

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Production Line Balancing Simulation: A case study in the Footwear Industry

João Tiago Martins Covas

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: A. Miguel Gomes

Co-Supervisor: Rui Rebelo

July 31, 2014

A Dissertação intitulada

“Production Line Balancing Simulation: A case study in the footwear industry”

foi aprovada em provas realizadas em 23-07-2014

o júri

Presidente Professor Doutor Fernando Manuel Ferreira Lobo Pereira
Professor Catedrático do Departamento de Engenharia Eletrotécnica e de
Computadores da Faculdade de Engenharia da Universidade do Porto

Professora Doutora Maria Teresa do Valle Moura Costa
Professor Adjunto do Departamento de Engenharia Mecânica da Instituto Superior
de Engenharia do Porto

Professor Doutor António Miguel da Fonseca Fernandes Gomes
Professor Auxiliar do Departamento de Engenharia e Gestão Industrial da Faculdade
de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua
exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente
autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou
inspirados em trabalhos de outros autores, e demais referências bibliográficas
usadas, são corretamente citados.

Autor - João Tiago Bacelar Martins Barreiro Covas

Resumo

O objectivo principal desta dissertação prende-se com a criação de um modelo de simulação que possa avaliar e validar balanceamentos de linhas de produção na indústria do calçado, onde estes terão de responder a pedidos customizados pelos clientes. Este modelo de simulação deve ter a capacidade de reproduzir o funcionamento da linha de produção de forma a avaliar e validar operações de balanceamento desta linha, as quais podem incluir a adição ou remoção de máquinas na linha de produção, modificação de ordens, etc.

Este problema está relacionado com um projecto em curso de uma investigação em maior escala, no qual participam algumas empresas da indústria do calçado, um centro tecnológico de calçado e o INESC TEC. Tradicionalmente, a indústria do calçado divide-se em três partes. Porém, nesta dissertação, vamos apenas abordar a parte da costura de sapatos, por ser a mais complexa entre as três, e por ter uma mão de obra bastante intensiva, onde todos os postos de trabalho necessitam de um operador. O modelo de simulação criado poderá ter a capacidade de mais tarde ser aplicado a um caso real, e pretende resolver um problema de balanceamento de linhas de produção através de uma ferramenta simulação.

Abstract

The main objective of this dissertation is to create a simulation model that can validate a problem of production line balancing, in a specific case that refers to the footwear industry, where they will need to respond to customized requests by the customers regarding the footwear industry. This simulation model must have the ability to reproduce the functioning of the production line in order to evaluate and validate balancing operations, which may include the addition or removal of machines in the production line, modification of orders, etc..

This problem is included in an ongoing project, that as has the participation of some companies in the footwear industry, a footwear technological center and INESC TEC. Traditionally, the footwear industry is divided in three stages. In this dissertation we are only going to approach the stitching stage, for being the most complex between the three of them, and a part with a lot of operators, where all the workstations have operations assigned. The simulation model created should be able to be applied to a real case, and it is intended to solve a problem of balancing production lines via a simulation tool.

Agradecimentos

Em primeiro lugar gostaria de agradecer ao Professor António Miguel Gomes, por toda a disponibilidade, apoio prestado e paciência enquanto orientador científico, especialmente na fase final do trabalho. Além disso, gostaria também de agradecer ao Engenheiro Rui Diogo Rebelo pelo apoio, pela dedicação e pela oportunidade de trabalhar num projecto tão motivador e com uma equipa dedicada e prestável. Ao Fábio Alves e ao Samuel Moniz, meus colegas de equipa, um muito obrigado pelos conselhos e pelo tempo que disponibilizaram para me darem ajuda no desenvolvimento do meu trabalho.

Agradeço ao INESC TEC, pela oportunidade de trabalhar num espaço com um muito bom ambiente de trabalho e com umas condições de trabalho excepcionais.

Um grande obrigado também aos meus pais pelo apoio que me deram e que nunca me deixaram ir abaixo, e que proporcionaram tudo o que eu precisava para a elaboração do trabalho. Também ao meu irmão que, mesmo à distância, não deixou de me dar ajuda quando precisei.

À Sónia que nunca me deixou de encorajar para enfrentar os dias de trabalho e ao restante "gang do i105" (à Joana, ao Daniel, ao Gonçalo, à Mafalda, à Marta, à Leonor, à Ana Zé, ao Mota, ao Filas e à Diana) que permitiu uma motivação extra para trabalhar e onde houve um sentimento de amizade, companheirismo e entreajuda enorme para o desenvolvimento dos trabalhos de cada um.

A todos os restantes amigos da FEUP que acompanharam o meu percurso e que de alguma forma contribuíram para que alcançasse com sucesso o fim desta etapa, de uma maneira tão magnífica e que irei sempre recordar para o resto da minha vida.

Ao "Ambrósio", que por me conhecer tão bem sabe quando eu preciso mais de ajuda, a qual nunca faltou quando precisei e onde encontrei um ombro amigo quando precisei, especialmente quando precisava de me afastar um pouco do trabalho e apenas estar com bons amigos.

Às bandas e artistas que me proporcionaram a melhor banda sonora enquanto trabalhava, especialmente aos Dead Combo, que me serviram de inspiração mesmo nos momentos em que ela estava mais escassa.

E, por último, agradeço à Sofia, por não me ter deixado desistir, quando o verdadeiro objectivo acabaria por me encontrar no fim.

Tiago Covas

*“A man is a success if he gets up in the morning and gets to bed at night,
and in between he does what he wants to do.”*

Bob Dylan

Contents

1	Introduction	1
1.1	Motivation for this project	1
1.2	Project Plan	2
1.3	Hosting Institution	3
1.4	Thesis Structure	4
2	Simulation of Assembly Lines	5
2.1	Assembly Lines	6
2.1.1	Assembly Lines Balancing Problem	6
2.1.2	Types of Assembly Lines Balancing	7
2.2	Job-Shop	10
2.3	Simulation	11
2.3.1	Why Use Simulation	11
2.3.2	Types of simulation	12
2.3.3	Running a Simulation	13
2.3.4	Simulation frameworks	14
2.3.5	Simio and Object-oriented Simulation	14
3	High Speed Shoe Factory	21
3.1	The Stitching Line	21
3.2	Comments on the Project	25
4	Approach and implementation	27
4.1	Input data	27
4.2	Building the model	28
4.2.1	Objects	28
4.2.2	User-defined Tables	30
4.2.3	Processes	30
4.3	Indicators	34
4.4	Deadlocks	35
5	Experimentations and Results	39
5.1	An example of a line balancing	43
5.2	Results Analysis	45
6	Conclusions and Future Works	47
A	Simulation 3 - Vehicle Status	49

References

51

List of Figures

2.1	ALB basic example	6
2.2	Classification of assembly line balancing	9
2.3	Two-Sided Assembly Line Structure	9
2.4	Job-Shop Layout	10
2.5	Flow-Shop Layout	10
2.6	Product-Process Matrix	11
2.7	Example of 3D animation in Simio	15
2.8	Example of an Excel Import to Simio	16
2.9	Example of a Simio process	17
2.10	Running a experimentation with different scenarios	18
3.1	U-Shaped Lines	22
3.2	Production line in focus	23
4.1	The 6 entities that represent 6 different model of shoes	27
4.2	The facility recreation, that include all the objects	29
4.3	A sample of the Sequence Table	30
4.4	A sample of the Mix Table	31
4.5	The Process that will decide to which machine to send the entity	31
4.6	Source Machine Decide Trigger	31
4.7	Machine Decide Server	31
4.8	Vehicle OK process	33
4.9	Machine Available Process	34
4.10	Trigger for MachineA1	34
4.11	Entities Produced	35
4.12	Entities Arrived	36
4.13	Machine's Status Pie	37
4.14	Vehicle's Status Pie	37
4.15	A Deadlock situation	37
4.16	Facility with storage implemented	38
5.1	Bloqued Machines	39
5.2	Processing Machines	40
5.3	Interarrival-Time Distribution	41
5.4	Week Schedule	42
5.5	Day Schedule	42
5.6	Simulations for the last two scenarios	46

List of Tables

4.1	Sequence of Operations	28
5.1	Interarrival Time Probability Distribution	41
5.2	Number of Simulation Runs	43
5.3	Simulation 1	44
5.4	Simulation 2 Mix	44
5.5	Simulation 2	44
5.6	Simulation 3	45
A.1	Vehicle Properties for simulation 3	50

Abbreviations and Symbols

3D	Three Dimensional
ALB	Assembly Lines Balancing
CESE	Centre for Enterprise Systems Engineering
FEUP	Faculty of Engineering
FIFO	First In First Out
HSSF	High Speed Shoe Factory
HTTP	Hypertext Transfer Protocol
INESC-TEC	Institute for Systems and Computers Engineering of Porto - Techonology & Science.
MMALB	Mixed-Model Assembly Line Balancing
MtMALB	Multi-Model Assembly Line Balancing
NEA	Number of Entities Arrived
NEP	Number of Entities Produced
QREN	National Strategic Reference Framework
R&D	Research and development
SALB	Single-Model Assembly Line Balancing
SALBP	Simple Assembly Line Balancing Problem
SALBP-1	Simple Assembly Line Balancing Problem - minimizing the number m of stations given the cycle time c
SALBP-2	Simple Assembly Line Balancing Problem - minimizing c given m
SALBP-E	Simple Assembly Line Balancing Problem - maximizing the line efficiency
WIP	Work In Progress

Chapter 1

Introduction

This dissertation regarding a Production Line Balancing Simulation is based in a specific case study in the footwear industry that is being developed in INESC TEC - Technology & Science. As this specific case involves high complexity in its configuration, it is important to find an efficient way to resolve it. And so, the mean that was used to find a solution for it was applying a simulation framework to the problem.

The industrial production has been changing its productive system due to the change in the clients demand. It becomes necessary to build faster and more flexible production systems, that can manage a high variety of models to be produced simultaneously in small quantities. Associated with this come production lines with high complexity layouts, for which there is a need of Production Lines Balancing, which consists in changing the operations and workstations so that the line is balanced, resulting in a continuous flow in production and in an idle time as minimum as possible. This balancing involves great amounts of time and costs to put into effect and to observe it actually working. Putting this, there is a need of reducing these cons and evaluate if an assembled line of production is going to have good results.

The approach used in this dissertation to avoid these high costs and great amounts of time is the utilization of a simulation framework. So, first it is needed to choose this framework and then we will use it to recreate the facility of the case study, build a simulation model and try to improve its results by asking questions such as "Does this machine need to be here when it would be a lot more productive in other place?", "Why are the operations in this sequence?" or "What could increase the performance of this production line?". In the end, it is expected to have a model that, when receiving input data, it can simulate the behaviour of a production line and so provide information about how to better distribute the workstations and the operations along it.

1.1 Motivation for this project

Each person has his own identity and, although people may have similar needs, each person has a unique taste and a unique personality. And lately, industrial production has been focusing on identifying these unique characteristics of the individuals and then satisfying them in a way that

approaches the efficiency of a mass production. This can be called mass customization and although these two words may seem antagonistic, technology is making this possible. And we can already see this happening in very small (and almost imperceptible) ways. For instance, when we browse the internet, the data that we receive can be automatically stored in a HTTP cookie file. This will allow that the website records the information in which the user was interested and this way the website can show advertisements customized to each user depending on what the user has browsed before. This is one example of mass customization. However, when we talk about mass customization related to the footwear industry, we are describing the possibility of providing a customer an option to personalize his shoes as he wants (e.g. choosing the color of the shoe, the pattern of the shoe laces, the size of the shoe, etc.)

Providing this possibility to the customers via web can increase exponentially the number of customers. However it raises not only the challenge of being capable of deliver in their houses the shoe exactly as they ordered it, just as they saw the shoes in their computer screens, but also to produce faster and with lower prices. To hundreds, thousands or even millions of clients.

The motivation for this project comes from the fact that, as we can understand easily, it is a problem that doesn't strict to this company neither to the footwear industry. On the contrary, it may be applied to a lot of other companies that are trying to modify their productive system so they can satisfy these individual customer needs and in completely different areas. The fact that balancing production lines is such a big and complex problem and the fact that it may not have one perfect solution but is also a challenging and motivator factor.

1.2 Project Plan

Before starting this dissertation, it's necessary to make a plan of the intended objectives in order to have a guideline so later we can analyse the final work and check if they were all completed. In case they weren't, it's important to find out what could have been done differently and that can be done in the future.

1. Getting comfortable with the simulation framework, getting to know its tools and characteristics and learning how to work with it.
2. Data research and gathering about the company in general, its purpose and goals, its history.
3. By starting with a specific part of the whole facility, we will find more specific aspects about this part, such as timing variables, workstations quantity, operators capability. Without forgetting the role of this part in the whole factory.
4. Conception and proposal of a first simulation model that can recreate the real case.
5. Optimization and implementation of this model by applying the researched values and find changeable scenarios in order to search for a line configuration with better results than the first one.

6. Doing a study of the analytical performances so that we can have some guidelines before producing the results. This way we can more easily evaluate the obtained results
7. Running the simulations and observe the results. Draw conclusions on them about what can be improved in the studied system

1.3 Hosting Institution

The work was developed in INESC TEC - Institute for Systems and Computers Engineering of Porto - Science and Technology. INESC TEC is a private non-profit association that has as associated founders INESC and FEUP.¹ INESC TEC is recognized as a Public Utility Institution and in 2002 it acquired the statute of Associated Laboratory. It was created to act as an interface between the academic world, the world of industry and services and the public administration in Information Technologies, Telecommunications and Electronics (ITT&E), and it is based on the following strategic objectives:

- Develop science and technology that is capable of competing on a national and international level.
- Participate in the technical and scientific training of high-quality human resources to enhance the nation's capacities and encourage modernisation.
- Contribute to the development of the scientific and technological education system, modernising it and helping it to adapt in order to meet the needs of society and the economy.
- Promote and incubate business initiatives in order to improve R&D activities and encourage young researchers to take risks and use their initiative.
- Create a modern Portugal, a well-established economy and a high calibre society by following the objectives that have been outlined.

The R&D made in INESC TEC is divided by twelve R&D centres and one R&D Associate Unit. This project was developed in CESE - Centre for Enterprise Systems Engineering, whose activities are related to Operations Management and Enterprise Information Systems applied to industrial companies and enterprise collaborative networks. The Centre is committed to conduct high quality research with a strong application focus.

The Centre conducts R&D in the following domains: Logistics (supply-chain management systems, logistic systems, transportation, distribution and warehouse systems), Operations Research (optimisation methods, Decision Support Systems) and, the most approached domain by this dissertation, Manufacturing (operations management, advanced information systems for industrial management, planning and control systems, rationalisation and optimisation of manufacturing processes, intelligent automation systems, decision support systems for production management). CESE is based in the following strategic objectives:

¹Information taken from <http://www2.inescporto.pt/>

- Strongly contribute for the performance improvement of industrial companies, through R&D projects, consultancy, technology transfer and training.
- Foster high quality research initiatives in specific areas where the elements of the group are internationally recognised, and start innovative research programmes in new emerging topics.
- Transfer the resulting knowledge and technologies to software houses, equipment producers and industrial companies, through applied research, technology transfer and consulting projects.

1.4 Thesis Structure

The document is divided in 6 chapters, in which the evolution of the work developed for this thesis was documented, that contain the following information:

- The present chapter, Chapter 1 gives a brief introduction of the problem addressed by this thesis. It is also possible to find a description of the institution where the work was made and the motivation for it.
- In Chapter 2 are presented some theoretical concepts about Simulation of Assembly Lines. First there's an explanation of both of these terms with the contextualization of the concept Mass Customization and then the idea of a Job Shop environment. After this, the possible frameworks for the problem's approach were presented and, after choosing the ideal one, it is presented.
- Chapter 3 presents the problem and the case study in particular, including its functioning, the characteristics of its components and what it's intended to obtain for the project.
- Chapter 4 describes how the implementation was done and how the model was built as well as the procedures to recreate the facility as close as possible to the real case. The indicators provided by the framework are then presented and it is done a brief study of the analytical performances, so that the results obtained can be compared to other results.
- Chapter 5 is where the simulations are runned and the results are explained.
- In Chapter 6 the final conclusions about the dissertation are drawn and the possible future works are referred.

Chapter 2

Simulation of Assembly Lines

This chapter will expose the subjects that this dissertation approaches: Simulation and Lines Balancing. We will start on Assembly Lines Balancing (ALB) - how to balance them and the problems that it involves - and then the job-shop concept will be presented (and why it is important to the project). Only then, the term *simulation* will be addressed so that it is easier to understand why simulation helps solving these kind of problems. And finally, after the possible frameworks are presented, the chosen one will be described, where the tools will be indicated and the vantages of it will be pointed out. But first, it is important to do a contextualization of these terms with the concept of *Mass Customization*. That way it easier to understand what is intended with the Lines Balancing Simulation - what should enter the system and then what should come out.

Mass Customization is an antagonistic term that connects two opposite concepts (one referring to mass production and other referring to customized production) reflecting the way in which the reality of production is going these days – to satisfy individual demands by the costumers with mass production efficiency. Major companies are trying as much as they can to provide to their clients this mass customization. In 2010, Coca-Cola invested \$100 million in a plant in its home town of Atlanta, Georgia, to churn out concentrates for its Freestyle soda dispensers, which offer more than 100 drink choices to mix and mash up. [1]

Mass customization is similar as saying "the technologies and systems to deliver goods and services that meet individual customers' needs near "mass production efficiency". [1] The following quote summarizes the actual concept of mass customization and makes it really easy to understand.

"Mass customization refers to a customer co-design process of products and services, which meet the needs of each individual customer with regard to certain product features. All operations are performed within a fixed solution space, characterized by stable but still flexible and responsive processes. As a result, the costs associated with customization allow for a price level that does not imply a switch in an upper market segment." [2]

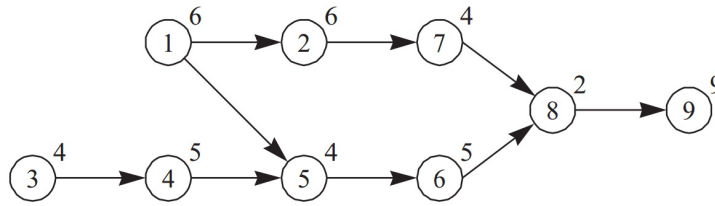


Figure 2.1: ALB basic example, adapted from [4]

Now that we are contextualized about the market's demand, the next sections will introduce more concepts that are essential to easily follow this work.

2.1 Assembly Lines

When we introduce the term of Assembly Lines we are referring to something that has a big importance in most production systems. In fact, it is something that is constantly being target of investigation. The traditional Assembly Line consists in a set of workstations sequentially organised to which the work pieces will go successively, through some sort of transportation - usually a conveyor belt, in order to perform assembly operations during a cycle time. According to Nancy Nardin [3], Henry Ford - the one who first came with the concept of Assembly Lines - had 3 principles for eliminating waste in the assembly process:

1. Place the tools and the operators in the sequence of the operation so that each component part shall travel the least possible distance while in the process of finishing.
2. Use work slides or some other form of carrier so that when a workman completes his operation, he drops the part always in the same place - which place must always be the most convenient place to his hand - and if possible have gravity carry the part to the next workman for his operation.
3. Use sliding assembling lines by which the parts to be assembled are delivered at convenient distances.

2.1.1 Assembly Lines Balancing Problem

In order to have a continuous work flow that allows the tasks to be optimally assigned to the workstations in the assembly line, we have to solve the assembly line balance problem. This is a concept that since his first appearance that academic work has been mainly focused on the core of the problem - the assignment of tasks to the stations. This means that there is a large variety of regarded extensions and that start from a general assembly line balancing.

For better understanding of this concept, a basic example will be presented.

As it is shown in Figure 2.1, which represents a precedence graph, the numbers from 1 - 9 in the nodes represent the activity number. Then, to each activity we can see a node weight associated

representing the task times (in this case we will just say the numbers represent time units). And then, there are arcs representing direct precedence constraints from where we can take paths for indirect precedence constraints.

The arrows in the image ensure that no precedence relationship is violated in the assignment of tasks to the stations. The set of the tasks S_k assigned to a station k will constitute the work load of that station, while the cumulated task time given by 2.1 is called *station time*.

$$t(S_k) = \sum_{j \in S_k} t_j \quad (2.1)$$

In this example, we could come up with a feasible line balance with cycle time $c = 11$ and $m = 5$ stations that is given by the station loads of $S_1 = \{1; 3\}$, $S_2 = \{2; 3\}$, $S_3 = \{5; 6\}$, $S_4 = \{7; 8\}$, $S_5 = \{9\}$.

The cycle time referred above is given by the equation 2.2

$$c = \frac{\text{Total Operation Time}}{\text{Quantity of Production Produced}} \quad (2.2)$$

The long-term effect of balancing decisions means that the objectives need to be carefully chosen considering the strategic goals of the enterprise. However, measuring and predicting the cost of operating a line over months or years and the profits achieved by selling the products assembled is complex and besides it is very likely to involve big errors, it also has big costs associated. A usual surrogate objective consists in maximizing the line utilization which is measured by the line efficiency E , which is given by the equation 2.3

$$E = \frac{W}{N * T} \quad (2.3)$$

W = Total Work Content

N = Number of Workstations

T = Longest Operation

2.1.2 Types of Assembly Lines Balancing

The problem that was described in the previous subsection is known as **Simple Assembly Line Balancing Problem** (SALBP). It is based on a set of limiting assumptions which reduce the complex problem line configuration to the main problem - assigning the tasks to stations. However, the assembly lines balancing in real-world problems will require the observation of a large variety of more technical or organizational aspects (such as the the workstations that are capable of doing what tasks, alternative lines, U- shaped assembly lines or the technical capacities of the operators.

[4] More extensions of SALBP were considered afterwards: SALBP-E - maximizing the line efficiency E , SALBP-1 - minimizing the number m of stations given the cycle time c , SALBP-2 - minimizing c given m and SALBP-F - seeking a feasible solution between m and c .

We may also refer to another kind of Assembly Lines Balancing types depending on the number on the number of models. In the case that the output is only one type and the work pieces are identical, then we call this **Single-Model Assembly Line Balancing** (SALB). Also, if there more than one product being assembled on the same line, but aren't any changes in the setups or significant operation times variations then it can be referred as single-model line as well. SALB Problems focus mainly on minimizing the cycle time for a fixed number of workstations and minimizing this number required.

On the other hand, there is the case where batches are produced, each one containing units of only one or similar models, with intermediate setup operations. And so, a lot sizing problem comes associated with the batch sequencing part. This is a **Multi-Model Assembly Line Balancing** (MtMALB)

Finally, and the most important one since it is going to be the most discussed one over the next pages, we have the **Mixed-Model Assembly Line Balancing** (MMALB). It refers to the production of units of different models in an arbitrarily intermixed sequence. Therefore, the Mixed-Model ALBP brings a sequencing problem - with it, comes the need to find a sequence of models and tasks in order to reduce all the inefficiencies. And although the balancing and the sequencing problem are too complex to be solved simultaneously, because the sequence depends on the short-term model-mix, a horizontal balancing objective is usually used to equalize the work content of stations over all models. [5]

In figure 2.2 we can see a diagram that makes it easier to understand the division of these different types of ALB.

We can also make the distinction between one-sided and two-sided Assembly Lines. Here the difference may be obvious - in the first one, we can only find workstations in one side of the conveyor while in the second one we can find them in both sides, such as Figure 2.3 illustrates.

[7] The advantages of a Two-Sided Assembly Line Structure over an One-Sided one include the possibility of a line length reduction - and consequentially fewer operators needed, reducing the amount of throughput time and it also can reduce material handling, workers movement and set-up time, which otherwise may not be easily eliminated. The Assembly Lines Balancing that will be focus of attention in the next chapter is a Two-Sided one, so we will give more attention to it.

By analysing the possible classifications of production lines balancing problems, we may conclude that we can classify this problem as a Mixed-Model Production, where we want to find a possible solution to reduce the idle time of the transporting robot and increase the machines' percentage of processing time.

In Mixed-Model balancing problems, the cycle time no longer exists so now it is considered the average of the operation times depending on the desired production rate, to what we can call *horizontal balancing* [5]. Associated to this MMALBP, must be referred as well the concept of

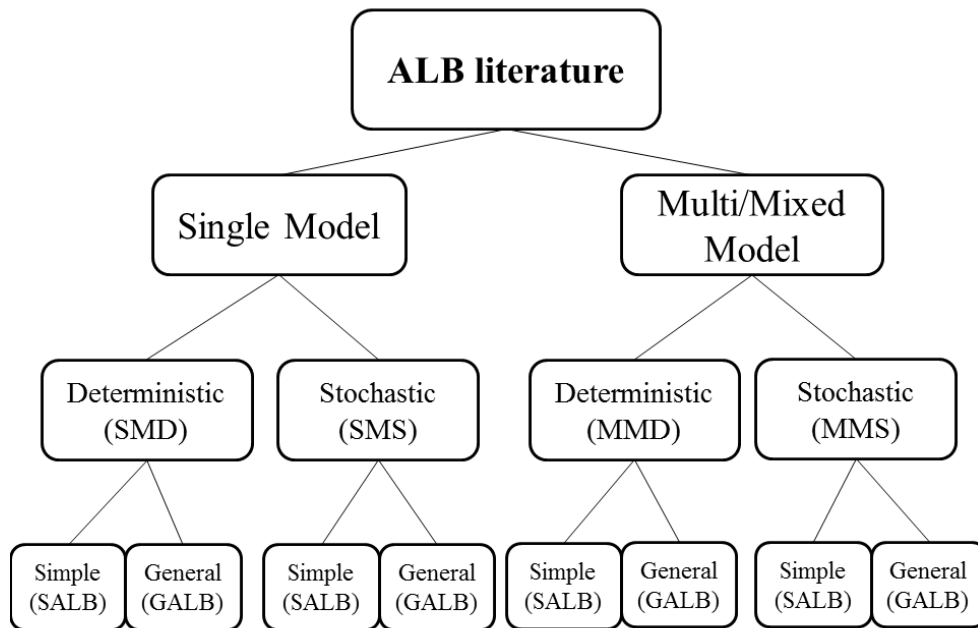


Figure 2.2: Classification of assembly line balancing, adapted from [6]

Mixed-Model Sequencing Problem, because besides finding the best lines balancing, it is also essential to find the best sequence of the entrance with a known objective.

It was proposed to apply the method of Mixed-Model Lines Balancing to a mass customization problem. The methodology consists in an heuristic with several steps, where first it used [8] the method of the equivalent diagram so that later it is used an algorithm that applies an heuristic called RPW - Rank Positional Weight. Through this, it is possible to obtain the initial solution and transform it until we obtain the final solution.

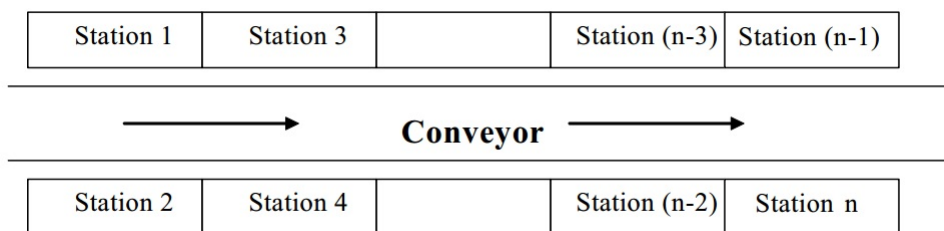


Figure 2.3: Two-Sided Assembly Line Structure, adapted from [7]

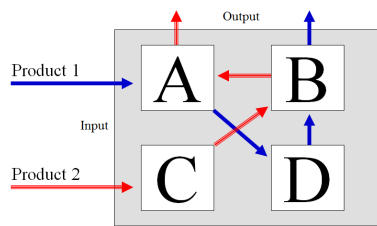


Figure 2.4: Job-Shop Layout, adapted from [9]

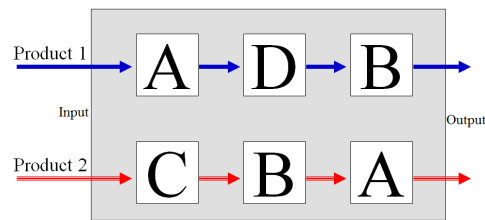


Figure 2.5: Flow-Shop Layout, adapted from [9]

2.2 Job-Shop

If we are referring to the footwear industry, the term *Job-Shop* becomes relevant. It describes the need for practical order release and dispatching mechanisms in an environment where there are sequence dependent setup. It is possible to improve the performance of a production system by systematically determine when jobs should be released to the shop floor - the area where the production is carried out, either by an automated system, workers or a combination of both.

The job-shops machines are aggregated by their type of operations and technological processes involved, meaning that each shop may contain different machines. And, as there's no need to restrain jobs to a single machine, we end up having higher production flexibility in the system.

On the other hand, this type of production line means that a part will encounter other parts competing for the same resources and queues at each machine, as it goes through the shop-floor. Although it isn't a perfect balance between the priorities in the job, if we can study one system long enough, it is possible to gradually find best results for our system.

Job-Shop is ideal when it's intended to build a system with the following characteristics:

- High variety / customized products
- Flexible resource
- Skilled human resources
- Mixed work flows
- High material handling and large volume of inventories
- Long flow time
- Highly structured information system

What, as we will see in the next chapter, are indeed the characteristics of the system that is intended to build. In opposition to Job-Shop, and to understand better its advantages to this project, we will present the term of Flow-Shop. It is characterized by a line layout where resources are arranged according to the sequence of the operations. Usually it requires a duplication in the source's storage. It is more proper to systems that demand high standardization and speed, with

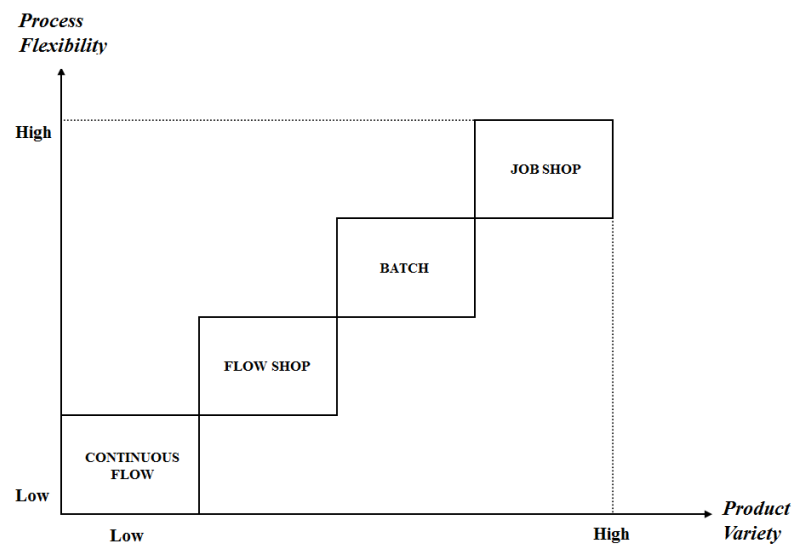


Figure 2.6: The Product-Process Matrix, adapted from [9]

low material handling, high flow unit processing cost and low skilled labour prevent flexibility. Basically is a system directed to mass production when, as presented earlier, we want a system directed to mass customization. Figure 2.4 and figure 2.5 illustrate the differences between these two concepts and make them easier to understand.

For better analysing these two kinds of terms, we may appeal to the use of a Product-Process Matrix, a tool for improving strategic fit. With it, we can compare the product attributes (defined by the market segment as "What is the client willing to pay for it) and the process capabilities to deliver value. This combination is then represented by a covered area in the matrix, as we can see in Figure 2.6. By the same figure it is possible to conclude that in fact a Job-Shop approach is the best one for this project.

2.3 Simulation

Now that the concept of Line Balancing was exposed, it is now possible to explain the reason to use Simulation and how can it be so useful in this field. First it is necessary to understand the concept of simulation and then, after a brief guide on how to run a simulation, we will see the possible frameworks the choose the right one.

2.3.1 Why Use Simulation

When a process or a system is created, the main focus is to develop a set of entities, which are logically related and that contribute to the interest of some application, with the best possible results. [10] And although it is possible to do a simple manual analysis of the system, it becomes important to reproduce its behaviour using a computer simulation as the complexity increases. With the use of worksheets or tables we can have an idea of the system statistics. However, as

these statistics are mostly based on averages, they are far from the actual system behaviour. For this, we mostly refer to a dynamic stochastic simulation. By being dynamic it means that we can closely follow the evolution of the process with detailed information. On the other hand, by being stochastic we assure that we are dealing with a process that has some uncertainty and risk on his behaviour and that's what the simulation is for – to be able to estimate the process' performance. Of course that as the process has this unpredictability, one simulation is never enough and that's why it's needed to run several simulations in order to draw some conclusion about it.

With the creation of a simulation model we can have mathematical recreation of the system random behaviour by using the proper key characteristics. This constitutes a model that can reveal how the system works and instead of only trusting in the average of the outcomes we get, we can better estimate of its real results in order to determine the best configuration without having to experiment with the actual system. Also, there are some systems that can be dangerous to test just for analysing – such as nuclear industry – so it's safer to understand how to configure the system in the best way. Putting this, we can say that simulation is fast, cheaper and safer than the actual implementation, and has the following advantages:

- The evolution of the system can be studied in compressed time rather than in real time, meaning that we can estimate the result of a lot of days in just some seconds. In fact, the speed of the time evolution is only limited by the processor power of the computers running the simulators.
- Instead of having to face the actual costs of implementing a certain configuration, which often needs a significant amount to be applied, a computer simulation will lower what is spent into finding the right configuration.
- Implementing something that we don't quite know how it works, just for experimenting, can bring several risks to the safety of people and their environment.

2.3.2 Types of simulation

[11] Although it is possible to distinguish different simulation types in several distinct ways, the most important would be depending on how state variable change. With this, we can be referring to one of two types of simulation:

- Discrete event simulation
- Continuous event simulation

In the first case, we're implying that the state of the system change in discrete times, and with a finite number of variable changes in some given time. On the second case, if we're talking about a continuous simulation, it normally involves a continuous change of the system state and often requires solving systems of differential equations. Although one of these two types is always predominant, in most cases they're both used in the whole system. For instance, in an orange juice

production, if we observe the production of the juice appears to be in a continuous outflow this work we'll mostly use a discrete event simulation.

2.3.3 Running a Simulation

Some time ago, a simulation was something that was done with a raw method, with very little care about the data involved. However, over the last decades, the simulation evolved from being a raw process to a group of activities that develop the validation, operation and the analysis of a simulation process. With this being said, to create and to operate a simulation in a correct way there are several important steps that can be divided in three stages: [12]

- **Simulation Developing Process**

The first step is to define a problem space. To run a simulation, we need to understand what are objectives and the requirements of the simulations. Then, after making sure we have precise definitions of the hoped results we can define a conceptual model, where there might be more than one appropriate conceptual model, which include the algorithms that represent the system behaviour and a description that holds some requirements for the model production and operation (amount of time, number of employees and equipment asset). Next we need to collect input data, where we gather the input parameters, whose accuracy is the key to get results with the least error. Then, before we actually run the simulation, is essential to verify, validate and accredit the model, because this is where we verify that the software model accurately reflects the conceptual model, validate that the operations done in the simulation actually reflect the functioning of the real world and we accredit the model, meaning we accept the software model for some purpose.

- **Running the Simulation**

Once the simulation developing process is finished we can execute the simulation, where the simulation uses the input data to generate the output data required to answer the problem early defined. Rarely happens that only one simulation is executed, so it's normal that hundreds or even thousands of simulations are done so that the results are closer to the ones that should be expected – particularly in the case of Monte Carlo models.

- **Results Processing**

While the executions are being made and results are coming, it's demanded to collect output data and store it in an organized way, especially because it's usual that there is a large quantity of information. When all the simulations are finished executing, comes the part where the data is analysed to find the answers to the questions the lead to the construction of the simulation. Once these answers are found, is useful to document the results, because they're might be needed to be used to other simulations in order to improve them. The same thing

should happen with the model, since once a model is built it is handy to modify it to other related projects, with new requirements.

2.3.4 Simulation frameworks

In the production lines balancing area, a great amount of research has been done on creating models using simulation. For a problem that includes stochastic elements such as demand and market price, stochastic simulations should be favoured. And so, the following simulation frameworks were considered:

- **Arena** - developed in 2000 by System Modelling and acquired by Rockwell Automation, Arena is a discrete-event simulation where the user builds an experiment by placing models (boxes of different shapes) that can represent either processes or logic, connected by lines that also specify the flow of entities. These models can do specific actions to entities, flow and timing. So it is actually up to the user how to represent each model and entity. Statistical data, such as cycle time and WIP levels, can be recorded and outputted as reports.
- **Flexsim** - this framework was developed by Flexsim Software Products, Inc. The FlexSim family includes both the basic FlexSim simulation software and FlexSim Healthcare Simulation (FlexSim HC). It uses an OpenGL environment to create real-time 3D rendering.
- **Simio** - It's a quite recent simulation tool, developed in 2007 by Dr. C. Dennis Pegden, and it represents a new approach in simulation - object orientation. Modelling is based on describing system's objects and evolution of system behaviour by interaction of these objects.

After some research and experiments on these simulation frameworks, the choice ended up on **Simio**, mainly for two reasons: In the first place it is a tool that was already available in the hosting institution; and then, the object-oriented simulation is an aspect that makes the building of production systems in job-shop environments much easier.

2.3.5 Simio and Object-oriented Simulation

First of all, we can see only by the name **Simio** that we are referring to a **Simulation** modelling software based on **intelligent objects**. The term object may be representing a wide list of things, such a robot, an assembly machine, a doctor, a patient or even a tank or a plane. Normally, an object oriented simulation system isn't very user friendly and by having a lot of code they aren't that easy to analyse. However, Simio is. The possibility of handling an object in 3D animated way makes it pretty accessible. Building a model or an object are very identical activities in Simio. In fact, there is no difference between them. And if you build a model it is by definition an object that can be instantiated into another model. For instance, if you combine two machines and a robot into a model of a work cell, the work cell model is itself an object that can then be instantiated any



Figure 2.7: Example of 3D animation in Simio, adapted from [14]

number of times into other models. The work cell is an object just like the machines and robot are objects.

Simio also allows the programming in languages such as C#, Python or Java in order to construct a set of cooperating objects that are instantiated from classes. These classes are designed using the core principles of abstraction, encapsulation, polymorphism, inheritance and composition. [13]

2.3.5.1 User Accessibility

The amount of tools for this kind of purpose is getting bigger and bigger, so it means that the easiness of access for the user has a big importance. This is one of Simio's biggest strengths because it provides:

- 3D animations on one step, importing 3D objects from Google 3D Warehouse (as you can see in Figure 2.7);
- By importing data from Excel worksheets it allows Simio to process data from real cases and to do simulations with less error. See Figure 2.8;
- Writing own logic function (e.g. priority rules) in many languages (C++, Visual Basic, etc.);
- Creating own intelligent objects and libraries that can be used in other projects;

When Simio is opened, the user can define several models within a project. However, usually a project only has one main model and one entity model (in fact, on creating a New Project, Simio automatically generates this two (where the main model and the entity model are respectively called "Model" and "ModelEntity"). In more advanced projects it is common to add more models as sub-models of the main one. The entity model is used to describe the characteristics of the

entities and their behaviour towards different events (although the default entity created by Simio doesn't have any explicit behaviour so it just goes "without reaction" through the model. As we it is possible to see in one of the example models provided by Simio - an emergency department - it is possible to distinguish different entities that represent either patients, nurses or doctors.

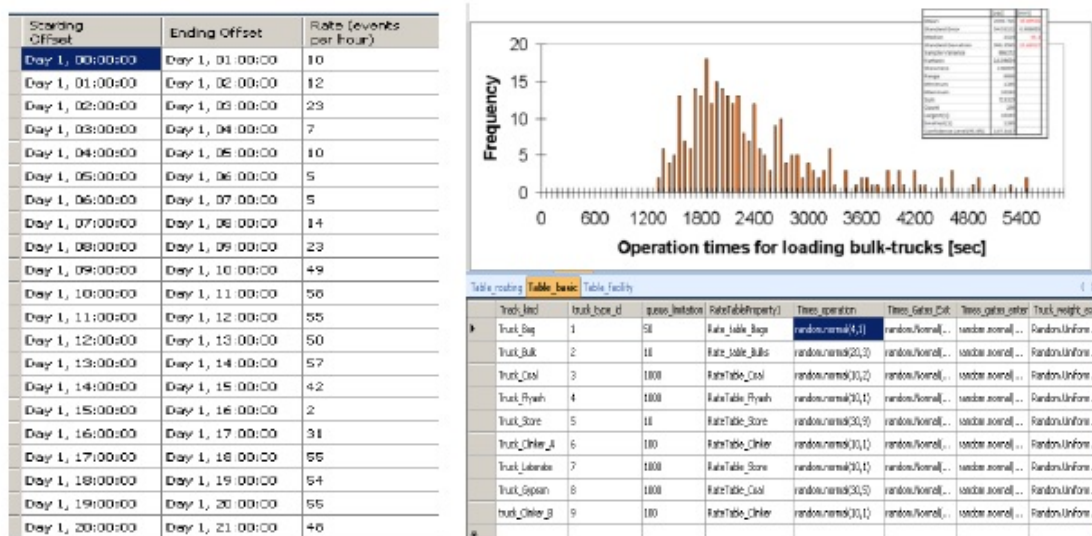


Figure 2.8: Example of an Excel Import to Simio, adapted from [14]

2.3.5.2 Creating Intelligent Objects Through Processes

As previously said, this kind of tools are good for quickly building models. The user just has to drag the objects wherever he wants, setting their properties and the model should be ready to run. And this kind of approach should be able to work in the simplest models. However, although this is a rapid solution for building models, it lacks in flexibility. It's really difficult (almost impossible) to design a set of objects that work in all situations across multiple and in so different areas (just imagine the variety of areas that Simio can simulate - since an Shoe assembly line, to a football match (the viewers that enter the stadium and all the rest).

Through the Simio Standard Library it is possible to address this problem with the concept of add-on processes. An add-on process is a small piece of logic that can be inserted into the Standard Library objects at selected points to perform some custom logic which can be used to change several properties. The processes are created as graphical flowcharts without the need for programming. Hence Simio combines the benefits of object based modeling with the power and flexibility of graphical process modeling.

Process logic can be inserted into an object on an instance by instance basis, without modifying or changing the main object definition. For example one Server instance might incorporate process logic to seize and move a secondary resource during processing, while another instance for the same Server definition incorporates special logic to calculate the processing time based on a learning curve, and a third instance of the same Server incorporates special process logic

for modeling a complex repair process. All these instances share the same Server definition but customize each instance as needed.

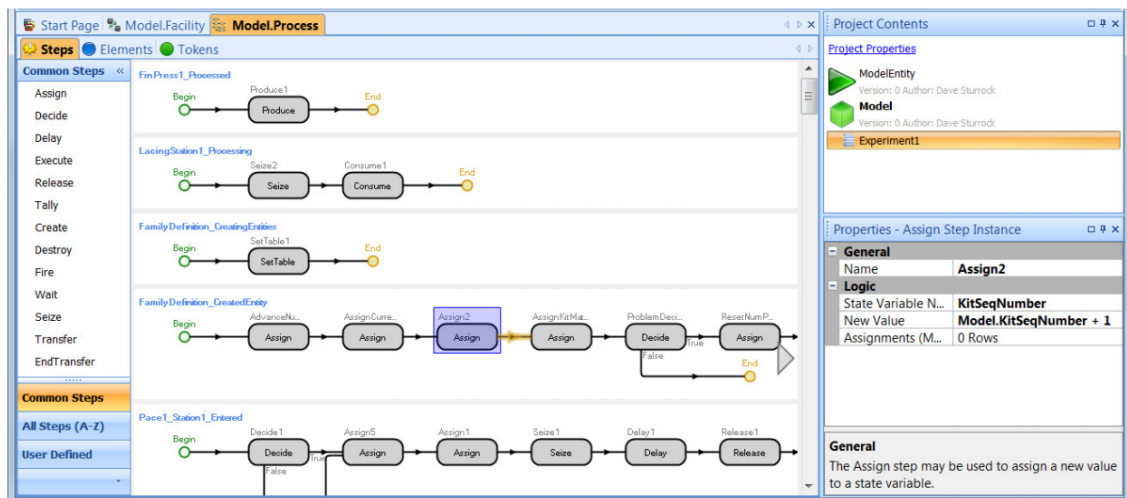


Figure 2.9: An example of a Simio process, adapted from [13]

2.3.5.3 Running the model

When the model is built, and with all the properties correctly set according to what we want, the next step is to run it. In Simio we can run it in two ways - we can either execute it in the *interactive mode* or in the *experimentation mode*.

2.3.5.3.1 Interactive Mode

This mode allows the user to watch the animated model in 3D and view in "real time" (the one considered by Simio, which is normally way faster than the actual real time) both the dynamic charts and the plots that summarize the system behaviour. With this it is possible to verify and validate the model and also getting a basic notion on how the system is performing (which is made much easier by Simio's animations). Then, once the model is considered to be validated, it is common to determine specific scenarios to test with the model, where some individual properties are changed to understand what is the difference in the model's performance when facing this changes. Normally, when running specific scenarios only individual properties of the objects are changed so it becomes irrelevant to observe the animation, because at the user's eyes there wouldn't be a significant difference. Instead, the focus should be in replicating each scenario to find the underlying differences in the system and to reach statistically valid conclusions from the model.

2.3.5.3.2 Experimentation Mode

On the other hand we have the experimentation mode, in which it is possible to run several simulations with different properties so we can see how the system reacts with them

In the experimentation model we can define one or more properties on the model that we can change to see the impact on the system performance. These properties, exposed in the experiment as *Controls*, might be used to vary things like conveyor speeds, the number of operators available, or the decision rule for selection the next customer to process. These model properties are then referenced by one or more objects in the model, that will make the difference in the produced statistic. You may also add *Responses*. These would generally be your *Key Performance Indicators* on which you make the primary decision about if your scenario may be considered a good one. Sometimes it is convenient to sort dynamically the columns created in the statistic (for instance to display the highest profit scenarios first) which is possible to do. The user can also add *Constraints* that will automatically be applied before or after a run to prevent running or to later discard a scenario that violates an input or output constraint. When the user runs an experiment, it takes full advantage of all processors available. Figure 2.10 illustrates several scenarios being executed simultaneously in an inexpensive quad-core machine.

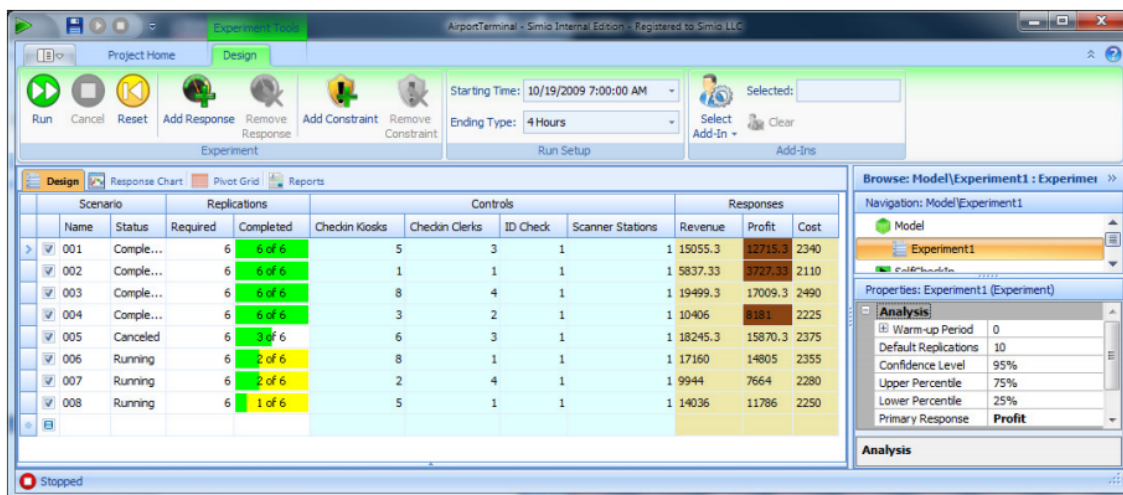


Figure 2.10: Running an experimentation with different scenarios, adapted from [13]

2.3.5.4 Summary - Why Simio?

Simio is a new modeling framework based on the main principles of object-oriented modelling. It provides the user unique features - that, as we will see in chapter 4 and 5, will be of a big importance - that expresses their presence in the following ways [14]:

1. Simio's framework is a graphical object-oriented modelling tool while most of the other ones are constituted simply by a set of classes in an object-oriented programming language that are useful for simulation modelling. The graphical modelling framework of Simio fully supports the core principles of object oriented modelling without requiring programming skills to add new objects to the system.
2. The Simio framework isn't specific to any area of business, and allows objects to be built that support many different application fields. The process modelling features in Simio

make it possible to create new objects with complex behavior - and if it isn't enough, the user can create his own object using coding languages such as C#, Java or Python

3. It supports multiple modelling paradigms as well as both discrete and continuous systems (although the most common is the discrete - and what will be used in this project), an event, process, object, and agent modeling view.
4. It provides specialized features to directly support applications in emulation and finite capacity scheduling that fully leverage the general modelling capabilities of Simio.

Chapter 3

High Speed Shoe Factory

Now that all the concepts and tools used in the project were presented, the next step is to present the actual project.

As previously said, although this thesis is based on a real case, it is intended that it is also directed to other small and medium companies that want to expand their online sales and that follow the same train of thought as the company in focus.

The case study is included in a wider scheme - the High Speed Shoe Factory. It is integrated in the National Strategic Reference Framework (QREN), has a system of incentives for research and technological development in companies that encourages the HSSF project. HSSF is being developed by a small group of companies and research institutes that have applied to their implementation, including the INESC TEC, where the author of this thesis was involved.

Its main objective is to conceive, develop and implement a new model of a shoe factory that can quickly answer to different requests in approximately 24 hours (maximum 2 days) and that is oriented to unitary production, in which it's produced one pair at a time, and that, without any stocks, is capable of responding to internet sales, to small orders and also to the reposition of products in the stores, while it also produces samples for new products and tests of new products for new collections. The project focuses on the modernization of assembly lines and replaces the traditional section of cutting, stitching, assembly and finish, by making them more flexible and prepared for producing several types of footwear simultaneously. Also, one of the other main objectives of this project is to provide to the client a big virtual shop, with a lot of interactivity and that brings him a customized environment, where there's advertisements directed to him and where he can make a custom purchase.

3.1 The Stitching Line

Obviously, the creation of a pair of shoes goes through several stages, since the feedstock arrives the factory until the output is 100% completed. The factory we will be focusing divides into 3 different stages:

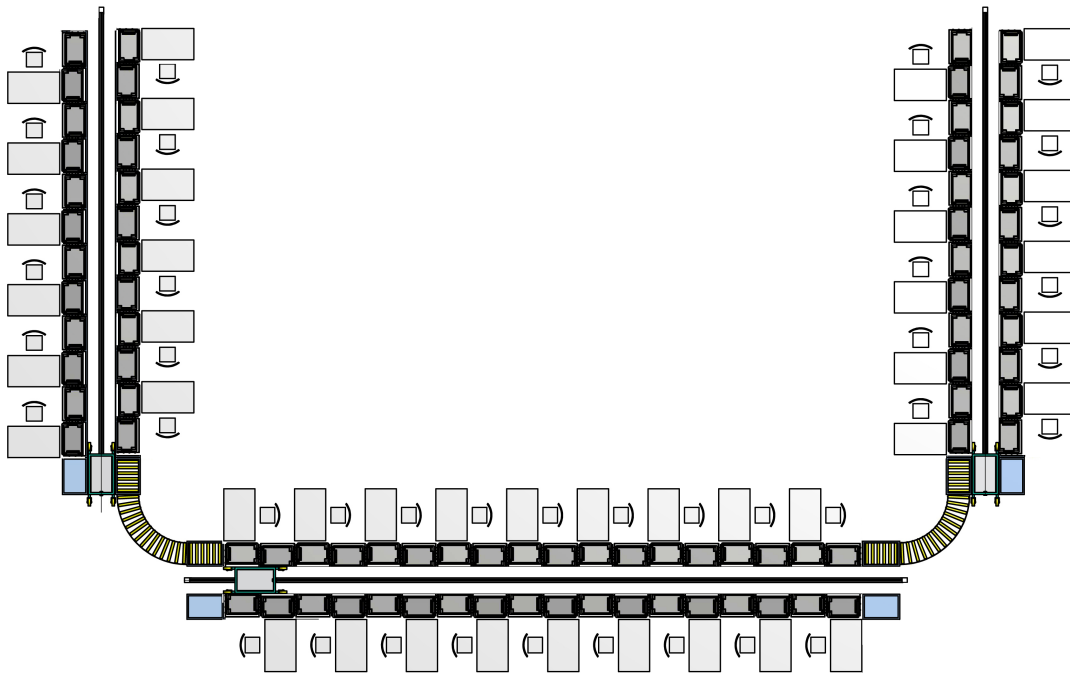


Figure 3.1: U-Shaped Lines

1. **Cutting** - This is the first stage of the shoe production. It is where the synthetic components will be cut into the several parts that constitute a shoe, with the help of operators or automatic systems of laser cutting or water jets, so that it has full efficiency.
2. **Stitching** - Once the parts are all cut in individual segments the next stage make sure that they are stitched and glued together, forming the upper. This is probably the most complex stage and the most difficult line to balance because each part must go through a lot of different operations and in different machines along the line.
3. **Assembly** - The final stage of the shoe production is the assembly where the upper formed in the previous stage is added to the shoe sole.

For this dissertation we will focus in the stitching stage, for having high complexity and difficulty in balancing problems. Figure 3.1 shows the production line of the stitching stage. We will consider it divided in 3 different parts, disposed in a U-Shaped configuration, due to the fact that each part has a robot that transports the parts along the respective line. Once a part is finished doing all the operations in that line, it follows to the end of it and the next stop should be either the next part of the line or the next stage of operation. This should be only to situate this work in the general configuration of the factory. Although there's a lot more in the factory besides this production line, we will only be focusing in one of its parts. Each part will focus in a different type of operations. However, the way they move along the lines it's pretty similar - as one part enters the line, it is transported through the workstations to get the operations done and then it exits the line. In this case we will be focusing in the one on the right (as seen in the figure 3.1).

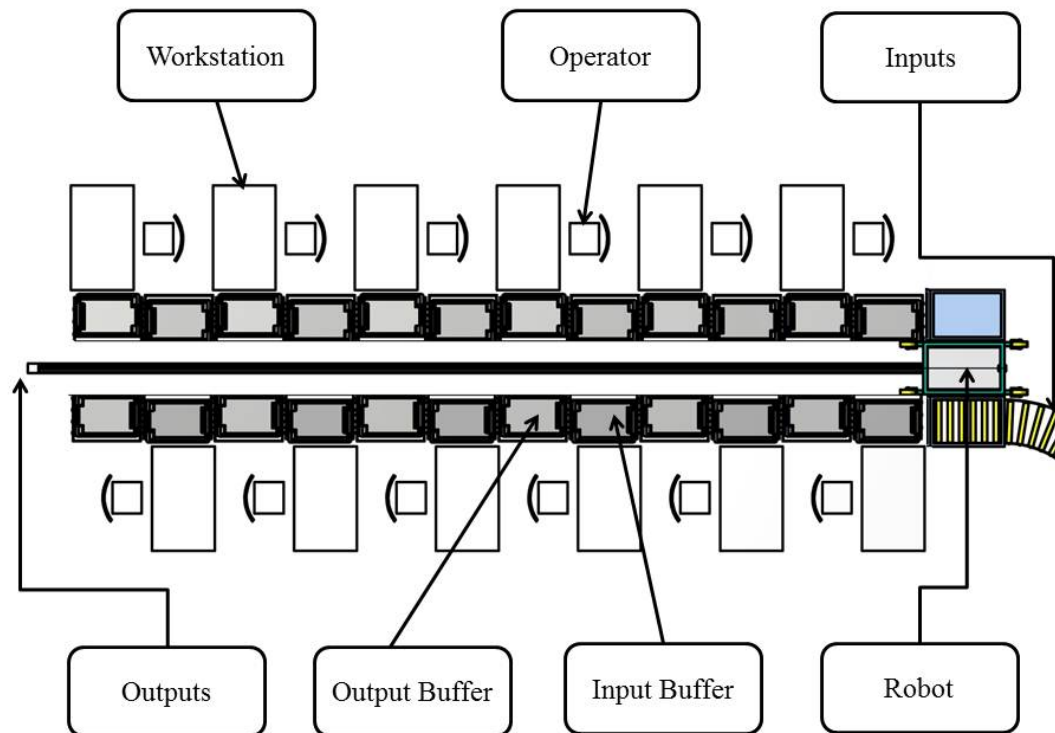


Figure 3.2: Production line in focus

The core simulation done in the next pages will focus in a sector of the whole factory - an assembly line that is constituted by several *workstations* (each one with an operator), *inputs* (pairs of shoes that have a list of operations that will be done in this line), *outputs* (pairs of shoes that are either completed or at least ready to go to the next assembly line) and, probably the most relevant, the robot that will transport the entities through the assembly line. Figure 3.2 gives an illustrative example of the line. In the same figure, we can see the workstation placed along the two sides of the assembly line. This way, gives us a easier way to understand how the line works.

Inputs

In the real line, the inputs in this line are pair of shoes that need to go through stitching operations. As we have seen above, this line follows a Job Shop concept, which means that each pair will have different sequences and so it will probably follow a different path from the previous one and from the next one.

When a new job enters the line, the information about it will be received and processed, so that the line can get to know this job. The information about it includes:

- **Job Type** - First of all, the line needs to know what type of job has just arrived. There are dozens of different kinds and it is essential to distinguish each one, although in this work we

will consider 6 different models, in order to easier analyse the results. The Unique Identifier that will guarantee to be unique among the other types is its Type Name.

- **Operations Sequence** - Each job need to do a lot of operations since it's entrance in the line until its end. So that all the operations are done and in its correct sequence, the jobs will have associated their Operations Sequence. They're essential and every time a new Job Type is created, it is needed to be created as well a new operations sequence. They're developed by a responsible that, experimentally, builds a new shoe, registering its operations done and the respective time. It's easy to understand that they have a great importance, not only because they're used as a guide by the operators, but they also are an input to the transporting robot to show him in which workstation the part will be operated.
- **Operations Machines** - This information transmits in which machines the operations can be done. As it happens a lot that a machine is already occupied with another job, and as there are machines that can to a lot of different jobs, there's a need to find an alternative machine when the first choice is unavailable, in order to avoid a unnecessary bottleneck.
- **Process Time** - Finally, the process time. It varies not only with each operation but also with each job (even if the operation is the same, it can take longer in some jobs than others)

Workstations

The sequential operations that need to be done by the jobs are processed by the workstations. As evidenced by the figure 3.2, we will have 12 workstation in this line we are working (6 on each side of the line). Some of them will do similar operations, so the robot can choose between these two machines.

Operators

All the operations need to be done by an operator, so we will find one worker assigned to each station. Although in the real case it happens that sometimes there aren't 12 operators working due to an eventual high idle time by some machines, and so one operator is enough to operate on both, we will consider that there are always 12 operators working (meaning that the idle time of each worker is the same as his respective machine).

Transporting Robot

This is the most important part of the line. It is what makes sure that the line is always flowing, so if it happens some failure to the robot, all the line is compromised and there isn't any more work, as the parts can't move neither to the next workstation nor the end of the line. At first we will give the robot a FIFO priority - the first part demanding the transport from the robot is the first one to be served. Then we will change this priority in order to see if it is the best approach to this problem.

Outputs

As soon as the parts are finished doing all the operations in this line, the stitching stage of the shoes production is over, so the transporting robot will move them to the end of the line where they will next be transported to the next stage (that is the final one - the assembly stage).

Input and Output Buffers

Each workstation will have an input buffer and an output buffer that will allow increasing the flow of the system. This way, when there's a part processing in a workstation there's the possibility of putting another part in the input buffer so as soon as the part is finished processing it moves to the output buffer and the other can start processing right away.

3.2 Comments on the Project

With this work it is pretended the development of a system where, by changing some properties of the components exposed above or by changing some sequence (in the operations routings or in the machines placement), it can process the input data that it will receive, process it and, through several experimentations, obtain a great quantity of simulations of the system whose results, after being treated, give clear information on how the line should be implemented.

It is intended that this work that is being developed here fits into the "bigger picture", i.e. that the manager proposes a solution for some line of work and then this simulation should recreate it, in order to see if it was a good solution. This means that these simulations should go along and work synchronously with the manager that is proposing this.

Chapter 4

Approach and implementation

In chapter 3, the previous one, we have understood the project and the project context has been explained as well as the intended goal. The next step is first to find the right approach to the problem. It goes through a simulation process where the real system is recreated, including the parts routings, the respective processing times as well as the transporting robot algorithm.

4.1 Input data



Figure 4.1: The 6 entities that represent 6 different model of shoes

In Simio, the parts are represented by Entities. Each entity will have its own routing sequence, its own processing times. We will call them by the name of the model of shoes produced. In this case only 6 entities will be represented, in order to be easier to understand the system. In reality there are dozens of models being produced each seasonal period. In figure 4.1 we can see the entities represented by the boxes, each one with a stripe of an individual colour so that during the running of the simulation it is easier to distinguish them.

Besides the entities, we have the sequence of operations that each entity will follow. We can see them in table 4.1, number of the operation of each entity associated with its respective machine and process time. The line's algorithm will assure that all the sequences are done only one time and in their proper sequence. Also, if there are two consecutive operations that can operate in the same machine, the operations will be done consecutively (so the robot doesn't come and pick the part and then put it again in the same machine).

Table 4.1: Sequence of Operations

Op.	Wrapper		Humpry		Sakie		Foxy		Jeffreson		Frazer	
	M.	P.T.	M.	P.T.	M.	P.T.	M.	P.T.	M.	P.T.	M.	P.T.
1	A	0,40	A	0,90	A	0,80	A	1,00	A	0,80	A	1,80
2	C	0,65	A	0,65	A	0,70	A	0,75	A	0,35	A	0,40
3	A	0,70	A	0,55	A	1,00	A	0,35	B	0,40	A	1,80
4	A	1,58	A	1,20	C	0,60	A	4,20	A	1,60	A	0,90
5	C	0,90	A	0,75	A	0,90	C	3,00	A	1,80	C	2,20
6	A	0,65	C	0,75	A	1,00	A	1,50	B	2,20	A	2,23
7	A	1,80	B	0,60	A	1,60	A	2,80	B	0,75	A	5,50
8	A	4,20	C	0,55	A	1,90	C	6,20	B	1,45	C	1,43
9	D	1,40	B	1,00	C	0,96	A	2,50	D	0,55	A	1,13
10	B	0,95	B	1,00	D	0,55	C	2,75	C	6,80	A	0,45

Op. - Operation Number

M. - ID of the machine where the part can be operated

P.T. - Process time of this operation in the machine in minutes

Also, there is interarrival-time, the variable related with the time interval between two successive entity arrivals to the line. In reality it will depend on the frequency of the previous lines, but here we will approximate this time to an ideal value that the robot never has idle time (or not idle time due to the arrival of new entities). This time will be defined in the next pages.

4.2 Building the model

Now that we know the input variables, the next step is to build the model that will process them and simulate the system as it behaves in the real case.

4.2.1 Objects

As explained in chapter 2, Simio's is based in an object-oriented simulation, what means that the first step to create the objects. The typical start in the creation of a new Simio model is the placement of a Source, a Workstation, a Sink and Paths connecting them. This model started the same way but then, of course, a lot of objects where added. Around 60 objects were used in the model (and they can be seen in figure 4.2) which can be divided in the following categories:

- **Source** - This object "creates" new entities. In reality there isn't one source, the parts just come as they're ready from the previous lines. In the source properties it is possible to define

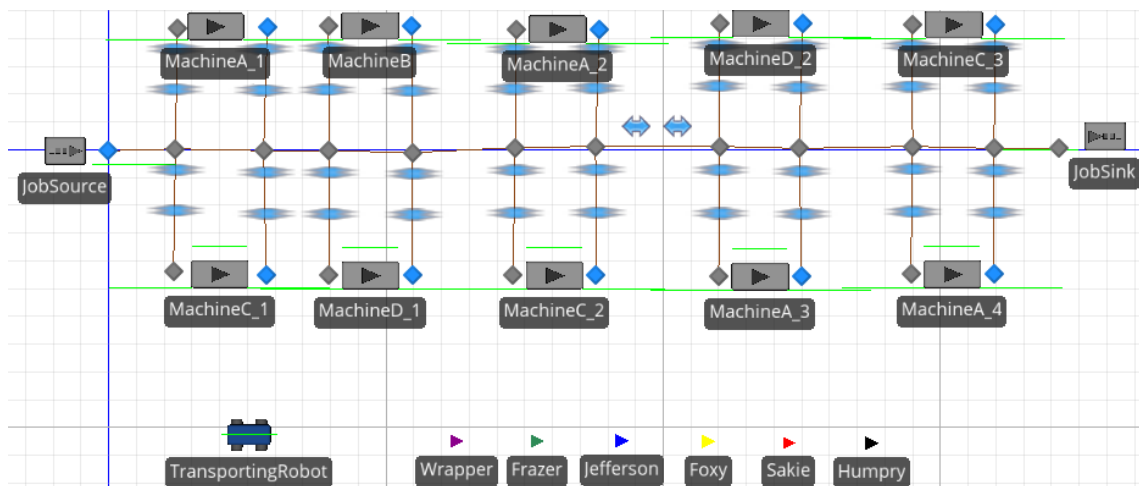


Figure 4.2: The facility recreation, that include all the objects

both the interarrival time between entities and which entity is going to be created. This last property was defined as being random (equally distributed) between the 6 entities. It was also stated that all the entities that leave the source must be transported by the vehicle (the Transported Robot).

- Workstations** - It is where the entities' operations are executed. They get the information about the entities' processing time. All of them have an input and an output buffer, both with capacity of one entity. This means that when a new part arrives the workstation, and if there's already a part in process, it is placed in the input buffer, so as soon as the part in process finishes its operations and is moved to the output buffer, a new processing can begin right away.
- Entities** - As explained before, they represent the parts. Each one has their Type name with the associated processing time and the operations sequence.
- Vehicle** - Or the transporting robot. It moves the entities from the source to the workstations, then from one workstation to other, following the correct sequence. The vehicle speed is changeable, so we will define it with a speed of 2 m/s .
- Paths** - The paths are the links that connect the other objects. They define the route that the robot must follow. Without paths, the transporting robot would just move freely in the facility, when in reality the robot follows a straight line. The paths were defined as being bidirectional, so that the vehicle can move back and front them.
- Basic Nodes** - they're simple nodes that make connection between links, in this case the paths. Without them, it would be necessary to create a lot more paths.

4.2.2 User-defined Tables

In order to build the model with some complexity, the user cannot do it using only the components standard library provided by Simio. There's a need to create auxiliary global variables, states, properties and tables. Here are variables created to help the creation of the simulation model:

Sequence Table As we can see in Figure 4.3, it relates information about the Type Name, of both the operations actual and next, it associates the node to where it should go next, a variable that says if it is available and its respective processing time.

	Type	Op Actual	Op Next	Next Node	Buffer Availability	ProcessTime (Minutes)
1	Wrapper	0	1	Input@MaqA1	MaqA1Available	.4
2	Wrapper	0	1	Input@MaqA2	MaqA2Available	.4
3	Wrapper	0	1	Input@MaqA3	MaqA3Available	.4
4	Wrapper	0	1	Input@MaqA4	MaqA4Available	.4
5	Wrapper	0	0	Input@Storage	Sinke 1	0.0
6	Wrapper	1	2	Input@MaqC1	MaqC1Available	.65
7	Wrapper	1	2	Input@MaqC2	MaqC2Available	.65
8	Wrapper	1	2	Input@MaqC3	MaqC3Available	.65
9	Wrapper	1	1	Input@Storage	Sinke 1	0.0
10	Wrapper	2	3	Input@MaqA1	MaqA1Available	2.28
11	Wrapper	2	3	Input@MaqA2	MaqA2Available	2.28
12	Wrapper	2	3	Input@MaqA3	MaqA3Available	2.28
13	Wrapper	2	3	Input@MaqA4	MaqA4Available	2.28
14	Wrapper	2	2	Input@Storage	Sinke 1	0.0
15	Wrapper	3	4	Input@MaqC1	MaqC1Available	.9
16	Wrapper	3	4	Input@MaqC2	MaqC2Available	.9
17	Wrapper	3	4	Input@MaqC3	MaqC3Available	.9
18	Wrapper	3	3	Input@Storage	Sinke 1	0.0
19	Wrapper	4	5	Input@MaqA1	MaqA1Available	6
20	Wrapper	4	5	Input@MaqA2	MaqA2Available	6

Figure 4.3: A sample of the Sequence Table

MixTable It is shown in Figure 4.4, and it shows associated the type of entity and the respective Mix number. This number is, as we will see in the next pages, an indicator of how many parts of each type we want to put going out the source. Mix doesn't say the precise amount of parts that come, but is only a sort of priority when facing other types.

4.2.3 Processes

A process in Simio is composed by *Steps*, *Elements* and *Tokens*. It is a sequence of actions that change the state of the model. Tokens flow through a process executing steps that can alter the state of one or more Elements. All the process used in this work are Add-On Processes. This means that the processes are executed by some pre-defined triggers.

4.2.3.1 MachineDecide

This is a process that will decide to where the entity should be moved next. It's triggered whenever an entity exits the object. In figures 4.6 and 4.7 we can respectively see the trigger being applied to

Entity Types		Op Sequence
	Types	Mix
1	Wrapper	30
2	Humpry	10
3	Sakie	20
4	Foxy	32
5	Jefferson	25
6	Frazer	30
▶		

Figure 4.4: A sample of the Mix Table

the Source and to one of the workstations. This means that whenever an entity exits the object it's created a Token associated with this entity. Now, as we can see in figure 4.5, the Token is created in the green circle in the left. Then it goes through the following steps:

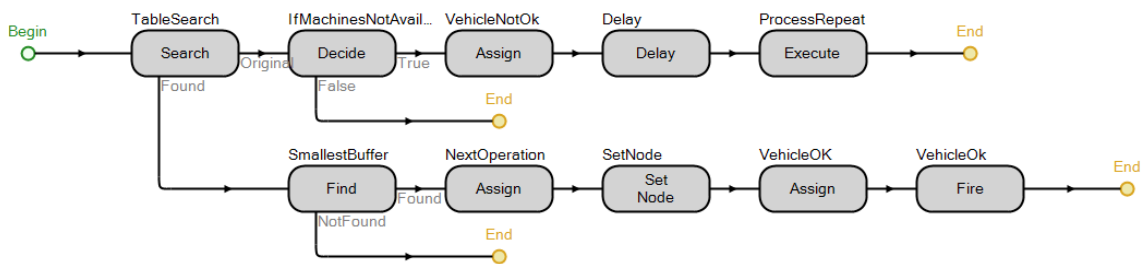


Figure 4.5: The Process that will decide to which machine to send the entity

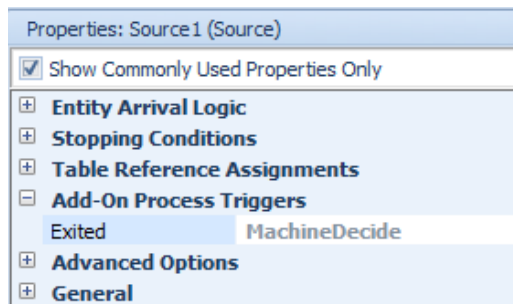


Figure 4.6: Source Machine Decide Trigger

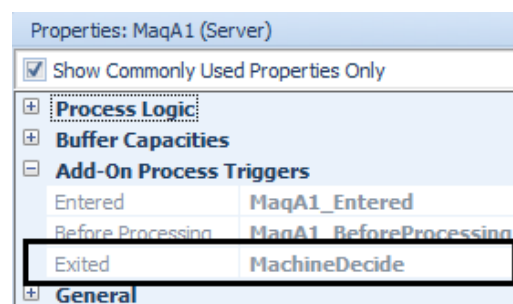


Figure 4.7: Machine Decide Server

- TableSearch** - This step runs through the sequence table (that will be explained ahead) and will find the machines where the entity can move next. This operation is done by match conditions. First it compares the Type Name of the Entity associated in the process with the Type Names in the table (so we will get between 15-20 results, that represent all the possible moves by that type). Then it compares the operation in progress with the operation row in the table (that should narrow the results to about 3-4 results). Finally, it is checked if

the input buffer is available (otherwise the robot would be transporting a part to a full input buffer so it would be blocked while waiting for it to be free). This Search step forwards the Found results to one side and the Original to the other one. It also stores the amount of results found into a variable *Machines Available*.

Original

- **Decide:** It compares the variable *Machines Available* (the one in which is saved the number of results found in the previous step TableSearch) with 0. And if it is false, it means that search found possible results so this token can end the process with no more actions. Otherwise, if there weren't any results available, the token will follow to the next step.

- **Assign:** This step assigns the global variable VehicleOK to 0 so the vehicle doesn't come to pick any entity to the source (otherwise the Transporting Robot would come to pick the part and then stop because it would have where to forward it).

- **Delay:** The delay step holds the token in it for the defined time and then it goes. Here, the delayed time defined was 10 seconds and its purpose is to prevent the process of going in infinite cycle.

- **Execute:** This last step is done in order to repeat the process. It will run the process again and check if, after the delay of 10 seconds there's a machine already available. If not it will run again and again, until there's a machine available.

Found

- **Find:** This step is going to go through the results found in search and find the workstation with less parts on it (considering Input Buffer, Processing and Output Buffer) so that the work is equally divided by all the machines.

- **Assign:** It increments the operation, so the next time this entity runs this process, it will move to the next operation, so that it doesn't neither repeat an operation nor skip it.

- **Set Node:** Now that we know the workstation where the entity should go next, this steps makes sure it goes there.

- **Assign:** Finally, now that the node where the entity should go is set, it is possible to give the vehicle permission to go pick the part.

- **Fire:** The purpose of this step is in case that an entity has already requested the vehicle when it couldn't transport it already. That means that the process ProcessVehicle is running (and should be on the step Wait, so this step let this process finish so the vehicle can go and pick the entity.

4.2.3.2 VehicleOK

This process will make sure that the Transporting Robot has permission to go pick up an entity. It will trigger everytime that an entity demands the service of the vehicle and it should assure that if

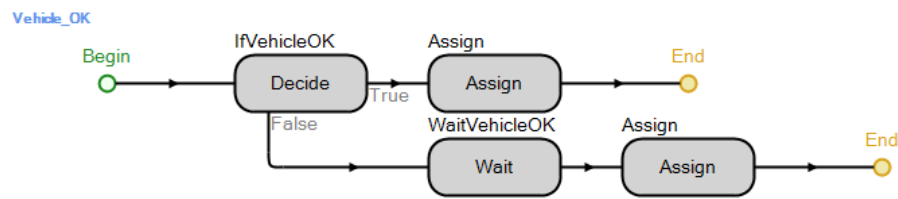


Figure 4.8: Vehicle OK process

this entity still hasn't been assigned a node where it should go, then the vehicle shouldn't go pick it up (and so, it should first pick the next entity available and then it will return to this). The figure 4.8 illustrates the functioning of the process, and here how they work:

- **Decide** - This step will check the variable `VehicleOK` that, as we have seen in the previous process, will have the information about if the vehicle can pick up the entity. If this entity has already been assigned a node where it should move next, then this step returns the value of *true*. On the other hand, if it doesn't have knowledge about the next node, it will return *false*.

True

- **Assign**: As the vehicle can transport the part, without any more actions of the delay or other types of action, then it will just assign the previous value and that should be it for this process

False

- **Wait**: This step will be active if the Transporting Robot doesn't have permission to go pick up the entity, so the natural thing to do is to wait until it is free to do it. It is important to highlight that this wait step doesn't mean that the whole system is stopped waiting (neither is the vehicle). It only means that the token associated with this entity is waiting so, while the **Machine Decide** process is finding a node to assign the entity, the token will be in this step. When, in the other process, the Fire step is activated, the vehicle will now have permission to continue. So when it finishes doing its current operation, will move and to this operation.

- **Assign**: The purpose of this Assign step is exactly the same as the one above.

4.2.3.3 MachinesAvailable

This is actually a pair of processes that is associated with each workstation and their purpose is to control the inflows and outflows of the input buffer of the workstations. They only run properly when combined with each other and their configuration is pretty simple:

The first process is triggered everytime an entity enters the input buffer of a workstation. And as we can see in Figure 4.9, this process has only one step - Assign. This step will change the value of the variable `MachineXAvailable` (where "X" is the name of the machine correspondent to

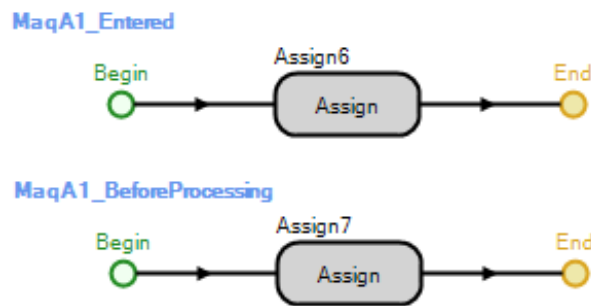


Figure 4.9: Machine Available Process

the one the entity just entered) to the value 0 so that the process **MachineDecide** use this to know if the buffer is available.

On the other hand, we have the process that does the opposite action. It is triggered exactly before the entity starts processing, which means that the input buffer is already free so, as the previous process, this one will have only one step as well - Assign. It will change the value of the variable *MachineXAvailable* back to 1, so that the process **MachineDecide** knows that the input buffer is free to transport entities.

In figure 4.10 we can see how these process triggers are applied to each machine. As it must be assigned a different variable to each workstation, it necessary to create as well a pair of processes to each machine. The Figure 4.10 only shows the triggers for the machine A-1, but in the program there are other triggers and other processes to the rest of the machines.

Add-On Process Triggers	
Entered	MaqA1_Entered
Before Processing	MaqA1_BeforeProcessing
Exited	MachineDecide

Figure 4.10: Trigger for MachineA1

4.3 Indicators

Simio provides indicators that can be placed along the facility and they transmit information to user about how the system is evolving. These statistics can be seen in real-time so they serve only to the user to see the evolution or, in other words, they aren't useful for producing the final results, because they wouldn't be so precise. For the actual results, a lot of simulations need to be executed. The following indicators were placed in the facility:

- **Number of entities produced** - This indicator shows the number of entities that were created in the source, so it actually intends to recreate the amount of parts that arrive the line from the previous lines (as shown in Figure 4.11).

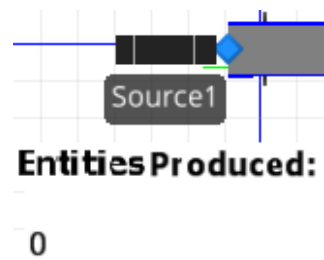


Figure 4.11: Entities Produced

- **Number of entities arrived** - On the other hand, this indicator will represent the amount of entities received by the sink, recreating the number of parts that finished doing all the operations in the line (as shown in Figure 4.12). With these two indicators we can calculate the Work in Progress (WIP), that is known by being an inventory between the start and end points of a product routing or, in other words, where all the products between but not including the ending stock points. The WIP is given by:

$$WIP = NEP - NEA \quad (4.1)$$

where NEP is the number of entities produced and NEA is the number of entities arrived.

- **Machines' Status Pie** - This is a way for the user to observe the performance of one of the machines. There is one status pie chart for each machine and they are very useful because they not only give information about if the machine's performance is good, but also if the blocked time increases too much, that means that there was a deadlock and the user can find a way to prevent it right away. We can see the status pie in figure 4.13 and it is possible to observe that there are a lot of subtitles for the different states that it may have. However, the most common are *Idle*, *Processing* and *Blocked*.
- **Vehicle's Status Pie** - This one has the same functioning as the previous one, but it is indicated for the vehicle. It is also important, because by observing it, we can tell if there could be more parts in the system (if the idle time is too big). For this system, it is ideal that the vehicle is the bottleneck. That means that it should always be processing and never waiting for some parts to arrive or to finishing process, so this status pie should be mostly green.

4.4 Deadlocks

One of the main problems in the implementation was the appearance of deadlocks. They normally happened when the vehicle put parts in all spaces available, so there wasn't any movement available to do. For instance, a entity Wrapper needs to move to a machine A but they are all full. And then, an entity Foxie on machine A wants to C, but it can't because the machines are all full. A



Figure 4.12: Entities Arrived

deadlock situation can be seen in Figure 4.15, where it is possible to see that all the input buffers are full and, as there are 3 parts in each machine (corresponding to the input buffer, processing and output buffer), it can't receive any more parts.

In this situation, the robot can't do anything else. As this problem appeared during the implementation, we needed to resolve this. Consequentially, two ways to resolve it:

- **Interarrival-time** This one was, due to lack of time for this work, the one that was implemented (as we are going to see while running the simulations). It is easier to implement but is not so effective and it doesn't produce as good results as it could.

It consists in the increasing of the interarrival-time so that the frequency of arrival is very close to the time that it take to one piece to complete all the operations. This way, there won't be any overload of pieces in the system and it can continue to flow normally.

- **Storage buffer** This one should be completely effective, but it wasn't possible to implement it, mainly because of the short time and to the lack of experience in the framework used. It consists in a storage buffer (with an infinite capacity) to where the transporting robot would transport the entities in case that the respective destination is full and there entities would wait until this would be free. In Figure 4.16 we can see an example of what the system would look like, with a storage implemented.

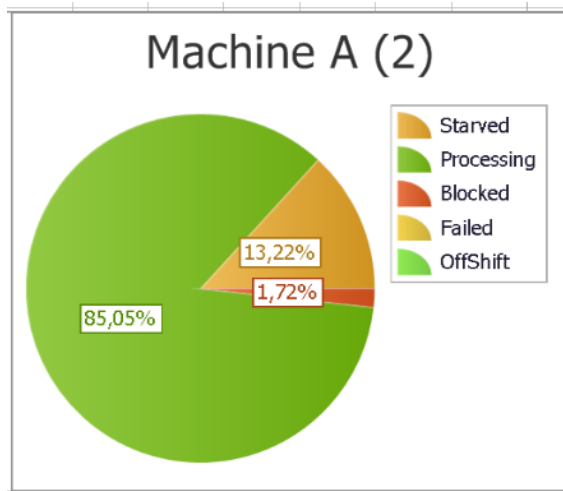


Figure 4.13: Machine's Status Pie

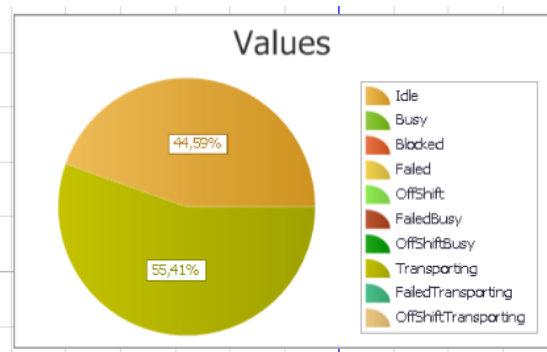


Figure 4.14: Vehicle's Status Pie

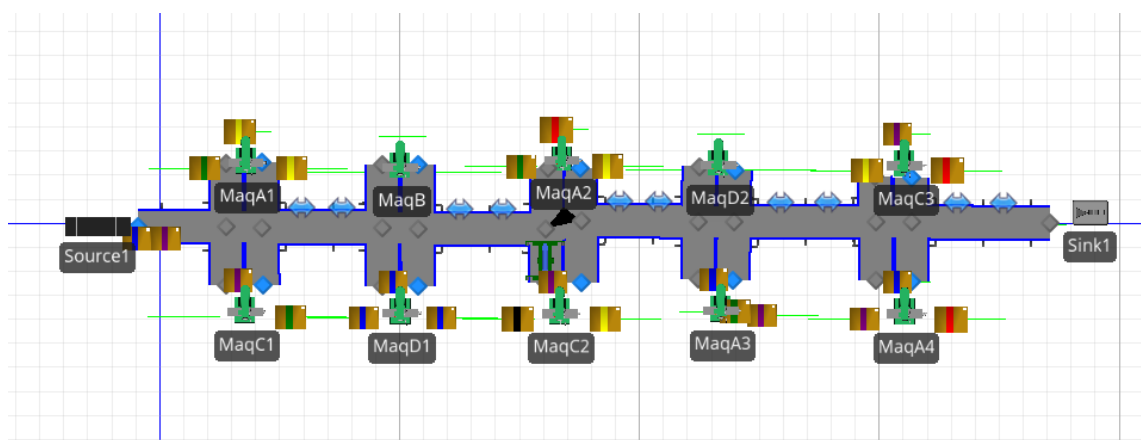


Figure 4.15: A Deadlock situation

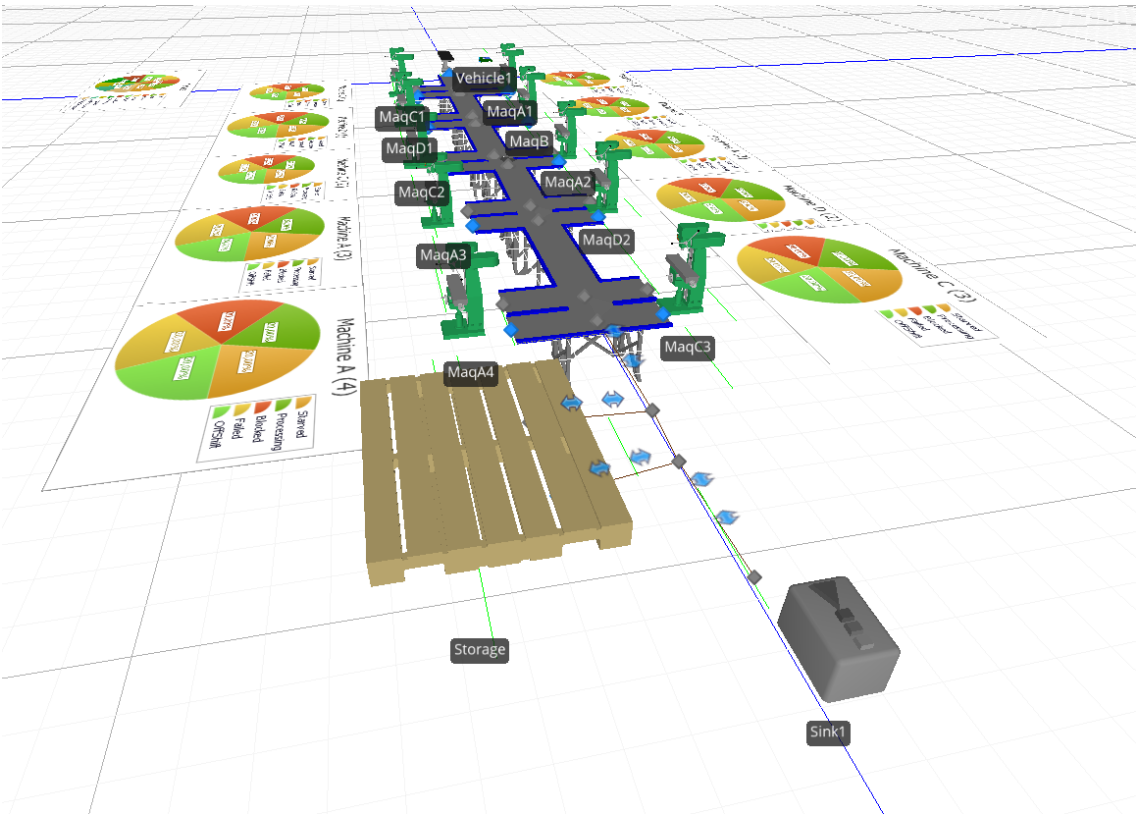


Figure 4.16: Facility with storage implemented

Chapter 5

Experimentations and Results

In the previous chapter 4 we have seen how the implementation was done. Now, the next step is to run these simulations and to analyse. That's what is going to be done in this chapter. First the different possible scenarios will be analysed and, in analogy to the real case, how can these scenarios change and, after implementing some changes, what the possibility of changing some configuration.

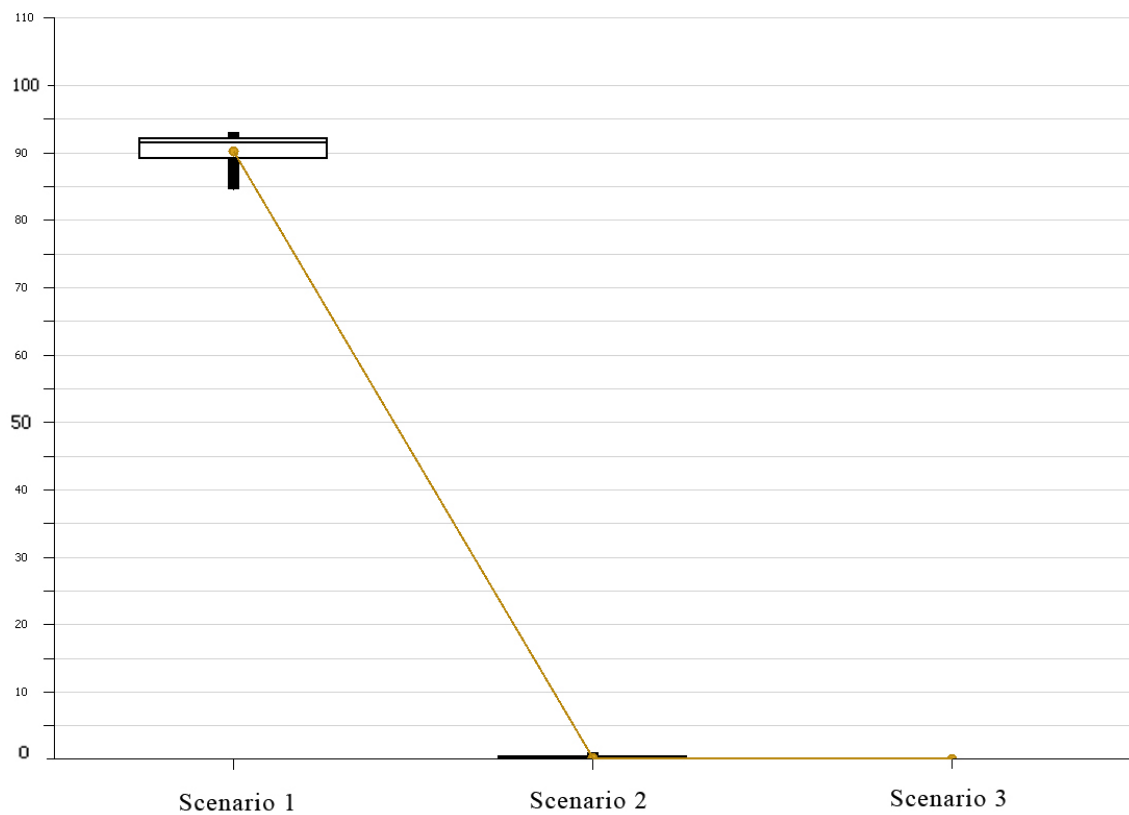


Figure 5.1: Bloqued Machines

Interarrival-Time

It represents the frequency that new parts arrive to the line. As there aren't any real values that we can use. For deciding the best Interarrival-time, we will run some simulations and analyse the machine's blocked time. We are looking for an interarrival-time that doesn't put too much parts in the system (and probably cause a deadlock) but we also want to take the most of the machines. And so, to test possible Interarrival-times, we will do 10 simulations of the system for values 2, 3 and 4 minutes between each arrival, corresponding respectively to scenarios 1, 2 and 3, and study how these scenarios respond with the machine percentage time of processing and blocked.

As we can see in Figure 5.1, that represents the percentage of blocked time, the first scenario (with interarrival-time of 2 minutes) has a very high percentage (more than 90%) of blocked time. This might be explained by the fact that a deadlock happens (more about deadlocks will be explained ahead) when the parts come with a frequency of between 2 and minutes. So if we only look at the blocked time, we may say that the bigger the interarrival-time is, the smaller will be the blocked time so it's for the best.

However we must also consider the processing time. If we take a look at Figure 5.2, when the processing time starts increasing apart from the second scenario , the processing time starts going down (there wasn't need to simulate more scenarios because the processing time would just continue to go down.)

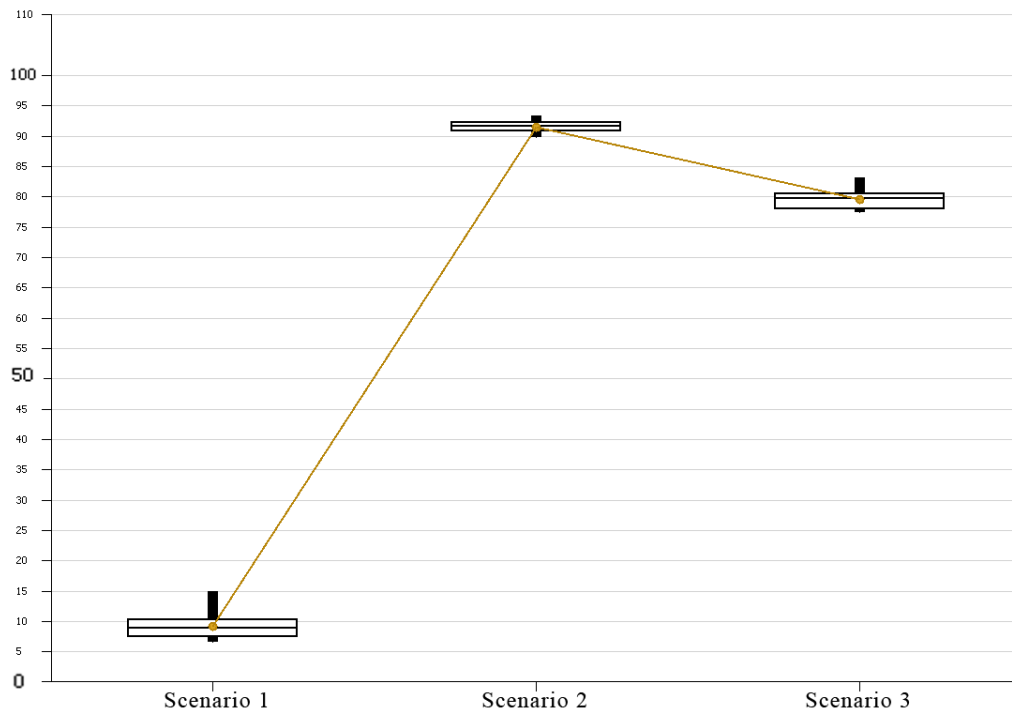


Figure 5.2: Processing Machines

With this, we can conclude that the best time would be around 2 and 3 minutes. So, and for the interarrival-time being a little synchronized with the values of the processing times, so it was decided to go to the values in the operations sequence, in table 4.1 and calculate a triangular distribution through it, where the minimum corresponds to the minimum process time of an operation, the maximum corresponds to the maximum time of an operation and then we have the average of all the values. Once we finish all the calculations, we end up with the following results:

Table 5.1: Interarrival Time Probability Distribution

Maximum	6,8
Most Likely	1,51
Minimum	0,38

So now we take these results and apply them into a triangular distribution, resulting in the graphic in Figure 5.3, where the x-axis represents the interarrival-time verified and the y-axis represents the number of occurrences, where the maximum will correspond to the Mix (the bigger the Mix is, the more occurrences of the entity there will be).

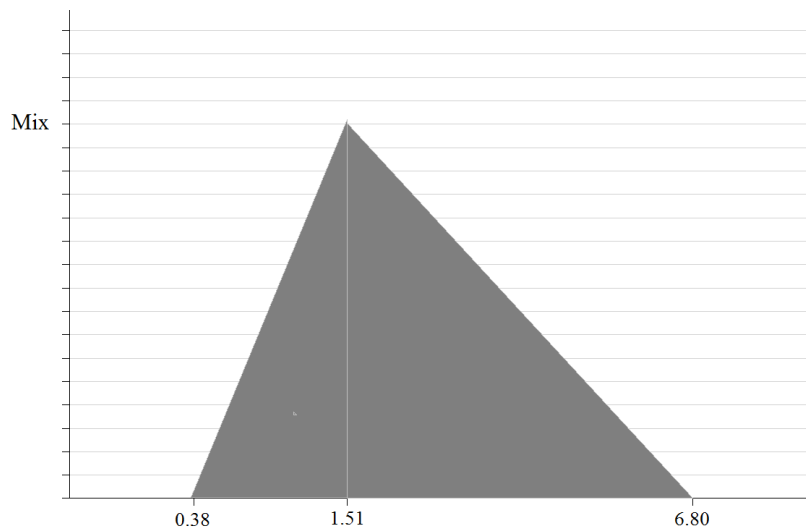


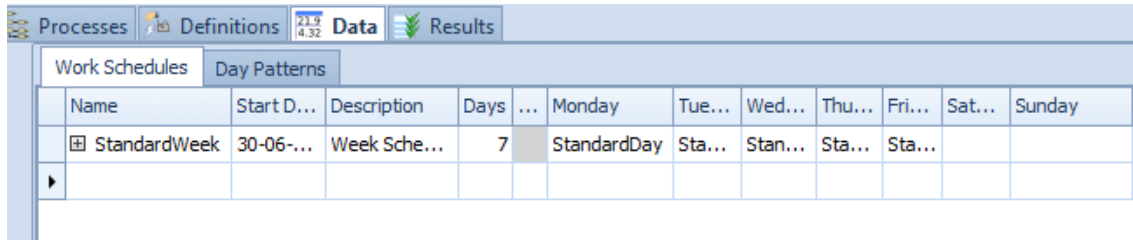
Figure 5.3: Interarrival-Time Distribution

Time of the simulation

This item has also a great influence on the final results and it depends on two aspects: the amount of time we are simulating (one day, one week, one month) and also the amount of simulations we do for each scenario.

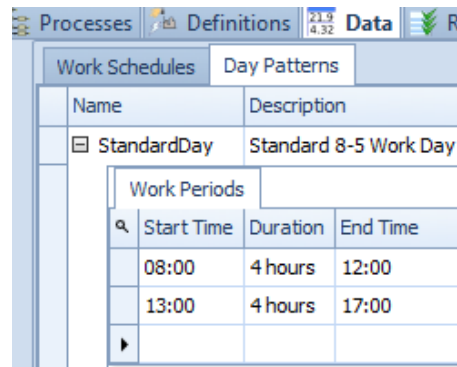
So first, the amount of time. Typically, new orders arrive to the facility at least once a week, so the simulations we will be doing will be for one week of work. A week of five days (Monday - Friday) with 8 hours of work each (08:00 - 12:00 and 13:00 - 17:00). For

simplifying the simulations, we won't focus on workers breaks or holidays, so in every simulation we will recreate 40 hours of work. If a more precise work, it would be possible and important to put these things into attention. For defining a week and day pattern in Simio, there were created two schedule tables. We can see them in Figures 5.4 and 5.5 how these were defined.



Name	Start D...	Description	Days	Monday	Tue...	Wed...	Thu...	Fri...	Sat...	Sunday
StandardWeek	30-06-...	Week Sche...	7	StandardDay	Sta...	Stan...	Sta...	Sta...		

Figure 5.4: Week Schedule



Name	Description
StandardDay	Standard 8-5 Work Day

Start Time	Duration	End Time
08:00	4 hours	12:00
13:00	4 hours	17:00

Figure 5.5: Day Schedule

Next step is to decide the number of simulations to be done in each scenario. The more simulations we do, the closer it is going to be to the expected value. So of course, the best thing to day would be do as many simulations as possible - thousands or even millions of them. However, doing this gigantic amount of simulations also takes a lot of time, and that is a big disadvantage. When creating this simulation model, we don't intend that it takes forever to produce results. The managers that requested the simulations don't want to wait so long to so the results of something that is more likely not to be implemented in real facility (normally, dozens of different scenarios are simulated so that only one change can be accepted).

Other thing that can speed up the simulation process is better hardware. With the available tools, for each time that a work week is simulated, it takes between 30 and 45 seconds to run. So if, for example, we want to run the simulation 1000 times, it would take over 10 hours. Here, we are going to work with a smaller number of simulations. For this, we are going to test for 5, 10, 25 and 50 simulations, and see the difference of results in the number

of entities created and destroyed (i.e., that arrived the line and that left it). So, from these simulations, we got the following values:

Table 5.2: Number of Simulation Runs

	Entities Arrived			Entities Produced			Total Running Time (mins)
	Min.	Av.	Max.	Min	Av.	Max.	
5 Simulations	827	828	829	817	818	820	3,64
10 Simulations	828	837	846	817	829	839	7.02
25 Simulations	825	838	849	816	830	840	18.81
50 Simulations	823	838	852	816	830	844	36.27

As it was expectable, the maximum and minimum limits go wider as the number of simulation occur. However, we can see that it takes to long to do this many simulations, so we have to decide to a intermediate value. We can see in table 5.2 that there is a significant difference between doing 5 or 10 simulations. But then, the major differences are between the limits (max. and min.) but the average stays more or less the same. And so, we chose to do 10 simulations, for each scenario.

Mix

The shoe's industry is always changing according to the season, to people and to fashion. In one week a model of shoes may be pretty famous and the next week almost no sales. That's why it is important to keep changing the amount of shoes that is produced of each type. And that can be controlled through the Mix, according to the table we have seen before. And this is where we will try to have values of a better simulation. In the next section, we are going to see an example of a Mix change and then what we can do with the line to make it with better results.

5.1 An example of a line balancing

Now that aspect of the simulation running in this model has been indicated, we can now proceed to the simulation. First we are going to run the system with a purpose of recreating no big changes in the market, what means that the Mix of all the parts will by the same. But then, we are going to see how the system reacts to a big change (one of the shoe models will come out more often) and analyse what can we do to improve that. We will first consider the configuration that can be seen in Figure 4.2 to observe its behaviour and then, a small change in this configuration will be made (a machine will be trade by another) so that the simulation model can validate this alteration.

So first we are going to proceed to an equal Mix simulation. See how many entities were created and how many were produced of each type. And the results can be seen in Table 5.3.

So, as we can see in table 5.3 (where A. stands for parts arrived to the line and P. for parts that were produced), all the values are more or less the same. There isn't much variation because all the Mix was the same.

Table 5.3: Simulation 1

	Humpry		Wrapper		Foxy		Frazer		Jefferson		Sakie	
	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.
Average	144,2	141,8	130,0	128,2	139,6	138,8	136,0	134,8	145,8	144,8	140,4	138,6
Min	126,0	122,0	123,0	121,0	128,0	127,0	121,0	118,0	134,0	133,0	127,0	125,0
Max	165,0	165,0	135,0	133,0	156,0	156,0	148,0	147,0	166,0	165,0	173,0	172,0

However, let's do it with a different Mix. Let's pretend that, for instance, the model Frazer was very fashionable, during one week, so it had a lot of demanding. Similar thing happened with the models Humpry and Wrapper. On the other hand, Jefferson and Sakie had too little demand. So we organised the Mix table like we can see in table 5.4

Table 5.4: Simulation 2 Mix

	Humpry	Wrapper	Foxy	Frazer	Jefferson	Sakie
Mix	75	75	50	100	5	5

And running the simulation, we have obtained the following values for both the pairs of shoes that arrived the line and that left:

Table 5.5: Simulation 2

	Humpry		Wrapper		Foxy		Frazer		Jefferson		Sakie	
	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.
Average	209,8	142	203,6	129,2	138,6	70,4	258	171,4	13,4	8,2	12,4	8,6
Min	189,0	21	184,0	37,0	116,0	17,0	247	24,0	7,0	1,0	6,0	1,0
Max	246,0	246	229,0	202,0	158	129,0	272	269,0	19,0	19,0	19,0	18

As we can see in table 5.5, there is a lot of WIP, specially in the parts that have more Mix. This can be resolved by a simple action of line balancing. As we can see in table 4.1, the most produced parts (Frazer, Wrapper and Humpry) don't use almost nothing the machine D. On the other hand, they use a lot of machine A. As we have two machines D, let's make a change and putting machine D doing operations of machine A (of course, in some cases this isn't possible, but for this example let us assume that it is).

So now, we will run the simulation again, in the same values of Mix as in table 5.4, but now we have added another machine A. Now after we ran the simulation, we have obtained the following results:

As we can see in table 5.6, there were more parts produced with around the same amount of material (there was some change, of course, because as said before, simulation recreates random processes, so it's very difficult for one simulation to produce the same results as other).

Figure 5.6 represents a chart with the results of tables 5.5 and 5.6. As we can see, there a big variance between the maximum and minimum values, specially in the amount of entities produced.

Also in the Appendix A we can see a table with the vehicle's performance in the last scenario, where are shown several values such as the amount of units transported, vehicle's idle time or its transporting time.

Table 5.6: Simulation 3

	Humpry		Wrapper		Foxy		Frazer		Jefferson		Sakie	
	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.	A.	P.
Average	209,8	170	203,6	149,4	138,8	89,8	258	201,4	13,4	5,2	12,4	5,6
Min	189	43	184	41	116	21	247	94	7	1	6	1
Max	246	246	229	213	158	140	272	269	19	19	19	18

5.2 Results Analysis

With this we can conclude that a simple line balancing that consisted in changing a couple of workstations took a significant effect in the systems performance. If we went through other examples, we could find more situations where we could improve the system's performance by implementing a line balancing operation. This model is more directed to this facility in specific, but we managed to understand how it can work if we apply simulation to other fields of line balancing. We just need to find where can we improve the system.

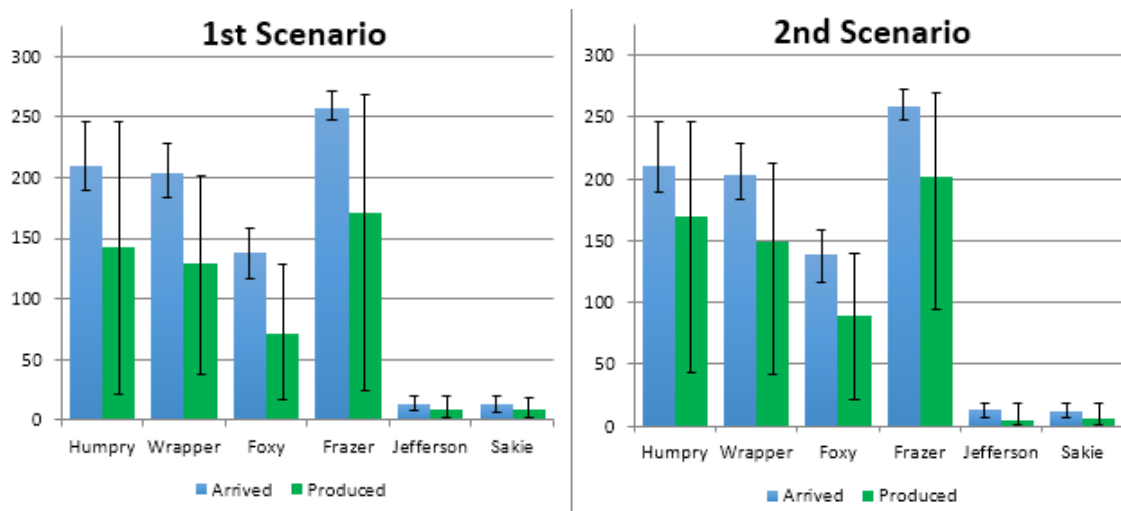


Figure 5.6: Simulations for the last two scenarios

Chapter 6

Conclusions and Future Works

Production lines balancing is a concept really very complex and that go into a lot of direction, which normally follow the evolution of technology. In this dissertation we have studied how we can apply new technology to this concept. The technology: a fairly new framework that represents a leap forward in simulation technology - Simio. Although much of the work spent on the development of this thesis was spent learning how to work with Simio, this time can be considered as very useful, as this is going to be a new simulation framework that is going to be frequently used in resolving old problems.

The fact that it was used a real case to study this problem helped a lot. Although the real case wasn't strictly followed (for simplification of the problems solving and also because the company couldn't provide really specific data about the facility configuration and other areas), isn't a problem. In fact, it opens doors to be used for other problems in other different areas. We've seen in the previous chapter 5 that it works to resolve a line balancing problem in the footwear industry through this way, so it is certain that it can be applied in other industries.

Therefore, for the future, this dissertation leaves two objectives:

- In the first place, it leaves the challenge to apply this model with more accurate data and use real information to test new configurations and perhaps implement one of them.
- Then, it motivates to continue this area and to use what was learned here to other fields.

Appendix A

Simulation 3 - Vehicle Status

Table A.1: Vehicle Properties for simulation 3

Object Name	Data Item	Statistic Type	Av.	Min	Max.	St. Deviation
Vehicle1	ScheduledUtilization	Percent	58,69	57,25	59,95	1,29
Vehicle1	UnitsAllocated	Total	5541,40	5444,00	5631,00	83,39
Vehicle1	UnitsScheduled	Average	1,00	1,00	1,00	0,00
Vehicle1	UnitsScheduled	Maximum	1,00	1,00	1,00	0,00
Vehicle1	UnitsScheduled	Minimum	1,00	1,00	1,00	0,00
Vehicle1	UnitsUtilized	Average	0,59	0,57	0,60	0,01
Vehicle1	UnitsUtilized	Maximum	1,00	1,00	1,00	0,00
Vehicle1	UnitsUtilized	Minimum	0,00	0,00	0,00	0,00
Vehicle1[1]	ScheduledUtilization	Percent	58,69	57,25	59,95	1,29
Vehicle1[1]	UnitsAllocated	Total	5541,40	5444,00	5631,00	83,39
Vehicle1[1]	UnitsScheduled	Average	1,00	1,00	1,00	0,00
Vehicle1[1]	UnitsScheduled	Maximum	1,00	1,00	1,00	0,00
Vehicle1[1]	UnitsScheduled	Minimum	1,00	1,00	1,00	0,00
Vehicle1[1]	UnitsUtilized	Average	0,59	0,57	0,60	0,01
Vehicle1[1]	UnitsUtilized	Maximum	1,00	1,00	1,00	0,00
Vehicle1[1]	UnitsUtilized	Minimum	0,00	0,00	0,00	0,00
Vehicle1[1]	IdleTime	Average (Hours)	0,01	0,01	0,01	0,00
Vehicle1[1]	IdleTime	Occurrences	2397,20	2291,00	2481,00	76,60
Vehicle1[1]	IdleTime	Percent	41,31	40,05	42,75	1,29
Vehicle1[1]	IdleTime	Total (Hours)	16,53	16,02	17,10	0,52
Vehicle1[1]	TransportingTime	Average (Hours)	0,01	0,01	0,01	0,00
Vehicle1[1]	TransportingTime	Occurrences	2397,40	2291,00	2481,00	76,71
Vehicle1[1]	TransportingTime	Percent	58,69	57,25	59,95	1,29
Vehicle1[1]	TransportingTime	Total (Hours)	23,47	22,90	23,98	0,52
Vehicle1[1]	NumberInStation	Average	0,29	0,29	0,30	0,01
Vehicle1[1]	NumberInStation	Maximum	1,00	1,00	1,00	0,00
Vehicle1[1]	NumberInStation	Minimum	0,00	0,00	0,00	0,00
Vehicle1[1]	TimeInStation	Average (Hours)	0,00	0,00	0,00	0,00
Vehicle1[1]	TimeInStation	Maximum (Hours)	0,00	0,00	0,00	0,00
Vehicle1[1]	TimeInStation	Minimum (Hours)	0,00	0,00	0,00	0,00
Vehicle1[1]	NumberEntered	Total	5541,40	5444,00	5631,00	83,39
Vehicle1[1]	NumberExited	Total	5541,20	5444,00	5631,00	83,33

References

- [1] Thorsten Simon Harzer. Value creation through mass customization: An empirical analysis of the requisite strategic capabilities. Master's thesis, Fakultät für Wirtschaftswissenschaften, 2013.
- [2] Frank T. Piller. Mass customization: Reflections on the state of the concept. *International Journal of Flexible Manufacturing Systems*, 16(4):313–334, 2004. doi:10.1007/s10696-005-5170-x.
- [3] Nancy Nardin. Smart selling tools: Sales waste and the production line theory of revenue generation. Published online, July 2013. Last visit on July 2014. URL: <http://www.pros.com/company/news/sales-waste-and-production-line-theory-revenue-generation/>.
- [4] Nils Boysen, Malte Fliedner, and Armin Scholl. Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 111(2):509 – 528, 2008. doi:10.1016/j.ijpe.2007.02.026.
- [5] Pravin Y Tambe. Balancing mixed-model assembly line to reduce work overload in a multi-level production system. Master's thesis, Louisiana State University, 2005.
- [6] Nuchara Kriengkarakot and Nalin Pianthong. The assembly line balancing problem: Review articles. *KKU Engineering Journal*, Vol. 34(2):133 – 140, April 2007.
- [7] Waldemar Grzechca. Assembly line balancing problem single and two-sided structures. In A D Rodiê, editor, *Automation Control - Theory and Practice*, chapter 1, pages 1 – 14. InTech, December 2009. doi:10.5772/7894.
- [8] Nick T. Thomopoulos. Mixed model line balancing with smoothed station assignments. *Management Science*, 16(9):593–603, 1970. doi:10.1287/mnsc.16.9.593.
- [9] Ardavan Asef-Vaziri. Product-process matrix. Lecture Slides from California State University Northridge. URL: <http://www.csun.edu/~aa2035/SOM306/powerpoints/Part1/Productivity/JobShopFlowShop.pptx>.
- [10] ProModel. Justifying simulation. Published Online. Last visit on July 2014. URL: <https://www.promodel.com/pdf/Justifying%20Simulation.pdf>.
- [11] William H. Sanders. Simulation basics, 2005. Lecture slides from the University of Illinois. URL: <http://users.crhc.illinois.edu/nicol/ece541/slides/simulation-basics.pdf>.
- [12] Roger D. Smith. Simulation article. In *Encyclopedia of Computer Science*. Groove's Dictionaries, New York, New York, 4th edition, July 2000.

- [13] David T. Sturrock and C. Dennis Pegden. Recent innovations in simio. In *Proceedings of the Winter Simulation Conference*, WSC '11, pages 52–62. Winter Simulation Conference, 2011. URL: <http://dl.acm.org/citation.cfm?id=2431518.2431526>.
- [14] C. Dennis Pegden and David T. Sturrock. Introduction to simio. In *Proceedings of the Winter Simulation Conference*, WSC '11, pages 29–38. Winter Simulation Conference, 2011. URL: <http://dl.acm.org/citation.cfm?id=2431518.2431524>.