MASTER IN BUSINESS ANALYTICS AND DATA SCIENCE

DEPARTMENT OF BUSINESS ADMINISTRATION

MASTER OF SCIENCE THESIS

SIMULATION IN PRODUCTION PROCESSES: THE CASE OF PIACENZA TEXTILE INDUSTRY

CHARALAMPIDOU MARIA RAFAELA

AUGUST, 2022

ABSTRACT

CHARALAMPIDOU MARIA RAFAELA

University of Macedonia

Master of Science Thesis, 73 pages

August 2022

Master's Degree in Business Analytics and Data Science

Examiner: Kaparis Konstantinos, Georgiou Andreas

Keywords: Discrete Event Simulation, Simulation Model, SimPy, Python

Technological development has generated more challenges in the field of manufacturing industries, leading to the need of advanced production processes. FACTLOG project initiated the idea of Cognitive Digital Twins in order for these processes to be optimized. Vital component for the construction of a DT, is the Simulation model.

This paper focuses on the case study of a Textile Industry, Piacenza, and the creation of a simulation model concerning a specific stage of its production process, weaving. The simulation is performed on Anaconda's Spyder environment, by the use of Python programming language.

The goal is to create a simulation model that takes under consideration components that can affect the levels of the production cost, and finally to calculate this cost for observation purposes and inspect its behavior in alternative simulation conditions.

PREFACE

The completion of this thesis has been an exciting yet challenging adventure.

I would like to thank my professors Konstantinos Kaparis and Andreas Georgiou for their guidance and tolerance during this long journey.

I would also like to express my gratitude to my parents and family for their love and support and lastly, I would like to specially thank my friend Nikos Efstratiou for getting me through tough days during this process.

CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background

The manufacturing sector is crucial to any economy due to its "domino effect" on the majority of other industries, encouraging employment and driving economic expansion.This industry is essential to the quest of sustainable economic growth in any economy.

This industry benefits from the advantages of competition, as the majority of enterprises are always innovating in order to create complex, yet inexpensive, new manufacturing techniques.As soon as one manufacturer discovers a method of producing things that is not only more efficient, but also more cost-effective, one can safely assume that another will follow suit.It is an industry that has been repeatedly reinvented by fresh ideas and new beginnings that alter the way things are conducted from the ground up (David, 2018).

The First Industrial Revolution saw the introduction of steam engines; the Second Industrial Revolution was characterized by electrical energy and mass manufacturing; and the Third Industrial Revolution, the most recent, saw the introduction of IT environments and other forms of digitalization (David, 2018).With the advent of the Fourth Industrial Revolution, also addressed as Industry 4.0, the manufacturing industry has undergone a comprehensive transition from conventional automated systems to those driven by the Internet of Things (IoT) and cloud computing involving cyber physical systems. This evolution has broadened manufacturing horizons in terms of cost reduction and production optimization and risen new challenges (David, 2018).

In the spotlight of the new era, one will find the Digital Twins; entities that can portray real life assets, simulate their procedures and can –among others- provide useful insights combining historical and real time data. In the core of a Digital Twin, there is a Simulation model. This model will represent all the co-existing components of the system, their functionality and in between relationships and dynamics, as well as simulate events that can occur during physical assets' procedures. Regarding manufacturing industries, a simulation model's results can lead to significant insights concerning production's cost reduction and effective resource distribution. This paper will focus on the use of simulation model for the observation and calculation of production cost.

1.2 Objective

The purpose of this master thesis is to develop a simulation model of the production process for a manufacturing industry specializing in the textile sector which can calculate losses that affect manufacturing cost. The simulation will be conducted on Anaconda's Spyder Environment, based on Python's SimPy module.

1.3 Thesis Outline

This paragraph refers to a comprehensive context description of the master thesis. Chapter 2 introduces the idea of the Digital Twin and as its core component, Simulation. Chapter 3 dives into a textile industry's production process simulation case, while providing information regarding the module that will be used for the implementation of the model. In this Chapter the construction of the model is described, as well as computational results. Finally, Chapter 4 provides a thesis summary and possible future research is proposed.

The following infographic provides an overview of the thesis structure and essential key points of each chapter.

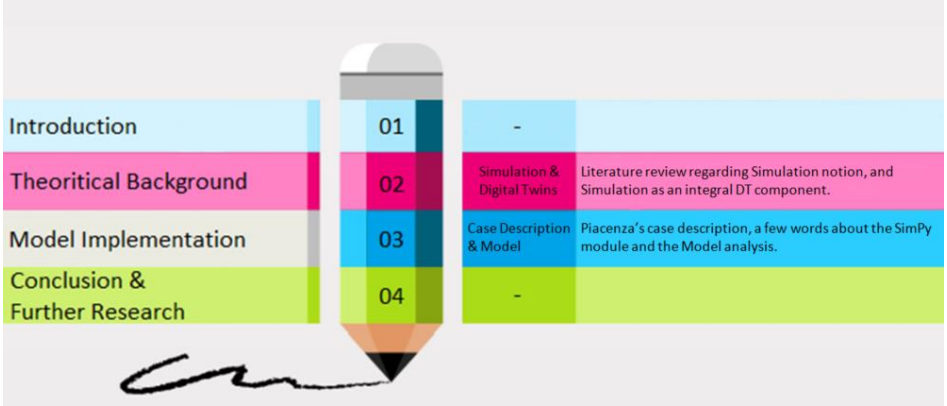


Figure 1: Structure Infographic

CHAPTER 2: LITERATURE REVIEW

2.1 Digital Twins

2.1.1 Introduction to Digital Twins

Ever since the evolution of engineering and manufacturing businesses, new needs have occurred for the optimization of their procedures and their products (David, 2018). These needs led to the development of new tools which could help to gain further insight as far as optimal decision making, useful predictions and reduction of risky activity were concerned. All these conditions resulted to the birth of the Digital Twins (DTs) idea.

NASA first introduced the idea of the "twin" during the Apollo program, where a twin of the spaceship was built, to simulate the conditions that the real space vehicle would probably face in the mission. Although the twin didn't include the digital factor at this point, it gained a fundamental role in testing rescue strategies on the ground and preparing the crew for possible obstacles. Nowadays, NASA adopts DTs as the key foundation in order to build next generation air force vehicles (Glaessgen & Stargel, 2012).

Digital Twins gradually gained more and more reputation and in 2017 were ranked in the top ten of the strategic technology trends (Erikstad, 2017). Yet, what is a Digital Twin, how does it work and which are the application areas?

2.1.2 Concept and Definition

A Digital Twin is described as a digital model of a real system, which is able to simulate the performance, function and behavior of the latter (Erikstad, 2017). Datta defines a DT as a *digital proxy*(Datta, 2017), that mimics the data flow, decision making and processes of a real time asset. These information about the real system are stored in a *software avatar*, and by addressing the relationships between the mentioned components, the digital twin is created.

According to Erikstad, the Digital Twin definition includes five core characteristics (Erikstad, 2017), the identity, the representation, the state, the behavior and the context of a digital twin. Identity refers to the imitation of the system by the DT. In a utopian scenario, the digital twin would be implemented through 1-to-1 cardinality between the system and the twin.

Nevertheless, in a realistic case, 1-to-N relationships are more likely to occur making the Digital Twin a difficult and challenging construction.

Representation is defined by the portrayal of the vital and physical operation of the system in a digital environment. CAD models can provide such format, yet such models and DTs differ in a matter of *state.* In State, the Digital Twin is divided by the classic CAD and CAE models, since the DT can present the real life system in real time, or at least give a quite accurate representation of it in a time point close to reality.

In a matter of Behavior, the DT reflects the effect of both internal and external factors in the real system, but in digital conditions. For instance, the alteration that an addition or a subtraction of an employee working in a retail sales center could cause to the center's availability to provide instant services to its customers, can be represented and investigated in the retail sales center's DT. In relation to Behavior, as far as Context is concerned, the digital twin stimulates the conditions where the real system exists, increasing the realistic essence that is provided through the DT.

As mentioned before, NASA started to build the foundation of the DTs when landing the Apollo project. To them, the Digital Twin was the virtual reflection of the real life asset, combining physics and scales, and exploiting both dynamic and historical data from the asset's life cycle (Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015). This understanding leads to further interpretations as multiple dimensions can be added. Rios et al, describe the digital twin of a product as the virtual image of its life cycle, from the birth of the idea and the implementation, to its servicing and impact. It uses the product's historical and current data to forecast future potential and develop further innovative services, related to the product (Ríos et al., 2015). Weber et al, approach a different dimension, the digital twin of a production process. To them, the DT of a production asset includes and represents all the modes and activities of the physical counterpart while being able to communicate and cooperate with other Digital Twins in order to accomplish an integrated intelligence system able to self-function (Kassner et al., 2017). In a factory dimension, Brenner and Hummel describe the digital twin as the digital representation of a physical factory, a machine, an employee and any other component, which is genuinely assembled and independently developed, automatically revised and accessible in real time (Brenner & Hummel, 2017).

2.1.3 Need for a Digital Twin

Yet, which conditions led to the creation of the Digital Twin notion and why it became extremely vital in modern technologies?

The main obstacle that all industries are trying to surpass is the risk factor. A possible alteration in their functionality would cause a chain of events that could have a negative impact on them. Industries have gradually managed to exceed the risk factor, at some point, through prediction. Nevertheless, a prediction is defined by a standard error, and the industries were in need for something more accurate. Thus, the idea of a digital model, representing the real asset, was born. The goal was to build a model that the industries would use to run certain scenarios, observe the results and act respectively.

Erikstad addresses the benefits in choosing the DT as the way to investigate a real life asset, in contradiction to the direct evaluation (Erikstad, 2017). He emphasizes DT's ability to monitor and examine the actual asset in a digital level, while saving resources and time that an inspection of the physical asset would require. Digital Twins enable accurate data aggregation and this is incredibly valuable, especially when the real life system is difficult to access, such as satellites in space or offshore installations.

The progenitor of the Digital Twin was the CAD model (Computer-aided design). The software was build to assist industries in designing processes and simulate several procedures. However, despite the fact that CAD models could answer in questions like "where", "when" and "how", through CAE (computer-aided engineering) analysis, they were lacking in operating as a PLM (product lifecycle management) environment (Fryer, 2019). The data flew into a CAD model were of a specific nature and they were also producing data of a specific nature. Nevertheless, Digital Twins exceeded CAD models in this area, as they draw data and information not only about the functionality of the process that they are representing, but also for other aspects such as cost of materials for instance. Thus, they are providing the user with the possibility of having a well-rounded idea not only of the functionality of the product and how it behaves, yet also the way it operates and how other factors are also influenced by that.

Digital Twins' abilities continue to unravel. Unlike more traditional methods, DTs are able to extract essential information by taking under consideration data deriving from both the

virtual and the physical environment. The comparison and analysis of these diverse data lead to the provision of more valuable insights and henceforth, to a more effective decision-making. Moreover, the Digital Twin can present various states of the physical asset in real time, giving the designers the ability of direct connection and interaction at any point of each procedure (Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015). Yet, the most important advantage that Digital Twins framework provides is the ability to predict future events by analyzing historical data and using virtual simulation. For all the abovementioned reasons, Digital Twins have granted more popularity than ever and their inclusion in data analyzing processes is almost mandatory. Besides, as CTO of Rockwell Automation Mike Loughran states, "It's bringing down the barriers".

2.1.4 State of the art

Digital Twins framework can be applied in multiple fields. Their applications vary from construction, healthcare and agriculture, to automobile, aerospace and smart cities etc(Adrien et al., 2020). Twenty first century has brought extreme development in digitalization, yet also in the complexity of the systems. Digital Twins seem to keep up with these challenges and they are considered to be the next generation of industrial simulation (Boschert & Rosen, 2016). As stated by Malakuti et al, the DT can describe and operate in any step of a process's life cycle, such as designing, operation and maintenance, by collecting the necessary data and information for simulation and predictive model (Malakuti et al., 2020). For instance, Tao et al. presented the advantages of utilizing a Digital Twin for a product's designing phase purposes (Tao, Cheng, et al., 2018), since the DT is also able to gather data and information regarding exogenous components such as customers' satisfaction and apply the insights in the designing process. Hence, the Digital Twin evolves into an accurate, virtual portrayal of a real-life asset. This enables the connection and the communication between the transmitter (e.g. designers) and the receiver (e.g. customers), especially now that the desire for visualization has increased. However, even with all the appropriate configurations and parameters, set and defined, each process needs to be tested via simulation. Yet, with no use of a Digital Twin, there is no connection and access to the real-time and environmental-affected data (Adrien et al., 2020). This is why using a DT to simulate a process and review its outcomes is important; the creation of real-life prototype for testing purposes is avoided, while the prediction capability is increased, since the Digital Twin is able to process historical data, mine and analyze hidden patterns and

provide robust solutions based on previous events and obtained information. Being able to predict and immediately respond in any occurred conditions, leads to a more efficient process evaluation, to a rapid correction in potential failures and to the avoidance of time and resources waste.

As mentioned above, Digital Twins has been characterized as the next generation of industrial simulation. This is because in modern manufacturing, the observation and data analysis during each operation is extremely vital, since it establishes a long-term sustainability and maintenance of equipment and resources. In terms of smart manufacturing, Wang et al. introduced a Digital Twin model for rotating machinery fault diagnosis (Wang et al., 2019). In order to distinguish unexpected deviations, the Digital Twin enables the consolidation and analysis of the physical knowledge and measurements using simulation models and their insights, instead of only relying on the sensor produced data (Adrien et al., 2020). This way, the DT simulation of mechanisms with expected failure, can lead to a critical event analysis for the determination of failure's principal cause and therefore, to the suitable solution. Based on these abilities, Wang et al. developed a prototype of a rotor system in order to establish the capabilities and effectiveness of Digital Twins framework on uneven quantification and detection for fault diagnosis (Adrien et al., 2020). The investigation's results have shown that the rotor DT model can effectively proceed with fault diagnosis and predictions regarding failure degradation.

Apart from the benefits in terms of simulation, Digital Twins also provide optimization contingencies through the production process. The DT holds information about every component of the production process, including available resources, assets, etc. This means that it is able to demonstrate in real time the current state of assets, outputs and procedures. As a result, any aftereffect a decision could have on a process, it's reflected through the Digital Twin which enhances the optimization opportunities, with avoiding the actual execution of the process for later investigation (Tao, Zhang, et al., 2019). The DT can also detect the outcomes of additional measurements regarding failure and malfunction behaviors for comparison and prevention purposes.

2.1.5 Architecture

The architecture of a Digital Twin can consist of five components; the physical asset, the sensors, the data, the analytics, the digital twin itself and the actuators. The Physical Asset stands

for the real-life asset that exists in the actual world. Regarding the Sensors, they have become the life energy of the manufacturing industry and Internet of Things has become the base framework where they function (Bi et al., 2014). This type of sensor makes real-time information about a real-lifecomponent's availability possible, with the later being an essential enabler for digital twins.In order to make information about the equipment (statuses), products (statuses, quality), processes (torque, displacement, etc.), and the environment (ambient temperature, humidity, etc.) available for the twin, the manufacturing facility is outfitted with a vast number of sensors that are dispersed throughout the asset.(David, 2018) Third component of the DT's architecture is the data. Data are collected from the sensors and are subsequently combined with the data obtained by the Enterprise Resource Planning and Manufacturing Execution systems. Following the proposed structure, there are the Analytics. In the field of artificial intelligence, analytics consist of state of the art technologies like pattern recognition, unstructured data analysis, and multi-modal data analysis. These technologies are used to analyze data in order to gain insights,given the available information (David, 2018). Continuing with the Digital Twin, this component stands for the virtual equivalent of the actual asset and it is software-based while portraying the asset in real time. Last but not least, Actuators are the mechanisms enabled by the Digital Twin for the later to able to interfere upon the real-life asset.

Tao et al. introduced the idea of a five-dimensional Digital Twin and identified the dimensions as i. Physical Entity, ii. Virtual Entity, iii. Digital Twin Data, iv. Services and v. Connections (Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015).

i. Physical Entity

Users are able to interact with and use the actual asset, which is the physical entity.Real-time data collection is made possible by sensor technology and Internet of Things technology. Data can be derived from both endogenous (maintenance data, environmental data, production running data etc.) and exogenous (reviews in social media platforms, download records etc.) sources. It is possible to store and analyze all of the data in the cloud platform, enabling the ability of easier data access to users from any location in the world, as long as there is Internet access available.

ii. Virtual Entity

The virtual entity is a reflection of the physical entity, and it is comprised mostly of four different kinds of models: geometry models, physics models, behavior models, and rule models (Tao, Zhang, et al., 2018). The first two models, characterize the asset in terms of its geometric properties (such as shape, location, and assembly relation), and its physics performances (e.g. hardness, strength, and wear resistance). Regarding the behavior models, not only they perform analysis on how the asset behaves in vary conditions, but they also concentrate on how the users and the environment behave, as well as the interactions and connections that occur between all of these factors. Moreover, the assessment, optimization, and forecasting models that were developed by adhering to the laws of asset operation and maintenance, among other laws, are the primary components of the rule models (Tao, Sui, et al., 2019).The cloud environment enables the building, launching, and maintenance of the virtual product. This environment also makes it easier for users and designers to get access to the actual entity.In addition to this, virtual reality (VR) and augmented reality (AR) make it possible for users and designers to interact directly with a virtual product in either an entirely virtual environment or a cyber-physical environment that combines both, all while maintaining a high level of authenticity and receiving feedback in real time(Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015).

iii. Digital Twin Data

In the DT, it is possible to evaluate, combine, and visualize data that has been acquired from both physical and virtual domains (Tao, Sui, et al., 2019). According to that, at first, data analytics is required in order to transform data into information that is more solid and that designers may directly query in order to apply decision-making processes.Secondly, since the data related to the asset are gathered from a variety of sources (such as the physical entity, virtual models, the Internet, cloud platforms etc.), the process of data integration makes it possible to identify underlying patterns that would otherwise be impossible to uncover using only a single data source.Thirdly, the inclusionof data visualization technologies enables the data to be presented in a manner that is effective, accurate and user friendly.In conclusion, state of the artmethods of artificial intelligence can be applied to improve the DT's cognitive ability (such as reasoning, problem solving, and information representation), allowing for some suggestions to be automatically model-derived.

iv. Services

Functional service and Business service are the two primary categories of services that are found in the Digital Twin (Tao, Liu, et al., 2019). Functional Service encases various data, models and algorithms, into services to support the effective working of a Digital Twin. Those services refer tomodel assembly, simulation, verification, DT data analytics, integration, and miningetc.(Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015). On the other hand, Business Service is offered to suit user expectations in areas such as machine fault prediction, risk evaluation, employee training, and customer satisfactionetc.(Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015).Users can be introduced to the BS using software interfaces that have standardized inputs and outputs for the purpose of providing clear instructions and making use of the system straightforward.

v. Connections

The connections that exist between the prior four dimensions are denoted by the notations CN PD, CN VD, CN SD, CN PV, CN PS, and CN VS. These notations refer to the connections that exist between the physical entity and the DT data, the virtual entity and the DT data, the services and the DT data, the physical entity and the virtual entity, the physical entity and the service, and the virtual entity and the service, respectively(Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, 2015).

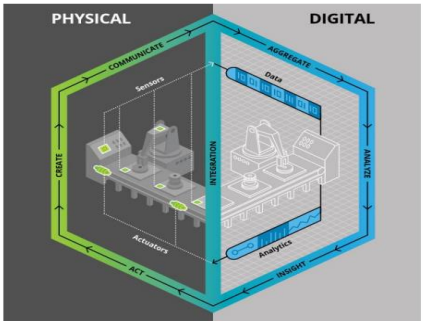


Figure 2: Architecture of Digital Twin(Parrot & Lane, 2017)

As it has been established, Data are the key factor related to the success of Digital Twins and any other AI related technology case. This is why the data should be defined by quality and reliability.It is essential to determine the type of data, quantity the quality of the data(Kuehn, 2018). The most important aspect of DTsis their real-time data feeds.Over the course of the past

few years, the volume and diversity of real-time data created all throughout the manufacturing value chain have rapidly increased.These data are originating not only from the assets themselves, but also from connected production equipment, fundamental manufacturing processes, enterprise information technology systems, and extrinsicparties such as customers or suppliers (Kuehn, 2018). In order to act as a bridge between the actual and the digital environment, Digital Twins need to receive data from their corresponding real-life systems in real time.The accuracy and dependability of these data, which have been obtained from a variety of sensors, is an essential concern.Anomaly detection of sensor data is thus required to be applied when Digital Twins are used in industrial applications.

2.1.6 Digital Twins Process

The implementation of a Digital Twin will be determined by the asset type, which will be connected to the requirements for precision, quality, availability, feasibility, and other similar factors, which will be traded off against cost and the readiness of the technology. The following components are required for a Digital Twin implementation to be considered complete at a minimum: 1) Capabilities at the edge for monitoring essential facets of the current state and behavior of the real asset. This often entails the use of sensors that are equipped with associated edge processing capabilities for the purpose of improving data quality in some way, be it by calibration, filtering, time synchronization, or some other method. 2) The basic runtime of the Digital Twin, which uses the input stream from the edge to produce a (near) real-time digital reflection of the asset's current state. As an essential component of a variety of business procedures, the application layer participates in the practice of subscribing to certain data streams provided by the Digital Twin. The data stream from the twin might be imported into data analytics and machine learning stacks for pattern identification and decision support. This could be particular end-user applications for observation and control, legacy systems for maintenance and asset management, or both(Erikstad, 2017).

2.1.7 Cognitive Twins

Regular Digital Twins are constructed as digital representations of physical assets. However, Cognitive Twins can be augmented with behaviors derived from algorithms, AI models, and KG models, rendering them actionable.An ontology model can abstract physical and digital entities, data sources, as well as algorithms and AI models(Rožanec et al., 2020).Consequently, each CT has its own ontology for its own contexts, and the ontology is

constructed based on a unified specification with more abstract concepts.Utilizing the Knowledge Graph and Artificial Intelligence provides cognitive skills, albeit their functions are distinct.

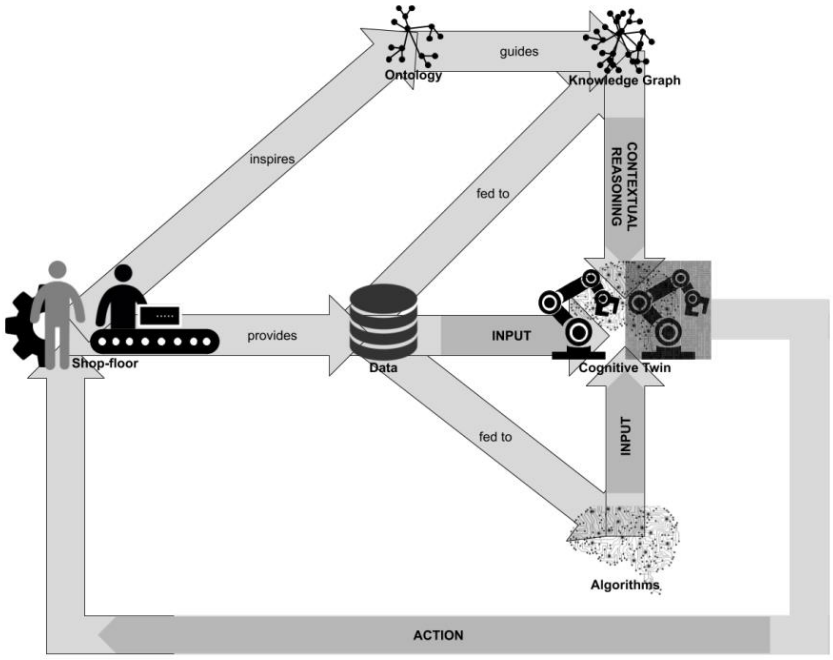


Figure 3: Cognitive twins supporting decision-making for manufacturing(Rožanec et al., 2020)

The cognitive capabilities can study the history behaviors of physical entities and the historical data of digital entities based on various use cases in order to make decisions for operational physical entities.

2.1.8 Impact and Value

The concept of the digital twin is quite powerful, and some of the common benefits of this concept are as follows:

1. Visibility: The Digital Twin enables visibility into the operations of a physical asset, a network of interconnected systems, or even an entire business.
2. Predictive: Digital Twin models can be utilized, using a variety of modeling methodologies, to make predictions regarding the future states of an asset.
3. Analysis of What-If Scenarios: With the help of well-designed interfaces, it is

possible to interact with a model, pose "what-if" scenarios to a virtual model, and mimic a variety of conditions that would be difficult or impossible to produce in real life.

4. Understand and explain behaviors: The Digital Twin model can be used as a communication tool to better understand and explain the behaviors of an individual asset, machine, or system. This can be accomplished by using the model to both comprehend and explain the behaviors.
5. Connect disparate systems such as backend business applications: Digital Twin models can be linked to backend business systems to achieve business results in the context of supply chain operations such as production, procurement, warehousing, transportation and logistics, and field service (General Electric Company, 2016), (Kuehn, 2018).

Digital Twins provide a sophisticated product, production, or logistic simulation model that can be used to answer specific questions. This model enables the monitoring and analysis of real-time data, which can then be used to improve design, controls, and tactics.The use of digital twins provides the opportunity to run the entire business or a portion of it in advance, as well as to explore and evaluate a number of potential solutions in a simulated setting before deciding which one to implement in the actual system.However, the concept of a digital twin calls for a robust link between the virtual world and the actual world, which must make use of real-time data.It is becoming increasingly vital to engage in continuous improvement in order to operate complex production and logistic operations in a manner that is both effective and efficient over the long term.It takes a powerful combination of digital and industrial strength, as well as a combination of in-depth physics knowledge, engineering design knowledge, advanced sensing and inspection technologies expertise, Artificial Intelligence and analytics experience, in order to design and realize digital twins(Kuehn, 2018).

Digital twins are becoming a vital aspect of the digital transition, and their implementation could impact any field.As a result, products were converted into services.This shift provides enterprises with a distinct chance to monetize their efforts beyond product sales. Digital Twin simultaneously generates data that leads to ongoing improvements in production processes and products. The transition from products to services is also likely to be beneficial to

the environment, since it extends product lifetimes, reduces the quantity of raw materials required to manufacture replacement products, and reduces the amount of waste that must be recycled.

Digital twins that provide new capabilities will encourage the development of service-related features by third parties, necessitating an open framework for the building of (soft) add-ons and a degree of standardization(Saracco, 2019).The latter could emerge from firms exposing their product ecosystems and devices that comply with specific platforms, such as smart-phones, where the platform is both the operating system (such as Android and iOS) and the market (for example, Google Play and iTunes)(Saracco, 2019).

2.1.9 Digital Twins Furthermore

The objective behind the creation of digital twins was to replicate actual entities with a degree of fidelity (precision) and synchrony that is suitable for certain applications.There is a good chance that digital twins will progress to a super entity that they constitute with their overlapping physical counterparts in a symbiotic relationship(Saracco, 2019). This would be similar to another outcome that is envisioned by the Symbiotic Autonomous System Initiative.The features of the super entity would have been inherited from both of the twins and their symbiotic relationship. For instance, the digital twin might function as an avatar that wanders throughout cyberspace and (partially) represents the real one.There is already an example of this, with UBSutilizing the digital twin of its chief economist Daniel Kaltin order to communicate with customers.

The idea of a super-organism can be significantly increased by creating digital twins out of bits, which can be replicated as many times as necessary.Applications that are used in conjunction with digital twins have the potential to expand the models' properties.Applications, for instance, are able to analyze the digital thread, which refers to the progression of the digital twin over time, and extract some significance from it.Imagine if there was a program that was aware of everything you had done over the course of the previous month and was able to converse with digital twins that represented the people you had been in contact with.It is possible that it will identify the initial symptoms of an infectious disease and advise you to visit a physician as soon as possible.It is even possible that it will supply the doctor with information on your friends and how they are responding to the treatments that have been prescribed for

them(Saracco, 2019). There is no upper bound on the potential capabilities of an extended digital twin.

A digital model, its shadow, and its thread are the three components that make up a digital twin.Each component presents its own unique set of technological hurdles.In addition, we shouldn't forget that in the years to come, digital twins will almost certainly increase their capabilities (by utilizing technologies such as AI) and eventually become a component of super-organisms, which will be the result of the symbiotic relationship between physical and digital twins.Both of these facets of that growth provide challenges.

Digital twins will continue to expand their presence throughout the coming decade.There is uncertainty over the adoption of personal digital twins and cognitive digital counterparts.Much will depend on the services that digital and cognitive personal twins provide.They create questions around confidentiality, ownership, and responsibility.A regulatory framework may be required, but it may be insufficient because a roaming digital twin may cross boundaries and play in cyberspace domains with different (or no) rules (Saracco, 2019). As this is a general issue that extends well beyond digital twins, it will require management at a higher level.

Digital twins exist in a variety of businesses.They are so effective and economically advantageous that their expansion to other regions seems inevitable.As they expand and become more prevalent over the coming decade, ownership and accountability issues will need to be addressed.In addition, human digitization will begin in the health-care industry (the first signs are already visible) and spread to other industries, including education, bringing privacy and the sense of self to the forefront(Saracco, 2019).As with any technology, security issues are paramount, and a regulatory framework is required.

2.2 Simulation

2.2.1 Introduction to Simulation

In order to proceed with the following research, it is vital to understand what a simulation is. In the Winter Simulation Conference in 1997, Anu interpreted simulation of a system as the "operation of a model of that system" (Anu Maria, 1997), which model can be redesigned and reformed. Experimenting with the system itself can be time consuming and expensive, leading to simulation in being the most proper procedure in evaluating the performance of the system under several conditions, both in short and long run. The simulation model will be tested on a variety of stimuli in order to prevent the real-life system from reaching unpleasant situations such as overuse of resources, failure to reach certain targets, elimination of future risks, etc. and finally to lead in the optimization of the system's performance. For instance, a simulation model can be used in order to inform the manager of a retail queue, on what will happen if the frequency of the costumers' arrivals changes. Answering "what if" questions, is one of simulation's biggest advantages.

2.2.2 Simulation Notion

As already mentioned, a simulation of a system relates to the model's operation of that system. The constitution of that model is based on several assumptions in the form of logical and/or mathematical relationships (Law & Kelton, 1994), about how the system operates. After the model is formed, it's used to gain knowledge on how the real-life systemwill respond on future stimuli. If the model is simple enough, the procedure of information extraction can just include mathematical methods. Yet, this is almost never the case, because real-life systems are defined by complexity and it's impossible to provide an analytic solution. To surpass this obstacle, the use of technical tools is important to collect the data, define the model's real attributes and finally provide a numerical solution.

Simulation appears to be the ultimate tool used in terms of operational research, while even since the late 19thcentury was ranked as one of the top three numerical techniques (Lane et al., 1993). As the years have gone by, simulation techniques have been tremendously developed and used in more recent technological initiatives such as Digital Twins that were discussed in the previous chapter. According to S. Erikstad, as a DT provides hindsight in the operational history of a system, as far as behavior based on previous patterns, and performance is concerned, simulation proves to be able to evaluate these procedures and furthermore to achieve predictions

related to the system's correspondence in both anticipated and unexpected future challenges and upcoming critical events (Erikstad, 2017). In correlation to future technology relevance, the idea of approaching a model with telemanipulation perspectives appeared (Willaert et al., 2012) aiming in achieving both stability and consistency in the model, while highlighting the ability to perform forecasting procedures using simulation in-the-loop (Cichon & Rossmann, 2017).

2.2.3 Adequacy

Since the mid 20thcentury, many authors are discussing the adequacy of simulation (Shannon, 1998). With this technique being the most widely accepted tool in operational research terms and as far as systems' analysis is concerned (Banks et al., 1999), the competence and efficiency of simulation needs to be examined. Where is simulation legitimate to be used and where is not?

Creating a simulation model for a system defined by complexity, the ability of experimenting and interacting with both external and internal functions of that very system, or a part of it, is now doable. The alterations caused by certain changes in the operational environment can be simulated, and the impact of these amendments on the model's behavior can be estimated. Furthermore, the observations of the effects that different simulating conditions cause to the model, can tremendously contribute in the whole system's evaluation and improvement. If any analytic solution methods where used, simulation can also provide verification of these methods and further reinforcement in terms of analysis and results.

Under evaluation is also the importance of the variables that enter the model. When in each simulation the variables' inputs are different, the influence and the effect of each variable subset is easily investigated and the variables' interaction within the inputs' set is observed. In addition to these observations on the impact of variables' alterations, simulation models can also provide proposals in terms of designing and policy changes (Banks et al., 1999) within the real-life system to prepare it for "what if" situations, before actual implementations take place.

As already mentioned, in order to manipulate complex systems and all the internal interactions that take place, simulation seems to be the most proper method to use. Internal interactions could be translated as a subsystem that specifically needs to be under observation and simulation, for instance, a machine. A machine is defined by certain characteristics and

capabilities. When these capabilities are altered through various scenarios in simulation, new information is being extracted relevant to the potential abilities of the subsystems-machines and new requirements are presented to optimize the performance and utilization of each subsystem.

Interfering in a system through a simulation model definitely reduces the cost, the time and the risk of doing it in the actual organism and provides a variety of information as far as future scenarios, predictions, optimized functionality and efficiency are concerned. Moreover, simulation also provides the ability of visualization on the model and through animation every distinct scenario can be also visibly approached.

However, despite the widespread use of simulation as an accepted tool for operational research, several scientists pointed out certain situations where simulation is not the most appropriate tool to use and provided ten principles where this technique could be avoided (Banks & Gibson, 1997). The first rule applies to the case where the solution of a problem can escalate through common sense. For instance, in a supermarket with automatic cash desks, where in an average rate the arrivals of costumers are 1000 per hour and the cash desks can serve in a mean rate of 100 per hour, then, in order to determine the minimum number of cash desks needed, the division 1000 / 100 = 10 provides the solution. The result indicates that the supermarket should provide at least 10 automatic cash desks for the costumers to be assisted properly.

In relation to the previous example, the second rule is applied. If the solution of a problem can be determined through logical and/or mathematical methods such as curves, simulation is not needed (Bhunia et al., 2019). Hillier and Lieberman developed such methods (curves) that could calculate the average waiting time of a costumer in the previous example (Hillier & Lieberman, 2002).

Third rule refers to the scenario that a direct experiment can be held, instead of the use of a simulation model, making it faster and less expensive to reach a solution. Such scenario could include for example a retail service that occupies one employee in order to assist the costumers. If the service adds another employee, the interaction of this addition with the costumers' waiting time variable, is easily detected without using simulation.

In their fourth rule, Banks and Gibson suggested that since simulation can become an expensive procedure (according to late 20thcentury's technology), if the costs of creating a

simulation model surpass the system's saving that the simulation provides, then it should be avoided. Thus, rules five and six that correspond to the availability of resources and time, suggest the very same thing.

Rule number seven refers to data. In order for a simulation to take place, data are needed. If there are no data, or their extraction is extremely difficult and not even estimates can be made, then simulation is not suggested. Hence, in rule number eight, if the lack of enough data leads to an inappropriate validation of the model, Banks and Gibson claim that simulation should be avoided. Additionally, in the same grounds of arbitrary expectations, might be more beneficial for simulation to be encountered likewise, according to the ninth rule.

Last but not least, rule number ten includes the complexity that a system could be defined by. Special focuses are made on the human factor since human behavior can become quite complex and unpredictable, and difficult to model.

In spite of some certain cases which are mentioned above, where simulation is not considered a suitable tool, it remains a technique that without any doubt brought operational research to a whole new era minimizing the costs, time-consume and risk needed to experiment on a real-life system, simplifying the analysis and the evaluation of the results, and optimizing the performance of the systems.

2.2.4 Advantages and Disadvantages

Simulation is a widespread and accepted technique, because it can simulatewith accuracy the external and internal behavior of a system both in designing or already existing and fully functioning stage. A simulation's results should coincide to the hypothetical output of the real-life system, if the simulation's scenario was applied on it. Moreover, the simulation technique includes all those cases which could be based on arguable hypotheses and solved by mathematical methods, ranking simulation as one of the most frequent used techniques in problem solving.

There should be a separation between simulation and optimization models. Optimization models are solved and present the optimal solution in a problem, while simulation models are "run" (Banks et al., 1999). In the case of simulation, each time a different set of inputs and model's attributes is provided and the running procedure begins. While the running procedure is

ongoing the model's behavior is observed. The alterations on the behavior that each given set of inputs and characteristics causes, create a set of scenarios. These scenarios fall under examination, in order to be evaluated and provide the best case scenario, which will be probably applied in the real-life system for implementation.

However, despite the beneficial results simulation can provide towards the system's performance, Pedgen(Pegden & Sturrock, 1990) suggests that the procedure has both advantages and disadvantages.

2.2.4.A Advantages

1. First of all, the greatest simulation's advantage, that is related to the very nature of this technique, is that without the real-life system being altered, procedures referring to operational research, such as decision making, new policies testing, inputs and outputs flows, organizational and functioning procedures etc., can be observed and be evaluated to be possibly applied in the future.

2. Secondly, since simulation executes digital made scenarios, the real resources of the system remain untouched. By that, any possible "extreme" scenario can be run without putting the real-life system at risk.

3. Thirdly, assumptions can be made related to how and why certain patterns seem to appear, and in the same time these hypotheses can be tested and evaluated. In terms of these phenomena being assessed, in the simulation, time can be both expanded or contracted to provide the ability for speeding up or slowing down the occurrence of the patterns for flexible observation and investigation.

4. Moreover, as mentioned before, the ability to insert each time different sets of inputs to create different scenarios, can lead to the observation of the interaction the groups of variables have with each other and also to a ranking of these variables based on their importance according to the simulation's results.

5. Furthermore, simulation also contributes on bottleneck analysis. This kind of analysis consists of a detailed process where information and data relevant to the manufacturing and operating flow of a certain procedure, that is being bottlenecked, are collected. Through this, simulation

can provide possible answers responding to questions such as why is this bottleneck happening or if it is possible to happen in the future. Information like that contribute in the understanding of certain procedures' functioning and in the improvement of them. For instance, bottleneck analysis could investigate the cause of the delaying of a particular ongoing process in the system.

6. Finally, simulation provides a clear image and an understanding of how the system really functions and not how individuals assume it does (Banks et al., 1999). Additionally, under the analysis of a simulation, "what if" questions are investigated which is really important, especially for systems that are still under development and in need for a strong base.

2.2.4.B Disadvantages

On the downside Pedgen(Pegden & Sturrock, 1990) claims that simulation is a procedure that needs certain training in order to be performed. The individuals need to be trained in model building and gain knowledge by gaining experience. Also, it's highly unlike for different individuals to create the same simulation model for a system. The models will share some basic foundation characteristics but they will be different.

Another argument states that sometimes simulation's outputs can be difficult to clarify. Considering the fact that the sets of inputs in the model can be random variables, the outputs might be defined by this randomness as well, leading in difficulties in the interpretation process, since it's not easily clarified whether an interpretation of an output is based on interrelationships within the system (Banks et al., 1999), or in haphazardness.

Simulation can also be a quite time consuming and expensive procedure. Yet, not providing the necessary resources to build an appropriate simulation model, could result in outputs, observations and assumptions that are not legit or adequate. The validity of the simulation model is extremely vital.

Finally, as it has been already mentioned, several time simulation is being used to provide solutions in problem which could have been solved with an analytical solution. In these situation, for instance when the problem includes procedures in waiting lines where queuing model could have been used (Banks et al., 1999), an analytical solution is desirable.

However, these disadvantages can be overly supplanted since experience and technology

development have shown otherwise. As Jerry Banks claims *"Simulation can be performed faster today than yesterday and it will be even faster tomorrow"*(Banks et al., 1999). Technology's evolution and especially simulation's development provides users with enhanced packages in the software, which packages support advanced output-analysis abilities in order not to be any room for misinterpretations. They incorporate models which demand only input data to run such as "simulators" and "templates" (Banks et al., 1999) and encompass hardware that allow accelerated running of the possible scenarios. Moreover, as far as closed form queuing models are concerned, experience has shown that such models couldn't encounter problems and systems defined by complexity and that analytical solutions could not become sufficient.

2.2.5 Application Fields

The cases where simulation is applicable, are unlimited and vary. Latest information about simulation's development both in practice and in theory, are announced in the Winter Simulation Conference (WSC) which takes place every year in the US. WSC is sustained under the aid of the National Institute of Standards and Technology (NIST) and six technical societies. These associations are: American Statistical Association (ASA), Association of Computing Machinery/Special Interest Group on Simulation (ACM/SIGSIM), Institute of Electrical and Electronics Engineers: Computer Society (IEEE/CS), Institute of Electrical and Electronics Engineers: Systems, Man and Cybernetics Society (IEEE/SMCS), Institute of Industrial Engineers (IIE), Institute of Operations Research and Management Sciences: College on Simulation (INFORMS/CS), and finally, the Society of Computer Simulation (SCS) (Banks et al., 1999). In these conferences, the areas of simulation's application have been explored and investigated. Banks et al summed them in eight categories (Banks et al., 1999):

1. Manufacturing Applications
2. Semiconductor Manufacturing
3. Construction Engineering and Project Management
4. Military
5. Logistics, Supply Chain and Distribution
6. Transportation Models
7. Business Process Simulation
8. Health Care

Law and Kelton (Law & Kelton, 1994) expanded these fields and added the fiscal analysis of a system and the regulations relevant to the hardware and software protocols that communication networks and computer systems could develop, as areas where simulation is not just applicable but also extremely beneficial to be used.

Simulation, though, is also thriving in the fields of insurance, options pricing, portfolio and call-center analysis (Banks et al., 1999). However, the last one is not suitable for application upon queuing models, because of its intricacy. In terms of expanded and more complex systems, both simulation hardware and software are enhanced with advanced abilities to cope with tremendously extensive numbers of entities within an acceptable time frame. Finally, Automated Material Handling Systems (AMHS) are experimenting with simulation models in order to develop and evaluate software related to control systems (Banks et al., 1999).

2.2.6 Systems and Models

Schmidt and Taylor (Schmidt & Taylor, 1970) referred to a system as an assemblage of entities, which entities, through processing and synergy, would provide some information and reasonable results. In each study, the term "system" is adjusted to meet the expectations and the needs of the inquiries. The "state" of a system is characterized by the assortment of the variables, related to the study's purposes, that are used to construe the system at a distinct time frame (Law & Kelton, 1994). The state can be whether discrete or continuous. Discrete systems are defined by discrete variables, while continuous systems are defined by continuous variables. Discrete variables change at a distinct set of points in time (Banks et al., 1999). For instance, a supermarket is a discrete system. The costumers that are waiting in line to pay for the products they bought are discrete variables, since their number is altered when a new costumer enters the supermarket, or an old one departs. In figure 4 one can see how a discrete variable behaves in time.

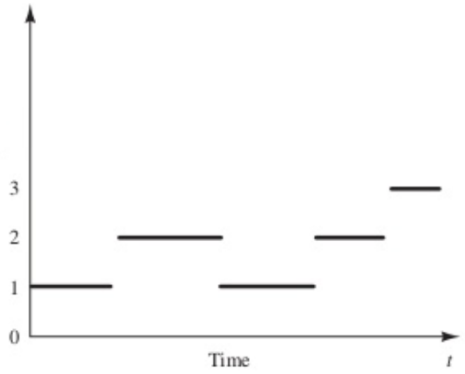


Figure 4: Discrete-system state variable (Banks et al., 1999)

On the other hand, in continuous systems, the variables are changing continuously over time (Banks et al., 1999). Continuous variables can take an uncountable set of real values into an interval. For instance, a temperature in a room is a continuous variable that can be transformed into a discrete one for easier estimate. In figure 5 one can see how a continuous variable behaves over time.

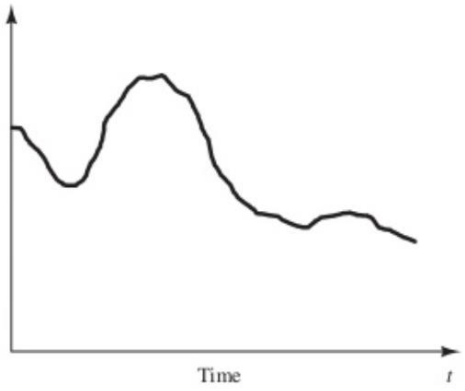


Figure 5: Continuous-system state variable (Banks et al., 1999)

In order to study systems, whether they are discrete or continuous, the creation of a model is inevitable. The reason is that most of the times, experimenting directly on the real-life system is not suggested or possible. A model is described as a portrayal of a system for studying and investigating purposes. In most researches, the features of the system that are under investigation are those which are directly connected with the presented problem. Those features

and aspects are considered in order to build the suitable model, which is a representation of the system in a much more simplified form.

Despite that simplified form, a model should nevertheless be sufficient and detailed in order to provide reasonable results that are actually based upon the real system's operation. Since the orientation of an investigation could change if new aspects occur and need to be examined, several models of the same system might be required to serve the new objectives. As a result, since just like a system, a model has its own entities, attributes and activities (Banks et al., 1999), they will have to be altered if a new model, serving a new purpose, is required.

When a system needs to be studied, for example for prediction purposes, several steps need to be taken under consideration in order to achieve a reasonable study. The ways that a system can be studied are presented in figure 6.

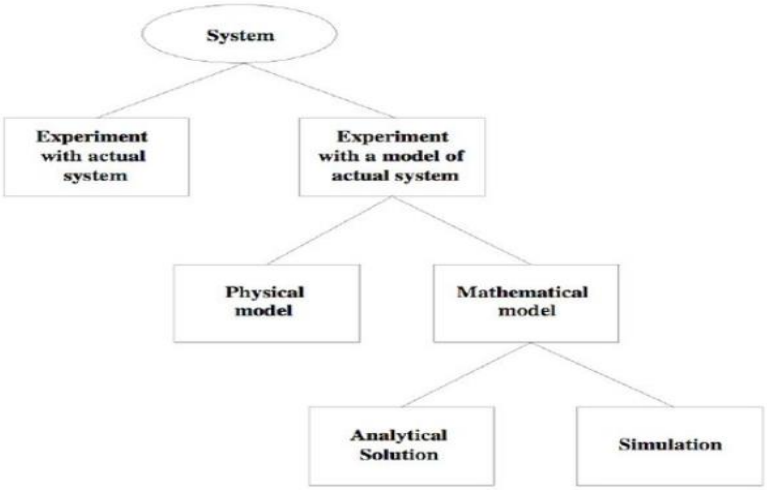


Figure 6: Perspectives when studying a system (Law & Kelton, 1994)

As presented in figure 6, first step in studying a system is to decide whether the experiment is going to be on the real-life system, or the model. Law & Kelton (Law & Kelton, 1994) claim that if a system can be studied physically, it's reasonable to be done this way. However, in most scenarios this isn't the case. As a result, a model is used to describe a system that it might not even exists in real life, yet it requires further investigation. This model will act as a portrayal of the system, however it needs to be validated in order to be accurate and true to

the system.

After deciding working on the model, it needs to be determined whether this will be a physical or a mathematical one. Although physical models can be useful in several studying areas such as engineering and management; tabletop scale models of material-handling systems for example (Law & Kelton, 1994), models based on mathematical and logical relationships between the components are more widespread, since they are easier to manipulate and change the variables when needed. Working on a valid mathematical model results to easier extraction of information about the system and to more accurate foundation in order to predict future behaviors based on the model's reaction to certain conditions.

The establishment of the mathematical model leads to further analysis and the solving procedure needs to be determined in order to complete the research's objectives and answering the inquiries of interest. The solution can surface whether by using the analytical solution method, or by using simulation. If the model is not defined by complexity, an analytical solution might be possible and even sometimes desirable, since simulation it's not a quite simple procedure itself, too. Nevertheless, in their majority systems that researchers come across are complex, leading to complex models as well. The very nature of these models excludes the possibility of an analytical solution, since the later is not applicable on composite systems.

Given that, simulation is inevitable and reasonable in order for the model to be investigated on the inputs and the outputs of the system's performance in a valid way and therefore, the simulation model now needs to be further examined. For this examination to take place, certain appliances could be proven useful, as well as the classification of the simulation models along three distinct dimensions (Law & Kelton, 1994), as presented in figure 7.

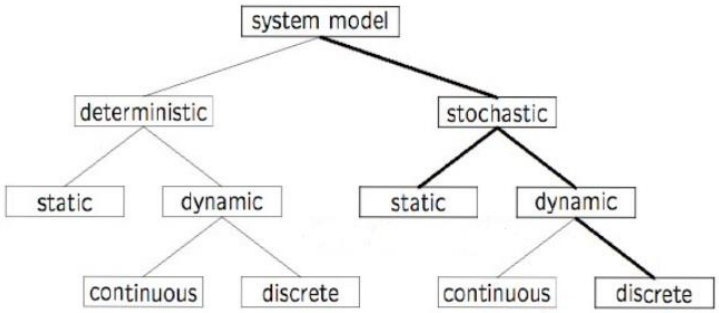


Figure 6: Models classification (Babulak & Wang, 2010)

First dimension refers to deterministic versus stochastic models. If a simulation model includes random elements, it's defined as deterministic. For instance, a complex system of differential equations depicting a nuclear reaction, would be classified as deterministic. In such models, the output is "determined" once the input is stated and evaluated (Law & Kelton, 1994). However, in other models there might be the need for certain components with a more probabilistic nature to exist. These models are the stochastic simulation models. Queuing models, for example, are usually stochastic. Due to their nature, though, stochastic models should be taken under consideration only as an appraisal of the actual attributes of the model.

Second dimension classifies a simulation model as static or dynamic. Static simulation models describe a system at a certain point in time, or they are used to present systems where time is a non-affecting factor. Monte Carlo models are often used in such occasions (Law & Kelton, 1994). Opposing to static ones, dynamic simulation models portray a system as it develops over time, taking under consideration all the possible changes that might occur.

Third and last dimension segregates models into discrete and continuous, which have been mentioned before. Nevertheless, it is useful to mention that discrete systems are not always chosen in order to describe a discrete system, and vice versa (Law & Kelton, 1994). The objectives of the simulation study will determine the type of model that is going to be used to describe a particular system. For instance, if someone wants to describe the car-flow in a highway and each car's attributes, such as speed, are critical to the study, then a discrete simulation model is suitable. On the other hand, if someone wants to study the car-flow as an aggregate phenomenon, differential equations in a continuous simulation model are more appropriate (Law & Kelton, 1994).

2.2.7 Simulation Models

2.2.7.A Monte Carlo Simulation

The Monte Carlo simulation is an approach which results are computed through the use of statistical analysis and repeated random sampling. This form of simulation is named after its namesake.This technique of simulation is quite similar to random experiments, which are tests in which the precise outcome cannot be predicted in advance.In this setting, the Monte Carlo simulation can be thought of as a methodical approach of performing an analysis known as a "what-if" study(Mason et al., 2008).

The Monte Carlo Simulation is based on the idea that the behavior of a statistic in a random sample may be evaluated by an empirical procedure that involves actually drawing a large number of random samples and monitoring how the statistic behaves in each of these samples.Creating an artificial environment, often referred as a pseudo-population, that is identical to the real world in all pertinent ways is the technique that will be used to accomplish this goal. This "pseudo-population" derives from mathematical processes that are used to generate sets of numbers that are similar in appearance to data samples that were selected from the "real" population. Many trials of the statistical method of interest are run, using the pseudo-population in order to investigate how the procedure behaves across different samples.

2.2.7.B Agent Based Simulation

An Agent Based Simulation takes a real-world system that consists of a set of entities and models and replicates that system.The ABS itself can be interpreted as a multi-agent system that is conducted by a collection of (software) agents. Thus, there is a correspondence between the real system and the multi-agent system, as well as between the (real) entities and the agents. Additionally, there is a correspondence between the multi-agent system and the real system. Words like "system" and "entity", are use to describe the real worlds, however when discussing simulation models, the correct terms are "multi-agent systems" and "agents"(Davidsson et al., 2007).

2.2.7.C System Dynamics Simulation

Whereas the common history of mechanical system dynamics has been already understood, breakthroughs in computational dynamics and related fields have only the recent years enabled the possibility to simulate the dynamic behavior of digital computers in real time. This development lays the groundwork for a wide range of applications involving operator-machine interaction, such as adapting the design of mechanical equipment to the capabilities of the human operator, conducting controlled human factors experiments associated with the safety of machine operations, and simulating designs prior to the fabrication and testing of expensive and potentially dangerous hardware (Schiehlen, 2013).

Real-time operator-in-the-loop dynamic modeling of large classes of mechanical systems has been hampered by the difficulty of formulating and solving the differential-algebraic equations (DAE) of dynamics until recently (Schiehlen, 2013). The capability to offer adequate visual and motion feedback to the operator, in order to generate an adequately realistic simulation of the real environment for human factors testing and design optimization, has also been hindered in some circumstances.

2.2.7.D Discrete-Event Simulation

In discrete-event simulation, the model has to describe a system that is continuously evolving through time. In the simulation model, though, the state variables are being altered promptly at distinct moments in time. These moments in time are defined by a single event that takes place at this particular duration. An “event” is described as a rapid appearance of a certain phenomenon, which leads in the possible alteration of the system’s state. Discrete-event simulation is analyzed with the aid of numerical methods over the analytical ones, and the model is “run”, rather than “solved”. During the simulation a huge amount of data needs to be stored or manipulated, which indicates that the use of a computer is mandatory.

In order for the discrete-event simulation to be easily understood, a discrete-event system’s example should be given. Consider, once again, a supermarket with only one server at the cash desk and costumers who are constantly arriving. We want to calculate the average waiting time (delay) in the queue that is formed, where the delay is defined as the time period between the moment that the costumer arrives in the supermarket’s line, until the moment the server begins to serve him. To estimate the expected delay, the simulation model’s state variables need to be determined. In the particular example, they would be the status of the server (free of busy), the amount of costumers waiting in the queue (if there are any), and the time that each costumer in the waiting line arrived. The status of the server is needed in order to establish whether a costumer can be immediately served at the point of his arrival, or he has to wait in the line. If the server is free, he proceeds and if the server is busy, the costumer joins the queue. When the server completes the procedure with a costumer, the number of the costumers waiting in line will determine whether his status will change to “free” (zero costumers in the queue), or it will continue to be “busy” (at least one costumer in the queue). The time of a costumer’s arrival is needed in order to determine his waiting time in the line, since the delay stands for the interval between the arrival time and the moment he begins being served (which at some point will be established). In this supermarket system there are two types of events that take place. The first event refers to the arrival of a costumer. When a costumer arrives, if there are no other people waiting in line he is going to be served immediately, changing the state variable “server’s status” from free to busy. On the other hand, if on his arrival the server is busy and he has to wait in the line, the state variable “amount of costumers waiting in line” is changing and it’s specifically increased by 1. In both occasions, a state variable is altered. Similarly, the second event refers to

a costumer's departure. When a costumer departs, two incidents can occur. If that costumer was the only one in the line, when his serving procedure is over, the server's status will change from busy to free. However, if there were actually more costumers in the line, the next one would proceed to the cash desk in order to be served, decreasing the costumers in the queue by 1. Again, in both incidents the state variables change.

A system like that could be represented by a discrete-event simulation model. However, not in all discrete-event simulation models the state variables change. This depends on the objectives and purposes of each simulation.

2.2.8 Simulation Study

Simulation is a whole procedure and every model built to represent a system is unique. However, researchers concluded in some certain steps that should take place into a simulation process (Shannon, 1975; Gordon, 1978). These steps are presented in figure 8.

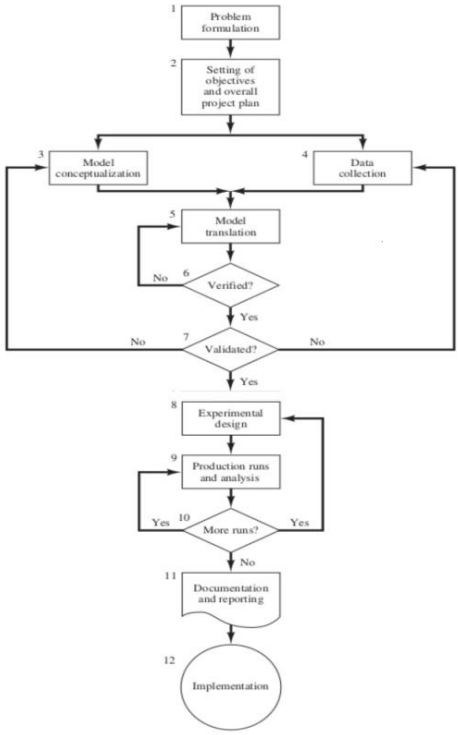


Figure 8: Steps in a Simulation Procedure (Banks et al., 1999)

Problem Formulation: At the beginning of every studying process, it's extremely important for the problem to be clear and understood.However, at certain occasions, the problem-statement

might need to change through the process of the simulation. All the parties participating in the study (analysts, policymakers, etc.) must be aware of the changes and agree to the new formulations. If the problem at the beginning is not appropriately stated, all the objectives and the procedures planned, might be in vain and will probably result to incorrect conclusions.

Setting of Objectives and overall Project Plan: At this point of the study it should be determined whether simulation is the appropriate method to proceed with in the study. Assuming it is, the objectives need to be defined, as well as, the project plan. The objectives represent the questions that the simulation study tries to answer and the project plan consists of alternative methods and ways to evaluate them. This step also includes all the practical information related to the study, such as the number of the people involved, the total cost, the expected results etc.

Model Conceptualization: In this step of the study, the building of the model begins. As W. Morris mentioned, *"although it is not possible to provide as set of instructions that will lead to building successful and appropriate models in every instance, there are some general guidelines that can be followed"*(Morris, 1967). The construction of the model includes the extraction of the vital attributes related to the problem, the selection and reformation of fundamental hypotheses that define the system and the enhancement of the model until it reaches its optimal state by a gradual evolution from a simpler to a more complex form. Nevertheless, the model should not be more complex than it actually should, since this could cause several dysfunctions during the simulation process.

Data Collection: This step of the study consists of the data gathering that will be used as input for the simulation model. As the model evolves in the model conceptualization step and it becomes more complex, there is a possibility that the data aspects might also alter. Data collection is a time consuming procedure and this is why it should be started in the early stages of the model's construction. The type of data that need to be collected for the purposes of the simulation study are determined based of the objectives. For instance, if the simulation model presents a supermarket-system that consists of a waiting line and a cash desk, and the objective is to determine the average delay of a customer in the queue, then the data that should be collected to run the model should be related with the arrival time of the costumers, the server's status etc.

Model Translation: Due to their complexity, the enormous amount of data that need to be

stored and manipulated and the estimates which have to take place, simulation models need to be loaded in a computer format. A suitable software or simulation language is being used to transform the model into a computer-friendly tool.

Verified?: Verification refers to whether the computer format built to represent the simulation model is appropriate. Models can be extremely complex and their translation in a computer environment requires a certain level of debugging (Banks et al., 1999). All the parameters related to the model need to be appropriately presented in the computer so that the verification can be completed.

Validated?: In order to achieve validation the model needs to be calibrated and constantly compared to the real life system. Through this continuous process new insights are gained and the model improves over time. The procedure will stop when the accuracy level between the simulation model and the real system is adequate.

Experimental Design: In a simulation procedure several scenarios are going to be run and tested. In the experimental design these scenarios need to be determined and designed. For each simulated scenario, several aspect need to be decided, such as the duration of the initialization period, the duration of the runs and the number of the distinct versions.

Production Run and Analysis: In this step, certain production runs and analysis occur in order to calculate and evaluate performance measures for the experimental designs of the system.

More Runs?: After the production runs and their analysis, the analyst has to decide whether further runs are needed and which design they should reflect.

Documentation and Reporting: In documentation there are two aspects to be considered. The first one refers to the computer format which was built in order to host the simulation model. If the same format is going to be used again, it is useful to understand how it functions, to provide results and decisions making according to the analysis. Keeping up with an appropriate documentation will be also useful in case the initial computer format (program) for the simulation model is going to be altered, or several parameters are going to change to bring the program in an optimal format. The second aspect refers to the progress report. According to K. Musselman (Musselman, 1998), progress reports are extremely vital since they document the

chronological evolution of the simulation, which includes every decision making aspect and completed objective. Regular progress reports are suggested to keep in track with the simulation model's evolution. The output of the analysis should be also reported. Through this procedure the users will be able to reassess the eventual formulation, the alternative scenarios, the results of the scenarios and the endorsed resolution to the problem.

Implementation: In order for a successful implementation to be achieved, it is required that the previous steps are executed appropriately. It can also be beneficial if the model user was involved in the building process of the model to better understand the statement of the problem that simulation is trying to resolve.

The steps shown in figure 8 can be divided in four phases. The first phase consists of the "Problem Formulation" and the "Setting of Objective and overall Project Plan" steps, where the building of the simulation model is preparing. The second phase refers to the next five steps ("Model Conceptualization", "Data Collection", "Model Translation", "Verification", "Validation"), where the actual building of the model takes place. The third phase engages steps 8, 9 and 10 ("Experimental Design", "Production Runs and Analysis", "Additional Runs"), where the simulation process begins and the *statistical experiment*(Banks et al., 1999) arises. Last but not least, the fourth phase includes the last two steps ("Documentation and Reporting" and "Implementation"). In this phase the implementation of the simulation takes place under the aid of documenting and reporting through the process, leading to optimal results. The most important step of the procedure seems to be the "Validation" one, since an invalid model can result to totally incorrect conclusions leading the implementation of the simulation in being vain.

During their research, Law and McComas (Law & McComas, 1989) encountered certain snags someone could face in a simulation study and should surpass in order for it to be successful. The first one refers to the inadequate specification of the objectives in the initial steps in the simulation study and to the usually improper level of details about the model. Moreover, another pitfall someone could face is the unsuccessful gathering of appropriate data, accompanied by an unsuitable choice of software which could lead to inappropriate appliance of the covetable modeling logic. Furthermore, a common misconception that could negatively affect the simulation process, is the assumption that more user friendly simulation packages demand little of no technical adequacy. Another failure could be caused, due to not taking into

consideration the random components that exist in the initial system, leading in frivolous choices of distributions as input in the model. In addition to the previous pitfalls, a usual misconduct is the use of certain packages which accept as a precondition, independence among the results of the simulation. This deficient assumption leads in the analysis of the output of a single simulation, without taking into account the results of the remained "runs". Relevant to that, comparing systems based on the results of only one single simulation run for each system's model, results to incorrect conclusions. Concurrently, inappropriate choice of performance measures leads to a misinterpretation of the simulation model's output. Last but not least, a poor selection of animation is also an aspect that should be taken under consideration during the simulation process.

2.2.9 Python and SimPy

As it has been already mentioned, new software has been developed and enhanced the simulation process. Examples of this kind of software are the Extend family, ARENA, MATLAB, Simulink etc. However, simulation can be also performed by using programming languages, such as C# or C++.

For the purposes of this master thesis, the programming language that will be used, is Python and more precisely the simulation is going to be performed in Spyder Environment with the use of the build in library for simulations, SimPy.

SimPy is an open-source software, released under the MIT License on December of 2002. It is a process-based simulation framework for discrete-event simulations and it's using Python's generators to model live parameters.

The simulation process through SimPy can vary from simple to complex, depending on the complexity of the real-life procedure. However, the simulation process in SimPy follows three basic steps. Firstly, one should establish the environment where the simulation is going to be performed. This includes the creation of an environment object that will handle the flow of the simulation, and it will lead the process though each subsequent event that needs to be completed before the next steps occur. Secondly, the creator of the simulation should set the variables that are going to take place in the simulation. These parameters stand for live components of the real-life procedure and are called through Python functions. Third step is of

course running of the simulation.

More information about SimPy framework will be given in the third chapter, where we are going to perform the simulation of the production process, more specifically the weaving process, of Fratelli Piacenza Industry.

CHAPTER 3: MODEL IMPEMENTATION

3.1 The FACTLOG Project

In Horizon 2020 innovation program, which was established by the European Union in order to maintain Europe's competitiveness in a global scale, a new initiative started to develop. The FACTLOG project with starting point November of 2019 and end date at April of 2023, was funded by Horizon 2020 to raise the standards in the field of technology. Yet, what is FACTLOG's main purpose?

The project focuses in the idea of Cognitive Digital Twins. A Digital Twin is a virtual simulation of a real-life asset and the main goal is to provide the ability to observe and follow the behavior of its physical twin. The ambition of the FACTLOG project is to develop a real-time process where observation, knowledge and experience are combined for the understanding of the behavior and processes of a system which is defined by complexity. In resemblance to ETC (European Credit Transfer) systems, which are autodidacts and they are able to detect and react accordingly to the situation when certain aberrations occur, the Cognitive Digital Twin is expected to act likewise.

In particular, the FACTLOG project's Digital Twins specialize in the creation of cognitive factories. There are twenty process industries taking part in this initiative, in the fields of steel and automotive manufacturing, oil refineries, waste-to-fuel transformer plants and textile industries. FACTLOG project offers a layer that combines Digital Twins driven by domain models, with data-driven models, for observation and monitoring purposes. The project aspires to novelty in the fields of Analytics, Artificial Intelligence and Optimization, in order for the development of contiguous Enhanced Cognitive Twins (ECTs) in the above mentioned fields to be achieved.

The project's intention is to embellish the European process industry and to provide both large and small enterprises with an optimizing production, enhanced with structured analyzing toolsets and adjustable to their needs and to any financial or sustainability concerned conditions. Furthermore, FACTLOG aims in the reinforcement of cognitive DTs with optimization means to identify aberration and variance in production processes, assess their impact and proceed with

rectifying actions while limiting human interference. In addition to that, FACTLOG's further goals include the radical transformation through ECTs notion, of all the production components, to a cognitive system that aims in the unsupervised re-optimization and management of its procedures. Moreover, the project focuses on the implementation of low cost manufacturing methods defined by asset-awareness and energy-efficiency, taking under consideration cases with automation and Internet of Things framework being in an introductory level.

In order for the FACTLOG project to achieve its goals, certain objectives need to be accomplished. Among others, these objectives include the description of the Cognitive Factory infrastructure model, the empowerment of automation as a process in manufacturing industries, as well as the implementation of model-driven analytics, and services based on the produced data. Moreover, the development of robust optimization procedures and toolsets, is also of great importance, as, additionally, the provision of a fully-developed 'as-a-service' apparatus, adjustable and easy to embrace and arrange, is. Furthermore, there has to be proof-of-concept, in a manner that the feasibility of everything that this project represents is evaluated and there is practical potential. FACTLOG needs to empower the association with academia and current research projects, not only under the auspices of SPIRE PPP, while also offering notable development of plant performance, especially in process and manufacturing industries. In addition to that, the detection of innovative business models and the transmission of knowledge regarding cognitive production components, taking under consideration scientific perspectives along with case studies, are essential.

3.2 Textile Industry Pilot

One of the pilots initialized in the FACTLOG project, concerns textile industries. The respective partner in this collaboration is the Fratelli Piacenza Industry. Piacenza is a woolen fabric manufacturer, as well as, a supplier to other fashion designers and manufacturers, such as Gucci, Prada, Louis Vuitton etc., located in Italy. Piacenza is competing in the field of fashion and designing innovation, customization and product quality, while withdrawing from the cost competition scene.

Piacenza carries the standards and peculiarities of a typical textile SME (Small Medium Enterprises). In the part of its production process relevant to the product quality, like weaving or finishing, Piacenza combines machines with a 10 years old ICT (Information and

Communications Technology) infrastructure. Nevertheless, due to the significant need for customization of fabrics and the rapid abatement of stack dimension, Piacenza has shown notable endeavors regarding the rejuvenation of its ICT infrastructures, in order to gather and use the produced data in an optimal way, and to upgrade its complex and heterogeneous production.

Through FACTLOG, the development of a progressive service-oriented architecture system for data collection and exploitation, integrating sensors, MES (Manufacturing Execution System), ERP (Enterprise Resource Planning System) and a production scheduler, all in one, is promoted, based on Case Base Reasoning (CBR), the process where current problems can be solved based on the solutions and gained knowledge from similar past problems.

Regarding the Piacenza case, this single service-oriented architecture (SoA), for its production process, requires a Production Unit Controller (PUC) to provide an initial set up to be applied, based on past cases concerning same or similar fabrics.

During the production process, a constant flow of data is delivered from the machines. In case of unanticipated events, the process will act based on CBR, and it will provide essential action approaches, based on similar past case scenarios. The gathered information from the Production Unit Controller are, at the same time, shared with the MES, ERP and production scheduler, as well as with the factory administrator. This SoA is constructed in a way to be open to exogenous data sources, especially information regarding the quality of input materials such as yarn for the respective production processes. It is also open to endogenous information sources regarding the processes' output (for instance, the produced fabric's quality), as well as the process's performance, such as machine speed. More information can produce a well detailed database, which is rather crucial for the SoA, since the more variant the data it processes, a more precise action list can be suggested, based on each detected event.

Regarding the majority of Piacenza's facilities, the current foundation is able to collect the respective data, essential in accordance to FACTLOG's perspective. However, there are existing processes and steps where technologically former equipment is used and the data obtaining process appears quite hard. In those cases, the establishment of extra sensors might be necessary in order to process and produce data.

As mentioned above, cognition has become the popular solution regarding the ‘unknown unknowns’, risks derived from situations with unexpected and unforeseeable conditions. Cognition’s vital role is also recognizable in production processes, like finishing. Regarding the weaving procedure, given that all the other variables remain fixed and continuous, a gradual wearing of certain components would eventually cause the production process to stop, with the later being an anticipated event. Yet, in processes such as finishing which are usually one shot procedures and the attributes of the fabrics are altered for each design, an unexpected outcome can always occur. This leads to the realization that in these cases an immediate data driven solution is not attainable. Thus, knowledge of previous cases is essential for a suitable solution to be provided, and that is the very fact that cognition represents. In the case of finishing procedure, for instance, the behavior of the process’s components and the intricate relations among them have to be examined and observed in order to for the dynamics of the procedure to be unraveled.

3.3 Case Description

By entering FACTLOG’s pilot, Fratelli Piacenza Industry sets two main goals; Firstly, to optimize her production and secondly, to decrease the cost, using the fundamental idea of FACTLOG; a Digital Twin. This Twin will be a virtual representation of Piacenza’s production process.

In order for the Twin to be implemented, there are two basic dimensions that need to be considered; Simulation and Optimization, leading in the construction of two models. The first one is the Simulation model, where all the variables, components and their in between relationships and dynamics, are portrayed in a structured framework and simulate the actual production process. The Optimization model on the other hand is a representation of the essential characteristics of the business problem that requires solution (in this case cost reduction) and consists of three main aspects; the objective function, the decision variables and the business constraints. These two models cooperate in order to implement the Digital Twin.

Nevertheless, this paper focuses on the implementation of the Simulation model for a specific part of Piacenza’s production process, weaving. In order to understand how the Simulation model will be applied on this procedure, one should explore the total production process of Piacenza.

Firstly, Piacenza proceeds with orders of raw materials such as wool and alpaca, from her suppliers. These raw materials are transforming into yarns through the spinning process, to be later used for Piacenza's production. This step of the process is carried out by a supplier company; Piacenza does not include it in its own production process. Due to that fact, the spinning cannot be optimized for either production costs or cost savings.This stage ought to be regarded as a consistent component across the entirety of the production process.

After the spinning process is completed and the yarns are ready, production enters the weaving phase, which is the phase of interest for the simulation model presented later on. In this phase, sets of yarns are aligned in order to be processed by the looms (devices used to weave cloth and tapestry) and be transformed into fabric. This is one of the most intricate procedures in Piacenza's production process since it requires the combination and cooperation of multiple components.

Finally, there is the finishing process. In this step of the production, the already manufactured fabrics go through finishing hand touches and shrinking, in order for the orders to be completed. Finishing process could also affect cost reduction and production optimization, yet this is something that will not be examined in this particular paper.

As mentioned above, the process of interest for the Simulation model will be weaving. The weaving department receives internal and external orders for fabric production. An order might be for a sample or it might be for regular production.In general, depending on the quantity of an order, it is possible to classify the orders into samples and regular production; sample length is less than 25 meters and for regular production is more than 25 meters. Each order has certain characteristics; an ascending identifying number that is unique for each order, a type status for sample or regular production classification, a priority scale from 1 to 10 with 1 being a high priority order and 10 being a low priority order, a quantity indicator (in meters) for the required amount of fabric in each order, CA and CC codes whose combination is vital for the determination of whether there will be different fabrics chaining or there will be set up of a new loom, a product code that indicates the number of strokes per meter, a complexity indicator that portrays the intricacy of each order, a release date before which the order cannot be scheduled for production and a due date that is the deadline of the order's delivery. After the orders are received, they are scheduled for production in the weaving department which is composed by a

set of looms with each loom having their own characteristics such as speed and performance index.

The goal is to create a Simulation model for the order scheduling of the weaving process. The dynamic scheduling however can be affected from several events such as delayed orders, loom failures, broken yarns etc., which also have an impact on the cost reduction goal (saving energy costs and respecting orders' deadline). These possibilities need to be portrayed in the model, while also taking under consideration variables contributing in cost's calculation and management, such as looms' set up cost or fixed energy cost.

3.4 Simulation in SimPy

As mentioned on Chapter 2, the simulation will be run on Anaconda's Spyder Environment using the basic framework of Python's build in library for simulations, SimPy.

The SimPy package is used for discrete-event simulations. Processes are used to model the behavior of active components such as automobiles, consumers, or communications. All processes reside in a specific environment and they interact with this surrounding environment through the use of events.

The Environment, Events, and the Process functions that the user develops are the basic components engaged in the simulation. The simulation time, as well as the scheduling and processing of events, are all managed by the environment in which the simulation is run. In addition to that, it provides the user with the ability to step through or run the simulation.

SimPy can be simplified to the point that it is understood as an asynchronous event dispatcher. The user can generate events and then schedule them to be triggered at a specific point in the simulation. The order of the events is determined by their priority, the amount of time remaining in the simulation, and an ascending event id. An event also has a list of callbacks, which are performed when the event is triggered and processed by the event loop. These callbacks are executed after the event has been processed. Event instances are used to define many types of events. Events can exist in one of the subsequent states; it might happen (not triggered), it is going to happen (triggered), or it has already happened (processed). These states are traversed exactly once in this order. Firstly, no events are triggered; only objects exist in memory. If an event is triggered, it is added to SimPy's event queue and scheduled for a specific

time. Moving to the final state of an event (processed), SimPy processes an event by removing it from the event queue and calling all of its callbacks. Return values are often produced from events as well.

Regarding the functions, the simulation model is implemented by the process functions in a way so that these functions are able to describe the behavior of the simulation. Simple Python generators (functions that allow the declaration of other functions that behave like iterators) are used to describe the processes. Depending on whether or not it is a typical function or method of a class, they are referred to as either a process function or a process method. During the course of their existence, they create events and then yield those events awaiting for them to be triggered. The environment stores these events in an events list and maintains the current simulation time.

When a process yields an event, that process is being put on hold until this event resurfaces. When the event occurs or more accurately, is being triggered, SimPy will start the process once again. It's possible for multiple processes to wait for the triggering of the same event at the same time. SimPy resumes them in the same order that they yielded this particular event. Thus, a process is passivated and can respectively be reactivated.

These are the fundamental components when simulating with SimPy. In the next paragraph, where the Weaving process Simulation model of Piacenza is going to be presented, the reader will have the opportunity to further explore SimPy's simulation module.

3.5 Model Description

As already mentioned, the environment where the Simulation is going to be performed is Anaconda's Spyder. Usually, when Spyder is being installed, most of the build in packages are being installed as well. Nevertheless, SimPy might have to be manually installed using the Command Line and the command 'conda install -c conda-forge simpy'.

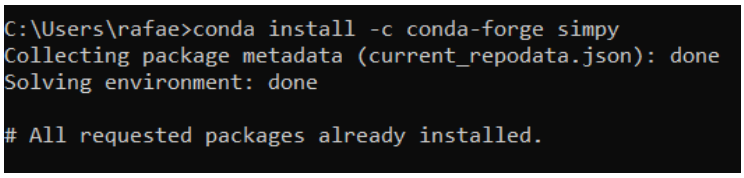


Figure 9: Installing SimPy in Anaconda's Spyder Environment via cmd

First step for the construction of the Simulation model is to import the necessary libraries.

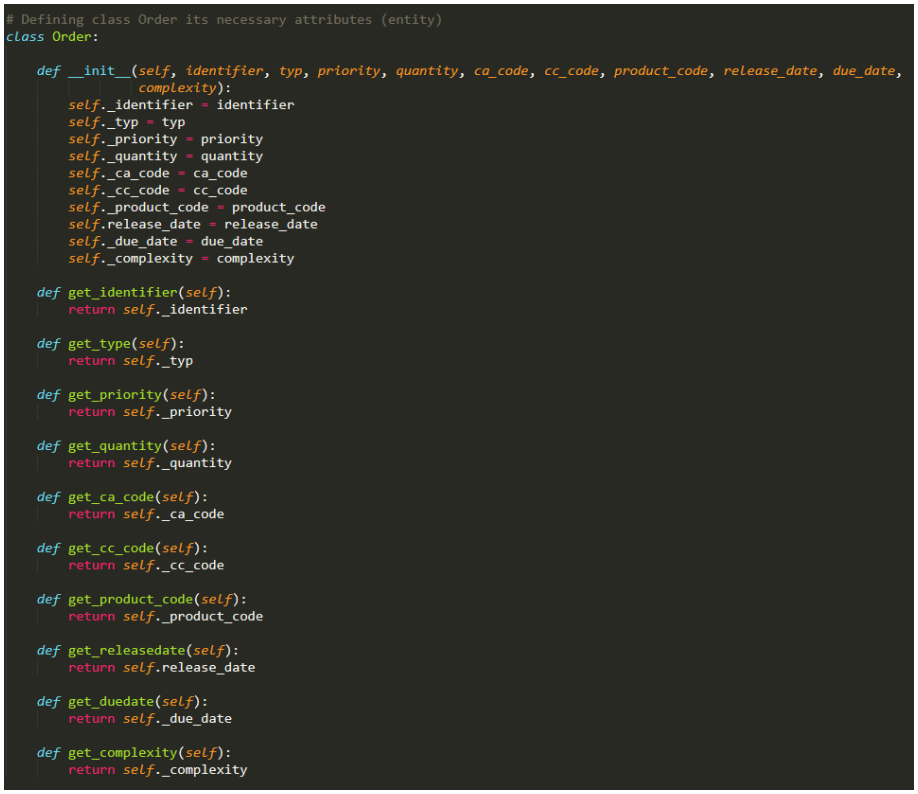
For this particular model, these are;SimPy, Random and Randint from Random, Pandas, DateTime and TimeDelta from DateTime, CSV, Math.

```
# Imorting the necessary modules
import csv
import math
from datetime import datetime, timedelta
from random import random, randint
import pandas as pd
import simpy as sp
```

Figure 10: Importing the required modules in Spyder

SimPy and its functionality have already been described above. Random module implements pseudo-random generators for various distributions. Random and Randint are methods of the Random module. The first one returns a pseudorandom double type number greater than or equal to 0.0 and less than 1.0, and the second one returns a random integer between a specified range. The sequence can be a string, a range, a list, a tuple, or any other kind of sequence. Pandas module is an open source library in Python and it provides ready to use high-performance data structures and data analysis tools. Moreover, there is DateTime module. DateTime supplies classes to work with date and time. The required classes for this simulation are DateTime and TimeDelta. DateTime class is a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond and tzinfo, while TimeDelta class stands for the duration expressing the difference between two date, time or datetime instances to microsecond resolution. Last but not least, CSV module is used to import or export spreadsheets and databases, while Math module includes from simple to quite complex mathematical functions.

Second step is to define the entity of the simulation. Entities are the objects flowing though the sequential processes in the model. In Piacenza's case, entities are the orders arriving and requiring the initiation of the weaving procedure. This entity in presented in the class Order. Classes are used in object oriented simulation in SimPy to provide information regarding the object (in this case the Orders) that is going to be used later on, on the model. This information is defined as attributes.



```
# Defining class Order its necessary attributes (entity)
class Order:

    def __init__(self, identifier, typ, priority, quantity, ca_code, cc_code, product_code, release_date, due_date,
                 complexity):
        self._identifier = identifier
        self._typ = typ
        self._priority = priority
        self._quantity = quantity
        self._ca_code = ca_code
        self._cc_code = cc_code
        self._product_code = product_code
        self.release_date = release_date
        self._due_date = due_date
        self._complexity = complexity

    def get_identifier(self):
        return self._identifier

    def get_type(self):
        return self._typ

    def get_priority(self):
        return self._priority

    def get_quantity(self):
        return self._quantity

    def get_ca_code(self):
        return self._ca_code

    def get_cc_code(self):
        return self._cc_code

    def get_product_code(self):
        return self._product_code

    def get_releasedate(self):
        return self.release_date

    def get_duedate(self):
        return self._due_date

    def get_complexity(self):
        return self._complexity
```

Figure 11: Defining class Order and its attributes

As presented in figure 11 class's Order attributes are; identifier (ascending number, unique for each order), typ (sample of regular production status), priority (scale from 1 to 10 with 1 being high priority), quantity (quantity indicator in meters), CA and CC codes as explained in Case Description, product code (number of strokes per meter), release date (date before which the order cannot be scheduled), due date (order's deadline) and complexity (indicator portraying the intricacy of each order).

Besides the entity, the Resources of the simulation should also be defined. Resources are necessary in order for the activities of the simulation to take place. For instance, in a retailing order queue, the entities are the clients and the resources the cashiers. In Piacenza's case, the Recourses are the Looms which are also defined on their own class.

```python
# Defining class Loom its associated properties (resources)
class Loom:

    def __init__(self, loom_id, loom_speed, performance_index):
        self.loom_id = loom_id
        self.loom_speed = loom_speed
        self.performance_index = performance_index
        self.prev_job = None
        self.current_job = None

    def set_prev_job(self, job):
        self.prev_job = job

    def set_current_job(self, job):
        self.current_job = job

    def get_loom_id(self):
        return self.loom_id

    def get_prev_job(self):
        return self.prev_job

    def get_current_job(self):
        return self.current_job

    def get_loom_speed(self):
        return self.loom_speed

    def get_performance_index(self):
        return self.performance_index

    def get_loom_original_speed(self):
        return self.loom_speed * self.performance_index
```

Figure 12: Defining class Loom and its attributes

Class's Loom attributes are; loom id, loom speed, performance index, current job and previous job. Current job refers to the order that is now being process on the loom, and previous job, to the order that we processed before. However, apart from these characteristics, there is also loom original speed which derives from the multiplication of the previous two attributes. It is a way to create a single variable that includes both speed and performance information for each loom.

Continuing, a database with the orders is imported in the simulation model (job_details.xlsx file). As presented in the figure above, all the attributes listed in class Order are columns to this dataset. This dataset consists of dummy data.

identifier	type	priority	quantity	ca_code	cc_code	product_code	release_date	due_date	complexity
id1	regular	3	1200	ca_1	cc_1	1474	7/7/2022	7/9/2022	1.2
id2	regular	1	2200	ca_1	cc_2	3751	7/7/2022	7/12/2022	1.3
id3	regular	3	2100	ca_2	cc_2	4363	7/11/2022	7/15/2022	1.5
id4	sample	6	22	ca_3	cc_2	7532	7/12/2022	7/15/2022	1.4
id5	regular	2	2300	ca_1	cc_3	1245	7/13/2022	7/16/2022	1.2
id6	sample	7	23000	ca_4	cc_1	3000	7/13/2022	7/16/2022	1.1
id7	regular	8	3500	ca_2	cc_4	1888	7/14/2022	7/17/2022	1.3
id8	regular	2	2000	ca_2	cc_1	5743	7/15/2022	7/19/2022	1.5
id9	regular	4	5000	ca_1	cc_2	2344	7/20/2022	7/29/2022	1.6
id10	regular	8	15000	ca_1	cc_2	2573	7/20/2022	7/23/2022	1.8

Figure 13: Orders' Database

Python opens the file as a pandas data frame, sorts the values according to the order

priority and iterates through the jobs storing the information for each unique Order in an object named order. This object is appended in the list all_jobs which now holds all the information for each order.

```
# Defining a list to store all jobs
all_jobs = list()
# Opening the excel file of jobs/orders
jobs = pd.read_excel("job_details.xlsx")
jobs.sort_values(by=["priority"], inplace=True)
# Iterating over jobs in excel file
for j in jobs.iterrows():
    # Getting current iterated job
    job_details = j[1]
    id, typ, prior = job_details["identifier"], job_details["type"], job_details["priority"]
    cap, ca, cc = job_details["quantity"], job_details["ca_code"], job_details["cc_code"]
    pc, rd, dd = job_details["product_code"], job_details["release_date"], job_details["due_date"]
    complex = job_details["complexity"]

    # Defining an order object with job details
    order = Order(id, typ, prior, cap, ca, cc, pc, rd, dd, complex)

    # Appending the list of all jobs with order
    all_jobs.append(order)
```

Figure 14: Creating list all_jobs with information for each Order

During the simulation, several events take place and certain calculations are being performed; they are portrayed in Python as functions. These functions can be called through any part of the code and can be nested inside other functions.

Setup_time function: When two different fabrics are following one another on a loom, extra time might be required in order to setup the loom accordingly for the transition. The required amount of time depends on the combination of the CA and CC codes of the fabrics/orders. If the CA code in fabric A equals CA code in fabric B and CC code in fabric A equals CC code in fabric B then no extra setup is needed (setup time equals 0 hours). If CA code in fabric A equals CA code in fabric B, yet CC code in fabric A does not equal CC code in fabric B, then two hours are required for the respective setup (setup time equals 2 hours). Lastly, if neither CA code in fabric A equals CA code in fabric B, nor CC code in fabric A equals CC code in fabric B, then eight hours are required for the loom to be appropriately set (setup time equals 8 hours).

```
# This function will calculate setup_time of loom based on ca and cc codes of current and previous processed job
def setup_time(ca1, cc1, ca2, cc2):

    if ca1 == ca2 and cc1 == cc2:
        return 0
    elif ca1 == ca2 and cc1 != cc2:
        return 2
    else:
        return 8
```

Figure 15: Setup_time function

Processing_time function: This function calculates the processing time for each order based on the quantity, the complexity and the original speed of the loom that this order is appended to.

```
# This function will calculate the processing time of a job based on its complexity
def processing_time(job, loom):

    comp = job.get_complexity()
    time = job.get_quantity() * comp / loom.get_loom_original_speed()
    return time
```

Figure 16: Processing_time function

As mentioned above, certain events can take place in the simulation. Firstly, a new order may occur, and it will also have to be scheduled for the weaving process. Moreover, there might be a case of broken yarn that needs restoration or change, or more rarely, there might be a case of a damaged loom that will need to be restored. New_job, Damaged_loom and Broken_yarn functions describe the aforementioned events.

```
# This function checks if there is a new order entering the production (event of simulation)
def new_job():

    chance = random()
    if chance > 0.7:
        print("New Job!")
        return True
    return False
```

Figure 17: New_job function

```
#  This function will return True if loom is damaged (event of simulation)
def damaged_loom():

    chance = random()
    if chance > 0.99:
        print("Found a Damaged Loom!")
        return True
    return False
```

Figure 18: Damaged_loom function

```
# This function checks if there is a broken yarn case (event of simulation)
def broken_yarn():

    chance = random()
    if chance > 0.85:
        print("Found Broken Yarn!")
        return True
    return False
```

Figure 19: Broken_yarn function

These functions operate in the same way. Through random method, a haphazard float number is chosen between 0 and 1. If this number is greater than a certain fixed constraint then the event of a new job, damaged loom and/or broken yarn occurs (numbers 0.7, 0.99 and 0.85 are adjustable and can be changed).

Take datetime and make it days month year: This function holds as an argument a string object (that reflects a date) and identifies the year, month and day.

```
def take_datetime_and_make_it_days_month_year(str_date):
    dt = str_date
    return dt.year, dt.month, dt.day
```

Figure 20: Take_datetime_and_make_it_days_month_year function

Random job: Random job is a function that generates values for a new order. This new order's release and due date will be at some point among sixty days from the beginning of the simulation.

```
# This function generates a new order
def random_job(id):
    if int(randint(0, 1)) == 0:
        typ = 'regular'
    else:
        typ = 'sample'

    prior = randint(0, 10)
    if typ == 'sample':
        cap = randint(1, 25)
    else:
        cap = randint(25, 100000)

    ca = 'ca_' + str(randint(0, 10))
    cc = 'cc_' + str(randint(0, 10))
    pc = randint(1000, 9999)
    rd_d = randint(1, 30)
    rd_m = randint(7, 8)
    rd = datetime(2022, rd_m, rd_d)

    dd_d = randint(1, 30)
    dd_m = randint(7, 8)
    dd = datetime(2022, dd_m, dd_d)

    while rd > dd:
        dd_d = randint(1, 30)
        dd_m = randint(7, 8)
        dd = datetime(2022, dd_m, dd_d)

    complex = randint(1, 10)
    return Order('id'+str(id), typ, prior, cap, ca, cc, pc, rd, dd, complex)
```

Figure 21: Random_job function

Jobs available: This function identifies which jobs/order can enter the weaving process, based on their release date and the current date of the simulation.

```python
def jobs_available(jobs, current_day):
    a_jobs = []
    for job in jobs:
        y, m, d = take_datetime_and_make_it_days_month_year(job.get_releasedate())
        s_releasedate = datetime(y, m, d)
        if s_releasedate <= current_day:
            a_jobs.append(job)

    return a_jobs
```

Figure 22: Jobs_available function

Selected job: When a job is entering the weaving process, it has to be appended in a loom. This function, uses the setup_time function to calculate what the setup time will be if this job would be appended in each of the available looms. This will be later used in order to select the most appropriate loom (best case scenario, the loom that will need no setup time in order to execute the order) to append the job on.

```python
def selected_job(jobs, prev_job):
    if prev_job is None:
        return jobs[0]

    min_time = 10
    s_job = jobs[0]
    for job in jobs:
        if setup_time(job.get_ca_code(), job.get_cc_code(), prev_job.get_ca_code(), prev_job.get_cc_code()) < min_time:
            min_time = setup_time(job.get_ca_code(), job.get_cc_code(), prev_job.get_ca_code(), prev_job.get_cc_code())
            s_job = job

    return s_job, min_time
```

Figure 23: Selected_job function

New jobs & Delete random jobs: New jobs is a function that creates a new job list without the selected job (s_job). Delete random jobs, then utilizes this list in order to remove a job that has been executed and continue the process with the available ones.

```python
def new_jobs(jobs, s_job):
    n_jobs = []
    for job in jobs:
        if job != s_job:
            n_jobs.append(job)

    return n_jobs
```

Figure 24: New_jobs function

```python
def delete_random_jobs(jobs, original_jobs_total_number):
    n_jobs = []
    for job in jobs:
        num_str = ""
        for m in job.get_identifier():
            if m.isdigit():
                num_str = num_str + m

        if int(num_str) <= original_jobs_total_number:
            n_jobs.append(job)

    return n_jobs
```

Figure 25: Delete_random_jobs function

Weaver function: This is the main function of the weaving process, where all the tasks are being assigned.

```python
def weaver(environment, loom):
    global start_time, active_time, jobs, setup_cost, ending_times, jobs_total_number, all_jobs
    ending_times = dict()
    while len(jobs) > 0:

        # print("loom", loom.get_loom_speed())
        # print("all_jobs len: ",len(jobs))
        # for job in jobs:
        #     print("all_jobs: ", job.get_identifier())

        simulation_start_time = environment.now
        days, hours = simulation_start_time // 8, simulation_start_time % 8
        current_date = start_time + timedelta(days=days, hours=hours)
        a_jobs = jobs_available(jobs, current_date)

        # print("avl_jobs len: ", len(a_jobs))
        # for job in a_jobs:
        #     print("all_jobs: ", job.get_identifier())

        if len(a_jobs) > 0:
            if new_job():
                r_job = random_job(jobs_total_number)
                jobs_total_number += 1
                jobs.append(r_job)
                # for job in a_jobs:
                #     print("all_jobs: ", job.get_identifier(), job.get_priority())
                jobs.sort(key=lambda x: x._priority, reverse=False)
                # for job in a_jobs:
                #     print("all_jobs: ", job.get_identifier(), job.get_priority())
                all_jobs.append(random_job(jobs_total_number))

            if loom.get_prev_job() is None:
                s_job, setup_time = a_jobs[0], 0
            else:
                s_job, setup_time = selected_job(a_jobs, loom.get_prev_job())

            # Print the starting date of a job and the loom it has been assigned to
            print("Started working on", s_job.get_identifier(), "at", current_date.strftime("%m/%d/%Y"), "at Loom",
                  loom.get_loom_id())

            loom.set_current_job(s_job)
            jobs = new_jobs(jobs, s_job)
            setup_cost += s_job.get_quantity() * setup_time
            yield env.timeout(setup_time)
            yield env.timeout(processing_time(s_job, loom))
            active_time += processing_time(s_job, loom)
            # If a loom is damaged, repair it, which takes 15 minutes
            if damaged_loom():
                yield env.timeout(0.25)

            # If there is broken yarn found, repair it, which takes 15 minutes
            if broken_yarn():
                yield env.timeout(0.25)


            simulation_start_time = environment.now
            days, hours = simulation_start_time // 8, simulation_start_time % 8
            ending_time = start_time + timedelta(days=days, hours=hours)
            # Printing the ending date of a job
            print(s_job.get_identifier(), "ended at", ending_time.strftime("%m/%d/%Y"), "at Loom",
                  loom.get_loom_id())
            loom.set_prev_job(s_job)
            ending_times[s_job.get_identifier()] = environment.now

        yield env.timeout(1)
```

Figure 26: Weaver function

At first, the simulation runs with a specific dataset of orders. However, during the time that the orders are being processed, a new one may occur. This job is joined with the rest of the orders waiting to be scheduled and is sorted according to its priority. Using the jobs_available function, all the jobs that can actually be scheduled (based on their release date) enter the loom based on their priority. Each job has to be assigned to a loom. To get the optimized loom-job pair, the previous job processed in each loom is identified and the setup time is calculated. The job gets assigned to the loom with the minimum setup time needed and it is executed based on its calculated processing time (calculated using processing time function). While taking under consideration events like damaged looms or broken yarns that can delay the job execution completence, the start and end date are printed, as well as the loom that the job was assigned to.

In order for the simulation to run in success, looms' characteristics need to be set, along with the simulation's environment and the starting date of the simulation.

```
all_jobs = delete_random_jobs(all_jobs, original_jobs_total_number)
global ending_times
env = sp.Environment()
start_time = datetime(2022, 7, 7)  # The start date of simulation, adjustable
# We can define more looms, depending on the conditions for each simulation
# For each loom, id, speed and performance index should be defined here
L1 = Loom("l1", 200, 0.5)  # First loom characteristics
L2 = Loom("l2", 300, 0.5)  # Second Loom characteristics
L3 = Loom("l3", 400, 0.8)  # Third Loom characteristics
# For each loom, speed and performance index should be defined here
looms = [L1, L2, L3]
```

Figure 27: Setting the Looms

With the Resources set, it is time to define the whole setup for the weaving process for each loom, as well as the simulation duration.

```
jobs = all_jobs
jobs_total_number = len(original_jobs)+1
env.process(weaver(env, looms[0]))
env.process(weaver(env, looms[1]))
env.process(weaver(env, looms[2]))

# Number of days the simulation is running, adjustable
days_to_simulate = 60
# Running the environment for number of days based on 8 hours of work/day
env.run(until=days_to_simulate * 8)
```

Figure 28: Defining setup

The simulated days are set to 60, which is adjustable. The simulation model will run from

simulation date 7/7/2022 until 60 simulation days has been reached.

With the simulation ending, there might be some orders that are incomplete, meaning that they did not manage to resolve during the simulation time, or there might be Orders that have been completed violating their due date. As presented in figure 29, the delay (if any) for every order, the cost due to those delays (the quantity of the order multiplied by the delayed days), and also the amount of delayed orders and unfinished orders, are calculated.

```
# Defining a dictionary object to store the penalty loss of orders
lost_profit_from_delays = dict()
print()
# Calculating Profit Loss
count_delayed_orders = 0
count_unfinished_orders = 0
# Iterating over jobs to calculate penalties for late orders
for j in all_jobs:
    # print(j.get_identifier())
    try:
        id = j.get_identifier()
        delivered_on = ending_times[id]
        days, hours = delivered_on // 8, delivered_on % 8
        delivered_on = start_time + timedelta(days=days, hours=hours)
        delay = delivered_on - j.get_duedate()
        if delay.days > 0:
            fine = delay.days * j.get_quantity()
            lost_profit_from_delays[id] = fine
            count_delayed_orders = count_delayed_orders + 1
    except:
        print("Unfinished Order:", j.get_identifier())
        # Some jobs might remain unfinished at the end of simulation, print them
        count_unfinished_orders = count_unfinished_orders + 1
```

Figure 29: Calculating Cost due to delays and the amount of Delayed and Unfinished Orders

Last step is to provide a calculated result regarding the Cost of production. Cost is affected by three factors; the cost due to delayed orders as described above, the fixed energy cost which is the cost for the energy the looms consume when they are active (when they are processing an order) and the setup cost which is the cost for every meter of fabric that could be processed but it is not due to the need of resetting the loom (it's the multiplication of quantity and the setup time). The code prints the calculated fine per delayed job, the total lost profit due to delayed orders and the total loss which also includes the fixed energy and setup cost.

```
print("Amount of unfinished orders:", count_unfinished_orders)
print()
print("Total Fine per delayed job: ")
# Iterating over lost profit and printing
for j, f in lost_profit_from_delays.items():
    print(j, f)
total_lost_profit_from_delays = sum(lost_profit_from_delays.values())
print("Total Lost Profit at the end due to delays:", total_lost_profit_from_delays)
fixed_energy_cost = fixed_energy_cost * active_time
total_loss = int(total_lost_profit_from_delays + fixed_energy_cost + setup_cost)
# Printing the total fine taking under consideration setup and loom active time fee
print("Total Loss after including setup fees and active time fees:", round(total_loss))
```

Figure 30: Calculating and printing results

3.6 Single Simulation Results

The figure below is a portrayal of the simulation's results, as presented in Spyder's console. There are messages informing the user on when an order id is starting to be processed and in which loom and when the weaving process for this id is finished. There are also warning messages regarding the events of new jobs, broken yarns and/or damaged looms. After the simulation days have reached the deadline (in this case the simulation days were sixty), the ids of unfinished orders are printed, as well as their total amount. Moreover, the ids of the delays orders are presented, along with the respective cost effect that each delayed order had on the total loss. Finally the total lost profit due to delayed orders is printed, as well as the total loss after the inclusion of setup and looms' active time fees.

```
Started working on id2 at 07/07/2022 at Loom l1
Started working on id1 at 07/07/2022 at Loom l2
Found Broken Yarn!
id1 ended at 07/08/2022 at Loom l2
id2 ended at 07/10/2022 at Loom l1
Started working on id3 at 07/11/2022 at Loom l3
New Job!
Started working on id4 at 07/12/2022 at Loom l1
Started working on id11 at 07/12/2022 at Loom l2
id3 ended at 07/12/2022 at Loom l3
Started working on id5 at 07/13/2022 at Loom l3
id4 ended at 07/13/2022 at Loom l1
Started working on id6 at 07/13/2022 at Loom l1
id5 ended at 07/15/2022 at Loom l3
Started working on id8 at 07/15/2022 at Loom l3
id8 ended at 07/17/2022 at Loom l3
Started working on id7 at 07/17/2022 at Loom l3
id7 ended at 07/19/2022 at Loom l3
Started working on id9 at 07/20/2022 at Loom l3
id9 ended at 07/24/2022 at Loom l3
Started working on id10 at 07/24/2022 at Loom l3
id10 ended at 08/03/2022 at Loom l3
id6 ended at 08/14/2022 at Loom l1
id11 ended at 08/27/2022 at Loom l2

Unfinished Order: id12
Amount of unfinished orders: 1

Total Fine per delayed job:
id6 667000
id7 7000
id10 165000
Total Lost Profit at the end due to delays: 839000
Total Loss after including setup fees and active time fees: 1301152
```

Figure 31: Printed results of a single simulation

3.7 Multiple Simulations

As mentioned in paragraph 3.6, those were the results of a single simulation run. However, by running the simulation several times, different results are presented each time. This is due to the fact that each run of the simulation will have different random samples for possibility event occurrence, activity times etc, and the appearance of outliers might lead in

deceptive results and observations. Hence, it is highly important in stochastic models that the simulation run is not performed just once if there is need to draw insights from the results.

A more efficient way to draw accurate insights from the model is to perform the simulation several times and take summary statistics over the results of each run to get more representative results from the model. Thus, the necessary alterations are being implemented to the script in order for the simulation to run multiple times. For this simulation, the model will run 100 times.

At first, a csv file that is going to store the desired results of each simulation is created and labeled "weaving_process_results.csv". The columns of this file and in extension, the results' column are also being defined here. There is "Run" which is the ascending number for the simulation run that is being performed, "Delayed Orders" which is the amount of orders that have been delayed, "Unfinished Orders" which is the amount of orders that were not completed when simulation days reached their end, "Lost Profit due to Delays" which includes the sum of delaying costs and described before for each run, and finally "Total Loss" which incorporates "Lost Profit due to Delays" with setup and looms' active time fees, again, for each run.

```
number_of_simulation_runs = 100
#Defing the number of simulations that will run, adjustable

with open("weaving_process_results.csv", "w") as f:
    writer = csv.writer(f, delimiter=",")

    writer.writerow(["Run", "Delayed Orders", "Unfinished Orders", "Lost Profit due to Delays", "Total Loss"])
# Creating a file to store the results of each run and write the column headers
```

Figure 32: Creating weaving_process_results.csv file

Next step is to nest the definition of the set up (environment processes), Resources definition and delayed and unfinished orders calculation, inside a 'for' loop that will run 100 times.

```
for run in range(number_of_simulation_runs):
# Run the simulation that many times, storing the results of each run to file
```

Figure 33: Running the simulation 100 times

Each run's results are stored in a list (list_to_write), weaving_process_results.csv file opens and the list's information is being appended to the file.

```
list_to_write = [run, count_delayed_orders, count_unfinished_orders, total_lost_profit_from_delays, total_loss]
# Set up list to write to file

with open("weaving_process_results.csv", "a") as f:
    # Write list in file
    writer = csv.writer(f, delimiter=",")

    writer.writerow(list_to_write)
```

Figure 34: Opening weaving_process_results.csv file and appending list_to_write information

Run	Delayed Orders	Unfinished Orders	Lost Profit due to Delays	Total Loss
0	6	0	1267940	1662312
1	4	0	1812220	2101807
2	5	0	1493506	1843516
3	5	0	1714440	2023184
4	6	0	1920506	2268930
5	5	1	1180672	1531457
6	5	1	1482820	1886188

Figure 35: Screenshot of the weaving_process_results.csv file

Python reads the weaving_process_results.csv file as a pandas data frame and calculates mean values for each column.

```
# Reading weaving_process_results file as a dataframe and calculating means
results = pd.read_csv("weaving_process_results.csv")
mean_count_delayed_orders = results["Delayed Orders"].mean()
mean_count_unfinished_orders = results["Unfinished Orders"].mean()
mean_total_lost_profit_from_delays = results["Lost Profit due to Delays"].mean()
mean_total_loss = results["Total Loss"].mean()
```

Figure 36: Calculating means in weaving_process_results.csv file

Finally the means are rounded up (to consider the worst case scenario) and they are printed.

```
# Printing the means and rounding them up
print("Calculated Means for 100 performed Simulations:")
print("Mean of Delayed Orders:", math.ceil(mean_count_delayed_orders))
print("Mean of Unfinished Orders:", math.ceil(mean_count_unfinished_orders))
print("Mean of Total Lost Profit due to Delays:", math.ceil(mean_total_lost_profit_from_delays))
print("Mean of Total Loss:", math.ceil(mean_total_loss))
```

Figure 37: Printing the means

The simulation has run 100 times and the results of each simulation have been produced on the console, while the means are also printed in the end. This way, the results are more

representative and can provide insights. Thus, for the specific amount of resources, orders and days of simulation, there are going to be approximately five delayed orders, 3 unfinished orders, the cost for delayed orders will reach 998.238, while the total loss will be 2.037.609.

```
Calculated Means for 100 performed Simulations:
Mean of Delayed Orders: 5
Mean of Unfinished Orders: 3
Mean of Total Lost Profit due to Delays: 998238
Mean of Total Loss: 2037609
```

Figure 38: Printed results of multiple simulations

3.8 Scenarios

The previously described simulation model can be easily adjusted in condition alterations (addition of looms, changes in events' occurrence probability etc.). Altering the conditions and identifying differences between different versions' results, is also a way to reach simulation model optimization. For instance, if two additional looms L4 and L5 (with the attributes of Loom 2) were to be added in the model and the rest of the variables remained fixed, the results will probably vary from the three looms simulation model results, previously performed.

```python
# For each loom, id, speed and performance index should be defined here
L1 = Loom("11", 200, 0.5)  # First loom characteristics
L2 = Loom("12", 300, 0.5)  # Second Loom characteristics
L3 = Loom("13", 400, 0.8)  # Third Loom characteristics
L4 = Loom("14", 300, 0.5)  # Fourth Loom characteristics
L5 = Loom("15", 300, 0.5)  # Fifth Loom characteristics
# For each loom, speed and performance index should be defined here
looms = [L1, L2, L3, L4, L5]

fixed_energy_cost = 1  # Fixed energy cost of loom while in work, adjustable
setup_cost = 0  # Setup cost of a loom, initially 0

jobs = all_jobs
jobs_total_number = len(original_jobs)+1
env.process(weaver(env, looms[0]))
env.process(weaver(env, looms[1]))
env.process(weaver(env, looms[2]))
env.process(weaver(env, looms[3]))
env.process(weaver(env, looms[4]))
```

Figure 39: Scenario: Additional Looms (L4 & L5)

The results of the simulation after the new additions demonstrate that due to the extra looms, the mean (the model still performs 100 simulations) of delayed orders have been reduced, alongside with the mean of total lost profit due to delays and the mean of total loss. Hence, this addition leads the model closer to the goal which is the cost reduction of the production process.

The user can alter the variables' values in order to observe results' behavior.

```
Calculated Means for 100 performed Simulations:
Mean of Delayed Orders: 4
Mean of Unfinished Orders: 3
Mean of Total Lost Profit due to Delays: 865937
Mean of Total Loss: 1961146
```

Figure 40: Scenario: Printed results of means

CHAPTER 4: CONCLUSION AND FUTURE RESEARCH

4.1 Thesis Summary

This thesis started as an inspiration based on FACTLOG project regarding Cognitive Digital Twins in the manufacturing industry, focusing on Simulation process as the core component of a DT. The theoretical background offered the reader a high-level, conceptual overview of a number of important theoretical ideas that were essential to their comprehension of the remaining chapters of the thesis. Modeling Implementation chapter referred to Python's SimPy simulation module, while introducing the Piacenza's case study where a simulation model had to be developed to portray the textile industry's weaving process. In the same chapter, the model's script, performed on Anaconda's Spyder Environment, is described and analyzed, followed by an overview of the simulation's results.

4.2 Future Research

Regarding future research, the author proposes a few different directions regarding the simulation, taking under consideration further components. The amount of workers, their salaries and working hours could be also taken into account to develop a more complex cost calculating model.

Moreover, by the aid of data envelopment analysis and the use of historical data, an optimized model could be constructed to enhance the production process and its DMUs (decision-making units), while providing robust solutions to any possible scenarios. This model could be expanded to a Cognitive Digital Twin.

REFERENCES

Adrien, B., Maia, E., Feeken, L., Borchers, P., & Praça, I. (2020). *applied sciences A New Concept of Digital Twin Supporting Optimization and Resilience of Factories of the Future*. *May 2017*.

Anu Maria. (1997). Introduction to modeling and simulation. *Proceedings - Winter Simulation Conference*, *1*, 9–16. https://doi.org/10.1201/b17595-2

Babulak, E., & Wang, M. (2010). *Discrete Event Simulation: State of the Art*. *October*. https://doi.org/10.13140/RG.2.1.2068.1767

Banks, Jerry;, Nelson, B. L., Nicol, D. M., & Carson II, J. S. (1999). *DISCRETE-EVENT SYSTEM Part I . Introduction to Discrete-Event System Simulation*. Pearson Education.

Banks, Jerry, & Gibson, R. (1997). Don't simulate when... 10 rules for determining when simulation is not appropriate. *IIE Solutions*.

Bhunia, A. K., Sahoo, L., & Shaikh, A. A. (2019). Introduction to Operations Research. In *Springer Optimization and Its Applications* (Vol. 153, pp. 1–11). https://doi.org/10.1007/978-981-32-9967-2_1

Bi, Z., Xu, L. Da, & Wang, C. (2014). Internet of things for enterprise systems of modern manufacturing. *IEEE Transactions on Industrial Informatics*, *10*(2), 1537–1546. https://doi.org/10.1109/TII.2014.2300338

Boschert, S., & Rosen, R. (2016). Digital Twin-The Simulation Aspect. *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and Their Designers*, 1–259. https://doi.org/10.1007/978-3-319-32156-1

Brenner, B., & Hummel, V. (2017). Digital twin as enabler for an innovative digital shopfloor management system in the ESB Logistics Learning Factory at Reutlingen - University. *Procedia Manufacturing*, *9*, 198–205. https://doi.org/10.1016/j.promfg.2017.04.039

Cichon, T., & Rossmann, J. (2017). Simulation-based user interfaces for digital twins: Pre-, in-,

or post-operational analysis and exploration of virtual testbeds. *31st Annual European Simulation and Modelling Conference 2017, ESM 2017*, 365–372.

Datta, S. P. A. (2017). *Emergence of Digital Twins*. 14–50. http://arxiv.org/abs/1610.06467

David, J. S. (2018). Development of a Digital Twin of a Flexible Manufacturing System For Assisted Learning. *Mater Thesis, February*. https://doi.org/10.13140/RG.2.2.26398.08000

Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., & Persson, M. (2007). Applications of agent based simulation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4442 LNAI*, 15–27. https://doi.org/10.1007/978-3-540-76539-4_2

Erikstad, S. O. (2017). Merging Physics, Big Data Analytics and Simulation for the Next-Generation Digital Twins. *HIPER 2017, High-Performance Marine Vehicles, Zevenwacht, South-Africa, 11-13 September 2017, September*, 139–149. https://www.researchgate.net/profile/Stein_Erikstad/publication/320196420_Merging_Physics_Big_Data_Analytics_and_Simulation_for_the_Next-Generation_Digital_Twins/links/59d4821d4585150177fc44c9/Merging-Physics-Big-Data-Analytics-and-Simulation-for-the-Next

Fei Tao, Ang Liu, Tianliang Hu, A. Y. C. Nee, E. (2015). Digital Twin Driven Smart Design. In *Academic Press* (Vol. 53, Issue 9). https://doi.org/10.1017/CBO9781107415324.004

Fryer, T. (2019). Industry Benefits from the Digital Baby Boom. *Engineering & Technology*, *14*(1), 30–34. https://doi.org/10.1049/et.2019.0100

General Electric Company. (2016). Analytic Engine for the Digital Power Plant. *GE Digital Twin*.

Glaessgen, E. H., & Stargel, D. S. (2012). The digital twin paradigm for future NASA and U.S. Air force vehicles. *Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1–14. https://doi.org/10.2514/6.2012-1818

Hillier, F. S., & Lieberman, G. J. (2002). *Introduction to Operations Research*.

Kassner, L., Mitschang, B., Weber, C., & Jan, K. (2017). *M2DDM – A Maturity Model for Data-Driven Manufacturing*. *63*, 173–178. https://doi.org/10.1016/j.procir.2017.03.309

Kuehn, W. (2018). Digital twins for decision making in complex production and logistic enterprises. *International Journal of Design and Nature and Ecodynamics*, *13*(3), 260–271. https://doi.org/10.2495/DNE-V13-N3-260-271

Lane, M. S., Mansour, A. H., & Harpell, J. L. (1993). Operations Research Techniques: A Longitudinal Update 1973–1988. *Interfaces*, *23*(2), 63–68. https://doi.org/10.1287/inte.23.2.63

Law, A. M., & Kelton, W. D. (1994). Simulation Modeling and Analysis. *Technometrics*, *36*(4), 429. https://doi.org/10.2307/1269971

Malakuti, S., van Schalkwyk, P., Boss, B., Ram Sastry, C., Runkana, V., Lin, S.-W., Rix, S., Green, G., Baechle, K., & Varan Nath, C. (2020). Digital twins for industrial applications. *IIC Journal of Innovation*, 1–19.

Mason, S. J., Hill, R. R., Mönch, L., Rose, O., Jefferson, T., & Fowler, J. W. (2008). *Proceedings of the 2008 Winter Simulation Conference S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.* 91–100.

Morris, W. T. (1967). On the art of modeling. *Journal of Interdisciplinary Mathematics*, *4*(2–3), 165–177. https://doi.org/10.1080/09720502.2001.10700298

Musselman, K. J. (1998). *Guidelines for Success. Handbook of Simulation*. 721–743.

Parrot, A., & Lane, W. (2017). Industry 4.0 and the digital twin. *Deloitte University Press*, 1–17.

Pegden, C. D., & Sturrock, D. T. (1990). Introduction to SIMAN. *Winter Simulation Conference Proceedings*. https://doi.org/10.1080/00224065.1985.11978974

Ríos, J., Carlos, J., Oliva, M., & Mas, F. (2015). *Product Avatar as Digital Counterpart of a Physical Individual Product : Literature Review and Implications in an Aircraft*. https://doi.org/10.3233/978-1-61499-544-9-657

Rožanec, J. M., Jinzhi, L., Košmerlj, A., Kenda, K., Dimitris, K., Jovanoski, V., Rupnik, J.,

Karlovčec, M., & Fortuna, B. (2020). Towards actionable cognitive digital twins for manufacturing. *CEUR Workshop Proceedings*, *2615*, 1–12.

Saracco, R. (2019). Digital Twins: Bridging Physical Space and Cyberspace. *Computer*, *52*(12), 58–64. https://doi.org/10.1109/MC.2019.2942803

Schiehlen, W. (Ed. . (2013). Advanced multibody system dynamics: simulation and software tools. *Springer Science & Business Media*.

Shannon, R. E. (1998). Introduction to the art and science of simulation. *Winter Simulation Conference Proceedings*, *1*, 7–14. https://doi.org/10.1109/wsc.1998.744892

Tao, F., Cheng, J., Qi, Q., Zhang, M., Zhang, H., & Sui, F. (2018). Digital twin-driven product design, manufacturing and service with big data. *International Journal of Advanced Manufacturing Technology*, *94*(9–12), 3563–3576. https://doi.org/10.1007/s00170-017-0233-1

Tao, F., Liu, W., Zhang, M., Hu, T., Qi, Q., Zhang, H., Sui, F., Wang, T., Xu, H., Huang, Z., Ma, X., Zhang, L., Cheng, J., Yao, N., Yi, W., Zhu, K., Zhang, X., Meng, F., Jin, X., … Luo, Y. (2019). Five-dimension digital twin model and its ten applications. *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*. https://doi.org/10.13196/j.cims.2019.01.001

Tao, F., Sui, F., Liu, A., Qi, Q., Zhang, M., Song, B., Guo, Z., Lu, S. C. Y., & Nee, A. Y. C. (2019). Digital twin-driven product design framework. *International Journal of Production Research*, *57*(12), 3935–3953. https://doi.org/10.1080/00207543.2018.1443229

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

Tao, F., Zhang, M., Liu, Y., & Nee, A. Y. C. (2018). CIRP Annals - Manufacturing Technology Digital twin driven prognostics and health management for complex equipment. *CIRP Annals - Manufacturing Technology*, 7–10. https://doi.org/10.1016/j.cirp.2018.04.055

Wang, J., Ye, L., Gao, R. X., Li, C., & Zhang, L. (2019). Digital Twin for rotating machinery

fault diagnosis in smart manufacturing. *International Journal of Production Research*, *57*(12), 3920–3934. https://doi.org/10.1080/00207543.2018.1552032

Willaert, B., Van Brussel, H., & Niemeyer, G. (2012). *Stability of model-mediated teleoperation: Discussion and experiments. In International conference on human haptic sensing and touch enabled computer applications*.

APPENDIX

The script:

```
# Imorting the necessary modules
import csv
import math
from datetime import datetime, timedelta
from random import random, randint
import pandas as pd
import simpy as sp

# Defining class Order its necessary attributes (entity)
class Order:

    def __init__(self, identifier, typ, priority, quantity, ca_code, cc_code,
product_code, release_date, due_date,
                 complexity):
        self._identifier = identifier
        self._typ = typ
        self._priority = priority
        self._quantity = quantity
        self._ca_code = ca_code
        self._cc_code = cc_code
        self._product_code = product_code
        self.release_date = release_date
        self._due_date = due_date
        self._complexity = complexity

    def get_identifier(self):
        return self._identifier

    def get_type(self):
        return self._typ

    def get_priority(self):
        return self._priority

    def get_quantity(self):
        return self._quantity

    def get_ca_code(self):
        return self._ca_code
```

```
    def get_cc_code(self):
        return self._cc_code

    def get_product_code(self):
        return self._product_code

    def get_releasedate(self):
        return self.release_date

    def get_duedate(self):
        return self._due_date

    def get_complexity(self):
        return self._complexity

# Defining class Loom its associated properties (resources)
class Loom:

    def __init__(self, loom_id, loom_speed, performance_index):
        self.loom_id = loom_id
        self.loom_speed = loom_speed
        self.performance_index = performance_index
        self.prev_job = None
        self.current_job = None

    def set_prev_job(self, job):
        self.prev_job = job

    def set_current_job(self, job):
        self.current_job = job

    def get_loom_id(self):
        return self.loom_id

    def get_prev_job(self):
        return self.prev_job

    def get_current_job(self):
        return self.current_job

    def get_loom_speed(self):
        return self.loom_speed

    def get_performance_index(self):
```

```
        return self.performance_index

    def get_loom_original_speed(self):
        return self.loom_speed * self.performance_index

# Defining a list to store all jobs
all_jobs = list()
# Opening the excel file of jobs/orders
jobs = pd.read_excel("job_details.xlsx")
jobs.sort_values(by=["priority"], inplace=True)
# Iterating over jobs in excel file
for j in jobs.iterrows():
        # Getting current iterated job
    job_details = j[1]
    id, typ, prior = job_details["identifier"], job_details["type"],
job_details["priority"]
    cap, ca, cc = job_details["quantity"], job_details["ca_code"],
job_details["cc_code"]
    pc, rd, dd = job_details["product_code"], job_details["release_date"],
job_details["due_date"]
    complex = job_details["complexity"]

    # Defining an order object with job details
    order = Order(id, typ, prior, cap, ca, cc, pc, rd, dd, complex)

    # Appending the list of all jobs with order
    all_jobs.append(order)

# This function generates a new order
def random_job(id):
    if int(randint(0, 1)) == 0:
        typ = 'regular'
    else:
        typ = 'sample'

    prior = randint(0, 10)
    if typ == 'sample':
        cap = randint(1, 25)
    else:
        cap = randint(25, 100000)

    ca = 'ca_' + str(randint(0, 10))
    cc = 'cc_' + str(randint(0, 10))
    pc = randint(1000, 9999)
    rd_d = randint(1, 30)
```

```
    rd_m = randint(7, 8)
    rd = datetime(2022, rd_m, rd_d)

    dd_d = randint(1, 30)
    dd_m = randint(7, 8)
    dd = datetime(2022, dd_m, dd_d)

    while rd > dd:
        dd_d = randint(1, 30)
        dd_m = randint(7, 8)
        dd = datetime(2022, dd_m, dd_d)

    complex = randint(1, 10)
    return Order('id'+str(id), typ, prior, cap, ca, cc, pc, rd, dd, complex)

# This function will calculate setup_time of loom based on ca and cc codes of
current and previous processed job
def setup_time(ca1, cc1, ca2, cc2):

    if ca1 == ca2 and cc1 == cc2:
        return 0
    elif ca1 == ca2 and cc1 != cc2:
        return 2
    else:
        return 8

# This function will calculate the processing time of a job based on its
complexity
def processing_time(job, loom):

    comp = job.get_complexity()
    time = job.get_quantity() * comp / loom.get_loom_original_speed()
    return time

#  This function will return True if loom is damaged (event of simulation)
def damaged_loom():

    chance = random()
    if chance > 0.99:
        print("Found a Damaged Loom!")
        return True
    return False

# This function checks if there is a broken yarn case (event of simulation)
def broken_yarn():
```

```
    chance = random()
    if chance > 0.85:
        print("Found Broken Yarn!")
        return True
    return False

# This function checks if there is a new order entering the production (event
of simulation)
def new_job():

    chance = random()
    if chance > 0.7:
        print("New Job!")
        return True
    return False


def jobs_available(jobs, current_day):
    a_jobs = []
    for job in jobs:
        y, m, d =
take_datetime_and_make_it_days_month_year(job.get_releasedate())
        s_releasedate = datetime(y, m, d)
        if s_releasedate <= current_day:
            a_jobs.append(job)

    return a_jobs


def selected_job(jobs, prev_job):
    if prev_job is None:
        return jobs[0]

    min_time = 10
    s_job = jobs[0]
    for job in jobs:
        if setup_time(job.get_ca_code(), job.get_cc_code(),
prev_job.get_ca_code(), prev_job.get_cc_code()) < min_time:
            min_time = setup_time(job.get_ca_code(), job.get_cc_code(),
prev_job.get_ca_code(), prev_job.get_cc_code())
            s_job = job

    return s_job, min_time
```

```
def new_jobs(jobs, s_job):
    n_jobs = []
    for job in jobs:
        if job != s_job:
            n_jobs.append(job)

    return n_jobs


def take_datetime_and_make_it_days_month_year(str_date):
    dt = str_date
    return dt.year, dt.month, dt.day


def delete_random_jobs(jobs, original_jobs_total_number):
    n_jobs = []
    for job in jobs:
        num_str = ""
        for m in job.get_identifier():
            if m.isdigit():
                num_str = num_str + m

        if int(num_str) <= original_jobs_total_number:
            n_jobs.append(job)

    return n_jobs


active_time = 0


def weaver(environment, loom):
    global start_time, active_time, jobs, setup_cost, ending_times,
jobs_total_number, all_jobs
    ending_times = dict()
    while len(jobs) > 0:

        # print("loom", loom.get_loom_speed())
        # print("all_jobs len: ",len(jobs))
        # for job in jobs:
        #     print("all_jobs: ", job.get_identifier())

        simulation_start_time = environment.now
        days, hours = simulation_start_time // 8, simulation_start_time % 8
```

```
        current_date = start_time + timedelta(days=days, hours=hours)
        a_jobs = jobs_available(jobs, current_date)

        # print("avl_jobs len: ", len(a_jobs))
        # for job in a_jobs:
        #     print("all_jobs: ", job.get_identifier())

        if len(a_jobs) > 0:
            if new_job():
                r_job = random_job(jobs_total_number)
                jobs_total_number += 1
                jobs.append(r_job)
                # for job in a_jobs:
                #     print("all_jobs: ", job.get_identifier(),
job.get_priority())
                jobs.sort(key=lambda x: x._priority, reverse=False)
                # for job in a_jobs:
                #     print("all_jobs: ", job.get_identifier(),
job.get_priority())
                all_jobs.append(random_job(jobs_total_number))

            if loom.get_prev_job() is None:
                s_job, setup_time = a_jobs[0], 0
            else:
                s_job, setup_time = selected_job(a_jobs, loom.get_prev_job())

            # Print the starting date of a job and the loom it has been
assigned to
            print("Started working on", s_job.get_identifier(), "at",
current_date.strftime("%m/%d/%Y"), "at Loom",
                  loom.get_loom_id())

            loom.set_current_job(s_job)
            jobs = new_jobs(jobs, s_job)
            setup_cost += s_job.get_quantity() * setup_time
            yield env.timeout(setup_time)
            yield env.timeout(processing_time(s_job, loom))
            active_time += processing_time(s_job, loom)
            # If a loom is damaged, repair it, which takes 15 minutes
            if damaged_loom():
                yield env.timeout(0.25)

            # If there is broken yarn found, repair it, which takes 15
minutes
            if broken_yarn():
```

```
                yield env.timeout(0.25)


            simulation_start_time = environment.now
            days, hours = simulation_start_time // 8, simulation_start_time %
8
            ending_time = start_time + timedelta(days=days, hours=hours)
            # Printing the ending date of a job
            print(s_job.get_identifier(), "ended at",
ending_time.strftime("%m/%d/%Y"), "at Loom",
                  loom.get_loom_id())
            loom.set_prev_job(s_job)
            ending_times[s_job.get_identifier()] = environment.now

        yield env.timeout(1)


number_of_simulation_runs = 100
# Defing the number of simulations that will run, adjustable

with open("weaving_process_results.csv", "w") as f:
    writer = csv.writer(f, delimiter=",")
    writer.writerow(["Run", "Delayed Orders", "Unfinished Orders", "Lost
Profit due to Delays", "Total Loss"])
# Creating a file to store the results of each run and write the column
headers

original_jobs = all_jobs
original_jobs_total_number = len(all_jobs)
for run in range(number_of_simulation_runs):
    # Run the simulation that many times, storing the results of each run to
file
    all_jobs = delete_random_jobs(all_jobs, original_jobs_total_number)
    global ending_times
    env = sp.Environment()
    start_time = datetime(2022, 7, 7)  # The start date of simulation,
adjustable
    # We can define more looms, depending on the conditions for each
simulation
    # For each loom, id, speed and performance index should be defined here
    L1 = Loom("l1", 200, 0.5)  # First loom characteristics
    L2 = Loom("l2", 300, 0.5)  # Second Loom characteristics
    L3 = Loom("l3", 400, 0.8)  # Third Loom characteristics
    L4 = Loom("l4", 300, 0.5)  # Fourth Loom characteristics
    L5 = Loom("l5", 300, 0.5)  # Fifth Loom characteristics
```

```
    # For each loom, speed and performance index should be defined here
    looms = [L1, L2, L3, L4, L5]

    fixed_energy_cost = 1  # Fixed energy cost of loom while in work,
adjustable
    setup_cost = 0  # Setup cost of a loom, initially 0

    jobs = all_jobs
    jobs_total_number = len(original_jobs)+1
    env.process(weaver(env, looms[0]))
    env.process(weaver(env, looms[1]))
    env.process(weaver(env, looms[2]))
    env.process(weaver(env, looms[3]))
    env.process(weaver(env, looms[4]))

    # Number of days the simulation is running, adjustable
    days_to_simulate = 60
    # Running the environment for number of days based on 8 hours of work/day
    env.run(until=days_to_simulate * 8)

    # Defining a dictionary object to store the penalty loss of orders
    lost_profit_from_delays = dict()
    print()
    # Calculating Profit Loss
    count_delayed_orders = 0
    count_unfinished_orders = 0
    # Iterating over jobs to calculate penalties for late orders
    for j in all_jobs:
        # print(j.get_identifier())
        try:
            id = j.get_identifier()
            delivered_on = ending_times[id]
            days, hours = delivered_on // 8, delivered_on % 8
            delivered_on = start_time + timedelta(days=days, hours=hours)
            delay = delivered_on - j.get_duedate()
            if delay.days > 0:
                fine = delay.days * j.get_quantity()
                lost_profit_from_delays[id] = fine
                count_delayed_orders = count_delayed_orders + 1
        except:
            print("Unfinished Order:", j.get_identifier())
            # Some jobs might remain unfinished at the end of simulation,
print them
            count_unfinished_orders = count_unfinished_orders + 1
```

```
    print("Amount of unfinished orders:", count_unfinished_orders)
    print()
    print("Total Fine per delayed job: ")
    # Iterating over lost profit and printing
    for j, f in lost_profit_from_delays.items():
        print(j, f)
    total_lost_profit_from_delays = sum(lost_profit_from_delays.values())
    print("Total Lost Profit at the end due to delays:",
total_lost_profit_from_delays)
    fixed_energy_cost = fixed_energy_cost * active_time
    total_loss = int(total_lost_profit_from_delays + fixed_energy_cost +
setup_cost)
    # Printing the total fine taking under consideration setup and loom
active time fee
    print("Total Loss after including setup fees and active time fees:",
round(total_loss))


    list_to_write = [run, count_delayed_orders, count_unfinished_orders,
total_lost_profit_from_delays, total_loss]
    # Set up list to write to file

    with open("weaving_process_results.csv", "a") as f:
        # Write list in file
        writer = csv.writer(f, delimiter=",")

        writer.writerow(list_to_write)

# Reading weaving_process_results file as a dataframe and calculating means
results = pd.read_csv("weaving_process_results.csv")
mean_count_delayed_orders = results["Delayed Orders"].mean()
mean_count_unfinished_orders = results["Unfinished Orders"].mean()
mean_total_lost_profit_from_delays = results["Lost Profit due to
Delays"].mean()
mean_total_loss = results["Total Loss"].mean()

# Printing the means and rounding them up
print("Calculated Means for 100 performed Simulations:")
print("Mean of Delayed Orders:", math.ceil(mean_count_delayed_orders))
print("Mean of Unfinished Orders:", math.ceil(mean_count_unfinished_orders))
print("Mean of Total Lost Profit due to Delays:",
math.ceil(mean_total_lost_profit_from_delays))
print("Mean of Total Loss:", math.ceil(mean_total_loss))
```