FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Desenvolvimento de um Modelo de Simulação para análise e avaliação de colaboração homem máquina em célula de assemblagem

João Pedro Caldas Ferreira

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Américo Azevedo

Second Supervisor: Prof. António Almeida

November 5, 2025

Resumo

Esta dissertação apresenta o desenvolvimento de uma framework de simulação destinada à avaliação de estratégias de colaboração homem-máquina em celulas de assemblagem industriais. A ferramenta foi concebida para replicar ambientes de produção baseados em eventos discretos, utilizando dados reais fornecidos pela Bosch Security Systems, S.A. Foram modeladas duas linhas distintas servindo como casos de estudo para análise comparativa de desempenho e proposta de melhorias. A framework permite simular diferentes configurações, como variações no número de operadores, integração de cobots e cenários de balanceamento de linha. São extraídos indicadores de desempenho como produtividade, tempos de ciclo, tempo com valor acrescentado e taxa de ocupação por estação, possibilitando comparações objetivas entre cenários. Foram desenvolvidas duas propostas de melhoria, ambas com introdução de cobots em sequências operacionais específicas. Foi realizada uma análise custo-benefício, estimando o esforço de implementação, retorno do investimento e ganhos operacionais. Os resultados demonstram que a simulação constitui uma ferramenta eficaz no apoio à decisão industrial, ao permitir prever com rigor o impacto da robótica colaborativa em ambientes de trabalho mistos.

Abstract

This dissertation presents the development of a simulation framework for evaluating human-machine collaboration strategies in industrial assembly lines. The framework was designed to replicate discrete-event production environments using real operational data from Bosch Security Systems, S.A. Two distinct assembly lines were modeled to serve as case studies for performance benchmarking and improvement proposals. The tool allows the simulation of different configurations, including variable numbers of operators, cobot integration, and line balancing scenarios. Key performance indicators such as throughput, station utilization, value-added time, and cycle time are extracted to compare configurations. Two improvement scenarios were proposed, each introducing cobots to automate specific sequences of operations. A cost-benefit analysis was conducted to estimate the implementation effort, return on investment, and operational gains. The results demonstrate that simulation can support structured decision-making in industrial environments by predicting the impact of collaborative robotics in mixed workspaces.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. n the context of this report, the implementation of Industry 4.0 technologies, particularly Human-Robot Collaboration (HRC) and Digital Twins, aligns with several SDGs, most notably:

- **Goal 9: Industry, Innovation, and Infrastructure** – The integration of advanced technologies such as HRC and Digital Twins into manufacturing processes supports innovation, fosters infrastructure development, and promotes resilient industrialization.
- **Goal 8: Decent Work and Economic Growth** – Human-robot collaboration can enhance productivity and create safer, more sustainable workplaces, contributing to decent work conditions and sustainable economic growth.
- **Goal 12: Responsible Consumption and Production** – Digital Twins and simulation models can optimize resource use, reduce waste, and improve the sustainability of industrial production processes, directly contributing to responsible consumption and production.
- **Goal 13: Climate Action** – By enabling energy-efficient processes and predictive maintenance through Digital Twins, Industry 4.0 technologies can help reduce the environmental footprint of manufacturing, supporting efforts in climate action.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Professor **António Almeida**, and to my co-supervisor, Professor **Américo Azevedo**. I am deeply grateful to my family and friends for their unwavering support. Finally, I would like to thank all the individuals and institutions that have supported me.

João Pedro Caldas Ferreira

"No one knows what the future holds. That's why its potential is infinite."

Okabe Rintarou

Contents

List of Figures

List of Tables

List of Acronyms

ABM Agent-Based Modeling

CoBots Collaborative Robots

DT Digital Twin

DPs Design Principles

HRC Human-Robot Collaboration

DSS Decision-Support Systems

IOs Inputs and Outputs

KPIs Key Performance Indicators

GUI Graphical User Interface

Chapter 1

Introduction

This dissertation lies at the intersection of industrial automation, industrial computing, and operations management. In this context, strategically allocating tasks between human operators and robotic systems is essential to improving efficiency, ergonomics, and overall system performance in industrial assembly environments. The topic aligns closely with the principles of Industry 4.0, which promotes the integration of advanced technologies to enable intelligent, flexible, and adaptive manufacturing systems. As organizations increasingly face dynamic and complex production requirements, there is a growing demand for decision-support tools that allow engineers to explore alternative task allocation strategies, simulate their operational impact, and make informed, data-based decisions.

1.1 Context and Motivation

The increasing integration of human operators and robotic systems within industrial assembly processes represents a significant challenge for optimization in modern manufacturing [5]. Within this context, Bosch Security Systems in Ovar operates in a particularly large-scale manufacturing environment, where a high number of assembly lines function simultaneously. This industrial setting provides a representative and relevant case study for this dissertation, as balancing human-centered assembly lines with the efficiency of autonomous systems is critical to achieving optimal system performance, scalability, and production consistency.

Human operators offer cognitive flexibility, adaptability, and creative decision-making, particularly in tasks requiring fine motor skills, context awareness, or handling process exceptions. On the other hand, robotic systems excel in repetitive tasks by providing high precision, consistency, and the ability to relieve workers from physically demanding operations [17, 29]. Finding the right trade-off between these aptitudes is essential not only to streamline operations and reduce product errors but also to lower production costs and improve working conditions.Collaborative Robots (CoBots) represent a hybrid solution that combines the capabilities of humans and robots

[13]. CoBots are designed to work safely alongside humans, automating low-value or ergonomically challenging tasks [17]. By integrating CoBots into the workforce, production systems can facilitate more flexible task sharing and leverage the strengths of both human and robotic labor [35].

As Bosch Ovar continues to pursue innovation in its internal logistics and production lines, there is a growing need for tools that support strategic decision-making in task allocation between humans, machines, and CoBots. A simulation-based decision-support framework would offer transformative changes by enabling engineers to evaluate multiple configurations in a risk-free environment before implementation. This approach supports the selection of optimal distributions of human-robot task allocations within assembly lines. This dissertation addresses this need for a simulation framework that can model and simulate various hybrid operational scenarios in assembly lines, allowing for the identification of the most efficient solutions. Furthermore, this aligns with the principles of Industry 4.0 by leveraging data-driven modeling and virtual experimentation to support continuous improvement in complex manufacturing systems [23].

1.2 Problem Statement

In modern industrial assembly factories, companies are pressured to rapidly adapt to dynamic production demands and pursue continuous improvements for efficiency gains. In the context of Bosch Ovar, one of the key industrial challenges is how to optimally allocate tasks among humans, machines, and CoBots. The goal is to enhance productivity while also reducing production costs, all without compromising flexibility and adaptability.

One of the central challenges addressed in this dissertation is the absence of a structured, flexible, and user-friendly decision-support tool specifically designed for human-machine-cobot assembly lines. Most existing simulation platforms are too general-purpose, offering limited configuration capabilities, restrictive licensing, and steep learning curves. These limitations make them impractical for rapid prototyping and the evaluation of alternative layouts by industrial engineers, who often lack the time or specialized skills to build complex models from the ground up. Moreover, engineers who are expected to use these tools often do not have the time or specialized knowledge needed to model a production system from the ground up. As a result, tools designed for real-world industrial decision-making must be user-friendly, modular, and able to provide actionable results in a short amount of time.

To overcome these limitations, the proposed simulation tool should enable engineers to test and compare structural reconfigurations such as merging or splitting assembly lines, assessing the ideal number of operators for a given production goal, or exploring alternative layout configurations. It also allows the evaluation of whether the inclusion of CoBots justifies the associated investment.

These capabilities are crucial for comparing cost-performance trade-offs across multiple scenarios and supporting strategic decisions grounded in operational data.

Applying this approach to the Bosch Ovar production context serves not only as a validation of the tool's effectiveness but also as a real-world demonstration of how simulation can support strategic decision-making, drive continuous improvement, and ultimately enhance operational efficiency while promoting more sustainable workforce practices.

1.3 Objectives

The main objective of this dissertation is to develop a simulation-based decision-support framework that enables the analysis and optimization of task allocation strategies within Human-Robot Collaboration (HRC) assembly lines. The tool is designed to address Bosch Ovar's need for flexible planning and operational adaptability in a multi-line production context, with the ultimate goal of improving workflow efficiency, reducing costs, and ensuring safe and ergonomic working conditions.

To achieve this, the project is structured around the following specific objectives:

- **Process Mapping** - Conduct a detailed study of the general assembly line at Bosch Ovar to identify the sequences and patterns of operations, as well as the distribution of tasks. This mapping will serve as the foundation for introducing a simulation-based tool developed using appropriate modeling methodologies.
- **Development of the Simulation Framework** - Develop and establish a simulation platform designed specifically for the structure and requirements of real industrial assembly environments. The framework must be modular, user-friendly, and capable of testing various operational scenarios.
- **Model Development** - Utilizing the simulation framework, create a digital model of a specific Bosch Ovar assembly line chosen as the case study. This model should accurately reflect the logic, structure, and flow of the actual system based on observed processes and gathered data.
- **Model Validation** - Validate the developed simulation model by comparing its outputs with real-world data from the selected Bosch Ovar assembly line. This comparison should include an assessment of process times, cycle times, and production throughput to ensure that the model accurately reflects the behavior of the actual system.
- **Scenario-Based Simulation and Evaluation** - Simulate and analyze multiple scenarios involving variations in the number of operators, introduction of CoBots, and layout reconfigurations. Each scenario should be assessed in terms of productivity, efficiency, and system flexibility.

- **Cost-Benefit Analysis and Strategy Assessment** - Compare the outcomes of each scenario by studying trade-offs between operational performance and resource investment, in order to identify the most cost-effective configurations.
- **Improvement Recommendations** - Based on the simulation results and industry insights, formulate recommendations to improve assembly line performance and guide future process changes.

1.4 Research Approach and Methodology

The research methodology for this dissertation was structured to combine an applied research with the development of simulation-based Decision-Support Systems (DSS) framework. This methodology follows a systematic, multi-phase approach that includes gathering relevant literature, study and understanding of contemporary methos, building a simulation model, conducting scenario analysis, and validating the results through real-world data comparison. The process involved iterative refinement and the use of advanced research tools to ensure both theoretical rigor and practical applicability.

1.4.1 Literature Review

The first step in the research process was conducting an extensive literature review to establish a strong theoretical foundation for the dissertation. The review focused on key areas such as Industry 4.0, HRC, simulation frameworks options, and Digital Twin (DT) technologies, all of which are central to the research goals. This phase aimed to synthesize existing knowledge and lay the groundwork for developing the simulation-based tool.

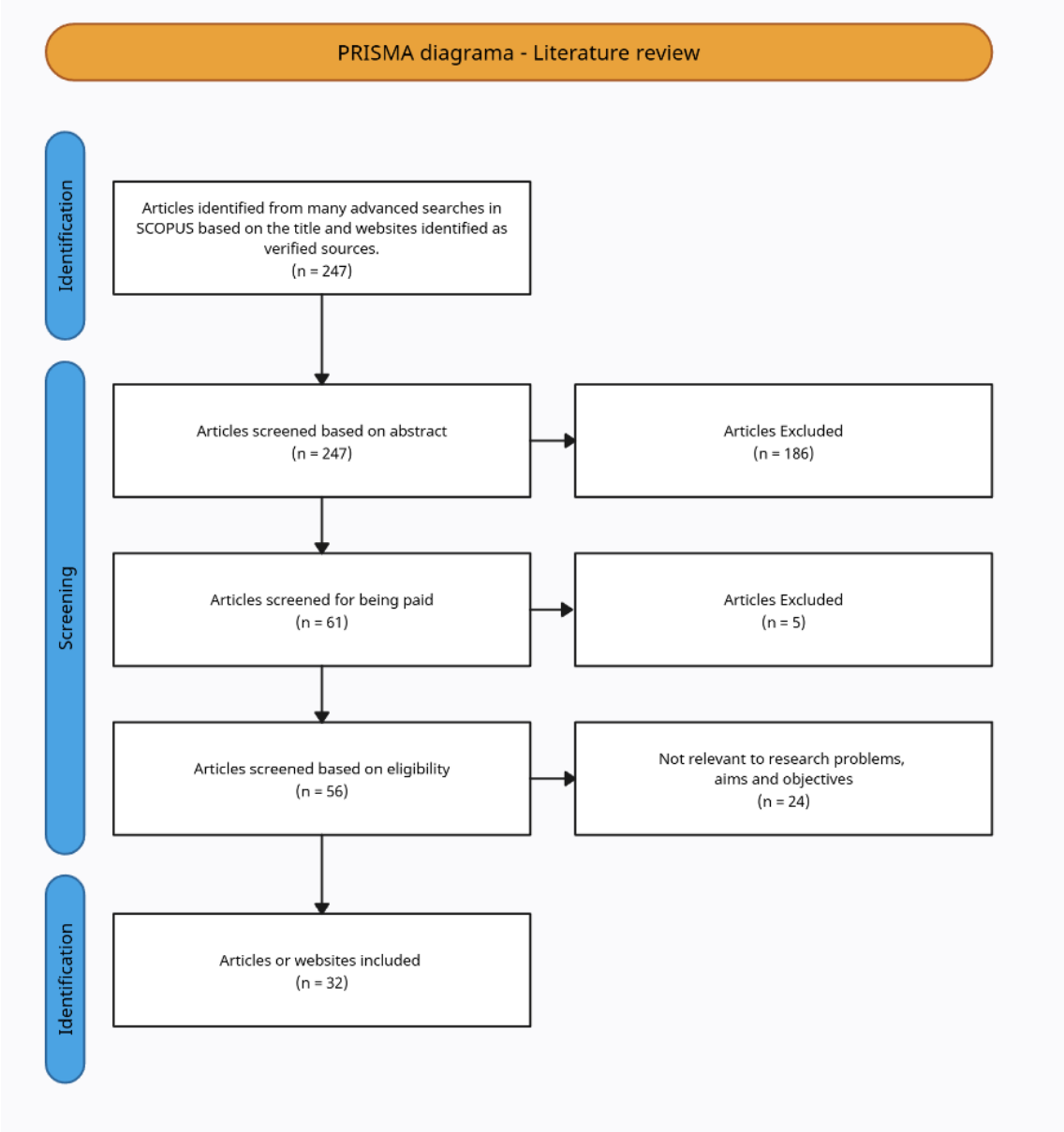


Figure 1.1: PRISMA diagram

To obtain the final list of articles, the PRISMA method (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) was employed. As illustrated in Figure 1.1, this method structured the literature review process to ensure a transparent, reproducible, and systematic approach. The PRISMA method involved multiple stages, starting with broad searches and filtering out irrelevant studies, followed by a more refined selection based on quality and relevance to the research scope.

The process began with screening articles based on their titles and abstracts to assess their relevance to the dissertation's focus. This initial screening helped narrow down a broad pool of articles to those most pertinent to the key themes of the research. The articles that passed this first

screening were then stored in curated SCOPUS lists, which were further refined through a more focused review.

At this stage, the articles were reviewed in detail and those that most closely aligned with the research objectives were selected for in-depth analysis. The review aimed to gather information that could contribute directly to the development of the simulation tool and improve understanding of the key technologies involved.

In addition to SCOPUS, Google NotebookLM was used as a supplementary tool to aid in organizing and analyzing the shortlisted articles. NotebookLM's AI-powered features were particularly useful in the following ways:

- **Summarization**: It generated concise summaries of the key articles, which helped save time in identifying the most important findings.
- **Querying Specific Topics**: NotebookLM enabled me to query specific topics across multiple articles, allowing for focused extraction of relevant information.
- **Organizing Information**: It assisted in organizing large volumes of data from the literature, creating a more structured and accessible reference database for the dissertatio

To organize and manage the citations and key information from the articles reviewed, Mendeley was used as the primary reference management tool. Mendeley facilitated the saving of full-text articles, specific quotes, and important parts extracted during the research process. These critical pieces of information were collected through a combination of personal reading and insight from NotebookLM. This approach ensured that all references were well organized and readily accessible for analysis and synthesis.

Together, these tools and methods formed an integrated research strategy. SCOPUS provided the breadth of access to academic literature, while NotebookLM and Mendeley streamlined the analysis and organization of the most relevant sources. This combination ensured a robust foundation for the development of the state-of-the-art.

1.4.2 Simulation Framework Development

The development of the simulation framework was the first significant step in the research process. Tthe framework was structured in two main components: logic and Graphical User Interface (GUI), each with different phases of development.

The logic component was developed using SimPy, a Python-based, open-source discrete-event simulation library. The first phase involved building simple test models to understand SimPy's syntax and functionality. These models, such as a basic processes simulation helped grasp the

event-driven structure of the tool. This foundational work was then expanded to simulate more complex processes which introduced greater complexity and more accurately mirrored industrial assembly tasks.

The GUI was developed using PyQt5, allowing users to interact with the simulation, adjust parameters, and view results. The GUI was initially basic, serving as a debugging tool, and later evolved into a comprehensive interface capable of managing simulation scenarios, running tests, and outputting key performance metrics such as cycle time and throughput.

1.4.3 Data Collection and Case Study

The real-world data from Bosch Security Systems was obtained later in the research process, once the simulation framework had already been developed. This data, which included cycle times, task completion rates, and resource utilization, was essential for testing and validating the simulation model.

The data was collected through a case study of L05B and L05C assembly lines at Bosch, which involved both process mapping and direct interactions with Bosch engineers and operators. Initially, process diagrams were created to visualize the workflows of the assembly lines, including interactions between operators and machines. These diagrams served as the inspiration for developing the simulation models.

Once the real-world data were received, it was used to calibrate the simulation model. The model was refined by comparing simulated results with Bosch's actual performance, ensuring that it reflected the operational realities of the assembly lines. This data-driven approach enabled the validation and fine-tuning of the framework, ensuring that the simulation tool accurately reflected Bosch's production system.

1.4.4 Scenario Analysis and Testing

After the simulation model was calibrated with Bosch's real data, scenario analysis was conducted to evaluate the impact of different configurations on production efficiency. The key scenarios tested included variations in task allocation, the introduction of CoBots at different stages of the assembly process, and changes to the line configuration.

Each scenario was assessed based on Key Performance Indicators (KPIs) such as cycle time, throughput, resource utilization, and efficiency. This phase of the research helped identify the configurations that yielded the highest productivity and provided actionable insights into optimizing human-robot collaboration at Bosch.

1.4.5 Model Validation

The validation of the simulation model was conducted by comparing the simulation results with the real-world data provided by Bosch. This step was crucial to ensure that the simulation accurately reflected Bosch's actual production processes.

Discrepancies between the simulated and real-world data were identified and analyzed. The model was refined iteratively based on this analysis, improving its accuracy and alignment with Bosch's operational performance. Feedback from Bosch engineers was also incorporated into the validation process, ensuring that the simulation tool met both technical and practical requirements.

1.4.6 Analysis and Conclusion

Upon completing the validation and testing phases, the focus shifted to critically evaluating the proof-of-concept nature of the simulation framework. The primary goal of this dissertation was not only to design and develop a simulation tool but to demonstrate that it could effectively model and optimize human-robot collaboration in assembly lines. While the cost-benefit analysis was initially intended, the research pivoted toward establishing the viability and functionality of the tool, ensuring it could serve as a working prototype capable of addressing real-world challenges.

The simulation model was validated through a comparison with real-world data from Bosch's assembly lines. This validation ensured that the tool could accurately reflect operational realities and provide meaningful insights into potential improvements. The testing phase focused on proving that the simulation tool could be used to evaluate different task allocation configurations, test the integration of cobots, and model line configurations in a way that is both realistic and actionable for engineers.

While the analysis did not involve a formal cost-benefit study, the results strongly indicated that the tool has the potential to optimize human-robot collaboration in real-world manufacturing environments. The proof-of-concept nature of the tool was successfully demonstrated.

1.5 Structure of the Dissertation

This dissertation unfolds into the development of a simulation framework designed to optimize task allocation within HRC assembly lines. The introduction establishes the motivation for the research, outlines the problem addressed, and clearly defines the objectives of the study. It also presents the research methodology, offering a structured approach to the analysis and providing a clear roadmap for the work to follow.

Chapter 2 presents a comprehensive review of the literature, focusing on key advancements in Industry 4.0, HRC, DTs, and simulation technologies. This chapter contextualises the research

within broader academic and industrial contexts, emphasising the concepts and knowledge necessary for understanding the simulation framework.

Chapter 3 delves into the development of the simulation framework itself. It provides an in-depth examination of the goals and design principles that guided the framework's creation. The chapter outlines the rationale behind the selection of specific technologies and tools, as well as the process through which both the simulation logic and GUI were designed and implemented.

Chapter 4 focuses on a case study involving Bosch's assembly lines, where the simulation framework is applied to real-world production data. This chapter details the process of data collection, the modeling of assembly line operations, and the evaluation of different scenarios. It illustrates the practical utility of the framework and provides a validation of its effectiveness in a real industrial context.

In Chapter 5, the dissertation shifts to the analysis of the results generated by the simulation framework. This chapter compares the performance outcomes of various scenarios, evaluating key performance indicators such as cycle time, throughput, and resource utilization. By comparing the simulated results with real-world data, the analysis demonstrates the framework's ability to accurately reflect operational realities and offers insights into potential improvements in production efficiency.

Chapter 6 concludes the dissertation by summarizing the findings, assessing the contributions of the research, and proposing avenues for future work. This chapter reflects on the success of the simulation framework in meeting the research objectives, while also suggesting potential improvements and expansions that could further enhance its applicability in industrial environments. It concludes with a discussion on the broader impact of the framework and its potential to support ongoing innovation in human-robot collaboration.

Chapter 2

State-of-the-Art

The literature review in this chapter provides a comprehensive foundation for this dissertation, contextualizing its objectives within the wider academic and industrial landscape. By examining existing research and methodologies, this chapter highlights the advancements and challenges in Industry 4.0, HRC, DTs, and Simulation tools. These areas are crucial to understanding the technological and theoretical underpinnings required to develop an effective simulation framework for optimizing task allocation in HRC environments.

This chapter begins with an overview of Industry 4.0 and its significant influence on manufacturing systems, followed by an exploration of HRC and its evolving role in modern assembly lines. The discussion then delves into the DT technology, emphasizing its applications for validation and scenario testing. Following this, the methodologies for modeling processes on assembly lines are succinctly examined to emphasize their significance within the framework of DTs. Finally, the chapter examines simulation frameworks and tools, focusing on open-source solutions and their integration with DTs. Together, these sections form the backbone of the dissertation's guiding the subsequent development of a simulation framework tailored to Bosch Ovar's assembly processes.

2.1 Industry 4.0 in Modern Manufacturing

The emergence of Industry 4.0 signifies a pivotal period in manufacturing, characterized by the convergence of digital and physical systems to create intelligent and adaptive production environments. The core principles that guide the development and implementation of smart manufacturing systems are often referred to as the Design Principles (DPs) of Industry 4.0, which aim to create interconnected, flexible, and data-driven environments [20, 23]. The DPs pertinent to the theme of this dissertation are:

1. **Interoperability**: is the ability of "Two or more components to cooperate, communicate, and interact despite differences in language, interface, and execution platform" [20]. In

practical terms, interoperability allows manufacturers to integrate various technologies, ensuring that they work together seamlessly. This capability facilitates the creation of more intelligent, versatile, and efficient factories [24].

2. **Virtualization**: is a principle that enables the creation of DTs, virtual representations of physical systems or processes [20, 27]. These digital models integrate data from sensors and other sources to provide a comprehensive, real-time reflection of the physical environment [20].

3. **Decentralization**: highlights the importance of distributing decision-making abilities and data generation across different points within a system, rather than consolidating them under a central authority [24]. According to [20], decentralization "refers to the development process data that are not centrally gathered, elaborated or controlled. Indeed, data can be accessed from anywhere.".

   This approach allows individual components, such as machines, devices, and systems, to autonomously make decisions utilizing real-time data in an effort to optimize their own objectives. Decentralization improves the robustness of a system by minimizing bottlenecks related to centralized authority. By distributing decision-making, the system can continue to function even if one part encounters a failure, as other components can adjust accordingly [24].

4. **Real-Time Capability**: Allows systems to process and analyze data instantaneously to make informed decisions and adjustments [20]. This principle is crucial to maintaining the responsiveness and flexibility required in modern manufacturing, where production lines and supply chains must react dynamically to changing conditions and demands [20].

5. **Modularity**: "refers to the capability of a system to be decomposed into modules. The modules can be changed and adapted to the specific product according to given requirements, increasing the manufacturing agility and flexibility." [20].

   This enables "decreasing levels of complexity to a point where simulation, visualization, and decision-making can be distributed and used to facilitate control and actuation on the real world." [24].

2.2 Human-Robot Collaboration

HRC is a modern work concept that holds great promise in improving efficiency and ergonomics, particularly in assembly line environments [17, 29]. HRC integrates the cognitive and creative abilities of human workers with the precision, strength, and consistency of robotic systems [35]. CoBots are a specialized type of robot designed to share workspaces and tasks with human workers. Unlike traditional industrial robots, which often require segregation for safety, CoBots are equipped with advanced sensing systems that enable them to operate safely alongside humans [35].

2.2.1 Collaborative Scenarios

Performed tasks of CoBots and humans in processes can be categorized into various types:

- **Independent**: "In this scenario, a human worker and a cobot process two tasks separately on two different workpieces" [17]. As clarified by [29], "the human and robot work side to side but do not share a working space".
- **Sequential**: "This scenario occurs when a human worker and a cobot process two different tasks on the same workpiece" [17], each contributing to the overall completion of the task.
- **Simultaneous**: "Here, a human worker and a cobot process two different tasks on the same workpiece at the same time" [17], with "robots reacting in real-time to human worker movement" [35].
- **Supportive**: "When a human worker and a cobot interactively process a single task on a single workpiece" [17], such as when "a cobot holds the workpiece and a human worker fastens the screw" [17].

2.2.2 Human-Robot Collaboration in Assembly Lines

Implementing HRC is vital for assembly lines in industrial manufacturing, as integrating robots into workspaces shared with humans enhances productivity, minimizes fatigue from repetitive tasks, and decreases the likelihood of job-related complexity or accidents for human workers [35]. CoBots perform tasks such as part handling, fastening, and precision assembly, complementing human workers who focus on quality control [4]. In addition, the flexibility of CoBots enables manufacturers to quickly adjust assembly lines to meet evolving production demands [17].

The adoption of CoBots is increasing in the manufacturing sector as they offer several benefits, such as:

- **Increased Efficiency and Productivity**: In striving for such adaptability on production lines, "manufacturing companies have started to heavily invest in 'collaborative workspaces', and as a consequence, in the close interaction between humans and robots for higher levels of efficiency, productivity" [29].
- **Enhanced Flexibility**: CoBots can be reprogrammed and redeployed with minimal effort, allowing manufacturers to adapt to new products or market demands [35]. The adaptability of CoBots makes them essential in ever-changing production settings [6].
- **Improved Ergonomics and Safety**: CoBots can improve ergonomics at work by helping with tasks that may cause strain or injuries to human workers [17, 35]. They can also be used in hazardous environments, such as disassembly lines, where workers may be exposed to dangerous materials or tasks [17].

- **Job Enhancement and Upskilling**: HRC does not necessarily replace human workers; instead, it has the potential to enhance their roles by allowing them to focus on tasks that require problem solving, creativity, and decision making, which leads to more engaging and fulfilling work experiences [17].

2.2.3 Safety Concerns in Human-Robot Collaboration

Safety is a crucial aspect of HRC, representing one of the most important considerations when designing and deploying CoBots in industrial environments. The close interaction between humans and CoBots in a shared workspace requires careful consideration of safety measures to prevent accidents and injuries. The design of CoBots places safety at the core, as emphasized by the *ISO 10218-1* and *ISO 10218-2* standards, which outline specific safety requirements for collaborative work systems [17]. These standards allow workers and CoBots to performe in the same workspace without safety risks, representing a shift away from the restrictive environments that were once necessary for industrial robots. In addition, newer guidelines such as *ISO/TS 15066* provide detailed biomechanical limits and measurement regulations to implement safer shared workplaces between humans and robots [29, 35].

The safety strategies defined in these standards can be categorized into four operational modes [4]:

- **Safe Stop**: The robot ceases all movement when a human enters the shared workspace.
- **Hand Guiding**: The robot operates only when directly guided by a human.
- **Speed and Distance Surveillance**: The robot adjusts its speed based on its proximity to a human, ensuring a safe distance is maintained.
- **Force and Power Limitation**: The construction or control mechanisms of the robot prevent injuries by limiting the force and power it can exert.

These strategies aim to mitigate risks through comprehensive procedures such as risk analysis and the development of risk minimization concepts, as prescribed by standards such as *DIN EN ISO 12100* and *DIN EN ISO 13849* [4]. Functional safety is further defined by *IEC 61508*, which is tailored to various technologies to improve the safety of shared workspaces between humans and robots [4].

Although significant progress has been made in the implementation of these safety measures, challenges remain. Current industrial applications predominantly utilize the first three safety strategies. Inherent safety achieved through the limitation of force and power, based on biomechanical research, is still an active area of study and development [4, 17].

2.3 Digital Twin Technology

DTs are digital replicas of physical systems, processes, or objects. The key difference between a DT and other digital models is the automatic and bidirectional exchange of data between the

digital and physical twins in real time [31]. The purpose of creating a DT is to improve the performance of the physical object using computational simulations and techniques [27]. The connection between the DT and its physical counterpart allows for real-time monitoring, analysis, and optimization [15]. This data can then be used to drive innovation, improve production processes and performance, and ensure continuity and traceability of information [27].

2.3.1 Digital Twins and Industry 4.0

The concept of DTs aligns seamlessly with the principles of Industry 4.0, which emphasize connectivity, adaptability, and data-driven decision making. DTs embody virtualization, creating dynamic digital models that mirror physical systems. These models leverage real-time data exchange to continuously update and adapt, enabling predictive analytics and scenario testing [15, 16]. They facilitate the integration of cyberphysical systems by acting as virtual counterparts to physical components, enabling seamless interaction between the physical and digital layers of manufacturing systems [22]. By supporting interoperability, DTs seamlessly connect to IoT devices, cloud systems, and advanced analytics platforms, forming the backbone of smart manufacturing environments [22, 31]. A core advantage of DTs in Industry 4.0 lies in their predictive capabilities. By simulating potential scenarios in a virtual environment, manufacturers can identify and address inefficiencies, predict failures, and optimize workflows before implementing changes in the physical system [15, 22, 31].

2.3.2 Digital Twins and Human-Robot Collaboration

In the context of HRC, DTs play a pivotal role by providing a platform to optimize the interaction between humans and robots. DTs enable the simulation of HRC scenarios, allowing manufacturers to design workflows that maximize productivity while ensuring worker safety. DTs also help design and optimize HRC tasks, guaranteeing that physical implementations are efficient and safe [16].

Within collaborative settings, DTs track real-time data from both robots and human operators, allowing for adaptive workflow modifications. For example, on an assembly line, a DT can simulate various task allocation methods, determining the optimal working distribution between humans and robots. This configuration ensures that robots handle repetitive or physically demanding tasks, while humans focus on making decisions and solving problems. DTs also simulate shared work environments to detect possible hazards and refine task assignment strategies. This methodology not only guarantees safety but also improves collaboration by ensuring smooth interaction between humans and robots [2, 16, 27].

2.3.3 Digital Twins and Simulations

Simulations are an integral component of DTs, enabling manufacturers to predict results, test scenarios, and optimize operations without disrupting the physical system. DT-driven simulations

are essential in dynamic and complex environments, where real-time changes can have significant impacts. For example, on reconfigurable assembly lines, DTs simulate the effects of new task sequences, robot configurations, or product variations, ensuring that physical implementations meet performance expectations [27].

The iterative nature of simulations within DTs supports continuous improvement. It further diminishes the hazards and expenses linked to trial-and-error testing on actual production lines, allowing manufacturers to explore "what-if" scenarios to assess the effects of alterations, like implementing new robotic systems or adjusting human-robot interactions. This capability is particularly valuable in HRC, where task allocation and safety require careful planning and refinement [16]. Furthermore, the combination of simulations and real-time data analytics enables DTs to offer predictive maintenance insights, reducing downtime and improving system reliability [31].

2.4 Process Modeling in Assembly Lines

Process modeling is a crucial preparatory step in developing effective simulation frameworks on assembly lines. By providing a structured representation of workflows, resource allocations, and task interdependencies, process modeling ensures that simulations are accurate and meaningful. A precise and dynamic process model provides the foundation for evaluating and optimizing workflows, which is essential before implementing simulations [27]. Without this step, simulations risk being incomplete or misaligned with real-world operations.

In assembly lines, where tasks are interconnected and involve humans, robots, and automated systems, process modeling clarifies the behavior of the system. Identifies bottlenecks, inefficiencies, and safety risks, which can be addressed during simulations. For example, before introducing CoBots, process models define task allocation and ergonomic factors to ensure seamless integration. Task allocation models are critical for simulating HRC scenarios, ensuring that physical implementations are efficient and ergonomic [29].

Process modeling also defines the scope of simulations, guiding what aspects of the system, such as task sequences or system configurations, need to be simulated. This clarity ensures that the simulations are targeted and comprehensive, avoiding wasted resources on irrelevant aspects.

Moreover, process modeling supports iterative refinement. By establishing an accurate baseline, manufacturers can adjust simulations as new data or requirements emerge, minimizing errors and disruptions. For example, on reconfigurable assembly lines, process models allow virtual configuration testing to ensure smooth implementation [14, 22]. This approach ensures that the simulation results are actionable, predicting the impact of new technologies or workflow optimizations [18, 35].

Finally, process modeling fosters collaboration among stakeholders by providing a visual representation of system processes. This shared understanding ensures that the simulations are tailored to address key challenges and align with the organization's operational objectives. By creating clear and comprehensive process models, stakeholders can work together more effectively, ensuring that simulations are both relevant and aligned with strategic goals [17].

2.4.1 Continuous-Time and Discrete-Time Modeling

In the realm of system simulation, understanding the distinction between continuous-time and discrete-time modeling is fundamental. These approaches define how systems are represented and analyzed over time, influencing the choice of simulation tools and methodologies.

2.4.1.1 Continuous-Time Modeling

Continuous-time modeling represents systems where variables change continuously over time. This approach is particularly suited for physical systems governed by differential equations, such as mechanical, electrical, and fluid systems. In continuous-time models, the state of the system is described by variables that evolve smoothly, without abrupt changes. This modeling is essential for simulating processes where changes occur in a continuous manner, such as the flow of electricity in a circuit or the movement of a mechanical component [26].

2.4.1.2 Discrete-Time Modeling

Discrete-time modeling represents systems in which variables change at distinct, separate points in time. This approach is ideal for systems where events occur at specific intervals, such as digital systems, queuing systems, or processes with discrete events. Discrete-time models are characterized by variables that change in steps, with each step occurring at a specific time point. This modeling is crucial for simulating systems where changes are triggered by specific events or occur at fixed intervals, such as the processing of transactions in a computer system or the arrival of customers at a service point [26].

2.4.2 Hybrid Modeling

Many real-world systems exhibit the characteristics of continuous and discrete behavior. Hybrid modeling addresses this by integrating continuous-time and discrete-time models into a unified framework. This approach is particularly useful in complex systems where continuous processes interact with discrete events, such as in manufacturing systems where machines operate continuously, but parts are processed in discrete steps. Hybrid models enable a more comprehensive simulation of such systems, taking into account the interactions between continuous and discrete components [26].

2.4.3 Agent-Based Modeling

Agent-Based Modeling (ABM) is a computational approach that simulates the actions and interactions of autonomous agents, individual or collective entities such as organizations or groups, to understand complex system behaviors and outcomes. In ABM, agents operate on the basis of defined rules and can adapt their behaviors based on interactions with other agents and their environment. This modeling technique is widely used across various disciplines, including economics, social sciences, ecology, and engineering, to analyze phenomena such as market dynamics, social behaviors, and ecological systems. By focusing on individual behaviors and interactions, ABM provides insight into how complex patterns emerge from simple rules, making it a valuable tool to understand and predict system behaviors in diverse fields [30].

2.5 Simulation Tools

Simulation tools play a crucial role in the effective deployment and enhancement of DTs systems within Industry 4.0. These tools allow manufacturers to replicate, assess and refine complex systems prior to their application in actual settings, thereby minimizing both risk and expense substantially. This section explores four widely used simulation tools in the development of DTs: OpenModelica, AnyLogic, FlexSim, and SimPy when combined with Blender. We examine the features and advantages each tool offers. In addition, the discussion includes a comparison of open-source and proprietary solutions, highlighting essential factors to consider when choosing a simulation tool.

2.5.1 Open-Source Simulation Tools

Open-source simulation tools are software platforms available at no cost for use, modification, and distribution. These tools allow users to model and simulate without the restrictions of proprietary software licenses. In academic settings, where cost-effectiveness and the ability to adapt software to unique requirements are essential, open-source simulation platforms offer substantial advantages. In addition, these tools are frequently supported by active communities, which encourage ongoing improvements and innovation.

2.5.1.1 OpenModelica

OpenModelica is an open-source model, simulation, and development environment ([11, 12]). It is based on the equation-based and object-oriented Modelica language and offers a wide range of features, including debugging, optimization, visualization, and 3D animation. OpenModelica also includes various scripting languages such as Python, Julia, and MATLAB [12]. The environment aims to provide a complete implementation of Modelica with industrial strength for various purposes such as modeling, simulation, system optimization, and model-based development [11]. It is particularly well-suited for building complex DTs due to its support for visual design, predefined model building blocks, reusable components, and multi-domain modeling capabilities [12].

The OpenModelica framework provides several integrated subsystems, such as the OpenModelica Compiler, OMEdit for graphical model editing, and OMSimulator for co-simulation [12]. Furthermore, OpenModelica supports 3D visualization and animation, enabling real-time representation of system behavior, which is crucial to interact with and analyze DTs [11, 12].

- **Open-source and Cost-effective**: OpenModelica is completely free to use, making it an attractive option for academic research and commercial applications where budget constraints may be a concern [12, 34].
- **Extensibility**: OpenModelica's support for the Modelica language allows users to easily extend and create custom models. According to [12], OpenModelica Environment also supports hybrid modeling: "Modeling of both continuous-time and discrete-time aspects of systems is supported in an integrated way." This is important for realistic simulations of physical systems.
- **Extensive Model Libraries**: OpenModelica comes with a rich set of libraries, including "The most important library is the Modelica Standard Library (MSL) (Modelica Association 2020). MSL version 4.0.0, released in 2020, contains about 1400 model components and 1200 functions from many domains" [12].
- **Inclusive Modeling**: OpenModelica supports the Functional Mock-up Interface (FMI) standard, which allows models from non-Modelica tools compiled into FMUs to be simulated [11].
- **Interactive Scripting**: OpenModelica allows integration with Python, Julia, and MATLAB for scripting and automation of tasks, enhancing flexibility in system analysis [11].

2.5.2 SimPy

SimPy is an open-source process-oriented discrete-event simulation framework that is implemented in standard Python [25]. Python's flexibility and broad library support make it a favored option for carrying out simulation tasks. With SimPy, you can define resources such as machines or workers and manage processes such as the assembly stages of a product [28]. By varying the number of workers, machine speeds, or part arrival rates, SimPy allows you to simulate various scenarios and assess the results to determine the most efficient setup for your assembly line [28]. Its simplicity and the availability of numerous libraries facilitate the creation and execution of intricate simulations in multiple fields, including the manufacturing sector [38]. SimPy provides strong simulation features, but lacks a built-in GUI for model configuration and visual interaction. Nevertheless, its implementation in Python allows seamless integration with a wide range of libraries, such as NumPy, Pandas, SciPy, and Matplotlib, which support data analysis, statistical modeling, and result visualization. Moreover, by combining SimPy with GUI frameworks

like PyQt or Tkinter, it is possible to build fully customized interfaces that enhance usability and provide interactive design environments for discrete-event simulations. This flexibility enables developers to overcome SimPy's minimal visual layer while retaining full control over simulation logic and data flow [28, 38].

- **Ease of Use**: SimPy provides a simple and intuitive syntax that is easy to learn and use, even for users without extensive programming experience [25].
- **Extensive Libraries**: As part of the Python ecosystem, SimPy integrates a vast collection of libraries for data analysis, visualization, and optimization capabilities [38].
- **Community Support**: A large and active community provides extensive documentation, tutorials, and forums that facilitate problem solving and knowledge sharing [28, 38].
- **Flexibility**: SimPy can model a wide range of systems, from simple queues to complex manufacturing processes and service centers [25, 38].

Integrating SimPy with a GUI enhances the simulation experience by allowing users to visually configure and interact with complex system models. Rather than relying on immersive 3D environments, this approach focuses on intuitive and accessible 2D representations that facilitate decision-making and productivity analysis.

2.5.3 Closed-Source Simulation Tools

Closed-source simulation tools are proprietary software platforms that are developed, maintained, and distributed by commercial entities under restrictive licenses. These tools typically offer advanced features, a well-developed interface, comprehensive documentation, and robust support, making them suitable for complex simulations in industrial and enterprise environments. As a result, closed-source simulation tools are widely adopted in commercial applications where reliability, performance, and long-term vendor support are essential.

2.5.3.1 AnyLogic

AnyLogic, a highly adaptable simulation modeling software, facilitates three main simulation techniques: discrete event, ABM, and system dynamics modeling, allowing thorough analysis of manufacturing systems [33]. The software features an intuitive drag-and-drop interface and includes libraries tailored to specific industries. These libraries improve the modeling process by offering predesigned components suitable for different sectors [7]. AnyLogic also integrates with cloud platforms, enabling users to run simulations online, collaborate with team members, and share results efficiently. This cloud integration supports real-time data analysis and decision making, streamlining the simulation process [8].

- **Versatility in Modeling**: AnyLogic can simulate systems of any complexity, which makes it suitable for a wide range of applications, from simple to highly intricate models [1].
- **User-Friendly Interface**: AnyLogic has a modern graphical interface and allows the use of the Java programming language to develop simulation models [1, 7].
- **Visualization**: AnyLogic allows for the visualization of the production system and provides key performance indicators for production planning, which can be displayed in a variety of chart types, including bar charts, pie charts, and plots. The user can select between predefined representations provided by AnyLogic or create individual representations of resources, warehouses, and products [1].
- **Multiple Modeling Support**: AnyLogic supports multiple modeling paradigms, including ABM, discrete event modeling, and system dynamics modeling [1].

Limitations

- **Cost**: As a proprietary software, AnyLogic requires a paid license, which may be a consideration for smaller organizations or individual users.
- **Complexity for Beginners**. Although the interface is user-friendly, mastering the full range of features may require a significant investment of time and training.
- **Limited Real-Time Simulation**: Although AnyLogic supports real-time simulation, its capabilities in this area may not be as advanced as those of specialized real-time simulation tools.

[19, 32, 33]

2.5.3.2 FlexSim

FlexSim is a discrete event simulation software praised for its intuitive interface and robust 3D modeling and visualization features, which improve the precision of the representation of real-world systems [19].

- **User-Friendly Interface**: FlexSim offers an intuitive setting for model creation, making it well known for its ease of use in providing a specialized environment for manufacturing simulation [19].
- **Object-oriented approach**: FlexSim utilizes an object-oriented approach to simulation modeling. This approach facilitates the creation, reuse, and sharing of objects [21].

- **Integration with C++**: It integrates effortlessly with the C++ compiler, allowing developers to utilize C++ libraries and functions that support FlexScript, a precompiled C++ library [3].
- **Advanced Animation Capabilities**: Using OpenGL, FlexSim provides impressive virtual reality animation. Users can simultaneously view animations in 2D, 3D and virtual reality [3]. The 3D animation, in particular, stands out with its realistic, game-like graphics quality [3].
- **Real-Time Capabilities**: FlexSim offers real-time monitoring capabilities for dynamic flow systems through its connectivity to external programs [3]. This allows real-time information transfer to and from FlexSim, enabling users to monitor and control systems effectively [3].

2.5.4 Open-Source VS Closed-Source - Simulation Tools

This section compares open-source tools (OpenModelica and SimPy with Blender/Unity) with closed-source tools (AnyLogic and FlexSim), focusing on usability, flexibility, scalability, and visualization capabilities. The table below highlights the key differences between these tools to provide a clear understanding of their strengths and trade-offs for various simulation applications.

Feature	OpenModelica	SimPy + 2D GUI	AnyLogic	FlexSim
License Type	GPLv3	MIT	Paid License	Paid License
Flexibility / Customization	High	High	Moderate	Moderate
Ease of Use	Moderate	High	High	High
Community Support	Strong	Strong	Limited	Limited
Visualization	Moderate	Limited	Advanced	Advanced
Extensibility	High	High	Moderate	Moderate
Real-Time Capabilities	Moderate	Limited	High	High
Simulation Paradigms	Moderate	Low	High	Low
Documentation & Support	Extensive	Extensive	Comprehensive	Comprehensive

Table 2.1: Comparison of Open-Source and Closed-Source Simulation Tools

2.6 Simulation-Driven Decision Support in Hybrid Production Systems

A DSS is a computer-based system designed to assist decision-makers in solving semi-structured or complex problems by combining data, models, and user interaction. DSSs are not intended to replace human judgment but to enhance decision quality through structured analysis and informed exploration of alternatives [9].

Simulation-based DSS are increasingly adopted in flexible assembly lines to support tactical and strategic decisions. These systems allow managers to perform "what-if" scenario evaluations without interfering with the real production system, making it possible to compare the outcomes of different resource allocation strategies, operator quantities, and automation levels [10].

These capabilities are especially important in hybrid production systems, where human operators and robots collaborate across various workstations. Simulation models enable a modular and flexible analysis of these systems, allowing for the evaluation of resource allocation strategies and the identification of potential bottlenecks or underutilized resources. In the work of [37], a hybrid simulation-optimization methodology is proposed for automated assembly lines operated by robotic agents. Their approach combines discrete-event simulation with mathematical programming. While primarily focused on optimization, this study highlights the essential role of simulation in validating system performance by leveraging the power of DSS to explore alternative configurations while ensuring feasibility and efficiency.

Similarly, [36] applies a two-level decision-support approach in a HRC assembly line, using simulation to evaluate the impact of scheduling decisions on both human and robotic resources. This study reinforces the notion that DSS in hybrid production settings must incorporate human variability, task dependencies, and operational constraints. Although human behavior is not explicitly predicted, the simulation includes stochastic process durations and shift-level planning to facilitate robust production decision-making.

DSS simulation is an iterative problem-solving strategy that employs interactive interfaces and model-based reasoning to address the complexities of hybrid production environments. By enabling the exploration of multiple operational scenarios, it provides a structured way to assess the impact of different task distributions, resource configurations, and productivity assumptions [10].

2.7 Research Gaps

Despite growing interest in simulation-based DSS for industrial applications, significant gaps remain in the literature. While commercial platforms like FlexSim and AnyLogic are widely used, there is limited availability of open-source frameworks tailored for hybrid assembly line simulation. Their adoption is often hindered by a lack of standardized architectures, user-friendly interfaces, and comprehensive documentation, especially for simulating complex human-robot interactions.

Although discrete-event simulation is widely recognized as effective for modeling industrial processes, there is a shortage of examples using SimPy, a popular open-source library, to model assembly systems in realistic settings. Existing studies often do not offer reusable templates or

practical guidance, making it challenging to use SimPy effectively. Furthermore, no unified framework exists that combines cobot interaction, human task allocation, buffer management, and production flow simulation in an open architecture. This lack of comprehensive models complicates efforts to explore new production strategies or validate automation investments.

After an extensive review of the literature and careful screening of articles, no model-based simulation frameworks were identified that support the modeling of hybrid assembly lines in open-source environments. This absence of native modeling structures in publicly available or newly developed tools restricts the accessibility and reproducibility of simulation studies in this field. The significant lack of such frameworks highlights the originality and importance of the simulation approach proposed in this dissertation, which aims to fill this gap by offering a reusable and adaptable solution grounded in industrial reality.

Chapter 3

Development of the Simulation Framework

This chapter describes the conceptual and technical development of the simulation framework designed to model and simulate assembly lines. The framework was designed to support scenario evaluation and decision-making within industrial assembly cells, while maintaining a high degree of versatility and customization. Its open-source foundation facilitates ongoing development and adaptation to meet contemporary and future manufacturing needs. And it presents the structure and implementation of the simulation tool, along with the underlying modeling principles, the reasoning behind the selected toolchain, and the design strategies adopted to ensure extensibility, realism, and usability. The simulation logic and GUI were developed from scratch to meet the specific requirements of the Bosch Ovar assembly environment, ensuring ease of use and facilitating upgrades and changes.

3.1 Goals and Design Principles

The primary objective of the proposed framework is to provide a flexible, configurable, and realistic simulation environment to support scenario analysis and decision-making in industrial assembly cells. The simulation framework serves as a decision-support tool, allowing engineers and planners to manually model the production systems before running the simulations. This modeling process is streamlined through a GUI specifically tailored to the concepts and structures used within Bosch assembly lines.

Several key DPs were adopted throughout the development of the framework:

- **Modularity** - The simulation system is composed of distinct modules, simulation logic, GUI, data logging, and statistical methods. This modular implementation promotes easier future integrations, updates, and upgrades.

- **Interactive Visual Model Construction** - Users can craft the simulation models by interacting with a 2D graphical interface that allows dragging, connecting, and configuring elements.
- **Model Validation Mechanism** - To ensure simulation reliability, the framework includes a logic validation function that verifies essential conditions and model consistency before execution.
- **Open-Source Foundation** - The framework was built entirely using open-source technologies, granting full access to the simulation logic and enabling future adaptation to evolving requirements without licensing restrictions.
- **Configurability** - The tool enables users to modify parameters such as the number of operators in an assembly line, the integration of CoBots, processing times, movement durations, buffer sizes, and routing logic.

By combining these principles, the framework provides not only a powerful simulation engine but also a practical and accessible interface for modeling and evaluating the dynamics of hybrid human-machine systems in industrial settings.

3.2 Justification of the Selected Toolchain

To effectively simulate complex interactions between human operators and automated systems within industrial assembly environments, it was necessary to select a set of development tools that offered full control, adaptability, and transparency. For this reason, the framework was developed entirely in Python using two key open-source libraries: SimPy for discrete-event simulation and Qt (via PyQt5) for building the GUI.

3.2.1 Open-Source Motivation and Control over Logic

The primary motivation for selecting Simpy and Python as the simulation backbone was to retain full control over the modeling logic and system behavior during the development process. By avoiding proprietary or closed-source platforms, it became possible to design a fully transparent simulation framework, where all elements, such as machines, operators, buffers, and error and statistical conditions, can be defined with full flexibility.

SimPy's process-oriented paradigm and event-driven core are particularly well-suited for modeling discrete operations, such as processing times, operator availability, buffer delays, and routing between machines. While some initial learning was necessary, SimPy ultimately proves to be a robust foundation for building a custom simulation engine tailored to the specific needs of HRC.

Using an open-source foundation eliminates licensing concerns, enabling the framework to be used, adapted, and scaled freely within the company. SimPy's permissive MIT license further supports this advantage by permitting unrestricted use, modification, and distribution, even in commercial applications, without imposing legal or proprietary limitations. In an industrial context, all developed logic belongs to the organization that adopts the tool. Additionally, open-source development enhances long-term maintainability, as there are no external dependencies that could hinder the framework's evolution.

3.2.2 Graphical Interface Suitability

To complement the simulation logic and improve usability, a custom GUI was developed using Qt framework (PyQt5). This was done to provide an intuitive, visual environment in which users, particularly industrial engineers, could model assembly lines without writing code. PyQt5's 2D graphical and visual elements enable users to drag, position, and connect elements. The flexibility of Python and Qt meant that the interface could be designed specifically for this use case, free from the limitations typically imposed by general-purpose simulation platforms.

3.2.3 Limitations of Existing Tools

Several commercial simulation platforms, such as FlexSim and AnyLogic, were initially considered in the early stages of the project. While these tools offer robust features and advanced graphical environments, they were ultimately deemed unsuitable for this application. A primary concern was the restrictive licensing models of these platforms, which limited the ability to freely distribute or modify the simulation logic. In some cases, even the export of a fully defined model required a paid version of the software. Furthermore, many of the specialized elements needed to model hybrid human-robot assembly lines, such as manual task phases, semi-automatic logic, and operator allocation rules, were either unavailable or accessible only through paid extensions.

Another challenge with these commercial platforms was the lack of transparency in their underlying simulation processes, which hindered the ability to adjust or validate model assumptions. By contrast, developing the framework from scratch using Python and SimPy provided complete control over the simulation logic. This approach allowed for full transparency, offering a clear overview of how simulation entities interact at every level. SimPy's open-source nature and flexibility enabled a higher degree of customization, making it possible to tailor the framework to both the broad needs of assembly lines and the specific intricacies of the task allocation process. This openness facilitated the integration of complex human-robot collaboration scenarios while ensuring the tool could evolve and adapt as new requirements emerged.

3.3 Architecture Overview and Module Responsibilities

The simulation framework was designed in a modular architecture, ensuring a clear separation of responsibilities between the GUI, simulation logic, and data handling. Each part of the framework is organized into separate files, with functionality encapsulated in distinct classes. This structure ensures clear separation of concerns, making the project easier to maintain, reuse, and extend in the future.

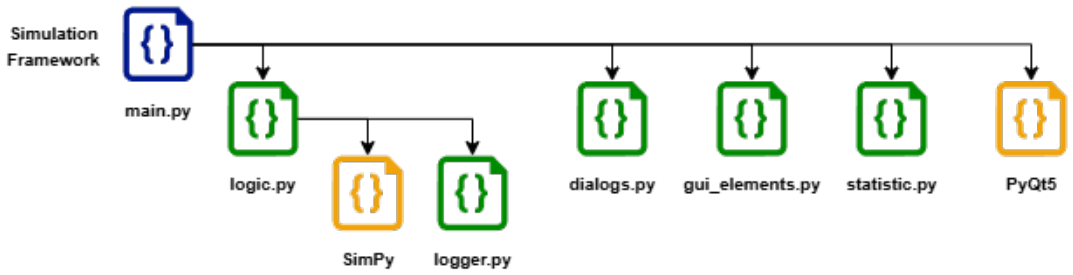


Figure 3.1: Simulation Framework Python File Hierarchy

At the core of the application lies the **main.py** module, which initializes and orchestrates the user interface, built upon the PyQt5 framework. It is responsible for creating the main application window, rendering the visual workspace, and managing user interactions such as adding elements, saving/loading configurations, launching simulations, and handling tab management for multiple simulation scenarios.

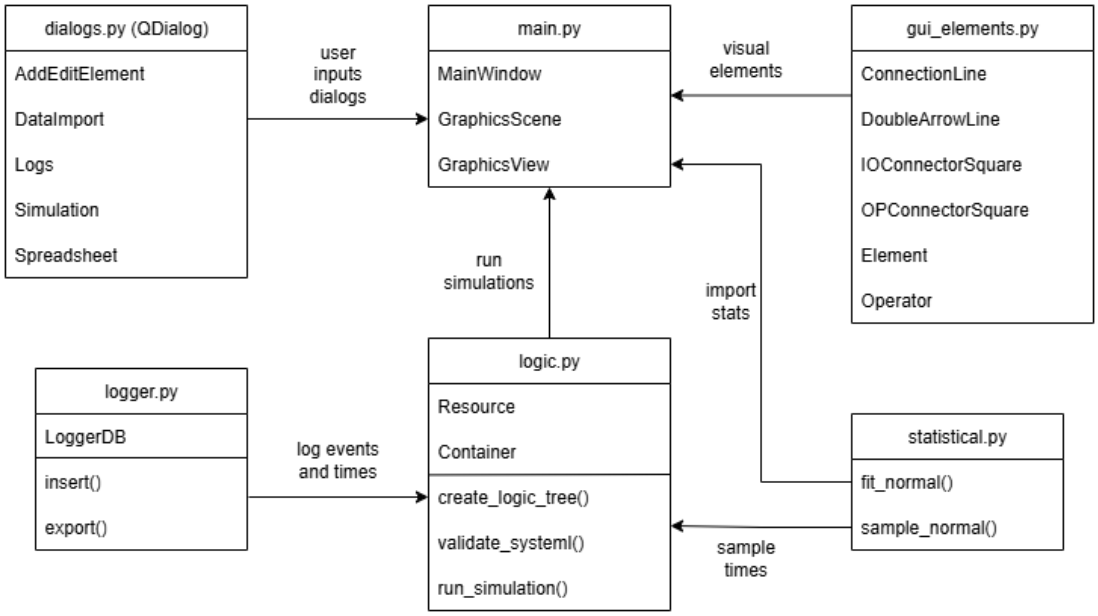


Figure 3.2: UML 'Architecture' Diagram (Shortened Version)

To support a visually intuitive experience, **main.py** interacts closely with two key modules: **gui_elements.py** and **dialogs.py**. The first defines the visual components of the assembly line, with graphical representations of logic nodes and connectors, all implemented through classes that inherit from QDialog. These elements represent various simulation entities such as machines, buffers, conveyors, operators,Inputs and Outputs (IOs), and connectors, that can be placed and configured on a 2D canvas. The **dialogs.py** module handles all dynamic dialog windows, including those for adding or editing elements, setting simulation parameters, and displaying success or error messages.

The **logic.py** module translates the graphical elements into SimPy entities and manages the event-driven simulation environment. It executes the process flow of each piece, handles resources, assigns operators to tasks, and controls the timing of operations based on user-defined parameters. Additionally, it includes a system validation function to ensure connectivity and consistency before running the simulation.

The **statistics_func.py** module acts as a utility layer for probabilistic modeling, providing functions to fit normal distributions from data and to sample process and loading times during the simulation for processes.

3.4 Discrete-Event Simulation with SimPy

The simulation core of the framework is built using SimPy, a process-based discrete-event simulation library for Python. SimPy allows the modeling of asynchronous, event-driven operations,

where multiple resources interact dynamically. This section details the main simulation entities, the model validation strategy, and the modeling logic used to simulate the behavior of hybrid assembly lines.

3.4.1 Simulation Entities

In logic.py, each modeled system is represented through a set of modular logical entities, each corresponding to a key physical component of an industrial assembly line. These entities are implemented as classes in the module, allowing for easy reuse and parameter customization. They utilize SimPy primitives like Resource and Container to model behavior and interactions.

A **simpy.Resource** models limited capacity resources that must be requested before use and released afterward. A **simpy.Container** represents a fixed-capacity storage unit, making it ideal for modeling buffers or inventories.

- **Operator** *(simpy.Resource)* - Models a human resource responsible for performing manual or assisted tasks in processes and transport operations.
- **Source** *(simpy.Resource)* - Represents the entry point of the system, where raw materials are introduced. Each simulation instance starts with the creation of piece processes in this Element.
- **Process** *(simpy.Resource)* - Represents a transformation operation within the assembly line flow. A process can be configured as automatic, manual, or semi-automatic, enabling the simulation of tasks performed by machines, human operators, or CoBots.
- **Buffer** *(simpy.Container)* - Stores intermediate parts temporarily between processing stages. Buffers are used to decouple timings and manage flow accumulation under variable processing durations.
- **Conveyor** *(simpy.Container)* - Represents a real conveyor responsible for transporting materials between processes. It can also be seen as a moving buffer that requires operator intervention for loading, while the unloading phase depends on the type of the subsequent Element in the process.
- **Target** *(simpy.Resource)* - Represents the endpoint of the assembly line. When a piece reaches a Target, it is marked as completed, and its process is terminated from the simulation.

3.4.2 Model Validation and Simulation Parameters

Before execution, the framework performs a structural node flow validation of the assembly line model. This step ensures the integrity and feasibility of the scenario for simulation:

- It verifies that at least one **Source** and one **Target** exist.
- It checks that all elements are correctly connected.
- It confirms that all assigned operators are valid and logically placed.
- It ensures that every possible path from a **Source** leads to at least one **Target**, forming a closed, executable system.

This validation is implemented in the ***validate_system()*** function in **logic.py** and is automatically triggered before the simulation starts. Users are informed of issues through the GUI.

Beforehand, the user-created visual assembly line is internally converted into structured logical nodes by the dedicated function ***create_logic_tree()***, which scans the canvas, classifies each Element, and builds the graph-based data model with properties like process type, assigned operators, time distributions, buffer capacities, and IOs connections.

3.4.3 Process-Based Modeling Concepts

The simulation follows a process-oriented modeling approach, where each piece acts as a discrete process moving through the assembly system. This allows for concurrency and flexibility, with each entity independently handling its flow and interactions with resources.

In SimPy, the simulation environment is created using an instance of **simpy.Environment**, which manages time progression and serves as the central event scheduler. It executes all processes and events in a **FIFO** order based on their scheduled times, ensuring chronological accuracy and resolving resource contention based on arrival times.

The core building blocks of SimPy include:

- **env.process()** , starts a new concurrent process in the environment
- **yield env.timeout(t)** , suspends a process for **t** units of simulated time
- **resource.request()** , requests exclusive access to a resource
- **yield resource.request**() , blocks the process until the resource is available
- **resource.release()** , returns the resource to the pool
- **env.run(until=x)** , runs the simulation until a given time or event

This modeling approach enables entities like operators and machines to operate independently while competing for shared resources within the system. Each entity follows its own logic and timing, creating a highly interactive and realistic simulation environment. As these entities progress through the system, resources are dynamically allocated and released based on availability and

demand. This configuration enables complex behaviors to naturally emerge, such as delays from resource contention, bottlenecks due to process queues, and congestion resulting from overlapping tasks.

3.4.4 Simulation Flow and Resource Scheduling

In the context of this framework, each piece begins its life cycle at a Source, where a corresponding SimPy process is instantiated to process its route through the assembly line. This process encapsulates the sequence of operations, resource requests, delays, and transitions that the piece will undergo until it reaches a **Target**.

At each step in the flow, the piece performs the simplified sequence of operations:

1. **Request access** to the current node in the path.
2. **Request an operator**, if the current process requires human intervention.
3. **Simulate a delay** corresponding to process time or handling time.
4. **Release a resource** once the operation is complete.
5. **Move to the next node**, following the defined path.
6. **Simulate transport**, transitioning to the next process.
7. **Release or retain an operator** based on the defined flow strategy.

The movement is dynamically defined at runtime, with each piece flowing independently and automatically queuing if resources are unavailable. Operators are created has discrete resources that can serve multiple elements depending on the scenario. The algorithm is equipped with safeguards to prevent over-allocation, and SimPy provides fair scheduling through its built-in queue mechanism. This process continues until a piece reaches the **Target**, at which point it is logged and its process is terminated. The simulation concludes either when a predefined number of pieces have been processed or when the specified simulation time has elapsed.

3.4.5 Time Handling

All time-related behaviors in the simulation are driven by stochastic models using normal distributions, defined for each Element and configured through the GUI or imported from real empirical datasets provided by Bosch for the case study.

The following durations are simulated:

- **Processing time**: Task duration per operation manual or automatic.
- **Loading/Unloading time**: Represents times to remove or insert pieces in machinery.
- **Transport time**: Movement times of parts as they travel through the assembly line.

Distributions are defined by a mean (**mu**) and standard deviation (**sigma**) and sampled using functions from ***statistics_func.py***. These simulated times help enhance realism by replicating the variability seen in real-world operations.

3.4.6 Error Events and Failure Simulation

To more accurately reflect the dynamic and uncertain nature of real-world assembly systems, the simulation framework incorporates two complementary event types related to operational reliability: piece conformity errors and process failures due to breakdowns.

Each process in the system includes an optional error rate parameter, which defines the probability that a processed part will be rejected due to non-conformity. This rate can be configured independently for each node through the GUI and represents issues such as operator-detected defects or rejections from quality tests. During simulation, each time a part completes its operation at a node, a random draw based on the defined error rate percentage determines if the piece is classified as defective.

If a defect is detected, the piece is immediately removed from the system and registered in the logs with a **Rejected** comment. It does not continue through the remaining nodes, and no further resources are allocated to it. This behavior reflects the logic commonly applied to automated quality control stations, particularly test machines, where faulty units are discarded on the spot.

The framework also supports the simulation of unexpected machine failures, representing unplanned downtimes such as mechanical failures or maintenance interruptions. This behavior is controlled by the following parameters:

- **MTBF** - average time between machine failures, relative to manual or semi-automatic processes or conveyors.
- **MTTR** - average time to repair.
- **MTBM** - average time between machine maintenance, relative to manual or semi-automatic processes or conveyors.
- **TSLM** - time since the last maintenance or repair was completed.
- **MTM** - average time required for maintenance completion.

These values can be configured by the user in the process configuration dialog. Internally, the simulation engine handles unexpected failures and scheduled maintenance as parallel mechanisms. Failures are triggered probabilistically based on the **MTBF** parameter, and result in the node being temporarily blocked for the duration defined by the **MTTR**. During this time, no new part may enter the process, and any pending tasks must wait.

When the **TSLM** value reaches the **MTBM** threshold, a preventive maintenance is triggered. This also blocks the node for the specified **MTM** duration. Importantly, the completion of a maintenance event resets the failure timer, postponing the occurrence of subsequent breakdowns. This model implements realistic Bosch OVar industrial practices, incorporating preventive maintenance to reduce the risk of unplanned failures.

Both types of downtime are automatically managed by the simulation engine, with all related events recorded in the log files. This enables users to evaluate how equipment reliability strategies affect overall system performance.

3.4.7 Features Not Implemented in the Simulation Framework

During the development of the simulation framework, several features were initially considered to enhance the model's realism and functionality. However, due to limitations in available data, these features were not implemented in the final version of the framework. Each of these potential additions is discussed below, along with the reasons for their exclusion.

3.4.7.1 Operator Characteristics

One of the features that was initially planned but ultimately not implemented was the incorporation of operator-level fatigue factors and skill variation. This feature aimed to account for the physical and mental fatigue of operators, as well as differences in their skill levels, which could influence task performance. While the concept of modeling fatigue and skill variation would have provided a more nuanced simulation of human performance, the absence of sufficient data to accurately model these factors led to their exclusion. Without reliable data to quantify fatigue levels or skill differences across operators, implementing this feature would have introduced unnecessary assumptions and potentially compromised the accuracy of the results.

3.4.7.2 Error Events and Failure Simulations

The parameters detail in the Section 3.4.6 were not included in the final model, they were set to 0 in each process. These features would have enabled the simulation to account for unplanned downtimes, maintenance needs, or machine failures, which are common in real-world production environments. However, the dataset used in this study was "clean," meaning that no maintenance or failure data was available to inform the modeling of such events. Given the lack of supporting data, introducing error events could have led to unrealistic assumptions and unreliable results. Therefore, the decision was made to proceed with the parameters set to null, focusing instead on simulating ideal operational conditions to assess performance under normal circumstances.

3.4.7.3 Simulations Results Dashboard

Another feature that was initially considered was the ability to save simulation runs and create a dashboard for uploading and comparing results within the framework. This feature would have

allowed for a more interactive and user-friendly experience, enabling users to store simulation results, visualize performance metrics, and perform comparative analyses across different scenarios. While this functionality would have enhanced the usability and analytical capabilities of the framework, it was ultimately deferred due to time constraints and the need to focus on core functionalities. This feature remains a potential avenue for future development, as it would add significant value in terms of data management and decision support.

3.5 Graphical User Interface with Qt

The simulation tool includes a GUI built with Qt that allows users to create, configure, and manage simulation scenarios visually. Instead of writing code, the user can drag and connect elements directly on a 2D canvas, assign operators, adjust simulation parameters, and validate the system before execution. The interface is organized to provide quick access to all key functionalities, such as element management, file handling, and simulation control, making it easier to build and run models that reflect real assembly line systems.

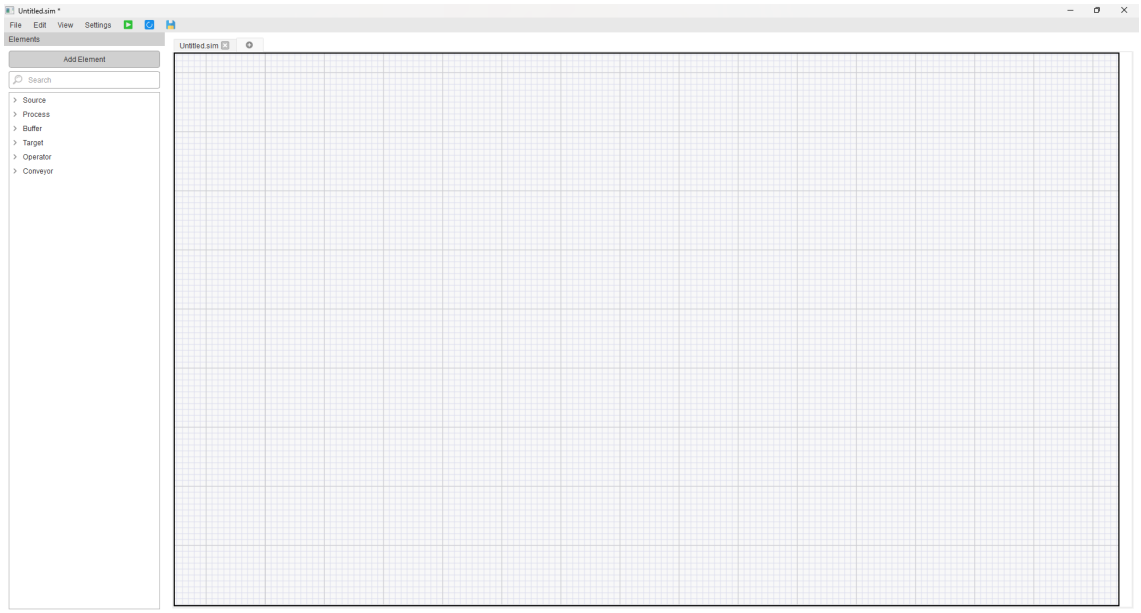


Figure 3.3: Simulation Framework Full Window

3.5.1 MainWindow Layout and Controls

The main window is the central component of the interface, integrating all primary actions for simulation creation, editing, validation, and execution. Its layout follows a familiar desktop application structure, combining a menu bar, toolbar icons, and a centered canvas for scenario overview.

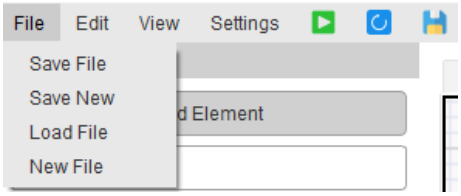


Figure 3.4: Top Menu Bar

At the top of the interface, a menu bar offers access to textual options. Currently, only the File menu is implemented, providing basic file operations:

- **Save File**: Saves the current scenario, either by overwriting or by creating a new filename if none exists.
- **Save New**: Saves the current scenario in complete new file.
- **Load File**: Loads an existing .sim file and reconstructs the corresponding scenario.
- **New File**: Opens a new blank scenario in a separate tab.

Next to the menu bar, three icon buttons allow quick access to core simulation operations:

- The **green play button** opens the simulation parameters dialog and allows the simulation to be run. Internally, it first validates the current model and, if successful, starts the simulation engine using the parameters defined by the user.
- The **blue validation button** checks the logical structure of the modeled system without executing the simulation, providing immediate feedback in case of missing connections or invalid configurations.
- The **save button** simulates the operation of the previously mentioned "Save File" button.

3.5.2 Element Palette and Canvas

On the left side of the interface, the element palette is organized into a vertical panel that allows users to add, search, and browse the elements required to model an assembly line.

At the top of the panel, the **AddElementt** button opens a configuration dialog where the user can define the properties of a new element.

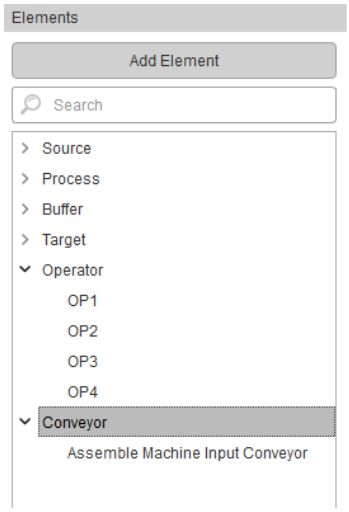


Figure 3.5: Elements Tree

The search bar allows filtering the element tree by either the name or class type, facilitating quick access in large projects. The element tree itself is structured by class, grouping elements together for easy navigation. The user can expand or collapse each class to view the available elements within.

To place an element on the workspace, the user can simply double-click it in the list, which spawns a copy directly onto the canvas.

The central area of the interface is the canvas, implemented using Qt's **QGraphicsScene** and **QGraphicsView**. This is where the simulation model is visually constructed, better viewed in Figure 3.3. Users can:

- Drag elements to reposition them freely.
- Connect elements using interactive IOs squares and operator connectors.
- Zoom in and zoom out, and move through the canvas.
- Right-click on any element to access a context menu with options to delete or edit its properties.

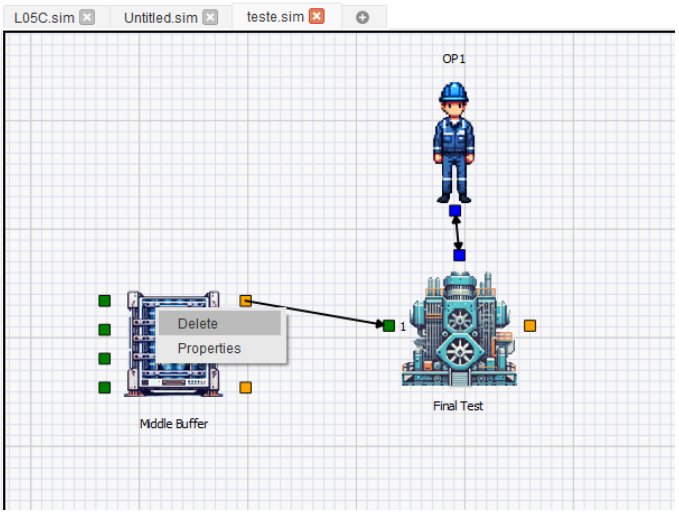


Figure 3.6: Canvas With Elements

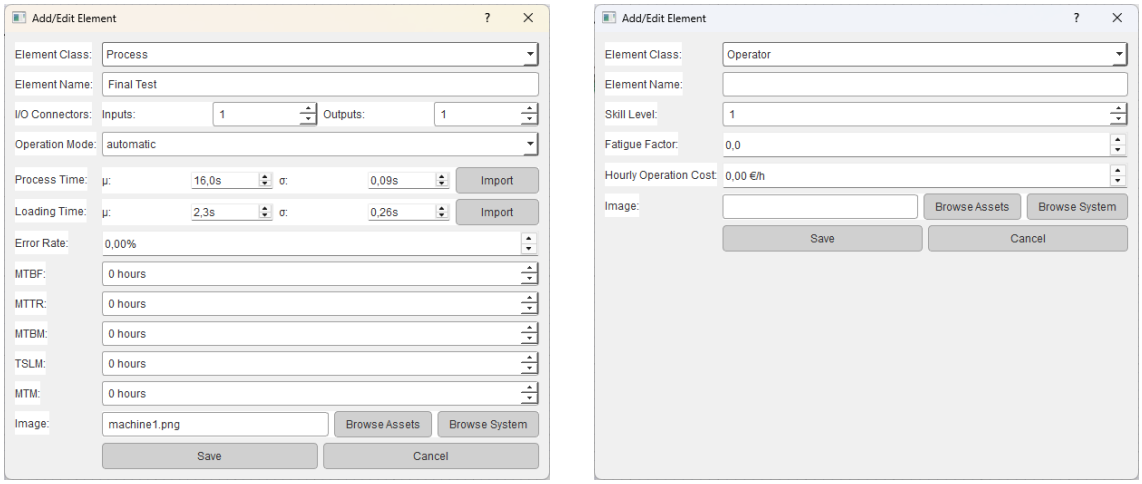
Above the canvas, a tab bar enables (just like a web browser) users to manage multiple scenarios within the same session. Each tab represents an independent scene, allowing separate configurations to be developed and tested in parallel.

3.5.3 Dialogs and Parameters

To configure the behavior of each element and define key simulation parameters, the interface makes use of a set of dialog windows that are displayed contextually as the user interacts with the model. These dialogs simplify data entry and enforce structure, ensuring that all necessary parameters are defined before execution.

3.5.3.1 Add/Edit Element Dialog

Whenever a user creates a new element using the **Add Element** button or edits an existing one from the canvas, the application opens the **Add/Edit Element Dialog**. This dialog automatically adjusts its layout based on the class of the selected element. The interface is organized into sections, with each section representing a specific group of parameters.



(a) AddEditElement Dialog Window - Process (b) AddEditElement Dialog Window - Operator

Figure 3.7: AddEditElement Dialog Windows

Time-based settings use paired input fields for average and standard deviation. When necessary, users can import time values directly from Excel or other sources by pasting numeric data into the input fields. The application then automatically fits a normal distribution to the imported values using statistical tools from **statistics_func.py**.

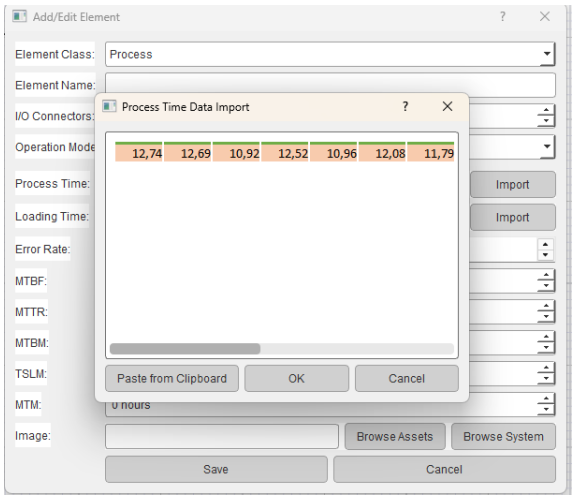
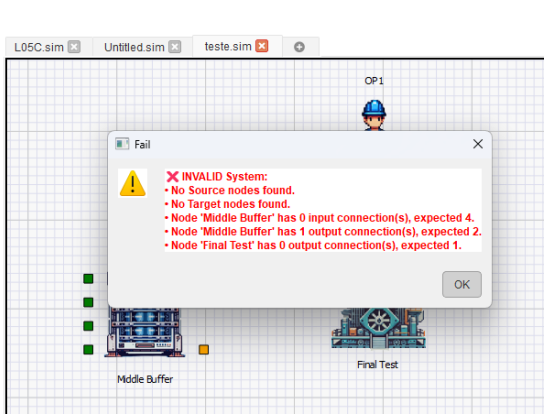


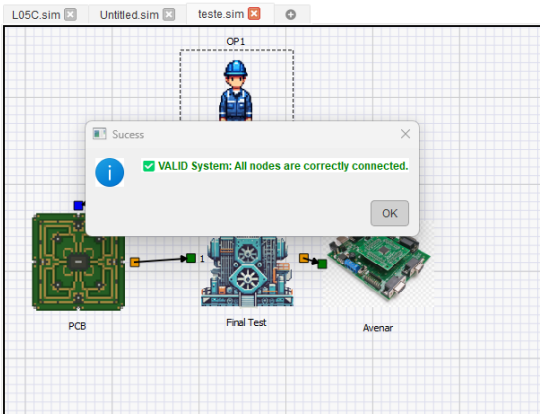
Figure 3.8: Process Time Import Dialog

3.5.3.2 Validation Feedback

When the blue validation button is pressed, the application performs a series of logical checks on the current model, described in Section 3.4.2.



(a) Validation Window - Error



(b) Validation Window - Success

Figure 3.9: Validation Window

Feedback is provided in a pop-up message. If the model is valid, a success message appears. If issues are found, the dialog lists all errors and guides the user to correct them. This validation logic is also invoked automatically when the user attempts to run a simulation

3.5.3.3 Simulation Parameters Dialog

Before running the simulation, a dedicated dialog prompts the user to define key global parameters that control the simulation runtime and evaluation:

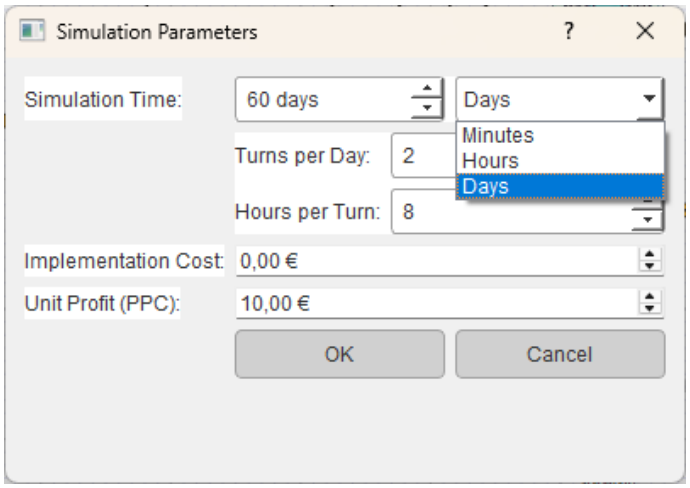


Figure 3.10: Simulation Parameters Dialog

- **Simulation Time**: The user selects whether the simulation is defined in minutes, hours, or days.

  If **Days** is selected:

  - **Turns per Day** and **Hours per Turn** are shown to determine the working time per day.

- **Implementation Cost**: A fixed value used later for cost-benefit analysis.
- **Unit Profit** (referred to as **PPC** at Bosch): The gain per successfully completed part.

These parameters are passed to the simulation engine as arguments.

3.6 Output Metrics and Logging Mechanism

To support post-simulation analysis and scenario comparison, the logic component of the framework records detailed data throughout the execution of each piece's process. All relevant events, including part movements, process durations, and operator interactions, are automatically logged and exported in structured formats. These logs form the foundation for calculating KPIs and enable engineers to thoroughly analyze the modeled system behavior. The following sections explain how this data is organized and which metrics can be extracted from the recorded results.

3.6.1 Logging Architecture and Export Formats

The simulation framework includes an internal logging system designed to register all relevant events that occur during the execution of a scenario. Two distinct logs are maintained in parallel: one for the behavior of each piece as it flows through the assembly line, and another for the operators, tracking their activity and allocation over time. These logs are automatically extracted

and saved once the simulation concludes and are presented to the user through popup spreadsheet interfaces, which can exported in the Excel format (**.xlsx**).

3.6.1.1 Part Flow Logging

Each time a piece enters, moves through, or exits a node in the assembly line, a log entry is created in the part log.

- The **ID** of the piece (unique identifier),
- The **node** name involved in the event,
- The current total **cycle time** this piece has spent in the system.,
- The cumulative simulation **process time**,
- A **comment** describing the action.

	ID ↑	NODE	CYCLE_TIME	PROCESS_TIME	COMMENT
1	1.1.1.1	PCB	0.0	0.0	In
2	1.1.1.1	PCB	0.0	0.0	Transport to Stick a Label
3	1.1.1.1	Stick a Label	0.0	0.0	Input Inserted Need:1 Have:1
4	1.1.1.1	PCB	0.0	0.0	Out
5	1.1.1.1	Stick a Label	0.0	0.0	In
6	1.1.1.1	Stick a Label	0.0	0.0	Manual Work Start
7	1.1.1.1	Stick a Label	3.3	3.3	Manual Work End
8	1.1.1.1	Stick a Label	3.3	3.3	Transport to Assemble Machine Input Conveyor
9	1.1.1.1	Stick a Label	3.3	3.3	Out
10	1.1.1.1	Assemble Machine Input Conveyor	3.3	3.3	In
11	1.1.1.1	Assemble Machine Input Conveyor	3.3	3.3	Conveyor Transport
12	1.1.1.1	Assemble Machine Input Conveyor	3.3	3.3	Transport to Assemble Components
13	1.1.1.1	Assemble Components	3.7	3.7	Input Inserted Need:1 Have:1
14	1.1.1.1	Assemble Machine Input Conveyor	3.7	3.7	Out

Figure 3.11: Spreadsheet Piece Log

This log allows reconstructing the complete trajectory of each piece across the system and analyzing bottlenecks, machine and station times, or excessive cycle times. The data is sorted by process time but can be reordered or filtered interactively in the output popup or through external spreadsheet tools.

3.6.1.2 Operator Activity Logging

In parallel, the framework logs all interactions involving human resources. For each operator, the log records:

- The operator **ID** and **name**,
- The **node** where the **interaction** occurred,

- The type of **action**,
- The simulation **process time** at which the action took place,
- The **piece ID** of the associated.

	ID	NAME	NODE	TYPE	PROCESS_TIME ↑	PIECE
1	26.0	OP1	PCB	Request	0.0	1.1.1.1
2	26.0	OP1	Stick a Label	Release	0.0	1.1.1.1
3	26.0	OP1	Stick a Label	Request	0.0	1.1.1.1
4	26.0	OP1	Stick a Label	Hold for Conveyor	3.3	1.1.1.1
5	26.0	OP1	Stick a Label	Release	3.3	1.1.1.1
6	26.0	OP1	Assemble Machine Input Conveyor	Request	3.3	1.1.1.1
7	26.0	OP1	Assemble Components	Release	3.7	1.1.1.1
8	26.0	OP1	PCB	Request	3.7	1.1.2.1
9	26.0	OP1	Stick a Label	Release	3.7	1.1.2.1
10	26.0	OP1	Stick a Label	Request	3.7	1.1.2.1
11	26.0	OP1	Stick a Label	Hold for Conveyor	7.5	1.1.2.1
12	26.0	OP1	Stick a Label	Release	7.5	1.1.2.1
13	26.0	OP1	PCB	Request	7.5	1.1.3.1
14	26.0	OP1	Stick a Label	Release	7.5	1.1.3.1

Figure 3.12: Spreadsheet Operator Log

This structure enables evaluation of operator utilization, identification of processes that rely heavily on manual labor, and tracking of how long operators remain active or idle during the simulation. These logs are especially valuable for pinpointing stations that may be overburdened or underutilized in terms of human resources.

3.6.1.3 Export and Visualization

Both logs are exported automatically when the simulation ends. The logs are presented to the user in dedicated spreadsheet pop-up windows, where they can be reviewed immediately without the need to access the raw files. The export is done in Excel format (.xlsx) to preserve formatting.

For improved interoperability, the logs can be post-processed to reorder entries and format the data by aligning events chronologically, and grouping by piece ID or operator. These logs form the primary source for extracting quantitative indicators, as detailed in the following section.

3.6.2 Key Performance Indicators

The structured logs generated by the simulation framework provide the foundation for extracting a variety of performance metrics. These indicators are essential for assessing the efficiency, workload balance, and potential constraints of a given assembly configuration. Some KPIs are calculated automatically at the end of the simulation and shown to the user, while others are derived from interpreting the exported data.

For the **Automatically Extracted Metrics**, the framework calculates several KPIs by default based on the events registered during execution, such as:

- **Process Task Time**
  The time each part spends in individual processes is recorded and aggregated, allowing for an immediate assessment of process-level efficiency.
- **Station Cycle Time**
  A "station" is defined as a set of processes sharing the same operator. The total time a piece spends across all processes within a station is calculated, providing insight into how this station affects throughput.
- **Hold Time**
  Time spent by a piece waiting in buffers or being delayed due to unavailable resources is tracked explicitly. This allows engineers to quantify non-value-adding time in the system.
- **Rejection Rates**
  The simulation framework tracks rejected parts and calculates rejection ratios globally, per station, and per process. These values are derived from the system logic and help identify areas prone to failure or instability.
- **Implementation Cost and Economic Return**
  During simulation setup, users can define two key economic parameters: the implementation cost of the proposed system configuration and the profit per completed piece. These values are saved with the simulation parameters and used to estimate the economic viability of each scenario.

While the logs provide all the necessary data, certain **Engineer-Driven Metrics** require manual interpretation or external post-processing by engineers to extract deeper insights from the recorded simulation results.

- **Bottleneck Identification**
  By analyzing **Station Cycle Times**, operator engagement, **Hold Times**, and **Process Task Times**, engineers can identify bottlenecks within specific processes or stations. These metrics identify areas where the system operates inefficiently, enabling engineers to detect underperforming points and resource imbalances.
- **Operator Utilization**
  The operator log can be used to estimate active versus idle time per resource, enabling evaluations of workload balance and potential underuse or overworkload.
- **Throughput and Flow Balance**
  By comparing entry and exit times across parts and nodes, engineers can calculate average system throughput and detect imbalances, such as areas where buffers become overloaded with pieces, indicating inefficiencies in the configuration.

3.7 Model Assumptions and Known Limitations

While the simulation framework provides a detailed and functional representation of assembly systems, it necessarily incorporates a number of assumptions and simplifications that influence its behavior and scope. These choices were primarily driven by the data provided by Bosch, practical constraints observed during line visits, and the objective of creating a decision-support tool that is both flexible and maintainable.

1. **Behavior of Operators** Human decision-making is inherently autonomous and reactive, often influenced by experience, intuition, and contextual awareness. However, the current simulation models operator behavior in a deterministic and linear fashion: each operator proceeds with tasks in sequence, moving from one assigned element to the next, unless blocked by an automatic machine or by an unavailable downstream node. When blocked, the operator selects the oldest pending task among the accessible ones. This approach is consistent with practices observed on the shop floor at Bosch, where task execution is mostly sequential and driven by proximity and availability rather than dynamic prioritization. In terms of operator profiles, no distinctions were made regarding experience or skill levels, for example: the operator "OP1" represents the average performance of multiple workers rotating through the assigned station. Consequently, the simulation does not account for variations in task speed, error rates, or decision-making among individual human resources.

2. **Simulation of Collaborative Robots** While the tool allows modeling semi-automatic processes,where both machine and operator are involved,these behaviors were implemented based on field observations and internal assumptions. Due to the lack of controlled experiments and specific performance data for CoBots in the studied lines, their behavior was only theorized, not validated.

3. **Physical World Modeling** The simulation provides a reasonably accurate temporal representation of the real assembly system, especially in terms of process duration, operator use, and flow constraints. However, it does not capture all aspects of physical layout, safety procedures, or ergonomic factors.

4. **Artificial Intelligence and Digital Twin System** From a technical standpoint, the framework does not include automatic balancing, intelligent routing, or optimization features. All decisions related to layout, operator assignment, and system configuration are left entirely to the user. This emphasizes the tool's role as a decision-support system rather than a DT. Although it draws on DT concepts like scenario exploration and virtual testing, the framework lacks real-time synchronization, bidirectional feedback, and autonomous control, which are essential characteristics of a true DT.

Chapter 4

Case Study: Bosch Assembly Lines

This chapter outlines the case study used to validate and demonstrate the simulation tool developed in this dissertation and presented in the previous chapter. It begins with an overview of the company and project context, followed by a description of the product manufactured in the case study. Then, the modeling methodology used to replicate the real assembly lines is detailed, covering both the preliminary study conducted prior to simulation implementation and the subsequent modeling within the simulation tool. The chapter concludes with the description of the simulation setup and configuration applied to all modeled scenarios.

4.1 Company and Project Context

Bosch Security Systems, S.A., located in Ovar, operates a significant manufacturing facility that produces security and fire detection systems. Within this industrial site, multiple assembly lines are organized throughout the shop floor, each dedicated to assembling complex product components while adhering to high-performance and quality standards.

This case study supports the research promoted by this dissertation on the integration of CoBots within industrial assembly environments, aiming to evaluate whether and how CoBots can enhance existing production systems by improving productivity and flexibility. Two assembly lines were modeled: L05B and L05C, an automated version of the former. Both lines produce the same range of products and were chosen to enable a comparative analysis of different production configurations. By replicating their structure and operation within the simulation environment, the objective is to explore potential improvements, in particular for L05B, by increasing automation through the integration of CoBots.

4.2 Product Overview

As mentioned earlier, the assembly lines examined in this dissertation are focused on producing a consistent range of products, specifically the Avenar fire detector. This device plays a vital role

in fire safety systems and requires a precise, consistent assembly process that includes mechanical, electronic, and testing operations. Due to the product's complexity and quality requirements, its assembly relies on a blend of manual labor and automated processes and tests to maintain high standards of reliability.

Figure 4.1: Avenar fire detector , Image credit: Bosch Security

4.3 Modeling Methodology

This chapter presents the comprehensive modeling methodology followed to replicate the Bosch L05C and L05B assembly lines. The work was conducted in two main phases: (i) conceptual modeling and process mapping using diagrams, and (ii) digital implementation in the discrete-event simulation framework developed for this dissertation. Both phases were supported by real production data and on-site observations conducted at Bosch Ovar.

4.3.1 Conceptual Modeling and Data Analysis

The modeling work began with the development of process diagrams for the L05C and L05B. These diagrams were used to visualize the full piece flow, the interaction between stations, and the specific tasks performed by each operator or automated machine. In the diagrams, directional arrows denote process flow, and accompanying labels identify operator actions. The diagrams were constructed based on:

- On-site observations of the Bosch assembly floor,
- Discussions and feedback from production engineers and operators,
- A detailed Excel file provided by Bosch, listing all processes, responsible entities (machine or human).

4.3.2 Simulation Modeling Methodology

Following the conceptual design, each assembly line was implemented in the custom-built simulation framework. The modeling adhered to a modular and scalable logic, with each process station recreated using discrete-event simulation elements.

Timings for each operation, or process, were assigned based on measured task times that were provided in the Excel dataset provided by Bosch engineers. These values were used to parameterize normal distributions through statistical fitting. For modeling purposes, a clear distinction was made between loading_time, which is associated with setups, operator movements, or input actions, and processing_time, that are related to transformative or value-added actions.

Each simulation element block was named and connected according to the flow defined in the diagrams. The goal was to preserve the production logic while ensuring that each station could later be analyzed independently or in combination with others. This modular block structure also allows easy modification of process times or operator roles for what-if analysis.

4.3.3 Operator Modeling

The simulation model of the case study assembly lines includes human operators, each assigned to specific workstation groups based on the real-life layout observed on the shop floor. Although operators do not directly introduce delays, they are treated as resources with dynamic availability constraints and an associated cost per hour of operation. Every operator-related task is embedded within the process blocks using either loading_time or processing_time parameters, as explained in 4.3.2. In essence, each operator task is modeled as a process linked to the corresponding operator. Some processes are manual, requiring operator involvement, while others are automated, that is, processed out by machines in an autonomous manner.

4.4 Assembly Line L05C

This section presents the modeling of the L05C assembly line, which was the first to be provided for this case study and consequently the first to be modeled within the simulation framework. As a more automated line, L05C serves as a reference for evaluating a more structured and mechanized production environment, higher-throughput production. The conceptual modeling outlines the full flow of the product throughout the assembly line, describing each station, the tasks performed, and the operator-machine interactions. This is followed by the simulation model, which details the implementation of each element in the digital environment, including process mappings, timing distributions, and resource allocation logic.

4.4.1 Conceptual Modeling of L05C

This section presents process flow diagrams that were constructed to map each station, define operator roles, and identify the interaction between automated and manual tasks. The diagram in Figure 4.2 provides an overview of the entire line, showing the main assembly stages and the approximate flow of the product from raw material input to the packed product.

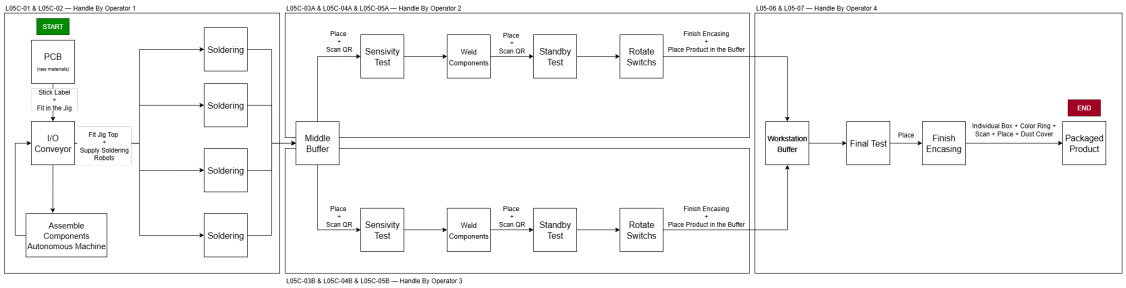


Figure 4.2: L05C Diagram

Four human operators are assigned to the line, each responsible for one or more stations based on the operational layout and real distribution observed at the plant. L05C features a combination of autonomous machines and manual operations spread across sequential workstations. The layout also includes two intermediate buffers that decouple different stages of production. The following sections offer a detailed view of each operator's working area within this configuration.

4.4.1.1 L05C-01 and L05C-02 – Initial Assembly and Soldering

Operator 1 initiates the production process by handling the raw material, which consists of the *PCB*. The first task is to apply a barcode identification label, ensuring traceability throughout the assembly workflow. The operator then places the *PCB* into a dedicated jig, specifically designed to interface with the autonomous component placement robot.

Once secured, the jig is fed into the automated station via a conveyor, where electronic components are mounted onto the *PCB* with high precision. After this step, a second jig is assembled around the product to prepare it for the robotic soldering phase. This jig provides structural support during the multi-point soldering process carried out by an automatic welding robot.

Finally, the operator removes the unit from the jig and places it in the intermediate buffer, allowing the downstream section of the line to operate independently and continuously.

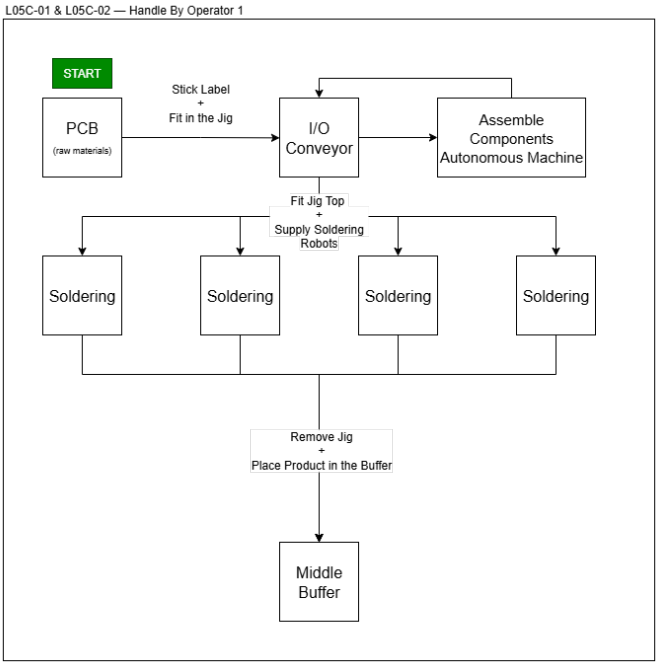


Figure 4.3: L05C-01 & L05C-02 - Handle By Operator 1

4.4.1.2 L05C-03A/B and L05C-04/05 – Sensitivity Compensation, Testing and Encasing

Operator 2 and Operator 3 handle parallel sections of the line, processing units from the middle buffer simultaneously by performing identical task sequences on separate branches to boost throughput.

Each operator begins by performing a sensitivity test on the partially assembled fire detector. This test identifies the appropriate resistor needed to compensate for potential deviations in ground voltage, a critical step to ensure the product functions within specified parameters. After determining the correct resistor, an automated dispenser drops it for the operator to weld manually onto a specific location on the board.

Then the component undergoes a standby test to ensure its electrical stability under normal operating conditions. After successful validation, the operator rotates a mechanical switch on the device and proceeds with the casing of the bottom plastic housing, preparing the unit for final assembly in the downstream stations.

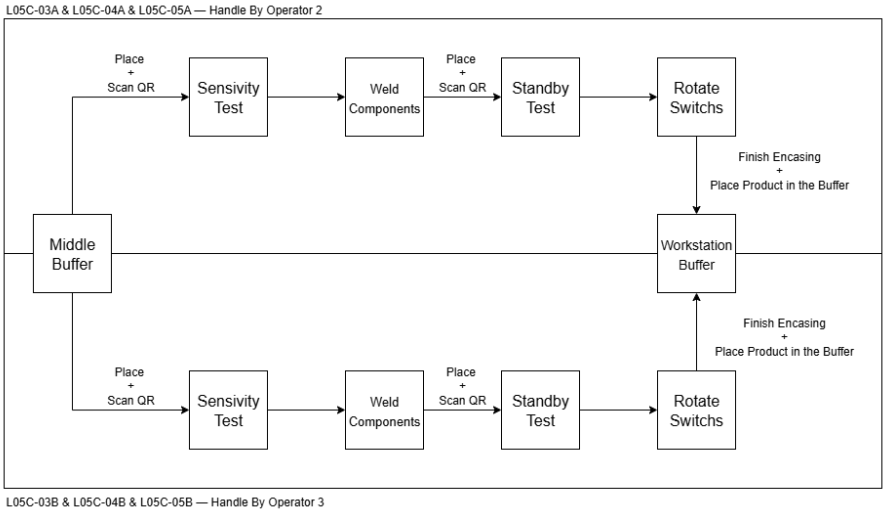


Figure 4.4: L05C-03 & L05C-04 & L05C-05 - Handle By Operators 2 and 3

4.4.1.3 L05C-06 and L05C-07 – Final Testing and Packaging

Operator 4 is responsible for the final stages of the production process, which include comprehensive functional testing, product finalization, and individual packaging. The process begins with the placement of each unit into a test machine capable of executing extensive validation routines automatically.

While a unit undergoes testing, the operator prepares the individual cardboard box by folding and assembling its base structure. Upon successful completion of the test, the operator sticks a colored identification ring on the top of the device. This ring is scanned to verify that the correct color was selected, and the piece is placed in its individual box.

Afterward, the barcode applied during the initial stages is scanned again, triggering the printing of a new external label specific to the product. While the label prints, the operator completes the assembly by attaching the final plastic components, sealing the Avenar detector's enclosure. The finished product is then placed into a cardboard box, the external label is applied, and the boxed unit is added to the final collective packaging for shipment.

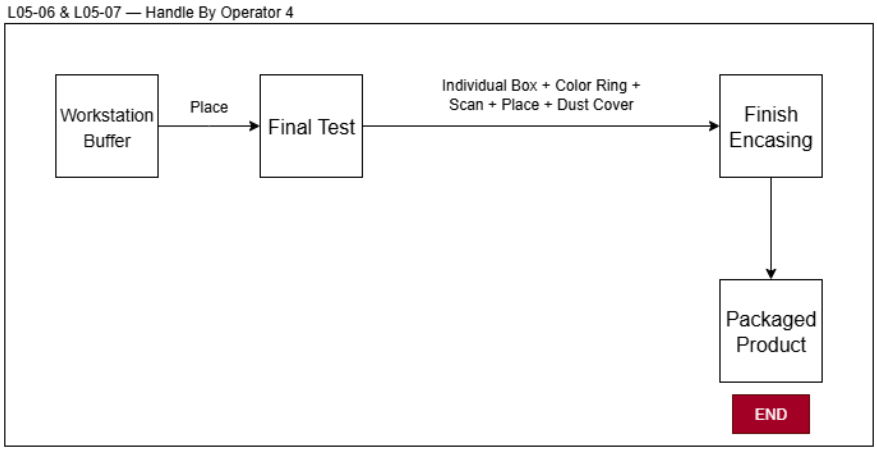


Figure 4.5: L05C-06 & L05C-07 - Handle By Operator 4

4.4.2 Simulation Modeling for L05C

The models were constructed using real-world observations and process data collected on-site, along with timing information extracted from historical datasets provided by Bosch. Each station was translated into discrete simulation elements, with human operators also included and connected to their respective processes. Processing and loading times were assigned through normal distributions, fitted to the average values derived from large data samples.

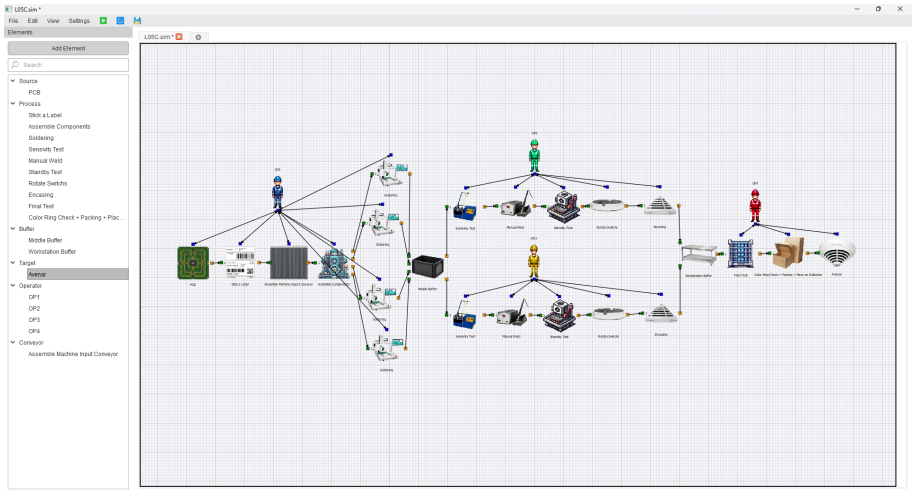


Figure 4.6: L05C Simulation Model

The modeling was conducted in three distinct segments, each corresponding to specific groups of stations and operators, which are detailed below.

4.4.2.1 L05C-01 and L05C-02 – Initial Assembly and Soldering

#	**Process**	**Average Time (s)**
L05C-01		
1	Stick the Label	3.73
2	Fit PCB in the Jig	4.04
3	Assemble Components	19.70
4	Fit Jig Top + Supply Welding Robot	4.23
L05C-02		
5	Weld Components	13.875
6	Remove Jig + Place in the Buffer	4.81

Table 4.1: Processes and Corresponding Times for Section L05C-01 and L05C-02

The simulation process begins with a *Source* element labeled *PCB*, representing the starting point and the raw PCB. The first *Process* block *Stick a Label* models the application of a label with an identification barcode, implemented with processing_time set to be process #1.

Next, the PCB is inserted into the jig and transferred to an autonomous assembly machine using a conveyor system that loops through a complex machine block. This system is modeled with three distinct inert elements: *Assemble Machine Input Conveyor*, *Assemble Components Machine*, and *Assemble Machine Output Conveyor*.

The first element, *Assemble Machine Input Conveyor*, is responsible for guiding the jig into the machine and is associated with loading_time from process #3. The central unit, labeled *Assemble Machine*, is modeled as a *Process* block and simulates the automated placement of electronic components onto the PCB using processing_time from process #3.

After processing, the piece flows through the *Assemble Machine Output Conveyor*, which has a rendezvous time of 2.5 seconds to reach the end of the conveyor. This conveyor connects directly to the *Assemble Machine Outputs Buffer*, representing the tip of the physical conveyor, which holds up to four units simultaneously. This buffer is implemented with four parallel outputs, each connected to one of the four *Automatic Soldering Machines*.

Each of the four *Automatic Soldering Machines* is modeled as an individual *Process* element. These elements simulate the preparation phase using loading_time corresponding to process #4 and the actual soldering operation using processing_time from process #5.

All soldering machines are connected to a shared *Middle Buffer*, which decouples this stage from the following ones. The time required to remove the jig and place the unit in this buffer is modeled with loading_time set to process #7.

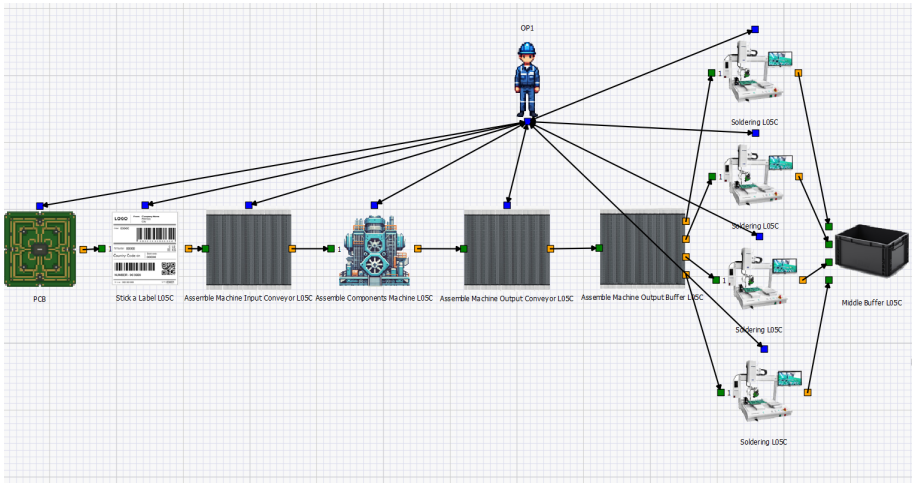


Figure 4.7: Simulation Model for L05C-01 & L05C-02

4.4.2.2 L05C-03A/B and L05C-04/05 – Sensitivity Compensation, Testing and Encasing

#	Process	Average Time (s)
L05C-03A/B		
7	Place in the Sensitivity Test + Scan Barcode	2.64
8	Sensitivity Test	10.54
9	Place in the Welder + Weld Components	4.305
10	Place in the Standby Test + Scan Barcode	1.57
11	Execute Standby Test	8.335
12	Rotate Switches	2.19
L05C-04/05		
13	Finish Encasing + Place in Buffer	3.505

Table 4.2: Processes and Corresponding Times for Section L05C-03, L05C-04, and L05C-05

After the middle buffer, the flow diverges into two symmetrical branches managed by Operator 2 and Operator 3. Each branch mirrors the same logic and consists of a linear sequence of simulation blocks.

The first task in each branch is the *Sensitivity Test*, implemented as a *Process* block with loading_time corresponding to process #7 and processing_time set to process #8.

Once the correct resistor is selected, it is manually soldered to the PCB. This step is modeled with a *Process* block using processing_time process #9.

Following the soldering, the unit undergoes a *Standby Test*. The placement and scanning are simulated with loading_time from process #10, and the test execution is defined by processing_time from process #11.

Next, the operator rotates a mechanical switch, which is modeled with a *Process* block using processing_time from process #12.

Finally, the product undergoes the *Finish Encasing* process, followed by its transfer to a buffer. This final step in the segment is captured by processing_time associated with process #13.

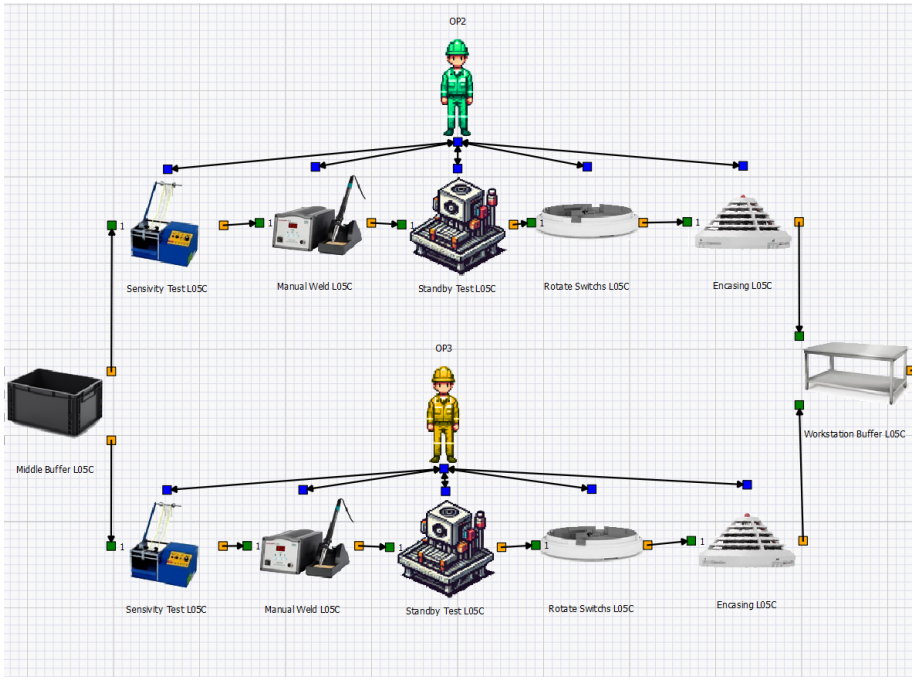


Figure 4.8: Simulation Model for L05C-03 & L05C-04/05

4.4.2.3 L05C-06 and L05C-07 – Final Testing and Packaging

#	Process	Average Time (s)
L05C-06		
14	Stick Color Ring + Place in the Final Test	2.27

#	**Process**	**Average Time (s)**
15	Execute Final Test	16
L05C-07		
16	Individual Box + Color Ring + Scan + Dust Cover + Place in Collective	11.97

Table 4.3: Processes and Corresponding Times for Section L05C-06 and L05C-07

The final part of the simulation begins when the product is retrieved from the *Workstation Buffer* by Operator 4 and placed into the *Final Test Machine*. In the real assembly line, this machine accepts up to four pieces simultaneously, processes them sequentially, and automatically transfers the validated units to a dedicated output buffer. To reproduce this behavior within the simulation, three distinct elements were modeled.

The first is the *Final Test Input Conveyor*, which simulates the operator action of placing the unit into the machine. This conveyor is modeled as a zero-delay *Buffer* with a capacity of four units and uses loading_time from process #14 to reflect the manual effort.

Once placed, the units are autonomously retrieved by the *Final Test Machine*, modeled as a *Process* block that executes the validation routines using processing_time from process #15. If the test is successful, the product is automatically transferred to the *Final Test Output Buffer*, an element that mirrors the physical collection tray used in the real-world setup.

From a successful test group of pieces, the product proceeds to a final *Process* block that represents a sequence of actions: inserting the color ring, scanning it for validation, printing the external label, completing the plastic enclosure, placing the device inside the individual box, and transferring it to the collective box. This set of tasks is modeled using processing_time from process #16.

The simulation ends with the unit reaching a *Target* element named *Avenar*, representing the completion of the assembly process.

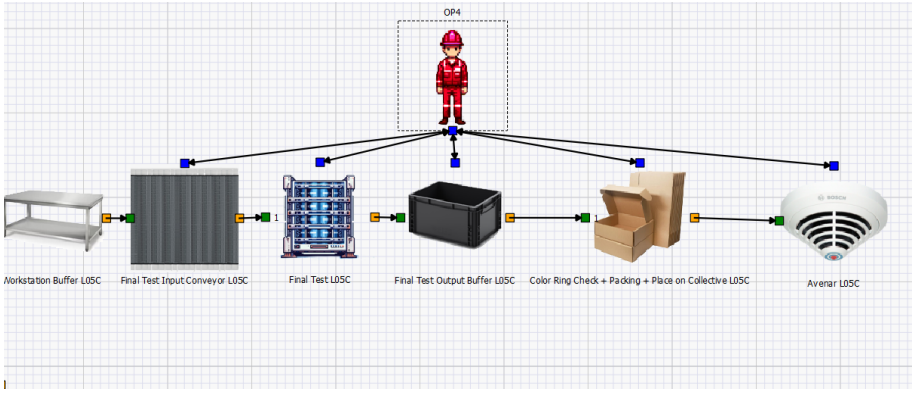


Figure 4.9: Simulation Model for L05C-06 & L05C-07

4.5 Assembly Line L05B

4.5.1 Conceptual Modeling of L05B

L05B is a simplified, more manual version of the L05C assembly line, producing the same final product and comprising the same number of workstations, but following a more linear structure with significantly reduced automation.

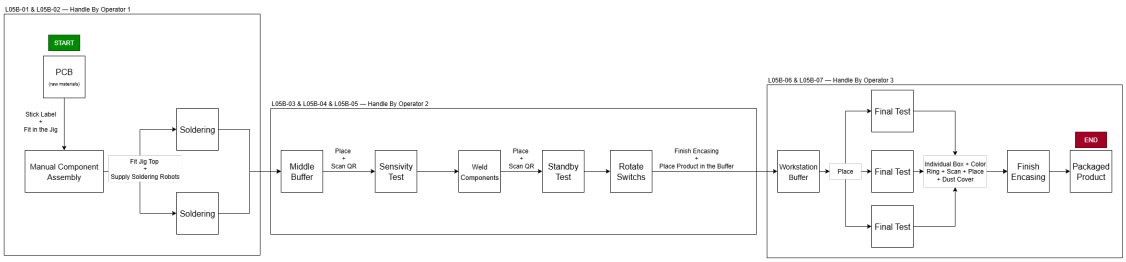


Figure 4.10: L05B Diagram

Unlike L05C's parallel branches, L05B operates as a single continuous flow, managed by three human operators, each overseeing a consecutive segment of the assembly process. While the processes mirror those in L05C conceptually, they are carried out more manually, especially in the initial stages, where component assembly is handled directly by the operator instead of automated machines.

4.5.1.1 L05B-01 and L05B-02 – Initial Assembly and Soldering

Operator 1 begins the assembly by placing a barcode label on the raw PCB, ensuring traceability throughout the line. In contrast to L05C, the component placement is conducted manually by the operator, who inserts each electronic component into the PCB. Once this is complete, a jig is fitted over the unit to provide structural support during the soldering phase.

A small local buffer is placed after this stage, allowing the operator to temporarily store partially assembled units while waiting for an available soldering machine. This ensures smoother flow, especially when both soldering stations are occupied.

Two automatic soldering machines complete the initial assembly by welding the components into the PCB. The units are then moved into the middle buffer, decoupling the first operator's workflow from the next stage.

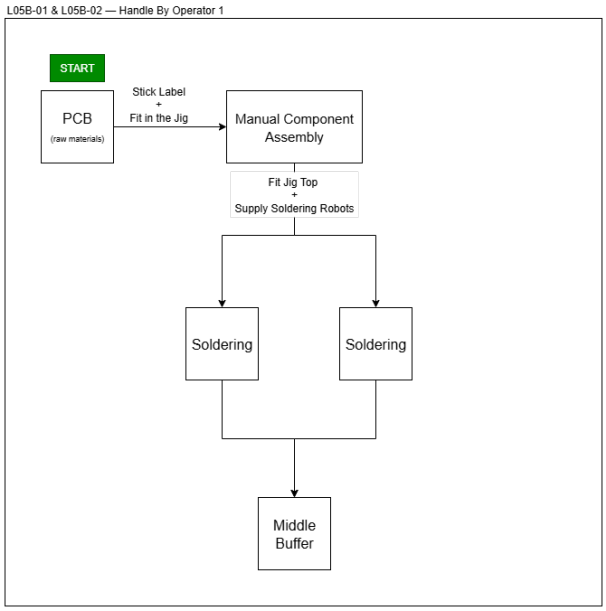


Figure 4.11: L05B-01 & L05B-02 - Handle By Operator 1

4.5.1.2 L05B-03 to L05B-05 – Sensitivity Compensation, Testing and Encasing

Operator 2 handles the middle section of the line, which in this configuration follows a single branch instead of two parallel ones as in L05C. The product retrieved from the middle buffer undergoes a sensitivity test, allowing the identification of the correct resistor to be applied for ground voltage compensation.

Once identified, the resistor is manually soldered onto the PCB. A standby test follows, verifying the product's functionality under normal conditions. The operator then rotates a mechanical switch to its final position and completes the lower casing of the unit, which is subsequently placed into the workstation buffer.

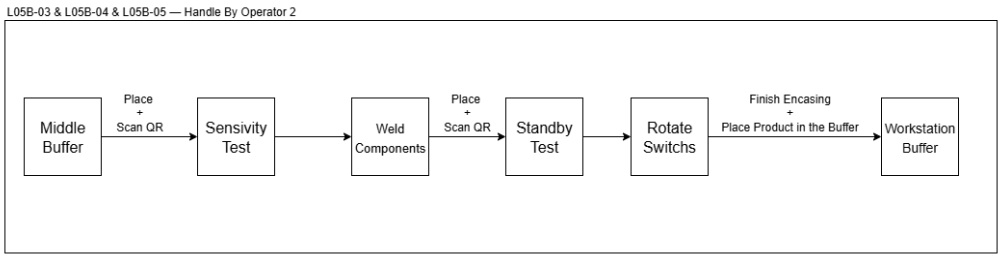


Figure 4.12: L05B-03 & L05B-04 & L05B-05 - Handle By Operator 2

4.5.1.3 L05B-06 and L05B-07 – Final Testing and Packaging

Operator 3 is responsible for the final operations. The partially assembled unit is placed into one of three available automated machines for final testing. These machines are not as well automated as the single unit used in L05C, which processes four pieces simultaneously. As a result, each individual machine in L05B operates much more slowly in terms of task time compared to the combined machine in L05C.

Next, the bar code applied earlier in the process is re-scanned, prompting the system to print a new external sticker label to the cardboard box of the product. While the label is being printed, the operator finishes the assembly by attaching the last plastic parts, completing the enclosure of the *Avenar* detector. Once assembled, the product is placed into a cardboard box, the external label is placed, and the packed unit is then moved into the final collective packaging for dispatch.

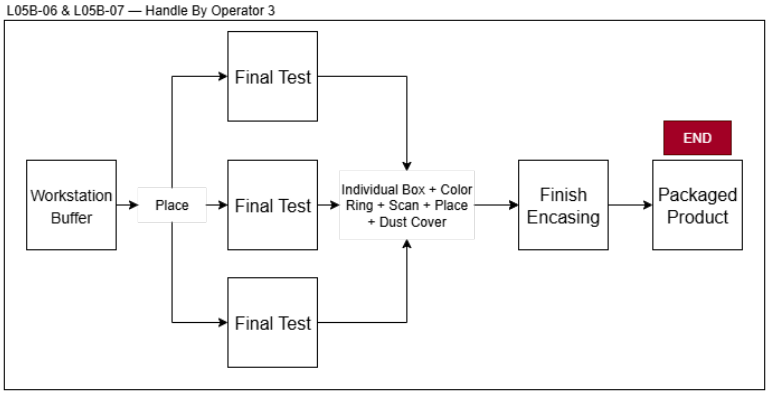


Figure 4.13: L05B-06 & L05B-07 - Handle By Operator 3

4.5.2 Simulation Modeling for L05B

The simulation model for L05B was developed following the same methodology adopted for L05C, with adaptations to reflect its simplified and more manual nature. The same number of workstations was implemented, although the structural layout is linear rather than branched, and

automation is significantly reduced. The entire line is operated by three human operators, each responsible for one continuous segment of the process.

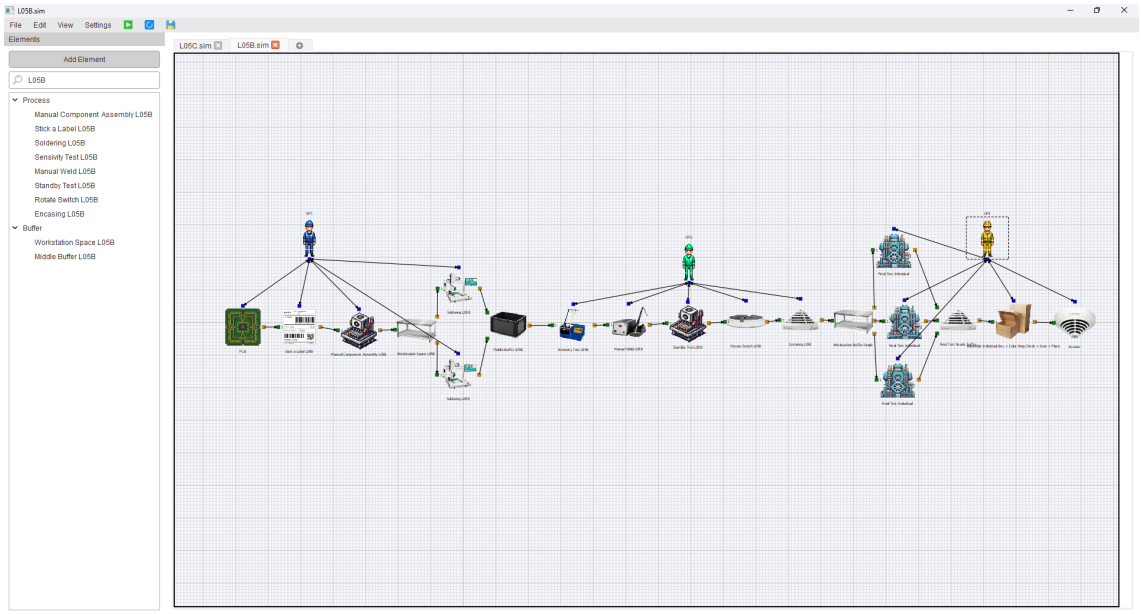


Figure 4.14: L05B Simulation Model

4.5.2.1 L05B-01 and L05B-02 – Initial Assembly and Soldering

#	Process	Average Time (s)
L05B-01		
1	Stick the Label	3.74
2	Fit PCB in the Jig	4.36
3	Manual Component Assembly	24.70
4	Fit Jig Top + Supply Welding Robot	4.33
L05B-02		
5	Weld Components	13.88
6	Remove Jig + Place in the Buffer	5.54

Table 4.4: Processes and Corresponding Times for Section L05B-01 and L05B-02

The simulation begins with a *Source* element named *PCB*, representing the arrival of the raw PCB. The first task in the simulation is the application of a barcode label for product traceability. This is modeled with a *Process* block using processing_time from process #1.

Following this, a second block simulates the manual placement of electronic components onto the PCB by the operator. This action is captured in a single *Process* element, the *Manual Component Assembly*, which uses a loading_time associated with process #2 and a processing_time corresponding to process #3.

After the manual assembly, the unit reaches a node that represents a physical space used by the operator to temporarily store units. This is implemented as an inert buffer named *Workstation Space* and introduces no delay, serving only to reflect the real layout where operators often offload parts when the soldering machines are occupied.

From this point, the flow continues into one of two automatic soldering machines. These are modeled as individual *Process* blocks. The preparation step, which involves fitting the jig top and supplying the soldering robot, is simulated using loading_time from process #4. The soldering action itself is captured by the processing_time, corresponding to process #5. Once completed, each machine directs the unit to the middle buffer.

The middle buffer decouples the initial tasks from the subsequent section. It is modeled with a loading_time aligned with process #6, simulating the time required to remove the jig and place the unit in the buffer.

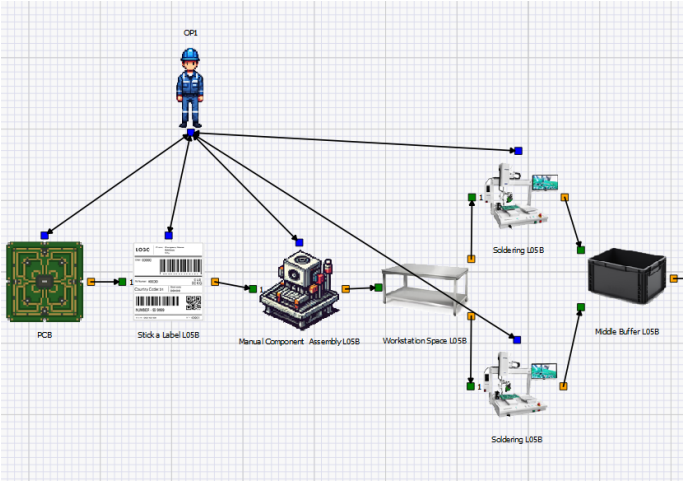


Figure 4.15: Simulation Model for L05B-01 & L05B-02

4.5.2.2 L05B-03 to L05B-05 – Sensitivity Compensation, Testing and Encasing

#	Process	Average Time (s)
L05B-03A/B		
7	Place in the Sensitivity Test + Scan Barcode	2.45
8	Sensitivity Test	10.54

#	Process	Average Time (s)
9	Place in the Welder + Weld Components	4.56
10	Place in the Standby Test + Scan Barcode	1.72
11	Execute Standby Test	8.42
12	Rotate Switches	2.14
L05B-04/05		
13	Finish Encasing + Place in Buffer	3.505

Table 4.5: Processes and Corresponding Times for Section L05B-03, L05B-04, and L05B-05

The second part of the simulation follows a linear structure and is handled entirely by Operator 2. Upon exiting the middle buffer, each unit undergoes a sensitivity test. This is modeled as a *Process* element that includes a loading_time to represent the action described in process #7, followed by a processing_time that refers to the execution of the test itself, which is process #8.

Once the required resistor is identified, the operator manually welds it onto the PCB. This task is modeled using a processing_time linked to process #9. Next, the unit enters a standby test block. The placement and scanning for this test are represented with loading_time from process #10, and the execution of the test uses processing_time from process #11.

The operator then rotates the mechanical switches on the unit, modeled as a standalone *Process* block with processing_time corresponding to process #12. The final task in this segment is the completion of the casing process that is modeled with a processing_time equal to process #13, followed by the transfer of the product into the workstation table buffer.

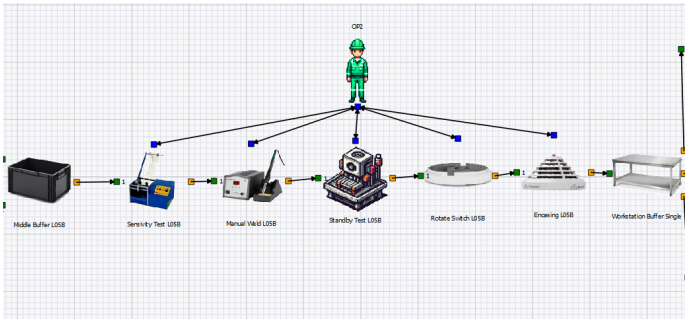


Figure 4.16: Simulation Model for L05B-03 & L05B-04 & L05B-5

4.5.2.3 L05B-06 and L05B-07 – Final Testing and Packaging

#	**Process**	**Average Time (s)**
L05B-06		
14	Place product in Final Test	2.56
15	Execute Final Test	24
L05B-07		
16	Assemble Individual Box + Color Ring Check + Scan + Place	7.10
17	Close Individual Box + Stick Label + Place	5.96

Table 4.6: Processes and Corresponding Times for Section L05B-06 and L05B-07

The third and final section of the simulation is managed by Operator 3. It begins with the product entering one of three available automatic final test machines. Each machine is modeled as an individual *Process* block. The operator places the product and initiates the test, which is simulated using a loading_time derived from process #14. The test execution is then captured by a processing_time corresponding to process #15.

After testing, the outputs of the three machines converge into a linear path using an inert merging buffer named *Final Test Ready Buffer*. This buffer introduces no delay but is used to connect the parallel machines to the final station.

The last production stage includes assembling the individual box, placing the colored ring, scanning the barcode, and completing the plastic closure of the product. All these actions are grouped and modeled as a single *Process* block with a processing_time based on process #16.

Finally, the unit is sent to the *Target* element labeled *Avenar*. The placement of the product in the collective shipping box and the application of the external label are modeled using loading_time from process #17.

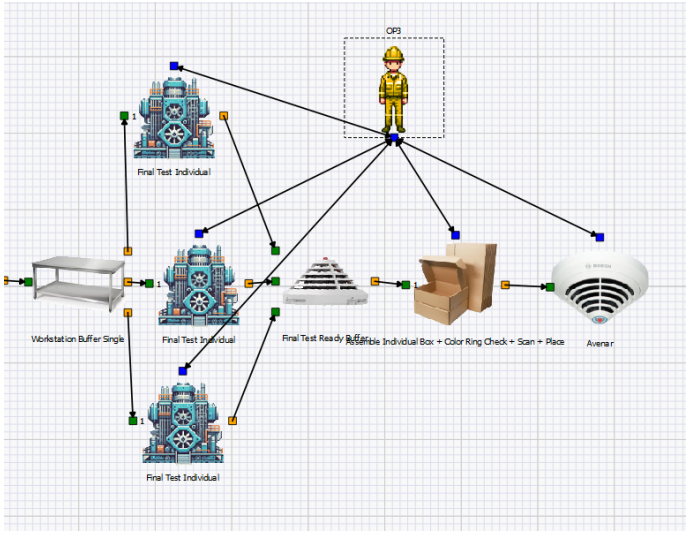


Figure 4.17: Simulation Model for L05B-06 & L05B-07

4.6 Simulation Setup and Configuration

To evaluate the modeled scenarios under realistic conditions, each simulation, both for the baseline case studies and the improved configurations, was executed for a total duration of 25 simulated hours. This duration was selected to align with the 25-hour monitoring campaign conducted by Bosch to assess product yield and determine key performance metrics, as detailed in the historical production data. Mirroring the timeframe used in the real environment enables direct comparison of the simulation results with Bosch's operational measurements.

From the simulated runs, the framework extracts several critical performance indicators, including:

- Average, minimum, and maximum **task times** for each individual process;
- Aggregated **station performance**, particularly for zones operated by human workers;
- Identification of **bottlenecks** along the production line;
- Real-time calculation of the **hourly output** throughout the simulation;
- An overview of the **financial performance**, based on resource usage, operator costs, and production rate.

These indicators serve as the basis for evaluating process efficiency and for comparing the effectiveness of new configurations, such as the introduction of CoBots or shifts in operator allocation.

Chapter 5

Results and Evaluation

This chapter presents the results obtained through the simulation framework applied to the Bosch assembly lines L05Cand L05B. It aims to evaluate the performance of different operational scenarios and explore the benefits and limitations of introducing CoBots into a manual production line.

The first part of the chapter describes the KPIs used to assess each configuration, followed by an analysis of the baseline simulation results and their comparison with real operational data from Bosch. This validation step confirms the reliability of the simulation model before introducing any modifications.

We subsequently introduced a solution that integrates a cobot and redefines operator task assignments for improved task distribution management. This solution has been evaluated for its impact on productivity, task distribution, and operational efficiency. We compared these scenarios with the original L05B line and the more automated L05C line.

Finally, a cross-scenario comparison is conducted to analyze performance differences, estimate cost-benefit trade-offs, and identify strategic implications for future improvements. The chapter concludes with a summary of the key insights derived from this evaluation.

5.1 Evaluation Metrics and Validation Approach

To assess the performance of each simulated scenario, a set of quantitative indicators was defined. These indicators are automatically extracted by the simulation framework upon completion of each run, and allow for both operational and financial evaluation of each assembly line configuration.

5.1.1 Key Operational Metrics

Three main classes of operational metrics were considered:

- **Throughput (Pieces per Hour)** - the total number of finished products successfully reaching the output per simulated hour. This is a direct measure of the productivity of the system.

  Processing Time - the total time a piece spends in each process (or node), from entry to exit. This includes value-added time such as loading_time and processing_time, along with any delays caused by resource contention. It reflects the total occupation of that node.

- **Value-Added Time** - the actual execution time of the process itself, which corresponds to the manual or automatic operation and the defined loading/unloading steps for each node, as specified in the previous chapter. It reflects the effective task duration without idle or waiting components.

- **Overall Cycle Time ($\mathbf{CT_{med}}$)** - represents the average time interval between the completion of two consecutive pieces at the end of the line. This value reflects the rhythm at which the line delivers output and is directly tied to throughput (a lower cycle time implies a higher production rate).

For both Processing Time and Value-Added Time, the framework computes their percentage relative to the total processing duration of the piece. This helps identify stations that contribute most to operational time and highlights potential bottlenecks or optimization opportunities. Distinguishing between total and value-added time is essential for assessing process efficiency, as excessive time spent on non-value-adding activities often signals poor flow synchronization, unbalanced workloads, or design limitations.

The Overall Cycle Time provides a high-level indicator of the production rhythm, representing the average time interval between the completion of two consecutive pieces. It is calculated by dividing the total operation time (in seconds) by the number of produced pieces within that period. This metric reflects the effective cadence of the line and is directly linked to throughput. While it does not reveal how time is distributed across stations, it serves as a valuable reference for evaluating the impact of process changes, interruptions, or performance improvements on the system's output rate.

5.1.2 Key Financial Indicators

The simulation also provides financial estimations based on predefined parameters, such as operator hourly wage, unit production value, and cobot investment cost. The following indicators are used:

- **Production Profits** - calculated based on the number of pieces produced and their unit sale price.

- **Operator Cost** - the total cost is calculated based on the number of operators, their hourly wage, and the total hours worked.

- **CoBots Investment Cost** - considers all expenses related to the acquisition, integration, programming, safety infrastructure, layout adaptation, and training required to implement the CoBots systems within the assembly line.

5.1.3 Validation Strategy

Before introducing improvements or modifications to the system, it is essential to ensure the validity of the simulation results. This is achieved by comparing the simulated outputs of the L05Cline to real operational data collected from Bosch. The comparison focuses on throughput and average process times, ensuring that the model accurately represents the behavior of the real system.

5.2 Case Study Results

This section presents the simulation results for the real assembly line L05C and, later, for the proposed manual line L05B. The analysis includes operational and financial metrics extracted from the simulation, as well as insights into individual station performance. The results will serve as a baseline for evaluating improvements in subsequent scenarios.

5.2.1 L05C: Current Real Assembly Line

5.2.1.1 Processing Time per Node and Station Utilization

Table 5.1 presents a detailed analysis of the time distribution across all stations in the L05C assembly line. For each node, the table reports the average time a piece spends at that location, encompassing all phases of its presence..

In addition to the absolute processing time, the table also includes the effective throughput at each station and its respective utilization percentage. The utilization value reflects the proportion of total simulation time during which each station is actively occupied by a piece. This metric is particularly relevant for identifying bottlenecks and underused resources, and provides insight into the overall efficiency and synchronization of the line's operational flow.

Table 5.1: Processing Time and Utilization per Station - L05C

Station	**Avg Time (s)**	**Throughput (pcs/h)**	**Utilization (%)**
PCB	22.73	158.40	7.75
Stick a Label L05C	22.72	158.36	7.75
Assemble Components Machine L05C	22.67	158.12	7.73
Assemble Machine Input Conveyor L05C	18.56	158.16	6.33
Assemble Machine Output Buffer L05C	109.77	157.88	37.43
Assemble Machine Output Conveyor L05C	21.24	158.08	7.24
Soldering L05C	18.64	157.84	6.36
Middle Buffer L05C	2.57	157.84	0.87
Sensitivity Test L05C	10.52	157.84	3.59
Manual Weld L05C	5.80	157.80	1.98
Standby Test L05C	8.34	157.80	2.84
Rotate Switches L05C	2.35	157.80	0.80
Encasing L05C	3.44	157.80	1.17
Workstation Buffer L05C	6.44	157.80	2.19
Final Test L05C	16.11	157.72	5.49
Final Test Input Conveyor L05C	1.37	157.76	0.47
Final Test Output Buffer L05C	0.02	157.72	0.01

The process with the highest processing time and utilization is the *Assemble Machine Output Buffer*, consuming approximately **43.61 seconds per piece** and accounting for **36.73%** of the total active time in the system. This data suggests the presence of a structural bottleneck, caused by the accumulation of pieces in this buffer awaiting transfer to subsequent stages. The cause of this bottleneck may be attributed to the limited availability of downstream stations or to synchronization inefficiencies within the flow logic of the automated system.

The remaining stations exhibit relatively even processing times ranging between 18 and 23 seconds, with utilization values clustering between 6-8%. This uniformity suggests a well-balanced flow across automated components of the line, with little idle time and consistent pacing. The lowest utilization values correspond to the final quality control stations , *Final Test*, *Sensivity Test*, and *Standby Test* , where process times are significantly shorter, and delays are minimal due to automation and low variability.

5.2.1.2 Value-Added Time Analysis

To complement the station-level analysis, Table 5.2 breaks down the average time each piece spends at every station of the L05C line. This includes all value-added operations, whether performed manually, automatically, or during loading/unloading phases.

Table 5.2: Value-Added Time Breakdown per Station - L05C

Station	**Manual (s)**	**Automatic (s)**	**Loading (s)**	**% of Total Time**
Stick a Label L05C	0.00	19.70	0.00	17.3%
Assemble Components Machine L05C	0.00	13.84	4.76	16.3%
Soldering L05C	0.00	10.44	0.00	9.2%
Sensitivity Test L05C	0.00	8.25	0.00	7.2%
Standby Test L05C	0.00	8.25	0.00	7.1%
Rotate Switch L05C	3.45	0.00	0.00	3.0%
Encasing L05C	0.00	3.95	0.00	3.0%
Workstation Buffer L05C	0.00	0.00	2.74	2.2%
Final Test L05C	0.00	15.95	0.00	14.0%
Final Assembly L05C	3.85	0.00	0.00	11.1%

A significant observation is the disproportionate weight of the *Assemble Components Machine L05C* and the *Stick a Label L05C* tasks, which account for a substantial portion of value-added time. Specifically, *Stick a Label L05C* contributes 17.3% of the total time, which is notably high for an operation that is entirely automated.

The *Assemble Components Machine L05C* also stands out, contributing 16.3% of the value-added time. This suggests that while automation handles the core tasks efficiently, manual operations, such as those associated with *Rotate Switch L05C* (3.0%), still contribute a significant amount of time that could be optimized.

The stations dedicated to final testing, like *Final Test L05C*, consume a large proportion of the total cycle time (14.0%).

Overall, the value-added time analysis reveals that while automation contributes to reducing cycle time at most stages, certain manual tasks continue to have a high impact on overall efficiency. This calls for strategic automation, especially in areas that could significantly benefit from CoBots, particularly in the final parts of the process, which involve repetitive and ergonomically challenging tasks.

5.2.1.3 Simulation Output Summary

The simulation of the L05C line was executed over a virtual period of 25 hours. A total of 3970 pieces were successfully completed, corresponding to an average throughput of **158.80 pieces/hour**. The average value-added cycle time per piece was approximately **325.61 seconds**. The simulation results are summarized in Table 5.3.

Table 5.3: Summary of Key Performance Indicators - L05C

Metric	**Value**
Simulation Duration	25 hours
Completed Pieces	3962
Throughput	158.48 pcs/h
Overall Cycle Time (CT_{med})	22.72 s
Average Value-Added Time per Piece	113.79 s
Average Processing Time per Piece	293.29 s

5.2.1.4 Comparison with Real Data

The Table 5.4 compares the simulation results of the L05C line with real production data from the corresponding physical assembly line operated at Bosch. The comparison is based on 25 hourly measurements on three different shifts, focusing on throughput and median cycle time as key performance indicators.

Table 5.4: Simulation vs Real Production Data - L05C

Metric	**Shift 1**	**Shift 2**	**Shift 3**	**Average**	**Simulation**
Throughput (pcs/h)	146.44	152.51	152.11	150.30	158.48
CT_{med} (seconds)	24.61	23.62	23.68	23.97	22.72

The comparison reveals a close alignment between simulated and real-world values, with a deviation of approximately 5.4%. This level of variation is acceptable and expected, particularly given that the simulation framework does not model human behavior.

Despite its deterministic nature, the simulation tool captures the essential dynamics of the real L05Cline and can therefore be considered a valid approximation for scenario testing and optimization analysis.

5.2.2 L05B: Manual Line Configuration

5.2.2.1 Processing Time per Node and Station Utilization

The processing time per station is shown in Table 5.5, along with each station's utilization percentage. Although individual station times are longer than in L05C, utilization remains low due to the slower throughput.

Table 5.5: Processing Time and Utilization per Station - L05B

Station	**Avg Time (s)**	**Throughput (pcs/h)**	**Utilization (%)**
PCB	34.79	103.48	1.16
Stick a Label L05B	13.12	103.44	0.44
Manual Component Assembly L05B	30.70	103.40	1.02
Workstation Space L05B	68.04	103.32	2.27
Soldering L05B	31.56	103.28	1.05
Middle Buffer L05B	1970.46	99.28	65.70
Sensitivity Test L05B	27.15	99.24	0.91
Manual Weld L05B	31.71	99.20	1.06
Standby Test L05B	29.21	99.16	0.97
Rotate Switch L05B	31.46	99.12	1.05
Encasing L05B	31.90	99.08	1.06
Workstation Buffer L05B	678.34	98.28	22.62
Final Test Individual L05B	20.95	98.28	0.70

The results presented in Table 5.5 provide valuable insights into time distribution across the L05Bassembly line. As expected from a predominantly manual setup, the average processing time per station tends to be higher than in the more automated L05C line.

One of the most notable observations is the performance of the *Middle Buffer*, which holds each piece for an average of 1970.46 seconds, accounting for 65.70% of total utilization, which is a clear indicator of a bottleneck. This prolonged holding time suggests significant accumulation and potential congestion caused by limited availability in downstream stations or general flow imbalances. Similarly, the *Workstation Buffer* holds pieces for an average of 678.34 seconds, contributing to 22.62% of total utilization. Overall, buffers in this line account for 88.32% of total utilization. This highlights that a substantial portion of production time is spent waiting rather than undergoing active processing.

These findings expose a critical inefficiency in the current flow design. Although several stations maintain processing times around 30-35 seconds, their utilization remains under 1%, indicating that they remain idle for most of the time.

Overall, the L05B configuration demonstrates a highly unbalanced system. Despite individual stations being capable of performing tasks efficiently, the line suffers from poor synchronization and flow control. The dominance of idle buffer times over active processing time underlines the need for improved scheduling, possible intermediate automation, or the integration of CoBots, precisely the focus of the dissertation project.

5.2.2.2 Value-Added Time Analysis

In this version of the line, most operations are performed manually. Table 5.6 shows the breakdown of average value-added time per station, including manual and automatic work, as well as loading activities.

Table 5.6: Value-Added Time Breakdown per Station - L05B

Station	**Manual (s)**	**Automatic (s)**	**Loading (s)**	**% of Total Time**
Stick a Label L05B	0.00	3.76	3.95	6.60%
Manual Component Assembly L05B	21.59	0.00	0.00	18.30%
Workstation Space L05B	0.00	0.00	3.15	2.70%
Soldering L05B	0.00	13.80	5.45	16.30%
Middle Buffer L05B	0.00	0.00	2.44	2.00%
Sensivity Test L05B	0.00	10.46	0.00	8.50%
Manual Weld L05B	3.76	0.00	1.55	4.30%
Standby Test L05B	0.00	8.25	0.00	6.70%
Rotate Switch L05B	2.14	0.00	0.00	1.70%
Encasing L05B	0.00	3.95	0.00	3.20%
Workstation Buffer L05B	0.00	0.00	2.74	2.20%
Final Test Individual L05B	0.00	20.95	0.00	16.90%
Assemble Individual Box + ... + Place L05B	7.05	0.00	5.85	10.40%

A closer look at the distribution of processing time across L05B reveals key contributors to the overall workload. Manual Component Assembly stands out, accounting for 18.3% of total processing time and relying entirely on manual labor, making it one of the most labor-intensive operations in the line.

In contrast, *Final Test Individual*, a fully automated process, contributes 16.9% to the total value-added time. *Soldering* follows closely, representing 16.3% of the processing time, with its workload divided between automated execution and manual loading.

The *Assemble Individual Box + Color Ring Check + Scan + Place* station, which aggregates several actions, has a relatively high impact at 10.4%, and is also a fully manual and loading-heavy task.

Overall, the value-added time analysis confirms the presence of both high-impact manual tasks and potential automation points. The distribution also supports the idea that selectively integrating CoBots to target tasks with high manual input or repetitive handling could significantly enhance efficiency without disrupting the overall workflow.

5.2.2.3 Simulation Output Summary

The L05B line, modeled as a predominantly manual configuration, was simulated over a total duration of 25 hours. Throughout this period, 2589 pieces were completed, resulting in a throughput of **103.56 pieces/hour**. This value is significantly lower than that of the automated L05Cline, reflecting the slower pace of production typically associated with manual operations. Table 5.7 summarizes the main operational indicators for this initial configuration.

Table 5.7: Summary of Operational Performance Indicators - L05B

Metric	**Value**
Simulation Duration	25 hours
Completed Pieces	2589
Throughput	103.56 pcs/h
Overall Cycle Time (CT_{med})	34.76 s
Average Value-Added Time per Piece	120.84 s
Average Processing Time per Piece	2999.39 s

5.2.2.4 Comparison with Real Data

This section compares the simulation results of the L05B line with real production data obtained from a similar manual assembly line operating under Bosch conditions. The comparison focuses on throughput and median cycle time, based on three 25-hour shifts of recorded production.

Table 5.8: Simulation vs Real Production Data - L05B

Metric	**Shift 1**	**Shift 2**	**Shift 3**	**Average**	**Simulation**
Throughput (pcs/h)	92,94	96,14	95,76	94,95	103.56
CT_{med} (seconds)	35.32	34.77	35.37	35.15	34.76

The comparison reveals a strong alignment between simulation and real data with a 9% deviation, which is acceptable given the manual nature of the L05B line, where performance is highly influenced by human factors such as operator fatigue, pace variability, short pauses, or inconsistencies in task execution.

As the simulation model assumes ideal and uninterrupted flow, without modeling individual human behavior, slight overestimation in output is expected. Nevertheless, the results confirm that the simulation provides a valid and representative baseline for testing improvement scenarios in predominantly manual environments.

5.3 Solution for L05B

This section presents the proposed improvement solution to the L05B assembly line, referred to as **L05B Version 2 (L05B/V2)**. The solution aims to alleviate the bottleneck identified in the middle section of the process by introducing partial automation through a cobot.

5.3.1 Description of the Cobot Integration

In the baseline configuration of the L05B line, three sequential tasks , *Sensitivity Test*, *Welding*, and *Standby Test* , are carried out independently, with long idle periods due to buffer accumulation and lack of synchronization. In L05B/V2, these operations are consolidated and automated using a CoBots capable of executing the sequence in a continuous and coordinated manner.

The cobot receives a piece from an input buffer (fed by a human operator), performs the sensitivity test, places the piece into an autonomous welding machine, and in parallel retrieves another piece to begin the next sensitivity test. Once the welding is complete, the cobot transfers the piece to the standby test station. After finalizing this last operation, the part is placed into a slide buffer, marking the end of its processing.

The cobot executes this loop autonomously and without physical interaction with human operators. To ensure safety, the cobot operates within a closed area protected by acrylic or glass walls, physically separated from the operator. The human operator is responsible only for rotating the switch, performing the encasing operation, and feeding the input buffer.

5.3.2 Modeling Approach for Collaborative Robots

In the simulation framework, the cobot was modeled using a structure composed of an *Cobot Input Buffer*, a Cobot which simulates the *Sensitivity* Test, the *Welding* and the Standby Test, and an *Cobot Output Buffer*. The process includes a probabilistic loading and processing time, based on real-life measurements carried out at Bosch using a stopwatch:

- **Loading Time:** normally distributed with a mean of 2.43 seconds and a standard deviation of 0.41 seconds.
- **Processing Time:** normally distributed with a mean of 17.42 seconds and a standard deviation of 0.80 seconds.

The loading_time is modeled in the *Cobot Input Buffer*, and the processing_time in the Cobot.

This setup encapsulates the time required for the cobot to manage one full cycle, including part manipulation, coordination with the autonomous welding machine, and transfer between stations. The output buffer introduces no additional delay, allowing continuous flow when downstream stations are available.

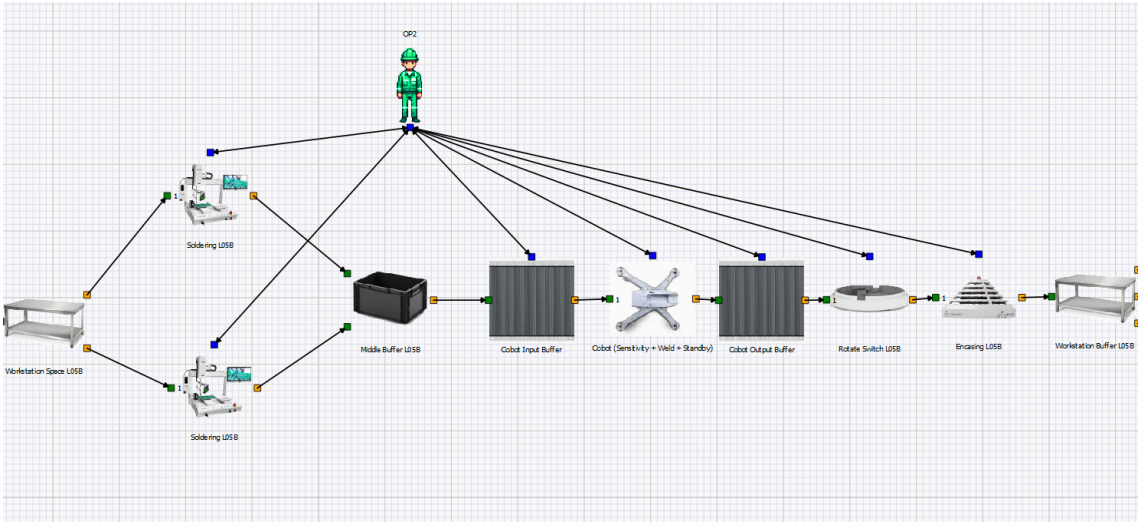


Figure 5.1: Solution L05B/V2

As seen in Figure 5.1, to address imbalances observed in the manual configuration of the L05B line, the operator OP2 was reassigned to take over the soldering task previously handled independently. In this revised layout, OP2 retrieves the piece directly from the Workstation Buffer (previously the output of OP1) and performs the soldering operation as part of their expanded responsibilities. This decision was made with the goal of redistributing workload more evenly across operators, as OP2 was initially underutilized compared to OP1. By concentrating both tasks , retrieval and soldering , under the same operator, the workflow becomes more fluid, reduces idle time between stations, and contributes to a more balanced and synchronized production line.

5.3.3 Implementation Cost Estimation

To support a later cost-benefit evaluation and return-on-investment (ROI) calculation, a detailed cost estimation was developed for the implementation of the L05B/V2 configuration. This estimation considers all required components, including hardware acquisition, layout modification, programming, training, and safety integration.

Table 5.9 provides a breakdown of each cost component, including ranges based on recent industrial data and manufacturer specifications.

Table 5.9: Estimated Implementation Cost for L05B/V2

Cost Component	**Estimated Cost (€)**	**Description**
Cobot arm (including controller and base software)	30,000 - 50,000	Medium-payload CoBots (about 10 kg), suitable for repetitive multi-stage handling and test placement tasks.
Custom gripper/end effector	5,000 - 10,000	Specialized tooling to handle delicate parts during the test and welding sequence.
Safety enclosure and physical isolation (acrylic/glass walls)	5,000 - 15,000	Safety structure compliant with collaborative workspace regulations, to isolate human-robot interaction zones.
Integration, programming, and validation	15,000 - 30,000	Configuration of task sequence (Sensitivity Test → Welding → Standby Test), synchronization logic, and automation testing.
Layout adaptation and auxiliary equipment	5,000 - 10,000	Physical reconfiguration of stations, buffers, and workspaces. Includes electrical and structural adjustments.
Operator training and knowledge transfer	3,000 - 7,000	Safety training, workflow interaction protocols, and basic cobot maintenance.
Total Estimated Investment	**63,000 - 122,000**	

The physical and logical integration of the cobot is estimated to require **3 working days** of installation time, assuming an effective daily workload of 7.70 hours. An additional **2 working days** are reserved for programming, testing, and fine-tuning of the system. This results in a total of **5 working days** during which the line is assumed to be non-operational.

Given an average production rate of 102.4 pieces per hour and an estimated profit margin of 8.6 EUR per piece, the temporary suspension of production leads to an opportunity cost of:

$$\text{Losses} = 5\,\text{days} \times 7.7\,\text{hours/day} \times 94.95\,\text{pcs/hour} \times 8.6\,\text{EUR} = \mathbf{31\,438\ \ EUR}$$

This lost profit must be considered in addition to the implementation cost. Assuming an minimum expected installation investment of approximately 63,000 EUR, the total effective cost of adopting the L05B/V2 solution, including the production halt, rises to a minimum of:

94 438EUR

This cost sheet will later serve as the basis for calculating the ROI, by comparing the cost of implementation against productivity gains in L05B/V2.

5.3.4 Simulation Results and Analysis

Table 5.10 summarizes the operational results obtained from simulating the improved L05B/V2 configuration. Over a 25-hour simulation period, the system produced 3085 pieces, which corresponds to a throughput of **123.40 pieces/hour**, an improvement of approximately 19% compared to the baseline L05B configuration (103.56 pieces/hour).

Table 5.10: L05B/V2 Results Comparison

L05B	**Real**	**Simulation**	**Solution**
Throughput (pcs/h)	94,95	103,56	123,40
CT_{med} (seconds)	35.15	34.76	29,17

The average value-added time per piece decreased from 120.84 seconds to 103.67 seconds, indicating that the line spends less time on useful tasks per product due to increased efficiency. Furthermore, the **Overall Cycle Time (CT_{med})** was reduced from 34.76 seconds to 29.17 seconds, a gain of nearly 5.6 seconds per piece. This represents an **effective 16% improvement in production rhythm**, confirming that the proposed solution accelerates output without compromising individual operation quality.

Table 5.11 details the value-added time distribution across all active stations in L05B/V2. The most notable change lies in the addition of the Cobot station, which now concentrates the execution of the *Sensitivity Test*, *Welding*, and *Standby Test*. This station alone accounts for 11.30% of total value-added time and absorbs much of the workload that previously caused buffer congestion.

Manual Component Assembly continues to be the most labor-intensive station (23.10%), confirming its importance as a potential target for future automation. The cobot's integration allowed a reduction in loading-related time across other stations (e.g., *Soldering* loading fell from 5.45s to 1.95s), revealing improved flow coordination.

Additionally, the station *Assemble Individual Box* + ... + *Place* remains one of the most demanding in terms of human effort (11.00%), suggesting that while automation mitigated central bottlenecks, peripheral tasks still represent an opportunity for optimization.

Table 5.11: Value-Added Time Breakdown per Station - L05B/V2

Station	**Manual (s)**	**Automatic (s)**	**Loading (s)**	**% of Total Time**
Stick a Label L05B/V2	0.00	3.74	3.97	8.30%
Manual Component Assembly L05B/V2	21.59	0.00	0.00	23.10%
Workstation Space L05B/V2	0.00	0.00	3.15	3.40%
Soldering L05B/V2	0.00	13.80	1.95	16.80%
Middle Buffer L05B/V2	0.00	0.00	0.50	0.50%
Cobot (Sensitivity + Weld + Standby)L05B/V2	0.00	12.33	0.00	11.30%
Rotate Switch L05B/V2	2.11	0.00	0.00	1.90%
Encasing L05B/V2	0.00	3.92	0.00	3.50%
Workstation Buffer L05B/V2	0.00	0.00	2.76	2.40%
Final Test Individual L05B/V2	0.00	20.95	0.00	17.90%
Assemble Individual Box + ... + Place L05B/V2	7.07	0.00	5.83	11.00%

Table 5.12 provides a detailed breakdown of processing times and utilization across stations in theL05B/V2 configuration. The most significant improvement is the reduction of processing time in buffers, particularly the *Middle Buffer*, which dropped from an average of 1970.46 seconds to 810.31 seconds , a 58.9% reduction. Likewise, the *Workstation Buffer* saw a decrease of over 35 seconds in average time.

Despite these improvements, the buffers still account for considerable utilization: 40.42% at the *Middle Buffer* and 32.06% at the *Workstation Buffer*. This suggests that while the system is less congested than before, the line flow still suffers from pacing mismatches in certain transitions.

Active stations such as *Manual Component Assembly*, Cobot, and *Rotate Switch* remain under 1.5% utilization, consistent with high-throughput environments where individual operation times are short and well-synchronized. Overall, the station distribution indicates a healthier flow compared to the baseline, but with remaining potential for buffer control refinement.

Table 5.12: Processing Time and Utilization per Station - L05B/V2

Station	**Avg Time (s)**	**Throughput (pcs/h)**	**Utilization (%)**
PCB	29.25	123.0	1.46
Stick a Label L05B/V2	7.70	123.0	0.38
Manual Component Assembly L05B/V2	21.59	122.8	1.08
Workstation Space L05B/V2	3.30	122.8	0.16
Soldering L05B/V2	15.82	122.6	0.79
Middle Buffer L05B/V2	810.31	105.6	40.42
Cobot (Sensitivity + Weld + Standby) L05B/V2	29.19	104.6	1.46
Cobot Input Buffer L05B/V2	96.31	104.8	4.80
Cobot Output Buffer L05B/V2	268.51	102.6	13.40
Rotate Switch L05B/V2	29.09	102.4	1.45
Encasing L05B/V2	29.80	102.2	1.49
Workstation Buffer L05B/V2	642.70	98.2	32.06
Final Test Individual L05B/V2	20.95	98.2	1.05
Final Test Ready Buffer L05B/V2	0.00	98.2	0.00

The results obtained from L05B/V2 validate the effectiveness of cobot integration in mitigating structural bottlenecks and improving line performance. The reduction of the cycle time from 34.76 to 29.17 seconds confirms that output rhythm was significantly enhanced. Moreover, shifting the burden of three consecutive tasks to a single autonomous agent streamlined coordination and reduced the time parts spend in intermediate states.

Nevertheless, while the cobot solution addressed one of the line's most critical limitations, the presence of residual buffer accumulation and high manual workload in specific stations. such as *Manual Component Assembly*, highlights that the line remains partially unbalanced.

In conclusion, L05B/V2 demonstrates a measurable step forward in operational efficiency and provides a solid foundation for future improvement iterations. The simulation confirms that even partial automation , when well-targeted , can unlock significant productivity gains in manual assembly environments.

5.3.5 Cost-Benefit Considerations

To evaluate the financial viability of the proposed L05B/V2 solution, a cost-benefit analysis was performed comparing the productivity gains of the improved configuration against its implementation cost. The main objective was to estimate how many working days are saved and how quickly the investment is recovered through increased output.

Table 5.13 presents a summary of the key figures used in this analysis. The productivity increase from 103.56 to 123.40 pieces per hour translates to approximately **19% gain in throughput**. When considering the same production target of 2589 units (as produced in the baseline scenario), the improved configuration completes the task in just **2.72 working days**, compared to **3.25 days** in the original setup , a time reduction of roughly **0.52 days**.

Alternatively, if the line continues operating for the same duration as the baseline (3.25 days), the enhanced setup produces an additional **484 units**, resulting in an estimated profit increase of **4,163.53€**. This provides a clear opportunity for enhanced production flexibility, either by meeting the same targets faster or producing more within the same schedule.

Considering an implementation cost of **94,438€**, the estimated payback time for the investment is approximately **73.26 working days**, assuming the additional profit continues at this rate. This confirms the solution as a medium-term investment with scalable benefits.

Table 5.13: Cost-Benefit Summary - L05B/V2

Metric	**Value**
Baseline Daily Output	797.41 pcs
Improved Daily Output	950.18 pcs
Baseline Days to Produce 2589 pcs	3.25 days
Improved Days to Produce 2589 pcs	2.72 days
Time Saved	0.52 days
Extra Pieces in Equal Time	484 pcs
Extra Profit Generated	4,163.53 €
Implementation Cost (incl. losses)	94,438 €
Estimated Payback Time	73.26 days

In conclusion, the proposed solution demonstrates not only a clear productivity improvement, but also financial feasibility in the medium term. The integration of targeted automation through CoBots represents an effective investment in improving production flow, reducing idle times, and aligning resource usage with demand.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The simulation framework developed in this dissertation offers a lightweight, event-driven environment for executing rapid what-if analyses of assembly-cell configurations. By encapsulating operators, machines, and buffers into discrete parameterization classes, the framework transforms complex production scenarios into modular components. Execution times measured in minutes effectively compress days of physical testing into computational experiments, enabling cognitive evaluation of layout adjustments, task reallocation, and automation strategies without interrupting real-world operations.

6.1.1 Critical Assessment of Tool Capabilities

A neutral assessment of the framework reveals both strengths and limitations:

- **Configurational Agility:** The core abstractions support easy instantiation of alternative cell layouts. For example, introducing a CoBots into a single workstation and reassigning tasks previously handled by Operator 2 demonstrated the framework's flexibility with minimal code changes.
- **Computational Efficiency:** The discrete-event kernel handles queuing, resource contention, and task preemption with low-latency performance in standard use cases. However, simulations that lack dynamic buffer logic may return unrealistic throughput results under high utilization.
- **Model Fidelity vs. Practicality:** While cycle times are based on averages or literature-derived values, the absence of vendor-specific data for CoBots limits the predictive precision of the tool. Empirical calibration or integration with DT infrastructure would improve accuracy.

- **Analytical Transparency:** The framework provides detailed logs and timestamped records of all events. However, the lack of built-in visualizations or automated reporting means users must rely on external tools to interpret results effectively.
- **Human Factors Exclusion:** Current limitations include the absence of ergonomic metrics or models for operator fatigue. Safety margins and reach constraints are assumed rather than measured, which could lead to oversights in process design.

6.1.2 Alignment with Dissertation Objectives

The dissertation set out to achieve the following goals:

1. Map the main processes and operational routing in the Bosch Ovar assembly cells that could be automated using robotic systems.
2. Develop a simulation model using an appropriate platform (e.g., Arena, AnyLogic, FlexSim).
3. Simulate various operational scenarios and task-distribution strategies between operators and robots, taking into account production times and cadence, operating costs, and assembly-cell flexibility.
4. Perform scenario analyses of operator-robot task allocation, including a sensitivity analysis based on different levels of operator experience and training.
5. Conduct a cost-benefit analysis of the proposed solution.
6. Propose improvements, optimizations, and best practices for the current production processes in Bosch assembly cells, based on the simulation results.

All objectives have been addressed except for the final one, the 6th. Due to time constraints, detailed optimization proposals were not completed. However, the simulation tool and results form a strong basis for future analysis and recommendation work.

6.2 Future Work

To evolve the current framework from a proof-of-concept into a full-scale decision-support system and DT prototype, several directions are recommended:

6.2.1 Broadened Resource Modeling

The framework can be expanded to support a wider range of industrial elements by incorporating:

- **Machine Variants:** Include more models of machines, vision systems, and manual processes-

- **Dynamic Buffers:** Implement FIFO, LIFO, and priority-based buffer types to more accurately represent real queuing behaviors and reduce the risk of simulation deadlocks.
- **Cobot Diversification:** Add support for multiple CoBots platforms with detailed profiles.
- **Human Ergonomics:** Integrate ergonomic models to simulate fatigue, posture, and physical load, enabling better assessment of operator safety and task distribution.

6.2.2 Automated Analytics and Visualization

Future development should also focus on automating result interpretation through:

- Built-in generation of key performance indicators such as throughput trends, cycle-time histograms, and utilization heatmaps.
- Interactive dashboards or compatibility with business intelligence tools for real-time scenario exploration and sensitivity analysis.

6.2.3 Empirical Validation and Use-Case Expansion

Strengthening the framework's credibility requires broader deployment and testing:

- Conduct validation studies that compare simulation results with real measured data from active production lines.
- Collaborate with industrial partners to fine-tune simulation parameters and assess prediction reliability.
- Apply the framework to varied production contexts, ranging from high-volume, low-mix to high-mix, low-volume operations, to test adaptability and scalability.

6.2.4 Pathway to Digital Twin Realization

The long-term goal of transforming the simulation tool into a real-time DT system includes:

- **Real-Time Data Integration:** Connect the simulation engine to live production data using protocols like OPC/UA or MQTT to enable up-to-date status monitoring.
- **Continuous Calibration:** Employ feedback mechanisms such as Kalman filters or Bayesian inference to adjust model predictions based on real-world performance.
- **Predictive Maintenance:** Overlay statistical failure models on machine components to forecast breakdowns and optimize maintenance schedules.

6.3 Concluding Statement

This dissertation has demonstrated the development of a simulation framework designed to analyze and optimize human-robot collaboration within industrial assembly lines, with a particular application to Bosch Security Systems. The framework provides a flexible, scalable tool capable of modeling both real and hypothetical production scenarios with a high degree of accuracy. Through its discrete-event simulation core, the framework enables the detailed representation of human-machine interactions and allows for the evaluation of various configurations and task allocations in the assembly process.

The primary contribution of this work lies in its ability to provide manufacturers with a DSS that can simulate complex, hybrid production environments, offering valuable insights into the optimal deployment of human operators and CoBots. The tool's modular structure ensures that it can be adapted to suit the specific needs of different production lines, providing engineers with a means to experiment with various configurations, test different scenarios, and predict the impacts of changes in production processes before implementing them in the real world.

While the current framework demonstrates significant potential, several areas remain to be explored and refined. For example, operators behavior, ergonomic modeling, and integration with live factory data represent areas for future improvement. These enhancements would enable the framework to provide a more comprehensive, real-time planning tool for manufacturers.

Looking ahead, the framework could evolve into a DT platform, capable of supporting Industry 4.0 objectives by offering continuous, real-time analysis and optimization of manufacturing systems. With further empirical calibration and the integration of live data streams, the tool could transform into an even more powerful asset for production planning, improving efficiency, reducing downtime, and fostering smarter, more flexible production processes. As such, the work presented in this dissertation lays a solid foundation for future research and development, opening the door to new possibilities for the optimization of hybrid human-robot systems in industrial manufacturing.

References

[1] V.M. Antonova et al. “Some Features of AnyLogic Integration into the University’s Education System”. In: 2023. DOI: 10.1109/IEEECONF56737.2023.10092076.

[2] U. Asad et al. “Human-Centric Digital Twins in Industry: A Comprehensive Review of Enabling Technologies and Implementation Strategies”. In: (2023). DOI: 10.3390/s23083938.

[3] A. Attajer et al. “Benchmarking Simulation Software Capabilities Against Distributed Control Requirements: FlexSim vs AnyLogic”. In: (2021). DOI: 10.1007/978-3-030-69373-2_38.

[4] T. Bänziger, A. Kunz, and K. Wegener. “Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions”. In: (2020). DOI: 10.1007/s10845-018-1411-1.

[5] Alessio Baratta et al. “Towards Real-Time Task Allocation in Human-Robot Collaboration: Defining Key Requirements and Features for a Multi-Simulation Digital Twin System”. In: 2024. DOI: 10.46354/i3m.2024.emss.036.

[6] C. Cella, A. Maria Zanchettin, and P. Rocco. “Digital Technologies for the Design of Human-Robot Collaborative Cells”. In: 2023. DOI: 10.1109/MetroXRAINE58569.2023.10405765.

[7] The AnyLogic Company. *AnyLogic Features*. Accessed: November 1st 2025. 2025. URL: https://www.anylogic.com/features.

[8] The AnyLogic Company. *AnyLogic Simulation Software*. Accessed: November 1st 2025. 2025. URL: https://www.anylogic.com/features/enterprise-simulation.

[9] Sean Eom. “Reference Disciplines of Decision Support Systems”. In: *Handbook on Decision Support Systems 1 Basic Themes* (2008). DOI: 10.1007/978-3-540-48713-5_8.

[10] *Evolution of Decision Support Systems*. 2022. DOI: 10.1007/978-3-030-87790-3_3.

[11] P. Fritzson et al. “The OpenModelica integrated environment for modeling, simulation, and model-based development”. In: (2020). DOI: 10.4173/MIC.2020.4.1.

[12] Peter Fritzson. “The Openmodelica Environment for Building Digital Twins of Sustainable Cyber-Physical Systems”. In: 2021. DOI: 10.1109/WSC52266.2021.9715443.

[13] N. Gjeldum et al. “Collaborative robot task allocation on an assembly line using the decision support system”. In: (2022). DOI: 10.1080/0951192X.2021.1946856.

[14] S.E. Hashemi-Petroodi et al. “Workforce reconfiguration strategies in manufacturing systems: a state of the art”. In: (2021). DOI: 10.1080/00207543.2020.1823028.

[15] Jannicke Baalsrud Hauge et al. “Employing digital twins within production logistics”. In: 2020. DOI: 10.1109/ICE/ITMC49519.2020.9198540.

[16] S. Kassen et al. “Concept and case study for a generic simulation as a digital shadow to be used for production optimisation”. In: (2021). DOI: 10.3390/pr9081362.

[17] A. Keshvarparast et al. “Collaborative robots in manufacturing and assembly systems: literature review and future research agenda”. In: (2024). DOI: 10.1007/s10845-023-02137-w.

[18] M. Klumpp et al. “Production logistics and human-computer interaction—state-of-the-art, challenges and requirements for the future”. In: (2019). DOI: 10.1007/s00170-019-03785-0.

[19] Kateryna Kovbasiuk et al. “Analysis of the Selected Simulation Software Packages: A Study”. In: (2021). DOI: 10.22306/atec.v7i4.120.

[20] J. Lettori et al. “Transdisciplinary Evaluation of Simulation Software for Industry 4.0 Assembly Lines”. In: 2022. DOI: 10.3233/ATDE220671.

[21] Wojciech Lewicki, Mariusz Niekurzak, and Jacek Wróbel. “Development of a Simulation Model to Improve the Functioning of Production Processes Using the FlexSim Tool”. In: (2024). DOI: 10.3390/app14166957.

[22] Q. Liu et al. “Digital twin-based designing of the configuration, motion, control, and optimization model of a flow-type smart manufacturing system”. In: (2021). DOI: 10.1016/j.jmsy.2020.04.012.

[23] E. Mahmoodi et al. “Data-driven simulation-based decision support system for resource allocation in industry 4.0 and smart manufacturing”. In: (2024). DOI: 10.1016/j.jmsy.2023.11.019.

[24] Maria Marques et al. “Decentralized decision support for intelligent manufacturing in Industry 4.0”. In: (2017). DOI: 10.3233/AIS-170436.

[25] Gerald Nqobile. *Simulation Programming with Python*. Accessed: November 1st 2025. 2016. URL: https://www.academia.edu/26475684/Simulation_Programming_with_Python.

[26] Onur Özgün and Yaman Barlas. *Discrete vs. Continuous Simulation: When Does It Matter?* Accessed: November 1st 2025. 2009. URL: https://proceedings.systemdynamics.org/2009/proceed/papers/P1199.pdf.

[27] T.Y. Pang et al. “Developing a digital twin and digital thread framework for an ‘industry 4.0’ shipyard”. In: (2021). DOI: 10.3390/app11031097.

[28] Real Python. *SimPy: Simulating Real-World Processes With Python*. Accessed: November 1st 2025. 2022. URL: https://realpython.com/simpy-simulating-with-python/.

[29] A.C. Simões et al. “Designing human-robot collaboration (HRC) workspaces in industrial settings: A systemic literature review”. In: (2022). DOI: 10.1016/j.jmsy.2021.11.007.

[30] Anmolika Singh. *Agent-Based Modeling*. Accessed: November 1st 2025. 2025. URL: https://builtin.com/articles/agent-based-modeling.

[31] M. Singh et al. “Applications of Digital Twin across Industries: A Review”. In: (2022). DOI: 10.3390/app12115727.

[32] AnyLogic Team. *AnyLogic Cloud Models*. Accessed: November 1st 2025. 2025. URL: https://cloud.anylogic.com/models.

[33] AnyLogic Team. *AnyLogic Simulation Software*. Accessed: November 1st 2025. 2025. URL: https://www.anylogic.com/.

[34] OpenModelica Team. *OpenModelica User's Guide*. Accessed: November 1st 2025. 2025. URL: https://openmodelica.org/doc/OpenModelicaUsersGuide/latest/.

[35] C. Thongdonnoi et al. "Application of Collaborative Robots for Increasing Productivity in an Eyeglasses Lenses Manufacturer". In: (2023). DOI: 10.4186/ej.2023.27.10.93.

[36] Miguel Vieira et al. "A two-level optimisation-simulation method for production planning and scheduling: the industrial case of a human–robot collaborative assembly line". In: (2022). DOI: 10.1080/00207543.2021.1906461.

[37] Miguel Vieira et al. "Simulation-optimization approach for the decision-support on the planning and scheduling of automated assembly lines". In: 2018. DOI: 10.1109/CONTROLO.2018.8514297.

[38] Dmitry Zinoviev. "Discrete Event Simulation: It's Easy with SimPy!" In: *Suffolk University, Boston* (2023). DOI: 10.5281/zenodo.7575671.