Gonçalo Alexandre Roque Rolo

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Simulation-based Digital Twin for Distributed Manufacturing Control Systems

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: André Dionísio Bettencourt da Silva Parreira Rocha, Professor Convidado, Universidade Nova de Lisboa

Júri

Presidente:	Prof.ª Doutora Anabela Monteiro Gonçalves Pronto
Arguente:	Prof. Doutor Ricardo Alexandre Fernandes da Silva Peres
Vogal:	Prof. Doutor André Bettencourt da Silva Parreira Rocha

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Novembro, 2020

Simulation-based Digital Twin for Distributed Manufacturing Control Systems

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

"Deus quer, o homem sonha, a obra nasce."
Fernando Pessoa

Acknowledgements

This project bears such importance to my future and professional career that some acknowledgements are due.

Firstly, I would like to express my gratitude for Dr. André Rocha's pieces of advice and for his proficient guidance which brought this project to a successful end. I would also like to thank João Tripa, Professional Engineer, for his availability and for simplifying the integration between my project and his.

Secondly, here goes a well-deserved acknowledgement to my friends, João Meneses, Leandro Filipe and, lately, Filipa Gouveia and Marta Gameiro, who took on this journey alongside me, making it a lot easier and enjoyable.

At last but not least, to my parents without whose help nothing I have ever achieved had been possible.

Overall, to everyone who contributed, in someway, to my personal success. Thank you all.

Abstract

The fast pace at which industrial revolutions have been taking place depicts the importance of production systems for human beings.

The increasing need to improve products' quality and production lines' efficiency without disregarding the human factor has compelled engineers to come up with innovative solutions. Consequently, the ongoing Industrial Revolution is leading to the emergence of new concepts, amongst which Digital Twins (DTs) stand out.

Given their early stage, the already existing implementations are far from standardised, meaning that each practical case has to be analysed on its own and solutions are often created from scratch. Despite their current lack of modularity, the powerful predictive and monitoring capabilities featured in such implementations make DTs a rather interesting subject.

This work suggests an architecture which allows the integration between a previously programmed manufacturing unit simulator and its DT, implemented and calibrated within the scope of this project. In spite of the physical asset itself being represented by a simulator, the suggested solution is equally applicable to a real-life scenario with a full-size conveyor network.

Moreover, a thorough validation process is carried out, allowing the broadening of the knowledge on the physical system's performance traits. Nevertheless, its somewhat unforeseeable behaviour hinders the modelling, thus making room for future work on how to make the DT's predictions more accurate.

Keywords: Cyber-Physical Production System, Digital Twin, Integration, Key Performance Indicator, Simulation

Resumo

A celeridade com que se têm verificado revoluções industriais retratam a importância dos sistemas de produção para os seres humanos.

A crescente necessidade de melhorar a qualidade dos produtos e a eficiência das linhas de produção sem ignorar o fator humano tem obrigado os engenheiros a pensar em soluções inovadoras. Consequentemente, a Revolução Industrial que se encontra em curso está a levar ao aparecimento de novos conceitos, entre os quais se destacam os Gémeos Digitais.

Dada a sua fase embrionária, as implementações já existentes estão longe de padronizadas, pelo que cada caso prático tem de ser analisado por si só e as soluções são muitas vezes criadas de raiz. Apesar da sua atual falta de modularidade, as poderosas capacidades de previsão e monitorização apresentadas por tais implementações tornam os Gémeos Digitais num tópico bastante interessante.

Este trabalho sugere uma arquitetura que permite a integração entre um simulador de uma unidade de manufatura, previamente programado, e o seu Gémeo Digital, implementado e calibrado no âmbito deste projeto. Apesar de o próprio sistema físico ser representado por um simulador, a solução sugerida é igualmente aplicável a um cenário com uma rede de transporte à escala real.

Adicionalmente, é descrito um processo de validação minucioso, que permite alargar o conhecimento sobre o desempenho geral do sistema físico. Não obstante, o seu comportamento algo imprevisível impõe obstáculos à modelação, algo que poderá vir a ser melhorado, futuramente, de modo a que as previsões feitas pelo Gémeo Digital sejam mais exatas.

Palavras-chave: Gémeo Digital, Indicador-Chave de Desempenho, Integração, Simulação, Sistema de Produção Ciber-Físico

Contents

List of Figures

List of Tables

Listings

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
AR	Augmented Reality.
BIW	Body In White.
CAD	Computer-Aided Design.
CPPS	Cyber-Physical Production System.
CPS	Cyber-Physical System.
DA	Deployment Agent.
DT	Digital Twin.
GUI	Graphical User Interface.
HCPS	Human-Cyber-Physical System.
HMI	Human-Machine Interface.
HPS	Human-Physical System.
I4.0	Industry 4.0.
ICT	Information and Communications Technology.
IL	Integration Layer.
IoT	Internet of Things.
JADE	Java Agent DEvelopment Framework.
JSON	JavaScript Object Notation.
KPI	Key Performance Indicator.
KRI	Key Result Indicator.

MAS	Multi-Agent System.
NASA	National Aeronautics and Space Administration.
NSF	National Science Foundation.
OPC-UA	Open Platform Communications Unified Architecture.
OWL	Ontology Web Language.
PA	Product Agent.
PI	Performance Indicator.
PL	Physical Layer.
PLC	Programmable Logic Controller.
PLM	Product Lifecycle Management.
RA	Resource Agent.
RAMI4.0	Reference Architecture Model for Industrie 4.0.
RDF	Resource Description Framework.
RICS	Robotics & Industrial Complex Systems.
SL	Simulation Layer.
SoS	System of Systems.
TA	Transport Agent.
URL	Uniform Resource Locator.
WSN	Wireless Sensor Network.

Chapter 1

Introduction

This chapter mainly intends to introduce the emerging problem whose solution was sought. In addition, a general preview of the developed work is provided, alongside the main contributions to the research community, derived from this project. Finally, the document's structure is unveiled.

1.1 Problem Presentation

Manufacture has always played a pivotal role on human beings' lives. Since the very first handcrafted object for survival purposes, humans have fancied the process of creating tools which facilitate their tasks, from the materials made available by the surrounding environment.

The unrelenting need to create new products has become exponentially evident with three industrial revolutions taking place in less than two centuries and a fourth one already in progress. Regarding the latter, a wide range of technologies await implementation with the main goals of increasing the quality of manufactured products, enhancing the production systems' efficiency and, as always, improving employees' working conditions.

One example of such technologies are Digital Twins (DTs), which hold applications in a wide range of fields and whose features undoubtedly improve the systems' performance. However, as with any technological innovation, the lack of standards makes each implementation a completely new challenge [1].

As a result, an innovative architecture for a simulation-based DT is suggested. However, unlike the majority of the existing solutions, its main focus is not being continuously synchronised with its physical counterpart. Instead, the model endeavours to accurately predict the system's outcome, by mirroring its behaviour at higher speeds.

1.2 Work Overview

This project's aim was to provide an unprecedented point of view on how to use DTs to help users making decisions. For that, several stages had to be completed.

Firstly, a three-layered architecture was designed. The bottom layer contains the already programmed physical system, which underwent a series of alterations that enabled both the communication between the physical and digital worlds and the calculation of the real Key Performance Indicators (KPIs). The middle layer is constituted by a database, where the simulation-relevant information is stored, and an Application Programming Interface (API), that directly interacts with it. The top layer includes the simulation model, which relies on the database's content to predict the system's performance. The second phase consisted of actually developing each of the foregoing elements. In the end, the developed model was calibrated and the predictions were compared with the real indicators so as to assess the solution's performance.

1.3 Main Contributions

This project's contributions are not confined but mostly related to the suggested architecture, since a more specific framework than the ones currently available is put forward.

On the one hand, the idea of representing a system's state using a simple file format is widely applicable. Furthermore, given that a similar API will be suitable for interacting with such files in most cases, an identical integration layer can be often adopted.

Despite the fact that the simulation model is only applicable to this specific system, AnyLogic was considered to be a versatile tool for developing and running digital models. This is due to its ability to read external files and export the developed models as stand-alone applications. Hence, the simulation layer is not expected to need significant alterations on future implementations either.

Ultimately, from a practical standpoint, the intense processing activity of the physical system is highlighted as one of the likely sources of error. Future approaches will, thus, be able to take this into consideration if they are to provide more exact results.

1.4 Thesis Outline

The remaining document is organised as follows. Chapter 2 introduces the core concepts for this work. Chapter 3 details the proposed architecture for tackling the aforementioned problem. Chapter 4 intends to clarify which technologies and algorithms were found more suitable for the implementation step. Chapter 5 describes the calibration method used for the digital model, presents the obtained results and puts forward explanations for some of the discrepancies. Finally, chapter 6 starts by assessing the coverage degree of the detected gaps and, then, leaves some guidelines on how to improve the developed solution.

Chapter 2

Literature Review

The current chapter serves the purpose of introducing some theoretical concepts which are essential for the comprehension of this project. Aside from that, the most evident unsolved issues related to the topic are underlined as a way of showing which aspects researchers shall place their efforts on, in the upcoming years.

2.1 Industry 4.0

The First Industrial Revolution dates back to the second half of the 18th century. Leveraged by the creation of the steam engine, which produced energy out of steam pressure, this revolution introduced the mechanisation of production processes. Less than a century later, the Second Industrial Revolution occurred. The emergence of electrical power allowed the implementation of production lines and, consequently, led to the mass production paradigm. Nearly a century after that, developments in the fields of electronics and computing promoted the automation of production environments. No sooner had the Third Industrial Revolution taken place than the concept of Fourth Industrial Revolution emerged.

The term, which used to be connected with breakthroughs in the area of nanotechnologies [2, 3], was designated Industry 4.0 (I4.0) for the first time in 2011, on the occasion of the Hannover Fair [4]. Alongside the German government, which was pioneer embracing the paradigm with the aim of innovating manufacturing systems, other countries came up with similar strategical plans, originating equivalent terms such as Smart Manufacturing, Advanced Manufacturing and Production 4.0, amongst many others [5, 6].

Given its considerably extensive nature, the term's definitions vary depending on the authors. However, some concepts are common to most of them, such as automation systems, interlink between the physical and the digital worlds and Internet-based technologies [5]: in [7, 8], I4.0 is seen as a meeting point of several technologies, such as big data, smart sensors and the Internet of Things (IoT), whereas, in [9], the author defines I4.0 as *"a set of technologies based on digitisation and interconnection of all production units present within an economic system"*.

The main goal of I4.0 is to improve the efficiency and profitability of manufacturing systems [10], integrating human beings, machines and data [4] and making use of the wide scope of devices which have been endowed with connectivity features during the last years. Hence, the exponential growth of Information and Communications Technologies (ICTs) stands out as one of the factors which have helped shortening the gap between the physical and the digital worlds [11]. Consequently, factories are progressively turning into smart factories, whose main features are [12]:

- Capacity to accept on-the-fly changes to the ongoing manufacturing process.
- Remote access to the information obtained by monitoring every single resource.
- Self-recognition of their capabilities and autonomous organisation of production tasks.

So as to materialise the transition to I4.0, some recent technological breakthroughs, the so-called enabling technologies, must be effectively integrated. Despite there being more than one thousand [13], most authors agree that the following nine technological advancements are of the utmost importance for the implementation of I4.0:

- Additive manufacturing - layer-by-layer manufacturing technique that allows the production of complex objects at reduced costs [14].
- Advanced manufacturing - consists of developing enhanced manufacturing systems provided with autonomous robots, capable of co-working with humans [5, 9].
- Artificial Intelligence (AI) - intends to give robots the ability to adapt to unexpected situations and decide what to do in order to complete the required tasks [15].
- Augmented Reality (AR) - refers to the extension of the real world by granting access to elements from the virtual world [16].
- Big data - methods to effectively deal with the wide amount of data obtained through a multitude of sources are required, in order to allow efficient access to information [14, 16].
- Cloud computing - describes the ability to compute data which is stored in an external environment, making use of the IoT [17], i.e., the fact that devices are continuously interconnected.

- Cyber security - through the IoT, physical environments become remotely accessible, making it vital to guarantee the integrity and confidentiality of the involved data [14].
- IoT - allows the communication between devices regardless of their physical location, thus providing immediate access to any information they hold [14].
- Simulation - consists of creating virtual models to accurately represent physical objects [5]. The benefits of such technology lay on the fact that, before taking risks with actual equipment, simulations can be conducted in order to predict possible failures.

Similarly to any technological innovation, I4.0's implementation entails considerable efforts in spite of its indisputable benefits. Some of the most relevant factors that are preventing companies from adopting I4.0 include considerable investments at an initial stage, uncertainty about the profitability of the transition, risk of cyber-attacks, seeming increase in the unemployment rate, lack of skill from the employees and unwillingness to change due to fear of personal data leakage [4].

From the aforementioned drawbacks, the risk of cyber-attacks definitely stands out as one of the most challenging problems. The fact that, nowadays, communication mostly takes place via wireless interfaces makes data more vulnerable to external threats. Such attacks may compromise three different data security requirements [18]:

- Confidentiality - data shall not become available to unauthorised people. Violating this requirement might mean loss of confidence from customers [19].
- Integrity - data must be protected from improper changes, so that it always represents reliable information. Not fulfilling this requirement may lead to reduced product's quality, due to incorrect settings on the machines, for instance [19].
- Availability - data must be always accessible in order to avoid jeopardising systems' proper operation. Should this requirement be compromised, the manufacturing system stops working, since it is prevented from accessing the needed settings [19].

That being said, cyber-security is one of the first challenges to be tackled, given that fulfilling all data security requirements leads to decreasing machine downtime and, consequently, improved productivity, therefore maximizing I4.0's performance [19].

In order to standardise future implementations of I4.0's scenarios, BITCOM, VDMA and ZVEI suggested the Reference Architecture Model for Industrie 4.0 (RAMI4.0) [20], represented in figure 2.1.

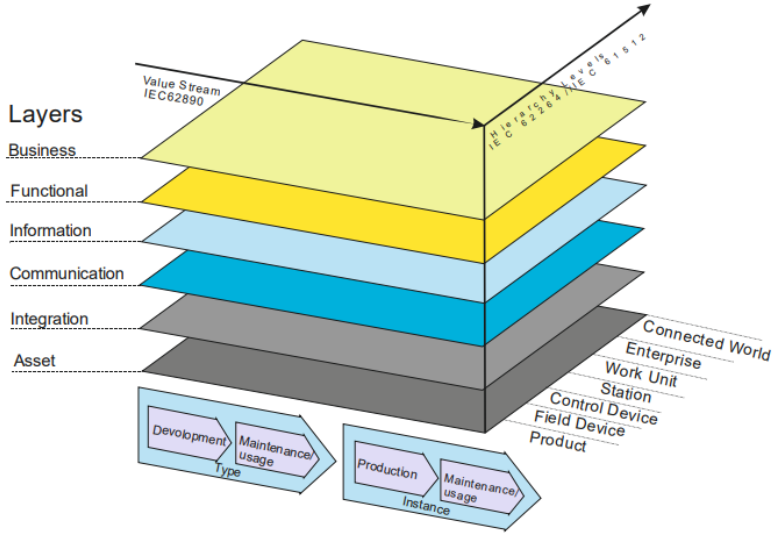


Figure 2.1: Reference Architecture Model for Industrie 4.0 [21]

This three-dimensional model is constituted by two horizontal axis, and six layers along the vertical axis. The left-hand horizontal axis refers to the value stream during products' life cycle, from the development stage up until their disposal. The right-hand horizontal axis, on the other hand, represents the hierarchical levels, starting from the simplest product up to the most macro level, characterised by a fully connected world. The layers and respective functions are listed below [20]:

- Asset layer - symbolizes the real world, thus including its physical components. These can be objects but also include human beings, i.e., workers.
- Integration layer - supports the interaction between the low-level layer, providing information from physical entities in a meaningful way for the digital world.
- Communication layer - is responsible for standardising data formats, allowing successful communications between components.
- Information layer - assures data is constantly available, maintaining its integrity and high quality standards.
- Functional layer - describes components' behaviour according to a set of rules.
- Business layer - ensures integration between different processes at the business level.

Ultimately, it is worth mentioning that, despite the great focus on the manufacturing sector, the Fourth Industrial Revolution holds a range of potential benefits in other areas, e.g., the services sector [22], where the RAMI4.0 is equally applicable.

2.2 Cyber-Physical Systems

The term Cyber-Physical System (CPS) is tightly related to the I4.0 paradigm [23]. The implementation of such systems is equally dependent on technological tendencies such as IoT and big data. Moreover, CPSs are of paramount importance to the successful development of areas like smart manufacturing, smart cities, etc. [24].

The concept of CPS itself was coined by Helen Gill at a workshop on CPSs, hosted by the National Science Foundation (NSF) in 2006 [24], and its emergence was due to the need of a new representative form for systems with increased levels of complexity [25]. Despite the fact that research in the area is still mainly theoretical [26], not only did the term become worldwide accepted but steps have also been taken towards its implementation. Such fact is backed up by a more recent study from Bitkom dated April 2018, which found out that 25% of the machines were already endowed with connectivity capabilities [27].

A widely accepted notion of the term is put forward in [28]: "*CPSs are systems of collaborating computational entities which are in intensive connection with the surrounding physical world and its on-going processes, providing and using, at the same time, data-accessing and data-processing services available on the internet*".

As suggested by its name, a CPS comprises two main elements [26], both illustrated in figure 2.2:

- Physical part - includes all tangible entities, e.g., raw materials, machinery and workers [17], and is responsible for collecting data from sensors and carrying out tasks, through actuators, at the shop-floor level.
- Digital or cyber side - encompasses smart data management methods, services and computing skills [29], in order to ensure proper decision-making, which is then transmitted to the physical entity.

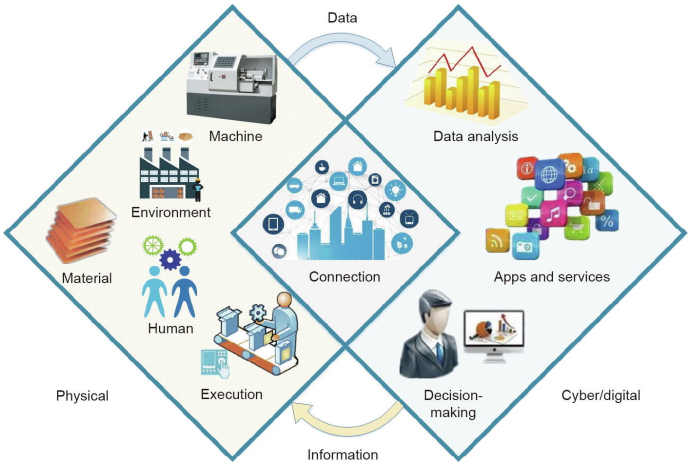


Figure 2.2: Elements of a Cyber-Physical System [26]

Given the indisputable interdependence between the two previously mentioned elements, a third component is often considered. It refers to the interconnection between the physical and the digital worlds and highlights the capacity of both sides to influence one another [30].

Regarding the structure of a CPS, there is a wide acceptance of a 5-layer representation, also known as the 5C architecture, presented in figure 2.3. This model adopts a triangular shape that displays, by increasing levels of complexity from bottom to top, the following features [31]:

- Smart Connection level - includes the sensors' network responsible for the data acquisition and refers to the system's ability to gather data from its surroundings.
- Data-to-Information Conversion level - is responsible for transforming the acquired data into contextualised information.
- Cyber level - makes use of the information conveyed by the previous layer to implement consistent and viable twin models.
- Cognition level - ensures that the knowledge obtained from the previously applied models is properly presented to users so that the right interventions can be made.
- Configuration level - is related to the systems' ability to adapt to changes in the environment, according to the feedback from the digital space.

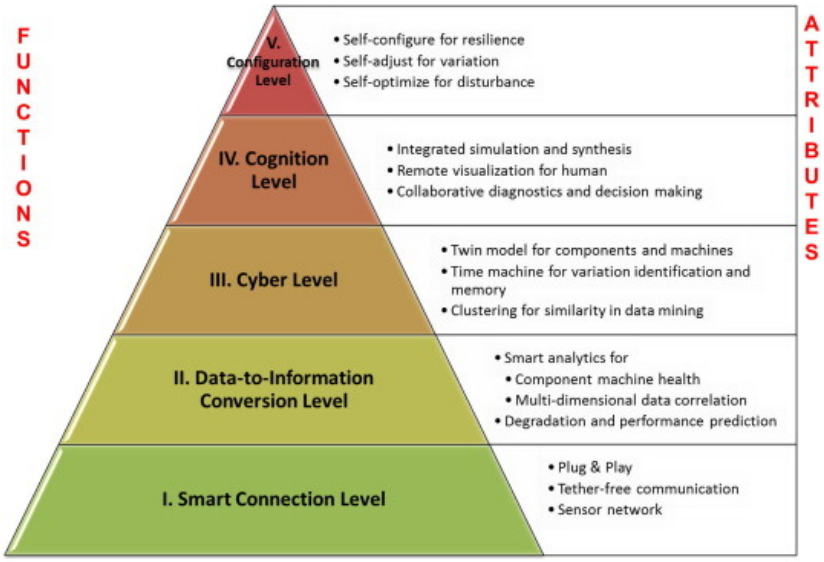


Figure 2.3: 5C architecture of a Cyber-Physical System [31]

However, seldom can this architecture be integrally found in actual systems. As a matter of fact, this model is most commonly used to assess the level of independence of systems rather than guide their implementation process [32].

The CPSs' modularity allows the recombination of several systems so that more sophisticated ones can be developed [33]. Regarding their hierarchical structure, three levels can be distinguished [34]:

- Unit level - consists of each individual component of a system, e.g., a machine.
- System level - encompasses the integration of a multitude of simpler CPSs, i.e., represents a production module formed by several unit-level components.
- System of Systems (SoS) level - refers to the interconnection between several system-level CPSs, i.e., between different companies [35].

Among the benefits of implementing CPSs, the adaptability to the environment, the improved automation levels and the learning capacity of the systems undoubtedly stand out as unprecedented features [36].

Nevertheless, the emergence of CPSs raised a number of ethical issues. Due to the fear of losing relevance in the value chain and the unwillingness to change their *modus operandi*, employees are reluctant to accept the implementation of such concept.

Furthermore, the CPSs' operation relies on Wireless Sensor Networks (WSNs), which may lead to a number of security breaches [24]. Taking into account that CPSs are real-time systems, demanding requirements are set and, consequently, common security methods are inadequate [37]. Hence, modelling security breaches and designing architectures capable of dealing with cyber-threats [38] are of extreme importance, if the information's security requirements from section 2.1 are to be fulfilled.

2.2.1 Human-Cyber-Physical Systems

Implementing a CPS often implies the existence of human beings who either supervise or directly interact with the system [39]. The inclusion of human beings in a CPS originates Human-Cyber-Physical Systems (HCPSs).

The first manufacturing systems, also known as Human-Physical Systems (HPSs), were introduced by the First Industrial Revolution and remained unchanged during the Second Industrial Revolution. They encompassed two main entities: humans, who played the role of masters, and machines, which executed the commanded tasks [40].

The Third Industrial Revolution was characterised by the fast development of ICTs which, at the level of HCPSs, led to the insertion of a new entity between humans and the machines: the cyber system, responsible for making decisions and carrying out control tasks, which were previously assigned to human beings. These HCPSs, referred to as HCPS1.0 in [40], triggered the shift of some mental endeavour to the cyber entity [40], even though humans still behaved as the masters.

With the emergence of the Internet and its application to manufacturing systems, HCPSs were upgraded to HCPS1.5, whose main feature was the appending of Internet technologies to the cyber entity as a way of integrating all three components of such systems [40].

Finally, the most recent version of HCPSs, HCPS2.0, came to light, making use of AI in order to provide the cyber world with knowledge-based abilities, which used to be exclusive to human beings, as represented in figure 2.4. As a result, workers tend to be freed from weary affairs and, consequently, allowed to focus on tasks which they are the only entity capable of executing, such as creative work.

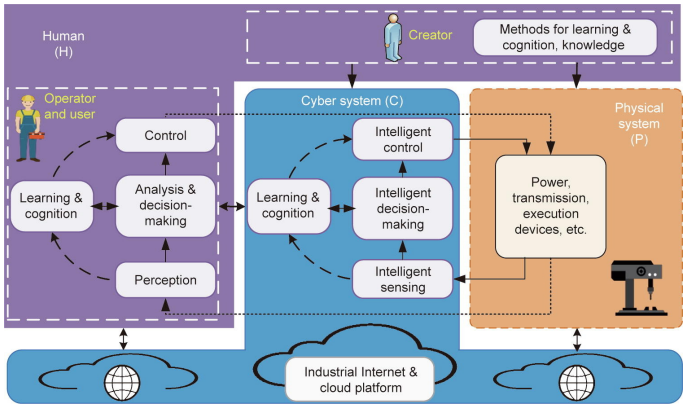


Figure 2.4: Human-Cyber-Physical Systems 2.0 [40]

The evolution of manufacturing systems, from the simplest up to the most recent version of HCPSs, is summed up in figure 2.5.

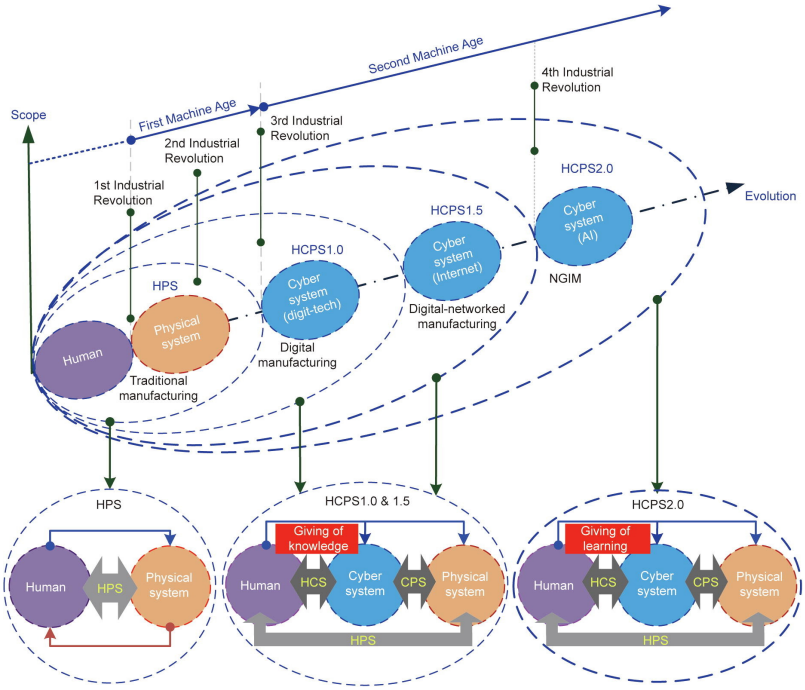


Figure 2.5: Evolution of Human-Cyber-Physical Systems [40]

In spite of all the benefits conveyed by the implementation of HCPSs, three main barriers have already been identified [40]:

- System modelling - the considerable systems' complexity and the vast amount of data involved hinder the creation of accurate and efficient models.
- Knowledge engineering - the fact that knowledge is dealt with by the cyber entity and actual tasks are carried out in the physical world makes it vital to successfully integrate both realities.
- Human-Machine symbiosis - the best way to combine workers and machines has to be studied, i.e., tasks must be thoughtfully assigned to each entity so that the best possible outcome is achieved.

2.2.2 Cyber-Physical Production Systems

Cyber-Physical Production System (CPPS) is the term used to refer to the application of CPSs for manufacturing purposes [41]. In [32], a definition considering critical aspects of CPPSs, such as their adaptability to the environment and learning capacity, is suggested: "*Cyber-Physical Production Systems are systems of systems of autonomous and cooperative elements connecting with each other in situation dependent ways, on and across all levels of production, from processes through machines up to production and logistics networks, enhancing decision-making processes in real-time, response to unforeseen conditions and evolution along time*".

Regarding their structure, a framework for CPPSs is displayed in [42], as illustrated by figure 2.6.

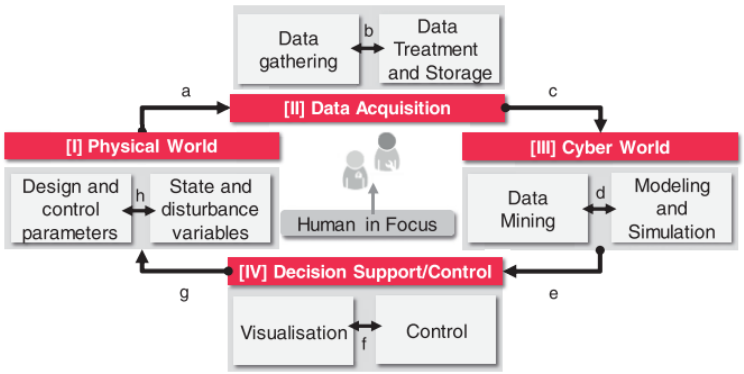


Figure 2.6: Cyber-Physical Production System's framework [42]. Adapted from [39]

Besides the physical and digital worlds mentioned in section 2.2, the bidirectional data flow between these two entities is emphasised:

- Data acquisition - encompasses the process of obtaining and storing data.
- Decision support and control - receives the simulations' outputs from the cyber world so as to intervene in the physical world, with or without human aid.

This framework also highlights the idea of human-centric systems, i.e., despite the fact that a lot of the processes are becoming increasingly autonomous, humans' importance in manufacturing environments has not decreased. In fact, workers will always be vital to complete whichever tasks machines are not capable of executing.

For that, CPPSs can be classified according to their dependence on human beings [32]:

- Full - the system is 100% independent and humans only play the role of supervisors.
- Automation - the only feature that the system lacks is the adaptability to its surroundings.
- Tool - the system works as an instrument and humans are responsible for the decision-making process.
- Manual - the system is only able to convey information to the human operator.

A CPPS is usually endowed with three main features [29], whose correspondence with the 3C model for CPSs is established in figure 2.7:

- Intelligence - portrays a systems' capacity to operate without human intervention.
- Connectedness - refers to the network formed by all the machines inside a production environment.
- Responsiveness - relates to systems' adaptability according to their surroundings.

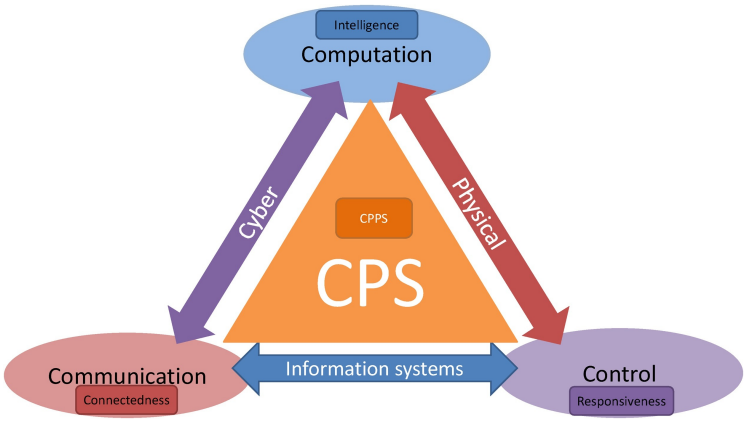


Figure 2.7: Comparison between CPSs and CPPSs [32]. Adapted from [43]

The concept of CPPS thus stands as a promising tool regarding the optimisation of industrial environments' efficiency [32] through reconfigurable production lines and autonomous decision-making strategies. Not only does the concept allow applications during production but it also enhances the design phase as well as the post-production stage [44]. These allow a reduction in the number of faulty products and the monitoring of products' life cycle [10], respectively.

The price of increasingly autonomous production environments is a considerable growth in the amount of equipment, namely sensors, actuators, etc., that need to be monitored and, from time to time, repaired [45]. However, recent technological advancements on predictive maintenance tools allow users to foresee which maintenance operations each equipment will require as well as when they will be necessary [46]. Consequently, machines' downtime is reduced [47] and long-term productivity increases [45, 48].

Given humans' role in manufacturing systems, the wide range of ongoing alterations demands higher skills from workers. Each person's abilities will have to be reassessed according to their role and competence development methods will be inevitable. Ultimately, it may be even necessary for workers to keep up with the engineering stage [49].

Learning factories have a word to say, in this regard. Such environments provide the perfect manufacturing scenarios allowing participants to acquire essential skills [50]. Moreover, they help attendants to understand the benefits of CPPSs and, consequently, break some of the barriers to the implementation of such systems [39].

Even disregarding the aforementioned security issues, there still is one tough barrier to the implementation of CPPSs: the undeniable economical benefits may not pay off the eventual environmental unsustainability. The need for considerable amounts of sensors and other equipment implies damages to the environment due to their production and disposal [51]. Besides, the extent to which CPPSs are really necessary to tackle certain problems is yet to be determined [42].

In order to reduce the amount of waste due to old equipment's disposal, retrofitting has been considered. It intends to adapt outdated physical assets, providing them with new functionalities without losing their original features, in lieu of spending considerable sums of money on new equipment. Aside from being inexpensive in relation to buying new machines, reconditioning old equipment is much more environmental-friendly. Despite not being mentioned in the RAMI4.0, this method is still seen as a feasible way of making the transition to I4.0 happen in a sustainable manner [52].

To sum up, should the implementation of CPPSs be carried out, the following changes are expectable [53]:

- Human presence will remain equally important, be it as the working force or playing the role of supervisory.
- The increasing tasks' complexity will force workers and engineers to work together, to some extent [54].
- Intuitive Human-Machine Interfaces (HMIs) will give a helping hand simplifying humans' interaction with devices [49].
- Repetitive and physically demanding tasks will be performed by machines whereas humans will be in charge of higher-level operations [55].
- Human-exclusive capacities will be more important than ever due to their costly automation [55].

2.3 Digital Twins

The concept of DT has been continuously submitted to a number of modifications since its introduction. The most remarkable stages from the history of this concept are shown in figure 2.8.

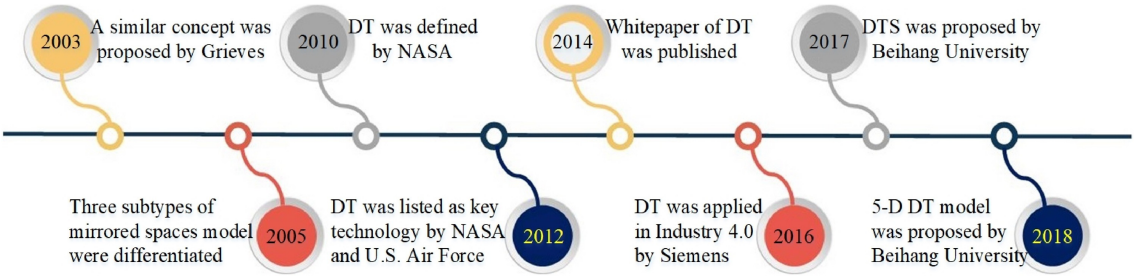


Figure 2.8: History of Digital Twins [1]

An identical concept to what is now referred to as DT was originally suggested by Michael Grieves on the occasion of an industry lecture on Product Lifecycle Management (PLM). According to its first definition, a DT was no more than a virtual entity similar to its physical counterpart [56]. More than half a decade later, National Aeronautics and Space Administration (NASA) came up with one of the currently most well-accepted definitions for the concept, according to whom, in the context of aeronautics, a DT is *"an integrated multi-physics, multi-scale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, etc., to mirror the life of its corresponding flying twin"* [57].

Another widely accepted view on the term was proposed by Grieves himself, in 2014, who suggested the following three-dimensional model [1, 56]:

- Physical asset - consists of any entity that lives in the real world and obeys to physical laws.
- Virtual model - mirrors the asset's physical properties, status and behaviour.
- Interconnections - allow the synchronisation of both entities.

In 2015, the wide dissemination of technologies, such as machine learning and wireless communication, leveraged the actual implementation of DTs [41]. The introduction of the term for I4.0's purposes was first carried out by Siemens, in 2016, two years before the update to the five-dimensional model was proposed [58]. This reference framework is well-known for standardising the development of DTs in different application areas. The additional entities, relative to the prior model, are mentioned below [56]:

- DT data - the data processing before being stored is vital due to the multitude of data sources and formats.
- Services in DT - access to software applications, virtual models and remotely stored data is granted by services.

However, the vast majority of the definitions create a misconception by claiming that the term refers to both the physical asset and its digital counterpart. Despite the fact that a DT does not live by itself, one shall not use the terms IoT, CPS and DT interchangeably.

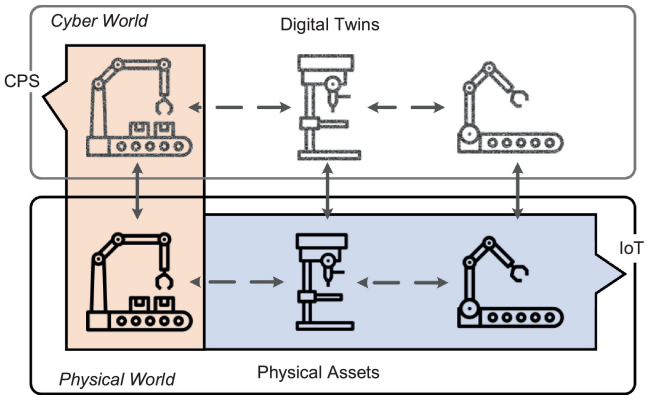


Figure 2.9: Comparison between IoT, CPSs and DTs [41]. Adapted from [59]

As portrayed by figure 2.9, the infrastructure that allows data exchange between physical assets is referred to as the IoT [59]. On the other hand, a DT solely refers to entities from the virtual world, whereas a CPS encompasses elements from both physical and cyber worlds [60].

Furthermore, not every virtual model can be considered a DT, as represented in figure 2.10.

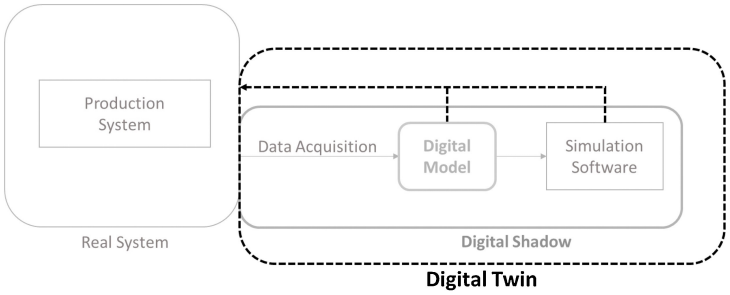


Figure 2.10: Digital Model vs Digital Shadow vs Digital Twin. Adapted from [61]

Depending on the level of integration between the physical asset and its virtual corresponding, three designations are proposed in [62]:

- Digital Model - the data exchange between the physical and the cyber entities is manually performed in both ways.
- Digital Shadow - the digital entity is automatically updated when a change occurs in the physical asset.
- DT - the digital entity autonomously influences its physical twin and vice-versa.

A reference model of a DT is proposed in [41] and presented in figure 2.11.

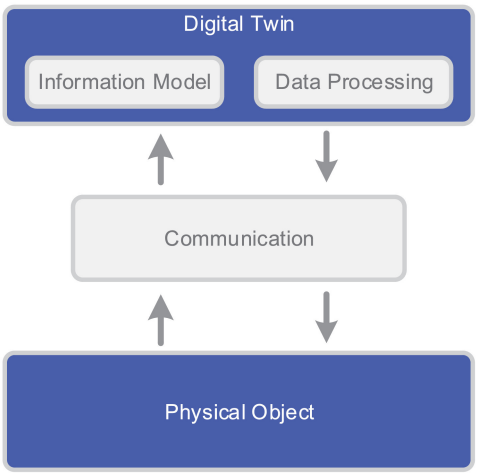


Figure 2.11: Reference model of a Digital Twin [41]

This model emphasises three main components [41]:

- Information model - is responsible for the physical description of the asset and the operations it undergoes. ISO 10303, also known as STEP, is the most popular existing standard which allows Computer-Aided Design (CAD) systems to communicate. Complementarily, Open Platform Communications Unified Architecture (OPC-UA) stands as one of the most common solutions to provide data exchange between devices, machines and systems.
- Industrial communication - ensures the bi-directional data flow between the physical object and its digital counterpart. A few options are available: original from the 70s, Fieldbus networks are well-known for their deterministic communication period, despite their low throughput; Ethernet-based networks are compatible with Internet integration and capable of transferring data over larger distances; Wireless networks do not entail wiring costs but are less reliable for real-time data exchanges.
- Big data processing - efficient data processing methods are required to deal with a multitude of data formats. It involves tackling problems, such as the presence of noise and missing values from the acquired data and storing information in either relational or non-relational databases, depending on the context.

In spite of the great interest in DTs and their applications, actual implementations are far from happening at a large scale. This is mostly due to the lack of standards across all the aforementioned components of the suggested architecture. Since the integration of the physical and digital worlds to its full extent is still a complex challenge, the process of developing a DT tends to prioritise the simulation of physical assets leaving the interconnection of both entities to a later stage, which, in the end, is often disregarded [41].

According to [63], the development of a DT may be tackled as follows:

- Modelling and simulation - consists of creating the virtual model and ensuring that it successfully mirrors its physical twin's behaviour.
- Data fusion - in this stage, sensors gather data from the real world and, when combined with data analysis methods, are able to extract significant features.
- Interaction and collaboration - ensure the bi-directional data flow between the physical asset and its digital counterpart, so that both entities are always synchronised.
- Service - refers to the implementation of the final objectives of a DT, namely monitoring and maintenance and downtime prediction.

In order to model and simulate an object from the real world, several software might be used, namely ANSYS Twin Builder, AnyLogic and Simio. ANSYS Twin Builder allows users to build exportable models from scratch, simulate systems considering realistic physics and easily prototype HMIs [1]. However, the graphical requirements to run this software are above the average [64]. On the other hand, AnyLogic and Simio are applications that allow 3D agent-based modelling through intuitive graphical interfaces and feature a Monte Carlo analysis tool, important for optimising parameters [65]. Furthermore, these lightweight software may be connected to databases, e.g., MySQL, ensuring efficient access to data and considerable storage capacity.

The countless existing technologies presented in [1] highlight that the major problem regarding a DT's implementation lays on the lack of infrastructure to integrate every necessary tool for the development of a fully autonomous DT, rather than on a particular stage of the process.

2.3.1 Digital Twins for Manufacturing

Despite finding applications in several fields, such as smart cities, civil engineering, agriculture, aeronautics, among many others [66], DTs for manufacturing will be in the spotlight of this project. In this area, DTs have already been used to ensure reasonable proximity between established goals and real outcome, enhance production processes and quality check and monitor and make predictions related to the manufacturing systems [67].

As a result, the term has become much more embracing since not only does it entail the holistic virtualisation of machines but also the representation of products during every stage of their life cycle [41]. Consequently, DTs' usefulness ranges from design validation through simulations without costly prototypes [68] to obtaining the best maintenance approach for a specific failure through iterative methods [69]. Furthermore, reconfiguration strategies for complex systems can be better-planned by DTs, over workers [70]. Moreover, acquired expertise can be documented, in order to be transmitted throughout a company's staff [1].

However, DTs' applications for manufacturing purposes are not confined to the equipment level. Immediately above the latter, the factory level can be considered, which abstracts the types of production made available by the factory as a whole, rather than focusing on a specific piece of equipment. Finally, at the enterprise level, DTs are used to detail the evolution of the production at the factory by keeping track of previous production records, including types of products, failures, and others. Nevertheless, applications of DTs are still in their infancy and, for that reason, mostly focus on the equipment level [12] for being less dependent on integration than the other positions from the aforementioned hierarchy.

In any case, a method to assess whether the set of goals is being achieved is of the utmost importance. For this, KPIs are used [71]. According to [72], if we consider a Body In White (BIW) manufacturing system, for instance, the idle time of each machine and the duration of each stage are some of the values that a DT should track.

Some of these values can be directly obtained, such as the number of products at the end of a production line. However, a considerable amount are derived from calculations. One example is the cycle time of a robot, which is obtainable by subtracting the time of arrival from the time when the product left the station. The onion model for KPIs, presented in [71], is representative of these two distinct approaches.

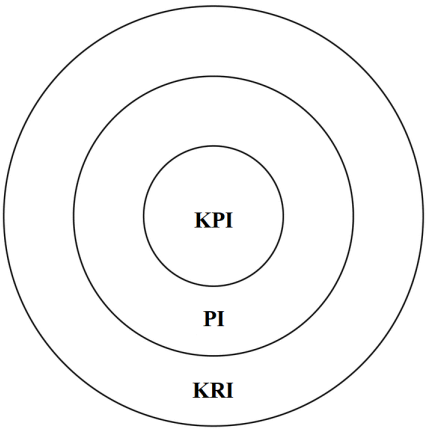


Figure 2.12: Onion model for KPIs. Adapted from [71]

According to figure 2.12, KPIs result from the combination of Key Result Indicators (KRIs) and Performance Indicators (PIs). The former consist of values directly obtained from machines, whereas the latter are mathematically derived from KRIs [71].

In a nutshell, thanks to recent breakthroughs in AI, big data analytics and machine learning algorithms, DTs stand out as powerful tools for monitoring systems and maximising earnings [73, 74]. Machines' failure, which used to be predicted based on signal analysis and assets' visual examination, are now autonomously foreseeable by DTs [75]. Therefore, factories' efficiency tends to increase and expenses are significantly reduced due to lower machines' downtime [76].

2.4 Research Gaps

According to the literature review on DTs carried out in [62], more than half of the analysed articles approached the paradigm in a conceptual way, which highlights the early stage of DTs' implementations. Regarding the level of integration, 35% of the considered publications only achieved the Digital Shadow level, 28% dealt with Digital Models and no more than 18% were considered Digital Twins.

More recent studies on DTs for manufacturing suggest that 85% of the applications tackle the lowest level of the hierarchy referred in subsection 2.3.1, whereas 11% reach the factory level and only a single application deals with human integration. The focus on abstracting manufacturing assets has led to applications in different areas, such as monitoring and prognostics. However, most applications only support humans in the decision-making process and do not implement autonomous feedback from the cyber entity to the physical asset [41].

In [12], a semantic model was developed with the aim of tackling resource virtualisation. Describing a machine's attributes and abilities and the interaction between resources through semantic web languages allows the representation of real-world information in a comprehensive format for computing. Furthermore, Protégé, a free open-source software for developing ontologies, enables the improvement of the created models' scalability through its objects' properties [70].

The proposed methodology is represented in figure 2.13.

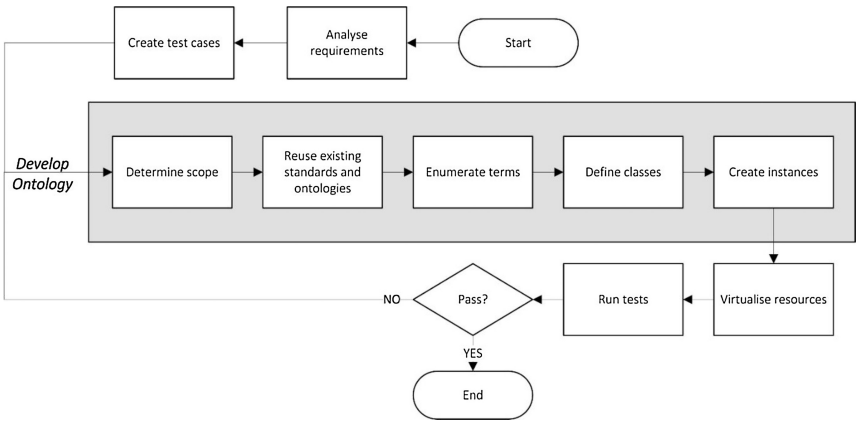


Figure 2.13: Resource virtualisation procedure [12]

For starters, one has to determine which resources will be abstracted in the digital world and which situations will be used to test the model. Afterwards, the ontology is developed: the scope must be defined, preferably, using existing standards [77] and classes, object and data properties and, finally, instances are created. Ultimately, resources are abstracted, tests are conducted and the created ontology is either approved or rejected, in which case it has to be rebuilt. The proposed framework used Ontology Web Language (OWL) as the ontology language and Jena as the rule language for the resources' capabilities description, due to its interface with Resource Description Framework (RDF) and OWL [12].

Another practical application was implemented in [78], even though, in this case, the goal was to produce a high-fidelity model of a physical object focusing on its geometry. So as to achieve the objective, the already mentioned ISO 10303, or simply STEP, was used. Besides the precise representation of geometrical objects through standardised terms, STEP excels due to the fact that *.stp* files are usable by other software. These may be imported to the application developed in MATLAB, GEMINI, which is also capable of exporting *.stp* files, updated through the Graphical User Interface (GUI). Nevertheless, this application only tackles the challenge of representing the physical shape of an object, disregarding its functionalities.

In [61], a DT was developed with the aim of monitoring the energy consumption of a I4.0 laboratory. The OPC-UA was used to ensure communication among machines and, consequently, for data acquisition. Aside from being a step towards standardisation, this protocol has an embedded security layer, ruling out the need for further safety rules. Once again, MATLAB was used, in addition to Simulink, and a GUI was developed. Furthermore, the OPC Toolbox for MATLAB was necessary with the purpose of integrating data exchange between the servers and the simulation, given that all the Programmable Logic Controllers (PLCs) were compatible with the aforementioned communication protocol.

Finally, in [79], the DT concept was applied to a rear lights' production system. In order to overcome one of the major barriers regarding DTs implementations, the integration of the physical asset and its digital counterpart, two models were created using AnyLogic. These were the front-end, representing the physical production system, which was only capable of executing tasks and the back-end, representing the actual DT, which was responsible for giving orders to the frontend. Both models were connected using a Java interface. Simulations were carried out for different scenarios with a variable amount of sensors along the railway, availability of feedback from each station and respective failure rate. Additionally, throughput time, lost messages and total messages were used as KPIs. The following conclusions could be derived:

- Throughput time was reduced by lower failure rates, available feedback from stations and higher amounts of sensors.
- The amount of lost messages increases with the number of sensors due to concurrent access to the communication resources.
- In spite of delaying communications, a higher number of sensors reduces throughput time, in particular when stations' failure rate increases.

Overall, implementing DTs raises a number of issues related to communication speed constraints, the data acquisition method, the lack of standards that ensure scalability of existing solutions and unawareness of how to integrate human beings in the process [41]. Consequently, the majority of applications fail to offer the supposed services [69] and, since most implementations' cyber entity is unable to control the physical asset [61], virtual entities only reach the level of Digital Shadows.

Chapter 3

Architecture

According to what was stated in section 2.3, DTs' solutions currently lack standards, which turns every implementation into a completely unfamiliar challenge. Depending on the system whose DT is to be created, different software might be needed in order to develop the digital model and distinct technologies may be considered for integration purposes as well.

For that reason and despite having been inspired by the reference model portrayed in figure 2.11, a three-layered architecture featuring some peculiarities was proposed, given the need to adapt to the already programmed physical system. Nevertheless, the fact that, in this solution, the feedback from the digital model does not autonomously affect the physical system does not prevent the following architecture from being extended, in order to include such feature.

For this project, the bottom layer, hereinafter referred to as the Physical Layer (PL), contains the manufacturing unit simulator, i.e., the physical entity of the CPPS, where the real KPIs are obtained from. This unit follows the principles of a Multi-Agent System (MAS) and corresponds to the physical object whose DT was developed for this thesis.

Immediately above the PL, an Integration Layer (IL) was added. Encompassing an API and a database that stores the physical system's state jointly with the production plan, this segment of the architecture enables the communication between the manufacturing unit simulator and its digital counterpart.

At the top lies the Simulation Layer (SL), which contains the simulation model. The latter relies on the database's information from the IL to emulate the physical system's behaviour and is responsible for providing users with real-time predictions for each KPI.

All the aforementioned components from the suggested architecture and the interactions among them are represented in figure 3.1.

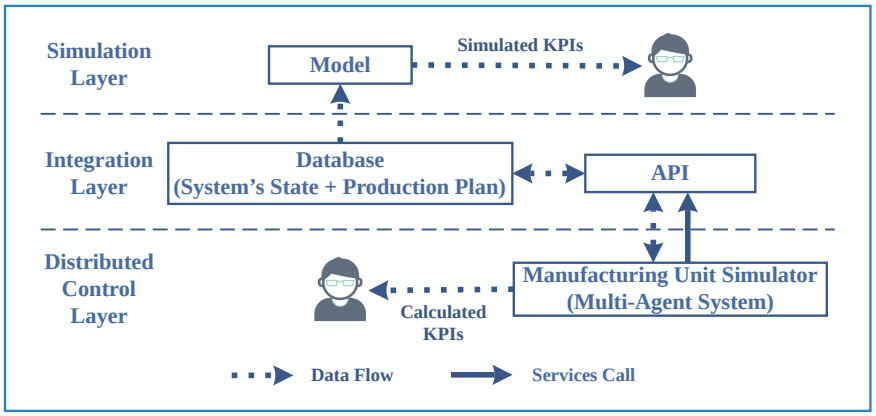


Figure 3.1: Suggested architecture

In the following sections, each of the aforementioned architecture's segments are looked at in depth.

3.1 The Physical Layer

As previously mentioned, the PL consists of a manufacturing unit simulator. Even though its implementation relies on a whole architecture by itself, some aspects must be highlighted in order to ease the comprehension of the remaining project. As a result, only the core concepts from the architecture will be approached. Nevertheless, further information on this system is available in [80].

The above-mentioned MAS essentially contains four types of agents:

- Product Agent (PA) - is responsible for abstracting a product since it is deployed until it reaches the end of the line and for ensuring that all due tasks are performed on the respective product.
- Transport Agent (TA) - represents a conveyor, which allows products to move throughout the line.
- Resource Agent (RA) - abstracts a processing station, where tasks can be performed.
- Deployment Agent (DA) - is responsible for launching and managing all other agents.

Additionally, in order to describe these entities and their properties in a meaningful way, a data model is required. Otherwise, concepts such as product, conveyor and resource, as well as the relationships between them, are unknown to the system. Thus, the ontology outlined in figure 3.2 was considered.

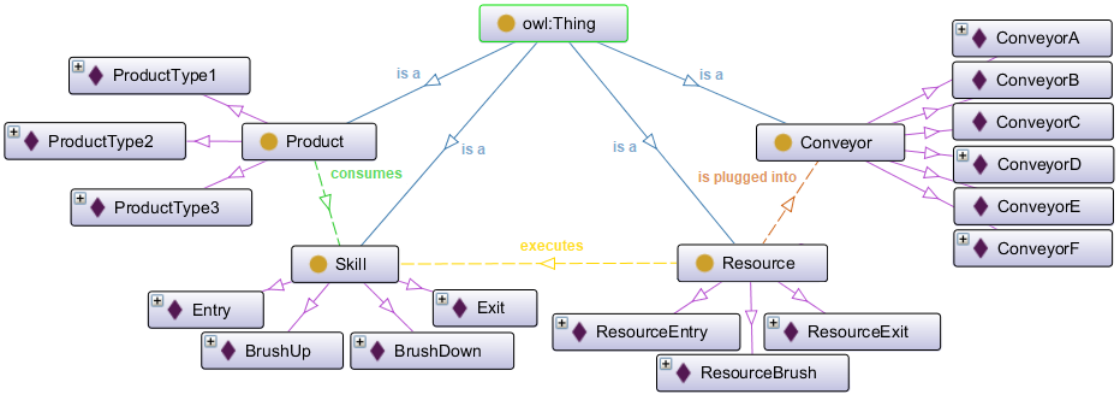


Figure 3.2: Multi-Agent System's data model

According to the previous image, the adopted data model describes four types of entities:

- Conveyor - there are six individuals of this type, whose designations range from conveyor A, the entry conveyor, to conveyor F, the exit conveyor. Each conveyor is able to carry one product at a time.
- Resource - each resource can perform one out of three tasks: adding a product to the line, removing a product from the line or brushing a product. A resource must be associated with one of the conveyors.
- Skill - apart from entering and leaving the production line, a product's top and/or bottom can be brushed. Each skill is always executed by a resource.
- Product - product types differ from one another on the amount of skills of each type they consume, i.e., products from type one consume one BrushUp, products from type two consume one BrushDown and products from type three consume one BrushUp and one BrushDown. In spite of there only being three preset product types, more variants can be easily added by editing the ontology.

Finally, an overview of the PL's behaviour is provided by figure 3.3.

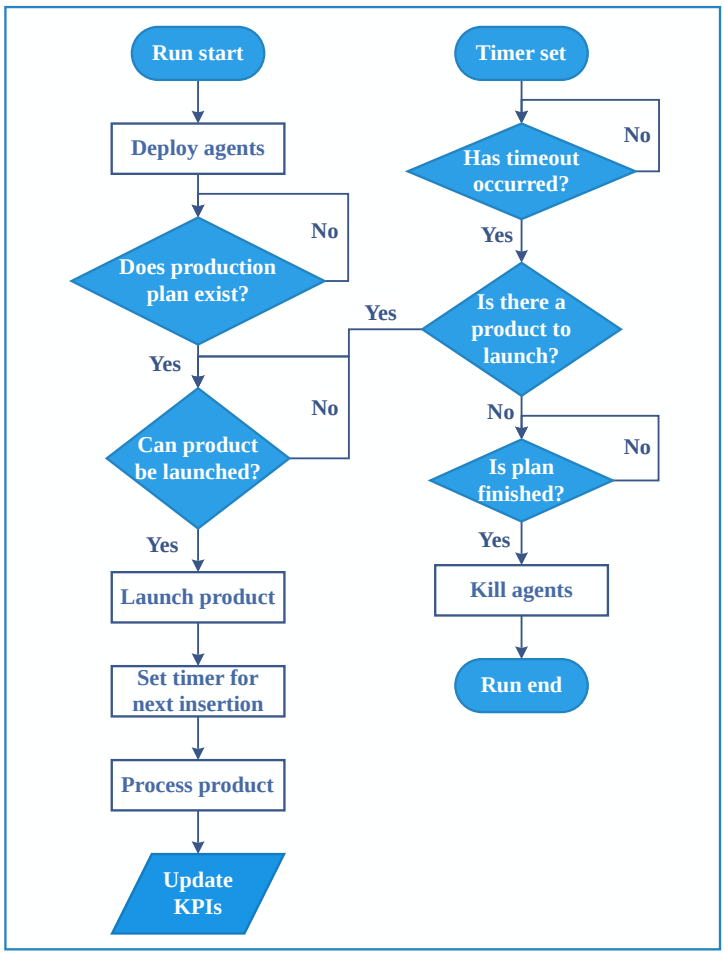


Figure 3.3: Physical Layer's behaviour

When launched, the MAS starts by deploying all agents and then remains idle until a production plan is introduced through the GUI. Following the introduction of a suitable production plan, the first product is added to the line and a timer is set in order to control the insertion of the next product. The newly-added product follows its course and, by the time it is removed from the line, the KPIs are updated. Meanwhile, as soon as the timer reaches the timeout condition, the next product is deployed, provided that the entry conveyor is free. This sequence is repeated until the whole production plan has been processed, by which time all agents are killed and the run ends.

3.2 The Integration Layer

The IL's role is of utmost importance as it allows the MAS to provide the digital model with crucial information from the physical system. For that, not only does it include a database, where the system's current state and the remaining production plan are stored, but it also features an API, whose main purpose is to interact with the latter.

Given that this layer cannot be looked at on its own, the following flowchart resuming the interactions between its components also refers to the MAS from the PL, which directly benefits from the API's functionalities.

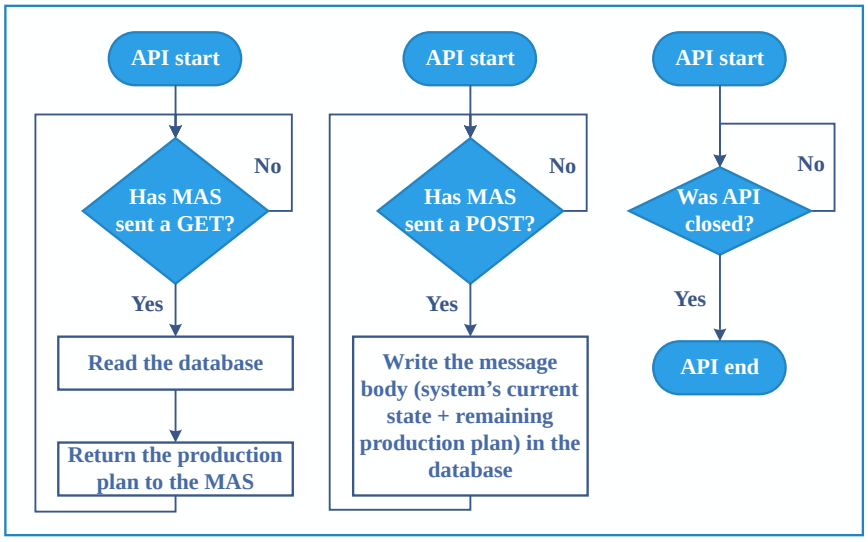


Figure 3.4: Integration Layer's behaviour

As tacitly depicted in figure 3.4, the API entails two services that are capable of working concurrently.

One of them allows the obtainment of the production plan contained within the database. For that, a *GET* request with no specific message must be sent to the API, which reads the database and then returns the production plan. This type of operation comes in handy before the insertion of a product, as it allows the client to check which type of product is to be inserted next.

The other service is responsible for updating the database. In this case, a *POST* request, whose message body must contain the system's current state and the remaining production plan, has to be sent to the API, which writes the content of the message in the database. In its turn, this service is called whenever a product is added to the production line, changes position or its processing ends, or when the production plan is first created.

3.3 The Simulation Layer

The SL solely includes the digital model. So as to serve its purpose, a correct description of the initial system's state and the remaining production plan are required. This way, the model knows where to start simulating from and the type of products that must be inserted afterwards. Additionally, the user is expected to configure the time interval between the insertion of two consecutive products and the delay associated with the consumption of each type of skill, according to the ontology.

In spite of being able to consider any desired KPI, this work only focused on the cycle time and the throughput. These are updated whenever a product leaves the system, even though the simulation only ends by the time that the production plan is completed.

The sequence of events that take place from the start of the model until the final predictions of each KPI are obtained is summarised in figure 3.5.

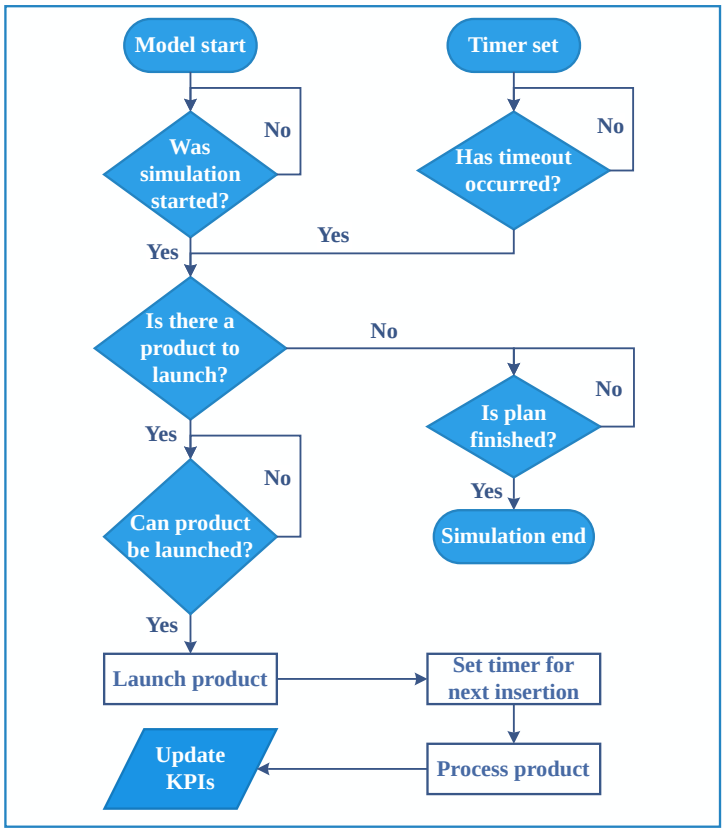


Figure 3.5: Simulation Layer's behaviour

Expectedly, this diagram resembles the one presented in 3.3. The main difference lays on the fact that the DT is able to start simulating from a desired initial state, unlike the physical system which always starts running without any products on the conveyors. Consequently, the production plan may be empty, in this case, making it necessary to check if there is a product to launch immediately after the beginning of the simulation.

Chapter 4

Implementation

The implementation hereinafter described relied on different technologies depending on each layer's role.

For developing the simulation model, AnyLogic, a powerful and well-documented tool that endures simulations based on agents, discrete events and systems' dynamics was used. The applets generated by this software are developed in Java and, using its Professional License, models can be exported in order to be run independently.

Regarding the database, a *csv* file was chosen due to the low complexity level of the information to be stored. Additionally, this format's high compatibility with a wide range of programming languages turned it into the best option.

The API used to interact with the *csv* file was developed using Flask. Not only does this framework enable the development of web applications but it also benefits from the versatility of the Python programming language.

Regarding the MAS, given that it was already implemented using JADE, the classes that were added in order to integrate the already existing system with the newly-created DT were developed in Java as well.

This chapter contains four sections: The Database File, The Simulation Model, The API and The Multi-Agent System. Each of them intends to clarify the implementation of the corresponding component from the architecture, in accordance with the order by which they were developed.

4.1 The Database File

The first component to be settled was the file where the initial state and the production plan are kept. Deciding which information was needed in order to fully describe the system's state and the next products to be inserted was fundamental, making it the first priority at the implementation level.

After considering all of the system's relevant aspects, the below-described structure emerged.

The first six lines of the file are mandatory and contain a maximum of five fields, that describe each conveyors' initial state, sorted as follows.

- Timestamp - UNIX timestamp, in milliseconds, of the moment when the corresponding conveyor's state was last updated.
- Station Flag - indicates whether a conveyor has a station attached to it (1) or not (0).
- Brush Ups - amount of BrushUp skills consumed by the product which is currently placed at a conveyor (-1 means that the conveyor is empty).
- Brush Downs - amount of BrushDown skills consumed by the product which is currently placed at a conveyor (-1 means that the conveyor is empty).
- Process Finished Flag - indicates whether the product placed at the respective conveyor has already been processed (0) or not (1). This field is only inserted when the corresponding conveyor is not empty and has a station plugged into it.

Each one of the remaining lines is optional and has two mandatory fields: Brush Ups and Brush Downs. Each line, thus, represents one product from the production plan, respecting the order by which they are to be inserted.

A possible sample of such file is shown in listing 4.1.

Listing 4.1: Sample *csv* file

```
1600443880822;0;-1;-1
1600443877467;0;-1;-1
1600443875446;0;-1;-1
1600443880767;1;0;1;0
1600443880756;0;-1;-1
1600443872305;0;-1;-1
1;0
1;1
```

The example above represents a scenario in which conveyor D has a station plugged into it and is carrying a processed product of type two. The next product to be inserted will be of type one, followed by a type three product.

4.2 The Simulation Model

The development of the simulation model itself went through a variety of stages. For that reason, this section is subdivided in order to provide a clearer explanation of each phase.

4.2.1 The Visual Elements

The first stage involved a scaled representation of the physical system using elements from the Space Markup category available in the Material Handling Library, which led to the conveyor network represented below.

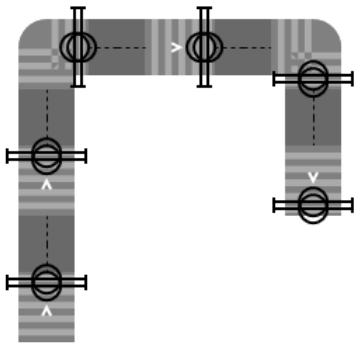


Figure 4.1: Conveyor network's 2D view

As portrayed by figure 4.1, the main components used were:

- Conveyors - grey-striped objects with a white arrow on top representing the direction in which products flow.
- Positions on Conveyors - black symbols placed near the end of each conveyor, whose importance will be explained in subsections 4.2.2 and 4.2.3.
- Transfer Tables - dark-grey objects used to connect consecutive conveyors. The physical system does not have such elements but they are necessary in order to connect conveyors in AnyLogic.

A 3D view of the system was also made available through the 3D Window object from the 3D section of the Presentation Library.

Furthermore, a new agent type, *Product*, was created. The object consists of a cube with side of 5 centimeters and two numbers on top of it representing the amount of skills of each type that the product consumes. The top view presented in figure 4.2 does not portray those numbers because it refers to a generic product.

Figure 4.2: Generic product's top view

Finally, a section for displaying the model's predictions was designed, as shown in figure 4.3.

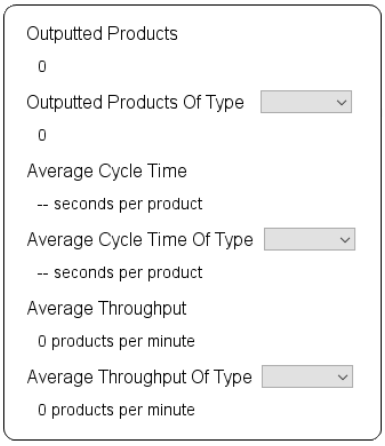


Figure 4.3: Simulation model's results section

The aforementioned section displays global statistics as well as type-focused results depending on the product type selected on the respective combo box. Nevertheless, its implementation will only be introduced in subsection 4.2.3.

4.2.2 The Blocks Diagram

In order to add products to a specific conveyor, control their flow throughout the conveyor network, among other tasks related to controlling the overall behaviour of the system, a blocks diagram is required. For this, the Blocks category from the Material Handling Library was used, resulting in the chart represented in figure 4.4.

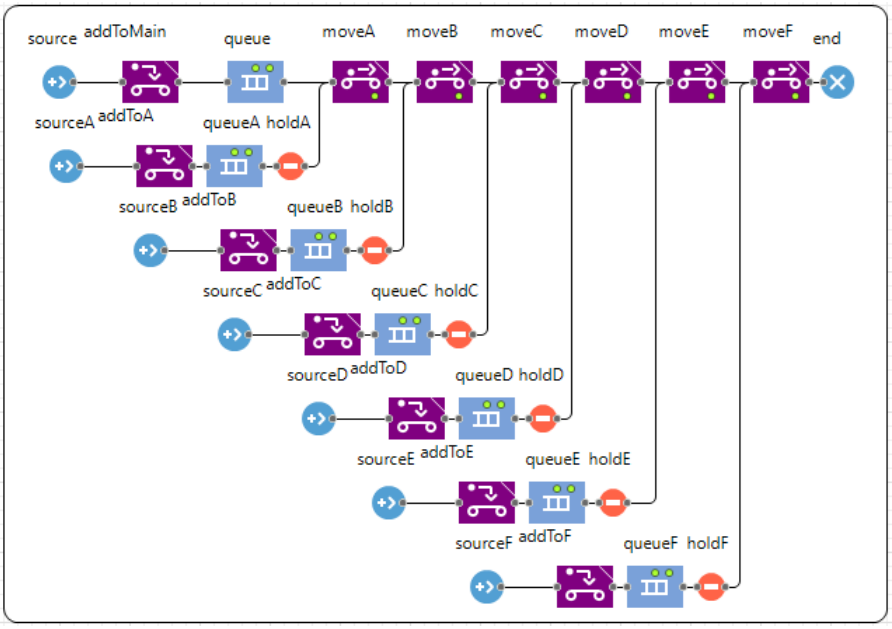


Figure 4.4: Simulation model's blocks diagram

This generic description of the system's behaviour required several types of blocks.

- Source - launches a new instance of a product by invoking the *Product* agent's constructor. The blocks from *sourceA* to *sourceF* are responsible for deploying a product on the respective conveyor during the first stage of the simulation, when the initial state is read from the file. On the other hand, the block named *source* launches all the remaining products from the production plan.
- Conveyor Enter - is placed after a source block and establishes, through Position on Conveyor elements, the conveyor where the newly-created product will be placed.
- Queue - stacks products in case they cannot be immediately added to the conveyor network.
- Hold - blocks the natural course of an agent. It is used to synchronise the beginning of the simulation by waiting for all products, from the initial state, to be added before the conveyors start moving.
- Convey - settles the sequence of conveyors that constitute the production line by defining each segment of the path.
- Sink - works as the endpoint of the production line, where products are dispatched.

However, the blocks diagram is not enough to describe some of the nuances of the physical system's behaviour, namely:

- AnyLogic does not directly limit the amount of simultaneous products per conveyor, which in the manufacturing unit never exceeds one product at a time.
- The diagram is unable to emulate the delays associated with the processing stations.

4.2.3 The Agents' Components

The simulation model, essentially, comprises two types of agents:

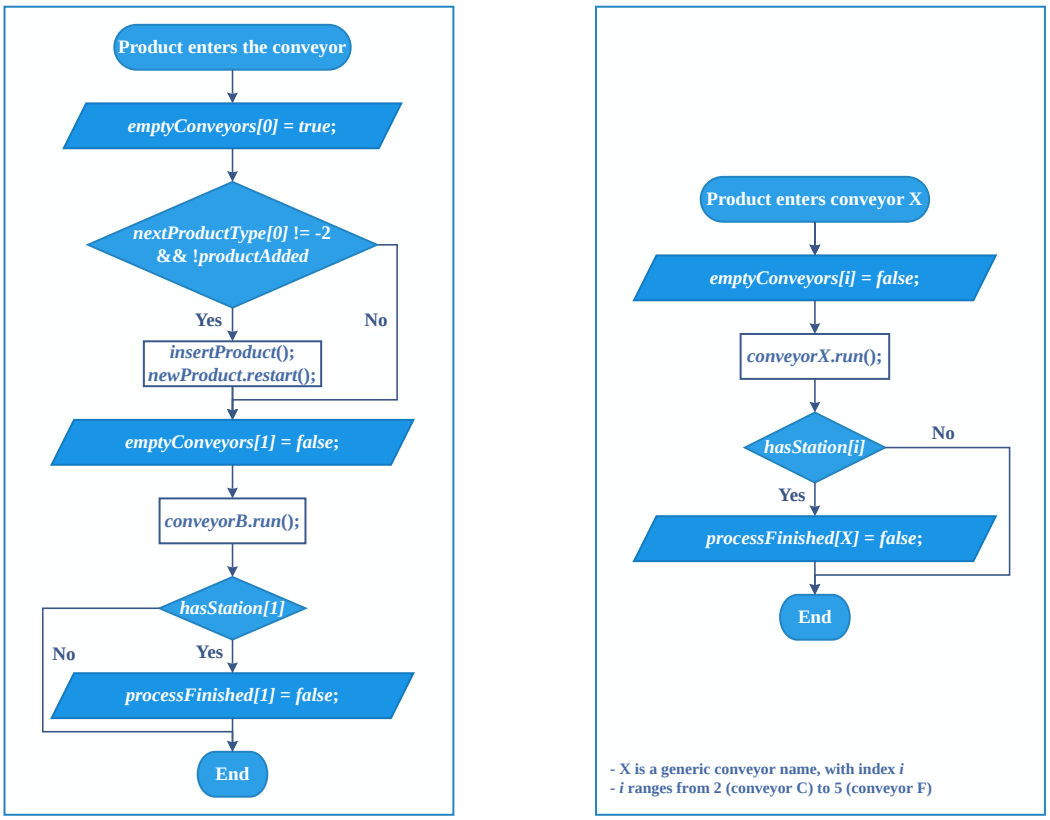
- *Product* - as already mentioned in subsection 4.2.1, it abstracts a product during its stay at the conveyor network. This object is described by four parameters: *brushUp* and *brushDown*, which are responsible for specifying the product type, and *entryTime* and *exitTime*, where the timestamps of its insertion and removal are stored, respectively. The type is also represented as a *String* as follows: "*brushUp*;*brushDown*". The *String* format will be mostly used henceforth.
- *Main* - includes all graphical and logical elements of the model and ascertains its correct behaviour until the final predictions are derived. Unlike *Product* type agents, no more than one instance is created per simulation.

In order to cover the gaps mentioned in 4.2.2, some extra logic was needed on the properties of the elements introduced in subsection 4.2.1. For that, some events, parameters and variables, available at the Agent Components category from the Agent Library, had to be defined. The core components from the *Main* agent are summarised in table 4.1.

Table 4.1: *Main* agent's components

Type	Name	Description
Event	*newProduct*	When a timeout occurs, a new product is inserted, in case the last attempted insertion was successful.
Event	*stationX*	There is one of these for each possible station's location. When timeout is reached, the corresponding *processFinished* flag is set back to *true* and conveyor X starts moving, if the following conveyor is free.
Parameter	*interArrivalTime*	Represents the time between the insertion of two consecutive products.
Parameter	*skillsDelay*	Array of two *double* where the delays of a BrushUp (first position) and a BrushDown (second position) are stored.
Variable	*emptyConveyors*	Array of six *boolean* flags that tell whether each conveyor is empty.
Variable	*hasStation*	Array of six *boolean* flags that tell whether each conveyor has a station plugged into it.
Variable	*nextProductType*	Array of two *int* that represent the type of the next product to be inserted. If there are no products left, both positions are set to -2.
Variable	*processedProducts*	Counts the amount of products that have left the line.
Variable	*processFinished*	Array of six *boolean* that tells whether the processing on each station is finished.
Variable	*productAdded*	Flag that is set to *false* whenever there is an unsuccessful attempt to insert a product, due to the entry conveyor being busy.
Variable	*productsToIgnore*	Counts the amount of products to be ignored for statistical purposes, for having been inserted prior to the simulation start.
Variable	*simulationStartDate*	Stores a *Calendar* instance relative to the simulation beginning.
Variable	*statistics*	Maps the amount of processed products of each type and their average cycle time to the product type.
Variable	*timeSpan*	Stores the timestamp of the simulation start and of the last product's removal.

Afterwards, some actions were added to the *On leading edge enter* and *On trailing edge exit* fields of each conveyor. The former's algorithm for each conveyor is portrayed in figure 4.5.



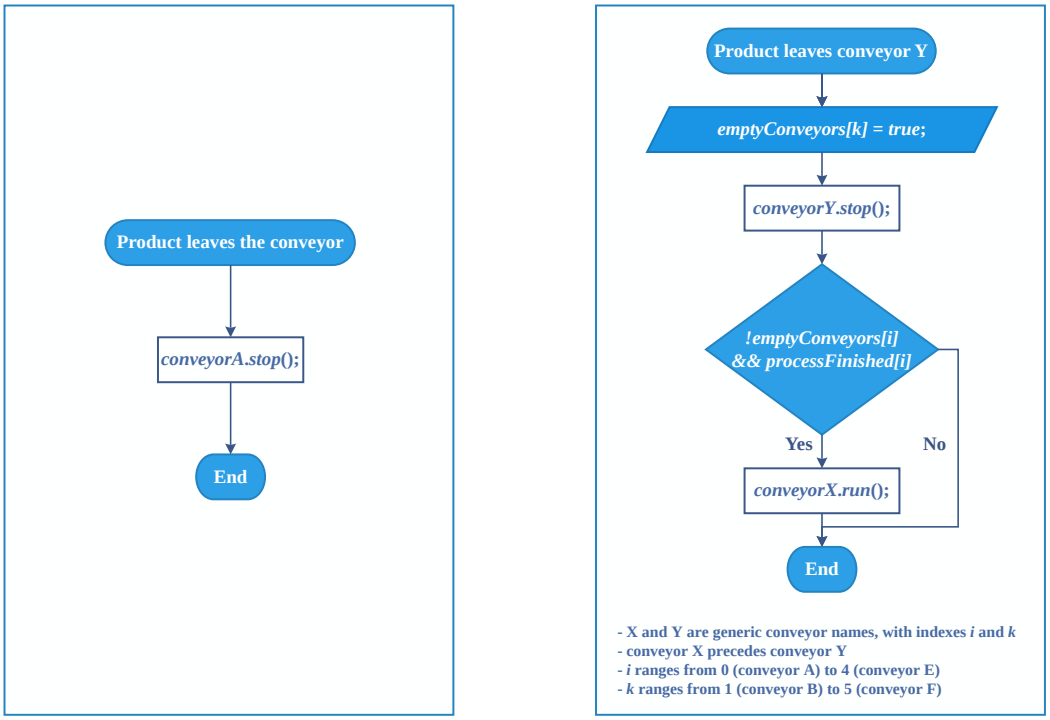
a Algorithm for conveyor B

b Algorithm for conveyors C to F

Figure 4.5: Conveyors' *On leading edge enter* algorithms

When a product enters conveyor B, the *emptyConveyors* flag for conveyor A is updated. Then, if there is a product to insert and the last attempted insertion was unsuccessful, the *insertProduct* method is called and the *newProduct* event is restarted. The *emptyConveyors* flag is set to *false* and conveyor B starts moving. Ultimately, if conveyor B has a station plugged into it, the *processFinished* flag is set to *false*. The algorithm for conveyors C to F is similar, except for the fact that it does not include the insertion of a new product on the production line. Conveyor B is the only one that needs such extra logic for being placed immediately after the entry conveyor, which does not have any code on this field because products never cross its beginning.

Similarly, the *On trailing edge exit* algorithm for each conveyor is shown in figure 4.6.



a Algorithm for conveyor A

b Algorithm for conveyors B to E

Figure 4.6: Conveyors' *On trailing edge exit* algorithms

When a product leaves one of the conveyors from B to E, its *emptyConveyors* flag is set back to *true* and the conveyor is stopped. Furthermore, if the previous conveyor is not empty and its *processFinished* flag equals *true*, it starts moving. As for conveyor A, the only taken action is stopping it.

On the account of the product being removed from the line before its trailing edge crosses the end of conveyor F, the equivalent algorithm for the exit conveyor had to be placed on the *On enter* field of *end* from the blocks diagram.

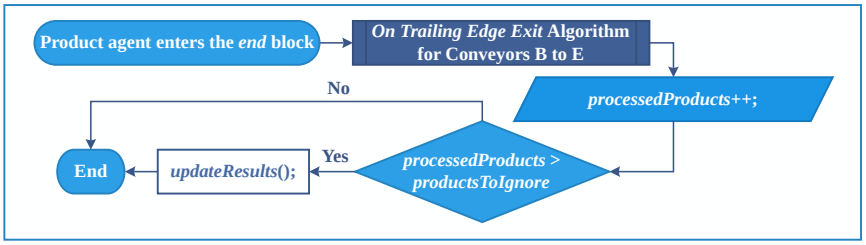
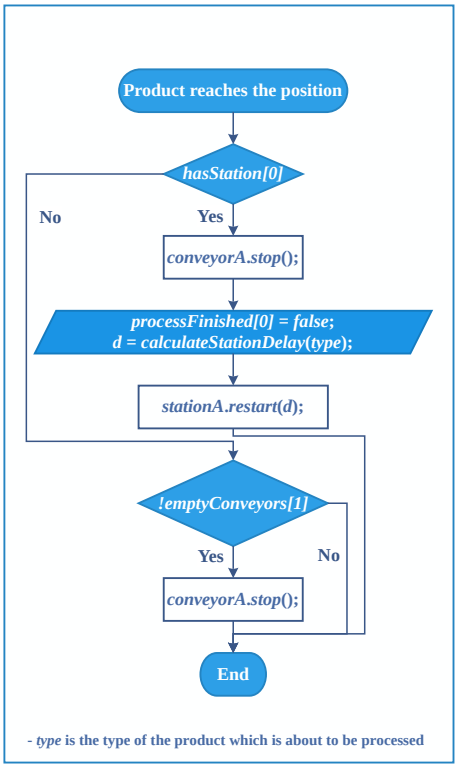


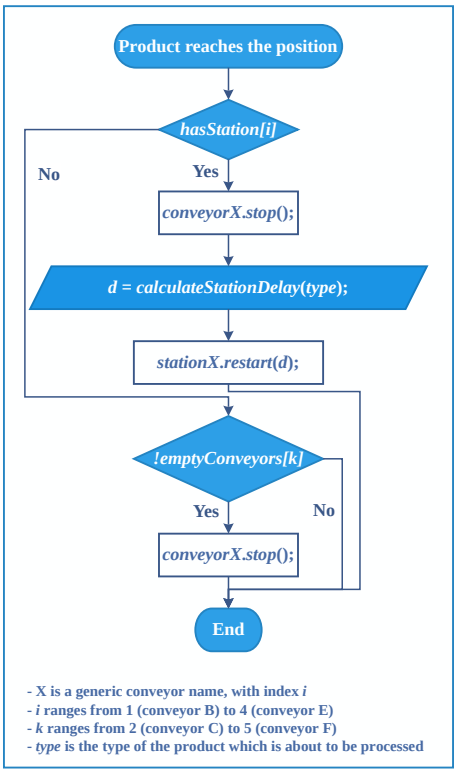
Figure 4.7: *End*'s *On enter* algorithm

In this case, the *processedProducts* counter is also incremented and, in case it overtakes the amount of *productsToIgnore*, the results are recomputed, as shown in figure 4.7.

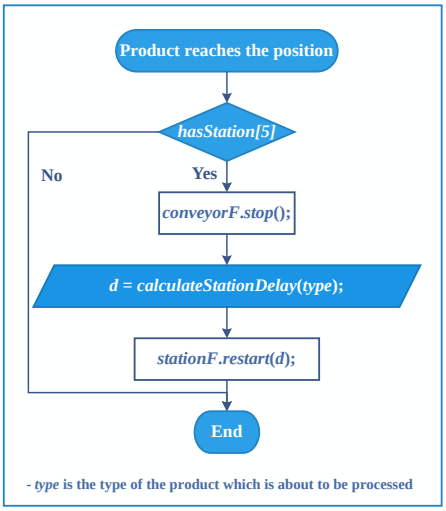
Additionally, the first *Position on Conveyor* element of each conveyor was endowed with some code so as to emulate the processing stations. As for the second *Position on Conveyor* element of each conveyor, no extra logic was necessary as they were only used to define the position where products are placed at the initial state.



a Algorithm for conveyor A



b Algorithm for conveyors B to E



c Algorithm for conveyor F

Figure 4.8: Positions on Conveyors' *On leading edge enter* algorithms

The previous algorithms behave according to the existence of a station. If there is a station plugged into conveyor A, the latter is stopped, *processFinished* is set to *false* and the *stationA* event is restarted with the delay obtained through *calculateStationDelay*. If otherwise, it is checked whether the following conveyor has a product, in which case conveyor A is stopped. The algorithm for conveyor B to E is exactly the same except for the fact that *processFinished* is not updated, because it is set to *false* as soon as the product enters the conveyor. As for conveyor F, for being the last one of the network, its algorithm is simpler since there is no following conveyor whose availability needs to be ascertained.

Ultimately and as it may be seen in some of the previous flowcharts, a certain amount of functions had to be developed. The *loadState* method, described in figure 4.9, is responsible for reading the initial state from the file.

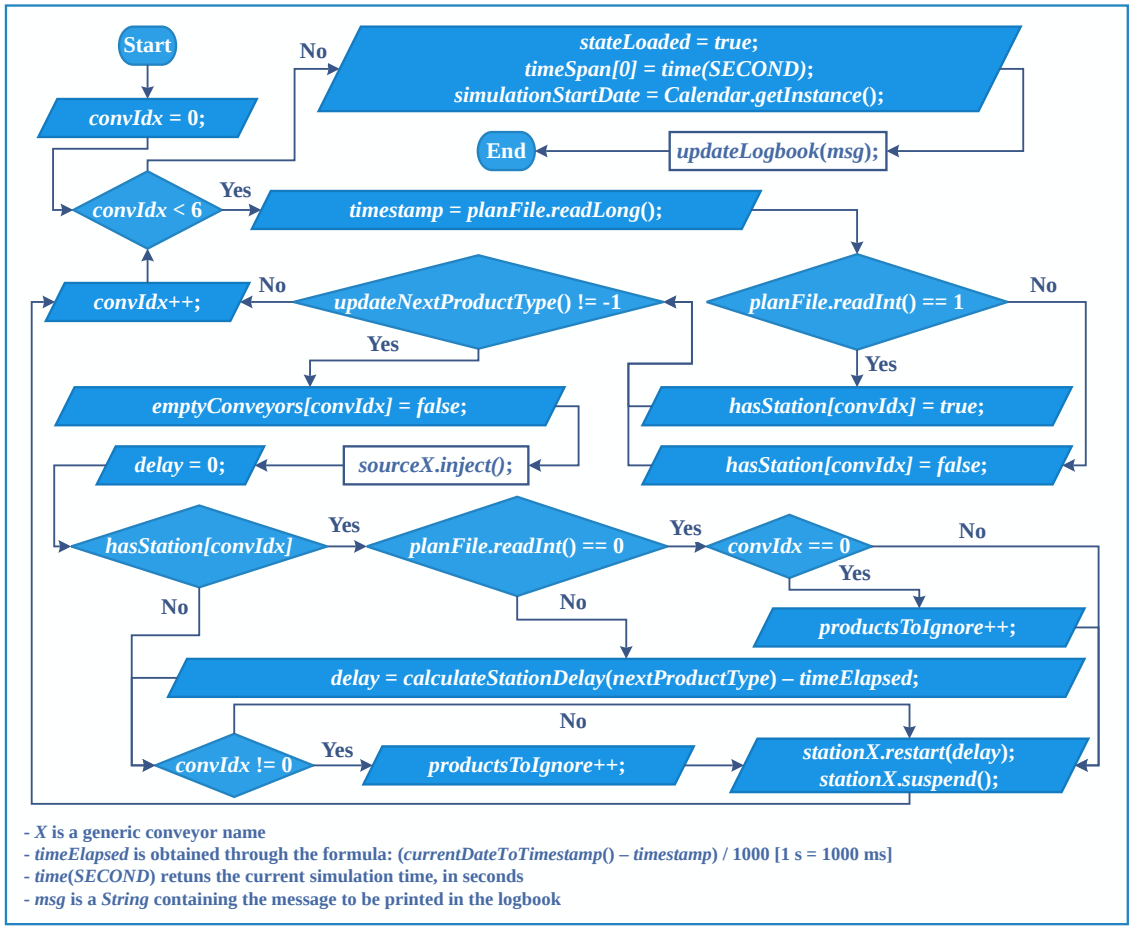


Figure 4.9: *loadState* method's algorithm

For each conveyor, the timestamp of the last update, the *hasStation* flag, the product type and, if there is a station with a product, the processing state are obtained. For each station with a non-processed product, the corresponding event is set and suspended. Finally, for each product which is not on the entry conveyor or that was already processed there, *productsToIgnore* is incremented.

In figure 4.10, the behaviour of the *insertProduct* method can be observed.

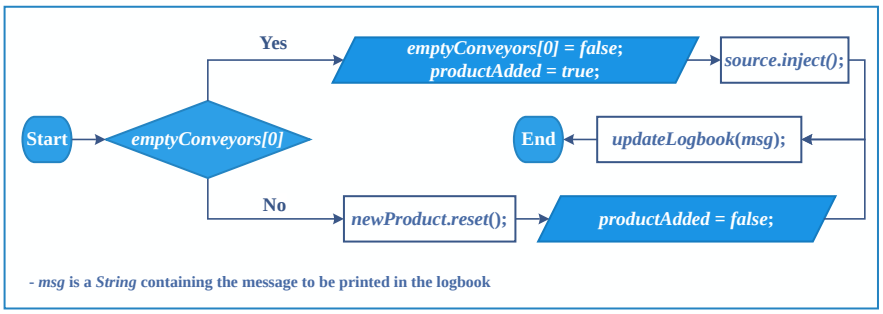


Figure 4.10: *insertProduct* method's algorithm

When this function is invoked, the *emptyConveyors* flag is checked. In case the entry conveyor is busy, the *newProduct* event is cancelled and the *productAdded* flag is set to *false*, representing that there was a missed attempt to insert a product. If otherwise, *emptyConveyors* is set to *false*, *productAdded* is set to *true* and the product is injected through the main source. In either case, an output message informing the result of the insertion is added to a logbook, at the end.

The description of the secondary functions can be consulted in table 4.2.

Table 4.2: *Main* agent's auxiliary functions

Name	Description
calculateStationDelay	Returns the processing time of a product of the given type through the *skillsDelay* parameter and the amount of skills from each type consumed by the product.
colours	Returns a colour (green, red or orange) according to the current state of a conveyor or station.
currentDateToString	Calculates the current date based on the *simulationStartDate* and the simulation elapsed time.
currentDateToTimestamp	Returns the current date as a timestamp, in milliseconds.
updateComboBoxes	Adds the given type of product to the combo boxes' options list, if it is not there yet.
updateGlobalResults	Updates the amount of processed products and the averages cycle time and throughput, using the values from *statistics*.
updateLogbook	Adds the given message to the logbook.
updateNextProductType	Resets *newProduct* and returns 0 when there are no more products to insert. If not, returns -1, if the current conveyor is empty, or 1, if otherwise. In either way, *nextProductType* is always updated.
updateResults	Updates *timeSpan* and invokes *updateStatistics*, *updateComboBoxes*, *updateGlobalResults* and, for each combo box whose selection matches the type of the last-removed product, *updateTypeResults*.
updateStatistics	Updates *statistics* considering the given product.
updateTypeResults	Is similar to *updateGlobalResults* but only deals with type-specific data.

4.2.4 The GUI

The last stage involved the design of a GUI whose main goal is to provide the end-user with an intuitive way of interacting with the digital model.

As soon as the application starts running, the homepage pops up. According to figure 4.11, it contains the title of this project, a picture of the physical system and three sliders that allow users to adjust the parameters described in table 4.1.

Figure 4.11: Simulation model's homepage

Once the model is run, the main screen is presented.

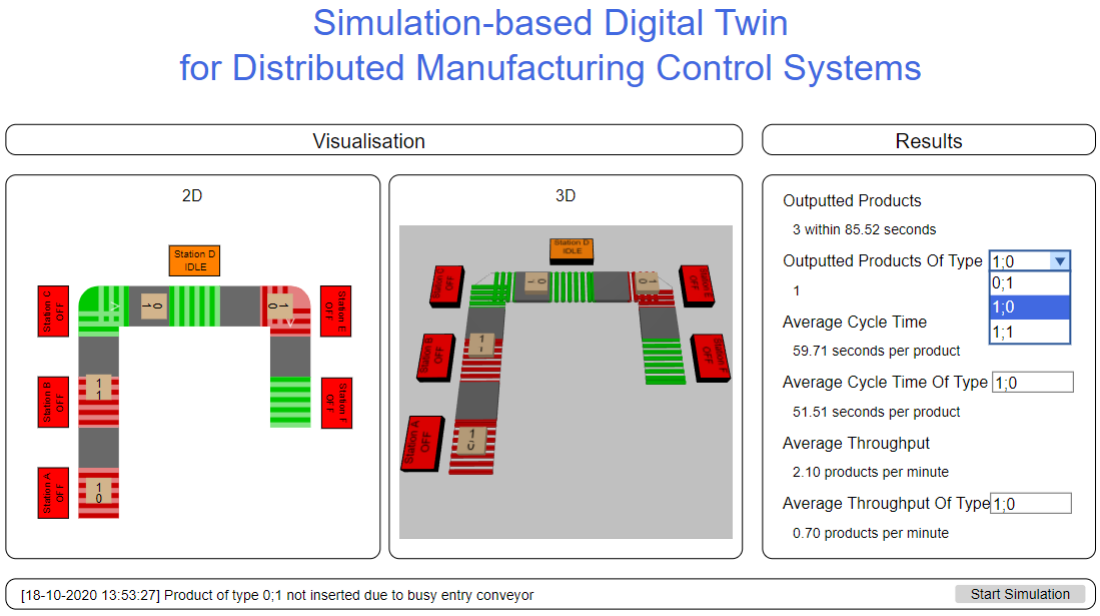


Figure 4.12: Simulation model's main screen

As shown in figure 4.12, aside from the elements introduced in subsection 4.2.1, the simulation screen includes an outputs' section, where feedback messages are printed, and an interactive button, used to start the simulation. The latter can only be pressed once, by which time the *loadState* method is invoked, every *hold* block is unlocked and the first attempt to insert a product is made. For this reason, the model must be reset before running a new simulation. Furthermore, the logbook is only saved when the application is stopped.

4.3 The API

As stated in the beginning of this chapter, the Flask framework was used in this stage of the implementation. Taking into account that its main purpose is getting information from the *csv* file or editing its content, the API only has two services.

On the one hand, a *POST* request may be sent through the `localhost:5000/api/v1/updateCSV` Uniform Resource Locator (URL), triggering the *updateCSV* method with the aim of modifying the database file. For that, the request's body must have a JavaScript Object Notation (JSON) message containing the fields introduced in section 4.1. A sample of the JSON that would generate the *csv* file from listing 4.1 is shown below.

Listing 4.2: Sample JSON message

```
{
  "conveyorA": [1600443880822,0,-1,-1],
  "conveyorB": [1600443877467,0,-1,-1],
  "conveyorC": [1600443875446,0,-1,-1],
  "conveyorD": [1600443880767,1,0,1,0],
  "conveyorE": [1600443880756,0,-1,-1],
  "conveyorF": [1600443872305,0,-1,-1],
  "nextProduct1": [1,0],
  "nextProduct2": [1,1]
}
```

The *updateCSV* method behaves as portrayed by figure 4.13.

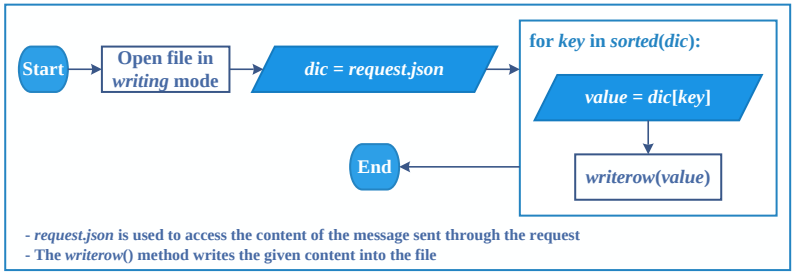


Figure 4.13: *updateCSV* method's algorithm

After opening the file for writing, the content of the message is stored in a dictionary. Then, for each key, the corresponding value is written into the file.

On the other hand, a *GET* request can be sent through the `localhost:5000/api/v1/getProducts` URL, which calls the *getProducts* method with the goal of obtaining the list of products to be added to the production line, i.e., the remaining production plan. In this case, no information needs to be sent in the request's body. The *getProducts* method's behaviour is illustrated in figure 4.14.

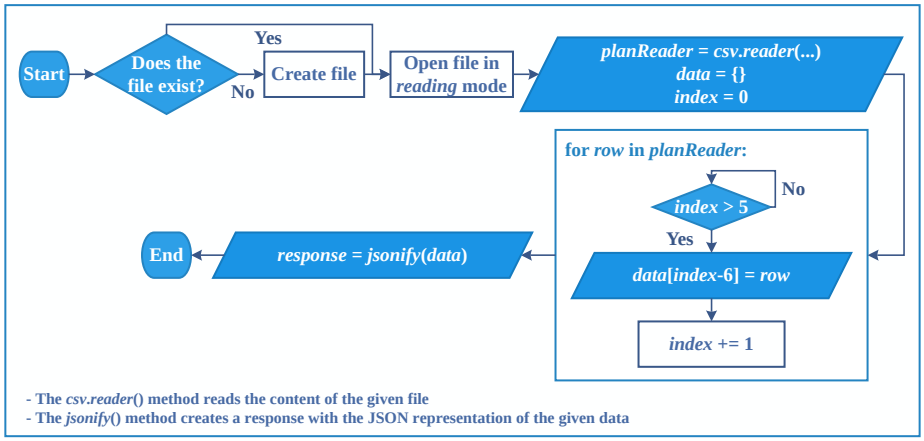


Figure 4.14: *getProducts* method's algorithm

Whenever the previous service is called, the file is created if it does not exist yet and, afterwards, it is opened for reading. Followingly, the file's content is stored in a variable and an empty array as well as a control variable are created. Then, from the seventh line of the file's content on, the information is stored in the aforementioned array. This is due to the fact that the first six lines are always reserved for the initial state information. Finally, the data is converted into the JSON notation and returned to the MAS.

4.4 The Multi-Agent System

The original Java project of the MAS was only capable of controlling the physical system's behaviour, because it did not need to communicate with external applications. Furthermore, in the previous version, products were added manually through a graphical interface, which did not match the desired working mode for this project, where they ought to be inserted periodically. Consequently, in order to ensure the synchronisation between the manufacturing unit and the newly-created DT, a new Java class and a GUI, for inserting the production plan, were added to the initial project.

4.4.1 The *restClient* Class

The *restClient* class is responsible for ascertaining the synchronisation between the physical system and its digital counterpart. For that, the variables described below are vital.

Table 4.3: *restClient* class's variables

Type	Name	Description
HashMap	*conveyorStatus*	Maps an array with the current status of each conveyor, i.e., the fields from section 4.1, to the conveyor's name.
HashMap	*productsMap*	Maps the product name to the name of the conveyor where it is currently located.
HashMap	*productionPlan*	Maps each product type from the plan, in the form of an array, to a number representing the order of insertion, e.g., 0 is the key of the next product to be inserted.
HashMap	*productTypes*	Maps a *String* representing the product type to an array with the amount of BrushUp and BrushDown skills consumed by the products of such type.
HashMap	*entryTimes*	Maps the UNIX timestamp relative to each product's insertion to its name.
int	*totalProducts*	Stores the amount of removed products.
long	*startTime*	Keeps the UNIX timestamp relative to the simulation start.
double	*totalCycleTimes*	Stores the sum of cycle times from every removed product.
double[4]	*statistics*	Contains the total time spent at stations, the simulation elapsed time and the averages cycle time and throughput.

With the aim of guaranteeing that the aforementioned variables are properly dealt with and that the MAS keeps the file updated on a near real-time basis, some functions were also necessary. The main methods created in this class may be consulted in table 4.4.

Table 4.4: *restClient* class's main methods

Name	Description
createProduct	Adds a product with the given number and type to *productionPlan*.
resetStatus	Is invoked at the beginning of the execution or whenever a product changes position. In the first case, all entries from *conveyorStatus* and *productsMap* are reset, because all conveyors are empty. If a product leaves the production line, *updateEntriesAndExitsFile* and *updateStatistics* are called.
updateStatus	Is called when a product is added to the line, changes position or its processing ends. Updates *conveyorsStatus*, *productsMap* and, in the last case, *statistics*.
sendProductionPlanUp	Is invoked in the previous situations or when the production plan is created. Posts the content of *conveyorStatus* and *productionPlan* to the file through the API and prints the current statistics.
getNextProduct	Returns the ID of the next product's type and removes the respective product from *productionPlan*.
readyForNextProduct	Is called by a *Ticker Behaviour*, from Java Agent DEvelopment Framework (JADE), every 5 seconds. Invokes *updateProductionPlan* and returns *true*, if the entry conveyor is free and *productionPlan* is not empty, or false, if otherwise.

Aside from the ones described above, other simpler algorithms were developed in order to simplify the main methods, ensuring a cleaner overview of the *restClient* class. Their description is available in table 4.5.

Table 4.5: *restClient* class's auxiliary methods

Name	Description
fillProductTypesHashMap	Is called from the *restClient*'s constructor. Consults the ontology and fills in the *productTypes* hash map.
validateProductType	Returns *true*, if the given product type exists in *productTypes*, or *false*, if otherwise.
hasStation	Returns *true*, if the given conveyor has a station plugged into it, or *false*, if otherwise. The ResourceEntry and ResourceExit types are not considered for this.
getNewStatus	Creates and returns the status array for a specific conveyor, taking into account the given product type.
updateProductionPlan	Gets the production plan, through the API, and stores it in *productionPlan*.
updateStatistics	Updates the *statistics* array whenever a product either finishes being processed at a station or leaves the conveyor network.
updateEntriesAndExitsFile	Adds the given array, containing the entry and exit UNIX timestamps of the last-removed product, to a *csv* file, *entriesAndExits*.

4.4.2 The Production Plan GUI

At last, an interface that enabled the insertion of a production plan was lacking. Consequently, the GUI shown below was created.

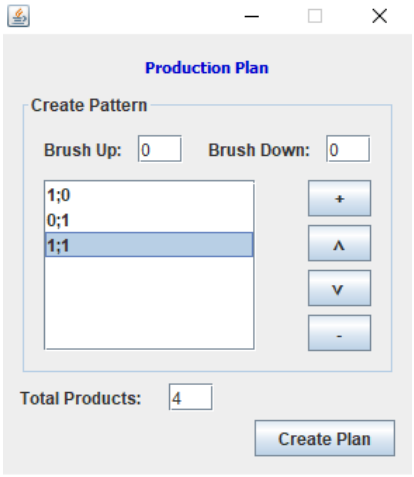


Figure 4.15: Production plan GUI

As portrayed by figure 4.15, this interface has two sections. The first one is dedicated to the creation of the production pattern, which is defined by inserting the products' types and sorting them into the desired order, whereas the second one is where the production plan size is defined and the instruction to actually create the plan is given.

In order to add a new product type, the amount of BrushUp and BrushDown skills has to be written in the respective text fields and, in the end, the "+" button has to be pressed. Note that, if the content of at least one of the text fields is not numeric, the product type is not inserted. In case the user inserts an undesired product type, it can be removed by selecting it and pressing the "-" button. In the example above, if this button was pressed, the last product type would be removed. It is also important to mention that the products' types do not have to be inserted in the correct order, given that there are two buttons which allow moving the selected product type up or down in the displayed pattern.

After describing the intended production pattern, the total amount of products to be inserted shall be written into the respective text field and, finally, the "Create Plan" button may be pressed. Similarly, if the content of the text field is not numeric or if the pattern contains an invalid product type according to the ontology, the production plan is not created. In the example from the previous figure, the production plan would consist of a complete pattern, i.e., ProductType1 → ProductType2 → ProductType3, followed by a product of type one.

The most important method from the class that implements the production plan GUI is *createPlanButtonActionPerformed*, which can only be invoked once per execution, since the button is disabled as soon as a valid production plan is inserted. Due to its role and complexity, the corresponding algorithm is provided below.

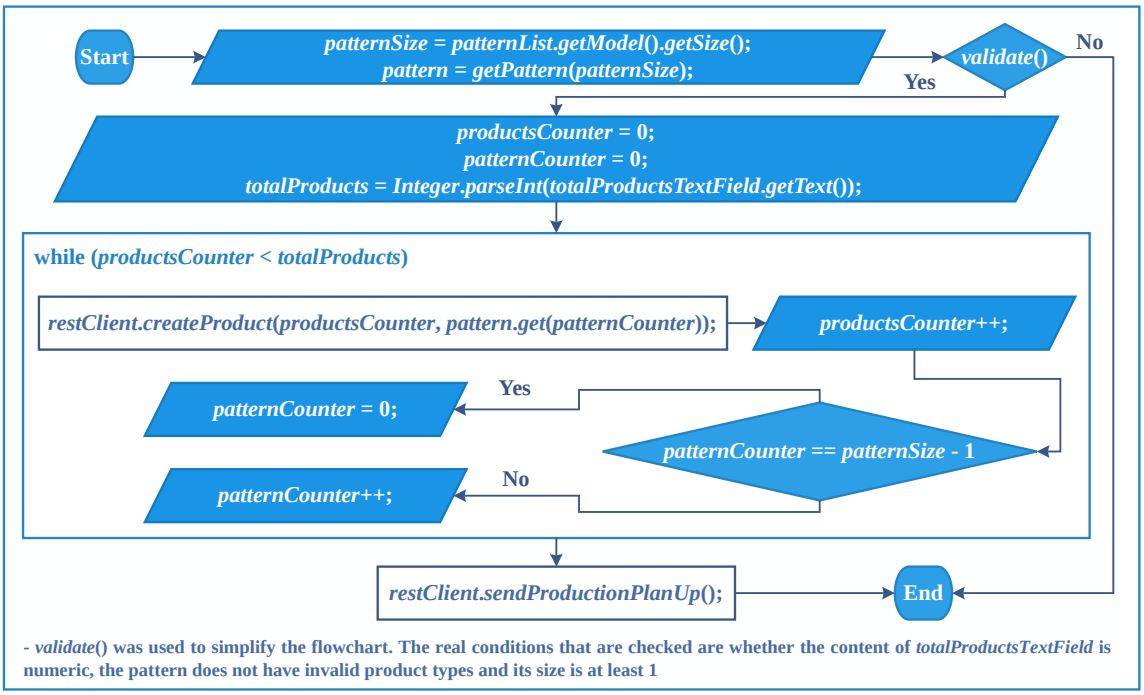


Figure 4.16: *createPlanButtonActionPeformed* method's algorithm

According to figure 4.16, the first stage involves obtaining the pattern's size and constitution. Then, as explained by the flowchart's footnote, the content of the text field and the pattern's size and elements are verified. In case the validation is unsuccessful, no more actions are taken. If otherwise, for each product to be added to the plan, the *createProduct* method from the *restClient* class is called. Ultimately, *sendProductionPlanUp* is invoked with the aim of sending the newly-created production plan to the *csv* file, introduced in section 4.1.

Chapter 5

Tests and Validation

This chapter intends to describe the method used for obtaining the results and validating the developed solution.

For that, the manufacturing unit simulator is firstly introduced from a practical point of view, i.e., the most relevant aspects that must be taken into consideration before using the simulator are summarised.

Then, the calibration methods are described, jointly with the conducted tests. Additionally, in the results analysis, not only are the pros of the suggested solution highlighted but also justifications for the least successful aspects are given.

5.1 The Manufacturing Unit Simulator

In order to validate the digital model, a source of reference values was needed. Given that the DT mirrors the physical system, the predictions may be compared with the values derived from the manufacturing unit simulator, shown in figure 5.1.

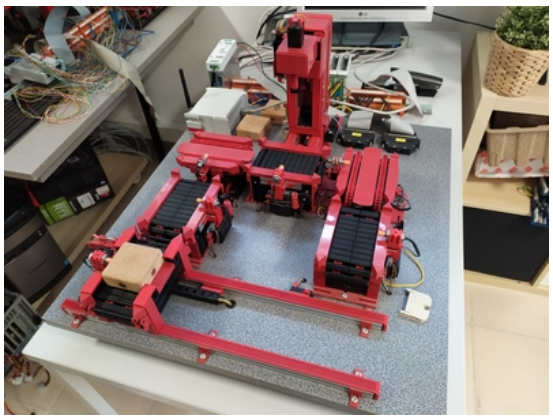


Figure 5.1: Manufacturing unit simulator

This production line, from Staudinger GMBH, encompasses one movable conveyor (A), which for this project is always located at the entry position (left-hand side of the picture), three unidirectional conveyors (B, D and F) and two conveyors with rotational movement (C and E), for being at the corners of the U-shaped network. The system also has a station, able to move back and forth as well as up and down, which is responsible for executing BrushUp and BrushDown skills. The figure below depicts the previous description.

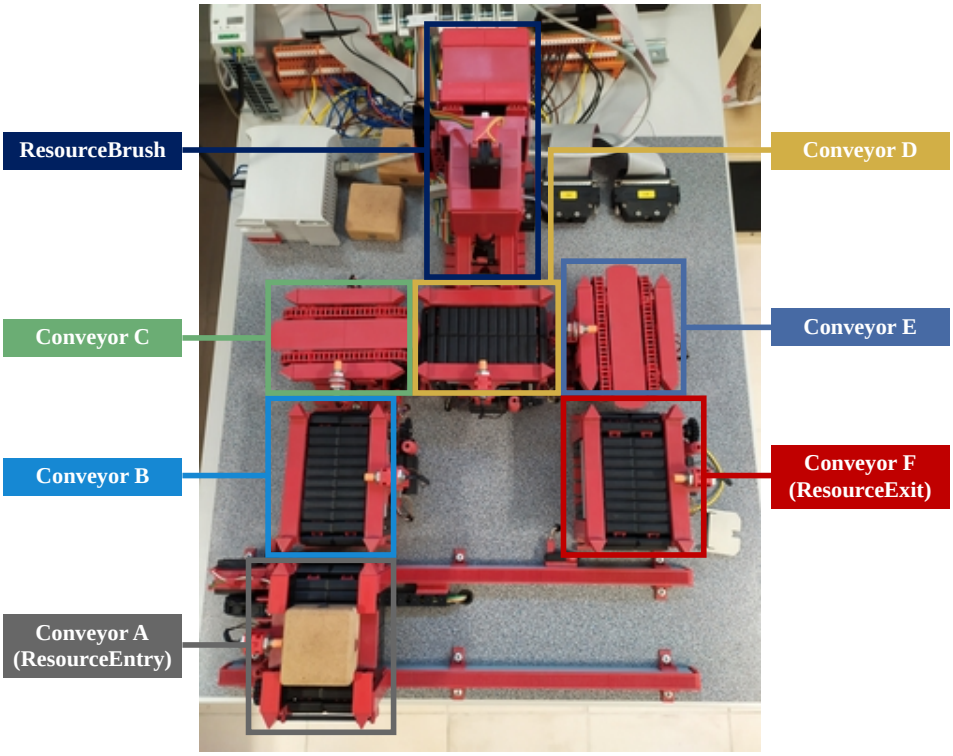


Figure 5.2: Manufacturing unit simulator's components

For each simulation, the ensuing steps must be followed. After executing the Java project and before anything else, a product has to be put on conveyor A. In fact, whenever this conveyor becomes empty, a new product must be added so that, when the next PA is launched, the product starts moving immediately and no delays are forced. Afterwards, the product travels from one conveyor to the following until it reaches the position of the ResourceBrush. Then, the due BrushUp and BrushDown skills are executed, after which the product resumes its journey towards the end of the network, i.e., conveyor F. By the time it gets there, the PA is instantly killed and the product must be manually removed.

Note that, in spite of the physical station being immovable, the ResourceBrush may be plugged into any conveyor, in the data model. If, for instance, this resource is associated with conveyor B, the product stops as soon as it arrives at that conveyor, where it is processed before resuming its natural course. However senseless in a real-life context, this nuance does not affect the obtained results because the processing delays are independent of the resource's location.

5.2 Results Analysis

5.2.1 First Calibration

While implementing the DT, the conveyors' speed and the processing time of each type of skill were arbitrarily set. Before obtaining the results, however, these parameters had to be tuned so that the model could accurately mirror the physical system.

In the first calibration, the following scenarios were considered.

Table 5.1: Scenarios considered in the first calibration

Scenario	Product Type	Total Products	Station Position
1	ProductType1 (1;0)	1	Conveyor D
2	ProductType2 (0;1)	1	Conveyor D
3	ProductType3 (1;1)	1	Conveyor D

As portrayed in table 5.1, all test cases are constituted by a single product because, that way, a product's behaviour can be analysed without it having to stop due to the next conveyor being busy.

The first parameters to be calibrated were the processing time of each skill, for which each scenario was run three times in the physical system. Given that the results fluctuated only a few tenths of a second, a value above the maximum obtained was chosen as the duration of the processing for the model, because that is the worst-case scenario, theoretically speaking. The results obtained for the kit and the ones that were chosen for the model are represented below.

Table 5.2: Calibration of the processing times

Scenario	Processing time (s)	
	Kit	Model
1	8.986	9.000
2	5.278	5.300
3	14.138	14.300

Secondly, the conveyors' speed was adjusted. For that, the previous tests were reused but, this time, focusing on the duration of the complete process. Once again, the highest values were taken as a reference, for corresponding to the worst case. Knowing the total duration of each test and the corresponding elapsed time at the station, the duration of the transport throughout the line was calculated. Notice that, in fact, the transport time is not only associated with the conveyors' speed because, according to this formula, any action which is not related to the station is assigned to the transport stage.

For that, it was necessary to get the model's transport time in order to assess whether the current conveyors' speed was too high or too low. Therefore, the same scenarios were tested and, using the previous formula, the transport time, in each case, was derived. All the simulation times are shown below.

Table 5.3: Simulation times with a station at conveyor D, before the first calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
1	17.206	8.986	26.192	13.714	9.000	22.714
2	17.107	5.278	22.385	13.714	5.300	19.014
3	16.900	14.138	31.038	13.714	14.000	28.014

Knowing that the initial conveyors' speed was 0.07 $m.s^{-1}$ and that the transport takes 13.714 s, in every scenario, the travelled distance was obtained.

$$d = v \times t \Rightarrow d = 0.07 \times 13.714 \approx 0.95998\ m$$

Given that the kit's transport delay is not the same in every scenario as it should, 17.3 s were considered. With the previous values, the adjusted conveyors' speed was calculated.

$$v = \frac{d}{t} \Rightarrow v = \frac{0.95998}{17.3} \approx 0.05549\ m.s^{-1}$$

With the aim of mitigating the possible communications delays, a slightly lower speed (0.055 $m.s^{-1}$) was selected.

Finally, the model's tests were repeated using the new velocity. The model's results and the reference values from the kit are shown below.

Table 5.4: Simulation times with a station at conveyor D, after the first calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
1	17.206	8.986	26.192	17.455	9.000	26.455
2	17.107	5.278	22.385	17.455	5.300	22.755
3	16.900	14.138	31.038	17.455	14.000	31.755

As expected, the model became a little slower than the kit for production plans with a single product. The pessimistic nature of the results may be explained by the fact that the calibration was carried out considering the worst-case scenarios.

5.2.2 First Analysis

So as to assess the simulation's accuracy, several longer and more complex tests were conducted.

Table 5.5: Scenarios considered in the first analysis

Scenario	Production Pattern	Total Products
1	0;1	5
2	0;1	20
3	1;0	5
4	1;0	20
5	1;1	5
6	1;1	20
7	0;1-1;0-1;1	10
8	0;1-1;0-1;1	20
9	0;1-1;1-1;0	10
10	0;1-1;1-1;0	20
11	1;0-0;1-1;1	10
12	1;0-0;1-1;1	20
13	1;0-1;1-0;1	10
14	1;0-1;1-0;1	20
15	1;1-0;1-1;0	10
16	1;1-0;1-1;0	20
17	1;1-1;0-0;1	10
18	1;1-1;0-0;1	20

Each of the above-described scenarios was tested six times, one for each possible station's location, which adds up to one hundred and eight runs. Once again, for each test case the total processing time and two subdivisions of this value, transport time (T) and station time (S), were registered. Additionally, the two KPIs below were extracted.

- Cycle Time (CT) - average time spent by products on the line since they arrive until they are removed, in $s/product$.

$$CT = \frac{\sum(exitTime - entryTime)}{exportedProducts}$$

- Throughput (Tp) - average amount of exported products per time unit, in $products/min$.

$$Tp = \frac{exportedProducts}{simulationEnd - simulationStart}$$

The results obtained with the station plugged into conveyor A are represented below.

Table 5.6: Results with a station at conveyor A, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	43.44	44.63	88.07	25.83	3.41	24.73	45.00	69.73	26.45	4.30
2	143.43	185.06	328.49	25.99	3.65	52.00	180.00	232.00	26.45	5.17
3	37.55	26.22	63.77	22.06	4.70	24.73	26.50	51.23	22.75	5.86
4	115.91	104.23	220.14	22.64	5.45	52.00	106.00	158.00	22.75	7.59
5	40.90	78.59	119.49	32.33	2.51	24.73	71.50	96.23	31.75	3.12
6	142.91	284.50	427.41	30.89	2.81	52.00	286.00	338.00	31.75	3.55
7	73.24	93.40	166.64	26.01	3.60	33.82	94.80	128.62	26.93	4.66
8	137.46	187.90	325.36	25.95	3.69	52.00	185.90	237.90	26.75	5.04
9	73.10	93.60	166.70	25.93	3.60	33.82	94.80	128.62	26.93	4.66
10	136.87	191.86	328.73	26.07	3.65	52.00	194.90	246.90	27.20	4.86
11	73.53	89.83	163.36	25.62	3.67	33.82	91.10	124.92	26.56	4.80
12	135.16	183.09	318.25	25.78	3.77	52.00	185.90	237.90	26.75	5.04
13	73.27	92.97	166.24	25.82	3.61	33.82	91.10	124.92	26.56	4.80
14	137.51	191.12	328.63	26.06	3.65	52.00	191.20	243.20	27.01	4.93
15	72.79	98.59	171.38	26.37	3.50	33.82	100.10	133.92	27.46	4.48
16	137.28	191.82	329.10	26.13	3.65	52.00	194.90	246.90	27.20	4.86
17	73.33	98.60	171.93	26.39	3.49	33.82	100.10	133.92	27.46	4.48
18	137.26	188.40	325.66	26.29	3.68	52.00	191.20	243.20	27.01	4.93

When taking into account the time fractions, it becomes clear that the transport is the main contributor for the differences between the predictions and the kit's values. As to the station times, the predictions are considerably accurate, particularly if we bear in mind the amount of products involved. Nevertheless, it is interesting to see that, in spite of the station's processing being slightly slower in the model, it ends up being faster.

Overall, the model is surprisingly optimistic, even though the conveyors are set to run at an inferior velocity. For this reason, we can infer that increasing the total amount of products leads to larger discrepancies between the predictions and the physical system, possibly because of the amount of communications established between agents.

Regarding the KPIs, the predicted cycle time is pessimistic but its relative error never exceeds 5%, meaning that the communications delays compensate for the lower speed of the model's conveyors. On the other hand, the throughput presents the least precise results. Since the station is placed at the entry conveyor, the delay caused by the presence of a product on conveyor A highly influences the overall process time as no products may be added before the entry conveyor becomes free.

Afterwards, the station was moved to conveyor B and the tests were rerun.

Table 5.7: Results with a station at conveyor B, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	39.76	44.41	84.17	32.94	3.56	35.64	45.00	80.64	35.20	3.72
2	126.09	177.65	303.74	34.75	3.95	103.82	180.00	283.82	37.44	4.23
3	39.58	26.08	65.66	26.94	4.57	35.64	26.50	62.14	28.54	4.83
4	126.93	103.56	230.49	27.74	5.21	103.82	106.00	209.82	30.22	5.72
5	39.67	70.23	109.90	42.19	2.73	35.64	71.50	107.14	44.74	2.80
6	126.14	288.47	414.61	45.26	2.89	103.82	286.00	389.82	47.77	3.08
7	68.54	93.35	161.89	35.06	3.71	58.36	94.80	153.16	37.65	3.92
8	125.65	182.50	308.15	35.49	3.89	103.82	185.90	289.72	38.21	4.14
9	68.48	98.30	166.78	36.25	3.60	58.36	94.80	153.16	37.65	3.92
10	126.52	191.78	318.30	35.64	3.77	103.82	194.90	298.72	38.66	4.02
11	68.49	90.96	159.45	35.47	3.76	58.36	91.10	149.46	37.28	4.01
12	126.85	188.64	315.49	35.49	3.80	103.82	185.90	289.72	38.03	4.14
13	68.54	89.55	158.09	34.66	3.80	58.36	91.10	149.46	37.28	4.01
14	126.83	191.68	318.51	36.02	3.77	103.82	191.20	295.02	38.29	4.07
15	69.05	103.06	172.11	36.19	3.49	58.36	100.10	158.46	38.18	3.79
16	126.20	200.95	327.15	36.90	3.67	103.82	194.90	298.72	38.93	4.02
17	68.29	98.44	166.73	35.68	3.60	58.36	100.10	158.46	38.18	3.79
18	126.41	187.60	314.01	35.99	3.82	103.82	191.20	295.02	38.74	4.07

By moving the station to conveyor B, the simulation globally becomes more accurate. In the first place, the relative error of the total time is always below 10%, possibly because, while the product is being processed on conveyor B, a new product may be added to conveyor A. That way, by the time conveyor B becomes empty, the product is ready to proceed. Furthermore, the relative error of the station time does not exceed 5% and, consequently, the transport time is also closer to the kit's.

On the other hand, the predicted cycle times are worse. This happens because this metric is counted from the product's insertion. Given that the model's conveyors are slower and the station is on conveyor B, while a product is being processed a new one is added to the network and the cycle time starts counting. Logically, the predicted throughputs are more exact, because they only depend on the total amount of products and on the elapsed time, which, in this case, closely match the real world's.

Then, with the station plugged into conveyor C, the results below were obtained.

Table 5.8: Results with a station at conveyor C, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	44.04	44.38	88.42	40.25	3.39	37.09	45.00	82.09	40.38	3.65
2	147.44	177.15	324.59	45.27	3.70	110.73	180.00	290.73	46.03	4.13
3	44.44	25.99	70.43	31.56	4.26	37.09	26.50	63.59	31.50	4.72
4	147.79	103.75	251.54	34.40	4.77	110.73	106.00	216.73	35.48	5.54
5	42.70	70.16	112.86	50.30	2.66	37.09	71.50	108.59	53.10	2.76
6	147.87	280.22	428.09	60.00	2.80	110.73	286.00	396.73	61.13	3.02
7	78.05	92.88	170.93	43.82	3.51	61.64	94.80	156.44	45.06	3.84
8	145.79	183.36	329.15	44.93	3.65	110.73	185.90	296.63	47.28	4.05
9	78.63	92.82	171.45	45.22	3.50	61.64	94.80	156.44	45.96	3.84
10	146.28	190.52	336.80	46.25	3.56	110.73	194.90	305.63	47.73	3.93
11	87.41	89.14	176.55	46.04	3.40	61.64	91.10	152.74	44.69	3.93
12	147.08	182.09	329.17	45.47	3.65	110.73	185.90	296.63	47.10	4.05
13	78.89	89.09	167.98	44.38	3,57	61.64	91.10	152.74	45.22	3.93
14	146.62	186.84	333.46	45.60	3.60	110.73	191.20	301.93	47.36	3.97
15	78.67	105.92	184.59	48.49	3.25	61.64	100.10	161.74	46.49	3.71
16	147.89	196.59	344.48	47.76	3.48	110.73	194.90	305.63	48.00	3.93
17	78.06	97.85	175.91	44.57	3.41	61.64	100.10	161.74	46.12	3.71
18	147.42	187.04	334.46	46.74	3.59	110.73	191.20	301.93	47.81	3.97

The results above show that, with the accumulation of products on the line, the pessimistic approach adopted in the first calibration becomes insufficient to compensate for the communications delays. The total time and, consequently, the throughput predicted by the model start deviating considerably from the theoretical values and only the predicted cycle time is fairly accurate. However, the latter shall not be overrated since the prediction is the arithmetical average of a range of values with high standard deviation, i.e., some products have excessively low cycle times whereas others have excessively high ones, which results in a falsely accurate average.

After that, the station was placed at conveyor D and the tests were repeated. The corresponding results are shown below.

Table 5.9: Results with a station at conveyor D, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	45.17	44.53	89.70	44.23	3.34	35.64	45.00	80.64	42.58	3.72
2	151.16	183.80	334.96	54.09	3.58	103.82	180.00	283.82	52.51	4.23
3	45.34	26.14	71.48	33.16	4.20	35.64	26.50	62.14	32.22	4.83
4	151.02	104.06	255.08	38.94	4.70	103.82	106.00	209.82	38.82	5.72
5	44.19	70.47	114.66	57.62	2.62	35.64	71.50	107.14	57.42	2.80
6	151.46	281.31	432.77	71.79	2.77	103.82	286.00	389.82	72.12	3.08
7	80.44	93.30	173.74	51.12	3.45	58.36	94.80	153.16	50.43	3.92
8	150.33	186.13	336.46	55.19	3.57	103.82	185.90	289.72	53.98	4.14
9	80.83	93.57	174.40	52.08	3.44	58.36	94.80	153.16	51.33	3.92
10	157.17	191.95	349.12	57.30	3.44	103.82	194.90	298.72	54.88	4.02
11	80.22	89.61	169.83	50.16	3.53	58.36	91.10	149.46	49.69	4.01
12	155.36	182.94	338.30	55.04	3.55	112.82	185.90	298.72	53.79	4.14
13	80.15	89.70	69.85	50.59	3.53	58.36	91.10	149.46	50.22	4.01
14	160.49	188.04	348.53	56.45	3.44	103.82	191.20	295.02	54.32	4.07
15	81.11	98.57	179.68	53.07	3.34	58.36	100.10	158.46	52.39	3.79
16	157.95	191.68	349.63	56.62	3.43	103.82	194.90	298.72	55.14	4.02
17	80.69	98.50	179.19	52.68	3.35	58.36	100.10	158.46	52.02	3.79
18	151.06	188.02	339.08	55.01	3.54	103.82	191.20	295.02	54.77	4.07

The results obtained with the station plugged into conveyor D reinforce the justifications provided during the analysis of the test case in which the station was located on conveyor C. As expected, the total time presents a larger deviation because it is now possible to accumulate three products on the line, while another one is being processed on conveyor D. Therefore, the throughput predictions exhibit a higher relative error whereas the cycle time forecasts remain falsely accurate, with relative errors below 5%.

Subsequently, the same scenarios were examined with the station at conveyor E.

Table 5.10: Results with a station at conveyor E, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	45.64	44.30	89.94	46.26	3.34	37.09	45.00	82.09	44.27	3.65
2	155.40	177.20	332.60	60.63	3.61	110.73	180.00	290.73	60.77	4.13
3	46.10	25.95	72.05	35.39	4.16	37.09	26.50	63.59	33.17	4.72
4	156.13	103.52	259.65	44.29	4.62	110.73	106.00	216.73	44.12	5.54
5	46.17	70.19	116.36	62.12	2.58	37.09	71.50	108.59	60.17	2.76
6	157.42	280.53	437.95	84.35	2.74	110.73	286.00	396.73	84.62	3.02
7	83.03	93.20	176.23	57.72	3.40	61.64	94.80	156.44	56.82	3.84
8	157.40	182.42	339.82	62.92	3.53	110.73	185.90	296.63	62.64	4.05
9	83.35	93.12	176.47	58.93	3.40	61.64	94.80	156.44	57.72	3.84
10	156.21	196.73	352.94	64.56	3.40	110.73	194.90	305.63	63.54	3.93
11	83.57	89.55	173.12	57.14	3.47	61.64	91.10	152.74	56.08	3.93
12	156.97	182.55	339.52	62.04	3.53	110.73	185.90	296.63	62.27	4.05
13	83.48	89.54	173.02	58.14	3.47	61.64	91.10	152.74	56.61	3.93
14	157.36	187.65	345.01	62.94	3.48	110.73	191.20	301.93	62.80	3.97
15	82.79	102.81	185.60	62.07	3.23	61.64	100.10	161.74	58.78	3.71
16	155.04	198.34	353.38	65.23	3.40	110.73	194.90	305.63	64.07	3.93
17	82.47	98.28	180.75	58.87	3.32	61.64	100.10	161.74	58.41	3.71
18	156.93	187.96	344.89	63.48	3.48	110.73	191.20	301.93	63.70	3.97

The results from table 5.10 show that the station's transition from conveyor D to E is a turning point in the system's behaviour. Unlike what happened when the station changed from conveyor C to D, the total time's error did not increase. The main reason is that, by advancing the station on the network, the products accumulation takes longer to start which, as already mentioned, is the main cause of the increasing delays associated with the communications between agents. Thus, the model retards in relation to the physical system until the accumulation begins, by which time the kit starts losing the advantage and ends up being globally slower.

Finally, the scenarios were tested with the station plugged into conveyor F. The obtained results are represented in table 5.11.

Table 5.11: Results with a station at conveyor F, after the first calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	31.19	44.24	75.43	39.01	3.98	32.00	45.00	77.00	41.73	3.90
2	85.83	185.02	270.85	58.56	4.13	86.65	180.00	266.65	61.90	4.50
3	30.94	26.04	56.98	28.07	5.26	32.00	26.50	58.50	30.63	5.13
4	85.24	103.35	188.59	31.35	6.36	86.55	106.00	192.55	42.46	6.23
5	31.12	70.00	101.12	54.48	2.97	32.00	71.50	103.50	57.63	2.90
6	85.18	289.74	374.92	81.29	3.20	86.55	286.00	372.55	89.73	3.22
7	48.77	92.74	141.51	48.44	4.24	50.18	94.80	144.98	56.23	4.14
8	85.83	187.91	273.74	54.98	4.38	86.55	185.90	272.45	64.17	4.40
9	49.25	92.76	142.01	49.97	4.23	50.18	94.80	144.98	58.03	4.14
10	86.02	200.18	286.20	56.89	4.19	86.55	194.90	281.45	65.07	4.26
11	49.17	89.00	138.17	46.21	4.34	50.18	91.10	141.28	55.46	4.25
12	85.44	181.55	266.99	52.49	4.49	86.55	185.90	272.45	63.78	4.40
13	54.86	88.97	143.83	48.95	4.17	50.18	91.10	141.28	56.52	4.25
14	85.41	195.55	280.96	54.78	4.27	86.55	191.20	277.75	64.31	4.32
15	49.02	100.41	149.43	51.03	4.02	50.18	100.10	150.28	59.09	3.99
16	85.31	190.40	275.71	53.91	4.35	86.55	194.90	281.45	65.60	4.26
17	53.59	97.72	151.31	51.64	3.97	50.18	100.10	150.28	58.35	3.99
18	84.86	186.74	271,60	53.85	4.42	86.55	191.20	277.75	65.23	4.32

These results support the hypothesis put forward during the analysis of the previous case, in which the station was plugged into conveyor E. In this case, the relative errors associated with the total time and the throughput predictions do not exceed 3% and 10%, respectively. As a consequence of the station being at the exit conveyor, the products accumulation is less likely and takes longer to occur, meaning that the amount of simultaneous communications is reduced and the model's accuracy increases.

5.2.3 Second Calibration

The unexpected differences between the real values and the model's predictions led to the need of a second calibration.

In the first place and for simplicity's sake, only two possible locations for the station, conveyors B or D, were considered hereafter. Besides, given the different types of responses from the system depending on the station's position, distinct conveyors' speeds where chosen in accordance with this parameter. For this stage, only scenarios 8 and 18 from table 5.5 were used.

Before the second tuning, the simulation times with the station at conveyor B were the following.

Table 5.12: Simulation times with a station at conveyor B, before the second calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
8	125.65	182.50	308.15	103.82	185.90	289.72
18	126.41	187.60	314.01	103.82	191.20	295.02

Through a trial-and-error process, the ideal speed of 0.0467 $m.s^{-1}$ for the conveyors was determined and the results below were obtained.

Table 5.13: Simulation times with a station at conveyor B, after the second calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
8	125.65	182.50	308.15	122.27	185.90	308.17
18	126.41	187.60	314.01	122.27	191.20	313.47

On the other hand, the simulation times with the station plugged into conveyor D, prior to this stage, are summarised below.

Table 5.14: Simulation times with a station at conveyor D, before the second calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
8	150.33	186.13	336.46	103.82	185.90	289.72
18	151.06	188.02	339.08	103.82	191.20	295.02

Following the same method, the ideal speed of 0.038 $m.s^{-1}$ was chosen, which led to the results from table 5.15.

Table 5.15: Simulation times with a station at conveyor D, after the second calibration

Scenario	Time (s)					
	Kit			Model		
	Transport	Station	Total	Transport	Station	Total
8	150.33	186.13	336.46	150.26	185.90	336.16
18	151.06	188.02	339.08	150.26	191.20	341.46

5.2.4 Second Analysis

With the aim of attesting the impacts of the second calibration on the model's performance, all scenarios from table 5.5 were repeated using the new conveyors' speed.

The table below contains the obtained results with the station plugged into conveyor B. Notice that, since the calibration was only applied to the model, the kit's results are the same from table 5.7.

Table 5.16: Results with a station at conveyor B, after the second calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	39.76	44.41	84.17	32.94	3.56	41.97	45.00	86.97	38.75	3.45
2	126.09	177.65	303.74	34.75	3.95	122.27	180.00	302.27	41.02	3.97
3	39.58	26.08	65.66	26.94	4.57	41.97	26.50	68.47	32.09	4.38
4	126.93	103.56	230.49	27.74	5.21	122.27	106.00	228.27	33.80	5.26
5	39.67	70.23	109.90	42.19	2.73	41.97	71.50	113.47	48.29	2.64
6	126.14	288.47	414.61	45.26	2.89	122.27	286.00	408.27	51.35	2.94
7	68.54	93.35	161.89	35.06	3.71	68.74	94.80	163.54	41.22	3.67
8	125.65	182.50	308.15	35.49	3.89	122.27	185.90	308.17	41.79	3.89
9	68.48	98.30	166.78	36.25	3.60	68.74	94.80	163.54	41.22	3.67
10	126.52	191.78	318.30	35.64	3.77	122.27	194.90	317.17	42.24	3.78
11	68.49	90.96	159.45	35.47	3.76	68.74	91.10	159.84	40.85	3.75
12	126.85	188.64	315.49	35.49	3.80	122.27	185.90	308.17	41.61	3.89
13	68.54	89.55	158.09	34.66	3.80	68.74	91.10	159.84	40.85	3.75
14	126.83	191.68	318.51	36.02	3.77	122.27	191.20	313.47	41.87	3.83
15	69.05	103.06	172.11	36.19	3.49	68.74	100.10	168.84	41.75	3.55
16	126.20	200.95	327.15	36.90	3.67	122.27	194.90	317.17	42.51	3.78
17	68.29	98.44	166.73	35.68	3.60	68.74	100.10	168.84	41.75	3.55
18	126.41	187.60	314.01	35.99	3.82	122.27	191.20	313.47	42.32	3.83

Even though, after the first adjustments, the results with a station at conveyor B were the most accurate of all, the second calibration was still able to improve the model's accuracy for this case. The total time and the throughput predictions became, thus, limited to 5%. On the other hand, the cycle time's relative error increased and, in some situations, is now above 20%. Such fact is mainly due to the decrease in the conveyors' velocity, which improves the predictions of the total time and the throughput at the cost of enlarging each product's cycle time.

Finally, the station was moved back to conveyor D and the same experiments were run. The reference values from the kit and the corresponding model's predictions are presented below.

Table 5.17: Results with a station at conveyor D, after the second calibration

Scenario	Kit					Model				
	Time (s)			CT	Tp	Time (s)			CT	Tp
	T	S	Total			T	S	Total		
1	45.17	44.53	89.70	44.23	3.34	51.58	45.00	96.58	52.78	3.11
2	151.16	183.80	334.96	54.09	3.58	150.26	180.00	330.26	61.71	3.63
3	45.34	26.14	71.48	33.16	4.20	51.58	26.50	78.08	42.42	3.84
4	151.02	104.06	255.08	38.94	4.70	150.26	106.00	256.26	48.02	4.68
5	44.19	70.47	114.66	57.62	2.62	51.58	71.50	123.08	67.62	2.44
6	151.46	281.31	432.77	71.79	2.77	150.26	286.00	436.26	81.32	2.75
7	80.44	93.30	173.74	51.12	3.45	84.47	94.80	179.27	59.96	3.35
8	150.33	186.13	336.46	55.19	3.57	150.26	185.90	336.16	63.18	3.57
9	80.83	93.57	174.40	52.08	3.44	84.47	94.80	179.27	60.86	3.35
10	157.17	191.95	349.12	57.30	3.44	150.26	194.90	345.16	64.08	3.48
11	80.22	89.61	169.83	50.16	3.53	84.47	91.10	175.57	59.22	3.42
12	155.36	182.94	338.30	55.04	3.55	150.26	185.90	336.16	62.99	3.57
13	80.15	89.70	169.85	50.59	3.53	84.47	91.10	175.57	59.75	3.42
14	160.49	188.04	348.53	56.45	3.44	150.26	191.20	341.46	63.52	3.51
15	81.11	98.57	179.68	53.07	3.34	84.47	100.10	184.57	61.92	3.25
16	157.95	191.68	349.63	56.62	3.43	150.26	194.90	345.16	64.34	3.48
17	80.69	98.50	179.19	52.68	3.35	84.47	100.10	184.57	61.55	3.25
18	151.06	188.02	339.08	55.01	3.54	150.26	191.20	341.46	63.97	3.51

The first results for this scenario were far from accurate. For this reason, the second calibration played an important role improving the predictions to some extent. For instance, the relative error of the total time and the throughput were limited to approximately 10%. Still, the average cycle time increased and, in some scenarios, its relative error surpasses 30%. Once again, this happens because the conveyors' speed had to be harshly decreased in order to counteract the communications delays.

5.2.5 Final Validation

In spite of the influence of the calibration process in the model's accuracy, the results failed to fulfill the initial expectations. As already discussed, the products' accumulation leads to an exponential increase in the amount of communications between agents and, consequently, impairs the overall system's behaviour.

In spite of being possible to adjust the total time of a specific production plan, it is achieved by reducing the conveyors' velocity. Thus, the second calibration, regardless of enabling the model to predict the KPIs for production plans with twenty products, spoils each product's cycle time. Moreover, the station's delay varies between executions and it is not correct to compensate such differences by changing the conveyors' speed. Taking the aforementioned into account, it is safe to assume that it is impossible to accurately predict all of the performance indicators by simply changing the conveyors' velocity.

Since modelling the delays transcends the scope of this project, a last verification was performed so as to ascertain whether processing a sufficiently large sum of products helps dissimulating such latency. For that, the scenarios used in the second calibration were rerun with larger production plans, but the KPIs' calculations only considered sets of twenty consecutive products.

The predictions for the last twenty out of forty or sixty products, with a station at conveyor B, are shown below.

Table 5.18: Results with a station at conveyor B, for the last twenty products out of forty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	318.06	36.07	3.77	320.54	42.82	3.74
18	339.54	37.09	3.53	320.00	42.52	3.75

Table 5.19: Results with a station at conveyor B, for the last twenty products out of sixty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	324.18	36.18	3.70	318.63	42.27	3.77
18	321.48	35.58	3.73	321.92	42.44	3.73

A closer look at the previous values suggests that, regardless of the growth of the production plan, the predictions' relative error does not increase and, sometimes, it may even decrease.

The results for the last twenty products out of forty, with a station at conveyor D, are represented below.

Table 5.20: Results with a station at conveyor D, for the last twenty products out of forty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	376.17	57.74	3.19	342.30	61.39	3.51
18	372.39	57.77	3.22	340.41	61.30	3.53

For the last twenty products out of sixty, with a station at conveyor D, the following predictions were obtained.

Table 5.21: Results with a station at conveyor D, for the last twenty products out of sixty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	364.10	57.25	3.30	339.13	61.24	3.54
18	359.65	57.15	3.34	333.38	60.97	3.59

Once more, the increase in the production plan's dimension did not lead to least accurate predictions and, sometimes, the results were better for the longer plan.

Finally, for the curiosity sake and since the necessary values were available, the KPIs for the twenty intermediate products out of sixty were calculated. The corresponding results are summarised in tables 5.22 and 5.23.

Table 5.22: Results with a station at conveyor B, for the twenty intermediate products out of sixty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	333.57	37.38	3.60	320.54	42.82	3.74
18	330.72	36.74	3.63	320.00	42.52	3.75

Table 5.23: Results with a station at conveyor D, for the twenty intermediate products out of sixty

Scenario	Kit			Model		
	Total Time (s)	CT	Tp	Total Time (s)	CT	Tp
8	369.35	57.51	3.25	342.30	61.39	3.51
18	369.37	56.96	3.25	340.41	61.30	3.53

At long last, these results were compared with the values obtained for the last twenty products out of forty in the corresponding test cases (tables 5.18 and 5.20) with the aim of checking their proximity, since they represent the same practical situation. The mismatches derived from this confrontation indicate that the kit will often originate different results for the exact same test case. The most likely cause is the casual network instability which may be exacerbated by the affluence of communications between agents depending on the amount of products present on the conveyor network.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The usage of DTs is still at an early phase. As a result, standardised implementations that can be easily adapted to different situations are lacking. With the aim of covering this major gap, the need to implement a digital model of a physical system was taken to a whole new level. Aside from developing the model itself, the architecture had to be pondered so that its specificity evenly allowed future implementations to be based on it.

The suggested architecture focuses on logically dividing each component of a CPS, i.e., there is a layer fully dedicated to the physical entities, one which embraces the digital components and a layer responsible for interconnecting them. Moreover, none of the used technologies narrows down the range of target-problems, given that widely-compatible programming languages and data formats were opted for. This way, it is believed that a considerable percentage of the oncoming solutions will be safe to rely on an architecture along the lines of the one hereby introduced.

Another great contribution of this project was provided by the detailed validation process carried out. Thanks to that, not only did it become clear that the developed model is capable of anticipating, to some extent, the physical system's behavioural trends but also useful tips for future work could be deduced from the least successful predictions.

To sum up, as foreseeable as it could be, the discovered research gaps were not fully covered. On the one hand, the results are highly dependent on the network's stability. Furthermore, the fact that the physical entity is a MAS implies that the total messages exchanged increases exponentially with the amount of products being simultaneously processed. Thus, in some cases, the results are inevitably worse, possibly due to the communication delays. Nonetheless, the aforementioned breakthroughs may well be the cornerstone of future research on the topic.

6.2 Future Work

As future work, two major aspects of the DT were left with room for improvement: the DT's overall performance and its ease of using.

In the first place, the analysis of the obtained results shed a light on the cause of the not-so-accurate predictions. Consequently, a crucial hint for improving the model's accuracy emerged. Since the communication delays were found to be the main cause of failure, the idea of modelling the actual delays, considering the amount of products being processed simultaneously, arose. Expectedly, the results will benefit from the model's ability to take such important factor into consideration.

Furthermore, the digital model is unable to deal with changes in the resource's location halfway through the simulation. Given that the manufacturing unit simulator has this feature, it would be valuable to provide the DT with such capability.

Finally, the DT's usage may become slightly more intuitive. Thus, it is advisable that the developed model be exported as a stand-alone app so that, in order to use it, people need not install AnyLogic. Additionally, if desired, the exported application may be launched from the NetBeans project that controls the MAS. Thereafter, the DT becomes attached to its physical counterpart's implementation.

Bibliography

[1] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A. Y. Nee. "Enabling technologies and tools for digital twin." In: *Journal of Manufacturing Systems* October (2019). ISSN: 02786125. DOI: 10.1016/j.jmsy.2019.10.001. URL: https://doi.org/10.1016/j.jmsy.2019.10.001.

[2] V Parthasarathi and G Thilagavathi. "Synthesis and characterization of zinc oxide nanopartilce and its application on fabrics for microbe resistant defence clothing." In: *International Journal of Pharmacy and Pharmaceutical Sciences* 3.4 (2011), pp. 392–398.

[3] S.-W. Hung, A.-P. Wang, and C.-C. Chang. "Exploring the evolution of nano technology." In: *2012 Proceedings of PICMET'12: Technology Management for Emerging Technologies*. IEEE. 2012, pp. 2598–2604.

[4] A. Raj, G. Dwivedi, A. Sharma, A. Beatriz, and L. D. Sousa. "International Journal of Production Economics Barriers to the adoption of industry 4 . 0 technologies in the manufacturing sector : An inter-country comparative perspective." In: *International Journal of Production Economics* August 2018 (2019), pp. 107–546. ISSN: 0925-5273. DOI: 10.1016/j.ijpe.2019.107546. URL: https://doi.org/10.1016/j.ijpe.2019.107546.

[5] G. Büchi, M. Cugno, and R. Castagnoli. "Smart factory performance and Industry 4.0." In: *Technological Forecasting and Social Change* 150.June 2019 (2020), pp. 119–790. ISSN: 00401625. DOI: 10.1016/j.techfore.2019.119790. URL: https://doi.org/10.1016/j.techfore.2019.119790.

[6] A. P. T. Pacchini, W. C. Lucato, F. Facchini, and G. Mummolo. "The degree of readiness for the implementation of Industry 4.0." In: *Computers in Industry* 113 (2019), pp. 103–125. ISSN: 01663615. DOI: 10.1016/j.compind.2019.103125. URL: https://doi.org/10.1016/j.compind.2019.103125.

[7] L. Li. "China's manufacturing locus in 2025: With a comparison of "Made-in-China 2025" and "Industry 4.0"." In: *Technological Forecasting and Social Change* 135 (2018), pp. 66–74. ISSN: 0040-1625. DOI: https://doi.org/10.1016/j.techfore.2017.05.028. URL: http://www.sciencedirect.com/science/article/pii/S0040162517307254.

[8] K. Schwab. *The Fourth Industrial Revolution*. Crown Publishing Group, 2017. ISBN: 9781524758875. URL: https://books.google.pt/books?id=ST_FDAAAQBAJ.

[9] G. Li, Y. Hou, and A. Wu. "Fourth Industrial Revolution: technological drivers, impacts and coping methods." In: *Chinese Geographical Science* 27.4 (2017), pp. 626–637. ISSN: 1993-064X. DOI: 10.1007/s11769-017-0890-x. URL: https://doi.org/10.1007/s11769-017-0890-x.

[10] E. Negri, L. Fumagalli, and M. Macchi. "A review of the roles of digital twin in cps-based production systems." In: *Procedia Manufacturing* 11 (2017), pp. 939–948.

[11] S. Weyer, M. Schmitt, M. Ohmer, and D. Gorecky. "Towards Industry 4.0-Standardization as the crucial challenge for highly modular, multi-vendor production systems." In: *Ifac-Papersonline* 48.3 (2015), pp. 579–584.

[12] Y. Lu and X. Xu. "Resource virtualization: A core technology for developing cyber-physical production systems." In: *Journal of Manufacturing Systems* 47.April (2018), pp. 128–140. ISSN: 02786125. DOI: 10.1016/j.jmsy.2018.05.003. URL: https://doi.org/10.1016/j.jmsy.2018.05.003.

[13] F. Chiarello, L. Trivelli, A. Bonaccorsi, and G. Fantoni. "Extracting and mapping industry 4.0 technologies using wikipedia." In: *Computers in Industry* 100 (2018), pp. 244–257.

[14] H. Ç. Bal and Ç. Erkan. "Industry 4.0 and Competitiveness." In: *Procedia Computer Science* 158 (2019), pp. 625–631. ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.09.096. URL: https://doi.org/10.1016/j.procs.2019.09.096.

[15] K. W. Chun, H. Kim, and K. Lee. "A Study on Research Trends of Technologies for Industry 4.0; 3D Printing, Artificial Intelligence, Big Data, Cloud Computing, and Internet of Things." In: *Advanced Multimedia and Ubiquitous Engineering*. Springer, 2018, pp. 397–403.

[16] S. Vaidya, P. Ambad, and S. Bhosle. "Industry 4.0–a glimpse." In: *Procedia Manufacturing* 20 (2018), pp. 233–238.

[17] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman. "Intelligent manufacturing in the context of industry 4.0: a review." In: *Engineering* 3.5 (2017), pp. 616–630.

[18] G. Stoneburner, A. Goguen, and A. Feringa. "Risk Management Guide forInformation Technology Systems." In: *NIST Special Publication* (2002), pp. 800–30.

[19] M. L. Angelo Corallo, Mariangela Lazoi. "Cybersecurity in the context of industry 4.0: A structured classification of critical assets and business impacts." In: *Computers in Industry* 114 (2019), pp. 1–15. ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2019.103165. URL: https://doi.org/10.1016/j.compind.2019.103165.

[20] F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl. "Industry 4.0 – An Introduction in the phenomenon." In: *IFAC-PapersOnLine* 49.25 (2016), pp. 8–12. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.12.002. URL: http://dx.doi.org/10.1016/j.ifacol.2016.12.002.

[21] T Bangemann, C Bauer, H Bedenbender, M Diesner, U Epple, F Elmas, J Friedrich, T Goldschmidt, F Göbe, S Grüner, et al. "Industrie 4.0-Technical Assets: Basic terminology concepts life cycles and administration models." In: *VDI/VDE and ZVEI* (2016).

[22] O Rehse, S Hoffman, and C Kosanke. "Tapping into the Transformative Power of Service 4.0." In: *The Boston Consulting Group* (2016).

[23] J. Zeng, L. T. Yang, M. Lin, H. Ning, and J. Ma. "A survey: Cyber-physical-social systems and their system-level design methodology." In: *Future Generation Computer Systems* (2016).

[24] R. Alguliyev, Y. Imamverdiyev, and L. Sukhostat. "Cyber-physical systems and their security issues." In: *Computers in Industry* 100.July 2017 (2018), pp. 212–223. ISSN: 01663615. DOI: 10.1016/j.compind.2018.04.017. URL: https://doi.org/10.1016/j.compind.2018.04.017.

[25] H Gill. "NSF perspective and status on cyber-physical systems. In National Workshop on Cyber-physical Systems." In: *Austin, TX* (2006).

[26] F. Tao, Q. Qi, L. Wang, and A. Y. Nee. "Digital Twins and Cyber–Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison." In: *Engineering* 5.4 (2019), pp. 653–661. ISSN: 20958099. DOI: 10.1016/j.eng.2019.01.014. URL: https://doi.org/10.1016/j.eng.2019.01.014.

[27] *Industrie 4.0: Jede vierte Maschine ist smart | Bitkom e.V.* URL: https://www.bitkom.org/Presse/Presseinformation/Industrie-40-Jede-vierte-Maschine-ist-smart.html (visited on 01/20/2020).

[28] L. Monostori. "Cyber-physical production systems: Roots, expectations and R&D challenges." In: *Procedia Cirp* 17 (2014), pp. 9–13.

[29] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda. "Cyber-physical systems in manufacturing." In: *Cirp Annals* 65.2 (2016), pp. 621–641.

[30] E. A. Lee. "Cyber-physical systems-are computing foundations adequate." In: *Position paper for NSF workshop on cyber-physical systems: research motivation, techniques and roadmap*. Vol. 2. Citeseer. 2006, pp. 1–9.

[31] J. Lee, B. Bagheri, and H.-A. Kao. "A cyber-physical systems architecture for industry 4.0-based manufacturing systems." In: *Manufacturing letters* 3 (2015), pp. 18–23.

[32] O. Cardin. "Classification of cyber-physical production systems applications: Proposition of an analysis framework." In: *Computers in Industry* 104 (2019), pp. 11–21. ISSN: 01663615. DOI: 10.1016/j.compind.2018.10.002. URL: https://doi.org/10.1016/j.compind.2018.10.002.

[33] M. Broy. "Innovation durch softwareintensive eingebettete Systeme." In: *Cyber-Physical Systems. Berlin Springer* (2010).

[34] N Guo and C Jia. "Interpretation of Cyber-Physical Systems Whitepaper (2017)." In: *Information Technology & Standardization* 4 (2017), pp. 36–47.

[35] Q. Qi, D. Zhao, T. W. Liao, and F. Tao. "Modeling of cyber-physical systems and digital twin based on edge computing, fog computing and cloud computing towards smart manufacturing." In: *ASME 2018 13th International Manufacturing Science and Engineering Conference*. American Society of Mechanical Engineers Digital Collection. 2018.

[36] P. Sobhrajan, S. Y. Nikam, D. Pimpri, and P. D. Pimpri. "Comparative study of abstraction in cyber physical system." In: *Int. J. Comput. Sci. Inf. Technol* 5 (2014), pp. 466–469.

[37] Y. Zacchia Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto. "State of the art of cyber-physical systems security: An automatic control perspective." In: *Journal of Systems and Software* 149 (2019), pp. 174–216. ISSN: 01641212. DOI: 10.1016/j.jss.2018.12.006. URL: https://doi.org/10.1016/j.jss.2018.12.006.

[38] C. H. Liu and Y. Zhang. *Cyber physical systems: architectures, protocols and applications*. Vol. 22. CRC Press, 2015.

[39] S. Thiede, M. Juraschek, and C. Herrmann. "Implementing Cyber-physical Production Systems in Learning Factories." In: *Procedia CIRP* 54 (2016), pp. 7–12. ISSN: 22128271. DOI: 10.1016/j.procir.2016.04.098. URL: http://dx.doi.org/10.1016/j.procir.2016.04.098.

[40] J. Zhou, Y. Zhou, B. Wang, and J. Zang. "Human–Cyber–Physical Systems (HCPSs) in the Context of New-Generation Intelligent Manufacturing." In: *Engineering* 5.4 (2019), pp. 624–636. ISSN: 20958099. DOI: 10.1016/j.eng.2019.07.015.

[41] Y. Lu, C. Liu, K. I. Wang, H. Huang, and X. Xu. "Digital Twin-driven smart manufacturing: Connotation, reference model, applications and research issues." In: *Robotics and Computer-Integrated Manufacturing* 61.April 2019 (2020), pp. 101–837. ISSN: 07365845. DOI: 10.1016/j.rcim.2019.101837. URL: https://doi.org/10.1016/j.rcim.2019.101837.

[42] S. Thiede. "Environmental Sustainability of Cyber Physical Production Systems." In: *Procedia CIRP* 69.May (2018), pp. 644–649. ISSN: 22128271. DOI: 10.1016/j.procir.2017.11.124. URL: http://dx.doi.org/10.1016/j.procir.2017.11.124.

[43] J. Klimeš. "Using formal concept analysis for control in cyber-physical systems." In: *Procedia Engineering* 69 (2014), pp. 1518–1522.

[44] S. J. Oks, M. Jalowski, A. Fritzsche, and K. M. Möslein. "Cyber-physical modeling and simulation: A reference architecture for designing demonstrators for industrial cyber-physical systems." In: *Procedia CIRP* 84 (2019), pp. 257–264. ISSN: 22128271. DOI: 10.1016/j.procir.2019.04.239. URL: https://doi.org/10.1016/j.procir.2019.04.239.

[45] S. Kröning. *Integrierte Produktions-und Instandhaltungsplanung und-steuerung mittels Simulationstechnik*. PZH Verlag, TEWISS-Technik und Wissen GmbH, 2014.

[46] M. Engels-Lindemann. *Optimierung von Programm-und Budgetentscheidungen der betrieblichen Instandhaltung*. Jost-Jetter, 2002.

[47] S Kowalewski and K. Bettenhausen. "Cyber-physical systems: Chancen und nutzen aus sicht der automation." In: *VDI/VDE-Gesellschaft Mess-und Automatisierungstechnik* (2013).

[48] S. Zhao, L. Wang, and Y. Zheng. "Integrating production planning and maintenance: an iterative method." In: *Industrial management & data systems* (2014).

[49] I. Graessler and A. Poehler. "Human-centric design of cyber-physical production systems." In: *Procedia CIRP* 84 (2019), pp. 251–256. ISSN: 22128271. DOI: 10.1016/j.procir.2019.04.199. URL: https://doi.org/10.1016/j.procir.2019.04.199.

[50] M. Tisch, C. Hertle, J. Cachay, E. Abele, J. Metternich, and R. Tenberg. "A systematic approach on developing action-oriented, competency-based Learning Factories." In: *Procedia CIRP* 7.0 (2013), pp. 580–585.

[51] L. Schebek, J Kannengießer, A Campitelli, J Fischer, E Abele, C Bauerdick, R Anderl, S Haag, A Sauer, J Mandel, et al. "Ressourceneffizienz durch Industrie 4.0." In: *Potenziale für KMU des verarbeitenden Gewerbes (VDI Zentrum Ressourceneffizienz GmbH (VDI ZRE), Ed.), Berlin. Accessed* 11 (2017), p. 2017.

[52] T. Lins and R. A. R. Oliveira. "Cyber-physical production systems retrofitting in context of industry 4.0." In: *Computers and Industrial Engineering* 139 (2020), pp. 106–193. ISSN: 03608352. DOI: 10.1016/j.cie.2019.106193. URL: https://doi.org/10.1016/j.cie.2019.106193.

[53] T. Becker and H. Stern. "Future trends in human work area design for cyber-physical production systems." In: *Procedia Cirp* 57 (2016), pp. 404–409.

[54] A. Botthof and E. A. Hartmann. *Zukunft der Arbeit in Industrie 4.0*. Springer Vieweg Berlin, 2015.

[55] S. E. Humphrey, J. D. Nahrgang, and F. P. Morgeson. "Integrating motivational, social, and contextual work design features: a meta-analytic summary and theoretical extension of the work design literature." In: *Journal of applied psychology* 92.5 (2007), p. 1332.

[56] M. Grieves. "Digital twin: manufacturing excellence through virtual factory replication." In: *White paper* 1 (2014), pp. 1–7.

[57] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang. "Modeling, simulation, information technology & processing roadmap." In: *National Aeronautics and Space Administration* (2012).

[58] F Tao, W Liu, M Zhang, T. Hu, Q. Qi, H Zhang, et al. "Five-dimension digital twin model and its ten applications." In: *Comput Integr Manuf Syst* 25.1 (2019), pp. 1–18.

[59] B. Bagheri and J Lee. "Big future for cyber-physical manufacturing systems." In: *Des World, September* (2015).

[60] T. H.-J. Uhlemann, C. Lehmann, and R. Steinhilper. "The digital twin: Realizing the cyber-physical production system for industry 4.0." In: *Procedia Cirp* 61 (2017), pp. 335–340.

[61] C. Cimino, E. Negri, and L. Fumagalli. "Review of digital twin applications in manufacturing." In: *Computers in Industry* 113 (2019), pp. 103–130. ISSN: 01663615. DOI: 10.1016/j.compind.2019.103130. URL: https://doi.org/10.1016/j.compind.2019.103130.

[62] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn. "Digital Twin in manufacturing: A categorical literature review and classification." In: *IFAC-PapersOnLine* 51.11 (2018), pp. 1016–1022. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.08.474. URL: https://doi.org/10.1016/j.ifacol.2018.08.474.

[63] F. Tao, H. Zhang, A. Liu, and A. Y. Nee. "Digital twin in industry: State-of-the-art." In: *IEEE Transactions on Industrial Informatics* 15.4 (2018), pp. 2405–2415.

[64] *ANSYS Twin Builder | Systems Design Simulation*. URL: https://www.ansys.com/products/systems/ansys-twin-builder (visited on 01/29/2020).

[65] *AnyLogic vs Simio - 2020 Feature and Pricing Comparison*. URL: https://www.capterra.com/simulation-software/compare/95940-110473/AnyLogic-vs-Simio (visited on 01/29/2020).

[66] F. Tao, M. Zhang, and A. Y. C. Nee. *Digital twin driven smart manufacturing*. Academic Press, 2019.

[67] Q. Qi, F. Tao, Y. Zuo, and D. Zhao. "Digital twin service towards smart manufacturing." In: *Procedia Cirp* 72 (2018), pp. 237–242.

[68] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S. C.-Y. Lu, and A. Nee. "Digital twin-driven product design framework." In: *International Journal of Production Research* 57.12 (2019), pp. 3935–3953.

[69] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui. "Digital twin-driven product design, manufacturing and service with big data." In: *The International Journal of Advanced Manufacturing Technology* 94.9-12 (2018), pp. 3563–3576.

[70] C. Zhang, W. Xu, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham. "A reconfigurable modeling approach for digital twin-based manufacturing system." In: *Procedia CIRP* 83 (2019), pp. 118–125. ISSN: 22128271. DOI: 10.1016/j.procir.2019.03.141. URL: https://doi.org/10.1016/j.procir.2019.03.141.

[71] K. Samir, M. R. Khabbazi, A. Maffei, and M. A. Onori. "Key Performance Indicators in Cyber-Physical Production Systems." In: *Procedia CIRP* 72 (2018), pp. 498–502. ISSN: 22128271. DOI: 10.1016/j.procir.2018.03.036. URL: https://doi.org/10.1016/j.procir.2018.03.036.

[72] F. Biesinger, D. Meike, B. Kraß, and M. Weyrich. "A digital twin for production planning based on cyber-physical systems: A Case Study for a Cyber-Physical System-Based Creation of a Digital Twin." In: *Procedia CIRP* 79 (2019), pp. 355–360. ISSN: 22128271. DOI: 10.1016/j.procir.2019.02.087. URL: https://doi.org/10.1016/j.procir.2019.02.087.

[73] V. Zaccaria, M. Stenfelt, I. Aslanidou, and K. G. Kyprianidis. "Fleet monitoring and diagnostics framework based on digital twin of aero-engines." In: *ASME Turbo Expo 2018: Turbomachinery Technical Conference and Exposition*. American Society of Mechanical Engineers Digital Collection. 2018.

[74] Y. Cai, B. Starly, P. Cohen, and Y.-S. Lee. "Sensor data and information fusion to construct digital-twins virtual machine tools for cyber-physical manufacturing." In: *Procedia Manufacturing* 10 (2017), pp. 1031–1042.

[75] Q. Qiao, J. Wang, L. Ye, and R. X. Gao. "Digital twin for machining tool condition prediction." In: *Procedia CIRP* 81 (2019), pp. 1388–1393. ISSN: 22128271. DOI: 10.1016/j.procir.2019.04.049. URL: https://doi.org/10.1016/j.procir.2019.04.049.

[76] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen. "About the importance of autonomy and digital twins for the future of manufacturing." In: *IFAC-PapersOnLine* 48.3 (2015), pp. 567–572.

[77] Y. Lu, H. Wang, and X. Xu. "ManuService ontology: A product data model for service-oriented business interactions in a cloud manufacturing environment." In: *Journal of Intelligent Manufacturing* 30.1 (2019), pp. 317–334.

[78] S. Haag and R. Anderl. "Automated Generation of as-manufactured geometric Representations for Digital Twins using STEP." In: *Procedia CIRP* 84 (2019), pp. 1082–1087. ISSN: 22128271. DOI: 10.1016/j.procir.2019.04.305. URL: https://doi.org/10.1016/j.procir.2019.04.305.

[79] A. Ait-Alla, M. Kreutz, D. Rippel, M. Lütjen, and M. Freitag. "Simulation-based Analysis of the Interaction of a Physical and a Digital Twin in a Cyber-Physical Production System." In: *IFAC MIM 2019 Proceedings* 13.2008 (2019), pp. 1331–1336. DOI: 10.1016/j.ifacol.2019.11.383.

[80] A. D. Rocha, J. Tripa, D. Alemao, R. S. Peres, and J. Barata. "Agent-based plug and produce cyber-physical production system - Test case." In: *IEEE International Conference on Industrial Informatics (INDIN)* 2019-July (2019), pp. 1545–1551. ISSN: 19354576. DOI: 10.1109/INDIN41052.2019.8972169.