systems

MDPI

Article

Automated Generation of Simulation Models and a Digital Twin Framework for Modular Production

Filip Jure Vuzem [1,2], Miha Pipan [1], Hugo Zupan [2], Marko Šimic [1,*] and Niko Herakovič [1]

[1] Faculty of Mechanical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia; filip.j.vuzem@digiteh.eu (F.J.V.); miha.pipan@fs.uni-lj.si (M.P.); niko.herakovic@fs.uni-lj.si (N.H.)
[2] DIGITEH d.o.o., Cesta na Jezerca 8, 4240 Radovljica, Slovenia; hugo.zupan@digiteh.eu
* Correspondence: marko.simic@fs.uni-lj.si; Tel.: +386-1-4771-797

Abstract

This study presents the development of a Digital Twin (DT) framework that is capable of generating and adjusting simulation models of production processes and systems automatically and in real-time. A Machine Vision (MV) system is used to detect newly added or already existing production module locations and rotations, as well as changes in both their location and rotations. This subsystem primarily functions as an External database and is used for new Asset Administrative Shell (AAS) creation, housing, and data gathering, which also includes a visualization platform. Tecnomatix Plant Simulation (TPS) is used for simulation model building, simulation execution, and high-level scheduling based on work orders and technological plans. Different subsystems were integrated into the DT framework using fast and reliable communication protocols. The automation of the proposed framework significantly reduces manual intervention, thus eliminating human factors, reducing the time needed for model creation, improving simulation fidelity, and providing the fundamentals for robust connectivity within the DT framework. The findings highlight the transformative potential of this method for streamlining simulation processes and enhancing system adaptability in complex environments.

Keywords: production layout; Digital Twin framework; simulation; self-building; attributes; parameters; connectivity

check for updates

Academic Editors: Khaled Benfriha and Esma TALHI

Received: 2 July 2025
Revised: 6 September 2025
Accepted: 10 September 2025
Published: 13 September 2025

Citation: Vuzem, F.J.; Pipan, M.; Zupan, H.; Šimic, M.; Herakovič, N. Automated Generation of Simulation Models and a Digital Twin Framework for Modular Production. *Systems* **2025**, *13*, 800. https://doi.org/10.3390/systems13090800

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Today's market demands and the trends toward factories of the future require agile production systems and processes, including those for internal logistics. To meet these requirements, factories are increasingly adopting the principles of Industry 4.0, which leverage established technologies to keep production systems responsive, efficient, and adaptable to customer needs [1–3].

The development of smart factories (factories of the future) has been guided by reference architectures, namely 3D Reference Architectural Model Industry 4.0 (RAMI 4.0), which organizes key aspects of Industry 4.0 into layers and dimensions. Building on this foundation, the LASFA model was introduced as a simplified 2D representation to illustrate building blocks and their interconnections [3]. These building blocks are supported by communication technologies such as ModBus, WiFi, or 5G, while edge computing is emphasized as a core enabler for distributed systems [4].

With such frameworks in place, smart factories rely on technologies including Digital Twins (DTs), IIoT, modular production, AI, robotics, and cloud/edge computing. Among

these, DTs are central, as they connect physical systems with their digital counterparts. The concept of the DT was first introduced by Grieves at the University of Michigan in 2003 [5] and later applied by NASA, which, in 2012, formalized the paradigm for spacecraft monitoring and simulation [6]. In 2015, Grieves [5] formalized the concept in a white paper. DTs link real systems with digital environments by automating data collection, processing the collected data in simulation models, and returning insights to the physical system. Simulation models within DTs enable bottleneck detection, what-if scenario analyses, and production and logistics optimization [7].

While essential in manual optimization, agile and lean methodologies remain relevant in digital optimization as well. Approaches such as Value Stream Mapping (VSM), Kanban, 5S methods, and layout optimization provide the foundation for optimizing processes and operations [8]. Their integration into DTs through Digital Twin as a Service approach enables higher digitalized optimization [9]. Lean improvements, however, must first be implemented physically; otherwise, inefficiencies are digitalized as well [10]. Building on this principle, Digital Lean and Digital Value Stream Mapping (DVSM) extend lean practices into digital domains, offering new tools for waste detection and process analysis [11].

While lean and digital optimizations focus on processes, the structure of the production system must also support flexibility. This is achieved through modular production, where modules can be combined in different sequences to manufacture various product variants or entirely different products. The Plug&Produce principle enhances this by allowing modules to be quickly connected or reconfigured, enabling fast layout changes and material flow adjustments. To fully exploit modular production, DTs and expert systems (ESs) with integrated AI can be applied [12]. This reduces throughput times, accelerates delivery, and allows for real-time scaling of production [13]. Modular systems can be organized as standalone machines, production cells, or fully connected lines arranged in L, U, or linear configurations as shown in Figure 1. The arrows show the material flow that can be performed by mobile units such as AGVs, AMRs, forklifts (dashed arrows) or in-line conveyers assuring the material flow across connected production modules in the production line (solid arrows).

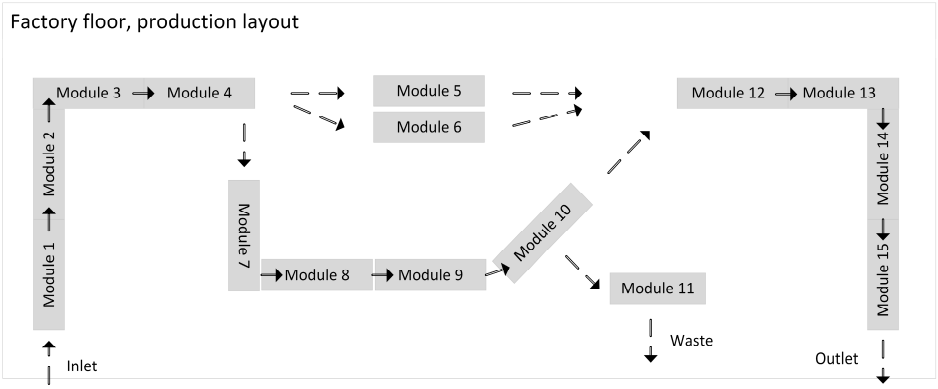


Figure 1. Example of a modular production layout.

As market demands and trends lean toward modular and agile production, our approach focuses on technologies that will allow for such production to function and to outperform traditional production.

The rest of this article is structured in the following way: The Self-building simulation models and Digital Twins section are focused on two main areas: adaptation of parameters of simulation models and adaptation of the structure of simulation models. The Scheduling in Digital Twins sub-section describes the general view of working principles within Digital Twins and supports the importance of the two main areas. At the end of this section,

the main gaps, research questions, and potential scientific contributions are highlighted. The Materials and Methods section describes our approaches for identifying the data needed and developing the necessary systems/subsystems within the DT framework. The Results and Discussion sections present the results and validation of our research work. In the Conclusions and Future Directions section, the scientific contribution, potential future research work for integration into Digital Twins, and the potential for scaling up are discussed.

2. Self-Building Simulation Models and Digital Twins

The idea of DTs starts with digitalizing the physical assets and gathering, structuring, and preparing all the important data that can be used as input. The main challenge here is to adapt the simulation models, their parameter setups, and the Digital Twin's functionality efficiently in real-time and without human intervention. Studies concerning the adaptation of simulation models to dynamic changes in real systems have been summarized in literature reviews. The reviewed research can be categorized into two categories: adaptation of parameters of simulation models [a], encompassing studies where only the parameters of elements in the simulation models are dynamically adapted (see Section 2.1); and adaptation of the structure of a simulation model [b], encompassing studies in which the authors discuss the adaptation of the entire model or model structure (see Section 2.2). This includes updating parameters and, for instance, changing the positions of objects in the model, deleting them, or adding new objects. There are several main development paths for Digital Twins with simulation model changes (hereinafter, structural Digital Twins), with some of these concerning the structural adaptation of products [14] and others delving into the structural adaptation of machines, tools, and processes. Figure 2 illustrates the timeline of advances made in DTs, focusing on the two categories mentioned above.

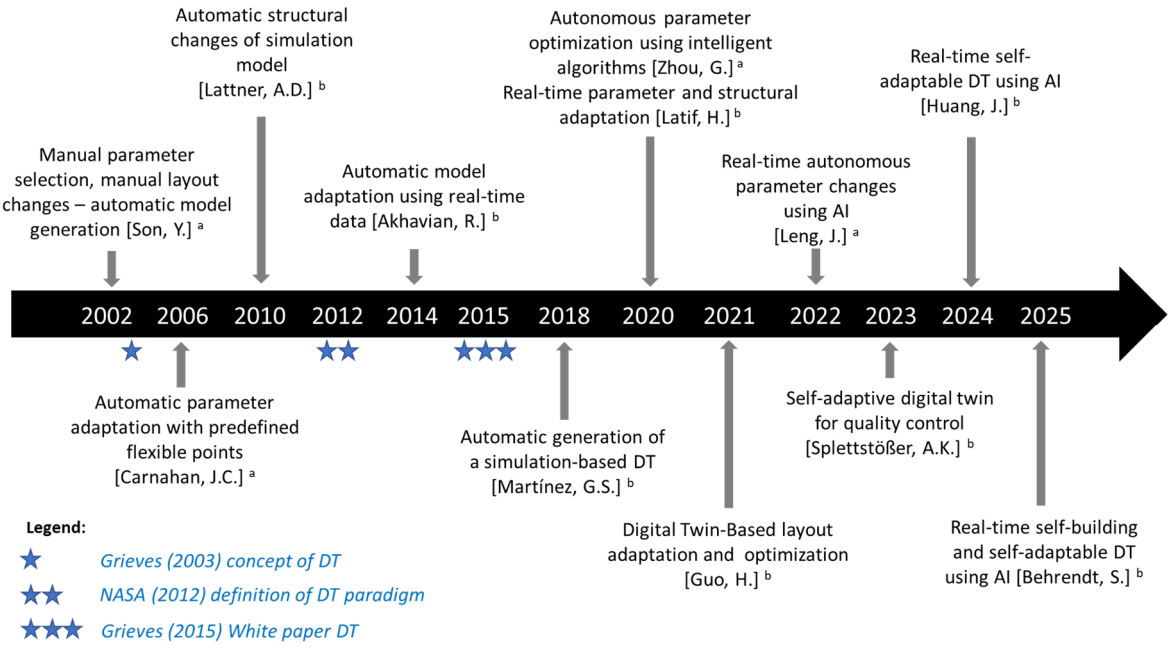


Figure 2. Timeline of Digital Twin (DT) development, with a focus on the self-adaptability of simulation models [5,6,15–26].

2.1. Adaptation of Parameters of Simulation Models

In 2002, Son, Y.-J. et al. [15] proposed a manual system for shop-floor planning and simulation parameter changes. In their 2006 research, Carnahan, J.C. et al. [16] highlighted the challenges involved in automatically adjusting simulation models when experimental

data suggests that changes are necessary. Their approach involved semi-automatic parameter adjustments, with operator intervention to determine which parameters needed to be modified. In 2020, Zhou, G. et al. [17] emphasized the role of Smart Behavior Digital Twins in maintaining alignment between digital and physical systems. It allowed for real-time parameter adaptation and represented a significant advancement in the field of parameter adaptability. In 2022, Leng, J. et al. [18] upgraded the concept presented in [17] by introducing AI into real-time parameter-adjustable Digital Twins. This allowed DTs to quickly respond to unexpected disturbances and variations in production processes.

From the research, it is evident that developments in the field of simulation model parameter adaptation range from fully manual methods through to automated parameter adjustment and tuning, even in real-time, and to the use of AI for automatic detection and parameter modification.

2.2. Adaptation of the Structure of Simulation Models

In their 2010 study, Lattner, A.D. et al. [19] presented a novel approach aimed at surpassing parameter optimization to enable automatic structural changes in simulation models, thereby generating model variants. In their 2014 paper, Akhavian, R. et al. [20] discussed the implementation of a software framework for the automatic adaptation of discrete event simulation models to dynamic changes using data mining methods and unsupervised learning in a construction environment. In 2018, Martínez, G.S et al. [21] developed a framework for the automatic generation of a simulation-based Digital Twin by extracting structural data and machine parameters. While the simulation model is fixed once created, the parameters can be adjusted in real time during simulation.

Latif, H. et al. [22] (2020) developed a DT that combined simulation, real-world data, and reinforcement learning for output predictions. In 2021, Guo, H. et al. [23] developed a framework for continuous layout adaptation during simulation based on incoming work orders, which included material distribution and tooling route optimization. In 2023, Splettstößer, A.K. et al. [24] upgraded an already existing Self-Building Digital Twin (DT) architecture with a Quality Assessor component. The enhanced architecture follows the MAPE-K (Monitor–Analyze–Plan–Execute over a shared Knowledge base) loop, enabling the DT to autonomously monitor system behavior and detect quality deviations. In 2024, Huang, J. et al. [25] developed a process for the automated reconfiguration of smart tools in smart manufacturing systems using Deep Q-Network (DQN), a type of deep reinforcement learning. This system allows for automated real-time tool optimization and work-order scheduling. In 2025, Behrendt, S. et al. [26] developed a real-to-simulation framework for the automatic generation of simulation models to support the creation of Self-Building Digital Twins. The approach uses graph-based methods to extract a system structure and machine learning techniques to model dynamic behavior from sensor and process data.

It is evident that the development of structural Digital Twins has progressed over time from the manual building of simulation models and the automation of simulation models and Digital Twin creation to the real-time building of simulation models and Digital Twins based on live data and, finally, to the use of AI-based approaches for automatic Digital Twin generation. Previous works have described (a) the adaptation of parameters and (b) the adaptation of structure, providing a solid basis for our development direction. Although recent research has focused on the direction of implementing AI in Digital Twins, several areas of self-building and adjustable frameworks remain unexplored. We intend to surpass previous works by creating a Digital Twin framework with self-building simulation model capabilities, which is an upgrade of the work in [19,21–24,26]. Most articles focused only on the self-building or adjustable aspects of the simulation models in the DT framework while overlooking other aspects, which we intend to cover in full.

2.3. Scheduling in Digital Twins

One of the main challenges for factories is scheduling and planning. Scheduling is mostly conducted in ERP systems or programs outside DTs, which becomes a problem when we have an adaptable DT framework. Scheduling is either static or dynamic; however, it is often not used as an input for DT optimization. It can be used for material flow planning or logistics planning. For scheduling, multiple approaches are used; these range from the use of exact algorithms to heuristic rules, metaheuristics, and machine learning approaches. In [27], the authors used machine learning approaches, heuristic rules, and metaheuristics to dynamically assign storage places and to select the shortest route for picking orders. The principles and methodology are well developed and can be applied in material flow optimization as well. As scheduling is such a challenge for factories, several approaches have been researched to connect it with Digital Twins; however, research studies on adaptable DTs and self-building and those involving adjustable simulation models are still rare.

2.4. Research Gaps and Scientific Contributions

To summarize, researchers focusing on Digital Twin frameworks with real-time adaptable simulation models have made notable progress; however, several critical limitations and open research challenges remain:

- The process of capturing the factory floor layout and integrating it into a Digital Twin system is still underexplored and lacks standardized methodologies;
- There is a need for clearer identification, structuring, and processing of the data required for Digital Twin operations—specifically, determining which data is necessary for simulation, how it should be formatted, and which communication protocols are best suited;
- Although self-building and self-adjustable simulation models have been studied, several promising approaches remain uninvestigated, particularly in terms of real-time responsiveness and integration with factory floor systems;
- The integration of scheduling into the DT framework is severely under-researched.

To address these identified limitations, this study makes the following key contributions:

- A Machine Vision system was developed and implemented to enable automatic detection and digitization of the factory layout;
- The essential data required for both the self-building capability of the simulation model and the runtime operation of the simulation model were systematically identified and structured;
- A complete data pipeline was designed to enable seamless communication between the Machine Vision module, other factory floor modules, and the Digital Twin—ensuring continuous data flow in both directions;
- A real-time Digital Twin system was developed, capable of autonomously constructing and adjusting its internal simulation model in response to dynamic changes on the factory floor;
- Scheduling was integrated into the DT, whereby rescheduling is carried out when a layout changes, real system module errors are detected, or new work orders arrive.

3. Materials and Methods

A Cyber-Physical System (CPS) consists of a physical system, sensors, connectivity protocols, and a virtual replica. As part of the CPS, Digital Twins represent a virtual replica of a real system with integrated intelligence and a return connection to the real system. Our

research combined the approaches of parameter self-adaptation and the self-building of a simulation model.

The methodology is divided into several steps: (1) describing the production modules and their potential for creating different production layouts; (2) defining the data and attributes of the production modules; (3) developing a concept of a Digital Twin framework that allows for the automatic self-building of simulation models; (4) developing methods for preparing the External and Internal databases for simulation models and methods for data manipulation; (5) developing new algorithms for automatic simulation-model building and production objects' interconnection (material flow); and (6) the testing and verification of a Digital Twin framework by considering different production layouts.

3.1. Modular Production in the Laboratory Factory Floor

In this paper, the laboratory modular production with internal logistics is used as a case study. Building a simulation model from scratch or modifying it to replicate the real industrial layout can be a time-consuming process, which leads to the need for a self-building simulation model. With that in mind, our concept is based on modular production, where each module can be arbitrarily placed on the factory floor. Production modules can be used individually or arranged together to form a production line or production cell. In this paper, we consider production modules with the following operations: handling, assembly, warehousing, and logistics. The Plug&Produce (P&P) design is implemented to achieve the efficient interconnection of production systems in different layout configurations. Figure 3 shows one of the possible production layout configurations for our investigation. Since we have mobile production modules, we are able to create different production layouts according to the production plan and the products that have to be assembled. In Figure 3a, one of the use cases is shown, which describes individual production systems, production cells, or production lines with several interconnected modules. The arrows on modules showcase possible moving direction of products in/out of modules. Figure 3b shows a photo of the physical testbed, a real laboratory environment showing modular production systems placed on the laboratory floor and a Machine Vision system located on the laboratory roof. To characterize the modules' existence on the laboratory factory floor, at least their location (X, Y), orientation (φ), and the module type (QR codes) should be recognized. For this purpose, we used a Machine Vision (MV) system, which is described in more detail later (see Section 3.3). The camera view has its own coordinate system and is synchronized with the coordinate system of Tecnomatix Plant Simulation (TPS).

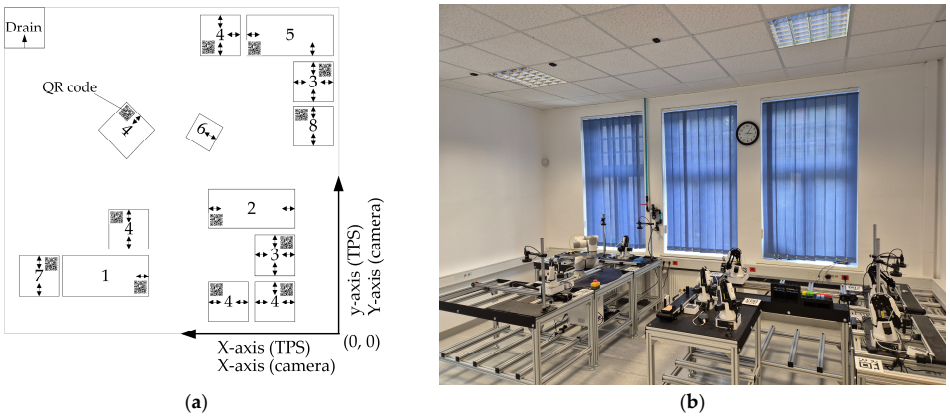


Figure 3. Modular production: (**a**) production layout configuration and (**b**) photo of real systems and the laboratory environment.

A short description of their characteristics and the main components of the production modules used in our use cases are described in Table 1. For experimental purposes,

Dobot development automation kits and components were used to develop and build different production modules (Modules 1–7). Raspberry PI was used for each production module as the master control unit to synchronize the operations, and WiFi was used for communication and data transfer between production modules.

Table 1. Production modules used in this study.

Module No.	Module Title	Description
1	Warehouse 1	To store colored cubes and to accommodate the need for assembly. Main components: Robot sliding rail with Magician robot arm, storage places and containers, and a loading/unloading place. It has one entry/exit point prepared for connection with AMR (loading and unloading of the pallets or parts).
2	Warehouse 2	To store engraving cards. Main component: Robot sliding rail with Magician robot arm, and storage places and containers. It has one entry/exit point prepared for connection with AMR (loading and unloading of the pallets or parts).
3	Robot cells 1 and 2	Handling operation between production modules; serves as a central unit with four potential connection points for production modules. Main components: CR3 collaborative robot and controller unit, storage places, and several robot grippers.
4	Robot cells 3, 4, 5, 6, and 7	Modules 3, 4, and 5 are used for assembly operations, and modules 6 and 7 are used for quality control. Main components: Magician robot arm, conveyor belt with optical sensors, robot vision with HD color industrial camera, storage places for colored cubes, and engraving cards or pallets. It has one exit point and one entry point prepared for AMR to allow for the loading and unloading of parts and pallets onto conveyor belts or into storage places.
5	Robot rail	Module for transferring the parts from one production module to another. Main components: Robot rail with Magician robot arm, and loading/unloading system for pallets and parts. It has one entry point and one exit point, which allows for connections to other production modules or AMR.
6	AMR	Used for internal logistics between production modules. Main components: Turtlebot3 AMR with conveyor belt and optical sensor to recognize pallets and parts. It has one entry/exit point, which is used to connect with other production modules.
7	Engraving	Laser engraving to engraving cards. Main components: Two Magician robot arms (one for engraving and one for handling), conveyor belt, engraving station to position, and engraving cards. It has one entry point and one exit point prepared for AMR (loading and unloading the pallets; cards).
8	Sorting	Module for sorting, positioning colored cubes, and preparing the pallets for the production modules (Robot cells 3–7). Main components: Magician robot arm, conveyor belt, and sorting station. It has one entry point and one exit point prepared for connection to other production modules or AMR.

3.2. Data and Attributes of Production Modules

A set of attributes characterizes each production module; some of them are common to all modules, whereas others are specific to individual modules. Each attribute can take on various values, represented in different data formats. To ensure systematic recognition and tracking, we decided to record all attributes, as detailed later in Tables 2 and 3, in an External database.

Table 2. General data and attributes of production modules.

General Data	Location and Rotation	Other Attributes	Visualization (Sim)	Visualization (Real)
ID [str]	location *X* [real] *	module status [bool] **	working time [time]	working time [time]
name [str] *, **	location *Y* [real] *	state of the process [json]	waiting time [time]	waiting time [time]
width [real]	rotation [real] *	operation [int]	down time [time]	down time [time]
length [real] **		operation description [str]	throughput [int]	throughput [int]
height [real]		start of operation [timestamp]	working cycle [time]	working cycle [time]
/		end of operation [timestamp]	type of error [str]	type of error [str]
/		errors [json]	/	/
/		is OK [int]	/	/
/		conveyor belt speed [real]	/	/

Legend: *: data needed for creating objects in the simulation model; **: data needed for the execution of the simulation.

Table 3. Special data and attributes of production modules.

Special Attributes	1	2	3	4	5
Robot cells 1 and 2	Number of grabbed products [int]	Conveyor belt speed [real]	Product grabbed [bool]	Product dropped [bool]	
Robot cells 3, 4, 5, and 6	Conveyor belt speed [real]				
Quality control	Is it OK? [bool]	Conveyor belt speed [real]			
Warehouse	Current warehouse inventory [int]				
Conveyor belt	Number of grabbed products [int]	Conveyor belt speed [real]			
Engraving	Conveyor belt speed [real]	Engraving finished [bool]			
Sorting	Sorting system ready [bool]	Conveyor belt speed [real]			
AMR	Is the conveyor working [bool]	Safety switch closed [bool]	AMR speed [real]	AMR direction [int]	Conveyor belt speed [real]

To facilitate the recording of attributes, these are classified as general data and attributes (Table 2) and special attributes (Table 3). Table 2 contains several important pieces of data and sub-attributes, such as General data in Column 1, which includes attributes related to the identification and basic dimensions of the module; "Location and rotation" in Column 2, which lists attributes describing the position of the module, and these that are collected using an MV system; Column 3 ("Other attributes") includes the operational attributes representing the module's current state during its functioning; Columns 4 and 5 ("Visualization (sim)" and "Visualization (real)") provide the data necessary for the visualization and monitoring of the production systems and processes. This data enables the monitoring of both real-time module performance and simulated behavior, illustrating how the module is expected to perform.

Table 3 outlines the special attributes of each module, providing the additional data necessary for the comprehensive "inventory" and proper characterization of each module. Not all of the data presented in Tables 2 and 3 is currently used for simulation, as all of the modules were treated as "black boxes"; however, all the prepared and monitored data can be used for the future expansion of the modules, where each elementary operation will be simulated. Based on our research questions, we defined the initial potential data and attributes needed for the self-building of simulation models by marking these with *, and the data and attributes needed for running the simulation model were marked with **. The special attributes listed in Table 3 (from 1 to 5) were gathered from real-world modules and subsequently stored in a database for further utilization. All monitored and recorded data was systematically prepared for future module expansions and for developing the Asset Administrative Shells (AASs), as well as the final DT of a given production process. These expansions aim to simulate, monitor, and analyze each module's elementary operations in detail.

As shown in Tables 2 and 3, each attribute is accompanied by its respective data format, indicated alongside the attribute name in brackets. The inclusion of data formats is crucial for accurately capturing and presenting digital information. The supported data formats include the following:

- [timestamp]: Timestamp is used to record time in a specific format (Day/Month/Year, Time) to monitor module activities, orders, and other tasks.
- [bool]: Boolean, where the value can be true or false; it is used for the attributes with only two states.
- [int]: Integer represents the value without decimal points; used for counting.
- [real]: Real is a number that includes decimal values. It is used for a more precise representation of values.
- [str]: String is a sequence of characters such as letters, numbers, and special symbols. It is used for attributes that represent a description.
- [time]: Time measures time during the module operation, where the basic unit is a second.

Attributes gathered from the modules are categorized as either static or dynamic, with dynamic attributes changing over time. For static attributes, the Internal database was structured and prepared, which served as a starting point for static data gathering. These static attributes remain constant and serve as static basic data for the production modules. The static attributes are as follows:

- Object detection time [time];
- Conveyor belt direction [boolean];
- Length of conveyor belt [real];
- Loading time onto the conveyor belt [time]; **
- Unloading time from conveyor belt [time]; **

- Electric energy required to move the robot arm [real];
- Electric energy required to move the conveyor belt [real];
- Electric energy required for the Machine Vision system [real].

These static data and attributes encompass all the currently gathered data for each production module. They are used for the Digital Twin framework, which includes several intelligent algorithms. The other data can be used as possible inputs for a Digital Twin of production processes and systems.

3.3. Development of the Digital Twin Framework

Figure 4 shows the system architecture and main subsystems and is used as the base for development of a Digital Twin framework. The main subsystems are (1) a laboratory factory floor with real production modules; (2) a Machine Vision system that consists of several cameras to cover the entire factory floor; (3) Thingsboard as an External database and a Human–Machine interface (HMI); and (4) a Tecnomatix Plant Simulation tool with an Internal database that represents the virtual environment for performing simulations of production processes. Meanwhile, Figure 5 shows detailed view of each subsystem, their inner working, the interconnections between subsystems and their roles within DT framework. Each subsystem is enclosed in dashed colored bracket and the connections between them are presented with colored arrows.

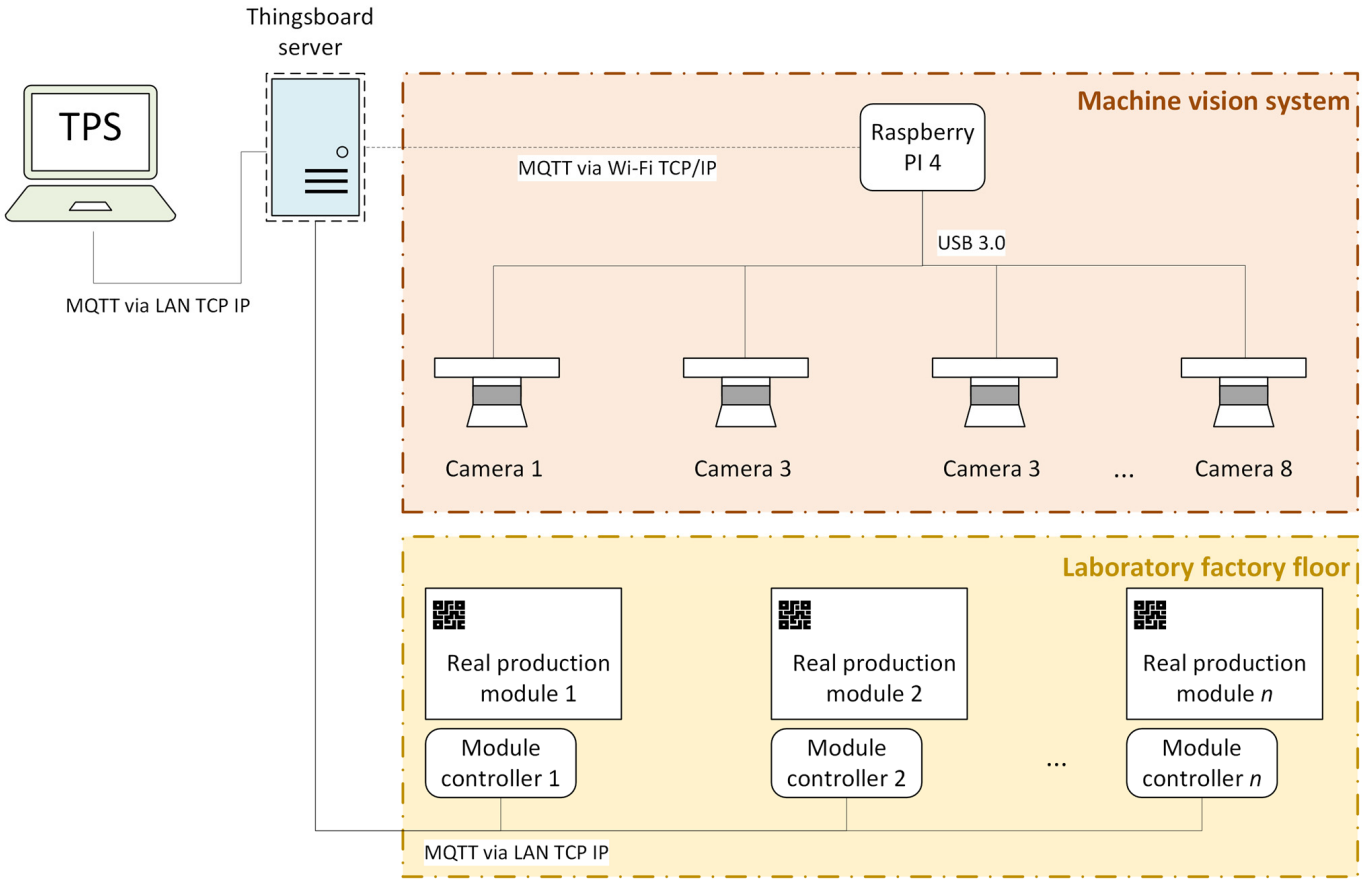


Figure 4. System architecture diagram and schematics of the subsystems used.

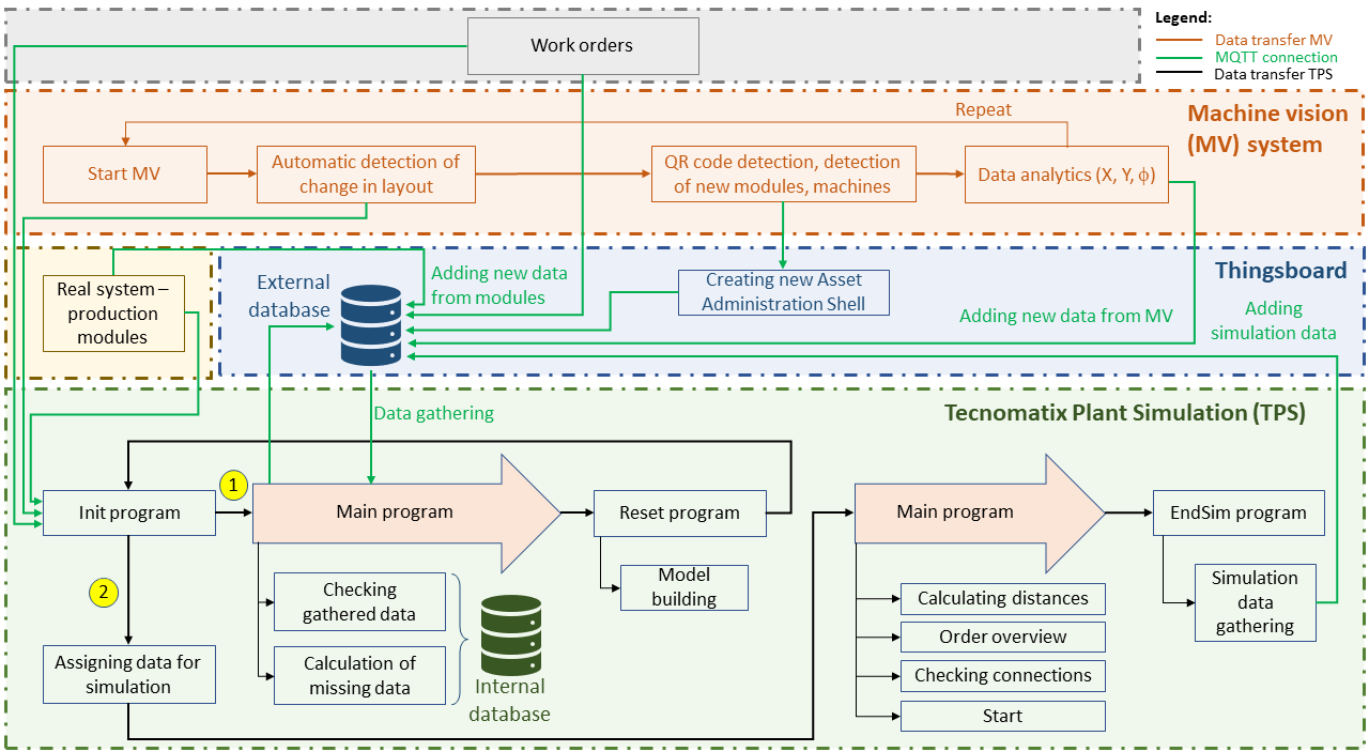


Figure 5. Digital Twin framework and all the important subsystems.

The real state of the production modules on the laboratory factory floor is recognized in real time by the Machine Vision system through the detection of QR codes placed on the modules, which defines the ID of each module, their location, orientation, and entry/exit points. Recognition of the entire laboratory factory floor requires sufficient image quality and a wide enough camera field of view to cover the area effectively; therefore, eight cameras were used for our case study. Cameras, chosen for the Machine Vision system, were equipped with a Sony IMX708 sensor (Sony Semiconductor Solutions Corp., Tokyo, Japan), as this offers a resolution of 12 MP, supports image capture up to 4608 × 2592 pixels, and features automatic focus. The camera's field of view is 66° horizontal and 61° vertical, with enough resolution to recognize QR codes on production modules. For the synchronization of multiple cameras, a Raspberry PI 4 (Raspberry PI Ltd., Cambridge, UK) and a multi-camera board (V2.2, Arducam Technology Co., Ltd., Nanjing, China) were used.

The Thingsboard platform serves as the main External database as well as a visualization platform. Data gathering in Thingsboard works in two modes: SQL (for sub-5000 data points received per second) and NoSQL (for over 5000 data points received per second), which allows for efficient scalability. Data from the MV system, as well as from production modules, is stored in asset administrative shells (AASs) inside the External database. From the MV module, the creation of new AASs is carried out if new production modules are recognized. In [28], automatic AAS creation was conducted using data-enriched 3D CAD drawings. In our use case, new AASs are created using MV and QR code reading; although it should be noted that this AAS is merely a shell, prepared to be populated with data from production modules themselves.

The simulation model was built using the Tecnomatix Plant Simulation 2404 (TPS) platform, which allows for real-time connectivity with different programs, the creation of simulation models, and parametric adjustable modules, as well as a return connection to the real system. TPS incorporates discrete event simulation (DES) to model and analyze complex production systems. In DES, events occur at specific points in time, with simu-

lations advancing only during these events. For example, when a part enters a module, an event is triggered, and another event is triggered when the part exits. The time between these events represents the module's operation time. In our case, TPS starts by gathering data from the External database and the real system. Once the data from the real system is collected, it is transferred to the simulation model through the Internal database, which is used to manage and analyze the data, ensuring synchronization between the real system and its digital counterpart [29,30]. The Internal database created in TPS has no real limitations in scalability. The scalability of the Internal database is limited only by TPS itself as well as hardware limitations of the system TPS runs on. After all the data is gathered and analyzed in the Internal database, the simulation model is created (marked as 1 in TPS, Figure 5). Afterwards, the following steps are conducted: workorder sequence generation, material flow optimization and feedback generation (marked as 2 in TPS, Figure 5.

The principle for the material flow optimization algorithm is based on work orders and the technological plan of products. Based on the gathered technological plan of the products, which specifies what operations are needed for the creation of a specified product (based on the known functionality of modules), the algorithm optimizes the material flow.

Lastly, data from the simulation model is returned to the External database for HMI application. The returned data from TPS encompasses simulation data, which includes a high-level scheduling plan.

Work orders, alongside real production modules and layout changes detected by the MV system, are the main input for the self-building and adjustable simulation model execution. Once new work orders are created, they are saved in the External database (Thingsboard), and a prompt is sent to TPS to start rescheduling. A similar principle is applied to changes in layout, detected by the MV system, and errors in production modules.

The real system modules and the MV system use Raspberry Pi and run Python version 3.11.2 code, allowing for a wide range of communication protocols. As a well-established platform, Tecnomatix Plant Simulation also allows for a similar range of standard communication protocols as version 3.11.2 code. The Thingsboard platform has a smaller range of available communication protocols (MQTT, HTTP, CoAP, SNMP, and LwM2M). Thus, our choice of connection between the main components of the Digital Twin framework was the MQTT protocol. This protocol was selected as it supports rapid data transfer and the handling of up to 1 million messages per second and provides high reliability with three levels of quality-of-service (QoS). It is also bandwidth-efficient, making it ideal for large-scale systems. However, latency problems were still noticed due to both the underlying medium for message transportation and MQTT itself when higher QoS was used, adding to server processing delays. WiFi caused the main latency problems, as it allows for a communication speed of only between 5 and 20 ms and is prone to interruptions.

The flow of data within the DT framework using the MQTT protocol is extensive, as all subsystems are interconnected. Data flows between the main components of the Digital Twin framework are the following:

- New work orders are saved in the External database (Thingsboard);
- Work orders send a prompt to TPS about needed rescheduling;
- The MV system sends layout data to the External database (Thingsboard);
- The MV system sends a prompt to TPS about layout changes;
- Production modules constantly stream operational data to the External database (Thingsboard);
- Production modules send a prompt to TPS about critical errors, such as module breakdown;
- TPS gathers data from the External database for the MV system as well as production modules;
- TPS returns the simulation results and optimized scheduling plan to the External database (Thingsboard).

The presented methods and concepts of the DT framework provide a solid foundation for the development and direction of a true self-building and structurally adaptable Digital Twin, where work orders are the driving force of the entire framework. The work orders send a prompt to the Self-Building Digital Twin framework, which creates a layout, checks new work orders, and then, based on those work orders, refines the layout to achieve an optimal layout for the sequence of work orders given. This is carried out iteratively when new work orders are added to the queue. At the same time, based on optimal layout for the work orders, optimal scheduling is created, using advanced scheduling techniques such as metaheuristics with heuristic rules or AI approaches.

3.4. Data Manipulation and Usage for Digital Twin Framework

The DT framework relies on well-structured data and a robust connection to the real system or database to function effectively. The connection is established using the MQTT protocol, which enables the transfer of all relevant data necessary for the initialization of objects in the simulation. Additionally, some of the data collected from the External database is crucial for the proper execution of the discrete event simulation used by Tecnomatix Plant Simulation (TPS). In order to guarantee proper data, we propose three main steps.

Step 1: Data Collection and Processing Workflow

The collected data from the real system is stored in the Thingsboard database, which serves as a central repository for all information, including the specifications, dimensions, and operational parameters of the production modules. When data is transmitted through the MQTT protocol, it is encoded in JSON format for efficient and reliable communication. Tecnomatix Plant Simulation receives the data in JSON format, unpacks it, and writes the information for each module into its Internal database in string format. The data is gathered from each operating module and the Machine Vision system (MV). MV captures images of the real system's layout and identifies modules by reading their QR codes. It uses this information to determine the type of module, its specifications (such as width, length, and name), and its precise location and orientation.

Step 2: Data Format Analysis and Conversion

To ensure that the proper data is used within the DT framework, an algorithm was developed to analyze the incoming data and convert it into the appropriate format (e.g., real, integer, or Boolean). This ensures that all data maintains its intended structure and accuracy for use within the simulation. After analysis and conversion, the data is consolidated into a unified Internal database.

Step 3: Integration with the TPS

The Internal database within TPS serves as a central hub for storing all processed data. It supports the initialization of simulation models and provides a basis for the DT framework to execute discrete event simulations. These simulations allow for the analysis of system behaviors, performance metrics, and the identification of potential improvements.

3.5. Algorithms for Automatic Model Building and Material Flow Optimization

To achieve self-development/self-building of a simulation model, three main algorithms that represent the core of the DT framework are proposed. The optimization platform represents the intelligence of the DT framework. This platform includes algorithms for the self-building of the simulation model. Algorithm 1 places the production modules in simulation model, Algorithm 2 creates interconnections between modules and Algorithm 3 optimizes material flow.

Algorithm 1 is responsible for positioning all objects in the simulation model, replicating their positions as observed from the real-world system (Figure 6). These positions are

scaled appropriately to fit the dimensions of the simulation model. This component of the algorithm is referred to as the "Program for Self-Building." It ensures that the simulation model accurately mirrors the layout of the physical system, forming a basis for further simulation and analysis.

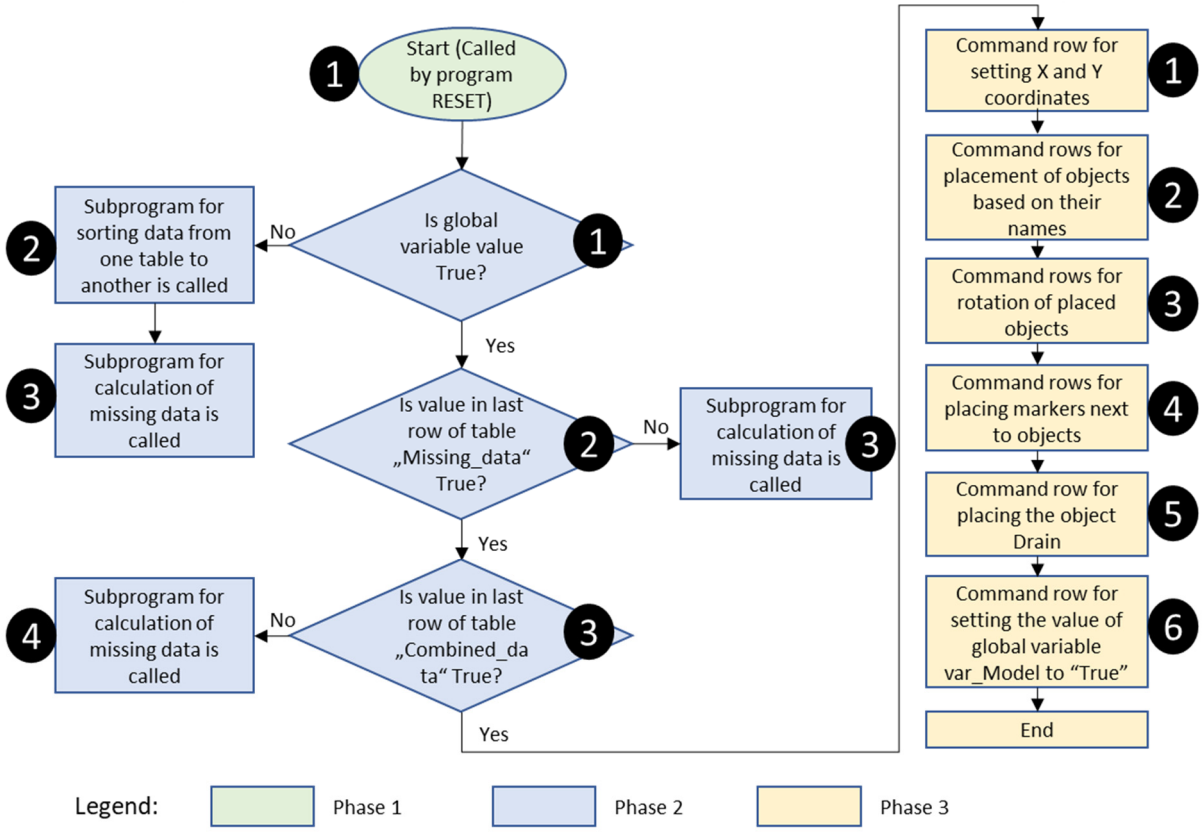


Figure 6. Block diagram showing the program for the self-building of the simulation model.

Algorithm 1 operates in several sequential phases, as illustrated in the block diagram. The initial phase consists of system initialization, whereby all existing data is cleared from the Internal data tables. This step includes the integration of a failsafe mechanism to ensure that any potentially corrupt or unnecessary data is removed, establishing a clean and stable starting point for execution. Phase 2 consists of checking whether all the essential and required data is available (e.g., object coordinates, rotations, module names, etc.). This verification acts as a prerequisite for continuing to the next phase. If all the required data is present, the algorithm proceeds to the third phase, which involves constructing the simulation model.

In step 1 of phase 1, the algorithm performs an initialization by clearing all the data from the Internal database. In phase 2, step 1 checks whether one of the global variables has changed its state and, based on that state (True/False), decides which branch of programs it will follow. If this global variable is set to False, the program follows on to steps 2 and 3, where it sorts the received data from the External database and calculates the missing data. If the variable is set to True, then it proceeds to another decision: whether the missing data has already been calculated. If it has not yet been calculated, then step 3 calls a subprogram to calculate the missing data. If, however, the data has already been calculated, then another decision is made, where the sorted Internal database is checked as per step 3. In step 4, either a subprogram for the calculation of missing data is executed or the program goes on to the third phase: the creation of a simulation model. Step 1 of phase 3 consists of selecting the correct X and Y coordinates for each module, step 2 consists of placing objects (real

system modules) in the simulation model, and step 3 consists of rotating the objects based on the rotation data from the Internal database. Step 4 places markers next to the entry and exits of objects for AMR navigation, step 5 places a Drain object into the simulation model to act as the final infinite storage, and step 6 changes the state of the global variable for the successful building of the simulation model to True.

For each type of production module, a corresponding object is instantiated when the simulation model is being built. The key portion of the program involves the following two code lines:

Line 1: `obj_class: = str_to_obj (‘‘.Resources.’’ + Combined_Data [2,i] + ’’Pool’’)`, where the type of object to be built is determined by reading its information from the Internal database named “Combined_Data.”
Line 2: `obj: = obj_class. createObject (.Models.Model, 0,0, Combined_Data [0,i])`, where the object is placed into the simulation model. In this step, the two zero (“0, 0”) values represent the X and Y coordinates, while the final part specifies the name of the object as stored in the Internal data table.

This systematic approach ensures that each module is accurately represented in the simulation model, maintaining alignment with the real-world system and laying the groundwork for further simulation and analysis. The object placement begins from the upper-left corner, which serves as the coordinate origin of the simulation model’s frame (Figure 7). This coordinate system ensures consistent placement of objects within the model, aligning their positions with the real-world layout and maintaining spatial accuracy during the simulation-building process.

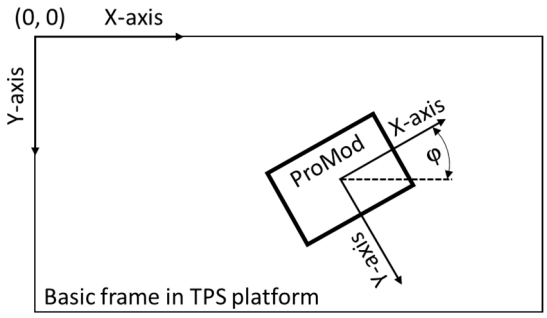


Figure 7. Basic frame in the TPS platform and its coordination system.

For every device in the simulation model, except for AMR-pool (which serves as the starting point for AMRs), markers must be created. Markers are special objects in the Tecnomatix Plant Simulation designed to assist AMRs in navigating the simulation model.

To ensure that the objects in the simulation model align with their real-world orientation, the program includes a line of programming code, `obj.objectAngle: =Combined_Data [14,i]`, to rotate the objects based on their actual rotation in the production system. As TPS uses radians for rotation, the rotational data—which is sent in degrees—must be converted to radians. This step ensures that the objects are accurately oriented within the simulation model, maintaining consistency with the real production system.

All rotations in TPS are performed clockwise, with the starting position being parallel to the X-axis. To facilitate accurate representation and analysis of the production flow, a programming object called “Drain” was created. It serves as the endpoint for all material flow within the simulation model and represents the output of the production line.

This addition allows for the tracking and analysis of production metrics, such as the total number of parts produced, their processing times, and other performance indicators. By

incorporating the Drain object, the simulation model provides a comprehensive overview of the production system's output and efficiency.

Algorithm 2 is responsible for calculating distances between production modules and creating connections between them (Figure 8). The algorithm uses the data and results from Algorithm 1 as inputs to calculate the distances between production modules. While Algorithm 1 creates all objects, including markers representing the potential inputs/outputs and the connections in the simulation model, Algorithm 2 refines the model by removing unnecessary markers and establishing connectors between objects. Connectors function similarly to markers in the simulation environment but serve a distinct operational role. Whereas markers support AMRs in localization and navigation within the simulation model, connectors are responsible for managing material handoffs between production modules, enabling seamless and continuous material flow throughout the system.

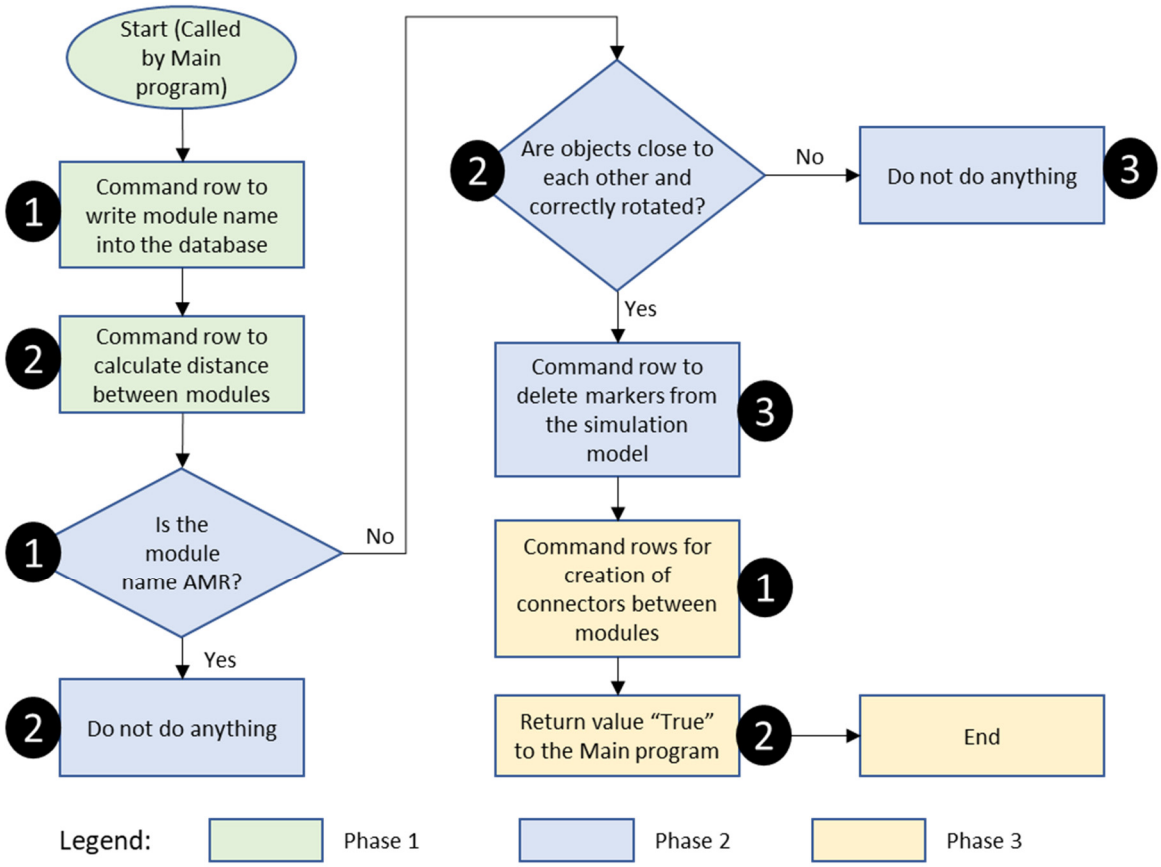


Figure 8. Block diagram showing the program for calculating distances between production modules.

The operation of Algorithm 2 can be summarized by three main phases.

Phase 1—Distance Calculation: In step 1, the program creates a matrix in the Internal database with the names of all the objects. In step 2, it calculates the distances between all objects placed in the simulation model, excluding markers.

Phase 2—Marker Refinement: In step 1 of phase 2, the algorithm loops through all the objects, and a decision is made. It checks whether the object is called AMR. If it is, then step 2 involves doing nothing; if it is not, then the algorithm checks whether two objects are close enough and rotated correctly for connector creation. If the answer to that is no, then step 3 enacts nothing; otherwise, step 3 deletes the markers between the objects.

Phase 3—Connect or Creation: The first step in phase 3 creates connectors between two objects whose markers were de-leted in step 3 of phase 2. When the connectors for all the relevant modules are created, step 2 of phase 3 consists of returning the

value True to the main program, which notifies it that the simulation model has been successfully created. The criteria for creating a con-nector in step 1 of phase 3 involve the following:

- The calculated distance between the two objects, as well as their position relative to each other in the simulation frame.
- The type of object: For example, warehouses have a unique configuration where the entry and exit points are in the same location (as shown in Figure 9). This must be accounted for during connector placement to ensure logical material flow.
- The rotation of the object: The program checks the rotation of the objects to determine the alignment of their entry and exit points. If the exit of one object is not correctly aligned with the entrance of another, a connector will not be built. This ensures that material flow follows realistic and logical pathways in the simulation model.

By automating these processes, Algorithm 2 ensures that the simulation model accurately represents the real production system while maintaining efficiency and functionality. This refinement allows for a more effective simulation of material flow and system operations.

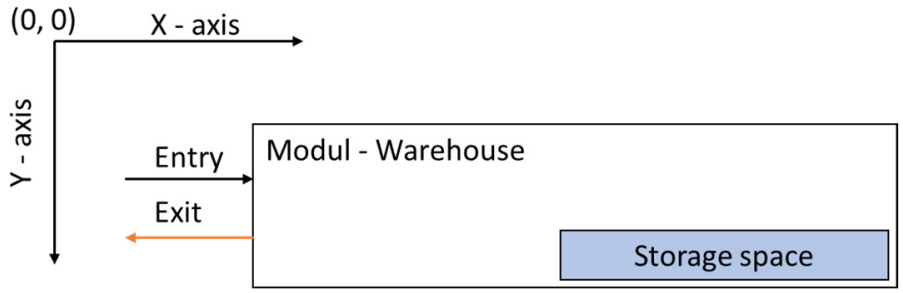


Figure 9. Production module Warehouse and its markers/connections.

The following two lines in Algorithm 2 handle the deletion of irrelevant markers and the creation of connections between two modules, forming a production cell or line.

```
Line 1:  obj3.deletObject
Line 2:  .materialflow.connector.connect(obj5,obj4)
```

Here, variables *obj3*, *obj4*, and *obj5* represent different objects in the simulation model, assigned dynamically during each iteration of the "for loop". These steps ensure logical material flow by refining the navigation points and establishing connections between modules.

Algorithm 3 was developed for material flow optimization and was built using an exact algorithm. It utilizes the database created by the previously described Algorithm 2. Using the production process plan for each product type, gathered from the External database in Thingsboard, the program identifies the necessary modules, checks their availability, and ensures their functionality (Figure 10).

The program evaluates the connections between all modules and determines if they are operational. Based on the distances between modules, their orientation, and their relative position on the factory floor, the shortest route is selected as the optimal path, and the corresponding modules are recorded in the Internal database. Additionally, Algorithm 3 verifies whether the modules are connected via connectors and if they are included in the technological plan. If any module or connection does not meet the requirements, the route is discarded, and an alternative path is chosen. The technological plan is gathered from the External database and consists of the sequence of operations and the modules on which those operations can be made.

Algorithm 3 works in two phases. In phase 1, step 1, the algorithm checks if any new work orders have arrived. If they have, then it gathers them and continues to step 2; if not, then the algorithm continues to step 2, where it checks whether the layout has changed. If the layout has not changed and no new work orders have been received, then in step 3, the algorithm does nothing. If new work orders have arrived or the layout has been changed, then the algorithm goes to phase 2, where step 1 consists of a loop going through all the objects from the technological plan. Steps 1, 2, 3, and 4 consist of checking whether objects are operational, checking if they are connected to other objects using connectors, determining the shortest routes, and choosing the shortest route for a pair of objects. Steps 1–4 are repeated until all the possible routes have been tested, and the shortest possible path is chosen for all objects. Step 5 checks whether all objects were checked, and step 6 writes the new technological plan.

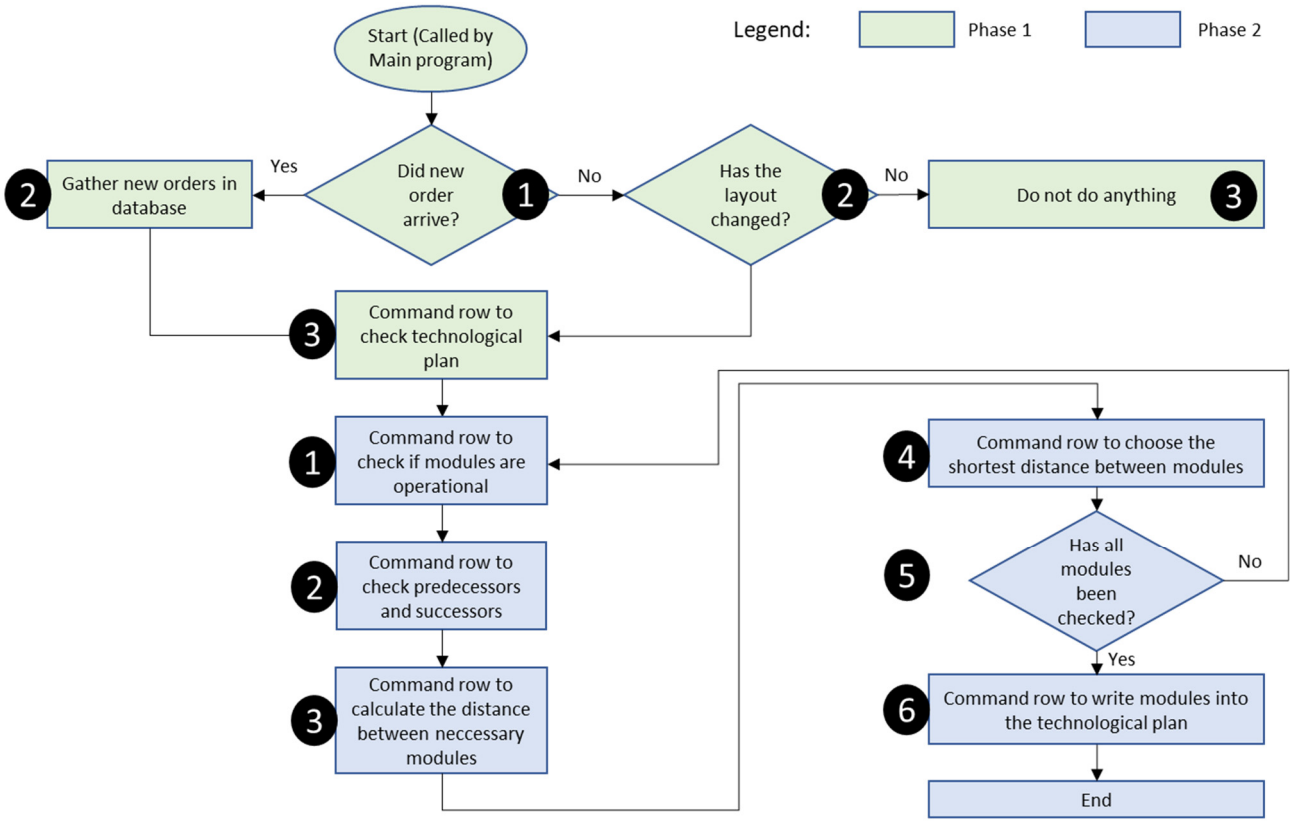


Figure 10. Block diagram for material flow optimization and the selection of proper modules.

3.6. Testing, Troubleshooting, and Verification of the Digital Twin Framework

The verification of the DT framework was conducted in several steps:

- The Machine Vision (MV) system was verified and validated based on the correct coordinates of the production modules detected. A Bosch laser measuring device, GLM150-27 C, was used to measure the actual location of production modules in the laboratory factory, which served as a reference for comparing the results obtained by the MV system. Some challenges can affect the image quality and accurate recognition of the production modules: ambient lighting and light reflections (eliminated by stable room conditions), image distortions (correction of image distortion was executed), image overlap (overlap of different cameras was corrected), and the correct placement and orientation of QR codes on the production modules. These challenges impacted the accurate recognition of modules as well as the correct positioning of modules in the TPS.

- The suitability and adequacy of the selected data for both the self-building of the simulation model and the simulation process were validated.
- Simulation model self-building capabilities and repeatability were verified and validated. Several use cases were considered to check the adjustability of the simulation model. The exact locations and rotations of modules in the real system were compared to the extracted locations and rotations by the MV system, which was translated to the TPS. The average standard deviation was calculated for the location and rotation parameters for each module. The time needed for self-building and the adjustability of the simulation model were compared to real factory needs. The correctness of the simulation model (connection between modules; module specifics) was also tested on the considered use cases.
- The material flow optimization algorithm was tested, verified, and validated based on the effectiveness of the results. Scalability testing and how the algorithm responds to higher complexity were not tested.

Troubleshooting steps were carried out during the development phases of the DT framework. All of the subsystems were troubleshot upon finding errors.

- The development phases of Machine vision system were troubleshot when errors in camera overlap, recognition of modules, or the position of modules were recognized. The code that detected QR codes for new or already existed modules was stopped at the beginning using breakpoints and was subsequently run line by line to see where the errors occurred. External sources were also contacted for help with camera distortion and overlap.
- The External database was examined when new production modules were discovered or data for already existing modules was sent to it. Troubleshooting in this case consisted of examining flow-based programming and seeing where the data went to and how it was saved.
- On the Tecnomatix Plant Simulation platform, both the Internal database as well as algorithms for sorting and filtering of the data, algorithms for self-building the simulation model, as well as algorithms for material flow optimization were troubleshot. Breakpoints were inserted into the code and algorithms were run line by line to detect errors. The Internal database was checked to determine whether the data was gathered and sorted correctly.
- Communication between subsystems was troubleshot in several ways. First it was checked if the connection can actually be established between different subsystems via a ping test and a port accessibility and credentials check. Then it was checked with one attribute to determine whether it was sent/received and if it was sent/received in the correct format.

Our primary objective was to assess the accuracy of the layout correspondence between Tecnomatix Plant Simulation (TPS) and the physical system, as well as the time needed for self-building and adjusting the simulation model. The real picture of layout use case is presented in Section 4.1 (Figure 11). It provides a robust basis for validating the Digital Twin framework's simulation model self-building capabilities. The tests confirmed that the proposed framework reliably adapts to different real-world configurations.

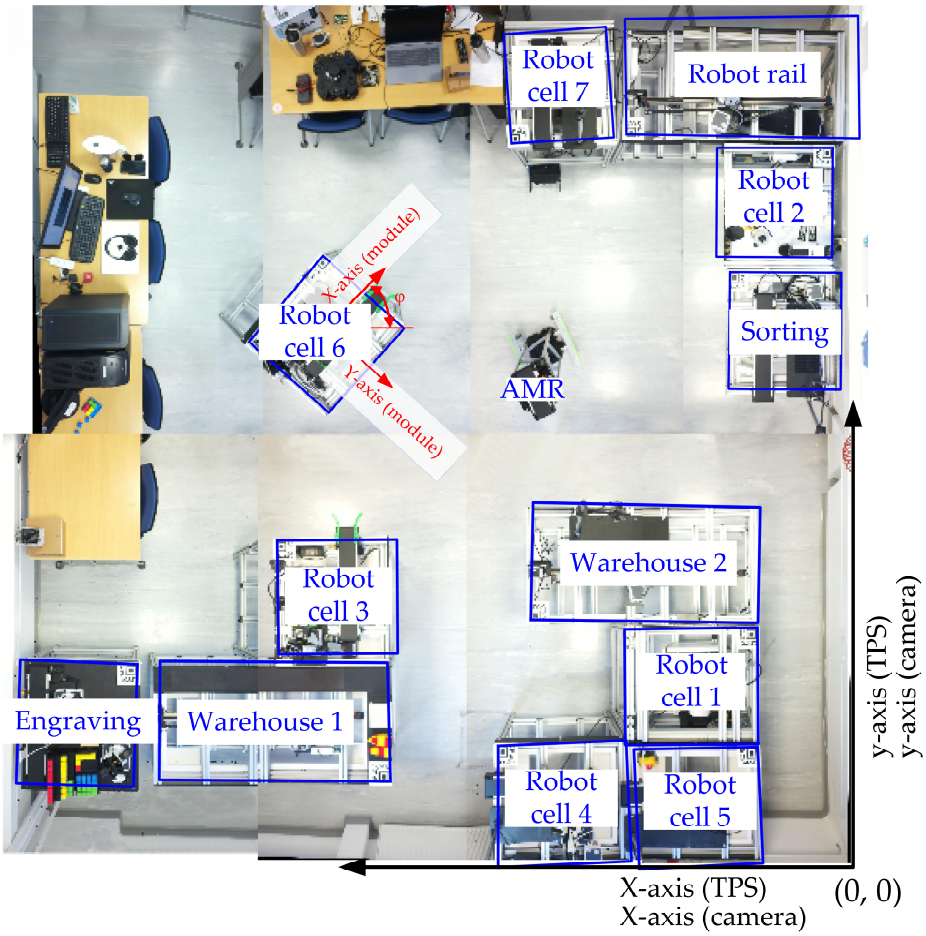


Figure 11. Laboratory factory floor use case: real picture of production modules, as recognized by the MV system.

4. Results and Discussion

This section presents the results focused on our proposed Digital Twin (DT) framework. Three main outcomes or results were obtained, analyzed, and evaluated during the research study:

- **Data requirements for self-building and execution**: The first result focused on identifying the essential data needed for both the self-building process and the execution of the simulation model;
- **Capacity of TPS for self-building**: The second result examined the ability of Tecnomatix Plant Simulation (TPS) to autonomously construct the simulation model based on the provided data, ensuring accuracy and scalability;
- **Optimization algorithm for module selection**: The third result addressed the development and effectiveness of the optimization algorithm, which uses multiple criteria to select appropriate modules based on the work order required to produce specific parts.

4.1. Data Requirements for Self-Building and Execution

After researching how TPS works and developing Algorithms 1 and 2, the following attributes from the production modules were defined to automatically build the simulation model: (1) the location of the module alongside the X-axis, (2) the location of the module alongside the Y-axis, (3) the rotation of the module, and (4) the name and functionality of the module (QR code and data from database). The real picture of the use-case scenario for

production module recognition (blue boundaries defined by the Machine Vision system) is shown in Figure 11.

The data about the exact location (X, Y) and rotation (φ) of the modules is obtained from the Machine Vision (MV) system, and these are combined with other data obtained from the QR code (the name of the production module and its function) linked to the data server (database). Since production modules have a typical rectangular shape (modular assembly production), the coordinates (X, Y) of the modules are defined as the centers of the modules with a known rectangle size. The rotation of the module (φ) starts at the X-axis and is directed toward the Y-axis.

The simulation needs several attributes to run properly as well as different attributes to replicate the state of the production floor. The attributes needed for the execution of the simulation are the following:

- Length of the modules, L_i (to calculate the cycle time according to the length and speed of the conveyor);
- Conveyor speed, v_i;
- Time needed to place objects on the module, t_1 (inlet time);
- Time needed to take objects from the module, t_2 (outlet time);
- Calculated time of module operation: $t_\mathrm{i} = t_2 - t_1$ or $t_\mathrm{i} = L_\mathrm{i}/t_\mathrm{i}$;
- AMR speed.

Table 4 presents 5 out of 26 data points gathered from one production module—Robot cell 3. The data related to Robot cell 3 (columns 1 and 2 from Table 4) is gathered from the Raspberry PI module control unit, while the data shown in columns 3 and 4 is gathered from Machine Vision system. Alongside is the data about the same production module, gathered from the Machine Vision system (name, location on X-axis, location on Y-axis, rotation).

Table 4. Data gathered from Robot cell 3 (Raspberry PI control unit) of the Real system and data from the Machine Vision system.

Robot Cell 3		**Machine Vision System**	
Attribute	**Value**	**Attribute**	**Value**
ID [str]	Module1_Standard	Robot_cell3_Name	Robot_cell3
Name [str]	Robot_cell3	Robot_cell3_LocationX [m]	2
Module status [bool]	true	Robot_cell3_LocationY [m]	4
Start of operation [timestamp]	10.6.2025 14:31:35	Robot_cell3_Rotation [°]	90
Errors [json]	{“Error”:false,”Type”:null}	...	...
...	...		

In our case study, the specific operation time t_i of a module is not essential or required. However, this data was pre-gathered for simulation purposes, allowing for the temporal evaluation of processes, process optimization, and feasibility testing. The time required for certain operations, such as machining, is included in the simulation as part of the time it takes for the part to move from the inlet of the module to its outlet.

As mentioned earlier, the data gathered from MV is important for the self-building of the simulation model. However, the MV system, which includes multiple cameras, encountered challenges such as overlap, distortions, and translation from the camera coordinate system to the global coordinate system when detecting production modules. To address distortions, we trained the MV system using machine learning techniques. Algorithms were also developed to calculate the overlap of different cameras and to

recalculate the position of modules if they were found in the overlap area. An assumption was made that calculating the translation from the coordinate system of each camera to the global coordinate system is carried out without errors. The main deviation in the location coordinates (X, Y) and rotation (φ) came from distortion and image overlap. Tables 5–7 present the gathered data on real measured location coordinates (X, Y) and rotation (φ) from five use cases and the data obtained from the MV system: location coordinates X_{MV} and Y_{MV} and rotation φ_{MV}.

Table 5. Measured X-coordinates of the real system (Real) and the obtained X-coordinates from the MV system.

Use Case	**1**		**2**		**3**		**4**		**5**	
Module	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**
Warehouse 1	3.75	3.72	2.85	2.77	4.51	4.62	4.44	4.43	4.44	4.48
Warehouse 2	1.31	1.35	0.90	0.91	0.90	0.93	0.73	0.73	1.22	1.24
Robot cell 1	1.03	1.04	0.97	1.00	0.97	0.95	3.77	3.66	1.71	1.73
Robot cell 2	0.73	0.73	0.82	0.84	2.04	1.99	0.78	0.76	0.78	0.76
Robot cell 3	3.39	3.32	2.43	2.49	2.92	2.83	2.67	2.63	3.99	3.95
Robot cell 4	1.84	1.80	1.75	1.76	2.52	2.45	2.99	2.91	2.67	2.73
Robot cell 5	1.01	0.98	0.95	0.94	0.95	0.92	4.69	4.63	4.69	4.78
Robot cell 6	3.31	3.38	3.09	3.11	3.65	3.65	3.28	3.37	3.93	3.84
Robot cell 7	1.87	1.88	1.91	1.87	1.91	1.89	2.02	2.04	2.34	2.36
Robot rail	0.8	0.81	0.97	0.99	0.97	0.96	1.92	1.92	0.99	0.98
Engraving	4.92	4.78	3.76	3.76	3.68	3.58	0.74	0.76	0.74	0.72
Sorting	0.74	0.76	2.85	0.88	1.96	1.96	3.34	3.32	2.72	2.7

Table 6. Measured Y-coordinates of the real system (Real) and the obtained Y-coordinates from the MV system.

Use Case	**1**		**2**		**3**		**4**		**5**	
Module	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**
Warehouse 1	0.71	0.70	0.84	0.82	0.79	0.81	1.53	1.53	1.53	1.54
Warehouse 2	2.25	2.31	5,29	5.36	5.29	5.44	4.85	4.84	4.99	5.06
Robot cell 1	1.51	1.53	0.66	0.68	0.66	0.65	0.81	0.79	2.45	2.48
Robot cell 2	4.42	4.45	4.46	4.59	3.65	3.56	2.87	2.80	2.87	2.80
Robot cell 3	1.49	1.46	2.30	2.35	3.67	3.56	3.73	3.68	3.25	3.21
Robot cell 4	0.75	0.73	0.75	0.75	1.64	1.60	0.81	0.79	1.23	1.26
Robot cell 5	0.78	0.76	1.43	1.41	1.43	1.39	0.78	0.77	0.78	0.79
Robot cell 6	3.32	3.39	3.65	3.67	3.63	3.63	3.14	3.22	4.53	4.43
Robot cell 7	5.17	5.20	5.18	5.06	5.18	5.11	4.28	4.32	4.25	4.29
Robot rail	5.15	5.21	2.56	2.61	2.56	2.55	0.80	0.80	0.87	0.86
Engraving	0.71	0.69	3.36	3.36	2.81	2.73	3.76	3.84	3.76	3.68
Sorting	3.68	3.78	3.70	3.80	4.36	4.36	1.52	1.51	3.16	3.14

Table 7. Measured rotations (φ) of the real system (Real) and the obtained rotations (φ) from the MV system.

Use Case	1		2		3		4		5	
Module	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**	**Real**	**MV**
Warehouse 1	180	181	0	0	0	0	0	0	0	0
Warehouse 2	180	179	0	0	0	0	90	90	90	91
Robot cell 1	−90	−89	180	180	180	181	90	89	270	270
Robot cell 2	−90	−89	0	0	0	0	270	268	270	271
Robot cell 3	−90	−90	270	269	0	0	315	316	0	0
Robot cell 4	0	0	180	182	135	134	0	0	270	271
Robot cell 5	180	180	90	89	90	90	0	0	0	0
Robot cell 6	135	134	315	312	270	268	315	313	270	269
Robot cell 7	−90	−90	90	90	90	90	315	314	0	0
Robot rail	0	0	90	90	90	89	0	0	0	0
Engraving	−90	−90	315	316	270	270	90	89	90	91
Sorting	180	179	90	89	90	90	180	179	180	182

Figures 12 and 13 present the calculated standard deviation (σ) of the location coordinates (X, Y) and rotation (φ) of each production module, considering five use cases. The central line in each box represents the median (close to the mean value in symmetric data), while the box edges show the interquartile range. The whiskers indicate variability outside the upper and lower quartiles. Figure 12 shows that most modules exhibit deviations within ±3%, confirming the MV system's accuracy in detecting module location. Figure 13 shows an even lower standard deviation (within ±1%), which also confirms the MV system's accuracy in detecting module rotation. These results confirm that the MV system is generally reliable, although further calibration could improve accuracy in certain areas.

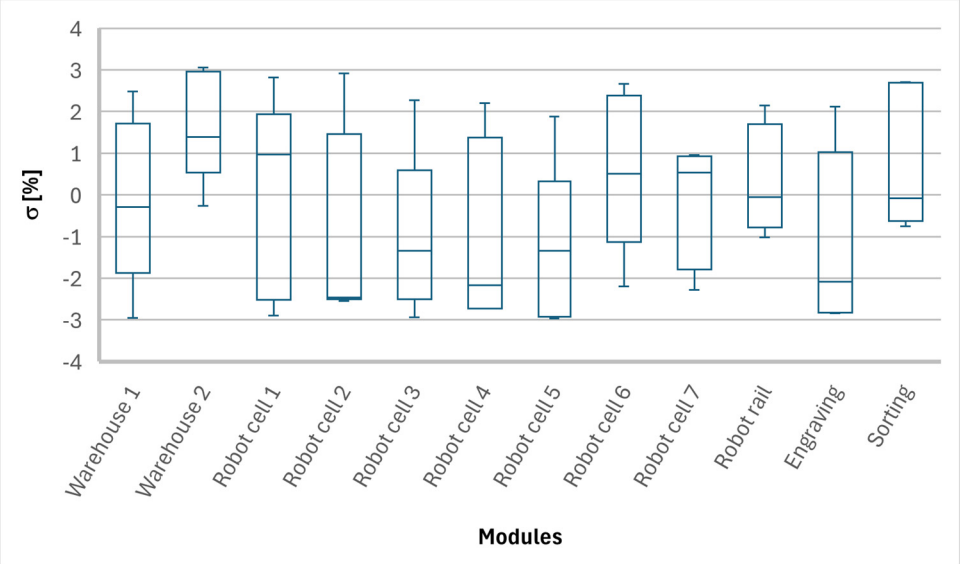


Figure 12. Standard deviation (σ) for the location (X, Y) for each production module.

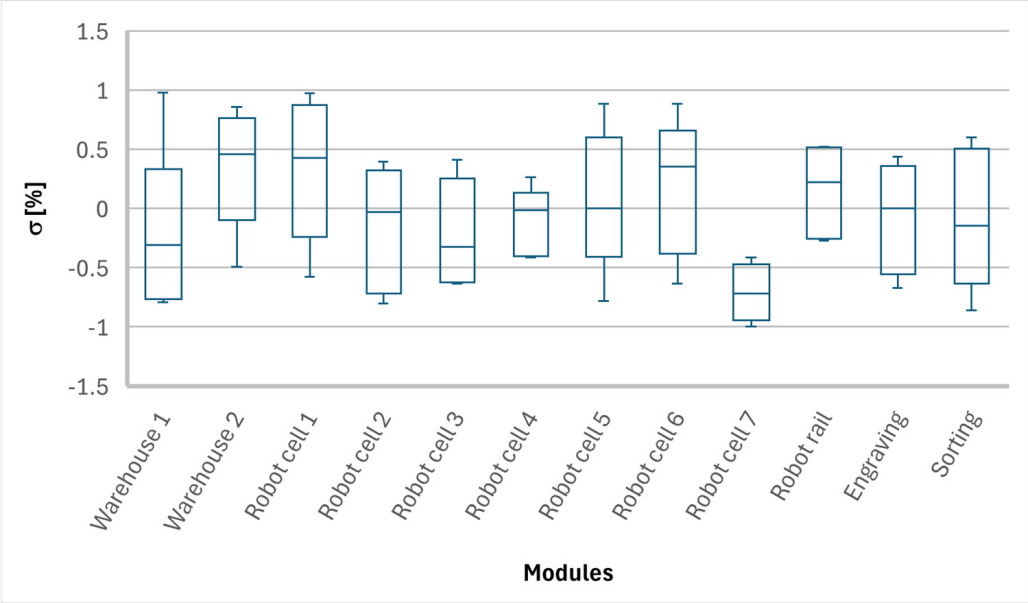


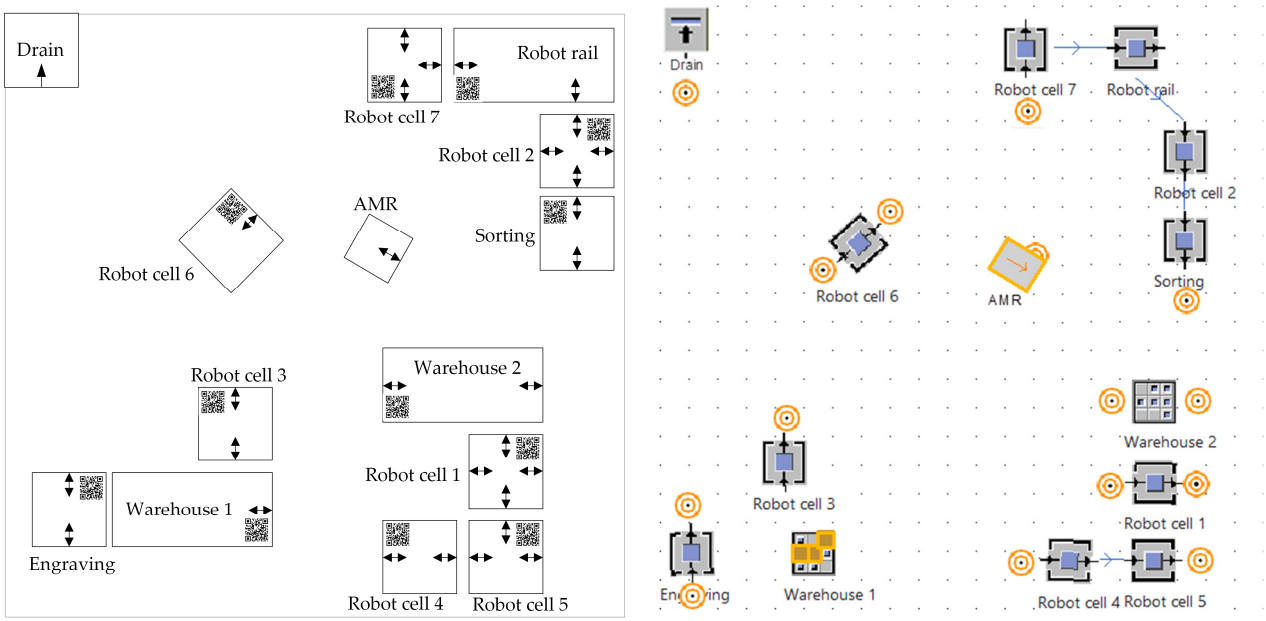
Figure 13. Standard deviation (σ) for rotation (φ) for each production module.

Key Findings:

- In order to create a self-building simulation model, the exact name or ID of the modules should be recognized and its data gathered from the server to know the modules' functionality and to define the seamless interconnections and material flow (inlet/outlet), its location (X, Y) on the factory floor, and the rotation (φ) related to the factory floor origin.
- The MV system proved to be effective in module recognition (their X and Y position as well as their rotation). With the data from the MV system and in connection with the External database, all the data needed for self-building and running of the simulation model can be gathered.
- The data required for the simulation (TPS) are determined using discrete events, their times (times of operation on modules, including the loading and unloading of objects), and AMR speed.
- The MV system proved challenging when the ambient lighting was too bright. Due to QR codes having a black pattern on white surface, reflections sometimes prevented the MV system from recognizing them. Distortions and camera angle overlap also caused problems and were the main contributors to the deviation in the captured and real coordinates of production modules. Stable environmental conditions were assured to prevent reflection. Machine learning approaches were used to eliminate distortion. Algorithms were developed to calculate the overlap between camera images.

4.2. Capacity of TPS for Self-Building

The second key result relates to the self-building capabilities of the simulation model. Our assumption about the data needed for simulation model creation and simulation running proved to be correct. If any of the specified data was missing, either the simulation model could not be built or the simulation would not run correctly. Following the execution of Algorithms 1 and 2, the outcome of one use case is illustrated in Figure 14a.



(**a**) (**b**)

Figure 14. Self-building process for a use case: (**a**) real system layout and (**b**) the results of module placement in the simulation model.

The layout includes both isolated modules and production module groups configured as production lines and production cells. Tecnomatix Plant Simulation (TPS) successfully generated the simulation model, placing modules in positions that accurately reflect the proposed real-world configuration. In addition to module placement, TPS automatically inserted markers for AMR control at module entry and exit points, as well as connectors to link relevant modules. While these markers and connectors do not exist in the physical system, they are essential for simulation functionality within TPS.

This configuration allowed for comprehensive testing of both Algorithms 1 and 2. As shown in Figure 14b, TPS effectively instantiated individual modules and correctly assembled them into production cells or lines within the virtual environment. The deviation between the placement of production modules in the real system and in TPS was within 3%, as the deviation stayed the same between the coordinates extracted by the MV system and those used in TPS; thus, the error came from the points described in Section 4.1.

The repeatability of self-building and the adjustability of the simulation model were tested through five different use cases, where the positions of modules were changed and the simulation model rebuilt itself in real time. The time needed for self-building and adjusting the simulation model was 36 s on average, which included communication between the External and Internal databases using the MQTT protocol. As quality-of-service (QoS) level 2 was used, the connection to each AAS in the External database took between 2 and 3 s. The time needed from the Internal database to the fully built simulation model was 1–2 s, on average.

Key findings:

- The self-building capacity of the simulation model shows an exact replication of the self-standing (isolated) modules. All production modules are in the proper location and orientation. The only caveat is that all the data needed for self-building and the running capacity of the simulation model is needed; otherwise, the simulation model will not be built or it will run incorrectly.
- The self-building capacity of the simulation model with regard to the production modules forming production lines and cells was equally sufficient. The algorithms

predicted the locations and rotations of the modules as well as their connection points. The real/predefined interconnections were correctly included in the simulation model. Due to proper interconnections between production modules, the correct material flow was defined according to module data.

- The specifics of certain production modules, such as the Warehouse module, were properly considered and implemented. The Warehouse module has entry and exit points in the same location; thus, the placement of other production modules next to the Warehouse module and the Warehouse module's rotation had to be considered. For instance, if the entry point into the Warehouse module was on its left side and there was a different production module on its right side, the algorithm correctly recognized this fact; it did not connect both production modules due to misalignment between the entry and exit points of the production modules.
- Communication between TPS and Thingsboard (Internal and External databases) took most of the time needed for self-building and adjusting the simulation model. In our research facility, the time it took to build the simulation model was sufficiently short. In reality, in actual factories, there are different definitions of real time. For some factories, real time is 1 min, whereas, for others, 1 h or even 1 day/week is sufficient. Suppose that real time is considered to equal 1 min, and the system's scale is 10 times bigger than our tested system. In this case, the scalability of this approach is questionable, and alternative communication protocols or methods should be used. One of the solutions could be a cloud database, where all the data from the AASs would be gathered. In this case, communication between TPS and the cloud database would only need one call instead of multiple calls to each AAS.

4.3. Optimization Algorithm for Module Selection

The final result pertains to Algorithm 3, which is used for calculating the optimal interconnection between production modules and routes (material flow) based on the production plan. This algorithm employs a multi-criteria approach to ensure efficient decision-making. When determining the modules for the optimal route, Algorithm 3 first verifies whether the modules are operational/functional or not. Non-functional modules are excluded from the candidate pool. Next, it evaluates whether the selected module has predecessors or successors (connections to other modules). If such connections exist, the algorithm checks the operational status of the connected modules and removes any non-functional ones.

Subsequently, the algorithm calculates the distance between the previously selected module, as specified in the production plan, and the current candidate module. It selects the module that offers the shortest route, ensuring optimal performance. Figure 15 illustrates an example of several production modules placed on the factory floor and two possible routes, selected by the decision-making process employed by the optimization algorithm. The highlighted route and the production modules represent the final solution of an example considered in the TPS.

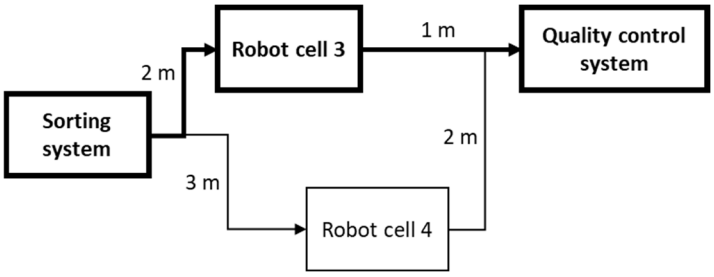


Figure 15. Module selection based on the production plan and best possible material flow.

The illustration demonstrates how the optimization algorithm selects the optimal route between modules based on known distance. It depicts two possible routes:

- Route 1: Sorting system → Robot Cell 3 → Quality Control System, giving an overall distance of 3 m (2 m + 1 m = 3 m).
- Route 2: Sorting System → Robot Cell 4 → Quality Control System, giving an overall distance of 5 m (3 m + 2 m = 5 m).

In this example, Route 1 is selected due to the shortest total distance (3 m). The other criteria for module selection, such as module functionality and connector verification, are not applied here, as all modules are considered operational, and none have connectors with other modules in this scenario.

Once the optimal route is determined, the selected modules and their connections are written into the Internal database of the Digital Twin (DT) framework. This data is then used during the simulation run to replicate the production process and validate the system's performance according to the technological plan. The algorithm used is an exact algorithm.

We tested one product with the following technological plan:

1. Picking operation from Warehouse 1 (where parts are stored);
2. Sorting of picked objects on module Sorting (functionality);
3. Manipulation enacted on parts, and selection from Robot cells 3–5;
4. Quality control enacted on assembled products, with selection coming from Robot cells 6 and 7;
5. Placement of assembled object in the final warehouse—Drain (placed in upper-left corner of Figure 14b).

The scaling factor used in the TPS frame was 0.015, which means that one square on the grid was 0.3 × 0.3 m. From use case 1 presented in Sections 3.1 and 3.2, the following result was achieved using the exact algorithm: using the known functionalities of the modules and technological plan for the selected product, the shortest route was selected. The combined length of the route was 23.5 m, including the length of the modules. The benchmark value was the longest route possible at 29.2 m, using only the modules with functionalities needed as specified in the technological plan.

Although we are aware that exact algorithms suffer severe restrictions with the scaling of the model, for our research, it was sufficient, as our focus was on the DT framework and not on material flow optimization

Key findings:

- The exact algorithm worked sufficiently fast and accurately, with the results being as expected. The algorithm's multi-criteria for production module selection worked correctly. If certain production modules were out of operation, they were not considered candidates for selection during material flow optimization. The interconnectivity of production modules and the distance between them were also correctly considered. Subsequently, the material flow was optimized using the exact algorithm.
- The optimized route of 23.5 m was 19.5% shorter than the benchmark route of 29.2 m, which confirmed the efficiency of our optimization algorithm.
- The exact algorithm is too slow and computationally demanding when scaling up production, which is its known limitation. Due to job-shop scheduling being an NP-hard problem, using heuristic rules, metaheuristics, or a combination of both would resolve the problem of real-time scheduling. Exact algorithms become too time-consuming above 20 modules. Adding to the constraints with module operations and real-time planning and re-planning when adding new work orders, exact algorithms can already become too computationally heavy and time-consuming at 5–10 modules [31]. Al-

though heuristic rules, metaheuristics, or machine learning yield sub-optimal results, those results are very near optimal and, in most cases, good enough for manufacturing and production uses.

- Using standard hardware (16 GB ram, Intel i7 11th generation) the time needed for material flow optimization of 50+ modules using the exact algorithm is between 1 to 3 h. At the same time, using a HPC (High-Performance Computing) system to solve the same problem would need 10–40 min. To make material flow optimization run in real-time with 50+ modules, the use of algorithms other than an exact algorithm is necessary.
- Machine learning techniques could be used for predictions and generating additional data. Neural networks can be used for tool wear [32], maintenance, stock, and incoming work order predictions. This could enrich data for scheduling and create a more robust system.

4.4. Comparison of Research Work

As was described in the Introduction, there have been relatively few articles describing the topic of self-building simulation models within Digital Twin frameworks, specifically for production systems. A few of them are focused on self-building production modules or cells, and only one article focused on the layout reconfiguration using a self-building simulation model. However, the approaches and principles used in state of the art can be compared to our developed Digital Twin Framework. We defined the most important components of DTs, which are listed in the first column of Table 8.

Table 8. Comparison of research performed.

Important Components of DT	Our Research	[23]	[24]	[26]
Real system	Modular production modules	Discrete manufacturing workshop	CPPS	Semiconductor fab
Physical system recognition	Machine Vision system	/	/	Material flow and tool behavior from event logs
Database	External database (Thingsboard) with AAS, Internal database (Tecnomatix Plant Simulation) for simulation model building and simulation running	Internal database in Tecnomatix Plant Simulation	Data Lake	Lot-tracking data and resource-tracking/control data
Connectivity protocols	MQTT protocol	TPC/IP, Fieldbus, Ethernet	OPC UA	/
Simulation model	Real-time self-building and reconfigurability of simulation model within Digital Twin framework	Manual simulation model building, automated layout reconfiguration	Self-adaptive DT reference architecture (MAPE-K loop), tool reconfigurability, quality assessment	Automatic simulation model generation/reconfiguration, ML/AI for tool behavior
Scheduling integration	Integrated exact algorithm for material flow optimization	/	Case-based reasoning, Event–Condition–Goal models for tool scheduling	Simulation-based validation
Additional Digital Twin component	/	/	Quality Assessor component	/

5. Conclusions and Future Directions

This research presented a Digital Twin framework designed to enhance adaptability and operational efficiency in Industry 4.0 production environments. From a scientific point of view, all of the research objectives defined in the Introduction were successfully achieved.

To bridge a gap in the integration of systems for capturing factory layout, a Machine Vision subsystem was developed that detects and interprets production module layouts in real time, enabling accurate recognition of module locations and rotations. Data gathered by the Machine Vision system is gathered in an External database in AASs for each production module, and new AASs are created upon recognition of new modules. At the same time, the MV system is connected directly to the digital model to provide data for layout changes.

The data needed for both the creation of the digital model as well as the running of the digital model has been researched, and necessary data, such as the location of the modules, their rotation, and name/functionality, was chosen. The MQTT protocol was chosen as the connectivity protocol between the MV system, External database and Internal database due to its lightweight and fast nature.

Based on minimal essential data, the framework autonomously creates a simulation model without manual intervention. The simulation model is created and adjusted in real time with changes to it being made when new work orders arrive, the layout changes, or when production modules encounter errors. This capability serves as a basis for a self-building and adjustable Digital Twin.

The scheduling algorithm, based on an exact algorithm, optimizes material flow in real time when new work orders arrive or changes on work floor are detected. The algorithm is integrated into the Digital Twin and is executed, if necessary, once new data is gathered.

The framework is parametric and broadly applicable across industrial sectors with varying production layouts. It supports real-time synchronization between physical and digital environments, distinguishing it from existing research, as most previous approaches either lack real-time adaptability or only offer partial automation. Unlike earlier works, the developed framework achieves full integration from detection to simulation model generation and scheduling. Although factories with full modular production do not yet exist, research cases in laboratory environments do exist, such as in [33]. The closest application of these technologies is Siemens Electronics Works Amberg, which is a Siemens use-case factory for Industry 4.0 technologies. The factory represents the modular production of electronics such as PLCs and circuit boards. Our developed Digital Twin framework can be used in similar production environments. Another example is the textile industry, where full customization of products is possible. Such customization needs agile and reconfigurable production, which can be covered with the developed framework. Lastly, any production going through rapid layout changes can benefit from using the developed Digital Twin framework. An example of such a factory was an electronics and semiconductor factory, where production was moved from one factory location to another and new production lines were bought for their original location. Consequently, their factory layout changed on a weekly basis.

Practical benefits of such an approach to Digital Twins include a significant reduction in simulation setup time, improved responsiveness to layout changes, and enhanced material flow optimization. However, the framework's effectiveness could be sensitive to the reliability of the MV system and QR code recognition. Scalability is also constrained by the use of exact algorithms, which become computationally intensive as the number of modules increases. To address this, heuristic and metaheuristic strategies were developed to deliver near-optimal results within real-time constraints.

Future work can be carried out in three main directions: (1) Refinement of material flow optimization: Although exact optimization algorithms can provide optimal results,

they are often computationally intensive and unsuitable for real-time applications. Future research could focus on exploring heuristic rules such as Shortest Processing Time (SPT), Earliest Due Date (EDD), or others, which offer fast, rule-based decision-making. Additionally, metaheuristic approaches, such as Genetic Algorithms, may provide a good trade-off between computational efficiency and solution quality. (2) Integration of advanced analytics: Expanding the framework to incorporate machine learning approaches could enable more proactive and data-driven decision-making. Neural networks could be used to predict machine failure rates, forecast incoming work orders, or estimate future stock levels. These predictions could then be used as additional inputs to enhance the scheduling algorithm. (3) Scalability testing: Further research could evaluate the framework's performance in large-scale systems by introducing additional modules, more simultaneous work orders, and varying operation sequences on modules. This could help in assessing the robustness and adaptability of the approach under increased system complexity. However, successful scaling up would depend on the improvements made in material flow optimization, as the currently used exact algorithm does not allow for scalability testing.

Author Contributions: Conceptualization, F.J.V. and M.P.; methodology, F.J.V. and M.P.; software, F.J.V. and M.P.; validation, F.J.V., M.P., M.Š. and N.H.; formal analysis, F.J.V. and M.P.; investigation, F.J.V. and M.P.; resources, N.H. and H.Z.; data curation, F.J.V. and M.P.; writing—original draft preparation, F.J.V., M.Š. and M.P.; writing—review and editing, M.P., M.Š., H.Z. and N.H.; visualization, F.J.V., M.P., M.Š. and N.H.; supervision, M.Š. and N.H.; project administration, N.H.; funding acquisition, M.Š., H.Z. and N.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovenian Research and Innovation Agency—ARIS, research project J2-4470 and research program P2-0248. This research was funded and implemented within the framework of the European Union under the Horizon Europe Grants N°101087348 (INNO2MARE), N°101058693 (STAGE), and the NextGenerationEU project GREENTECH.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article, and further inquiries can be directed to the corresponding author.

Acknowledgments: We would like to express our sincere gratitude to the company DIGITEH for their financial support and for providing the necessary software and hardware. Our thanks also go to the LASIM laboratory at the Faculty of Mechanical Engineering, University of Ljubljana, for their assistance in providing the physical infrastructure and conducting the experiments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Popkova, E.G.; Sergi, B.S. Human capital and AI in industry 4.0. Convergence and divergence in social entrepreneurship in Russia. *J. Intellect. Cap.* **2020**, *21*, 565–581. [CrossRef]
2. Feldkamp, N.; Bergmann, S.; Straßburger, S. Simulation-Based Deep Reinforcement Learning For Modular Production Systems. In Proceedings of the 2020 Winter Simulation Conference (WSC), Orlando, FL, USA, 14–18 December 2020; pp. 1596–1607.
3. Resman, M.; Pipan, M.; Šimic, M.; Herakovič, N. A new architecture model for smart manufacturing: A performance analysis and comparison with the RAMI 4.0 reference model. *Adv. Prod. Eng. Manag.* **2019**, *14*, 153–165. [CrossRef]
4. Poonpakdee, P.; Koiwanit, J. Accuracy of Distributed Systems towards Industry 4.0: Smart Grids and Urban Drainage Systems Case Studies. *Int. J. GEOMATE* **2018**, *14*, 70–76. [CrossRef]
5. Grieves, M. Digital Twin: Manufacturing Excellence through Virtual Factory Replication. *White Pap.* **2015**, 1–7.
6. Glaessgen, E.H.; Stargel, D.S. The Digital Twin Paradigm for Future NASA and U.S. Air force Vechicles. In Proceedings of the 53rd Structurs, Structiural Dynamics, and Materials Conference: Special Session on the Digital Twin, Honolulu, HI, USA, 23–26 April 2012.

7. Fabri, M.; Ramalhinho, H.; Oliver, M. Internal logistics flow simulation: A case study in automotive industry. *J. Simul.* **2022**, *16*, 204–216. [CrossRef]
8. Mourato, J.; Ferreira, L.P.; Sá, J.C.; Silva, F.J.G.; Dieguez, T.; Tjahjono, B. Improving internal logistics of a bus manufacturing using the lean techniques. *Int. J. Prod. Perform. Manag.* **2021**, *70*, 1930–1951. [CrossRef]
9. Aheleroff, S.; Xu, X.; Zhong, R.Y.; Lu, Y. Digital Twin as a Service (DTaaS) in Industry 4.0: An Architecture Reference Model. *Adv. Eng. Inform.* **2021**, *47*, 101225. [CrossRef]
10. Jordan, E.; Berlec, T.; Rihar, L.; Kušar, J. Simulation of Cost Driven Value Stream Mapping. *Int. J. Simul. Model.* **2020**, *19*, 458–469. [CrossRef]
11. Berlec, T.; Kleindienst, M.; Rabitsch, C.; Ramsauer, C. Methodology to Facilitate Successful Lean Implementation. *J. Mech. Eng.* **2017**, *63*, 457–465. [CrossRef]
12. Legowo, M.B.; Indiarto, B. Issues and Challenges in Implementing Industry 4.0 for the Manufacturing Sector in Indonesia. *Int. J. Progress. Sci. Technol.* **2021**, *25*, 650–658. [CrossRef]
13. Wang, X.; Wang, Y.; Tao, F.; Liu, A. New Paradigm of Data-Driven Smart Customisation through Digital Twin. *J. Manuf. Syst.* **2021**, *58*, 270–280. [CrossRef]
14. Liu, C.; Chen, Y.; Xu, X.; Che, W. Domain generalization-based damage detection of composite structures powered by structural digital twin. *Compos. Sci. Technol.* **2024**, *258*, 110908. [CrossRef]
15. Son, Y.-J.; Joshi, S.B.; Wysk, R.A.; Smith, J.S. Simulation-Based Shop Floor Control. *J. Manuf. Syst.* **2002**, *21*, 380–394. [CrossRef]
16. Carnahan, J.C.; Reynolds, P.F., Jr. Requirements For DDDAS Flexible Point Support. In Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, 3–6 December 2006; Perrone, V.L.F., Wieland, F.P., Liu, J., Lawson, B.G., Nicol, D.M., Fujimoto, R.M., Eds.; pp. 2101–2108.
17. Zhou, G.; Zhang, C.; Li, Z.; Ding, K.; Wang, C. Knowledge-driven Digital Twin manufacturing cell towards intelligent manufacturing. *Int. J. Prod. Res.* **2020**, *58*, 1034–1051. [CrossRef]
18. Leng, J.; Chen, Z.; Sha, W.; Lin, Z.; Lin, J.; Liu, Q. Digital Twins-based flexible operating of open architecture production line for individualized manufacturing. *Adv. Eng. Inform.* **2022**, *53*, 101676. [CrossRef]
19. Lattner, A.D.; Bogon, T.; Lorion, Y.; Timm, I.J. A knowledge-based approach to automated simulation model adaptation. In Proceedings of the 2010 Spring Simulation Multiconference, San Diego, CA, USA, 11–15 April 2010; pp. 200–207.
20. Akhavian, R.; Behzadan, A.H. Dynamic Simulation of Construction Activities Using Real Time Field Data Collection. In Proceedings of the 18th Workshop of Intelligent Computing in Engineering, Cardiff, UK, 16–18 July 2014; Hartmann, T., Rafiq, Y., de Wilde, P., Eds.; pp. 1–9.
21. Martínez, G.S.; Sierla, S.; Karhela, T.; Vyatkin, V. Automatic Generation of a Simulation-Based Digital Twin of an Industrial Process Plant. In Proceedings of the IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 3084–3089. [CrossRef]
22. Latif, H.; Shao, G.; Starly, B. A Case Study of Digital Twin for a Manufacturing Process Involving Human Interactions. In Proceedings of the 2020 Winter Simulation Conference (WSC), Orlando, FL, USA, 14–18 December 2020.
23. Guo, H.; Zhu, Y.; Zhang, Y.; Ren, Y.; Chen, M.; Zhang, R. A Digital Twin-Based Layout Optimization Method for Discrete Manufacturing Workshop. *Int. J. Adv. Manuf. Technol.* **2021**, *112*, 1307–1318. [CrossRef]
24. Splettstößer, A.K.; Ellwein, C.; Wortmann, A. Self-adaptive digital twin reference architecture to improve process quality. *Procedia CIRP* **2023**, *119*, 867–872. [CrossRef]
25. Huang, J.; Huang, S.; Moghaddam, S.K.; Lu, Y.; Wang, G.; Yan, Y.; Shi, X. Deep Reinforcement Learning-Based Dynamic Reconfiguration Planning for Digital Twin-Driven Smart Manufacturing Systems With Reconfigurable Machine Tools. *IEEE Trans. Ind. Inform.* **2024**, *20*, 13135–13146. [CrossRef]
26. Behrendt, S.; Altenmüller, T.; May, M.C.; Kuhnle, A.; Lanza, G. Real-to-sim: Automatic simulation model generation for a digital twin in semiconductor manufacturing. *J. Intell. Manuf.* **2025**, 1–20. [CrossRef]
27. Ho, G.T.S.; Tang, V.; Tong, P.H.; Tam, M.M.F. Demand-driven storage allocation for optimizing order picking processes. *Expert Syst. Appl.* **2025**, *272*, 126812. [CrossRef]
28. Lu, Q.; Li, M.; Zhu, D. Model-based definition-assisted asset administration shell as enabler for smart production line. *Int. J. Comput. Integr. Manuf.* **2025**, 1–17. [CrossRef]
29. Resman, M.; Herakovič, N.; Debevec, M. Integrating Digital Twin Technology to Achieve Higher Operational Efficiency and Sustainability in Manufacturing Systems. *Systems* **2025**, *13*, 180. [CrossRef]
30. Resman, M.; Protner, J.; Simic, M.; Herakovic, N. A Five-Step Approach to Planning Data-Driven Digital Twins for Discrete Manufacturing Systems. *Appl. Sci.* **2021**, *11*, 3639. [CrossRef]
31. Zupan, H.; Herakovič, N.; Žerovnik, J. A robust heuristics for the online job shop scheduling problem. *Algorithms* **2024**, *17*, 568. [CrossRef]

32. Dominguez-Caballero, J.; Ayvar-Soberanis, S.; Curtis, D. Intelligent real-time tool life prediction for a digital twin framework. *J. Intell. Manuf.* **2025**, 1–21. [CrossRef]
33. Arnarson, H.; Mahdi, H.; Solvang, B.; Bremdal, B.A. Towards automatic configuration and programming of a manufacturing cell. *J. Manuf. Syst.* **2022**, *64*, 225–235. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.