Politecnico di Torino

Master of Science in Engineering and Management (Class LM-31)
Department of Management and Production Engineering

Degree Thesis

Digital Twin Application for Dynamic Task Allocation in a Robotic System: A Flexsim-Based Case Study

Supervisor:
Professor Giulia Bruno

Candidate:
Lucia Ines Codesal
307114

A.y. 2023/2024
October 2024

Para mi familia

Acknowledgements

With the submission of this document, I bring to a close one of the most challenging and rewarding stages of my life, and I can only express my deepest gratitude to all those who have supported and accompanied me along the way.

First and foremost, I want to thank my family, who gave me their unconditional support every step of the way. To my parents, who helped me navigate the challenges of this degree, thank you for instilling in me the values that define who I am today and for supporting me in all my plans, no matter how far-fetched they may seem. To my sister Pili, for always knowing how to make me laugh and helping me relax, and to my Abuelita, who, even though still cries because I moved to Europe, never stops encouraging me to pursue my dreams.

I also want to extend my heartfelt thanks to my friends, both those I have known for years and those I have met along the way, now spread across different countries. To my childhood friends, for their unwavering support, understanding, and belief in me even when I doubted myself.

To my friends from ITBA, who made my academic journey in back in Argentina far more enjoyable. I especially appreciate Benja for always sharing notes and answering my countless questions, and Luli and Valen, with whom it was always a pleasure to do group projects and study.

To my friends at Polito, with whom I share unforgettable moments. Thank you for welcoming me into your group so warmly, even when communication was not always easy, and for your support during the thesis experiments.

And, to the friends I made during my internship at Amazon, who made my start in the professional world so much more enjoyable and fun.

Lastly, I would like to express my sincere appreciation to the institutions that have enabled me to achieve this degree. My admiration and gratitude to the Instituto Tecnológico de Buenos Aires for providing me with a world-class education through its outstanding professionals, and to the Politecnico di Torino for welcoming me so warmly and fostering an inspiring academic environment that allowed me to complete my studies at both universities.

In particular, I would like to thank Professor Giulia Bruno and Khurshid Aliev for guiding me through my thesis, supporting my work in the lab, and including me in the Digital Twin project that we were fortunate to receive a prize for, an achievement that I will always remember.

To every one of you, thank you for being part of this journey.

Contents

Abstract

In contemporary manufacturing systems the use of the Digital Twin technology provides valuable insights for enhancing real-time performance and adaptability. By creating virtual replicas of physical systems with bidirectional data flow and constant synchronization, Digital Twins enable live monitoring and optimization of processes, increasing operational efficiency, minimizing downtime and enhancing the quality of products.

In the context of dynamic scheduling, it serves as a valuable tool for identifying issues that may disrupt the planned schedule. When failures are detected, Digital Twin models can dynamically adjust the original schedule, implementing real-time response strategies to effectively mitigate disruptions.

This paper explores the utilization of a Digital Twin in optimizing the performance of a physical system within a practical application developed in the Mind4Lab Lab of the Politecnico di Torino. The study examines how the use of a Digital Twin for dynamic scheduling and reallocation of tasks can effectively identify and mitigate production problems and failures, especially when applied to robotic systems.

The research is focused on the application of the Digital Twin for processes under changing conditions, as it addresses challenges related to failures and inefficiencies within a production system. The impacts of different dynamic scheduling approaches on mitigating failures are evaluated and compared against non-dynamic scheduling, highlighting the critical advantages of implementing a Digital Twin in modern manufacturing environments. The study includes an analysis of scenarios involving different reallocation strategies to optimize the system's parameters (Throughput, Cycle Time, and Utilization).

The work is structured into three main parts. Initially, the concepts and technologies behind Digital Twins are introduced, including their application in the industry. Secondly, a detailed Case Study is presented to demonstrate the implementation of the Digital Twin in managing a process in which a system of robots experiences time failures. Finally, an analysis of the performance metrics collected from the different scenarios is provided.

The obtained results demonstrate that the use of the Digital Twin for dynamic scheduling improves the performance of the system by mitigating the effects of errors and time failures and optimizing the allocation of resources. The findings validate the effectiveness of Digital Twin technology in improving operational efficiency, thereby highlighting its potential when applied to manufacturing and production processes.

Digital Twin technology

A Digital Twin is a dynamic virtual model of a physical system, with a bidirectional data flow that enables real-time synchronization. The Digital Twin is constantly updated with data, allowing it to reflect the live state of the physical system it represents. Moreover, it is also able to send information back to the physical system, thus responding and adapting to changes in the environment. These abilities enable the Digital Twin to be used for monitoring, analysis, and optimization of operational systems.

To better understand the concept of Digital Twin, it is important to differentiate between the related concepts of Digital Model and Digital Shadow.

A **Digital Model** is a static representation of a physical system. It is characterized by the lack of information exchange between the virtual and physical systems and by the lack of influence on the virtual model due to changes in the physical system. Therefore, unlike the Digital Twin its ability to reflect the current state of the system is limited. Nevertheless, it is a useful tool for performing initial design and analysis.

A **Digital Shadow** represents a one-way flow of data from the physical system to the virtual model. While it is able to capture the past or current states of the physical system, thus updating the virtual model with it, it is not able to provide continuous updates, feedback or changes to the physical system.

In contrast, the **Digital Twin** presents the most complete model, offering bidirectional data flows: from the physical system to the virtual model and from the virtual model to the physical system. This way, both components are fully integrated, and the Digital Twin is not only an accurate current representation of the physical system, but it also provides a feedback loop to it, that enables for optimization and improvement of the performance of the system.

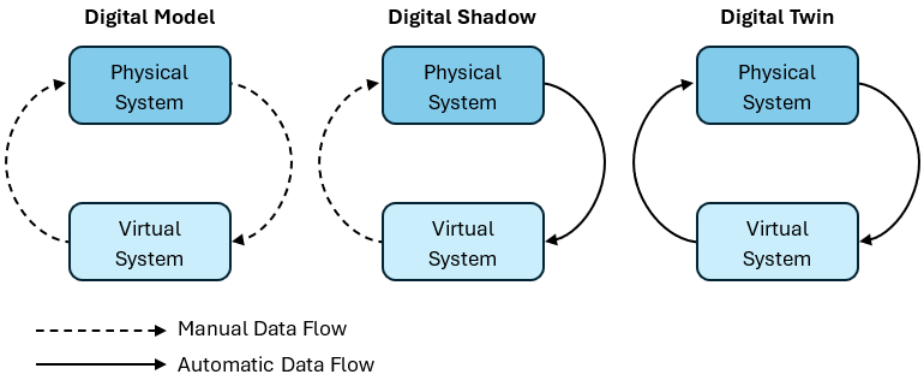

Fig 1. Digital Model, Digital Shadow and Digital Twin.

Components of the Digital Twin

The main components of the Digital Twin concept model are a) the physical system (Real Space), b) the virtual model (Virtual Space) and c) the connections of data between the two, consisting of information flows from the physical system to the virtual model and from the virtual model to the physical system.[1]

1. **The physical system** is the tangible, actual object or system in the real world, which is represented by the Digital Twin. The asset's physical characteristics, behavior, and operational conditions are the primary inputs for the Digital Twin.
2. **The virtual model** is the digital representation of the physical system, and it reflects the physical asset's properties, behavior, and lifecycle. To accurately represent the physical system in a virtual environment it integrates different data sources, including physical models, simulations, and historical data. This component is used for real-time monitoring, simulation, and analysis.
3. **The information flow** refers to the bidirectional data exchange between the physical and virtual systems. It ensures that the Digital Twin remains synchronized with the physical system by continuously updating the virtual model with real-time data and vice versa.

Moreover, for the Digital Twin to work correctly, its interaction with other elements is needed. Sensors and Data Acquisition Systems are used to collect information from the physical system. Sensors, IoT devices and other data acquisition technologies allow the transformation of physical states, performances, environments and characteristics of the physical system into readable and actionable variables that serve as inputs for the Digital Twin.

Furthermore, Simulation and Analytics Engines, that are the software tools and platforms used to analyze and simulate the data collected from the physical system, allow for the performance of analysis, predictive modelling and simulations based on the data gathered. Other tools, like Machine Learning or Artificial Intelligence (AI) can be integrated for driving predictions and feedback.

User Interface and Visualization Tools allow users to interact with the Digital Twin and visualize its data and simulations, helping in understanding complex data and facilitating decision-making. This could refer for example to 3D Models and live performance dashboards.

[1] Grieves, M., & Vickers, J. (2011). *Digital twin: Manufacturing excellence through virtual factory replication*. In *Proceedings of the Fifth Annual IEEE International Conference on Cyber Physical Systems* (pp. 1-7). IEEE.

Security protocols for the data transmitted and for the flow of information between the various components involved in the Digital Twin are required. Authentication and authorization mechanisms are highly suggested.

Finally, a Digital Twin performance evaluation is needed to ensure the correct functioning of the Digital Twin; by using evaluating metrics like accuracy, resilience and robustness of the Digital Twin model, evaluation methods and tests should be performed. [2]

Brief history of Digital Twin technology and State of Art

The concept of Digital Twin was introduced in 2002 by Michael Grieves as part of his presentation on PLM at the SME Conference, referring to it as "a digital copy of one or a set of specific devices that can abstractly represent a real device and can be used as a basis for testing under real or simulated conditions". [3]

In 2011, Michael Grieves and John Vickers would propose a renewed definition for the Digital Twin concept, emphasizing the need for a dynamic interaction between the physical and virtual systems, and defining its three necessary components previously mentioned: the physical system, the virtual model and the bidirectional flow of data between the two. This definition laid the groundwork for the Digital Twin technology.

NASA further developed the Digital Twin concept by applying the technology in the context of spacecraft and advanced aerospace vehicles. They were able to advance the concept beyond its initial theoretical framework by using Digital Twin models to simulate and test spacecraft systems and components in a virtual environment, mirroring operational conditions, helping in its design, testing and operational management.

In the following years, the Digital Twin technology has become widely applied with the rise of Industry 4.0. The integration of Digital Twin with the technologies of Industry 4.0, such as Internet of Things (IoT), Artificial intelligence (AI), Cyber-Physical Systems (CPS), Big Data and advanced simulation technologies, expanded its use to different industry sectors.

[2] Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., & Calinescu, A. (2022). Digital twins: State of the art theory and practice, challenges, and open research questions. *Journal of Industrial Information Integration, 30*, 100383.
[3] Grieves, M. (2003). *Digital twin: Manufacturing excellence through virtual factory replication*. Presented at the *Society of Manufacturing Engineers Conference*.

With these technological advancements, more detailed and accurate digital models could be created, thus facilitating the predictive maintenance, real-time monitoring and optimization of production processes.[4]

The use of IoT devices and big data analytics increased the capabilities of Digital Twins, as IoT sensors facilitated real-time data collection from physical systems and big data tools enabled the analysis and integration of the data into the virtual models. This allowed for the expansion of the Digital Twin into various areas of application, such as predictive maintenance and fault detection in production systems, improvement of manufacturing processes and smart city developments, detection of anomalies in patient care, fault detection and traffic management in smart cities, among others.[5]

More recent developments focus on integrating Digital Twins with emerging technologies such as 5G, edge computing, and advanced robotics, which promise to expand the capabilities of Digital Twins.

Digital Twin applied to Manufacturing and Production Environments

The Digital Twin technology has become a key tool for decision making in all steps of the Product Lifecycle Management, as it is used for monitoring and analyzing products during their entire life cycle. Its use allows for optimizing production processes by enhancing operational efficiency, reducing downtime, and improving product quality.[6] Moreover, a Digital Twin System can provide support to the design, reconstruction, integration, monitoring, operation and maintenance of production lines. [7]

The application of the Digital Twin in the manufacturing industry and in production environments is particularly promising, as factories are becoming increasingly more connected and reliant on data-driven decisions, and the amount and variety of information generated by productive processes is growing at a fast rate. Moreover, emerging technologies and the extended use of sensors, controllers and actuators, make possible both the acquisition of large amounts of data from production systems, and the transmission of information to them.

[4] Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics, 15*(4), 2405-2415.
[5] Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges, and open research. IEEE Access, 8, 108952-108971.
[6] Methuselah, J. (2024). Digital twin technology for smart manufacturing. Journal of Technology and Systems, 6(4), 52–65.
[7] Yan, D., Sha, W., Wang, D., Yang, J., & Zhang, S. (2022). Digital twin-driven variant design of a 3C electronic product assembly line. *Scientific Reports*, *12*, 3846.

Besides, physical systems can act upon the received signals, making use of the full capabilities of the Digital Twin technology. Fig 2. illustrates a possible flow of information within manufacturing systems and when implementing a Digital Twin.

A Digital Twin model acts on data collected by sensors as well as on information obtained from other internal or external environments. With these data, and considering the physical system logics, the Digital Twin is able to create an accurate representation of the physical system, allowing for predictions to be made on it, and ultimately enabling autonomous real-time decision-making. By using actuators, the decisions transmitted are then used to modify the physical environments, making use of the full capabilities of the Digital Twin.

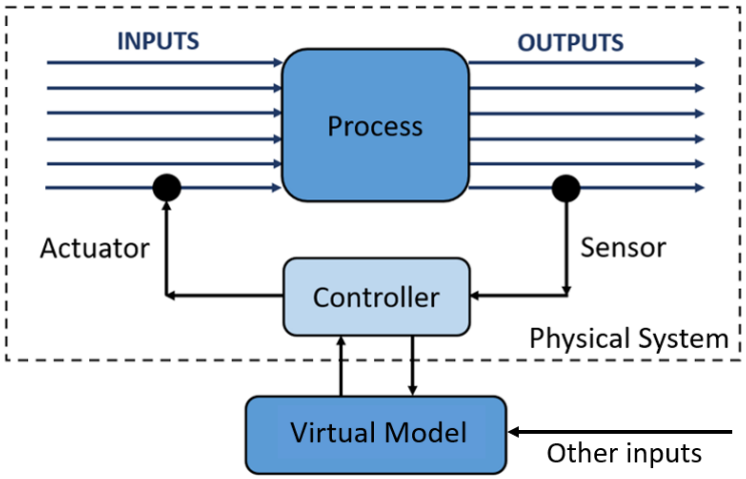

Fig 2. Digital Twin interaction with Manufacturing and Production Environments.

In the context of the Digital Twin technology, sensors allow for the collection of live data from physical systems, that then serve as inputs for the virtual model of the Digital Twin; they detect physical or chemical magnitudes and transforms them into electrical variables. Different types of sensors are commonly used in the manufacturing industry depending on the process performed and the measured variable, such as temperature, humidity, proximity and force sensors, among others.

Controllers, such as PLCs and other servers, collect the data gathered by the different sensors and send the information to the virtual model of the Digital Twin. Moreover, as the Digital Twin can analyze the data collected and drive feedback into the system, it is also possible to manage and automate physical system processes based on the data received from the model, by commanding specific actions on the actuators.

Finally, actuators take the signals received from the controller and upon it perform physical adjustments to the process, acting according to the optimization suggestions made by the

Digital Twin. Actuators in production environments can be of different types, such as electric, pneumatic, hydraulic, thermal, among others.

As the technologies used for retrieving and transmitting data evolve, allowing for more parameters to be measured and controlled, so do the functionalities and applications of the Digital Twin. Additionally, the integration of robotics and other automation tools in factories is elevating the complexity of manufacturing systems, as they allow for a broader range of tasks to be performed.

Some of the main applications of Digital Twin within manufacturing and production systems include product development, optimization of production, predictive maintenance, quality assurance, training of operators, energy management and supply chain and logistics.

Related to product development, Digital Twins facilitate the creation of virtual prototyping for the optimization of product designs. The simulation and testing of products in a virtual environment reduces costs and time and allows for the creation of products with enhanced qualities.

For production optimization, the use of Digital Twin systems allows for the creation of process simulations and its real time monitoring. This helps identify possible bottlenecks and mitigate them, optimize production lines, and subsequently, improve the efficiency of the overall process. Moreover, the Digital Twin technology allows for the identification of failures and errors within the production process and for its mitigation and resolution.

Another great application for the Digital Twin is the predictive maintenance of production systems. By using the technology, the condition of equipment can be monitored and its failure predicted and mitigated, thus minimizing downtown of machinery and extending its useful life.

Digital Twins for quality assurance help ensure that products are consistent with the required quality, detect deviations from it and implement corrective measures.

Related to the training of operators, it is used for their practice and development of skills in a riskless environment. It can also be used for training security protocols and how to handle unexpected issues that may arise.

For energy management, simulations related to energy usage can be done, helping identify opportunities for energy savings and testing and implementing energy efficient and cost reduction practices.

Finally, in the supply chain and logistics area, Digital Twins can help optimize inventory, improve storage, retrieval and transportation processes, and monitor the logistics and supply chain sector in general.

Furthermore, Digital Twins can be applied to different levels within the production process, depending on the system to be modeled and the amount of detail needed. At component level, Digital Twins are used when the focus is on a single critical component within the process. Asset level Digital Twins are used for modeling single assets within a production line, while process level Digital Twins aim at optimizing processes like design, development and production, and system level Digital Twins monitor and potentially improve entire production line system. Moreover, product level Digital Twins helps monitor a single product in real-time as used by real customers or end-users. [8]

In particular, implementing a Digital Twin is highly beneficial when physical prototyping involves high costs, requires resources and is time-consuming, when it involves extreme testing conditions, when real-time monitoring is essential, and for products or products lifecycles with multiple parameters which could be optimized jointly. [9]

Digital Twins for Industrial Robotics

The Digital Twin technology in robotics has come a long way in terms of domains of implementation and possible applications. Current areas of application for Digital Twins for robotics include space robotics, medical and rehabilitation robotics, soft robotics, human-robot interaction and industrial robotics.

For the context of this thesis, the last one is of interest. Robots have become an important element in the automation of industrial manufacturing systems, as they enhance task performance by reducing both the time required to complete tasks and the operational quality of repetitive processes. This results in faster production cycles and greater overall efficiency. Moreover, by handling repetitive and hazardous tasks, the utilization of robots frees up human resources, allowing them to engage in more complex and creative activities, thus driving innovation and growth within the industry.

---

[8] Radiant Digital. (n.d.). *Digital twin: Converging the virtual and physical worlds to accelerate transformational innovation*. Radiant Digital. https://www.radiant.digital/digital-twin-converging-the-virtual-and-physical-worlds-to-accelerate-transformational-innovation/

[9] Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., & Calinescu, A. (2022). Digital twins: State of the art theory and practice, challenges, and open research questions. *Journal of Industrial Information Integration, 30*, 100383.

Current needs require robots to adapt to real-time situations, to be able to address issues that arise within production, such as fluctuations in demand or unexpected disruptions. While robots can develop effective control strategies through interactive learning in simulated environments, inconsistencies between the simulated environment and reality can arise, posing a challenge to the performance of the robots. In this context, the Digital Twin technology serves as a powerful tool for monitoring and optimizing robotic systems.

Digital Twin systems enable the creation of detailed models and simulations of robotic environments, providing valuable insights into robot performance. This facilitates the optimization of key areas, such as navigation planning, perception analyses, and decision-making, while allowing for testing and validation to be performed in a virtual setting. Additionally, Digital Twin models can be used for real-time remote control and monitoring of robotic systems, allowing for automatic responses to changes in parameters. This offers immediate real-time insights into the performance of the system and enables proactive adjustment to be made swiftly, ensuring smoother workflows, reducing downtime and improving the overall performance and efficiency of robotic systems.

Furthermore, the Digital Twin enhances autonomous learning and decision-making capabilities for robots. By integrating algorithms, such as deep learning and reinforcement learning, with the Digital Twin framework, robots can undergo effective training and optimization, leading to improved performance and adaptability.

Within industrial robotics, Digital Twins can be used for monitoring robot states, planning assembly and disassembly sequences, learning grasping techniques, navigating environments, generating collision-free paths, and facilitating human-robot interaction, among other applications. Recent trends include work related to robot work-cell simulations, Digital Twin-aided plant maintenance, Digital Twin-aided AI implementations, industrial cloud and edge robotics cloud and blockchain in industrial robotic Digital Twins.

In practice, Digital Twin systems applied to industrial robotics are increasingly being implemented to reduce the costs and risks associated with production processes.[10]

[10] Zhang, X., Wu, B., Zhang, X., Duan, J., Wan, C., & Hu, Y. (2023). An effective MBSE approach for constructing industrial robot digital twin system. *Robotics and Computer-Integrated Manufacturing, 80*, Article 102455.

Digital Twin for Dynamic Scheduling and Reallocation Processes

Production scheduling plays a central role in the manufacturing process, as it directly influences the efficiency of production systems. Challenges such as unforeseen events, unexpected disturbances, and information gaps often affect the quality of the scheduling, for which traditional scheduling methods lack effectiveness in mitigating.

In traditional scheduling the parameters used, such as the processing time of a machine, are estimated based on statistics regarding the production process and taken as constants for calculating the schedule to be followed. In reality, these parameters depend on several factors and are not static; a significant difference between the values estimated and reality can pose a real threat to the planned schedule.

To address this issue, predictive scheduling provides a solution by using data analytics and forecasting tools to enhance the accuracy of the scheduling process. By integrating historical and real-time data and predictive models it anticipates potential disruptions, thus developing an improved schedule. Nonetheless, it still involves creating the schedule in advance, which can introduce some rigidity when handling unforeseen events.

Dynamic scheduling is an evolution of predictive scheduling that allows for the creation of both a baseline schedule to be followed and a strategy on how to respond to unexpected events in real time, allowing for real-time adjustments to be done to the proposed schedule. It continuously updates and modifies the schedule as new information becomes available, with the aim of reducing downtimes, improving responsiveness and adaptiveness of the schedule and hence improving the efficiency of the overall process.

In the area of dynamic scheduling the Digital Twin technology is a valuable tool for detecting problems that could pose a significant change in the planned schedule of a production system, as it enables real-time monitoring and facilitates optimization improvements within the process. Upon detection of failures, Digital Twin models could activate mechanisms to adapt the original baseline schedule to mitigate the failures; it allows for the creation of a strategy stating how to respond to events in real time.

Furthermore, dynamic scheduling could allow for the interaction between different machines when an unexpected problem compromises the planned schedule. Mitigating strategies could involve the reallocation of items to be produced or processed between different machines working in parallel or within a system of machines.

The implementation of Digital Twins for dynamic scheduling issues can have two different approaches based on the way they operate when an issue occurs: reactive or preventive.

In the reactive situation, upon the occurrence of an issue, the new information is fed back to the Digital Twin model for recalculation and re-execution of the program. It involves continuous recalculation of outcomes based on new information taken.

With the preventive approach, the need for constant rescheduling is minimized. This is done either by minimizing the adjustments needed in each rescheduling situation, and therefore reducing the organizational challenges associated with each change in the schedule, or by establishing a certain threshold for the foreseen deviations on the planned schedule and activating the rescheduling process only when the threshold is met. This threshold limit then represents the critical point for when measures need to be taken in order to comply with a specific timing taking into account normal deviations in the production.

Software and Tools used in the Application of Digital Twin - Flexsim software

FlexSim was selected as the software for the Digital Twin Application as it allows for both good user interface and visualization tools and advanced simulation and analytics. It not only enables the real-time virtual representation of processes, but it also allows for the generation of logics within the program. These logics can then dictate the behavior of elements within the physical system, thanks to Flexsim's Emulation Tool. By using Flexsim one can create a Digital Twin within the software and without the need for additional programs.

With respect to the user interface and visualization tools, Flexsim's 3D Model can offer a live representation of the physical system for the user to have a real-time notion of what is happening in its physical counterpart. Moreover, Flexsim's dashboards can provide a live display of the different measurable parameters and variables of interest of the system.

Simulations and analysis can be also done within the Flexsim environment, as it allows for logic creation both within its 3D Model and the Process flow. Flexsim's Emulation Tool allows for the connection of the simulation with PLCs and other servers of the physical system, allowing for the bidirectional data flow necessary for the Digital Twin application.

In short, FlexSim offers a complete suite of tools and functionalities to model and analyze complex systems across various industries, enabling in-depth analysis, optimization, and experimentation within a virtual environment[11], which makes it a great tool for the development of a Digital Twin application.

Flexsim's main uses, tools and functionalities

FlexSim is a simulation software with high versatility and a user-friendly interface. It allows users to construct detailed 3D models of systems, applying them to various applications across industries, including manufacturing, logistics, healthcare, and service operations. FlexSim's scalability allows it to adapt to the needs of its users, as it can accommodate models of different complexities and scales.

Within its functionalities Flexsim allows users to create 3D Models, enabling the creation of functional visual representation of real processes. Moreover, Process Flow models can also be

[11] Flexsim. (n.d.). Flexsim: A powerful simulation modeling software. Flexsim. Retrieved August 14, 2024, from https://www.flexsim.com/flexsim/

generated, for users to build the model's logic. Both modelling interfaces can then be connected to ensure the correct functioning of the simulation, and dashboards and statistical analysis can be generated. It also provides an Emulation tool that allows for real-time connections with PLCs or other servers.

Flexsim's Library provides a wide range of pre-built objects and resources to optimize simulation modeling and analysis. Users can simply drag and drop elements into their simulation models, combining them to best represent the system, and thus reducing modeling time and effort. These objects can then be customized to match specific requirements.

Moreover, its Toolbox includes tools for model creation, object manipulation, data analysis, visualization, experimentation, and optimization, providing users with a wide range of modeling tools and features to create and customize simulation models according to their specific requirements.

FlexScript and code customization

FlexSim runs on FlexScript as its scripting language. While it has functions made specifically for the FlexSim simulation environment, including concepts such as variables, loops, conditionals, functions and data types, it shares syntactical similarities with C programming language.

FlexScript allows for the writing of custom scripts inside objects within the 3D Model or on its Process Flow. These features allow users to interact with simulation objects, events, and data. For this reason, FlexScript is a powerful tool for automating tasks, implementing logics, and analyzing simulation results within FlexSim.

3D modeling

Objects of the Model

One of the main advantages of Flexsim is that it allows 3D modelling, enabling the creation of very accurate virtual representations of real processes. FlexSim offers a bast variety of different 3D objects available to build a Simulation Model.

Starting with Flow items, these are the items that flow through the Simulation model, following its path through the process. They can represent different things within the Simulation, such as

material, products, customers, and orders. In the Simulation Model, Flow Items interact with Fixed Resources and Task Executers.

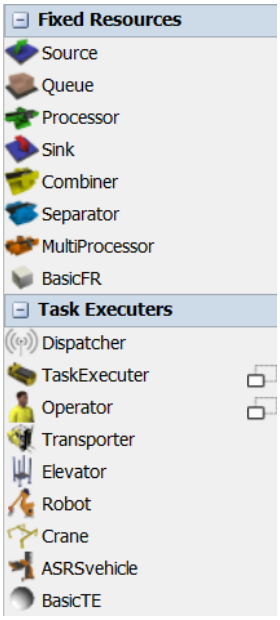

Fig 3. List of available Fixed Resources and Task Executers.

Fixed Resources are objects that remain fixed or immobile as they interact with Flow Items. Flow Items flow from one Fixed Resource to the next one downstream, continuing until reaching the end of the Simulation. Fixed resources can represent various processes within the model, the most common ones being workstations, storage centers and processing stations.

Among the fixed resources that can be used to build models, the most popular ones are the *Source*, the *Queue*, the *Processor* and the *Sink*, as they represent the starting point, waiting area, processing stage and endpoint of Flow Items within a simulated system. By combining and connecting these components in a logical sequence, users can create dynamic simulations to analyze processes, ranging from simple to more complex ones.

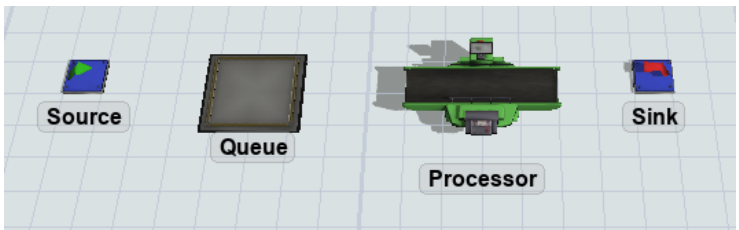

Fig 4. Common fixed resources used for 3D Modelling.

The *Source* represents the starting point of a process flow in FlexSim since it is where the Flow Items are generated. These can be introduced following a specific rate or a predefined schedule, among other options.

The *Queue* is a storage area where Flow Items are retained, waiting for the next step of the process. *Queues* in FlexSim can represent physical storage areas or virtual buffers. Items can exit the queue following different configurations, such as first-in-first-out (FIFO) or last-in-first-out (LIFO), or even some special logics. *Queues* may include additional parameters to control capacity, priorities, and routing.

The *Processor*, also known as an activity or workstation, represents a step in the process where Flow Items are transformed, processed, or worked on. *Processors* in FlexSim simulate the actions or tasks performed on items as they move through the system, such as assembly, machining, inspection, or any other type of processing required in a production line. *Processors* can have specified processing times, resource requirements, and work schedules.

The *Sink* represents the endpoint of Flow Items; entities that reach the sink are deleted from the simulation, and thus, end their journey through the system. The *Sink* can be used not only for finished products after a processing line, but also for waste or defective materials or other outputs of the process.

Finally, Task Executers are objects that interact with Flow Items and Fixed Resources as they move through the Model. As its name suggests, Task Executers can be assigned tasks and task sequences, such as transporting Flow Items, setting up or operating machines, acting as a shared resource for processing stations, among other simulation tasks. They represent workers, AGVs, vehicles or transportation machineries and more.

Element Connection and Port Properties

The connection between the different elements in a Simulation is done through ports. Elements in Flexsim allow for connection through two ways: input-output, and center port connections.

As the names suggest, an input port of an element is where entities enter the object, while an output port is where they exit it. Input/output port connections are generally used to link two fixed resources, allowing for flow items exchange and enabling their progression through the process. They are represented by a small triangle or arrow at the top of the 3D Model elements, as can be seen in Fig 5. The direction of the arrow indicates the direction of the process flow, with input ports arrows pointing towards the object and output ports arrows away from the object.

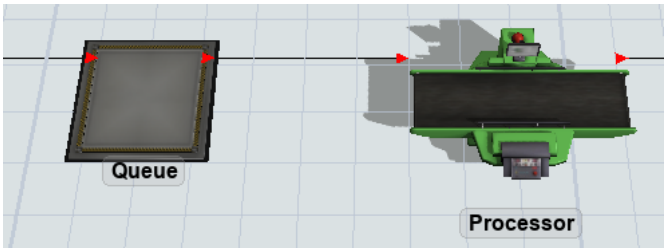

Fig 5. Input/Output connection in Flexsim.

Center ports on the other hand, create an abstract reference point between objects, and are used to link task executers to fixed resources or any objects that reference each other.

3D Model elements can be connected to multiple other elements, allowing for complex process modelling. Whenever a connection is made between two objects a rank is assigned to it. This ranking allows for easy referencing of these ports for various purposes, such as for creating conditional logics for items exiting an object.

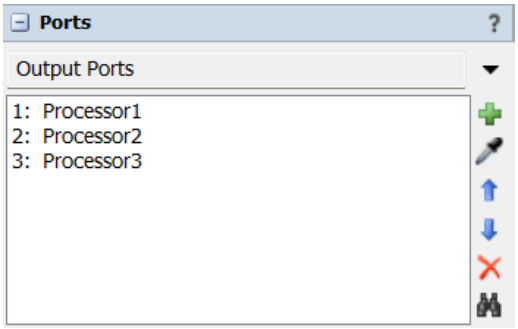

Fig 6. Output ports of a Queue element connected to various Processors.

Close and Open Port

Ports can be open or closed, and thus be available to receive or send items, or not. During a simulation run, open ports take the color green, while close ports the color red. Moreover, ports can be opened and closed as needed by using certain *Triggers* and *Events*.

The State of both input and output ports can managed by four different instructions: Close, Stop, Open and Resume.

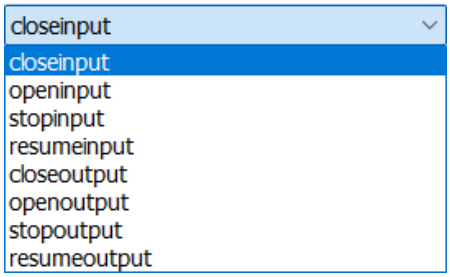

Fig 7. Instructions for managing the state of the ports.

The Close input or Close output instructions block the input or output of the object respectively. This way, ports are blocked when this instruction is called.

The Open input or Open output instructions act as the counterpart of the Close port instructions and unlocks the ports of the object after they have been previously closed with the Close port instructions.

The Stop input or Stop output instructions operate like the Close port commands, but it also keeps track of the consecutive stop calls on the object and will only open the port after all stop calls have been resumed with its respective Resume input or Resume output instruction.

Finally, the Resume input or Resume output commands opens the input or output of the object just like the Open port command does, but because they keeps track of previous Stop port calls on the object, they will only open the respective port after all stops have been resumed.

In summary, if the Stop port commands are used, one will need as many Resume port instructions as times one has called the respective Stop port command. If the Close port commands are used however, one will need only one Open port instruction to open the port. When modelling the opening and closing of ports it is advised to use the Stop port and Resume port commands, rather than the Close port and Open port instructions.

Properties of the objects and customization

Flexsim offers an extensive range of features for personalizing objects within the simulation, to best represent the physical system, both in appearance and functionality.

Within the 3D Modelling features FlexSim provides tools that allow users to customize the appearance of objects, modifying parameters such as texture, color, shape, among others. It is also possible to import custom 3D objects. Additionally, it is possible to add custom labels and graphics to objects within the 3D Model, providing additional information on it.

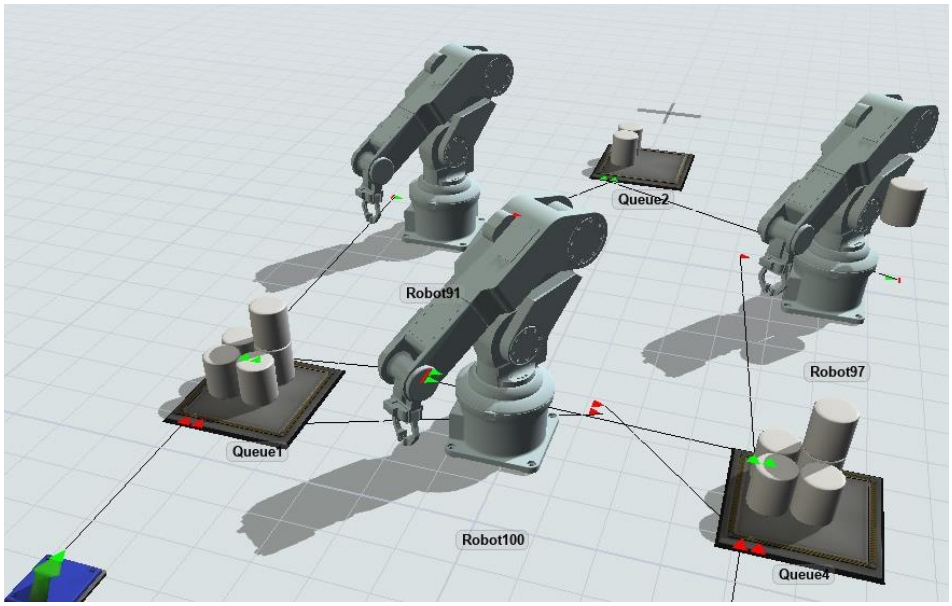

Fig 8. Object customization done for the Application of Digital Twin.

Related to its functionality, Flexsim also allows for the customization of the object's properties and attributes. Parameters such as processing time, capacity, size, weight, and resource requirements can be defined.

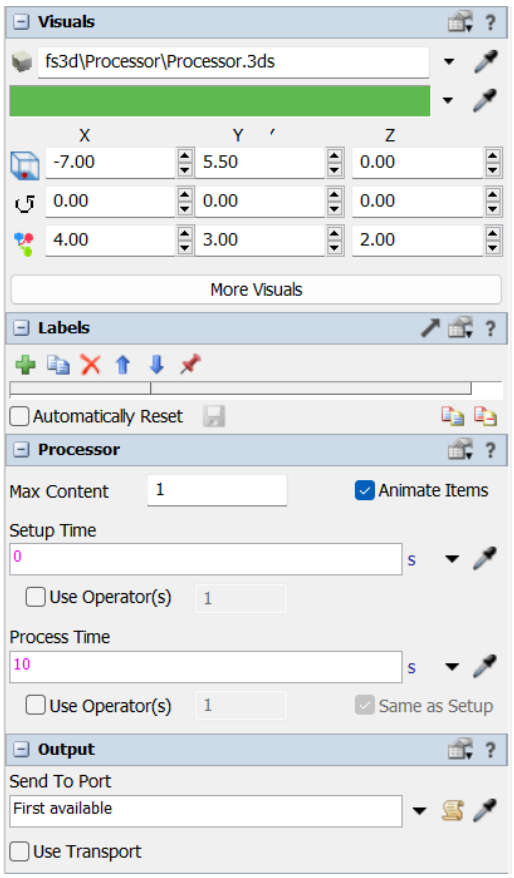

Fig 9. Predetermined parameter values of a Processor object.

Process Flow Modeling

Flexsim also allows for Process Flow modelling. With pre-built activity blocks, it is possible to build the logic of the simulation with a flowchart approach. Dynamic process flows, decision trees and routing rules, can be created, enabling the creation of all types of simulation logics, from simple to more complex ones.

Moreover, building the logics of the simulation throughout Process Flow modelling eases the generation of logics, allows for the centralization of the logics in one place and facilitates scalability and debugging as the model changes and progresses.

Elements of the Process Flow

To build logics in Flexsim a list of predefined activities is provided. Activities can be combined in different orders as needed, allowing tokens to circulate through the process, activating different logics as they do so. Among the most common activity blocks, it is worth mentioning the *Source*, the *Decide*, the *Delay*, the *Wait for Event* and the *Sink*.

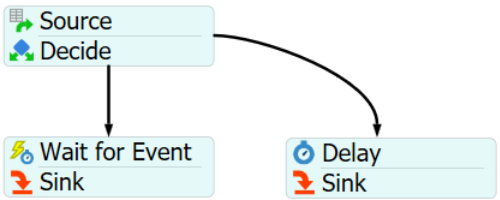

Fig 10. Common elements of the Process Flow.

The *Source* element marks the starting point of the process flow, where tokens are generated and introduced into the system. There are different types of sources that can be used depending on whether tokens arrive at a specified rate, according to a schedule, based on predefined *Events* or other conditions.

The *Decide* element allows for different scenarios to take place depending on if certain defined conditions are met or not; they are generally connected to two or more activities and depending on if the condition is met or not tokens follow its way to one or other subsequent activity. They can be used to control the flow of tokens or trigger specific actions, allowing for different scenarios to take place conditionally according to the current state of the simulation. The conditions inside the *Decide* block can be of various types, such as based on a condition, a case, a time, a probability percentage, following a statistical distribution, among others.

The *Wait for Event* element momentarily stops the execution of the Process Flow until a specified Event takes place. Once a token arrives to this activity, it is held while listening for a specific event to occur in the Simulation, both in the 3D Model and Process Flow. Tokens can be released to the next activity in the Process Flow only when the *Event* occurs.

The *Event* to wait for can be related to entity arrivals, resource availability or other user-defined events. The *Wait for Event* block is used to synchronize activities, coordinate actions, or manage timing within the simulation, ensuring that actions occur only when specific events or conditions are met and not before.

The *Delay* block introduces a waiting time before the token proceeds to the next activity. It is used when there is a specific fixed time between the occurrence of events, but it can also be used as a security buffer time to ensure the correct functioning of the model. The delay time can be fixed or follow a statistical distribution.

Like the *Sink* in the 3D Model, the *Sink* element in the Process Flow is the endpoint of tokens. Consequently, Tokens that reach the Sink are deleted from the Process Flow.

Moreover, FlexSim offers other specific blocks that allow for the correct functioning of the model, like the Merge and Split element, that combines or separates flow paths for entities within the process flow. Also, Flexsim has a Custom Code element, that allows for the execution of custom code defined by the user, for more specific applications.

Variable, Set and Get Variable

Flexsim's Process Flow allows for the creation of internal variables. These variables can be connected to the simulation by the *Set Variable* and the *Get Variable* elements, which allows assigning and retrieving values to and from variables respectively.

The *Set Variable* element modifies the value of variables within the Process Flow. Among other possible uses, the *Set Variable* block allows for fixed values, calculations or state updates based on predefined logics, to be assigned to variables. *Set Variables* are used to track and manipulate data, parameters, or states within the simulation.

Conversely, The *Get Variable* element retrieves the value of variables from the simulation allowing its use within the Process Flow. *Get Variables* are used for getting attributes or properties of entities, objects, or resources of the simulation.

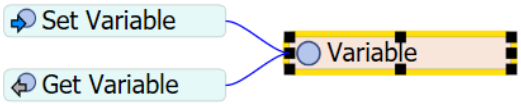

Fig 11. Variable connected to a Set Variable and a Get Variable element.

Events, States and Triggers - Connecting the 3D Model to the Process Flow

Events, States and *Triggers* enable complex interactions between different elements of the model. For this reason, they represent a key component for creating the Model's logic.

Flexsim's *Events* consist of a series of pre-programmed logics that instructs the 3D object how to interact with Flow Items. *Events* can be used to modify the logic and behavior of 3D objects, initiating, modifying or terminating processes and activities within the Model. Flexsim's *Events* represent real-life events that can occur over time, like the arrival of a customer, order or material, the transportation of products between stations, the processing of a product, or the breakdown of a machine.

For *Events* to interact with the system, Flexsim includes a feature for event listening. This way, specific *Events* can be monitored within the Simulation and responses can be defined tied to its occurrence. This capability allows the simulation to respond to real-time changes, thus allowing for the creation of dynamic and interactive simulation models.

Moreover, event listening is a way of connecting the 3D Model with the Process Flow Model: event-listening activities in the Process Flow Model can be used for listening to certain *Events* in the 3D Model. Upon the occurrence of the *Events*, the event-listening activity will create and/or release a token, that could consequently execute some action in the Process Flow. The two main Process Flow activities that are used for event listening are the *Event-Triggered Source* and the *Wait for Event* activities.

Events in a simulation model can cause a *State* change in the 3D objects, for both *Fixed Resources* and *Task Executers*, with the *State* of the object being its current condition or status. Example of *States* include processing state, idle state, down state, traveling state and utilized state, among others. *States* are relevant for statistical and data-gathering purposes.

Finally, *Triggers* are responsible for the execution of actions, and are implemented whenever an *Event* needs to occur in the Model; they are used to create responsible behaviors in the

simulation. Most 3D objects have a specific set of associated *Triggers*, that can be activated upon the execution of an *Event*.

Additionally, certain logics can be assigned to the *Triggers* allowing for the generation of actions when the *Trigger* is activated. FlexSim has a wide variety of pre-programmed logics that can be added to *Triggers*. Moreover, it also allows for the creation of custom logic using the Process Flow tool or FlexScript.

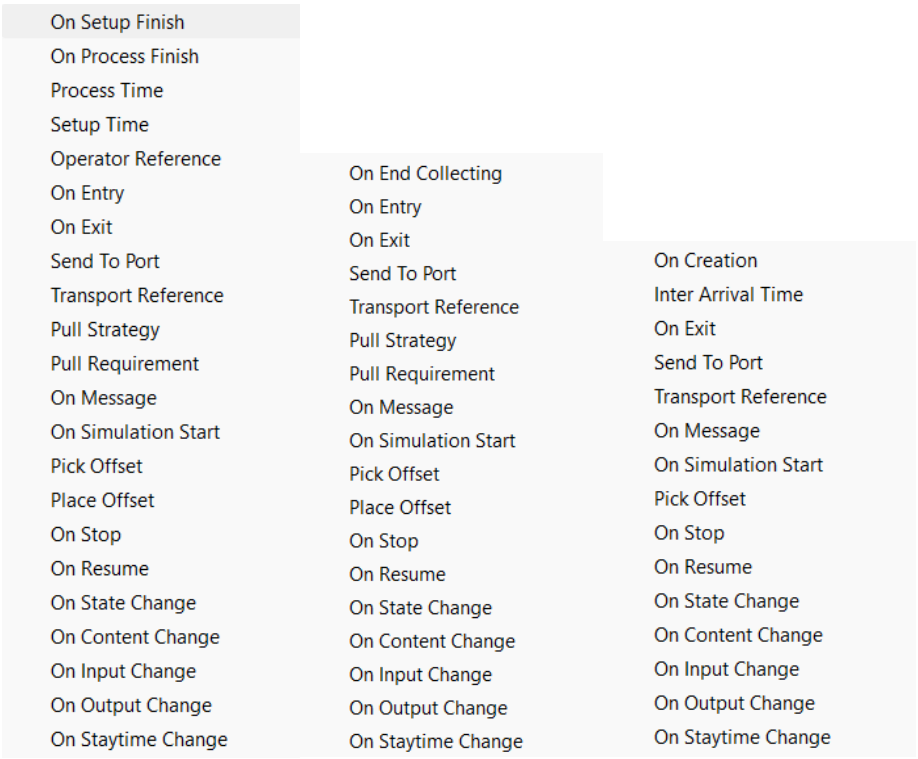

Fig 12. List of available pre-programmed Triggers and Events associated to common Fixed Resources.

Labels

Labels are another important component for building the logics of the Model. They can be assigned to *Flow Items, Tokens* or objects in the 3D Model, and store valuable information about them. Different Labels can be assigned to the same element type, allowing for differentiation among the same group of items.

Labels can be used for getting data from the Model for tracking statistics, representing changes to a *Token* or object, linking *Tokens* to 3D objects or other *Tokens*, stablishing sorting and conditional routing, filtering and restricting *Flow Items* and *Tokens,* and for enabling conditional decision making, among other uses.

Flexsim's Emulation Tool

In addition to its core functionalities, Flexsim offers an Emulation tool that facilitates the bidirectional data flow necessary for Digital Twin applications. It serves as a connection between the virtual model and the physical system, enabling communication and interaction between the two, allowing to send and receive signals, data, and commands between the model and external devices. The Emulation Tool creates a connection between Flexsim and external PLCs or other servers, and supports various protocols, including OPC UA, OPC DA and Modbus.

By establishing this connection, FlexSim enables real-time data exchange, where information from the physical system can be captured and incorporated into the simulation, and information sent by the virtual model can alter the physical system. This real-time feedback loop ensures that the virtual model remains synchronized with the physical system, allowing it to respond and act upon the data received.

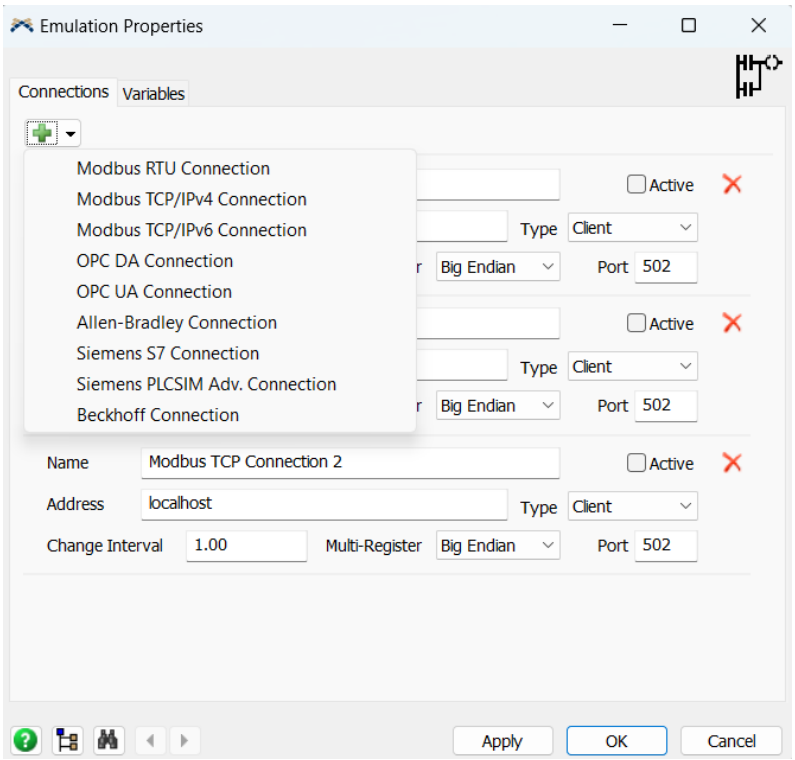

Fig 13. Creating a connection using Flexsim's Emulation Tool.

Multiple simultaneous connections can be created for one simulation, to integrate various sources of data. When a connection is not actively needed for the simulation, it is possible to deactivate it; consequently, the data would be retrieved from the simulation model instead of from the PLC.

The Emulation Tool can be accessed from Flexsim's Toolbox or can be generated within the Process Flow Model. Moreover, Variables related to the connection made through Flexsim's Emulation Tool can be accessed through its window interface or can be generated within the Process Flow Model.

Internal Emulation Variables and Internal Emulation Connection

Flexsim allows for the creation of *Internal Emulation Variables* and *Internal Emulation Connections* as part of the Emulation Tool inside the Process Flow Model. This allows for the *Variables* to be modified by elements within the Process Flow, like the previously discussed *Set Variable* block.

Flexsim allows for a wide range of Variables and Connections to be created, depending on the protocol needed. Moreover, the Variables can be *Sensor Variables* and *Control Variables*.

Sensors are the inputs of the PLC, that provide environmental data and upon which the PLC takes specific actions based on the data received. When connected to the simulation, *Sensor Variables* are the variables which send data from the Simulation and to the PLC.

Flexsim allows for *Sensor Variables* to be connected to an object within the 3D Model and be associated with a series of *Events*, depending on what kind of object it is connected to. Upon the realization of the selected *Event* a signal is sent from the Simulation and to the PLC. The PLC can then perform a task associated with the signal received.

Controls are the outputs of the PLC; the results achieved and sent to the simulation, based on the inputs it received. *Control Variables* are then the simulation's variables which receive data from the PLC, and upon which certain actions can be taken in the Model.

Like the *Sensor Variables*, each *Control Variable* can also be connected to an object of the 3D Model, and can trigger a series of actions, depending on what kind of object it is connected to. Upon receiving the right signal, the simulation executes a specific action within the model.

Finally, to ensure the functioning of the Emulation Tool it is necessary to connect the Variables to the Connection in the Process Flow, selecting also the register type and register number of the Connection.

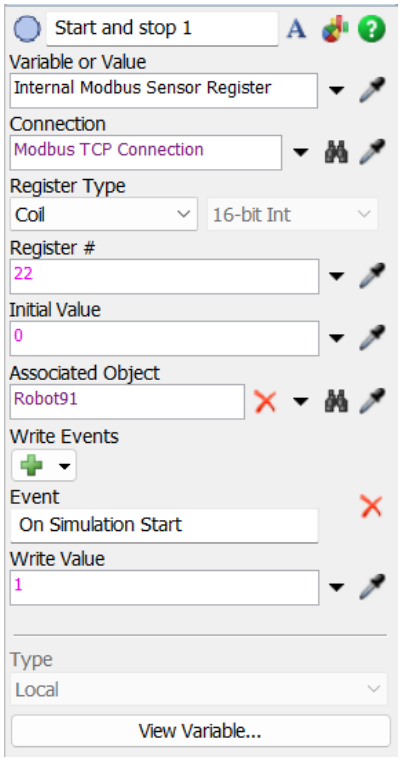

Fig 14. Sensor Variable used in the Application.

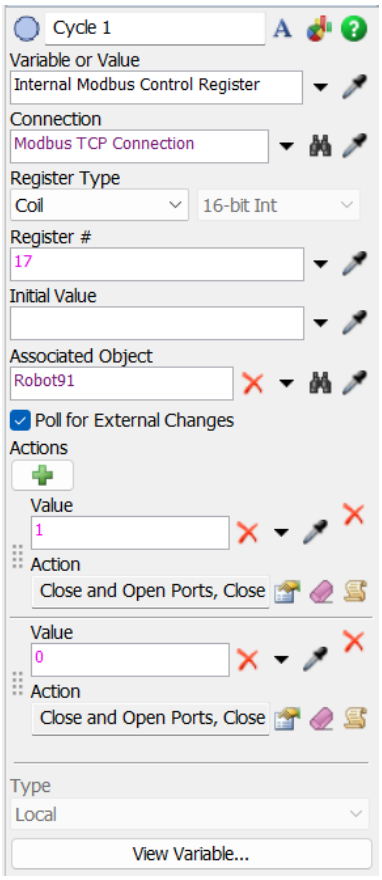

Fig 15. Control Variable used in the Application.

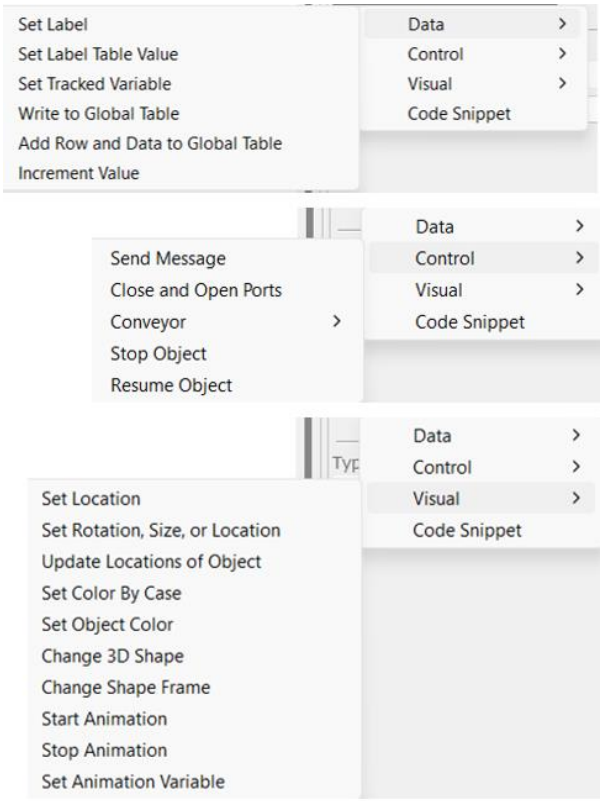

Fig 16. List of possible actions associated with a Processor.

Dashboards

Flexsim's dashboard tool is a graphical user interface and visualization tool that enable users to make use of various visualization elements like charts and graphs to create custom dashboards. Between its available elements, users can choose between time plots, histograms, Gantt charts, pie charts, bar charts, table charts, box plots, and more.

It is a particularly useful tool, as it allows for the real-time monitoring of the simulation, visualizing its main statistics, metrics and KPIs. This can be used to perform insightful analysis of the performance of the simulated system, and even take corrective actions on it, adjusting parameters, inputs, and settings.

The dashboard's graphical elements are supported on data from the simulation model, such as variables, attributes, outputs or any other custom data source and they can be integrated with Process Flow and 3D Modelling.

Fig 17. Example of Flexsim's Dashboard.

Application of Digital Twin developed in the Lab

Setting and scenarios

To illustrate how the Digital Twin technology can enhance the flexibility of a production system, by enabling real-time detection of failures and triggering a dynamic task reallocation response, the following Application is proposed.

The setting presents two collaborative robots tasked with handling items within a manufacturing environment. The task assigned to these robots is to retrieve an item from a designated pick-up point, process it, and subsequently transport it to a releasing point.

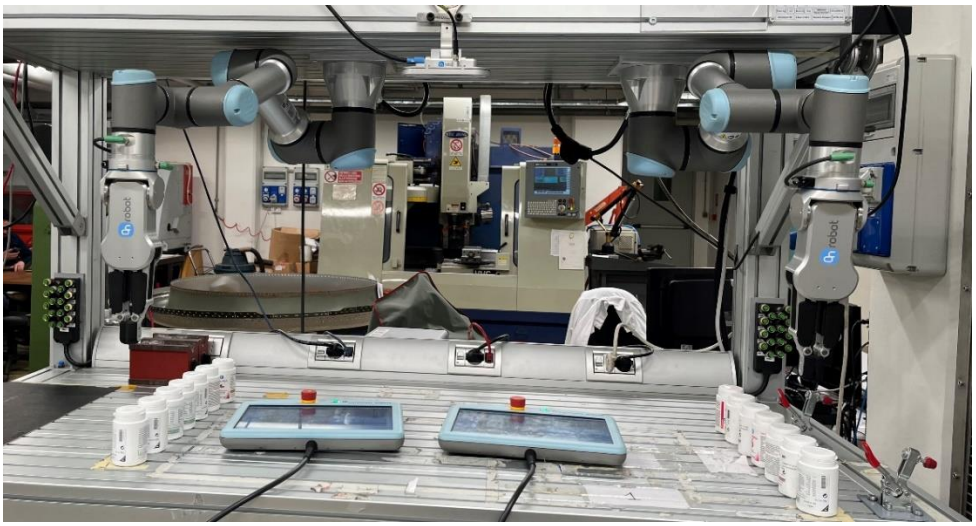

Fig 18. Set up of the Application in the Laboratory.

Seven items are assigned to each robot for its processing. In an optimal scenario, both robots, being identical, work at approximately the same speed and perform the same tasks, thus completing its respective tasks at nearly the same time.

If a time failure occurs however, and one of the two robots operates at a slower pace, maintaining the initial schedule, the slower robot would take a longer time to accomplish the seven assigned tasks.

To mitigate this problem, a Digital Twin is implemented to perform a dynamic task reallocation between the two robots. The Digital Twin can automatically detect the time failure, and subsequently reassign the remaining items to be processed by the faster robot. This can be done through two different reallocation strategies: by reallocating one item at a time, or by moving an optimal calculated batch of items at once.

For the reallocation of items, a third robot is used, thus automatizing, and controlling the movement of the items.

Fig 19. Set up of the Application in the Laboratory, with the addition of the third robot performing the dynamic task reallocation.

Four scenarios are then considered for the Application:

1. **Non-dynamic Scheduling without Failure:** This scenario represents the optimal production case where no unexpected events, such as time failures, disruptions, or slowdowns, occur. In this scenario, each of the two collaborative robots is assigned seven items to handle and the tasks are executed according to the predefined schedule.
2. **Non-dynamic Scheduling with Time Failure:** In this scenario, one of the two robots experiences a time failure, resulting in a slower execution of tasks. Despite the occurrence of this time failure, the initial task allocation and scheduling remain unchanged. Consequently, the slower robot takes a longer time to complete the seven assigned tasks, resulting in an increase in the processing time of the system.
3. **Dynamic Scheduling with Time Failure, reallocating one item at a time:** This scenario demonstrates the capability of the Digital Twin system to detect time failures in real-time and automatically initiate corrective actions. When a time failure is detected, the Digital Twin dynamically adjusts the task allocation by reassigning jobs to the faster robot. This is achieved by initially reallocating one item, upon which operation is resumed; if this was not enough to mitigate the time failure, another item is reallocated. This is iterated until the situation is corrected.
4. **Dynamic Scheduling with Time Failure, reallocating an optimal batch of items:** Like the previous scenario, the Digital Twin system detects a time failure in real-time. In this scenario however, the Digital Twin dynamically modifies the original scheduling by reassigning a calculated optimal batch of items to the faster robot at once.

Digital Twin implementation

The Digital Twin implemented in this Application comprises three main components: the Physical System, the Virtual System, and the Connections between the two.

Physical System consists of two robots performing the processing tasks and one robot performing the reallocation of items between the two. The Virtual System on the other hand, is a Simulation Model made using the software Flexsim. The connection between Physical and Virtual System is done by using the Flexsim's Emulation Tool, which allows sending and receiving signals between the robot and the simulation, in the form of digital inputs and digital outputs.

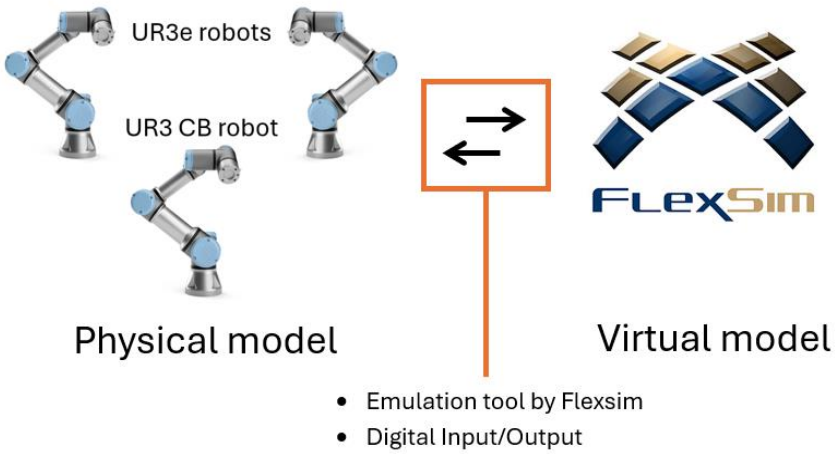

Fig 20. Digital twin Physical System, Virtual System and the connection among them.

Physical System

The physical system consists of two collaborative robots that work in parallel to perform a given task, and a third collaborative robot that performs the reallocation of items between the two, when needed.

For the two robots performing the processing tasks, the UR3e collaborative robots, manufactured by Universal Robots, were chosen.

The UR3e-series is a small collaborative robot designed for various industrial and research settings. Made primarily of plastic and aluminum, these robots are lightweight and agile, facilitating easy coordination of movement to perform tasks accurately. Moreover, thanks to its small size it can work in confined workspaces, improving the efficiency of a given production

facility. It can integrate into any production environment, as it can work side-by-side with human workers or within a separate station. [12]

Each UR3e robot is equipped with its own controller and teach pendant, facilitating their control and programming as well as the coordination and communication between the robots. The control units are interconnected via TCP/IP communication, enabling real-time data exchange and synchronization between the robots. Additionally, each control unit is equipped with digital/analog input and output ports that enable connectivity with the robot arms, the Teach Pendant, and other peripherals within the robotic cell.

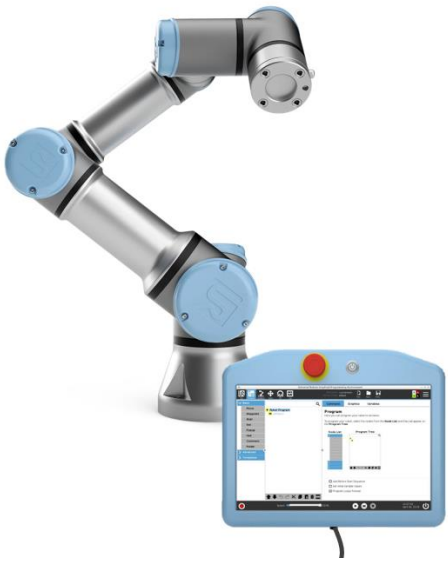

Fig 21. UR3e robotic arms with its teach pedant.

Moreover, to facilitate dynamic task allocation and reallocation of items between the two UR3e robots within the application of the digital twin, a UR3-CB-series robot is utilized. Similar to the UR3e-series robots, the UR3-CB-series robot is a collaborative robotic arm developed by Universal Robots, that offers precision and flexibility for various industrial and research applications.[13]

Additionally, for the three robots to be able to perform the pick and place tasks, a RG gripper was attached to the wrist of each of the robots.

[12] Universal Robots. (n.d.). UR3 robot. Universal Robots. Retrieved August 14, 2024, from https://www.universal-robots.com/products/ur3-robot/
[13] Universal Robots. (n.d.). CB3 series robots. Universal Robots. Retrieved August 14, 2024, from https://www.universal-robots.com/products/cb3/

Virtual System

For the Application of Digital Twin the Virtual Model was developed using the software Flexsim. It consists of several key elements that represent the various components of the physical system and their interactions. Fig 22. shows the 3D Model of the Virtual System, consisting of two *Source* elements, three *Queue* elements and three *Processor* elements.

The process starts with the *Source* elements, that generate the *Flow Items* that move through the Model: each *Source* element generates a batch of seven items at the start of the simulation. Flow elements continue the process as they enter the first two *Queue* elements. They are the storage area where the items wait to be processed by the robots.

Two *Processor* elements, representing the two UR3e robots working in parallel, are the next step of the process. They pick *Flow Items* from their respective *Queue* and place it on another final *Queue*, after the processing is done. The latter is used for storage of the items at the end of the process.

Another *Processor* is used to represent the UR3 CB robot in charge of the reallocation process. It is connected to the *Queues* of the other two *Processors*, as it redistributes *Flow Items* between them.

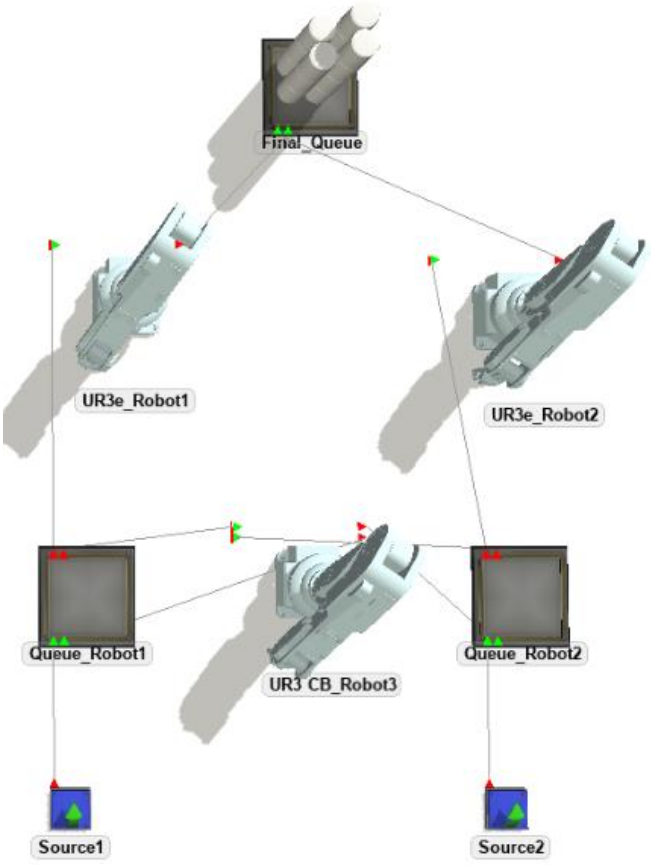

Fig 22. Virtual System of the Application.

Connections - Flexsim's Emulation Tool

For the Digital Twin, the connection between the physical system and the virtual model plays a fundamental role in enabling the necessary bidirectional flow of data. This connection allows for the real-time synchronization of the virtual model and the physical system, for the virtual model to be an accurate live representation of the physical system and for the physical system to be able to respond and adapt to changes.

- **Data Flow from Physical System to Virtual Model:** This flow of data transfers information regarding the status of the robots from the physical system to the virtual model. This data is used to synchronize the movement of the robots with the simulation, ensuring that the virtual model accurately reflects the behavior of the physical system.
- **Data Flow from Virtual Model to Physical System:** This flow of data involves the transmission of instructions and control signals from FlexSim to the robots. Based on the decisions and actions taken within the virtual model, following the different dynamic task reallocation strategies, instructions are generated and sent to the robots. These instructions guide the behavior and actions of the robots in the physical system, by controlling their start and stop.

In the context of this Application, the connections between the robots and FlexSim are established using the Emulation tool provided by the software. The Modbus protocol was used, generating three Modbus TCP Connections, one for each robot, as can be seen in Fig 23.

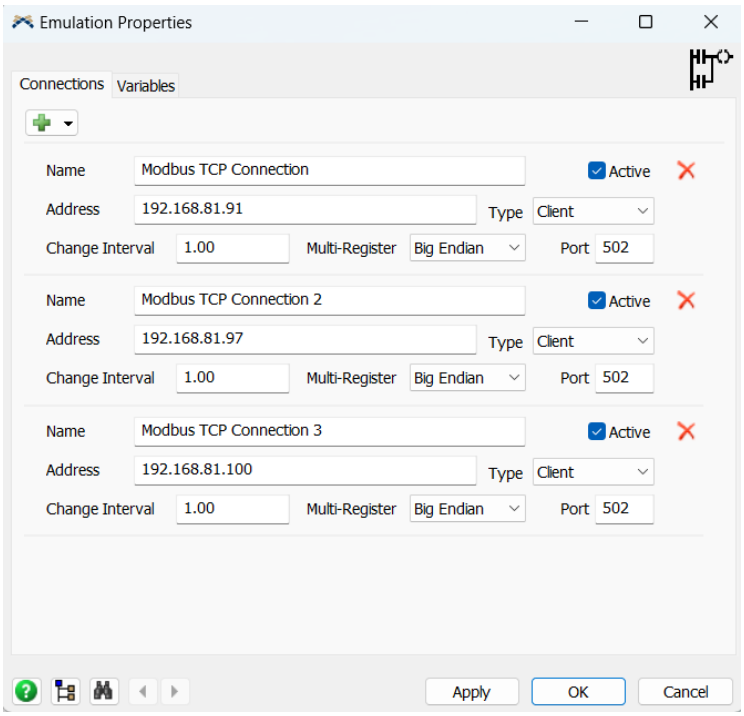

Fig 23. Flexsim's Emulation Tool, connection between the robots and the simulation.

Logics and functioning of the Physical System

In order to make sure that the robots work correctly, a series of logics were implemented. Waypoints were set to define the movements and path to be followed by the robots, each Waypoint being a determined position the robot must take. Moreover, signals were used to control the opening and closing of the gripper, for the robot to be able to perform the pick and place tasks.

Concerning the two UR3e series Robots, a 15 second wait was added to the cycle of the robots to simulate the processing of the items. In a production environment this would be a real delay time due to the processing work done by a human worker or by another machine.

Regarding the robot's functioning logics, the process starts when a signal is sent to both robots. As the robots work simultaneously in a single productive cell, they wait until receiving a signal from the other robot to start the process. If the signal is not received within 20 seconds, the process is ended; for this a thread is run in parallel of the main robot program, independent of the robot tasks.

However, if the signal is received the robot enters a loop, in which upon receiving another signal it is instructed to move an item. When no signal is received (digital output 4 = False) it waits for three seconds before entering the loop again, as can be seen in Fig 24. and Fig 25. This ensures that the robot is constantly checking for the signal, and that upon receiving it, it will start the movement within a three second delay. Moreover, this allows for the robots to be momentarily stopped if needed, but without exiting the program, which is a very useful feature to facilitate the reallocation of items.

Finally, when an item is picked the signal *DO[1]* is set to On and when the item is placed in its destination the signal is set back to Off. These signals are then picked up by the Digital System to ensure the synchronization of the movement between both systems.

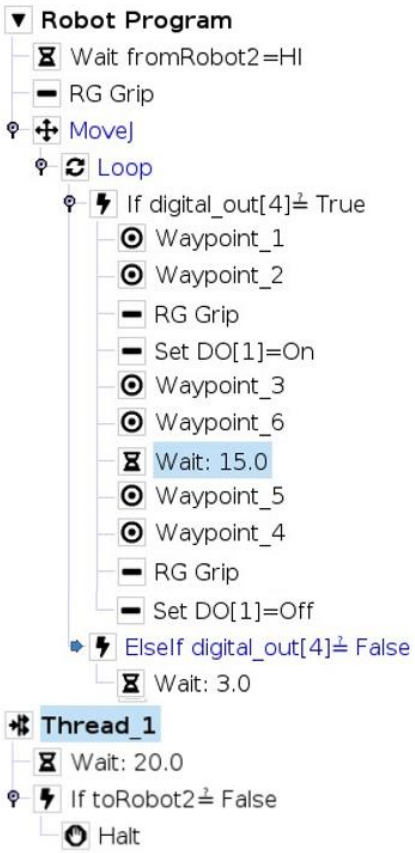

Fig 24: Robot UR3e's logics.

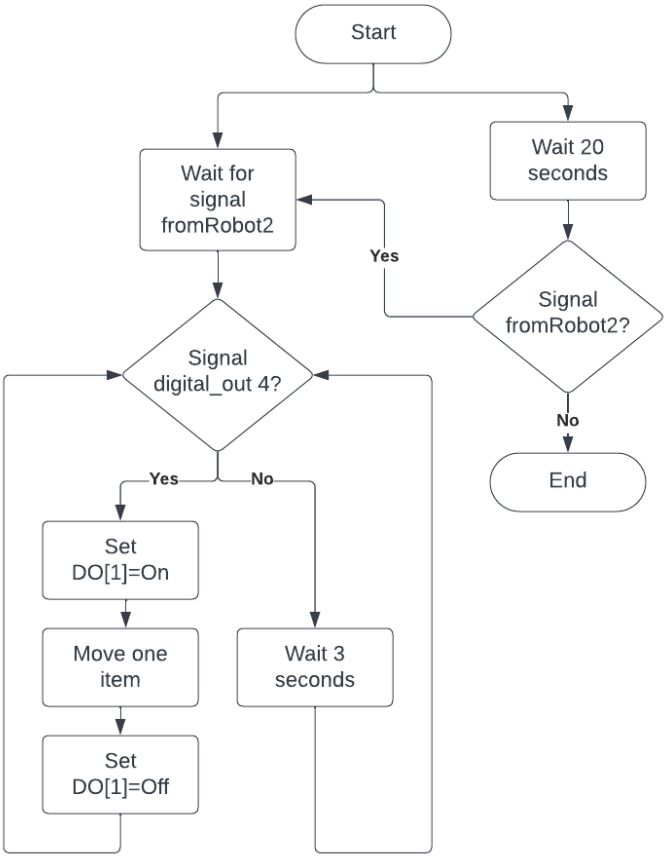

Fig 25: Robot UR3e's process flow.

With respect to the UR3 CB series Robot, responsible for reallocating the items between the two processing robots, its logic can be observed in Fig 26. and Fig 27. The reallocation process starts when the robot receives a signal. There are two possible signals the robot can receive that prompt two different movements: from Robot 1 to Robot 2 or vice versa. To ensure that the robot is constantly waiting for signals, a loop is utilized. If no signal is received it waits for 0.5 seconds before checking for any of the two signals again.

Moreover, like for the other two robots, when an item is picked the signal *DO[2]* is set to On and when the item is placed in its destination the signal is set back to Off.

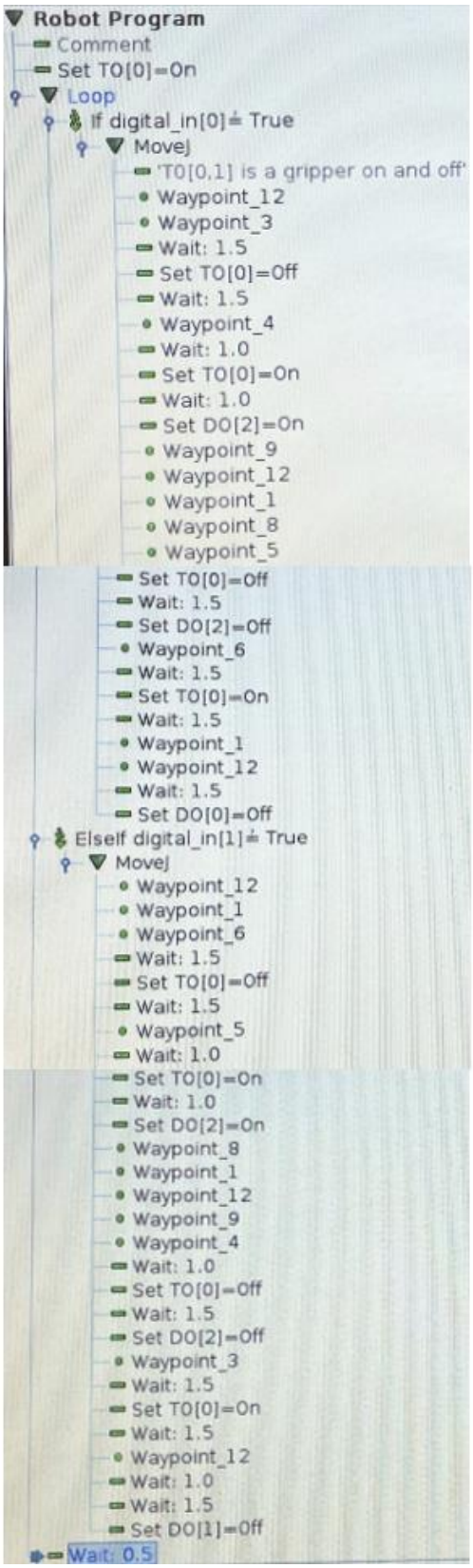

Fig 26. Robot UR3 CB's logics.

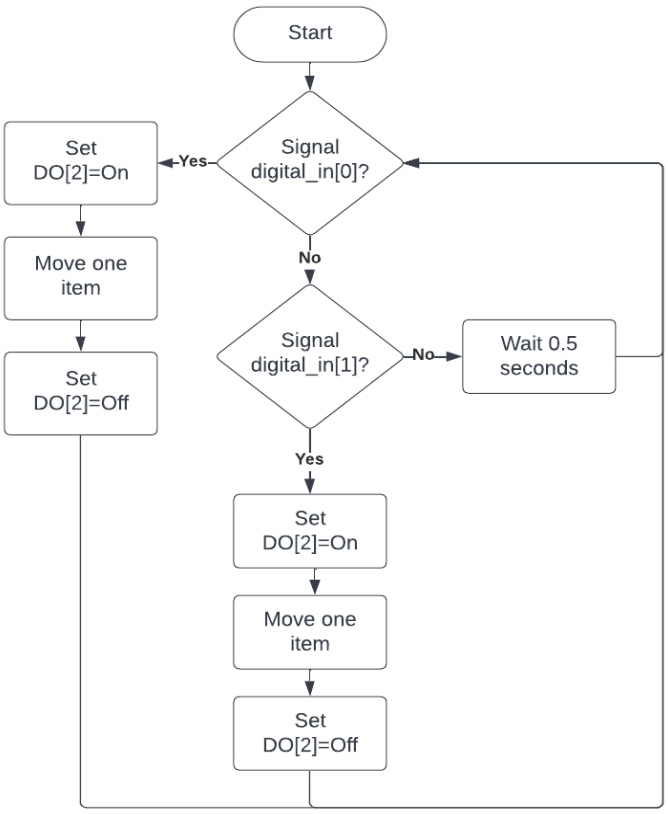

Fig 27. Robot UR3 CB's logics process flow.

Logics and functioning of the Virtual System

To ensure the correct functioning of the simulation, logics were implemented both in the 3D Model and the Process Flow Model. While logics within the 3D Model deal with the general functioning of the simulation, logics within the Process Flow Model handle the connection and communication between the physical and the virtual systems, thus enabling the Digital Twin Application.

Logics and functioning within the 3D Model

The process starts with the arrival of fourteen items to the system. This is done with two *Source* elements, each delivering seven items. An arrival schedule was used, with a table to determine the arrival time of seven items at the start of the Simulation.

Each *Source* element is connected to one *Queue* element. Using a *Trigger*, once a *Flow Item* enters a *Queue*, a *Label* is assigned to it. The *Label* is called Type and it can hold values of either 1 or 2, contingent upon the queue of entry and the robot assigned for processing of the item. This *Label* designation is important, as it allows for logics and computations to take place.

It is relevant to mention that the *Label* is assigned to the items each time they enter a *Queue*, and so if an item is moved from one *Queue* to another the label will take the value of the last *Queue* it entered to, consequently losing the value of the first one.

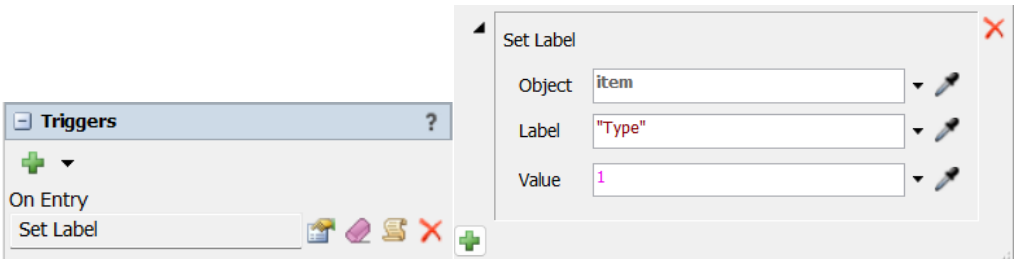

Fig 28. Trigger inside the Queue elements.

Items are stored in each *Queue* until retrieved by the *Processors*. Each *Queue* is connected to two *Processors*, one representing the UR3-CB-series robot in charge of the processing and the other the UR3 CB robot doing the reallocation of items when implementing the Digital Twin.

Each *Processor* element contains a trigger, that activates when the simulation is started, which stops the input port, preventing the entrance of items to the *Processor*. This is to ensure perfect coordination with the physical system.

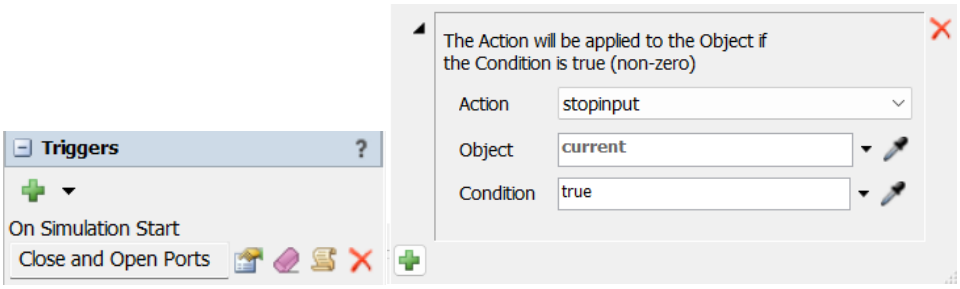

Fig 29. Triggers inside the Processor elements.

Within the Application, the processing of the *Flow Items* is controlled by opening and closing the input and output ports of the *Processors*, as will be explained in another section bellow. This way, the real processing time of the *Processors* is defined by the time between an item is allowed in and out of the *Processor*. For this reason, a processing time of 10 seconds is set in the *Processor* element as its processing time. This is not the real processing time, but a time short enough to ensure the correct functioning of the Model.

Moreover, the *Processor* associated with the robot performing the reallocation of items follows a specific logic when deciding which port to send the items upon exiting the *Processor*: it examines the *Label* Type of the item and then decides the port accordingly. This logic is necessary because the *Processor* operates between two *Queues* and to ensure that an item, once taken for processing, does not return to the same *Queue* from which it came from. Thus, if the item's Type is 1, it is directed to *Queue* 2, and vice versa.

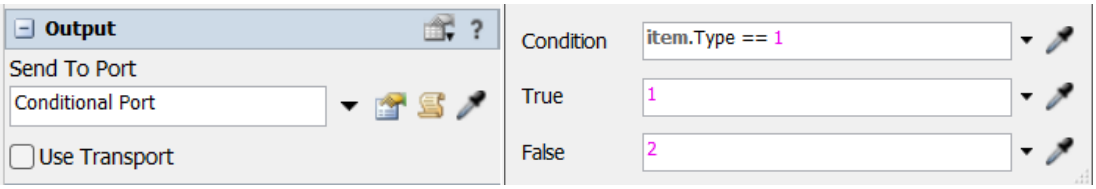

Fig 30. Logics inside the output port of the Processor.

Finally, the two other *Processors* are connected to a final *Queue* that collects the processed items. In this final *Queue* the number of items processed by each *Processor* is computed, also considering the Label Type assigned before; for this a custom code is used as a *Trigger.* This *Queue* element also has another trigger to reset the count upon resetting the program.

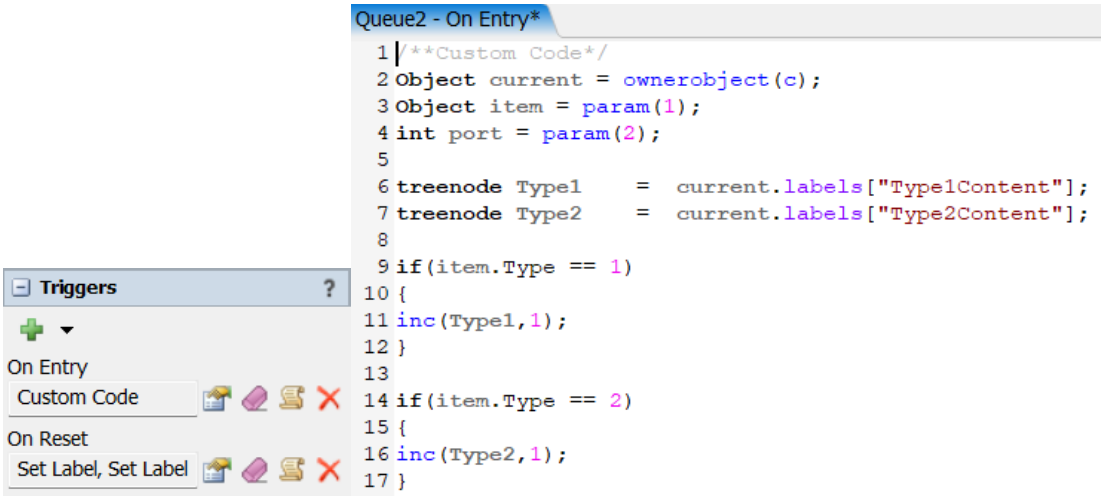

Fig 31. Triggers inside the Queue element.

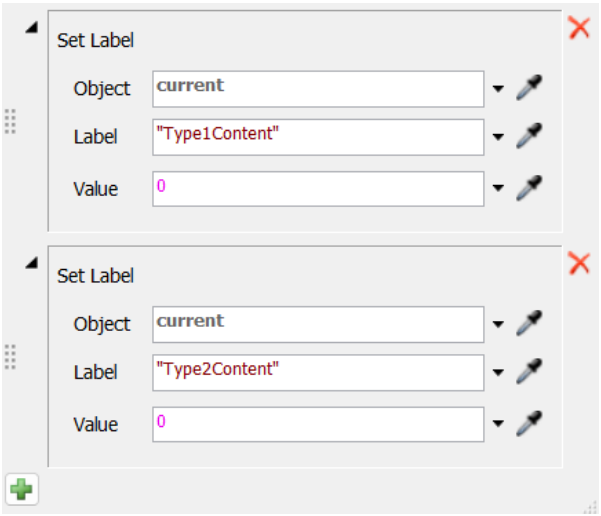

Fig 32. Triggers inside the Queue element.

Logics and functioning within the Process Flow

Variables of the Process Flow

To explain the functioning of the logics of the Process Flow it is necessary to first describe the variables that intervene in said logics. Nine *Internal Emulation Variables* are used, comprising six *Sensor Variables* and three *Control Variables*, as depicted in Fig 33.

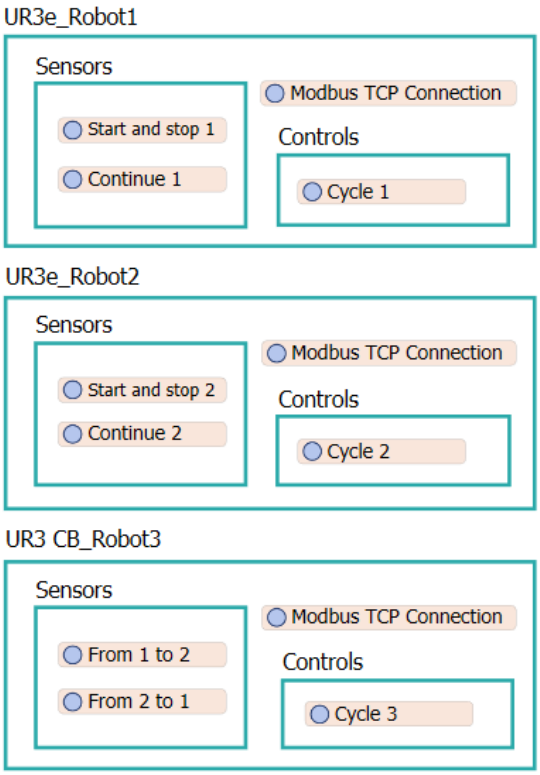

Fig 33. Process Flow Variables used in the Application.

These variables have different purposes and interact with the Virtual and Physical Systems in different ways. The connection of the variables to the Physical System through the Emulation Tool will be discussed in another section bellow.

As explained before, *Sensor Variables* send information from the Virtual to the Physical System, upon which the Physical System takes certain actions. The *Sensor Variables* used in the Application are the following:

- *Start and Stop* Variables (1 and 2 for each robot respectively): These variables are responsible for the activation of the Robots at the start of the Simulation.

  They can take the values of 0 or 1 and have an associated event at Simulation Start that assigns the value of 1 to the variable. This way, they send a signal (value of the signal equal to 1) at the start of the simulation to start the robots. When the signal is no longer received (value equal to 0) robots stop working and the program is stopped.

- *Continue* Variables (1 and 2 for each robot respectively): These variables are responsible for the momentarily stop and restart of the robots. They enable the reallocation of items, by stopping the functioning of the robots while the third robot reallocates the items between them.

  Like Start and Stop, they also take the values of 0 or 1 and have an associated event that on Simulation Start assigns the value of 1 to the variable to start the functioning of the robots.

  These variables send a signal to the robots (value of the signal equal to 1); when the signal is no longer received (value of the signal equal to 0) robots momentarily stop working until signal is received again.

- *From Robot 1 to 2* Variable and *From robot 2 to 1* Variable: These variables control the movement of the robot in charge of the reallocation of items. They send a signal to the robot to activate the right movement.

  They can take 1 and 0 as their values, having an associated event that on the start of the simulation that assigns the value of 0 to them. This way the robot is not activated at the start of the simulation, and instead is waiting for the signal to start its functioning.

Control Variables receive data from the Physical System, upon which the Virtual System reacts. The following *Control Variables* were used for the Application:

- *Cycle* Variables (1, 2 and 3 respectively): These variables are responsible for the movement synchronization between the Physical and Virtual Systems, as they control the performance of the Processors within the Model.

  The variables can take 1 and 0 values and have a series of associated actions according to the value taken by them. They receive a signal (signal value of 1) once the robot grabs an item and stop receiving it (signal value of 0) once the robot places the item in its destination.

  To ensure the precise synchronization between the Physical System and the Virtual System, the input and output port of the *Processors* are controlled. For this, a time short enough to ensure the correct functioning of the Model is set to be the processing time in the *Processor* element of the 3D Model, allowing the real processing time to be defined by the input and output ports. This way, by controlling the ports of the *Processor*, not only the processes are completely synchronized, but also the processing times are equal.

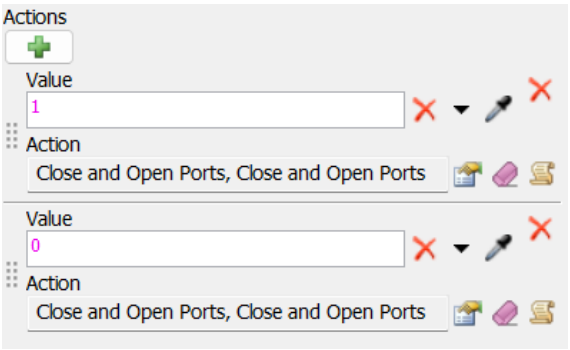

Fig 34. Actions taken by the Cycle Variable.

Upon receiving a value of 1 for the Variable, the Model instructs the *Processor* to resume its input of items while stopping its output, thereby forcing the element to remain in the processor until the signal to end its processing is received.

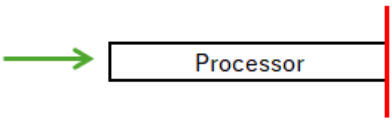

Fig 35. Resume Input and Stop Output graphical representation.

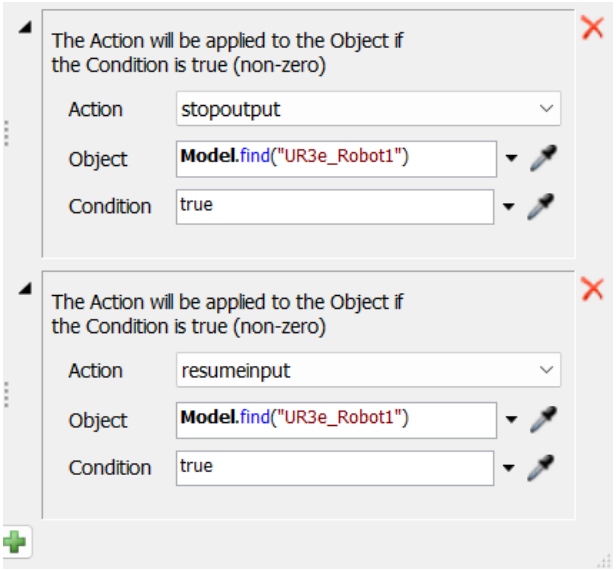

Fig 36. Resume Input and Stop Output in the Application.

Conversely, upon receiving a value of 0, the Model sets the processor to resume the output of elements, enabling the items to exit the processor, while stopping its input until the signal indicating that the robot has grabbed another item is received.

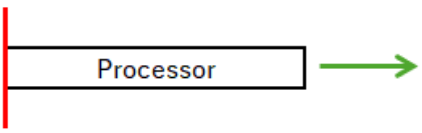

Fig 37. Stop Input and Resume Output graphical representation.

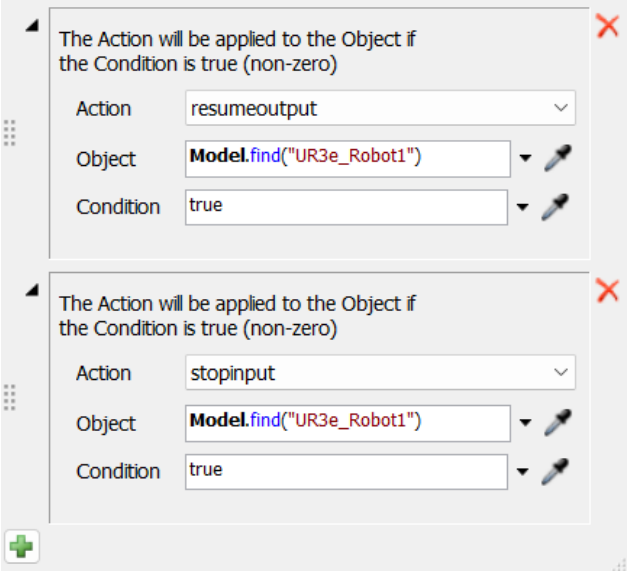

Fig 38. Stop Input and Resume Output in the Application.

Simulation start and functioning before Reallocation of items

Before the Simulation begins, the initial value of all the variables within the Application is 0. As the Simulation starts various events and actions are triggered within the Process Flow, turning the *Start and Stop* and *Continue* Variables values to 1. This triggers the functioning of the robots within the Physical System, allowing them to pick the first item. [14]

As previously mentioned, as each a robot retrieves an item from its respective queue, a signal is sent, and upon placing the item in its destination, the signal is ended. This triggers the *Cycle* variable, which manages the input and output ports of the Processor element in the 3D Model, allowing the Model to work accordingly to the real process.

For the process to be iterated, a simple logic is implemented in the Process Flow. Whenever an item exits the *Processor,* signifying a change in the number of items in its queue, a token is generated with an *Event Triggered Source* in the Process Flow.

This token then moves to a *Decide* block that assesses whether there are still items in the Processor's queue. This is done by using the *subnode.lenght* expression, which counts the number of items inside a 3D Model element, as it can be seen on Fig 39. If the response is affirmative, indicating that there are still items present in its queue, the *Continue* Variable is set to 1, allowing the robot to pick another item and the process to continue.

[14] The implicit assumption being that each robot has at least one item in its queue.

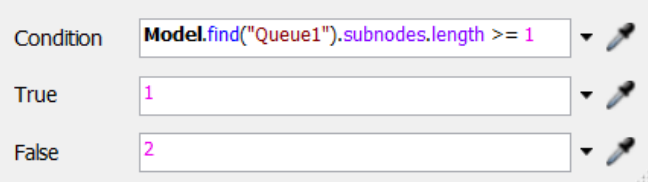

Fig 39. Condition applied in the first Decide Block.

If there were no more items in queue however, the *Continue* Variable is set to 0, discontinuing the signal that enables the functioning of the respective robot, bringing it to a halt.

The process then continues with a *Decide* block that asses the number of items in queue of the other robot and determines whether there is a need for a reallocation of items: if there are less than two items in queue no reallocation takes place, case contrary the reallocation logic is executed. The reasoning behind this is that if there is only one item in the queue of the other robot or if the other robot is processing its last item, it would be faster and more efficient for that robot to complete its work rather than reallocating the item to the other robot for processing, considering the time taken for the reallocation of items. Therefore, in such scenarios, no reallocation of items is deemed necessary.

The complete process flow before the reallocation is implemented can be seen in Fig 40.

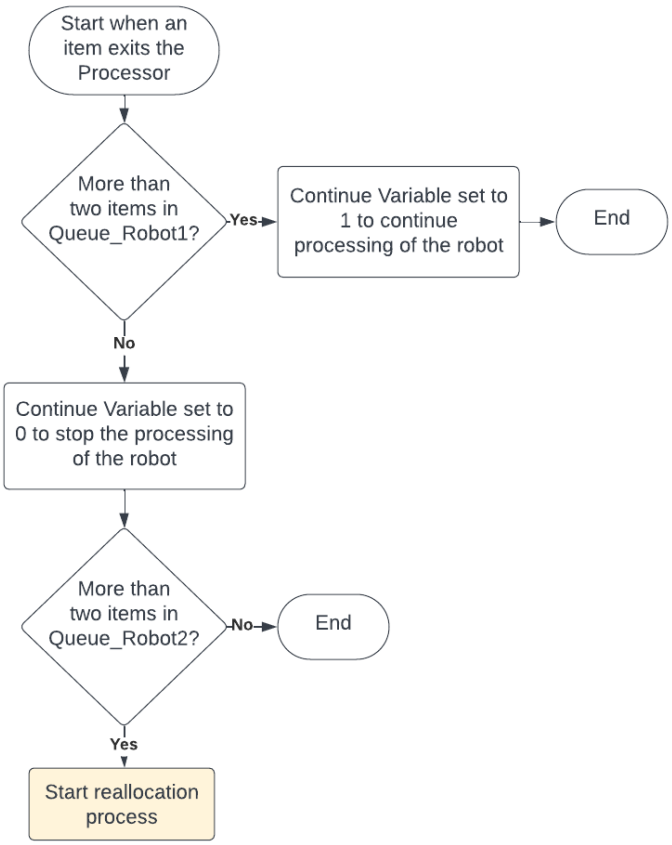

Fig 40. Process Flow before the reallocation of items.

Digital Twin for dynamic task allocation – Third scenario: Reallocation of one item at a time

In this scenario one of the robots experiences a time failure. Consequently, would the initial schedule be maintained, the slower robot would take a longer time to accomplish its assigned tasks. For this reason, a Digital Twin is implemented to mitigate the issue, by reassigning items from the slowest to the fastest robot.

This reallocation of items is done once the Digital Twin detects that one of the robots has finished its assigned tasks, while the other still has items to be processed. The Digital Twin then momentarily stops the processing of the items and moves one item from the queue of the slowest robot and to the queue of the fastest robot, reassigning the processing of the items. Once this is completed both robots resume their work. If another reallocation is needed, once the fastest robot finishes processing the new item, the reallocation process is performed once again, moving one item from the slowest and to the fastest robot.

For this and the following scenarios, the UR3e_Robot1 represents the robot that experiences the time failure, and thus the reallocation process is done from its queue and to the queue of the UR3e_Robot2; were the UR3e_Robot2 the robot to experience the time failure, the reallocation process would be activated from its queue and to the other robot's queue.

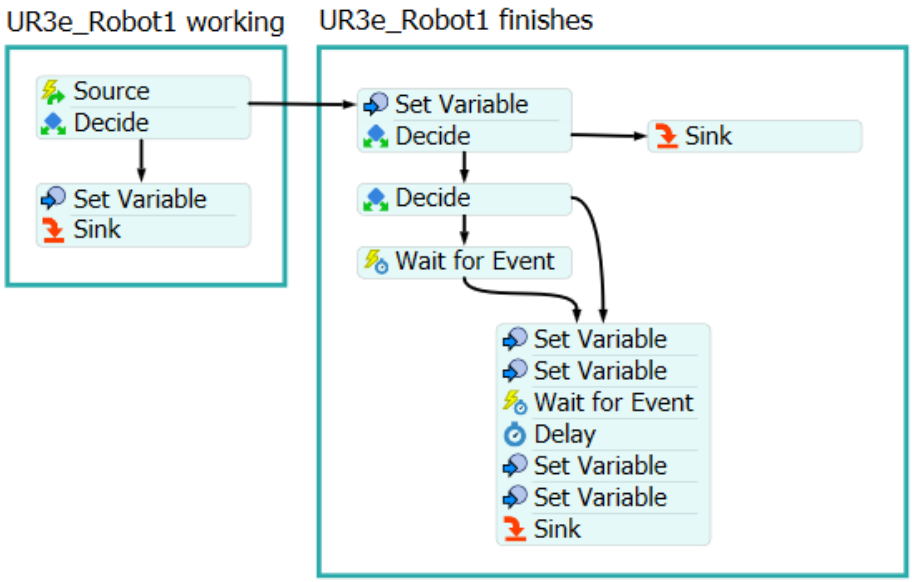

Fig 41. Process Flow of the Third scenario of the Application of Digital Twin for dynamic task allocation.

Following with the logic within the *Process Flow*, once the decision to reallocate items is made, the Model checks if the second robot is currently processing any item. This is implemented to guarantee that the robot is not stopped mid operation, and to prevent any potential glitching. For this, the *subnode.lenght* of the *Processor* item is assessed, as was done before for the *Queue* elements.

If the robot is currently engaged in a processing task, the simulator awaits the completion of the task, ensuring that the item is placed in its final location before proceeding. A *Wait for Event* element is used, the event being the exit of an item from the *Processor*.

When the processing task is concluded, or if the second robot is not currently processing any item, the signal enabling the robot to work is promptly discontinued, by setting to 0 the Variable *Continue* of the respective robot. As a result, the second robot halts its operations, ensuring that the reallocation process can be executed properly.

Subsequently, the Variable *From Robot 2 to Robot 1* is set with a value of 1, instructing the robot designated for the reallocation process to commence its task. The Variable being activated (*From Robot 1 to Robot 2* or *From Robot 2 to Robot 1*) is determined by which of the two robots has zero items in its queue, ensuring that the movement is done from the robot with items in its queue to the one without.

The system then detects the completion of the reallocation process when there is a change in the number of items in the queue of the *Processors*. Another *Wait for Event* element is implemented, with the event being the increase in the content of the *Queue* that had zero items before the reallocation.

For safety reasons, to guarantee that the reallocation robot has finished its movements before resuming the movement of the other robots, a *Delay* element is added to the Process Flow, with a delay time of 30 seconds.

Two *Set Variable* elements are used to assign a value of 1 to both *Continue* Variables to signal the robots to resume their respective processing tasks.

Once one of the robots runs out of items in queue again, if the above logics are satisfied, a second reallocation of items could take place. This iterative process continues until there are no more items remaining in the queues of the robots, indicating that all items have been successfully processed.

The complete process flow for the Third scenario of the Application of Digital Twin for dynamic task allocation can be seen in Fig 42.

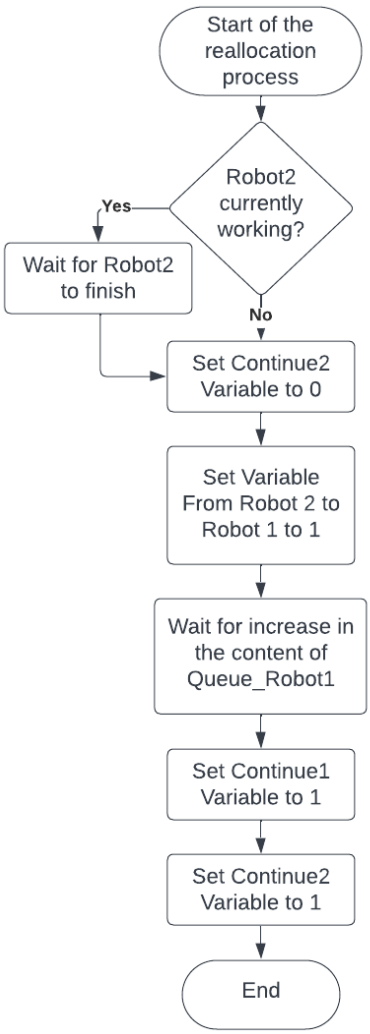

Fig 42. Process Flow for the Third scenario of the Application of Digital Twin for dynamic task allocation.

Digital Twin for dynamic task allocation - Fourth scenario: Reallocation of an optimal batch of items

In this scenario one of the robots also experiences a time failure and the Digital Twin is implemented to reassign the items between the robots. This is done by reallocating an optimal batch of items, as opposed to reallocating one item at a time as in the third scenario.

The operational logics mirror those of the third scenario: once one of the robots completes its assigned tasks while the other robot still has items in queue to be processed, the reallocation process is triggered. The robots are temporally halted, allowing for a safe reallocation of items. Once the items are reallocated both robots resume their processing tasks.

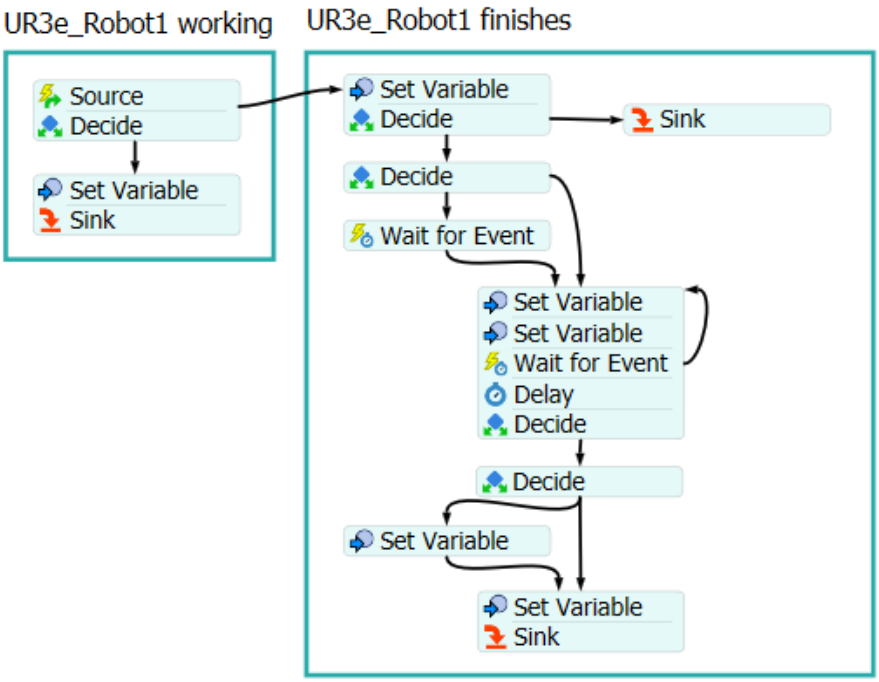

Fig 43. Process Flow of the Fourth scenario of the Application of Digital Twin for dynamic task allocation.

With respect to the logic within the *Process Flow*, the first steps mirror that of the third scenario. When it is determined that a reallocation needs to take place, the Model first assesses whether the second robot is actively processing any item. If so, it waits for the completion of the task, ensuring that the item is placed in its final location before continuing. Subsequently, the Variable *Continue* is set to a value of 0, halting the robot's process, while the Variable *From Robot 2 to Robot 1* is set with a value of 1, prompting the robot designated for the reallocation process to work. The process proceeds once the reallocation process is completed, as indicated by an increase in the number of items of the *Queue* that had zero items before the reallocation.

Additional logics were implemented at this stage due to the need to reallocate more than one item within this scenario. The reallocation of items follows an iterative process in which the number of items in the queue of the robots is continually compared against the optimal number of items required for the system to operate at its peak efficiency. For this a *Decide* Block was utilized.

Each time an item is reallocated, the system evaluates the current state against said optimal number. If the logic is not satisfied, meaning that the optimal number of items moved has not yet been reached, an additional item is moved. This iterative approach ensures that the reallocation process continues until the optimal batch of items is achieved.

The optimal number of items to be reallocated is calculated considering both the total number of items to be moved and the ratio of processed items by each robot (indirectly measuring the speed of the robots, assuming a constant speed). It can be summarized as the following expression:

Optimal number of items to be moved = Total number of items x Ratio of processed items

The Total number of items is calculated simply by summing the number of items in the *Queues* of both *Processors*, using the *subnode.lenght* feature.

For the Ratio of processed items, two item *Labels,* previously calculated in the *Final_Queue* in the 3D Model, are used. These *Labels*, *Type1Content* and *Type2Content,* track the total number of items processed by each robot, each type corresponding to each robot respectively. The ratio is then computed by dividing the number of items processed by one robot by the total number of items processed by both robots.

Following the Process Flow, upon completion of the iterative reallocation process and once the number of items in the queue aligns with the optimal calculated number, the robots can resume performing their respective tasks.

Unlike the third scenario, the fourth scenario allows for the condition where one robot has zero items in its queue, due to the reallocation of all its items to the other robot. Consequently, a specific logic must be implemented to determine whether it is necessary to send a signal to that robot to restart operations. This is done with another *Decide* element. Then, if the robot has more than one item in queue a signal is sent for it to resume operations, setting the *Continue* Variable to 1; case contrary no signal is sent, *Continue* Variable equal to 0.

Finally, another *Set Variable* element is used to assign a value of 1 to the *Continue* Variable of the other robot for it to resume its processing tasks.

The complete process flow for the Fourth scenario of the Application of Digital Twin for dynamic task allocation can be seen in Fig 44.

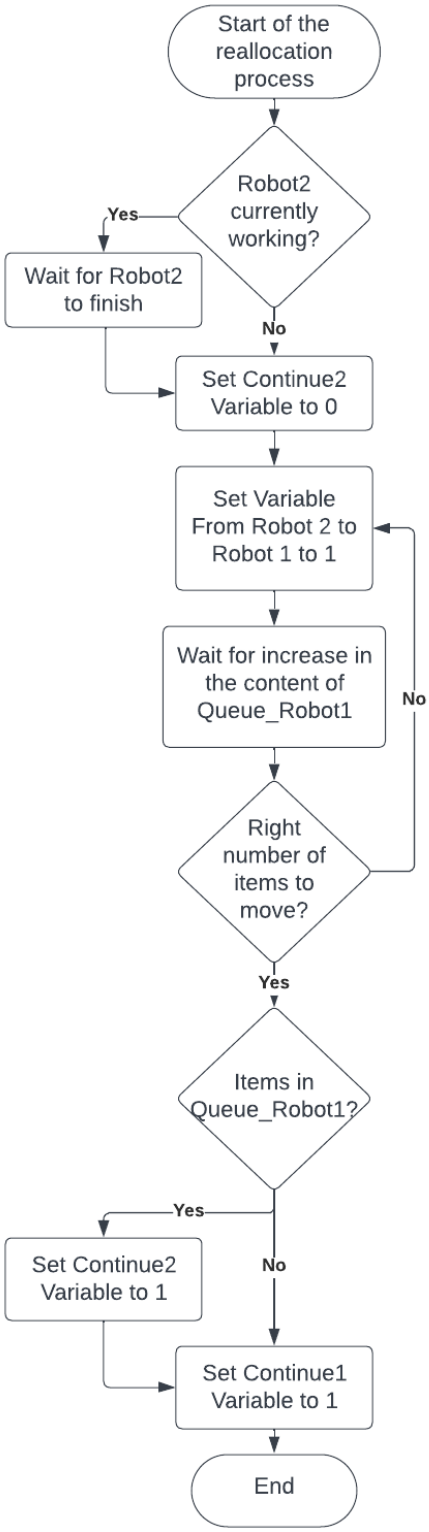

Fig 44. Process Flow for the Fourth scenario of the Application of Digital Twin for dynamic task allocation.

Connections between the Virtual and Physical Systems

For the correct functioning of the Model, the Virtual and Physical Systems must be connected, always sending and receiving signals. For this reason, each Variable within the simulation has an associated signal within the Physical System; this can be seen in Table 1.

Variables of the Virtual System	Signals of the Physical System		
	UR3e Robot1's Signals	UR3e Robot2's Signals	UR3 CB Robot's Signals
Start and Stop 1	ToRobot2	FromRobot1	
Start and Stop 2	FromRobot2	ToRobot1	
Continue 1	digital_out 4		
Continue 2		digital_out 4	
From Robot 1 to Robot 2			digital_in [0]
From Robot 2 to Robot 1			digital_in [1]
Cycle 1	DO[1]		
Cycle 2		DO[1]	
Cycle 3			DO[2]

Table 1. Variables of the Virtual System and its Signals of the Physical System.

Flexsim's Emulation tool plays and important role, enabling the communication to take place. Utilizing the Modbus Protocol, each variable has an associated specific register of operation with the robots' PLCs.

Run of the Simulation, simplifications made and its modifications when applied to a production environment

In order to simulate the time failure of one of the robots, the operation speed of UR3e Robot1 was reduced to 15% of its normal operation speed. This adjustment means that UR3e Robot1 operates at approximately 15% of UR3e Robot2's speed, considering that both robots operate at nearly the same speed under normal conditions. This 15% speed reduction was chosen to ensure that the simulation adequately demonstrates the reallocation process, allowing for the reallocation of several items between the robots, thus accurately reflecting the impact of the time failure.

Moreover, this 15% speed reduction was set to be constant throughout the entire simulation process. This approach ensures that the calculation of the optimal number of items to be moved within the second reallocation scenario is accurate, as the rate of production remains constant over time.

In a real-world scenario, the time failure would occur over time, with the incremental reduction of the speed of one of the robots. Therefore, the optimal number of items to be moved would

be calculated considering the average processing speed of the robots over time, as opposed to a fixed speed reduction. As a consequence, the calculated number of items to be moved might not be as accurate, depending on how the speed reduction of the robots occurs. To address this, an alternative method for calculating the number of items to be moved, which takes into account the speed progression over time, could be implemented. This would allow for a more precise adjustment in response to the gradual changes in the robot's operating speed.

Another simplification made was the implementation of a 15 second wait time in the robot's logics, to simulate the processing work performed by a human worker or another robot. In a production environment, this time would be replaced by the real time required for processing of the items.

Results and Analysis

The four scenarios covered in the Application of Digital Twin were implemented. For each scenario a set of indicators was computed, facilitating the comparison between them:

- Throughput of the System (TH): This metric quantifies the number of items processed per unit of time.
- Average cycle time of the System (CT): Defined as the average time needed to process all the items in the system, measured from the moment the simulation starts (robots start functioning) and until it ends (robots finish their tasks).
- Utilization of the robots (U): This metric represents the average percentage of time during which the robots are in the processing state, actively processing items.

The results obtained for these indicators are presented in Table 2. In the following sections a more detailed explanation will take place.

Indicator	First scenario (no failure)	Second scenario (failute without DT)	Third scenario (failure with DT, item reallocation)	Fourth scenario (failure with DT, batch reallocation)
Throughput (TH)	288 items/h	120 items/h	144 items/h	152.7 items/h
Cycle time (CT)	175 s	420 s	350 s	330 s
Utilization Robot1 (U1)	100%	100%	84%	72.73%
Utilization Robot 2 (U2)	100%	41.67%	64.29%	64.29%
Utilization Robot 3 (U3)	0%	0%	16%	25.45%

Table 2. Indicator values for the three scenarios.

Before discussing each scenario in detail, it is important to mention that for the third and fourth scenarios, the reallocation process done by the UR3 CB Robot is taken into consideration for computing some of the above indicators. This way, the Cycle time encompasses not only the processing time of the items done by the UR3e Robots, but also the time associated with the reallocation process.

Conversely, since no real processing is done by the UR3 CB Robot, as it merely handles the internal reallocation of items within the same system, its activity is not considered when computing the Throughput of the process.

First Scenario - Non-dynamic Scheduling without Failure

The first scenario represents the optimal outcome, in which no time failure, disruptions or slowdowns occur, allowing tasks to be executed according to the predefined schedule. As a result, this scenario reports the best performance.

As mentioned before, in this set up both robots are assigned seven items to process. The two robots are assumed to be nearly identical and operate under the same conditions, resulting in identical processing times of 25 seconds, within a tolerance range due to natural variability inherent to the manufacturing process.

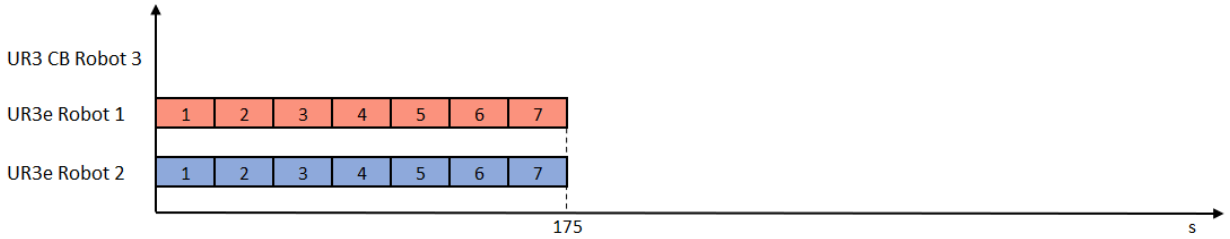

Fig 45. First scenario, non-dynamic scheduling without failure.

The average Cycle time of this scenario can be approximated at 175 seconds, with a Throughput of 288 items per hour, considering the contribution by both robots. The Utilization of the system is at 100%, as no time failure occurs.

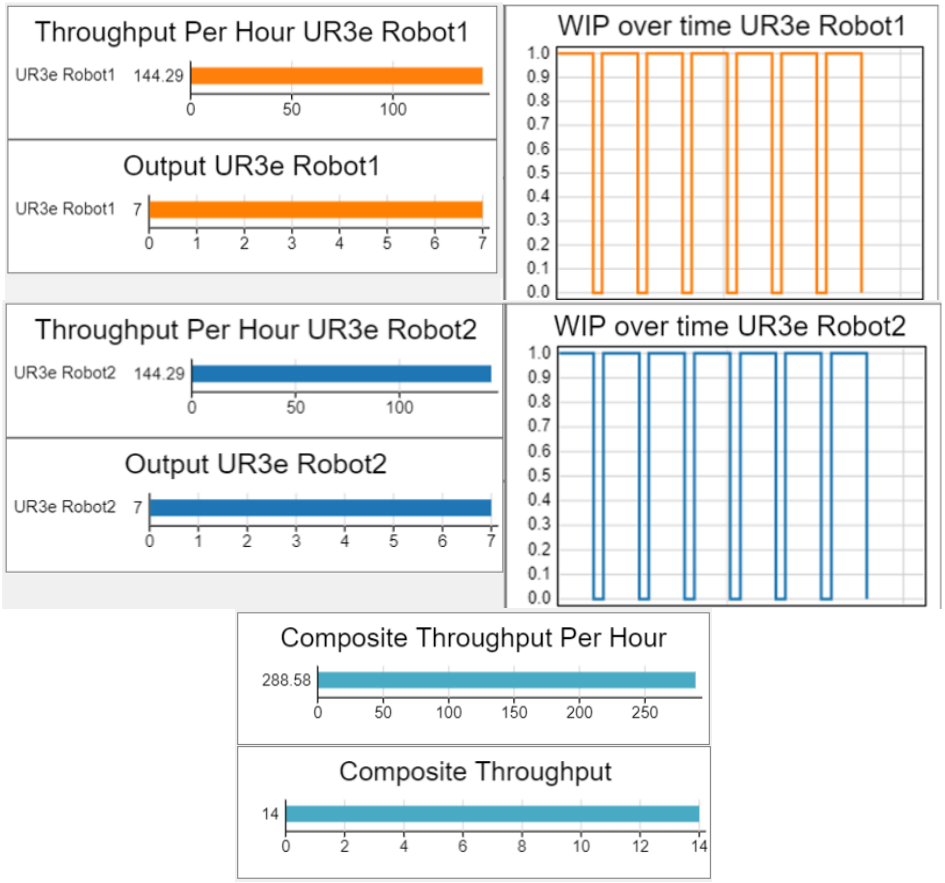

Fig 46. Flexsim's dashboard for the First Scenario.

Second Scenario - Non-dynamic Scheduling with time Failure

In this scenario, UR3e Robot1 experiences a time failure, resulting in a slower execution of its assigned tasks. Despite the occurrence of the time failure, the initial task scheduling remains unchanged, and no reallocation of items takes place. Consequently, UR3e Robot1 takes a longer time to complete the seven assigned tasks. As expected, this scenario displays the worst values for the selected performance indicators.

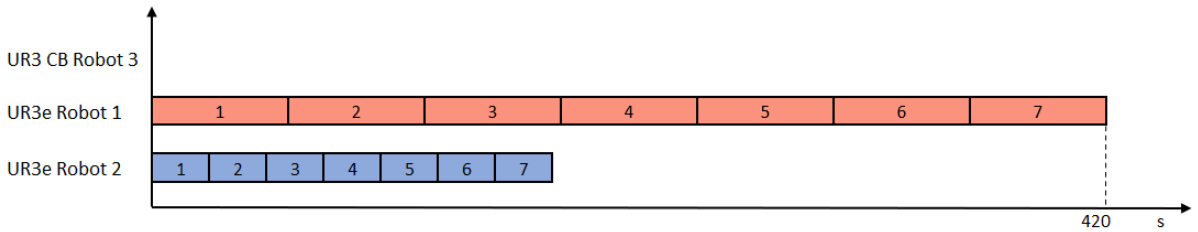

Fig 47. Second scenario, non-dynamic scheduling with time failure.

UR3e Robot1 takes approximately 2.4 times longer than expected to process an item, with a processing time of approximately 60 seconds, compared to the 25 seconds it would take in the ideal scenario, without the occurrence of a time failure. Due to the absence of a reallocation strategy, the total System processing time increases to approximately 420 seconds. Thereby UR3e Robot 1 becomes the bottleneck for the overall system.

As a consequence, the Utilization of UR3e Robot2 is reduced to approximately 41.67%, as opposed to the 100% of the best-case scenario, causing a decrease in the total Utilization of the System. This reduction is due to idle time experienced by UR3e Robot2 after completing its processing tasks and while waiting for UR3e Robot1 to complete its tasks. The Throughput of the System is estimated at 120 items per hour.

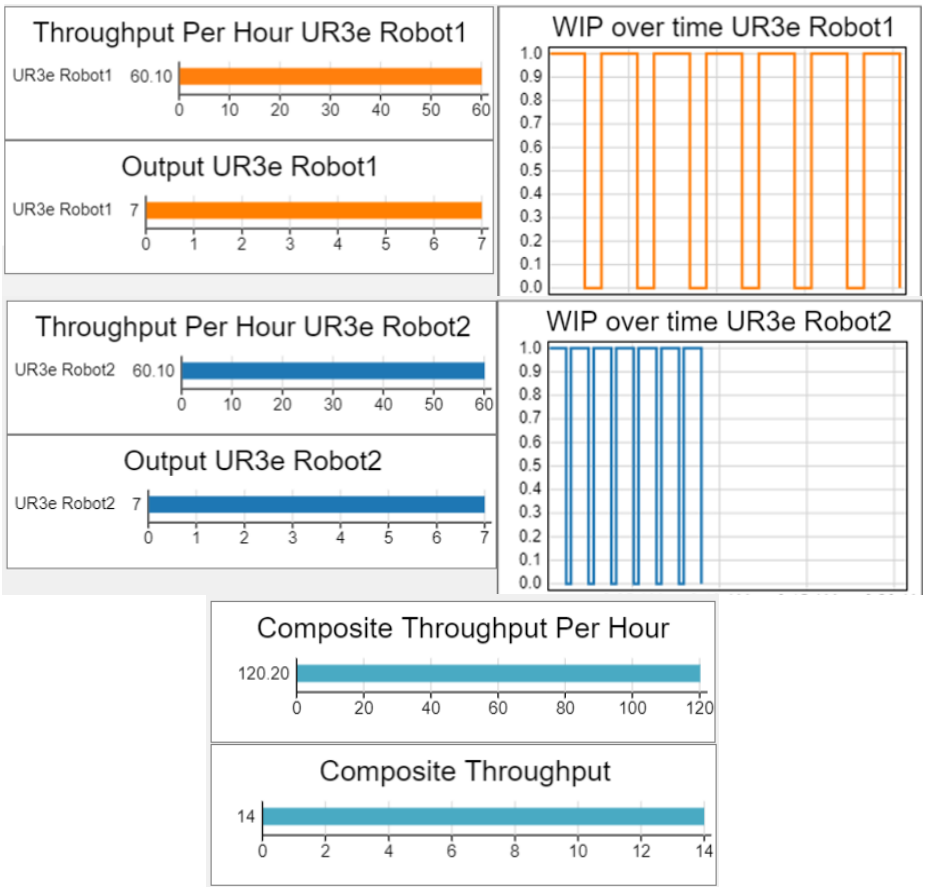

Fig 48. Flexsim's dashboard for the Second Scenario.

Third Scenario - Dynamic Scheduling with Time Failure, reallocation of one item at a time

Similar to the second scenario, in this scenario UR3e Robot1 also experiences a time failure, which results in a slower execution of its tasks. This scenario however, upon the detection of the time failure, and with the use of the Digital Twin system, the reallocation process is activated, reassigning the items to be processed between the two UR3e robots and thus mitigating the negative consequences of the failure.

The reallocation of items between the two robots is done by the UR CB Robot, following a sequential one-at-a-time reallocation strategy. This way, when the Digital Twin system identifies the need for a reallocation, a single item is transferred between the two UR3e robots for processing. Once this reallocation is completed, the robots resume their operations. If further reallocations are necessary, they will be detected and executed after the robots have finished their current processing cycle. More information on the logics governing this reallocation strategy can be found on previous sections.

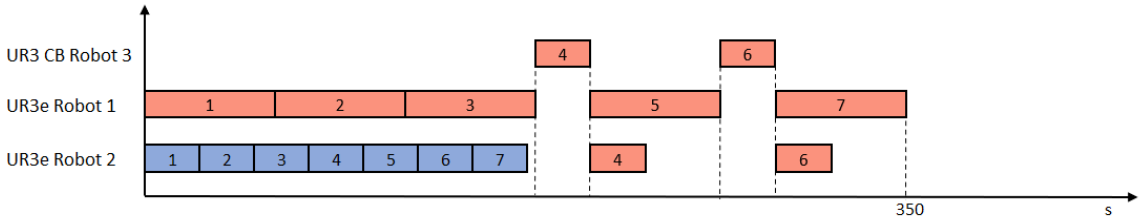

Fig 49. Third scenario representing the dynamic scheduling with time failure, with the relocation of one item at a time.

In this scenario, the processing time of UR3e Robot2 remains at 25 seconds, as the robot does not experience any time failure. Conversely, UR3e Robot1's processing time increases to 60 seconds due to its time failure. UR3 CB Robot performs the reallocation of items at 25 seconds per item.

As a result, the Throughput of the system improves to 144 items per hour, while the Cycle time reduces to 350 seconds, as opposed to the previous scenario. As expected, although results are less favorable than those obtained in the first scenario, they represent an improvement over the second scenario where no reallocation strategy was implemented. This can be attributed to the intervention of the Digital Twin in mitigating the issue; in this application the Digital Twin is used for reducing the impact of the time failure, but it does not address the cause of the issue nor solves it completely.

The Utilization for UR3e Robot1 is computed at 84%, for UR3e Robot2 at 64.29%, and for the UR3 CB Robot at 16%. It is important to note that the reduction in Utilization for UR3e Robot1 is entirely attributable to its operation being momentarily stopped during the reallocation process for safety reasons, to avoid the collision of the robots. Similarly, part of the reduced Utilization for UR3e Robot2 is also explained by this factor. Conversely, UR3 CB Robot is only active during the reallocation of items, hence getting its utilization percentage solely from that.

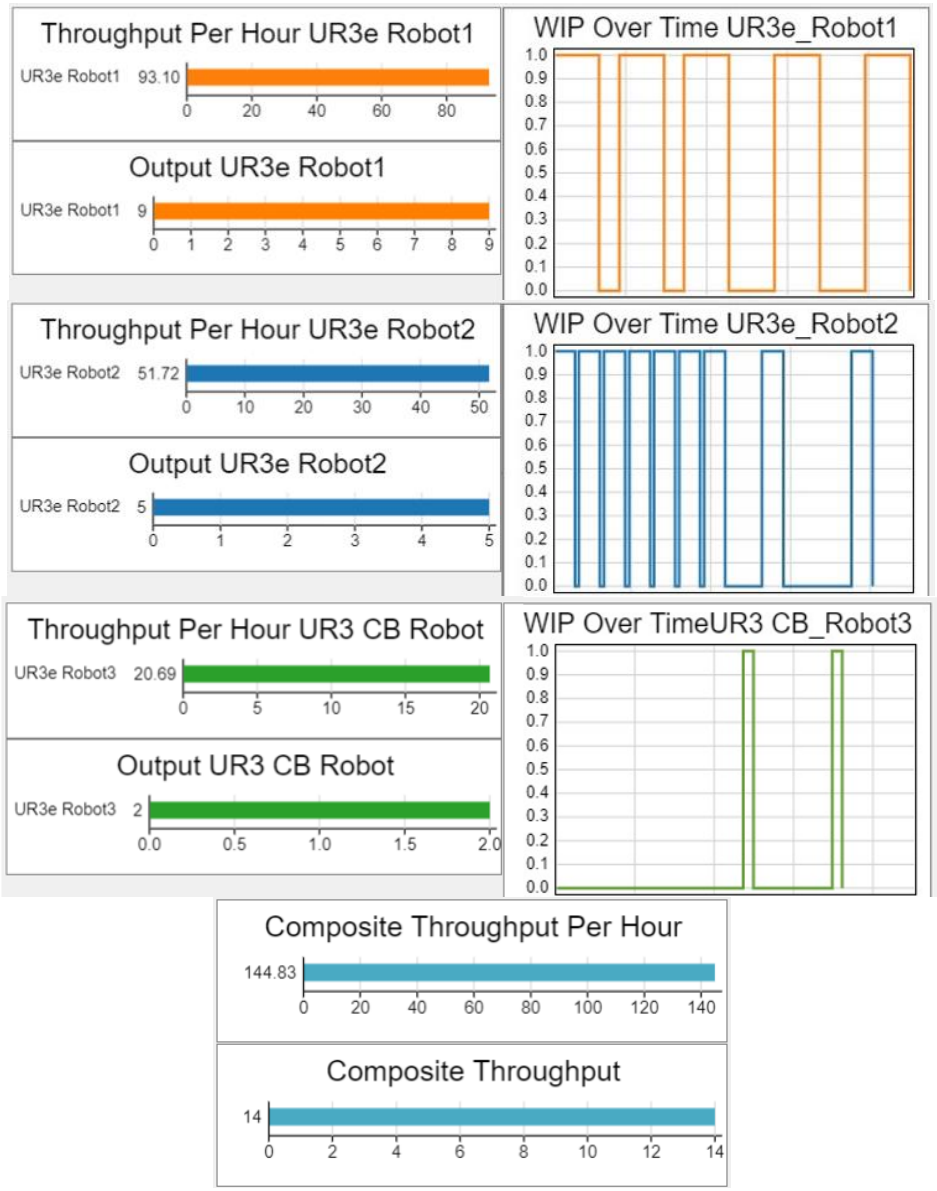

Fig 50. Flexsim's dashboard for the Third Scenario.

Forth Scenario - Dynamic Scheduling with Time Failure, reallocation of an optimal batch of items

In this scenario, similar to the previous one, the Digital Twin system detects in real-time the occurrence of a time failure affecting the performance of UR3e Robot1 and implements a mitigation strategy by adjusting the original production schedule and reallocating items between the robots. However, unlike the earlier approach where only one item was transferred at a time, this scenario involves reallocating an optimal batch of items in a single event. By transferring multiple items at once, the system minimizes the frequency of the reallocation process, which in turn reduces the number of interruptions to the production process, not only accelerating the completion of the tasks but also minimizing operational disruptions. The way this optimal batch of items is calculated, as well as the logic behind this scenario, has been previously explained.

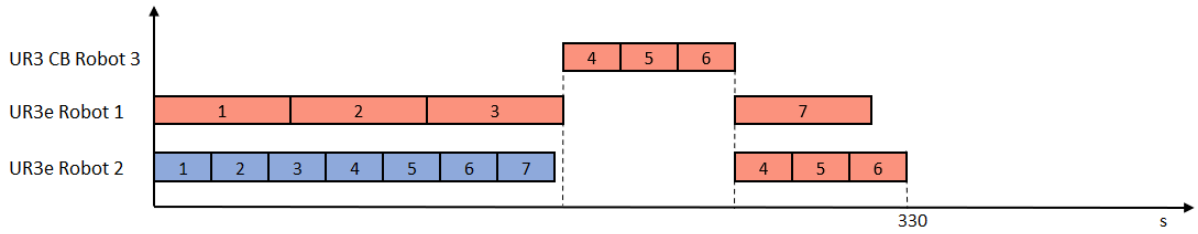

Fig 51. Forth scenario representing the dynamic scheduling with time failure, relocating a batch of three units.

Both the throughput and the Cycle time are improved in this scenario when compared to the previous one, with the Throughput reaching 152.7 items per hour, and the Cycle time reducing to 330 seconds.

The Utilization values are recorded at 72.73% for UR3e Robot1, 64.29% for UR3e Robot2 and 25.45% for the UR3 CB Robot. As observed in the previous scenario, these values can be explained (fully for UR3e Robot1 and partially for UR3e Robot1) by the periods of forced inactivity of the robots during the reallocation process.

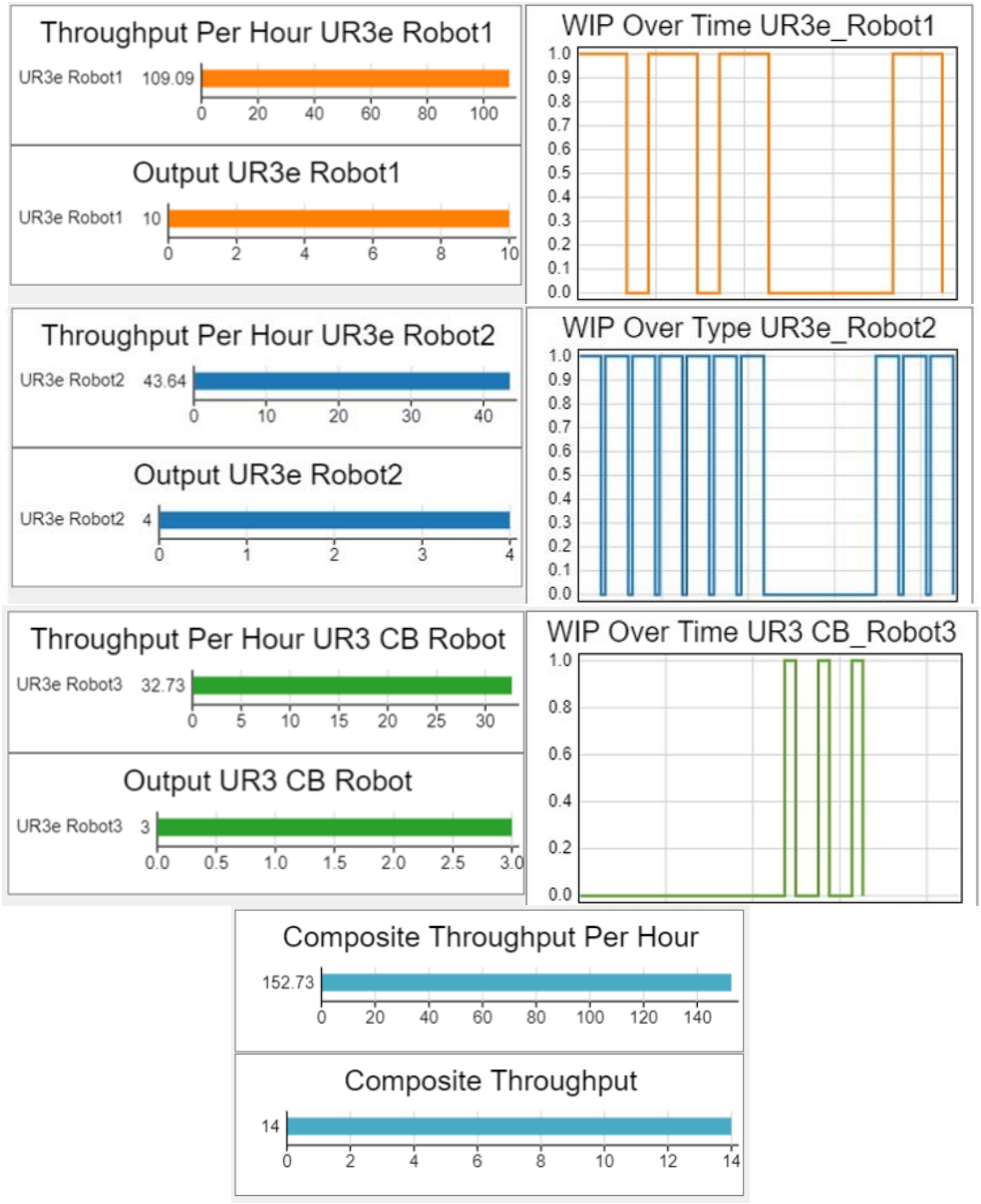

Fig 52. Flexsim's dashboard for the Forth Scenario.

When comparing the Third and Fourth scenarios, there is a significant difference in the values taken by the different parameters. In the Fourth scenario, the Utilization values are slightly lower for UR3e Robot1 and marginally higher for the UR3 CB Robot. This discrepancy is driven by the difference in the reallocation strategies employed in the two scenarios.

Given the difference in the logics governing each scenario, in the Third scenario the Digital Twin system determines the reallocation of two items, while in the Fourth scenario it calls for the reallocation of three items. Consequently, and when compared to the Fourth scenario, in the Third scenario UR3e Robot1 processes one item more, as there is one reallocation of items

less, leading to an increased Utilization. Moreover, UR3 CB Robot is engaged only twice, which reduces its overall Utilization.

Overall, the Fourth scenario proves to be the most efficient when compared to the Third scenario. Its primary advantage lies in the calculation of the optimal number of items for reallocation between the robots as soon as the time failure is detected for the first time, allowing for a single, consolidated reallocation event to take place. This reduces the frequency of the reallocation of items between the robots, reducing the operational disruptions. Hence, robots experience less downtime while waiting for the reallocation process to be complete. Moreover, the Fourth scenario has a better performance with respect to resource allocation than the Third scenario, and consequently present a lower cycle time for the system.

Finally, it is important to note that, in the context of the application developed in the laboratory, as outlined in the simplifications and modifications section of this thesis, the time failure of the UR3e Robot1 is modeled as constant over time. This justifies the calculation of the optimal number of items to be reallocated in the Fourth scenario and it is aligned with the Fourth scenario being the most efficient approach in this case.

However, in a real production environment, the occurrence of time failures may not follow such a constant pattern. In such a case, the Third Scenario, which sequentially reallocates one item at a time, may prove to be more suitable, as the Digital Twin decides in each iteration if an item needs to be reallocated or not.

Conclusion

The paper explored the application of the Digital Twin technology with the intention of generating an accurate real-time virtual representation of the physical system, that could detect the occurrence of time failures and consequently activate appropriate mitigation strategies. The main goal of the application was to demonstrate the use of the Digital Twin applied to dynamic task allocation and flexible scheduling, that could effectively address failures and problems and optimize production.

For this, two different mitigation strategies were contrasted. The first strategy entailed the sequential reallocation of items on a one-at-a-time basis. In this approach, upon the detection of a necessary reallocation by the Digital Twin system, a single item is transferred between the task-executing robots for processing. After this reallocation, operations are resumed, and should another reallocation be required, it would be implemented only after the robots have completed their current cycle.

Conversely, the second strategy encompasses the reallocation of a calculated optimal batch of items between the task executers. This methodology facilitates a single comprehensive reallocation event, where all necessary items are moved at once, thereby streamlining the process and minimizing operational disruptions.

The results obtained show that the implemented Digital Twin is successful in mitigating time failures, within the application developed; they are not only consistent with the expected outcome of the research, but also highlight the significance of employing Digital Twin technology in modern manufacturing systems. Both mitigation strategies proved to be useful, although the analysis showed that reallocating an optimal batch of items yielded the best results when compared to reallocating one item at a time.

The reallocation strategies improved the Throughput and reduced the Cycle Time of the system, compared to the scenario in which a time failure occurs and the Digital Twin is not implemented to mitigate the issue. Thus, it is demonstrated that in scenarios where time failures happen, the system's ability to adjust its task allocations dynamically, through the use of the Digital Twin, ensured better overall performance and resource utilization.

The results then reveal that the use of the Digital Twin significantly improves the adaptability of production environments. By generating a virtual representation of a physical system, mirroring operating conditions and leveraging real-time data, the Digital Twin can foresee problems and act upon it by activating accordingly mitigation strategies.

The integration of the Digital Twin technology into production environments brings several advantages. As displayed in the application, it can improve the system's efficiency, mitigate issues and optimize the use of resources by balancing workloads dynamically. Moreover, from an economic perspective, it helps mitigate financial losses associated with inefficiencies in production.

Looking forward, there are promising applications for further exploration. Expanding the implementation of the Digital Twin to a larger production environment could provide a more comprehensive understanding of the potential of the Digital Twin. Additionally, more advanced algorithms for the task reallocation strategies could be developed for the Digital Twin to better adapt to occurring failures. Moreover, the Digital Twin model could be improved to better represent the physical system by including information gathered from additional data sources of the physical system.

In conclusion, the study confirms that the Digital Twin represents a significant advancement applied to manufacturing and production systems. As the technology evolves, its role in managing failures and optimizing performance is likely to become increasingly valuable.

References

[1] Grieves, M., & Vickers, J. (2011). *Digital twin: Manufacturing excellence through virtual factory replication*. In *Proceedings of the Fifth Annual IEEE International Conference on Cyber Physical Systems* (pp. 1-7). IEEE.

[2] Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., & Calinescu, A. (2022). Digital twins: State of the art theory and practice, challenges, and open research questions. *Journal of Industrial Information Integration, 30*, 100383.

[3] Grieves, M. (2003). *Digital twin: Manufacturing excellence through virtual factory replication*. Presented at the *Society of Manufacturing Engineers Conference*.

[4] Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics, 15*(4), 2405-2415.

[5] Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges, and open research. *IEEE Access, 8*, 108952-108971.

[6] Methuselah, J. (2024). Digital twin technology for smart manufacturing. *Journal of Technology and Systems, 6*(4), 52–65.

[7] Soori, M., Arezoo, B., & Dastres, R. (2023). Digital twin for smart manufacturing: A review. *Sustainable Manufacturing and Service Economics, 2*, 100017.

[8] Hamza Zafar, M., Langås, E. F., & Sanfilippo, F. (2024). Exploring the synergies between collaborative robotics, digital twins, augmentation, and industry 5.0 for smart manufacturing: A state-of-the-art review. *Robotics and Computer-Integrated Manufacturing, 89*, 102769.

[9] Röhm, B., & Anderl, R. (2022). Simulation data management in the digital twin (SDM-DT) – Evolution of simulation data management along the product life cycle. *Procedia CIRP, 105*, 847-850.

[10] Li, Y., Tao, Z., Wang, L., Du, B., Guo, J., & Pang, S. (2023). Digital twin-based job shop anomaly detection and dynamic scheduling. *Robotics and Computer-Integrated Manufacturing, 79*, 102443.

[11] Whitmore, D. (2024). Digital twins in the asset life cycle: Are we there yet? *Proceedings of the Institution of Civil Engineers - Management, Procurement and Law*. Available online 30 July 2024.

[12] Arumugam, T., Kamble, N. K., Guntreddi, V., Sakravarthy, N. V., Shanthi, S., & Ponnusamy, S. (2024). Analysis and development of smart production and distribution line

system in smart grid based on optimization techniques involving digital twin. *Measurement: Sensors, 34*, 101272.

[13] Qiu, F., Chen, M., Wang, L., Ying, Y., & Tang, T. (2023). The architecture evolution of intelligent factory logistics digital twin from planning, implementation to operation. *Advances in Mechanical Engineering, 15*(9).

[14] Zhang, F., Bai, J., Yang, D., & Liu, X. (2022). Digital twin data-driven proactive job-shop scheduling strategy towards asymmetric manufacturing execution decision. *Scientific Reports, 12*, 1546.

[15] Liu, X., Jiang, D., Tao, B., Xiang, F., Jiang, G., Sun, Y., Kong, J., & Li, G. (2023). A systematic review of digital twin about physical entities, virtual models, twin data, and applications. *Advanced Engineering Informatics, 55*, 101876.

[16] Javaid, M., Haleem, A., & Suman, R. (2023). Digital twin applications toward Industry 4.0: A review. *Cognitive Robotics, 3*, 71-92.

[17] Tao, F., Qi, Q., Wang, L., & Nee, A. Y. C. (2019). Digital twins and cyber–physical systems toward smart manufacturing and Industry 4.0: Correlation and comparison. *Engineering, 5*(4), 653–661.

[18] Yan, D., Sha, W., Wang, D., Yang, J., & Zhang, S. (2022). Digital twin-driven variant design of a 3C electronic product assembly line. *Scientific Reports*, *12*, 3846.

[19] Ayankoso, S., Kaigom, E., Louadah, H., Faham, H., Gu, F., & Ball, A. (2024). A hybrid digital twin scheme for the condition monitoring of industrial collaborative robots. *Procedia Computer Science, 232,* 1099–1108.

[20] Mazumder, A., Sahed, M. F., Tasneem, Z., Das, P., Badal, F. R., Ali, M. F., Ahamed, M. H., Abhi, S. H., Sarker, S. K., Das, S. K., Hasan, M. M., Islam, M. M., & Islam, M. R. (2023). Towards next generation digital twin in robotics: Trends, scopes, challenges, and future. *Heliyon, 9*(2), e13359.

[21] Albini, T., Brocchi, A., Murgia, G., & Pranzo, M. (2023). Real-time optimization for a Digital Twin of a robotic cell with human operators. *Computers in Industry, 146,* 103858.

[22] Fang, Y., Peng, C., Lou, P., Zhou, Z., Hu, J., & Yan, J. (2019). Digital-twin-based job shop scheduling toward smart manufacturing. *IEEE Transactions on Industrial Informatics, 15*(12), 6425–6435.

[23] Baratta, A., Cimino, A., Longo, F., & Nicoletti, L. (2024). Digital twin for human-robot collaboration enhancement in manufacturing systems: Literature review and direction for future developments. *Computers & Industrial Engineering, 187,* 109764.

[24] Li, X., He, B., Wang, Z., Zhou, Y., Li, G., & Zhu, Z. (2024). A digital twin system for task-replanning and human-robot control of robot manipulation. *Advanced Engineering Informatics, 62*(A), 102570.

[25] Yao, B., Xu, W., Shen, T., Ye, X., & Tian, S. (2023). Digital twin-based multi-level task rescheduling for robotic assembly line. *Scientific Reports, 13*, Article 1769.

[26] Zhang, Z., Ji, Y., Tang, D., Chen, J., & Liu, C. (2024). Enabling collaborative assembly between humans and robots using a digital twin system. *Robotics and Computer-Integrated Manufacturing, 86*, Article 102691.

[27] Zhang, X., Wu, B., Zhang, X., Duan, J., Wan, C., & Hu, Y. (2023). An effective MBSE approach for constructing industrial robot digital twin system. *Robotics and Computer-Integrated Manufacturing, 80*, Article 102455.

[28] IBM. (n.d.). What is a digital twin? Retrieved July 16, 2024, from https://www.ibm.com/topics/what-is-a-digital-twin

[29] Flexsim. (n.d.). *Flexsim: A powerful simulation modeling software*. Retrieved January 20, 2024, from https://www.flexsim.com/flexsim/

[30] FlexSim. (n.d.). Emulation. In *FlexSim documentation*. Retrieved July 7, 2024, from https://docs.flexsim.com/en/21.2/Reference/Tools/Emulation/Emulation.html

[31] FlexSim. (n.d.). PLC emulation. *FlexSim*. Retrieved July 7, 2024, from https://www.flexsim.com/plc-emulation/

[32] Universal Robots. (n.d.). *UR3 robot*. Retrieved August 14, 2024, from https://www.universal-robots.com/products/ur3-robot/

[33] Universal Robots. (n.d.). *CB3 series robots*. Retrieved August 14, 2024, from https://www.universal-robots.com/products/cb3/

[34] Radiant Digital. (n.d.). *Digital twin: Converging the virtual and physical worlds to accelerate transformational innovation*. Radiant Digital. https://www.radiant.digital/digital-twin-converging-the-virtual-and-physical-worlds-to-accelerate-transformational-innovation/

List of Figures

List of Tables