Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

School of Engineering and Sciences

Discrete event simulation with integration of optimizations and the internet of things.

A thesis presented by

Rubén Febronio García Martínez

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

In

Manufacturing Systems

Monterrey, Nuevo León, June 12th, 2020

Dedication

To God and my beloved parents,
Aurora and Rubén.

To my sister and my grandmother,
Aurora and Mª Guadalupe.

Acknowledgements

I would like to express my deepest gratitude to all those who have been side by side with me:

To God for his wisdom, care and protection.
To my dear and beloved parents, sister and grandmother, who have been my source of inspiration and have gave me unconditional confidence, support, patience, and encouragement.
To my friends, Abraham, Karla, José, Fernanda, Sergio, Cindy, Diego, Jafet, for their love and unconditional support.
To my advisor for guiding and support me during this period.
To my professors for their motivation.
To Tecnológico de Monterrey for the support with tuition.
To CONACyT for the support for living.

Discrete event simulation with integration of optimizations and the internet of things.

by

Rubén Febronio García Martínez

Abstract

There is a lack of development in the discrete event simulation area. Representing a real process through a digital model has proven to be a useful tool because modifications can be carried out at no cost to corporations within a simulated environment. Even so, within the creation of DES for specific processes, it has different limitations (data collection, visual representation of the model, platform integration for the final simulation). That is why the present work proposes the integration of discrete event simulation (DES) with the environment of industrial internet of things and an industrial process. A simulated process was developed using Simpy, a Python tool, and Siemens Plant Simulation. Variables obtained from Python simulation were saved on a remote server using Google Scripts. The results of the Plant simulation were sent to an Excel data sheet. Comparisons of both simulations were performed, with variables such as cycle time, number of parts per day and other indicators.

After comparing the simulations, other scenarios were tested to illustrate how the Python model can work with different data. Then, optimizations were carried out to maximize the number of items produced in different scenarios. Future work includes implementation in real factory environment and the interconnection with other technologies as augmented reality.

List of Figures

List of Tables

Contents

CHAPTER 1. INTRODUCTION

In different ways, technology has been present in the life of humans since ancient times. Nowadays the technology has reached one of the most important points in the history of humanity thanks to all the advances that are being created. The internet is not a new discovery but now it can play a very important role because mixing technology and Internet can result in many new and innovative things and processes. Thanks to the IoT and the connections machines-cloud, data is being generated at every moment, creating new opportunities to work with it through the integrations of different Industry 4.0 tools.

Talking about manufacturer companies, Industry 4.0 has gained a lot of strength among them and it is like a new trend to evolve the company to a more technological one. The processes and the consumers nowadays demand more, fast, and best products and that is why the companies need to evolve, to reach the customer needing [1].

To make the processes more efficient and with more quality it is not only necessary to have a good production system and a good design of facilities. What is necessary now is to have a well informed and feedback process and with great speed. Industry 4.0 allows all of this and more due to the interconnections of things, machines and humans that it makes and the capability of analyze huge amount of data. Another important part in the

transition to a 4.0 process is the participation of a digital architecture in which the process will work and send information over time, where sensors, the network and the equipment will be interconnected. [2]

With the advent of industry 4.0, companies have new challenges. One of the challenges is to achieve or improve the integration of new technologies within the framework of industry 4.0 with the processes and methodologies existing in the industry. New communication and connectivity systems need to be implemented in order to evolve the industry in a comprehensive and joint way. [3]

One of the applications that has gained fame and boom has been the creation of digital twins to predict, measure and analyze the behavior of a certain process, machine or industry, in a safe way in terms of use of resources and time [4]. Simulations and data analysis digitally is another approach to the combination of industry 4.0 with reality [5]. Another approach of representing the reality in a digital way is by the creation of systems and representing them through virtual reality [6].

Some software such as Plant Simulation from Siemens has been widely used as a resource to simulate all kinds of processes, from simple [7] production lines with few processes to complete [8] factories. This software was one of the options to carry out this work, however, despite being able to represent reality in an adequate way, it has limitations regarding communication between different software's, with the cloud and with other IoT tools; nor does it allow much flexibility in modifying process parameters easily.

For this reason, it was decided to use Python and the Simpy library [9] for the development of DES, as it is a free, free and easy-to-use platform for developing interconnectivity with other tools of IoT. With this knowledge, a discrete event simulation is developed and its representation in Python will be analyzed, compared and optimized to create a visual system that represents in real time the process.

1.1 Background

1.1.1 Industry 4.0: Evolution, Technology and Human Resources

Industry 4.0 can be defined as an initiative where the technologies based on the Internet are used to improve and upgrade the industrial processes. This initiative is new, but it has been evolving since the first industrial revolution through the years. The first industrial revolution happened during the 1700's and 1800's and in this period of time the manual labor work which involved people and animals, evolved to a more efficient process in which the steam and water were the main materials to power the engines, machines and tools [10].

After many years, in the 1900's started the second industrial revolution which consisted now in the introduction and use of steel and electricity in manufacturing processes. With this new power the production processes were more efficient, fast, and new concepts appeared like the mass production and the assembly lines, combining human resources and machines.

Then during the 1950's, a third industrial revolution started to appear when the manufacturers started to combine not only electricity in assembly lines but also electronic components with computers in the assembly lines. This shift to more digital processes instead of the traditional opened the guide to focus into the digital and the automation of the production lines.

Finally, the fourth industrial revolution emerged just a few decades ago when the Internet and the processes entered a new era: the use of sensors to collect data, which transforms into big data and the need of manage this big data in a more efficient way using less time and resources, having access to real-time data and the introduction of cyber-physical systems. This is how the Industry 4.0 born. Industry 4.0 offers a new comprehensive, holistic and interlinked manufacturing network in which it is connected the digital and physical parts of everything and allows a better collaboration between the whole company and the whole supply chain [10].

1.1.2 Internet of Things: Everything's interconnected

Internet of Things definition starts with the Internet definition which is a global interconnected system or network that uses the standard protocol TCP/IP. So, the IoT involves the addition of Internet as connectivity for certain objects that need to be connected, and it also refers to connections between physical objects as sensors and machines with the Internet, working in parallel to send and receive data, learn, take decisions and achieve better results [11]. Nowadays it has appeared a similar concept which is IIoT, Industrial Internet of Things that refers to the connections also but know between people, data, and machines, all of them related to manufacturing. [12]

1.1.3 Big Data: Manage the information

Thanks to the actual technology every person in the world can generate a lot of data every single second. When someone upload a new photo to a social network or even when the location is activated in a mobile device, information is being generated. Something similar is happening in the industries around the world. They are all the time creating products, receiving material and a huge amount of information is generated and in the actual world all that information is been stored in the networked, digitized internet world. All of that is called Big Data and nowadays the problem is no more how to store but how to use it to generate value and to take advantage of it, either to improve a process, or to connect different machines to transform a production process from a robust to a lean one. [13]

1.1.4 Cyber-Physical Systems

In some processes the different involved parts have to be connected in some way in order to work correctly, without loss of data, time and money. Recently this has been achieved by making a connectivity between every part of a process using cyber physical systems altogether with Internet of Things, Big Data and Cloud Services. [14]

Also called Integration of computation and physical processes, cyber physical systems are online networks of connected equipment's that use Internet of Things technology and mechanic or electronic accessories. [11]

1.1.5 Analytics and Statistics

When data is collected it can be stored in a computer, in a server, on the internet ow in many different places for later use. But it is important to keep in mind that the raw data does not mean anything until a person analyze it and decides what to do with it. To analyze data is necessary to have a storage, a purpose and a methodology or a software to analyze it. [13] There are different kind of statistical software like Minitab or MATLAB [15] where the information can be stored to obtain different results or even optimizations according to what is been looking for.

1.1.6 Sensors

The most common sensor, or the most known could be a thermometer, which was designed to measure the temperature of the environment, of the body of a person or just the temperature of any object. Currently sensors have been widely used in several applications ranging from medicine applications to industrial applications. [16] They detect different things depending of its use and receive a signal from a device, and then transform that signal into an output that can easily be understood and used. Sensors are important in Industry 4.0 because thanks to their versatility can be used in several applications, from putting it in a machine to measure the vibration until connect them to the internet to send and receive data and feedback a production system. [17]

1.2 Problem Statement and Context

The present work aims to analyze a process of item creation from raw material that enters to a machine by an operator, who subsequently activates it and while a percentage of the material is used to create an item some material is wasted but recovered. Once finished, the machine is stopped, and the item is removed.

This process is isolated and currently does not have an analysis system in charge of obtaining process data (cycle times, created items, etc.).

Taking this process as a case study and under that premise, the problem was found that in current simulation systems there are various limitations related to the connectivity of the simulation with external systems or programs as well as with IoT tools, during its

execution or for the subsequent analysis of results. In addition to this, the current systems to represent DES are expensive and represent an expense for organizations, who choose to keep their operations as they are because they do not see the need for the implementation of a DES analysis and optimization system. That is why the need arises for a free DES system, that uses IoT tools, that is flexible and that is also capable of optimizing parameters of the analyzed process.

1.3 Objectives

1.3.1 General

To simulate a process through a DES that have the capacity to feedback itself with data, using IoT tools such as optimizations and a channel of communication between the real process and simulation.

1.3.2 Specific

Determine the best simulation software for this work.

Identify the best communication simulation-cloud and implement it.

Achieve the system communication.

Prove the similitude between the DES and real process with commercial software.

Demonstrate the DES versatility through analysis of different scenarios.

Define the way to optimize the DES.

1.4 Hypothesis

A DES can be correctly represented and improved with real time data produced in the factory?

1.5 Research Contribution

Different authors have worked with specialized simulation software [7], or developed their own DES systems to represent process from reality. Also optimizations have been done to improve processes parameters such KPI's [18]. But there is a lack into combining different IoT tools to a DES analysis creation.

The main contribution of this work will be the creation of the connections of a flexible DES made in Simpy (a tool of Python) and compared with Plant Simulation, capable of represent different scenarios of a process and create optimizations in MatLab, allowing to generate improved results according with the tested scenario and make changes in the real process if needed.

1.6 Justification

This research work was carried out with the purpose of exposing the scope of different tools through their practical combination, relating the analysis of data from a process, the simulation of it in an interactive and technical way, also applying optimizations, which served of simulation validating agent. Similarly, analyzing different scenarios derived from the baseline DES can show how versatile a digital twin can be, being able to make little changes to the process or modify its scope with simplicity and obtaining good and expected results. Also, optimized data opens the possibilities to train machines and processes to improve automatically and semi autonomously.

1.7 Thesis Organization

This research work is divided into six chapters, beginning with this chapter named Introduction..

Chapter 2: It contains the literature review of the work. It discusses previous researchers work and knowledge about related topics.

Chapter 3: Describes the followed methodology used in the development of the work: creation of DES, different scenarios and optimizations.

Chapter 4: It explains the results obtained for each of the simulations, the comparison between them, the results of the different scenarios and the optimizations made.

Chapter 5: This chapter describes the whole research work in a single conclusion.

Chapter 6: It describes the future work that can be done to continue and improve this research work.

CHAPTER 2. STATE OF THE ART

2.1 Literature Review

The literature analysis aims to illustrate the relationship of various topics and their impact and importance in the present work. It will be divided in five different topics:

- Cyber Physical Systems and Digital Twins, showing the importance of its relation between each other and the impact in the generation of better digital twins.
- Discrete Event Simulations
- DES Optimizations
- Simulation in Digital Twins
- Integration of Sensors

2.1.1 Cyber Physical Systems and Digital Twins

Digital Twin concept first mention was in a University of Michigan conference in 2003; since then, the term of DT has been widely studied due to its capacity of synchronize and represent physical activities and behavior with the virtual world, allowing to make predictions and designs [4]. The definition of cyber physical systems and digital twins is related in a certain way due to its emphasis on the integration of the physical world with the digital one, in order to obtain a beneficial feedback for the complete system

that allows improvements, which can be of autonomous way but compared with DTs, CPS emphasize the computing and communication capabilities of the cyber world, which can enhance the accuracy and efficiency of the physical world [19].

2.1.2 Discrete Event Simulations

The use of Discrete Event Simulations (DES) in the design of production systems, although increasing, is still sporadic [20]. DES is a tool that can be used during different stages of a production system design that are created to answer detailed questions about how a complex system will behave, through analyzing many variables and operations. During the model of the DES the system is represented as a series of events and the changes to this system are dynamically evaluated. According with [21] the use of DES can be separated into three process phases: design, development and deployment as shown in Figure 1.

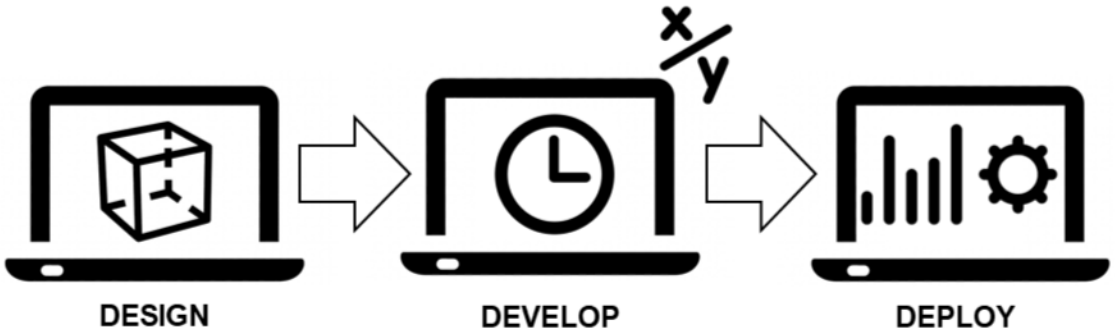

Figure 1. DES Phases

The DES design phase consists into generate conceptual models, goals and deliverables of the system. Then in the development phase the different parts, activities and operations of the system are identified, as well as de input data for the simulation. In the deployment phase the activities include experimentation, simulation, output analysis and the use of these results. As mentioned by [20], although DES is a very popular technique in the design of production systems, the full benefits of DES for manufacturing companies remain to be realized. There are common thoughts between different authors about the “hidden” benefits or capabilities of the use of DES in the industrial area. The lack of research in these topics is an open window to explore and improve them. For the

development of discrete event simulations there are several tools that can be open source or that are on the market. It is worth noting one of them which exists within the Python programming environment, Simpy. Simpy is a programming library or framework that is based on the Python standard and that has features that facilitate DES programming, such as the environment, resources, etc. [22]

2.1.3 DES Optimizations

In the literature it has been found that most of the time the development of a digital twin focused on manufacturing, production, process optimization or product design process goes hand in hand or starts from the creation of a discrete event simulation [5]. DES use has greatly increase in recent years due to its increased competition and tool availability; likewise, the appearance of various simulation software has been of great impact in the development of DES focused on the optimization of processes and layouts, identification of KPIs, production line design, etc [23].

Nowadays the information and interconnection trend are more a necessity for all the industries, and this has led to investigate new ways to make things possible. The actual problem is to achieve an effective communication between machines and the internet, maintaining a simple language between them and to have it standardized to make further processes easier and fastest.

2.1.4 Simulation in Digital Twins

Simulation tools such as Siemens Plant Simulation [24] and Arena [25] are part of the most used group of tools because they allow to evaluate and represent with great accuracy the model you want to simulate based on reality. Various simulation trends have been identified by authors, such as the shift in the application of DES from design to operations enabling the benefits of simulation to be realized further downstream in manufacturing processes. A key trend noted by authors is the increased use of hybrid modelling methods and optimization algorithms and tools [26]; these new links greatly increase the capability of DES and provide further benefits for manufacturers.

It is known that DES provide information about how a process can become depending on the variables that have been introduced to the system, be it time, processes, labor, etc., but do not interact with the system in the physical world, real. The demolition of this barrier was the one that brought to reality the implementation of the simulations of discrete events in conjunction with other areas of Industry 4.0 such as the Internet of Things, Digital Twins, Sensorization and Analysis [27].

Approaches of the integration of DES in digital twins can prove the advantages of the use of these technologies for the analysis and representation of systems that are not yet existing [28].

The good relationship and integration between DES and Digital Twin has been proven several times through various works such as the development of different works using software like Plant Simulation [7], with results that were a good approximation to reality and with the possibility of make optimizations to the new developed model.

The virtual reality is another topic that is enlisted in the road to Industry 4.0 and goes hand in hand with simulations of discrete events and digital twins. Virtual reality offers immersive 3D scale- one visualization, realistic rendering, natural gesture interactions, collaboration functionalities, and a quick navigation tools in wide area [29]. VR can provide a new way to show the results of a simulation, an optimization or a representation of a digital twin in the real life, in real time and with the possibility to interact with the digital system [30].

2.1.5 Integration of sensors with DES

In the context of detection of errors, failures and malfunctioning's in machines there has been made several researches during the last years. Industrial processes involve that every single part of the production system needs to be working as well as the first time and that's why there is no room for failures during a process. Also, a machine stop represents lots of money lost for the company which simply cannot happen. One study shows that a company process oil and that they use machines like presses and pumps

and that there can be measured some characteristics of the machines using sensors, analyzing data and using SDAEM (Sensor Data Analysis for Equipment Monitoring) [31].

To summarize the concepts mentioned in the literature review, a morphological matrix was developed. In it the different works related to the present are listed, as well as the topics they address. The morphological matrix is shown in Table 1.

Table 1. Morphologic Matrix

Name	References	Topic/Tool		
		DES	Plant Simulation	Optimizations
Design and simulation based assessment of lean material flows considering imprecision	[18]	✓	✓	✓
Integration of Software Tools with Heterogeneous Data Structures in Production Plant Lifecycles	[32]	✓		
Creation of simulation model of expansion of production in manufacturing companies	[24]	✓	✓	✓
Discrete Event Simulation Output Data-Handling System in an Automotive Manufacturing Plant	[8]	✓	✓	✓
Study of Production Scenarios with the Use of Simulation Models	[33]	✓	✓	✓
Analysis of the production process in the selected company and proposal a possible model optimization through PLM software module Tecnomatix Plant Simulation	[7]	✓	✓	✓
Cyber-physical systems in manufacturing	[14]	✓		✓
Methodology of the creation of human and robot operation in the tecnomatix process simulate	[34]	✓		✓

CHAPTER 3. METHODOLOGY

This section presents the methodology followed in this work. The developed process contains several parts that allow the creation of a system capable of simulating a DES which can be modified to represent different scenarios and subsequently optimized, using IoT tools, as seen in Figure 2.

Figure 2. Process

The system process has three sections:

User: Refers to the operator of the process/machine which is the one that enters the data into the next part, cloud services.

DES Simulation: Contains the simulation environment developed in this work, the different scenarios tested and the optimization script in charge of improving DES results.
Cloud Services: Is the Internet service that allows communication between the User and the DES.

With the big picture of the system, it can be divided into other specific sections to explain more deeply the activities and methodology of each one as shown in Figure 3. This work presents detailed information about the DES creation process, its comparison, the different scenarios tested and the optimizations:

- Physical process.
    - Identification of variables to simulate.
- Discrete Event Simulations
    - Process definition and design of Python simulation.
    - Design of comparison in Plant Simulation.
    - Bayesian Inference.
    - Main Python Scenarios.
    - Other Scenarios.
    - Main Scenarios Comparison.
- Optimization.
- System Communication.

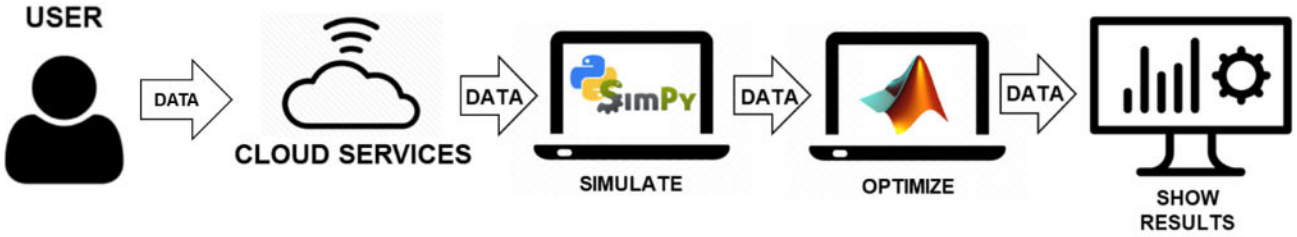

Figure 3. Complete process

3.1 Physical Process

The present work context is based in a machine that consists in a process of creation of items. The material is deposited in the storage chamber by a human operator, a setup is done in the homogenizer box, the operator turns on the machine and the material start to flow from the SC to the Homogenizer box. Then once the item is complete the operator turns off the machine and removes the item. During the process some material is wasted but recovered into the Recovery Chamber trough the recovery mechanism. Process definition is illustrated in Figure 4. The machine is divided in three parts and different processes:

-Storage Chamber
-Homogenizer Box
-Recovery Mechanism

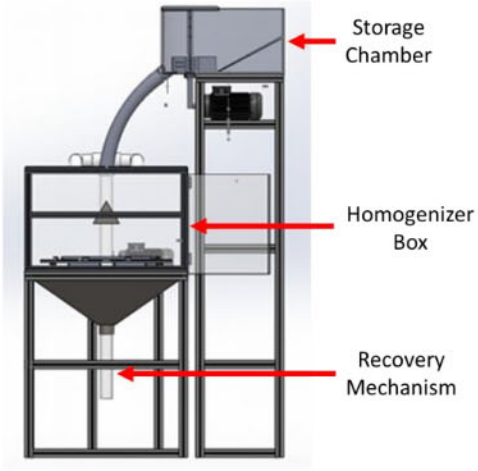

Figure 4. Main parts of the machine

Storage Chamber is divided in main tank and the dosing propeller. The functions of these two components is to storage the raw material and to separate and distribute it into the whole chamber, to allow its flow to the next segment of the machine, passing through a pvc tube.

The *Homogenizer Box* is located in the middle part of the machine. This segment is composed by a box, a triangular prism that serves as a material distributor and a homogenizer mechanism that is activated by a small engine. In the homogenizer box is

where a mold is located by an operator to later empty material into it. Once the mold is filled, it is removed as a final product.

At last, the *Recovery Mechanism* was designed to store the material that has been wasted from the previous segment of the machine. Once this Recovery Chamber is full, the material is resent to the Main Chamber in order for the material to be used.

3.1.1 Identification of variables to simulate

To identify the variables in the process, the process had to be analyzed using the information of the previous section, the data of the machine and its different parts, obtaining the variables shown in Table 2.

Table 2.Variables to simulate

VARIABLES
SC level
SC fill
Homogenizer Box setup
Turn On/Off Machine
Product creation
Removing item
Recovery process

SC level: Can have the values from 20 to 100. It represents the quantity of material that is inside the SC.

SC fill: Amount of time that the human operator takes to fill the storage chamber with material.

Homogenizer box setup: Amount of time that the operator takes to do the setup of the homogenizer box to create the item.

Turn On/Off Machine: Amount of time that the human operator takes to turn on/off the machine operation.

Product creation: Amount of time that the machine takes to create the item.

Removing item: Amount of time that the operator takes to remove the finished item from the homogenizer box.
Recovery process: Amount of time that the operator takes into moving the recovered material to the SC.

The times for each of the operations were defined using an approximation of criteria that could take the operator to perform each task and the machine to execute the operations. The variables that are affected by the operator's work are: Fill the SC, Box setup, Turn On / Off machine, Move item and Recuperate material.

3.2 Discrete Event Simulation

The next step was to program the DES simulation capable of call the data from an external source through internet and use it in the variables of the DES, to obtain results from the real data and check which parameters can change to optimize these results into the simulation itself to perform better, improving the real process with future actions. This DES is to represent and adapt to similar processes to that described in 3.1. A digital twin approach was developed in Pyhton, using the library called Simpy. Then a comparison of the simulation created in Python was created with Plant Simulation software, from Siemens. The next sections of 3.2 explain how the DES was structured in Python, how it was compared using Plant Simulation, how different scenarios aside from the Baseline were tested to show the flexibility of the DES, and the optimizations made.

3.2.1 Process definition and design of Python simulation

Once the general process was defined as can be seen in Figure 3, variable operating times were assigned to each of the process steps, which are its average operating times and standard deviations. These times were partially defined arbitrarily and based on historical operation data in similar processes and represent the process of Figure 5.

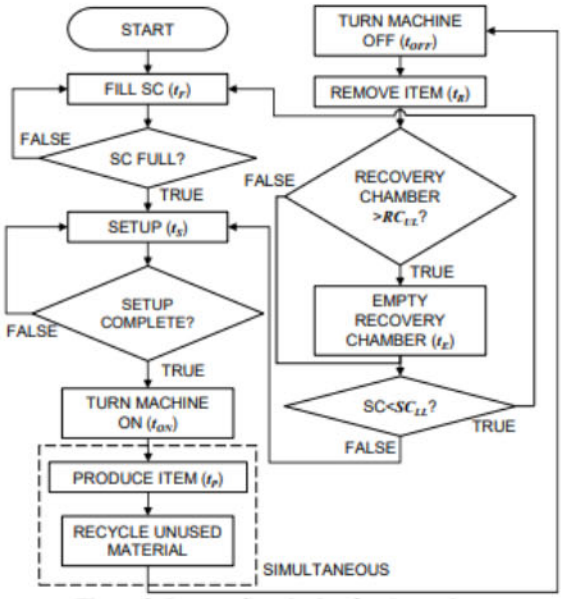

Figure 5. Python Flow Chart [9]

With the defined variables, the simulation began. These variables were introduced directly in the simulation.

The purpose of the simulation was that, through the reception of data from an external source, data from a real physical process, be analyzed to yield results about the production process of the machine, as well as its performance and capacity, for later analyze the simulation runs and use the data thrown to optimize and improve the physical machine.

The steps for the Python simulation are shown in Figure 6. Step one is make the HTTP request and save the data into a string array and then into the variables that represent the operations of the machine. Then, through the use of string arrays the times of the operations were stored in variables and once all variables have been placed the principal code for the DES was coded with python language and different libraries such as Simpy and RequestHTTP. At the end of DES execution, the results of the main parameters (cycle time per item, items created, remaining material in SC and recovered material in SC.)

Once the results of the DES have been obtained, they can be used to perform an optimization of different parameters of the process, in order to maximize the number of items created by each execution of the machine during one hour of work.

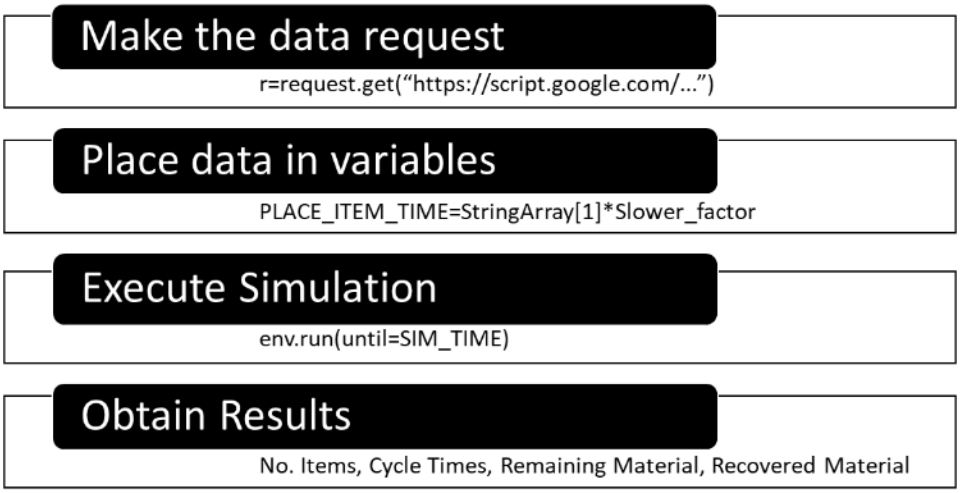

Figure 6. Python Structure

The framework created for the DES coded using Python has seven main parts in its structure as shown in Figure 7: Libraries selection, Request, Variables definition, Processes times definition, Machine environment, Methods and Show results.

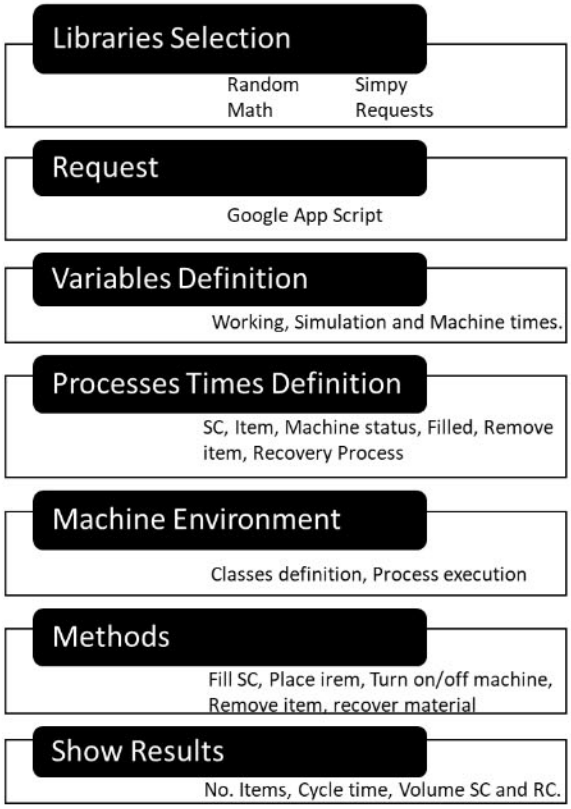

Figure 7. Code structure [9]

- Libraries selection. In this initial part, four different libraries of Python were selected to work with: Random, Simpy, Math and Requests. *Random* library allow to use pseudo-random numbers with statistical distributions. Simpy allows to create DES environment, parts and actions of the represented process. Math includes mathematical functions in C standard and Requests allow to make HTTP requests. The libraries into the code are shown in Figure 8.

```
1  import random
2  import simpy
3  import time
4  import requests
5  from math import exp
```

Figure 8. Libraries of Baseline Code

- Request. In this part of the code a request was programmed using a tool from the library *Requests,* HTTP Request. This tool allows the Python code to make a call to the information stored in the provided link. A Google App Script was previously done and fed with the information of the process. The way that the request is in the code is presented in Figure 9.

```
r = requests.get("https://script.google.com/macros/s/AKfycbwVlOuXyiDjJrwOqrFy-ckeVgy7KtbmDtmTM_rsZQYV4kWKBgZ-/exec")

print(r.status_code, r.text)

Times=[int(e) if e.isdigit() else e for e in r.text.split(',')]
print(Times)
```

Figure 9. HTTP Request

- Variables definition. In this section the variables of the machine are declared as in Figure 10. These variables are the operations executed in the machine and start with the Time that the operator takes into filling the SC, the time that takes to make the setup of the homogenizer box, the duration of turning the machine on and off, the duration of creating an item, the time that the operator takes into remove the item and the duration of move the recovered material into the SC. Once the variables were defined, the variables for the sigma of each operation were added to the code to add variation to the process. A slower factor variable was defined to represent a slow operator in the process and a variation factor was added to increase the variation of the operation times. The time of the execution represents 8 hours of work, that equals to 480 minutes that the machine is doing the process.

```
WEEKS = 1 / 5
SIM_TIME = WEEKS * 5 * 8 * 60 * 60
Slower_factor=1
Variability_factor=1

TANK_T = Times[0]*Slower_factor
PLACE_MOLD_T = Times[1]*Slower_factor
MACHINE_ON =Times[2]*Slower_factor
FILL_MOLD = Times[3]*Slower_factor
MACHINE_OFF = Times[4]*Slower_factor
REMOVE_MOLD_T = Times[5]*Slower_factor
RECOVERY_T = Times[6]*Slower_factor

TT_SIGMA = Times[7]*Variability_factor
MT_SIGMA = Times[8]*Variability_factor
MON_SIGMA = Times[9]*Variability_factor
FM_SIGMA = Times[10]*Variability_factor
MOF_SIGMA = Times[11]*Variability_factor
RM_SIGMA = Times[12]*Variability_factor
RT_SIGMA = Times[13]*Variability_factor
```

Figure 10. Variables of Time and Sigma

- Processes time definition. Methods were defined representing all the operations of the machine and the operator. These operations are: The functioning of the Storage Chamber, Place the item, do the setup of the homogenizer box, turn on and off the machine, remove the finished item and move the recovered material from RC to SC. In this section the times and sigmas requested from the Google App Script are saved into specific variables to calculate through a statistical distribution (normal) the real times with variation. An example is presented in Figure 11.

```
54  def tank_process():
55      return random.normalvariate(TANK_T, TT_SIGMA)
56
57  def mold_process():
58      return random.normalvariate(PLACE_MOLD_T, MT_SIGMA)
```

Figure 11. Example of defining methods

- Machine environment. This part was coded to create a “class” named Machine, as a representation of the physical machine. In this class, the states, processes and variables of the process were included and set to the initial state (for the SC to 100 of material, the machine turned off, etc.). The class defined was coded with two states: working and resting. In the first state the machine executes all the process as normal, from the start to the end and in the other state the machine stays in a standby mode.
- Methods. In this point it was defined in Python language how each of the operations of the physical process works, that is, the function structure of each operation was created from the operation “filling of the storage chamber” to the “recover material” part. The methods are within the Machine environment, specifically in the Working part.
- Show results. At last the results of the simulation are presented in the console of Python, showing information of cycle time per item, level of material in SC, level of recovered material in RC and items created during simulation time.

3.2.2 Design of comparison in Plant Simulation

General Structure

Once the simulation environment was created in Python, it was replicated using a commercial software named Plant Simulation. This software allows the user to model, simulate, analyze, visualize and optimize processes and systems, flow of material and logistic operations. The followed structure is shown in Figure 12, starting with Object Selection and following with Path Creation, Controls Definition, Methods Definition, Displays and Tables and Export Data.

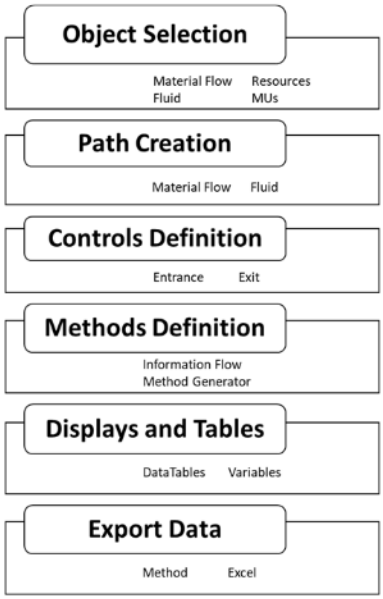

Figure 12. Model Structure

- Object Selection. The first step in Plant Simulation is to select the objects that are going to be used in the DES. For this, objects were selected from Material Flow (Sources, Stations, Store), Fluids (Tanks), Resources (Workplaces, Workers, Workpool, Exporter, Broker) and MUs (Entities). These objects represent the main parts of the process being simulated (machines, human resources, materials).
- Path Creation. In this step was defined the flow of the material through all the process, using Material Flow objects (Connectors) and Fluids (Pipes).
- Controls Definition. Some of the objects previously inserted into the simulation model can have several controls. In this step the control of those objects was

defined, allowing to determine whether the entrance or exit of material as seen in Figure 13.

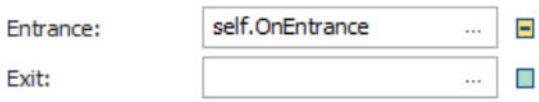

Figure 13. Example of Entrance Control

- Methods Definition. Once the objects, controls and paths were defined, it was necessary to create the Methods of how the simulation was going to work. From the Information Flow objects, Methods and Generators were used to program the flow of the material, the relation between different machines, and all the restrictions that were previously determined in the Python simulation. Plant Simulation has its proper programming language, Simtalk 2.0. A visualization of the Methods and Generators in Plant is shown in Figure 14 and an example in Figure 15.

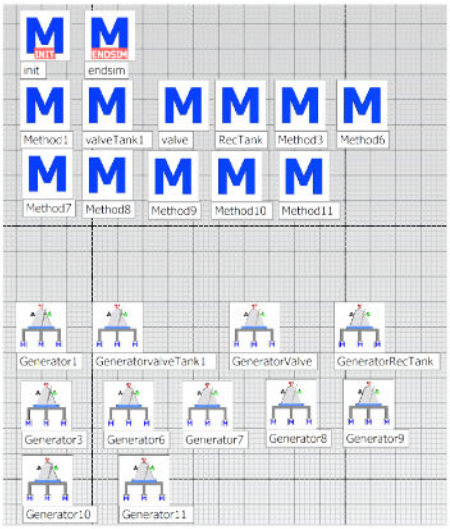

Figure 14. Methods And Generators

```
var i:integer;
var obj:object;

DataTable2.delete

for i:=1 to current.numNodes loop
    obj:=current.node(i)
    if obj.class = .MaterialFlow.Station
        current.DataTable2.writeRow(1,
            current.DataTable2.YDim+1,
            obj.name,obj.statWorkingPortion);


    end
next
current.DataTable2.writeExcelFile("C:\Users\febro\Documents/PlantSimulationFiles/Stat.xlsx")
```

Figure 15. Example of SimTalk Method

- Displays and Tables. From Information Flow objects, DataTables and Variables were added to the model in order to store data generated through the simulation run, like Process times, Cycle Time, Items produced, etc.
- Export Data. Finally, the data stored in the different DataTables was sent to Excel, using specific methods. Once the data was exported, it was analyzed and graphed.

The categories, the objects and their functions are detailed in Table 3.

Table 3. Functions of objects

CATEGORY	NAME	FUNCTION
Material Flow	Connector	Connect objects in the model
	Source	Produces Mus in a single station
	Station	Process a part in a lapse of time
	PickAndPlace	Pick a part at one station and places it in another
	Store	Stores any number of Mus
Fluids	Pipe	Transfers flowing material between fluid objects
	Tank	Temporarily stores a single material
	Portioner	Transform a Fluid into a solid MU
	DePortioner	Transform a solid MU into a Fluid
	MaterialsTable	Define properties of fluid materials

Resources	Workplace	Place at station where Worker performs its job
	WorkerPool	Where the workers are stored
	Worker	Represents a person who works on a Workplace
	Broker	Where the services of workers are defined
InformationFlow	Method	Queries information from an object and returns a value
	Variable	Can be defined with a value
	DataTable	A list with 2 or more columns
	Generator	Activates Method objects at a specific defined time
UserInterface	Display	Shows a value at all times the simulation run
MUs	Part	Moves through material flow objects
	Container	Tansports other MUs

Model Design

Plant Simulation is a very complete program, whose function is to create simulation models that can represent isolated / individual processes, an entire production line or all the processes within a plant. However, this simulator has limitations in some objects that cannot be used in a versatile way because they have very specific functions. That is why the present simulation adapt the utilized objects to represent the desired process.

The first step was to add these objects: Two *Tanks*, *DePortioner*, two *Buffers*, two *Portioners*, *Store*, three *Stations*, *PickAndPlace*, *Source*, *WorkerPool*, *Broker* and six *WorkPlaces*. Then, three *Pipes* were added to the model and using *Connectors* the flow of the material throughout the process was defined, as seen in table 4.

Table 4. Connections

Connect From:	To:
Tank1	Pipe1 Pipe2
Pipe 1	Portioner1
Portioner1	Station
Station	PickAndPlace
PickAndPlace	Buffer

Buffer	Store
Pipe2	Tank
Tank	Pipe
Pipe	Portioner
Portioner	Buffer1
Buffer1	Station1
Station1	DePortioner
Deportioner	Pipe3
Pipe3	Tank1
Station2	DePortioner

Once having all the connections made, the *WorkPlaces* were assigned where needed, defining this in the Attributes of the object and marking “Worker stays here after completing the job”, as presented in Figure 16.

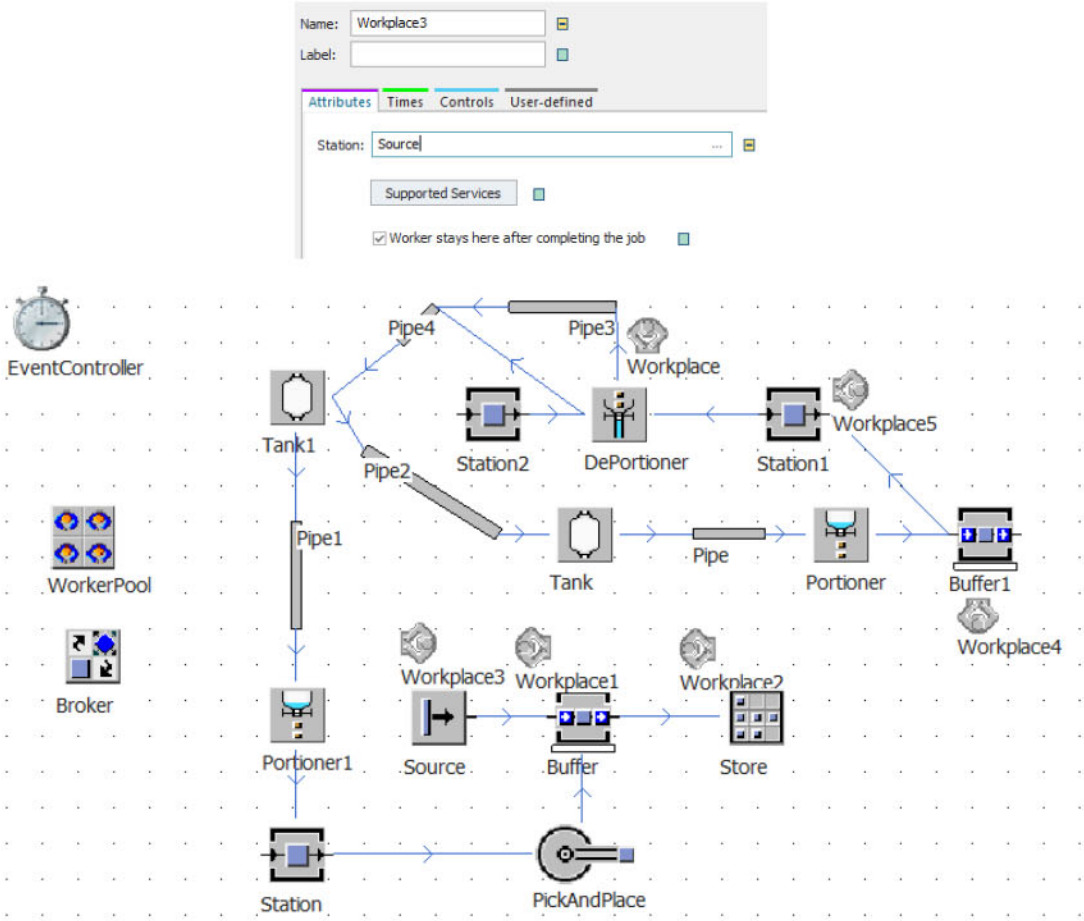

Figure 16. Assignation of WorkPlaces

Each object has its own attributes, like processing time, amount per MU, Capacity, etc. These attributes were defined in each of the objects inserted in the baseline model, as follows in Table 5.

Table 5. Attributes of Objects

Attribute	Objects											
	Tank	Tank1	Portioner	Portioner1	Station	DePortioner	Buffer	Buffer1	Store	Pipe	Pipe1	Pipe2
OutflowRate	1	1	-	-	-	100	-	-	-	100	10	2
Volume	100	140	-	-	-	-	-	-	-	-	-	-
MU	-	-	Part	Part	-	-	-	-	-	-	-	-
Amount per MU	-	-	90	10	-	100	-	-	-	-	-	-
Fluid from predecessor	-	-	1	1	-	-	-	-	-	-	-	-
Processing Time	-	-	-	-	Normal 0:50 0:02 0:02 0:50	-	-	-	-	-	-	-
Fluid depends on	-	-	-	-	-	Fixed	-	-	-	-	-	-
Material	-	-	-	-	-	Part	-	-	-	-	-	-
Materials Table	-	-	-	-	-	Materials Table	-	-	-	-	-	-
Setup Time	-	-	-	-	-	Normal 0:32 0:02 0:02 0:32	-	-	-	-	-	-
Capacity	-	-	-	-	-	-	1	1	-	-	-	-
Buffer Type	-	-	-	-	-	-	Queue	Queue		-	-	-
X dimension	-	-	-	-	-	-	-	-	10	-	-	-
Y dimensión	-	-	-	-	-	-	-	-	40	-	-	-
Z dimension	-	-	-	-	-	-	-	-	1	-	-	-

Different controls were defined in the objects. The controls allow an object to realize specific tasks as define how or when the material or an item enters or exit a station, define the setup times for a machine, or the way the items are being stored (frequency, pull, push, etc.). For the simulation the control were defined in four objects: *Station*, *PickAndPlace*, *Buffer* and *Store*. For the *Station,* a control was defined on the Entrance and is shown in Figure 17.

```
var myPatient : string := to_str(@)
var index:integer:=DataTable.ydim+1
   if timeOfDay(eventController.absSimTime) >= 0
    DataTable["EntryT",index] := EventController.Simtime
    DataTable["ExitT",index] := EventController.Simtime + station.procTime
    DataTable["ProcTime",index] := Station.ProcTime
```

Figure 17. Entrance Control for Station

This control is a *Method* that stores in a *DataTable* the data generated in *Station:* Process Times of each item that passes through Station, and the times which the item enters and leave the Station.

For the *PickAndPlace* Target control was defined. This method specify that the PickAndPlace has to wait until the setup was done (container in place), to place the item (part) on the container. The control defined in the *Buffer* checks if the container is already in the buffer, waits for the container to be filled by the *PickAndPlace* and once occupied moves it to the next station. In the store, a control was defined to record the number of items that were already produced. This process was done by declaring a Method in the Entrance Control, establishing that the name and the time of each item entering the store will be recorded in a DataTable.

For the processes to follow the flow of the real process within the simulation, it was necessary to use Methods in different operations of the simulated process. The methods can work independently and automatically for some operations and simple processes, but in this case the majority of methods were used within Generators.

Methods

The methods that were used without the necessity of an activation with a Generator were the ones to move data into DataTables and Excel Sheets. These methods are shown in Table 6.

Table 6. Methods of Model

Method	Function
init	Define the inital state of the objects in the simulation.
Method	Write in DataTable number and time of item created.
endSim	Define actions made at the end of the simulation.

The init method is the first one that is triggered when the simulation starts. In the model, the init method structure is as follows in Figure 18:

```
Pipe3.pipeOpened:=true
Pipe1.pipeOpened:=true
Pipe2.pipeOpened:=true
Pipe.pipeOpened:=true
Tank.entranceLocked:=false
Tank.exitLocked:=true
DataTable.delete
Stats.delete
wait 100
.MUs.Molde.create(Moldes)
```

Figure 18. init Method

At the start of the simulation all the pipes are open, the entrance of the Recovery Chamber (RC) is open, the exit of RC is locked and all previous records from DataTable are deleted. Then after 100 seconds of simulation pass, the first part for setup is created, after the operator has filled the Storage Chamber (SC) with material. The Method called "Method" is assigned to the Entrance Control in the *Store*. It writes in "DataTable" the number of each of the created items and the time that an item entered the *Store*. endSim Method declares that at the end of the simulation run, the Table called DataTable is written into an Excel File.

Generators

A Generator is a tool that within its Control (Interval or Duration) has a Method assigned to it. In the Generator times, the Start, Stop, Interval and Duration times are defined. The Generators and Methods used are summarized in Table 7.

Table 7. Generators

Generator	Times		Controls		Function
	Start	Interval	Interval	Duration	
3112	0:01	1:00	Method9	self.OnDuration	Create material to refill Storage Chamber.
31121	0:01	0:02	Method10	-	Delete material if SC full.
311211	0:01	0:02	Method101	-	Detele mat. if SC and DePortioner occ.
4	0:02	0:30	Method3	0:01	Creation rate of Pallet for Setup Op.
0	0:02	0:02	valve	-	Open/Close pipe2 depending on buffer.
1	0:02	0:02	RecTank	-	Close pipe2 depending on Buffer1.
2	0:02	0:02	valveTank1	-	Open/Close Pipe1 depending on buffer.
3	0	1:00:00	0:20	-	Save statistics of Store, Station and Buffer.
31	0:02	0:02	Method6	-	Control the exit of material in RC.
311	0:00.5	0:00.5	Method7	-	Allow the exit of material in RC.
3111	0:02	0:02	Method8		Close Pipe2 depending on RC.

Each Generator has Start and Interval times which are specified to determine when the Generator is starting to work and at which rate time is being executed after activated. For example, Generator 3111 was activated since the first second of simulation run and it is executed every two seconds, also running the Method10 that checks if the Storage Chamber (SC) is full and deleted the material if true.

It should be noted that the use of a PickAndPlace Robot was only made for the simulation to work and represent correctly the real process. The PickAndPlace robot is representing the flow of material between two objects, the Station and the Buffer. Also the DePortioners, Portioners, and Buffers are intermediaries in the whole process. They were necessary to make the flow of material through the process, that is why these objects have no operating time and only allow the passage of material. The Stations are also intermediaries in the process, but "*Station"* does have processing time, representing the times of different operations of the process: Produce item, Remove item and Empty RC. The "*DePortioner*" does have setup time, representing other operations of the real process: Fill Storage Chamber (SC), Setup, Turn On/Off Machine.

DataTables

These objects are lists of information that can have two or more columns. Each column can have its own data type (Boolean, Integer, String, Real, Object, Table, List, Stack, Queue, Time, Money, Length, Weight, Speed, Acceleration, Date, DateTime). The DataTable used in the model stores information from PickAndPlace, Station and Method in six different columns: Mold, EndTime, ProcTime, EntreT, Robot_entry and ExitT. Each column has different Data Type and is shown in Table 8.

Table 8. Data Type of DataTable

Column	Name	Data Type
1	Mold	String
2	EndTime	Time
3	ProcTime	Time
4	EntryT	Time
5	Robot_entry	Time
6	ExitT	Time

3D Representation of Process.

The models made in Plant Simulation can also be represented in a 3D visualization. The Figure 19 is presenting the final model of the real process, presented in the environment of Plant Simulation. As seen, there is a human operator in the top right side located in its workplace. The SC is represented by the biggest *Tank* and the RC by the little one. A *Buffer* is representing the Machine, and the *Store* is where the operator puts the final items. The two Chamber had fill levels to observe how their capacity is going. Other objects showed in Figure 16 (Methods, Generators, Variables, DataTables) are hidden in the final representation.

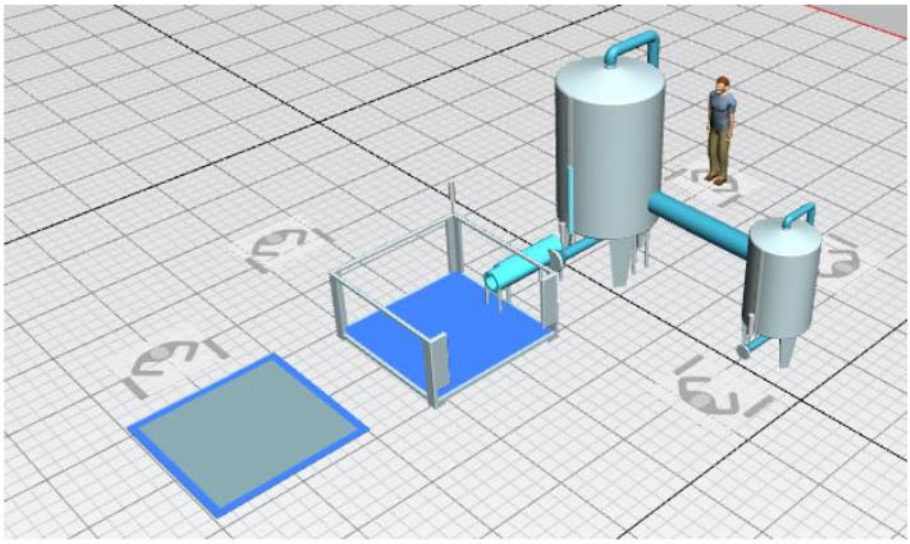

Figure 19. 3D Visual Representation

3.2.3 Bayesian Inference.

In order to test as a demonstration that the simulation elaborated in Python is capable of being analyzed in different ways, a Bayesian technique was used, using data generated in the simulation itself.

The Naive Bayes algorithm it's based on Bayes Theorem. It represents a classification procedure with an assumption of independence among predictors. It considers each operation as a mutually exclusive state or hypothesis. According to Eq. (1), the algorithm found the probability of a delay in the process P_D given that an operation in the process was delayed, identified as **x**.

$$P(P_D|x) = \frac{P(x|P_D) \cdot P(P_D)}{P(x)}$$

$j\ \{1, \ldots, 7\}$

Equation 1. Naive Bayes Eq.

The technique applied to this process is intended to analyze its results in order to forecast the level of probability that there are delayed items in the entire process, having found delayed items somewhere in it. This will allow identifying which part of the process must be attended or which section is the bottleneck, in order to improve or decrease it.

In the original Python simulation, an addition was made to the final part of the code so that the data provided by DES was stored as a DataFrame within a local variable in the same simulation, in order to group them into columns and rows to its subsequent evaluation with the Bayesian technique.

As seen in figure 20, the program begins by calling the data previously collected and stored in the previous simulation, which correspond to the data of Figure 5.

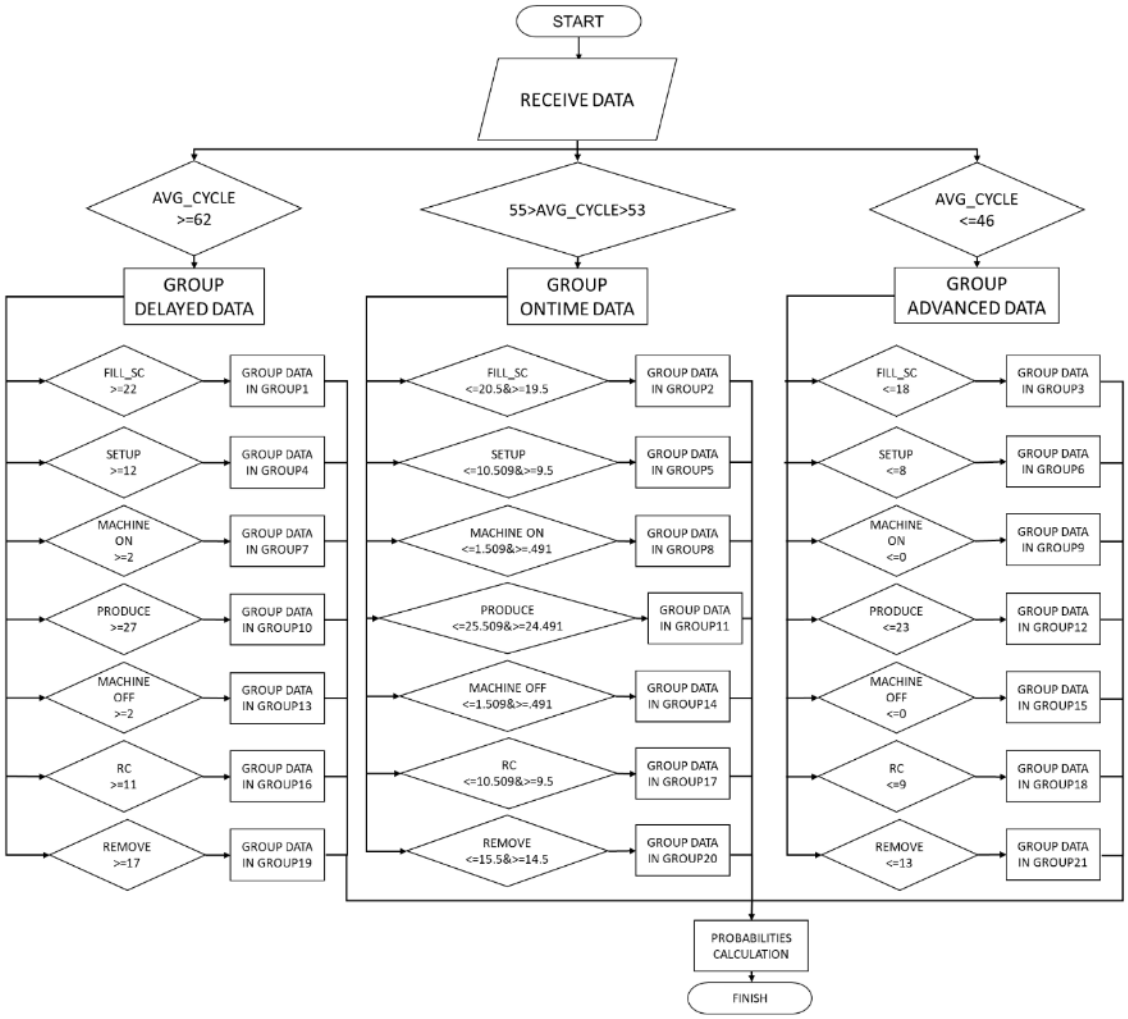

Figure 20. Flow Chart of Bayesian Analysis in Python

Then, these data is grouped into three different variables (Delayed Items, Advanced Items, Items On Time). From these three groups of data, a sub-grouping of data is performed once again to identify delayed items, on time and out of time within each of the processes. Subsequently, the program analyzes each of the data groups and calculates probabilities for each of them, following the Naïve Bayes methodology. Finally, the forecast results are presented in the console.

3.2.4 Main Python Scenarios

Based on the Baseline DES done in Python and to guarantee and demonstrate that the Python model can be flexible enough to represent all different situations that the real process can perform before applying them, three primal scenarios were made in order to test different situations using the same variables as the initial DES:

- Baseline
- High Variation
- Slow Operator

I. Baseline

This scenario was the first one made. It represents the identical process of the real one. First, it was made in Simpy, with the programming language Python. The process for creating the structure and code for this simulation is explained in 3.2.1. It was necessary to store data from the process into Google App Script and then made a request in Python to call the data. After that, the code structure, shown in Figure 7, was developed. It represents the whole real process, and how it works from the entrance of material into the Storage Chamber (SC) until the products are produced and the Recover Chamber (RC) is emptied. The values used in Python Simulation are shown in Table 9.

In baseline, both for Python and Plant Simulations, the Sigma used were 2 seconds and a normal distribution to represent the little variation of operation times.

II. High Variation

High Variation Scenario represents the same real process of fill the Storage Chamber (SC), do the Setup, Create Items, etc. In this model a different variation factor is used. The times of each operation remain the same. The sigma for each operation time is different from baseline, going from two to six. The data variables is shown in Table 9. The Flow chart for this scenario remains the same as in Figure 5.

III. Slow Operator

Slow Operator Scenario involves the process presented in baseline simulation and in Figure 5. In this model is represented a slow human operator. For that, a Slower Factor was introduced into the source code of the Python simulation. The value of this variables is two and was intended to multiply by 2 all the operation times of the process. The data variables are shown in Table 9.

3.2.5 Other Scenarios

In order to ensure that the DES was representative enough to simulate various changes in the process, three other scenarios were created where the variables, operations, and other elements of the main process were modified or added.

- Multiple Machines with the same process.
- Multiple machines with one general storage chamber.
- New process added to the baseline.

IV. Multiple Machines with the same process.

This simulation was made using the same structure of the DES made in Python. Modifications were made into the code of the DES, to allow the user to enter the number of machines that are involved in the process and proceed with the evaluation. The data variables for this scenario are shown in Table 9.

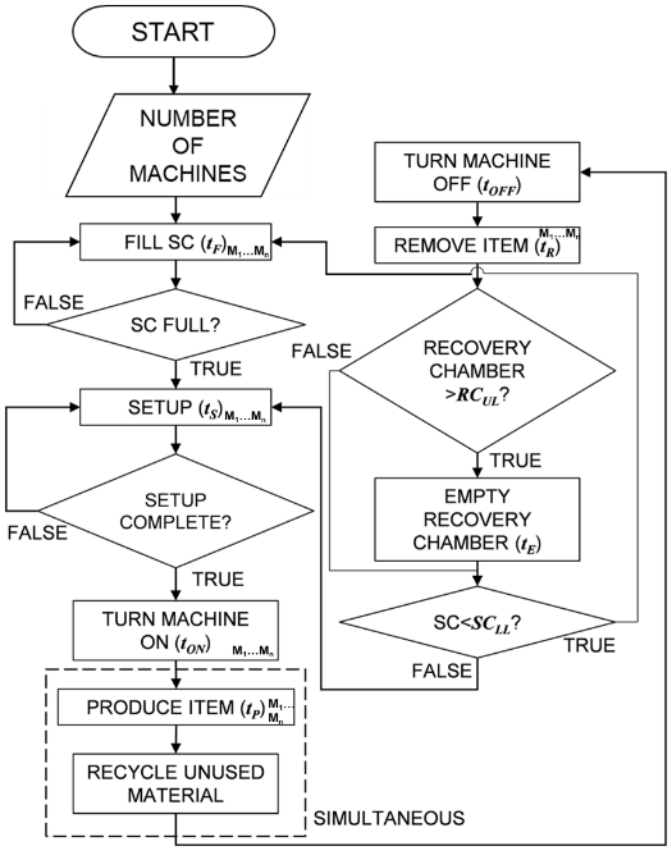

Figure 21. Multiple Machines flowchart

The time data required for the simulation are being obtained in the same way as in the baseline simulation, with the HTTP Request and the Google App Script link to save the data in specific variables. The rest of the process continues the same as in the baseline DES as seen in Figure 21.

V. Multiple machines with one main storage chamber.

In this scenario, what was sought to simulate was the baseline process with multiple machines and only one storage chamber. For this, the code adapted for the previous scenario of multiple machines was taken and the part of the program where the material is deposited in the storage chamber and sent to the different machines was modified so that the chamber now functioned as one for n machines. The process for this scenario is in Figure 22. The data variables for this scenario are shown in Table 9.

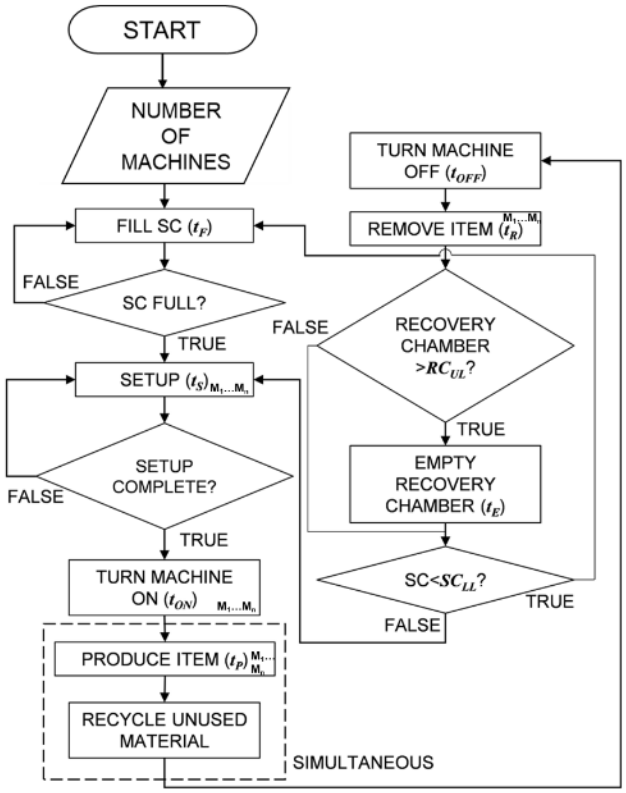

Figure 22. Multiple machines with one general SC flowchart

It is important to mention that in this scenario like in the others, one operator is who realizes all the activities involved in the process. The difference here resides after the operator has filled with material the Storage Chamber (SC). It goes and does the setup in the first machine, turn on it and then goes to the second machine until it finishes with all of them.

VI. New process added to the baseline.

For this last scenario, it was used the simulation of scenario I, where there can be multiple machines with similar processes. What was done was to add a new operation to the machine process. As seen in figure 23, the simulation starts in the same way, requesting the data on the number of machines from the user and fetching the time data of each process from each machine from a Google App Script. The first task of the operator is to fill the storage chamber with material, and then the added process, “crushed”, enters. Once the material is inside the storage chamber, the operator starts

the crushing process, it is carried out and turns it off. From this point on, the simulation continues as in the baseline. The data variables for this scenario are shown in Table 9.

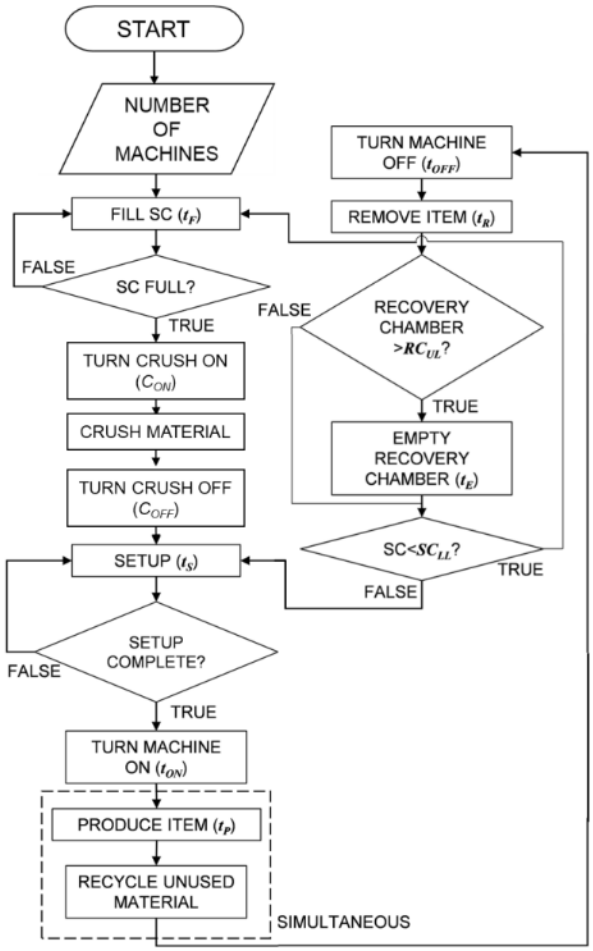

Figure 23. New Process Added flowchart

The data necessary to work with each of the simulations developed was determined using an approximation of criteria that could take the operator to perform each task and the machine to execute the operations. Different operating times of each process were collected, as well as the sigma values of each one of them; these data were used in baseline simulation. The machine parameters are the average times of operation as well as their respective sigma. For the simulation with high variation, it was required to modify the sigma values that were used in the baseline simulation. That is why some of the values were increased to a sigma of 6, adding variation to the times of each operation.

For the simulation with slow operator, it was decided to add to the baseline simulation a variable called “Slower Factor” with a value of 2. Its function in the simulation is to multiply by 2 to each of the values of the operating times, simulating with this that the operator is slower when performing its functions.

Each of the values are summarized in Table 9.

Table 9. Data for Python Simulation Scenarios

VARIABLES	Baseline	High Variation	Slow Operator	Multiple Machines Same Process	General SC	New process added
Machines in process	1	1	1	2	3	1
Feed SC	20 sec	20 sec	40 sec	20 sec	20 sec	20 sec
Configure	10 sec	10 sec	20 sec	10 sec	10 sec	10 sec
Activate process	1 sec	1 sec	2 sec	1 sec	1 sec	1 sec
Create product	25 sec	25 sec	50 sec	25 sec	25 sec	25 sec
Deactivate process	1 sec	1 sec	2 sec	1 sec	1 sec	1 sec
Move finished product	15 sec	15 sec	30 sec	15 sec	15 sec	15 sec
Empty RC	10 sec	10 sec	20 sec	10 sec	10 sec	10 sec
New Process On **New Process** **New Process OFF**						1 sec 25 sec 1 sec
Feed SC	2 sec	6 sec	2 sec	2 sec	2 sec	2 sec
Configure	2 sec	6 sec	2 sec	2 sec	2 sec	2 sec
Activate process	1 sec for all scenarios					
Create product	2 sec	6 sec	2 sec	2 sec	2 sec	2 sec
Deactivate process	1 sec for all scenarios					
Move finished product	2 sec	6 sec	2 sec	2 sec	2 sec	2 sec
Empty RC	1 sec for all scenarios					
RC upper limit	90% for all scenarios					
SC lower limit	20% for all scenarios					

3.2.6 Main Scenarios Comparison

After being designed the Baseline DES and the main scenarios in Python, through the representation of Plant Simulation they were compared. As mentioned in Introduction, Plant Simulation is a simulation software capable of represent processes from reality in a very complete way, but with limitations such as the difficulty to connect the Plant models with cloud services, or to export simulation results to other software or IoT tools. For that reason Plant Simulation was selected only to compare the Python DES, as follows.

Baseline Comparison.

In Plant Simulation DES, the Baseline Simulation was done as explained in 3.2.2. The utilization of objects, a framework and different methods was necessary to represent the reality and also the times of each process were fitted to the operations represented in the model. In Plant Simulation, each object required different configurations and attributes and are detailed in Table 10.

High Variation Comparison.

For the comparison, the Baseline sigmas were changed to a value of 1 to achieve greater variation.

Slow Operator Comparison.

For Plant Simulation model, the times used in Baseline were multiplied by two and assigned to their object.

The data used in Plant Simulation DES was the same as the used in the Python Simulation but distributed in different way between the machines of the Plant Simulation Model, in order to achieve the best representation of the process. The values of Plant Simulation Scenarios are described in Table 10.

Table 10. Plant Simulation Data

Attribute	Baseline		High Variation		Slow Operator	
	Station	DePortioner	Station	DePortioner	Station	DePortioner
Processing Time	Normal 0:50 0:02 0:02 0:50	-	Normal 0:50 0:01 0:02 0:50	-	Normal 1:40 0:02 1:40 0:50	-
Setup Time	-	Normal 0:32 0:02 0:02 0:32	-	Normal 0:32 0:01 0:02 0:32	-	Normal 1:04 0:02 1:04 0:32

The times are splitted just in two objects, Station and DePortioner because in those objects is where it is allowed to insert times, in order for the simulation to work correctly.

3.3 Optimization

To optimize the DES performed in Python, MatLab was chosen due to its faster analysis capabilities than Python as well as the ease of use and the friendly presentation of results. This integration of Python and MatLab demonstrates the flexibility and adaptation that DES code can have to interact with other IoT tools.

Within the simulations it was observed and determined that the variables that could be optimized were those related to the levels of both storage chambers. In order to determine the best relationship between the initial level of the Storage Chamber and the percentage of equivalence that the Recovery Chamber represents with respect to the Storage Chamber to Maximize the number of items produced, it was decided to carry out different simulation runs.

For this, the code of scenario V “Multiple machines with one main storage chamber" was used as the basis.

In the baseline simulation and in all the simulations that were carried out, the storage chamber fill value was 100%. The relationship between the Recovery Chamber and the Storage Chamber is 10%, that is, if the storage capacity of the SC is 100, that of RC is 10. This is called the Recovery chamber ratio. For the purpose of optimizing, within the Python simulation code, the storaged constant values of 100 SC and Recovery chamber ratio of 10 were transformed to constants.

To carry out the optimization, MatLab was chosen to work with because it is a software which presents optimization tools in a more user-friendly way than Python, in addition to allowing the creation of visualization graphics of results in a simple way.

3.3.1 MatLab Configuration.

1.- Call Python Code.

The first step was to bring the Python code to MatLab. The source code was added to the project folder. Then it was transformed into a text file saved into the same folder. Then a MatLab function stores the simulation results, about the items produced on average by each execution.

2.- MatLab Script.

A MatLab script was coded to run the executions and the optimization. It was divided in four sections: Combination Matrix, Get maximums, Optimize and Plot results.

- Combination Matrix: The first step of the script is to obtain the average results of items produced in each of the simulation runs derived from each of the combinations of SC init_level and RC tank_ratio. These results are stored in a matrix. The init_level and tank_ratio values are found in Table 11.

Table 11. Data for combinations

	Range
SC init_level	20 – 100 Interval: 5
RC tank_ratio	0.05 – 1 Interval: 0.05

The total combinations resulting of the iterations are 340 and are stored in a 17 x 20 matrix.

-Get Maximums: This part of the script analyzes the previous matrix finding the 5 maximum values of average items produced and store them with their values of init_level and tank_ratio. The 5 highest values found can have a different combination of init_level and storage / recovery ratio, that is why there can be more than five points in the final graph.

-Optimize: In this section two function were selected to execute the optimization. First the function *optimset* was selected to plot the results of the optimizations. In conjunction with optimset, the *fminsearch* function was used. *fminsearch* is a function that finds the minimum of unconstrained multivariable function using derivative-free method [15]. The operation of this function is as follows:

* Start by iterating with the Simplex algorithm of Nelder-Mead around initial values, ordering them from least (1) to greatest (+1).
* Each iteration discards the current worst point (+1).
* Then it accepts the new point.
* The function continues iterating until converging in a local minimum.

This algorithm finds the local minimum of a function. Because it is needed to maximize the average number of items produced on each machine, an arrangement was made in the script code so that instead of the algorithm discarding the maximum values of each iteration, the minimum ones are discarded.

Then the variable *fun* was created to work within fminsearch. The *fun* function stores the script CallPy, that brings the Python code into MatLab. Then, during the optimization, each maximum value is evaluated, searching for an optimization of maximization of the produced items.

-Plot Results: At this point, the found values obtained from the “Get Maximums” section and the optimized values are plotted in a visualization.

The whole process of MatLab is represented in Figure 24.

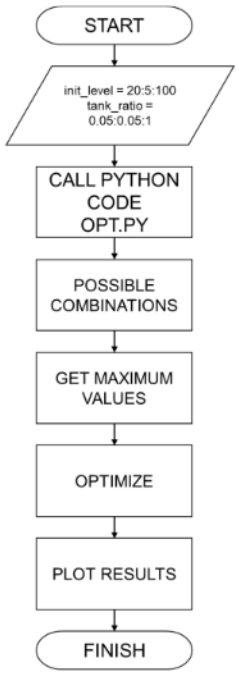

Figure 24. MatLab process flowchart

3.3.2 Optimization Data

Three different scenarios with different number of machines were tested and optimized. The data for each one of them are summarized in Table 12.

Table 12. Optimization Data

		Number of Machines		
		1	**3**	**5**
Process Times	Fill SC	20	20	20
	Setup	10	10	10
	Turn machine ON	1	1	1
	Produce item	25	25	25
	Turn machine OFF	1	1	1
	Remove item	15	15	15

Empty Recovery Chamber	10	10	10
fill_level	20 – 100 with intervals of 5		
tank_ratio	0.05 – 1 with intervals of 0.05		

The execution time was of one hour. It is worth mentioning that in order for the results of the optimizations to be representative, it was decided to use a variation of 0 within the processes.

3.4 System Communication

The parts that make up the system presented in this work communicate using different communication channels. Figure 25 shows the communication of the parts of the system as well as the data that is flowing through it.

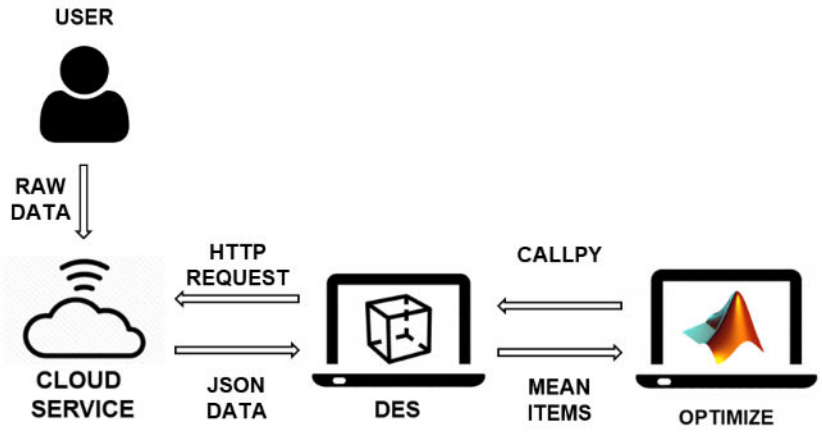

Figure 25. Digital thread

As seen in Figure 25 the system is made up of the user, the cloud service, the discrete event simulation and optimization. Each of these parties communicates with some other through the sending of different data.

User: In this part, the first data entry is made to the system. It is done by a user. The data were previously established and are made up of the times and standard deviations of each of the processes carried out on the machine of the case study. This information is entered into the system manually and directly to a cloud service.

Cloud Service: Here the data that was entered by the user is stored. A Google App Script was created specifically to empty the necessary data from the process and store it for later use.

HTTP Request: This request is made from Python to the Google App Script and executes the communication between Python and the cloud, calling the data from Google App Script and storing it into different variables in the code corresponding to the times and sigmas of the process.

JSON data: Once the request was made, Google App Script send the data in Json format to the Python DES.

DES: The discrete event simulation made in Python, make the HTTP request to the Google App Script and receive the data in Json format. Then, the information is distributed into the different variables concerning the simulated process.

For the optimization of DES, a connection of Python code with a script in MatLab was carried out, two communication channels were used for this:

CallPy: This is a script created in MatLab which has the function of activating the Python code locally, running an execution and collecting average values of items produced as well as the combinations of SC initial level and storage/recovery ratio.

Mean Items: In this part the data collected regarding average items produced in the execution of DES in Python are sent to the MatLab environment to continue optimization there.

Optimize: Here the collected Python data is analyzed by the MatLab optimization and script and the results are displayed graphically.

CHAPTER 4. RESULTS

This part shows the results of the simulation and the scenarios made in Python using Simpy and the ones made with Plant Simulation; describes the comparison obtained from Python and Plant Simulation; presents the results of the Bayesian technique and the results of the other scenarios developed in Python. Also presents the results of the optimizations executed. To explain this, it will be divided in different topics:

- Simulation results
- Comparison
- Bayesian prognosis
- Other Scenarios Results
- Optimization Results

4.1 Simulation Results

The results of the Python simulation are the main parameters of the process: Cycle time, remaining material, recovered material, number of items created and time of creation. A preview of the results of an execution in Python is shown in Figure 26.

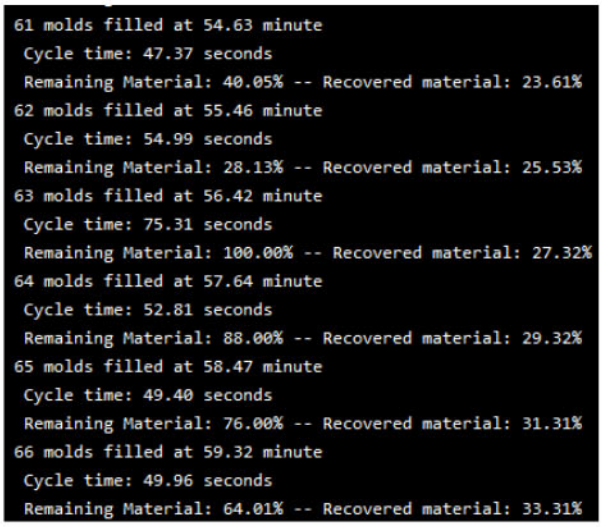

Figure 26. Results preview Python DES

Python simulation was performed several times, during one hour of simulated process in order to test the six different scenarios. The baseline scenario, slow operator, process with high variation, new operation added, one SC multiple machines and baseline with multiple machines. The results of these executions are shown in Table 13.

Table 13. Python Results

	Scenarios in Python					
	Baseline	**Slow Operator**	**High Variation**	**Baseline MultiMach**	**General SC**	**New process added**
Items	66	44	65	63	63	45
(%) Volume of SC	52%	76%	76%	51%	73%	64%
(%) Volume of RC	33%	85%	31%	50%	10%	87%

4.1.1 Main Scenarios Results.

The three main scenarios results were analyzed at first since they were the most common to happen in the real process. Three different analysis were made: Time per item, Level of SC nominal capacity and RC capacity.

Product Creation Time.

The Figure 27 shows the behavior of the time per item through 60 minutes of simulation.

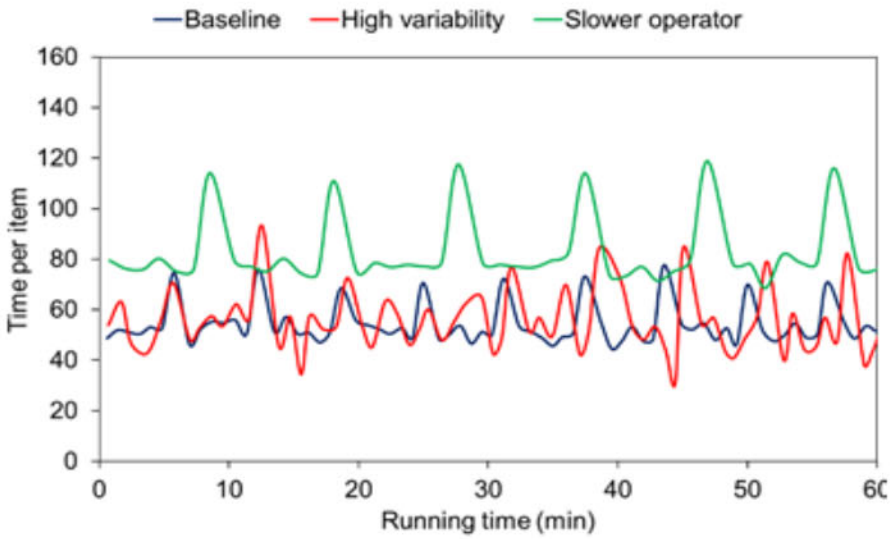

Figure 27. Product Creation Time (Main scenarios)

An average of the results per scenario was obtained, resulting in a behavior of 54 seconds per item in the baseline simulation, 82 seconds per item when there was a slow operator and 55 seconds per item with high variation, that is to say, when the entered sigma was higher. The results are shown in Table 14.

Table 14. Creation time

Scenario	Avg Duration (s)
Baseline	54
High Variation	55
Slow Operator	82

Level of SC nominal capacity.

The next analysis was between the level of capacity of the storage chamber and the hour of simulation and it is shown in Figure 28.

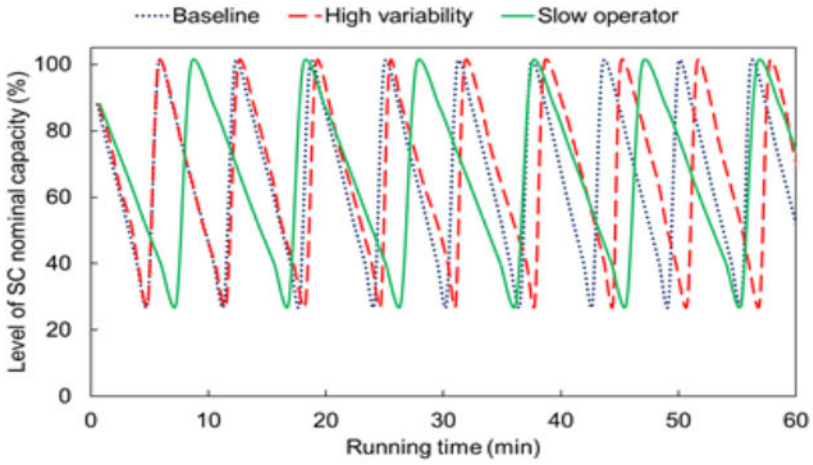

Figure 28. Level of SC nominal capacity % (Main scenarios)

The storage chamber level starts decreasing in each of the scenarios because of the production of items until it reaches the defined limit of 20% of the capacity in tank. Mean times of how long the SC takes to empty were obtained and are shown in Table 15. The difference is significant between the different scenarios only in a matter of time. Whereas in the baseline simulation the storage chamber takes approximately 5 minutes to reach its lower limit as in the DES with high variation, it takes 9 minutes to reach the same limit in the DES with a slow operator, because It takes more time to move around to perform their tasks adding extra time to the whole process.

Table 15. Average Duration of SC filling

Scenario	Avg Duration (min)
Baseline	5
High Variation	9
Slow Operator	5

RC capacity.

The last analysis shows the level of capacity of the recovery chamber during 250 minutes of simulation and is represented in Figure 29.

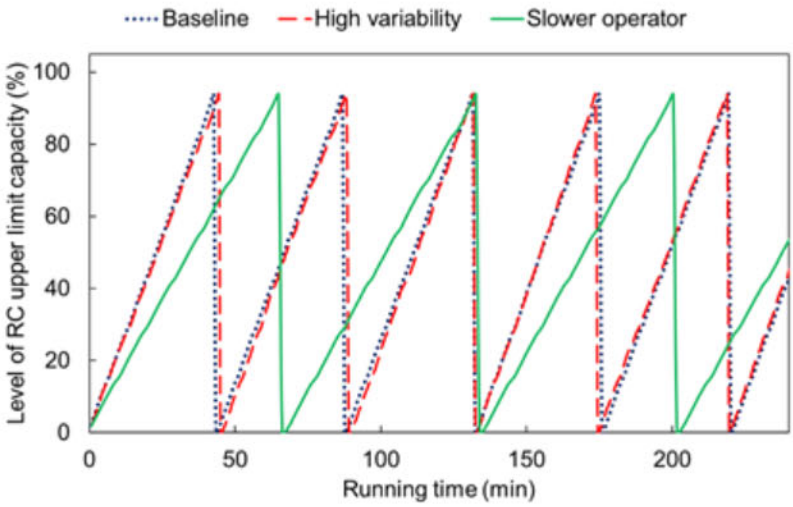

Figure 29. RC capacity % (Main scenarios)

In the three scenarios the recovery chamber upper limit is 90%. The difference is that during baseline and high variation DES, the process of filling the Recovery chamber is 43 min versus the 66 minutes of the DES with a slow operator. This difference happens because, as the operator is slower, the rhythm of the items will decrease and consequently the recovered material that fills the capacity of the recovery chamber will enter more slowly. The results are shown in Table 16.

Table 16. Average Duration of RC filling

Scenario	Avg Duration (min)
Baseline	43
High Variation	66
Slow Operator	43

4.1.2 Other Scenarios Results.

Multiple machines with the same process

This scenario was tested by running the simulation with two machines performing the same baseline process each and with a single human to operate both machines. The results of the run show the behavior of the production of items for one hour in Figure 30. On machine 1 the item with the longest creation time was 37 with 77.84 seconds and the lowest was item 53 with 46.47 seconds. On machine 2, the item with the longest time was 52 with 78.16 seconds and the lowest was 61 with 46.27 seconds.

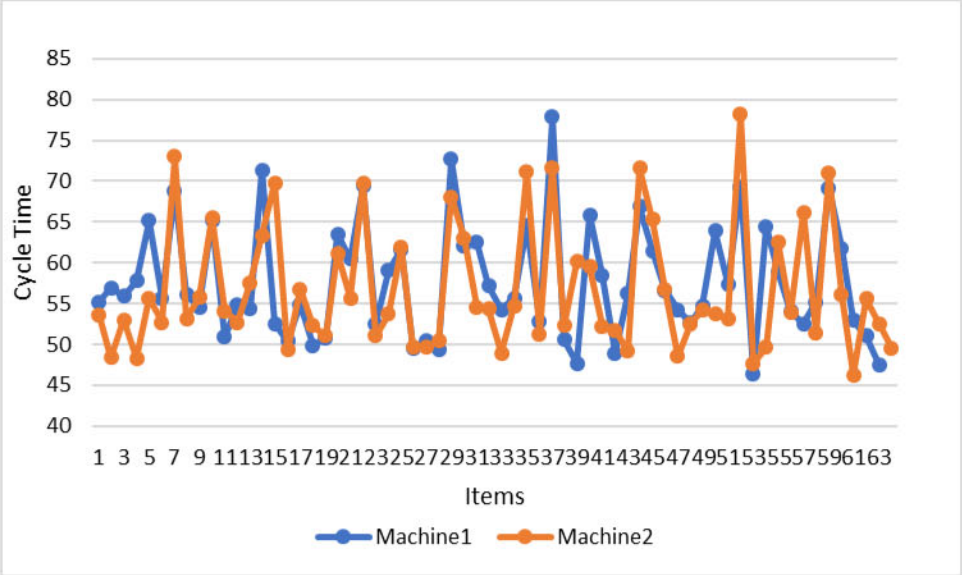

Figure 30. Multiple machines – Same process results

New process added to baseline

This scenario was tested by running the simulation with a machine performing the same baseline process but incorporating a new operation into the machine process. The new operation involves turning the crush on, crushing the material, and turning it off. The results of the run show the behavior of the production of items for one hour. As can be seen in Figure 31 the items produced at the end of the operation were 44.

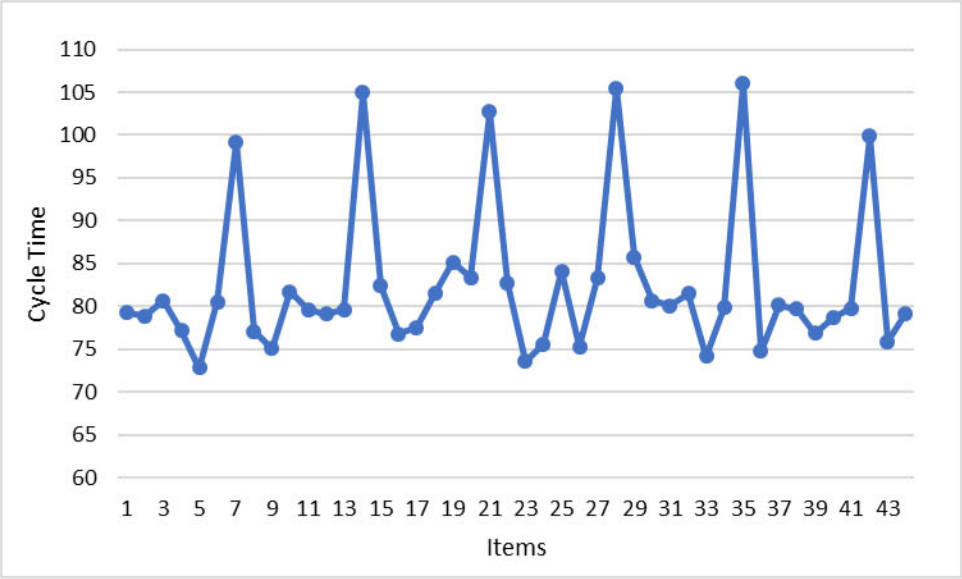

Figure 31. New process added results

Multiple machines with one general storage chamber.

This scenario was tested by running the simulation with three machines, performing the same baseline process with the difference of having a storage chamber for N number of machines. The results of the run in Figure 32 show the behavior of the production of items from the 3 machines for one hour and being operated by one person. Machine 1 produced 63 items and Machine 2 and 3, 62 items.

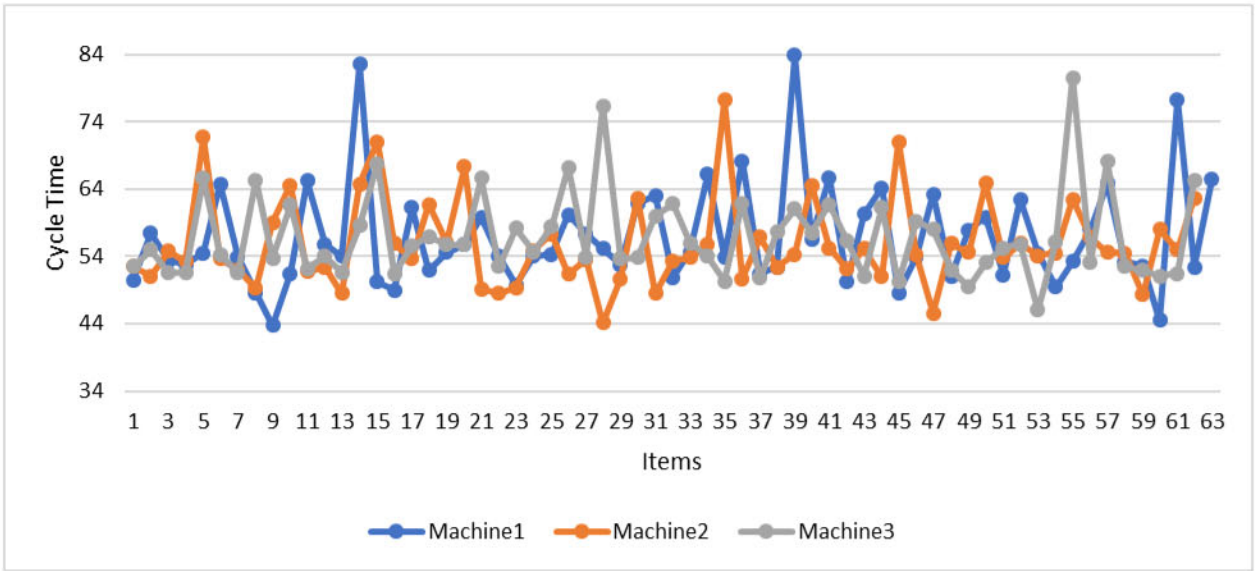

Figure 32. Multiple Machines – One SC Results

The individual results of each scenario were made to show in a separate way the diversity that each scenario can offer. Other comparisons were plotted, showing the behavior of the six scenarios at the same time in different categories, as follows.

Product Creation Time.

These results show the six scenarios tested. In Figure 33 it is represented 30 minutes of process, to allow a better view of the behavior of the lines.

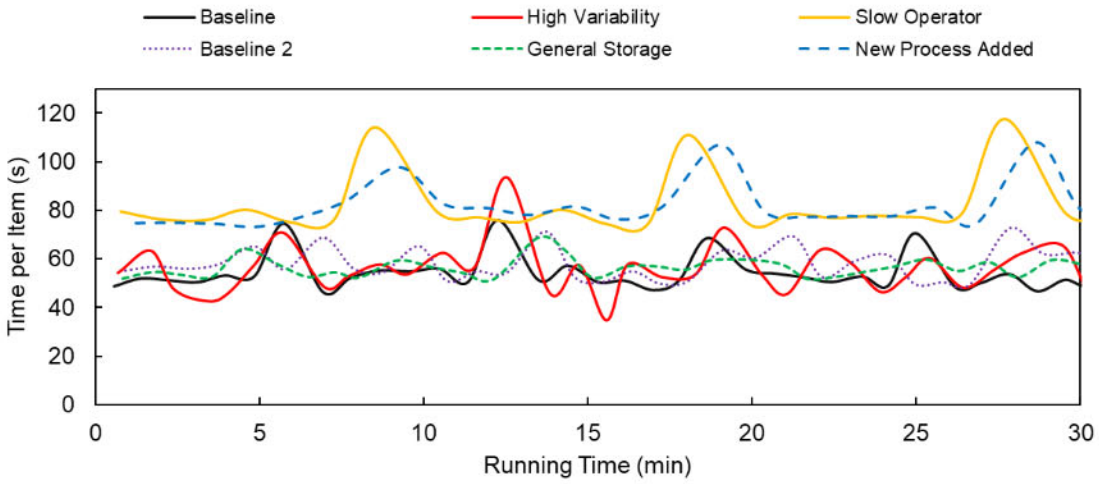

Figure 33. Product Creation Time (All scenarios)

Level of SC nominal capacity (%).

Here, the results of the Storage Chamber capacity in percentage are shown during 30 minutes of simulation. The behavior of each of the six scenarios can be seen in Figure 34.

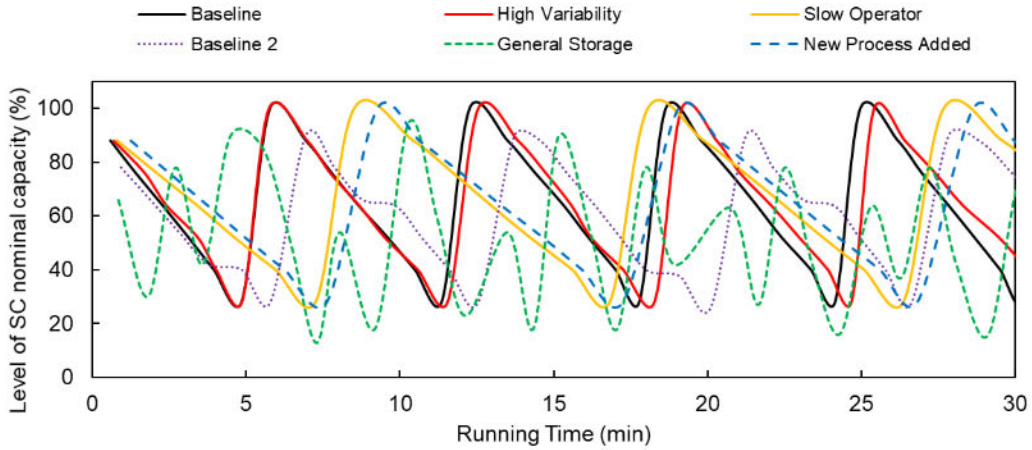

Figure 34. Level of SC nominal capacity % (All scenarios)

RC capacity (%).

This result is showing in Figure 35 the behavior of the Recovery Chamber capacity during 60 minutes of simulation.

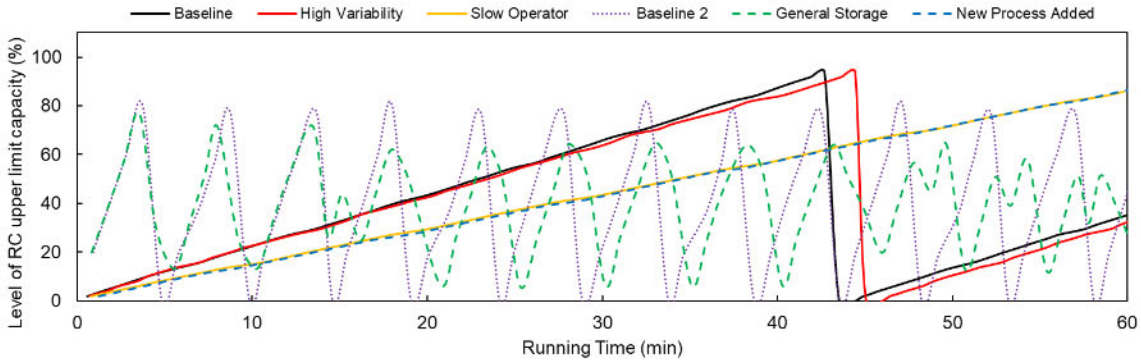

Figure 35. RC capacity % (All scenarios)

It is worth to mention that in the last two results there are notable differences in the other scenarios added. The reason for this to happen is because these scenarios are configured in different ways than the main ones, to represent specific different scenarios.

4.2 Comparison of Pyhton Simulation

4.2.1 Plant Simulation Results

At the same time, the Plant Simulation baseline DES was fed with the machine parameters of Table 9 and tests were performed to compare the three main scenarios previously done in the Python simulation. The simulation ran the same, during one hour of process. The results are shown in Table 17.

Table 17. Plant Simulation Results

	Scenarios in Plant Simulation		
	Baseline	**Slow Operator**	**High Variation**
Items	**69**	**43**	**70**

4.2.2 Comparison

Once having obtained the results of both simulations of each of the different scenarios, a comparison was made between them in order to determine if Python simulation was functional.

Table 18. Items per simulation

	Scenarios		
	Baseline	**Slow Operator**	**High Variation**
Items Python	**66**	**44**	**66**
Items Plant Sim	**69**	**43**	**67**

As seen in Table 18. the results obtained from Python DES were 66 items produced in first scenario, 44 in second and 65 in third. Moreover, the simulation in Plant obtained 66 items in first scenario, 43 in the second and 70 in the third.

The charts for each of the comparison are presented below. In the three graphs, the X axis represents the number of items produced and the Y axis represents the time the item was produced. The Python simulation is represented by the orange line and the Plant Simulation simulation by the blue line. The results of both baseline simulations are illustrated in Figure 36. A good similarity is observed in the data. Each point represents an item. The peaks generated by both lines represent an item with a time greater than 65 seconds. These items with high processing times occurred because prior to their creation, the operator performed the operation of filling the storage chamber with new material or with material left over from the Recovery Chamber. The difference of item production was of 3 items.

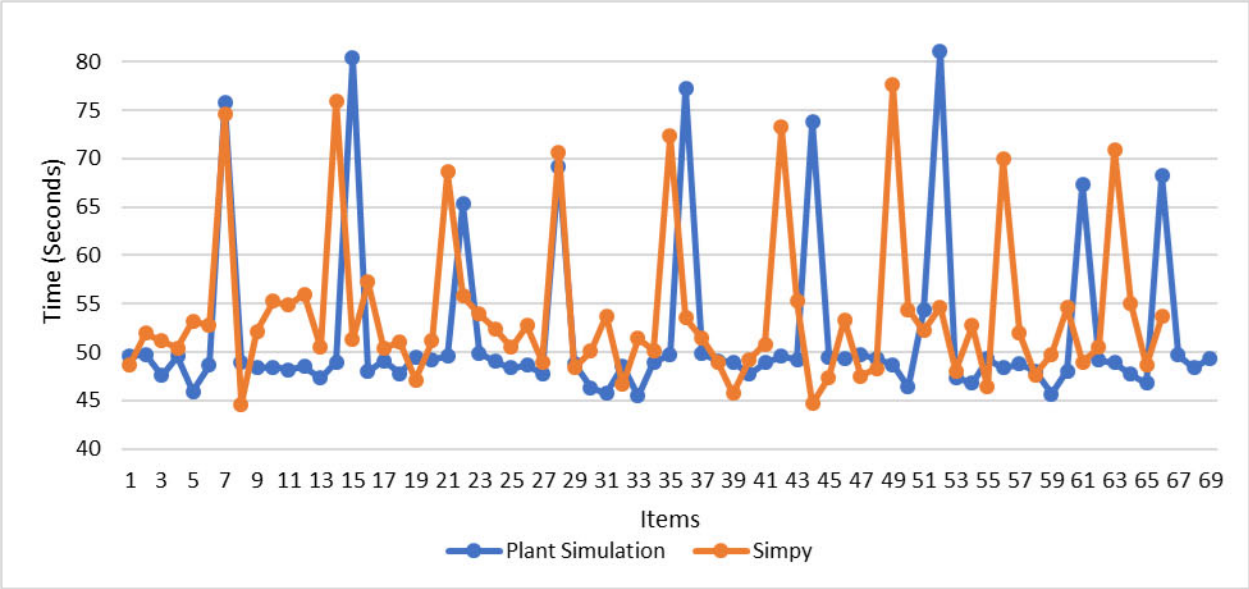

Figure 36. Baseline Comparison

The results and comparison of the two simulations with high variation are shown in Figure 37. In this comparison, the Plant Simulation model line achieves a more stable behavior than Python's. Even so, both representations managed to produce approximately the same number of items, with 66 in Python and 67 in Plant Simulation. The number of times the operator went to refill the main tank in both simulations can be seen on the graph as the high peaks.

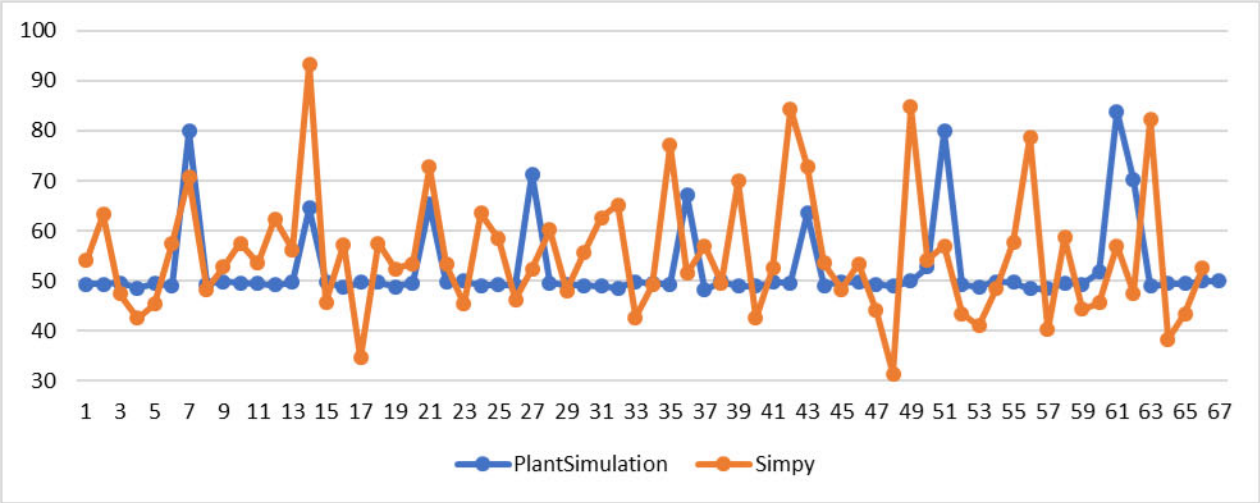

Figure 37. High Variation Comparison

The results and comparison of the two simulations with the slowest operator are presented in Figure 38. In this comparison, a significant difference is observed in the times that the operator had to fill the Storage Chamber with new material or with the recovered material. Whereas in Python he had to do it six times over an hour, in Plant Simulation he did it 3. On the other hand, the number of items produced was similar, with 43 in Plant Simulation and 44 in Python.

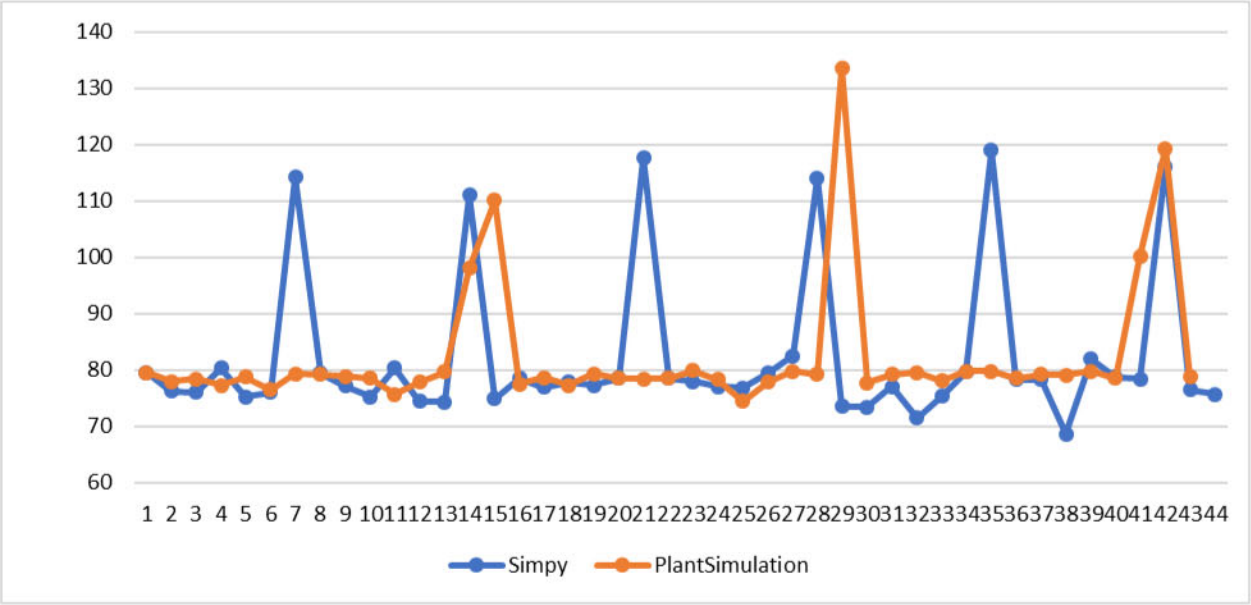

Figure 38. Slow Operator Comparison

4.3 Bayesian Prognosis

For the Bayesian analysis the baseline simulation was used. Data of times of each of the operations from the process were collected and analyzed with help of the code in Python. The different probabilities obtained are shown in Table 19.

Table 19. Bayesian Technique Results

Probabilities of items being delayed in the general process having found delays in:						
Storage Chamber	**Setup**	**Turn On Machine**	**Create Item**	**Turn Off Machine**	**Recovery Chamber**	**Remove Item**
42.8 %	40%	19.9%	46.6%	19.9%	23.91%	50%

4.4 Optimization Results

Communication between MatLab and Python was made to optimize the result of the DES. Three different simulations were executed in order to optimize the number of items produced: The process with 1, 3 and 5 machines with a storage chamber.

Optimization of process with 1 Machine.

The first execution corresponded to the scenario with 1 machine. Using the times stored in Table 9 and running the simulation for one hour the results are as follows.

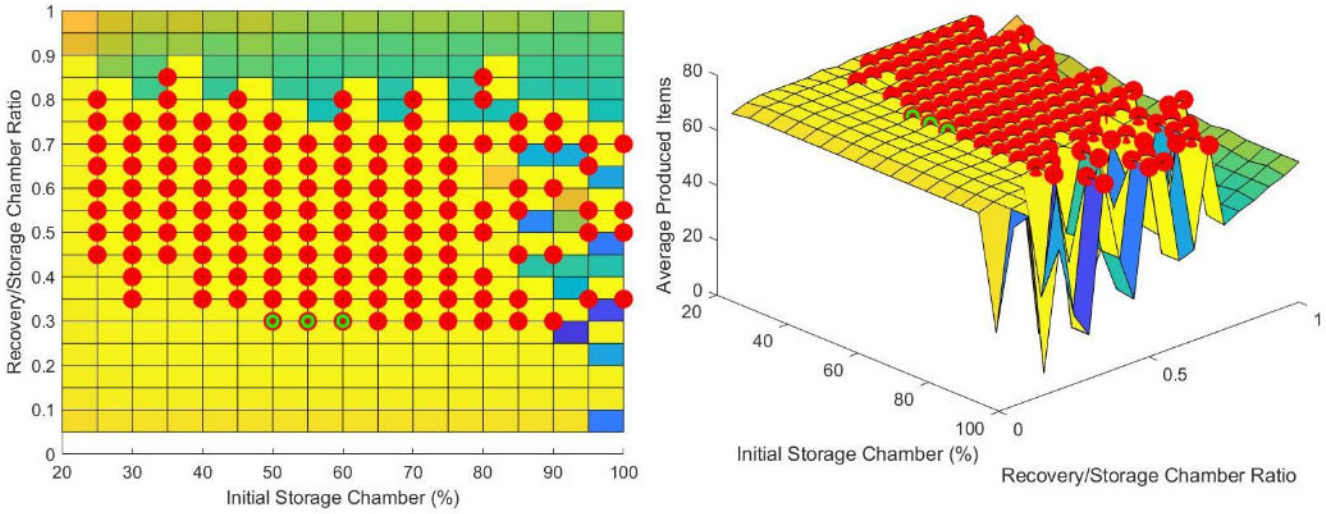

Figure 39. First Optimization Results

Figure 39 shows the representation made in MatLab, where the resulting values after carrying out the simulation are plotted. Each of the red points in the graph are the maximum values found within the combination matrix resulting from the simulation with each of the values within the range 20-100 of the initial Storage Chamber (X axis) and 0.05-1 of Recovery / Storage Chamber ratio (Y axis).

Once the maximums were located, they were optimized in order to find the best combination of ISC and RSCratio to maximize the number of items produced.

As seen in both graphs, the green dots represent the optimal values found. The qty. of items is represented on the Z axis. The optimal results are illustrated in Table 20.

Table 20. Optimization 1 Results

Optimal Values for 1 machine process		
Initial Storage Chamber	Recovery/StorageChamber ratio	Items Produced
50	0.3	69
55	0.3	69
60	0.3	69

Optimization of process with 3 Machines.

The second execution was made to represent the scenario with 3 machines. The times of the process are shown in Table 9 and the results below in Figure 40.

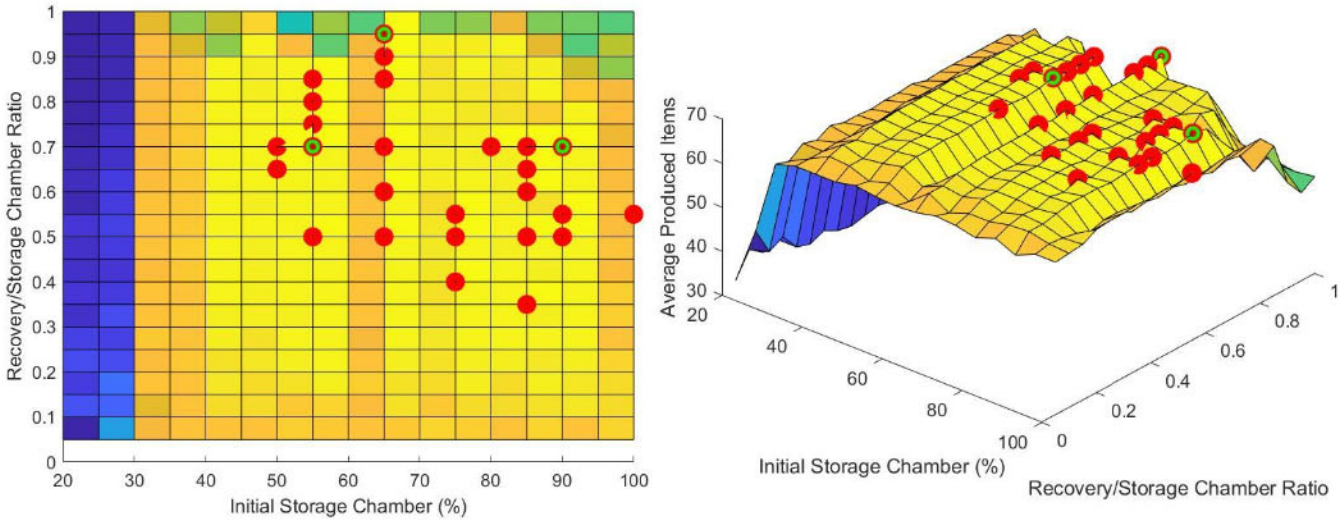

Figure 40. Second Optimization Results

According with the graph, MatLab script found 26 maximum points, identified with the red color. From those points, the optimization was made and then three points were identified as the optimal, as seen in Table 21.

Table 21. Optimization 2 Results

Optimal Values for 3 machine process		
Initial Storage Chamber	Recovery/StorageChamber ratio	Items Produced
55	0.7	68.333
90	0.7	68.333
65	0.95	68.333

Optimization of process with 5 Machines.

The last execution represented a process with 5 machines, producing items for one hour.

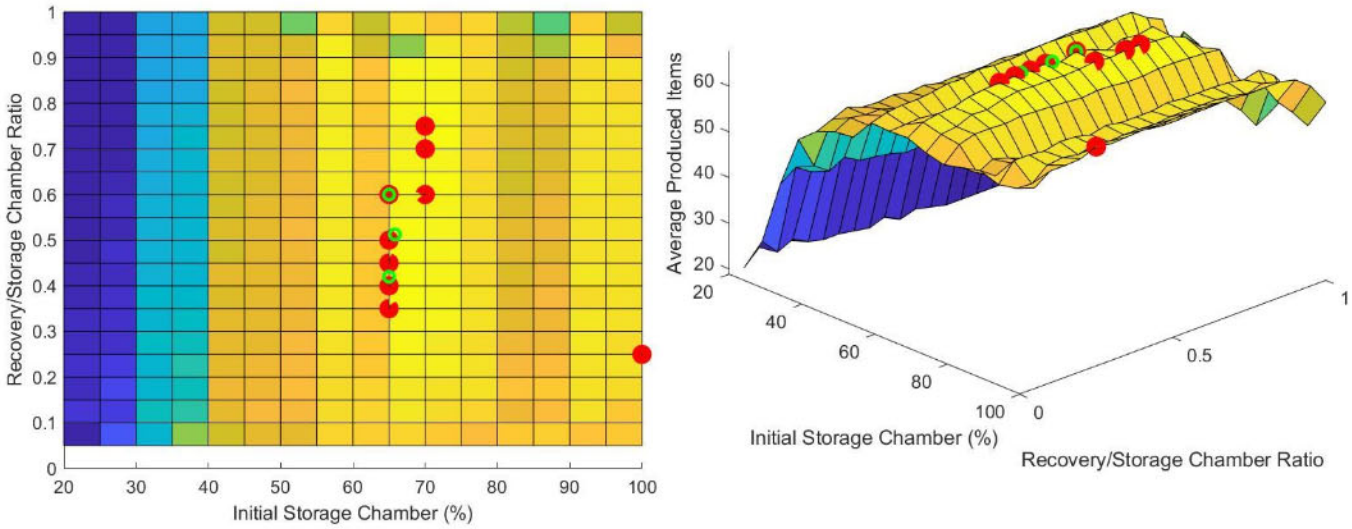

Figure 41. Third Optimization Results

It can be seen in Figure 41 that 9 maximum points were found, and after the optimization three points were the optimal to maximize the item production. These values are summarized in Table 22.

Table 22. Optimization 3 Results

Optimal Values for 5 machine process		
Initial Storage Chamber	Recovery/StorageChamber ratio	Items Produced
65	0.6	67.4
65.8125000000000	0.5125	67.6
65	0.42	67.2

CHAPTER 5. CONCLUSIONS AND RECOMMENDATIONS

As shown in the results chapter, in this thesis six simulated scenarios representing the process illustrated in Figure 4 were carried out. A comparison of the results of three simulated scenarios was performed in Python and Plant Simulation and the comparison of both simulations. Likewise, an analysis was performed to find different probabilities using a Bayesian technique and the optimal arrangement between Storage Chamber and Recovery Chamber capacities was also found. Finally, an optimization was carried out linking the work made in Python with MatLab in order to maximize the number of items produced.

According with the comparison results it was proven that Python model can simulate in a very precise way the process from the reality.

In Baseline comparison, the results of both simulations were very similar, obtaining several similar items produced. The results showed that Python simulation could produce 66 items and Plant Simulation 69 items. This was Python's first comparison using Plant Simulation.

In the comparison of the simulations with high variation, a difference was observed regarding the stability of production times of both models. The model of Plant Simulation was more stable than Python's but it was due to the variation inserted at the times, as well as the way of working of Plant Simulation. Both simulations obtained a number of similar produced items with 66 and 67 items.

In the third comparison, the slow operator, 43 and 44 items were produced in Python and Plant Simulation representation, respectively. The reduction in the results was due to the increase in the operating times of the operator, who when taking longer to carry out his tasks lengthens the production time per item.

The Bayesian technique was used in this work to represent the versatility of the code created in Python. Thanks to the simplicity of Python, it was possible to recreate the necessary modeling to obtain the different probabilities obtained within the naive bayes theorem. The results were as expected, finding the probabilities of producing delayed items having found delays in some of the operations of the process. If it was found that there was a delay in some operation with high execution time, it was expected that the prognosis of delays in the complete process would be high. The gotten results were as expected.

Through the representation of three more scenarios (One Storage Chamber/Multiple Machines, New process added and Baseline with Multiple machines), it was possible to reaffirm the conclusion that Simpy, within Python, is an effective tool for simulating discrete events.

With the changes made to the simulation base code, it was possible to observe different behaviors in terms of production, thus confirming different possible digital scenarios, and obtaining approximate results which can be used in the real world to modify the current process.

The optimization section was carried out in three different scenarios: the baseline process, with a single storage chamber and multiple machines (tested with 1, 3 and 5). Its purpose was to maximize the average quantity of items produced in one hour.

With the results of the first simulation, it was concluded that the optimal combination of the level at which the storage chamber should be filled and the recovery / storage chamber ratio to maximize the production of items with a machine and a storage chamber can be 50/0.3, 55/0.3 and 60/0.3, producing 69 items per hour. The second execution was tested with three machines obtaining an average of 68 items per hour with the combinations of 55/0.7, 90/0.7 and 65/0.95, and the third run was tested with five machines, obtaining an average result of 67 items per hour with the combinations of 65/0.6, 65.81/0.5125 and 65/0.42

In general, the optimizations carried out showed successful results applicable to the real process. Thanks to this, it is possible to simulate different scenarios and determine the optimal one in order to reduce production costs or unnecessary layout changes and increase production.

The optimizations section developed in this work is an approach to represent the capabilities of DES to interconnect with other software as well as the ease of use that MatLab represents, which is why a Python-MatLab connection was used. It is possible to move the optimization work towards an opensource software like Python, keeping the same optimization functionalities.

MatLab was selected to perform the optimizations due to its fast analysis, its easy graphical way of representing results and also because it has extensive documentation that is freely accessible to anyone who wants to learn to work in this software. Plant Simulation, on the other hand, is a specific software to represent discrete event simulations, which offers a wide variety of specific tools to represent and solve analyzes of this type, however, the limitations are greater due to the fact that equal to MatLab, Plant Simulation is commercial and expensive software, unlike there is no extensive

documentation for the general public, making it difficult to learn and use it also because it has its own programming language.

The established objectives were met. The discrete event simulation performed in Python (Simpy) was able to test different scenarios and be compared with a simulation commercial program (Plant Simulation). With this, it was observed that the simulation was able to work together with other tools such as Google App Scripts, MatLab and Unity. Also, the code generated to simulate the different scenarios was flexible enough to make different changes (number of machines, times) and even a bayesian analysis to find probabilities of how the process will behave. Likewise, optimizations were made in MatLab to maximize the number of items produced, regardless of the type of scenario being tested. Connections made for information management throughout the DES system were successful.

CHAPTER 6. FUTURE WORK

With the simulation model made in Python, different scenarios can be represented from what was tested in this work in order to improve some parameter of the real process, such as represent how machine downtime or failure affects the process.

An improvement to be made is to place sensors on the real machine to obtain process data and send them directly to the simulation, instead of using Google App Scripts to simulate the stored data times and sigmas.

Optimize the process in Python instead of using commercial software such as MatLab, to make the system completely open source.

Also, the DES can be improved to be more automatic and autonomous by implementing communication protocols of Industry 4.0 such as MtConnect and OPCUA, automating and allowing it to have an self-feedback of data.

Delve into how to relate Plant Simulation to the Internet of Things, to create new connections between that platform and those used in this work.

Experiment with other types of optimizations such as neural networks or genetic algorithms.

Create a generic simulation environment where obtaining data from any process allows modifying the parameters and operation of the simulation to adapt to any desired process.

José Abraham Valdivia Puga, a master colleague, developed an application for the iOS and Android platforms in parallel to my work, in which visual data regarding the simulation of the process and its results are displayed. The optimization data performed in MatLab could be represented visually within the application, thus reflecting how the process would look with different changes such as more or less machines, etc.

Appendix A

Abbreviations and acronyms

Table A.1 Abbreviations

Abbreviation	Description
DES	Discrete Event Simulation
SC	Storage Chamber
RC	Recovery Chamber
IoT	Internet of Things
RSCratio	Recovery/Storage Chamber ratio

REFERENCES

[1] Y. A. Sukhodolov, “The notion, essence, and peculiarities of industry 4.0 as a sphere of industry,” in *Studies in Systems, Decision and Control*, vol. 169, Springer International Publishing, 2019, pp. 3–10.

[2] A. Schumacher, S. Erol, and W. Sihn, “A Maturity Model for Assessing Industry 4.0 Readiness and Maturity of Manufacturing Enterprises,” in *Procedia CIRP*, 2016.

[3] N. Jazdi, “Cyber physical systems in the context of Industry 4.0,” in *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*, 2014.

[4] F. Tao, Q. Qi, L. Wang, and A. Y. C. Nee, “Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison,” *Engineering*, vol. 5, no. 4, pp. 653–661, 2019.

[5] A. Gatsou, “Discrete event simulation for manufacturing system analysis: An industrial case study.” [Online]. Available: https://www.researchgate.net/publication/237045327_Discrete_event_simulation_for_manufacturing_system_analysis_An_industrial_case_study. [Accessed: 28-Oct-2019].

[6] Z. Tuma, J. Tuma, R. Knoflíček, P. Blecha, and F. Bradác, “The process simulation using by virtual reality,” in *Procedia Engineering*, 2014.

[7] M. Kliment, R. Popovič, and J. Janek, “Analysis of the production process in the selected company and proposal a possible model optimization through PLM software module tecnomatix Plant Simulation,” in *Procedia Engineering*, 2014.

[8] C. A. Barrera-Diaz, J. Oscarsson, S. Lidberg, and T. Sellgren, “Discrete Event Simulation Output Data-Handling System in an Automotive Manufacturing Plant,” in *Procedia Manufacturing*, 2018.

[9] P. D. Urbina, R. F. García, J. A. Valdivia, A. Ortigoza, and H. Ahuett-Garza, “TOWARDS A DIGITAL TWIN BY MERGING DISCRETE EVENT SIMULATIONS AND THE INTERNET OF THINGS,” no. February 2020, pp. 1–4.

[10] E. G. Popkova, A. V. Bogoviz, U. A. Pozdnyakova, and N. V. Przhedetskaya, “Specifics of economic growth of developing countries,” in *Studies in Systems, Decision and Control*, vol. 135, Springer International Publishing, 2018, pp. 139–146.

[11] S. Ramaswamy and R. Tripathi, “Internet of Things (IoT): A Literature Review,” *J. Comput. Commun.*, vol. 3, pp. 164–173, 2015.

[12] J. Wan *et al.*, “Software-Defined Industrial Internet of Things in the Context of Industry 4.0,” 2016.

[13] M. S. Reis and G. Gins, “Industrial process monitoring in the big data/industry 4.0 era: From detection, to diagnosis, to prognosis,” *Processes*, vol. 5, no. 3, Sep. 2017.

[14] L. Monostori *et al.*, “Cyber-physical systems in manufacturing,” *CIRP Ann.*, 2016.

[15] S. A. S. Alkadhim, “Solar Cell Parameter Extraction From Data using MATLAB and Simulink,” *SSRN Electron. J.*, no. December, 2019.

[16] Y. Cai, B. Starly, P. Cohen, and Y. S. Lee, “Sensor Data and Information Fusion to Construct Digital-twins Virtual Machine Tools for Cyber-physical Manufacturing,” *Procedia Manuf.*, 2017.

[17] E. Negri, L. Fumagalli, and M. Macchi, “A Review of the Roles of Digital Twin in CPS-based Production Systems,” *Procedia Manuf.*, 2017.

[18] M. P. Roessler, M. Wolff, and E. Abele, “Design and simulation based assessment of lean material flows considering imprecision,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2013.

[19] J. Lee, B. Bagheri, and H. A. Kao, “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems,” *Manuf. Lett.*, 2015.

[20] E. Flores-García, J. Bruch, M. Wiktorsson, and M. Jackson, “CHALLENGES OF DISCRETE EVENT SIMULATION IN THE EARLY STAGES OF PRODUCTION SYSTEM DESIGN,” 2019.

[21] J. W. Fowler and O. Rose, “Grand Challenges in Modeling and Simulation of Complex Manufacturing Systems,” *Simulation*, vol. 80, no. 9, pp. 469–476, Sep. 2004.

[22] G. Dagkakis, I. Papagiannopoulos, and C. Heavey, “ManPy: an open-source software tool for building discrete event simulation models of manufacturing systems,” *Softw. - Pract. Exp.*, vol. 46, no. 7, pp. 955–981, Jul. 2016.

[23] N. Prajapat, T. Waller, J. Young, and A. Tiwari, “Layout Optimization of a Repair Facility Using Discrete Event Simulation,” in *Procedia CIRP*, 2016.

[24] P. Trebuňa, M. Kliment, M. Edl, and M. Petrik, “Creation of simulation model of expansion of production in manufacturing companies,” in *Procedia Engineering*, 2014.

[25] R. R. Neeraj, R. P. Nithin, P. Niranjhan, A. Sumesh, and M. Thenarasu, “Modelling and simulation of discrete manufacturing industry,” in *Materials Today: Proceedings*, 2018.

[26] R. Söderberg, K. Wärmefjord, J. S. Carlson, and L. Lindkvist, “Toward a Digital Twin for real-time geometry assurance in individualized production,” *CIRP Ann. - Manuf. Technol.*, 2017.

[27] M. Schluse, M. Priggemeyer, L. Atorf, and J. Rossmann, “Experimentable Digital Twins-Streamlining Simulation-Based Systems Engineering for Industry 4.0,” *IEEE Trans. Ind. Informatics*, vol. 14, no. 4, pp. 1722–1731, 2018.

[28] H. Zhang, Q. Liu, X. Chen, D. Zhang, and J. Leng, “A Digital Twin-Based Approach for Designing and Multi-Objective Optimization of Hollow Glass Production Line,” *IEEE Access*, vol. 5, pp. 26901–26911, 2017.

[29] V. Havard, B. Jeanne, M. Lacomblez, and D. Baudry, “Digital twin and virtual reality: a co-simulation environment for design and assessment of industrial workstations,” *Prod. Manuf. Res.*, vol. 7, no. 1, pp. 472–489, 2019.

[30] Z. Zhou, Y. Feng, G. Rong, and F. Zhu, “Virtual factory integrated manufacturing system for process simulation and monitoring,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2011.

[31] A. C. B. Garcia, C. Bentes, R. H. C. de Melo, B. Zadrozny, and T. J. P. Penna, “Sensor data analysis for equipment monitoring,” *Knowl. Inf. Syst.*, vol. 28, no. 2, pp. 333–364, Aug. 2011.

[32] C. Brecher *et al.*, “Integration of software tools with heterogeneous data structures in production plant lifecycles,” in *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 2010.

[33] M. Kikolski, “Study of Production Scenarios with the Use of Simulation Models,” in *Procedia Engineering*, 2017.

[34] P. Trebuňa, R. Popovič, and S. Kłos, “Methodology of the creation of human and robot operation in the tecnomatix process simulate,” in *Procedia Engineering*, 2014, vol. 96, pp. 483–488.

TOWARDS A DIGITAL TWIN BY MERGING DISCRETE EVENT SIMULATIONS AND THE INTERNET OF THINGS

Ruben Febronio Garcia Martinez [1], Jose Abraham Valdivia Puga [1], Axel Alejandro Gómez Ortigoza [1], Pedro Orta-Castañon [1], Horacio Ahuett-Garza [1], Pedro Daniel Urbina Coronado [1*]

(1) : Escuela de Ingeniería y Ciencias, Tecnologico de Monterrey,
Av. Eugenio Garza Sada 2501 Sur 64849, Monterrey, Nuevo Leon, Mexico.
Email : a00826791@itesm.mx , a00826790@itesm.mx , gomezortigoza@tec.mx, *urbinacoronado@tec.mx
Corresponding author

Short Abstract: This work proposes the integration of discrete event simulation (DES) in the environment of the industrial internet of things. A simulated process is defined for which its parameters are stored in a remote server. A local computer with an internet connection runs a DES which performs a request that updates the input parameters of the DES simulation to the latest data. With that, results of the cycle time, number of parts a day and other indicators are obtained. Future work includes optimization and implementation in real factory environment.

Key words: Discrete event simulation, Industry 4.0, IoT, digital twin.

1- Introduction

The fourth industrial revolution is linked to the concept of the Digital Twin (DT). The DT is a concept under discussion and for which there are several conceptions **[KK2]**. A simple definition of the DT "refers to a virtual representation of the real physical system that mirrors its state and behaviour" **[KJ1]**. The virtual representation requires models (that can be simulated) of the data, functionality and communication interfaces **[SA1], [SR1]**. According to Kunath and Winkler, the DT constitutes the Cyber part of the Cyber-Physical System (CPS) and in order to obtain the real time virtual representation of the physical part, the latter must be digitalized, or in other words, produce data through sensors and communication systems **[KW1]**.

Discrete Event Simulation (DES) is widely used in the manufacturing sector to research energy consumption **[RE1]**, reducing bottlenecks **[IY1]** and assembly line optimization **[KJ1]**. DES has been around since the 50's however, with the application of recent technologies, new opportunities to generate value have been found, for example, the integration or virtual reality and DES **[TH1]**. The integration of automated data generation to serve as inputs of the DES was studied by Ingemansson et al. **[IY1]**. An evolution of this idea is shown in a research about an Internet of Things (IoT) platform that monitors an assembly line and feeds information to the DES **[KJ1]**. Both of these works rely on the gathering of information for several days and then based on this information run the DES for analysis and optimizations.

The contribution of this work is to present a DES that has the capability of continually update its results with real time data stored remotely. The data can be generated by machines equipped with sensors in an Industry 4.0 or smart factory environment as expressed in the diagram of Figure 1. In the Figure, the DES is integrated into a CPS to contribute to generate the real time DT of physical objects in the plant. This work provides a case study of a theoretical machine simulated in a remote server. The DES simulation makes an internet request at a fixed time interval to query the last parameters and update the results. This work is organized as follows. Section 2 shows the setup of the DES and the simulated machine. Section 3 offers the results of the full implementation and Section 4 provide the conclusions of this study and the future direction of research.

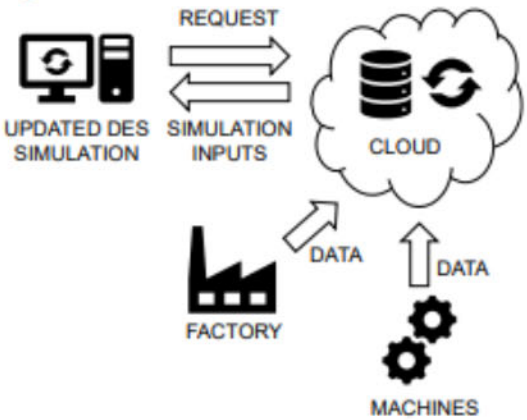

Figure 1: Concept of the DES simulation updated by real time data from smart factory.