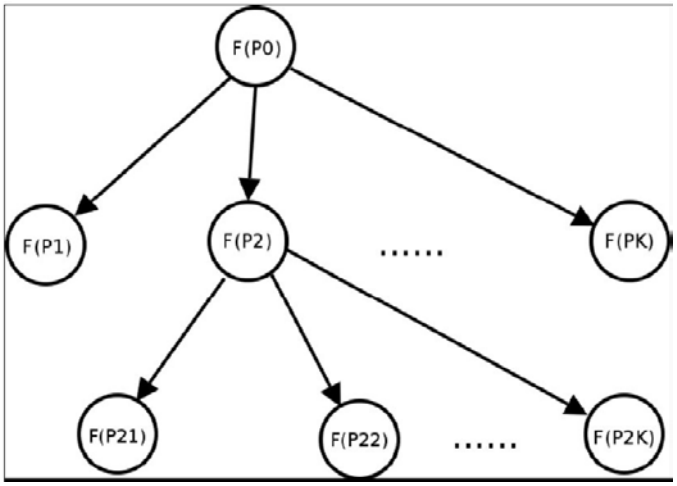


Figure 2.5: Example of "tree" generated from a B&B analysis

The figure above represents the branching process. It can be depicted as a branch decision tree, where every knot represents the sub-problem whereas every arch is a descendant relation.

Dynamic Programming

The Dynamic Programming method works on sub-problems like division method and division-based methods. It has been widely used in optimization problems since Richard Bellman developed it in the 1950s. A basic condition for using the method to calculate the optimal case is known as *optimality principle*. Optimality principle is to solve the problem optimally including optimal solution of all the sub-problems. That is, the problem should be such that when finding its optimal solution, optimal

solutions of all its sub-problems is also obtained. The number of sub-problems in this method is $2n^n$ and each sub-problem can be solved in a time in linear order.

2.3.3 Heuristics methods

When a scheduling problem and/or the background of the solution does not allow to apply exact solving techniques, it is necessary to reach "good" admissible solutions in "reasonable" computational time. Heuristic methods come from that need. These methods took their name from the Greek word *euriskein*, that means - to find-. In fact, the reason why these methods have spread in many fields of application is that they allow finding reasonably good solutions in a relatively short time; rather than an optimal solution but in a big amount of time. They tend to be fairly generic and can be adapted easily to a large variety of scheduling problems.

The literature on heuristics is wide, and it is supposed to become wider for the ease of adaptability of this method. Many and different techniques have been developed to the point to make really tough every attempt in giving an acceptable classification. A possible list of categories of heuristics methods sees:

1. Constructive heuristics: they start without a schedule and gradually construct a schedule by adding a job at a time. *Dispatching* or *priority rules* fit in this category.

2. Meta-heuristics: generic methods that define components and their interaction in order to reach a good solution. Meta-heuristics can be classified also as algorithms of the improvement type. They differ from the previous one because they start out with a complete schedule, which may be selected at random, and then try to obtain a better schedule by manipulating the current one.

3. Approximate algorithms: they offer, for each instance of the problem, a solution that might not be worse than the optimal one, within a given percentage.

4. Iper-heuristics: pioneering field at the edge between artificial intelligence and machine learning. The aim of the research is to define algorithms that are

able to find themselves certain optimization methods and automatically adapt them to different problems.

The first two methods are better presented in the next section.
As regards meta-heuristic, there will be a presentation of local search procedures such as Simulated Annealing (SA), Tabu-search (TS), Hill Climbing (HC), Genetic Algorithm (GA). In addition to them, a framework that combines local search techniques, dispatching rules and other techniques is also showed: the Ant Colony Optimization (ACO) algorithm.

Constructive heuristics

Constructive heuristics determine an admissible solution by starting only from the input data of the given problem. A common feature is the lack of backtracking: it is started from an empty solution and through an iterative way, new elements are added to a solution until the solution is completed. This is called expansion criteria. The computational complexity of the techniques of this type is polynomial. The expansion criteria for each iteration is based on the choice of the element that produces a better improvement of the objective function, according to the constraints of the problem. For this reason, some constructive heuristics are also named *greedy*.

Dispatching rules belong to the set of greedy algorithms since they take advantage of the order of the element in the sequence. During the initialization phase, operations are ordered based on rules or priority index, and they are assigned to the available machines following this order. Dispatching rules are useful in scheduling problems when one attempts to find a reasonably good schedule with regard to a single objective such as the makespan, the total completion time or the maximum lateness.
The most used dispatching rules are:

- First Come First Served - FCFS
- Weighted Shortest Processing Time - WSPT
- Longest Processing Time - LPT

- Earliest Due Date - EDD
- Minimum Slack Time - MST

Examples of their application are given when studying the single-machine case in the next section 2.4.

2.3.4 Meta-heuristic methods

The first type of meta-heuristics investigated regards a crucial class of improvement type algorithms: the local search procedures. A local search procedure does not assure an optimal solution. It usually attempts to find a schedule that is better than the current one in the *neighbourhood* of the current one. Two schedules are neighbours if one can be obtained through a well-defined modification of the other. At each iteration, a local search procedure performs a search within the neighbourhood and evaluates the various neighbouring solutions. The procedure either accepts or rejects a candidate solution as the next schedule to move to, based on a given acceptance-rejection criteria.

Local search algorithms may be further divided into punctual and population-based algorithms. *Punctual* algorithms, such as Simulated Annealing, Tabu-Search and Hill Climbing, build a punctual trajectory in the solutions space, considering one solution at each iteration.
There are instead meta-heuristics that keep a population of solutions; the so-called *population based* algorithms. That is, at each iteration they combine the set of solutions in order to achieve a new population. These methods are often inspired by natural mechanisms since in nature there are situations where the subjects tend to arrange themselves in "optimized" structures. During the last years, many studies concerning this topic have been carried on; also, with strong links between different disciplines such as Operative Research, Artificial Intelligence, Soft Computing, etc. These researches led to the definition of various optimization methods. The ones that will be further presented are the Genetic Algorithm and Ant Colony Optimization. Other population-based algorithms are Evolutionary Computation, Scatter Search, Swarm Optimization, etc.

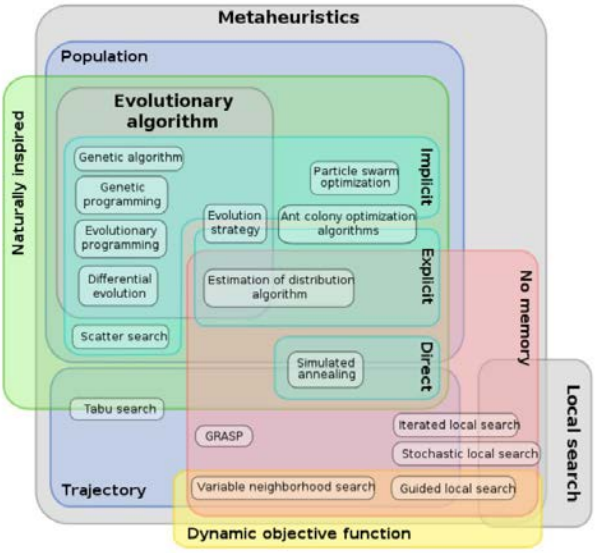


Figure 2.6: Classification of meta-heuristic algorithms by J. Dreo (2007)

The ones just introduced are only a side of a possible list or classification of meta-heuristic algorithms. For a deeper investigation on meta-heuristic see J. Dreo (2007), Naepolitan and Naimipour (2004), Voss et al. (1999). Beheshti et al. (2013) reviewed population-based algorithms.

Simulated Annealing

Simulated Annealing is a computational procedure developed in 1983 by Kirckpatrick in order to solve combinatorial optimization problems. The name recalls the metallurgy treatment that establishes to heat a metal at high temperatures and then cool it down slowly, to let the crystal lattice modify its asset in order to achieve defined characteristics. In the same way, SA process states that each solution is to be associated with a certain T temperature. Decreasing the temperature following a cooling rate, is it possible to find more solutions in the neighbour of the actual one. The process is shown in the flow diagram below (Mc Mullen, 1998).

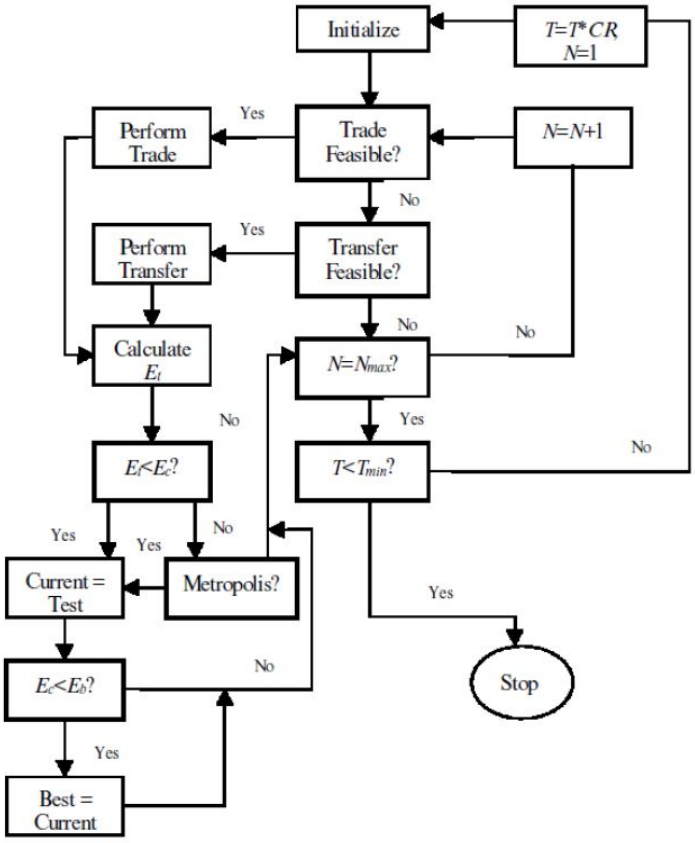


Figure 2.7: Flow chart for the Simulated Annealing algorithm

Tabu-Search

Building upon some of his previous work, Fred Glover proposed in 1986 a new approach, which he called Tabu Search, to allow local search methods to overcome local optima. The basic principle of TS is to pursue local search whenever it encounters a local optimum by allowing non-improving moves; cycling back to previously visited solutions is prevented by the use of memories, called *tabu lists*, that record the recent history of the search, a key idea that can be linked to Artificial Intelligence concepts. The flowchart below illustrates the decision moments of Tabu Search (Mc Mullen, 1998).

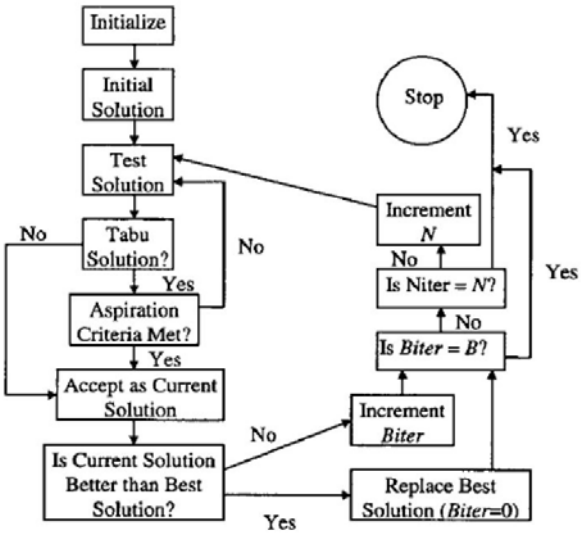


Figure 2.8: Flow chart for the Tabu-search algorithm

Hill Climbing algorithm

Hill Climbing algorithm is one of the simplest local search algorithms. It takes a random point in the search space and it states this as the initial solution for which calculating the objective function. In the next step, the neighbours of the initial solutions are investigated. If a neighbour with better value of the objective function exists, the algorithm changes the location to that point. Otherwise, if there is no better neighbour, the current location is selected as the optimal solution. High climbing may belong to greedy algorithms since it selects a good neighbour without

thinking where to go. At any point in state space, the search moves in the only direction that optimizes the cost of function with the hope of finding the optimal solution at the end. This aspect makes the algorithms fast in improving bad states, but it can give results different from real ones and the optimal one. The Hill Climbing algorithm may be associated with a state space diagram. State space diagram is a graphical representation of the set of states the search algorithm can reach (*X-axis*) related to the values of the objective function (*Y-axis*). The best solution will be that state space where objective function has a maximum value(global maximum).

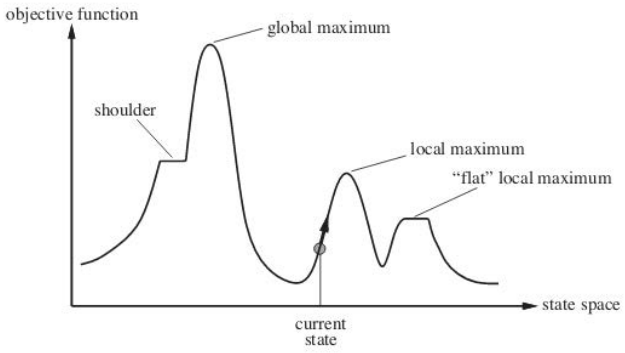


Figure 2.9: State space diagram for HC algorithm

The state space diagram may present different regions:

- Local maximum : It is a state which is better than its neighbouring state however there exists a state which is better than it(global maximum). This state is better because here the value of the objective function is higher than its neighbours.
- Global maximum: It is the best possible state in the state space diagram. This because at this state, objective function has the highest value.
- Flat local maximum: It is a flat region of state space where neighbouring states have the same value.
- Ridge: It is a region which is higher than its neighbours but itself has a slope. It is a special kind of local maximum.
- Current state: The region of state space diagram where we are currently present during the search.

- Shoulder: It is a plateau that has an uphill edge.

Genetic Algorithm

Basic principles of Genetic Algorithms were set up by Holland in 1975, inspired by Darwin's theory of evolution, published in 1859. Genetic algorithms, when applied to scheduling, view sequences and scheduling as individuals or members of a *population*. Each individual is characterized by its *fitness*. The fitness of an individual is measured by the associated value of the objective function. The procedure works iteratively, and each iteration is referred to as a *generation*. The population of one generation consists of survivors from the previous generation plus the new schedule, the *offspring* of the previous generation. The offspring is generated through reproduction and mutation of individuals that were part of the previous generation, the *parents*. Individuals are sometimes also referred to as *chromosomes*. In each generation, the fittest individuals (the best solutions) reproduce while the least fit die.

Ant Colony Optimization

Ant Colony Optimization (ATO) algorithm was firstly proposed by Dorigo et al. in 1996 as a tool for solving the Travelling Salesman Problem. It derives from principles of another type of heuristic techniques and it is inspired by the trail following behaviour of ant colonies. Ants, when moving along a path to a destination, leave along with their path a chemical called pheromone as a signal for other ants to follow (left figure in 2.10). An ACO algorithm assumes that a colony of (artificial) ants iteratively construct solutions for the problem at hand using (artificial) pheromone trials that are related to previously found solutions as well as to heuristic information.

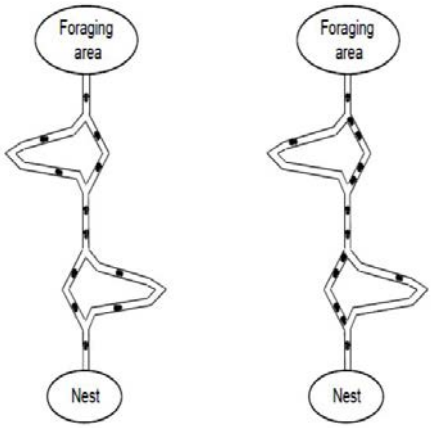


Figure 2.10: Food finding behaviour of real ants

The ants communicate with one another only indirectly through changes in the amounts of pheromone they deposit on their trails during the algorithm's execution. The more amount of pheromone is deposited, the better is the solution (right figure in 2.10). Because the solutions constructed by the ants may not be locally optimal, many ACO algorithms allow the ants to improve their solutions through a local search procedure.

2.4 Single-machine scheduling problems

The single-machine environment is worth to be analysed because he represents a building block for more complex configurations. Many researchers have dealt with job scheduling problems on a single-machine under different constraints. A fundamental issue is the difficulty bound with the one machine scheduling problems that involve sequence dependent setup times. It means that setups are separate from the processing times. Pinedo (2008) proved that the makespan optimization on a single machine with sequence dependent setup times is strongly NP-hard. That is, it is not possible to find optimal solutions in reasonable computational time for large-sized instances but searching local optimal solution via other applications.

In the first place, this section deals with generic single-machine scheduling problems that may be encountered in manufacturing environments. The problems are explained, and a resolution method is given. Later on, the focus is switched towards what better represents the multi-format bottling line case: the sequence-dependent setup times scheduling problems on a single machine, also denoted as SDST-SMSP.

2.4.1 Analysis of the single-machine case

It is firstly considered the problem of scheduling a set of n jobs on one machine. Job j, with $j = 1, ..., n$, is characterized by its processing time p_j. Afterwards, more constraints are added in the β field with different objective functions to optimize.

Minimize the total completion time

The problem of minimizing the sum of the completion times of each of the n jobs, is denoted as:

$$1||\sum_{i=1}^{n} C_i$$

The input of the problem are n given jobs to be processed on a machine, with related processing times p_i, i=1,..., n. The aim is to sequence the jobs in order to minimize the objective function, that means to minimize the total completion time.

An optimal solution can be achieved by using the *Shortest Processing Time First*, also known as SPT rule. The theorem implies to process before the jobs with the minimum processing time.

Minimize the weighted total completion time

The problem of minimizing the sum of the completion times of n jobs with a given associated weight w_i and processing time p_i, is denoted as:

$$1||\sum_{i=1}^{n} w_i C_i$$

The theorem applicable in this situation is the *Weighted Shortest Processing Time* (WSPT) rule. This heuristic procedure jumped out for the first time in a seminal paper by W. E. Smith (1956) and it has been further developed. In the basic form $1||\sum_{i=1}^{n} w_i C_i$ of the problem, a solution is achieved by processing the n jobs ordered by the smaller ratio $\frac{p_i}{w_i}$.

Minimize the maximum Lateness

In this problem there are n given jobs, all available at the beginning (release date $r_i = 0$) and each job have an assigned due date d_i to be respected. The aim is to minimize $L_{max} = max_r L_r$. The notation for this case is:

$$1||L_{max}$$

The lateness of a job i is the difference between its completion time and its due date. The maximum Lateness may be minimized by using the *Earliest Due Date* (EDD) rule due to Jackson (1955). It states that to achieve the solution the n job must be ordered by increasing due dates.

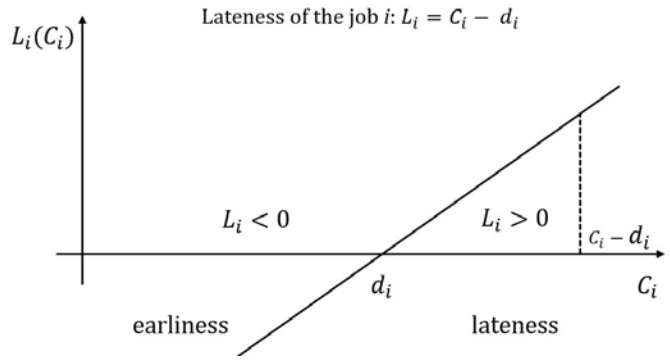


Figure 2.11: Representation of the lateness of a given job i

As it is possible to notice in the figure above, lateness is positive when job i is completed late and negative when it is completed early. If the due date is exceeded, positive lateness, a penalty occurs. The *tardiness* of a job i is instead defined as $T_i = max(C_i - d_i, 0)$. Tardiness is never negative.

Sometimes it may also happen that a firm accepts to delay some jobs in order to finish within the due date other jobs if it is justified by economic reasons.
The *unit penalty* of job i is defined as:

$$U_i = \begin{cases} 1 \text{ if } C_i > d_i \\ 0 \text{ otherwise} \end{cases}$$

Minimize the number of late jobs

In this case, n independent jobs are given, with relevant and known due dates. The preemption is not admissible; it means that the jobs cannot be overtaken. Setup times do not depend on the sequence so they may be either null or included in the processing times.
An algorithm to solve this problem was developed by Moore and Hodgson (1968). The outputs are a set E that includes the sequencing of the non-late jobs and a set L for the late jobs. The late jobs are to be sequenced in any order after the relative jobs of the set E. The steps to follow to solve the algorithm are:

1. Create the sets E* and L*. E* includes the jobs ordered by increasing due date whereas L* is initially empty.

2. Based on the order in E*, determine the completion time of each job, and identify the late jobs.

3. If set E* remains still empty, it means that there are no late jobs so E = E* and L = L*. Stop.

4. If inside set E* there is at least one late job, being k the first late job in the sequence.

5. Identify the job with the longest processing time within the first k jobs of the sequence E*. Remove that job from the set E* and place it in L*. Go to step 2.

2.4.2 SDST-SMSP

Sequence-dependent machine setup times are here considered. That is, if job k is carried out on the machine straightaway after job j, setup time is needed during which the machine cannot process any job. It is denoted as s_{jk} and it is also related to the changeover cost that occurs, c_{jk}. In a bottling line system or a similar multi-model line, when the production switches from one format to another all the line stops to let the required adjustments be done. The objective considered concerns the minimization of the makespan of the schedule or total completion time of all jobs. The aim is a result of minimizing the total setup time.

Solving the SDST scheduling problem

Using the classical notation in Scheduling Theory, this problem is noted as $1|s_{jk}|C_{max}$. This problem is equivalent to the *Travelling Salesman Problem* (TSP), which is one of the most known route and scheduling problems.

The problem of job scheduling with sequence-dependent setup times have been deeply studied in the literature. State of the art surveys can be found in Allahverdi, Gupta and Aldowaisan (1999), Allahverdi et al. (2008), Zhu and Wilhelm (2006). For the one machine case, complexity analysis is not encouraging. Despite this, researchers have developed exact approaches based on branch and bound, dynamic

programming or integer linear programming. The objective function under study in this section is the makespan and can be expressed as:

$$C_{max} = \sum_{j=1}^{n} p_j + \sum_{j \to k} s_{jk}$$

When setup times are dependent on the sequence, minimizing makespan becomes equivalent to minimizing the total setup time. That is because the sum of processing times remains a constant through the whole scheduling when all information about jobs is deterministic and known at the initial time of scheduling (Montoya et al., 2010).

This problem corresponds to what is usually called the Travelling Salesman Problem (TSP). Travelling salesman problem was proposed by mathematicians, Carl Menger and Hustler Wietni in 1930. The problem is that a travelling salesman wants to visit a large number of cities and his goal is to find the shortest path; such that it passes all cities and each city is only passed once and finally returns to the starting point. Linking TSP to the SDST scheduling problem, each city corresponds to a job and the distance between cities corresponds to the time required to change from one job to another. If the setup times for all pairs of jobs are indifferent to their sequencing order when scheduled consecutively, the scheduling problem is equivalent to a symmetrical TSP, otherwise, it is equivalent to asymmetrical TSP.

Related works

One of the initial works on the sequence-dependent setup time problem was presented by Gilmor and Gomory in 1964. They presented a solvable sequencing case applying TSP on a one-state variable machine. Presby and Wolfson (1967) provided an optimization algorithm that is suitable only for small problems. Bianco et al. (1988) formulated the problem with jobs characterized also with non-negative integer release dates r_j as a mixed integer linear program. They developed a heuristic algorithm using lower bounds and dominance criteria. He and Kusiak (1992) examined the SDST scheduling problem with precedence, proposing a simpler mixed-integer formulation and a fast heuristic algorithm of low computational time complexity. Ozgur and Brown developed a two-stage travelling salesman heuristic procedure for

the problem where similar products produced on the machine can be partitioned into families.

There are plural works in literature that consider other objective functions. Barnes and Vanston (1981) combined branch and bound with dynamic programming to solve the problem noted as $1|s_{jk}|\sum w_jC_j + \sum s_{jk}$. For the case of precedence limitations with a special structure (chains), Uzsoy et al. (1992) developed a branch and bound and a dynamic programming algorithm for $1|prec, s_{jk}|L_{max}$. They also developed a dynamic programming algorithm for $1|prec, s_{jk}|\sum U_j$, where the objective function corresponds to the minimization of the number of tardy jobs. Tan and Narashiman (1997) proposed a simulated annealing algorithm to minimize total tardiness. The problem they wanted to solve is denoted as $1|s_{jk}|\sum T_j$. Later, in 2000, they compared the performance of branch and bound, genetic search, simulated annealing and random-start pairwise interchange heuristics for the same problem. Different versions of genetic algorithms have also been presented; see Tan et al. (2000) and Armentano and Mazzini. Gagne et al. (2002) proposed an Ant Colony Optimization (ACO) algorithm for the same problem. Montoya et al. (2010) analysed the problem on a single machine with also release dates, denoted as $1|r_j, s_{jk}|C_{max}$ and proposed a heuristic based on a random insertion strategy.

2.4.3 Karg-Thompson's algorithm

The case at issue is an advancement of $1||\sum_{i=1}^{n} C_i$, where the sequence depends on setup times. There are n given jobs with not relevant due dates and preemption is not admissible. The characteristic of this environment is that in case of changing from a just processed job i to a next job k, a certain amount of time to arrange the machine is needed. The notation for this kind of scheduling problem is:

$$1|s_{jk}|\sum_{i=1}^{n} C_i$$

Good order for the sequence results in a reduction of the total amount of setup times and therefore also the makespan decreases. The heuristic algorithm developed by Karg-Thompson (1968) is a way to reach the aim. The procedure is composed of the following steps:

1. Select two random jobs from the set of the n jobs to be processed.
2. Select a third new job and attempt to place it in the available spots of the current sequence.
3. Calculate the total setup for each new position.
4. Allocate the job to the sequence that allows having the lowest setup time.
5. Repeat from step 2 since all the jobs have been allocated.

The computational complexity of this algorithm is quite low.

$$\sum_{j=1}^{N} j = \frac{N \cdot (N+1)}{2} << N!$$

The K-T algorithm presents a double limit. The result depends on the initial couple that has been picked and on the order of insertion of the other jobs. Although the algorithm of Karg-Thompson does not bring to an optimal solution, it can be considered a good tool to solve scheduling problems with dependent setup times. Moreover, it is possible to iterate it by seeking better solutions by changing the initial couple of jobs.

2.5 Flow shop scheduling problems

Flow shop machine environment better represents the generic case of an automated bottling line. In a flow shop model n jobs (n formats of bottle) require to be processed on m machines in series. Here, the attention is on the permutation flow shop problem, where the process sequences of all the jobs are the same. But the processing times for various jobs on a machine may differ. If an operation is absent in a job, and then the processing time of the operation of the job is assumed to be zero. The objective is to find a job sequence that minimizes the completion time (makespan) of the last job. Often, between successive machines are located buffers. A buffer is an inter-operational warehouse with a defined capacity. The figure below depicts a flow line with K stations and *K-1* buffers of capacity B_i.

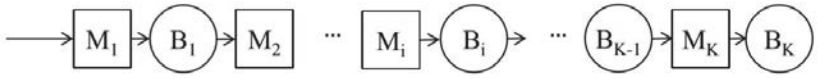


Figure 2.12: Flow shop with intermediate buffers

Random processing times and unreliable stations with stochastic failures and successive repair may lead to blocking and starvation in the line. A station starves if it cannot produce due to lack of material in the upstream buffer, whereas a blocked station ceases production due to a full downstream buffer. The resulting throughput losses can be mitigated by larger buffer capacities. However, buffer capacities can be costly or limited. The decision on the allocation of buffer capacities between station is known as the Buffer Allocation Problem (BAP). The case study will face also this problem. Several researches have carried out models that concern Flow Shop with both unlimited and limited intermediate storage. Broadly speaking, the storage or buffer capacities in between successive machines may be virtually unlimited in case of products physically small (e.g., printed circuit boards, integrated circuits), making it relatively easy to store large quantities between machines. When the products are physically large (e.g., television sets, copiers), then the buffer space in between two successive machines may have a limited capacity. The latter is the situation when a block occurs.

2.5.1 Solving methods

Permutation flow shop is one of the traditional combinatorial optimization problems since the work of Johnson (1954), which is widely applied in real life, such as computer work, industrial engineering, mathematics. Johnson presented an algorithm for solving a scheduling problem in a 2-machine flow shop, noted as $F2||Cmax$; processing times of the machines are assumed to be different. This method was successively advanced by Palmer (1965) who adapted it for a m-machine case; recently Hossain et al. (2014) carried out a study starting from Palmer's in order to solve a 4-jobs and 10-machines flow shop scheduling problem using heuristics. Other extensions of Johnson's rule can be found in Smith and Dudek (1967), Campbell et al. (1970), Baker (1974).

Many other heuristics have been developed for $F_m||C_{max}$; see, for example, Gupta (1972, 1993), Dannenbring (1977), Widmer and Hertz (1989) and Taillard (1990). For complexity results with regard to various objective functions, see Gonzalez and Sahni (1978b) and Du and Leung (1993a, 1993b). Also, exact methods have been developed to solve the flow shop scheduling problems such as the works of Ignall et al. (1965) and McMahon et al. (1967), who applied the branch & bound technique.

The flow shop with limited intermediate storage $F_m|block|C_{max}$ is studied in detail by Levner (1969), Reddy and Ramamoorthy (1972) and Pinedo (1982).

2.5.2 SDST-FSSP

The flow shop scheduling problem with sequence dependent setup times is considered and shown to be NP-complete. It has been an investigated object for years as one of the most popular scheduling problems in manufacturing systems. It means that setup times of the processing jobs depend both on the preceding job and the job to be processed. Due to the too high computation cost of exact methods, more heuristics and meta-heuristics have been developed. See the paper by Wang et al. (2017) for a review on the most related solving techniques for the permutation flow shop scheduling problem with sequence-dependent setup times.

2.6 Implementation of the K-T algorithm

It is now explained the procedure that has been developed to solve the scheduling problem in a multi-model automated bottling line with sequence-dependent setup times and equal machine cycle times $p_j = p$. This explanation will regard only the theoretical and computational aspects, whereas the practical features of the case study will be carried out in a further chapter. The need of the work arises from the cons and the limits of the Karg-Thompson algorithm introduced in 2.4.3. Karg-Thompson's algorithm deals with the sequence whose setup times are relevant. Multi-model lines are one example of a situation of this kind. This is the reason why I focused on this algorithm to solve the $1|s_i k|\sum_{i=1}^{n} C_i$ scheduling problem, stated that in a case like this all the work-stations of the line are set to the same productive speed or cycle time. The limits of Karg-Thompson's algorithm are that the sequence found by the algorithm depends on the first couple of jobs that have been picked and the order of inserting the next jobs. A solution to this issue can be achieved by iterating the algorithms many times.

Given a review on K-T algorithm, before introducing the codified version is it better to sum up the characteristics of the scheduling problem that this codified algorithm will solve.

- The sequence depends on setup times, given in a matrix either symmetric or asymmetric.
- It is attributed to a single-machine scheduling problem or a flow shop environment with equal processing times $p_j = p$.
- There is a list of n jobs to be processed.
- No preemption is admitted. That is, jobs cannot be overcome.
- The first job picked may be either random or chosen by the user. For example, when planning the production of different weeks, it is to be considered that the week *w+1* the machine may still be arranged to process the last job of the week *w*.

The algorithm is built with the programming language Python. It translates the Karg-Thompson algorithm in a code. The user has to enter as input the set of jobs that need to be processed and create the setup matrix. This matrix needs to be read from row to column (predecessor to successor).

```
import random

needed_jobs = 'ABBBBCDEHILK'


setups = [[20, 60, 220, 220, 220, 240, 240, 240, 235, 235, 235, 220, 260],
          [60, 20, 240, 240, 240, 220, 220, 220, 255, 255, 255, 240, 220],
          [220, 240, 20, 210, 210, 210, 60, 230, 225, 225, 225, 210, 210],
          [220, 240, 210, 20, 160, 60, 210, 180, 175, 175, 175, 240, 220],
          [220, 240, 210, 160, 20, 160, 220, 60, 35, 35, 145, 240, 220],
          [240, 220, 190, 60, 195, 20, 210, 180, 175, 175, 175, 220, 240],
          [240, 220, 70, 210, 210, 210, 10, 230, 225, 225, 225, 210, 210],
          [240, 220, 190, 160, 60, 180, 210, 20, 35, 35, 145, 220, 240],
          [290, 290, 200, 170, 110, 230, 260, 150, 35, 35, 215, 310, 310],
          [310, 310, 280, 250, 110, 280, 280, 110, 35, 35, 145, 310, 290],
          [290, 290, 280, 230, 230, 230, 260, 230, 145, 145, 0, 290, 290],
          [220, 240, 210, 240, 240, 220, 230, 220, 255, 255, 255, 10, 90],
          [220, 240, 210, 240, 240, 240, 230, 240, 255, 255, 255, 60, 0]]

letters = 'ABDFHECGKLMJI'

absolute_best_time = -1
absolute_best_list = []
```

Figure 2.13: First part of the algorithm codification in Python

The algorithm is iterated 1000 times in order to find the best possible solution that minimizes the total setup times and to overcome the limits of the traditional algorithm of Karg-Thompson. Given a set of jobs to process, *needed jobs*, and a setup matrix, the code at first translate the letters in numbers.

```
for k in range(1000):
    jobs = [letters.index(l) for l in needed_jobs]

    j1 = jobs[random.randint(0, len(jobs)-1)]
    jobs.remove(j1)

    j2 = jobs[random.randint(0, len(jobs)-1)]
    jobs.remove(j2)

    jobs_list = [j1, j2]
    current_time = setups[j1][j2]

    while len(jobs) > 0:
        new_job = jobs[random.randint(0, len(jobs)-1)]
        jobs.remove(new_job)

        best_pos = len(jobs_list)
        best_time = current_time + setups[jobs_list[-1]][new_job]

        for i in range(len(jobs_list) - 1):
            c = current_time - setups[jobs_list[i]][jobs_list[i+1]]
            c += setups[jobs_list[i]][new_job] + setups[new_job][jobs_list[i+1]]
            if c < best_time:
                best_pos = i
                best_time = c

        current_time = best_time
        jobs_list.insert(best_pos+1, new_job)

    if absolute_best_time == -1 or absolute_best_time > current_time:
        print(jobs_list, current_time)
        absolute_best_time = current_time
        absolute_best_list = jobs_list

print(absolute_best_list, absolute_best_time)
s = ''
for i in absolute_best_list:
    s += letters[i]

print(s)

input()
```

Figure 2.14: Second part of the algorithm codification in Python

The algorithm that is iterated follows the steps of the Karg-Thompson algorithm. Chosen the first two jobs j_1 and j_2 and added them to the *jobs list*, it is calculated the setup time to switch from the first to the second. It is denoted as *current time*. Current time represents the total setup time, that is the function that has to be minimized.

Until the jobs list contains $n > 0$ jobs, a random job from the needed ones is taken. At first, the last position in the sequence is assumed to be the best for a new job. The new setup time is then calculated, to switch from the second-last to the last (the new) job, and it is assumed to be the best one. This time is afterwards used

as a comparison with the other possible positions for the new job.
So, also for the other possible positions in the sequence, excluded the last one, two values are calculated:

- `c` is calculated as the current time less the setup time to switch from the positions between which the new job may be included. Denote them as j_i and j_{i+1}.
- To `c` is now added the setup times to switch from j_i to the new job and from the new job to j_{i+1}.

If `c` is lower than the current `best time`, then that position is assigned to be the best one and `c` the best time. The procedure is iterated until the list of available jobs is empty.
Finally, the optimal sequence and its total setup time are obtained and ready to be used for further studies.

Chapter 3

Simulation and manufacturing

Simulation modelling is considered an effective work tool in manufacturing allowing the system's behaviour to be learned, tested and improved in a low cost and quick way. In this chapter, the concepts behind simulation modelling are discussed, from nature and the purposes of simulation, with a look to simulation in manufacturing, to the classifications of simulation systems. Then, there is an explanation of discrete-event based simulation models and its elements, since the thesis mostly concerns a discrete system such as an automated bottling line. The final part introduces AnyLogic, the simulation software used for the case study.

3.1 Introduction to Simulation

In literature, it is possible to find various definitions of simulation. Ravindran et al. (1987) defined computer simulation as "A numerical technique for conducting experiments on a digital computer which involves logical and mathematical relationships that interact to describe the behaviour of a system over time". By means of computers, simulation allows to imitate, or *simulate*, the operations of different kinds of real-world facilities or processes over time. It can be used either to study and compare alternative designs or to troubleshoot existing systems. The prediction of the future behaviour of the system is then achieved by monitoring the behaviour of different modelling scenarios as a function of simulated time.

3.1.1 A system for each kind of environment

For the purpose of this dissertation, a system is defined as a set of interrelated components working together toward a common objective (Blanchard and Fabrycky, 1990). A deeper definition was pointed out by the Air Force System Command (1991): "A system is a composite of people, products, and processes that provide a capability to satisfy stated needs. A complete system includes the facilities, equipment (hardware and software), materials, services, data, skilled personnel, and techniques required to achieve, provide, and sustain system effectiveness". The components of a system are called *entities*. In practice, what is meant by the system depends on the objectives of a particular study. To describe a system at a particular time, we need a collection of variables, called *state*.

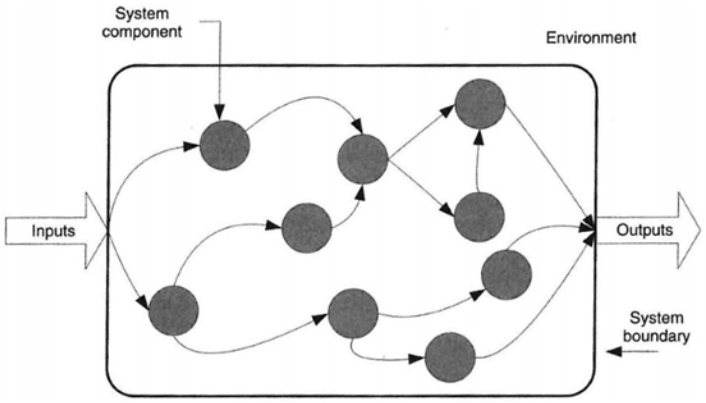


Figure 3.1: A conceptualization of a system

As depicted in the figure above, the components of a system work on the inputs that arrive to produce output. The decisions regarding how to conceptualize the system will drive the level of abstraction within the model. In fact, the model should include enough information to get confident answers for the specific questions asked from the study; at the same time too much not required details can bring confusion into the study or cause a waste of time.

Before going on with the modelling aspect of simulation, it is good to discuss some general system classification. This is triggered by the fact that the modelling phase

is driven by how a system is conceptualized. There are different ways to categorize the systems:

- man-made (e.g., manufacturing system) or natural (e.g., solar system);
- physical (e.g., an airport) or conceptual (e.g., a system of equations);
- stochastic if stochastic or random behaviour is an important component of the system, otherwise it is deterministic;
- static if it does not change significantly with respect to time, else it is dynamic. A dynamic system is said to be discrete if the state of the system changes at a discrete point in time, whilst it is said to be continuous if the state of the system changes continuously with time.

Few systems in practice are wholly discrete or wholly continuous, but since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous.

3.1.2 Application areas

Application areas for simulation are numerous and diverse. Below there is a list of some particular kind of problems for which simulation has been found to be a useful and powerful tool:

- Designing and analysing manufacturing systems;
- evaluating hardware and software requirements for a computer system;
- evaluating a new military weapons system or tactic. Designing communications systems and message protocol for them;
- determining ordering policies for an inventory system;
- designing and operating transportation facilities such as freeways, airports, subways, or ports;

- evaluating designs for service organizations such as hospitals, post offices, or fast-food restaurants;
- analysing financial or economic systems.

Simulation applications can also be sorted by the abstraction level of the corresponding models, as shown in the figure below.

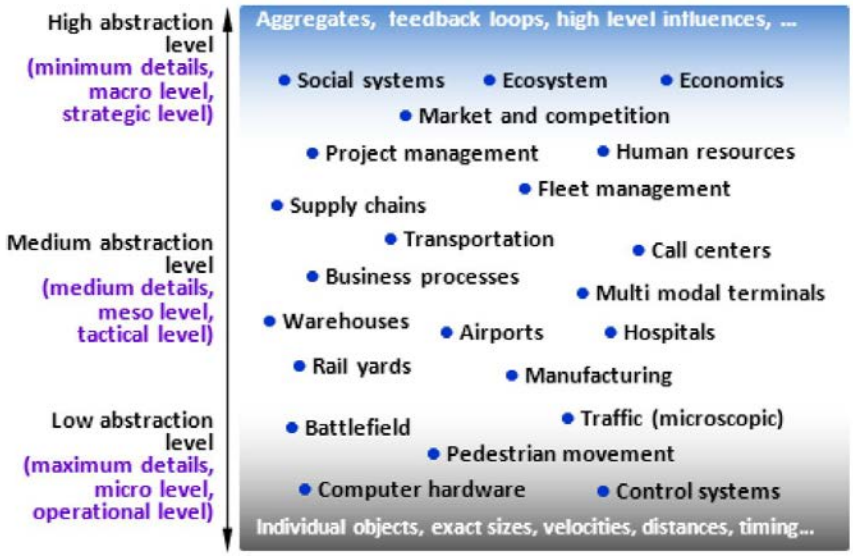


Figure 3.2: Application areas of simulation

At the bottom are the physical-level models that use highly-detailed representations of real-world objects. At this level, it has cared about physical interaction, dimensions, velocities, distances, and timings; problems of this category require low abstraction modelling.

The models at the top are highly abstract, and they typically use aggregates such as consumer populations and employment statistics rather than individual objects. Since their objects interact at a high level, they can help understand relationships without requiring modelling intermediate steps. Other models have an intermediate abstraction level.

3.2 Simulation modelling

At some point in the lives of most systems, there is a need to study them to try to gain insights into the relationship among various components or to predict performances under some new conditions being considered. The figure below maps out different ways in which a system might be studied. This thesis focuses on the mathematical models to be studied by means of simulation, henceforth referred to as simulation models. The interest in building a simulation model derives from the fact that real-world systems are often too complex for analytic models and often too expansive to experiment with directly.

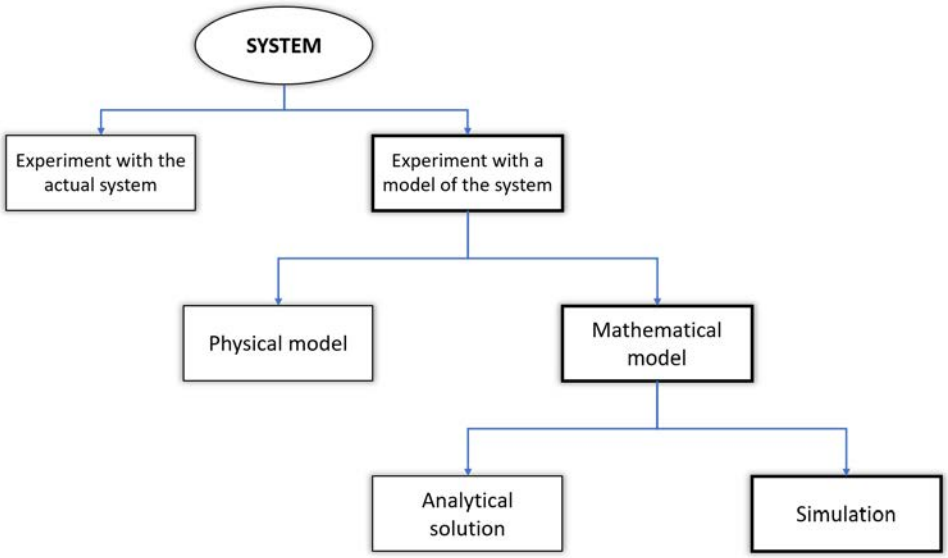


Figure 3.3: Ways to study a system

Simulation modelling has the capability of modelling an entire system and its complex interrelationship. The model usually takes the form of a set of assumptions concerning the operation of a system. These assumptions may be expressed in mathematical, logical, and symbolic relationship between the entities, or objects of interest, of the system. Once developed and validated, indeed, a model can be used to investigate a wide variety of "what if" questions about the real-world system.

The main purpose of a simulation model is to let people study a particular system

and collect observations as a function of time. Simulation can also be used to study systems in the design stage before such systems are built. Thus, simulation modelling can be used both as an analysis tool for predicting the effect of changes to existing systems and as a design tool to predict the performance of new systems under a varying set of circumstances. This is the use of *prescriptive* modelling technique: to convey the required behaviours or properties of a proposed system. Moreover, simulation modelling can be used also in a *descriptive* way to depict the behaviours or characteristics of an existing or proposed system. Even though the most valuable use of simulation is the first expressed, that is to recommend a solution.

3.2.1 Classification of the simulation models

The main purpose of a simulation model is to allow observations about a particular system to be gathered as a function of time. From that standpoint, there are distinct types of simulation models. In a parallel manner as the classifications in 3.1.1, simulation models can be classified along three different dimensions:

- Static or dynamic simulation models. A *static* simulation model, sometimes called a Monte Carlo simulation, represents a system at a particular point in time. *Dynamic* simulation models represent systems as they change over time.

- Deterministic or stochastic simulation models. *Deterministic* models have a known set of inputs, which will result in a unique set of outputs. A *stochastic* simulation model has one or more random variables as input. Random inputs lead to random outputs. Since the outputs are random, they can be considered only as estimates of the true characteristics of a model.

- Continuous or discrete simulation models. *Continuous* simulation requires that observations are collected continuously at every point in time. In a *discrete-event* simulation, observations are gathered at selected points in time when certain changes take place in the system.

The simulation models that will be dealt with in the remainder of this thesis will

be discrete, dynamic, and stochastic; they are the so-called *discrete-event-based simulation models*.

3.2.2 Advantages and disadvantages of simulation modelling

Simulation is intuitively appealing because it mimics what happens in a real system or what is perceived for a system that is in the design stage. Simulation models are "run" rather than solved. Given a particular set of input and model characteristics, the model is run, and the simulated behaviour is observed. This process of changing inputs and model characteristics results in a set of scenarios that are evaluated. A good solution, either in the analysis of an existing system or in the design of a new system, is then recommended for implementation. The advantages of using simulation in problem-solving are multiple, even though drawbacks are also present.

Advantages

1. Simulation models allow to analyse systems, also complex and with stochastic elements, and find solutions where methods such as analytic calculations and linear programming fail.

2. Once the abstraction level has been chosen, it is easier to develop a simulation model than an analytic model.

3. The structure of a simulation model naturally reflects the system's structure.

4. In a simulation model, it is possible to measure values and track entities within the level of abstraction and to add measurements and statistical analysis at any time.

5. "What if" questions can be answered.

6. New hardware design, physical layouts, transportation systems, and so on can be tested without committing resources for their acquisition or disrupting the current ones. Simulation can be seen as a money-saver solution.

7. Simulation allows to study a system with a long-time frame in compressed time, or alternatively to study the detailed workings of a system in expanded time.

8. It allows to play and animate the system behaviour in time. Animations can be useful for demonstrations, verification, and debugging.

9. Simulation models are far more convincing than other presentation tools (for example Excel spreadsheets). If a proposal is supported by simulation, there is a clear advantage over those who only use numbers.

Disadvantages

Simulation is not without its drawbacks. Some disadvantages are as it follows.

1. Each run of a stochastic simulation model produces only estimates of a model's true characteristics for a particular set of input parameters. On the other hand, an analytic model, if appropriate, can often easily produce the exact true characteristics of that model for a variety of sets of input parameters.

2. Simulation models may result expensive and time-consuming to develop.

3. Model building of complex systems might require special training.

4. A basic knowledge of language programming is needed or to be developed.

Finally, it should be noted that in some studies both simulation and analytic models might be useful. In particular, simulation can be used to check the validity of assumptions needed in an analytic model. An analytic model can also suggest reasonable alternatives to investigate in a simulation study.

3.2.3 Simulation languages

Attempting to implement the simulation model, from scratch, in a general-purpose language such as FORTRAN, Visual Basic, C/C++, or Java requires above-average programming skills. Luckily, the repetitive nature of computations in simulation allows the development of computer libraries that are applicable to simulation modelling situations.

The computational power and storage capacity have motivated the development of specialized simulation languages that provide standard programming facilities and will differ in how the user will take advantage of these facilities. There is normally some trade-off between how flexible the language is in representing certain modelling situations. Some languages are more programming oriented (e.g., SIMSCRIPT) and others are more "drag and drop" (e.g., ProModel, Arena, AnyLogic). The latter is the one picked to develop the simulation model of the case study. It will be introduced in chapter 4.

3.3 Simulation in manufacturing

Modelling and simulation are emerging as key technologies to support manufacturing in the 21st century. A great contribution of literature has been given to the modelling and analysis of manufacturing systems such as production lines since the early 1950s because of their economic importance and their academic interest. The cornerstone works are linked to Gershwin (1992), Buzacott and Hanifin (1978) and Papadopoulos (1996). Hosseinpour et al. (2009) presented a comprehensive literature review on the importance of simulation in manufacturing as a very helpful work tool in industrial fields to test the system behaviour. Different aspects of many different system configuration may be analysed in a fast, low cost and secure way through simulation. The use of simulation for solving manufacturing problems results in a simulation model. The concepts of simulation using a simulation software were presented by Kelton (2007). He pointed out the steps that a modeller should follow in order to reach the ability to carry out effective simulation modelling. Among the modelers who developed simulation models in this field we find Seraj (2008) who modelled a rusk production line to increase its capacity, Hecker et al. (2010) who analysed and optimized a bakery production line while Chassapis et al. (2009) simulated a production line to select a preventive maintenance schedule, which gives the best utility and performance values. Hesmat et al. (2013) used a simulation model to analyse and test several bottlenecks that cause severe congestion in different areas on a production line.

3.3.1 The value of production systems models

Simulation is a low cost, secure and fast analysis tool with many different system configurations. Simulation may be used either to design new manufacturing systems or to improve the performance of existing ones. Simulation has been used successfully as a supporting tool in the design of new production facilities, warehouses, or distribution centres. It can be used by analysts and engineers for evaluating the impact of capital investments in equipment and physical facility and of proposed changes to material handling and layout.

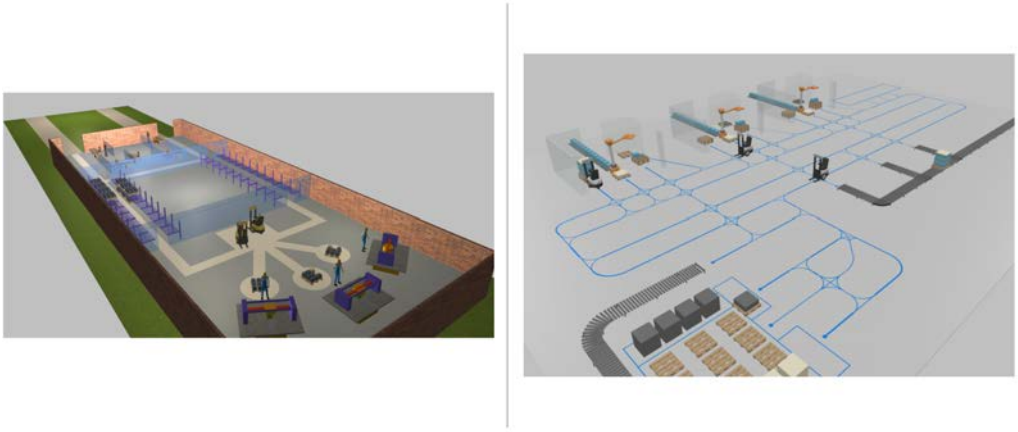


Figure 3.4: Models of a job shop and a palletizing system in AnyLogic

A simulation model can provide a "test drive" without making rushed capital investments in new systems or disrupting the existing system with untried changes. It has to be stated that simulation is not the only tool used in the decision-making process and not all the firms are used to adopt it, but its implementation in manufacturing has contributed to raise the quality of the insights that managers and engineers can have when evaluating a system.

3.3.2 Applications in manufacturing

Already seen the general application of simulation, possible applications of simulation addressed in manufacturing can be categorized listed as it follows (Law and Comas, 1997):

- The need for and the quantity of equipment and personnel: number of type of machines for a particular object, physical arrangement of transporters, conveyors, and other support equipment, location and size of inventory buffers, evaluation of a change in product volume or mix, evaluation of the effect of a new piece of equipment on an existing manufacturing system, evaluation of capital investment
- Performance evaluation: throughput analysis, time-in-system analysis, bottleneck analysis
- Evaluation of operational procedures: production scheduling, inventory policies, control strategies for an automated guided vehicle system, reliability analysis, quality-control policies.

3.4 D.E.S.: Discrete-event simulation models

A manufacturing system most of the time satisfies the requirements to be stochastic, dynamic and discrete. This system does not need to be observed on a continuous basis but only at selected discrete points in time, resulting in the applicability of a particular type of simulation models: the discrete-event simulation model, also referred to with *DES* model, previously disclosed in 3.2.1. DES was developed in the 1960s in industrial engineering and operation research to help analyse and improve industrial and business processes. The term "discrete" refers to the fact that this type of model moves forward in time at discrete intervals and then the events are discrete (mutually exclusive). These factors give the flexibility and efficiency to be used over a very wide range of problems and they allow to provide an intuitive approach in representing complex systems.

Discrete-event simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs; where an *event* is defined as an instantaneous occurrence that may change the state of the system.

Although discrete-event simulation could conceptually be done by hand calculations, the amount of data that must be stored and manipulated for most real-world systems dictates that discrete-event simulations be done on a digital computer.

3.4.1 Components of a DES model

The core concepts of a discrete-event simulation model are entities, attributes, events, resources and queues. It follows a brief explanation of them.

- *Entities* are the discrete items that enter the system, flow through the system, potentially use the resources of the system and depart the system;
- *attributes* are features specific to each entity that allow it to carry information;

- *events* are things that can happen to an entity or the environment;
- *resources* are something that is used by the entities and may constrain the flow of the entities within the system;
- if a resource is busy when an entity needs it, then the entity must wait, forming a *queue*.

3.4.2 The discrete-event clock

In DESs, an event is something that happens at an instant in time which corresponds to a change in the system state. It can be conceptualized as a transmission of information that causes this action of change. For this reason, when simulating a system, it is required to be able to generate a sequence of events so that at the occurrence of each event, the appropriate actions that change the state of the system are invoked. In simulation, events are created by adding logic to the normal state changing actions. This additional logic is responsible for scheduling future events that are implied by the actions of the current events.

There is generally no relationship between simulated time and the time needed to run a simulation on the computer. This happens because of the mechanism that advances simulated time from one value to another. This mechanism is the *simulation clock*. It does not "tick" at regular intervals. Instead, the simulation clock jumps from event time to event time. Historically, two principal approaches have been suggested for advancing the simulation clock: *next-event time advance* and *fixed-increment time advance*, but to stick to the discrete-event simulation models the focus will be on the first one. The next figure represents the next-event time approach for a general single-resource queuing system. The following notation is needed:

- t_i = time of arrival of the ith entity ($t_0 = 0$)
- $A_i = t_i$ - t_{i-1} = inter-arrival time between (i-1)st and ith entities
- S_i = time that the resource actually spends working the ith entity

- D_i = delay in queue of the ith entity
- $c_i = t_i + D_i + s_i$ = time that the ith entity completes service and departs
- e_i = time of occurrence of ith event of any type (ith value the simulation clock takes on, excluding the value $e_0 = 0$)

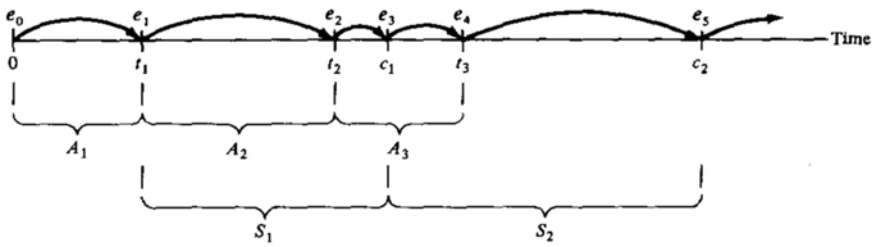


Figure 3.5: The next-event time-advance approach illustrated for a single-resource queuing system

With the next-event time-advance approach, the simulation clock is initialized to zero and the times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the most imminent (first) of these future events, at which point the state of the system is updated to account for the fact that an event has occurred. Then the simulation clock is advanced to the time of the (new) most imminent event, the state of the system is updated, and future event times are determined, etc. This process of advancing the simulation clock from one event time to another is continued until eventually, some pre-specified stopping condition is satisfied.

Since all state changes occur only at event times for a discrete-event simulation model, periods of inactivity are skipped over by jumping the clock from event time to event time.

Chapter 4

AnyLogic ®

Modelling is a way to solve real-world problems. In many cases, it is not affordable to experiment with real objects to find the right solutions: building, destroying, and making changes may be too expensive, dangerous, or just impossible. If that is the case, a model that uses a modelling language to represent the real system is built. This process assumes abstraction: are included the details believed as important and left aside those thought to be not important. The model is always less complex than the original system. Given the model of a system, in order to achieve insights or output data able to solve a problem related to it, it is necessary to translate the conceptual model within a calculator. The means available for doing it are general purpose languages such as Pascal, C, C++ or specialized one like SIMSCRIPT, MODSIM, GPSS. An alternative is to resort to interactive applications for simulating. The most knows are AutoMod, Simul8, Arena Simulation, Witness, Extend, Micro Saint and AnyLogic. They are easy to use and therefore suitable for building models quickly.

Among the developed software that predicate on modelling the process flow of "entities" through a system, AnyLogic is the one picked for carrying on this dissertation and to develop the case study. AnyLogic is a multimethod simulation modelling tool able to support agent-based, discrete event, and system dynamics simulation methodologies (Weimer et al., 2016).

Figure 4.1: AnyLogic's icon

With AnyLogic modellers can create prototypes of different systems during the phases of the study, designing or development. Models are further used to investigate aspects and details concerning the design and implementation of the related systems in an easy way and without risks. AnyLogic is written in Java. Moreover, it is characterized by a rapid and intuitive modelling style. It includes a graphical modelling language based on a drag-and-drop of items and also allows the user to extend simulation models with Java code. The model can be built in a 2D or 3D environment, where animations can play a crucial role in the representation of a real system.

The remainder of the chapter deals with the historical background related to the development of the software and then its key feature to be a multi-method simulation software. Agent-based, System Dynamics and Discrete-event based simulation models are introduced. Section 3 concerns the practical issues and AnyLogic's workspace. It presents the main components used to create a model and the main characters of it as well as the different libraries included in AnyLogic. The final sections are a brief look at the output analysis and the experimental phase.

It has to be clarified that not all AnyLogic's components, features and functions are named. Moreover, the practical explanations are not enough to get proper knowledge of the use of AnyLogic. However, the application of many concepts introduced can be seen in the Case Study's chapters.

4.1 History of AnyLogic

At the start of 1990, there was a big interest in the mathematical approach to modelling and simulation of parallel processes. This approach may be applied to the analysis of correctness of parallel and distributed programs. The Distributed Computer Network (DCN) research group at Saint Petersburg Polytechnic University developed such a software system for the analysis of program correctness; the new tool was named COVERS (Concurrent Verification and Simulation). This system allowed graphical modelling notation for system structure and behaviour. The tool was applied for the research granted by Hewlett-Packard.

In 1988 the success of this research inspired the DCN laboratory to organize a company with a mission to develop a new age simulation software. The emphasis in the development was placed on applied methods such as simulation, performance analysis, the behaviour of stochastic systems, optimization and visualization. New software released in 2000 was based on the latest advantages of information technologies: an object-oriented approach, elements of the UML standard, the use of Java, a modern GUI, etc. (Molderink et al., 2009). The tool was named AnyLogic because it supported all three well-known modelling approaches: system dynamics, discrete event simulation, agent-based modelling, and any combination of these approaches within a single model (Borshchev and Filippov, 2004; Bazan and German, 2012).

The first version of AnyLogic was AnyLogic 4 because the numbering continues the numbering of COVERS 3.0. A big step was taken in 2003 when AnyLogic 5 was released. The new version was focused on business simulation in different industries. AnyLogic 7, was released in 2014. It featured many significant updates aimed at simplifying model building, including enhanced support for multimethod modelling, decreased need for coding, renewed libraries, and other usability improvements. AnyLogic 7.1, also released in 2014, included the new GIS implementation in the software: in addition to shapefile-based maps, AnyLogic started to support tile maps from free online providers, including OpenStreetMap. 2015 marked the release of

AnyLogi 7.2 with the built-in database and the Fluid Library. Since 2015, AnyLogic Personal Learning Edition (PLE) is available for free for the purposes of education and self-education. The PLE license is perpetual, but created models are limited in size. The new Road Traffic Library was introduced in 2016 with AnyLogic 7.3. AnyLogic 8 was released in 2017. Beginning with Version 8.0, the AnyLogic model development environment was integrated with AnyLogic Cloud, a web service for simulation analytics. The platform for AnyLogic 8 model development environment is Eclipse.

4.2 A multi-method simulation software

One of the key advantages of AnyLogic is that its models can be based on any of the main simulation modelling paradigms: discrete event or process-centric (DE), systems dynamics (SD), agent-based(AB). It is also possible to have a model that combines the three kinds of models together.

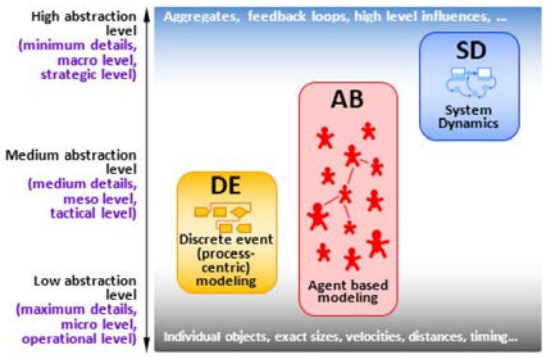


Figure 4.2: The three methods in simulation modelling

In simulation modelling, a method can be defined as a framework used to map a real-world system to its model. It can be seen as a type of language or a sort of "terms and conditions" for model building. The methods that can be considered in modern simulation modelling are three: DES, SD and AB. Each method serves a specific range of abstraction levels. System dynamics assumes very high abstraction, and it's typically used for strategic modelling. Discrete event modelling supports medium and medium-low abstraction. In the middle are agent-based models, which can vary from very detailed models where agents represent physical objects to the highly abstract models where agents represent competing companies or governments. A model should be always selected after having carefully considered the system to model and the goals of the study.

AnyLogic allows the modeller to combine these simulation approaches within the same model. As an example, one could create a model of the package shipping industry where carriers are modelled as agents acting/reacting independently whereas the inner workings of their transport and infrastructure networks could be modelled

with discrete event simulation. Similarly, one can model consumers as agents whose aggregate behaviour feed a systems dynamics model capturing flows such as revenues or costs which do not need to be tied to individual agents. This mixed language approach is directly applicable to a wide variety of complex modelling problems that may be modelled via any one approach albeit with compromises.

4.2.1 Agent-based modelling

Agent-based modelling is a relatively new method compared to system dynamics and discrete event modelling. In fact, agent-based modelling was largely an academic topic until simulation practitioners began using it some 15 years ago.

It was triggered by:

- A desire to gain deeper insights into systems that traditional modelling approaches don't capture well;
- Advances in modelling technology made possible by computer science, such as object-oriented modelling, UML, and statecharts;
- The rapid growth of CPU power and memory. Agent-based models are more demanding than system dynamics and discrete event models.

Agent-based modelling offers a modeller another way to look at the system. The idea behind is that even though it is not known how a system behaves, be able to identify its key variables and their dependencies or recognize a process flow, it is possible to have insights into how the system's objects behave. In this case, it is possible to start building the model by identifying the objects (agents) and defining their behaviours. Afterwards, the agents created can be connected and allowed to interact or put in an environment which has its own dynamics. Agents in an agent-based model may represent very diverse things: vehicles, units of equipment, projects, products, ideas, organizations, investments, pieces of land, people in different roles, etc. The system's global behaviour emerges from many (tens, hundreds, thousands, millions) concurrent individual behaviours. There is no standard language for agent-based modelling, and an agent-based model's structure comes from graphical editors

or scripts. There are many ways to specify an agent's behaviour. Frequently agent has a notion of state and its actions and reactions depend on the state; then the behaviour is best defined with statecharts.

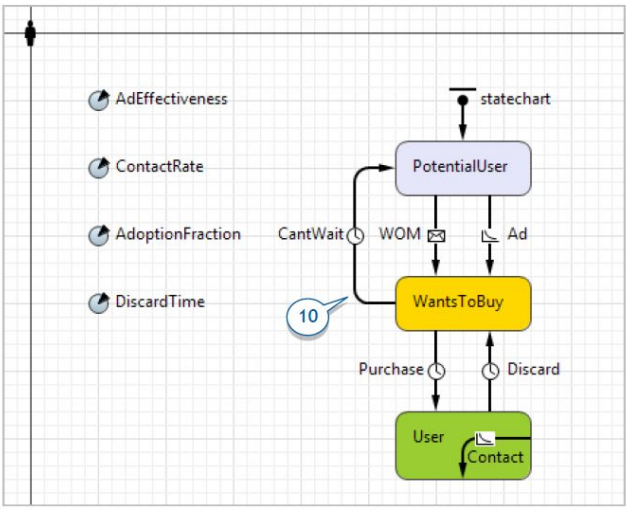


Figure 4.3: Example of statechart in an Agent-based model

Sometimes behaviour is defined in rules executed upon special events. In many cases, the best way to capture the agent's internal dynamics is to use system dynamics or a discrete event approach, and then place a stock and flow diagram or a process flowchart inside an agent. Similarly, outside agents, the dynamics of the environment where they live is often naturally modelled using traditional methods. It's why many agent-based models are multi-method models.

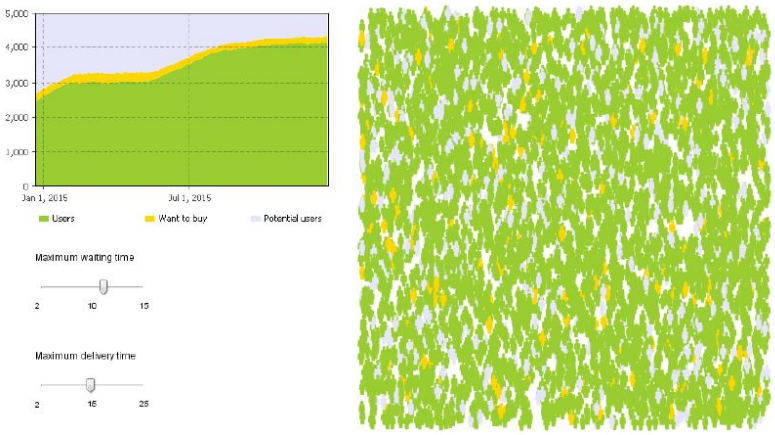


Figure 4.4: Example of Agent-based model

4.2.2 System Dynamics modeling

The system dynamics method was created in the 1950s by MIT Professor Jay Forrester. Drawing on his science and engineering background, Forrester sought to use the laws of physics, in particular, the laws of electrical circuits, to investigate economic and social systems.

Today, system dynamics is typically used in long-term, strategic models, and it assumes high levels of object aggregation: SD models represent people, products, events, and other discrete items by their quantities. System dynamics is a methodology to study dynamic systems. It suggests to:

- Model the system as a causally closed structure that defines its own behaviour.
- Discover the system's feedback loops (circular causality) balancing or reinforcing. Feedback loops are the heart of system dynamics.
- Identify stocks (accumulations) and flows that affect them.

Stocks are accumulations and characterize the system state. They are the memory of the system and sources of disequilibrium. The model works only with aggregates - the stock's items are indistinguishable. Flows are the rates at which these system states change. Stocks are usually expressed in quantities such as people, inventory levels, money, or knowledge, while flows are typically measurements of quantities in a given time period such as clients per month or dollars per year.

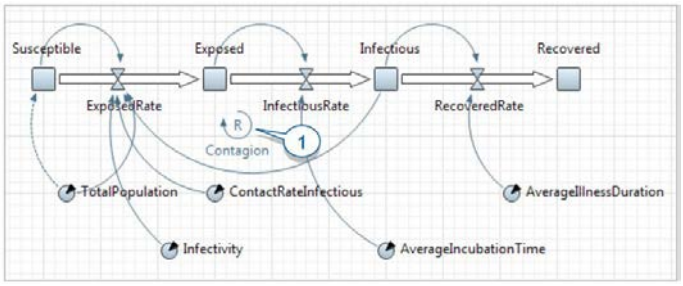


Figure 4.5: Example of System Dynamics model

4.2.3 Discrete-event modelling

Discrete event modelling is nearly the same age as system dynamics. In 1961, IBM engineer Geoffrey Gordon introduced GPSS, considered to be the first software implementation of the discrete event modelling method. Today, a number of programs - including modern versions of GPSS - offer discrete event modelling.

In a Discrete-event model, the system is seen as a process, a sequence of operation that agents perform. A model's operations can include delays, service by various resources, process branch selections, splits and many others. As long as agents compete for limited resources and can be delayed, queues will be part of nearly all discrete event models. The model is specified graphically as a process flowchart where blocks represent operations. The flowchart usually starts with "source" blocks that generate agents and inject them into the process and ends with "sink" blocks that remove them. Agents, originally named transactions in GPSS or entities in other simulation software, can represent clients, patients, phone calls, physical and electronic documents, parts, products, pallets, computer transactions, vehicles, tasks, projects, ideas, and so forth. Resources represent staff, doctors, operators, workers, servers, CPUs, computer memory, equipment, and transport. Service times and agent arrival times are usually stochastic, and since they are drawn from a probability distribution, discrete event models are themselves stochastic. In simple terms, this means a model must run for a specific amount of time or complete a specific number of replications before it produces meaningful output.

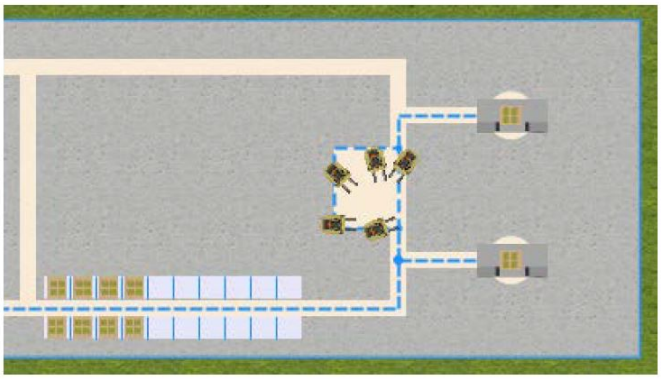


Figure 4.6: Example of DES model in AnyLogic: a job shop

Typical output expected from a discrete event model includes:

- Utilization of resources
- Time spent in the system or its part by an agent
- Waiting times
- Queue lengths
- System throughput
- Bottlenecks

4.3 Modelling in AnyLogic

It is now given a brief explanation of modelling with AnyLogic. This section deals with the basic concepts and items of AnyLogic's environment: agents, blocks, flowchart, diagram grid, statecharts, parameters, variables, etc. The software has available an interactive guide, the so-called *AnyLogic Help*, where all the elements are explained. In spite of that, the only guide is not sufficient for being able to build a model as soon as one starts to use AnyLogic. In fact, it can be considered a learning-by-doing software for the wide variety of possibilities that a modeller has for representing and characterizing its system in the simulation environment. A basis on Java's concepts is also required due to the nature of the software.

In AnyLogic's workspace, all the available model items are contained in the *palette* view grouped by categories in a number of stencils (palettes). The modeller drags the element for the palette and drops it in the diagram of the model that might be seen as a canvas for the model under construction. Each agent's type has its own diagram. *Libraries* are the bases of the simulation, and it is in this section of the palette where it is possible to change how the system acts. AnyLogic includes six standard libraries; they can be mixed together. The *System Dynamics* palette contains elements frequently used by system dynamics modelers while the *Statechart* palette contains elements of statecharts. Statecharts are advanced construct to describe event- and time-driven behaviours. The *Agent* palette contains general elements used for defining dynamics of the model, its structure and data such as parameters, variables and more. The *Space Markup* palette contains elements for marking up the space in models to define, for instance, agent locations. The *Analysis* palette contains elements, used for collecting, viewing and analysing output data. There are then palettes for the graphical aspects like the *Presentation* palette (shapes used to draw presentations of the models) and the *3D Object* (set of 3D images of frequently modelled objects).

In addition, the *Projects* view allows to access the AnyLogic models opened in the workspace, and the workspace tree helps to easily navigate them. The *Properties*

view allows to view and modify the selected item's properties. Another feature that helps make AnyLogic such a powerful modelling tool is 3D animation. AnyLogic's camera objects allow defining the view that displays in the 3D window.

The chapter continues with a more detailed look at the main items contained in the palettes.

4.3.1 Agents, attributes and behaviours

Agents are a model's building blocks, and they can be used to model all kinds of real-world objects, including organizations, companies, trucks, processing stations, resources, cities, retailers, physical objects, controllers, and so on. Each agent typically represents one of the model's logical sections. This allows decomposing a model into many levels of detail.

Parameters and variables

Attributes are characteristics linked to the agents in order to define them and to reflect the characteristics of the object they mimic. To describe objects statically it is used the *parameter* while to store the results of a model simulation or to model some data units or object characteristics, changing over time, *variable* are needed. They can both be found in the Agent palette and they can be of different value types. There are eight primitive data types in Java, but in AnyLogic models are typically used these four: double, int, boolean and String.

Figure 4.7: Parameter, variable and collection in AnyLogic

Another type of variable is the *Collection*. A collection represents a group of objects, known as its elements. Collections are used for defining data objects that group multiple elements into a single unit. Collections are used to store, retrieve and manipulate aggregate data. For example, a collection can be used to store a set of agents - e.g., neighbours or colleagues of a given agent.

Statecharts

The best way to define behaviour in AnyLogic is by using statecharts. A statechart is a state transition diagram. The statechart's states are alternative, which means the object can only be in one state at a time. A transition execution may lead to a state change that makes a new set of transitions active. Statechart is used to show the state space of a given algorithm, the events that cause a transition from one state to another, and the actions that result from state change. By using statecharts you can visually capture a wide variety of discrete behaviours, much richer than just idle/busy, open/closed, or up/down status offered by most block-based tools.

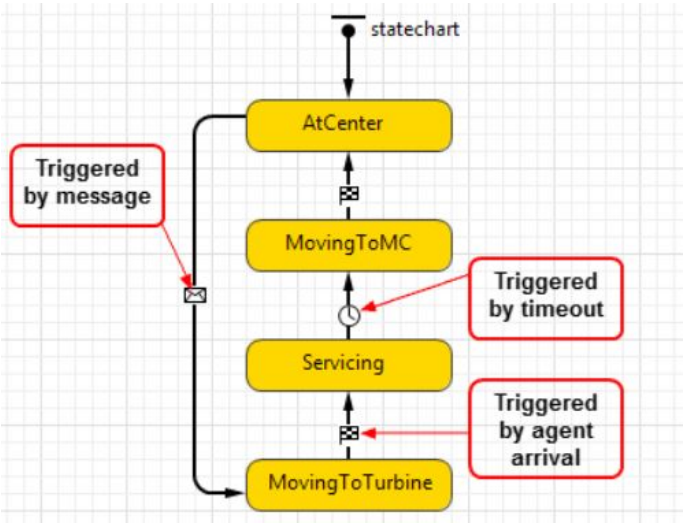


Figure 4.8: Statechart with different transition's triggers

States changes can be triggered as a result of various types of events occurred like timeout, rate, condition, agent arrival or message. The trigger type is specified in the "Triggered by" property of a transition.

4.3.2 The Network

Paths and nodes are space markup elements that define the locations of agents:

- A *node* is a place where agents may reside or perform an operation.
- A *path* is a route that agents can use to move between nodes.

Together, nodes and paths make up a network that a model's agents can use to move along the shortest paths between their origin and destination nodes. It is usually created a network when the model's processes take place in a defined physical space and it has moving agents and resources. Path and nodes are general space markup elements, like also attractors and pallet rack. They can be dragged into the diagram from the space markup palette. The Space markup palette contains also elements for a more aimed-purposes environment such as conveyors, stations, walls, railway tracks and more. Other tools to sketch the graphical environment can be found in the Presentation palette.

Resources

Resources are objects that agents use to perform a given action. An agent must obtain the resource, perform the action, and then release the resource. Some examples of resources include hospital model's doctors, nurses, equipment, and wheelchairs; supply chain model's vehicles and containers; warehouse model's forklift trucks and workers.
There are three types of resources: static, moving, and portable.

- Static resources are bound to a specific location, and they cannot move or be moved
- Moving resources can move independently
- Portable resources can be moved by agents or by moving resources.

In AnyLogic, the Process Modeling library's *ResourcePool* block defines each set or pool of resources.

Figure 4.9: ResourcePool block in AnyLogic

Resource units can have individual attributes, and each resource has a graphical diagram where adding elements such as statecharts, parameters, and functions.

4.3.3 AnyLogic Libraries

The blocks in AnyLogic's Libraries allow using of combinations of agents, resources, and processes to create process-centric models of real-world systems. It has been spoken about agents and resources earlier in this section, and the model is built upon that foundation by defining processes as operations sequences that include queues, delays, and resource utilization. The model's processes are defined by *flowcharts*, the graphical process representations constructed from the Modeling Library's blocks.

AnyLogic includes the following standard libraries:

- **The Process Modeling Library** supports discrete-event modelling paradigms. With Process Modeling Library objects it is possible to model the real-world systems in terms of agents (transactions, customers, products, parts, vehicles, etc.), processes (sequences of operations typically involving queues, delays, resource utilization), and resources. The processes are specified in the form of flowcharts - a widely adopted graphical representation used in many areas: manufacturing, call centres, business processes, logistics, healthcare, etc.

- **The Pedestrian Library** is dedicated to simulating pedestrian flows in a physical environment. It allows creating models of pedestrian-intensive buildings (like subway stations, security checks etc.) or streets (large numbers of pedestrians). In models created with the Pedestrian Library, pedestrians move in continuous space, reacting to different kinds of obstacles (walls, different kinds of areas), as well as other pedestrians.

- **The Material Handling Library** assists in process simulation in factories and warehouses. The library contains conveyors, transporters, and other elements simplifying the creation of detailed production models.

- **The Rail Library** supports modelling, simulating, and visualizing operations of a rail yard of any complexity and scale. The rail yard models can be combined with discrete event or agent-based models related to: loading and unloading, resource allocation, maintenance, business processes, and other transportation activities.

- **The Fluid Library** allows the user to model storage and transfer of fluids, bulk goods, or large amounts of discrete items, which are not desirable to model as separate objects. The library includes blocks such as a tank, pipeline, valve, and objects for routing, merging and diverging the flow.

- **The Road Traffic Library** allows users to simulate vehicle traffic on roads. The library supports detailed, physical level modelling of vehicle movement. Each vehicle represents an agent that can have its own behavioural patterns inside. The library allows users to simulate vehicle movement on roads, considering driving regulations, traffic lights, pedestrian crossings, priorities at junctions, parking lots, and public transport movements. The library is suitable for modelling highway traffic, street traffic, on-site transportation at manufacturing sites, or any other systems with vehicles, roads, and lanes.

Besides those above, the user can also develop a set of reusable agents and Java classes for a particular application area, package them and save as a library. Such custom library can be opened in the palette view along with the standard ones.

The case study in chapter 6 will use blocks from the Process Modeling Library to build a model that represents the bottleneck of an automated bottling line, therefore further explanations on the blocks as well as many other elements previously introduced are not present in this chapter, but they will be presented later.

4.3.4 Output Analysis

As stated in chapter 7 of "Simulation Modeling and Arena" by Rossetti (2015), the inputs in a simulation model are random; hence, the outputs are also random. In AnyLogic, randomness can be controlled through the dedicated pane in the simulation experiments. Managed the randomness and run the simulation, it is relevant to see the statistical output and analyse how the model performed. Results can be visualized through plots.

Randomness in AnyLogic

There are three possibilities for modelling the randomness in AnyLogic's simulations:

1. Random seed: the software initializes a different seed for each experiment performed. Thus, the model runs cannot be reproducible.

2. Fixed seed: the user sets a fixed seed for the randomness. This option is very valuable in the development phase as the simulations are reproducible.

3. Custom generator: it lets the user create his own random class.

Visualization

Visualization of data values is vital when the user has to get the right perception of the simulation. Therefore, making good visualization is very important. In AnyLogic is it possible to use different plots for the visualization of the results or of a value that changes during the run. Plots like time plot, bar plot, stack chart plot and more.

4.3.5 Experiments

Another key feature of AnyLogic is represented by the experiments. The basic one is the Simulation experiment, that runs the model in the way the user has set it. Other experiments may be used when the model parameters play a significant role and it is needed to analyse how they affect the model behaviour, or when it is wanted to find optimal parameters of the model.

AnyLogic supports the following types of experiments:

- Simulation experiment
- Parameters variation experiment
- Optimization experiment
- Monte Carlo experiment
- Compare runs experiment
- Sensitivity analysis experiment
- Calibration experiment

In addition to them, Custom experiment runs an experiment with custom scenario entirely written by the user. The custom experiment gives maximum flexibility with setting parameters, managing simulation runs, making decisions. It simply gives a code field where do all that (and a lot more) by using a rich Java API of AnyLogic engine (functions like run(), stop(), etc.). This experiment has no built-in graphical interface as well as no predefined behaviour.

Chapter 5

Problem-solving with Simulation

The successful application of a simulation approach in a study requires a lot of up-front work prior to building a computer simulation model. This chapter regards the steps to follow in order to build a simulation model by discussing the problem-solving process in the context of an iterative methodology. It will be explained what this iteration concerns and a proper methodology will be presented. The most followed within this dissertation is the *General Methodology for Applying Simulation to Problem Solving* (Rossetti, 2015), that is also the method followed in the developing of the case study. Five major phases are identified: problem formulation, simulation model building, experimental design and analysis, evaluation and iteration, implementation. They will be discussed in detail gradually by identifying the steps to follow the good execution of the methodology.

5.1 The methodology

Detailed modelling and coding are just part of an overall simulation effort to understand or design a complex system, and that attention must be paid to a variety of other concerns, ranging from statistical experimental design to problem formulation and model validation. Many papers have appeared in the past on the process of applying the simulation technique (also referred to with "simulation methodology"), the successful application of simulation and how to avoid the pitfalls of simulation. Among the most famous pioneers, we can find Law (1991), Musselman (1992), Sadowski (1989), Ulgen (1991) and Rossetti (2009).

In the following sections, the steps one must follow in applying the simulation

methodology to solve real problems are described. Five major phases are identified for the proper application of simulation; each phase is further broken into steps. The phases are:

1. Problem formulation
2. Simulation model building
3. Experimental design and analysis
4. Evaluation and iteration
5. Implementation

Although these phases are generally applied in sequence, one may return to the previous phases due to changes in the scope and objectives of the study. It is also possible that something that emerges in further steps implies to go back to a previous phase in order to apply changes.

5.1.1 The DEGREE problem-solving methodology

A methodology is simply a series of steps to follow. Since simulation involves systems modelling, a simulation methodology based on the general precepts of solving a problem through systems analysis is first presented. A general methodology for solving problems can be stated as follows:

1. Define the problem. It helps to ensure that the problem under solving is right.
2. Establish measures of performance for evaluation. It helps to ensure that the reason to solve the problem for is right and to check that the metrics are coherent with the problem.
3. Generate alternative solutions.
4. Rank alternative solutions. With the previous step, it ensures that there are looks at and evaluations of multiple solutions to the problem.

5. Evaluate and iterate during the process. It evaluates how the process is proceeding and allows for iteration.

6. Execute and evaluate the solution. If there is the opportunity, the solution should be executed by implementing the decisions.

This methodology can be referred to by using the first letter of each step. The DEGREE methodology for problem-solving represents a series of steps that can be used during the problem-solving process.

The concept of iteration means that the problem-solving process can be repeated until the desired degree of modelling fidelity has been achieved. The first model should be representative at a level that allows it to be initiated. It is suggested to start with simple models and then to build them up until the desired goals are reached. It is important to get started and get something established on each step and continually go back in order to ensure that the model is representing reality in the way that the modeller wants to intend.

5.2 Applying simulation to problem-solving

Despite the DEGREE problem-solving methodology works well for the modellers, often it is not enough when simulation involves certain unique actions that must be performed during the general overall problem-solving process. When applying the DEGREE to a problem that may require simulation, the general DEGREE approach needs to be modified to explicitly consider how simulation will interact with the overall problem-solving process. The next figure depicts a general methodology for applying simulation to problem-solving developed by Rossetti (2015) with its phases that will be further discussed. The phases, that are composed of many steps, can be collected in 5 main ones: problem formulation, simulation model building, experimental design and analysis, evaluation and iteration, implementation. The book of Rossetti (*Simulation Modeling and ARENA*, 2015) contains all the detailed procedures to carry on the methodology.

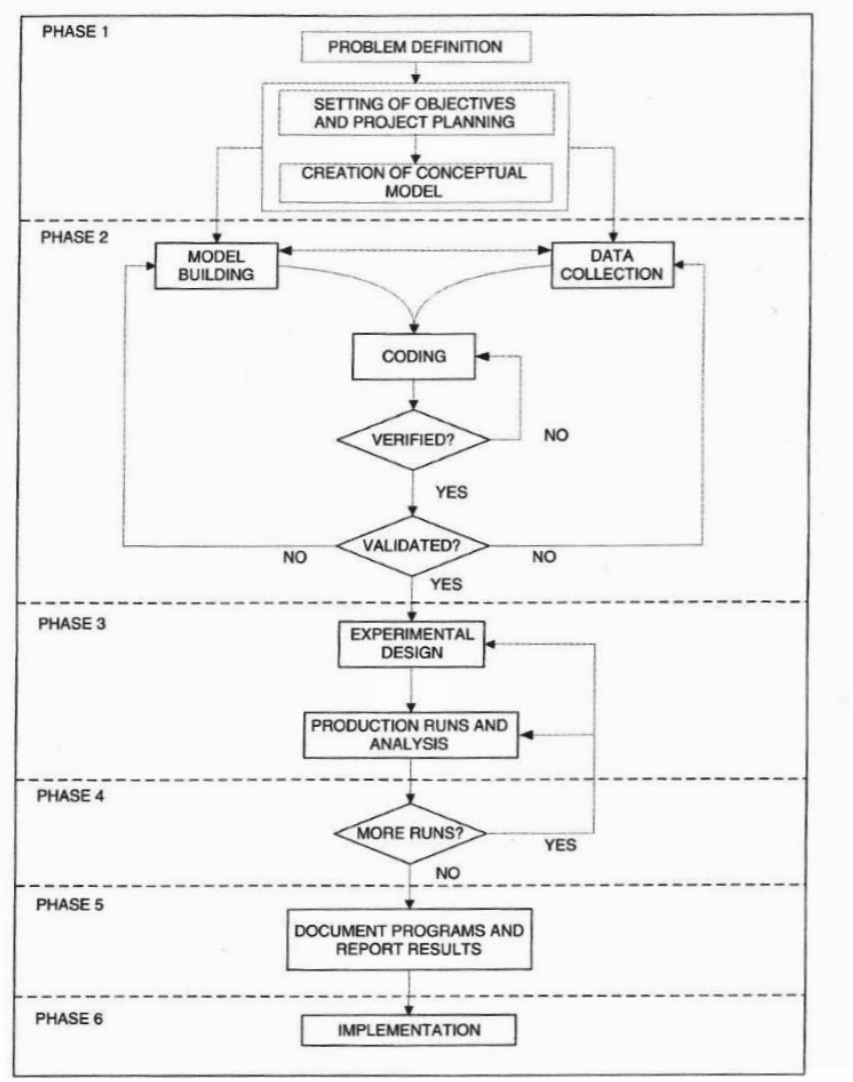


Figure 5.1: General simulation methodology

The phases are overlapped with the DEGREE methodology. The first phase, problem formulation, captures indeed the essence of the first two steps of the DEGREE process. The second phase, model building, captures the essence of step 3 of the DEGREE process. When building models, certain design alternatives are either explicitly or implicitly being developed. The third phase, experimental design and analysis, encloses some of steps 3 and 4 of the DEGREE process. In designing experiments, design alternatives are specified, and when analysing experiments, their worth is being evaluated with respect to problem objectives. The fourth phase, evaluate and iterate, captures the notion of iteration. Finally, the fifth and sixth phases, documentation and implementation complete the simulation process.

5.2.1 Problem formulation

Every study must begin with an evident statement of the study's overall objectives and specific issues to be addressed. The problem formulation phase of the study consists of the following activities:

- Defining the problem
- Defining the system
- Establishing performance metrics
- Building conceptual models
- Documenting model assumptions

These activities are useful in developing an appreciation for and an understanding of what needs to be solved, since a problem starts whit a perceived need.

Defining the problem

This phase of the simulation process has the most effect on the total simulation study since a wrong problem definition can waste a lot of time and money on the project. The output of this activity is a problem definition statement, that is a narrative discussion of the problem necessary to accurately and concisely represent the problem for the analyst and the problem stakeholders. It should include a detailed description of the objectives of the study, the desired outputs from the model, and the types of scenarios to be examined or decisions to be made.

In addition, during this preparatory phase, more steps might be developed:

- Estimate how long a project will take and which resources (financial, human, etc.) will be used for the study. A tool very useful to point out these aspects is the PERT chart, that gives the minimum, maximum and mode duration for each task in order to estimate the total project time at different levels of confidence.
- Perform a cost-benefit analysis.

- Create a planning chart (e.g., Gantt chart) of the proposed project.

Defining the system

A system definition statement is necessary to accurately and concisely define the system, particularly its boundaries. This ensures that the simulation study is focused on the appropriate areas of interests to the stakeholders and that the scope of the project is well understood.

Establishing performance metrics

In this phase, the required objective and measurable performance metrics for the model are needed to be figured out. They are necessary to meaningfully compare alternative scenarios. The performance metrics should include either quantitative statistical measures and qualitative assessments. The focus should be placed on the performance measures that are considered to be the most important to system decision makers and tied directly to the objectives of the simulation study.

Building conceptual models

Prior to create the simulation model with any software, it is good to use conceptual modelling tools in order to create a conceptual model. This aims to convey a more detailed system description so that the model may be translated into a computer representation. A conceptual model can be depicted as a context diagram, activity diagram or with the use of Software Engineering Diagrams.

When modelling, it is important to start with an easy conceptual model that captures the basic aspects and behaviours of the system. Then, adding details, considering additional functionality. Finally, it is to be remembered that the complexity of the model has to remain proportional to the quality of the available data and the degree of validity necessary to meet the objectives of the study.

Documenting model assumptions

Since the model is just a mimic of the real system, it would be impossible to recreate the system exactly as it is. The assumptions help us to fill the gap. They must be summarized at this step. This includes assumptions regarding the behaviour of model components, input data, model detail level, start-up conditions of the model, etc. It is to be decided which components of the real system will be excluded, included as a black box, included in moderate detail, or included in fine detail.

5.2.2 Simulation Model Building

During the simulation model building phase, alternative system design configurations are developed based on the previously developed conceptual models. Additional project planning is also performed to yield specifications for the equipment, resources, and timing required for the development of the simulation models. The simulation models used to evaluate the alternative solutions are then developed, verified, validated, and prepared for analysis. Within the context of a simulation project, this process includes input data preparation, model translation, verification and validation.

Input data preparation

Input data should be analysed and tested through the utilization of statistical tools. Patterns in data should be identified, if any, and incorporated as part of input data generation. Theoretical distributions should be fitted to actual data and used in the model whenever possible (Law et al., 1991).

Model translation

The procedure for coding the model are described, including timing and general procedures and the translation of the conceptual models into computer simulation program representations.

The simulation modeller must also decide whether to program the model in a general-purpose language such as FORTRAN, Pascal, Java or C or in a specially designed

simulation language (Flexsim, AnyLogic, Arena, etc.).

Verification

Verification of the computer simulation model is performed to determine whether or not the program performs as intended. To perform model verification, model debugging is performed to locate any errors in the simulation code. Model debugging also includes scenario repetition utilizing identical random seeds, "stressing" the model through a sensitivity analysis to ensure compliance with anticipate behaviour and testing of individual modules within the simulation code.

Validation

Validation of the simulation model is performed to determine whether or not the simulation model adequately represents the real system. That is, the model must have an acceptable level of confidence in the performances processing assumed. The simulation model is shown to personnel (of various level) associated with the system in question. Their input concerning the realism of the model is critical in establishing the validity of the simulation. In addition, further observations of the systems are performed to ensure model validity with respect to actual system performance. A simple technique is to statistically compare the output of the simulation model to the output from the real system and to analyse whether there is a significant difference between the two.

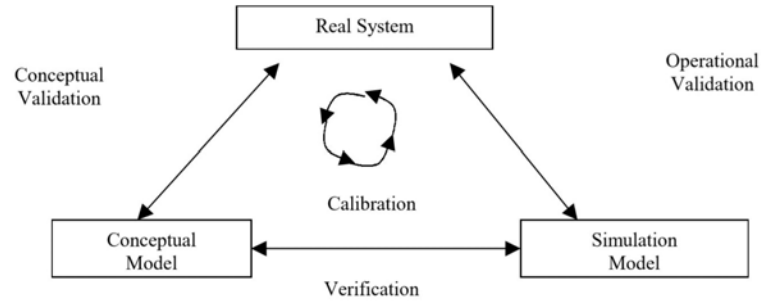


Figure 5.2: Conceptual Validation, Verification and Operational Validation in Simulation

The relationship between the conceptual validation, verification and operational

validation of a model is shown in figure 5.2. A rigorous validation procedure for the conceptual model is as important as the verification and operational validation of the model because, being earlier than the others, it saves time and redirects in the right direction before a lot of time is wasted in the study.

5.2.3 Experimental Design and Analysis

Experimentation with the model and applying the design of experiments techniques in order to investigate the results driven from the model is the third big phase of the simulation process. Preliminary simulation experiments should be performed to set the statistical parameters associated with the main experimental study. The experimental method should use the simulation model to generate benchmark statistics of current system operations. The simulation model is then altered to conform to a potential scenario, and it is rerun to generate comparative statistics. This process is continued cycling through suggested scenarios and generating comparative statistics to allow evaluation of alternative solutions.

5.2.4 Evaluation and Iteration

Utilizing the criteria specified by system decision makers, and utilizing the simulation model's statistical results, alternative scenarios should then be analysed and ranked to carry on an efficient study.

Iteration is required if the simulation has not achieved the objectives of the study. In fact, this procedure helps to determine if any additional data, models, experimentation, or analysis is needed to achieve the modelling objectives.

5.2.5 Implementation

When the simulation satisfies the goals of the study, it is time to document and implement the recommended solutions. Afterwards, the project should be evaluated as to whether or not the proposed solution met the intended objectives.

Chapter 6

Case Study: Automated Bottling Line

This chapter gathers all the topics prior discussed in a case study. The steps followed while developing it are the ones of the *General Methodology for applying Simulation to Problem Solving* presented in chapter 5, integrated with the job sequencing resolution method explained in 2.6. A brief introduction to the methodology carried on in order to solve the case study is shown in the following diagram. It reflects the combination of the use of both simulation modelling - by means of the software AnyLogic - and combinatorial optimization techniques through the implementation of Karg-Thompson's algorithm. This is just an overview to which details will be added along the case study.

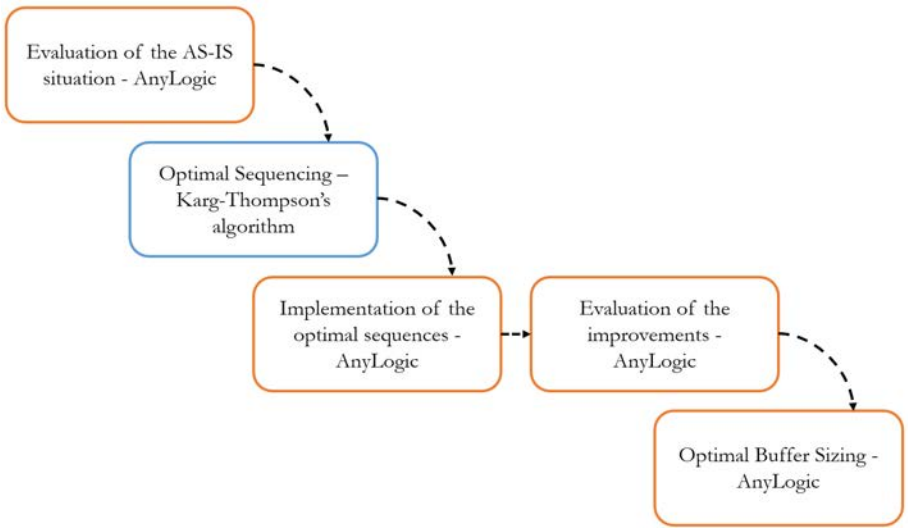


Figure 6.1: Interaction between simulation and combinatorial optimization in the case study

6.1 Problem formulation

6.1.1 Define the problem

The case study aims to improve the Overall Equipment Effectiveness, and therefore the production rate, of an automated bottling line that works different formats and types (returnable and one-way) of bottles of water. The ASME scheme of the line can be seen in the figure below.

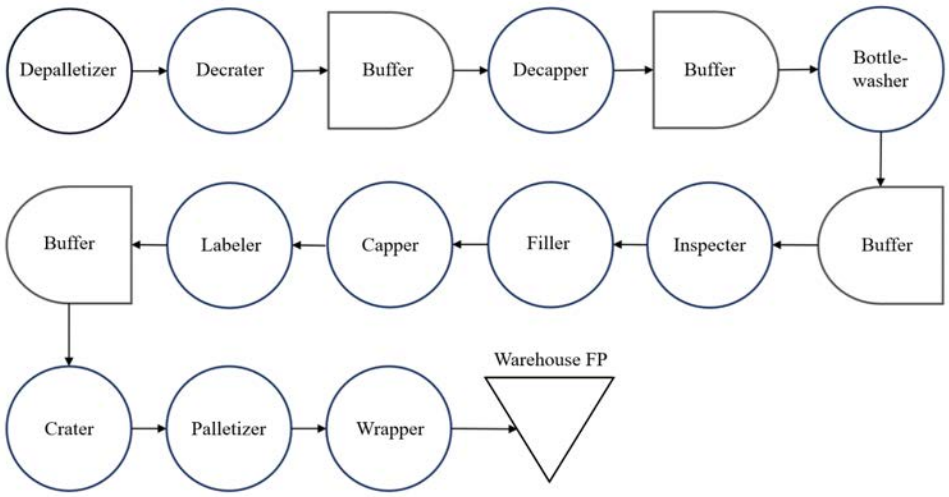


Figure 6.2: ASME scheme of the line

An important measure performance related to the throughput of the line and its efficiency is the OEE (*Overall Equipment Effectiveness*). The higher it is, the higher the production rate results. The factors that affect the line, and so the OEE, are failures and set-ups. The aim of this work is to raise the OEE of the line by solving two problems:

- Job sequencing
- Optimal buffer sizing

The way of doing it involves the development of a simulation model of the critical section of the line and the utilization of a codified heuristic algorithm. An improvement of the bottleneck results in an improvement of the whole line effectiveness.

6.1.2 Define the system

Our system is the critical section of the line, the so-called *bottleneck*. It has been identified by looking at the nominal cycle times of the line's work-stations.

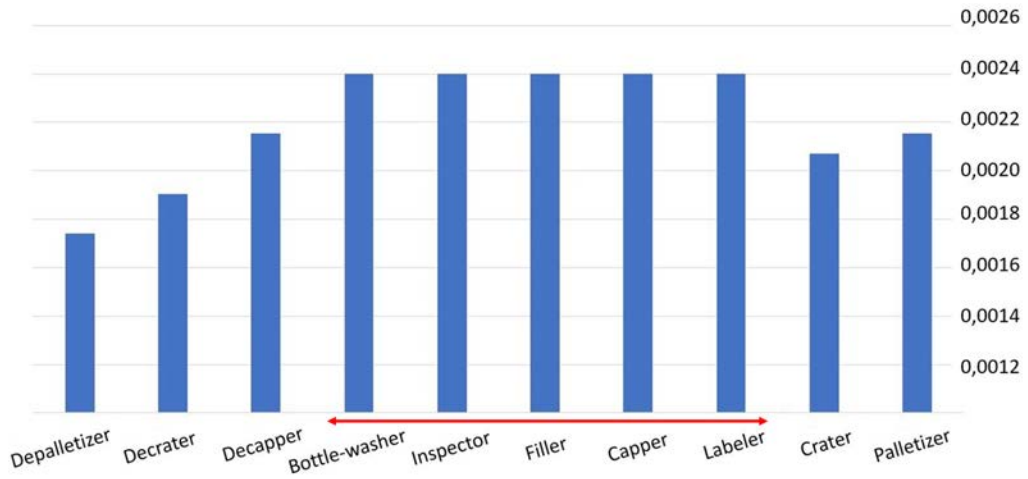


Figure 6.3: Histogram of line's machines nominal cycle time (min/bottle)

Having a look at the histogram, the bottleneck can be identified as the group of machines from the bottle-washer to the labeler. Their cycle time of 0,0024 minutes per bottle makes them the slowest section of the line. Since the nominal production rate of the work-stations included in the critical section is equal and they are all affected by failures, the real bottleneck can be either the bottle-washer or the group of work-stations from the rinser to the labeler. The latter works in sync, so during our analysis, we can consider only the last work-station: the labeler.

Structure

The system is therefore composed of three conveyors - the buffers - and two work-stations; we will refer to them as *elements of the line*. The first two buffers are related to the type of bottle being it of the returnable type or one-way.

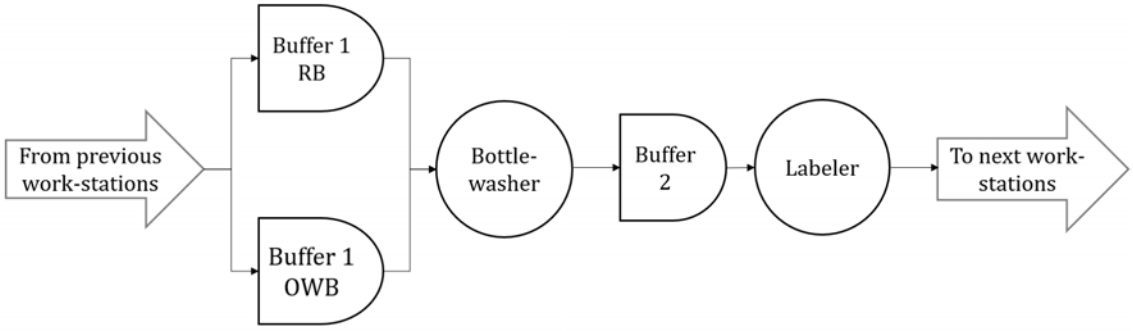


Figure 6.4: ASME scheme of the system

The bottles arrive at the entry point with a rate of 33000 bottles per hour after that they have been subjected to other works. They appear on of the first conveyors (buffer) depending on the type of the bottle being produced: if they are glass bottles the conveyor has a length of 33,5 meters while if it is plastic the conveyor's length is 100 meters. The buffers feed the bottle-washer, a work-station with a production rate of 25000 pieces per hour, followed by a second buffer that makes the bottles flow until a second work-station. The second buffer of the section is long 8 meters. The width of the buffers is 0,6 meters.

Down-times

Both the work-stations can be the real bottleneck since their availability is strictly influenced by the micro down-times. The output of this piece of line, that corresponds with the exit from our system, is represented by the bottles with the stuck labels. The main problem of this bottleneck is the capacity of the second buffers (directly proportional to its length) considered as not enough to let the flow of bottles continuous even when a micro-downtime occurs, that it should be its scope.
The failures can affect either the conveyors and the work-stations, their causes are many and each element might have different problems. They happen after a certain time (TTF) and then it takes a certain amount of time to the machine (or conveyors) to restart working (TTR). Once a failure occurs the elements of the line affected by it stops to work, so TTF can also be seen as up-time and TTR as down-time. In our study, the second ones are considered micro down-times because the time that the machine stalls is lower than 5 minutes.

The table below contains the elements of the line with their failures and their symbols.

	Failure	***Symbol***
Buffer 1	Lateral stuck	A
	Fallen bottle	C
Bottle-washer	Stuck load	A
	Stuck unload	D
	Out of sync	E
Buffer 2	Bottles block	B
Labeler	Unstuck labels	A
	Wrong positioning	B

Figure 6.5: Line elements with their failures

Set-ups

In addition, the entire line is also stopped by set-ups: one that happens for a change of the format of the bottles and the other one it occurs once a week for predictive maintenance. The setups for changeover influence the performances of the line too. In fact, the line is able to produce 13 different formats and their total setup time is sequence-dependent. It means that the order the formats are processed straight affects the efficiency of the line because the time to switch from a certain format i to j may differ from the time to switch from j to i. Therefore, the current sequence of jobs (formats) might not be the optimal one and make the system sub-performing.

6.1.3 Inputs of the system

The inputs given to build the model, analyse the system and improve its performances, are:

- ASME scheme of the line taken into the study. ASME (American Society of Mechanical Engineers) is a symbolism used to represent production systems. The ASME scheme is depicted in figure 6.2;
- Rate of bottles that enter our system per hour (33000 bottles per hour). They arrive from the previous section of the line;
- Product mix of a generic x month;
- Weekly product mix of the generic x month. Every day are processed n lots of the same and/or different formats of bottles;
- Nominal production rate of the work-stations (figure 6.3). It is expressed in pieces per minute;
- Length and width of the buffers (6.1.2);
- Data sets of Time to Failure and Time to Repair of the buffers and the work-stations for each kind of failure. They are present in the appendix at the end of the thesis;
- Set-up matrix for format changes;
- Weekly sequence of the jobs (format lots) being processed;
- Average weekly values of the Overall Equipment Effectiveness of the as-is situation of the four weeks. It is about 63/64 % the first and the fourth week, whereas it is about 55/57 % the second and the third week. To be useful the model needs to match as much as possible the current OEE values of the line;
- Shift timetable. During a year, the line works 24 hours a day for 8 months and 16 hours a week during the rest months.

The following sections will present some of these inputs in a more detailed manner.

Product mix

The overall product mix is shown through a table and a pie chart. It represents the product mix of the month x. The table indicates also the capacity of the formats expressed in litres; the bottle's diameter depends on the capacity of the bottle.

Letter	*Format (L)*	*Type*	*Mix*
A	0,25	One-way	5,06%
B	0,25	Returnable	0,00%
C	0,5	One-way	3,63%
D	0,5	Returnable	18,30%
E	0,75	One-way	12,10%
F	0,75	Returnable	20,50%
G	1	One-way	6,97%
H	1	Returnable	26,78%
I	0,65	One-way	1,46%
J	0,65	Returnable	3,49%
K	1	One-way	0,26%
L	1	One-way	0,47%
M	1	One-way	0,97%

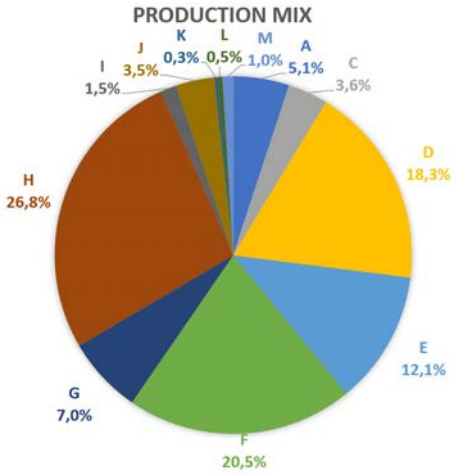


Figure 6.6: Monthly product mix

The data state that the predominant formats are D, E, F, G and H. Together they cover almost the 78 % of the total production. H, letter that stands for the glass bottle of 1 litre, is the format more produced with a percentage up to 26,8 %, more than a quarter of the total production.

Job sequence and weekly product mix

Every week the different lots are worked on a different given order, called *sequence*. The sequences of the four weeks are:

1. HHHEEEEEFDDDIDIIIIJJJJJDCCCCC (29 lots)
2. CDDFDDDDDFFFFHHHGGGGGGGGGHHHGGEFHHEEEEEEHEE (43 lots)
3. FFFEFEDDFDDDHHCCHGHGGGGFHHHHHEFFEE (34 lots)
4. EFFEEEEDDDDDHHAAAHKLHMKLKKKF (28 lots)

The tables there contain the product mix of the weeks (weekly demand) of the x month. This information is required to set up the model and as a comparison in order to verify its validity.

Week 1		Week 2		Week 3		Week 4	
Format	*Mix*	*Format*	*Mix*	*Format*	*Mix*	*Format*	*Mix*
C	17,86%	C	0,85%	C	0,49%	A	21,01%
D	22,92%	D	14,66%	D	15,85%	D	22,75%
E	11,50%	E	9,91%	E	14,94%	E	12,25%
F	4,64%	F	29,61%	F	27,49%	F	14,57%
H	15,86%	G	16,26%	G	8,48%	H	22,34%
I	8,04%	H	28,71%	H	32,75%	K	1,08%
J	19,18%					L	1,96%
						M	4,03%

Figure 6.7: Weekly product mixes

Set-up matrix for format changes

The set-up matrix contains all the theoretical times to change the format. It's the time to clean, prepare the tools, arrange the machines to work the next format. This time is better considered as a *changeover* time, meant by the specific time it takes to go from the last good part of one product run to the first good part of the next product run. Anyway, the two terms can be used also interchangeably.

	Successor												
Predecessor	A	B	D	F	H	E	C	G	K	L	M	J	I
A	20	60	220	220	220	240	240	240	235	235	235	220	260
B	60	20	240	240	240	220	220	220	255	255	255	240	220
D	220	240	20	210	210	210	60	230	225	225	225	210	210
F	220	240	210	20	160	60	210	180	175	175	175	240	220
H	220	240	210	160	20	160	220	60	35	35	145	240	220
E	240	220	190	60	195	20	210	180	175	175	175	220	240
C	240	220	70	210	210	210	10	230	225	225	225	210	210
G	240	220	190	160	60	180	210	20	35	35	145	220	240
K	290	290	200	170	110	230	260	150	35	35	215	310	310
L	310	310	280	250	110	280	280	110	35	35	145	310	290
M	290	290	280	230	230	230	260	230	145	145	0	290	290
J	220	240	210	240	240	220	230	220	255	255	255	10	90
I	220	240	210	240	240	240	230	240	255	255	255	60	0

Figure 6.8: Set-up matrix for change format

It is possible to notice that the matrix is asymmetrical, that is the time required to switch from a format i to a format j may be different from switching from j to i.

6.1.4 Establish performance metrics

The situation of the system can be evaluated by different types of performance metrics and they are generally listed under the five headings of quality, speed, dependability, flexibility and cost.

- Throughput: job exiting from the production line per unit time;
- Overall Equipment Effectiveness (OEE) that reflects the six major losses based on its Availability, Performance and the Quality rate of the output;
- Productivity: production rate in terms of bottles processed per hour.

As seen in chapter 1, the most practice way to calculate the Overall Equipment Effectiveness value for the entire plant (or production line) implies the using of the following presented formula. This will be used to calculate the OEE of the critical section of the line during our simulation study. The elements of the formula are prior explained in chapter 1.

$$OEE_{LINE} = \frac{Production\,rate}{Nominal\,production\,rate} = \frac{\frac{Throughput}{Production\,Time}}{25000} \tag{6.1}$$

6.1.5 Build conceptual model

To depict our system, we use an activity diagram where each shape represents something different. Zigzag lines indicate the creation or destruction of entities. The queues are shown as a circle and stand for the buffers of the line, instead of the activities shown as rectangles.

The resources that interact with the agents, in our case the working machines that work on them, are represented by small circles.

Lines/arcs indicate flows (precedence ordering) for engagement of entities in activities or for obtaining resources. Dotted lines are used to indicate the seizing and releasing of resources.

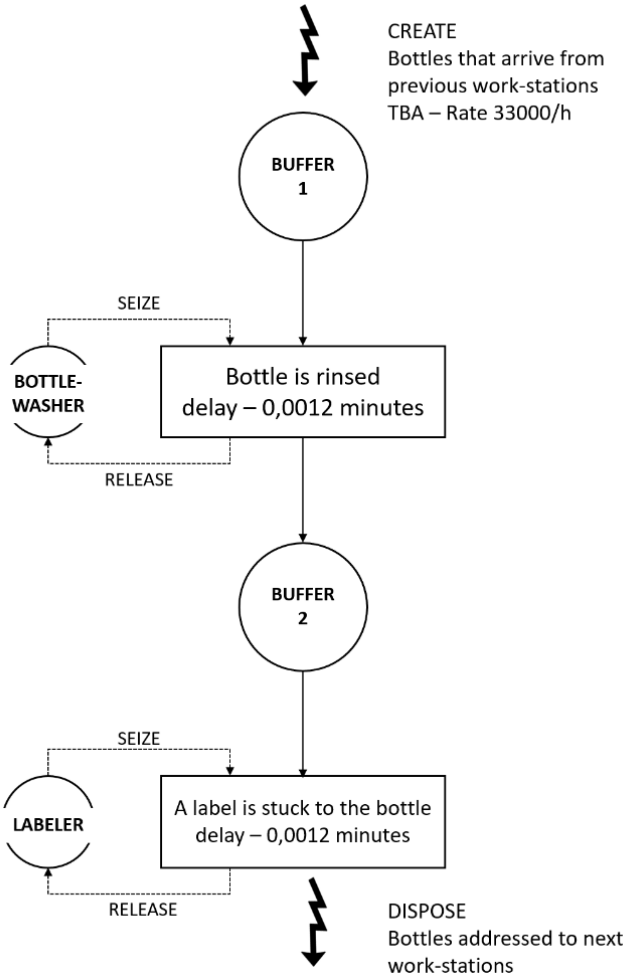


Figure 6.9: Activity diagram of the system

6.1.6 Document model assumptions

The model requires some assumptions due to needed works to do on the data or to the level of detail desired, that could not require to represent every aspect of the real line. The assumption made to build the model are the following:

1. the layout of the line in the model does not follow the real one;
2. the time spent by the agent on the conveyor is the same as its next work-station's cycle time;
3. the speed of the conveyors has been calculated using the data about the cycle time of the line we had as input and the length of the conveyors;
4. to represent the grouping of the bottles when they flow on the real line, we use a statistical approach to determine the batch size;
5. the diameter of 1 L bottle is about 9 centimetres. The diameter of the bottle of the other formats is calculated with proportions. Bottles of the same format have the same diameter regardless they are glass or plastic;
6. conveyors and work-stations' repair and failure times are randomly distributed;
7. the time required to change format is taken from the setup matrix. It is assumed that this time may vary plus or less 10% depending on the conditions, tiredness and knowledge of the workers;
8. for weeks 2, 3 and 4 it is present in the sequence also the last lot worked in the previous week. It is indeed assumed that when a new week begins, the line is still arranged as it was to work the last format of the previous week;
9. workers are not considered in the analysis;
10. there are no physical constraints to consider when performing the optimal buffer sizing;
11. the time of a model run is set to 10080 minutes, equal to a week of work.

6.2 Simulation Model Building

As stated in the *General simulation methodology for applying simulation to problem solving*, the phases of model building and data collection are strictly linked in order to realize a model as closest as possible to the real system (Rossetti, 2015). In fact, the model building has to depict the best trade-off between all the inputs that affect the study and the investigated level of detail. Since our study aims to improve the productivity on the line, a focus on the efficiency of the model rather than the graphical aspect is preferred. There are many elements and variables that interact between them in the model, therefore when building it, it is important to have a look also at the general set while working on the data preparation of the singular one. For this purpose, assumptions and calculations are made and repeatedly modified, moving up and down through the phases of the iterative process proposed by the methodology. The distributions for the TTF and TTR have been figured out with the use of the statistics. The information about the physics element of the line are important to make the line behaviour as realistic as possible, just like cycle times and buffer lengths.

The approach used for this phase can be seen as a data funnel: the inputs of the case study enter the funnel, where they are worked to exit it as input for the simulation model. The funnel reflects the phases of Input Data Preparation and Model Translation, as represented in the image below.

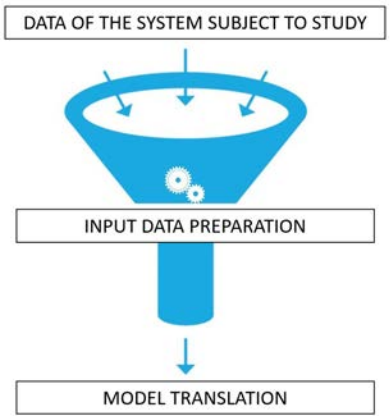


Figure 6.10: Funnel of the work done on the data

6.2.1 Input data preparation

This phase consists in starting from the input data and working them to use them as inputs for the on-building model. To reach this aim, various techniques are used. Statistical approaches and methods are needed to develop the proper distributions for Times to Fail and Times to Repair of both the work-stations and the buffers. The Cumulative Distribution Function helps to find out a suitable batch size for representing the flow of the bottles in the model, combined with logical considerations. Information about the physical elements of the line as work-stations and buffers are to be collected and managed with their production speed and lengths in order to be useful for the model.

Fit the availability parameters with a probability distribution

To achieve randomness in the simulation we need to find the proper probability distribution for the data samples regarding the Times to Failure and Times to Repair of the different causes of down-times of the line. This work is very important because the quality of the data used in a simulation study is vital for the validity of the result. That is the reason for which a thorough analysis of the data with the software Minitab is needed. The result of the analysis is the set of distributions to put into the software AnyLogic to manage the failures.

The first thing to do when analysing some data is to create a histogram and collect descriptive statistics. The histogram can show the frequency distribution of the data and then find out an appropriate distribution with its parameters at first sight. The next step brings to the use of the Anderson Darling *Goodness of Fit test* to investigate which probability distribution would fit the most with the data samples. The kinds of distributions taken into consideration for the test are the continuous distributions because they can be used to situations where the set of possible values occurs in an interval or set of intervals. Furthermore, within discrete-event simulation, they are often used for modelling time to perform a task (Rossetti, 2015).

The method used to investigate the distributions for Times to Failure and Times to Repair is the same. It is composed of due steps:

1. Create a histogram from the data sample;
2. Identify the probability distribution that fits the most the data through the Anderson-Darling test.

The Anderson-Darling test

The Anderson-Darling test measures how well the data follow a particular distribution. For a specific data set and distribution, the better the distribution fits the data, the smaller this statistic will be. It is defined as:

- H_0: the data follow a specific distribution;
- H_a: the data do not follow the specific distribution;
- Test Statistic: $A^2 = -N - S$.

Where: $S = \sum_{i=1}^{N} \frac{(2I-1)}{N}[lnF(Y_i) + ln(1 - F(Y_N + 1 - i))]$
F is the cumulative distribution function of the specified distribution. Note that Y_i are the ordered data.

The Anderson-Darling statistic (A^2) measures the area of the expected model (based on the chosen distribution) and the empirical distribution function. More precisely, it is a squared distance that will have a greater weight in the tails of the distribution. Low values of the Anderson-Darling statistic mean that the hypothesized distribution fits the data well.

Use the corresponding p-value (when available) to test if the data come from the chosen distribution. If the p-value is less than a chosen α (usually 0.05 or 0.10), then reject the null hypothesis that the data come from that distribution.
Thus, to choose the right distribution, it is needed to look in order at:

- AD value - The less it is, the better the distribution is;

- p-value - It should be >0.05 to make the distribution be considerable. The p-value is also used to choose the right distribution when the AD values of two alternative distributions are very close. In these cases, it is picked the distribution with a higher p-value.

In our case, the Anderson-Darling test is made for the first reason for the failure of the bottle-washer, denoted with the symbol "A". Before it, a histogram for the Times to Failure of the failure A of the work-station bottle-washer is created. This failure is caused by a lateral stuck of the bottles inside the machine.
The histogram displays statistical information with rectangles to show the frequency of data items in successive numerical intervals of equal size. The Times to Failure expressed in minutes that stay in the same interval class are grouped together and it, therefore, allows to identify the possible distributions for the sample. They are then checked with the Anderson-Darling Test for the data sample.

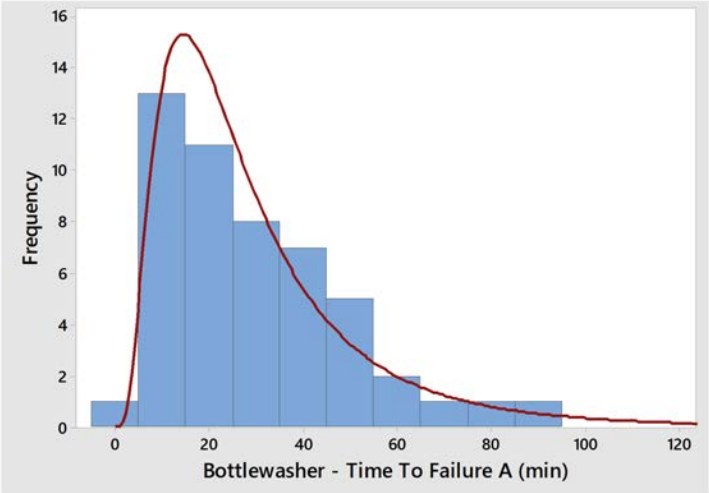


Figure 6.11: Histogram for the TTF A of the bottle-washer

It is followed by some statistical information about the data sample.

Bottle – washer "Lateral Stuck" TTF: Descriptive statistics

N	Mean	StDev	Median	Minimum	Maximum	Skewness	Kurtosis
50	29,6482	19,45	24,9622	4,7915	86,2514	1,07299	0,08036

Figure 6.12: Descriptive statistics for Bottle-washer Time to Failure A

The Anderson-Darling Goodness of Fit test shows that the Lognormal distribution fits the best the data sample for the Time to Failure of the failure A of the bottle-washer. This because its AD value of 0,246 is the lowest from the results of the test. Afterwards, the parameters of the chosen distribution are pointed out and they will be the input of model for the Time to Failure related to the failure A of the work-station bottle-washer.

Goodness of Fit Test		
Distribution	*AD value*	*P value*
Normal	1,284	<0.005
Box-Cox Transformation	0,246	0,747
Lognormal	0,246	0,747
3-Parameter Lognormal	0,221	*
Exponential	2,962	<0.003
2-Parameter Exponential	1,018	0,088
Weibull	0,308	>0.250
3-Parameter Weibull	0,176	>0.500
Smallest Extreme Value	2,801	<0.010
Largest Extreme Value	0,471	0,24
Gamma	0,217	>0.250
3-Parameter Gamma	0,18	*
Logistic	0,999	0,005
Loglogistic	0,299	>0.250
3-Parameter Loglogistic	0,304	*
Johnson Transformation	0,138	0,974

Figure 6.13: Anderson-Darling test for TTF A

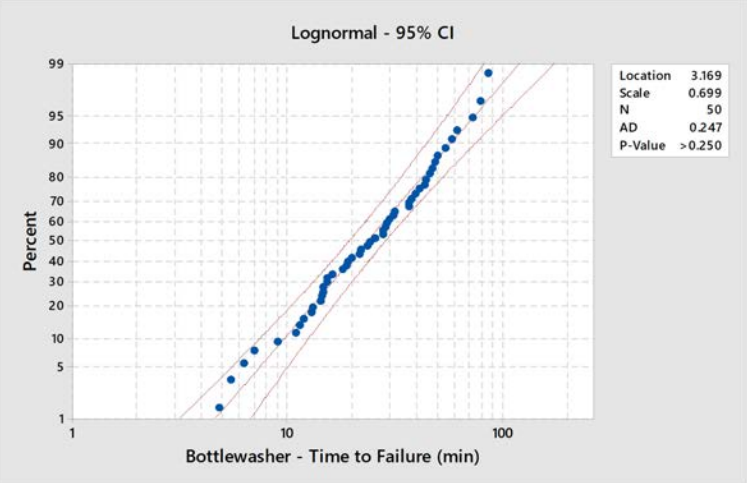


Figure 6.14: Probability plot for TTF A

Estimates of Distribution Parameters

Distribution	*Location*	*Shape*	*Scale*
Normal*	29,64824		19,44999
Box-Cox Transformation*	3,1694		0,69925
Lognormal*	3,169		0,699
3-Parameter Lognormal	3,28177		0,61701
Exponential			29,64824
2-Parameter Exponential			25,36401
Weibull		1,63066	33,2963
3-Parameter Weibull		1,26575	27,01085
Smallest Extreme Value	40,12026		22,78402
Largest Extreme Value	21,06857		13,96618
Gamma		2,42638	12,21914
3-Parameter Gamma		2,09595	13,39632
Logistic	27,43989		10,693
Loglogistic	3,19095		0,40372
3-Parameter Loglogistic	3,16665		0,41477
Johnson Transformation*	-0,01188		0,96352

* Scale: Adjusted ML estimate

Figure 6.15: Distribution parameters for TTF A

The steps are repeated for all the availability parameters. The results of the analysis and the chosen distributions are shown in the tables below.

Machine	*Symbol*	*Micro – downtime*	*TTF Distribution*	*Location*	*σ*	*Shape α*	*Scale β*
Buffer 1	A	Lateral stuck	Weibull			1,224	46,924
	C	Fallen bottle on FT	Lognormal	3,765	0,375		
Bottle-washer	A	Stuck load	Lognormal	3,169	0,699		
	D	Stuck unload	Weibull			1,252	38,541
	E	Out of sync machine	Lognormal	4,044	0,420		
Buffer 2	B	Bottles block	Weibull			1,127	53,908
Labeler	A	Unstuck labels	Weibull			0,996	58,335
	B	Wrong positioning	Lognormal	3,159	1,468		

Machine	*Symbol*	*Micro – downtime*	*TTR Distribution*	*Location*	*σ*	*Shape α*	*Scale β*
Buffer 1	A	Lateral stuck	Lognormal	0,557	0,459		
	C	Fallen bottle on FT	Weibull			2,568	2,454
Bottle-washer	A	Stuck load	Lognormal	0,403	0,547		
	D	Stuck unload	Lognormal	0,662	0,325		
	E	Out of sync machine	Normal	2,006	0,833		
Buffer 2	B	Bottles block	Normal	1,399	0,340		
Labeler	A	Unstuck labels	Lognormal	0,628	0,544		
	B	Wrong positioning	Weibull			2,852	3,416

Figure 6.16: Fitted distributions of the Times To Failure and Times To Repair

Set-ups

The predictive maintenance happens once a week and it lasts 90 minutes. Therefore, it has been modelled in the following way:

- rate: event that happens once during the simulation;
- duration time: 90 minutes.

The changeover time is modelled in the same way, even though the frequency is higher, and it depends on the number of lots in the sequence.

- Rate: number of lots processed during the week
- Duration time: triangular(min, max, mode) depending on the total set-up time of the sequence. *Min* and *max* are the minus and plus 10% of the set-up time due to the variability related to the worker (assumption 6)

The total set-up time for format changes of each week is calculated by considering the set-up matrix and the current sequences seen in 6.1.3. In the following table *frequency* is equal to the number of lots in the sequence.

Week	***Current sequence***	***Setup (min)***	***Frequency***
1	HHHEEEEEFDDDIDIIIIJJJJJDCCCCC	1650	29
2	(C)CDDFDDDDDFFFFHHHGGGGGGGGGHHHGGEFHHEEEEEEHEE	2525	44
3	(E)FFFEFEDDFDDDHHCCHGHGGGGFHHHHHEFFEE	2580	35
4	(E)EFFEEEEDDDDDHHAAAHKLHMKLKKKF	2740	29

Figure 6.17: Weekly sequences and setup times

Working Time is calculated as *10080 - Total setup time*; 10080 minutes is equal to the total amount of available time in one week. The other terms in the table are:

- *Time to Setup = Working Time / Frequency*
- $Setup\ Time_{MEAN}$ = *Total Setup / Frequency* where *Total Setup* is the total setup time of the given sequence according to the setup matrix

- $Setup\ Time_{MIN} = Setup\ Time_{MEAN}$ minus 10%
- $Setup\ Time_{MAX} = Setup\ Time_{MEAN}$ plus 10%

Calculations		***Model setup time (min)***		
Working time (min)	*Time to Setup* (min)	*Mean*	−10%	+ 10%
8430	290,7	56,9	51,2	62,6
7555	171,7	57,4	51,6	63,1
7500	214,3	73,7	66,3	81,1
7340	253,1	94,5	85,0	103,9

Figure 6.18: Modelling the setup times for format changes

Batch size

In automated bottling lines, the bottles are not worked one by one, but they flow together on the conveyors and they can enter the work-station grouped as well. This aspect has been subject to a particular analysis. It is indeed to be decided which size of the group, referred to as *batch size*, would fit the most the similarity with the real line. It is also needed a trade-off between the real line, without distorting its normal functioning, and a statistical explanation of the choice.

In the first place, it is assumed that 50 bottles as one agent might be a reasonable number for the desired batch size. The analysis carries out some calculations: the input of the line is 33000 bottles per hour, equivalent to 550 bottles per minute. Thus, 50 bottles are generated in 0,09 minute. The aim is to confirm that the batch size of 50 bottles is reasonable with the buffer capacity and that guarantees that the likelihood of finding a failure within a time span of 0,09 minute is less than 1%. The latter is the hypothesis of the analysis.

The tool used to evaluate the hypothesis is the *Cumulative Distribution Function*. Indeed, the Cumulative Distribution Function (CDF) of a distribution function of a real-valued random variable X is the function given by:

$$F_x(X) = P(X \leq x)$$

where the right-hand side represents the likelihood that the random variable X takes on a value less than or equal to x. In the case subject to study:

- X is the probability distribution for the Times To Failure;
- x is equal to 0,09 minutes, the rate of arrival of 50 bottles.

In the diagram below, we can see that for the bottle-washer there is a likelihood of 1% to find a failure *stuck load* before 2,72 minutes. This amount of time is bigger than 0,09 minutes, therefore the likelihood of finding a failure in 0,09 is less than 1% and our hypothesis is valid for this Time To Failure. If it happens the same for all the Times To Failure then our hypothesis is confirmed at all and we can use a batch size of 50 bottles.

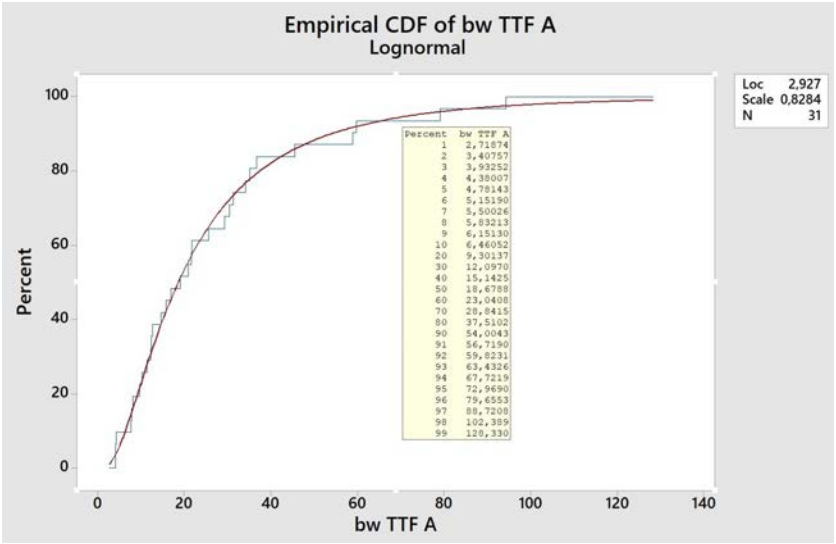


Figure 6.19: Plot of the *Cumulative Distribution Function* of TTF A

The results of the analysis have been collected in the following table. They confirm that the batch size of 50 bottles, decided to use to have an aggregation that would have been reasonable with the system properties, is valid also from a statistical point of view. Afterwards, the cycle time of the elements of the line becomes:

$$Cycle\,time = \frac{1}{25000/60} \times Batch\,Size = 0,12\,[minutes/batch] \tag{6.2}$$

	Failure	Distribution for X	$P\{X \leq 0,18\}$	Hypothesis confirmed?
Buffer 1	Lateral stuck	Lognormal(0,49; 2,87)	0 %	Yes
	Fallen bottle	Weibull(1,93; 19,29)	0,0121 %	Yes
Bottle-washer	Stuck load	Triangular(34,95; 28,97, 72;38)	0 %	Yes
	Stuck unload	Gamma(1,68; 25,96)	0,00155 %	Yes
	Out of sync machine	Lognormal(0,36; 3,60)	0 %	Yes
Buffer 2	Bottles block	Gamma(1,88; 26,42)	0,0047 %	Yes
Labeler	Unstuck labels	Exponential(2,1; 44,39)	0,4047 %	Yes
	Wrong positioning	Lognormal(1,34; 3,12)	0,0154 %	Yes

Figure 6.20: Results of the CDF test

Given the batch, one agent will represent a number of bottles equal to the batch size. In the model one agent will represent 50 bottles; then the output may be calculated either expressed in batches or bottles.

The rate of bottles that enter the system is 33000 bottles/hour so the time of arrival of one agent is set as:

$$Interarrival\,time = \frac{1}{33000/60} \times Batch\,Size = 0,091\,[minutes/batch] \tag{6.3}$$

This value is inserted in the AnyLogic's block *Source*.

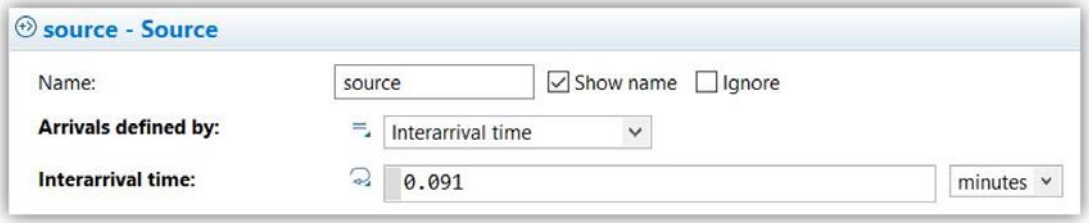


Figure 6.21: Source's block pane in AnyLogic

Conveyors' attributes

AnyLogic has a proper *Conveyor* block to set all the parameters related to this item. The cycle time of the conveyor is meant as the time spent by one agent to move on the entire length of the conveyor in a situation of normal functioning. The formula used is:

$$Speed_{CONVEYOR} = \frac{Conveyor\,Length}{Cycle\,time} \tag{6.4}$$

The second aspect to be considered is the capacity of one buffer. This because in the AnyLogic model, like in the real cases, the capacity of the buffer is directly proportional to its length: the more the buffer is long and the bigger is the number of bottles that the buffer can have on it (its capacity). For this reason, every batch length that is added increases the buffer capacity of 50 bottles. To calculate the length of the batch it has been used the assumption made on the diameter of a one-litre bottle: 0,089 meters. The diameter of the bottles of the other formats is calculated proportionally. The procedure to achieve the length of the batch is composed of the following steps:

1. consider the width of the conveyor of 0,6 meters and the diameter of one bottle in order to calculate how many bottles stay in a row;
2. calculate the number of rows that form a batch dividing 50 by the number of bottles in a row;
3. the length of one batch is obtained from the multiplication of the diameter of one bottle and the number of rows that form a batch.

Format (L)	***Diameter (m)***	***[Bottles in a row]***	***[Rows]***	***Batch length (m)***
0,25	0,022	26	2	0,045
0,5	0,045	13	4	0,178
0,65	0,058	10	5	0,289
0,75	0,067	8	7	0,467
1	0,089	6	9	0,801

Figure 6.22: Procedure to calculate the length of the batches

The batch's length corresponds with the *Agent length* in the AnyLogic's *Conveyor* pane. The capacity of the buffers can be calculated by considering its length, the length of the batch and the batch size of 50:

$$Capacity_{BUFFER} = \frac{Conveyor\ Length}{Agent\ Length} \times Batch\ Size\ [meters] \tag{6.5}$$

The next figure shows the properties pane of the block that manage the conveyor in the software.

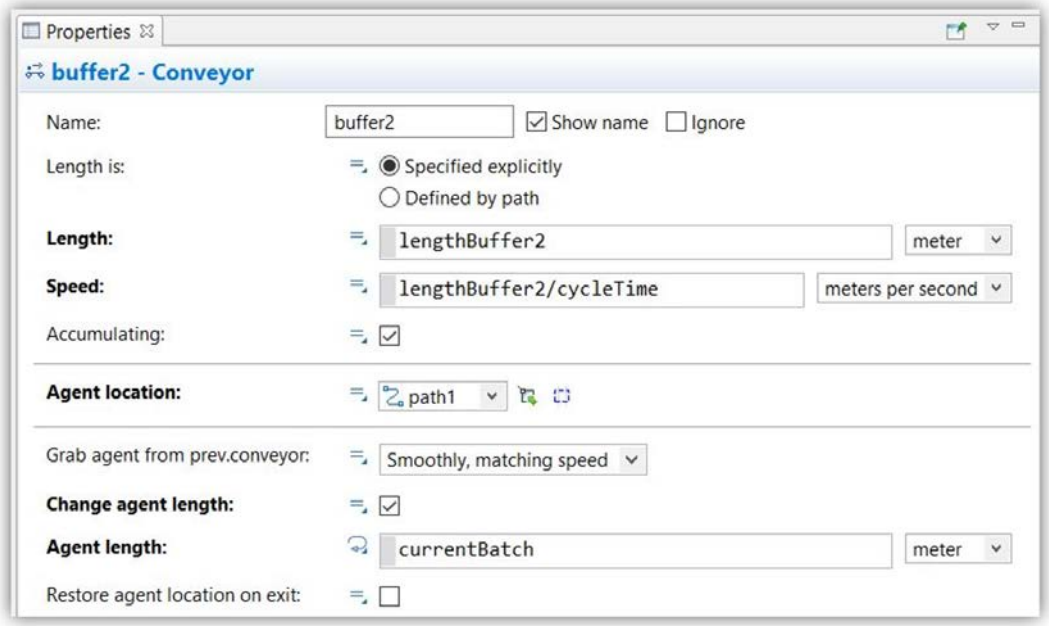


Figure 6.23: Conveyor's block pane in AnyLogic

6.2.2 Model's input recap

After the input data preparation phase, the data that are going to be inserted into the model as input are:

- Rate of arrival: 0,091 minutes/batch equal to 0,00182 min/bottle
- Batch size: 50 bottles
- Agent length = length of the current batch
- List of the different formats with their batch length
- Work-stations and conveyors cycle time: 0,12 minutes/batch equal to 0,0024 min/bottle
- Probability distributions for machines and conveyors' failures and repairs
- Set-up times
- Length of the conveyors
- Simulation time: 10080 minutes.

6.2.3 Model translation

This phase entails the description of the procedure carried out to code the model, in other words, to represent the real system with the software AnyLogic. The aspects explained in this section regard the translation of the conceptual model into computer simulation program representations.

The model aims to represent the critical section of the bottling line of our case study. It is composed of three main parts, that will be introduced and explained in the following sections: graphics, *flowchart* and *statecharts*. A fourth section is added in order to explain how the format changes are modelled.

Graphics

The software AnyLogic allows the model to have different graphical levels of detail, linked with the aimed level of detail of the study. In this case study, it is not that important the graphical issue since the main purpose is to increase the performance of the system and it can be done also without a rich design. For this reason, just few elements of the *Space Markup* palette have been used to depict the critical section of our bottling line in an easy way and a *3D object* as a bottle to stand for the batch of bottles. The buffers are therefore represented with *paths* and the work-stations with *rectangular nodes*.

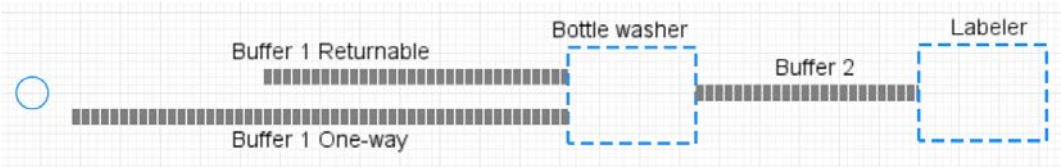


Figure 6.24: Critical section of the bottling line in the model

Flowchart of the system

The simulation model works following the logic created by the user, called *flowchart*. This is made with the blocks of the Process Modelling Library.

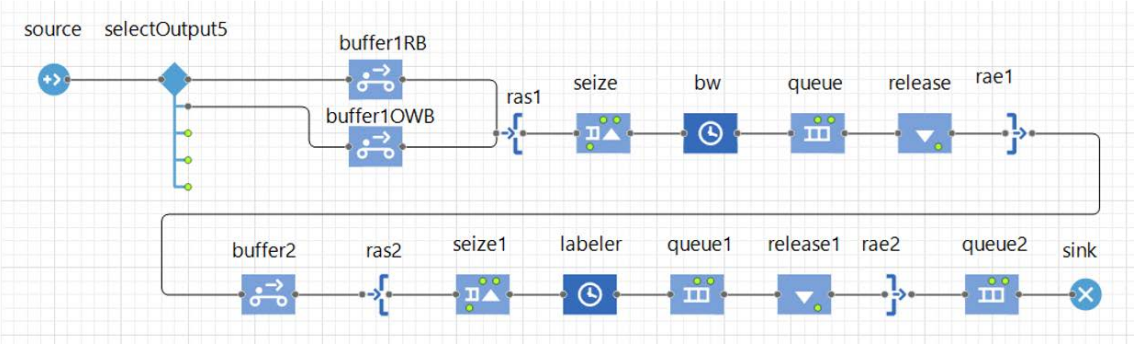


Figure 6.25: Flowchart of the model

The agents are generated in the block *source* with an arrival rate of 33000 bottles/hour, converted into 0,091 minutes/batch. The *selectOutput5* directs the agents towards the right buffer: *buffer1RB* if the bottles are returnable; *buffer1OWB* if they are one-way bottles. The agent is moved along the first path by one of this *Conveyor* block, to reach the first work-station. The work-stations are inserted in the model as resources initialized by the *Resource Pool* block.

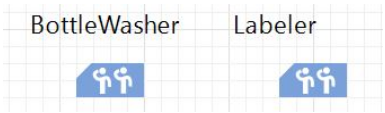


Figure 6.26: Work-stations

When the agent arrives at the work-station, it enters the *restricted area* and then it is seized by the resource. It reaches the *delay* block and it waits there for a delay time that corresponds with the cycle time of the machine, set at 0,12 minutes/batch. When the operation is completed, the agent is released, and it exits the *restricted area* to proceed along the line. It flows along the second buffer until the work-station labeler is reached. The functioning of the operation is the same as the previous station. Once the operation of labelling is completed as well, the agent exits the system through the block *exit*. In the real line, it will go to further work-

stations but the system of the case under analysis finishes here. In conditions of normal functioning, the agents flow along the line with a nominal speed of 0,12 minutes/batch, whether they are on buffers or work-stations. This condition is not always present since these elements are subject to failures that stop their functioning for a certain amount of time until they are repaired, or because of some action of set-up due to a change in the format that is going to be produced or to predictive maintenance. These stops are operated through the use of *statecharts*. The way through which they have been managed is discussed in the next paragraphs.

Managing the failures with statecharts

A thorough analysis of the data sample of the Times to Failure and Times to Repair has been carried out since this is a critical aspect of the system. Failures influence the system and decrease the performance of the line. More often the failures occur, more time the elements of the line subject to failure are stopped and the production output rate decreases. The bottleneck is characterized by many causes of failure and some elements are subject to more than one cause of failure too, as seen in 6.5. For example, the bottle-washer can be stopped by either a stuck while loading the bottles into the machine or while unloading them after the operation of rinsing, but also due to a problem of synchronization. The labeler instead can be stopped for a failure due to a wrong positioning or a label that is not well stuck on the bottle.

For these underlined reasons, the model presents a statechart for each kind of failure, for each work-stations and buffer. The set-ups are managed with statecharts too. Moreover, have been added functions, parameters and variables; they will be explained moving on.

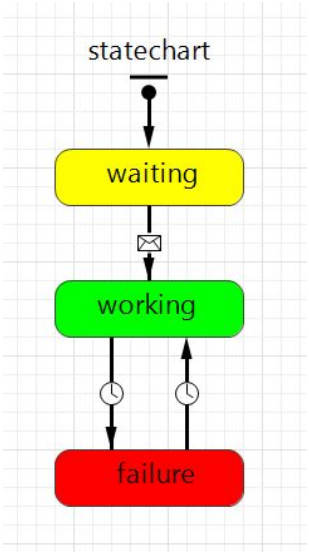


Figure 6.27: Statechart of a failure with its states

In our case, the statechart is basically composed of three different states. The first one, *waiting*, is active before the system starts. When the first agent is created, a trigger activates the *working* one and the machine (or the buffer) starts to run. After the Time to Failure, the state *failure* is triggered and a code *suspend()* (or *stop()* for the buffer) stops the machine. A new code *resume()* acts when the Time to Repair is over and the machine starts to work again.

The one just discussed is the simple case when an element of the line is subject to only one failure or stop. This situation happens with the second buffer, for which it has been observed only one cause of failure, due to a bottles block during their flowing on the conveyor. For the rest of the elements of the system, at least two causes of failure have been detected: two for the first buffers and the labeler, three for the bottle-washer. These sets of failures are managed with more statecharts in a single agent diagram and some codes and functions either in the states and in the transitions. The way to manage more failures set in our simulation environment follows the logic for which every time a failure occurs, it is verified whether there are other active failures or not. In the latter case, the failure just occurred is the dominant one and it will manage the *resume()*; otherwise, there is a comparison between the Time to Repair of the failure occurred and the remaining repair time of the other active failure (failures). The biggest amount of time states the dominant

failure, that will be the one responsible for the *resume()* of the machine (buffer). This logic is shown in the flowchart below, where there are two random failures (A, B), with their Time to Repair and Time to Fail (TTF_A, TTR_A, TTR_B), that acts on a machine.

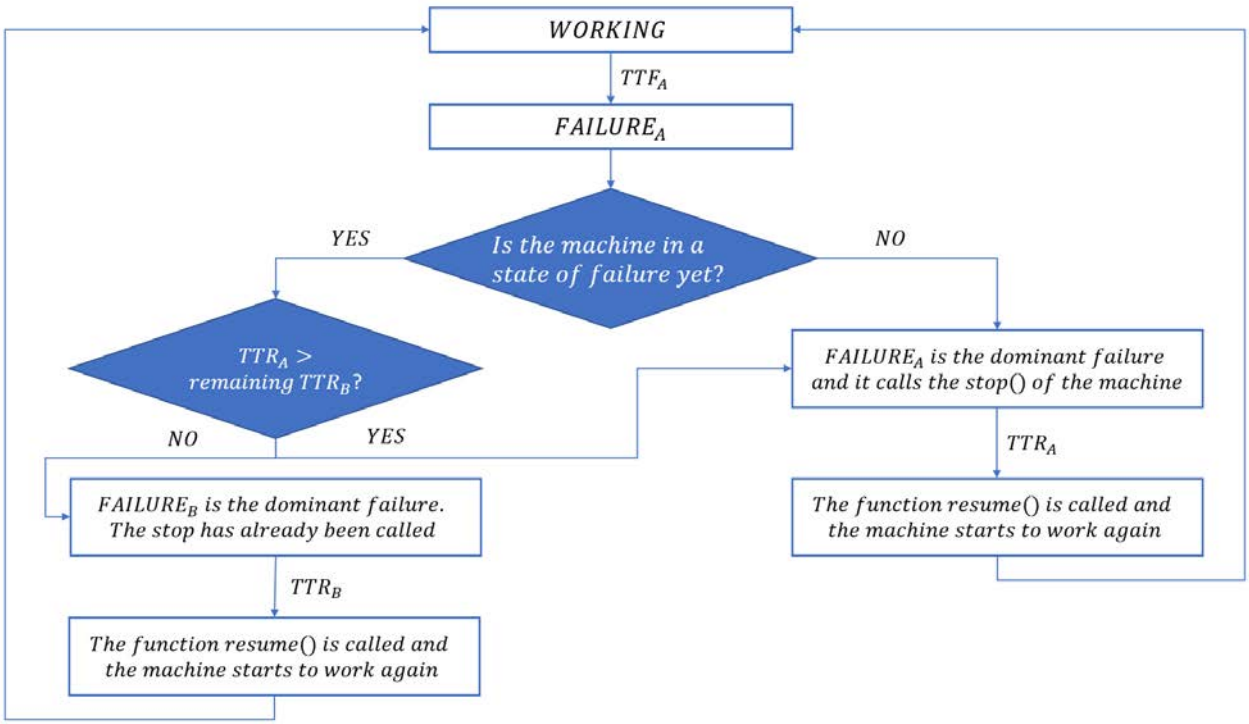


Figure 6.28: Flowchart of the operation of the statecharts

As stated before, the computational work implies functions and parameters, that are key elements in the creation of a model with the software AnyLogic. They are written in a Java language and they are needed to operate all the logic explained before about the failures. Moreover, codes are also written inside the transitions and the states. The functions that we have used are:

- **updateTTR**. When called, it swaps the Time to Repair presents in the collection *ttr* with the remaining time until the repair is over through another function *restTime*. The collection *ttr* contains the values of the TTR of the failures: 0 if the failure is not active, a certain amount of time otherwise. The code for this function is:

```
for (int i=0; i<ttr.size(); i++)
{
if (ttr.get(i)!=0)
    {
    ttr.remove(i);
    ttr.add(i, restTime(i));
    }
}
```

Figure 6.29: Function *updateTTR*

- **restTime**. When called, it initializes a double variable *rest* to which assigns a value if a failure is already ongoing. This value is equal to the remaining time of the Time to Repair related to the ongoing failure. The code for this function is:

```
double rest = 0;

if (i==0)
    rest = ttrA.getRest();
else if (i==1)
    rest = ttrB.getRest();

return rest;
```

Figure 6.30: Function *restTime*

Though in our model the coding method has been repeated for all the work-stations and the buffers, the following procedure represents the way used to code the state-charts that manage the failures of the flowchart 6.28. It is a generic approach that can also be used in other systems.

Each statechart is made by:

- States: waiting, working and failure;
- Transitions ttf and ttr. They are triggered by a timeout that follows the probability distribution related to that availability parameter.
- Variables: *restartA* and *restartB* are used to check which is the dominant failure when it is time to restart the machine; *maxTTR* is a boolean variable used to store the value of the current maximum Time to Repair.

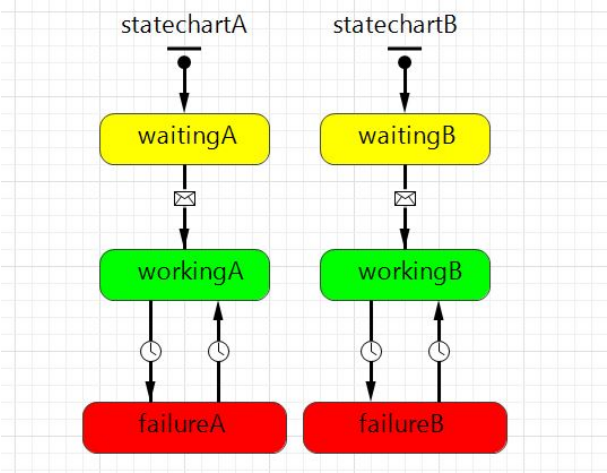


Figure 6.31: Statecharts of Failure A and Failure B of the machine

The codes are written as entry actions inside the status *failure* and inside the transition of the Time to Repair, that links the *failure* state to the *working* one. It is explained the meaning and the functioning of the statechart of the failure A taking into consideration that the procedure to code the failure B is similar.

When a failure occurs, the function *updateTTR* is called and it is calculated the maximum *ttr* between the active ones. It is also initialized the position 0 of the *ttr* collection with the value of ttr_A. An *if-else* statement declare which action is carried out. If there is not another ongoing TTR transition, the function *suspend()* is called and the machine stops to work, the double variable *restartA* is assigned to be true and the *dominantTTR* is declared to be the one that occupies the position 0 inside the *ttr* collection (equal to failure A; while the Time to Repair of the failure B occupies the position 1).
Instead, if the transition ttr_B is active (failure B has already stopped the machine) it is checked whether the Time to Repair A is the biggest one respect to the other ongoing failure. The Time to Repair of the ongoing failure B, during the comparison, is meant as the time remaining until ttr_B is over. It is so assigned a *true* value to the *restart* variable related to the failure with the maximum Time to Repair and *false* to the other one.

```
updateTTR();
ttr.remove(0);
ttr.add(0, ttrUnstuck);

maxTTR = Collections.max(ttr);

if(ttrB.isActive()==false)
    {
    machine.suspend();
    restartA = true;
    dominantTTR=0;
    }
else
    {
    if (ttrA == maxTTR)
        {
        restartA = true;
        restartB = false;
        dominantTTR=0;
        }
     else
        {
        restartB = true;
        }
     }
```

Figure 6.32: Code of the state *failureA*

The transition ttr_A contains the code that restart the machine if the variable *restA* is true. Moreover, it initializes the boolean variable $rest_A$ to *false* that it is its default value and to 0.0 the position related to the ttr_A in the collection *ttr*.

```
ttr.remove(0);
ttr.add(0, 0.0);

if (restartA==true)
{
 machine.resume();
}
 restartA=false;
```

Figure 6.33: Code of the transition ttr_A

Format changes

As already introduced, the line works different kinds and formats of bottles according to the demand (weekly product mix). This fact brings us to two decisions:

- it is needed to create one model of each week, that differs on the formats processed and the set-up times;
- it is required a way to model the change of format in order to reach the desired product mix.

Setups have their own diagram and the stops are managed in the same way of the failures. In 6.18 are present the times for the set-up related to the format change, that are inserted into the model as variables. The *Time to Setup* is a unique value while the *Setup Time* is inserted into the model with a triangular distribution according to assumption 7.

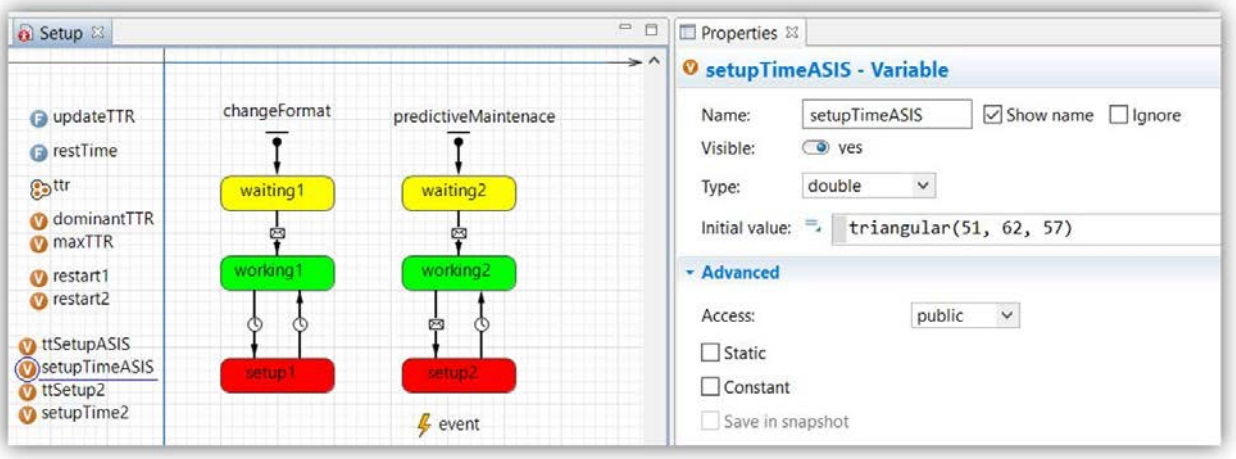


Figure 6.34: View of Setup's diagram in AnyLogic

AnyLogic allows using of tables, collections of related data held in a structured format within a database. It consists of fields (columns), and rows. It is possible to easily import a ready-to-use database with data in the AnyLogic project or create a database table(s) in the model and enter data manually. Since that, a table is created with the information regarding the different formats of the week such as format (letter), batch length, line and type (returnable or one-way). Moreover, the

table contains also the changing time, when the production of one format switches to the production of another format. In fact, reproducing several runs has helped to find out the production time needed to reach the correct percentage of each format, according to the weekly mix. The following image depicts the database table related to the first week.

	db_time	db_format	db_line	db_type	db_batch
1	0	C	OWB	2	0.178
2	1,800	D	RB	1	0.178
3	4,111	EE	OWB	2	0.467
4	5,270	F	RB	1	0.467
5	5,738	H	RB	1	0.801
6	7,337	I	OWB	2	0.289
7	8,147	J	RB	1	0.289

Figure 6.35: Database table for the first week model

The data in the table can be inserted in the form of a variable of different type (int, double, boolean, String). They are attached to the agent once it exits the *Source* block by means of codes in the "On exit" section of the block, based on the current value of the related variables. The change of the format is managed by the changing time present in the database table; when it happens, the Dynamic Event *Switch Format* is created whose purpose is to attribute information about the new format to the current ones.

The figure below illustrates the items such as variables that gather the outputs, the dynamic event *Switch Format* and the variables that characterize the production of the first week.

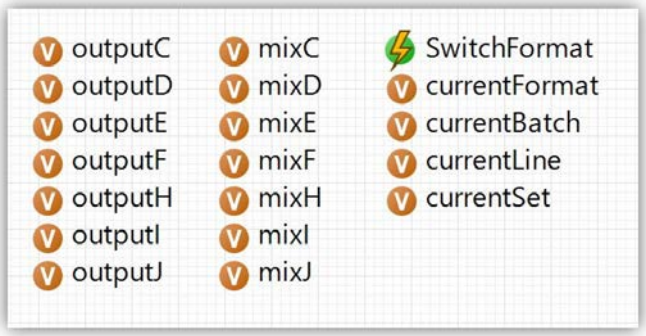


Figure 6.36: Items related to the format change

This attribution of the parameter to the agent is important for different reasons:

- it allows to characterize the agent and make it be representative of that particular format on production;
- it directs the agent to the first buffer it belongs to;
- it allows calculating an accurate output. The output is indeed calculated by a code in the *Sink* block: when an agent arrives, the output variable of the corresponding format is increased by 1.

Data collection

Since the object of study is the performance of the system, variables and elements of the *Analysis* palette to collect statistics during the runs have been added to the model. They are useful to:

- calculate the *Overall Equipment Effectiveness* of the machines;
- collect availability and efficiency parameters of the machines: availability, performance, MTTF, MTTR;
- calculate the product mix;
- achieve other performance measures: throughput of the machines and of the line.

6.2.4 Verification

The verification-phase consists of running the model many times to be assured that the codes work and that the model does what is supposed to do. It has also helped to change and improve the model during the whole building.

Before the validation, it was needed to run the model with 1000 replications. In fact, since the model is stochastic, the result of a single model run might not be representative of the system. This is due to the randomness of the simulation. For this reason, it is required a proper number of replications with independent random numbers in order to make valid conclusions. The approach used to determine the enough number of replications for a simulation is practical and it requires to create a steady-state plot. A *steady-state* plot is a plot of the average over the number of replications. It allows to find out graphically the number of replications from which on the average result is stable (it does not change a lot more replication after replication).

Figure 6.37: Steady-state plot to investigate the number of replications

In the plot, it is shown that the steady-state, where the values of the production rate become approximately constant, it is reached with about 300 replications.

6.2.5 Validation

In order to achieve the validation, parameters and performance values are checked to determine whether or not the simulation model adequately represents the real system. The technique used is to statistically compare the output of the simulation model to the output from the given input. The outputs considered are the OEE of the line, the production rate (bottles/hour) and weekly product mixes. For both the criteria, a result can be accepted only if it does not differ more than 2% compared to its target value.

The results obtained from the model runs after 300 replications are shown in the following tables and compared with the given ones. The *AS IS* situation reflects the values of the original line object of the study.

Effectiveness values

	OEE		***Productivity (bottles/hour)***	
	Target value	*Simulation*	*Target value*	*Simulation*
Week 1	63 - 64 %	63,3%	15500 - 16000	15818
Week 2	52 - 57 %	56,1%	14000 - 14500	14015
Week 3	53 - 57 %	55,6%	13500 - 14000	13897
Week 4	63 - 64 %	62,6%	15500 - 16000	15640

Figure 6.38: Target values VS Simulation results: OEE and productivity

Weekly Product mix

Week 1	*Target Values*	*Simulation*		***Week 2***	*Target Values*	*Simulation*	
Format	*Mix*	*Mix*	*|Δ|*	*Format*	*Mix*	*Mix*	*|Δ|*
C	17,86%	19,40%	1,5%	C	0,85%	1,86%	1,0%
D	22,92%	22,96%	0,0%	D	14,66%	15,07%	0,4%
E	11,50%	10,29%	1,2%	E	9,91%	9,79%	0,1%
F	4,64%	4,60%	0,0%	F	29,61%	28,86%	0,7%
H	15,86%	15,31%	0,6%	G	16,26%	16,19%	0,1%
I	8,04%	8,88%	0,8%	H	28,71%	28,23%	0,5%
J	19,18%	18,55%	0,6%				

Figure 6.39: Target values VS Simulation results: Product mix 1 and 2

Week 3	Target Values	Simulation	
Format	Mix	Mix	|Δ|
C	0,49%	1,09%	0,6%
D	15,85%	16,78%	0,9%
E	14,94%	14,48%	0,5%
F	27,49%	26,86%	0,6%
G	8,48%	8,64%	0,2%
H	32,75%	32,14%	0,6%

Week 4	Target Values	Simulation	
Format	Mix	Mix	|Δ|
A	21,01%	20,99%	0,0%
D	22,75%	24,43%	1,7%
E	12,25%	11,67%	0,6%
F	14,57%	14,25%	0,3%
H	22,34%	21,72%	0,6%
K	1,08%	1,32%	0,2%
L	1,96%	1,96%	0,0%
M	4,03%	3,66%	0,4%

Figure 6.40: Target values VS Simulation results: Product mix 3 and 4

Validation

It is possible to state that thanks to these results the model can be validated and used to run experiments. In fact, the results show that the proposed model has an acceptable level of confidence in the expected and required performances.
Hence, the model is ready to perform experiments.

Chapter 7

Case study: Experimental analysis and improvements

Model simulation runs are carried out to see at first the *as is* situation, expressed mainly with the values of the Overall Equipment Effectiveness and the production rate. The first optimization regards the sequence under which the lots of different formats are processed in the four weeks of study (*Format sequencing problem*). The aim is to reduce the total setup time, that implies a reduction of the makespan. This period of analysis is divided in the four weeks according to the weekly rate of updating of the production plan.

Afterwards, set the optimized scenario, it is performed a *Parameter Variation* experiment where the length of the second buffer is varied in order to maximize the throughput (*Optimal Buffer Sizing*). In fact, in the current situation, the second buffer is not able to effectively decouple the operations of the bottle-washer and the labeller. That is, it does not guarantee its proper function. The buffer has indeed the aim of allowing process continuity and should be placed between two critical areas from the point of view of the micro-downtimes, making it possible for each machine to continue operating also after the interrupting of the adjacent machines (Gershwin, 1992). An optimal buffer size allows to improve the OEE and maximize the throughput of the line.

To sum up, the implemented methodology sees an interaction between combinatorial optimization and simulation modelling in order to increase the performance of the automated bottling line. The procedure is shown in the following scheme.

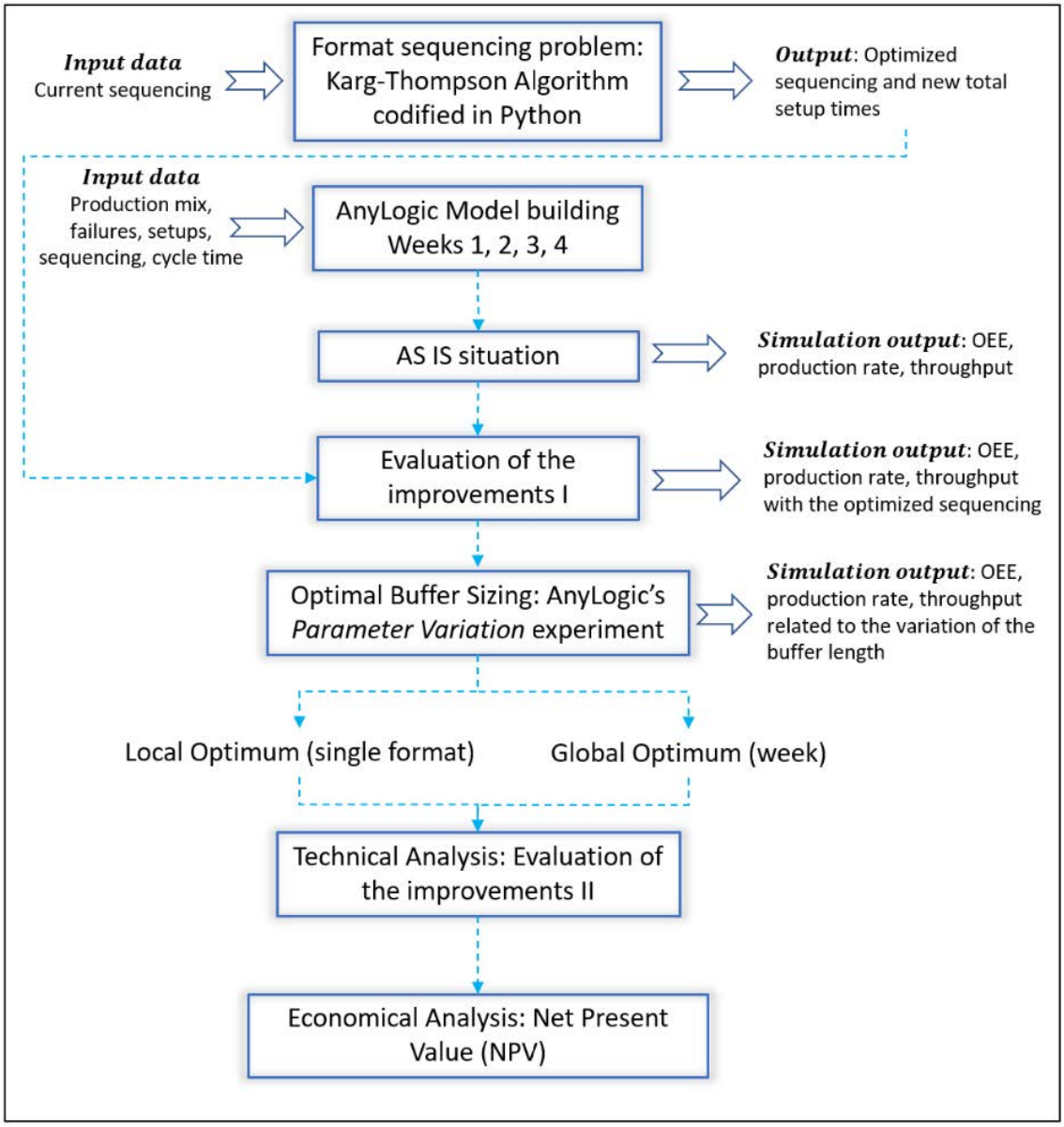


Figure 7.1: Experimental methodology

AnyLogic is used to recreate the line as a model, collect the data on the as-is situation and carry on the buffer optimization while the codified heuristic algorithm (Karg-Thompson algorithm) is helpful to find out the optimal format sequencing. An economic analysis is performed on the most interesting scenario by means of the Net Present Value method.

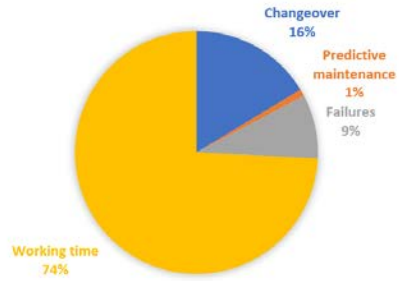
7.1 AS IS Situation

The first thing to do when performing an experiment is to collect the data about the current situation, in order to use them as a comparison with the further improved suggested scenarios. Some values already seen in 6.2.5 during the validation phase are now recalled, they regard the effectiveness parameters such as Overall Equipment Effectiveness, production rate, total output and format's output of the line during the four weeks object of study. In addition to them, there are the current weekly sequences with their corresponding total setup time for the format change. The simulation results of the AS-IS situation have been divided into the four weeks; gathered, the four weeks represent the starting scenario of the experimental analysis.

Week 1

The current situation of the first week of production is shown in the table. The figure below shows the current sequence of formats processing.

Output (bottles)	2659834
Productivity (bott/h)	15826
OEE	63,30%



Format	*Batches*	*Bottles*	*Mix*
C	10322	516076	19,40%
D	12216	610783	22,96%
E	5476	273808	10,29%
F	2447	122328	4,60%
H	8143	407140	15,31%
I	4725	236262	8,88%
J	9869	493436	18,55%
Tot	53197	2659834	

Figure 7.2: Results from the simulation: Week 1 AS IS

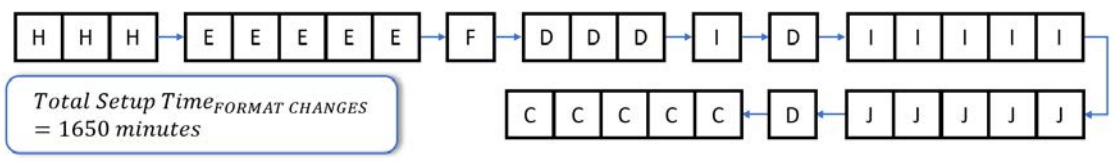


Figure 7.3: Current format sequence: Week 1

The production rate and the output consider the total number of bottles and therefore they are not dependent on the type of format. Overall Equipment Effectiveness is calculated as the ratio between the actual production rate and the nominal production rate. The current sequence results into a total setup time for format changes that is calculated from the set-up matrix (thus, to be considered as theoretical) and insert into the model as a variable. The changeover time, in this case, 1650 minutes, is one of the factors that decrease the availability and therefore the effectiveness of the line, like failures and the time spent still because of the action of predictive maintenance that happens once a week.

Week 2

The current situation of the second week of production is shown in the table, while the figure below indicates the current sequence and the total setup time due to the arrangements for the next productions. The sequence includes the first lot of the previous sequence, that is C. The resulting setup time is 2525 minutes.

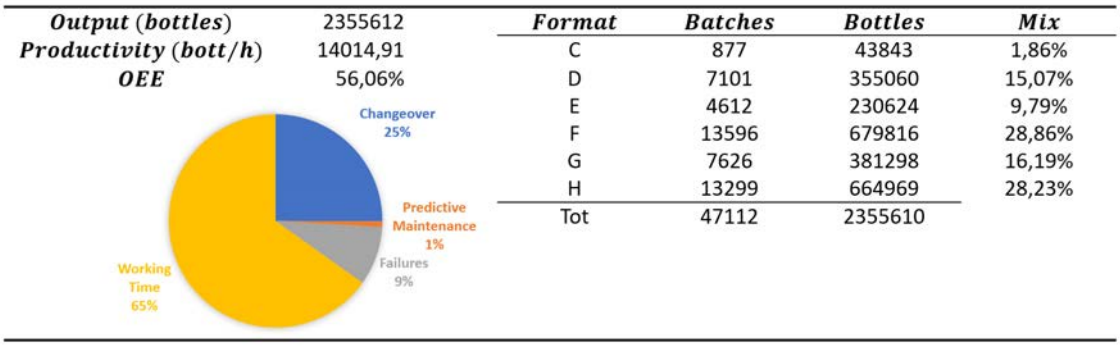


Figure 7.4: Results from the simulation: Week 2 AS IS

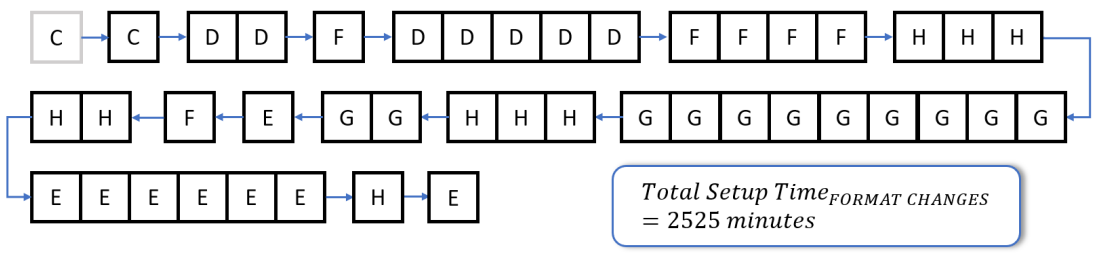
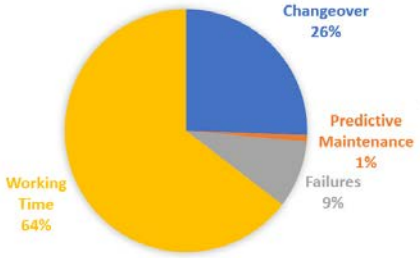


Figure 7.5: Current format sequence: Week 2

Week 3

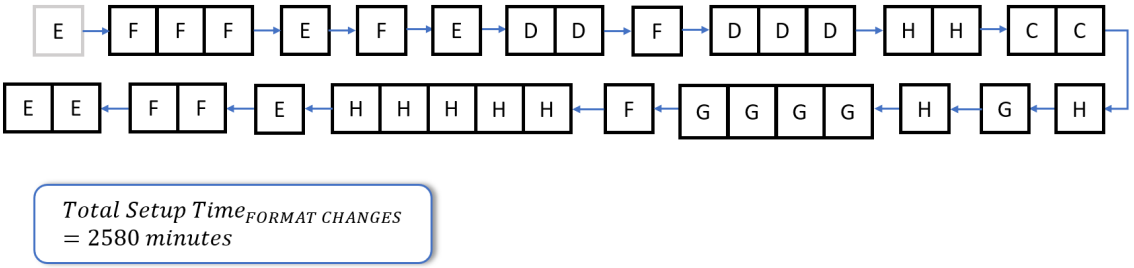
The current situation of the third week of production is shown in the table, while the figure below indicates the current sequence and the total setup time due to the arrangements for the next productions. The sequence includes the first lot of the previous sequence, that is E. This current sequence entails a total theoretical setup time for format change of 2580 minutes.

Output (bottles)	2335649
Productivity (bott/h)	13896,6
OEE	55,59%



Format	*Batches*	*Bottles*	*Mix*
C	511	25562	1,09%
D	7836	391822	16,78%
E	6765	338254	14,48%
F	12547	627368	26,86%
G	4037	201863	8,64%
H	15016	750780	32,14%
Tot	46713	2335649	

Figure 7.6: Results from the simulation: Week 3 AS IS



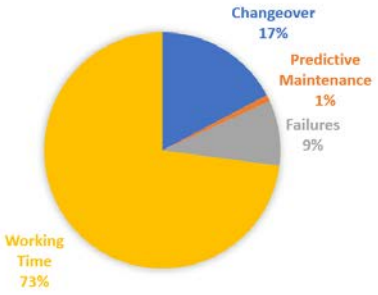
$Total\ Setup\ Time_{FORMAT\ CHANGES} = 2580\ minutes$

Figure 7.7: Current format sequence: Week 3

Week 4

The current situation of the last week of production (fourth) is shown in the table, while the figure below indicates the current sequence and the total setup time due to the arrangements for switching the formats production. The sequence includes the first lot of the previous sequence, that is E. The total setup amounts to 1740 minutes.

Output (bottles)	2628522
Productivity (bott/h)	15639,85
OEE	62,56%



Format	*Batches*	*Bottles*	*Mix*
A	11036,94	551846,9	21,01%
D	12841,38	642068,8	22,75%
E	6132,969	306648,4	12,25%
F	7490,516	374525,8	14,57%
H	11418,45	570922,7	22,34%
K	695,7188	34785,94	1,08%
L	1029,234	51461,72	1,96%
M	1925,234	96261,72	4,03%
Tot	52570,44	2628522	

Figure 7.8: Results from the simulation: Week 4 AS IS

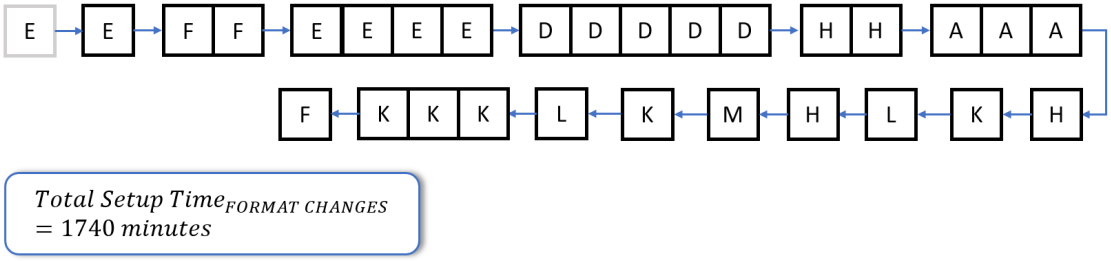


Figure 7.9: Current format sequence: Week 4

7.1.1 Overall view of the first scenario

The first scenario's values are collected as monthly OEE derived from the average of the four weeks' OEE and an average production rate, the average and total output. The total amount of bottles produced for each format is also present. After the tests and the suggested improvements driven from them, the *As Is* values derived from the current settings will be compared with the new ones in order to evaluate each proposed solution from a technical standpoint.

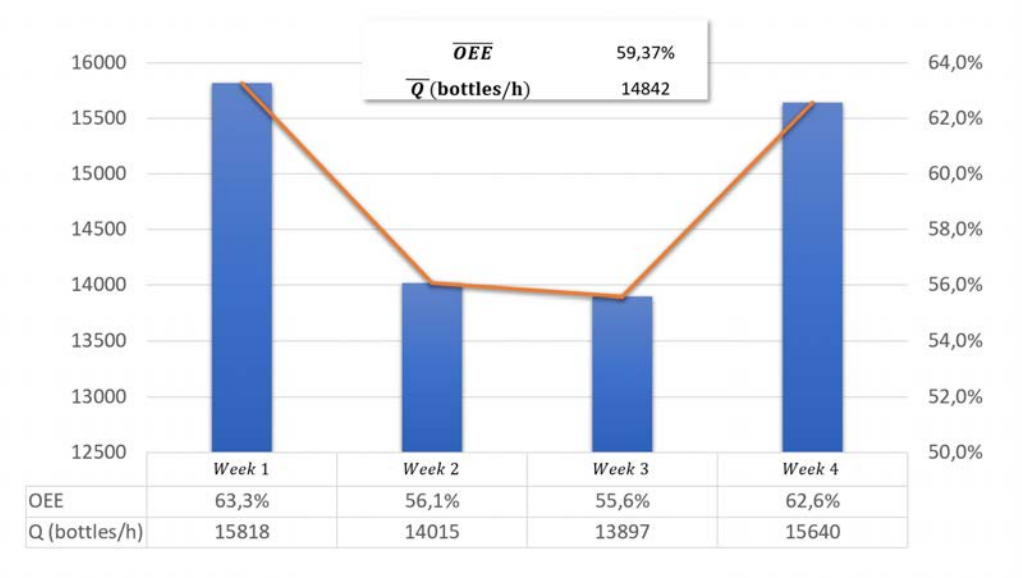


Figure 7.10: Effectiveness values of the AS IS scenario

Letter	*Format (L)*	*Type*	*Bottles*
A	0,25	One-way	551847
B	0,25	Returnable	0
C	0,5	One-way	585482
D	0,5	Returnable	1999734
E	0,75	One-way	1149335
F	0,75	Returnable	1804038
G	1	One-way	583161
H	1	Returnable	2393811
I	0,65	One-way	236262
J	0,65	Returnable	493436
K	1	One-way	34786
L	1	One-way	51462
M	1	One-way	96262

Figure 7.11: Bottle formats' output

7.2 Format sequencing problem

The first improvement wanted to be made regards the sequencing. Given the characteristics of the line, this can be called *Sequence-dependent setup times* scheduling problem. In fact, setup time depends either on the processing lot and on the next one in the sequence. The total amount of setup time is equal to the makespan; therefore, the aim is to reduce it in order to gather a greater amount of available working time. Since the cycle time of the bottleneck is equal either for the work-stations and the buffers, this problem can be reduced to a single machine scheduling problem: an optimized starting sequence is valid in order to increase the overall effectiveness values. The notation for this kind of scheduling problem is:

$$1|s_{jk}|\sum_{i=1}^{n} C_i$$

The solving method has been introduced in 2.6. It concerns the Karg-Thompson heuristic algorithm that has been codified in the programming language Python. By inserting the current sequence and launching the program, the algorithm is iterated and the optimal sequence with its related total setup time is given.

7.2.1 Execution of the program

The interface of the program is very simple. It is necessary just to enter the list of the lots of formats (jobs) to be processed during the week as input. The figure below shows the procedure done for the sequence of the first week.

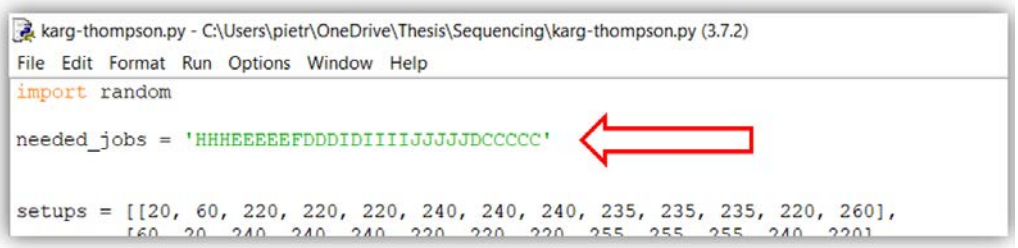


Figure 7.12: Input phase in Python

The outputs are the optimal sequence and the total setup time related to it.

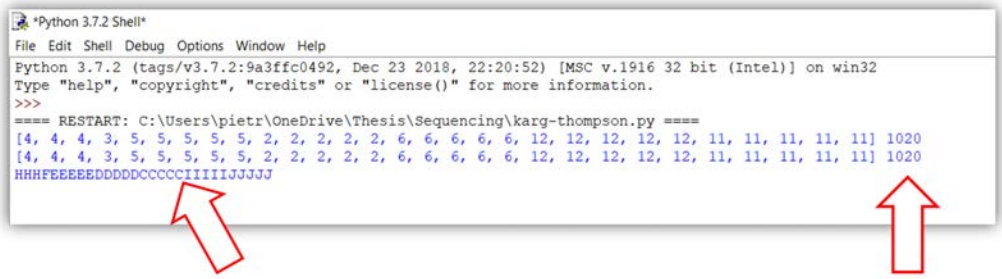


Figure 7.13: Output of the program

7.2.2 Results

The procedure repeated for all the four weeks gives back the following new sequences:

1. HHHFEEEEEDDDDDCCCCCIIIIIJJJJJ (1020 minutes of setup time)
2. J-GGGGGGGGGGGHHHHHHHHHFFFFFFEEEEEEEEEDDDDDDDC (1490 minutes)
3. C-CCDDDDDHHHHHHHHHGGGGGFFFFFFFFEEEEE (1120 minutes)
4. E-EEEEEFFFDDDDDAAAHHHHKKKKKLLM (1420 minutes)

These new setup times are compared with the ones related to the starting sequences.

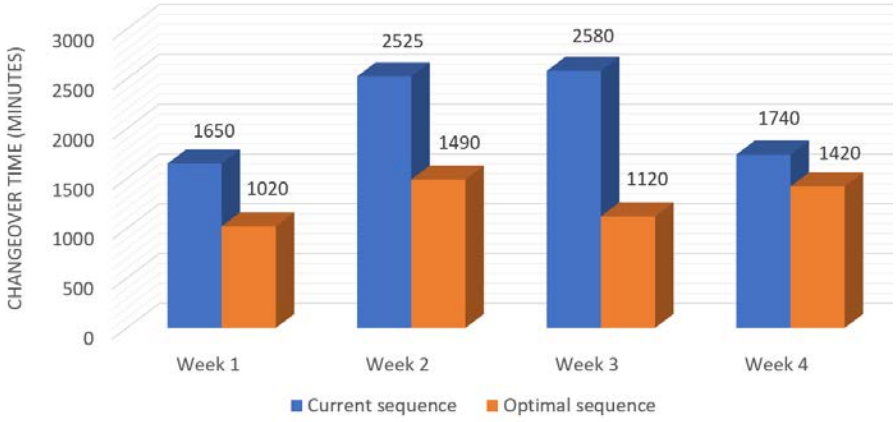


Figure 7.14: Results from the new sequences

It is possible to notice that with the optimized sequences the total setup time has been reduced as regards all the four sequences, with a higher impact on the second and the third ones. It results in a sequencing more balanced and effective. A small improvement could have been reached even without the algorithm; a simple hint is indeed to work in series the lots of the same format since they require the smallest setup time. The entity of these improvements is evaluated by implementing the optimized sequencing in AnyLogic's model.

7.2.3 Evaluation of the improvements with AnyLogic

The format sequences achieved by means of the algorithm are supposed to increase the effectiveness values of the line. To know how much the OEE, the production rate and the throughput of the line would be increased if the format were processed following these new sequences we need to implement the new scenarios in AnyLogic.

In the AnyLogic's model the current setup times in the *changeFormat* statechart are substituted with the ones figured out in the previous phase. The results from the simulation runs are reported.

	Overall Equipment Effectiveness			***Production Rate (bottles/hour)***		
	AS IS	*Optimal sequence*		*AS IS*	*Optimal sequence*	
Week 1	63,3%	68,0%	+ 4,7%	15818	16988	+ 1170
Week 2	56,1%	63,7%	+ 7,6%	14015	15923	+ 1908
Week 3	55,6%	66,4%	+ 10,8%	13897	16592	+ 2695
Week 4	62,6%	64,6%	+ 2,1%	15640	16159	+ 519

Figure 7.15: Simulation output of the new scenarios

	AS IS	***Optimized sequence***
OEE	59,37%	65,66%
Production rate (bottles/hour)	14842,36	16415,41

Figure 7.16: Monthly improvement of the line

The outputs from simulation show how the line effectiveness could increase by changing the order under which the different lots of formats are worked, the so-called *sequence*. The OEE increases by over 6% and with the new settings the line would produce 1500 more bottles per hour.

7.3 Optimal buffer sizing

The second action performed to improve the system regards the buffer capacity, strictly linked to the buffer's length. The buffer under analysis is the second one, considered not able to effectively decouple the operations of the bottle-washer and the labeller. It must mitigate the effect of the disturbance caused by failures and micro-downtimes and let the *n+1* machine continue working even though the *n* is stopped. To perform this test, we use the Parameter Variation Experiment of Anylogic. It affords the opportunity to run the model with different model parameters and analyse how some certain parameters affect the model behaviour. In the figure below it is possible to see the pan of this experiment in Anylogic.

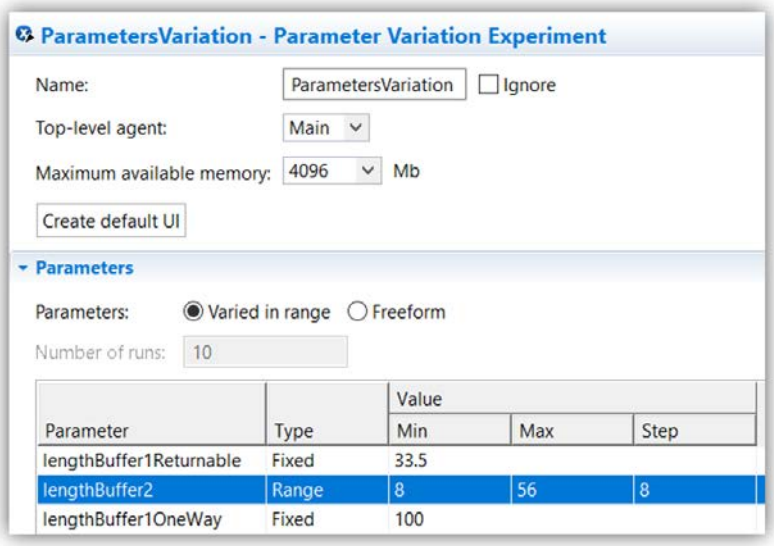


Figure 7.17: Pane of the Parameter Variation Experiment in the software Anylogic

As stated in 6.1.6, the simulation time is 10080 minutes and the number of replications per iteration is set to 300. This number of replications derives from the analysis of the steady-state plot in 6.37. The parameter set to vary in our test is the length of the second buffers. The length of the first buffers is fixed while the length of buffer 2 varies between 8 and 56 meters. The step this parameter will increase its value to reach the maximum is set to 8. The purpose is to assess the impact in terms of OEE and productivity of an increase or decrease in the length of the intermediate buffer through simulation. The final aim is to verify whether a modify

in the layout would improve the effectiveness of the line or not.

The experiment will concern either the weekly scenarios, set with the optimized sequence and some single format situations. Thus, the study is split into two searches that can produce useful insights in order to decide the optimum buffer sizes:

- Local optimum - running the test under the assumption that the line processes only a single format of bottles, therefore in this model setup times for format changes are not present;
- Global optimum - test on the weekly models with optimized sequencing.

Local optimum

The local optimum research regards the three main produced formats D, F and H. Together they cover about the 65% of the total production: D (18,3%), F (20,5%), H (26,5%). Because of this, they may have a certain weight on the choice of the suggested buffer's length.

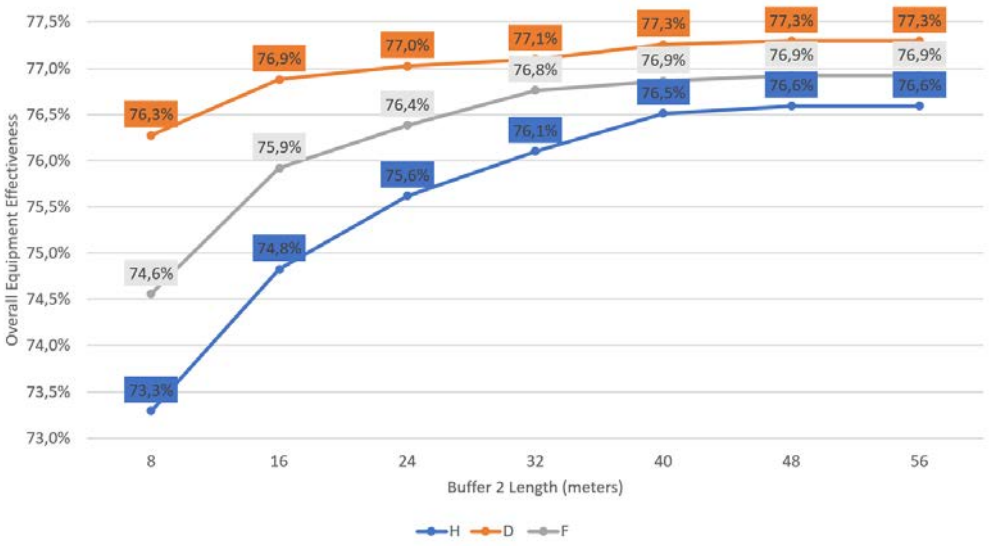


Figure 7.18: OEE vs buffer length (m) variation - Formats

The plot shows as the buffer length, and therefore the capacity, has a significant effect on the increase in the line efficiency till 16 meters for the format D, 32 meters for F and 40 meters for the format H; with further increase in the buffer size, the increase in efficiency is only marginal. These are the local optimum; if the line was mono-format then a change of the buffer length to these value would allow a great performance as regards the effectiveness.

Global optimum

Global optimum means a buffer length that, if implemented in the system, would guarantee an increase of the weekly performance measure of the line such as OEE and production rate. Finding out the optimal buffer size would result in increasing the throughput of the line; this is the purpose of the simulation tests. The results from the parameter variation experiments are to be compared with the local optimums and afterwards supported by an economic analysis. The tests run on the four weeks gives back the following results.

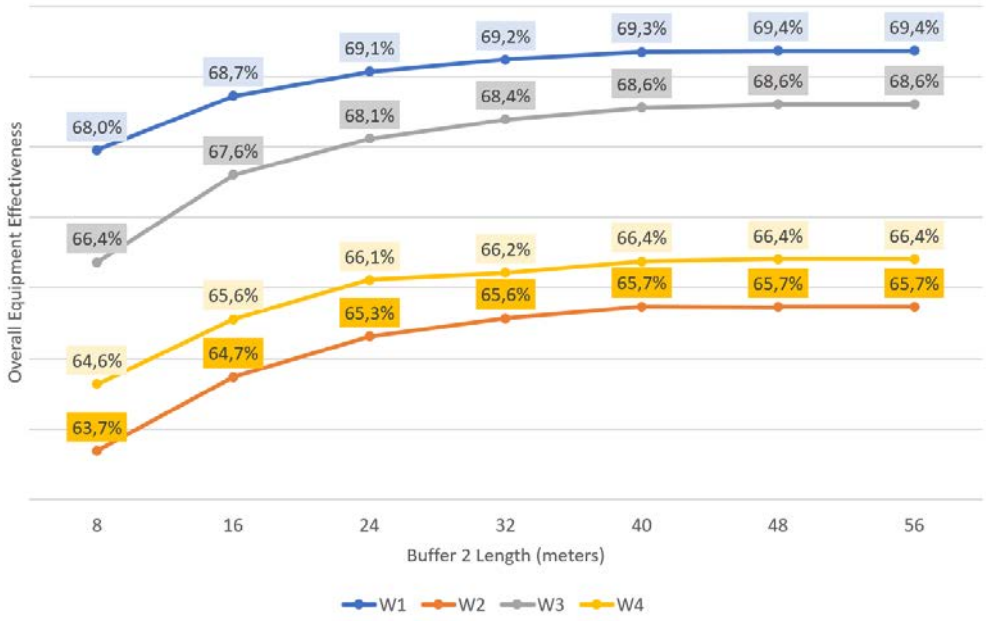


Figure 7.19: OEE vs buffer length (m) variation - Weeks

The plot shows how the increase of the length of the second buffer produces a significant effect on line efficiency from 24 meters on. Indeed, the average OEE

value changes from 65,7% with 8 meters of buffer length to 67,5% with a length of the second buffer equal to 24 meters. After 40 meters of length, corresponding with 67,5% of OEE and a productivity close to 16900 bottles/hour, a further increase in the buffer size results into a null or only marginal increase in efficiency. Since the aim is to maximize the throughput of the line, a buffer size of 40 meters is considered as the optimal one. The results in the following plot and table are calculated as the average of the values of the four weeks simulations.

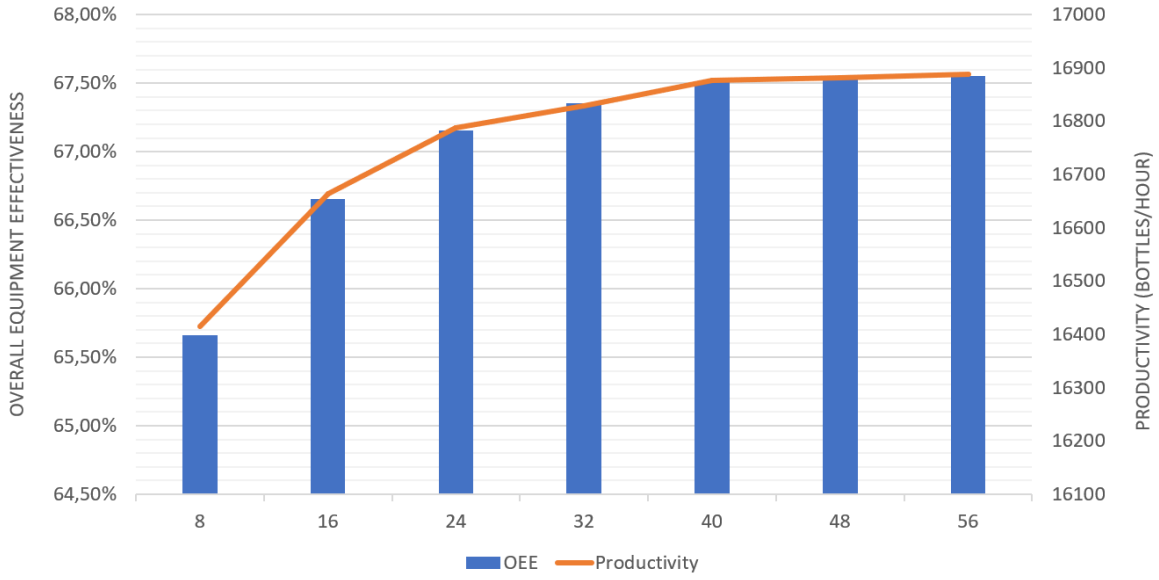


Figure 7.20: Plot: OEE and Production Rate vs buffer length (m) variation

Buffer 2 Length (m)	8	16	24	32	40	48	56
OEE	65,66%	66,66%	67,15%	67,35%	67,50%	67,53%	67,55%
Productivity	16415,41	16663,96	16787,98186	16828,35555	16876,09044	16882,06721	16887,5

Figure 7.21: Table: OEE and Production Rate vs buffer length (m) variation

7.4 Analysis of the results

The starting scenario had an average monthly OEE of 59,4% and average productivity of about 14850 bottles per hour. The simulation's results state that the adjustment of the second buffer size by itself would bring an improvement in the OEE by 2,5% while acting only on the sequence would increase the OEE by 6,3%. The jointly implementation of an optimized sequencing and a modification on the second buffer's length from 8 to 40 meters would instead increase the values of OEE by 8% and the production rate by 13%. The plot below depicts the actions carried on in order to improve the effectiveness of the line with the related effectiveness values. While the optimal sequencing does not imply an additional cost to be implemented, the suggested improvements regarding the buffer sizing do. Therefore, they need to be supported by an economic analysis.

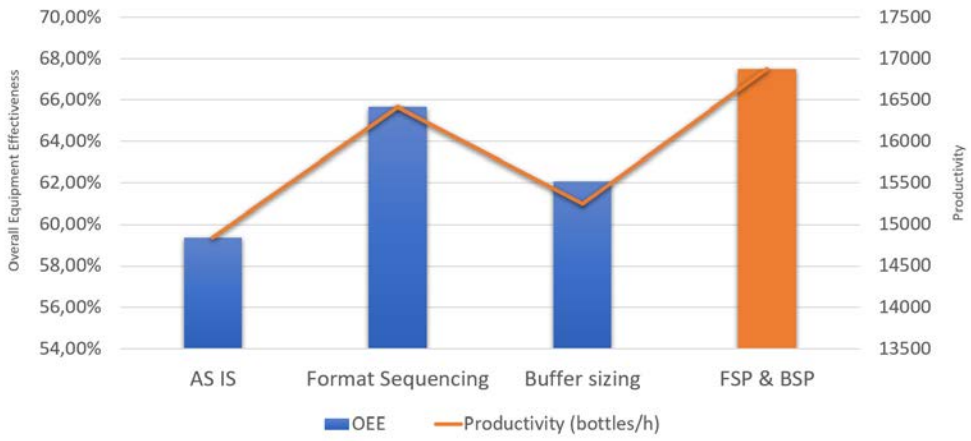


Figure 7.22: Suggested improvements and simulation results

7.5 Economic Analysis

The final step relates to the cost-benefit analysis to evaluate the solutions proposed. Economy data are fictitious. A solution proposed to improve the effectiveness and therefore the throughput of the line was an optimal buffer sizing able to mitigate down-times that affect the line. The solution that maximizes the throughput of the line suggests a change in the length of the second buffer from its original 8 meters to 40 meters; this could guarantee a great impact in the increase of the effectiveness parameter of the line.

The cost of investment and the recoverable OEE, as well as the payback period, have been calculated. The recoverable OEE is considered as the increase in the value from the scenario of optimized sequencing. The cost of the investment to increase the length of the second buffer is assumed to be €1000,00/m. The labour cost of the project and the rearrangement of the layout are assumed to amount to €8000,00 and about €10000,00, respectively. The modification of the buffer length brings additional fixed costs of the period; these cost items form the negative factor of the cash flow of the period in the NPV formula.

INVESTMENT COSTS		*INCREMENTAL FIXED COSTS (year)*	
Cost item	*Cost*	*Cost item*	*Cost*
Additional conveyor	32.000,00 €	Maintenance	2.785,00 €
Project	8.000,00 €	Cleanings	3.360,00 €
Building costs	9.600,00 €	Work-in-process	95,00 €
	49.600,00 €	Utilities	1.920,00 €
		Other costs	2.000,00 €
			10.160,00 €

Figure 7.23: Costs of the new scenario

Given an increase in productivity of 461 bottles/hour from the scenario of optimized sequencing, the annual increase amount to about 3590358 bottles. The contribution margin for the first level has been calculated as the production increase for the unitary contribution margin without fixed costs. The unitary contribution margin is supposed to be independent on the format and it is considered to vary in a range from €0,01 to €0,25 per bottle. For the second level, incremental fixed costs such as costs for maintenance, cleanings, utilities, work-in-process and others have also been considered.

	CM 1	CM 2	CM 3	CM 4	CM 5	CM 6	CM 7	CM 8
Investment cost (€)	32.000,00 €	32.000,00 €	32.000,00 €	32.000,00 €	32.000,00 €	32.000,00 €	32.000,00 €	32.000,00 €
Cost of the project (€)	17.600,00 €	8.000,00 €	8.000,00 €	8.000,00 €	8.000,00 €	8.000,00 €	8.000,00 €	8.000,00 €
Total (€)	49.600,00 €	49.600,00 €	49.600,00 €	49.600,00 €	49.600,00 €	49.600,00 €	49.600,00 €	49.600,00 €
CM1stLevel (€/pz)	**0,01 €**	**0,02 €**	**0,03 €**	**0,05 €**	**0,10 €**	**0,15 €**	**0,20 €**	**0,25 €**
Production increase (pz/week)	3590358	3590358	3590358	3590358	3590358	3590358	3590358	3590358
***CM 1st Level* (€)**	35.903,58 €	71.807,15 €	107.710,73 €	179.517,88 €	359.035,76 €	538.553,63 €	718.071,51 €	897.589,39 €
Fixed Costs (€)	10.159,89 €	10.159,89 €	10.159,89 €	10.159,89 €	10.159,89 €	10.159,89 €	10.159,89 €	10.159,89 €
CM 2ndLevel (€)	25.743,69 €	61.647,27 €	97.550,84 €	169.357,99 €	348.875,87 €	528.393,75 €	707.911,63 €	887.429,50 €

Figure 7.24: Contribution margin calculation

With the second level contribution margin the payback period was calculated, as well as the cash flow, using the *Net Present Value* (NPV) index for the following weeks and assuming an increase in production with an interest rate of 5 percent.

$$NPV_i = -C_0 + \sum_{t=0}^{N} \frac{C_t}{(1+i)^t} \tag{7.1}$$

where C_0 is the initial investment, C is the cash flow and i the interest rate.

	2^{nd} Level Cash Flow							
Year	**0,01 € *CM***	**0,02 € *CM***	**0,03 € *CM***	**0,05 € *CM***	**0,10 € *CM***	**0,15 € *CM***	**0,20 € *CM***	**0,25 € *CM***
0	-49.600,00 €	-49.600,00 €	-49.600,00 €	-49.600,00 €	-49.600,00 €	-49.600,00 €	-49.600,00 €	-49.600,00 €
1	-25.082,20 €	**9.111,68 €**	**43.305,56 €**	**111.693,33 €**	**282.662,73 €**	**453.632,14 €**	**624.601,55 €**	**795.570,96 €**
2	-1.731,91 €	65.027,57 €	131.787,05 €	265.306,02 €	599.103,43 €	932.900,85 €	1.266.698,26 €	1.600.495,67 €
3	**20.506,45 €**	118.280,80 €	216.055,14 €	411.603,82 €	900.475,53 €	1.389.347,23 €	1.878.218,94 €	2.367.090,65 €
4	41.685,85 €	168.998,15 €	296.310,46 €	550.935,06 €	1.187.496,57 €	1.824.058,08 €	2.460.619,58 €	3.097.181,09 €
5	61.856,71 €	217.300,40 €	372.744,09 €	683.631,48 €	1.460.849,94 €	2.238.068,40 €	3.015.286,87 €	3.792.505,33 €

Figure 7.25: Payback period calculation

Results show that the payback period decreases as the contribution margin increases. With a contribution margin of €0,01 the period of time required to recoup the funds expended in the investment, or to reach the break-even point, is equal to 3 months while this period decreases if the contribution margin increases.

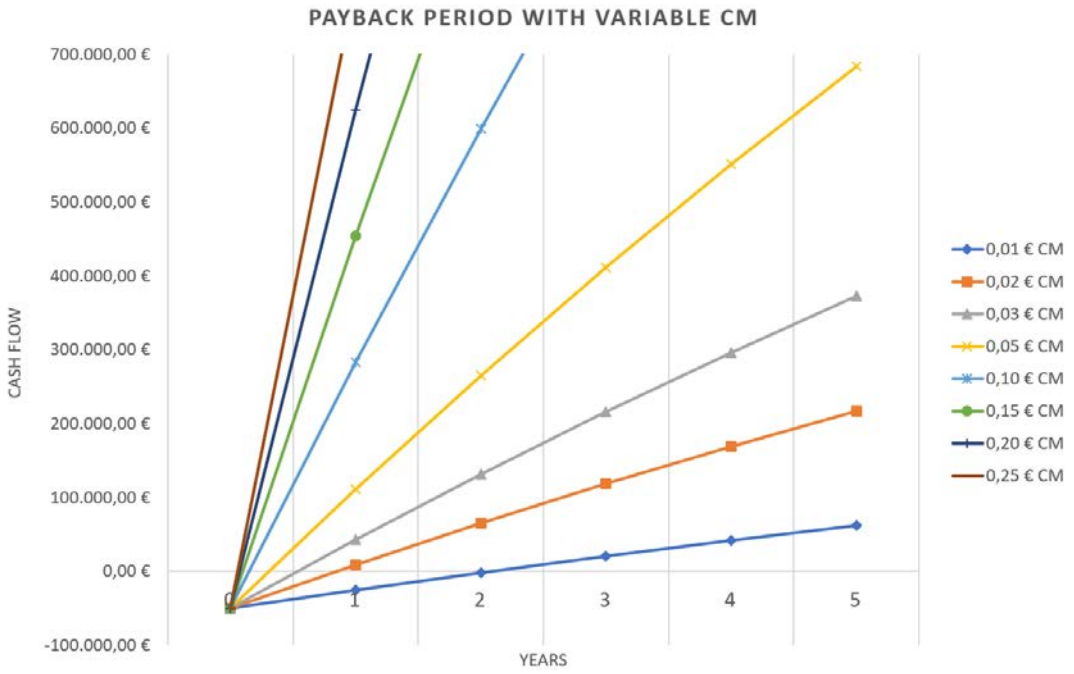


Figure 7.26: Payback period graphic with variable CM

Conclusion

Manufacturing companies have been facing a period in which the market is characterized by large variability in the requests of the customers. The aspects of flexibility, reduced lead time and differentiation must be a core issue mainly in companies that produce high volume and different product mixes such as the ones that belong to the food and beverage sector. Only adapting to the changes in a quick way and raising the effectiveness values can let a company stay competitive in the market. Simulation is the best tool that can be used to improve manufacturing systems since it allows to search for a good feasible solution without disrupting its operation, saving time and costs.

In literature, different works have been found out where researchers have developed simulation models to solve manufacturing cases: Seraj et al. (2008) achieved an increase in the production rate of a rusk production line by 50% replacing a machine with a new one, Chassapis et al. (2009) used simulation to select a preventive maintenance schedule for a production line. Hecker et al. (2010) analysed and optimized a bakery production line proving that high utilization of the equipment would increase the line productivity, therefore, the line performance. Hesmat et al. (2013) applied simulation to solve the bottleneck of a cement production line; the results achieved showed that a modification of work-stations utilization and buffer capacities would bring an increase about more than 15% of the production rate. Aamen et al. (2018) designed a simulation model and used it to evaluate the effect of the buffer capacity and the repair rates of the machines on production line efficiency; they achieved an increase by over 10%.

The purpose of this thesis project was to use simulation to improve the OEE of an automated bottling line. The aim is to operate on the inefficiencies losses that affect the line, that are due to failures, set-ups for predictive maintenance and set-ups for

format changes, the capacity of the buffers between the work-stations. Besides, the line is characterized by product mix constraints. The problems faced by the project are the sequencing according to which the different formats of bottles are weekly produced and the size (capacity) of the buffer of the bottleneck. The case study has been carried out following the General Methodology for Applying Simulation to Problem Solving (Rossetti, 2015). Input data have been studied and analysed to be used in a discrete-event based simulation model in AnyLogic that would reflect the behaviour of the starting one. The most interesting features of the simulation model created are the flowchart that controls the actions of the bottles along the line, the statecharts that allow managing precisely the failures that affect the different elements of the line and the functions that manage the change of formats. After reached the validation, the model was ready to perform simulation runs.

The first scenario simulated collected the results of the as-is situation, the starting situation: 59,37% of OEE and about 15000 bottles/hour of production rate. The first optimization regards the sequence under which the lots of different formats are processed in the four weeks of study (Format sequencing problem). The aim is to reduce the total amount of setup time in order to gather a greater amount of available working time. The sequencing problem of our bottling line is classified as sequence-dependent setup times scheduling problem because setup time depends either on the processing lot and on the next one in the sequence. This first matter has carried out two main considerations:

- Literature presents a great deal of solving techniques as regards scheduling problems. Therefore, when facing one, it is suggested to use the methodology that best fits with the problem to solve.

- The optimal sequencing problem happened to be resolved in a faster way by means of a heuristic algorithm rather than using simulation. This is due to a matter of computational time and the ease of using the algorithm.

Thus, the format sequencing problem has been solved codifying the Karg-Thompson algorithm in the language program Python. Inserted the original sequence of format

lots to be processed, the code run gives as output the optimal sequence with its corresponding setup time. The new scenarios have been implemented in AnyLogic model to collect the improvements in terms of OEE and productivity. The OEE increases by over 6% and with the new settings the line would produce 1500 more bottles per hour.

Afterwards, it is performed a test to achieve the optimal buffer size by means of the *Parameter variation* experiment in AnyLogic. The purpose is to assess the impact in terms of OEE and productivity of an increase or decrease in the length of the intermediate buffer. The maximization of the throughput corresponds with 67,5% of OEE and productivity close to 16900 bottles/hour. Ultimately, the jointly implementation of an optimized sequencing and a modification on the buffer's length would increase the values of OEE by 8% and the production rate by 13%. The figure below summarizes the steps carried out to improve the automated bottling line.

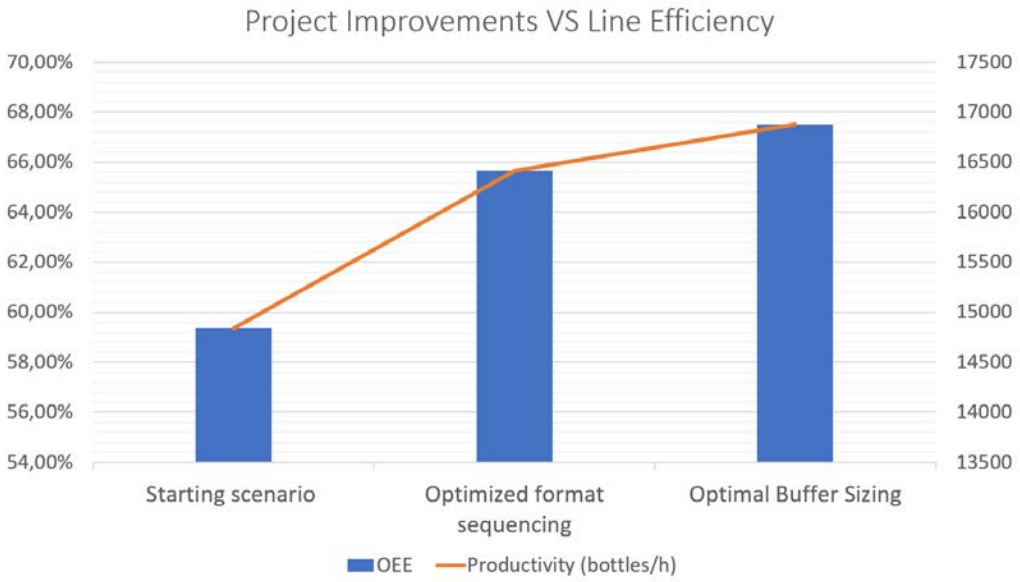


Figure 7.27: Final evaluation of the improvements

The dissertation has carried on a methodology that improved the OEE of an automated bottling line through the combination between the use of both simulation modelling - by means of the software AnyLogic - and combinatorial optimization techniques through the implementation of the Karg Thompson's algorithm.

The considerations pointed out suggest that a simulation model is a powerful tool either for having a benchmark of a manufacturing system to compare with improved situations and to be directly modified in order to gather values of a new possible scenario. AnyLogic has turned out as a very interesting multimethod modelling simulation software. Its main features such as the drag-and-drop mode to create the model, the suite of industry-specific tools gathered in its libraries, animation and visualizations, the data interoperability, and more, supported by a basic knowledge of the JAVA language, allow to create several environments and solve issues of different kinds of system and enhance their performances.

The analysis of and the improvements on the system have pointed out that the higher impact in terms of efficiency is due to the optimization of the format sequences, followed by the research for an optimal buffer size that would increase even more the values of OEE and productivity (bottles/hour). The line still presents margin of improvements, so future studies can be conducted to extend the developed model to test the performances in relation with a decrease or the elimination of certain failures, by means of actions of TPM. Moreover, it would be interesting to simulate a hypothetical replacement of machines with more performing ones in order to bridge the gap between the input of the line and the production rate of the current machines.

Appendix A: Data Samples

Bottle − washer			*Buffer 1*		*Buffer 2*	*Labeler*	
A	D	E	A	C	B	A	B
79,091	27,139	29,301	19,819	41,292	78,957	33,180	1,572
7,040	41,681	75,241	26,795	47,016	22,834	5,843	326,587
47,262	37,030	40,061	47,720	52,954	0,376	67,295	2,108
58,216	27,925	74,869	14,051	63,169	64,981	160,547	9,070
73,100	40,672	29,940	54,001	28,896	20,189	64,416	8,063
4,792	20,528	59,760	49,230	28,419	134,815	16,039	5,808
10,985	11,571	56,525	38,772	35,161	71,002	14,349	147,963
27,730	4,220	70,201	10,454	35,848	5,018	14,682	27,977
36,938	92,923	58,076	88,807	50,483	69,473	13,058	190,786
48,780	5,159	94,009	6,339	51,300	37,871	108,052	70,610
14,647	32,851	53,433	46,562	67,448	126,759	85,154	157,410
23,674	21,050	131,444	46,438	39,343	14,452	211,726	155,576
61,597	7,576	114,191	11,214	62,033	12,092	29,897	58,027
27,974	5,579	39,614	86,707	39,323	17,472	25,489	2,832
14,703	81,676	46,046	64,952	65,875	56,145	26,026	29,065
19,010	103,951	61,891	19,408	58,624	2,078	79,721	22,824
6,246	2,360	62,962	62,234	30,966	165,029	30,300	22,409
13,182	21,131	53,376	11,805	35,314	76,204	6,937	9,142
46,141	32,628	36,008	45,286	45,957	59,336	26,623	25,620
19,075	7,830	100,915	6,180	31,720	10,099	113,156	9,103
54,100	4,130	38,870	18,955	34,088	30,292	29,408	153,736
28,757	44,589	102,364	26,612	26,898	3,442	16,355	0,844
18,032	23,633	40,184	66,533	21,200	18,197	140,322	10,073
31,556	43,490	69,043	51,082	58,177	31,325	144,371	45,687
11,382	1,608	81,132	146,898	41,241	82,480	5,279	79,238
37,800	43,043	79,245	6,170	16,691	88,284	273,772	265,765
9,010	33,922	68,664	27,515	38,112	105,297	8,364	17,306
44,473	51,802	93,795	7,154	42,842	11,675	118,871	13,313
11,979	52,905	24,754	28,207	41,073	28,279	20,300	1,638
15,264	35,898	68,446	171,242	40,783	13,162	26,132	9,347
86,251	21,433	46,381	16,611	47,028	21,672	31,540	4,236
5,466	38,245	49,708	8,711	62,727	41,274	8,239	17,343
25,694	69,161	80,890	21,346	84,330	18,324	158,706	51,011
36,623	67,826	32,530	22,782	25,380	96,645	76,699	94,783
21,768	98,022	42,767	139,397	37,434	94,865	12,752	12,842
24,230	9,957	41,454	22,196	44,685	49,546	56,171	55,335
41,214	51,530	28,304	63,781	27,485	108,728	80,462	42,360
31,345	76,584	93,525	52,340	52,387	16,503	42,470	191,126
39,678	57,900	63,242	6,002	56,659	81,231	35,891	41,513
28,918	48,301	93,798	39,355	82,823	108,870	30,599	87,139
14,574	19,117	106,363	26,414	73,747	1,580	106,178	0,896
20,022	48,829	29,414	37,716	29,247	100,443	0,669	27,947
14,350	6,245	44,724	30,316	29,191	28,718	87,209	13,840
15,266	21,799	86,574	63,663	83,355	15,114	2,875	28,080
21,969	56,901	47,371	37,102	25,315	14,259	116,850	16,081
49,725	51,428	79,476	27,847	80,447	81,787	40,830	34,398
16,292	74,431	56,829	34,174	34,808	54,941	2,747	14,344
43,423	10,609	53,303	30,135	50,774	29,765	57,619	73,889
12,956	4,741	29,620	174,700	45,461	127,792	24,893	15,920
30,113	2,642	42,935	19,285	62,085	39,078	32,410	40,302

Figure 7.28: Times To Failure (minutes)

Bottle − washer			*Buffer 1*		*Buffer 2*	*Labeler*	
A	D	E	A	C	B	A	B
79,091	27,139	29,301	19,819	41,292	78,957	33,180	1,572
7,040	41,681	75,241	26,795	47,016	22,834	5,843	326,587
47,262	37,030	40,061	47,720	52,954	0,376	67,295	2,108
58,216	27,925	74,869	14,051	63,169	64,981	160,547	9,070
73,100	40,672	29,940	54,001	28,896	20,189	64,416	8,063
4,792	20,528	59,760	49,230	28,419	134,815	16,039	5,808
10,985	11,571	56,525	38,772	35,161	71,002	14,349	147,963
27,730	4,220	70,201	10,454	35,848	5,018	14,682	27,977
36,938	92,923	58,076	88,807	50,483	69,473	13,058	190,786
48,780	5,159	94,009	6,339	51,300	37,871	108,052	70,610
14,647	32,851	53,433	46,562	67,448	126,759	85,154	157,410
23,674	21,050	131,444	46,438	39,343	14,452	211,726	155,576
61,597	7,576	114,191	11,214	62,033	12,092	29,897	58,027
27,974	5,579	39,614	86,707	39,323	17,472	25,489	2,832
14,703	81,676	46,046	64,952	65,875	56,145	26,026	29,065
19,010	103,951	61,891	19,408	58,624	2,078	79,721	22,824
6,246	2,360	62,962	62,234	30,966	165,029	30,300	22,409
13,182	21,131	53,376	11,805	35,314	76,204	6,937	9,142
46,141	32,628	36,008	45,286	45,957	59,336	26,623	25,620
19,075	7,830	100,915	6,180	31,720	10,099	113,156	9,103
54,100	4,130	38,870	18,955	34,088	30,292	29,408	153,736
28,757	44,589	102,364	26,612	26,898	3,442	16,355	0,844
18,032	23,633	40,184	66,533	21,200	18,197	140,322	10,073
31,556	43,490	69,043	51,082	58,177	31,325	144,371	45,687
11,382	1,608	81,132	146,898	41,241	82,480	5,279	79,238
37,800	43,043	79,245	6,170	16,691	88,284	273,772	265,765
9,010	33,922	68,664	27,515	38,112	105,297	8,364	17,306
44,473	51,802	93,795	7,154	42,842	11,675	118,871	13,313
11,979	52,905	24,754	28,207	41,073	28,279	20,300	1,638
15,264	35,898	68,446	171,242	40,783	13,162	26,132	9,347
86,251	21,433	46,381	16,611	47,028	21,672	31,540	4,236
5,466	38,245	49,708	8,711	62,727	41,274	8,239	17,343
25,694	69,161	80,890	21,346	84,330	18,324	158,706	51,011
36,623	67,826	32,530	22,782	25,380	96,645	76,699	94,783
21,768	98,022	42,767	139,397	37,434	94,865	12,752	12,842
24,230	9,957	41,454	22,196	44,685	49,546	56,171	55,335
41,214	51,530	28,304	63,781	27,485	108,728	80,462	42,360
31,345	76,584	93,525	52,340	52,387	16,503	42,470	191,126
39,678	57,900	63,242	6,002	56,659	81,231	35,891	41,513
28,918	48,301	93,798	39,355	82,823	108,870	30,599	87,139
14,574	19,117	106,363	26,414	73,747	1,580	106,178	0,896
20,022	48,829	29,414	37,716	29,247	100,443	0,669	27,947
14,350	6,245	44,724	30,316	29,191	28,718	87,209	13,840
15,266	21,799	86,574	63,663	83,355	15,114	2,875	28,080
21,969	56,901	47,371	37,102	25,315	14,259	116,850	16,081
49,725	51,428	79,476	27,847	80,447	81,787	40,830	34,398
16,292	74,431	56,829	34,174	34,808	54,941	2,747	14,344
43,423	10,609	53,303	30,135	50,774	29,765	57,619	73,889
12,956	4,741	29,620	174,700	45,461	127,792	24,893	15,920
30,113	2,642	42,935	19,285	62,085	39,078	32,410	40,302

Figure 7.29: Times To Repair (minutes)

Appendix B: Output Statistical Analysis Minitab®

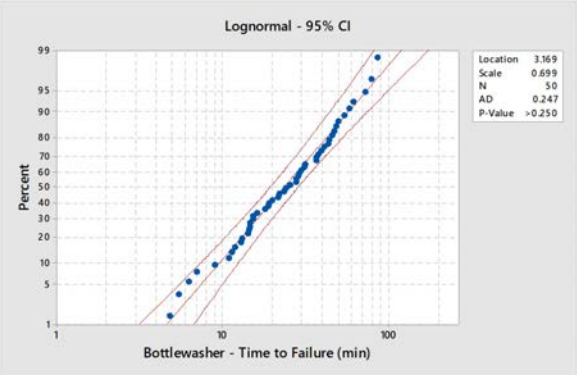
Distribution ID Plot: TTF (min) Bottle-washer A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	29,6482	19,45	24,9622	4,7915	86,2514	1,07299	0,803603

Goodness of Fit Test

Distribution	AD	P
Normal	1,284	<0.005
Box-Cox Transformation	0,246	0,747
Lognormal	0,246	0,747
Exponential	2,962	<0.003
Weibull	0,308	>0.250
Smallest Extreme Value	2,801	<0.010
Largest Extreme Value	0,471	0,24
Gamma	0,217	>0.250
Logistic	0,999	0,005
Loglogistic	0,299	>0.250
Johnson Transformation	0,138	0,974



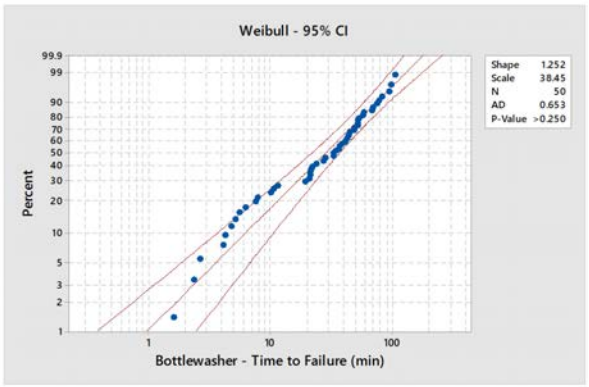
Distribution ID Plot: TTF (min) Bottle-washer D

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	35,924	27,0796	33,3865	1,60794	103,951	0,707773	-0,13075

Goodness of Fit Test

Distribution	AD	P
Normal	0,859	0,025
Box-Cox Transformation	0,477	0,228
Lognormal	1,579	<0.005
Exponential	0,998	0,116
Weibull	0,653	0,086
Smallest Extreme Value	1,911	<0.010
Largest Extreme Value	0,654	0,085
Gamma	0,728	0,07
Logistic	0,751	0,028
Loglogistic	1,396	<0.005
Johnson Transformation	0,433	0,292



Distribution ID Plot: TTF (min) Bottle-washer E

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	62,0714	25,5789	57,4527	24,7544	131,444	0,63317	-0,23415

Goodness of Fit Test

Distribution	AD	P
Normal	0,682	0,07
Box-Cox Transformation	0,303	0,56
Lognormal	0,303	0,56
Exponential	8,158	<0.003
Weibull	0,447	>0.250
Smallest Extreme Value	1,595	<0.010
Largest Extreme Value	0,356	>0.250
Gamma	0,316	>0.250
Logistic	0,66	0,049
Loglogistic	0,39	>0.250
Johnson Transformation	0,159	0,947

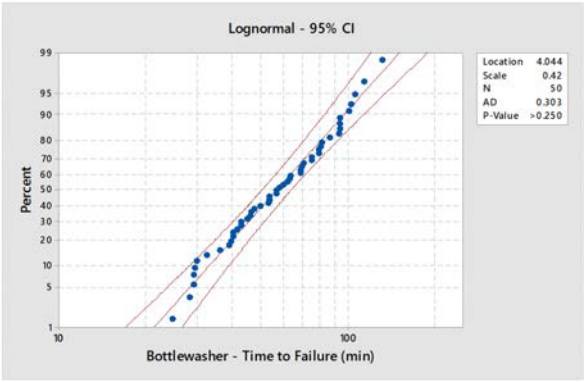


Figure 7.30: Probability distributions Identification for TTFs and TTRs

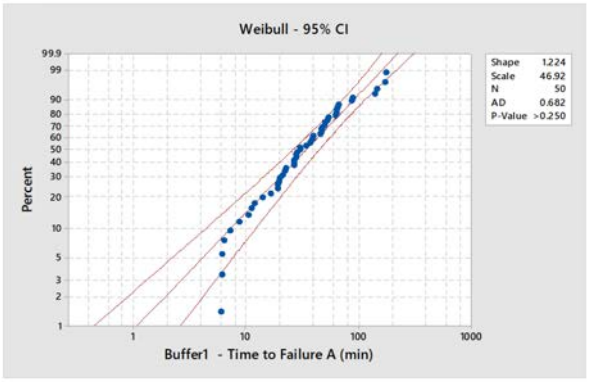
Distribution ID Plot: TTF (min) Buffer 1 A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	43,6203	40,0247	30,2254	6,00177	174,7	1,97634	3,86575

Goodness of Fit Test

Distribution	AD	P
Normal	3,449	<0.005
Box-Cox Transformation	0,338	0,489
Lognormal	0,338	0,489
Exponential	1,288	0,053
Weibull	0,682	0,073
Smallest Extreme Value	6,028	<0.010
Largest Extreme Value	1,041	<0.010
Gamma	0,546	0,192
Logistic	1,893	<0.005
Loglogistic	0,3	>0.250
Johnson Transformation	0,283	0,62



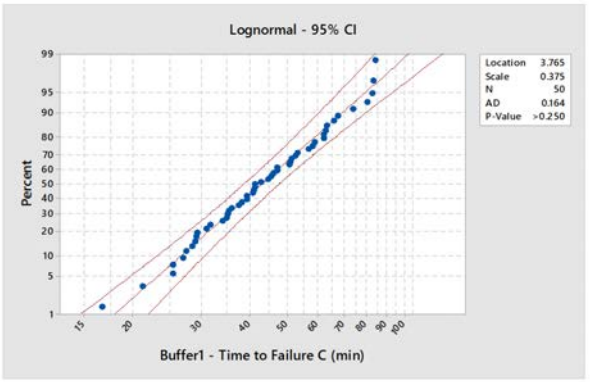
Distribution ID Plot: TTF (min) Buffer 1 C

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	46,1522	17,0646	42,067	16,6907	84,3301	0,639071	-0,2452

Goodness of Fit Test

Distribution	AD	P
Normal	0,73	0,053
Box-Cox Transformation	0,165	0,937
Lognormal	0,165	0,937
Exponential	9,474	<0.003
Weibull	0,571	0,146
Smallest Extreme Value	1,897	<0.010
Largest Extreme Value	0,189	>0.250
Gamma	0,241	>0.250
Logistic	0,622	0,068
Loglogistic	0,234	>0.250
Johnson Transformation	0,155	0,952



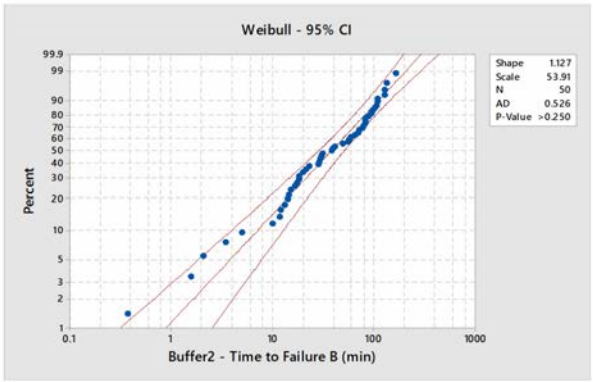
Distribution ID Plot: TTF (min) Buffer 2 B

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	51,775	41,9155	38,4746	0,376331	165,029	0,739693	-0,35375

Goodness of Fit Test

Distribution	AD	P
Normal	1,483	<0.005
Box-Cox Transformation	0,563	0,138
Lognormal	1,317	<0.005
Exponential	0,592	0,375
Weibull	0,526	0,188
Smallest Extreme Value	2,126	<0.010
Largest Extreme Value	1,159	<0.010
Gamma	0,524	0,216
Logistic	1,448	<0.005
Loglogistic	0,907	0,01
Johnson Transformation	0,264	0,683



Distribution ID Plot: TTF (min) Labeler A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	58,4294	58,8403	31,975	0,668593	273,772	1,62865	2,88675

Goodness of Fit Test

Distribution	AD	P
Normal	2,746	<0.005
Box-Cox Transformation	0,269	0,668
Lognormal	0,515	0,183
Exponential	0,305	0,812
Weibull	0,3	>0.250
Smallest Extreme Value	4,496	<0.010
Largest Extreme Value	1,64	<0.010
Gamma	0,311	>0.250
Logistic	2,216	<0.005
Loglogistic	0,419	>0.250
Johnson Transformation	0,217	0,833

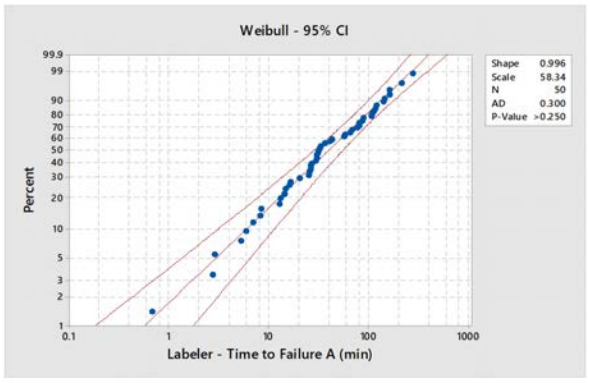


Figure 7.31: Probability distributions Identification for TTFs and TTRs

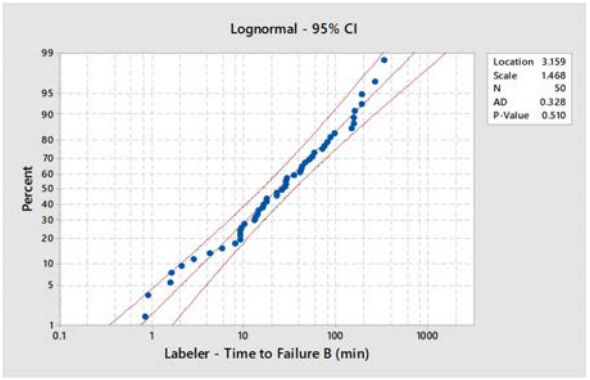
Distribution ID Plot: TTF (min) Labeler B

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	54,8577	71,7573	26,7837	0,843529	326,587	2,07525	4,30451

Goodness of Fit Test

Distribution	AD	P
Normal	4,966	<0.005
Box-Cox Transformation	0,328	0,51
Lognormal	0,328	0,51
Exponential	2,175	0,006
Weibull	0,372	>0.250
Smallest Extreme Value	6,602	<0.010
Largest Extreme Value	3,168	<0.010
Gamma	0,575	0,172
Logistic	3,835	<0.005
Loglogistic	0,273	>0.250
Johnson Transformation	0,2	0,878



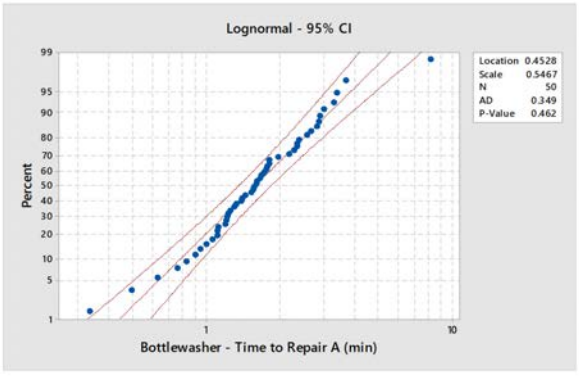
Distribution ID Plot: TTR (min) Bottle-washer A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	1,82842	1,19515	1,56748	0,334614	8,13752	3,17325	15,2166

Goodness of Fit Test

Distribution	AD	P
Normal	2,576	<0.005
Box-Cox Transformation	0,349	0,462
Lognormal	0,349	0,462
Exponential	6	<0.003
Weibull	1,399	<0.010
Smallest Extreme Value	7,411	<0.010
Largest Extreme Value	0,393	>0.250
Gamma	0,572	0,16
Logistic	1,039	<0.005
Loglogistic	0,188	>0.250
Johnson Transformation	0,153	0,956



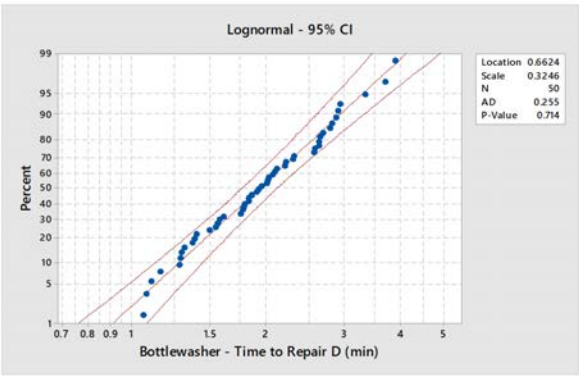
Distribution ID Plot: TTR (min) Bottle-washer D

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	2,04187	0,66994 6	1,93817	1,06274	3,88228	0,720388	0,224218

Goodness of Fit Test

Distribution	AD	P
Normal	0,624	0,098
Box-Cox Transformation	0,255	0,714
Lognormal	0,255	0,714
Exponential	10,824	<0.003
Weibull	0,6	0,118
Smallest Extreme Value	1,848	<0.010
Largest Extreme Value	0,292	>0.250
Gamma	0,295	>0.250
Logistic	0,55	0,111
Loglogistic	0,326	>0.250
Johnson Transformation	0,208	0,859



Distribution ID Plot: TTR (min) Bottle-washer E

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	2,00564	0,832743	2,05983	0,354669	3,54556	-0,17904	-0,82592

Goodness of Fit Test

Distribution	AD	P
Normal	0,35	0,46
Box-Cox Transformation	0,35	0,46
Lognormal	1,695	<0.005
Exponential	7,334	<0.003
Weibull	0,537	0,178
Smallest Extreme Value	0,417	>0.250
Largest Extreme Value	0,942	0,016
Gamma	1,055	0,01
Logistic	0,401	>0.250
Loglogistic	1,234	<0.005
Johnson Transformation	0,35	0,46

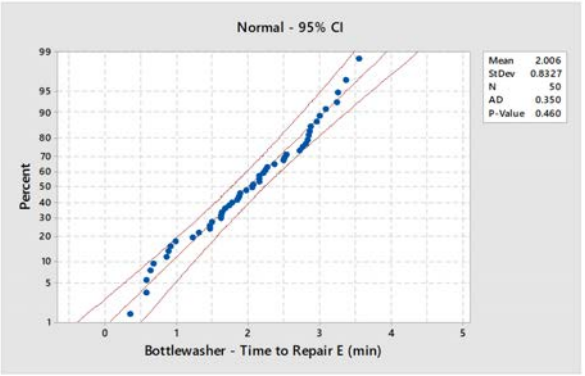


Figure 7.32: Probability distributions Identification for TTFs and TTRs

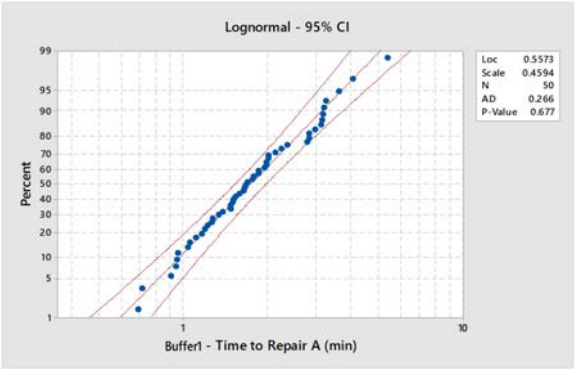
Distribution ID Plot: TTR (min) Buffer 1 A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	1,93973	0,948443	1,6799	0,689417	5,36615	1,33631	2,25691

Goodness of Fit Test

Distribution	AD	P
Normal	1,53	<0.005
Box-Cox Transformation	0,266	0,677
Lognormal	0,266	0,677
Exponential	7,236	<0.003
Weibull	0,95	0,016
Smallest Extreme Value	3,419	<0.010
Largest Extreme Value	0,461	>0.250
Gamma	0,515	0,21
Logistic	1,177	<0.005
Loglogistic	0,306	>0.250
Johnson Transformation	0,212	0,849



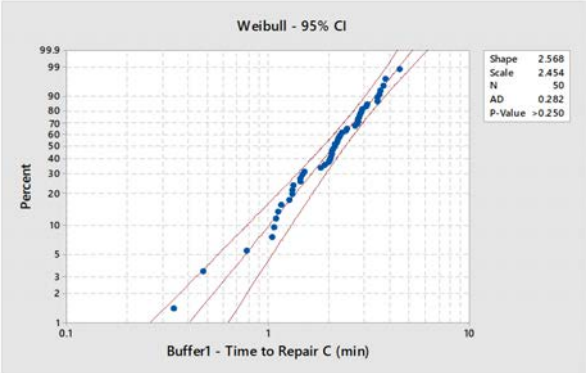
Distribution ID Plot: TTR (min) Buffer 1 C

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	2,17932	0,923037	2,11856	0,33901	4,46032	0,216134	-0,37403

Goodness of Fit Test

Distribution	AD	P
Normal	0,323	0,517
Box-Cox Transformation	0,368	0,417
Lognormal	1,004	0,011
Exponential	7,415	<0.003
Weibull	0,282	>0.250
Smallest Extreme Value	0,95	0,016
Largest Extreme Value	0,507	0,206
Gamma	0,529	0,198
Logistic	0,385	>0.250
Loglogistic	0,71	0,038
Johnson Transformation	0,323	0,517



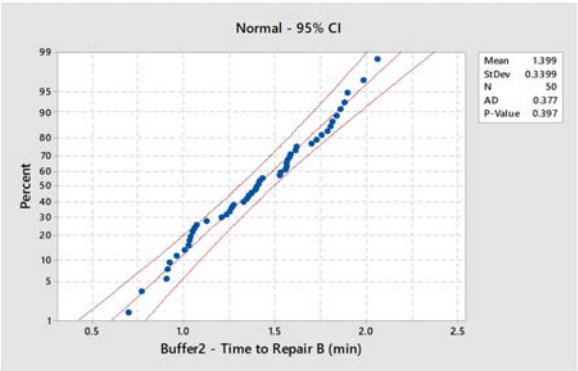
Distribution ID Plot: TTR (min) Buffer 2 B

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	1,39873	0,339884	1,40302	0,698326	2,05819	-0,08546	-0,85424

Goodness of Fit Test

Distribution	AD	P
Normal	0,377	0,397
Box-Cox Transformation	0,377	0,397
Lognormal	0,667	0,077
Exponential	13,142	<0.003
Weibull	0,344	>0.250
Smallest Extreme Value	0,52	0,194
Largest Extreme Value	0,768	0,043
Gamma	0,542	0,183
Logistic	0,47	0,199
Loglogistic	0,674	0,046
Johnson Transformation	0,377	0,397



Distribution ID Plot: TTR (min) Labeler A

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	2,16049	1,21494	1,85387	0,592028	6,48485	1,42223	2,41991

Goodness of Fit Test

Distribution	AD	P
Normal	1,815	<0.005
Box-Cox Transformation	0,393	0,363
Lognormal	0,393	0,363
Exponential	5,744	<0.003
Weibull	0,912	0,019
Smallest Extreme Value	3,889	<0.010
Largest Extreme Value	0,48	0,232
Gamma	0,515	0,212
Logistic	1,208	<0.005
Loglogistic	0,315	>0.250
Johnson Transformation	0,307	0,551

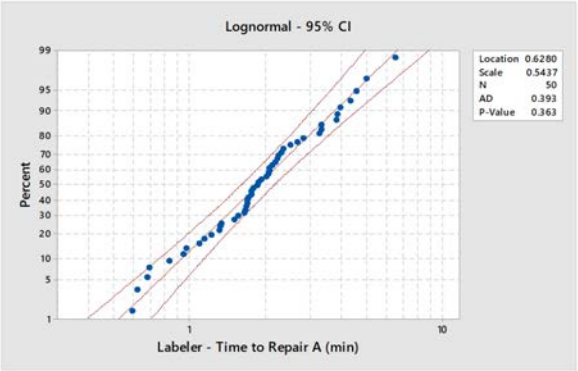


Figure 7.33: Probability distributions Identification for TTFs and TTRs

Distribution ID Plot: TTR (min) Labeler B

Descriptive Statistics

N	Mean	StDev	Median	Min	Max	Skewness	Kurtosis
50	3,04398	1,14849	3,12173	0,992272	6,77914	0,591773	0,98762

Goodness of Fit Test

Distribution	AD	P
Normal	0,274	0,649
Box-Cox Transformation	0,282	0,624
Lognormal	0,649	0,085
Exponential	9,174	<0.003
Weibull	0,265	>0.250
Smallest Extreme Value	1,667	<0.010
Largest Extreme Value	0,498	0,215
Gamma	0,374	>0.250
Logistic	0,255	>0.250
Loglogistic	0,571	0,094
Johnson Transformation	0,274	0,649

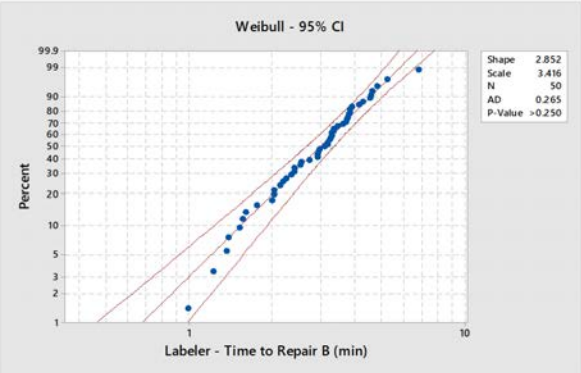


Figure 7.34: Probability distributions Identification for TTFs and TTRs

Bibliography

[1] Aamen, W., AlKahtani, M., Mohammed, M. K., Abdulhameed, O. and El-Tamimi, A. M. (2018) *Investigation of the effect of buffer storage capacity and repair rate on production line efficiency*, Journal of King Saud University - Engineering Sciences, Vol. 30, pp. 243-249.

[2] Allahverdi, A., Gupta, J.N.D. and Aldowaisan, T. (1999) *A review of scheduling research involving setup considerations*, Omega, 27, 219239.

[3] Allahverdi, A. and Soroush, H.M. (2008) *The significance of reducing setup times/setup costs*, European Journal of Operational Research, 187, 978984.

[4] Altuger, G. and Chassapis, C. (2009) *Multi Criteria Preventive Maintenance Scheduling Through Arena Based Simulation Modeling*, IEEE, pp. 1214–2134, Winter Simulation Conference 2009.

[5] Anderson, T. W. and Darling, D. A. (1952) *Asymptotic theory of certain "goodness-of-fit" criteria based on stochastic processes*, Annals of Mathematical Statistics 23: 193–212.

[6] The AnyLogic Company (2018) *AnyLogic Help*, available from: http://www.anylogic.com/anylogic/help/"

[7] Armentano, V.A. and Mazzini, R. (2000) *A genetic algorithm for scheduling on a single machine with setup and due dates*, Production Planning and Control, 11, 713720.

[8] Baker, K. R. (1974) *Introduction to sequencing and scheduling*, New York, NY: Wiley.

[9] Baker, K. R. and Scudder, G. D. (1990) *Sequencing with earliness and tardiness penalties: A review*, Operations Research, 38, 22–36.

[10] Bard, J., Dar-El, E., and Shtub, A. (1992) *An analytic framework for sequencing mixed model assembly lines*, International Journal of Production Research, 30(1):35–48.

[11] Barnes, J.W. and Vanston, L.K. (1981) *Scheduling jobs with linear delay penalties and sequence dependent setup times and release dates*, Operations Research, 29, 146154.

[12] Battini D., Manzini R., Persona A. and Regattieri A., (2006) *TPM Approach and new buffer design paradigm in plant layout*, 12th ISSAT International Conference on Reliability and Quality in design. Chicago, August 3-5, 2006.

[13] Bazan, P. and German, R. (2012) *Hybrid Simulation of Renewable Energy Generation and Storage Grids*, Proceedings of the 2012 Winter Simulation Conference.

[14] Beaty, S. (1992) *Genetic algorithms for instruction sequencing and scheduling*, Workshop on Computer Architecture Technology and Formalism for Computer Science Research and Applications. Naples, Italy.

[15] Beheshti, Z., Mariyam, S. and Shamsuddin, H. (2013) *A Review of Population-based Meta-Heuristic Algorithm*, International Journal of Advanced Computer Science and Applications, Vol. 5, No. 1, March 2013, ISSN 2074-8523.

[16] Richard, B. (1972) *Dynamic Programming and Partial Differential Equations*, Elsevier Science Technology Books.

[17] Bianco, L., Ricciardelli, S., Rinaldi, G., and Sassano, A. (1988) *Scheduling Tasks with Sequence-Dependent Processing Times*", Naval Research Logistics Quarterly, Vol. 35, pp. 177–184.

[18] Bolat, A. (1994) *Sequencing jobs on an automobile assembly line: objectives and procedures*, International Journal of Production Research, 32(5):1219–1236.

[19] Borshchev A. and Filippov, A. (2004) *From System Dynamics and Discrete Event to Practical Agent Based Modeling: Reasons, Techniques, Tools*, The 22nd International Conference of the System Dynamics Society, July 25–29, 2004, Oxford, England.

[20] Braglia, M., Frosolini, M., and Zammori, F. (2009) *Overall equipment effectiveness of a manufacturing line (OEEML): an integrated approach to assess systems performance*, Journal of Manufacturing Technology Management, 20 (1), 8–29.

[21] Buzacott, J. A. and Hanifin, L. E. (1978) *Automatic transfer lines with buffer stocks*, International Journal of Production Research, 5(3): 183-200.

[22] Campbell, H. G., Dudek R. A. and Smith, M.L. (1970) *A Heuristic Algorithm*

for the n Job m Machine Sequencing Problem, Management Science, Vol. 16, pp. B630–B637.

[23] Caridi M. and Sianesi A. (1999) *Multi-agent system in production planning and control: an application to the scheduling of mixed-model assembly lines*, International Journal of Production Economics 68, 29-42.

[24] Chung, C. A. (2004) *Simulation Modeling Handbook: A practical Approach*, CRC Press LLc.

[25] Conway, R.W., Maxwell, W.L. and Miller, L.W. (1967) *Theory of Scheduling*, Addison- Wesley, Reading, MA.

[26] Dallery, Y. and Gershwin, S. B. (1992) *Manufacturing flow line systems: a review of models and analytical results*, vol. 12, pp. 3–94, 1992.

[27] Dannenbring, D.G. (1977) *An Evaluation of Flowshop Sequencing Heuristics*, Management Science, Vol. 23, pp. 1174–1182.

[28] De Ron, A. J. and Rooda, J.E. (2006) *OEE and equipment effectiveness: an evaluation*, International Journal of Production Research, Taylor.

[29] Dorigo, M., Maniezzo, V. and Colorni, A. (1996) *The ant system: Optimization by a colony of cooperating agents*, IEEE Transactions on Systems, Man, and Cybernetics, Part B, 26(1):29–41.

[30] Dreo, J. (2007) *Dreaming of Metaheuristics*, available from: http://metah.nojhan.net.

[31] Drury, H. B. (1918) *Economy Scientific Management*, second ed., rev., Columbia University Press, New York, NY, 1918, p. 125 126.

[32] Du, J. and Leung, J.Y.T. (1993a) *Minimizing Mean Flow Time in Two-Machine Open Shops and Flow Shops*, Journal of Algorithms, Vol. 14, pp. 24–44.

[33] Fabricky, W.J. and Blanchard, B.S. (1991) *Life-Cycle Costs and Economic Analysis*, Prentice Hall.

[34] Flynn, B.B. (1987) *The effects of setup time on output capacity in cellular manufacturing.* International Journal of Production Research, 25, 17611772.

[35] Gagne, C., Prince, W.L. and Gravel, M. (2002) *Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times.* Journal of the Operational Research Society, 53, 895906.

[36] Gilmore, P.C. and Gomory, R.E. (1964) *Sequencing a One-State Variable Ma-*

chine: a Solvable Case of the Travelling Salesman Problem", Operations Research, Vol. 12, pp. 655–679.

[37] Gonzalez, T. and Sahni, S. (1978b) *Flowshop and Jobshop Schedules: Complexity and Approximation*, Operations Research, Vol. 26, pp. 36–52.

[38] Grygoryev, I. (2014) *AnyLogic in Three Days*, available from: https://www.anylogic.com/resources/books/free-simulation-book-and-modeling-tutorials/

[39] Gupta, J.N.D. (1972) *Heuristic Algorithms for the Multistage Flow Shop Problem*, AIIE Transactions, Vol. 4, pp. 11–18.

[39] He, W. and Kusiak, A. (1992) *Scheduling manufacturing systems*, Computers in Industry, 20, 163175.

[40] Hecker, F., Hussein, W. and T. Becker (2010) *Analysis and optimization of a bakery production line using ARENA*, International Journal of Simulation Modeling, vol. 9, no. 3, pp. 108–216, Dec. 2010.

[41] Herrmann, J., Lee, C.Y, and Snowdon, J. (1993) *A Classification of Static Scheduling Problems*, Complexity in Numerical Optimization, P.M. Pardalos (ed.), World Scientific, pp. 203–253.

[42] Heshmat, M., El-Sharief, M. A. and El-Sebaie, M. G. (2013) *Simulation Modeling of Automatic Production Lines with Intermediate Buffers*, International Journal of Scientific Engineering Research, vol. 4(7), Jul. 2013, pp. 2528–2535.

[43] Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press.

[44] Hossain, M. S., Asadujjaman M., Nayon, M. A. A. and Bhattacharya, P. (2014) *Minimization of Makespan in Flow Shop Scheduling Using Heuristics*" International Conference on Mechanical, Industrial and Energy Engineering at Bangladesh (2014).

[45] Hosseinpour, F. and Hajihosseini, H. (2009) *Importance of Simulation in Manufacturing*, World Academy of Science, Engineering and Technology International Journal of Economics and Management Engineering Vol:3, No:3, 2009.

[46] Huang, S.H. (2003) *Manufacturing productivity improvement using effectiveness metrics and simulation analysis*, International Journal of Production Research, 41 (3), 513–527.

[47] Ignall, E. and Schrage, L. (1965) *Application of the branch and bound technique*

to some flow-shop scheduling problems, Operations Research, 13(3), 400-412.

[48] Jackson, J.R. (1955) *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, University of California, Los Angeles.

[49] Jeong, K-Y. and Phillips, D.T. (2001) *Operational efficiency and effectiveness measurement*, International Journal of Operations Production Management, Vol. 21 No. 11, pp. 1404–1416.

[50] Johnson, D. B.(1977) *Efficient algorithms for shortest paths in sparse networks*, Journal of the ACM 24(1):1-13, January 1977.

[51] Johnson, S. M. (1954) *Optimal Two and Three-Stage Production Schedules with Setup Times Included*, Naval Research Logistics Quarterly, Vol. 1, pp. 61–67.

[52] Jonsson, P. and Lesshammar, M. (1999) *Evaluation and improvement of manufacturing performance measurement systems—the role of OEE*, International Journal of Operations and Production Management 1999, 19, 55–78.

[53] Kelton, W. D., Sadowski, D. T. and Sturrock, R.P. (2007) *Simulation with ARENA*, Fourth Edi. McGraw-Hill New York.

[54] Kogan, K. and Levner, E. (1998) *A polynomial algorithm for scheduling small scale manufacturing cells served by multiple robots*, Computers Operations Research, 25, 5362.

[55] Krajewski, L.J., King, B.E., Ritzman, L.P. and Wond, D.S. (1987) *Kanban, MRP and shaping the manufacturing environment*, Management Science, 33, 3957.

[56] Lageweg, B.J., Lawler, E.L., Lenstra J.K. and Rinnooy Kan, A.H.G. (1982) *Computer-Aided Complexity Classification of Combinatorial Problems*", Communications of the ACM, Vol. 25, pp 817–822.

[57] Lahmar, M., Ergan, H., and Benjaafar, S. (2003) *Resequencing and feature assignment on an automated assembly line*, IEEE Transactions on Robotics and Automation, 19(1):89–102.

[58] Land A. H. and Doig, A. G. (1960) *An Automatic Method of Solving Discrete Programming Problems*, Econometrica, Vol. 28, No. 3 (Jul., 1960), pp. 497-520.

[59] Law, A. M. and McComas, M.G. (1997) *Simulation of manufacturing systems*, Proceeding of the 1997 winter simulation.

[60] Law, A.M. and Kelton, W.D. (1991) *Simulation Modeling and Analysis*, McGraw-Hill, New York.

[61] Lawler, E.L., Lenstra J.K., Rinnooy Kan, A.H.G. and Shmoys, D. (1993) *Sequencing and Scheduling: Algorithms and Complexity*, in Handbooks in Operations Research and Management Science, Vol. 4: Logistics of Production and Inventory, S. S. Graves, A. H. G. Rinnooy Kan and P. Zipkin, (eds.), pp. 445– 522, North-Holland, New York.

[62] Lee, C. Y. (2004) *Machine Scheduling with Availability Constraints*, Chapter 22 in Handbook of Scheduling, J. Y.-T. Leung (ed.), Chapman and Hall/CRC, Boca Raton, Florida.

[63] Lenstra, J.K. (1977) *Sequencing by Enumerative Methods*,Mathematical Centre Tracts 69, Centre for Mathematics and Computer Science, Amsterdam.

[64] Levner, E. M. (1969) *Optimal Planning of Parts Machining on a Number of Machines*, Automation and Remote Control, Vol. 12, pp. 1972–1981.

[65] Liu, C.Y., Chang, S.C. (2000) *Scheduling flexible flow shops with sequence-dependent setup effects.* IEEE Transactions on Robotics and Automation, 16, 408419.

[66] McMahon, G.B. and Burton, B (1967) *Flow shop scheduling with branch and bound method*, Oper. Res., Vol. 15, pp. 473–481, 1967.

[67] McMullen, P. R. (1998) *JIT sequencing for mixed-model assembly lines with setups using Tabu Search*, Production Planning Control, 9:5, 504-510.

[68] McMullen, P.R. and Frazier, G.V. (1998) *Using simulated annealing to solve a multi-objective assembly line balancing problem with parallel workstations*, International Journal of Production Research, 36:10, 2717-2741.

[69] Molderink, A., Bosman, M.G.C., Bakker, V., Hurink, J. L. and Smit, G.J.M. (2009) *Simulating the Effect on the Energy Efficiency of Smart Grid Technologies*, Proceedings of the 2009 Winter Simulation Conference.

[70] Montoya-Torres, J.R., Soto-Ferrari, M. and González Solano, F. (2010) *Production scheduling with sequence-dependent setups and job release dates*, DYNA, 77 (163), pp. 260-269.

[71] Moore, J. M. (1968) *An n job, one machine sequencing algorithm for minimizing the number of late jobs*, Management Science 15, 102-109.

[72] Mourtzis D. and Doukas M., (2012) *Decentralized Manufacturing Systems Review: Challenges and Outlook*, Logistics Research, Springer. 1865-0368.

[73] Mussellman, K.J. (1992) *Conducting a Successful Simulation Project*, Proceedings of the 1992 Winter Simulation Conference (Ed.: Crain, Wilson, Swain, and Goldsman) Arlington, Virginia, pp. 115-121.

[74] Nachiappan, R.M. and Anantharam, N. (2006) *Evaluation of overall line effectiveness (OLE) in a continuous product line manufacturing system*, Journal of Manufacturing Technology Management, 17 (7), 987–1008.

[75] Neapolitan, R. E. and Naimipour, K. (2004) *Foundations of algorithms using Java pseudocode*, Sudbury, Mass: Jones and Bartlett Publishers.

[76] Nakajima, S. (1988) *Introduction to TPM: total productive maintenance*, Cambridge, MA: Productivity Press Inc.

[77] Nord C., Pettersson B. and Johansson B. (1997) *TPM: Total Productive Maintenance in the Volvo company*, Idrottens Grafiska I Goteborg AB, Molnlycke.

[78] Ozgur, C.O. and Brown, J.R. (1995) *A two stage traveling salesman procedure for the single machine sequence-dependent scheduling problem*, Omega, 23, 205219.

[79] Palmer, D.S. (1965) *Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining a Near Optimum*", Operational Research Quarterly, Vol. 16, pp. 101–107.

[80] Pandey, D.S. and Raut, N. (2016) *Implementing TPM by doing RCA*, International journal of advanced research in science, engineering and technology, vol. 3, no. 2, 2016.

[81] Papadopoulos, H.T. (1996) *Queuing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines*, European Journal of Operational Research, vol. 92, pp. 1–27, 1996.

[82] Pinedo, M. (2008) *Scheduling : Theory, Algorithms, and Systems*, Springer Science+Business Media, LLC.

[83] Pinedo, M. (1982) *Minimizing the Expected Makespan in Stochastic Flow Shops*, Operations Research, Vol. 30, pp. 148–162.

[84] Potts C.N. and Kovalyov, M.Y. (2000) *Scheduling with Batching: A Review*, European Journal of Operational Research, Vol. 120, pp. 228–249.

[85] Presby, J.T. and Wolfson, M.L. (1967) *An algorithm for solving job sequencing problems*, Management Science, 13, B454B464.

[86] Raghavachari, M. (1988) *Scheduling Problems with Non-Regular Penalty Functions: A Review*, Opsearch, Vol. 25, pp. 144–164.

[87] Ravindran, A., Phillips, D.T. and Solberg, J.J. (1987) *Operations Research: Principles and Practice*, 2nd Edition, John Wiley, New York.

[88] Reddi, S.S. and Ramamoorthy, C.V. (1972) *On the Flowshop Sequencing Problem with No Wait in Process*, Operational Research Quarterly, Vol. 23, pp.323–330.

[89] Rinnooy Kan, A.H.G. (1976) *Machine Scheduling Problems: Classification, Complexity and Computations*, Nijhoff, The Hague.

[90] Robinson, C. (2004) *Calculating line or process OEE*, Maintenace Technology, available at: www.mt-online.com/newarticles2/06-94mm.cfm

[91] Rossetti, M.D. (2015) *Simulation Modeling and Arena*, Wiley Publishing.

[92] Sadowski, R.P. (1989) *The Simulation Process: Avoiding the Problems and Pitfalls*, Proceedings of the 1989 Winter Simulation Conference (Ed.: MacNair, Musselman, and Heldelberger) Washington, D.C., pp. 72-79.

[93] Seraj, Y. (2008) *A Simulation Study To Increase The Capacity Of A Rusk Production Line*, Department of Industrial Engineering, vol. 5, no. 9, pp. 1395–1404, 2008.

[94] Smith, R. D. and Dudek, R. A. (1967) *A General algorithm for the solution of the n job, m machine sequencing problem of the flowshop*, Operations Research, vol.15, pp. 71–82, 1967. Also, see their Errata Operations Research 17, 756.

[95] Smith, W.E. (1956) *Various Optimizers for Single Stage Production*" Naval Research Logistics Quarterly, Vol. 3, pp. 59–66.

[96] Taillard, E. (1990) *Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem*, European Journal of Operational Research, Vol. 47, pp. 65–74.

[97] Tan K.C. and Narasimhan, R. (1997) *Minimizing tardiness on a single processor with sequence-dependent setup times: a simulated annealing approach*, Omega 25(6):619–634.

[98] Tan, K.C., Narashiman, R., Rubin, P.A. and Ragatz, G.L. (2000) *A comparison of four methods for minimizing total tardiness on a single processor with sequence*

dependent setup times, Omega, 28, 313326.

[99] Thompson, K. (1968) *Programming Techniques: Regular expression search algorithm*, Communications of the ACM, Volume 11 / Number 6 / June, 1968.

[100] Trovinger, S.C. and Bohn, R.E. (2005) *Setup time reduction for electronics assembly: Combining simple (SMED) and IT-based methods*, Production and Operations Management, 14, 205217.

[101] Ulgen, O.M. (1991) *Proper Management Techniques are Keys to a Successful Project*, Industrial Engineering, pp. 37-41.

[102] Uzsoy, R., Lee, C.Y. and Martinvega, L.A. (1992) *Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine*, Naval Research Logistics, 39, 369388.

[103] Voß, S., Martello, S., Osman, I.H. and C. Roucairol (1999) *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston.

[104] Wang, Y., Li, X. and Ma, Z. (2017) *A Hybrid Local Search Algorithm for the Sequence Dependent Setup Times Flowshop Scheduling Problem with Makespan Criterion*, Sustainability 2017, 9, 2318.

[105] Weimer, C. W., Miller, J. O. and Hill, R.R. (2016) *Agent-Based Modeling: an Introduction and Primer*, Proceedings of the 2016 Winter Simulation Conference.

[106] Widmer, M. and Hertz, A. (1989) *A New Heuristic Method for the Flow Shop Sequencing Heuristic*, European Journal of Operational Research, Vol. 41, 186–193.

[107] Zennaro, I., Battini, D., Sgarbossa, F., Persona, A. and De Marchi, R. (2018) *Micro downtime: Data collection, analysis and impact on OEE in bottling lines the San Benedetto case study*, International Journal of Quality Reliability Management, Vol. 35 Issue: 4, pp.965-995.

[108] Zhu, X. and Wilhelm, W.E. (2006) *Scheduling and lot sizing with sequence-dependent setup: A literature review*, IIE Transactions, 38, 9871007.