POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

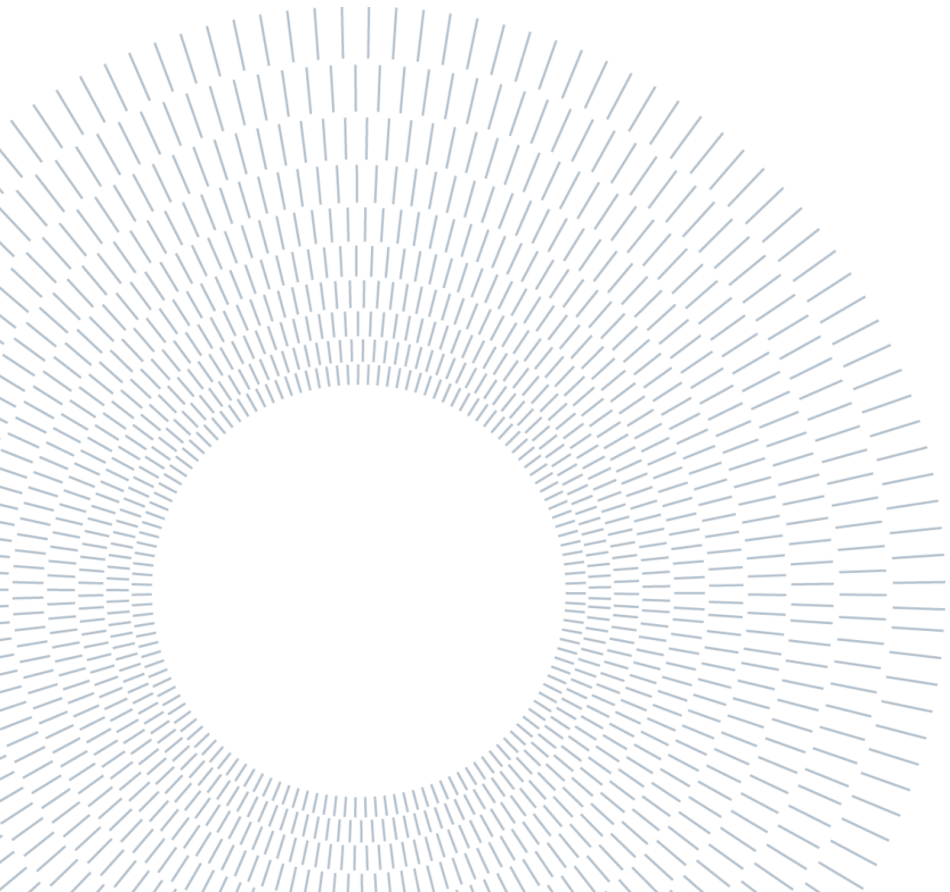
Online Synchronisation of Digital Twins: a Control-based Methodology for Manufacturing Systems Applications

MASTER OF SCIENCE THESIS IN
MECHANICAL ENGINEERING - INGEGNERIA MECCANICA

Authors: Edoardo Passarin 953371
Francesco Verucchi 926797

Advisor: Prof. Andrea Matta
Co-advisor: Ph.D. Giovanni Lugaresi

Academic Year: 2020-2021



Ringraziamenti

Un particolare ringraziamento al professor Andrea Matta per averci dato l'opportunità di sviluppare il nostro interesse verso lo studio di sistemi produttivi, e per la fiducia accordataci durante questo lavoro di Tesi. Un caloroso grazie anche al correlatore Giovanni Lugaresi, per il suo continuo supporto e infinita disponibilità.

Ringraziamo inoltre Sofia e Giulia, per la collaborazione e la reciproca motivazione durante lo sviluppo del progetto.

Infine vorremmo augurare un grande bocca al lupo per il futuro percorso dei ragazzi del "Digital Twin Lab", e ringraziare in particolare Stefano per l'aiuto datoci.

Abstract

Under the fourth industrial revolution, many technologies have emerged and contributed to a substantial advancement of the automation and data exchange in the manufacturing industry. Parallelly, the competition and uncertainty of the global markets have forced a continuous evolution in production systems. The digitalization of processes and the demanding industrial context has sparked an interest towards the research of Digital Twin (DT). It is defined as a virtual representation of a production system, providing the capabilities to improve productivity and support decision-making, using Discrete Event Simulations (DES). One of the challenges of this technology is to make the DT able to represent in real-time the complete behaviour of the physical system and adapt to possible transformations. This work aims at determining the most critical factors causing divergency between the two objects, and developing a methodology based the concept of "alignment", introduced as the capability of the DT to completely reflect the physical system's conditions. An indicator-based control manages efficiently the procedures of automatic DES model update, synchronisation, and stochastic input model update, integrated into a digital architecture applied to a lab-scale model. Experiments are performed to evaluate the limits and capabilities of the services provided by the DT in real-time, consisting of the monitoring of current performances and prediction of future ones. In conclusion a case study is developed to apply the techniques in the context of a proof-of-concept Digital Twin. The results of this work demonstrate that the methodology is effective in keeping the alignment between a digital model and the physical system.

Key-words: Digital Twin; Production System; Discrete Event Simulation; Synchronization; Automatic Model Generation

Abstract in lingua italiana

Durante la quarta rivoluzione industriale molte tecnologie sono emerse e hanno contribuito a un sostanziale progresso dell'automazione e dello scambio di dati nel settore manifatturiero. In parallelo, la concorrenza e l'incertezza dei mercati globali hanno richiesto una continua evoluzione dei sistemi di produzione. La digitalizzazione dei processi e l'esigente contesto industriale ha suscitato un interesse verso la ricerca riguardo il Digital Twin (DT). Questo è definito come una rappresentazione virtuale di un sistema produttivo, che fornisce le funzionalità per migliorare la produttività e supportare il processo decisionale, utilizzando simulazioni ad eventi discreti (DES). Una delle sfide di questa tecnologia è quella di rendere il DT in grado di rappresentare completamente in tempo reale il funzionamento del sistema fisico e adattarsi alle possibili trasformazioni. Questo lavoro mira a determinare i fattori più critici che causano la divergenza tra le due entità e a sviluppare una metodologia basata sul concetto di "allineamento", introdotto come la capacità della DT di riflettere completamente le condizioni del sistema fisico. Un controllo basato su indicatori gestisce in modo efficiente le procedure di generazione automatica del modello DES, sincronizzazione, e aggiornamento dei parametri stocastici, integrate in un'architettura digitale applicate ad un modello di produzione in scala. Vengono eseguiti esperimenti per valutare i limiti e le capacità dei servizi forniti dal DT in tempo reale, consistenti nel monitoraggio delle prestazioni attuali e nella previsione di quelle future. In conclusione, un caso di studio è stato ideato per applicare le tecniche sviluppate nel contesto di un proof-of-concept Digital Twin. I risultati di questo lavoro dimostrano che la metodologia è efficace nel mantenere l'allineamento tra un modello digitale e il sistema fisico.

Parole chiave: Digital Twin; Sistema di Produzione; Simulazione ad Eventi Discreti; Sincronizzazione; Generazione Automatica di Modello di Simulazione

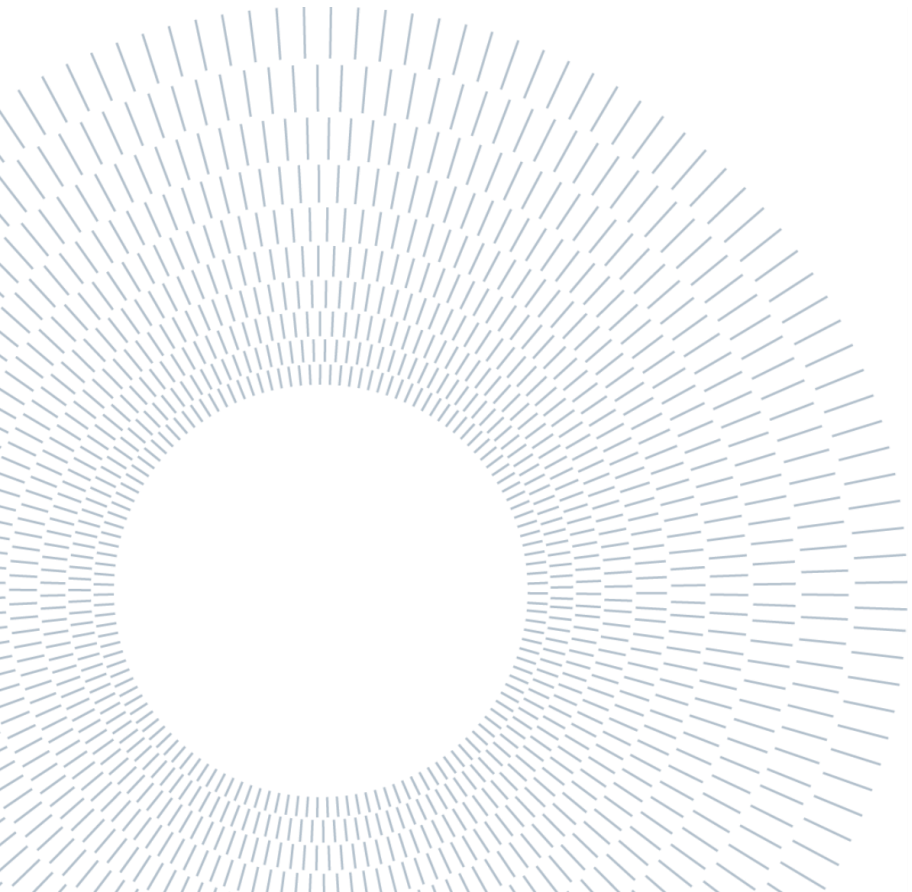


Table of contents

List of figures

List of tables

List of Abbreviations

ANOVA: Analysis of Variance

BAS: Blocking After Service

CI: Confidence Interval

CPS: Cyber Physical System

DES: Discrete Event Simulation

DoE: Design of Experiments

DM: Digital Model

DS: Digital Shadow

DT: Digital Twin

ERP: Enterprise Resource Planning

ID: Identifier

IoT: Internet of Things

JSON: JavaScript Object Notation

KPI: Key Performance Indicator

MHS: Message Handling System

MQTT: Message Queue Telemetry Transport

MLE: Maximum likelihood Estimation

PLM: Product Lifecycle Management

RTS: Real-Time Simulation

SSH: Secure Shell Host

UML: Unified Modeling Language

VBA: Virtual Basic for Applications

XML: eXtensible Markup Language

1. Introduction

Under the fourth industrial revolution, many technologies have emerged and contributed to a substantial advancement of the automation and data exchange in the manufacturing industry. Parallelly, the competition and uncertainty of the global markets have forced a continuous evolution in production systems.
The digitalization of processes and the demanding industrial context has sparked an interest towards the research of Digital Twin (DT). The concept extends from the integration of technologies like Cyber Physical Systems (CPS) and Internet of Things (IoT), providing the capabilities to improve productivity and support decision-making. This thesis investigates the DT framework and in particular the aspects involved In real-time synchronisation of Discrete Event Simulation (DES) models.

1.1. Digital Twin

The concept of Digital Twin, once known as conceptual ideal for Product Lifecycle Management (PLM), is first presented to the industry in 2002 by Dr. Grieves at the University of Michigan. Figure 1.1 shows all the fundamental structures of a modern DT: a real space linked to a virtual space, a bi-directional flow of information, and also a connection from virtual to following sub-spaces.

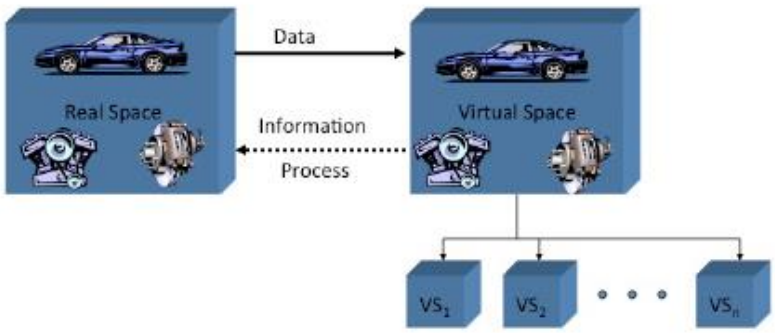


Figure 1.1: Conceptual ideal for a PLM. [Dr. Michael Grieves, University of Michigan, Luries Engineering Center, Dec 3, 2001]

The physical system is defined as an always present entity while the virtual side as constantly fed with all the information of the other. The terminology once used for

PLM wanted to underline the distance from a static representation, but instead, a linkage throughout the whole lifecycle of the system. A complete definition of Digital Twin is proposed by Dr. Grieves and Vickers [3]: *"The Digital Twin is a set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometric level"*. Ideally, any information obtainable from physical system's inspections could be equally taken from its Digital Twin. Also the NASA has been fundamental in paving the way of the Digital Twin evolution. As highlighted by Negri et al. 2017 [1], its contribution has introduced capabilities such as: the use of integrated sub-models to mirror the life of air vehicles, considering their interactions with the real world; also, a life-cycle view and the ability to conduct prognostic and diagnostic activities. While initially the DT is mainly applied at product level, the evolution of this technology has allowed the introduction on entire systems. As a matter of fact, this will be the context of application of this study.

Due to multiple existing solutions and applications causing misleading interpretations, Kritzinger et al. in [4] proposed a classification focused on the level of data interconnections. Figure 1.2 shows the three categories of digital architectures.

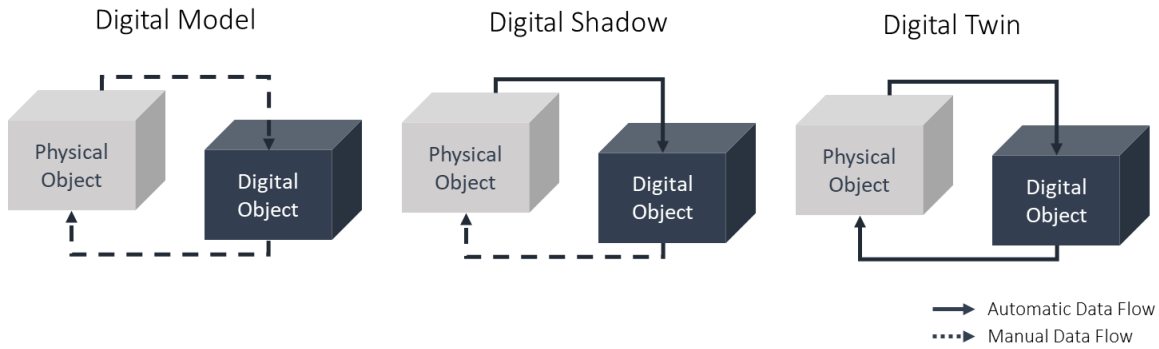


Figure 1.2: Classification based on data interconnections adapted from ref. [Kritzinger et al.2018]

- A Digital Model virtually represents an existing or planned physical object. Simulation models of any other type can be used for this scope, however data are manually exchanged. Furthermore, a possible variation in the physical system is not directly reflected in the digital one and vice versa.
- If an automated one-way exchange of data between the state of the physical object and the digital one exists, this is defined as Digital Shadow. Thus, a change in state in the physical only affect the digital counterpart.
- Once data flows are bi-directional between the two entities, the Digital Twin is achieved. Adding to the previous explained potentialities, now the digital object is enhanced by the capability of acting as controller over the physical system.

As shown in Figure 1.1, the dynamic model is intended to be able to follow the changes over the lifecycle of the system. During the design phase, the DT supports developers to test and understand how the real model will behave. In the development phase specific and potentially unique configurations are built on the physical system. These are reflected as instances in the virtual space, providing the means to perform analysis among the real objects while still in development. Next is the operational phase, during which the data interconnections are crucial to maintain the network between the two entities. This capability allows the continuous mirroring of the physical system conditions, also the prediction of performances and the reaction to solve failures. The application of the Digital Twin framework may be even more exploited by the creation of multiple virtual spaces (see Figure 1.1 $VS_1, VS_2 \dots VS_n$) allowing to put the system through destructive test inexpensively.

1.1.1. Digital Twin applications

The DT provides the opportunities to optimize activities and support decision-making. This aspect makes it largely applicable in many contexts of application. For instance, in the aerospace field, to investigate, through conspicuous time history of flights, future maintenance's needs and interventions; also, in robotic industries and especially for virtual commissioning, to optimize the control algorithms during the development phase of robots. In particular, this work focuses on the applications within production and manufacturing systems where the DTs are employed mostly to detect and tackle disruptive events, as well as providing supplementary performance measurements. In addition, they can be potentially deployed at various levels, from single components to the entire system, and in many disciplines such as: production planning and control, maintenance, and layout designing. Ultimately the provided means contribute to elevate competitiveness, productivity and efficiency [4].

1.1.2. DT and online DES model

Practically, the DT is consistent with a CPS type architecture, which involves a series of virtual components interconnected and communicating with a physical system's infostructure. The digital tools deployed are highly dependent on the results and type of physical system that are intended to be handled. Despite the several differences between implementations, some challenges are common for the virtual counterpart. Firstly, it must be able to perform rapidly in conditions of high complexity and randomness. Furthermore, it is required to handle the processing of vast amount of data given by the information source provided. Although adopting closed-form analytical algorithms would allow to achieve accurate and fast computing solutions, they are also limited in their deployment. Simulation is on the other hand a better candidate for flexibility and less restrained assumptions. However, considering the computational effort requested, it might show some

criticalities given the high solving speed required.
Differently from a classic approach, where often simulation is performed in the early stage of designing the physical object and then disposed. In a DT the application is carried throughout the operational phase, in the so called "Real-Time"/"Online" conditions.
Online simulation is a concept developed earlier than the DT, but for several characteristics it covers an essential role in the framework. Davis et al. [5] underlines the conditions which distinguish an "online simulation" from the others. Firstly, the simulation horizon is shorter relatively to the dynamic of the system. Consequently, the initial state of the system in the simulation represents a very influential point on the results and on the decision-making process. In particular, distancing from traditional "non terminating" simulation, the online one cannot have an initial transient phase for tuning purposes but must deliver results right from the start [6]. Thus, the simulation model has to be aligned in real-time with the physical system and every feature must be reflected digitally. Secondly, results are required to be delivered within a reasonable timeframe; therefore, the simulation engine has to be fast. Summarising, while this particular type of simulation becomes a fundamental piece of the technological puzzle it also adds further requirements and challenges that must be taken into account.

1.2. Problem introduction

Modern production plants are affected by a rapid evolution, the consequent transformations introduce several expected and unexpected changes between the Digital Twin and the physical system. In particular, while the introduction of new resources and the modification of the facility's layout are known and planned events, factors such as components aging, and degradation may not be predicted. The DT is required to consistently mirror the dynamic system for it to be a support tool during the operational phase. Specifically, the process of reflecting real-time conditions necessitates the DT to adapt to disruptions and evolutions affecting the physical plant. This is a challenging task and requires the study of numerous leading factors that might influence the Digital Twin's ability to correctly mirror the physical system. The research about this precise topic is scarce and a structured discussion addressing it is not present. Thus, the authors have originally defined the term "alignment", identifying the status of the DT as completely mirroring the real-time conditions of the physical system. Three distinct challenges are identified, each one of them producing a possible cause of misalignment:

- **Challenge 1 – Topological Misalignment.**
  The topology and layout of a manufacturing plant might vary, together with the functioning logic of the production processes. Classically these aspects are addressed during the design phase of the digital model, and assuming a non-changing system this would be enough. However, since disruptive

disturbances are part of modern real systems' life cycles, investigating methodologies applicable during the operational phase becomes of high interest.

- **Challenge 2 – Instant conditions**
  Instant conditions are defined by the real-time status of the elements and resources that compose a manufacturing system. These are affected by a various range of influencing factors; therefore, information related to their variations must be transmitted to the digital model satisfying all the constraints and requests needed for the conduction of an online simulation.
- **Challenge 3 – Stochastic behaviours**
  The third and conclusive element is determined by the stochastic behaviours of the processes that characterize the real system. These are implemented in the digital model as distributions and are required to be correct when performing predictive simulations.

These elements need to be addressed separately for the alignment action to be effective.

Ultimately, Kritzinger et al. [9] asses that the development of real case-studies with regards to Digital twin with bi-directional data transfer is still at its infancy. A majority of the papers reviewed show that the level of integration achieved is quite often not as in a Digital Twin, but more Digital shadow and Digital model highlighting the lack of maturity when it comes to defining the Digital twin term. In addition, the almost absence of concrete case studies shows the complexity of creating the data connections and platforms to perform the technology within real conditions.

1.3. Aim of the work

The aim of this work is to analyse the alignment between a Digital Twin and the physical system. The concept develops on multiple levels and contexts of application, therefore, considering the previously introduced potential causes of divergency, three distinct solving actions have been developed:

- Automatic model update of DES models, when they no longer reflect the physical system's logic.
- Synchronisation of the instant conditions, aligning the real-time status of the elements on the production plant with the corresponding objects in the DT.
- Input model update, involving a distribution fitting of the information provided within the historical data.

The actions are managed by means of an integrated control based on indicators, which are the outcomes of comparison procedures between the real and digital

objects and identify the alignment status. The result is an efficient solution, with alignment procedures deployed only when necessary, limiting the computational effort. Also, it is integrated in a digital architecture which provides the DT services of monitoring and forecasting. An aligned model assures the correctness of the provided monitoring service, consisting of using the simulating capabilities of the Digital Twin to obtain the real-time performances of the physical system. Furthermore, the predictions obtained by the forecast analyses are precise and updated to the changes affecting the production plant.
Finally a proof-of-concept Digital Twin is demonstrated with a case study, in which bi-directional data exchange features allow to perform monitoring, forecasting and what-If analyses on a lab-scale manufacturing system.

1.4. Thesis outline

Chapter 2 introduces a literature overview on the subjects of model generation and conversion, synchronisation, and initialisation. Chapters 3 and 4 outline respectively the proposed methodology aimed at addressing the alignment issue, and the physical system used to test it, together with the developed software architecture. In chapter 5 the single components are validated and tested, with the aim of understanding their capabilities and limitations. Furthermore, three experimental campaigns are introduced, analysing the behaviour of the synchronisation and forecast functions. Chapter 6 presents a case study, composed of a demonstration aimed at testing the developed methodology in its entirety, together with additional components that make up a proof-of-concept Digital Twin. Finally, the conclusions of this work, including the limitations and potential future developments, are outlined in chapter 7.

2. State of the Art

This chapter presents the state of the research regarding the alignment of a physical system and its digital counterpart. This originally introduced concept allows to consider the multiple factors causing lack of real conditions reflection in the DT. Contributions found in literature expands on one front, which consider automatic generation of DES models to align the logic and layout of the physical system. Alternatively, a significant work is done to tackle the evolution of instant conditions and the variation of stochastic behaviors by synchronising the appropriate parameters.
Section 2.1 introduces to automatic model generation and illustrates the methods of conversion. Consequently, section 2.2 presents the synchronisation techniques, anticipating the concept of initialisation.

2.1. Generation

In a simulation project, the procedure is to firstly generate a model representing the physical system's behavior, and then use it to perform analyses. The standard approach for the development of a simulation model is creating it manually using commercial software programs.

Chance, Robinson, and Fowler [11] as well as Yucesan and Fowler [12] define a process for the simulation of manufacturing system, composed of three major steps:

1. Model Design
2. Model Development
3. Model Deployment

In the design phase the scope of the project is identified, and a conceptual model is developed. An important aspect is the complexity of the behavior of the physical system and how that is modelled. A too detailed model may increase its accuracy, but it will render it too complex and hard to develop and maintain. Therefore understating the required detail of the simulation model is a crucial step in the design phase.
The model development phase consists of converting the conceptual model developed previously into a digital model, which is required to be tested and validated.

The last phase, defined as model deployment, is when the digital model created is used to perform experiments and subsequent analyses on the result obtained through discrete event simulations.
The first two phases of the simulation project are usually performed manually, which is a task that is very time and labor intensive. In fact, these operations require specialized personnel, in the form of simulation experts. A solution to this issue would be to implement an automatic discrete event model generation method, reducing the time and effort required for the design and development phase of a simulation project. Automatic Model Generation would also simplify the modelling of increasingly complex and larger systems. Vieira et al. [13] defines the automatic model generation as one of the major features of discrete event simulation that should contribute to the industry 4.0 revolution.
The automatic generation of a discrete event simulation model using data collected from the production floor has been partly solved by the work of Lugaresi et al. [14].

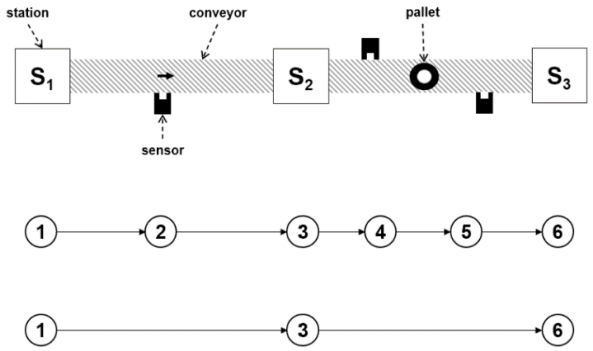


Figure 2.1: Schema of the generation and tuning method ref. [Lugaresi et al. 2020]

The authors develop a method for the generation and tuning of a graph model representing a production system behavior logic from information gathered from an events log. In Figure 2.1 is depicted a schema of the generation method at different levels.
The obtained model can be converted to other modelling formalization, i.e., petri nets, but unfortunately there still isn't a method to automatically translate it into a language able to perform discrete event simulations. This type of conversion would complete the procedure that generates an executable simulation model.

In their research they define the automatic model generation procedure as a sequence of six steps:

1. Data collection
2. Process topological discovery
3. Statistical analysis
4. Control policies identification
5. Model conversion

6. Model validation

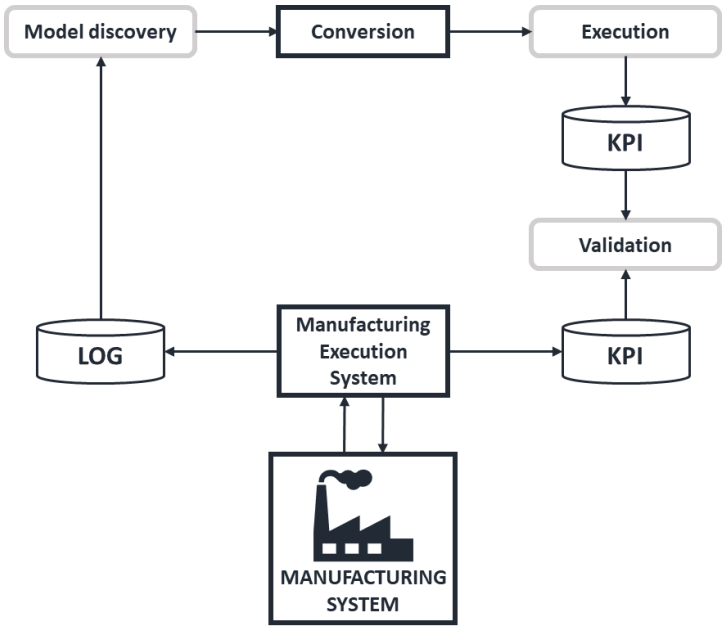


Figure 2.2: Typical procedure for automatic model generation, adapted from ref. [Lugaresi et al. 2020]

This thesis work focuses on the model conversion phase, which is composed of all the necessary steps in order to obtain an executable simulation code of the designed system.

Research is conducted on the different proposed methodologies for the conversion of input data into an executable simulation model, either using commercial off-the-shelf programs or open source methods based on programming languages like Python or Java.

2.1.1. Input source

To generate an executable simulation model, the necessary information regarding the system's topology and logic must be given to the conversion method. This includes data about all the resources, entities, policies, and their parameters. Since the necessary information is collected from different sources within the company, a file standard that is compatible with different applications surely helps the information exchange.

Literature presents different standards for the information exchange between manufacturing applications. One of the most used formats is XML (eXtensible Markup Language). Many standards use this format as modelling language. Lee et al. [15] analyze and compared three data exchange interface standards, precisely CMSD (Core Manufacturing Simulation Data), OAGIS (Open Application Group Integration Specification) and ISA-95 (Instrumentation, Systems and Automation

society). All these standards use as file exchange format XML. The authors' goals were to evaluate the CMSD standard in a case study regarding the operations of a car manufacturing plant.

2.1.2. Methods for model conversion

Mathewson in [16] elaborated the concept of translating a logic of a model from general symbolism into code that a computer can simulate. The author referred to this type of software as a "program generator". In the research an entity cycle diagram, shown in Figure 2.3, is transformed through the developed program DRAFT (based on FORTRAN) into an error-free code ready for a simulation.

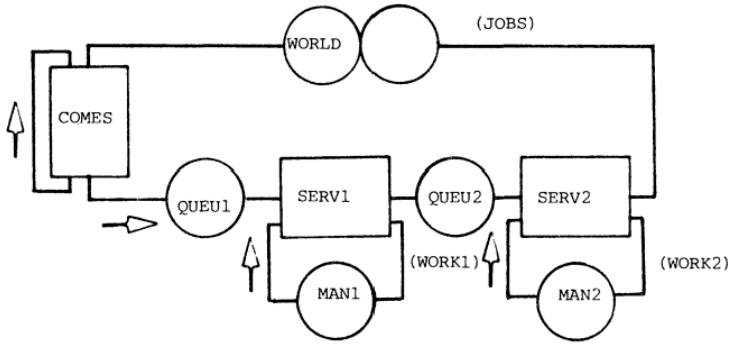


Figure 2.3: Entity cycle diagram ref. [Mathewson 1985]

Krenczyk et al. in [17] propose a solution that allows the semi-automatic conversion of external data into a discrete event simulation model. The focus of the paper is to implement an adaptable framework for the integration of ERP systems with simulation systems. The method is able to convert data about the production system, processes and plan into an executable model built using the FlexSim simulation software. The information exchange is done using files in XML format, which is then loaded into a custom developed software called RapidSim, which generates an input file for the FlexSim simulation system, containing all the objects which compose the model, data on the production system and the flow of production orders.

Haraszkó et al [18] develop a method for the automatic conversion of custom layouts of manufacturing systems into the simulation software Plant Simulation. The information regarding the type and order of operations, the type of manufacturing equipment, the type of material flow solutions and the material handling systems, the layout type (species and sub-species or sub-types), and the product mix. The data is manually introduced into the translator by using a custom GUI, where the user specifies the necessary information. The developed tool aids the design phase of a manufacturing system since it allows for easy and fast generation of simulation models with different layouts.

Bloomfield et al. in [19] develop a software called UA Translator, which is able to convert the information about an assembly process into a simulation model in ProModel and Arena. All the data exchange is done using XML files using the CMSD standard. The UA Translator is a VBA application, which through its processing logic is able to convert the assembly files (from Design Profit software) into a DES model, by populating it into the simulation software. The author presents two case studies, testing the translator's ability to convert correctly the data into a simulation model in the first one, and the interoperability with other applications, specifically a job shop scheduler and an order and inventory system. The benefit of the proposed solution is a saving in time and cost regarding the DES development.

Popovics et al. in [20] propose the framework EasySim for the automatic generation and conversion of a DES model. The information regarding elements composing a job shop manufacturing systems are stored in files (either excel sheets or XML files) which are then imported into the EasySim program, where model building algorithms generate the executable model. The framework, schematized in Figure 2.4,uses ISA-95 standard for data input and output between the simulation model and data sources. There is the added capability of validation by comparing the simulation results with the input data. The framework is tested with experiments in both EasySim and Plant Simulation.

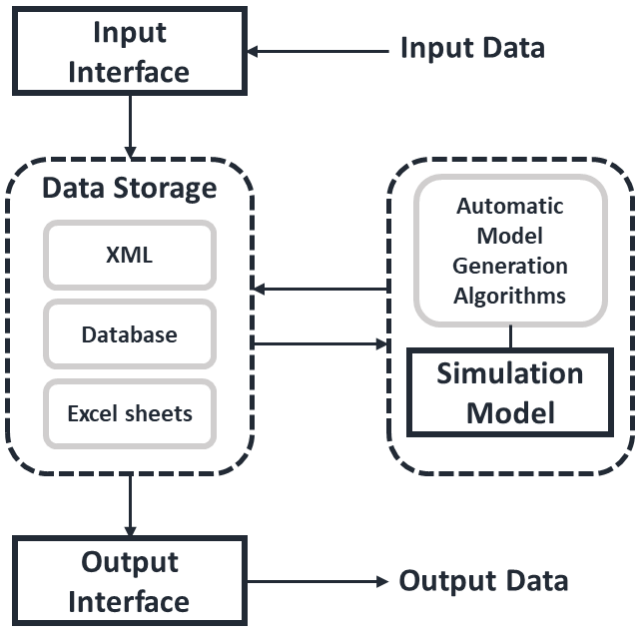


Figure 2.4: The architecture of the EasySim framework, adapted from ref. [Popovics et al. 2016]

Krenczyk et al. in [21] developed a method for the conversion of the information regarding a manufacturing system into a DES model. The information regarding the resources and production processes are stored in an XML file which uses a custom developed structure. Using an XSLT (Extensible Stylesheet Language Transformations) processor, the data is automatically transformed into a format

readable by simulation systems. The algorithm which generates the simulation code is a sequential process, generating different elements at each stage. Figure 2.5 describes the proposed framework was tested using the software Enterprise Dynamics, but the author suggests that other commercial programs could be used.

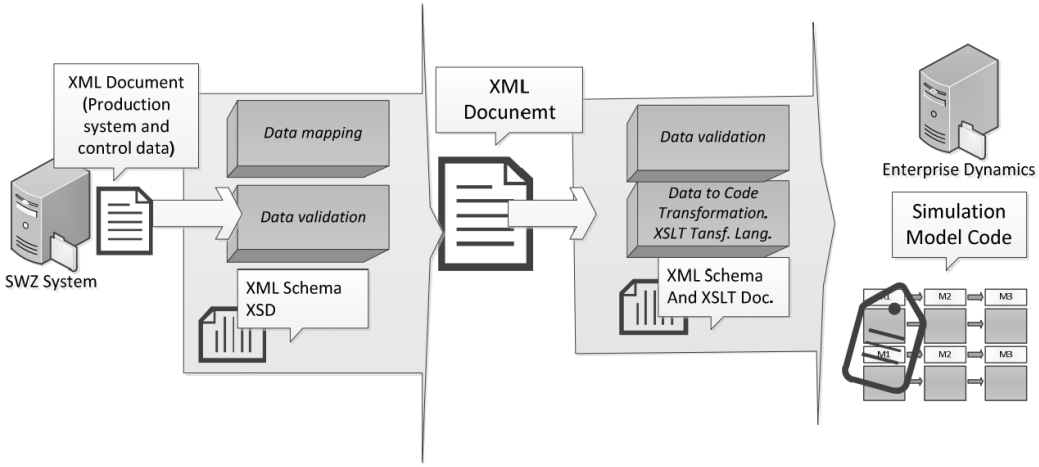


Figure 2.5: XML document transformation ref. [Krenczyk et al. 2014]

Meng et al in [22] propose a method for the automatic generation of material handling systems in coal mines. The author models the behavior of different equipment and resources as custom-built Petri nets. The information necessary to model the MHS is gathered from different production servers and stored into an excel file, which is then loaded into Arena using VBA. The simulation model is then populated with the necessary objects, verified, validated, and finally used to perform what ifs analyses. The developed framework was then implemented using real production data coming from a real open pit mine in the USA.

To summarise, eventhough some methodologies do exist they are either very specific to a use-case or only conceptualisations. In particular, they are lacking of capabilites allowing them to be used in a DT context.

2.1.3. Conversion requirements and software selection

Research is conducted on the main commercial simulation software programs as well as open source methods for the conversion of the graph model into an executable program able to perform discrete event simulations.
The requirements for the conversion method have been established, and are:

- Modelling capability: many production processes are really complex; therefore the simulator needs to be capable of replicating it.
- Operability: the software needs to be easy to use. Commercial software provides a pick and place interface that allows easy modelling, while most open-source solutions require an understanding of the code.

- Availability: commercial software can be expensive, an open-source solution may provide the necessary utility free of charge.
- Interoperability: since the digital model needs to be able to communicate with external functions, its ability to do that is required.
- Level of automation: the conversion and simulation processes need to be completely automatic, there should be no manual intervention required.

Based on these requirements a list of possible solutions has been created, presented in Table 2.1, and ranked with a qualitative score of low, medium, or high. A first distinction is between commercial software and open-source methods (based on Python and Java).
The research is constrained to only open-source software or available for academic use.

2.1.4. Commercial software

Arena: is one of the most used discrete event simulation programs. The software has very extensive modelling capabilities, being able to replicate very complex behaviors. The main shortcoming is the fact that Arena has limited automation capabilities. VBA can be used to perform simple tasks, like gathering data from excel sheets or text files or opening and closing a selected model. The ability to place block objects and connecting them not very accessible thus making impossible a complete automation of the model generation. The software also has limited interoperability with other applications.

Siemens Tecnomatix Plant Simulation: the other commercial software tested is Plant Simulation, offered by Siemens. Similarly to Arena, the software is able to replicate the behaviors of very complex systems. Plant Simulation offers interoperability with other application, for example Oracle databases or PLCs, through its interface package. Furthermore, the internal programming language of the software, SimTalk, allows for the creation of the simulation objects and connections. The biggest disadvantage of this solution is the fact that in order to run the model it is required to access Plant Simulation, rendering the automatic conversion and simulation impossible.

2.1.5. Open-source methods

JSimIO: It is a Python library able to generate JSIM simulation models and perform simulation with the JMT engine [23]. This framework can model different manufacturing system layouts and can output analytical results in the form of throughput at system and node level. The main limitations of JSimIO are the necessity of using the JMT software outside of Python to see the results of the simulations and the lack of customizability of the objects functioning. Since it is

based on Python, this library has some level of interoperability with different applications and programs.

SimPy: it is a framework based on Python which creates an environment for performing discrete event simulations. The main objects are the environment, resources. The main advantage of this framework is the incredible customizability it offers. Custom classes and resources can be created allowing the ability to model complex behaviors. It is main advantage is also the biggest limit, since every single process must be developed from scratch. This means manually coding every single class composing the model.

ManPy: it is a Python library built on SimPy which includes custom classes specifically built for simulating manufacturing processes. The scope of this library is to replicate the object of commercial software, thus easing the model development phase. The advantage of using this framework it is the fact that is completely open source, allowing the creation of new classes and the modifications of the existing ones. The fact of it being built on Python creates the possibility of connecting with different applications.

	Commercial off-the-shelf		**Open-source**		
	Arena	**Plant Simulation**	**JSimIO**	**SimPy**	**ManPy**
Modelling Capabilities	High	High	Low	High	Medium
Operability	Easy	Easy	Medium	Difficult	Medium
Availability	Paid	Paid	Free	Free	Free
Interoperability	Low	Medium	Medium	High	High
Automation Level	Low	Medium	Medium	High	High

Table 2.1: Software classification for conversion

Considering the proposed requirements of the conversion method and the application in which it will be implemented, an open source solution based on Python is identified as the better choice. ManPy, with its custom objects and functions, is selected as the optimal framework for the required application.

2.2. Synchronisation

The concept of synchronisation is dealt in literature, also various methodologies have been developed considering the type of application and the factors which influence

it. In addition, there is not still a clear definition about this procedure, but most of the authors agree that the common purpose is to reflect the real system conditions into the digital one.

This work focuses mainly on applications at a system level, which enclose those concentrating on the single or multiple products. While the latter has been discretely studied through scientific papers and conferences, the first is still fairly undeveloped.

2.2.1. Synchronisation methodologies

Yang et al [24] study a standard application of synchronisation, namely the implementation of a trace driven simulation, as soon as the data from the real system is available, then the digital model can be realigned and reflect the real conditions in the shortest time possible.
In the paper a very simple real system is used: a mini vehicle driven by dry battery and controlled by an on-off switch. It is implemented as a representation of a flow production line with a defined point assumed to represent a processing point.

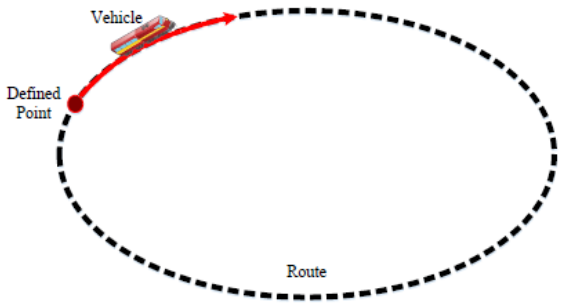


Figure 2.6: Image of the distributed model ref. [W. Yang et al. 2017]

The defined point is characterised by a light sensor, which record a passage when its intensity of light is affected by the passage of the vehicle underneath; This procedure is implemented in Excel VBA. Thus, the time difference between two succeeding signals will define the system cycle time, composing the real-time data then communicated to the digital counterpart.
The results of the paper shows the composition of a dashboard showing the effective shadowing in real-time of the digital model with respect to the real system. Also, a list of the cycle time from the real system and digital is given, showing the correctness of the results.
In Table 2.2 column A shows the real cycle time acquired by the real system in real time, column B the input data fed to the digital system and C the acquired cycle time from the digital model. Trivially column B and C will be equal, but this validates the actual functioning of the digital model.

A	B	C
6.13	6.13	6.13
6.09	6.09	6.09
6.02	6.02	6.02
6.05	6.05	6.05
6.04	6.04	6.04
6.16	6.16	6.16
6.10	6.10	6.10
6.08	6.08	6.08
6.16	6.16	6.16
6.18	6.18	6.18

Table 2.2: Experimental results ref. [W. Yang et al. 2017]

The type of methodology is therefore illustrated in Figure 2.7 and Figure 2.8. While the real system, identified in the mini vehicle running through the circuit complete a cycle, the digital model waits to receive the information.

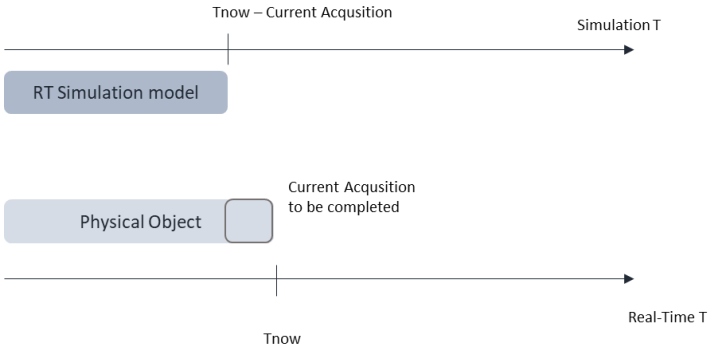


Figure 2.7: (a) Methodology developed adapted from ref. [W. Yang et al. 2017]

When the cycle is complete the information is computed using the difference between the two succeeding recordings then sent to the digital model. Simulation can be then run and aligned.

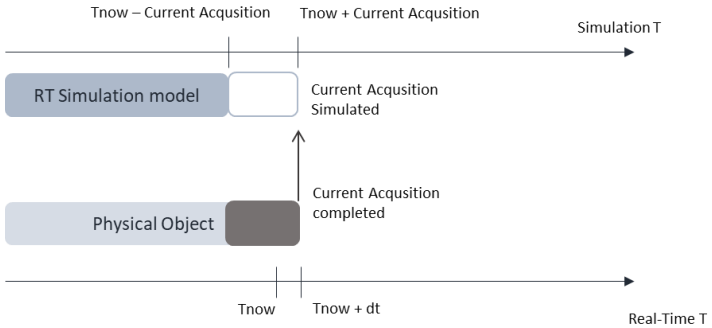


Figure 2.8: (b) Methodology developed, adapted from ref. [W. Yang et al. 2017]

Ideally the transmission of information should be instantaneous and therefore the synchronisation is achieved just for an instant of time, as the real system continues its cycle. It is possible to understand that with this methodology the digital model is only able to represent the past and present conditions.

Hanisch et al. [7] conceptualise this type of methodology as a permanent synchronisation. As it can be understood by the aforementioned explanation the time advance mechanism of the simulation has to be the same as for the real system.
In this paper the authors introduce also a methodology named Requested synchronisation.
The simulation works in sleep mode by being in stand-by between updating modes, during which a realignment of the conditions is obtained.

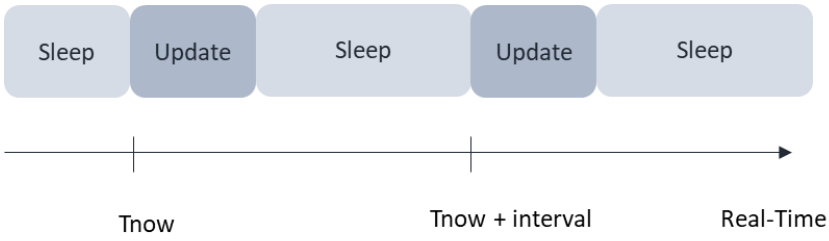


Figure 2.9: Requested synchronisation, adapted from ref. [W. Yang et al. 2017]

This type of synchronisation becomes more important in using the digital counterpart to conduct forecast and interrogative actions.

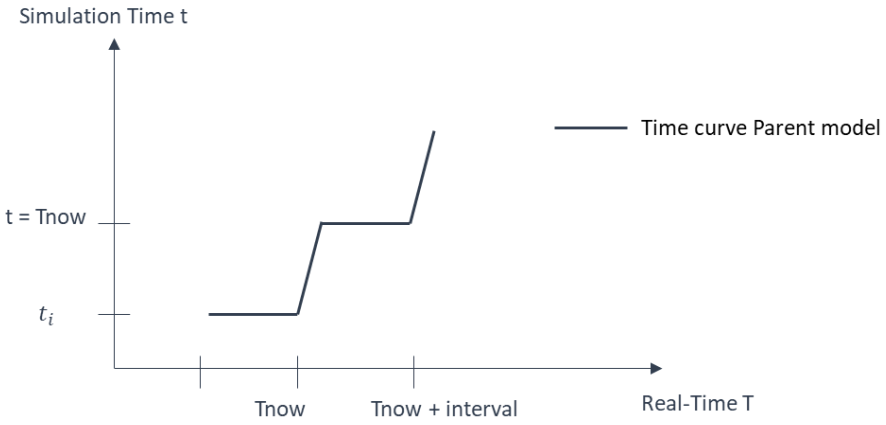


Figure 2.10: Correlation real and digital system, adapted from ref. [W. Yang et al. 2017]

Figure 2.10 shows the correlation between the digital simulation time and the real time. As it is possible to see when the synchronisation is requested, the simulation is run beginning from the initial condition of t_i to reach the conditions of the real system at Tnow.

Cardin et al. [25] implement the Digital Twin for the case study of decision process upon allocation of transporters to fabricate an order.

The first case sees the observer being ahead of time of the physical system. The progress of those parts, which information from the physical system is available, keeps on going, while the others hold until the information is communicated. On the other hand, the observer is late: in this condition, the authors propose a local acceleration of time in order to reach the real system's state.

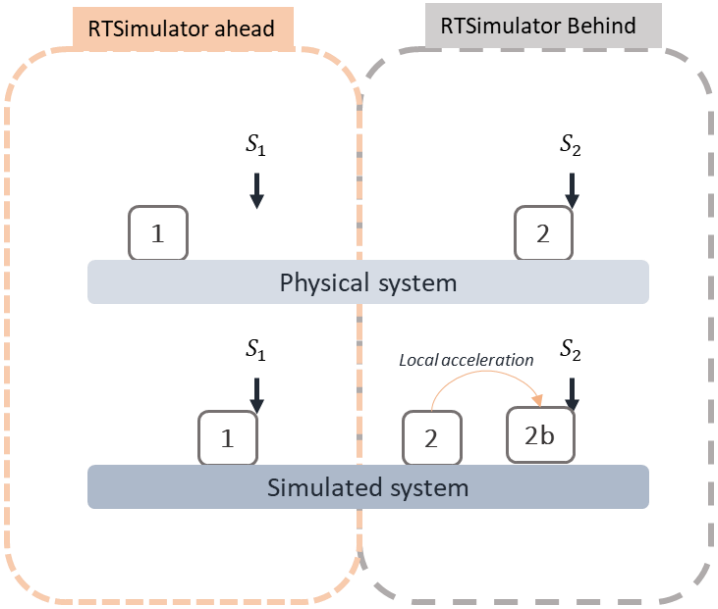


Figure 2.11: A synchronization example, adapted from ref. [Cardin et al. 2011]

Figure 2.11 shows s_1, s_2 which represent the synchronisation points, while 1 and 2 indicate the parcels whether simulated in the digital system or followed in the real one.

Another important aspect to consider, is that information between the two synchronisation points in real system cannot be acquired. A Digital twin correctly validated and synchronised used for shadowing purposes can additionally give information which otherwise would not be acquired from the real system.

From an implementation point of view the paper uses a Petri Net to introduce the synchronisation function principles.

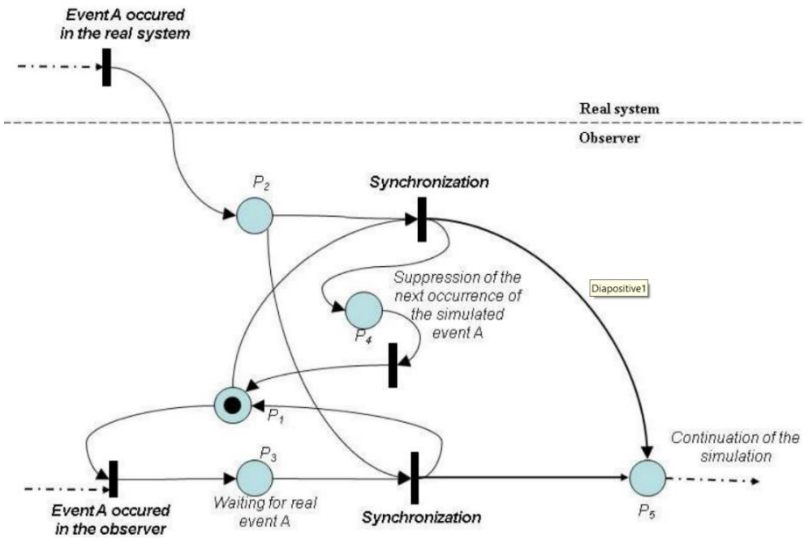


Figure 2.12: The observer synchronization function principles ref. [Cardin et al. 2013]

In Figure 2.12, the token in P1 indicates that the real event in A did not happen, in P2 that the real event in A happened. In P3 the token signals that there is the possibility that the event A has happened. The P4 function is quite particular, in fact it deletes the occurrence of the event and considers as it has happened for the purpose putting the simulated system in a "waiting" mode. Finally, in P5 the simulated system is synchronised.
The illustrated figure allows to introduce the topic of synchronisation points. Defined whether in time or space either on the physical or digital side, they are used as indicator of where the two systems must be able to reflect identical conditions. They are positioned consistently with the applications and the performance expectations.

2.2.2. Hardware in the loop and co-simulation

In the context of control systems, some literature works have developed interesting methodologies. Although the field of application might be characterised by different conditions and inputs, it completely presents the possible solutions to the synchronisation.

The use of a digital infrastructure in supporting the control system, can be incapsulated in the term hardware in the loop. This have been categorised by Wünsch G. [26], distinguishing between HILS (Hardware In The Loop Simulation) approaches for different control cycle times, in the range of 100, 10 and 1 millisecond. From the slower applications to the faster ones, windows operating system and real-time operating system (RTOS) can be used respectively [26].
The authors introduce a new architecture to be used in the contest of the digital factory for the virtual commissioning of production systems. The aim is to be able to model and control a production system at multiple levels, therefore integrate the

several interactions between the process, machine, control system and operator satisfying the requirements of a HILS (cycle time ~ 1ms).

Pritschow G. et al. 2004 [27] states that a machine simulation has to process the output of the control to then manage the inputs within the deterministic control cycle time to be able to satisfy the constraints of time-synchronous and lossless data processing. This Architecture is indicated as *closed simulation architecture* by Schmoll R. [28], shown in Figure 2.13.

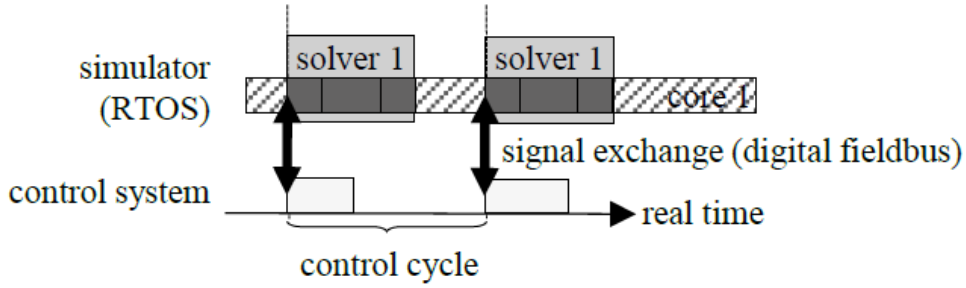


Figure 2.13: Closed simulation architecture ref. [Schmoll et al. 2015]

The individual solver executed as RTOS (Real-Time Operating System), although highly time performant, limits the computing capacity. The control inputs are then submitted to a simulation clock, synchronised with the control system.

A solution to this disadvantage is the introduction of a real-time co-simulation, a term quite often used in other areas then production systems. A simple comprehension of this architecture is the partitioning of the overall model and parallelization of the simulation calculation by coupled and synchronised partial model solvers.

The final result with regards to the application of a real-time co-simulation platform for virtual commissioning of production systems is illustrated by Scheifele et al [26].

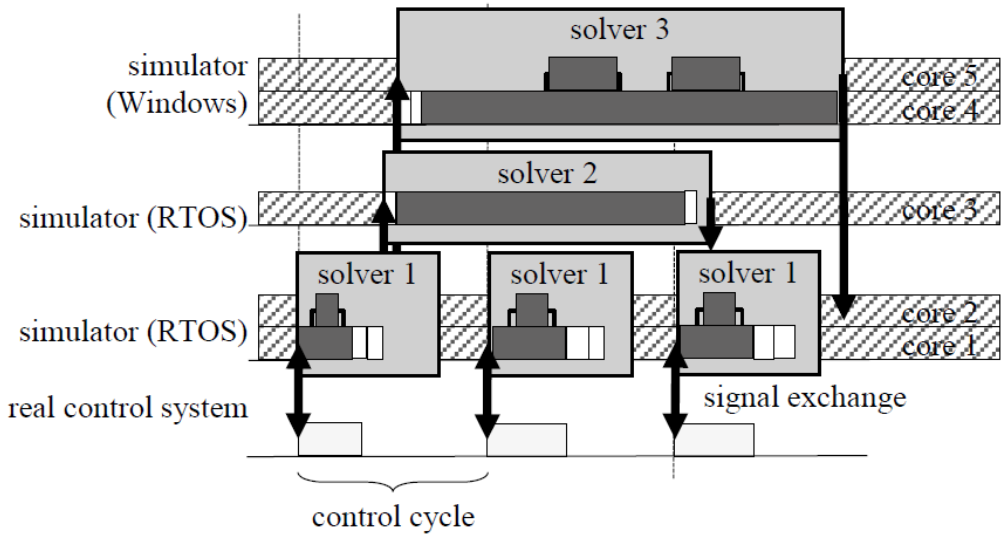


Figure 2.14: Real-Time co-simulation architecture ref. [Scheifele et al.]

While for Offline co-simulation, a virtual time axis for the simulation time is assumed and partial solvers are calculated one after the other. On the other hand, for Real-time co-simulation, each solver is executed parallelly and with independent timing. The numerous solvers must be synchronised satisfying the requirements of HILS. It is although quite interesting to underline the possibility of managing multiple levels of synchronisation, even though for this illustrated logic the purpose is of only controlling the various multidisciplinary systems; however, not much is done with regards of DT applications such as monitoring or interrogative actions.

Zipper H. 2019 [29] Introduces an architecture and an algorithm to synchronise the states of a plant with its Digital Twin while at the same time detecting changes. The concept of synchronisation is handled as an optimization problem for physical plant. This type of methodology is applied to synchronise especially manufacturing processes defined particularly by continuous evolving conditions due to aging, faults, and wear.

An example of possible architecture to develop an online simulation platform having knowledge of the input given by the controller to the physical system is shown in Figure 2.15.

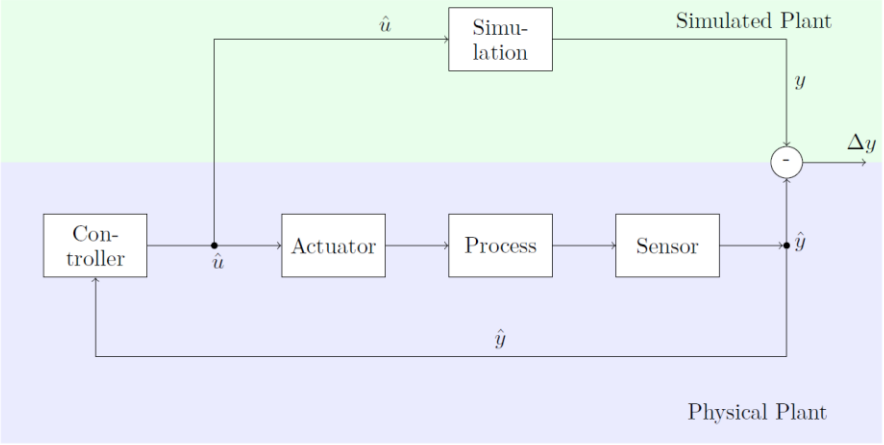


Figure 2.15: Online simulation ref. [Zipper et al. 2021]

With: $\hat{y}$ *: output measured from the physical plant*

y *: output acquired from the simulation model*

$\hat{u}$ *: input of the controller*

Depending on the actual problem to be solved, the signals given to the physical plant to be parallelly inserted in the simulation are chosen. A possible difference between the two sides, thus is used to calculate the error.

An innovative solution developed by Zipper et al. is to tackle the problems of missing feedback to the controller by auto-tuning the simulations inputs and parameters. This is indeed achieved by applying a dynamic optimization focused on minimizing the aforementioned difference, defining the concept of state synchronisation.

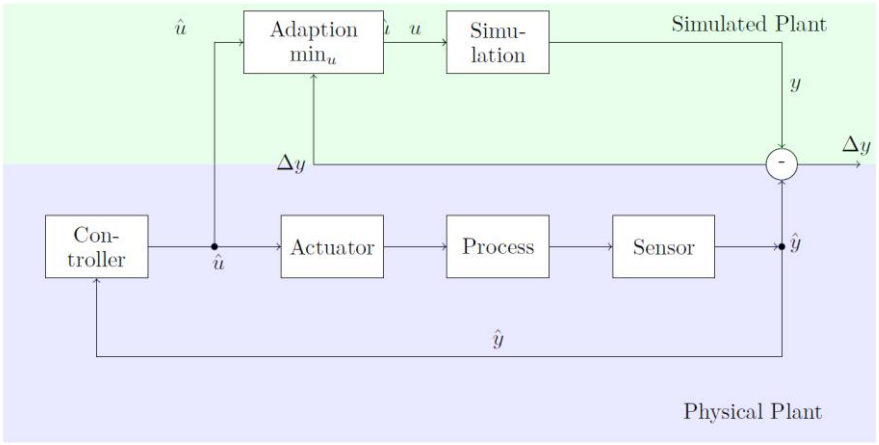


Figure 2.16: Real-time co-simulation ref. [Zipper et al. 2021]

The great potential which characterises this original architecture is the continuous adaptation of u for each time step i, iteration of the optimization algorithm, of the online simulation in order to minimise the difference $\Delta y = y - \hat{y}$.
The overall algorithm which determine the simulation logic is modelled as co-simulation, this allows the possibilities and advantages explained earlier and especially the independency between the two levels, thus using multiple iterations to acquire the outputs from the simulation. In addition, it can be used to work as predictor of behaviours.

To summarise, the inputs coming from the controller are adapted, receiving the correction estimated by the difference of the outputs of the Digital and physical plant. Since unpredictable changes can affect the output of the physical plant, the parameters and inputs in the simulation will be adapted as a consequence to dynamically minimize the error. Results have been shown in context of physical system such as electrical motors and pneumatic cylinder acting on a horizontal beam

2.2.3. Initialisation

Although this work will not concentrate on the methodologies for the purpose of predicting with the highest precision the future behaviours, some aspects of the synchronisation related to forecast analyses in the context of Digital Twin application remain fundamental.
A fairly minor and extensive challenge in the context of digital and real objects alignment is the initialisation of the conditions in the real-time simulation models. Particularly challenging and fundamental to allow the digital model to be ready to give correct results right from the start and avoid any initial transient phase.
Reporting a quite explanatory example from the work of Hanish et al. [7] which study the flow of pedestrians in a train station. Initialising a correct model at the wrong time, e.g., 7 am when the station is closed, would surely give wrong results given that the real time is 8 am and the station is opened.
Hanish et. al [6] introduce an architecture when it comes to correlate synchronised models for forecasting actions, similar in logic to the co-simulation explained in Section 2.2. Parent and child models: the first reflects the real system, while the latter inherits all system variables including values from the parent model. Based on the simulation software capabilities and operating systems, multiple child models might be used to execute forecasts. Their time advance mechanism must be as fast as possible and different from the parent model.

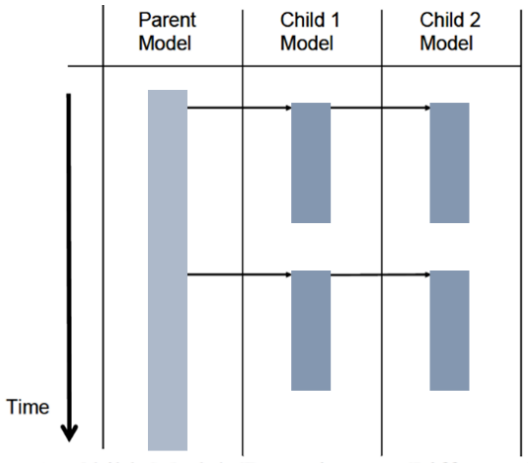


Figure 2.17: Parent and child model architecture, adapted from ref. [Hanish et. al 2005]

Synchronisation in the parent model can be maintained using the aforementioned methodologies, thus permanently or upon request.

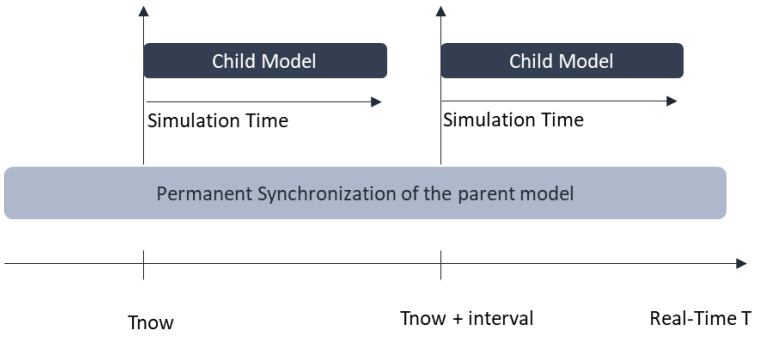


Figure 2.18: Permanent synchronisation,, adapted from ref. [Hanish et. al 2005]

To be considered is that if the synchronisation upon request is used, then it is a major constraint that the child model, entitled for forecasting, finishes its run before the parent model is updated again.

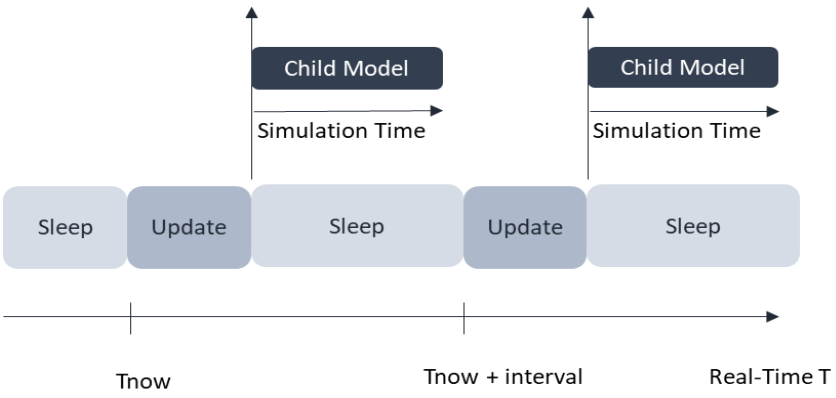


Figure 2.19: Request synchronisation, adapted from ref. [Hanish et. al 2005]

A further comparison when the parent model has been updated is incorporated, measured values from the real system are used to adjust the simulated ones before a forecast is run.

A third and the simplest approach for initialising online simulation models is the generation of a new model to conduct forecast analyses using historical data as in classic simulations.

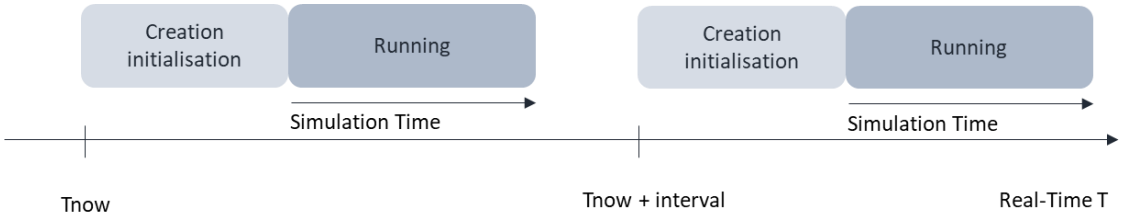


Figure 2.20: Initialisation with model generation, adapted from ref. [Hanish et. al 2005]

A comparison between these three methodologies, underlines that: generating a new model when needed it is surely simpler, however the data inputs must be complete and if not possible to be all acquired then they necessarily have to be assumed compromising the correctness of the results. Alternatively, synchronisation allows for the incompleteness of data, but a higher computational effort must be implemented.

2.3. Literature gap identification

The illustrated methodologies summarised from the literature are certainly a step forward in the research and for this must be taken into account. Furthermore, the works done seem to highlight that implementations have been performed only in certain stages of the DT application.
In Figure 2.21 a categorisation in consistency with the phases of the digital twin is illustrated.

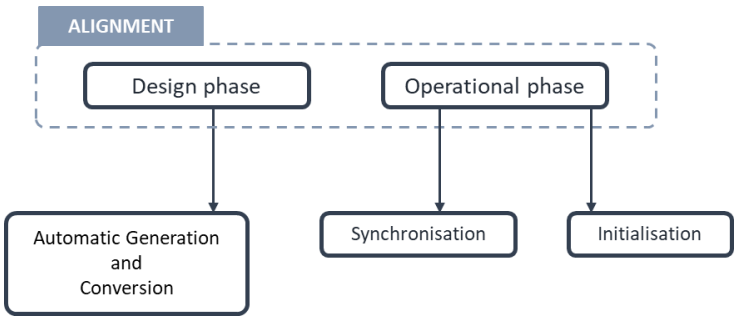


Figure 2.21: Summary of alignment methodologies & phases

In addition, previous works only concentrate on specific actions aim to achieve either monitoring or forecast.
While the methodologies are addressing the same issue with distanced perspectives and results, some common grounds can be highlighted. To commence, all practices underline a great need for interoperability between the various systems involved. Also, the flow of data transferred between the several technologies must follow a protocol which guarantees an effective comprehension of the information. A dynamic storage built or provided by external services is a fundamental component, given the great amount of data and the possibility of looking in the past to gain essential information. The components integrated in a digital twin framework should also reflect highly automated characteristics. Additionally, a previous knowledge about the possible variables involved is at the core of the methodologies implemented.
However, the authors of this study have been able to identify some gaps that are present in the literature. A lack of flexibility in solving the alignment is surely visible. Although each method developed optimally tackle a misalignment issue, they become inefficient when other multiple behaviours come into play affecting the real system. Furthermore, the actions are set to achieve the optimal results given a constant frequency and no decisional process is done, whether to act or not and what strategy to apply. In addition, the resolution for the alignment is focused on achieving the best performance. However, no consideration is done regarding the efficiency of the process, for instance the amount of data used or the relationship between computational effort and results obtained. In conclusion, although some real implementations are found in the literature, they are often referred to other context than production systems.
In the following chapter 3 a new methodology is introduced with the aim of tackling the explained limitations and contribute to the scientific community.

3. Proposed methodology for the alignment of the DT

3.1. Methodology overview

In this section a methodology is presented that aims at developing an adaptable and intelligent alignment function, which collects data from the physical and digital objects and acts on the DT. A scheme of the proposed alignment methodology is pictured in Figure 3.1.

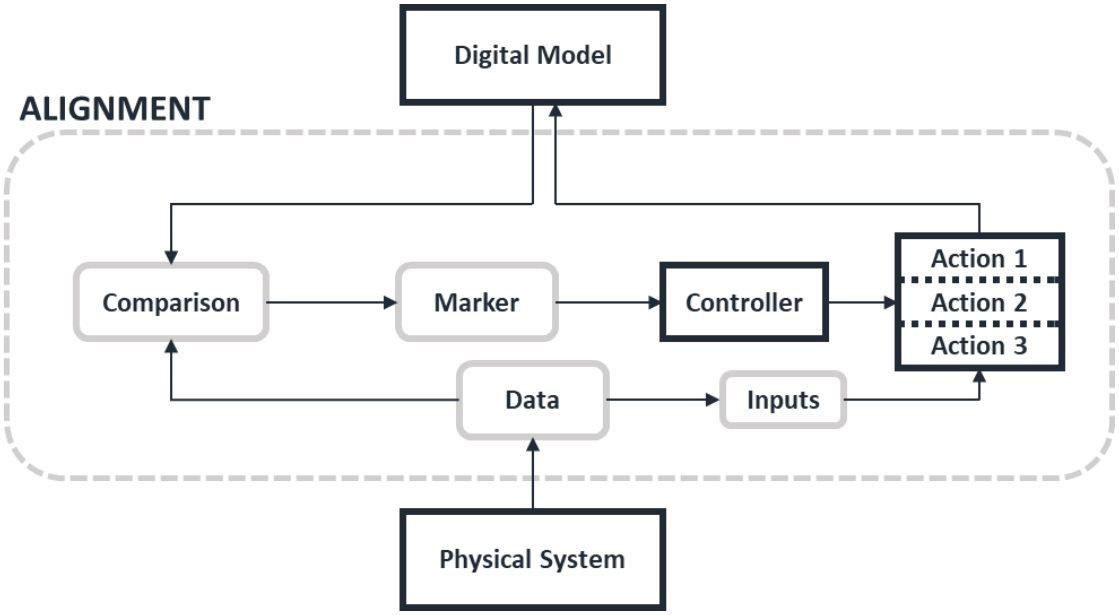


Figure 3.1: Scheme of the proposed alignment methodology

Some assumptions must be made regarding the data collected from the physical system, in fact information acquired are considered as:

- Correct: data acquired by the physical system are assumed as valid, thus lacking in inaccuracy or noise.
- Incomplete: the data collected does not represent all the information regarding the state of the physical system at a given moment in time.
- Intermittent: the data acquisition points in the production line are limited, therefore the physical system collects data exclusively when they are

encountered. They also act as "synchronisation points" defined as key moments in time and space where comparisons between the two entities can be performed.

This study focuses on solving three major kinds of misalignments: the model logic, the model's instant conditions and the model's stochastic behaviors. There could be other types of discrepancies between the two objects, but this framework would be still applicable, as long as a corresponding aligning action is implemented.

- The logic of a production line is described by the plant's layout, the number, and types of resources, and how the workpieces flow through them. The digital model must reflect these features to be used for performing useful and correct analysis.
- The instant conditions represent a static snapshot of the actual status of the production system at a given moment in time.
- The stochastic behaviors are defined by the parameters which model the time duration of the different activities characterizing a production system. They are obtained through a statistical analysis on the processes and afterwards are given as input to the digital object in the form of stochastic distributions.

The problem of misalignment is tackled with a methodology that enables a continuous control on the status of the digital model in relation to the real counterpart. For an evaluation of the alignment between the two objects, data from the physical system are collected and used in comparison procedures. The results are then summarized in a series of markers. Based on these, a component named Controller chooses if and which action needs to be deployed to re-align the digital model. The comparison techniques can vary depending on the kind of misalignment they analyze. In this work the procedures used are three: one for the logic of the model, one for the stochastic behaviors and one for the instant conditions. These measures elaborate a single marker that indicates whether the physical system and the digital model are aligned and if that is not the case, it indicates the type of discrepancy between the two.
The actions illustrated in Figure 3.1 to re-align the model are managed by the controller and are:

- Action 1: Model update, when the logic of the digital object no longer replicates the behavior of the physical system.
- Action 2: Synchronisation, when the instant conditions of the digital model differ from the ones of the real counterpart.
- Action 3: Input model update, when the stochastic behaviors implemented in the digital model are not able to describe anymore those characterizing the physical system.

This solution is innovative because it targets multiple kinds of misalignments, not only specific ones, and the control system calls the different actions only when they are needed. This way the computational effort is efficiently managed.

3.2. Comparison procedures and markers

Different comparison procedures are used to assess the alignment status between the physical system and the digital model. In this work three comparison procedures have been chosen to determine the level of discrepancy between the two objects:

- The logic validation compares the topology and general behavior of the two systems by comparing the real with the simulated performance obtained with a trace-driven simulation.
- The input validation functions in the same way; however, it modifies the trace by correlating it with the digital model's stochastic behaviors. It checks whether the stochastic behaviors of the two objects are equal or not.
- A synchronisation check determines if the instant conditions of the digital model are the same as the ones of the physical system. This measures the discrepancy using the results of a particular positioning function. In appendix A.2.4 an example is given.

The result is that these markers convey a binary indication whether the DT alignment is validated particularly to a specific factor.

3.3. Controller

The controller works on pre-defined request frequency f_{req}, in which every cycle it collects the input information in the form of markers and then it decides if deploying the corresponding alignment action. The request frequency can vary, depending on the use case of the framework, and also the required performance the platform has to satisfy.

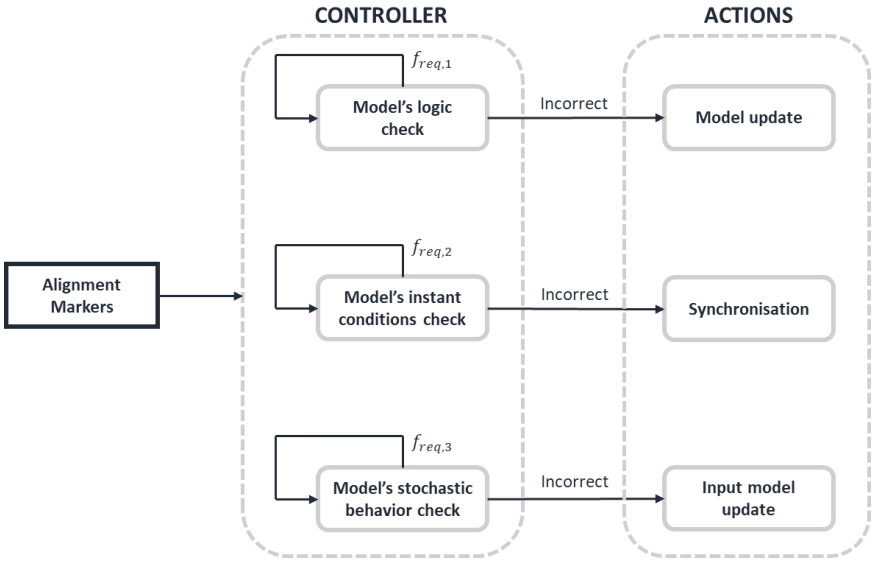


Figure 3.2: Flowchart of the behavior of the Controller in the case of un-coupled request frequencies

The check on the different markers can also be de-coupled, depending on the system analyzed and if the process of creating the marker is time-consuming. For example, the check on the model's logic can happen less frequently compared to the check on the model's instant conditions, because the latter evolves far more frequently than the former. In this case there will be multiple request frequencies $f_{req,i}$, as shown in Figure 3.2. The markers can also be neglected, this happens in the case in which there is no information regarding the alignment status of the physical and digital systems. In this configuration the different actions are always deployed, but since there is no comparison between the two objects, it remains uncertain the effectiveness of the alignment process.

In the next sections the re-aligning actions are explained, together with a running example to better explain the different procedures. The system in question is composed of two servers in series, as shown in Figure 3.3(a).

3.3.1. Model update

The simulation model automatic update is composed of two steps. First, a graph model containing all the necessary information for replicating the physical system's logic is generated. Consequently, the graph is converted into a simulation model using a conversion algorithm. The result of each phase of the two servers' system is shown in Figure 3.3.

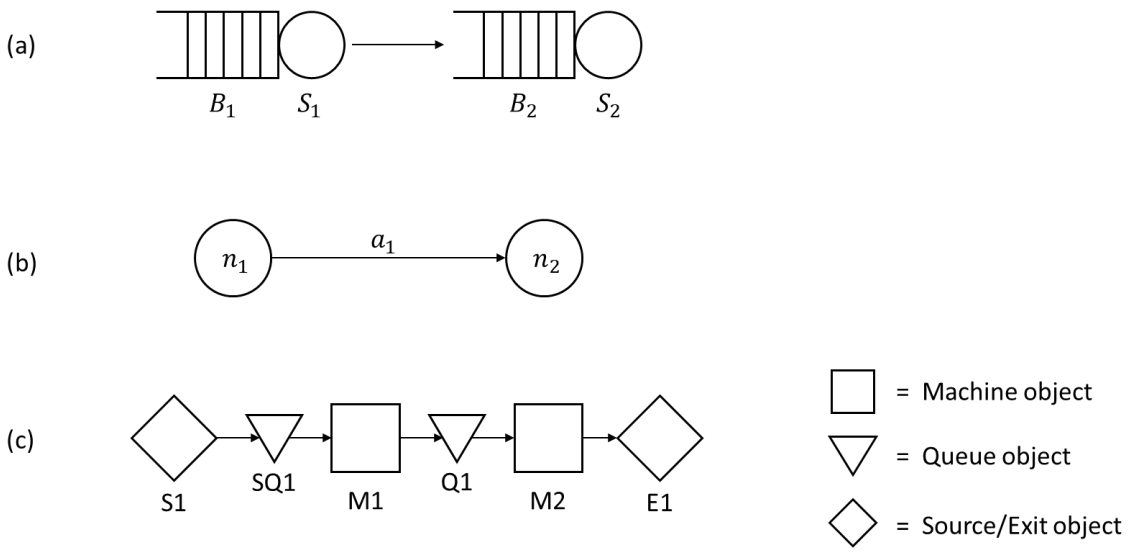


Figure 3.3: Original two-server system (a), representation of the generated graph model (b), representation of the converted simulation model (c).

3.3.1.1. Model Generation

The model generation procedure is based on the work of Lugaresi et al.[13], where a model discovery procedure generates a graph model Ω, which is defined as a tuple Ω=(N, A) where N is the set of nodes and A is the set of arcs. Each node represents one activity which compose the production system, while arcs are the connections between the different activities. Each node $n \in N$ and each arc $a \in A$ is defined by a set of attributes, summarized in Table 3.1 and Table 3.2.

Node parameter	
P_n	Predecessor node set
S_n	Successor node set
k_n	Capacity of a node
ϕ_n	Frequency of a node

Table 3.1: Node parameters notation

Arc parameter	
$\eta_a = (n, m)$	Nodes connected by an arc
c_a	Buffer capacity of an arc
f_a	Frequency on an arc

Table 3.2: Arc parameters notation

The relevant parameters of each node are:

- Predecessor and successor node sets, P_n and S_n, which indicate the other nodes in the model which the node in question is connected with.
- The capacity k_n of the node, which illustrate the maximum number of workpieces that can be processed at the same time.
- Frequency ϕ_n of the node, which depicts the number of times the activity was found in the event log.

The relevant parameters of each arc are:

- The tuple η_a, which contains the two nodes connected by the arc.
- The buffer capacity c_a, which indicates the maximum number of workpieces that stayed in the arc at the same time.
- Frequency f_a, which is the total number of workpieces that flowed through the arc.

In Figure 3.3(b) is shown the resulting graph model of the two-server system, composed of two nodes, n_1 and n_2, representing the two servers, and the arc a_1 connecting them.

3.3.1.2. Model Conversion

The graph model Ω is then converted into a simulation model. The conversion method takes each element of the graphs and creates the suitable objects to populate the simulation model. These are: Part, Machine, Queue, Source and Exit, all composing the digital model, as shown in Figure 3.4.

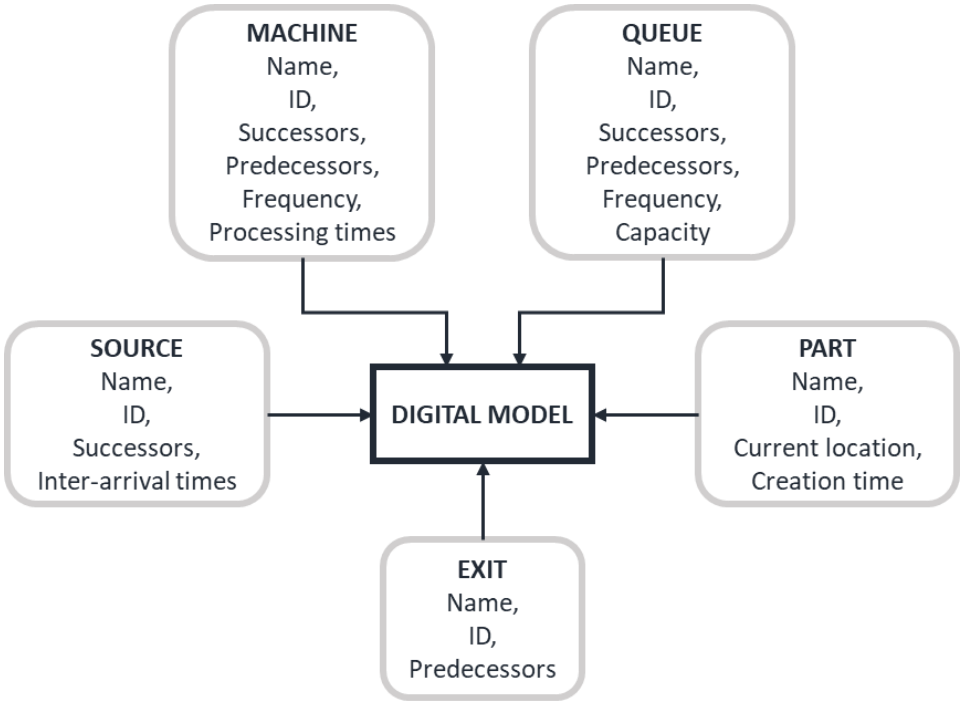


Figure 3.4: Diagram of the Digital Model components

- The Part object represents the workpieces that flow through the production system. Each Part object has a set of attributes that summarize its characteristics: id and name of the workpiece, its current location in the model and the timestamp associated to its creation.
- The Machine object is used to simulate the behavior of the different resources that perform activities on each workpiece. The processing action is summarized in a time delay that simulates the activity. Each Machine object is characterized by its name and ID, the other objects it is connected to, its frequency value, which is used to determine the probability of receiving a workpiece if it is part of a parallel system, and finally information regarding the processing times (divided in loading, unloading and actual processing times), in the form of a distribution and its parameters.
- The Queue object is used to replicate the behavior of a buffer, which is a resource that holds a certain number of workpieces. Every Queue object holds information regarding its identification (ID and name), its capacity, which is the maximum number of parts it can hold, the other object it is connected to, its scheduling rule, and finally its frequency value, which has the same use as in the Machine object.
- The Source object generates the Part objects in the digital model, is used to simulate the inter-arrival of workpieces into the system.
- The Exit object removes the Part object from the digital model, simulating the workpieces exiting the production system.

The methodology for the automatic conversion of the graph model Ω into a simulation model is composed of a series of steps shown in Figure 3.5.

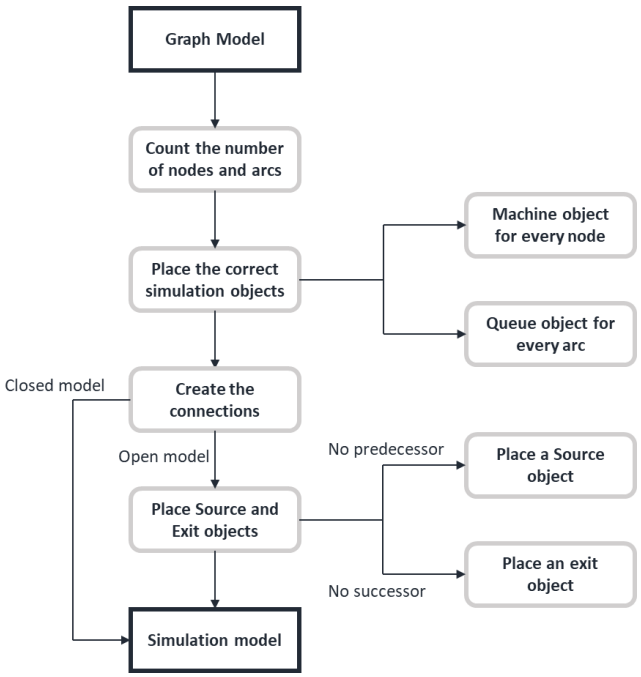


Figure 3.5: Flowchart of the model conversion procedure

The first phase consists of counting the number of nodes and arcs that compose the graph model. These values are used to create the correct number of simulation object to place in the digital object. The nodes represent each activity that takes place in the production system, therefore in a simulation model they must be represented by an object which is able to replicate the processing time of the activity. Consequently, for every node in the graph model the conversion algorithm creates a Machine object. On the other hand, the arcs represent the connections between the different nodes, but they also have the ability to hold a certain number of work-pieces. Because of this reason, they must be represented by a buffer-type simulation object, that is able to replicate the behavior as a temporary storage of workpieces. Therefore, the conversion procedure creates a Queue object for every arc in the graph model. The next step is composed of the analysis of the connections between the different objects in the simulation model. The arc represents the connection between the nodes; therefore this information is used in the conversion procedure to define the correct predecessor and successor object for each Machine and Queue. In open models, where workpieces enter the system, are processed, and then exit, there needs to be something that simulates the arrival and the disposal of parts. If a node does not have any element in the predecessor set P_n, then a Source object will be placed in the digital model and connected before the corresponding Machine. In the same way, if the successor set S_n of a node is empty, an Exit object will be placed in the simulation model after the corresponding Machine. In Figure 3.3(c) is shown a representation of the simulation model generated and converted from the original two-server system. The two nodes are converted into two Machine objects, M1 and M2, while the arc is replaced by one Queue, Q1. Since n_1 has no predecessor nodes, the Source object S1 has been added to the model, together with SQ1. Furthermore, and Exit object E1 has been placed after M2, because n_2 doesn't have successors.

The final output of the conversion procedure is a discrete event simulation model of the physical system in analysis. The model can be executed in two different ways, either through a stochastic or a trace-driven simulation. In both cases some parameters must be specified before. In the case of a standard stochastic simulation, the distributions generating the processing times of the different Machine objects must be given as input. The length of the simulation and the number of replicates is also necessary. Considering trace-driven simulations, where the processing time values of the Machine objects are taken from a list given as input, the number of replicates is just one, while the simulation runs up until there the processing time values are finished. In this mode also hybrid simulations can be performed, where some Machine objects take processing time values from a distribution, while others use values given as input. In both cases the output of the simulation are the average throughput and system time, the number of parts produced; together with the resources' utilization, starvation and blocking ratio.

3.3.2. Synchronisation

The second action called by the controller is the synchronization which addresses the re-establishment of the instant conditions of to the physical system into the digital model. These include the immediate status of the resources and the workpieces that compose the production line, as well as the information regarding the instantaneous performances that the real system has been able to obtain. In particular, the workpiece position and the actual number of parts produced are the subject of synchronisation.
The procedure is structured as a request type synchronisation, as illustrated in Figure 3.6.

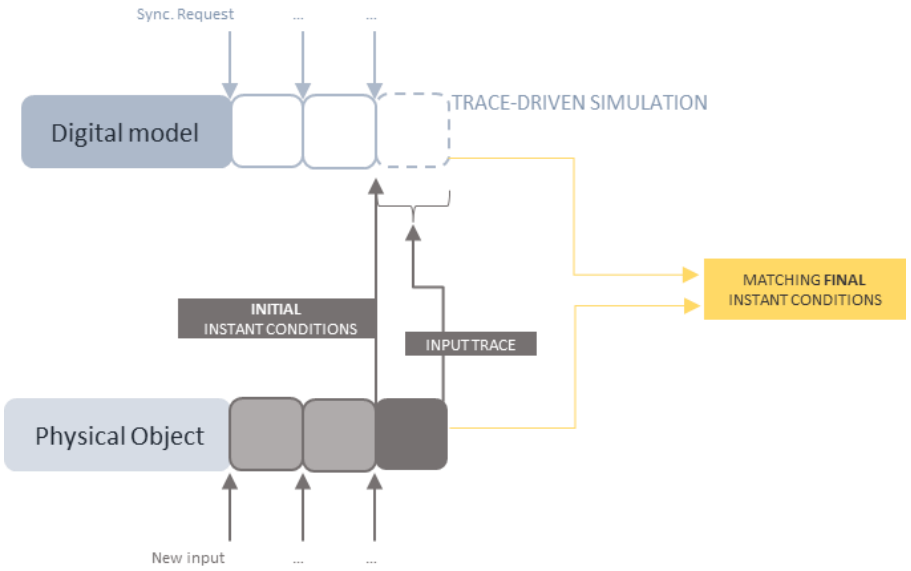


Figure 3.6: Logic of synchronisation method

The dark grey box in the physical system represents a newly acquired processing parameter, e.g., a processing time from a machine of the production line, thus the digital model is run with a trace driven simulation. To start the simulation, the initial instant conditions found before the last processing parameter are used to initialize. The processing parameters are used as input trace. Ideally when the digital system has simulated the real behaviors, it should be able to reflect the final conditions also find in the real system. This mark instantaneously that the synchronisation has been reached. In Figure 3.7 is shown the result of the procedure in the two-server line. Before the synchronisation the instant conditions of the simulation model are different from the ones on the real system, while after both the positioning of the parts and the number of produced parts are equal.

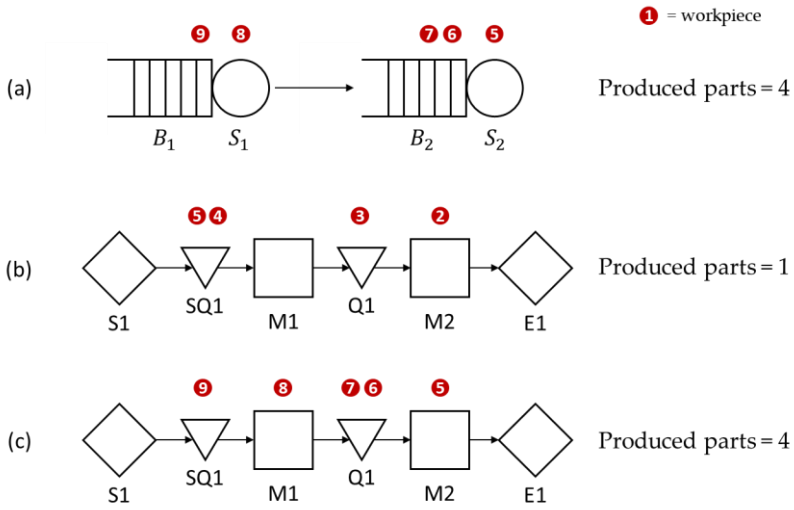


Figure 3.7: Two-server system instant conditions (a), digital model instant conditions before synchronisation (b), digital model instant conditions after synchronisation (c)

3.3.2.1. Synchronisation implementation

When a synchronisation request is performed at time T, the procedure is as follows:

1. The trace of the most recent processing time values of the entire system is acquired. The amount of data used determines a time interval dt.
2. The initial instant conditions at T-dt are computed using a part positioning function, introduced later in this section.
3. A trace-driven simulation is executed using the gathered traces, from T-dt to T.
4. The final position of the digital model at the instant T, and the respective digital performances are therefore obtained.

Given the type of system analysed, during operations the workpieces can either be: in a machine, where they are being processed, in a buffer, or in transport between two resources. A generalisation is introduced in the positioning procedure, as shown in Figure 3.8, where the production system is divided into zones, each include a machine and its downstream buffer.

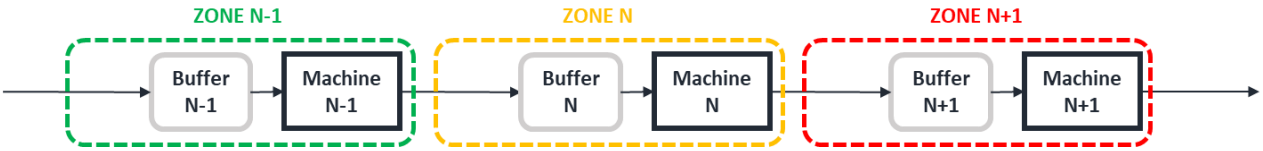


Figure 3.8: General production system layout and zones division

In order to compute the position of the workpieces from the information provided by the event log, the function acquires the last event for each part id still in the system. If the relative tag is a start, the part is placed in the buffer of the zone containing the

machine, otherwise if it signals a finish the part is placed in the subsequent zone. This procedure allows to identify the positions for both the real and digital system.

To summarise, the factors required to conduct this type of procedure are the input process parameters, which allow to describe the evolution of the physical system, and the initial conditions. The former is extrapolated from the physical system and are dependent on the type of simulation tool implemented as well the purposes and digital twin performance' expectations. They are required to completely be able to let the simulation advance and reach the physical system conditions. The latter have to be able to depict the instantaneous conditions that are essential to initialise the digital model and thus make it avoiding the initial transitory phase. This type of methodology is consistent with the so-called Online simulation.

3.3.3. Input model update

The last action tackles the issue related to the misalignment reflected by the deviation in the stochastic behaviors. This is certainly of great influence for forecast purposes, where fitted parameters are used to predict future conditions.

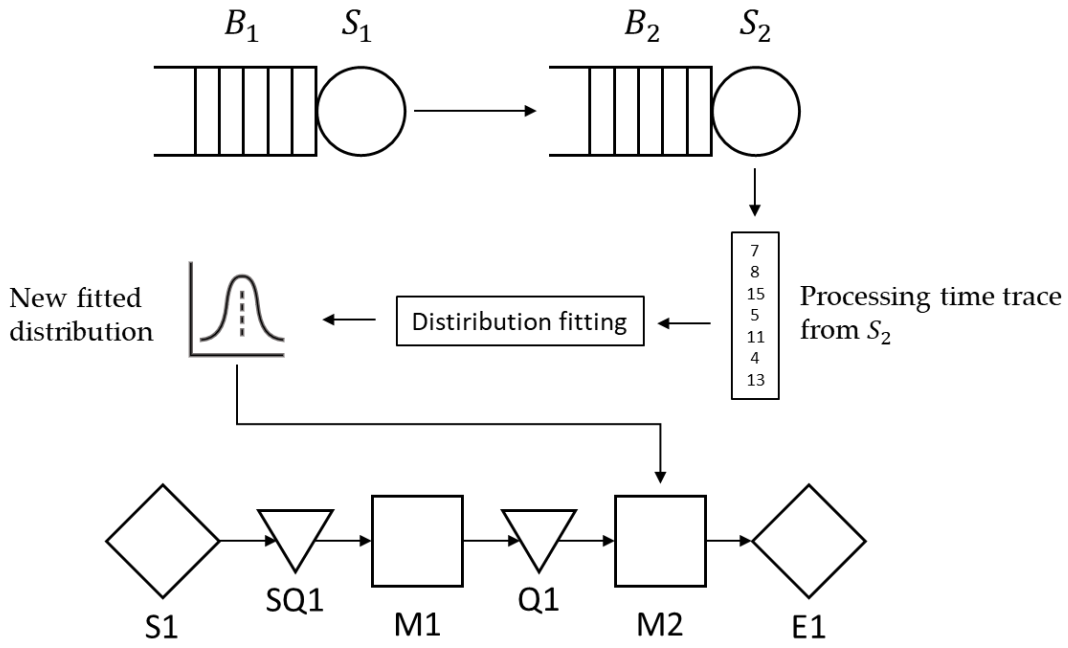


Figure 3.9: Example of the input model update method on the two-server system, where a new distribution replicating the processing times of S_2 is fitted and implemented in the corresponding element of the simulation model.

The historical data from the physical object are acquired and consequently fitted in order to find a stochastic pattern. This is updated into the digital model which is then ready to give any predictive results. The action of input model update is entirely determined by the controller not only in terms of frequency but also on the length of

historical data acquired. Figure 3.9 illustrates an example of the input model update logic on the two-server system. Specifically, the procedure works in three steps:

1. A trace containing the most recent processing time values of the machine is acquired.
2. The distribution that better fits the acquired values is found, by applying the Maximum Likelihood Estimation (MLE) method.
3. The new distribution is implemented in the correct object of the simulation model.

3.4. Services

The methodology introduced in the previous sections makes the digital twin meet the alignment requirement, enabling the services of monitoring, which provides real-time analysis of the current performances, and forecast, to predict future behaviors. In Figure 3.10 are shown the conditions that must be kept aligned by the controller for correctly performing each service.

Figure 3.10: Alignment conditions that the controller must satisfy to perform the two services.

3.4.1. Monitoring

The monitoring service consists of using the simulating capabilities of the digital twin to obtain the real-time performances of the physical system. Trace-driven simulations are performed using the actual processing parameters collected from the physical plant. The result is a more complete knowledge of the current status of the production system, allowing the calculation of performances that are too costly or impossible to obtain directly from the real object. An example would be the computation of the machine utilization values. As shown in Figure 3.10, for the monitoring action to be correct, both the logic and the instant condition of the digital model must be kept aligned by the controller.

3.4.2. Forecasting

The forecast service consists of the prediction of future performances of the production system. The main benefit that an aligned Digital Twin provides is the ability to perform real-time simulations at any time, allowing the prevention and reaction to unpredictable and disruptive events. As an example, an aligned digital model can perform what if analysis after a failure of a machine in the manufacturing line, by simulating possible solutions and implementing the better performing one. As Figure 3.10 shows, the requirement for the correct activity of the forecast function is the alignment of the logic, the instant conditions, and the stochastic behaviors of the digital model.

In conclusion, the controller is built to manage the alignment of the digital and the real object through the whole life cycle. The gaps found in the literature are satisfied by the capabilities and characteristics of the conceptualized solution. As a first contribution to the literature, the control logic is surely very flexible. In particular, the controller triggers the several actions in charge of realigning the digital model. The decisional process followed by the component is based on markers which summarize the various degrees of misalignment. In addition, the use of indicators allows to act only when necessary and save computational effort.
Although the methodology introduced is applicable to several kinds of production systems, it needs to be furtherly adapted based on the physical system's features and the digital tools' characteristics.

4. Alignment components implementation

This chapter introduces the implementation of the proposed methodology. This acts as a cyber-physical system including multiple entities interconnected. On one side the physical system, a lab-scale model which simulates the behavior of a real production line. On the other, the components correlated to the Digital Twin. The connection between the two objects is made possible by supporting services.

4.1. Physical system

The physical system is built using Lego MINDSTORMS. This and other types lab-scale models are available in the "Digital Twin lab" at Politecnico di Milano. The adoption of this kind of system has the advantage of being able to test multiple configurations of production systems, by changing the number and position of the different elements. This capability, together with the simplicity in its use and upkeep, is the reason for its selection.

The model is composed of stations connected by conveyors, which act also as buffer for the circulating pallets. Each component of the physical system is built using the MINDSTORMS components pictured in Figure 4.1.

a) b) c)

Figure 4.1: Lego MINDSTORMS components: motors (a), optical sensors (b), EV3 (c)

Stations include three sensors, pictured in Figure 4.1 (b), a pusher, and a dedicated conveyor. Figure 4.2 pictures a station. Both the pusher and the conveyors are moved by motors, shown in Figure 4.1 (a).

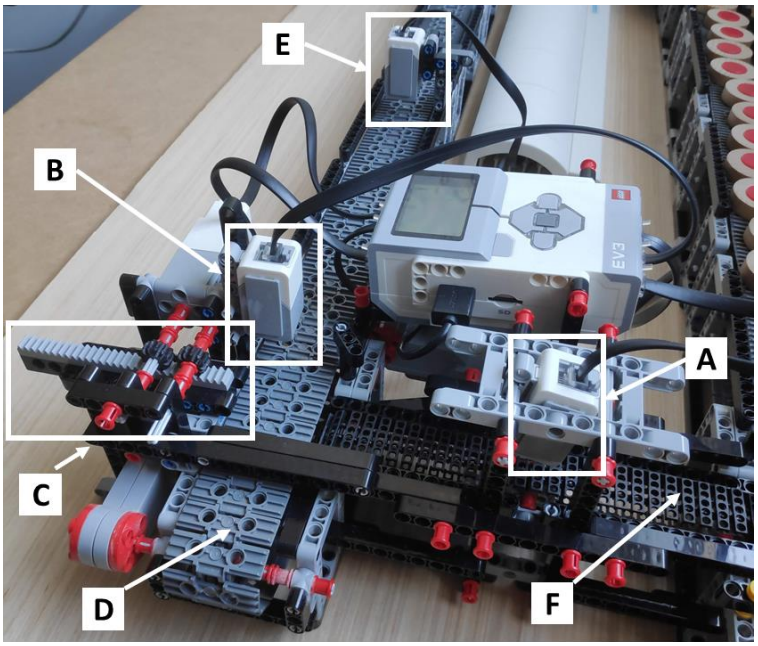


Figure 4.2: Example of a station and its components: the internal sensor (A), the upstream sensor (B), the pusher (C), the buffer sensor (D), the transport conveyor (E), and the internal conveyor (F)

The functioning of the stations aims to reproduce the behavior of a real processing machine. Each one has four states: idle, working, blocked, and failed. The logic of the station is explained in the flowchart pictured in Figure 4.3.

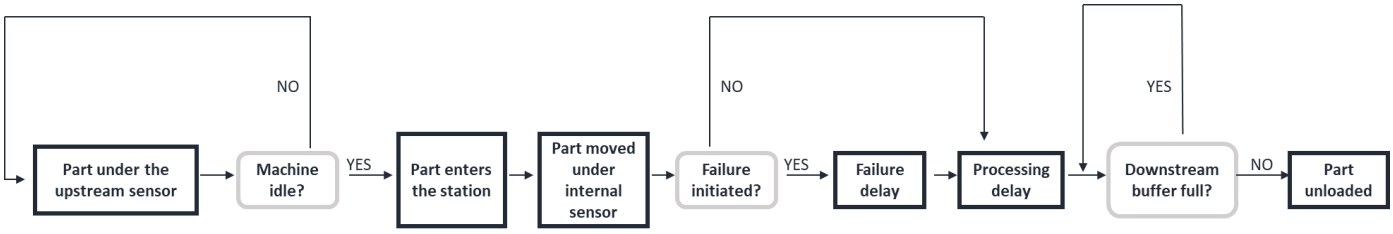


Figure 4.3: Station logic flowchart

Buffers are modelled as the areas between the upstream and the buffer sensor, and the distance between them defines the capacity. Pallets are represented by wooden discs with a diameter of 35mm. They present a colored marking on the top, which is necessary for the recognition by the sensors. The MINDSTORMS components are controlled by programmable logic controllers, pictured in Figure 4.1 (c), also called EV3.

For the implementation of this work a two-station closed-loop model has been chosen, pictured in Figure 4.4. The system is characterized by: processing of a part one at the time, Blocking After Service (BAS) discipline, buffers capacity of 8 pallet, and 12 pallets circulating the line.

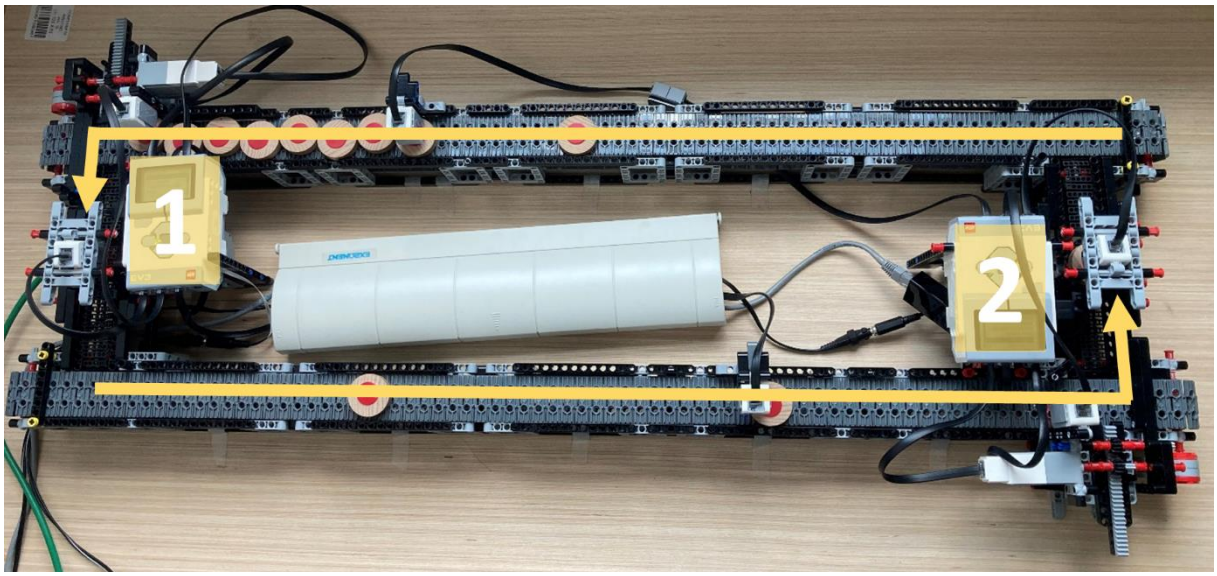


Figure 4.4: Lab-scale model of closed-loop, two-station line

4.2. Communication logic

Communications are managed by the EV3, which receives the signals from the sensors and controls the actuation of the motors. A scheme of the communication logic of an EV3 is shown in Figure 4.5. The device's logic requires a configuration input, it also sends out the information it collects. The EV3 bricks and the PCs are connected to a network router, the connection is established using a Secure Shell Host (SSH) protocol. The communication channel between the physical system and the other components is achieved using Message Queue Telemetry Transport (MQTT), a publish-subscribe network protocol, where each message is structured in JSON format. Mosquitto is used as broker.

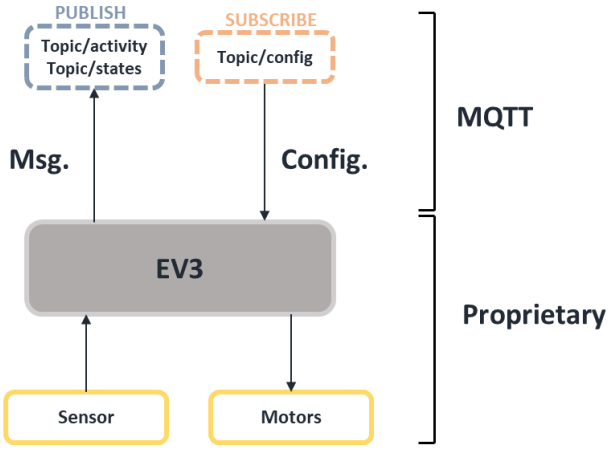


Figure 4.5: EV3 messaging logic scheme, with MQTT and proprietary connections

The EV3 is controlled through a Python script which determines all the logic of the inbound and outbound messages [29]. The latter interprets the information received by the sensor, specifically the buffers level, the stations' state, and the events. For

instance, a message published on the topic/activity, corresponding to an event on the system, can be: *{"id": 16, "activity": 1, "tag": "s", "ts": 1578443734.169}*, where:

- *"id"* is used to identify which workpiece is being processed.
- *"activity"* is a number corresponding to the machine which is doing the processing.
- *"tag"* identifies the type of activity. It can either be "s" or "f", corresponding respectively to the start and finish of the processing operation.
- *"ts"* is the timestamp of the event, saved in UNIX format.

The inbound messages are the configuration files, which are sent to each EV3 when the system is started. These parameters define the logic of the station, specifically the processing times, the failure probability, and the repair times. These values of the stochastic quantities (i.e., the processing and failure times) are generated by a RNG contained in the Python script, which uses as seed value defined in the configuration file. In Table 4.1 an example of configuration parameters is shown.

Component	Input
Station	1
Next station	2
Conveyor speed	-200
Pusher Speed	500
Station Speed	200
Unload time	3

Parameter	Input
Distribution Cycle time	Deterministic
Dist. Parameters	12
Failure prob.	5
Distribution repair time	Deterministic
Dist. R. Parameters	30
seed	120

Table 4.1: Configuration parameters

4.3. Digital architecture

Figure 4.6 depicts a class diagram containing all the main components of the digital architecture and their connections. These are developed using different software platforms, all interconnected and controlled through the use of Python script. The simulation capabilities are provided by Arena Rockwell and ManPy. Furthermore, InfluxDB is used to provide the database. MobaXTerm and Mosquitto handle the MQTT messaging, while a dashboard has been developed using Grafana.

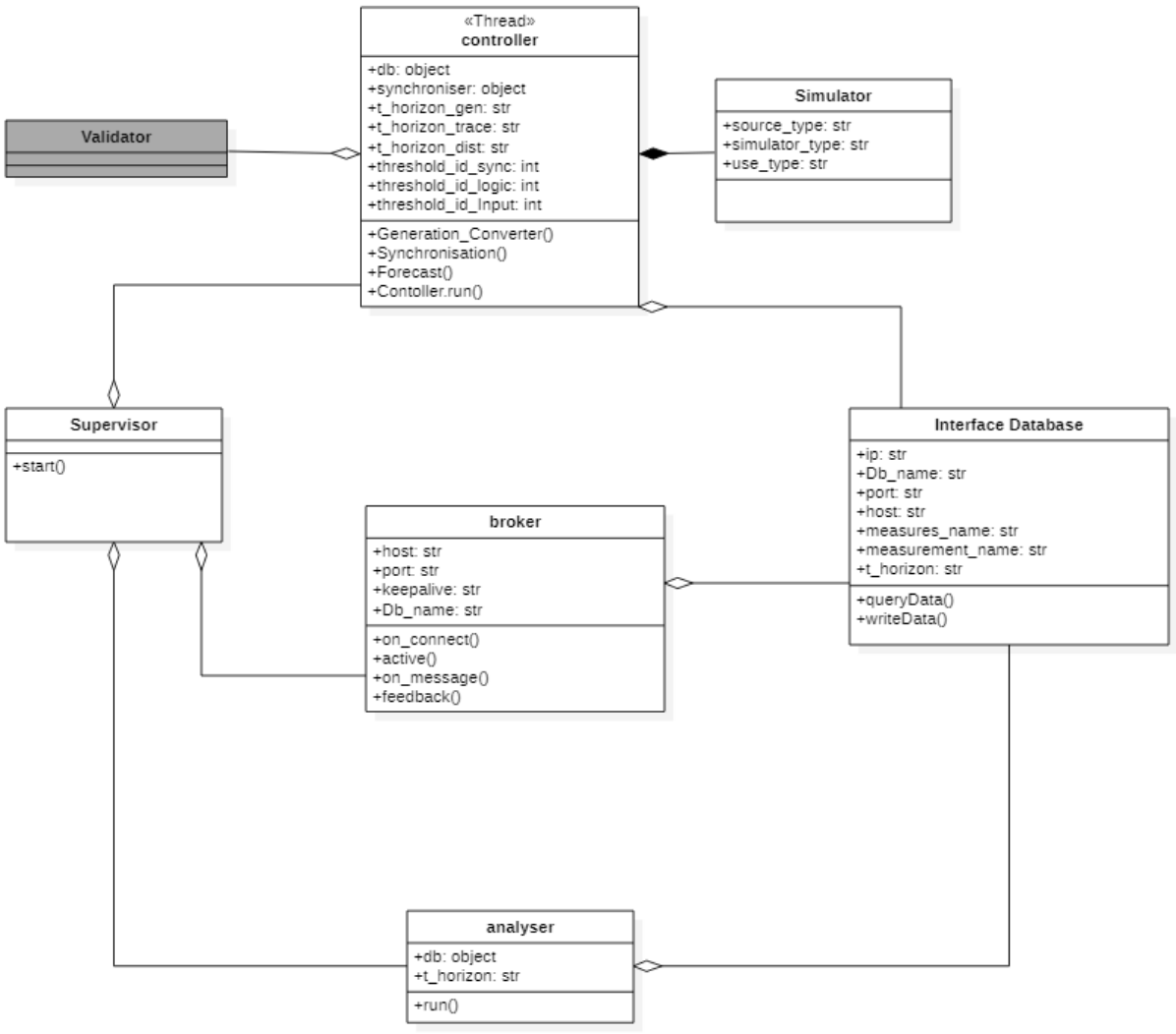


Figure 4.6: Supervisor class diagram

The Supervisor class manages the functioning of all the other components. With the method "start()" it initializes all the necessary objects and activates them. The database stores all the data coming from the physical and digital systems. The broker object has two functions: it manages the communication from and to the physical system and it also populates the event log with collected messages. Its attributes define the IP address and port number of the machine where it is located, while the methods perform the two functions. Raw data from the real system are manipulated to compose in real time derived performances by the Analyser. The components starts with the method "run()", and works on a frequency defined by the "t_horizon" attribute. Finally, the InterfaceDB class is used to handle the connection between the database and the other components. The Validator component is in charge of performing the comparison procedures and will be furtherly explained in section 6.
A flow chart of the supervisor is illustrated in Appendix A.2.2. A dedicated description of the controller and database is provided in the following chapters.

4.3.1. Controller

In Figure 4.7 is shown the class diagram of all the components that make up the Controller. With "run()" the object starts its function, calling the methods which check on the markers and, if needed, deploy the aligning action. "Model_Update()" checks the logic validation marker, generating and converting a new simulation model if needed, "Synchronisation()" controls the instant conditions and potentially performs a synchronisation, and "Forecast()", which checks on the stochastic parameters, updates them when requested, and performs predictive simulations.

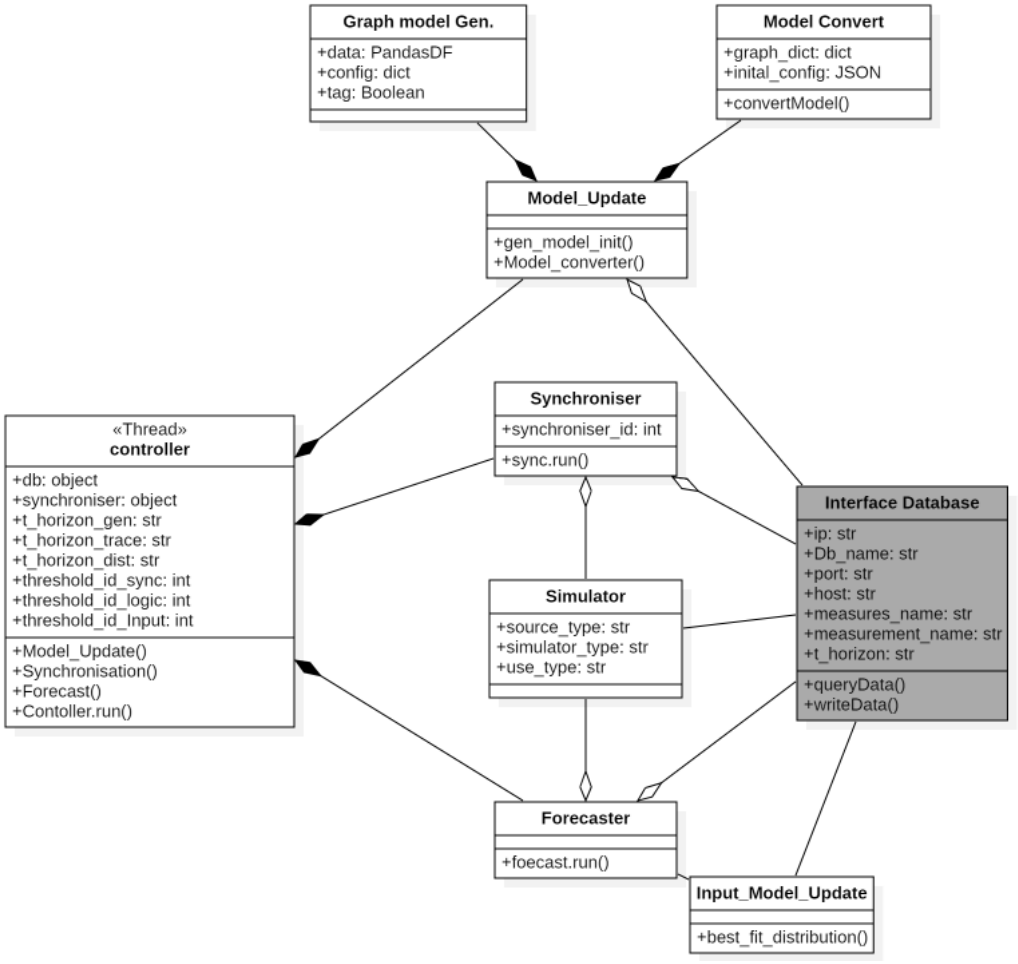


Figure 4.7: Class diagram of the components that make up the controller

- The Simulator component is in charge of performing simulations and operates using either ManPy or Arena Simulation software. In the first case, the model is automatically generated and converted, in the second it has to be purposely built for each physical system. It can perform either trace-driven simulations, where the processing parameters are given as trace, or using stochastic distributions.
- The implementation of the model update function is done through two classes, respectively the Graph_Model_Generator and the Model_Converter, aggregated in the Model_Update component. The procedure starts by

collecting datapoints from the event log included in the database. The "t_horizon_gen" parameter defines the amount of data to query. The update method then takes as input this information and a configuration file, containing possible manual modifications, and creates a graph model in the format of a JSON file with the method "gen_model_init()", which is then uploaded on the database. The method "model_convert()" is then activated, which produces a simulation model in the same format, also uploaded to the database. In Section 4.3.1.1 the flow chart illustrates in details the procedure to achieve a correct update.

- The Synchroniser is in charge of re-aligning the instant conditions of the digital model. It is activated by the Controller and the attribute "t_horizon_trace" determines the time interval dt in the procedure explained in section 3.3.2. The final position and the performances obtained are then uploaded to the Database. Section 4.3.1.2 proposes a flowchart to explain furtherly the process.
- The Forecaster component handles the update of stochastic parameters, with the method "best_fit_distribution()" in the Input_Model_Update component, and predicts the future performances of the line. The attribute of the controller "time_horizon_dist" determines the length of the processing times trace to query from the Database. Furthermore, before each forecast, the instant position computed by the synchronisation must be downloaded. With regards to Input_Model_Update the Section 4.3.1.3 illustrates in detail the process involved.

A sequence diagram of the controller and broker is depicted in Appendix A.2.3. A2.7.

4.3.1.1. Implementation of Model Update

The update of the model's layout and logic is performed through procedures implemented in Python, theses are illustrated in the flow chart of Figure 4.8.
The event log is first used by the Model Discovery Procedure to generate a graph model JSON file. This is read and to initialize the ModelConverter a configuration file is introduced. The method of convertModel() is activated by first having computed the number of nodes and arcs from the graph model generated. These are used as explained in Section 3.3.1.2 to create the Machine, Queue and Exit objects. Once the executable model is finalized with the connections between the objects, the simulation Model JSON file is uploaded in the Database.

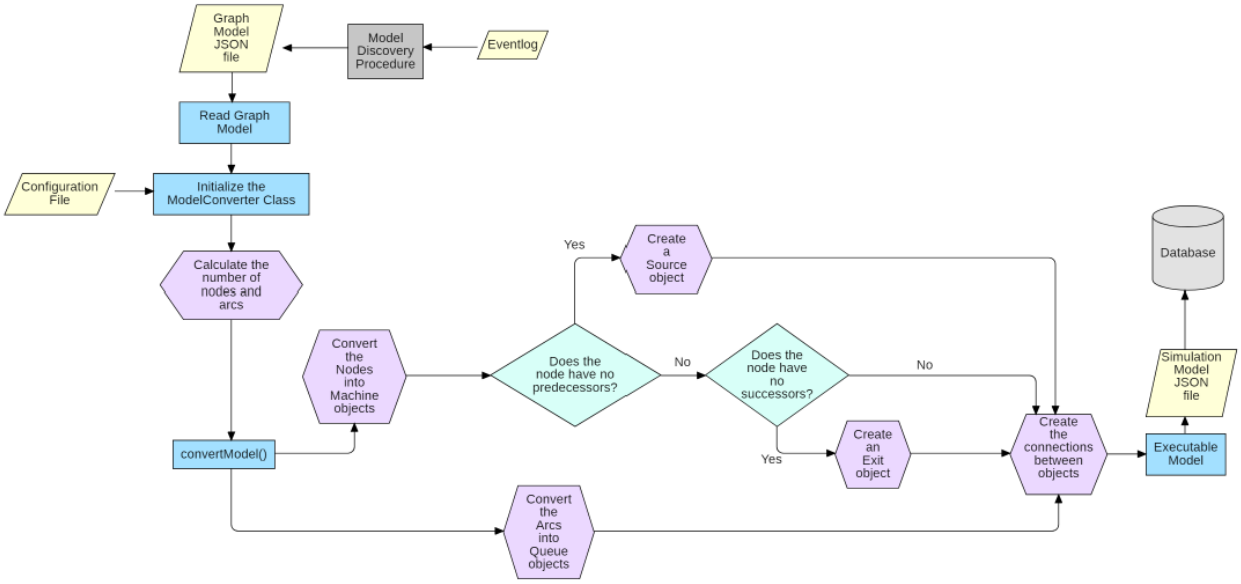


Figure 4.8: Model update flowchart

4.3.1.2. Implementation of Synchronisation

Through the synchroniser's methods and attributes, the instant conditions are aligned.

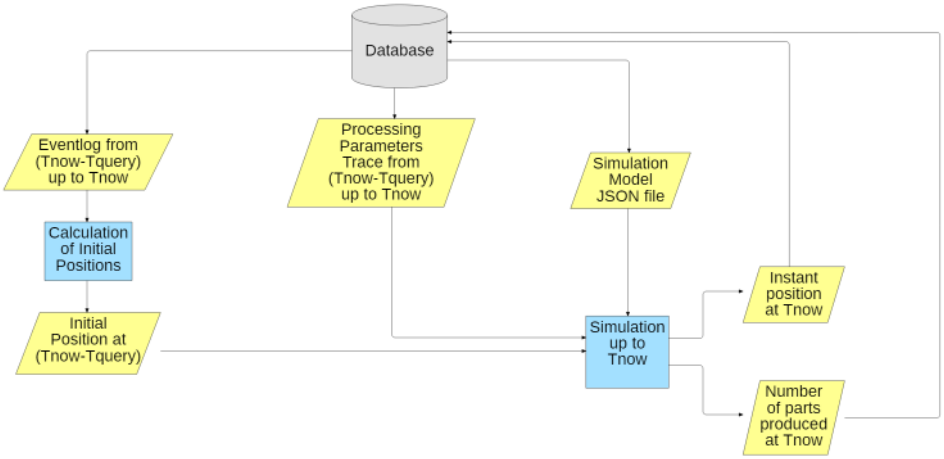


Figure 4.9: Synchroniser flowchart

Figure 4.9 pictures the sequence of performed procedures. Two inputs are acquired by the Database: the processing parameters acquired up until a certain time Tquery in the past composing the trace. Then, the event log is acquired using also Tquery so the initial positions of the pallet at the beginning of the acquisition is computed, then imposed in the Trace Driven simulation.
The simulation is run until Tnow so the instant conditions are aligned, the positions of the pallets and the parts produced are uploaded in the Database.

4.3.1.3. Implementation of Input Model Update

The process of input model update allows to align the stochastic behaviors. Figure 4.10 illustrates the procedure implemented.

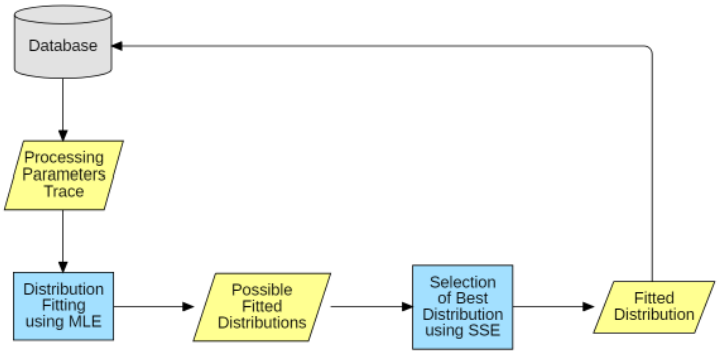


Figure 4.10: Input model update flowchart

Initially a determined trace of processing parameters is acquired by the Database. Using MLE (Maximum likelihood Estimation) method multiple types of distributions parameters are fitted e.g. Uniform, Triangular, Gamma, Normal etc.
The distribution parameters as well the type of distributions better fitting the trace are found using the SSE (Sum Squared Error) evaluation.

4.3.2. Database

InfluxDB is chosen for the database infostructure. It is an open-source time series database, based on InfluxQL query language. Its connection with the other components is handled by the Interface Database object, which contains functions for writing and querying the correct datasets. A class diagram containing the Interface Database and Database is shown in Figure 4.11. The methods “queryData()” and “writeData()” are used to respectively download and upload data to the database, while the attributes refer to values like the IP address and port number of the machine where the Database is located.

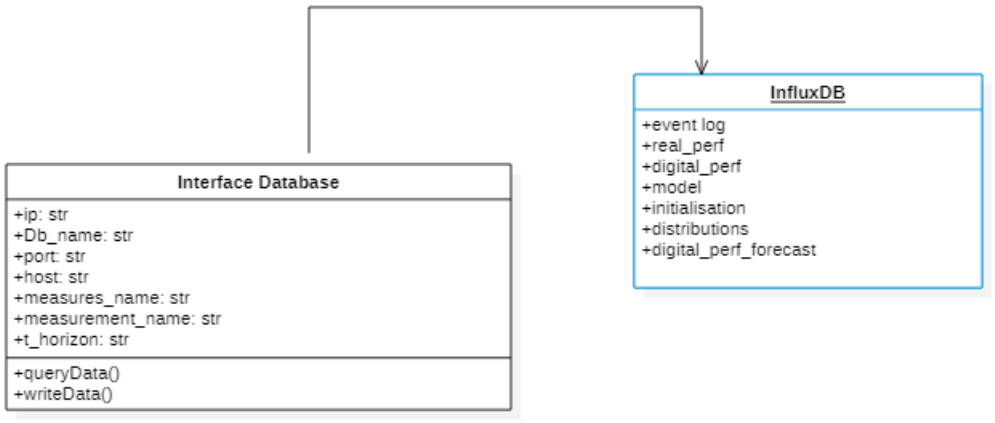


Figure 4.11: Class diagram of database and interface component

As shown in Figure 4.11 the database is organized in different tables, each containing data obtained from several sources. The objective is to separate the different information sets, simplifying the writing and querying processes. In Table 4.2 are shown two examples of how the tables are structured.

Table	Tags	Fields
eventlog	activity[str], part_id[int]	type[str]
real_perf	measures[str], activity[int], part_id[int]	value[int]

Table 4.2: Structure of the eventlog and real_perf tables

Every entry of the database has a set of tag and field values, which depend on the table, together with the associated UNIX timestamp. For example, as shown in Table 4.2, every entry in the table "eventlog" will have a value for each tag ("activity" and "part_id"), and one for each field ("type"), together with the time at which it is saved on the Database.

In the following chapter the platform is tested to validate the presented components, together with three experimental campaigns.

5. Numerical Experiments

This chapter presents in Section 5.1 the numerical experiments obtained with the purpose of assessing the functioning of the single components explained Chapter 4. Additionally, Section 5.2 illustrates a series of test cases performed to investigate the benefits of an aligned model in the context of forecasting.

5.1. Validation of the single components

This chapter deals with testing and validating the functionality of the single components. In section 5.1.1 model configurations used for this phase are presented. Different production scenarios are introduced with the objective of testing the developed methodology in multiple conditions. In section 5.1.2 the conversion procedure and the simulation model are analyzed, by comparing it to models developed in Arena. Then, the generated and converted model of the lab-scale production line is tested. In section 5.1.3 the synchronisation capabilities of the model are investigated. Finally in section 5.1.4 the forecast function is evaluated, together with how it is affected by the synchronisation of the parts positioning.

5.1.1. Intro to scenarios

Three configurations of the physical system are introduced in these sections, corresponding to different production scenarios of increasing variability. The lab-scale model presented in section 4.1 is used, and the variability is introduced by choosing the distribution parameters and the seed used by the random number generator, influencing the processing variables of the two stations.
In Table 5.1 the scenarios which will be dealt with during this testing phase are presented. Scenario 1 is characterized by deterministic processing times and is used to test the prediction capabilities in context without variability. For scenario 2 are used stochastic distributions, introducing variability in the behavior of the system. Scenario 3 substitutes the triangular distribution with a Weibull, which is characterized by a higher value of standard deviation, increasing furtherly the variability of the physical system.

	Station 1	Station 2	Demonstration length
Scenario 1	Det: 10	Det: 12	40 min
Scenario 2	Triang: [10,18,14]	Triang: [11,19,15]	40 min
Scenario 3	Weibull: [18,3]	Triang: [10,21,20]	40 min

Table 5.1: Scenarios list for validation and test bench

5.1.2. Conversion

In this section the capabilities and limitations of the conversion procedure and the resulting simulation model. Different manufacturing system configurations are tested, first by manually creating a model on Arena Rockwell simulation, then collecting the data from the simulations, use it to generate a graph model and finally converting it into ManPy. The two models are simulated and compared with the t-paired technique. Since the objective is to identify whether the logic of the ManPy model is the same as the Arena one, the same distributions and buffer capacity are used in both models. Additionally, the generated and converted digital model of the two-station lab-scale model, presented in section 4.1, is tested in trace-driven conditions. The results of the tests are detailly illustrated in Appendix A.3.1.

A six-machine flowline configuration is used to test the conversion procedure for a more demanding system, compared to the running example presented in section 3.3.1. Parts arrive at the first station, then are processed by other five subsequent machines before being disposed. Six fixed capacity buffers are positioned between the processing stations, which process one part at a time, with Blocking After Service (BAS) discipline. The Arena model is used to generate the events used by the generation algorithm. Figure 5.1 shows the blocks used in Arena to model one station of the system.

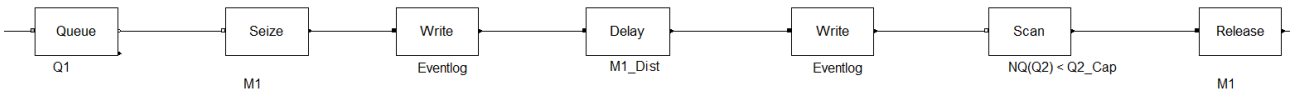


Figure 5.1: Arena six machine flow line model

Write blocks are placed before and after the machine, generating a text file containing the event log of the processes. In Table 5.2 is shown an extract of the file obtained from the Arena model. In the first column is recorded the timestamp of the activity,

the second refers to the station that performs the processing, in the third are recorded the activity types, "s" for start and "f" for finish, and finally in the last column are saved the processed part ID.

Timestamp [s]	Resource	Activity type	Part ID
190.8121	4	"s"	7
191.0250	5	"f"	5
191.0250	5	"s"	6
162.1587	6	"f"	4
162.1587	6	"s"	5
205.6218	3	"f"	8
207.9275	2	"f"	9
207.9275	3	"s"	9
211.0631	6	"f"	5
211.3121	4	"f"	7
211.3121	4	"s"	8
211.7133	1	"f"	10

Table 5.2: Example of the resulting event log file obtained from the Arena model, showing timestamp, station number, activity type, and part ID for each processing event.

One replication has been executed; the gathered event log file is then used to create the graph model through the generation algorithm. The result model, shown in Figure 5.2 (a), is composed of 6 nodes, representing the stations, with 5 arcs connecting them.

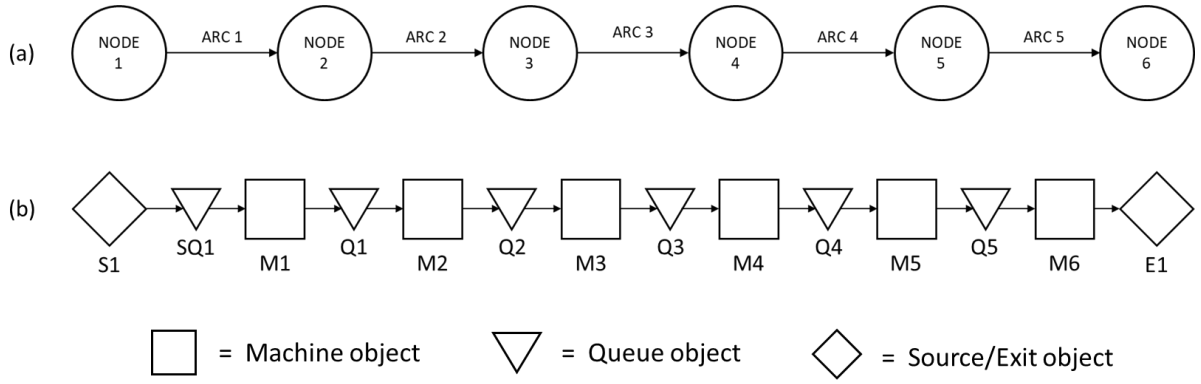


Figure 5.2: Six machine flowline generated graph model (a) and converted simulation model (b)

The resulting simulation model is pictured in Figure 5.2 (b), and is composed of six Machine objects, six Queue elements, one Source, and one Exit.

To assess the performance of the conversion algorithm, the ManPy model and the Arena one have been compared by calculating two performance indicators: the system and inter-departure times. Both models were executed for 5 replications with length of 24 hours, and the KPIs were collected and analyzed. The results are shown in Figure 5.3.

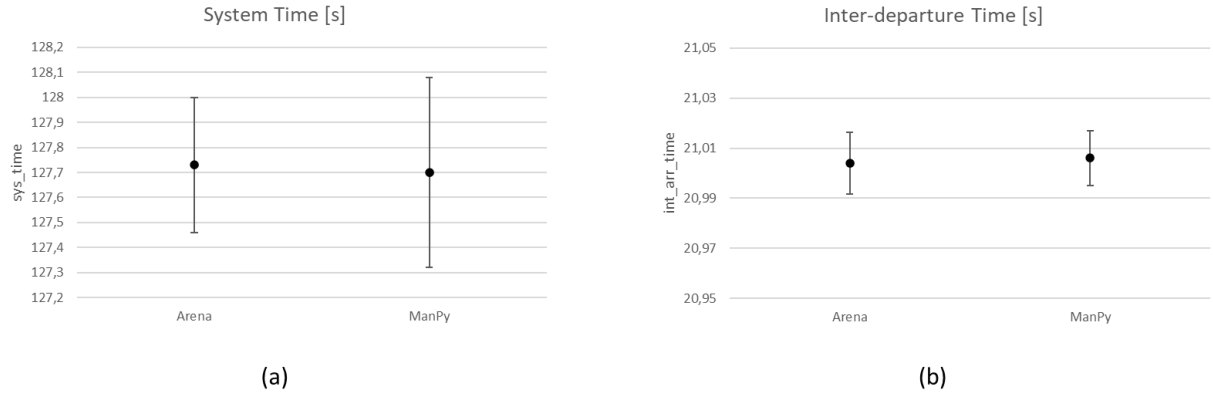


Figure 5.3: Comparison between the Arena and ManPy models for the six-machine flowline. KPIs used are system time (a) and inter-departure time (b)

The 95% confidence interval of the difference is equal [-0.4907 0.5491] for the system time and [-0.0206 0.0161] for the inter-departure time. Both these result show that there is not a significant difference between the logic of the two models.

The third tested configuration involves the analysis of a system with parallel servers. This has been done to check the capabilities of the parallel machine selection logic in the ManPy simulator. As previously, the assessment has been performed comparing two digital systems, with Arena as benchmark.

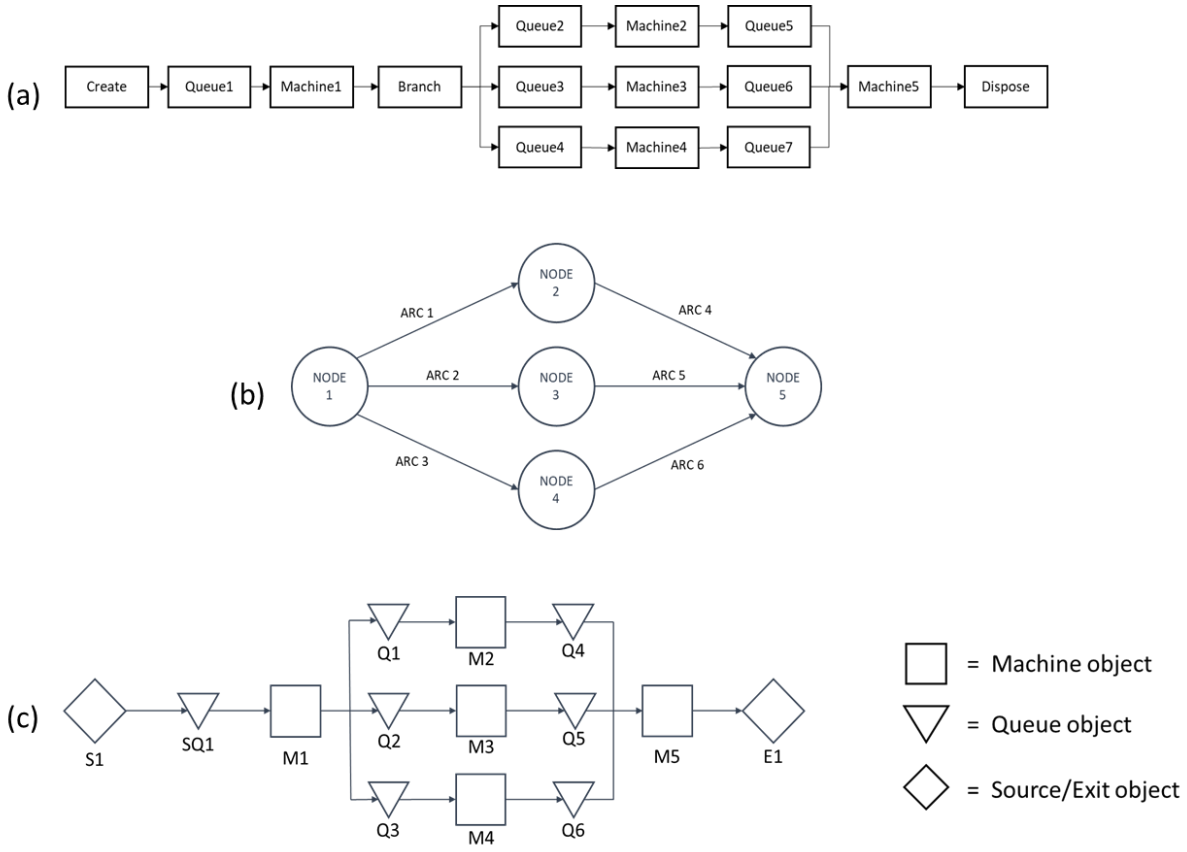


Figure 5.4: Arena parallel servers' model (a), generated graph model (b), and converted simulation model (c)

The production line is composed by three parallel servers followed and preceded by a single machine. A simplified representation of the Arena model is pictured in Figure 5.4 (a). The branch block determines the path which parts go through within the parallel servers. Parts flow is directed with equal split probability. The generated graph model (Figure 5.4 (b)) and the converted one (Figure 5.4 (c)) are shown. The first is composed of 5 nodes, equal to the number of machines, and 6 arcs, while the latter adds the Source and Exit objects.

The results are shown in Figure 5.5, showing the comparison between the two simulation models in terms of system time (a) and inter-departure time (b).

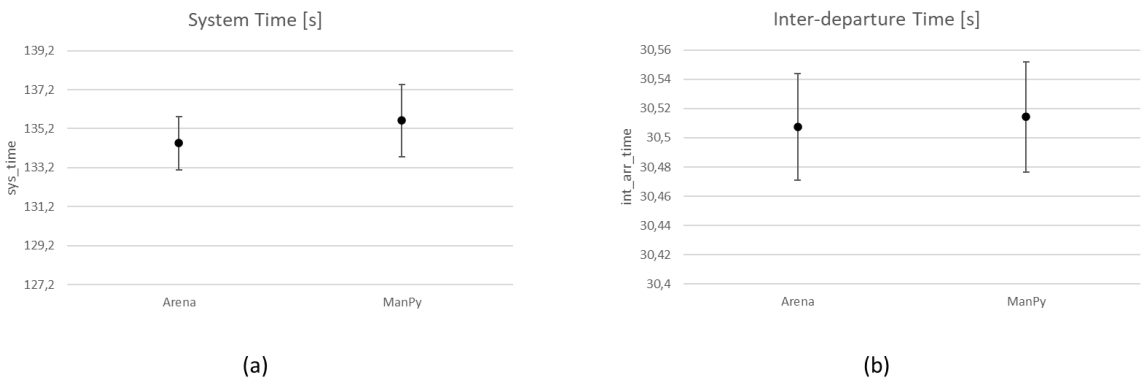


Figure 5.5: Comparison between the Arena and ManPy models for parallel servers' configuration. KPIs used are system time (a) and inter-departure time (b)

The 95% confidence interval of the difference is equal [-3.8296 1.5062] for the system time and [-0.0656 0.0517] for the inter-departure time. These result show that also the parallel logic of the ManPy model is comparable to the Arena one.

Finally, the conversion procedure is tested using data gathered from the two-station lab-scale model presented in section 4.1. A segment of the event log recorded in the database is given as input to the model discovery procedure. Due to the limitation of the data available and the capability of the generation method, the graph model cannot replicate some characteristics of the real system. This means that those parameters have to be inserted manually in the conversion function, allowing the final simulation model to perform at its highest in simulating the real system conditions.

The graph model generated is composed of two nodes and two arcs connecting them as a closed loop, as shown in Figure 5.6 (a). Unfortunately, due to the fact that the model generation method is optimized for open models, the capacity of the arc 2 cannot be correctly computed. Furthermore, the transportation conveyors are not discovered, and have to be manually added. To overcome this limitation a configuration file is given as input to the conversion procedure. Transportation time, conveyor capacity, buffer capacity and unload times have to be specified. The result, shown in Figure 5.6 (b) is a two-machine closed-loop line, with two buffers, each composed of one Queue object and three transportation objects, composed by modified Machine objects, simulating the behavior of the conveyors.

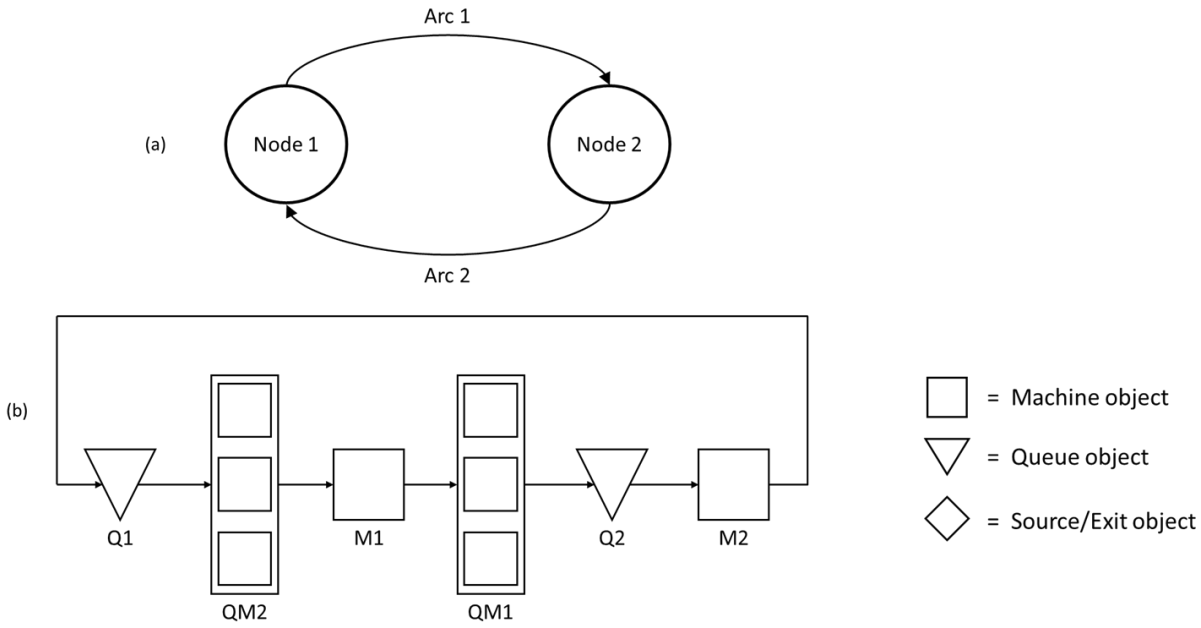


Figure 5.6: Simple converted model

The simulation model is tested in its capabilities of replicating the lab-scale model behavior in trace-driven conditions. Figure 5.7 shows an excerpt of a production demonstration, with the digital model replicating the physical system in trace-driven conditions. The simulation model system time trend is very similar to the one obtained by the real production line. Some discrepancies of maximum 2 seconds are present but can be explained either by imperfect behavior of the lab-scale model, such as pallets getting stuck, or by the different unloading logics during blocking conditions of the two systems.

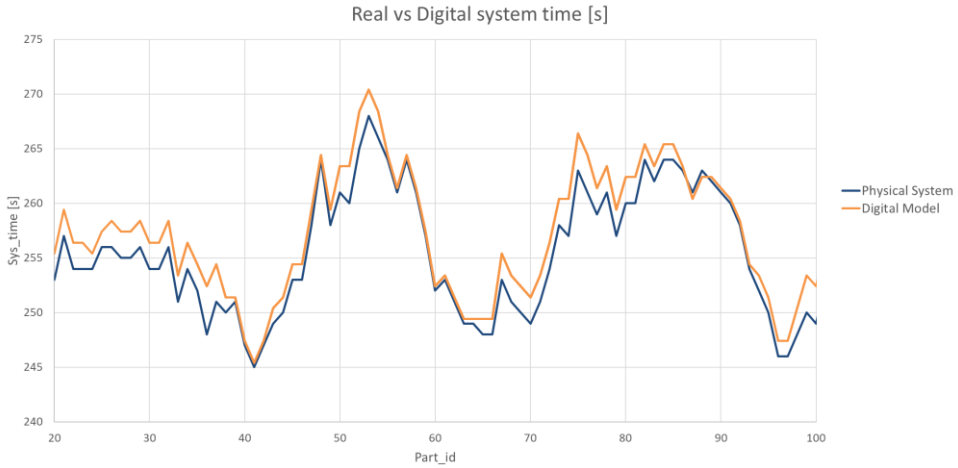


Figure 5.7: Comparison between physical system and digital model system time trend.

5.1.3. Synchronisation

The synchronisation procedure aims at maintaining the instant conditions present in DT aligned with those of the physical system during operations. As Section 3.2.3 explains, processing parameters are used as trace, while the initial positions of the pallet within the line are acquired at the beginning of the acquired trace. Every time is requested by the controller, a trace-driven simulation is run, and the instant conditions are recovered.
The following sections allows to validate and test the procedure of syncrhonisation and the control algorithm managing it. In addition, all tests are performed applying the platform on the lab-scale model introduced in Section 4.1.
First, a simple test using deterministic configuration setting, Scenario 1, presented in Table 5.1, is demonstrated. Subsequently, scenarios of higher variability are introduced and also results obtained by monitoring are illustrated.

At this stage the synchronisation component is used with a constant frequency and the indicator developed is not implemented.
With the deployment of the syncrhonisation, the DT is able to be aligned with the advance of the physical system. The trace-driven simulation uses the processing times computed by the events log acquired from the database. This implies that the precision of the results obtained from the simulation are directly dependent by that one of the sensors.
For this assessment the control is imposed to trigger the syncrhonisation every 10 seconds. With the aim of showing the two entities performance, the system time is acquired. Figure 5.8 shows the system time obtain from the digital model and the one from the real system.

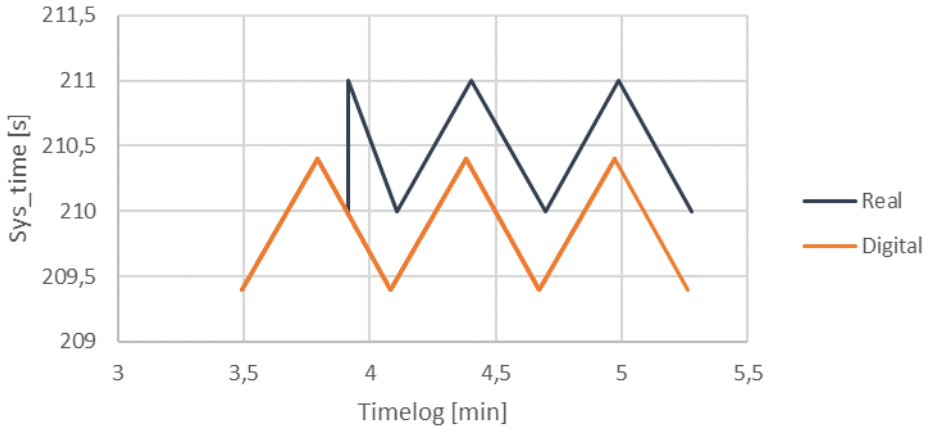


Figure 5.8: Scenario 1, synchronisation system time

Figure 5.8 confirms the capability of the procedure. Firstly, the real system time is appropriately reflected by the simulation results. Also, the final conditions depicted allows to demonstrate the efficacy in maintaining aligned the two entities.
As previously introduced the indicator is not used, for this test, by the control

algorithm to decide whether or not to trigger a syncrhonisation action. However, the indicator is anyway computed and allows to show the conditions of alignment when a request for synchronisation is made. Figure 5.9 shows the variation of the synchronisation check indicator with respect to time.

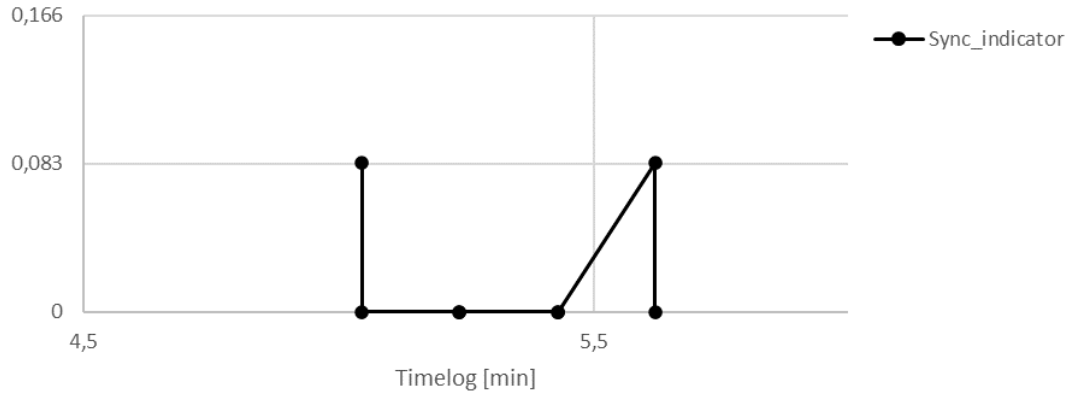


Figure 5.9: Scenario 1, synchronisation check indicator

For not aligned instant conditions the indicator is greater than 0. Ideally, the indicator should signal first a misalignment, then with the synchronisation the indicator should reach 0. However, since for this test no control is applied, as Figure 5.9 shows, the synchronisation is requested when the instant conditions are already aligned. i.e., indicator remains 0. In the next paragraph the syncrhonisation procedure is tested with Scenario 3 offering more variability, also the additional performance provided are shown to demonstrate the monitoring service.

The synchronisation procedure should be able to reflect the real conditions independently from the variability of the results obtained from the physical systems. For this reason, a test is performed using the synchronisation component in parallel with the lab-scale model set with Scenario 3 configuration settings (Table 5.1). Other Scenarios have been tested, performing similarly. The related results and for the entire evolution of synchronisation in time is illustrated in Appendix A.3.2.

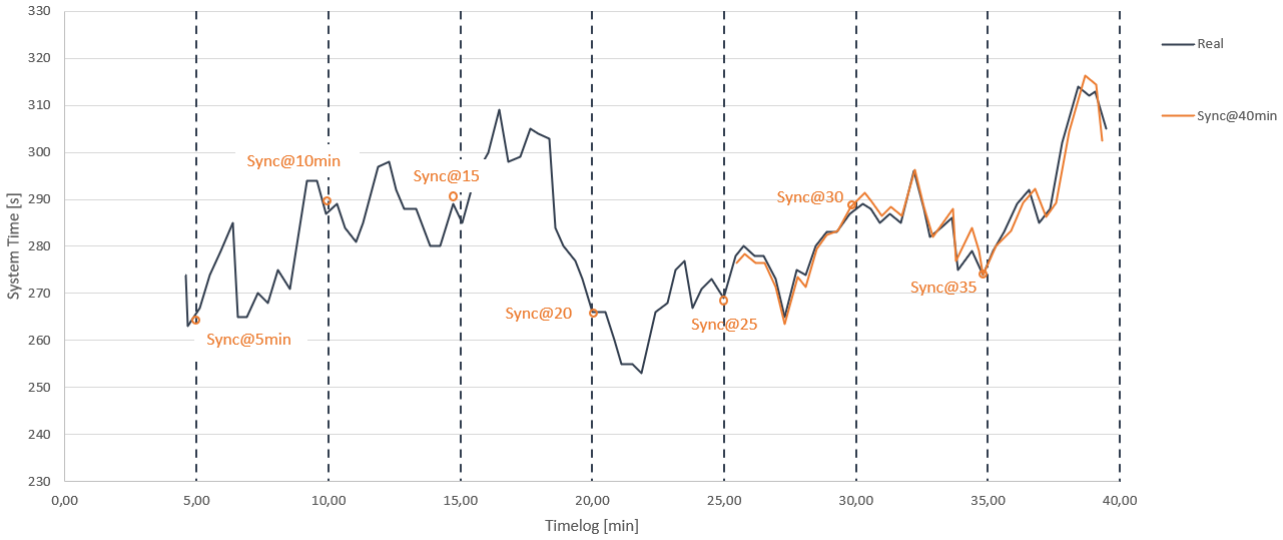


Figure 5.10: Scenario 3, synchronisation system time

Figure 5.10 illustrates the system time obtained from the real system and the digital model during synchronisation. Synchronisation are performed based on the request of the controller. As anticipated in section 3.3.2, the information gained by the syncrhonisation check about the alignment of the instant conditions are used by the controller to request a synchronisation. During the validation tests, this indicator is not deployed, in fact using a constant frequency imply the synchronisation to be used even if not necessary. The opposite is done for this case, the utilisation of the synchronisation check to assess the alignment allows the synchronisation to be requested only when needed and this effectively reduce the computational effort required.

In Figure 5.10 "sync@40min", the synchronisation performed at 40th min, illustrates, the trace used is of 20 min for this test. Specifically, the first results are delivered from the 25th min because the first pallet takes almost 5 minutes to complete a cycle. This setting is applied for two reasons: firstly, the simulation is able to perform quite quickly and realign with the final instant conditions. Secondly, by simulating 20 min of production it is possible to monitor the performance of the real system in real-time. In particular it is far easier to gain additional information from the simulation model rather than from the physical system. The comparison illustrated in Figure 5.10 demonstrates that the performance of system time obtained from the two entities are similar, thus also the digital performance used to monitor can be considered reliable. Although the performance indicators which can be acquired from the digital model are multiple, in this case the utilisation of the two stations is acquired. The final objective of this representation is to demonstrate the contribution that monitoring the performances of the real system with an aligned simulation model can provide.

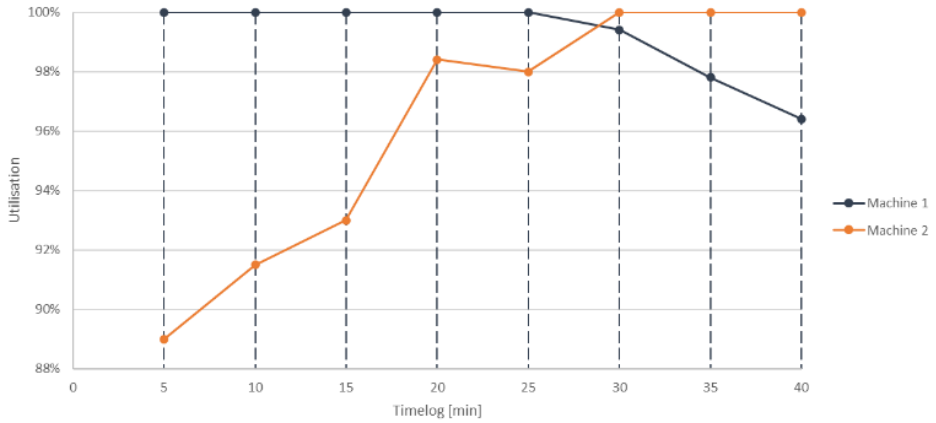


Figure 5.11: Utilisation obtained for station 1 and station 2 every synchronisation

Figure 5.11 depicts the utilisation of the two stations computed for the synchronisations spaced 5 min from each other. As the graph shows, machine 2 at presents a higher values at the beginning, since pallets are all loaded in buffer 1 at the start of the demonstration. As the production continues, the utilisation of station 1 decreases, while it increases for station 2. This is caused by the fact that the latter is the slower machine, causing it is to be the bottleneck of the line.
This section validates the alignment of instant conditions performed with synchronisation. Additionally, it shows how monitoring is performed and what additional outcomes are provided.

5.1.4. Forecasting

In this section the forecast capabilities of the aligned digital model are investigated. Test are performed on the two-station system with three different scenarios, presented in Table 5.1.

Firstly, scenario 1 has been used to validate how the forecast analyses are computed. A comparison is made with forecasts made at 5 mins of distance between each other and predicting the number of parts produced at two fixed moments in time, in particular, the 20th min and 40th min, corresponding respectively to the middle and end of the demonstration. Every simulation uses 5 replicates and runs until the end of production. It is important to underline that every time a forecast analyses is run, the model is previously synchronised, computing the initial positioning and the amount of parts produced until that moment. Figure 5.12 and Figure 5.13 illustrate the cumulative production and the results of the prediction for the 20th and 40th minutes.

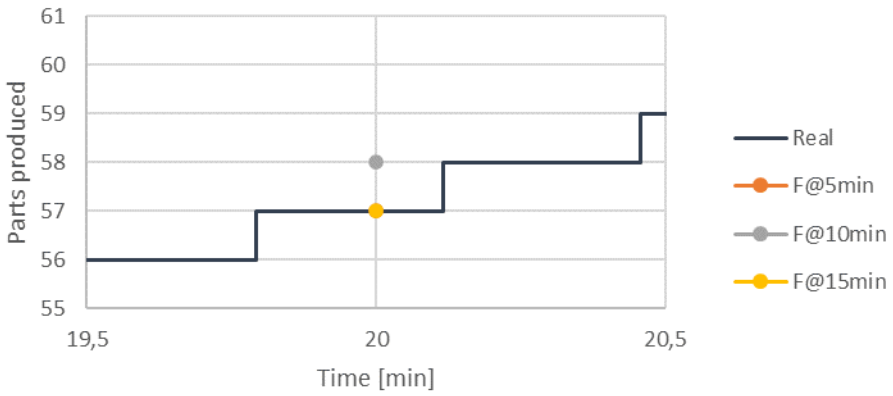


Figure 5.12: Cumulative production and prediction results at the 20th minute for scenario 1.

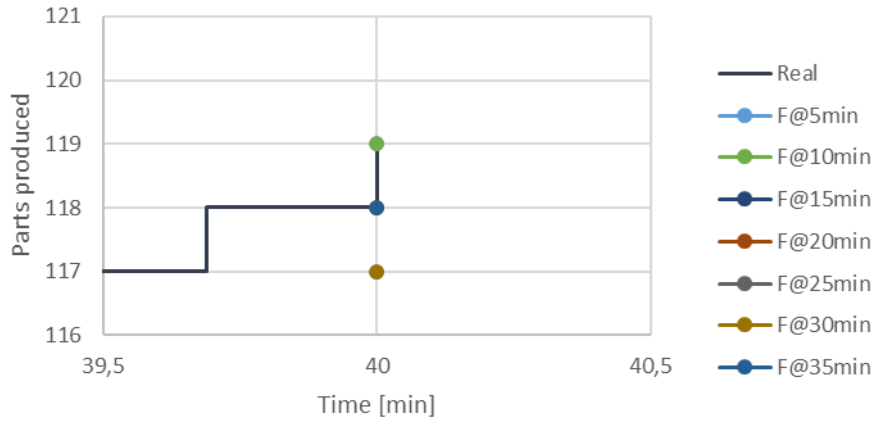


Figure 5.13: Cumulative production and prediction results at the 40th minute for scenario 1.

Due to the deterministic nature of this test, the results do not show any kind of trend in terms of accuracy of prediction.

The same test is performed for scenario 2, which presents a higher overall variability.

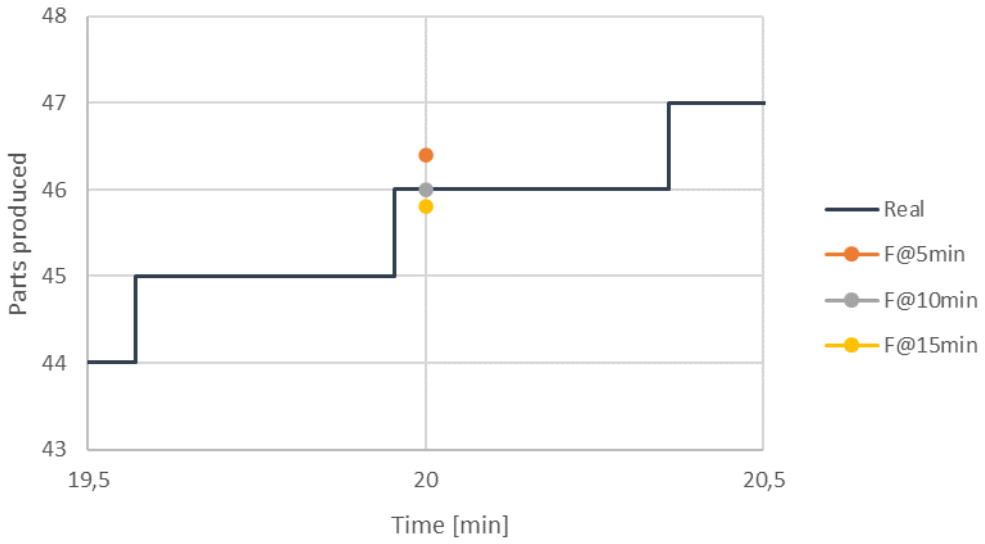


Figure 5.14 : Cumulative production and prediction results at the 20th minute for scenario 2.

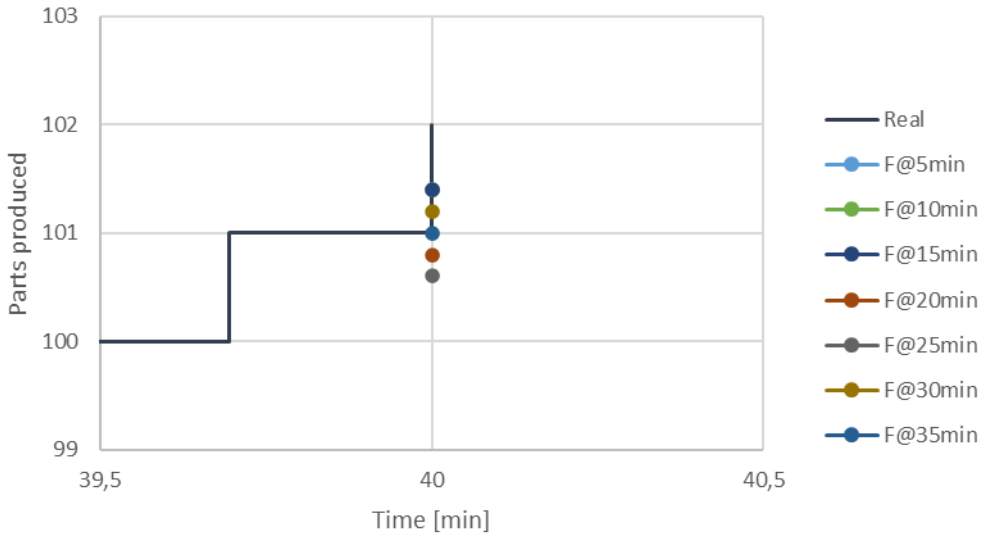


Figure 5.15: Cumulative production and prediction results at the 40th minute for scenario 2.

Differently from the previous test, a higher variability introduces some discrepancies in the results. However, it is majorly fundamental to highlight that for increasingly closer prediction to the moment, thanks to the synchronisation becomes more precise.

Finally, scenario 3 is tested, and the results are shown in Figure 5.16 and Figure 5.17.

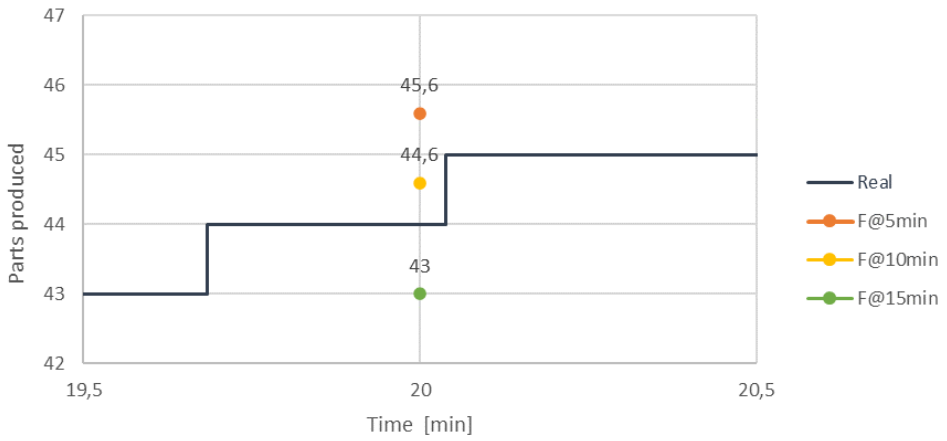


Figure 5.16: Cumulative production and prediction results at the 40th minute for scenario 3.

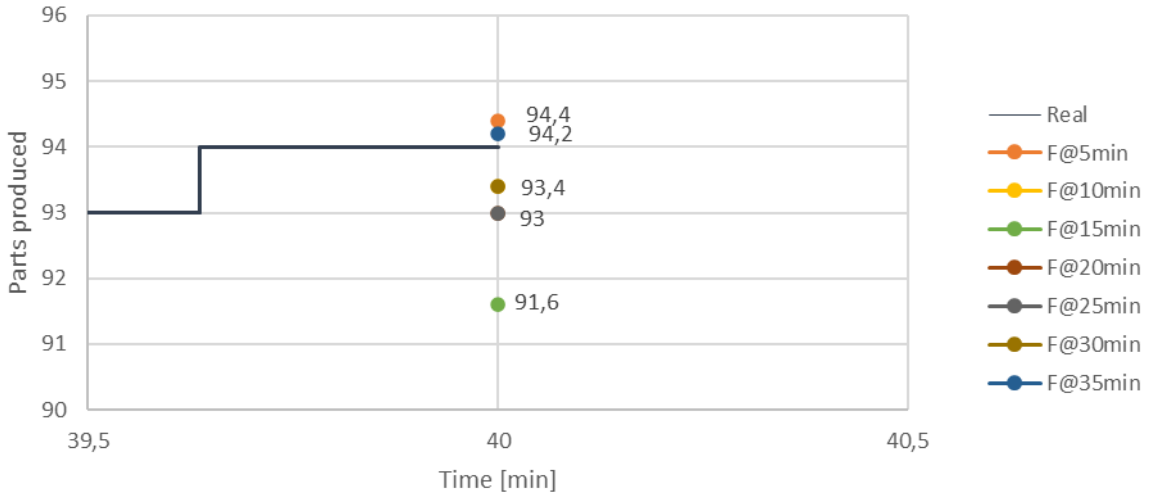


Figure 5.17: Cumulative production and prediction results at the 40th minute for scenario 3.

This scenario is more variable than the previous two, and an increasing trend in terms of forecast accuracy can be seen, particularly in the results for the 40th minute prediction. The predictions get closer to the actual production value as the forecast time gets closer to the end of the demonstration, the only outlier value is the forecast made at the fifth minute.

Further tests were performed without synchronizing the initial position before each forecast. The pallets were all located before station 1 in each forecast run. The aim of these experiments is to show the effect of not initializing the correct part position on the forecast. The results are pictured in Figure 5.18 and Figure 5.19.

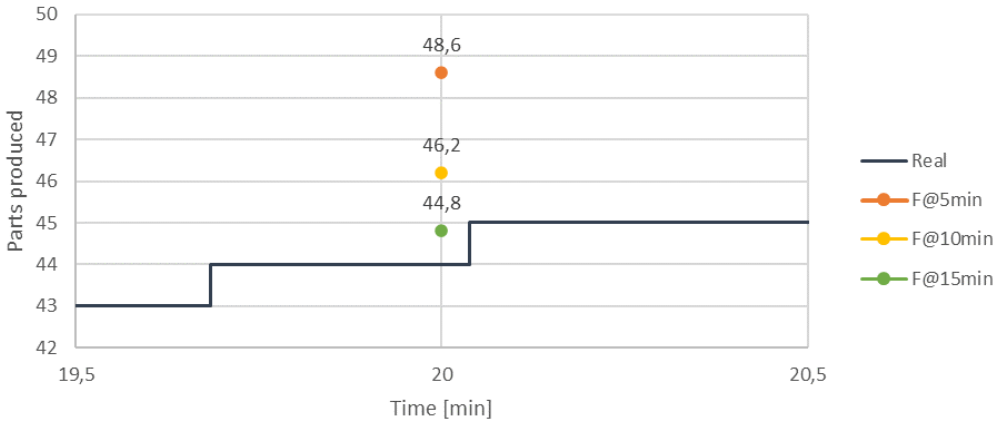


Figure 5.18: Cumulative production and prediction results at the 20th minute for scenario 3 with no part positioning synchronisation

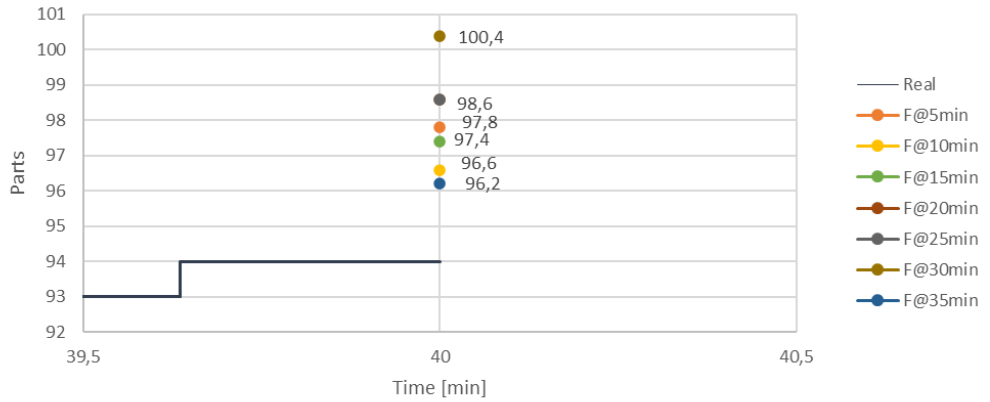


Figure 5.19: Cumulative production and prediction results at the 40th minute for scenario 3 with no part positioning synchronisation

These results show that by not initializing the correct initial position of the part at the start of each forecast, the predicted number of parts is overstimated.

5.2. Test cases

In this section experiments will be presented to assess the effect of parameters on the performances of the developed methodology. Three different campaigns were conceived and they are summarized in Table 5.3. The objective is to analyze the behavior of the integrated functions of synchronisation and forecast. The developed architecture presented in chapter 4 and validated in section 5.1 is utilized.

	Experimental campaign 1	Experimental campaign 2	Experimental campaign 3
Objective	Evaluate the relationship between the instant of synchronisation and the prediction.	Evaluate the effect of a partially misaligned model on the synchronisation and forecast.	Understand the impact of the forecast frequency parameter on prediction error.
Investigated factor	Time of forecast	Statistical parameters	Forecast frequency
Reference KPI	Prediction error	Prediction error Normalized error area	Normalized error area
Setting scenarios	Scenario 3	Scenario 3	Scenario 3 Scenario 4

Table 5.3: Description and objective of experimental phase

5.2.1. Reference KPIs

Given that the studied application is the forecast, the key performance indicators used as reference to analyze the result are based on the number of parts produced in a given time frame. More precisely, a prediction error E_p is the subject of evaluation, and it is computed for each simulation as the absolute difference between the forecasted value $N_{predicted}$ and effective number of parts produced by the real system $N_{effective}$.

$$E_p = |N_{predicted} - N_{effective}| \quad (5.1)$$

Where $N_{predicted}$ is defined as:

$$N_{predicted} = \frac{\sum_i^{nrep} N_{predicted,i}}{nrep} \qquad i = 1,2,\dots,nrep \quad (5.2)$$

Where $N_{predicted,i}$ is the prediction for the i-th replication of the simulation.

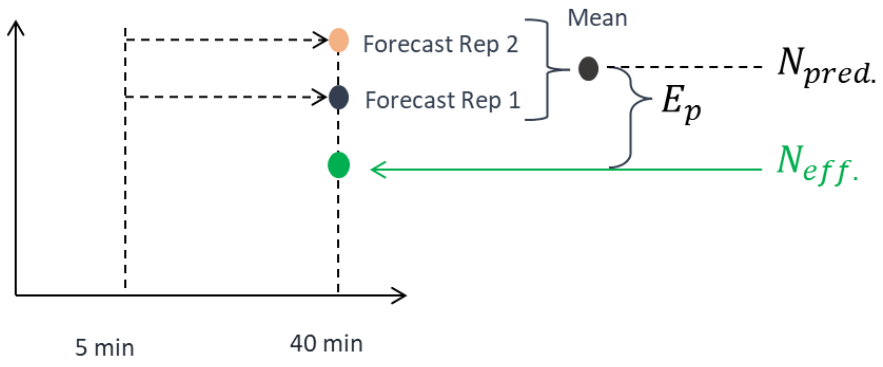


Figure 5.20: Graphical computational illustration of the prediction error

As a result each prediction analysis conveys into a single error value E_p, as shown in Figure 5.20. In following campaigns, the complexity increases and multiple results are obtained from a single experiment. The latter are made comparable by computing them into a single value, the normalized area error E_{area}. The computation of this proposed solution follows Equation (2.2), where $E_{p,i}$ is the prediction error for the i-th forecast, and $t_{forecast,i}$ is the moment in time at which the simulation is performed. Figure 5.21 shows two examples of normalized error area calculations.

$$E_{area} = \sum_i^n \frac{E_{p,i}}{(t_{forecast,i+1} - t_{forecast,i})} \qquad i = 1, 2, \dots, n \qquad (5.3)$$

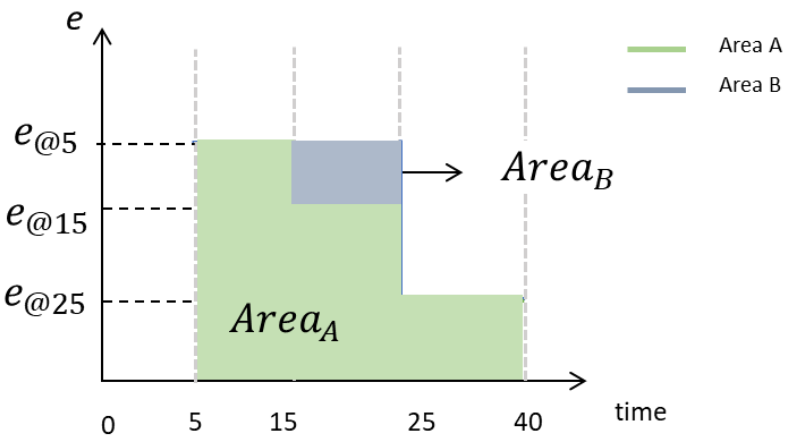


Figure 5.21: Normalized area error description

5.2.2. Experimental campaign 1: time of forecast

The first experimental campaign seeks to solidify the beneficial contributions of an aligned model when it comes to conduct forecast analyses. Through the tests presented, the aim is to demonstrate the effectiveness of the implemented methodology in being accurate to predict future performances.

5.2.2.1. Experimental setup

Scenario 3, introduced in section 4.1, is used as the configuration for the physical system. The mean and variance of the distribution parameters are shown in Table 5.4.

Station	Distribution type	Parameters	Mean [s]	Variance [s²]
Station 1	Weibull	[18, 3]	16.07	34.128
Station 2	Triangular	[10, 21, 20]	17.00	6.167

Table 5.4: Distribution settings used in Scenario 3

A real situation is replicated by performing 5 experiments of 40 min each during distinct moments of production. After 5 minutes of warm-up, the corresponding digital platform is activated. The length of the demonstration is limited by the lab-scale model, which starts to malfunction (i.e. parts get stuck in the system) when it remains active for too much time.

On the other hand, the forecast analyses are run with a 5 minute interval. To mitigate the influence of incorrect fitting, the imposed distributions for the physical object are also implemented in the digital one. Therefore, only the variability of the production processes can cause forecast errors.
For each prediction E_p is computed for the 20th and 40th minute, then gathered with the others made at same instant but from different experiments.

5.2.2.2. Results

The value $E_p(t,T)$ indicates the error for the predicted production at time T when the forecast is started at time t.

The results obtained for the experimental campaign are shown in Figure 5.22 and Figure 5.23.

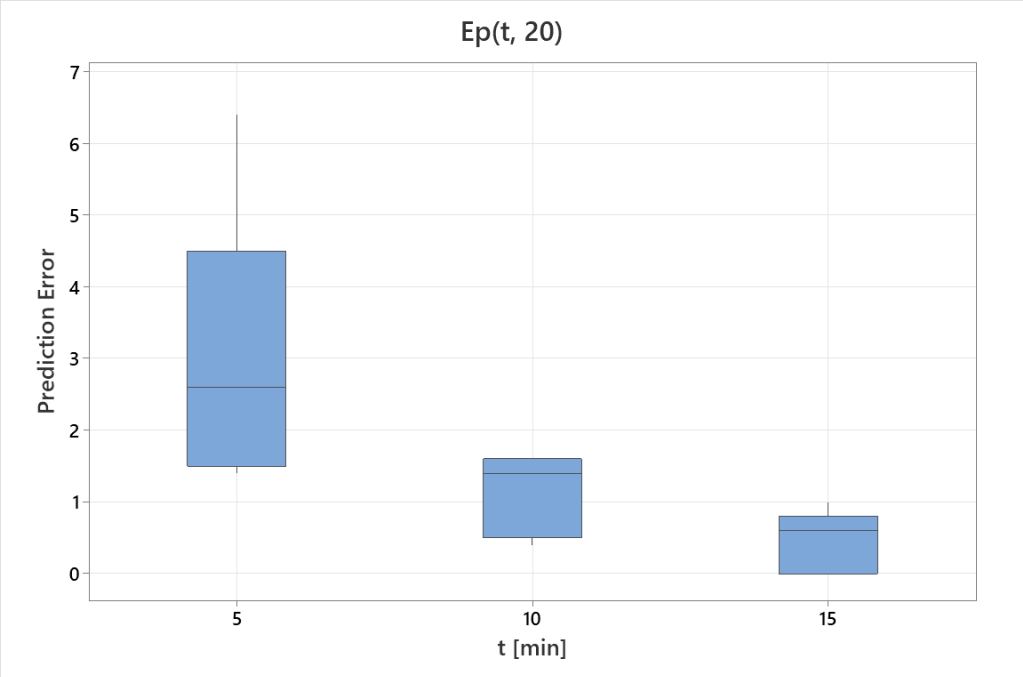


Figure 5.22: Boxplot of the forecast error of the parts produced at the 20th minute

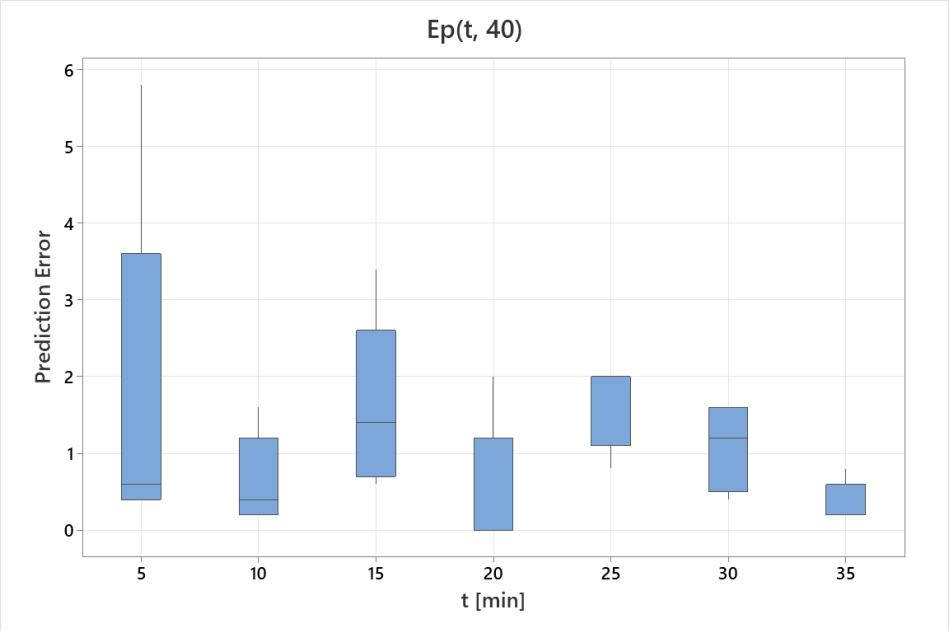


Figure 5.23: Boxplot of the forecast error of the parts produced at the 40th minute

One-way ANOVA is used, with the prediction error as response variable and significance level α=0.05, with the objective of understanding if the results are significantly different. For the analysis on $E_p(t, 20)$ the p-value is equal to 0.022, therefore the null hypothesis stating that all means are equal is rejected. Furthermore the results are grouped using Fisher LSD Method, as shown in Table 5.5.

t	Average $E_p(t, 20)$	Grouping
5	2.920	A
10	1.120	B
15	0.440	B

Table 5.5: Grouping using Fisher pairwise comparison.

For the analysis on $E_p(t, 40)$ the p-value is equal to 0.236, therefore the null hypothesis of all means being equal cannot be rejected.

These experiments aim to illustrate that as the prediction analysis gets closer to the target the error reduces. The results are only significant for the $E_p(t, 20)$ but not for $E_p(t, 40)$. These findings can be caused by the overall low variability characterizing the experiments, caused by factors such as the short demonstration time and the reduced layout complexity.

5.2.3. Experimental campaign 2: input mis-alignment

In this experimental campaign the aim is to analyze the behavior of the synchronisation and forecast functions when the available digital model is partially misaligned. The expected result is to understand the mitigation given by the

contributing action of synchronisation in the context of forecasting, given a prediction error caused by the imposed misalignment.

5.2.3.1. Experimental setup

To simulate the use of a partially misaligned model, non-valid distributions for the processing times are given as input to the digital object. These do not replicate the stochastic behavior of the physical system in question; therefore a prediction error of the future performances is expected. Predictions are performed every 15 minutes, therefore three simulation experiments are executed, precisely at the 5th, 20th, and 35th minute. The synchronisation function, introduced in section 3.3.2, is executed before each forecast, initialising the parts positions, and computing the number of parts produced until that moment. The time between forecasts is arbitrarily chosen to obtain a balance between the computational effort of the digital system, which runs both the synchronizing and predicting functions, and the number of simulations. To study the effect of this parameter is not the scope of this experiment, but it is investigated in the following experimental campaign.

5.2.3.2. Design of experiments

The physical system's settings are the same as the ones utilized in the first experimental campaign. The parameters of the configuration are shown in Table 5.4. The stochastic behavior of the processing activity is defined by three parameters: distribution mean, variance, and type. Those values will be modified in the experiments to simulate a discrepancy between the real and digital systems. A factorial design is used to set up the multiple experiments in this campaign. The three parameters mentioned in the previous paragraph are considered as the factors, with two levels indicating the correct and incorrect value. Therefore, a 2^3 full factorial design is obtained, composed of 8 different experimental configurations. This experimental campaign is designed following a 2^3 full factorial design including: the three previously explained factors and as levels the correct and incorrect values. Table 5.6 shows the eight resulting configurations.

Configuration	M	V	T
1	L	L	L
2	H	L	L
3	L	H	L
4	H	H	L
5	L	L	H
6	H	L	H
7	L	H	H
8	H	H	H

M = mean value of distribution
V = variance value of distribution
T = distribution type

L = correct value
H = incorrect value

Table 5.6: 2^3 full factorial design table

The modifications to the factors are chosen with the objective of overturning the dynamical behaviors in the digital model. In fact, the mean is increased so that the

bottleneck of the production line is moved from Station 2 to Station 1; the new variance is doubled; and a uniform distribution has a different shape from the triangular used. For each configuration five experiments are performed, changing the starting conditions and the seed parameter. The effect is to simulate different production conditions, obtained from analyzing the system in distinct moments. The resulting number of experimental runs is therefore 40. Table 5.7 shows the distributions used in each experiment configuration.

Configuration	M	V	T	Distribution
1	L	L	L	Weibull(18, 3)
2	H	L	L	Weibull(18, 3) + 4
3	L	H	L	Weibull(18.1350, 2)
4	H	H	L	Weibull(18.1350, 2) + 4
5	L	L	H	Uniform(5.9515, 26.1885)
6	H	L	H	Uniform(5.9515, 26.1885) + 4
7	L	H	H	Weibull(18, 3)
8	H	H	H	Weibull(18, 3) + 4

Table 5.7: Distribution used in the configurations of the factorial design

5.2.3.3. Results

The average results for each configuration are shown in Table 5.8, where the average error for the predictions performed at the different times are presented, together with the normalized area error.

Configuration	$E_p(5,40)$	$E_p(20,40)$	$E_p(35,40)$	E_{area}
1	0,88	0,72	0,88	0,81
2	11,24	5,84	1	7,46
3	1,16	1	0,84	1,05
4	11,52	5,56	1,16	7,49
5	1,44	0,72	0,76	1,03
6	11,32	5,92	1,04	7,54
7	1,84	1	0,84	1,34
8	11,6	5,36	1,24	7,45

Table 5.8: Numerical results of the experimental campaign

One-way ANOVA is used, with the normalized area error as response variable and significance level $\alpha = 0{,}05$. The results highlight the most influencing factors and possible combinate effects.

The first part of the analysis focuses on discovering which is the main influencing factor between the mean, the variance, and the distribution type. In Figure 5.24 are shown the main effects boxplots, indicating the impact of the factors.

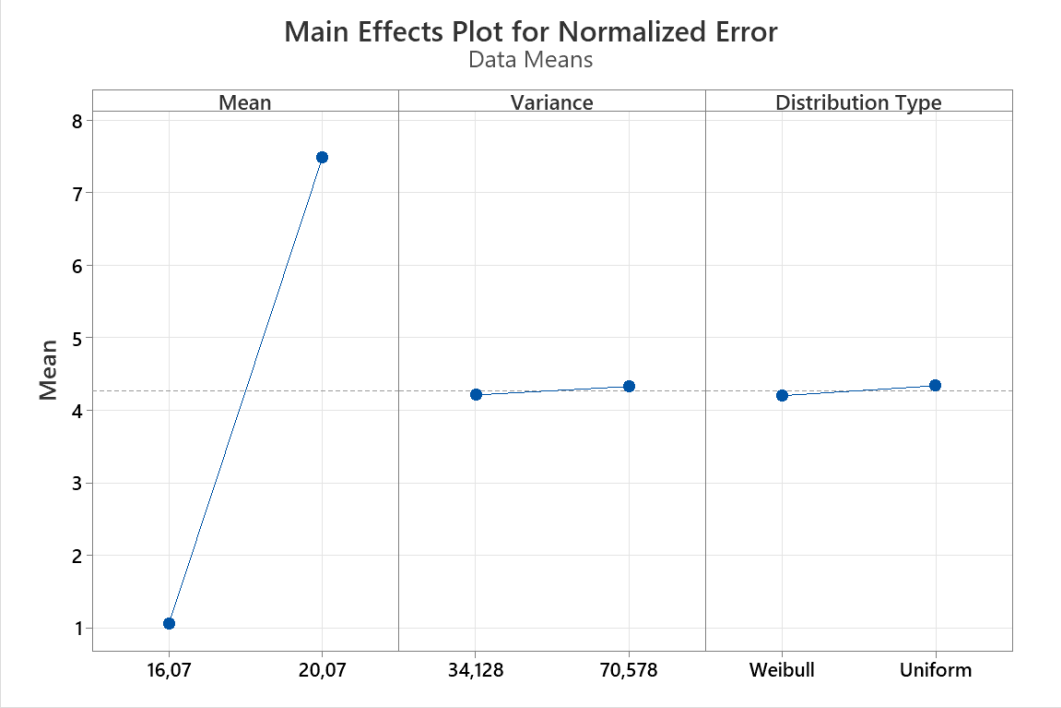


Figure 5.24: Boxplots of main effects on the normalized area error

From Figure 5.24 is noticeable the clear impact of the mean of the distribution on the prediction error. The other two factors, while still showing an increase in the normalized area error, do not show a significant influence on the results.

The combinate effects plots are shown in Figure 5.25.

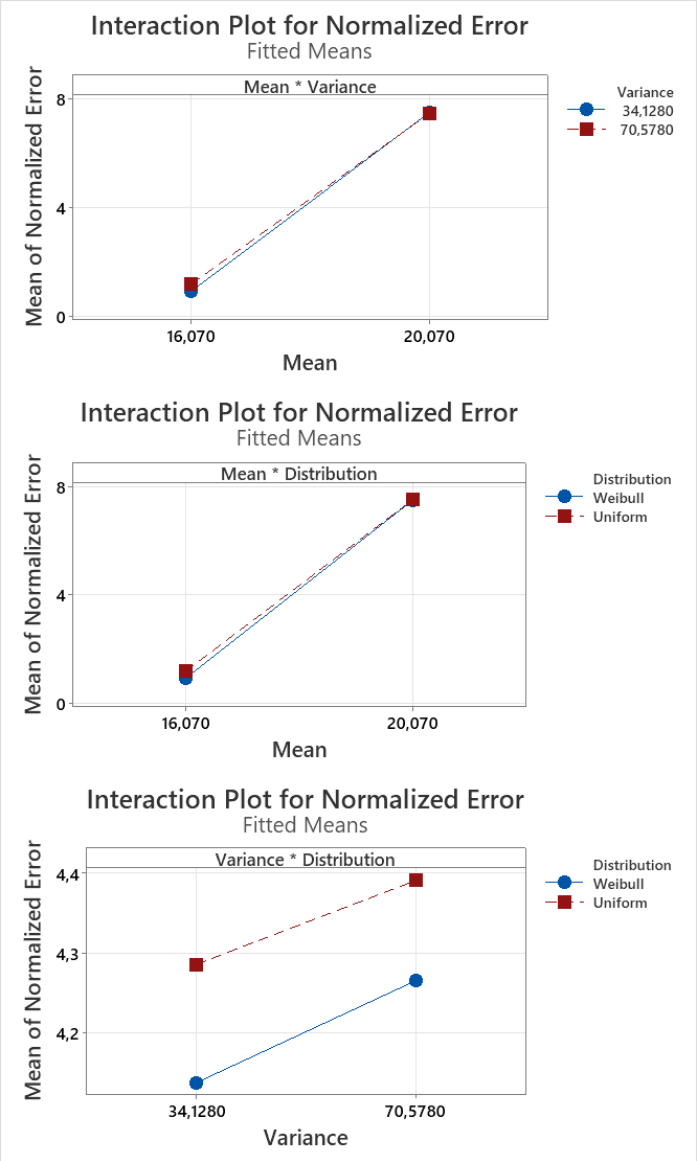


Figure 5.25: Combinate effects plot of normalized area error

This analysis is aimed at understanding if there is the presence of combinate effects, where the modification of two factors together impacts the result in a different way compared to the single effect. The results do not show the presence of significant combinate effects. The p-values of the analyses are shown in Table 5.9.

Configuration	M	V	T	P-Value
1	L	L	L	0.000
2	H	L	L	0.000
3	L	H	L	0.587
4	H	H	L	0.525
5	L	L	H	0.483
6	H	L	H	0.578
7	L	H	H	0.958
8	H	H	H	0.832

Table 5.9: P-values of experiments

5.2.4. Experimental campaign 3: forecast frequency

The object of experimentation is to investigate how the frequency at which the predictions are performed during production affects the forecast error. Therefore, these experiments are aimed at understanding if there is an optimal value for the time between forecasts that provides lower prediction error while still being computationally efficient.

5.2.4.1. Experimental setup

The physical system configured as introduced in section 4.1, with the same initial parameters present in Table 5.4, for the same reasons as the first experimental campaign. The demonstrations have length equal to 40 minutes, with 5 minutes of warm up, and the predictions initiated are at different frequencies. Two scenarios are tested, the Scenario 3 and the Scenario 4. The latter presents unpredictable disruptive events in the form of failures initiated in one of the stations at random times. The stations' configuration is the same for both scenarios, with the parameters presented in Table 5.3.

Five values of frequency are investigated, each corresponding to a time between forecasts, as shown in Table 5.10.

Forecast Frequency [min^{-1}]	Time Between Forecasts [min]	Number of Forecast per Demonstration
0.200	5	7
0.100	10	4
0.667	15	3
0.050	20	2
0.025	40	1

Table 5.10: Selected values of forecast frequency and corresponding time between forecasts and number of predictions per 40 minute demonstration

These values are selected with the objective of having a different number of predictions during the 40 minute demonstration.

As already mentioned, the second scenario tested in this experimental campaign is characterized by random disruptive events unknown to the digital model. These are introduced into the system as machine failures, manually initiated at predetermined times. The length of each event is equal to two minutes, after which the stations return to normal conditions, and are meant to replicate minor stoppages on a production line. These are introduced only to one of the machines, to better control the experiment. Station 1 is chosen, in this way during the failures the bottleneck shifts.
For each demonstration the selected number of failures is two, happening at different moments in the five replications. The minute at which the event is initiated is selected basing on a predetermined structure: the 35 minutes of activation of the synchronizing and forecasting functions is divided into four time intervals, as shown in Figure 5.26. The objective is to obtain five demonstration with failures happening at different times, mitigating the effect of the time of failure on the experiments.

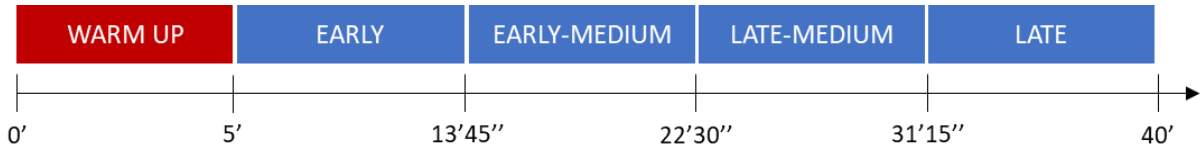


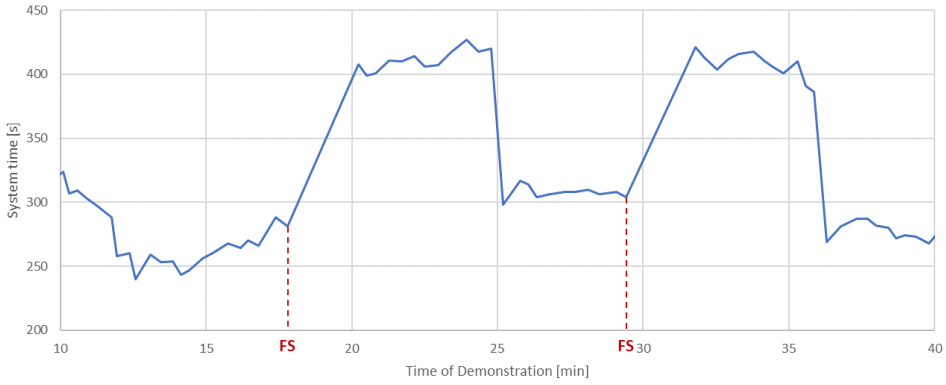
Figure 5.26: Division of the demonstration into periods

Five different combinations of time of failure are obtained, one for each replication of the experiment, as shown in Table 5.11. To obtain them, it was chosen to not initiate the failures in the same time interval for each experiment. The result are 6 possible combinations, out of which one is randomly discarded.

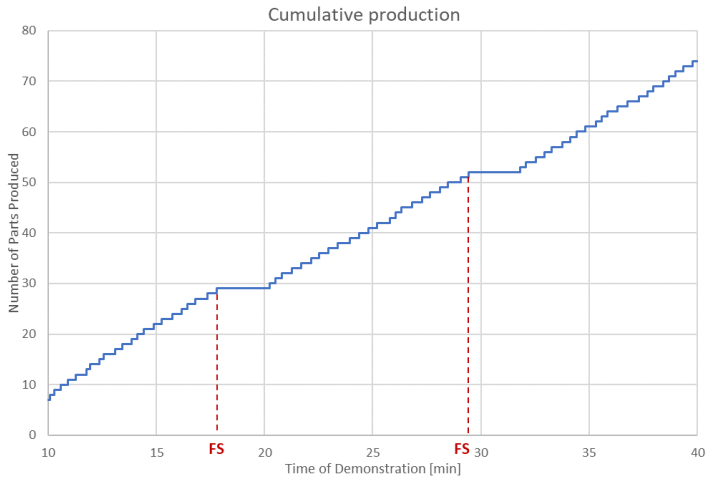
Experiment	Interval Combination	Time of 1st event [min]	Time of 2nd event [min]
1	EM-LM	18	30
2	E-L	11.5	32
3	EM-L	14	37
4	E-LM	10	27
5	LM-L	24	36

Table 5.11: Event timestamps for the 5 experiments (E = early, EM = early-medium, LM = late-medium, L = late)

The failures greatly influence the performance of the production line, as shown in Figure 5.27, where the system time trend and cumulative production of one experiment are plotted.



(a)



(b)

Figure 5.27: Effect of disruptive events on system time trend (a) and cumulative production curve (b).

5.2.4.2. Results

The results of the experiments conducted for scenario 3 are presented in Table 5.12 and Figure 5.28. Similarly to the previous experiment campaign the analyzed value is the normalized area error.

Forecast Frequency [min^{-1}]	Time Between Forecasts [min]	Normalized Area Error
0.200	5	1,07
0.100	10	1,47
0.667	15	0,99
0.050	20	1,69
0.025	40	1,72

Table 5.12: Numerical results of the experiments on scenario without failures (scenario 3)

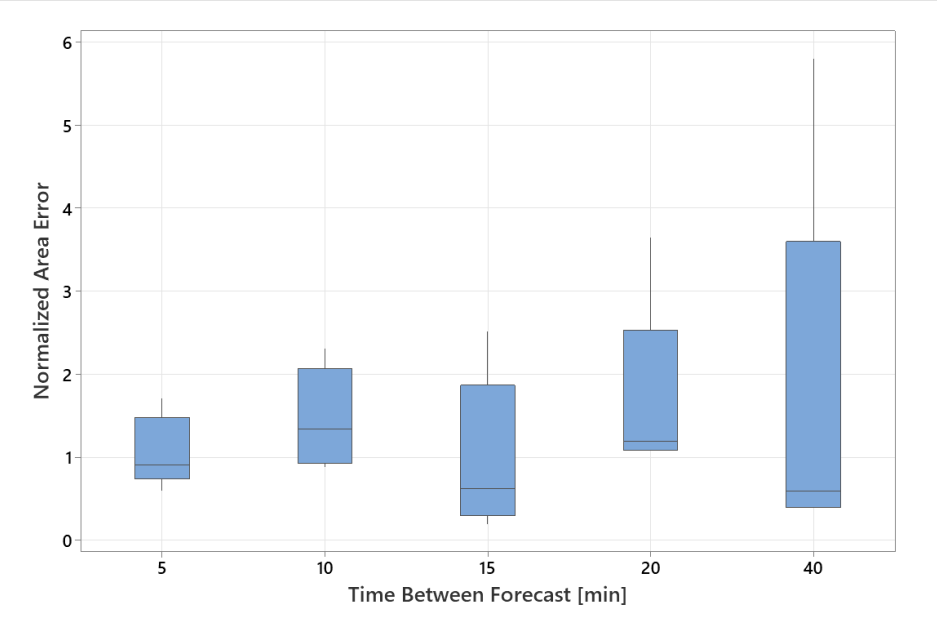


Figure 5.28: Boxplot of results of experiments using scenario 3

One-way ANOVA is used, with the normalized area error as response variable, with the objective of understanding if the results are significantly different. With a significance level α=0.05, the obtained p-value is 0.836. Therefore, the null hypothesis stating that all means are equal cannot be rejected. The overall result can be caused by the low variability which characterizes the physical system in this configuration.

The results of scenario 4 present more interesting findings, given the much higher variability in the performances. The normalized area errors are shown in Table 5.13.

Forecast Frequency [min^{-1}]	Time Between Forecasts [min]	Normalized Area Error
0.200	5	6,41
0.100	10	6,67
0.667	15	6,77
0.050	20	7,51
0.025	40	8,52

Table 5.13: Numerical results of the experiments on scenario with failures (scenario 4)

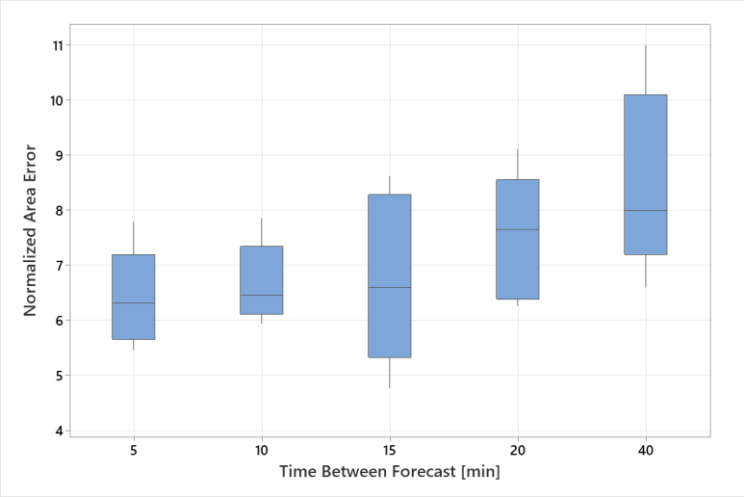


Figure 5.29: Boxplot of results of experiments on scenario 4

For scenario the obtained p-value is 0.021. Therefore, the null hypothesis stating that all means are equal is rejected. The values for the different forecast are grouped using Fisher pairwise comparisons, as shown in Table 5.14.

Forecast Frequency [min^{-1}]	Time Between Forecasts [min]	Grouping
0.200	5	A
0.100	10	A
0.667	15	A
0.050	20	A B
0.025	40	B

Table 5.14: Grouping using Fisher pairwise comparison.

The overall results of this experimental campaign prove the hypothesis that forecasts performed with a higher frequency result in more accurate predictions only in contexts with high variability, such as scenario 4. In lower variability contexts, the benefit is not significant.

5.2.5. Experimental campaign conclusions

The overall finding and results of the three experimental campaigns are summarized:

- Experimental campaign 1 aims to prove that the utilisation of an alignment model during prediction in operational phase is able to produce increasingly precise results, the closer the analyses get to the target point. The results obtained predicting the parts produced at the 20th min prove this hypothesis while for predictions targeting 40th min the results is not significant.
- Experimental campaign 2 tests the context of prediction with a misaligned model. While the mean of the distribution replicating the processing has

significant influence, the variance and distribution type factors do not show significant impact.

- Experimental campaign 3 intends to prove that an higher forecast frequency leads to an overall lower value of prediction error. Although in low variability contexts the results are rejected, a significant confirmation of the hypothesis is obtained in a context with unpredictable events.

In conclusion, some features characterizing the physical system affect the results of this experimental phase. The low variability offered by the production processes, the relatively short demonstration time, together with the simplicity of the lab-scale model do not allow to concretely determined the benefits provided by an aligned digital model in the context of forecasting.

In the next chapter the implementation of the proposed methodology in an industrial case scenario is presented. A fully functioning Digital Twin is implemented with the functionalities of real-time monitoring, prediction, what-if analysis, and subsequent feedback reaction to an unpredictable disruptive event.

6. Case study

This section illustrates a case study in which the methodology described in chapter 3 is applied. The work aims to produce a proof-of-concept Digital Twin of a manufacturing system. In particular, it extends the opportunity to test the developed platform in a more demanding and unstable context. Differently from chapter 4, the controller component is tested in its multitasking ability to manage all the three alignment actions of: model update, synchronisation, and input model update.

6.1. Use cases definition

Through a demonstration, a wide range of services is provided. Figure 6.1 shows the use cases which are meant to be assessed, clustered in three main categories:

1. Analytics, where historical performances and forecast evaluations can be found, together with the eventual what if analyses. For example, the manager requests the calculation of the future production rate of the analyzed system in its current configuration.
2. Monitoring deals with the real-time status of the manufacturing plant. For instance, the possibility of the manager to obtain information on the production status that would otherwise be impossible to directly acquire from the real system.
3. Reaction to unpredictable events that disrupt normal operations. For example, a failure on one element of the production line is detected, and the manager requests the analysis of possible solving actions that would be automatically implemented into the real system.

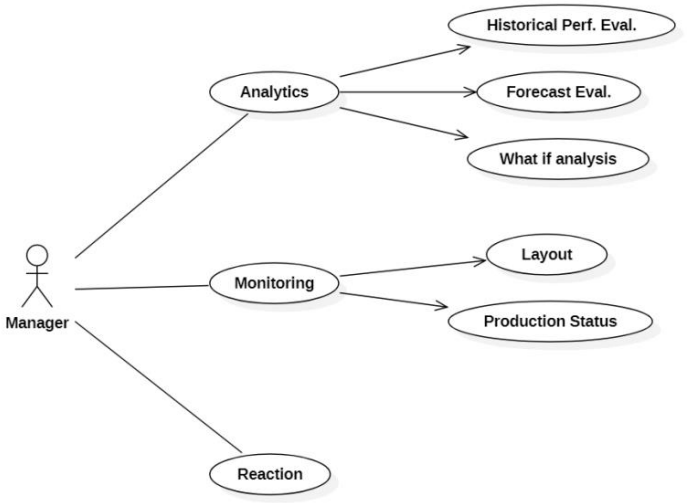


Figure 6.1: Use case diagram for the Digital Twin of a manufacturing system

Additional components, in particular the Evaluator developed by the authors and the Validator built by Gangemi et al. [30] are integrated to realize a complete DT framework.
The elements that compose the Digital Twin are shown in the component diagram pictured in Figure 6.2, where in white are the ones already presented in chapters 3 and 4, while shadowed are the ones introduced for this implementation.

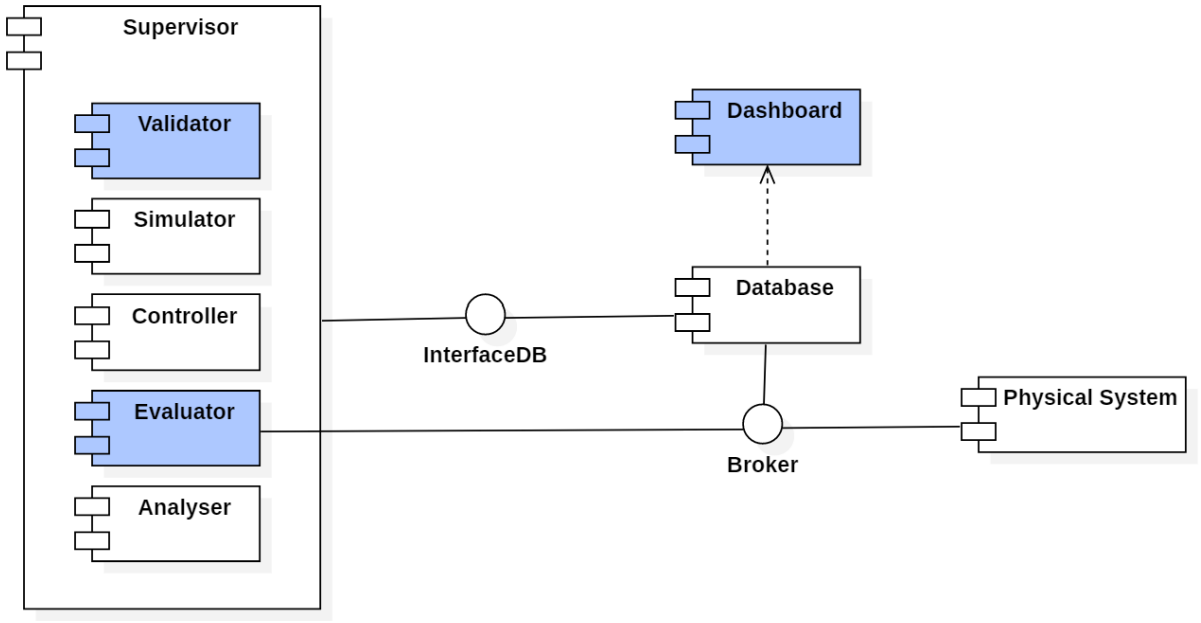


Figure 6.2: Component diagram of the Digital Twin

The physical system is interfaced with the database through the Broker, which transmits the inbound and outbound MQTT messages introduced in section 4.2.

The interconnection between the Supervisor and the Database is realized though the InterfaceDB. The Supervisor component administers several subcomponents:

- Simulator.
- Controller, which manages eventual aligning actions.
- Analyser, which is in charge of computing derivate performance values of the physical system in real time.
- Validator, in charge of the comparison procedure between the digital and physical objects.
- Evaluator, which detects disruptive events, performs what-if analyses, and introduces eventual feedback actions on the real system through the Broker.

The last two components have been specifically developed for this implementation and are furtherly explained in the following sections.
The complete class diagram of the architecture is present in Appendix A.4.1.
The real-time data and performances coming from the Database are visualized in the Dashboard.

6.1.1. Validator

The Validator component works in parallel with the Controller. It is in charge of all the necessary procedures for the computation of the markers indicating the alignment status between the physical and digital systems. Specifically, it performs the "logic validation" and "input validation" functions. The first uses trace-driven simulations to determines if the logic of the digital model is correct. The second performs quasi trace-driven simulations to check the alignment of the stochastic parameters. The procedures give as output two numerical indicators. These are numbers in the interval $\in [0,1]$ and indicate the alignment level between the two objects. For values lower than certain thresholds, the corresponding marker is set to indicate a failed alignment and is uploaded to the Database. These are consequently used by the Controller to trigger the re-aligning actions of Model update and Input model update.

The component is activated with a fixed and pre-determined frequency, and it performs the two validation procedures in sequence. In Figure 6.3 is shown the class diagram of the Validator.

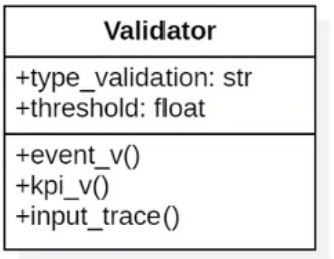


Figure 6.3: Class diagram of the Validator component

The attributes determine which type of validation is preformed and the corresponding threshold value, while the three methods are used to activate the procedures. In addition, the values of the loop frequency and query time must be given as input.

6.1.2. Evaluator

The Evaluator is in charge of three actions:

- Detection of disruptive events that may affect the physical system.
- What-if analysis, to determine if possible pre-determined solution actions can be executed to mitigate the effect of the detected events.
- Implementation of the selected feedback action on the physical system.

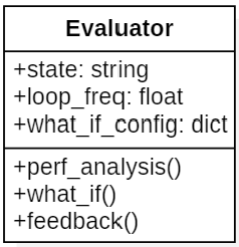


Figure 6.4: Class diagram of the Evaluator component

The class diagram of the component is shown in Figure 6.4. The "state" attribute can have three values: "waiting", "standby", and "evaluate".
During normal operations, the state is set to "standby", where periodically the marker indicating the alignment status of the stochastic behaviors is extracted from the Database, with a frequency determined by the "loop_freq" attribute.
If the distribution of the model are no longer validated, the Evaluator is set to the "waiting" state, where it still checks on the same marker, although this time it is waiting for new fitted distributions to be validated.
Eventually, when the marker indicates that the model is valid again, the component is set to the "evaluate" state and the "perf_analysis()" method is called, which starts a series of steps, shown in the sequence diagram pictured in Figure 6.5:

1. Performance analysis: a stochastic simulation is executed, with the objective of computing the new performance of the system, specifically the average throughput until the end of the shift. This value is then compared, using a paired t-test, to the one obtained before the event that disrupted the behavior of the production line. The objective is to detect if the new performance is significantly different from the old one, particularly if it is worse. If that is the case, the what-if phase is initiated by calling the "what_if()" method.
2. During the what-if phase, all the pre-defined solutions, contained in the "what_if_config" dictionary, are evaluated by simulating the system behavior until the end of the shift. Each scenario is replicated 10 times, and the best

performing solution is chosen as the one to be implemented in the physical system.

3. In the reaction phase, the command for the implementation of the selected solution is transmitted to the physical system's PLCs through the Broker, using a MQTT message. This is done by the "feedback()" method. Finally, the Evaluator is set back to the "standby" state.

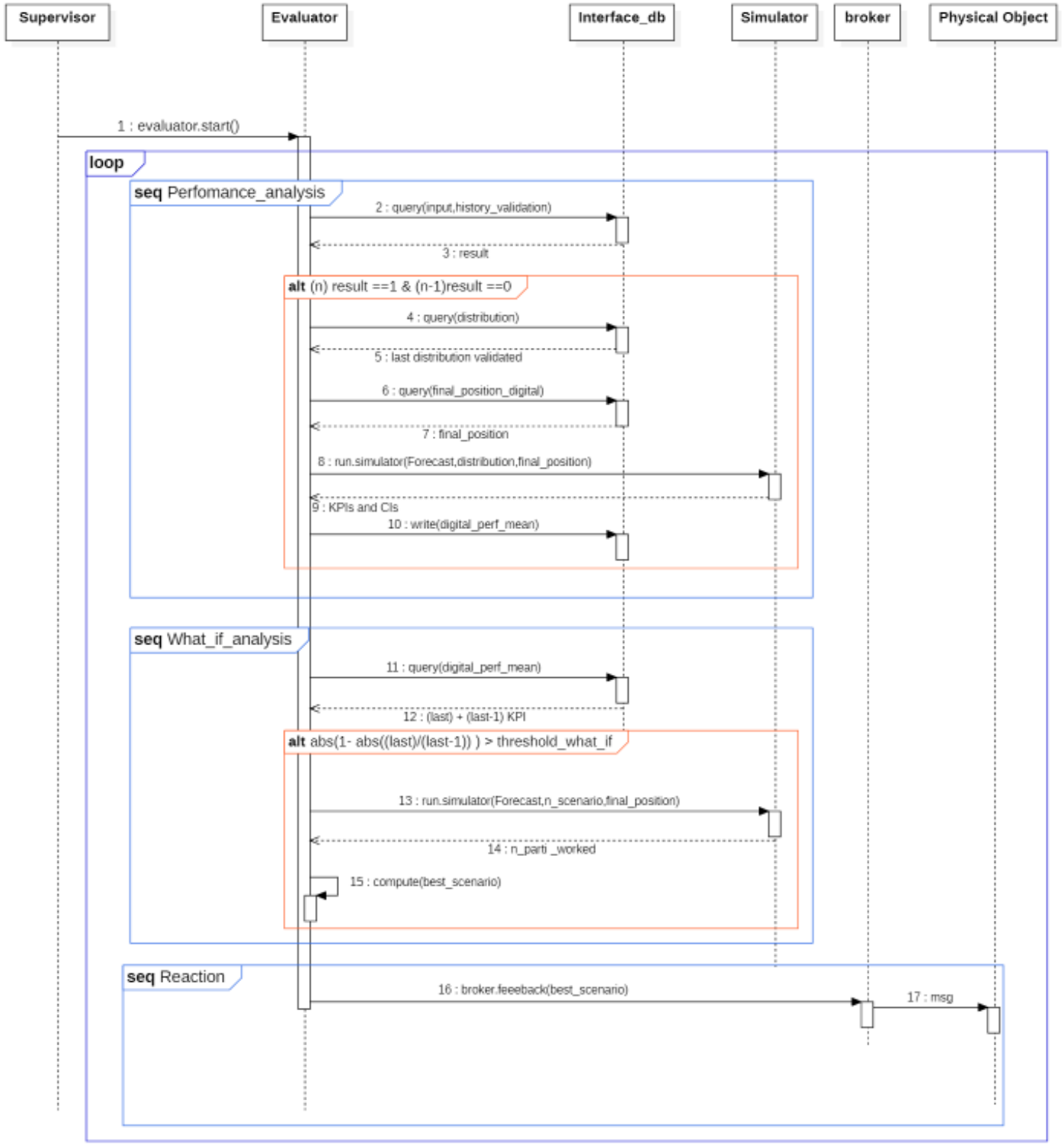


Figure 6.5: Sequence diagram of the Evaluator component

6.1.3. Dashboard

The Dashboard is developed using Grafana, which allows the real-time visualization of the results shown in Figure 6.6.

Figure 6.6: Dashboard layout

As the image shows, it is divided into four sections.
In the "Physical system performances" area the performance indicators of the real system are presented, all computed in real time by the Analyser component:

- 1 and 2 show the histograms of the processing times of Station 1 and Station 2 respectively of the last 5 minutes of acquired data. Below each graph is reported the last updated distribution.
- 3 depicts the system time values in seconds, calculated every time a part completes the processing loop.
- 4 depicts the instant throughput, calculated in parts/seconds.

The sector "Digital system performances" shows the performances calculated by the digital model. These are the results of the monitoring function.

- The last acquisition for system time and throughput indicators are shown in 5 and 6. By comparing them with the real performance, 3 and 4 respectively, it is possible to validate the aligned behavior.
- 7 represents the utilization values of the two machines. These cannot be calculated from the real system alone.

The "Validation results" area is dedicated to the validation functions:

- 8 shows the results of the input validation.
- 9 shows the results of the logic validation.

Finally, the section "Mean performances and what-if" depicts information regarding the results of the Evaluator component:

- 10 are the current average performances of the system, calculated as the throughput in parts/hour.
- 11 is dedicated to the eventual what-if analysis, with the "case in action" value indicating the happening of the disruptive event.
- 12 are the sub-sections dedicated to the two solution scenarios, with the forecasted performances and an indicator showing which of the two is implemented.

6.2. The demonstration: settings and results

A demonstration is set in order to verify the functioning and study the contributions of the platform. The purpose is to prove the capabilities of the DT framework to autonomously monitor the performance of the system, provide forecast analyses and react to solve possible deteriorating productive patterns.
The lab-scale production line explained in Section 4.1 has been set to represent a system operating for limited time-shifts distanced by switched-off periods. For this demonstration an operating shift of 50 minutes is used. The physical system is characterized by: perfectly reliable machines, processing of a part one at the time, Blocking After Service (BAS) discipline and buffers capacity of 8 pallet with 12 pallets circulating. The two stations are configured to process pallets with defined distribution parameters: Machine 1: *Triangular(3,8,5)* and Machine 2: *Triangular(2,5,3).*

The digital components' settings are tuned, with the knowledge acquired from the previous tests, to effectively and efficiently achieve the introduced purposes. Further studies might support the definition of the optimal parameters, although they would strictly be related to the objectives and real conditions of the system. The parameters used for the demonstration are presented in Table 6.1.

Components	Sub-components	Settings
Controller	Model_update	t_horizon: *10m*, freq: *marker dependant*
	Synchroniser	t_horizon: *5m*, freq: *marker dependant*
	Input_Model_Update	t_horizon_dist: *10m*, freq: *marker dependant*
Simulator		Simulator_type: *"Manpy"*, use_type: *"sync/forecast"*
Evaluator		Forecast_rep: *20*, sim_length: *remaining time of demonstration*
Analyser		t_horizon: *10 m*, freq: *0.5s*
Validator		t_hozion: *10 m*, freq: *1/60 s*

Table 6.1: Components' settings

During one shift an instantaneous degradation event is applied to one of the stations. This can represent a deterioration of an element in a real production machine, which is then required to produce at a slower pace. Once the event is detected, the digital platform runs a series of forecast analyses. Different options scenarios are tried with the aim of mitigating the disruptive effect and optimizing the production within the end of the shift:

- Scenario 1: do not stop the operations and repair the machine only when the shift ends. This way the production is not interrupted, but the line produces at an overall slower pace.
- Scenario 2: stop the line and allow the repair of the machine to its original conditions. When the intervention is done, operations are restarted.

The best solution, based only on the number of parts produced, is automatically implemented to the real system, without considering any cost-based information.

The steps which compose the demonstration timeline are shown in Figure 6.7.

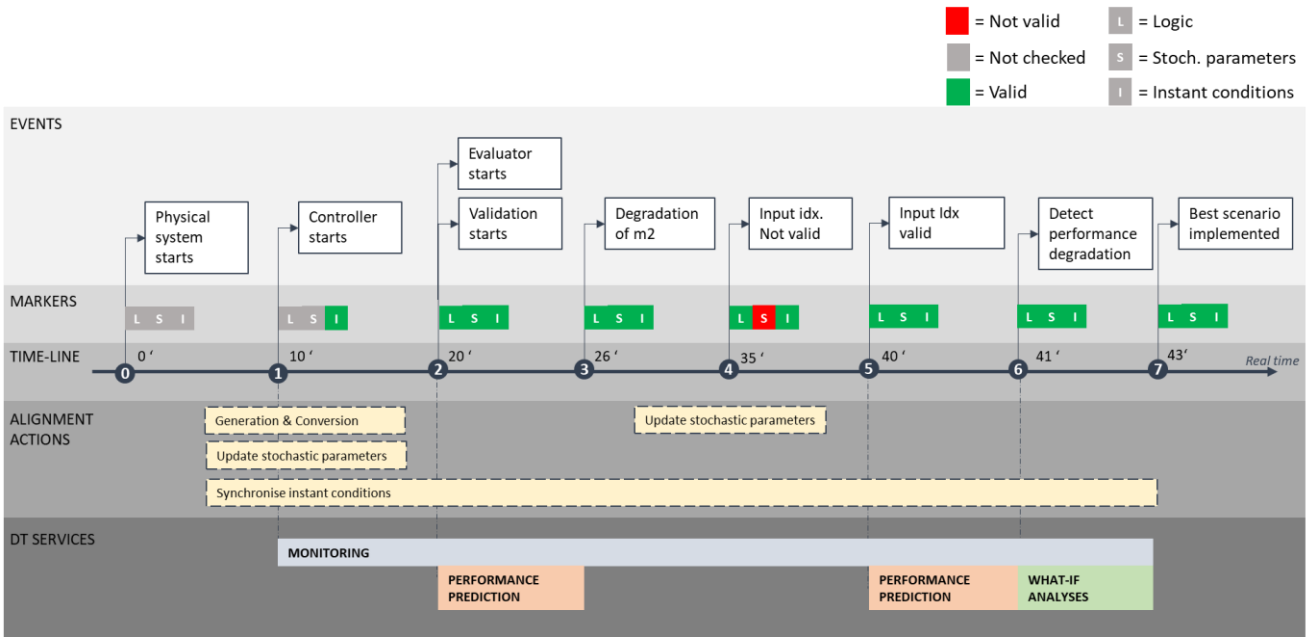


Figure 6.7: Demonstration events timeline

Each event is detailly explained below:

0. The physical system is started and initially all pallets are positioned in the queue of machine 1. Also, the Analyser begins computing in real-time the derivate KPIs shown in Figure 6.8, respectively: the histograms of the processing times from Station 1 (a) and 2 (b), the system time (c), and the throughput (d).

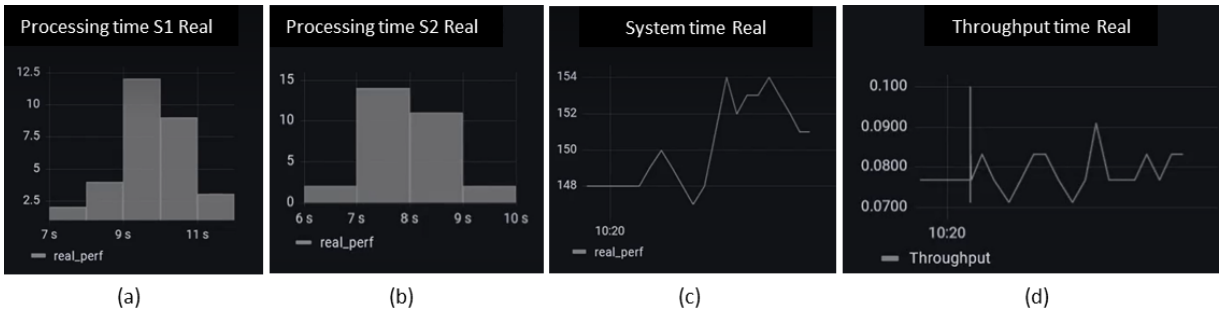


Figure 6.8: Dashboard illustration of real performance at real system start (at event 0)

1. After 10 minutes of operation, the controller has enough data to start the actions to align the digital system. Since the Validator is not activated yet, two markers are missing, but the controller acts anyway in the following sequence: it generates and converts the digital model, updates the stochastic distributions related to the processing times for machine 1 and machine 2, and finally begins to synchronise the instant conditions.
2. The Validator is activated, examining both the logic and the input of the digital system. All markers are checked; therefore, the alignment is completely confirmed and the monitoring of the real system performance is to be

considered correct. Also, the Evaluator can perform the first forecast analysis predicting the average production rate of the line.
In Figure 6.9 the instant system time and throughput calculated for the real system can be compared with the digital performance obtained through the implementation of trace-driven simulations. In particular the last value computed through the simulation is shown validating the alignment and resulting performances.

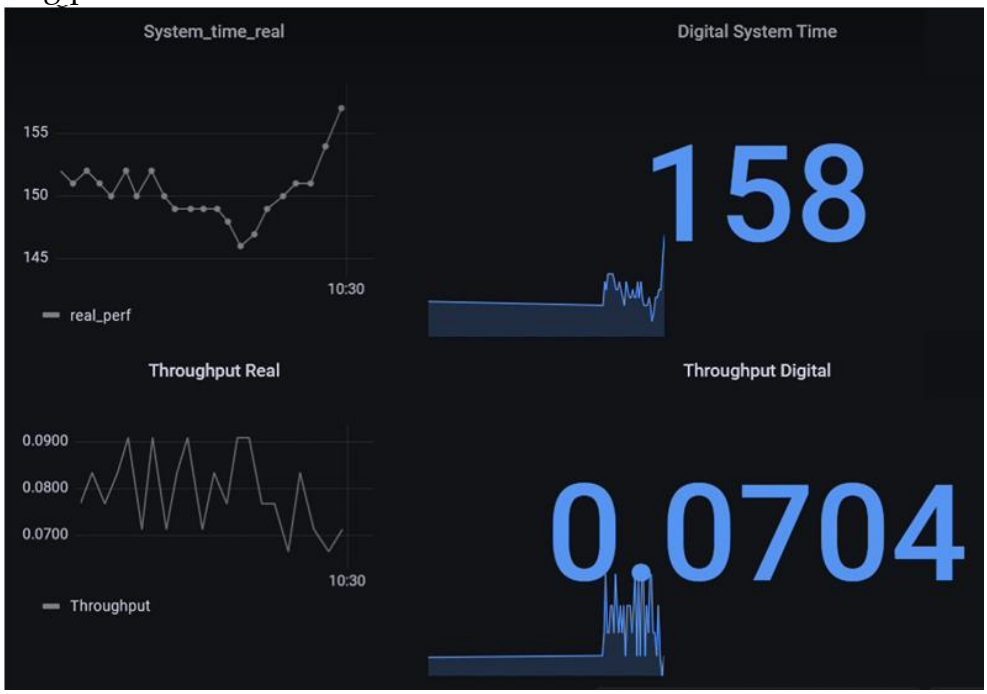


Figure 6.9: Dashboard illustration comparing system time and throughput both on the real and digital system (at event 2)

Figure 6.10: Dashboard illustration of additional digital performance (utilisation) and validator outcomes (at event 2)

Figure 6.10 illustrates the additional information provided through monitoring the real system's conditions. In particular, as (a) indicates, station 1 is the bottleneck initially.

3. An instantaneous degradation of machine 2 happens. By monitoring the digital performance, it is possible to appreciate that the bottleneck now is shifted to machine 1.
   Figure 6.11 shows the change in the dynamic of the system, while Figure 6.12 illustrates how the digital performance acquired by the monitoring action highlight this event.

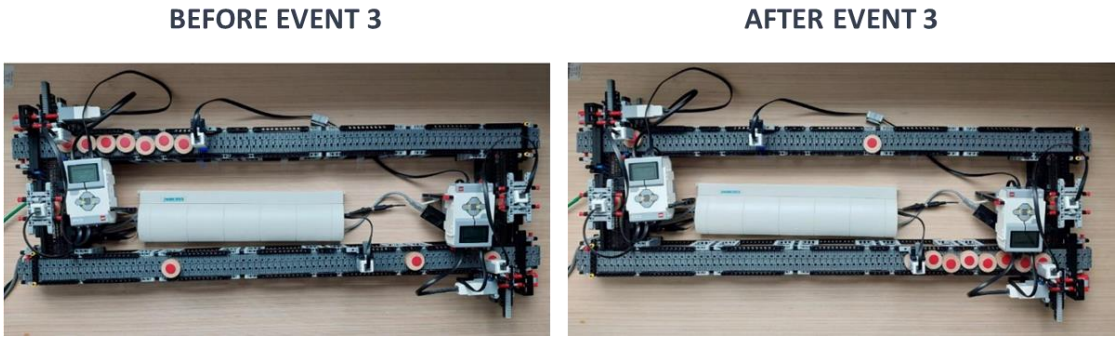


Figure 6.11: Real system state before and after the disruptive event

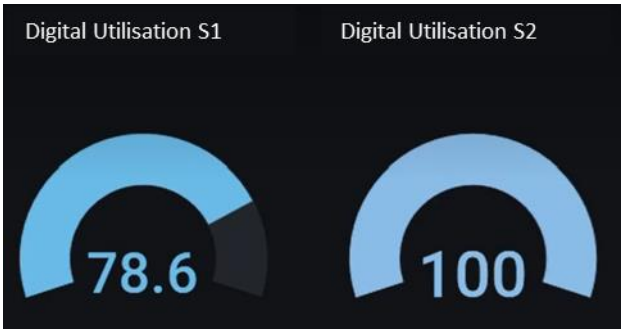


Figure 6.12: Dashboard illustration of digital performance after disruptive event (at event 3)

4. Consequently, the distribution parameters previously fitted are not valid anymore. This is signalled by the marker indicating the input validation.
   The controller is triggered into updating the stochastic parameters by fitting the processing times acquired from the real system historical data.
5. After some iterations, the Controller is able to correctly update the stochastic parameters, validated by the input marker. This activates the performance analysis function of the Evaluator, which detects a degradation of the performances, triggering the what-if analysis
6. The what-if analysis uses two predictive simulations based on the stochastic parameters validated and test the two scenarios pre-defined. The number of parts produced by the end of the demonstration are computed and the best scenario performing is found, Scenario 2 highlighted in green in Figure 6.13 is the resulting best choice.

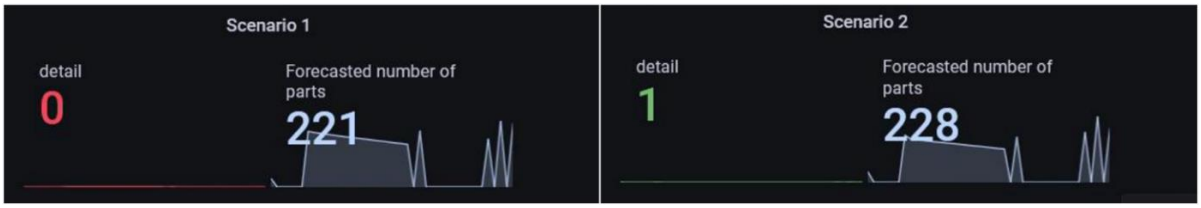


Figure 6.13: What if analyses results (at event 6)

7. The new configuration is sent as message by the broker component and applied onto the real system

6.3. Case study conclusions

This case study confirms the alignment capabilities of the developed methodology, integrating the actions managed by the controller and the information obtained through the comparison of real and digital systems.
The utilisation of particular indicators to identify the misalignment between real and digital conditions have effectively activated the expected actions. The performed automatic model update of a DES model demonstrates the opportunity of aligning morphological and logic characteristics when required. The synchronisation of instant conditions using the trace provided by processing parameters demonstrates the contribution of online simulation in the context of this precise framework. Additionally, the update of the distribution inputs by fitting the historical data allows the alignment of stochastic behaviours.

The introduction of other developed components, Evaluator and Validator allows the adoption of a complete Digital Twin framework offering many essential services. Through the demonstration, a continuous monitoring and prediction of the performances is accomplished. In addition, the detection of disruptive events offered by the Evaluator contributes to perform what-if analyses and the automatic deployment of corrective actions on the real system.

The application is characterized by a bi-directional flow of information supporting the several potentialities mentioned. Therefore, it can be considered as a proof-of-concept Digital Twin and as a starting point for future DT research. In conclusion, further implementations could be tested, extending the architecture developed on real industrial scenario and integrating additional components to foster extensive capabilities.

7. Conclusions & future developments

This work focuses on the challenge of maintaining aligned a Digital Twin and the production system. It is now evident that to reflect the real conditions is not only an essential requirement for every DT application, but it is of great impact on the results.
The methodology developed allows to expand the limited solutions found in literature. Also, it efficiently contrasts the critical causes of loss of alignment identified in the transformation of layout and logic, the advance of the instant conditions and the shift of stochastic behaviors. In particular, an indicator based control algorithm is integrated in a digital architecture to effectively administer the aligning actions. The digital platform developed includes several components which appropriately operate to provide the services of monitoring the performance of the real system, forecast future behaviors and conduct what-if analyses to anticipate and tackle possible disruptive productive patterns. A complete assessment of the DT capabilities is provided through several experiments a case study adopting a lab-scale model as physical representation of a real manufacturing system.

7.1. Main achievements

The most important contributions of this work concern the alignment between the DT and the physical system during the operational phase. In particular the procedures integrated in the architecture tackle three challenges causing misalignment between the two entities. The automatic model update of DES model solves the discrepancy in layout and logic, while the syncrhonisation allows the instant conditions present in the real system to be reflected in Real-Time in the DT. As process patterns evolve, the update of stochastic inputs based on historical data permits to achieve more precise predictions.
The control algorithm uses three indicators detecting modifications in the layout and logic, advance of instant conditions and the evolution of stochastic behaviours. Thus, only if a misalignment is detected than the related procedure is triggered. The integration of such management implies a better efficiency when procedures are deployed and at the same time creates room for new techniques and indicators to be introduced. In addition, the platform built shows great flexibility by interoperating with multiple digital tools allowing the introduction for the scope of the case study of new components. The Validator is introduced to compute the logic and input

indicators, while the Evaluator detect disruptive events and transmit the best alternative chosen, as a result of what-if analyses. The last achievement of this work is the realisation of a proof-of-concept DT with bi-directional capabilities applied on a lab-scale manufacturing system.

7.2. Limitations

The results obtained are a step forward for the exploration of DT applications in the manufacturing field. However, some limitations are to be highlighted to comprehend the direction of future works on the alignment concept.
Within the integrated procedures: the automatic model update is limited when deployed on more complex model's logics e.g., assembly or mixed products manufacturing. Also, synchronisation is dependent on the type of data received. This not only requires data to be complete otherwise results will be affected, but might induce some criticalities in applications with more complex systems. As a matter of fact, this work has been experimented on a fairly simple production line, therefore challenges related to more complicated systems have not been investigated.
Finally, the architecture is impacted strongly in precision by the hardware sensors installed onto the production plant.

7.3. Future developments

The next steps of this study might focus on the elaboration of procedures able to approach more challenging systems. The synchronisation as of now is highly dependent on the precision and completeness of the data acquired by the physical system. Thus, the introduction of machine learning algorithms could be used to reduce this dependency and therefore generate a new research discussion towards the decrease of cost maintaining reliable results. The indicators used by the control algorithm are the outcomes of particular functions intended to measure the divergency between the DT and the production system. Therefore, further studies related to these root-causes analyses might be helpful to improve the techniques used and introduce new ones. A conclusive further development would target the application of the methodology onto a real manufacturing system and the integration with other industrial software.

In conclusion, this thesis work aims at exploring the alignment of DT and physical system and proposes an initial methodology to achieve more flexible and adaptable applications. The further components implemented has allowed to build a proof-of-concept DT integrated onto a lab-scale model and pave the way for future research works.

8. Bibliography

[1] M. Grieves, *Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management*. 2011.

[2] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems(Excerpt)", doi: 10.13140/RG.2.2.26367.61609.

[3] E. Negri, L. Fumagalli, and M. Macchi, "A Review of the Roles of Digital Twin in CPS-based Production Systems," *Procedia Manufacturing*, vol. 11, pp. 939–948, 2017, doi: 10.1016/j.promfg.2017.07.198.

[4] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification," Jan. 2018, vol. 51, no. 11, pp. 1016–1022. doi: 10.1016/j.ifacol.2018.08.474.

[5] W. J. Davis, "On-Line Simulation: Need and Evolving Research Requirements," in *Handbook of Simulation*, John Wiley & Sons, Ltd, 1998, pp. 465–516. doi: https://doi.org/10.1002/9780470172445.ch13.

[6] A. Hanisch, J. Tolujew, and T. Schulze, "Initialization of online simulation models," in *Proceedings - Winter Simulation Conference*, 2005, vol. 2005, pp. 1795–1803. doi: 10.1109/WSC.2005.1574454.

[7] G. Shao, "Use Case Scenarios for Digital Twin Implementation Based on ISO 23247," Gaithersburg, MD, May 2021. doi: 10.6028/NIST.AMS.400-2.

[8] J. W. Fowler and O. Rose, "Grand challenges in modeling and simulation of complex manufacturing systems," *Simulation*, vol. 80, no. 9, pp. 469–476, Sep. 2004, doi: 10.1177/0037549704044324.

[9] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital Twin in manufacturing: A categorical literature review and classification."

[10] F. Chance, J. Robinson, and J. W. Fowler, "Supporting manufacturing with simulation," 1996, pp. 114–121. doi: 10.1145/256562.256586.

[11] J. W. Yücesan Enver and Fowler, "Logistics systems, simulation analysis ofSimulation analysisSIMULATION ANALYSIS OF MANUFACTURING AND LOGISTICS SYSTEMS," in *Encyclopedia of Production and Manufacturing*

Management, P. M. Swamidass, Ed. Boston, MA: Springer US, 2000, pp. 687–697. doi: 10.1007/1-4020-0612-8_881.

[12] A. A. C. Vieira, L. M. S. Dias, M. Y. Santos, G. A. B. Pereira, and J. A. Oliveira, "Setting an industry 4.0 research and development agenda for simulation – A literature review," *International Journal of Simulation Modelling*, vol. 17, no. 3. DAAAM International Vienna, pp. 377–390, Sep. 01, 2018. doi: 10.2507/IJSIMM17(3)429.

[13] G. Lugaresi and A. Matta, "Generation and Tuning of Discrete Event Simulation Models for Manufacturing Applications," in *Proceedings - Winter Simulation Conference*, Dec. 2020, vol. 2020-December, pp. 2707–2718. doi: 10.1109/WSC48552.2020.9383870.

[14] Y. T. T. Lee, F. H. Riddick, and B. J. I. Johansson, "Core Manufacturing Simulation Data - A manufacturing simulation integration standard: Overview and case studies," *International Journal of Computer Integrated Manufacturing*, vol. 24, no. 8, pp. 689–709, 2011, doi: 10.1080/0951192X.2011.574154.

[15] C. Author and S. C. Mathewson, "Simulation Program Generators: Code and Animation on a P.C.," 1985. [Online]. Available: https://about.jstor.org/terms

[16] D. Krenczyk, W. M. Kempa, K. Kalinowski, C. Grabowik, and I. Paprocka, "Integration of manufacturing operations management tools and discrete event simulation," in *IOP Conference Series: Materials Science and Engineering*, Sep. 2018, vol. 400, no. 2. doi: 10.1088/1757-899X/400/2/022037.

[17] C. Haraszkó and I. Németh, "DES Configurators for rapid virtual prototyping and optimisation of manufacturing systems," *Periodica Polytechnica Mechanical Engineering*, vol. 59, no. 3, pp. 143–152, 2015, doi: 10.3311/PPme.7888.

[18] R. Bloomfield, E. Mazhari, J. Hawkins, and Y. J. Son, "Interoperability of manufacturing applications using the Core Manufacturing Simulation Data (CMSD) standard information model," *Computers and Industrial Engineering*, vol. 62, no. 4, pp. 1065–1079, May 2012, doi: 10.1016/j.cie.2011.12.034.

[19] G. Popovics, A. Pfeiffer, and L. Monostori, "Generic data structure and validation methodology for simulation of manufacturing systems," *International Journal of Computer Integrated Manufacturing*, vol. 29, no. 12, pp. 1272–1286, Dec. 2016, doi: 10.1080/0951192X.2016.1187296.

[20] D. Krenczyk, "Automatic generation method of simulation model for production planning and simulation systems integration," in *Advanced Materials Research*, 2014, vol. 1036, pp. 825–829. doi: 10.4028/www.scientific.net/AMR.1036.825.

[21] C. Meng, S. S. Nageshwaraniyer, A. Maghsoudi, Y. J. Son, and S. Dessureault, "Data-driven modeling and simulation framework for material handling

systems in coal mines," *Computers and Industrial Engineering*, vol. 64, no. 3, pp. 766–779, 2013, doi: 10.1016/j.cie.2012.12.017.

[22] "JMT Overview - Virtual Learning Factory Toolkit (gitbook.io)."

[23] W. Yang, K. Yoshida, and S. Takakuwa, "Digital Twin-Driven Simulation for a Cyber-Physical System in Industry 4.0 Era," 2017, pp. 227–234. doi: 10.2507/daaam.scibook.2017.18.

[24] O. Cardin and P. Castagna, "Proactive production activity control by online simulation," 2011. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00784336

[25] C. Scheifele, A. Verl, and O. Riedel, "Real-time co-simulation for the virtual commissioning of production systems," in *Procedia CIRP*, 2019, vol. 79, pp. 397–402. doi: 10.1016/j.procir.2019.02.104.

[26] R. S. Pritschow G, "'Hardware in the Loop' Simulation of Machine Tools," in *CIRP Annals 53(1)* , 2004, pp. 8–295.

[27] Schmoll R., "Co-Simulation und Solverkopplung: Analyse komplexermultiphysikalischer Systeme," 2015.

[28] Holger Zipper and Christian Diedrich, "Synchronization of Industrial Plant and Digital Twin," 2019.

[29] G. Lugaresi, V. V. Alba, and A. Matta, "Lab-scale Models of Manufacturing Systems for Testing Real-time Simulation and Production Control Technologies," *Journal of Manufacturing Systems*, vol. 58, pp. 93–108, Jan. 2021, doi: 10.1016/j.jmsy.2020.09.003.

[30] Sofia Gangemi and Giulia Gazzoni, "Real-time validation of Discrete Event Simulation models in a digital twin framework: an approach based on sequence comparison techniques," Politecnico di Milano, 2022.

A. Appendix

A.1 Paired t-test

The paired t-test is a statistical procedure applied to determine if the mean difference between paired of observations is close to zero.

The test hypothesis are:

- H_0: the mean difference is equal to 0
- H_1: the mean difference is different from 0

The test assumptions are:

- Independence of observations.
- Variables approximately normally distributed.

The procedure of the test is composed by three steps:

1. Given n replications for both groups, compute the differences:

   $D_j = Y_{1,j} - Y_{2,j}$ for j $= 1, \dots,$ n assuming $D_1, D_2, \dots, D_n$ i. i. d. normal

   Where $Y_{1,j}$ and $Y_{2,j}$ are the observations of the j-th replication respectively of group 1 and group 2.
2. Calculate the sample mean and the variance of the differences D_j

$$\overline{D}(n) = \frac{1}{n}\sum_{j=1}^{n} D_j \qquad S^2(n) = \frac{1}{n-1}\sum_{j=1}^{n}(D_j - \overline{D}(n))^2$$

3. Compute the confidence interval based on the mean of the differences, given a significance level α:

$$\mu_1 - \mu_2 \in \overline{D}(n) \pm t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}}$$

If the confidence interval contains the value, the null hypothesis H_0 cannot be rejected.

A.2 Physical system and software implementation

A.2.1 Software code

The software code of all the digital components is available in the following GitHub repository: https://github.com/digital-twin-lab/

A.2.2 Supervisor

Figure A.1 illustrates the class diagram of the supervisor simplified, i.e., Evaluator and Validator components are not present.

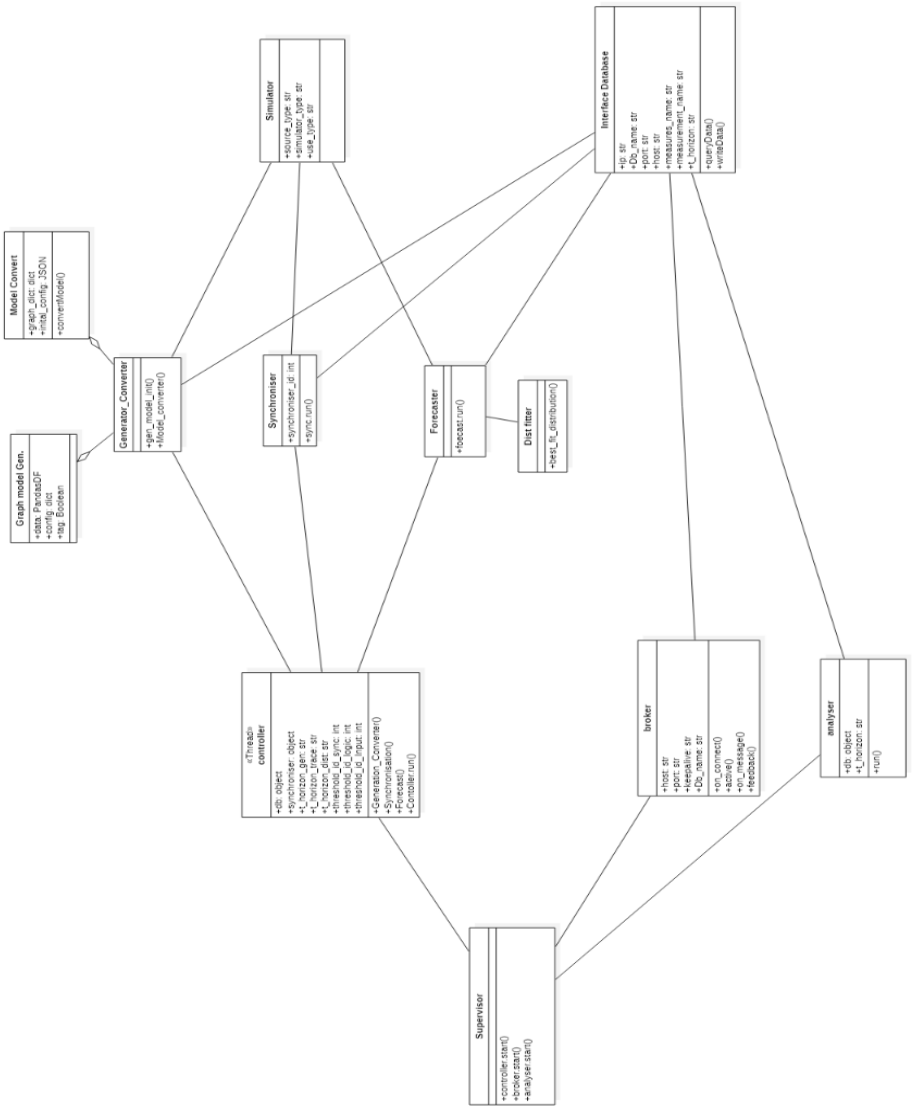


Figure A.1: Supervisor simplified class diagram

A.2.3 Controller

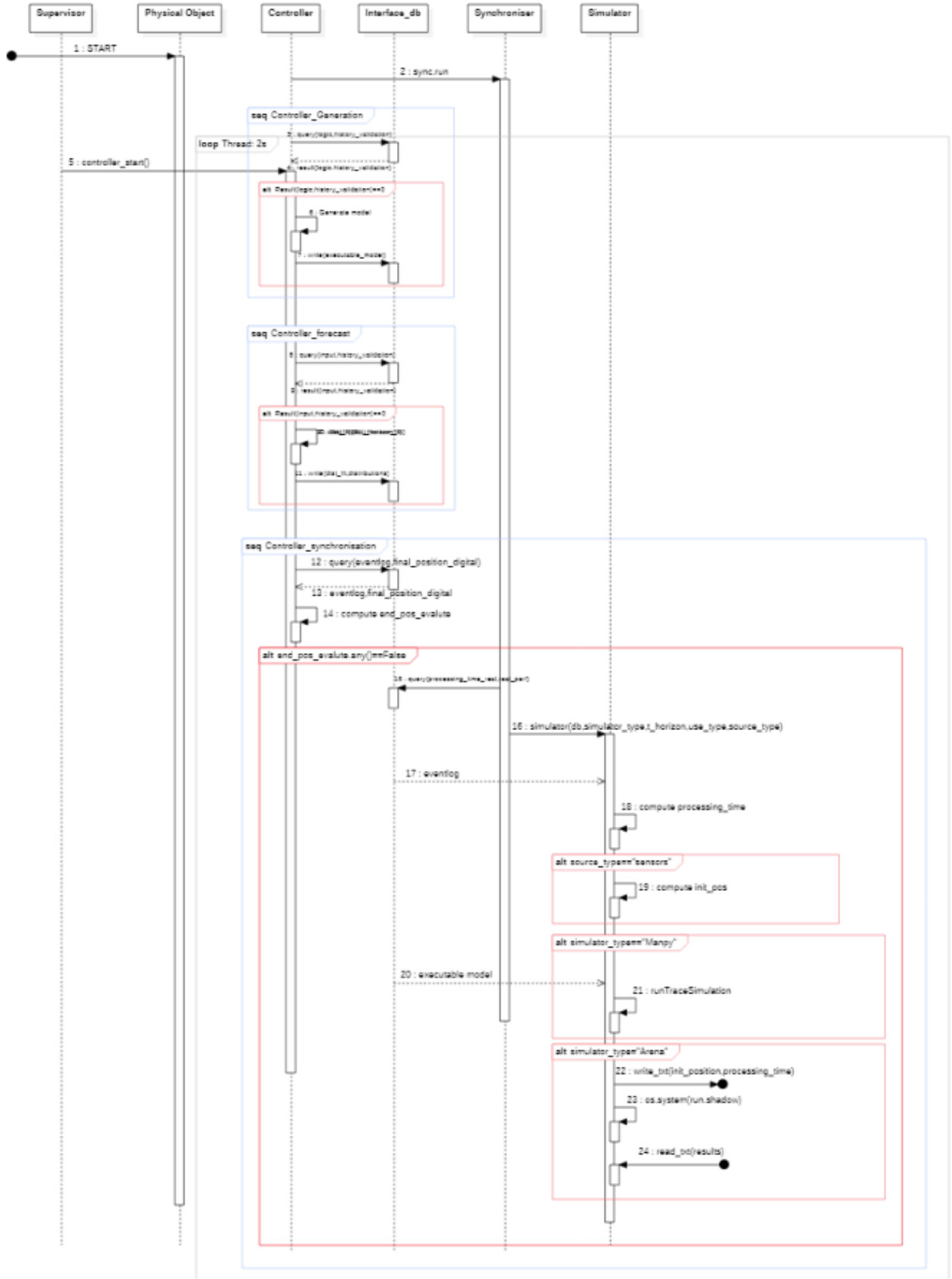


Figure A.2: controller sequence diagram

A.2.4 Synch check

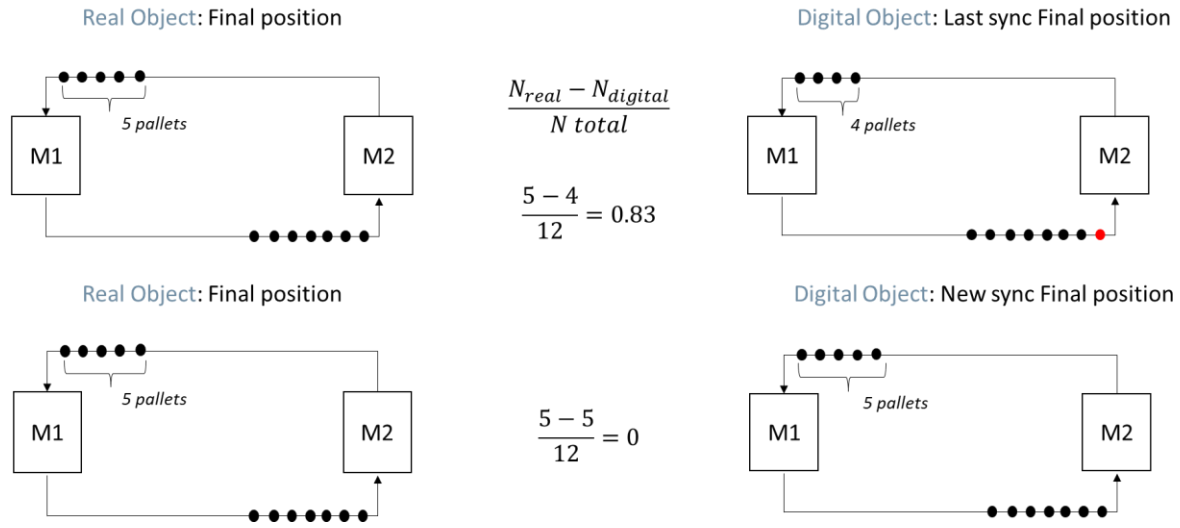


Figure A.3: Scheme of the synchronisation check function

A.2.5 Model update

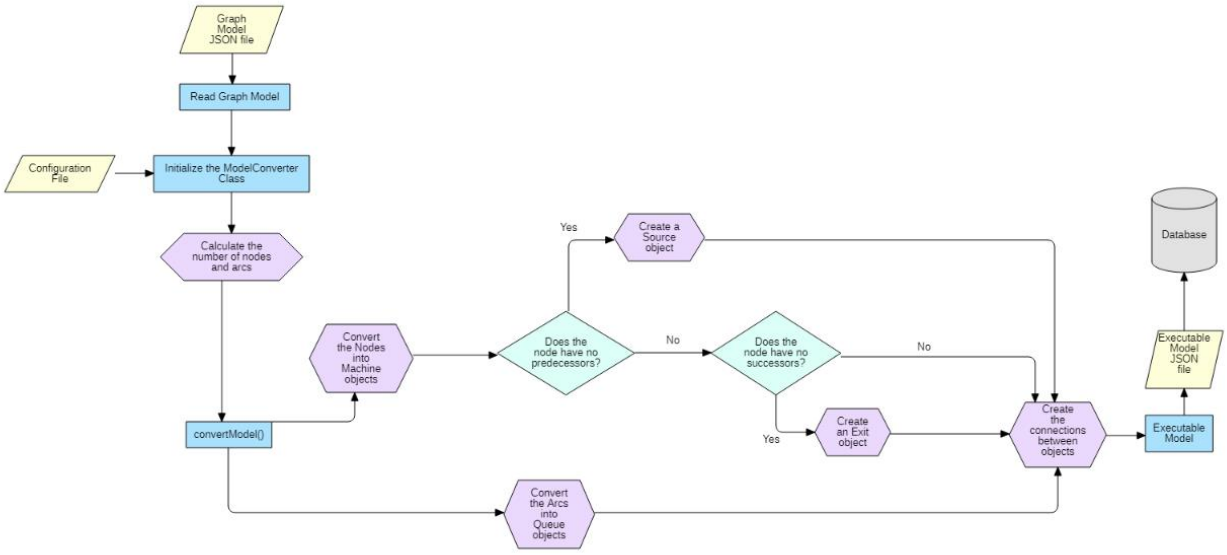


Figure A.4: Model Update Flow chart

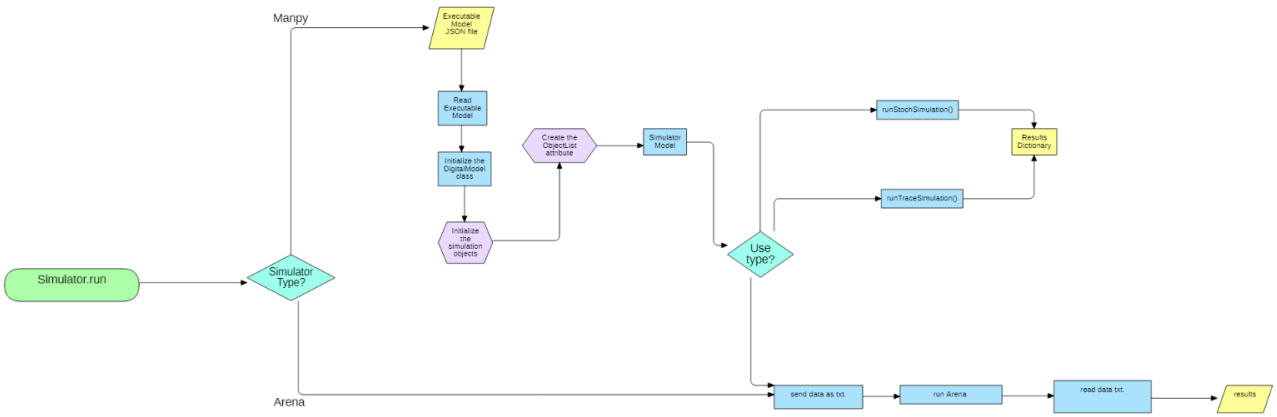


Figure A.5: Simulator flowchart

A.2.6 Synchronisation

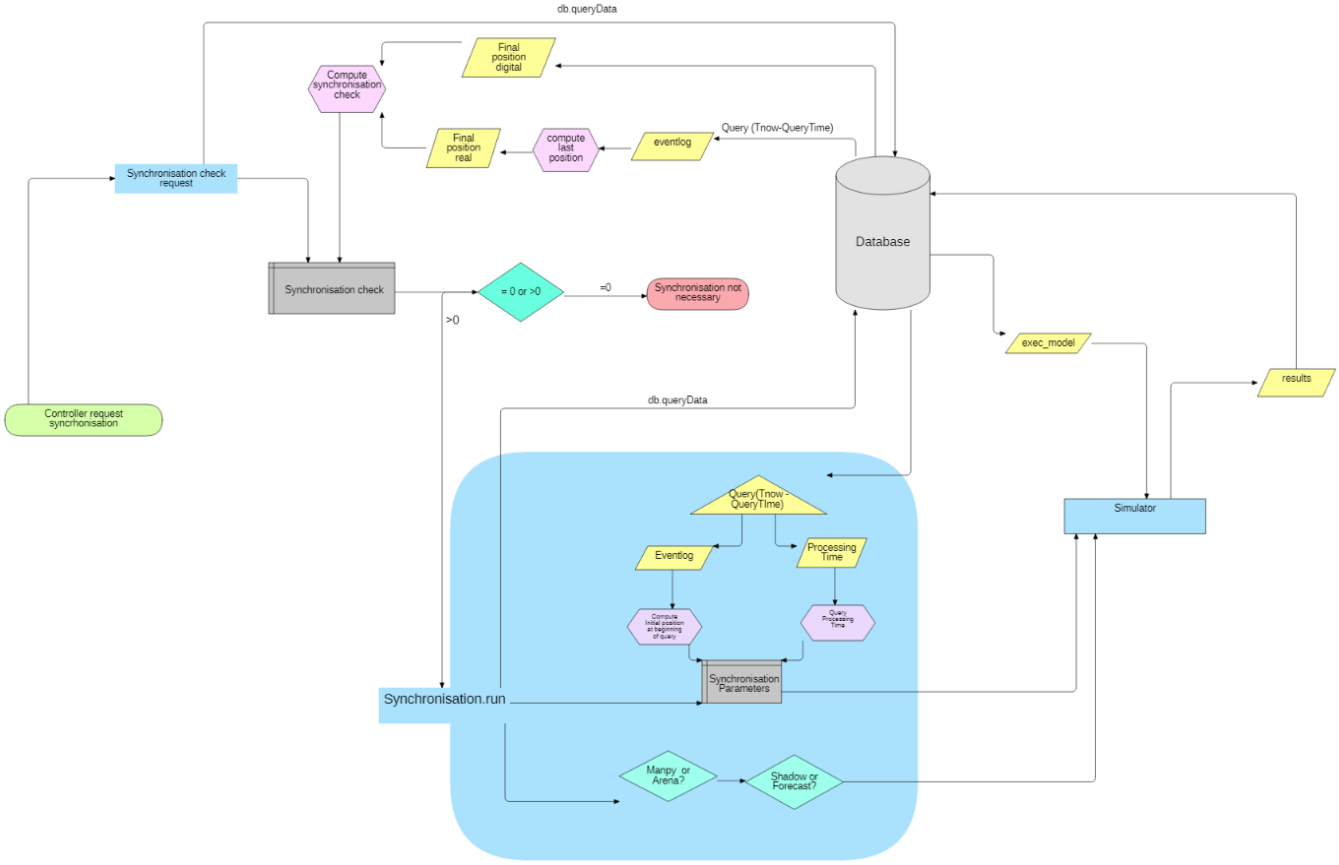


Figure A.6: Synchronisation flowchart

A.2.7 Broker

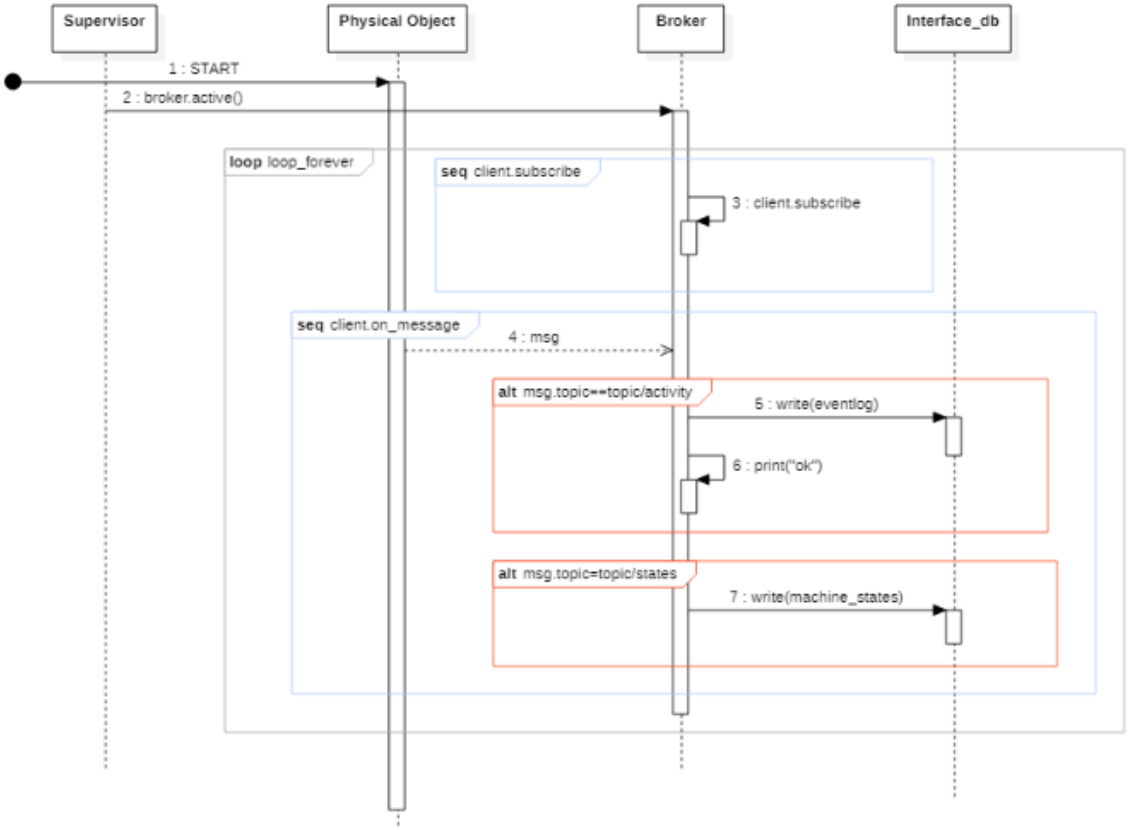


Figure A.7: Broker sequence diagram

A.2.8 Analyser

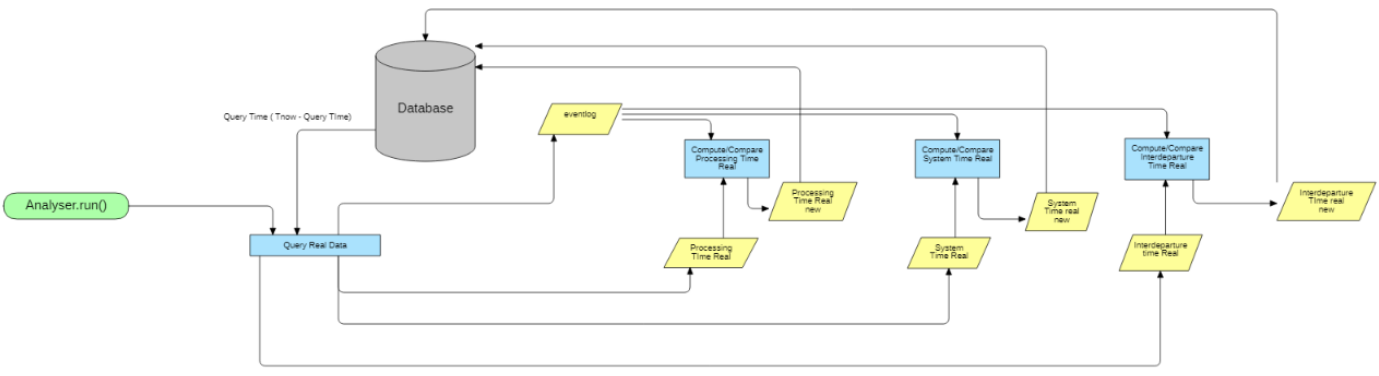


Figure A.8: Analyser flowchart

A.2.9 Forecast

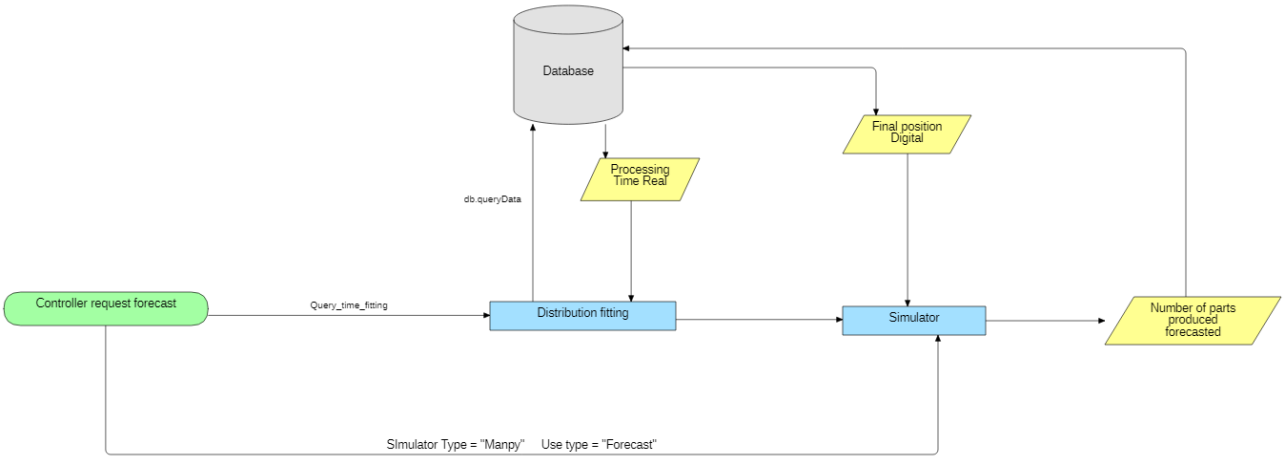


Figure A.9: Forecast flow chart

A.3 Test Bench

A.3.1 Conversion tables

	System Time [s]			Inter-arrival Time [s]		
	Mean	**St. Dev.**	**95%CI**	**Mean**	**St. Dev.**	**95%CI**
Arena	127.7295	0.2429	[127.4597 127.9993]	21.0039	0.0111	[20.9916 21.0162]
ManPy	127.7003	0.3411	[127.3215 128.0791]	21.0061	0.0098	[20.9952 21.0170]
Difference	0.0292	-	[-0.4907 0.5491]	-0.0022	-	[-0.0206 0.0161]

Table A.1: Conversion test results, Flowline

	System Time [s]			Inter-arrival Time [s]		
	Mean	**St. Dev.**	**95%CI**	**Mean**	**St. Dev.**	**95%CI**
Arena	134.4550	1.3552	[132.9500 135.9600]	30.5074	0.0329	[30.4708 30.5440]
ManPy	135.6167	1.6674	[133.7649 137.4685]	30.5143	0.0339	[30.47767 30.5519]
Difference	-1.1617	-	[-3.8296 1.5062]	-0.0069	-	[-0.0656 0.0517]

Table A.2: Conversion test results, Parallel

	Mean [s]	**St. Deviation [s]**	**95% CI [s]**	**%95 CI for Difference [s]**
Arena	302.88	0.74	[301.44, 304.31]	[-3.07, 0.63]
ManPy	304.0949	1.2916	[303.27, 304.31]	

Table A.3: Conversion test results, Real system LEGO

A.3.2 Synchronisation

8.1.1.1. Scenario 1

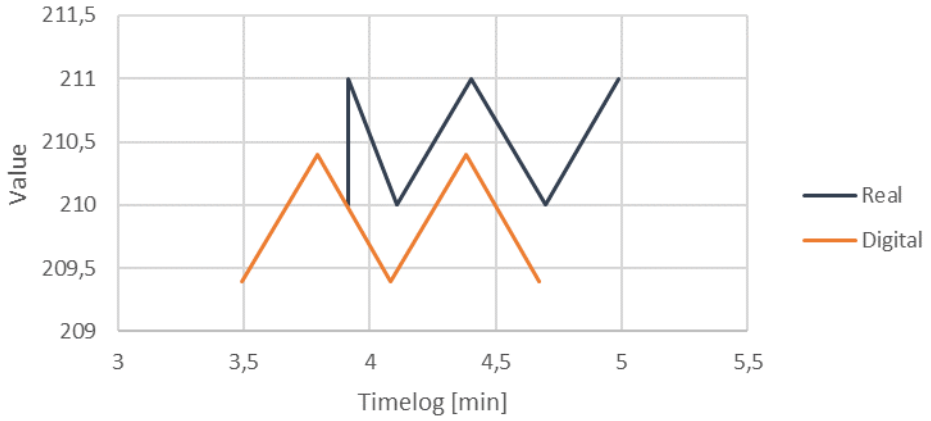


Table A.4: (a) Scenario 1, synchronisation system time

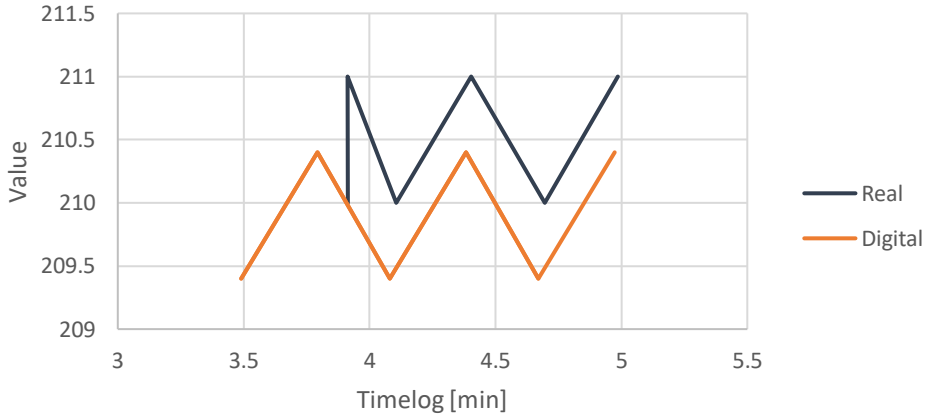


Table A.5: (b) Scenario 1, synchronisation system time

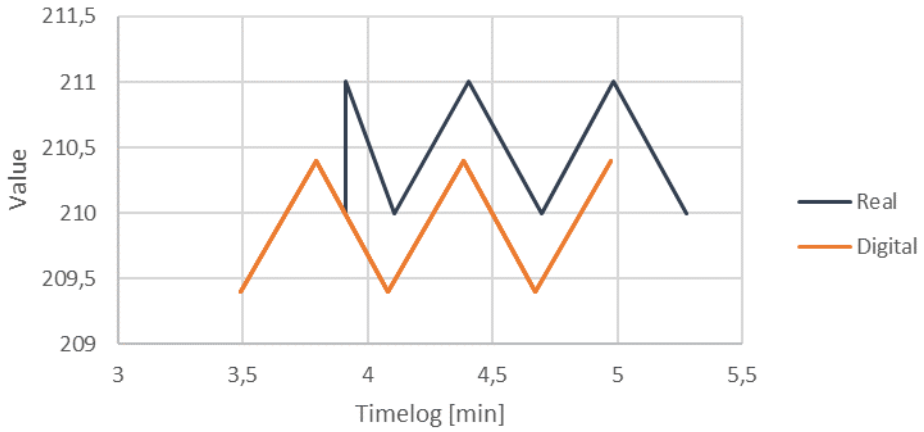


Table A.6: (c) Scenario 1, synchronisation system time

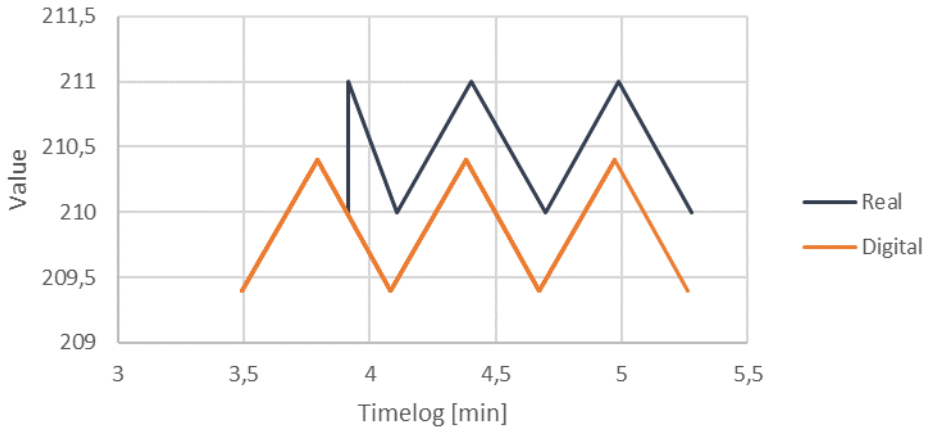


Table A.7: (d) Scenario 1, synchronisation system time

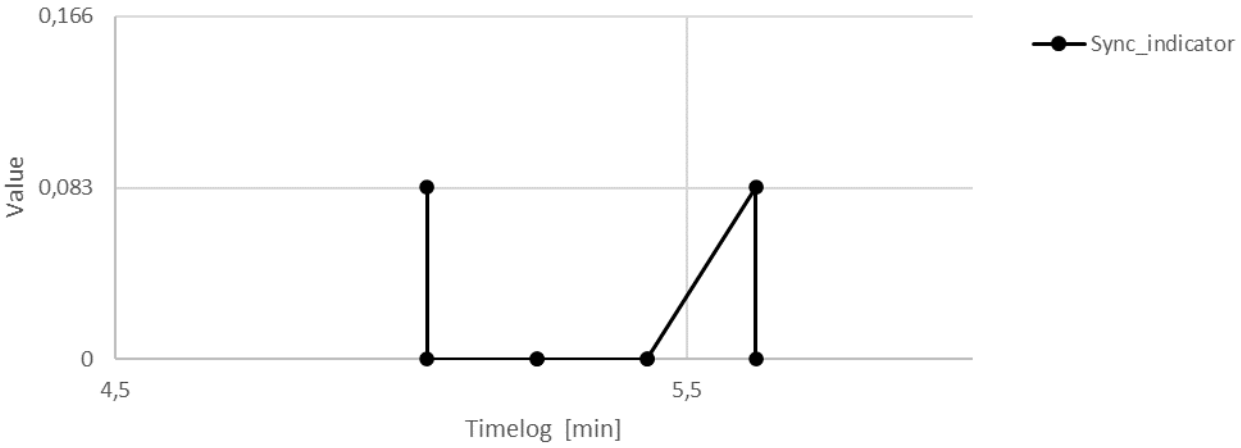


Table A.8: Indicator synchronisation check timeline

8.1.1.2. Scenario 2

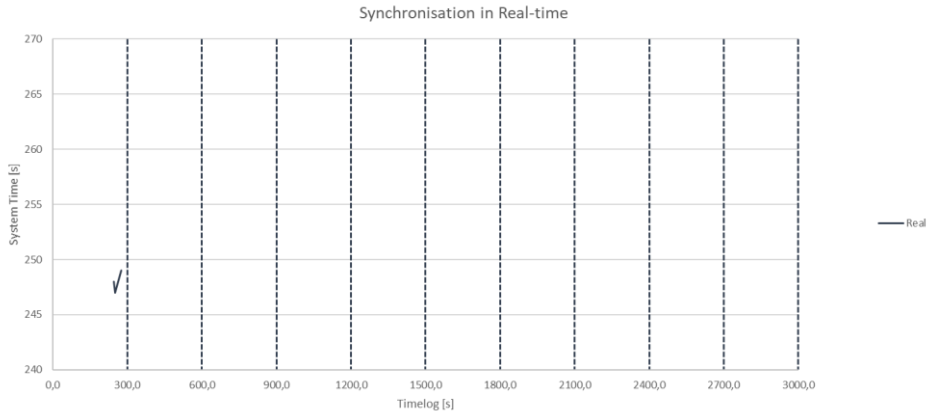


Figure A.10: (a) Scenario 2, synchronisation system time

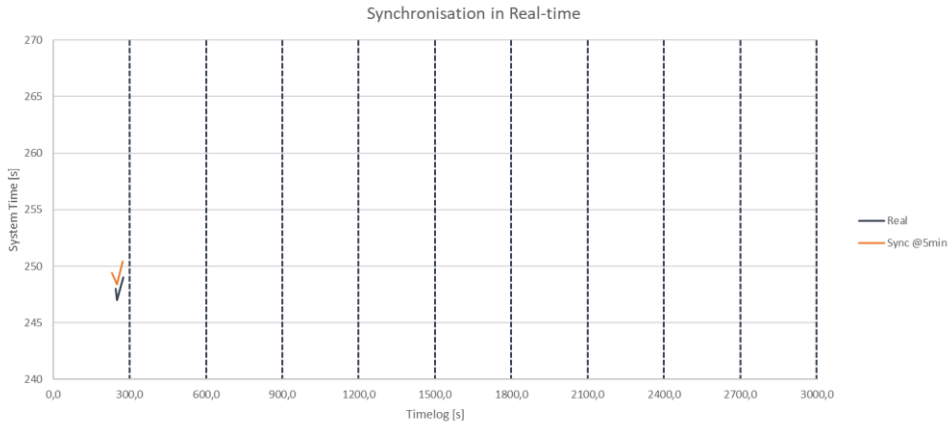


Figure A.11: (b) Scenario 2, synchronisation system time

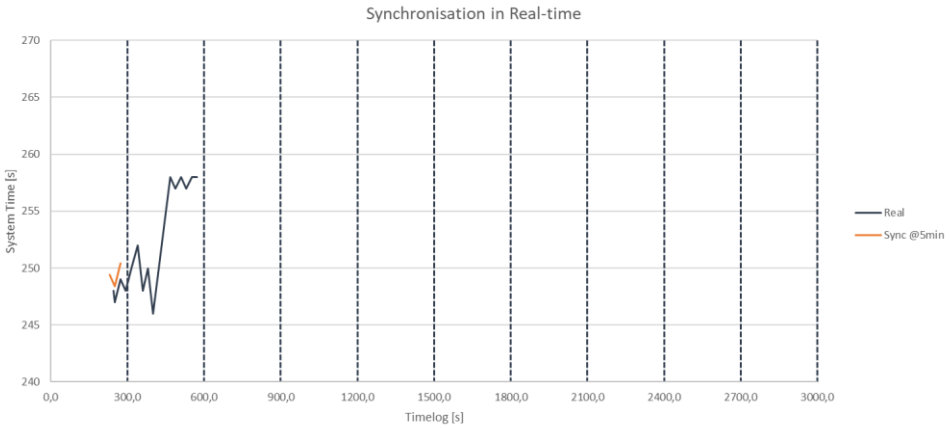


Figure A.12: (c) Scenario 2, synchronisation system time

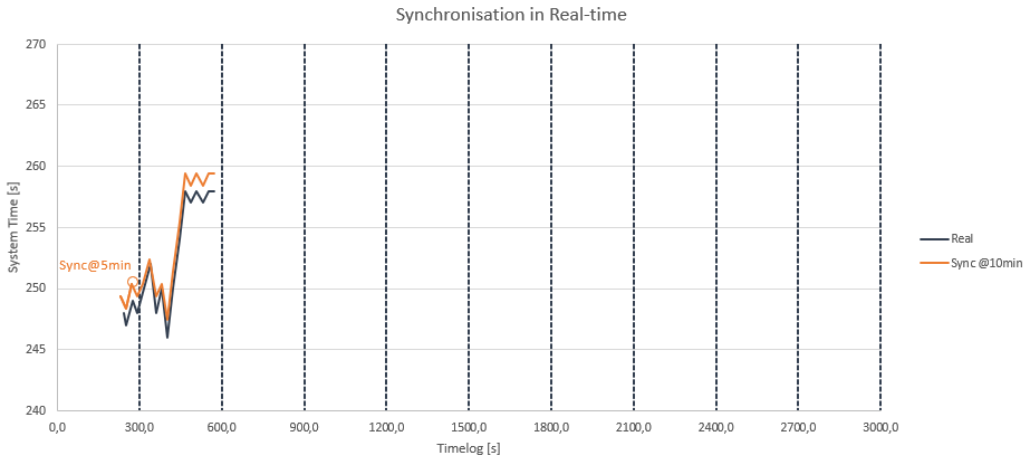


Figure A.13: (d) Scenario 2, synchronisation system time

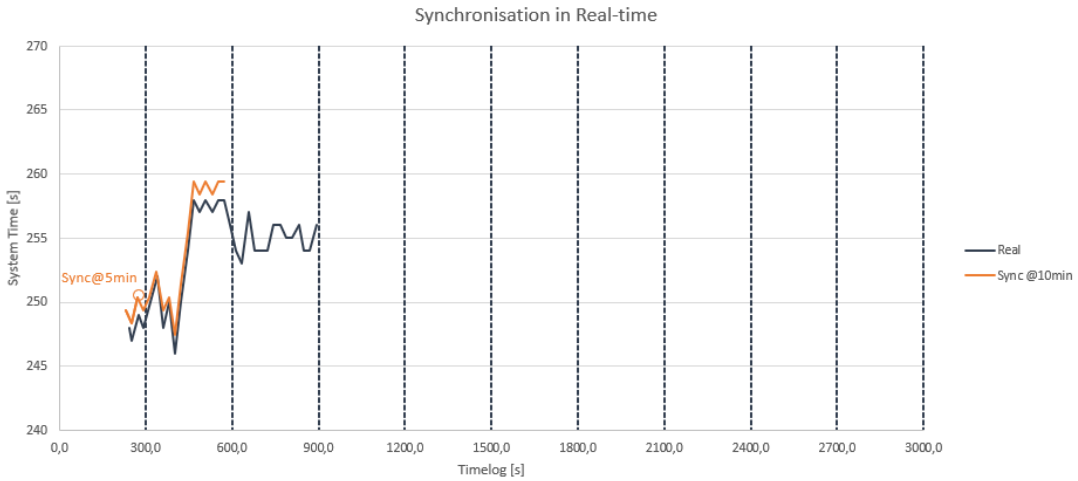


Figure A.14: (e) Scenario 2, synchronisation system time

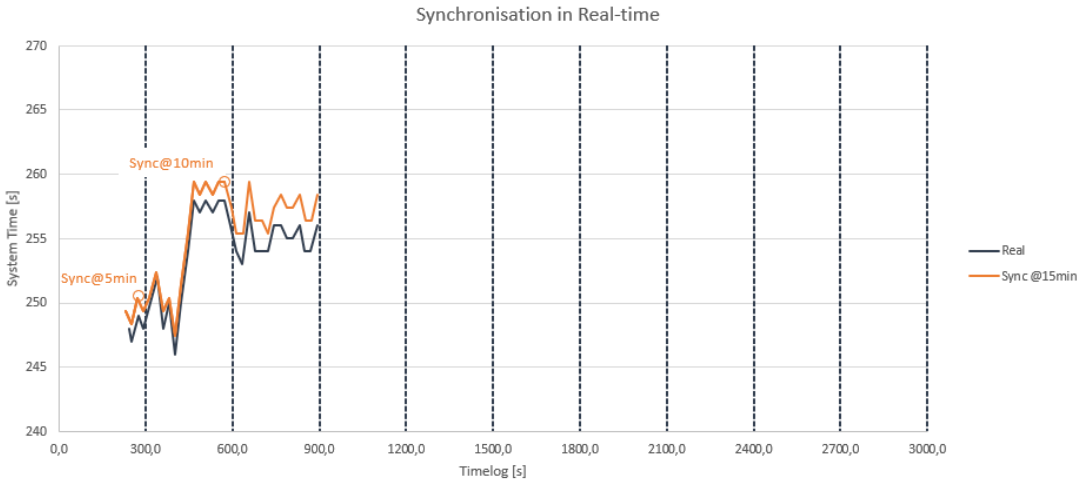


Figure A.15: (f) Scenario 2, synchronisation system time

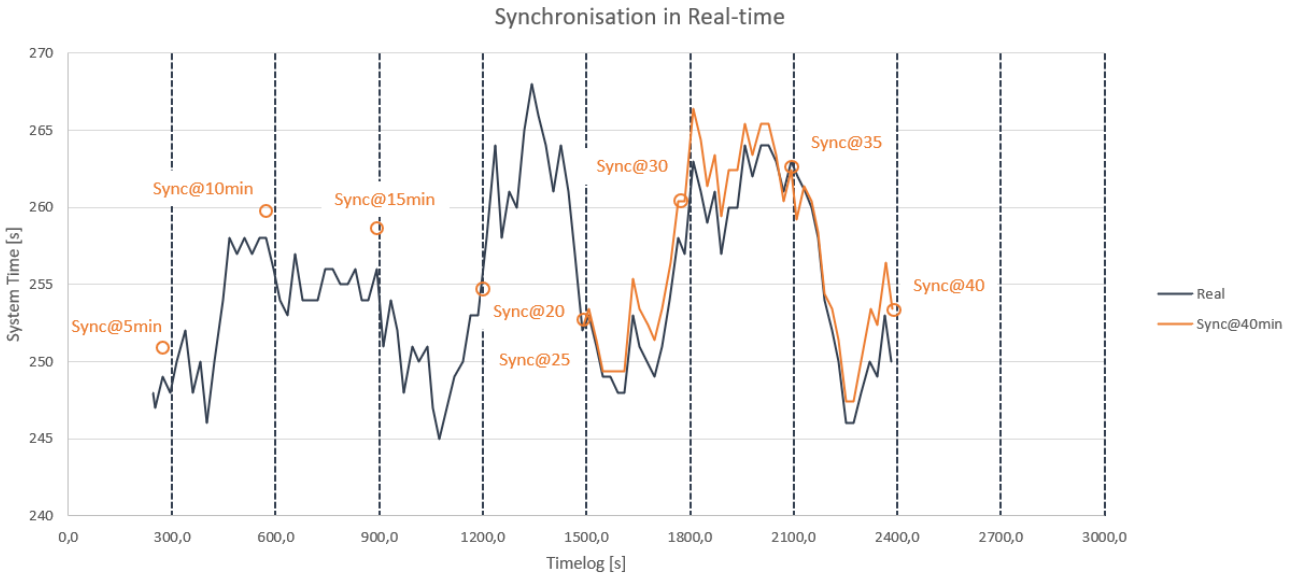


Figure A.16: (e) Scenario 2, synchronisation system time

8.1.1.3. Scenario 3

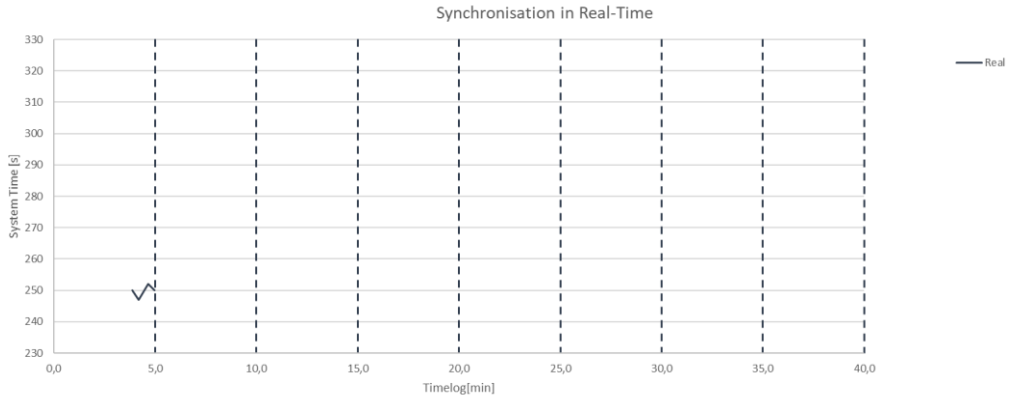


Figure A.17: (a) Scenario 3, synchronisation system time.

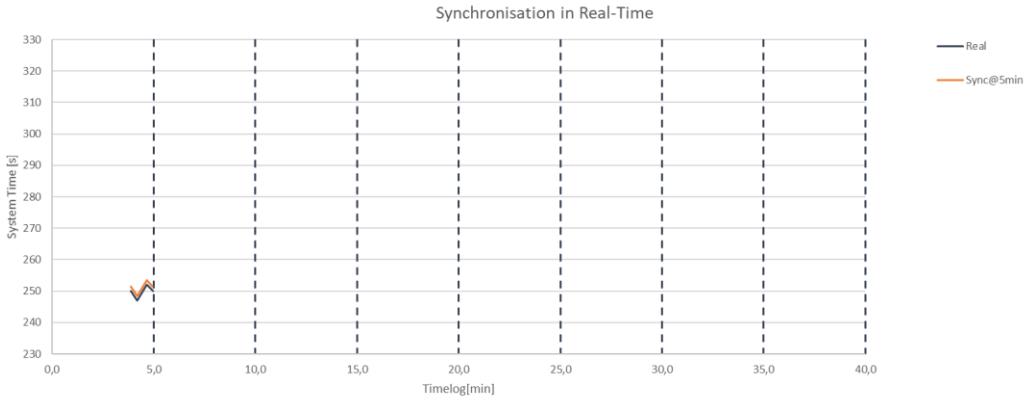


Figure A.18: (b) Scenario 3, synchronisation system time.

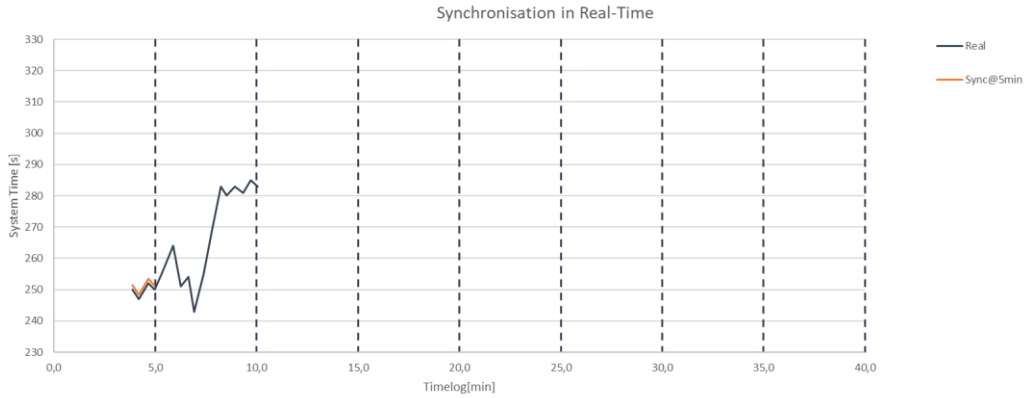


Figure A.19: (c) Scenario 3, synchronisation system time.

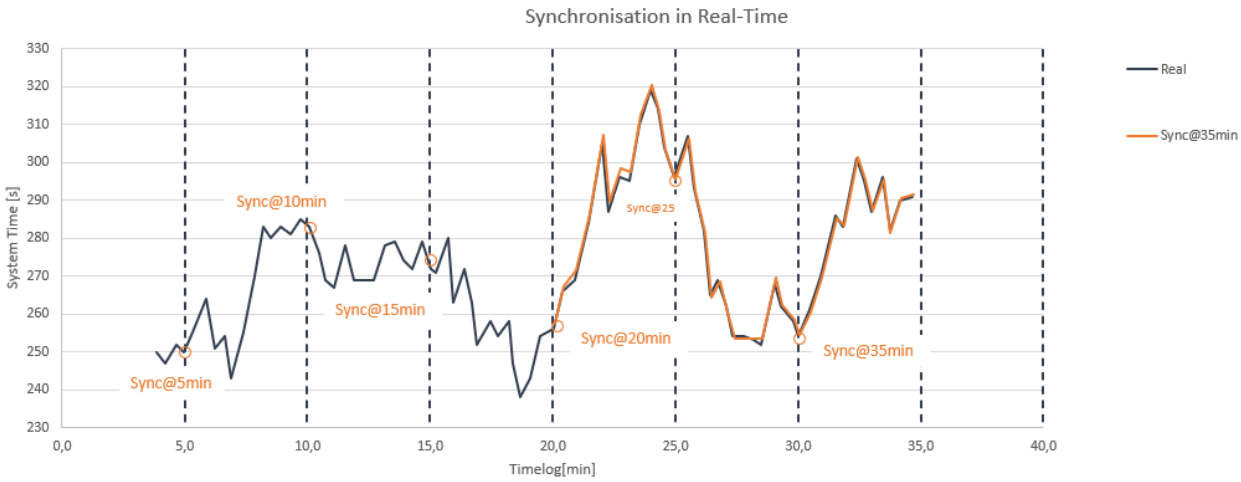


Figure A.20: (d) Scenario 3, synchronisation system time.

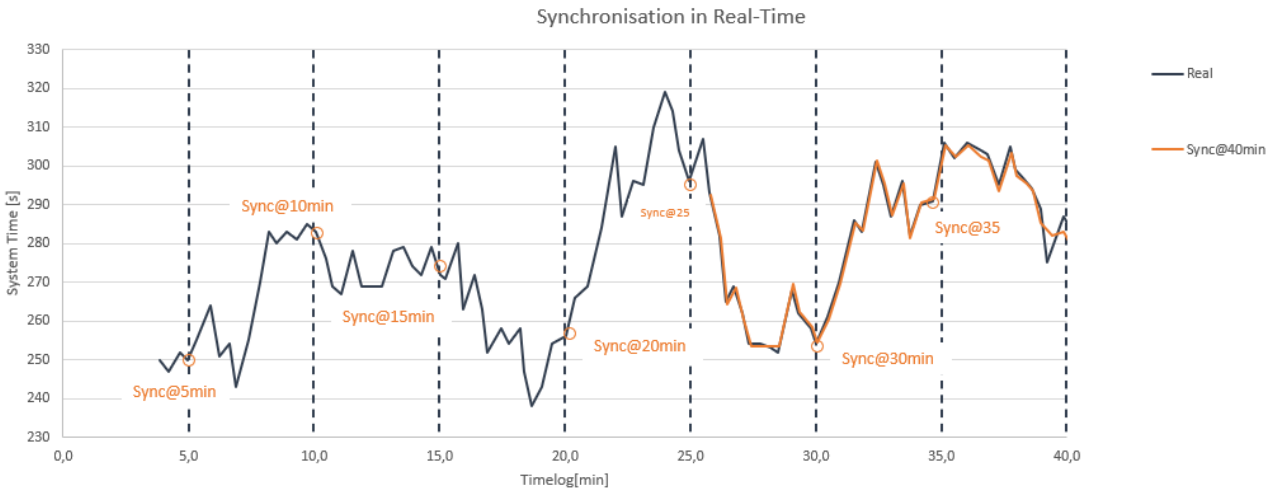


Figure A.21: (e) Scenario 3, synchronisation system time.

A.4 Case study

A.4.1 Supervisor new

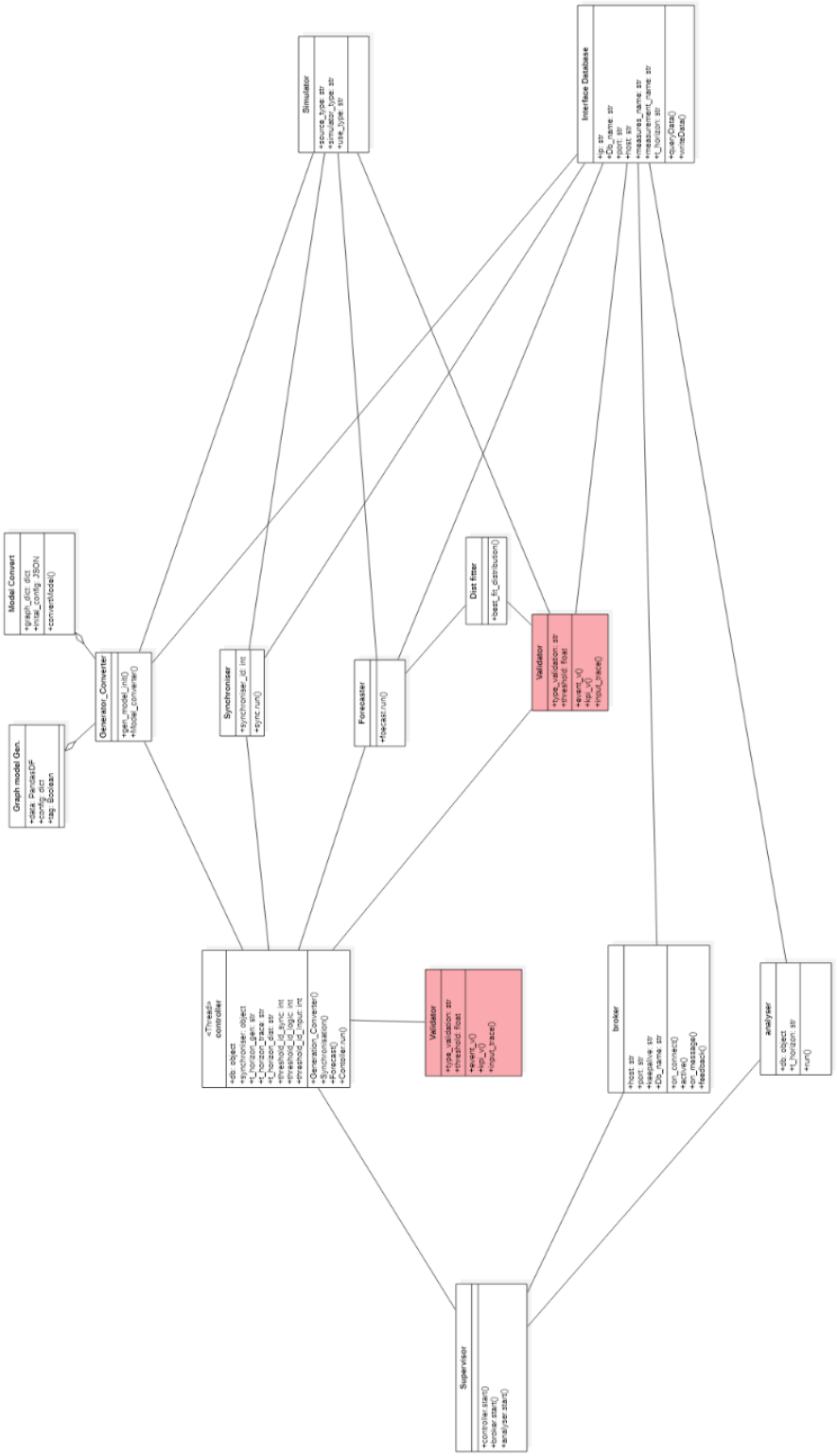


Figure A.22: Supervisor class diagram, Evaluator and Validator components highlighted