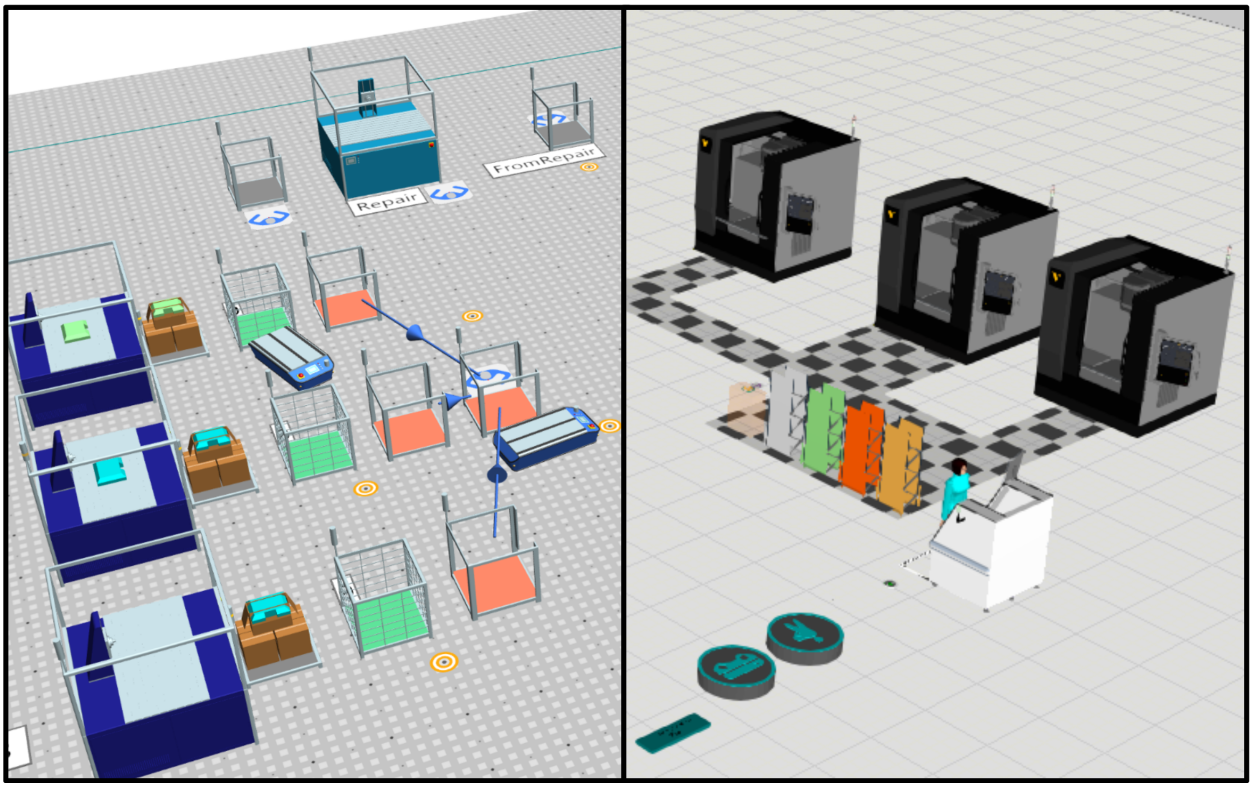
CHALMERS
UNIVERSITY OF TECHNOLOGY



Evaluation of Discrete Event Simulation as a Tool for Modelling and Analysing Manual-Robotic Production Systems

Simulation of an Automated Product Handling Solution for Testing Operations

Master's thesis in Production Engineering

Emelie Gillerstedt
Maja Svärd

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Evaluation of Discrete Event Simulation as a Tool for Modelling and Analysing Manual-Robotic Production Systems

Simulation of an Automated Product Handling Solution for Testing Operations

EMELIE GILLERSTEDT
MAJA SVÄRD

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
Gothenburg, Sweden 2026

Evaluation of Discrete Event Simulation as a Tool for Modelling and Analysing Manual-Robotic Production Systems
Simulation of an Automated Product Handling Solution for Testing Operations
EMELIE GILLERSTEDT
MAJA SVÄRD

Supervisors:
Jacob Gyllenberg
Torbjörn Hjorter
Rasmus Johansson
Anita Clara Notarianni, Department of Industrial and Materials Science
Examiner: Anders Skoogh, Department of Industrial and Materials Science

Master's Thesis 2026
Department of Industrial and Materials Science
Division of Production Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Screenshots of the simulation models made in Visual Components and Tecnomatix Plant Simulation.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Evaluation of Discrete Event Simulation as a Tool for Modelling and Analysing Manual-Robotic Production Systems
Simulation of an Automated Product Handling Solution for Testing Operations
EMELIE GILLERSTEDT
MAJA SVÄRD
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

This thesis investigates how Discrete Event Simulation (DES) can be used to model and analyse an automated solution for a production system. The simulation models built and run in this study are based on two different layouts of a quality control production system where two cobots have been implemented in order to automate the system. The aim of the thesis was to build the simulation models as well as evaluate the two different programs used. The 3D visualisation program used for this thesis is Visual Components and the advanced DES program used is Tecnomatix Plant Simulation. The results show that for the purpose of the simulation models, while Visual Components has clear visual benefits compared to Tecnomatix Plant Simulation, Tecnomatix Plant Simulation was the better equipped simulation tool. An effective method to build simulation models of this complexity is to work iteratively, introducing one behaviour at a time. Of the two different layouts, the second layout introduced seems to have the most benefits.

Keywords: Discrete Event Simulation, Tecnomatix Plant Simulation, Visual Components, Evaluation, Cobot, Industry 5.0, Simulation modelling.

Acknowledgements

The authors of this thesis would like to extend our gratitude to our supervisor at the department of Industrial and Materials Science at Chalmers University of Technology, Anita Clara Notarianni, who provided us with valuable feedback and positive encouragement. We would also like to thank Anders Skoogh, our examiner. Additionally, we want to thank our supervisors at the company that this thesis was done in collaboration with. Thank you Jacob Gyllenberg, Rasmus Johansson and Torbjörn Hjorter for your much appreciated guidance and support throughout the project. The members of the team that we got to work in at the company were also very welcoming and supportive. Lastly, we are grateful to the company that let us work with this exciting project and work on site at their office space.

Emelie Gillerstedt & Maja Svärd, Gothenburg, May 2026

Acronyms

AGV	Automated Guided Vehicle
AMR	Automated Mobile Robot
ATE	Automated Test Equipment
Cobot	Collaborative Robot
CPS	Cyber-Physical Systems
DES	Discrete Event Simulation
KPI	Key Performance Indicator
MTBF	Mean Time Before Failure
MTTR	Mean Time To Repair
TBL	Triple Bottom Line
TPS	Tecnomatix Plant Simulation
VC	Visual Components

Contents

List of Figures

List of Tables

1 Introduction

This report was conducted in collaboration with a company within the aerospace and defence sector, with focus on the surveillance part of the company. As the company expands and improve their production and move towards Industry 5.0 principles, there is an increasing need for efficient production planning tools such as discrete event simulation (DES) programs. These programs can therefore be used to analyse and visualise production systems before implementing changes in real-world system.

Existing research regarding simulation systems is currently either focused one development of one certain model or a broad comparison of several programs. Therefore, there is a lack of studies that compare a limited number of simulation programs more in depth.

1.1 Background

In recent years, the demand for surveillance related products has increased significantly, driven by factors such as increased security requirements and evolving geopolitical and societal conditions. According to OECD (2025), the economic security risk in the world have increased due to recent events such as the COVID-19 pandemic, volatile energy markets and war and therefore geopolitical and societal tensions have arisen. As a result, the global security market has experienced significant growth, and is expected to keep growing due to the increased security concerns as well as technological advances (Security Industry Association & ASIS International, 2024).

This situation increases the pressure on manufacturing systems to increase their volumes while keeping strict quality standards. As a result, optimised production and resource efficiency have become key focus areas within the industry, alongside the transition towards Industry 5.0. Specifically, maximising the use of existing equipment without compromising quality is highly relevant, as investing in new equipment is often time consuming and expensive.

1.2 Purpose

Within the surveillance sector at the company, there are several production processes that take place. The recent increase in demand has created a need for more efficient production. To address this, several parts of the production needs to be

investigated to identify possible optimisation opportunities. Due to the strict reliability requirements for products of this nature, the quality aspect of the production is critical. One area of focus is increasing the utilisation of current Automated Test Equipment (ATE) in one of the company's production areas. The purpose of the ATEs are to test the products and detect any electronic faults. Thereby, a suggested solution is to incorporate Automated Mobile Robot (AMR) systems, further referred to as collaborative robots (cobots) or Automated Guided Vehicles (AGV), that can deliver the products to these ATEs, including during overnight operations.

1.3 Aim

The main aim of the project is to build and use a DES model of the quality control process. A central outcome is therefore a functional 3D model of the selected production area. This model includes the test stations and the cobots. There are different versions of the models, which represent the first proposed setup given, as well as another setup that was introduced during the course of the project.

By analysing the simulation, production data such as queue lengths, waiting times, and throughput can be interpreted. The model can be used by the company to evaluate different scheduling strategies as well as how repair stations can be incorporated into the system. The simulation results can additionally be used to identify bottlenecks and compare different production strategies.

Finally, the project also assessed the suitability of two different simulation programs; Visual Components (VC) and Tecnomatix Plant Simulation (TPS), for this type of simulation task. This includes an evaluation of how well the software can represent production flow and the movement of the cobots. The findings will provide input for future work related to simulating and optimising production systems.

1.4 Research Questions

In the beginning of the project, the following research questions were formulated to guide the research:

- *"How can discrete event simulation be used to model a manual-robotic production system?"*
- *"What simulation software capabilities are important for modelling and analysing a manual-robotic production system?"*

By addressing these questions, the study investigates how to optimise the electronics quality control area at the company and how discrete event simulation can help simulate the production line to help planning around the clock quality control using cobots.

1.5 Delimitations

The production consists of testing, assembly and repair of electronic assemblies. The main focus for this simulation model is the testing. The assembly and repair parts of the production environment were not analysed as thoroughly; they are only considered to the extent necessary in order to set physical limitations to where the cobots can operate at the moment.

Limitations may also occur within the program itself. In VC, the simulation part may not function ideally, which could cause hinders and unsatisfactory results. Furthermore, time constraints may become a limitation. As this is a masters thesis within the Department of Industrial and Materials Science At Chalmers University, the timeline of the project is only 20 weeks which limits the possibility to thoroughly work through the simulations and identify all possible issues. The timeline is also limiting if unforeseen issues arise when building the simulation models.

1.6 Disclaimer

During the model building, the AI chatbots Chat GPT and Copliot have been used to look for bugs in both programs and assist with the SimTalk code in TPS. AI has also been used to assist with word choices and sentence structures while writing this report.

2
Literature Study

In order to compile relevant literature related to the subject, scientific papers, reports and handbooks from Chalmers Library online and Scopus database have been analysed using the following keywords when searching: "Discrete Event Simulation", "Production", "Cobot", "Digitalisation" and "Data Management". All articles before the year 2000 are subsequently filtered out to make sure that the information isn't outdated. Unrelated subjects have been filtered out through the filter options within the databases. On Scopus, the topic engineering was used as a filter to avoid looking through irrelevant articles. After searching for the keywords in different combinations with boolean operators, articles were hand-picked by reading the headlines as well as the abstract of the article. Some articles were removed at a later stage during a more in-depth reading due to not being relevant or useful enough for the project. In some cases, articles were considered too old and therefore less relevant. In other cases, a deeper analysis revealed that the articles did not align with the focus of this thesis, either because they did not address the type of simulation being studied or because they were not related to production systems. Additional sources have been found through recommendations from supervisors and recommended sources from relevant courses.

2.1 Industry 5.0

The company that this thesis is in collaboration with is currently undergoing significant expansion and is therefore exploring ways to optimise parts of its production while moving towards the principles of Industry 5.0.

Industry 5.0 is the latest phase of industrial development and aims to reverse potential disruptions that may have arisen from Industry 4.0 in production systems and regarding sustainability (Dacre et al., 2025).

As described by Zhang et al. (2026) and Fonseca et al. (2025) Industry 4.0 is characterised by the digital transformation of manufacturing and supply chains through the integration of key technologies such as cyber-physical systems (CPS), the Internet of Things, big data and Artificial Intelligence (AI). These contribute to increased information flow and connectivity and therefore also increased automation and data driven decision making. Industry 5.0 has subsequently evolved from Industry 4.0, aiming to further develop the technologies while addressing the societal and envi-

ronmental implications.

The purpose of Industry 5.0 is to focus on improving productivity while also addressing the well-being of workers and minimising the environmental impact (Pan et al., 2026). Built on the foundation of Industry 4.0, this next phase of industrial development moves beyond smart manufacturing and its multiple technologies in manufacturing processes (Gunal, 2019). Industry 4.0 was needed to address the complexity of new and evolving processes and products, however, Industry 5.0 represents the shift from efficiency oriented production to human-centric, sustainable and resilient industrial systems. The European Commission (2024), describes Industry 5.0 as a key factor in the economic and societal transformations that are currently happening, and in order to remain relevant industries must work toward leading the green and digital transitions.

A central pillar of Industry 5.0 is human-centric manufacturing systems as described by Pan et al. (2026). This entails designing manufacturing systems to allocate tasks based on the respective advantages of human or machine. the aim is for humans to focus on work requiring decision making and creativity, while the machine adopts the more repetitive or hazardous tasks. Pan et al. (2026) specifically mentions how this will improve operational efficiency while increasing well being among workers.

Another central characteristic of Industry 5.0 is resilience in relation to disruptions. According to Dacre et al. (2025), resilience entails identifying both vulnerabilities and capabilities. Vulnerabilities include anything that increases the systems susceptibility to disruptions. These disruptions can be both internal within the production concerning equipment breakdowns, safety, delays, or quality, or external, therefore originating outside the production such as geopolitical instability or delays/cancellations. (Pan et al., 2026) Capabilities denote the ability to anticipate, mitigate and recover from such disturbances.Dacre et al. (2025). Thereby, manufacturing systems must be adaptive and capable of responding to rapid changes in unpredictable environments.

Finally, sustainability is an important factor in Industry 5.0. The definition of sustainability is described by the United Nations Brundtland Commission as “meeting the needs of the present without compromising the ability of future generations to meet their own needs” (United Nations, n.d.). Therefore sustainability and thereby sustainable development has become a priority in modern manufacturing. Being able to save time by having access to much more data due to digitalisation is fundamental for quick decision making, lowering costs, and optimising resource use and waste management (Gunal, 2019). Within Industry 4.0 and 5.0, the sustainability perspective extends beyond environmental concerns to include social and economic factors. Dacre et al. (2025) describes how Industry 5.0 promotes reduced emissions, improved product efficiency while supporting employee well-being and maintaining sustainability as a core principle in manufacturing systems. Furthermore, as discussed by Pan et al. (2026), advanced technologies, such as artificial intelligence, support sustainability transformations by optimising energy consumption, and im-

plementing predictive maintenance and thereby reducing waste and improving overall system efficiency.

2.2 Discrete Event Simulation

Discrete event simulation (DES) is a methodology for modelling and analysis of systems that changes over time due to different events. Furthermore, it can be used to estimate and assess the performance of a system (Fishman, 2001).

Moreover, Fishman (2001) discusses how DES relates to modelling and why it is used. First, this method includes creating a model based on theory and empirical observation which then can be used for studying complex systems. Strong justifications for using DES according to Fishman (2001) include systems that are complex with many interactions and dependencies where several variations and variables are tested.

Banks (2000) provides a structural proposal when modelling production systems that supports the methodology this thesis follows. Firstly, Banks (2000) emphasises the collection of data for modelling which then moves on to creating a DES model, allowing for system testing and evaluation. The scenario based modelling can thereby be used to analyse the system performance.

2.3 Program Categories

The two different simulation programs used can be divided into two different categories. Advanced DES-programs and 3D-visualisation programs. Although both programs are capable in both categories, TPS has a DES core and excels at complex DES modelling (Ivanov et al., 2025), which sets it into the advanced DES-programs category. The strength of VC lies partly within the 3D and visual aspect of the program, making it fit into the 3D-visualisation category.

2.3.1 3D Visualisation Program

VC was developed in 1999 and has continuously been improved since. It is used for simulating and modelling manufacturing systems. It is widely used for building 3D factory layouts, simulating robotic operations and, validating throughput analysis. The program has a built in library of components and also supports Python scripting which allows for customisation within simulations. The software can also integrate real time and historical data which aligns with smart manufacturing principles (Bartoș et al., 2025).

2.3.2 Advanced DES Program

TPS is a discrete-event simulation software which is used to model, analyse, and optimise manufacturing and logistics systems. With it digital representations of production processes can be created that allow engineers to evaluate system performance without affecting real world operations. The software supports detailed modelling of production elements such as workers, machines, and material flows. Furthermore, it provides tools for analysing performance factors such as throughput, process times and bottlenecks. TPS also supports further statistical analysis and evaluation of different scenarios, making it valuable in production improvement and decision making (Hoang et al., 2024).

2.4 Cobots

A similar project was presented by Sze et al. (2024), who simulated a pick and place system for a cobot. A cobot is the short term for collaborative robot, which is a robot engineered to safely work alongside humans in various systems. The paper by Sze et al. (2024) demonstrates how tasks such as pick-and-place can be broken into smaller, sequential steps, which supports using the approach of DES to simulate a system.

The article by Nogueira et al. (2019), further explains cobot behaviour driven by events. While the paper does not explicitly use DES, its event driven view of cobot behaviour provides a useful basis for modelling cobots in a DES context.

2.5 Data Collection

Data collection is a big part of any simulation project (Skoogh & Johansson, 2008), and proper input data management is a crucial part of this step to reduce the amount of time that data collection can require. Skoogh and Johansson, 2008 presents a methodology as shown in figure 2.1, to reduce the time spent on identifying parameters needed for the simulation model. In addition to reducing the time, proper input data management reduces the risk of delays, insufficient model credibility and unnecessary reworks during later stages of the simulation.

As seen in 2.1, the data collection consists of four different stages; "identify and define relevant parameters", "specify accuracy requirements", "identify available data" and "choose methods for gathering of not available data".

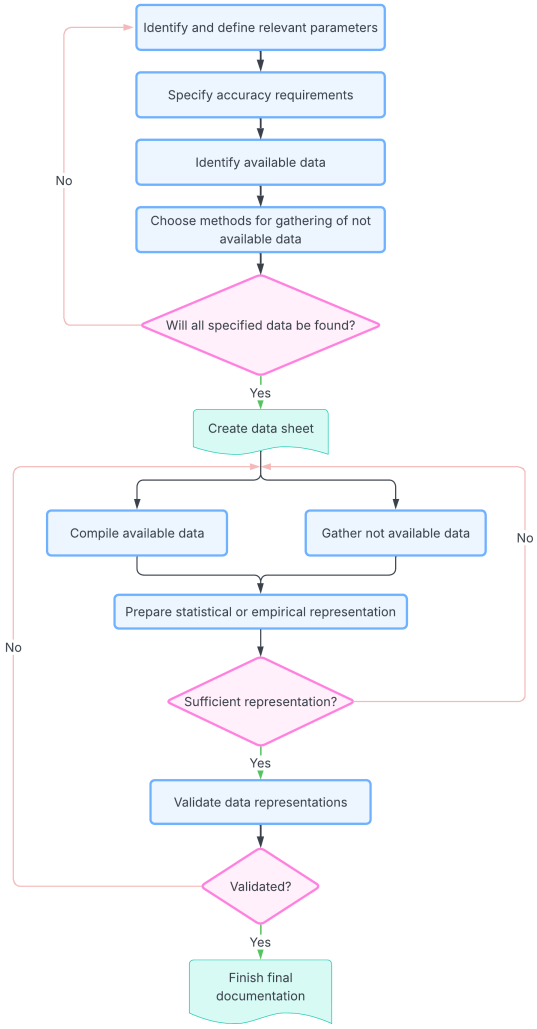


Figure 2.1: Visualisation of a proposed methodology for input data management based on Skoogh and Johansson (2008)

2.5.1 Identify and Define Relevant Parameters

The first stage of the input data methodology is to determine what data is required to make a realistic and credible simulation. This can be done by observing the system in detail, and conducting interviews with experts in the area (Skoogh & Johansson, 2008). The appropriate level of detail need to be determined and the relevant parameters. Moreover, a clear standard of how the parameters are measured need to be in place as it may not be clear how to define each parameter, to mitigate this problem, a clear discussion with company experts should be held to understand how parameters are normally defined (Skoogh & Johansson, 2008).

2.5.2 Specify Accuracy Requirements

After identifying and defining parameters, it's important to differentiate the different accuracies needed for each parameter (Skoogh & Johansson, 2008). While some parameters need a higher level of accuracy, others may not affect the simulation to the same extent. For these parameters, lesser consideration can be given to allocate more time to accurately represent the more important parameters. Similarly to when identifying the parameters, a discussion with experts in the system is important as they possess the most knowledge in the field. A factor that plays a significant role is the variability (Skoogh & Johansson, 2008). Parameters that vary frequently need extra attention so they can be represented in a realistic way in the simulation. Examples of these given in Skoogh and Johansson, 2008 are parameters related to repair and breakdowns, as they tend to vary for different cases.

2.5.3 Identify Available Data

The next step is to identify the currently available data, namely the secondary data. By following the previous steps, the gaps in the information can be systematically identified to prepare for gathering the non-available data.

2.5.4 Choose Methods for Gathering of Not Available Data

If the available data is insufficient, which is the most likely case in a simulation study (Skoogh & Johansson, 2008), methods for gathering required additional data need to be determined. A common way of doing this is to conduct a stopwatch study, depending on the system and the circumstances, other methods such as a time study with video, work sampling and predetermined time system could be more suitable (Almström et al., 2024). The advantages with a stopwatch study is that it's fairly easy to perform, as long as the start- and stop time of the parameters have been defined during the first stage of the input data management. One disadvantage with this method is that the results may not be at a satisfactory level of detail (Skoogh & Johansson, 2008; Almström et al., 2024). Further advantages and disadvantages with the different mentioned methods can be seen in table 2.1 below.

Method	Purpose	Pros	Cons
Stopwatch time study	Measure time for work sequences in production. Measure machine times.	Fast and easy to perform. Easy to understand the result.	The analyst disrupting workflow
Video time study	Measure time for work sequences in production. Method improvement	Easy to perform. Does not interfere with the workflow. Easy to involve operators.	Requires specific technology.
Work sampling	Pre-study for improvement projects. Measure the allowance time.	Easy to perform. The only way to measure allowances.	Complicated design and analysis.
Predetermined time system	Design a norm time. Method improvement. Performance measurement.	The only way to design a manual work time. Objective and fair. Detailed method analysis.	Requires training. Time consuming.

Table 2.1: Methods to collect data based on Almström et al. (2024)

Skoogh and Johansson, 2008 mention several other methods to choose from, each suitable for different situations.

Following the initial steps of the data input management, an evaluation is necessary to check if all the available data was collected and a data sheet is created. All the secondary data can be compiled into the data sheet while the primary data has to be gathered by using the previously selected method.

2.6 Simulation Study

Figure 2.2 by Banks (2000) shows the steps that can be taken when carrying out a simulation study with a systematic approach. First of a clear formulation of the problem needs to be made. Then objectives need to be clarified together with an overall plan of the project. The plan should describe software and personnel requirements, outputs and the different scenarios that will be investigated throughout the study.

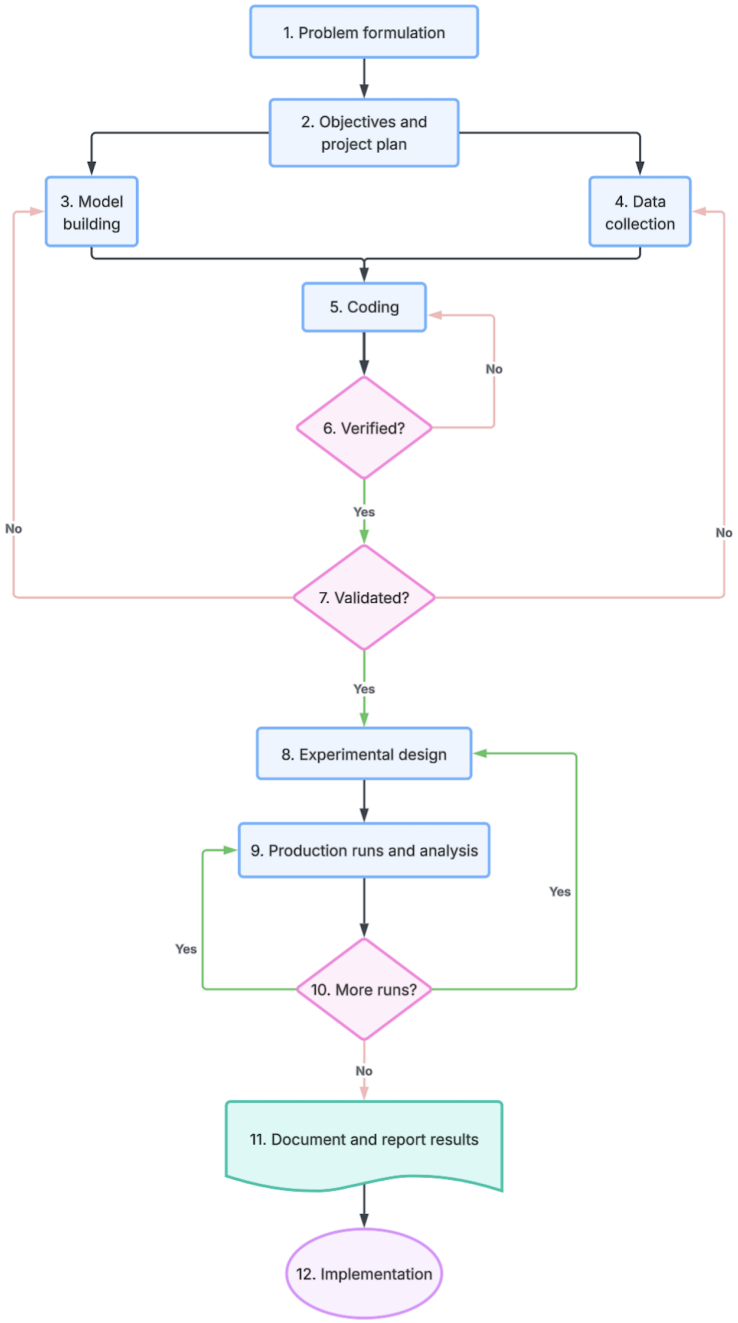


Figure 2.2: Steps to take in a simulation study based on Banks (2000)

2.6.1 Conceptual Model

Firstly, based on the project plan and other provided information, a conceptual model (step 3 in figure 2.2) can be created. This model represents the production system and contains key components regarding the mathematical relationships and capabilities within the production system. When creating this model, Banks (2000) also emphasises the importance of keeping it as simple as possible. Increased complexity may be useful to some extent, however, beyond a certain point it mainly increases the costs and time frame without increasing the quality of the final product.

Furthermore, "Model building" and "Data collection" from steps 3 and 4 in figure 2.2 have separate feedback loops to allow for simulation analysts to continuously work on the model while the data collection is advancing.

2.6.2 DES model

After finalising the conceptual model, it is converted into a operational DES model. This model is then analysed and tested to ensure it acts as the real-world system. When verifying the model, step 6 in figure 2.2, it is important to verify continuously instead of waiting until the model is finished (Banks, 2000). This involves reviewing process flows step by step, testing individual sections and debugging the system to detect errors early on.

Step 7 involves making sure that the conceptual model is a correct representation of the real production system. This ensures that the simulation model can be used as a virtual representation of the physical system.

2.6.3 Experimental Design

After creating the operational DES model, experimental design is applied to simulate each scenario. The system is then consequently evaluated and analysed under different circumstances. Banks (2000) further states the importance of documenting and reporting throughout the project in case the simulation is to be used again or modified for example. Finally, the reported outcomes from the simulation can be implemented in the production system.

2.7 Verification and Validation

Verification and validation are both essential steps in simulation modelling in order to ensure that the model accurately represents the real world system. Model verification refers to if the model has been correctly implemented according to the original design. Validation concerns identifying if the simulation model is an accurate representation of the real world problem (Sargent, 2010).

Furthermore, Hora and Campos (2015) mentions the importance of realising that a simulation model does not need to replicate the real world system perfectly in order

to be considered valid. Instead, it is important to note that a simulation model is constructed under specific conditions and a certain hypothesis, which should therefore be taken into account when evaluating the validity of the model.

When validating model, debugging and testing are important steps in order to ensure that the model behaves as it was intended. Debugging is also useful in verifying the code and make sure it functions correctly. Moreover, extreme cases can be used to stress test model in order to identify potential issues. However, Hora and Campos (2015) also mentions that it is not possible to exhaust all possible combinations and relevant tests, meaning that it is impossible to prove that a model is perfectly correct.

Therefore, the aim is instead to test the model sufficiently for its intended purpose. Some simulation models are designed to predict future behaviour, while others are made to to represent observations or historical data, which should be taken into consideration when verifying and validating the model (Hora & Campos, 2015).

2.8 Evaluation

Since an important aspect of the project is the evaluation of the two different programs, proper ways to evaluate and compare the program were required to a fair and reliable assessment. In the article by Ivanov et al., 2025, several simulation programs were reviewed based on criteria involving the different technical, analytical, organizational, economic and educational aspects. Among the nine different programs that were evaluated, both TPS and VC were included. This article is highly relevant for this project as it details the different criteria Ivanov et al., 2025 used, and how the criteria is defined, giving insight to how simulation programs can be evaluated and on what basis.

The criteria used to evaluate the programs were divided into five different categories; technical, analytical, organizational, economic and educational. Each category has a few criteria that was used for the evaluation. When setting this projects criteria, the choices were influenced by this article, the specific requirements of this project and the company as a whole.

In addition to evaluating the different simulation tools, the simulation models themselves were also analysed. This assessment focused on how well the simulations represent the real-world situation. The focus here is on how well the simulation can implement different functions, logic, and key system behaviours. Furthermore, iterative model building was used to validate the models throughout the simulation development.

Likewise to the evaluation framework proposed by Ivanov et al., 2025, Ciauşu-Sliwa et al., 2025 further emphasises the increasing importance of DES in smart manufacturing among other fields. Similar to Ivanov et al., 2025, they present different simulation tools and evaluates them on nine different criteria that was determined to be of relevance to industrial applications. While Ciauşu-Sliwa et al., 2025 presents

each program in more depth compared to Ivanov et al., 2025, little to no practical research was documented, as in making a simulation model in the different programs.

2.9 Thesis Contribution

Simulation tools are commonly used in production environments. Current research primarily focuses on the development of simulation models or the direct comparison of the program interfaces. This thesis adds to the research by instead comparing two simulation tools through modelling a real-world system.

By limiting the study to two simulation software, the analysis allows for a more detailed and in-depth comparison between the two. The same model was built in both programs, with emphasis on keeping the logic and functions of the models as similar as possible. This approach enables a more direct and fair comparison of how each software handles the making of a relatively complex production area.

While much of the current research, such as Ivanov et al. (2025), Hora and Campos (2015) and Ciauşu-Sliwa et al. (2025), evaluates strengths and weaknesses across multiple simulation software, this study prioritises depth over breadth. Focusing on only two simulation tools makes it possible to explore their capabilities and limitations in more detail.

By comparing the programs with respect to previously determined evaluation criteria, this study provides practical insight into the strengths and limitations of each program. Furthermore, this thesis also explores evaluating different system layouts in each program.

3
Methods

3.1 Gathering Data

Before developing the model and running simulations, data must be collected from the production line to ensure the simulation closely reflects real-world conditions. The data can be divided into two different categories; primary data and secondary data (Ajayi, 2023).

Primary data is the data that is gathered for the first time during the project and will be collected through consultations with employees and observations at the production line. During the project, the primary data collected was the time to change fixtures in between product types and the time to calibrate after changing them. This time is summed up to a total 'changeover time'.

Secondary data is the data that has already been gathered or produced by others. Since the defence company has security restrictions, the data was gathered through reaching out to the appropriate personnel. The secondary data that was used during this project is the yield of each product and the processing time of each product.

Some of the data was not available, and would not become available during the time frame of the project. This data had to be fabricated or estimated. The biggest contributor to the results of this data is the time it takes to repair a product. Multiple attempts to estimate a mean repair time was made, but due to the unpredictability of the repair, no data was collected. For the purpose of this project, the repair time was fabricated, while leaving room to adjust it at a later stage.

The speed of the cobots were similarly estimated, since there was no available information regarding which brand of mobile robot will be used. The cobot speed does not affect the simulation results in the same magnitude as the repair times, as the travel time for each product is such a small percentage of the products total time in the production area. When estimating the speed of the robots, the walking speed of an average human was used as a reference. This was suitable since the robots are integrated into the system in order to carry out tasks previously performed by humans. This comparison also allows for an evaluation into how much more efficient the robots can be through around-the-clock operation, even if they move at approximately the same speed as a human.

As for the estimation of batch sizes and how frequently new batches should arrive, several assumptions had to be made due to the lack of available data. In order to fully test the system, all products were incorporated with relatively small batch sizes in order to fully test changeovers and the versatility of the system. The number of fixtures per part, the part priorities as well as processing time was also taken into account when determining the batch sizes. An estimate was then made on roughly how long time it would take the machines to complete all the batches, and based on this, arrival times for new batches were estimated in order to optimise the system while still ensuring that all product types were introduced at some point.

3.2 Selection of Software

VC was selected as the primary simulation tool, as the company where the production line is located already uses this software and expressed a preference for developing the model within that environment. However, there was also an interest in exploring alternative simulation tools and their capabilities. Therefore, it was decided to develop a comparative model in a second software. TPS was selected for this purpose due to its availability through the university, as well as being a well known simulation program.

3.3 Conceptual model

Based on the project plan as well as other provided information a conceptual model of the system was created. This model shows the basic operational steps that will take place and provide a foundation for the operational DES model. The overall structure of this model was also kept very simple since increased complexity was strongly discouraged by Banks (2000).

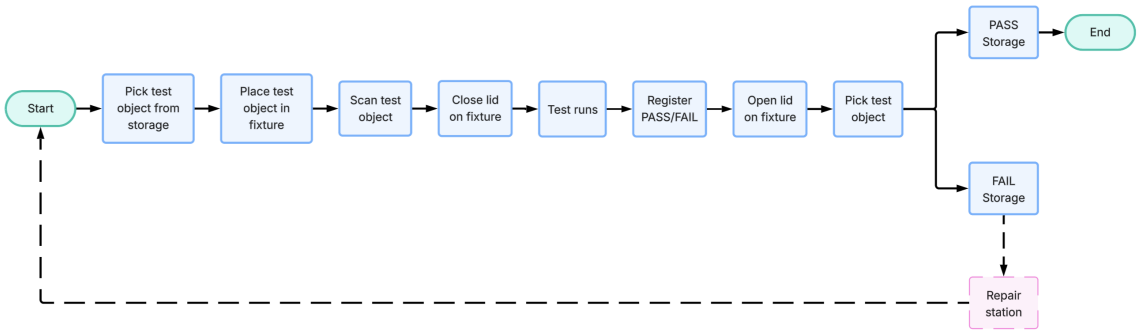


Figure 3.1: The conceptual model for the simulated system

the conceptual model represents the product flow through testing , including pass and fail outcomes. The failed products enter a repair loop and are reintroduced back into the system. The cobots are then responsible for transporting the products between these stages.

3.4 DES model

After gathering enough data and finalising the conceptual model, a realistic model was created in the programs VC and TPS. The models were used to simulate different scenarios and visualise the movement of the two cobots. The majority of the model building and simulation was conducted at the company, partly because of security reasons but also for the ability to discuss with employees how to improve the model and simulation further.

3.4.1 Visualisation of system

Moving on from the conceptual model, the real system was visualised as shown in figure 3.2. The layout consists of two separate systems each containing one cobot and three ATE machines. Each area also has storage where the cobot picks up new products as well as leaves finished ones. There is also a repair buffer where faulty parts are placed by the robot in wait for repair. Each cobot also has a charging station it returns to when idle. There are also some workplaces in the production that have been included in the illustration to further emphasise the restriction of the robots movement.

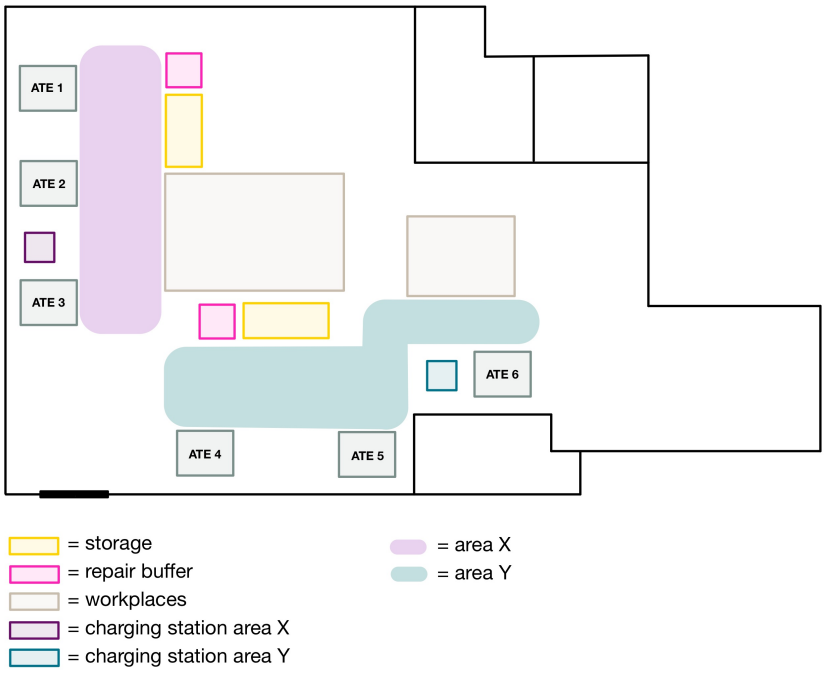


Figure 3.2: Layout 1 visualisation

Since the system doesn't exist in the real world yet, different concepts of the system emerged during the course of the project. Because of this, a different visualisation as can be seen in figure 3.3 was made to reflect another concept of how the system

should look and work. The purpose of this model is to reduce travel time of the robot by placing storage, pass and fail at each ATE respectively allowing the arriving robot to pick up the product, insert product into machine, all from the same spot. Similarly when picking up a finished product, it can easily be put directly into the pass or fail buffer.

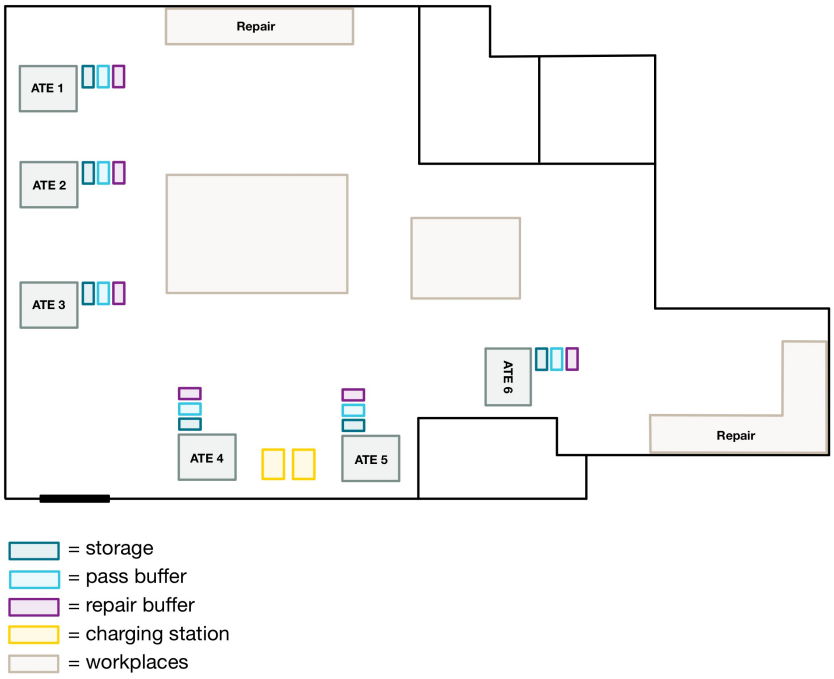


Figure 3.3: Layout 2 visualisation

3.4.2 Ideal system logic

The system was required to perform several basic functions. As previously mentioned, the system consist of two cobots and six different ATEs. The cobots' mission is to deliver products to and from the ATEs. The cobot is restricted so that it can't deliver more products to the ATEs than there are existing fixtures for that product. In between different products the fixture needs to be changed, which will lead to a changeover, therefore the cobot should also prioritise delivering products to an ATE that previously worked with the same product. Each product has different processing times and yields, so the system will take these into regard when calculating the processing time and whether the product passes or fails the test.

If a product passes, it should be delivered to the correct storage for passed units, and if it fails it should be delivered to a storage for failed units. The failed units then enter a repair process. The repair process has operators repairing the failed products and then returning them to the robot for it to be tested in an ATE again.

There are different types of products, type one and two. Three ATEs are designated for type one and the other three are designated for type two.

Initially each cobot was supposed to be responsible for each product type. As the second concept was introduced, where both cobots should be able to handle both types of products an additional scenario was created for that version. A small changeover should be added to account for a tool change the robot does before a it picks up a product of a different type and previously transported.

To simplify the creation of the systems in a simulation program, the logic was visualised with an UML (Unified Modelling Language) diagram, as show in figure 3.4. The UML diagram represents how the ideal logic of the simulations should work. The UML diagram doesn't show the different products or the ATEs, but as they all work in the same way, they aren't necessary to include in the diagram.

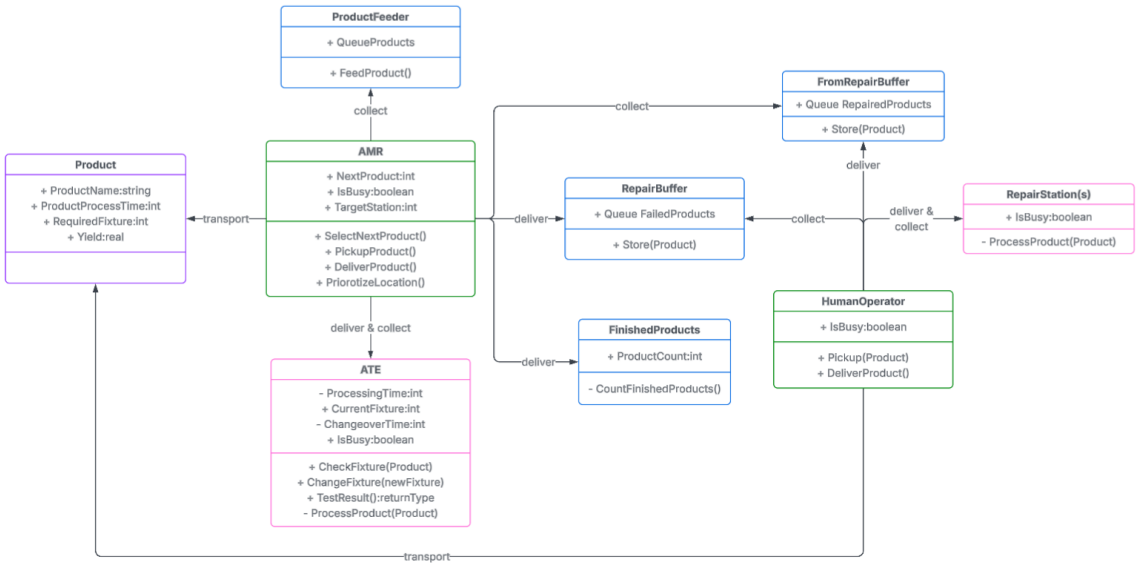


Figure 3.4: System logic visualisation

This logic may not be possible to execute in the desired way for every simulation program, but is a realistic representation of how the system should work. Ideally, the simulations should follow this logic in order to accurately represent real-world circumstances.

When starting to create the visualisations they were divided up according to the two separate systems. This method was adapted in both VC and TPS. The models were created in different iterations, beginning with the most basic functions to integrating more functions in different stages, staring from the simplest to the more complicated functions. While creating these versions of the system, the placement of the different components were done randomly. The components were placed in their proper places in the last iteration, as the first steps were solely focused on

implementing the required functions the system should have.

3.4.3 Model Building in Visual Components

To easier present the steps of each model, the model building is divided up into different stages of complexity. The work done was a much more iterative process than just doing each stage at once, but these stages are the main steps.

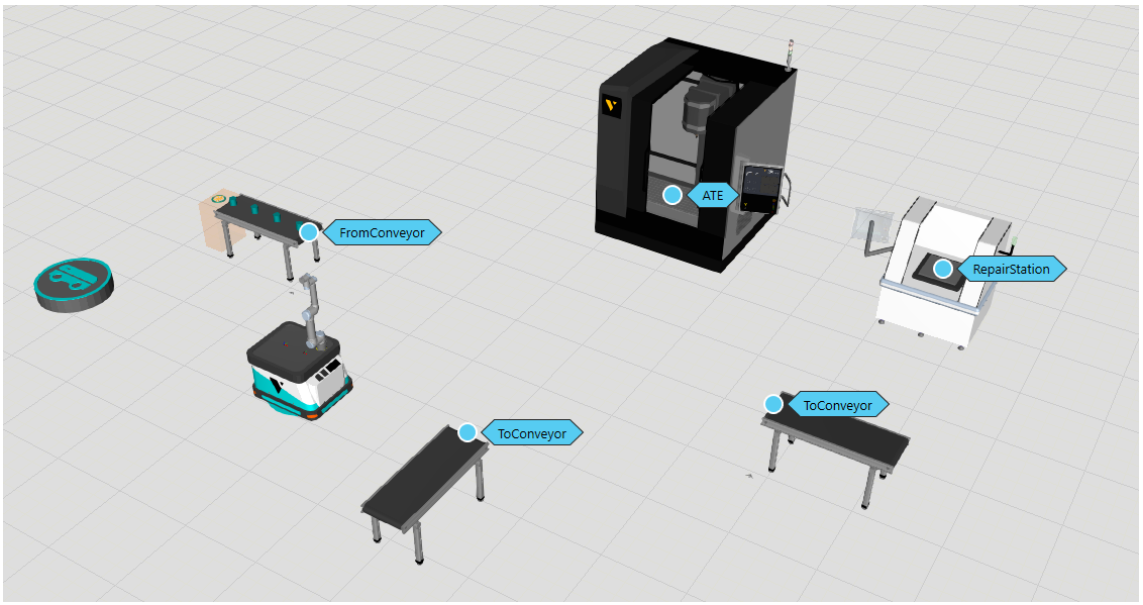
Model Building Stage 1

Following the suggested approach by Banks, 2000, the different models were developed in an iterative manner. Starting with the basic functions and adding more functions, progressively increasing the complexity to achieve the desired end result. With this approach, each stage of the model could be fully functional before moving on to the next, giving the models a stable base logic.

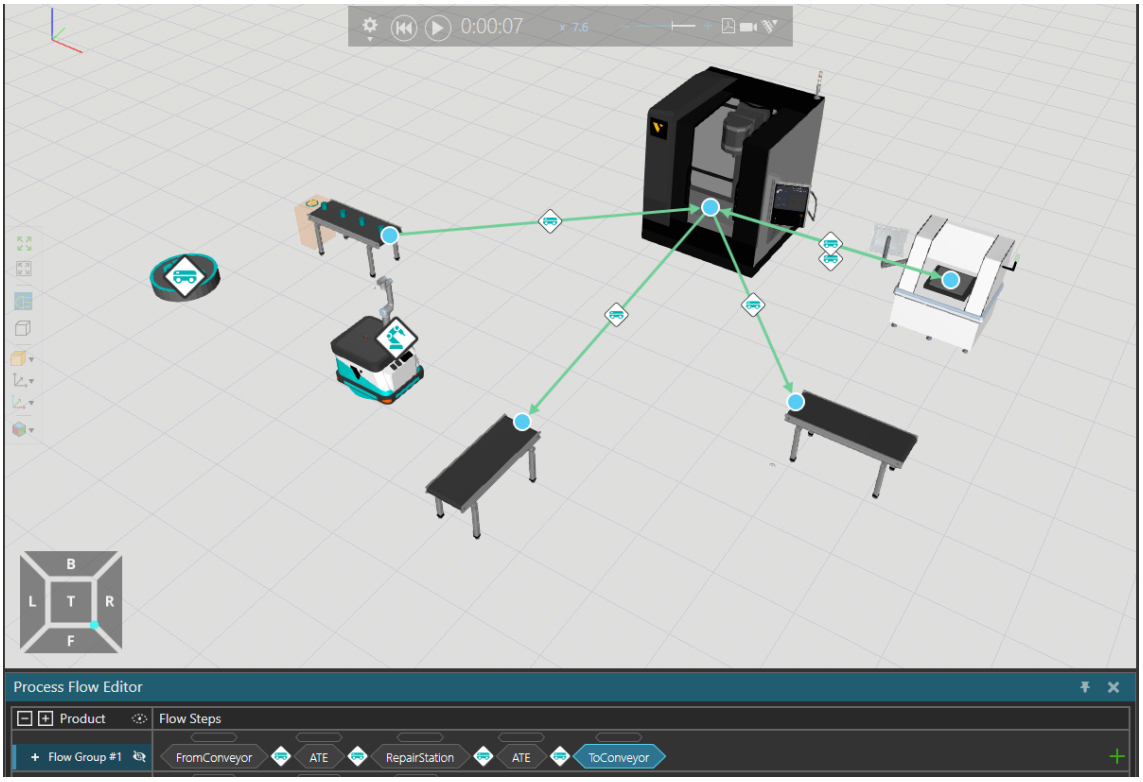
Firstly, a very basic model was created in VC as can be seen in figure 3.5a. The system contains a process feeder connected to a conveyor which has a "From Conveyor Process" node connected to it allowing the Mobile Robot resource to be able to pick up products from the conveyor. A robotic arm has also been placed on the robot to assist with loading and unloading products onto the robot. The specific arm used from the programs component library is called "UR5". The model also contains a "Parametric Vertical Mill" which is used to simulate the ATE machine. Furthermore, there is a very basic repair loop where a "Generic Work Station" represents a repair station. There are also two separate conveyors with a "To Conveyor Process" node on each which represent the locations for passed and failed products.

In VC, flow steps and flow connections are used to make the program aware of which paths the products flow in. In the following figure 3.5b, the process flow between the products as well as the process flow editor are shown. To further dictate where a product should go, the flow layout has a process flow editor. Each column in the process flow editor represents a flow step. The ATEs process executor access these flow steps and decides the next step for the product depending on if it passed or failed during the ATE process.

In this stage of the model all connections are done with the Mobile Robot Resource as the transporter.



(a) Process layout



(b) Flow layout

Figure 3.5: Visual Components Stage 1

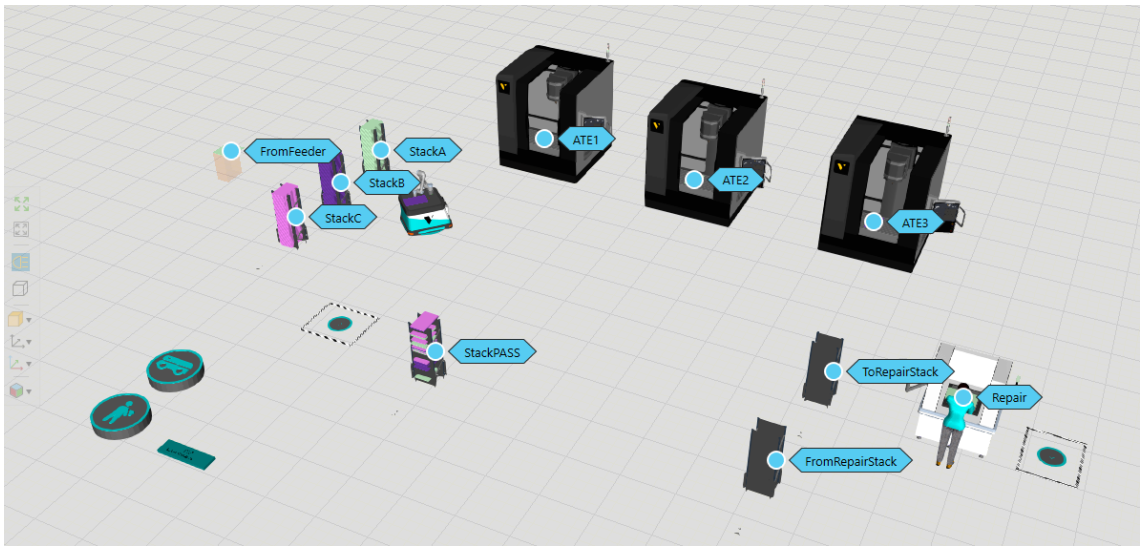
Model Building Stage 2

The next stage of the model building consisted of expanding the system to function with one robot, three ATE's, three different products and a human transporter in the repair loop. The following figure 3.6a shows the set-up for this stage. The visual aspect of the shelves have also been changed to better mimic the real world. Idle stations have been added for the robot and the worker. Figure 3.6b shows the process flows within the new system as well as which transporters are responsible for each transport segment.

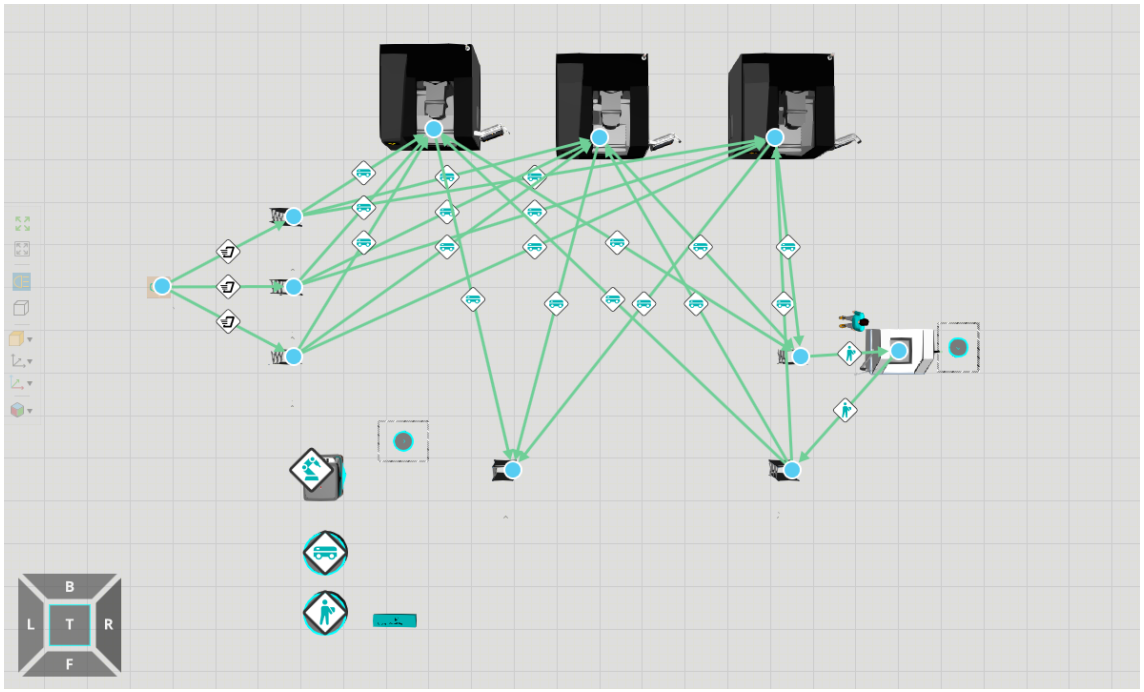
In this stage a process node, named 'Mission Controller', was implemented to tell the ATEs when they should request products. Since there was no way to directly control the robot, and all the ATEs process executors run at the same time, there was an issue that they all requested the same product at the same time, leading to the number of fixtures going into the negative. For the Mission Controller to work correctly, each product had a global counter variable, to ensure that the Mission Controller wouldn't assign a product that didn't exist to an ATE. With the Mission Controller an order of priority was made, making it impossible for two ATEs to request products at the exact same time. Furthermore, the function that if an ATE already has the appropriate fixture, the product would prioritise going to that same ATE again was added. This was to reduce excessive changeovers.

When implementing additional products, each products were assigned different properties since the products have different processing times, required fixtures and yield. By storing these variables in the products themselves, the machines can access these numbers and calculate the appropriate pass-rate and processing time. The ATEs additionally use product properties to determine whether a delay that represents a changeover is required.

In addition to this, the repair loop was configured to repair in batches as this function had been requested while discussing with employees. Once a certain number of products have failed, that amount of products gets repaired and subsequently return to the production flow.



(a) Process layout



(b) Flow layout

Figure 3.6: Visual Components Stage 2

Model Building Stage 3

The last stage of the model building was to implement all of the required functions in an environment with two cobots, six ATEs and all current products. In VC a pre-made model of the production environment was available, and the final simulation model was built in this model to properly visualise and reflect the real-world production environment.

Two versions of the final model was created. One with the initial layout requested by the company and another of the additional layout that was suggested at a later date.

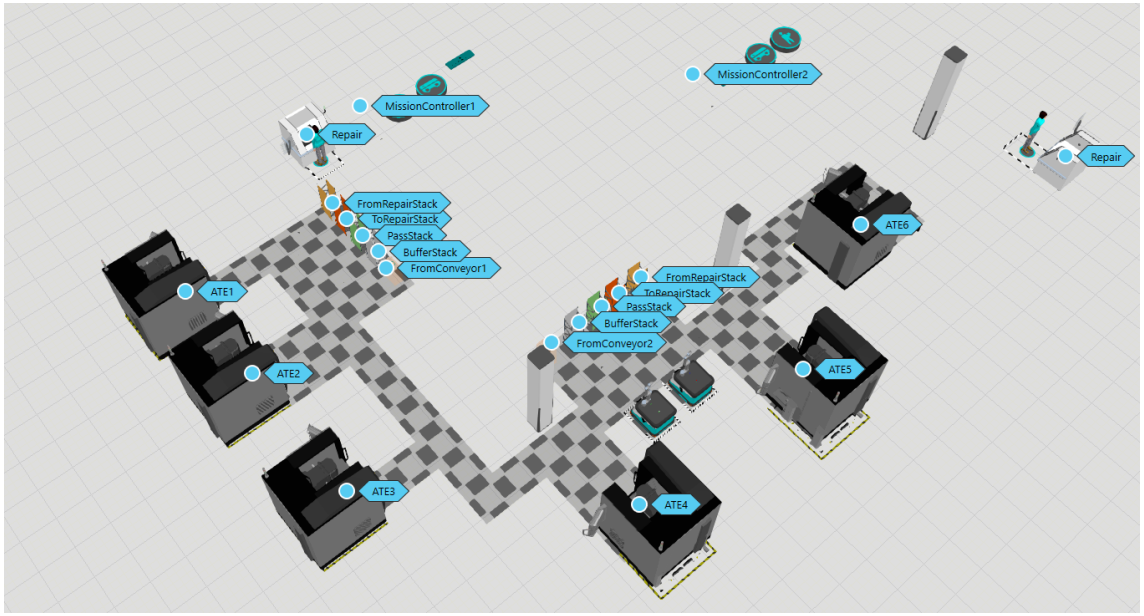
Prior to the final models, all the necessary products were implemented and tested in the stage 2 models, to ensure that all required properties functioned as intended.

The finished ProcessExecutor codes for each station can be found in the appendix under A.1.

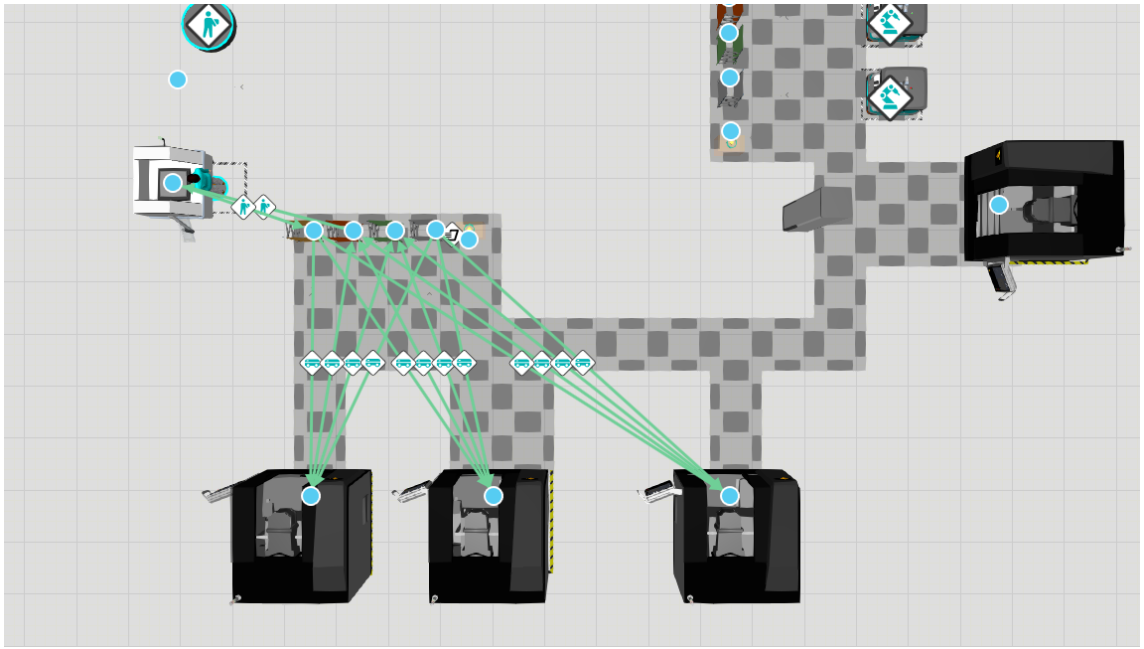
Stage 3 Layout 1

For the initial layout, two versions of stage 2 models were merged, one with all type 1 products and one with all type 2 products. Two Mission Controllers were run to simultaneously assign permissions for the ATE product requests for type 1 and 2 products. Products of type 1 and type 2 were divided into two different flow groups, ensuring that type 1 could only flow through ATE 1, 2 and 3 while type 2 products flow through ATE 4, 5 and 6. An additional mobile robot resource was added and each robot was assigned to one of the two flow groups, with the use of an additional mobile robot controller.

As shown in figure 3.7, pathways were added and connected to the mobile robot controller. These pathways ensure that the cobots avoid areas that may be occupied by other objects in the production.



(a) Process layout



(b) Flow layout

Figure 3.7: Visual Components Stage 3, Layout 1

The complete process flow editor for layout 1 is shown below in figure 3.8.

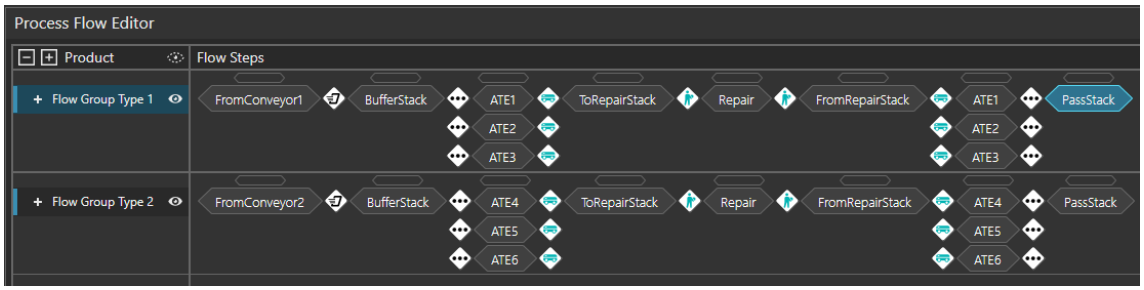


Figure 3.8: Visual Components Stage 3, Layout 1 - flow layout

Stage 3 Layout 2

The building of layout 2 was very similar to layout 1 but more complex. Every ATE has it's own feeder, meaning that there needed to be counters for each product at each ATE. An additional counter for products that come from the repair station was added as well.

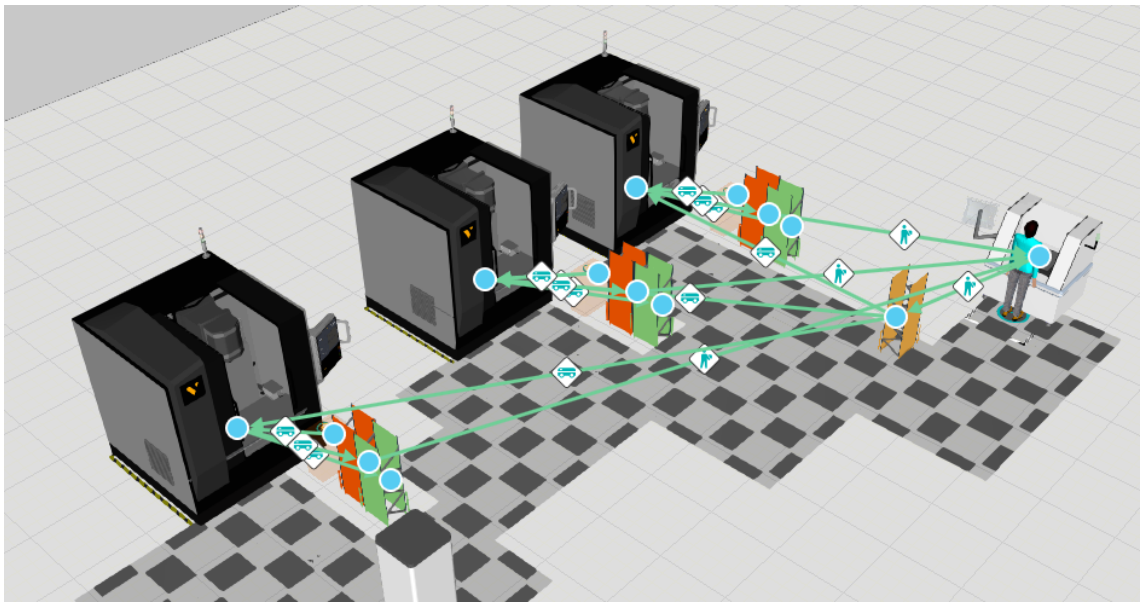
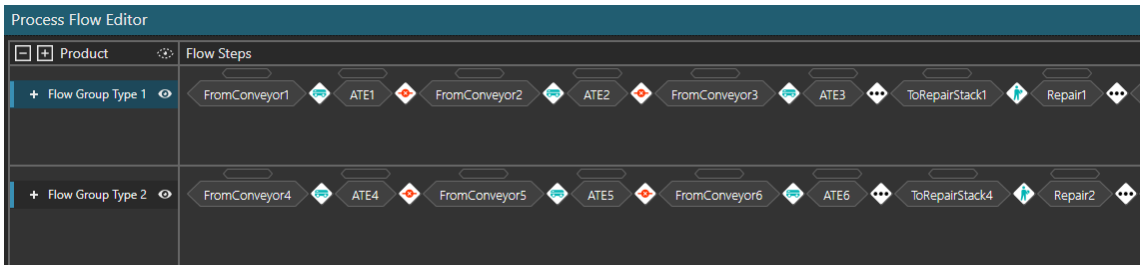
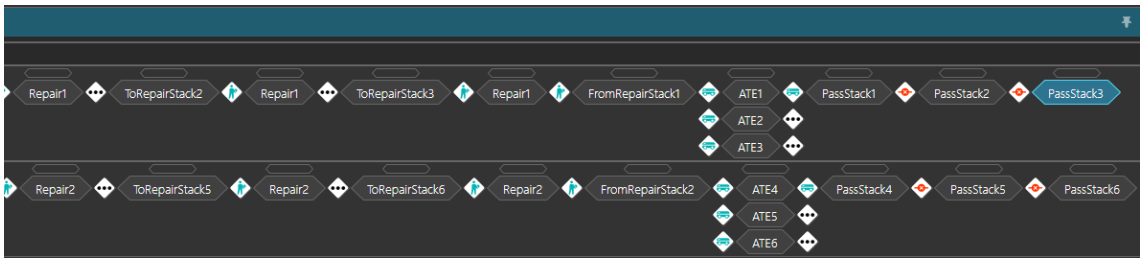


Figure 3.9: Visual Components Stage 3, Layout 2

In addition to this, the flow groups were modified to avoid products losing their flow steps. Since each location for a product is specific to which ATE the product is at, the flow group had to be restructured so that each stack location became it's own step, with each ATE selecting the next step for the product that corresponds to the ATE. The modified flow group is shown in figure 3.10.



(a) Visual Components Stage 3, Layout 2 - flow layout part 1



(b) Visual Components Stage 3, Layout 2 - flow layout part 2

Figure 3.10: Visual Components Stage 3, Layout 2 - flow layout

In this layout, the two robots share the same mobile transport controller as they are supposed to share the working areas.

3.4.4 Model Building in Tecnomatix Plant Simulation

This section describes the different stages of model building that were done in the program TPS. Similarly to VC, the stages represent different iterations and complexities of the models.

Model Building Stage 1

The first stage of the model building in TPS, shown by figure 3.11 is very similar to that in VC. A basic structure was created in order to get a functioning flow of products from a source to an ATE. In this stage a built in function was used (percentage in the "exit" tab in the stations settings), in order to simulate the yield, making sure 80% of products passed and 20% went to fail. The repair loop also goes back into the system and passed products pass on to the drain. At this stage the transporters have not yet been incorporated so connectors (the arrows) have been used instead.

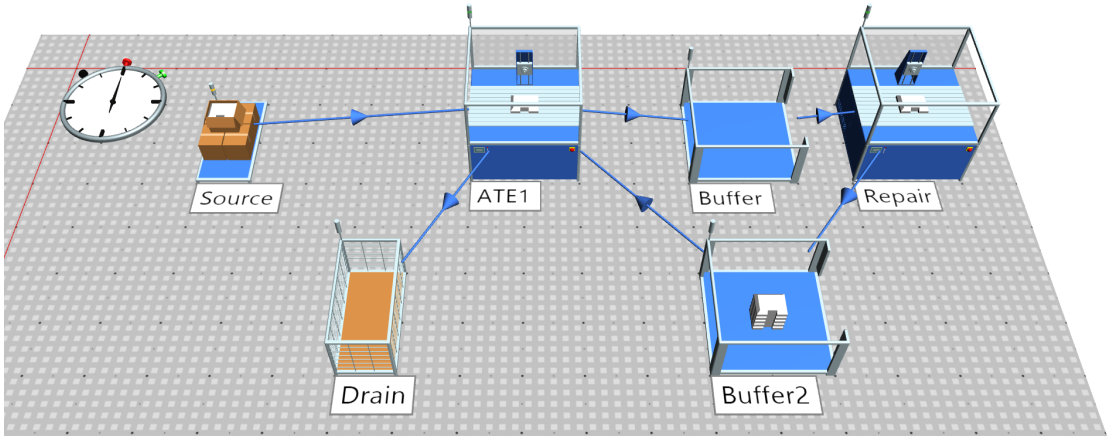


Figure 3.11: Tecnomatix Plant Simulation Stage 1

Model building Stage 2

Further on, once basic functions were eventually integrated. A more complete system was formed including three ATEs, three sources and a mobile robot transporter as well as a human transporter. In TPS, the component used to represent the cobots is the AGV. The AGVPool spawns the mobile robot and the "Init" method defines how the robot moves and the priority of movement.

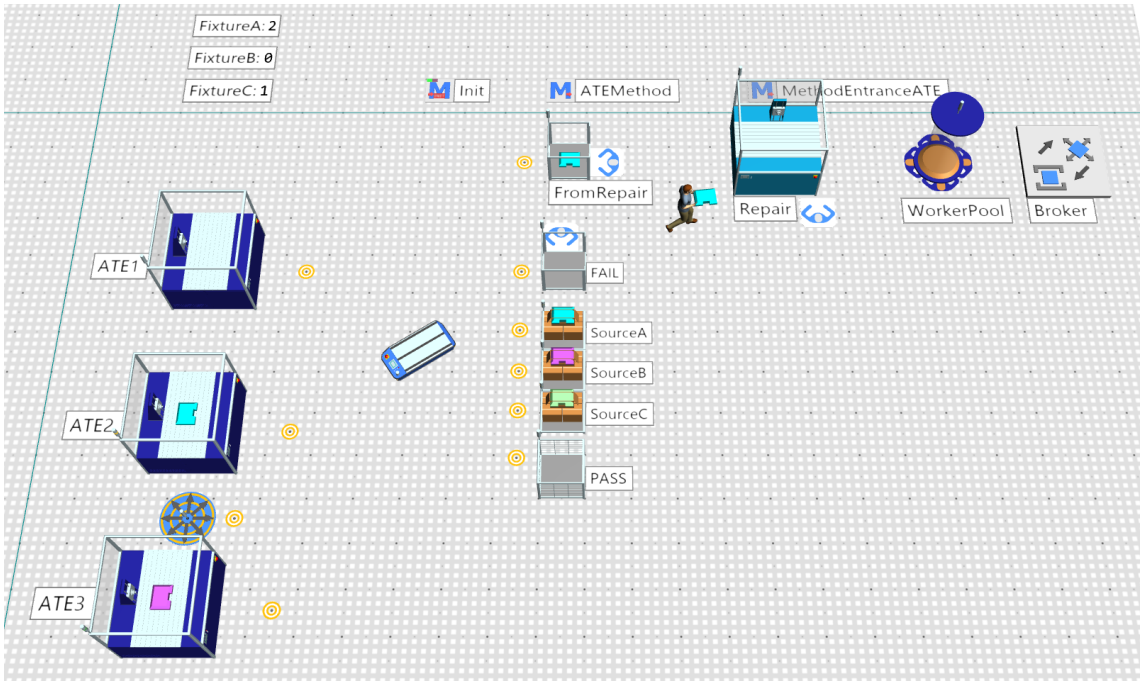


Figure 3.12: Tecnomatix Plant Simulation Stage 2

For the repair loop there is a WorkerPool and a broker which run the human transporter. The human picks up products from the "FAIL" buffer, delivers them to the "repair" station and when done places them in "FromRepair" buffer. Here the AGV robot takes over again and incorporates them back into the system. Since the different products have different yield, and changeover time the "MethodEntranceATE"

method is needed. This method checks the properties of the incoming product and adapts accordingly.

The "Init" function controls the robot and is responsible for assigning tasks to the AGV. As can be seen in figure 3.12, Markers have also been used instead of tracks to allow the robot to move freely.

This model was built according to availability of fixtures and makes sure products only run if there is a fixture available. If a machine recently has used a certain fixtures, that machine is prioritised to take on the same consecutive product in order to avoid unnecessary changeover times. The SimTalk methods also have a built in order of priority.

Model building stage 3

Stage 3 - Layout 1

For the first layout, the AGV transporters operate independently within their respective areas. Therefore, Layout 1 was created by duplicating the system developed in Stage 2. The two areas are combined to represent both areas simultaneously, forming the complete Layout 1. When duplicating the system, two init methods where formed. Since the two areas have different products the different areas have certain restrictions such as "product H can only be tested in ATE5 and ATE6", the two methods were kept separately, renamed as "Init_X" (see appendix A.2.1) and "Init_Y" (see appendix A.2.2).

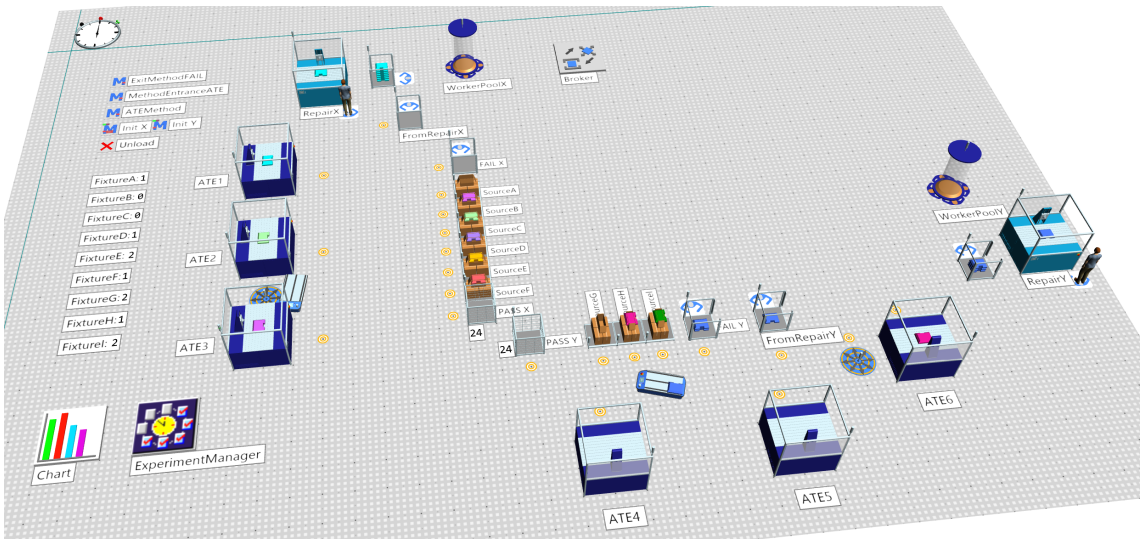


Figure 3.13: Tecnomatix Plant Simulation Stage 3, Layout 1

When the system was duplicated, adjustments were required to ensure that both areas could operate simultaneously. To guarantee that the initialization methods for the two separate AGVs start at the same time, each init method was connected directly to its respective AGVPool. With this solution both AGV codes start at the same time. Otherwise, since each method contains a loop, if one method is called

upon first, the other one will be ignored.

Stage 3 - Layout 2

For Layout 2 more changes had to be made. In the second layout, both AGVs operate in the same area and share tasks, which meant that the task assignment logic needed to be updated. The initial Init method was modified (see Appendix A.3.1 so that it only assigns tasks to AGVs. Then an additional method was added "ControlAGV" (see Appendix A.3.2) which includes the different tasks and paths for the robots. The tasks are also "reserved" by the AGV that is assigned so that both AGVs cant be assigned to the same task.

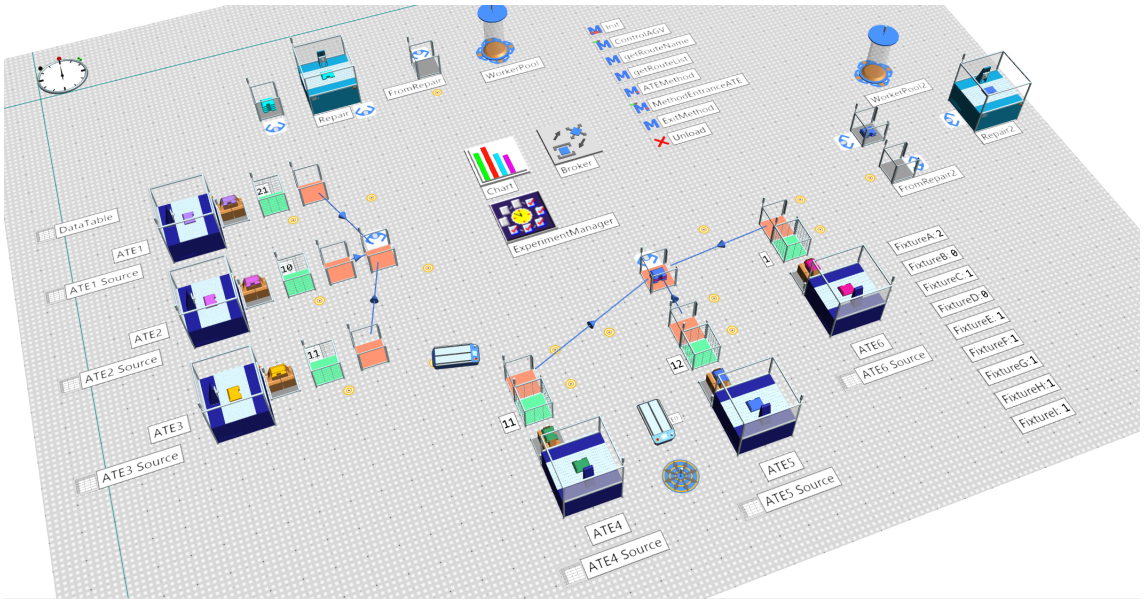


Figure 3.14: Tecnomatix Plant Simulation Stage 3, Layout 2

Regarding pathways, big adjustments needed to be made in order to stop the AGVs from driving shortest possible route between tasks, and thereby crossing straight through restricted areas such as workspaces. A data table was added (see appendix A.3.10) that includes all possible paths for the AGVs. Then by calling on the methods "getRoute" (see appendix A.3.3) the AGV could retrieve a path from the data table that takes it to its next destination by following a set of markers.

3.5 Experimental Design

After finalising the operational DES models in both VC and TPS, the simulations were used to simulate different scenarios. All stages have been documented in case the simulations need to be developed further or in case of modifications (Banks,

2000).

When everything was functioning correctly in Layout 1 and 2, the statistics from both programs was analysed. In TPS the "Experimental Manager" was added together as well as the "Chart" function in order to record statistics from the simulation while running 20 iterations for each scenario. In VC, statistics for all scenarios could still be collected, although only one simulation run was performed for each scenario.

3.6 Verification and Validation

Verification refers to making sure the final model is in accordance with the original design. In order to verify the simulation models the final models were compared to initial visualisation as well as initial desired logic and as much as possible has been implemented.

The validation for the final simulation models was more difficult since there is no real-world system for comparison. Therefore, experts working full-time with the production area, the different layouts and the cobot implementation were consulted. Since these employees are responsible for developing the future real-world system that the models are based on, this approach provided the closest possible validation against a real production environment.

3.7 Software evaluation method and criteria

Similar to Ivanov et al., 2025 and Ciauşu-Sliwa et al., 2025, this project evaluated the programs by comparing it based on different criteria, to limit the scope of the program evaluation.

As mentioned in Ivanov et al., 2025, their criteria for evaluating simulation tools could be divided into several categories, including technical, analytical, organisational, economic, and educational aspects. In this project, five different criteria were selected with inspiration from Ivanov et al., 2025.

The selection is based on the functionalities of the programs and the notable challenges and differences between simulation software encountered during the project. The criteria are mainly picked from the technical category, as the technical aspects of the two simulation tools were the main focus of this research.

These criteria were being evaluated throughout the model building in both programs. Since both models are built in a similar way, after each stage in the model building the advantages and disadvantages of the programs in each criterion could be evaluated. During the model building, notes regarding differences in each program were taken and then gone through to evaluate each criterion. The criteria chosen

are shown in table 3.1.

	Technical Category	Educational Category
Criteria	• User interface • Simulation performance • Visualisation • Scalability	• Academic support & training

Table 3.1: Evaluation criteria chosen from Ivanov et al., 2025

From the Ivanov et al., 2025's technical category (see table 3.1) the following criteria were chosen: **user interface**, **simulation performance**, **visualisation** and **scalability**. These criteria were deemed relevant to this research.

The user interface criterion was included to evaluate how intuitive and efficient each program was to work with during the process of building the models. This criterion was evaluated through personal observations during the entirety of the model building.

Simulation performance was considered important to assess how the different softwares handles the specific demands of the models. This criterion was evaluated by running the simulations and observing how well each program handled the implemented logic.

The visualisation criterion was chosen to evaluate how closely the program is able to represent the 3D-environment of the real world, since the company currently uses VC for the visualisation aspect of it. The visualisation criterion was evaluated through comparing the end result of both models to the real life production area, and seeing which one resembles the area more closely.

Lastly, scalability was included from the technical category to evaluate how well the software were equipped for increasing the complexity of the models, since the models built are fairly complex. Since the model building was an iterative process, constant upscaling of the models was performed throughout the project. Comparing how long each iteration took in both programs and how well the logic could be applied to bigger models set the base for this evaluation.

The last criterion, **academic support & training** (see table 3.1) was picked from the educational category in Ivanov et al., 2025's paper. This criterion was picked due to lacking prior knowledge in one of the simulation software, and not complete knowledge in the other considering the different functions needed for the simulation model. Because of the need for academic support and training in both programs, comparing the two software on this criterion was deemed important in order to evaluate the accessibility of learning resources, available documentation and support for developing and troubleshooting the simulation model.

The academic support and training criterion was evaluated every time there was a lack of knowledge, and how accessible information in regards to this was found.

4
Results

4.1 Software Functionality Evaluation

The simulation programs evaluated in this thesis represent two different categories. VC is a 3D Visualisation program, while TPS is an Advanced DES Program. This section presents an evaluation of the program categories based on the selected criteria.

4.1.1 Scalability

Throughout the model building scalability was continuously tested in both programs. When developing the different layouts, the initial structure only included one of each component, after which additional components were gradually added.

When scaling in VC, some issues were encountered that made the process more time consuming and complicated. Since each machine has a separate process executor, all programming for each component needs to be modified individually. Therefore, when scaling the system and making changes to a machine, the process becomes time-consuming, as the same change must be applied to each component. When it comes to adding additional mobile robots, VC handled the transition from one to two robots well and the modifications needed were very intuitive.

In TPS, scalability was easier from a programming perspective. With built-in Python functionality that allows code to be shared across multiple components, such as all ATEs, adding or removing components was fairly effortless. As for the mobile robots adding another one in TPS required some restructuring in the code but once the system is build for multiple mobile robots, additional ones can be added with ease.

While adding more mobile robots is well supported in TPS, adding more components in the models means that the pathing of the cobots requires changes. In comparison to VC more work has to be done for the cobots to move correctly if the cobots are using marker to traverse the production area.

4.1.2 Simulation performance

The simulation performance regarding execution speed was good in both programs throughout the model building. However, once the entire structure of the production

area was added to layout three in VC, some lagging occurred when the simulation was run.

Regarding the evaluation of the overall simulation performance in reference to the intended purpose, as discussed by Hora and Campos (2015), TPS could easily implement all the desired logic. Although, programming the robots in order to make them follow certain paths as well as controlling several robots in the same area was the part of the simulation that caused the most setbacks.

Regarding the simulation performance in VC, the functionality was only marginally sufficient for implementing the intended logic. In this program, parts of the logic had to be restructured in order to adapt to the built-in functions and limitations of the software. The specific flows of the products within the program are one of the cause of the limitations. The process flow editor also needed to be structured specifically for the purpose of the simulation, which in some cases resulted in an illogical flow order for the simulation to function correctly. Furthermore, the robots themselves could not be directly programmed as they relied on their built-in programming. As a result, instead of programming the robots to pick up finished products according to the desired logic, the robot is dependent on the ATE code which announces that a product is finished and ready for the next step in the process flow.

4.1.3 Visualisation

The visualisation in VC is very good since the components and machines all look very realistic. There is an extensive and easily accessible library with many different machines and components that can represent a real production process. This allows the user to personalise the simulation significantly which also improves visualisation and therefore, when using this program, almost no additional time had to be spent on making the simulation resemble real life.

In TPS, the initial structure of the components is very basic. For example, machines and workstations appear in one format in their default form. Although there are possibilities to modify the 3D attributes, the initial visualisation is primarily functional. Making the simulation more representative of real life is therefore time-consuming.

4.1.4 User interface

Concerning the user interface, VC performs very well, especially for users with no prior experience. The interface is clean, and tools are easy to find. The programming in the ProcessExecutors is also easy to use, as it is based on a “drag-and-drop” structure. The connection between flow groups and process flow can initially be somewhat confusing as in why both are required, although editing them is very user-friendly.

TPS requires some prior knowledge or an introductory course to fully understand

the program and its functions. However, once a basic understanding is achieved, the program becomes more intuitive. The programming is done in SimTalk, which is similar to Python. However, for users without prior programming experience, the program can be difficult to use initially.

4.1.5 Academic Support & Training

Overall, both programs have good support and training available. However, with VC course material is easier to access through web searches without having to subscribe to a access through a specific login.

The available material from VC is easily accessible online and includes courses, video tutorials and documentation. There was a good amount of beginner friendly material, allowing for a good introduction to the program. The Visual Components Academy page also provides a forum where users can get help directly from VC as well as from each other.

For TPS the training is mainly provided through Siemens Xceleraor academy, which includes extensive online learning material in the form of instructed courses, and certifications. Additional tutorials, university material and third party video tutorials are also available. Information is about how each tool is used can also be found within the program interface directly. Finally, similar to VC there is also an online forum.

4.2 Simulation Results

In order to compare the different programs, as many aspects as possible were kept identical when building the different layouts. For example, although it would have been ideal to include a warm-up time before recording the statistics, no warm-up time was added in either simulation because it did not function as intended in one of the programs.

Since no batch sizes were given, fabricated ones were made. As stated in section 3.1, several aspects were taken into account when deciding the amount of each product. The batch sizes used for the simulations are shown in table 4.1. In layout 1, a feeder will produce 26 product A products, 26 product B, 16 product C and so on. The cobots are able to pick any of the products until the batch runs out. In layout 2, at ATE 1 for example, 14 of product A will be introduced to the system, and then 16 of product C. That is how the batches were implemented to get simulation data from both software.

Bacthes of products	Layout 1	Layout 2 ATE1	Layout 2 ATE2	Layout 2 ATE3	Layout 2 ATE4	Layout 2 ATE5	Layout 2 ATE6
A	26		14	12	-	-	-
B	26	26	-	-	-	-	-
C	16	-	16	-	-	-	-
D	12	12	-	-	-	-	-
E	26	13	-	13	-	-	-
F	10		-	10	-	-	-
-	-	-	-	-	-	-	-
G	30	-	-	-	15	15	-
H	10	-	-	-	-	5	5
I	20	-	-	-	15	-	15

Table 4.1: Batch sizes in Layout 1 and 2

The exact numbers from the simulation data results can be found in the appendix, under A.4.

4.2.1 Tecnomatix Plant Simulation Results

Here, the simulation results from the two layouts in TPS are shown. When possible, the built in Experimental Manager and chart functions were used. For some data, the method EndSim was used in order to print data that was then manually entered into tables and graphs after compiling it in a Google Sheets file.

When the Experimental manager was used 20 iterations where performed for each scenario. In order to evaluate the variation between these runs the standard deviation can be observed for each value output in the table of the exact data extracted from the Experimental Manager report (see appendix A.5).

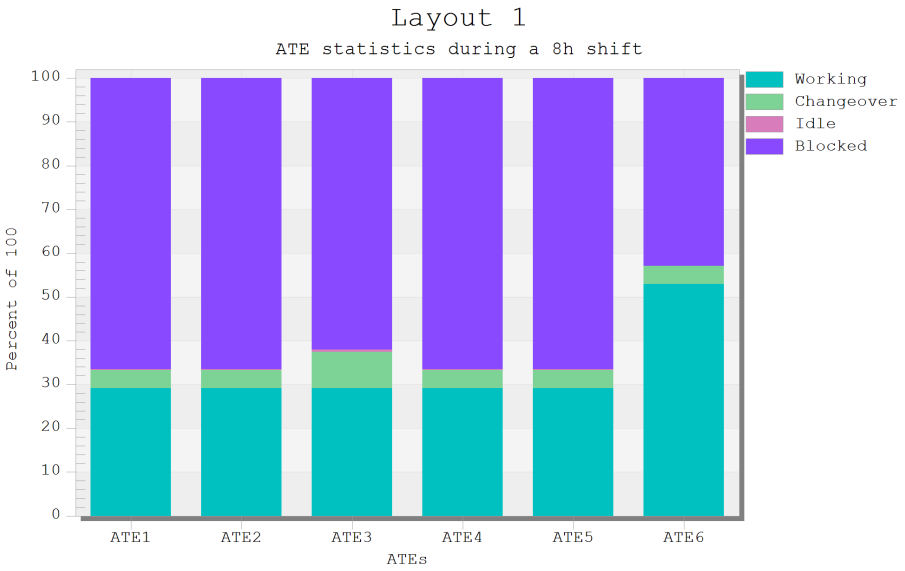
ATE utilisation

For TPS, the Experiment Manager was used to record the proportion of the total simulation time during which the machine was working, setting up, idle, blocked, or in a changeover. For each layout 2 experiments were done, each with 20 iterations in order to get the average result. Both experiments were done with a cycle time of 24 hours, however, in Experiment 1 the AGVs were only active for 8 hours while in Experiment 2 the AGVs were active during the full 24 hours. This is in order to simulate a normal working day and then compare ATE utilisation with 8 hours working time compared to 24 hours working time. The chart function was used to display the results graphically (the specific percentages can be found in the appendix A.4.2).

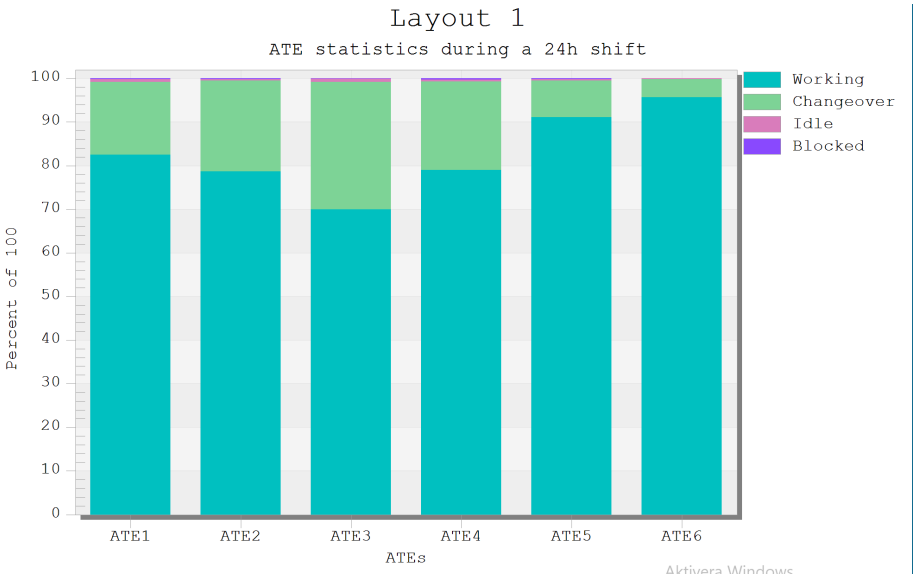
Since there was not enough real data available and the batch sizes had to be fabricated, the different layouts were compared to an environment where there are no cobots involved. To achieve this, the simulation was run with the cobots and repair station operating for 8 hours, after which the simulation continued to run without any further product transportation by the cobots.

The following figures show the results of the ATE utilisation presented as percentage portions. Figure 4.1 shows the graph for the 8 hour and 24 hour shift for Layout 1

and figure 4.2 shows corresponding data for Layout 2. From the 8h shifts it is clear that not having the AGVs working around the clock leads to a significant percentage of blocked time. The 24h shifts shows that having the cobots working around the clock significantly increases ATE utilisation from around 33% to roughly 84% (see appendix A.4.2).

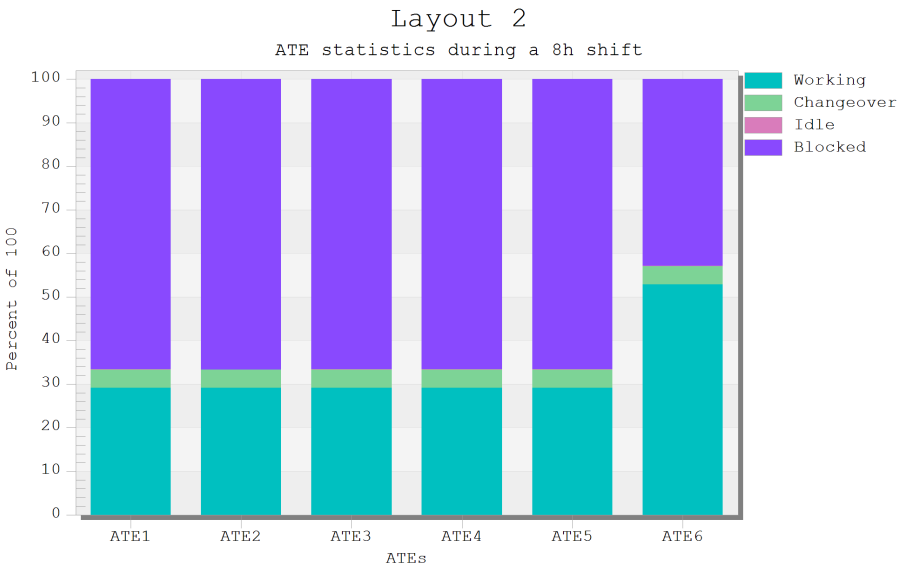


(a)

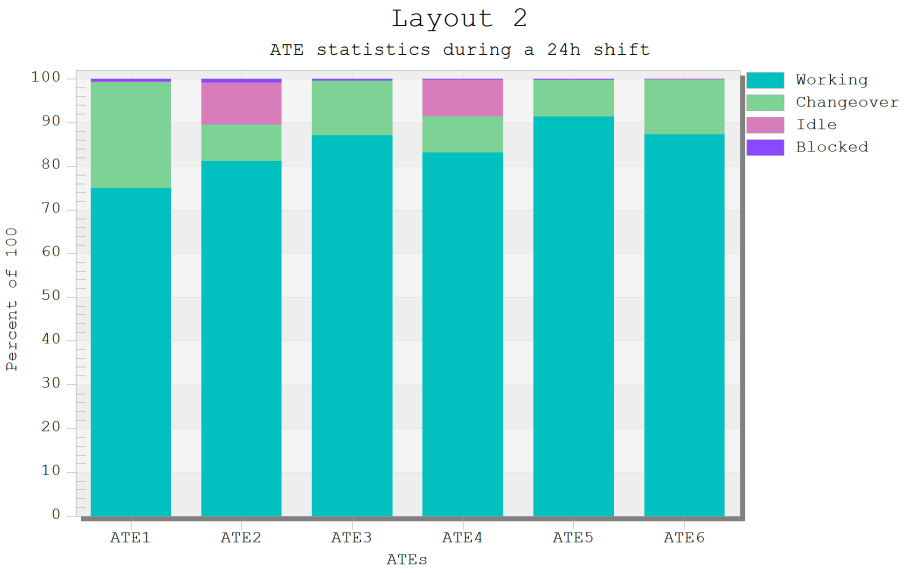


(b)

Figure 4.1: Comparison of Layout 1 for 8h and 24h work shifts in TPS



(a)



(b)

Figure 4.2: Comparison of Layout 2 for 8h and 24h work shifts in TPS

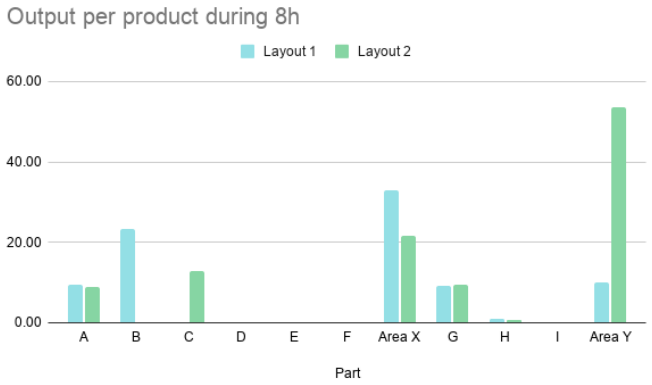
Product Output

Figure 4.3 shows the total product output per product in the 8h simulation as well as the 24h simulation (see exact data in appendix A.4.2). Area X refers to the total output produced by ATE 1, ATE 2 and ATE 3, while Area Y is the total output produced by ATE 4,ATE 5 and ATE 6.

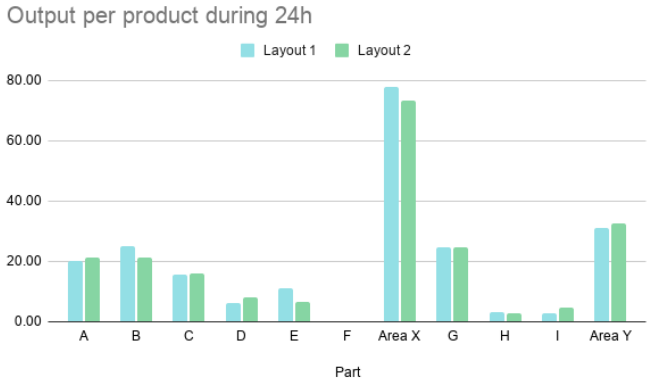
During the 8-hour shift scenario, relatively few products have enough time to complete processing due to the long processing times of the ATEs. However, in the 24-hour scenario, at least one unit of every product type is produced, except for product F.

When comparing the product output between the two layouts, the numbers differ slightly in the 8-hour scenario, while the results for the 24-hour scenario are very similar. For the 8-hour shift, Layout 1 produces 11.1 more products than Layout 2. In contrast, for the 24-hour shift, Layout 1 produces only 3 more products than Layout 2 (Figure A.4.2).

In order to collect the data for the product output the method "Counting"(see appendix A.2.7) was used to count how many of each product type entered the PASS drain during the simulation. At the end of the simulation, the method "EndSim"(see appendix A.2.8) was used to print the results.



(a)



(b)

Figure 4.3: Comparison of Layout 1 and 2 for product output during 8h and 24h work shifts in TPS

Distance travelled

The distance travelled by each cobot in the two layouts differs slightly. In Layout 1 each robot has its own assigned area, Area X and Area Y. In Layout 2, both cobots work together and are both able to tend to all machines and are not restricted to their own area.The 8h simulation was used to represent worker operation schedules. Therefore, this diagram, figure 4.4 only includes data from the 24h simulation, as it specifically focuses on the cobots and the difference in distance travelled between the layouts. According to this diagram, the cobot travels further in Layout 2 with a difference of 0.42km (for exact data see table 4.2).

The distance data was collected by using the "EndSim" method (see appendix A.2.8) and calling on the built in function "statTraveledDistance" of each cobot.

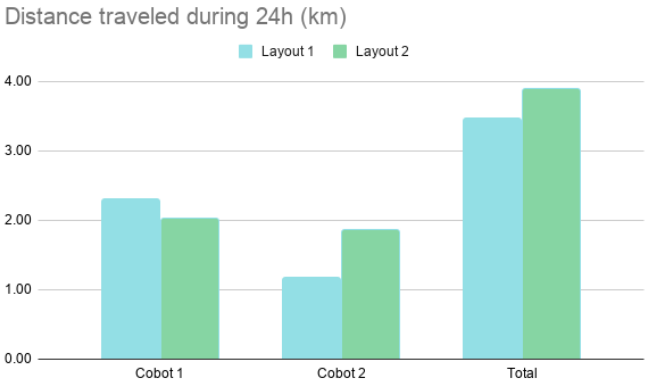


Figure 4.4: The distance in km travelled by each cobot during a 24h shift in TPS

Working Time

This shows the number of hours that the ATEs are working during a 24h shift. The figure 4.5 also compares the working time between the different Layouts. The total working time is very similar between Layout 1 and 2, only differing 0.1 hours (see exact values in appendix A.4.2). As can be observed in figure 4.5, the working time for each ATE was also very similar when comparing the two layouts.

The data for working time for each ATE was collected using the Experimental Manager (Appendix A.5).

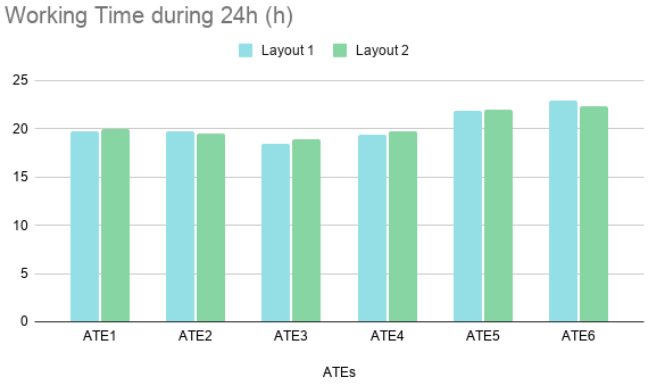


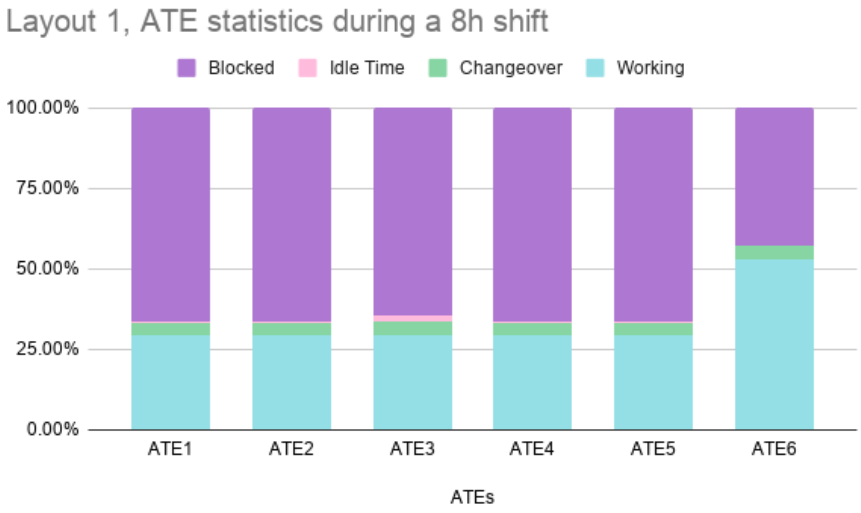
Figure 4.5: Comparison between Layout 1 and 2 for the total working time per ATE in TPS

4.2.2 Visual Components Results

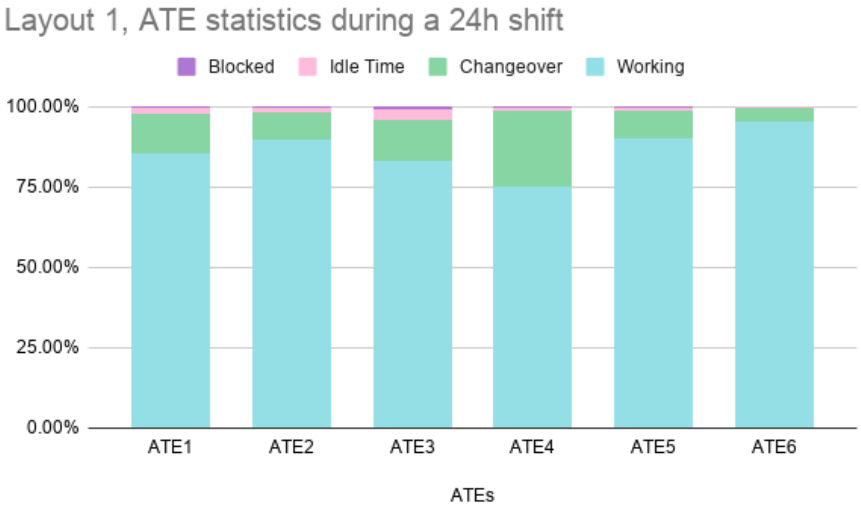
Here, the simulation results from the two layouts in VC are shown. To get the data, the 'Statistics' option was used. Each state of the ATEs had to be manually entered to get the corresponding data. The graphs were not automatically organised in a comprehensive manner, so the data was compiled in a Google Sheets file, and the graphs were generated in Google Sheets for a better visualisation of the data.

ATE Utilisation

Just as in TPS, to get the data for comparison between an 8 hour shift and a 24 hour shift, the simulation was run for 24 hours with the cobots and humans only working 8 of those hours.



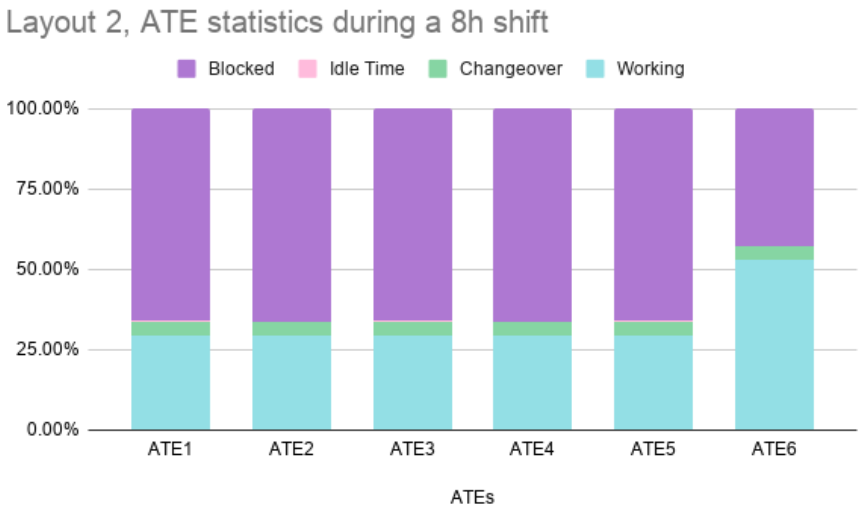
(a)



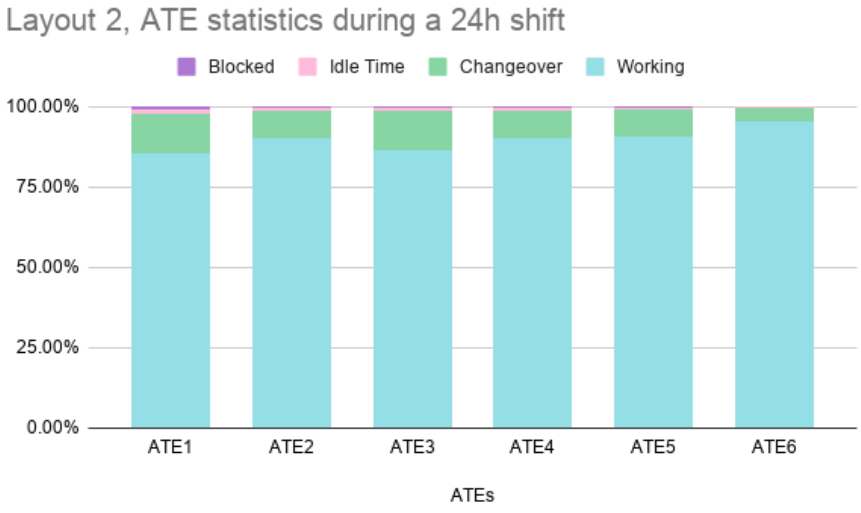
(b)

Figure 4.6: Comparison of Layout 1 for 8h and 24h work shifts in VC

The results from Layout 2 can be seen in Figure 4.7.



(a)



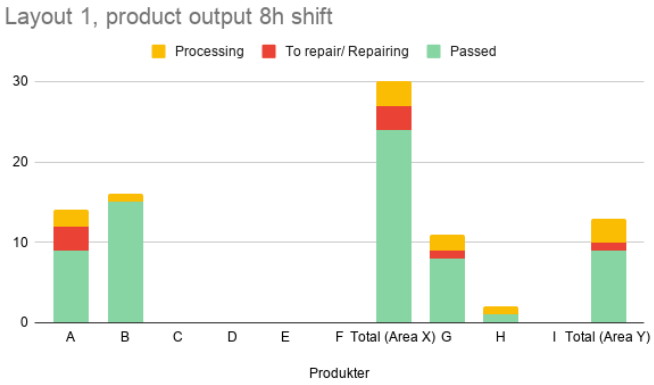
(b)

Figure 4.7: Comparison of Layout 2 for 8h and 24h work shifts in VC

The ATE utilisation results in VC show that, similarly to TPS, having the cobots working 24 hours compared to 8 hours during a 24 hour time span utilises the ATEs more. The working average per ATE is increased from approximately 33% to 87% in layout 1 and from 33% to 90% in layout 2.

Product output

The figures in 4.8 show the specific product outputs. Since each product takes different amounts of time to process, the total time required for the amount of products has been calculated as well, without regard to the changeover times.

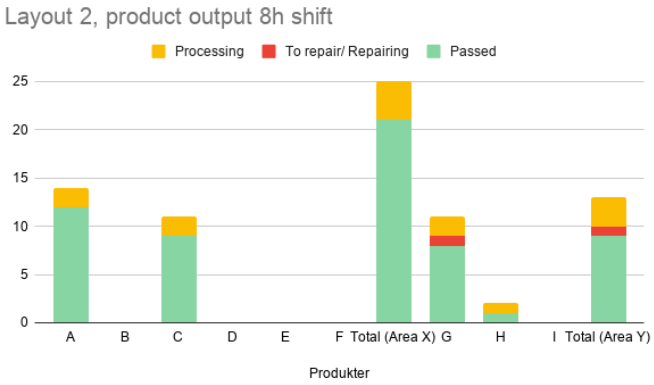


(a)

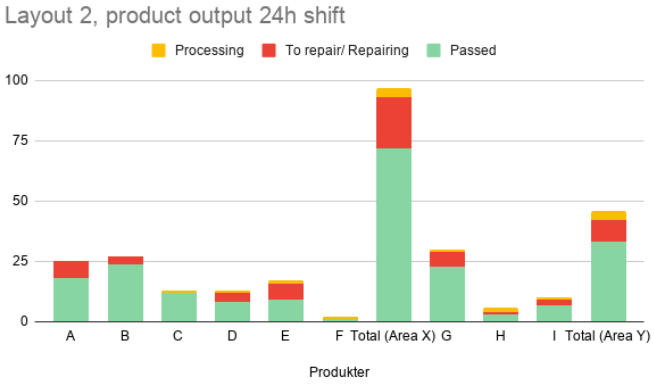
(b)

Figure 4.8: Comparison of product output in Layout 1 for 8h and 24h work shifts in VC

In layout 1, the total product time for a 24 hour shift was summed up to 115 hours and 53 minutes. In contrast, the total product time for an 8 hour shift was summed up to 31 hours and 6 minutes. Likewise to the simulation in TPS, during the 8 hour shift, not all products were processed as there was not enough time to get to the lesser prioritised products.



(a)



(b)

Figure 4.9: Comparison of product output in Layout 2 for 8h and 24h work shifts in VC

In layout 2, the total product time for the 24 hour shift was calculated to 116 hours and 7 minutes while the 8 hour shift added up to 31 hours and 51 minutes. However, these numbers does not take into account parts that have been repaired and gone back into the ATE. As shown in figure 4.9a, no B products were processed, compared to 4.8a where 15 of product B were finished. This is due to the placement of each product and layout 1 being able to prioritise product B while layout 2 did not have access to the product before the 8 hours were up.

Important to note with the product output is that it does not account for products that have been processed twice. So the calculated total product time for each product may be larger, since some products may have been processed more than once.

Cobot travel distance

Figure 4.10 shows how much each cobot travelled during the span of 24 hours in each layout. As can be seen in the figure, having the cobots being able to cover both

area X and Y leads to a more even work distribution, as well as reducing the total travel distance for the cobots by approximately 347 m.

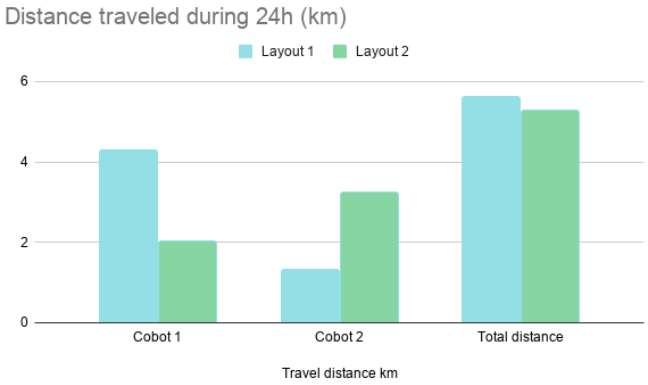


Figure 4.10: The distance in km travelled by each cobot during a 24h shift in VC

4.3 Comparing Layouts

This section contains an overview of the data collected in both programs.

4.3.1 Tecnomatix Plant Simulation Overview

The following table 4.2 compares the different layouts in both the 8h version as well as the 24h version. The working, changeover, idle and blocked average are the average values calculated from the percentages represented in table 4.1 and table 4.2.

	Layout 1		Layout 2	
Metric	**8h**	**24h**	**8h**	**24h**
Working Average	33.15%	84.81%	33.13%	84.88%
Changeover Average	4.86%	14.58%	4.17%	12.68%
Idle Average	0.20%	0.46%	0.04%	2.06%
Blocked Average	61.80%	0.15%	62.67%	0.39%
Travel Distance (km)	-	3.466	-	3.883
Total Passed Products	42.90	108.90	31.80	105.85
Working Time (h)	47.70	122.13	47.70	122.23

Table 4.2: Comparison between Layout 1 and 2 in TPS

The blocked average percentage clearly shows a significant increase in blocked time when the production was only run for 8h.

The distance travelled is also represented graphically in figure 4.4. The distance travelled in total by the cobots under the 24 hour shift is shorter in Layout 1.

The total number of passed products is shown in 4.3. This data is relevant when comparing the different simulation programs since both programs have been built up as similarly as possible with identical input data.

Finally, working time is included as it converts the working percentage into a value which is easier to interpret practically, showing the actual operating time in hours.

4.3.2 Visual Components Overview

Just as in TPS, a table has been created to represent the different data gathered from VC. The averages and working time have been calculated in the same way as in TPS as well. The compiled data from VC can be seen in table 4.3 below.

	Layout 1		Layout 2	
Metric	**8h**	**24h**	**8h**	**24h**
Working Average	33.23%	86.59%	33.37%	89.82%
Changeover Average	4.17%	11.55%	4.17%	9.04%
Idle Average	0.60%	1.51%	0.33%	0.77%
Blocked Average	62.00%	0.35%	62.13%	0.38%
Travel Distance (km)	-	5.614	-	5.267
Total Passed Products	33.00	114.00	30.00	105.00
Working Time (h)	47.85	124.69	48.05	129.34

Table 4.3: Comparison between Layout 1 and 2 in VC

A majority of the data reflect similar results as in TPS. The blocked percentage differs greatly between the 8 hour shifts and 24 hour shifts.

The working time is subsequently larger in the 24 hour shifts, with layout 2 having a slightly larger average working time for the ATEs.

The main difference between the programs is that in TPS, layout 1 had a total shorter travel distance than layout 2 due to differences in how the cobots are allowed to move in each layout. In VC layout 2 is the best layout when it comes to travel distances, and both cobots have the same area to move on in both layouts.

4.4 Comparing Simulation Program Data

The working, changeover, idle and blocked average percentages are relatively similar for both programs. At most the percentage differ with 4.94% between the results from the different programs.

Regarding the travel distance, there are some differences. In TPS, table 4.2, the distance travelled is longer in layout two by 0.42km, while in VC, figure 4.3, the distance travelled in Layout 1 is longer by 0.35km. The Total distance in VC is also 10,88km while in TPS it is only 7.35km during 24 hours.

Finally, the total number of passed products can be compared between programs since the input data for was identical for each. The batch sizes, processing times yields and fixture restrictions are all the same in both programs. From the results in table 4.2 and table4.3, TPS produces more products in every category.

4.5 Key Takeaways

Both simulation programs showed consistent results that running the simulation with the cobots active over 24 hours significantly reduced the bloked percentage for the ATEs compared to 8 hours. The two layouts in each program showed minor differences when it comes to production output, the difference was mainly in VC, where layout 2 performed slightly better. TPS showed a shorter travel distance in Layout 1, while Visual Components had a shorter travel distance in Layout 2. Overall, the shift duration had a much greater impact on system performance than the layout differences.

5
Discussion

5.1 Model Building

An effective approach to model building in this project was the use of an iterative method, which was well suited given the limited prior knowledge. Avoiding the implementation of too many changes at once gave a clear understanding of how each iteration of the model functioned and simplified the identifications of potential bugs. There was a vast difference noticed when implementing too much change at once, resulting in unidentified bugs and eventually time loss as the changes had to be removed to slowly be redone again.

Unfortunately, proper validation of if the system resembles reality was difficult as there were no comparison of the current production outputs for the system. This, in combination with the lack of data regarding batch sizes, meant that the system could not be directly compared to real-life data. Instead, some validation was carried out through discussions with employees and observations of the simulated model.

Furthermore, the final models could be verified by comparing the statistics from each simulation program. Since the input data in each model is identical, the output and utilisation percentages could be compared to ensure that they aligned with the intended functionality and original system design.

5.2 Comparison Between Layout 1 and Layout 2

The results from the simulation were as expected, showing minor differences between the two layouts in regards to product output. These differences are most likely because of the cobots not being able to choose which product should go to which ATE in layout 2, as the products are placed at their intended ATE. An additional contributor to this is the travel time that is slightly less in Layout 2. There was also a minor delay included in the Visual Component ATE codes after the TransportOut command. This delay was implemented to ensure that the cobot that offloaded the ATE, was also the same one that then loaded the ATE, since this was the intended behaviour of the cobots in layout 2. This change most likely contributed to the ATEs in layout 2 getting their products slightly slower in VC.

The difference presented between the 8 hour and 24 hour shifts in products and ATE

utilisation are large as expected. These simulations were run to get a comparison of what the model outputs were if the model was run without any cobots with the same conditions. An issue to keep in mind when comparing the simulations is that the repair station runs during the entire 24 hour shift. Since the repair station has human workers, this does not reflect what the real life situation would look like.

Something to be noted in regards to the travel distance of the cobots is that the cobots did not always pick the best task in VC, so there is an excess of travel distance in layout 2. This means that there is potential for an even greater difference between the travel distances of layout 1 and layout 2.

The travel distance was expected to be shorter in layout 2 since the cobots have the sources right next to the ATE, and in VC this was the case. However, in TPS it was the opposite. This could be due to that the routing for the cobots is different in the the two layouts. In layout 1, there were no obstacles in the path of the cobot and therefore no specific routing was implemented and the transporters were able to always pick the most optimal route straight to its next destination. However, when redesigning the simulation into layout two, the routing became an issue since the cobots were going straight through workplaces. Therefore, the second layout contains a routing table that ensures the robots follow a fixed route of markers. This could be the reason for why the difference in travel distance is longer in layout 2 than in layout 1.

5.3 Comparison of the Simulation Programs

When starting this project there was some prior experience among the authors regarding the simulation programs. One of the authors had completed a short introductory course in VC, while both authors had prior experience from a course in production system simulation using TPS. It should be noted that neither course covered the use of mobile robots, and this functionality was therefore explored independently during the project.

Considering the pre-existing knowledge of the programs, VC was more time consuming and difficult to work with compared to TPS. For reference, building the basic structure in VC took approximately two weeks, whereas the same structure was modelled in TPS in about three days. This time difference can partly be attributed to the difference in prior knowledge of the software. Another contributor to the time difference is that stage 1 and parts of stage 2 were built in VC before starting with stage 1 in TPS. Because of this, some knowledge of how the model could be built in a DES program was already acquired when starting to work with TPS.

Despite these contributors to the time difference in both programs, the main reason for the large difference was likely the different programs structures. This is largely due to the limitations associated with the “drag-and-drop” programming used in VC. While this approach is beneficial for users without prior programming experience, it introduces restrictions when implementing more complex logic and specific

conditions in the model. On the other hand, TPS uses SimTalk, which offers greater flexibility and control. However, for users without prior programming experience, SimTalk can be more difficult to use.

When exploring programming capabilities further with both programs, it was identified that VC supports Python script. However, this structure was less intuitive compared to the SimTalk structure in TPS. This is due to every machine having a built in Python script that controls its automatic functions. Based on a short investigation conducted, it was concluded that in order to implement custom Python scripts effectively, modifications to the existing component logic would be required. As this approach would involve changing or bypassing the built-in functions of the components, this was not considered suitable due to the limited knowledge in VC and the time frame. Therefore, the model development in VC was based on the built-in functions and existing component logic.

In terms of the execution speed for each simulation software, the comparison was slightly unbalanced. As mentioned, some lag was experienced in VC during stage 3 of the model building. However, in VC some additional environmental components were added for it to better represent the real-life environment. This contributed to the execution speed of the program.

During the the time span of the project, in addition to the experts and employees, AI chatbots were used to assist with the logic of the model building. Early in the model building, it was noted that for VC, AI chatbots such as ChatGPT and Copilot were not fit to properly assist with the work. The syntax and logic of the ProcessExecutors as well as some of the core logic in the program itself was not something the AI bots were well equipped to interpret and assist with. In TPS however, AI chatbots were significantly more efficient to assist, as SimTalk was something they could assist with.

Regarding the data retrieved from both programs, some differences occurred. The distance travelled by the cobot in TPS was significantly shorter than the distance travelled in VC. This is partly because the exact distances between the components have not been measured in TPS due to time limitations. More focus was given to the functionality and priority aspects of the simulation. In VC the model was build onto a pre-existing visualisation of the production area and is therefore expected to have the correct distances.

The total number of passed products also differed between the simulation programs. Since the distance travelled by the robot differs between the programs, this most likely causes this difference in product output. For example, if the ATEs are closer in TPS, more products can be delivered quicker.

The two different simulation programs still overall had very similar outputs, meaning that the models were successfully built to behave as closely as possible to each other and the four different models that were built could be validated by comparing

the two programs.

Compared to the findings of Ivanov et al., 2025, the results found in this study are similar when it comes to the five different evaluation criterion. The two programs perform similar when it comes to simulation performance. VC has strengths in 3D modelling and visualisation, and the user interface. TPS is efficient when it comes to scalability. In simulation performance both programs perform well. These findings all align with Ivanov et al., 2025. Something additional that was found in this study is that scaling a model in TPS with mobile robots poses slightly more work than in VC. Since markers were used to decide how the cobots move, additional markers had to be added and structured in the routing table. If the TPS model had pathways instead of markers, it is possible that scaling with more components would be easier to implement.

The academic support and training criteria was evaluated slightly differently. This projects' evaluation of the criteria was focused more on the ease of access to training material in each program. While Ivanov et al., 2025 comes to the conclusion that both programs are equal in this criteria, during this project it was experienced that the academic support and training was more widely accessible in VC without any additional licence.

5.4 Limitations

There were several limitations within this thesis project. First of all, the time frame of the project. The entire project is 20 weeks, which is not very long to undertake a larger project. Therefore, the selection of evaluation criteria has been important in order to focus on certain aspects of the simulation programs when comparing them. Due to the limited time frame, it was also not possible to explore all available functions within the programs in detail, such as the Python scripting functionality in VC.

Considering the different logic in the two programs, the main priority was to build complete, functioning models. Making the models behave exactly the same would have been ideal, but not achieved during the span of the project. There are some behavioural differences in the programs, such as the cobots' exact routings, curve radiuses and pickup- and placement durations.

In a similar way, given more time, the models could have been more accurate to the real-life production area, with more realistic details implemented.

Throughout the time span of the project, new information regarding the layout of the model, and new suggestions concerning functions within the simulation were discussed and requested. This led to more time needed to build the models, which affected the initial plans and, consequently, the time schedule for the thesis.

Working in both programs has also been an additional challenge, as they operate in different ways. The logic used when working therefore had to be adapted for each

program.

Another hinder was the available data. Most of the essential data was obtained, but some data, such as the time it takes to repair a product, was not available. This affected the model building slightly, by limiting what functions and behaviours could be implemented. Most functions could still be implemented but with inaccurate data.

Some of the data that was required for the model to work had to be fabricated to produce any statistics from the simulations. This does not affect the simulation results when it comes to academically comparing the different layouts, but for management at the company to use the models for planning and predictions, the correct data would be required.

The data about mean time before failure (MTBF) and mean time to repair (MTTR) in the ATEs was also unavailable, and in the end not included in the simulation.

Lastly, a lack of simulation competence in VC in the company hindered the possibility to resolve modelling issues through discussions with the team. The same can be said about TPS. Some experts within the simulation field of both VC and TPS were contacted, but the discussions were mainly held via e-mail, which made it more difficult to convey the problems and added delays. When comparing the available academic support for both programs VC was better in terms of providing a variety of beginner friendly material in contrast to TPS which was more technically advanced.

5.5 Managerial Implications

This project can be useful for the company in terms of production planning and decisions regarding what programs to use for their production visualisations. As of right now, VC is the program mainly used, but this may come to change in the future. Depending on the type of work that is required, the company can take this report into account when weighing the different potentials of simulation tools if needed. Furthermore, this project also supports the company's ongoing transition towards Industry 5.0 by analysing the implementation of this new production system with collaboration between cobots and humans.

Regarding the production planning, if the correct data is gathered and used, the models can predict approximate production outputs and the time it will take before certain orders can be considered complete. Different batch sizes can be tested to see what would fit them best. All four model layouts were created in a way that makes it as easy as possible to modify specific part values, processing times and other input data in order to test new scenarios. Finally, this thesis can also give analytical support for which layout the company eventually decides to use.

5.6 Future Work

Considering the time constraint as well as the prior knowledge in each program. There is a lot of potential to improve the current simulation models.

As mentioned earlier, MTBF and MTTR were not implemented at all in the simulations, as there was a lack of existing data and no time to acquire it. In a similar way, the actual mean duration of the repair process was not available, and the data was fabricated instead.

Adding the breakdown data for the ATEs and cobots, more realistic repair times would ensure that the models behave in a more realistic way. Furthermore, implementing the behaviour that if a certain amount of products fail in the same ATE in a row, an automatic breakdown of the ATE would be set.

Some analyses that could have been valuable to do were not done. For example a bottleneck detection analysis could be done if more accurate data was gathered. Since a few of the components are based on fabricated data, the bottleneck detection analysis would most likely not show actual bottlenecks.

To determine what variables affect the simulation the most a sensitivity analysis would be appropriate. This analysis was not performed however due to the time restriction.

Since distances between components in TPS were approximated, and not exact, spending time on adjusting so that each ATE, source and working station is at the correct place would improve the model's credibility.
In the simulation statistics from the models in VC, a full week for the robots had to be separated into two different simulations, as the repair loop should not be included during weekends. In addition to this, when running the simulation for 24 hours during 7 days, the repair loop still functions normally during the evening, night and weekend, which isn't a representation of the real world, as there is no evening, night or weekend shift.

For further improvements, a way to schedule the workers could be implemented, ensuring that during weekends and non-working hours, the repair loop does not continue to run. This is possible to achieve in both programs, but not done due to the time restriction of the project.

Only two simulation tools were used during this project, there are several others which could prove valuable to evaluate but were not due to the size of the project. In a similar way, more criterion could be used to evaluate the software for a broader comparison.

6
Conclusions

Through the iterative model building and continuous validation, the simulation was able to sufficiently represent the interactions that were to happen if two robots were brought into this specific production area. This in turn makes it possible to analyse system behaviour, resource utilisation and the production flow in a structured manner without disrupting the real life system.

For this type of model building, the advanced DES program category, represented by TPS was found to be the ideal simulation tool. Considering the machine complexity needed for the model, TPS supported the logic in a straightforward way compared to the 3D visualisation program, represented by VC. Although VC performed better when it came to the visualisation of the system, this project mainly focused on all the required behaviours functioning properly, which was more efficient to achieve in TPS.

As shown in the simulation results, under 4.2, the two different layouts produced similar numbers but layout 2 resulted in less travel distance for each cobot, while also distributing the work load for each cobot more evenly. This shows that under the given conditions, layout 2 is a better solution in regards to sustainability without affecting the output of the production area too much. However, since in layout 2 the each products are manually assigned to its designated ATE, more planning is required to ensure optimal product placement. This is because the cobots will have less flexibility in selecting the most suitable ATE destination for each product.

Bibliography

OECD. (2025). *Economic security in a changing world*. https://doi.org/10.1787/4eac89c7-en

Security Industry Association & ASIS International. (2024). *Complexities in the global security market: 2024 through 2026*. Omdia.

Dacre, N., Yan, J., Frei, R., Al-Mhdawi, M. K. S., & Dong, H. (2025). Advancing sustainable manufacturing: A systematic exploration of industry 5.0 supply chains for sustainability, human-centricity, and resilience. *Production Planning & Control*, *36*(11), 1499–1528. https://doi.org/10.1080/09537287.2024.2380361

Zhang, J., Hou, Y., Song, W., Shen, J., & Li, Y. (2026). Research framework and evolutionary trends of smart supply chain in industry 4.0: A hybrid methodology integrating bertopic and holt-arima [Published online: 7 April 2026]. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2026.2655808

Fonseca, C. V. C., Predo, R. M., & Blikstad, N. M. D. (2025). Defining a research agenda for industry 4.0 technologies in corporate strategies: Insights from complex networks and machine learning [Published online: 11 June 2025]. *International Journal of Production Research*, *63*(23), 8743–8760. https://doi.org/10.1080/00207543.2025.2514253

Pan, Y., Huang, Z., Lev, B., Xu, L., & Olson, D. (2026). A literature review on the artificial intelligence in manufacturing systems under industry 5.0. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2026.2623537

Gunal, M. M. (Ed.). (2019). *Simulation for industry 4.0*. Springer. https://doi.org/10.1007/978-3-030-04137-3

European Commission. (2024). Industry 5.0 [Accessed 2026-05-19]. https://research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/industry-50_en

United Nations. (n.d.). Sustainability [Accessed January 29, 2026].

Fishman, G. S. (2001). *Discrete-event simulation: Modeling, programming, and analysis*. Springer. https://doi.org/10.1007/978-1-4757-3552-9

Banks, J. Introduction to simulation. In: *1*. 2000, 9–16. https://www.scopus.com/inward/record.uri?eid=2-s2.0-0034429419&partnerID=40&md5=648b45032fbfe4d4454a08c449421df3

Ivanov, V., Amelin, M., & Haidabrus, B. (2025). A comprehensive comparison of modern simulation software for manufacturing systems using multi-criteria

decision analysis. *SN Applied Sciences*, *7*(1), 1–20. https://doi.org/10.1007/s42452-025-08012-y

Bartoș, V.-D., Brad, S., Balog, B., Cirlejan, A., & Ticudean, D. (2025). Enhancing production efficiency through visual components simulation in digital manufacturing systems. *Robotica & Management*, *30*(1), 38–43.

Hoang, L., Nguyen, H. L., Nguyen, X. T. L., Pham, L. B., & Pham, V. S. (2024). Application of tecnomatix in simulation and optimization of manufacturing processes in the factory. *HaUI Journal of Science and Technology*, *60*(5). https://doi.org/10.57001/hui5804.2024.148

Sze, D. L., Reyes, R. S. J., & Abu, P. A. R. (2024). Simulation of a pick and place system for electronic cards using a YuMi cobot. *Proceedings of the 10th International Conference on Automation, Robotics, and Applications (ICARA)*, 128–132. https://doi.org/10.1109/ICARA60736.2024.10553214

Nogueira, R., Reis, J., Pinto, R., & Gonçalves, G. (2019). Self-adaptive cobots in cyber-physical production systems. *Proceedings of the IEEE International Conference on Industrial Informatics (INDIN)*, 521–528.

Skoogh, A., & Johansson, B. (2008). A methodology for input data management in discrete event simulation projects. *Proceedings of the 2008 Winter Simulation Conference*, 1727–1735. https://doi.org/10.1109/WSC.2008.4736055

Almström, P., Mårdberg, P., Carlson, J., Högberg, D., Jeong, Y., Wiktorsson, M., Mattsson, S., Backman, B., & Gåsvaer, D. (2024). *Time data management: A handbook*. MTM Nordic.

Sargent, R. G. (2010). Verification and validation of simulation models. *Proceedings of the 2010 Winter Simulation Conference*, 166–183.

Hora, J., & Campos, P. (2015). A review of performance criteria to validate simulation models. *Expert Systems*, *32*(5), 578–593. https://doi.org/10.1111/exsy.12111

Ciauşu-Sliwa, M. R., Ciauşu-Sliwa, D., & Dodun, O. (2025). Discrete-event simulation for smart manufacturing: Assessing software, industry adoption, and future research challenges. *Bulletin of the Polytechnic Institute of Iaşi. Machine Constructions Section*. https://doi.org/10.2478/bipcm-2025-0009

Ajayi, V. (2023). A review on primary sources of data and secondary sources of data. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5378785

A

Appendix

A.1 Visual Components Layout 1 and 2

This section of the appendix contains all the logic used in Visual Components to program the simulation in Layout 1 and 2.

A.1.1 Feeder codes

As can be seen in the figures A.1 and A.2 below, the code for the product feeders are almost identical. The main difference is that for layout 2 the flow step is set so the product goes to the correct ATE and it also adds to the corresponding counter.

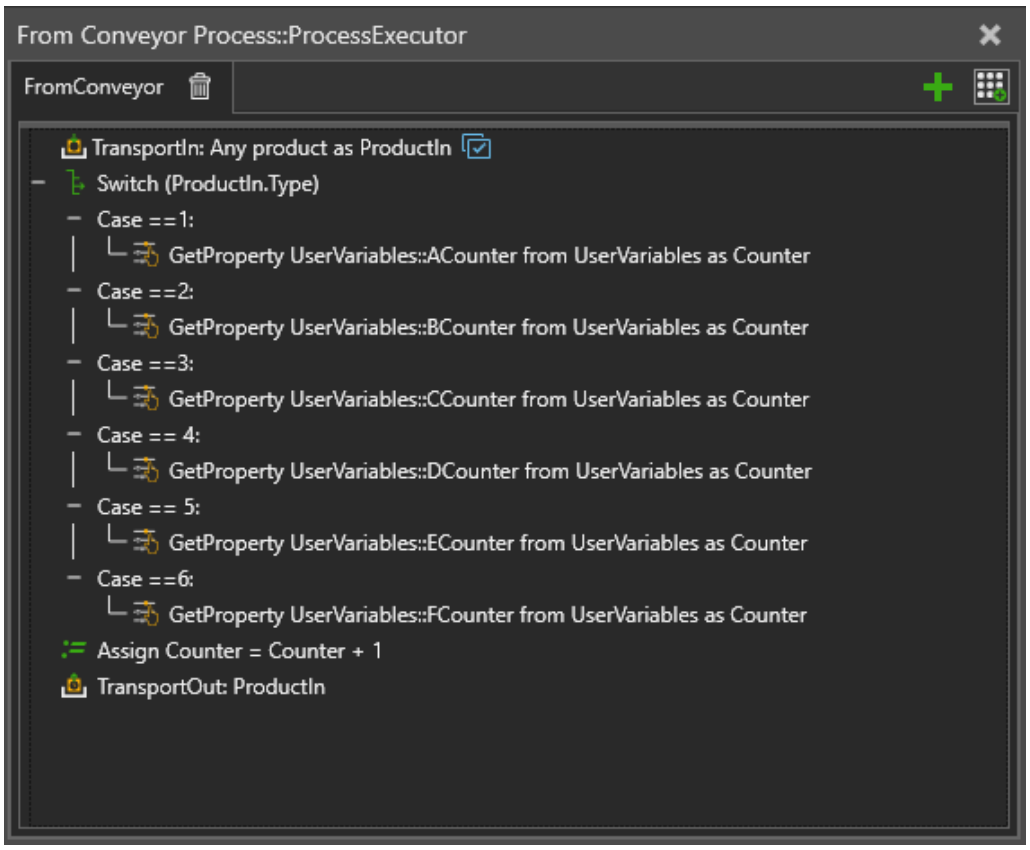


Figure A.1: Feeder code Layout 1

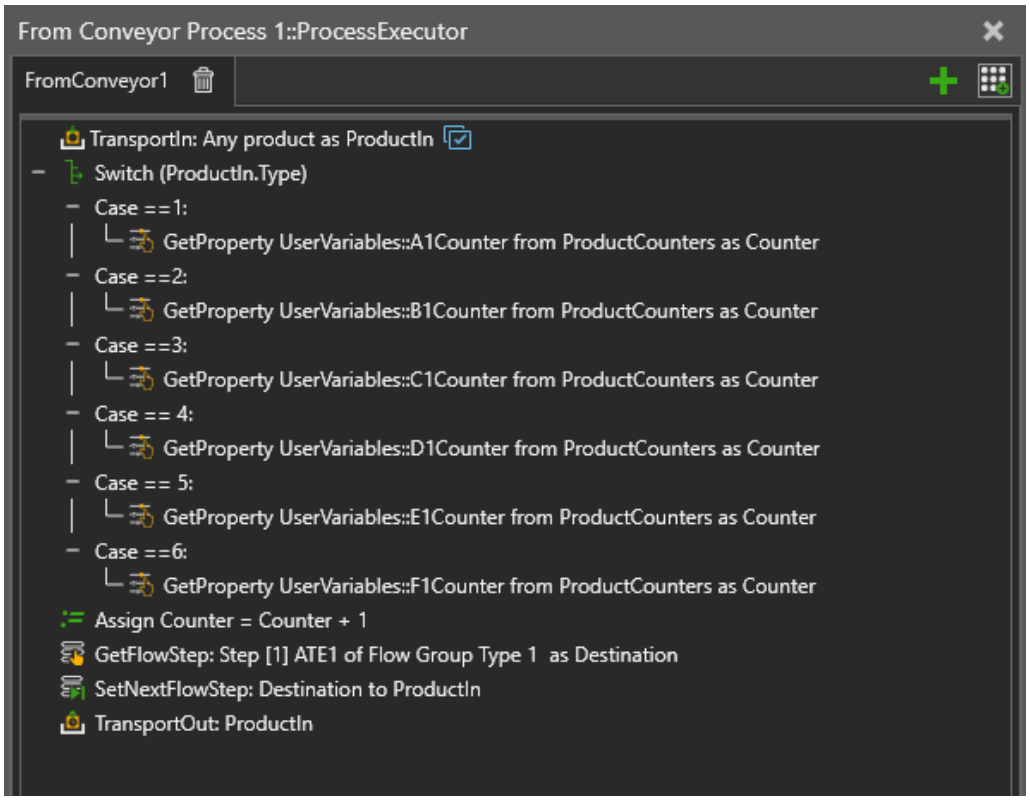


Figure A.2: Feeder code Layout 2

A.1.2 ATE codes

The following figures are screenshots from the ATE1 codes in layout 1 and 2. Each ATE has a very similar code, with some lines that are unique to each ATE. The codes for ATE4, 5 and 6 follow the same logic but with fewer products.

Figure A.3: ATE1 code Layout 1 - Part 1

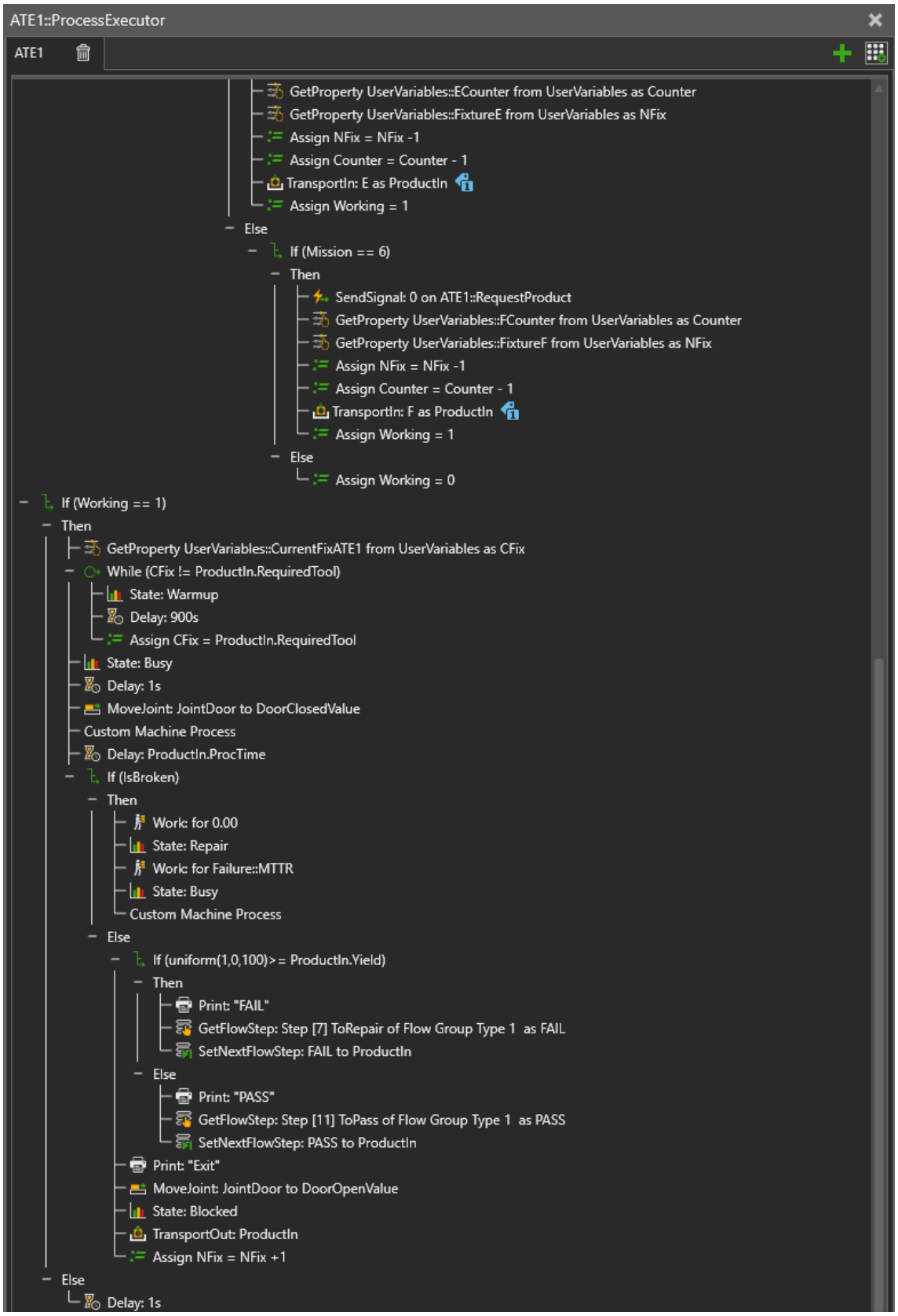


Figure A.4: ATE1 code Layout 1 - Part 2

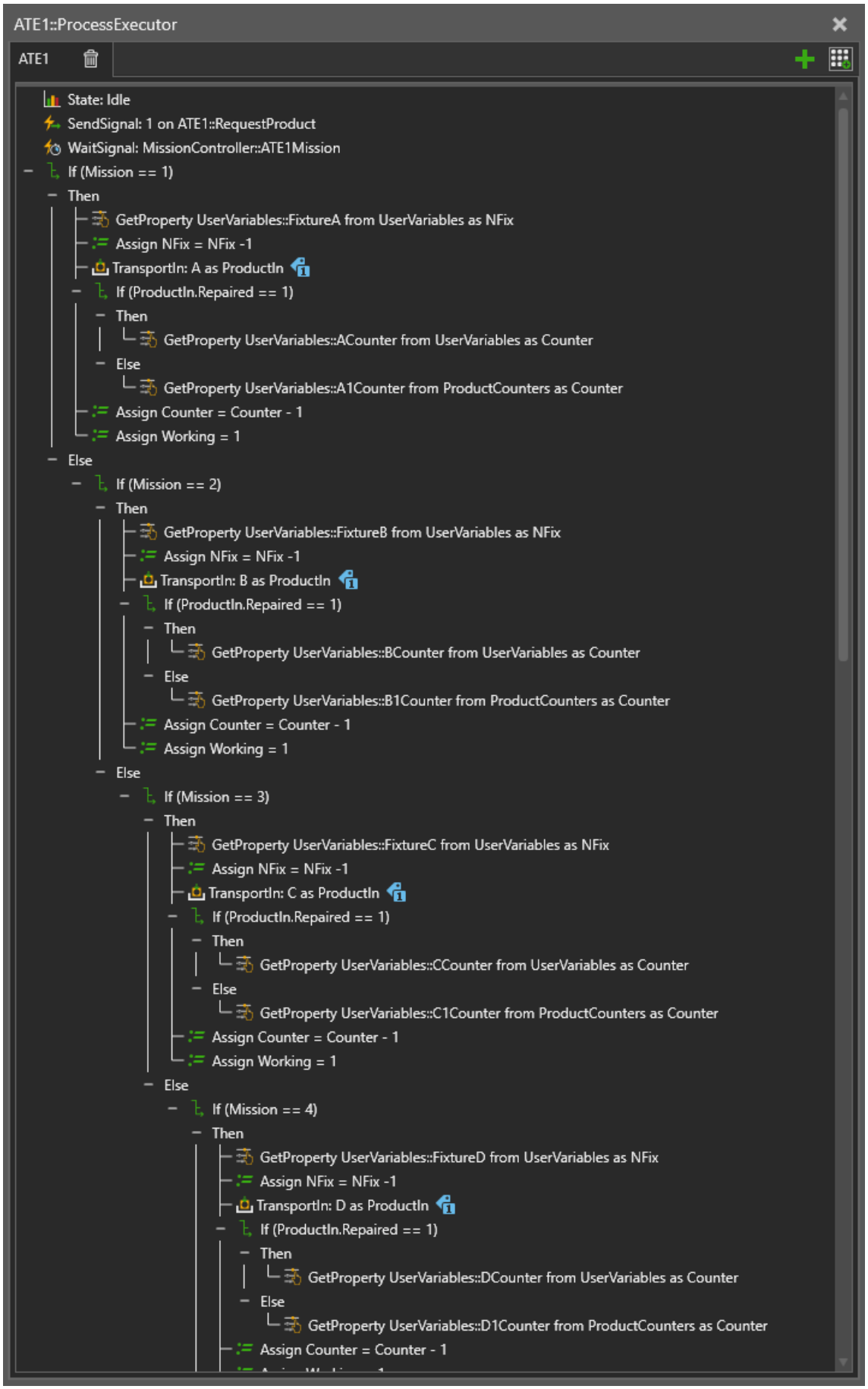


Figure A.5: ATE1 code Layout 2 - Part 1

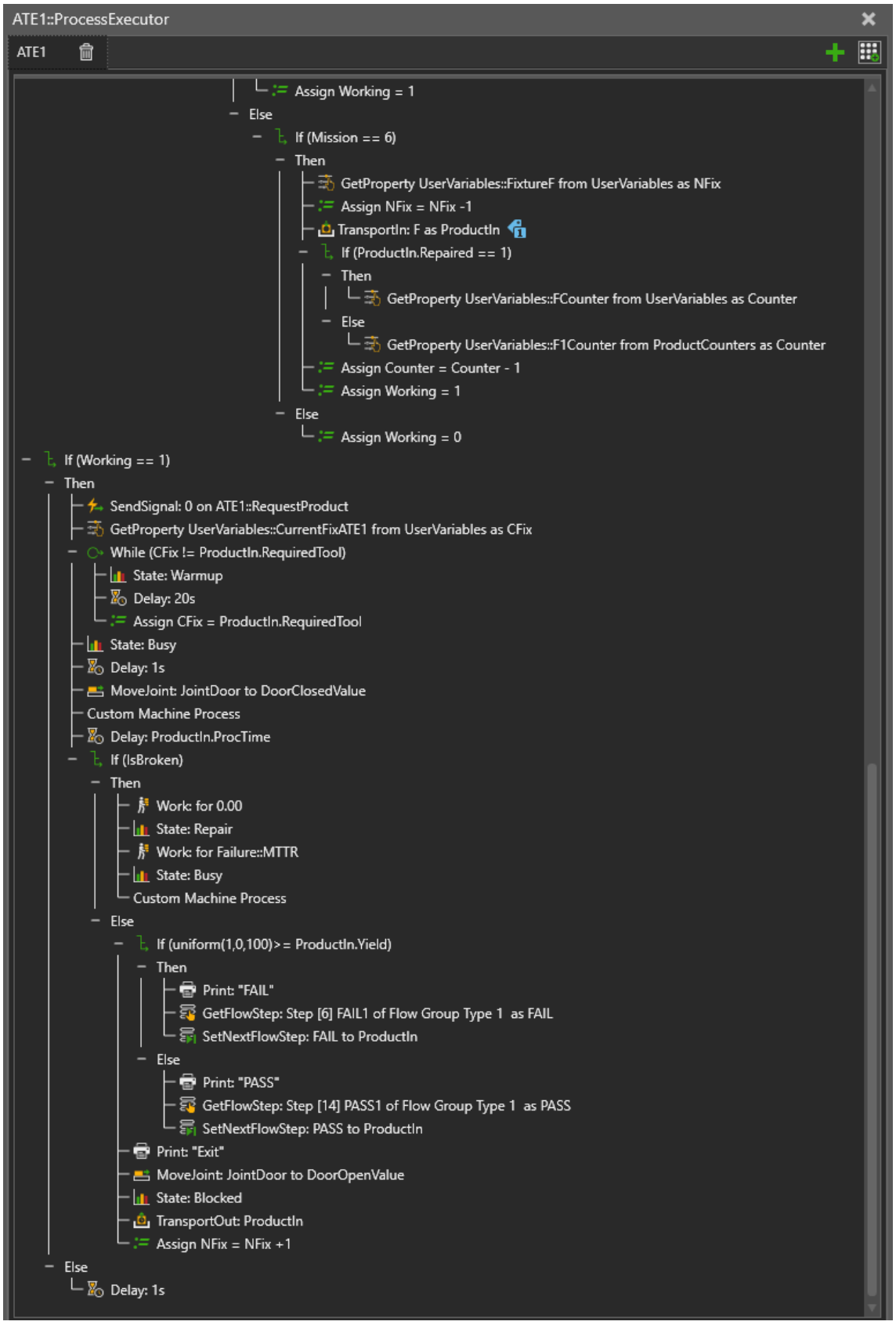


Figure A.6: ATE1 code Layout 2 - Part 2

A.1.3 Mission Controller codes

The only difference between the Mission Controller codes between the two layouts was that for layout 2, the code used the proper product counters for each ATE as well as accounting for potential products coming from the repair loop when assigning missions. The following figures show the code that decides which ATE should request which product.

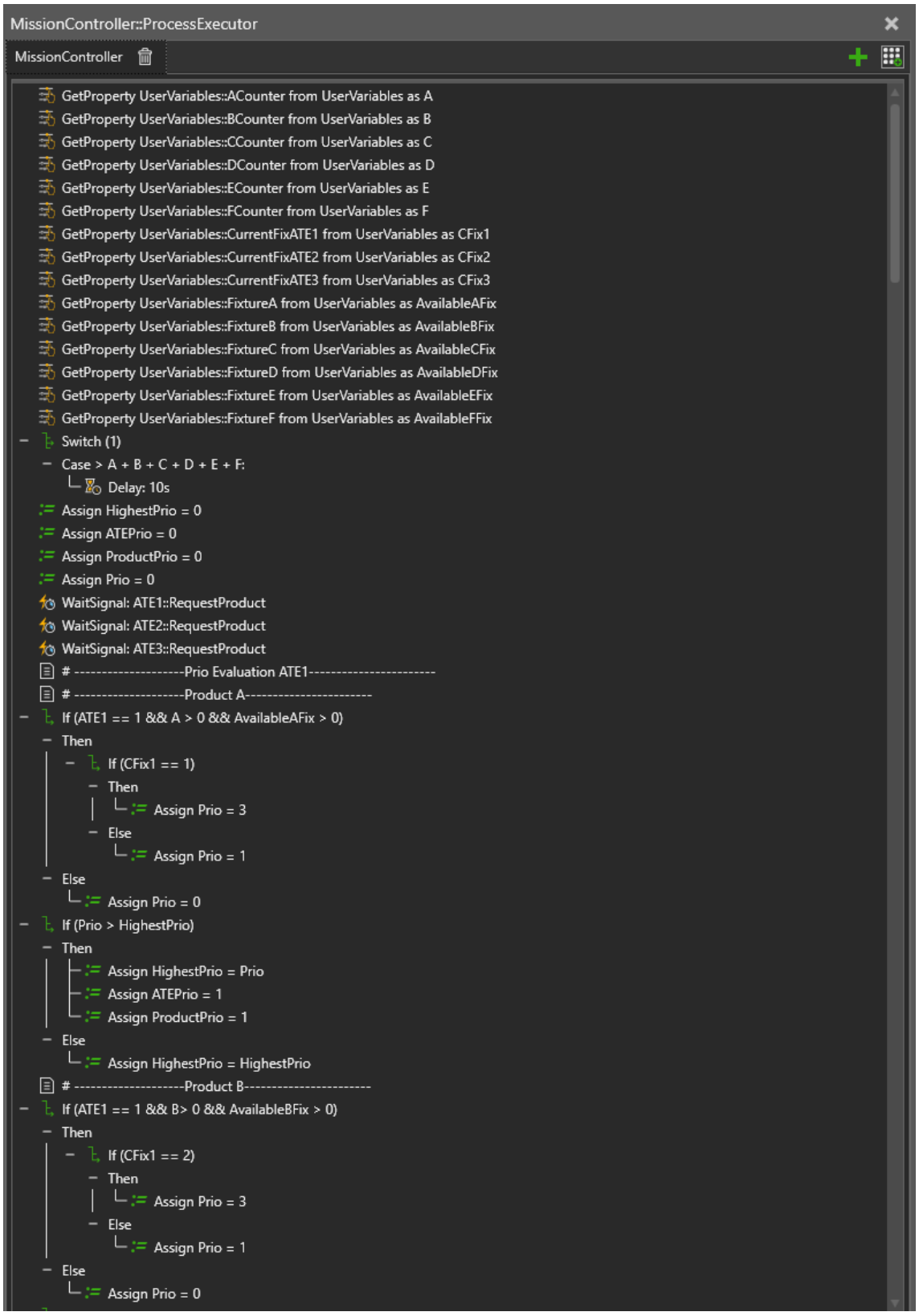


Figure A.7: Mission Controller code Layout 1 - Part 1

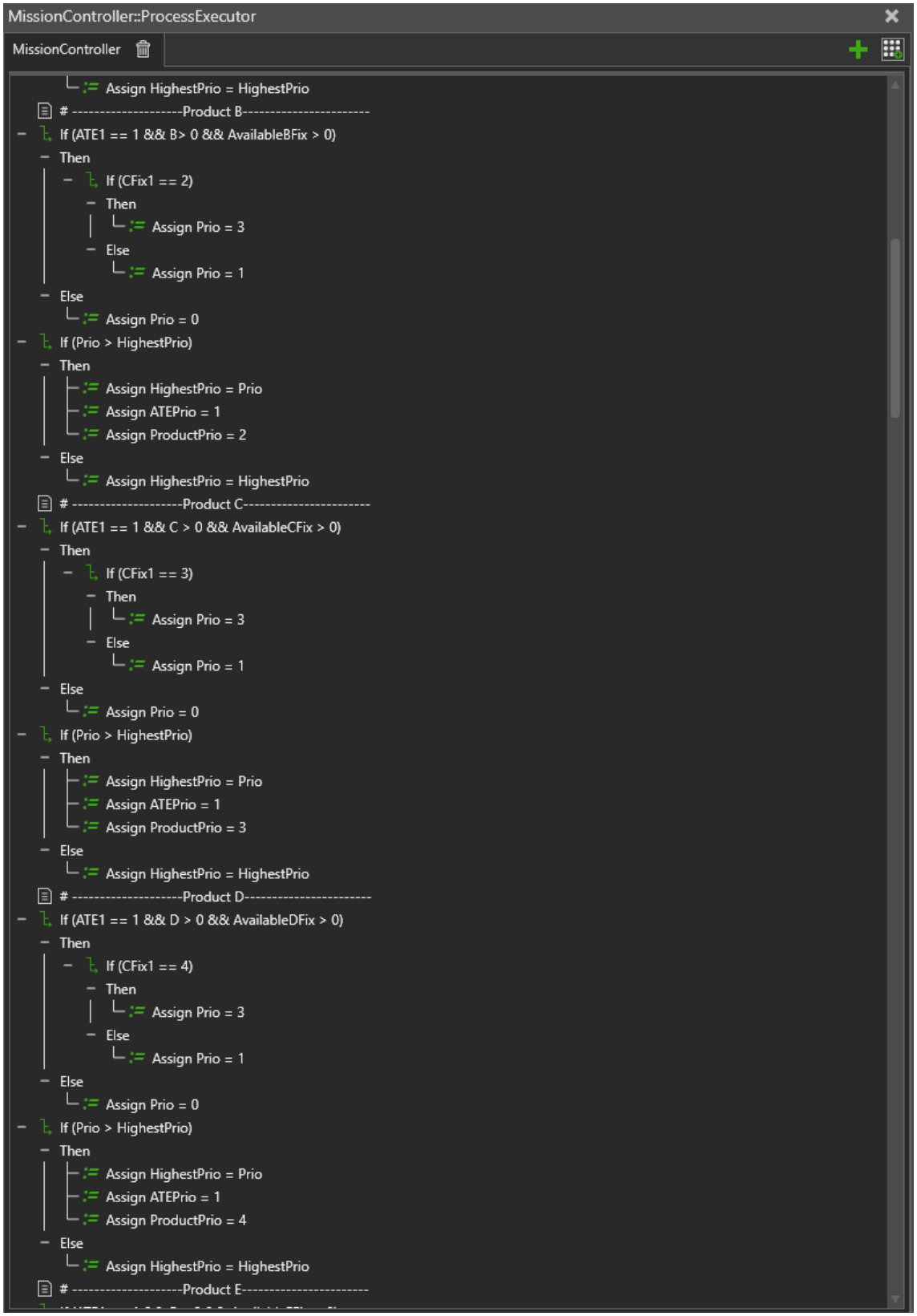


Figure A.8: Mission Controller code Layout 1 - Part 2

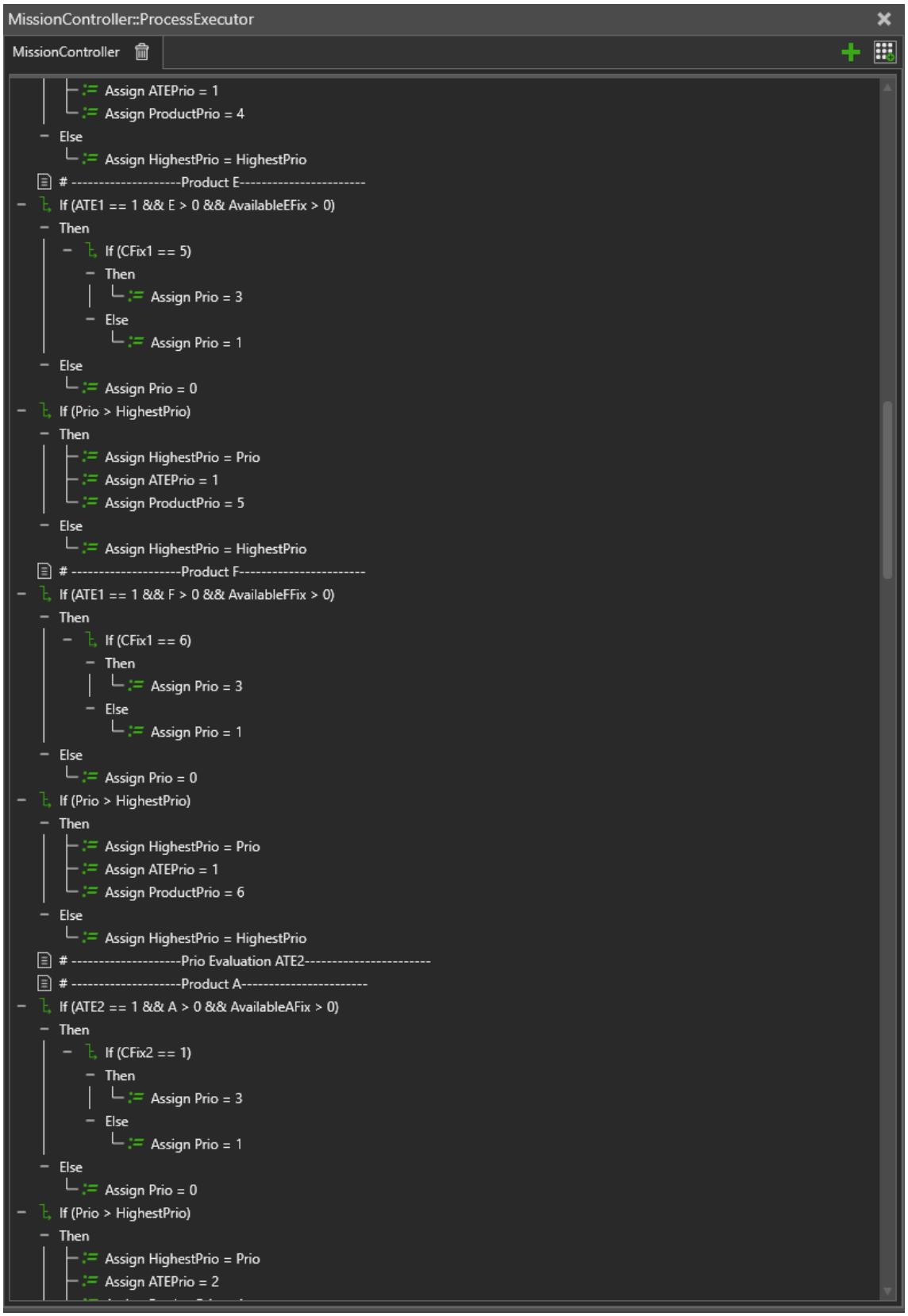


Figure A.9: Mission Controller code Layout 1 - Part 3

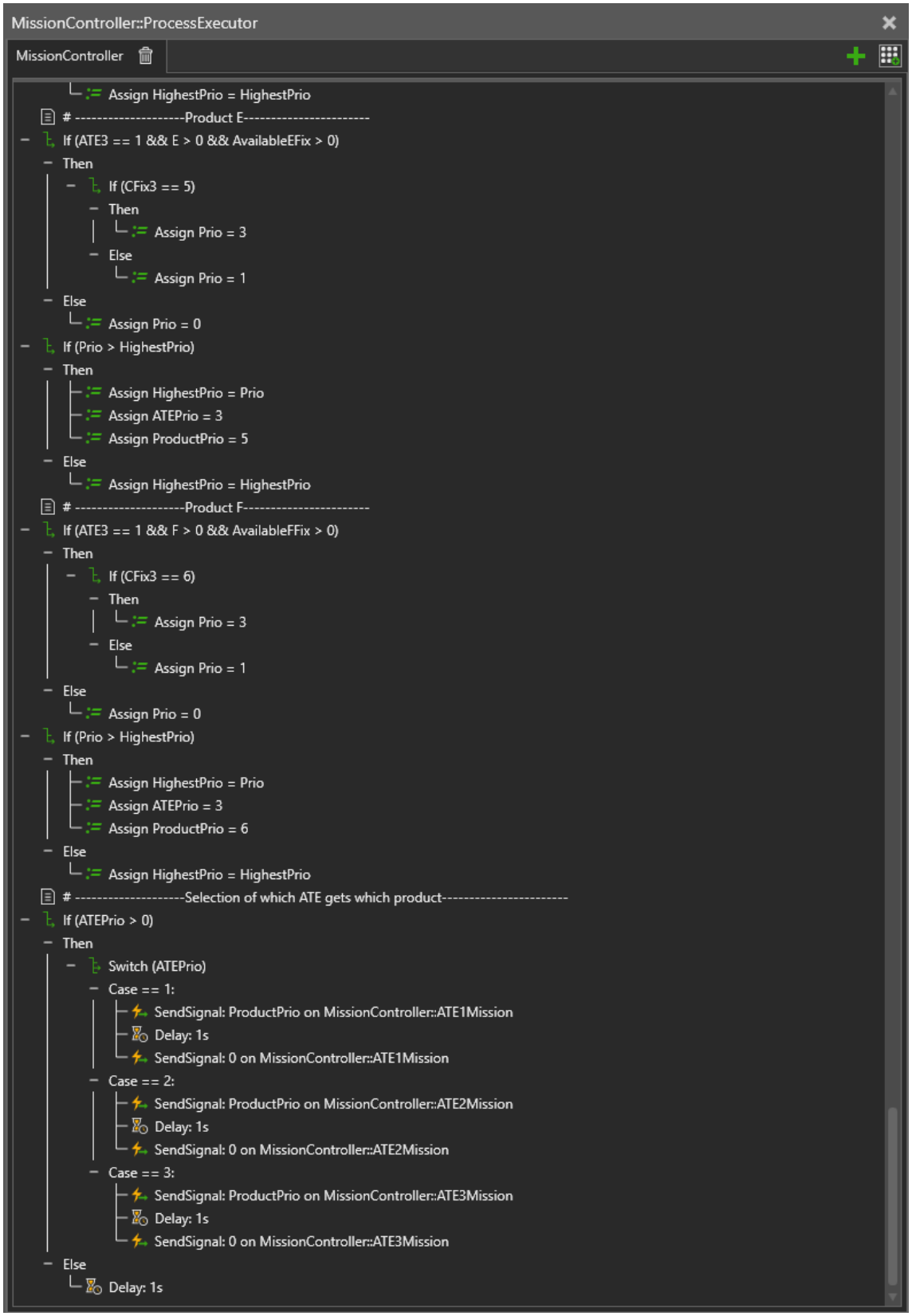


Figure A.10: Mission Controller code Layout 1 - Part 4

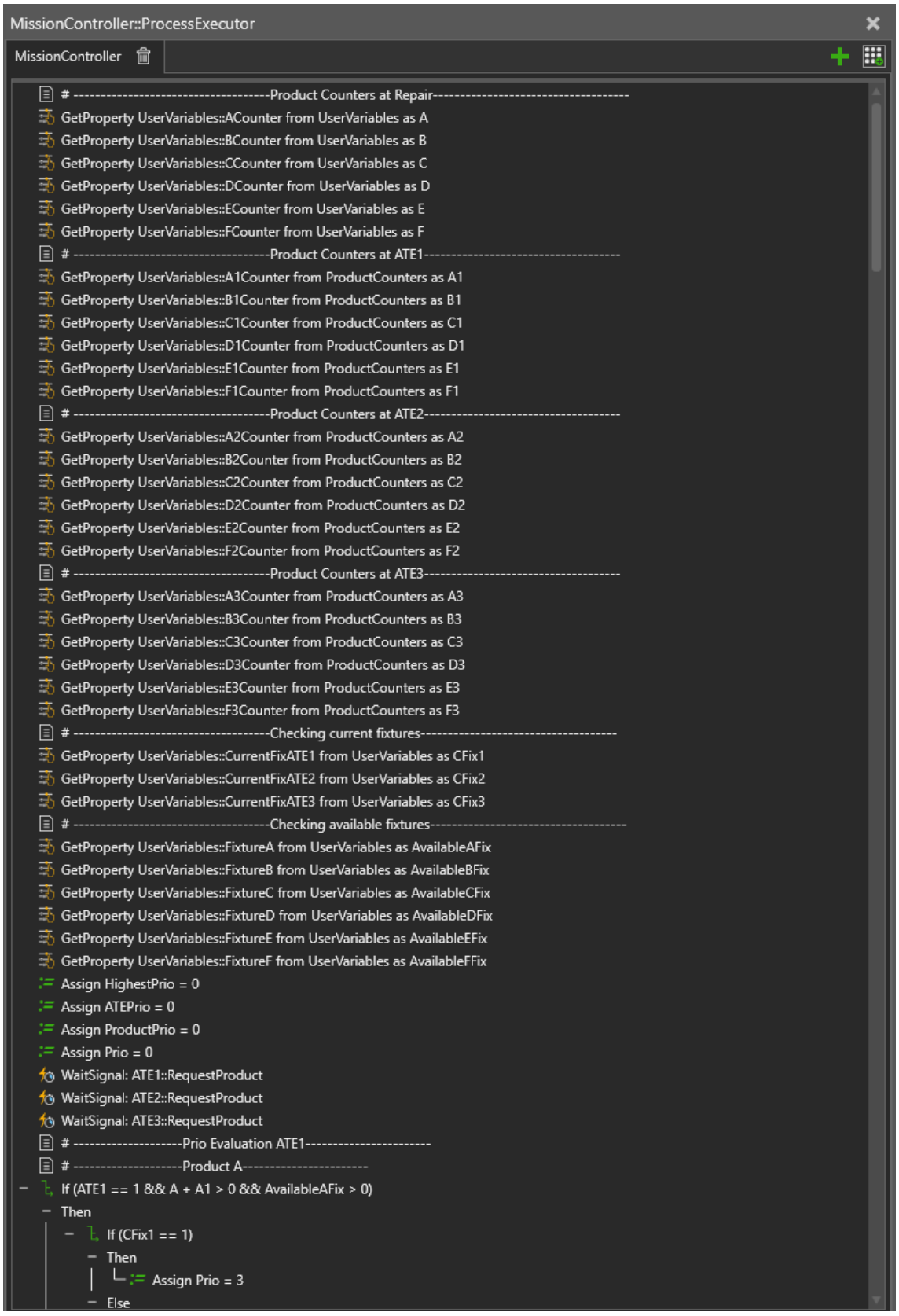


Figure A.11: Mission Controller code Layout 2 - Part 1

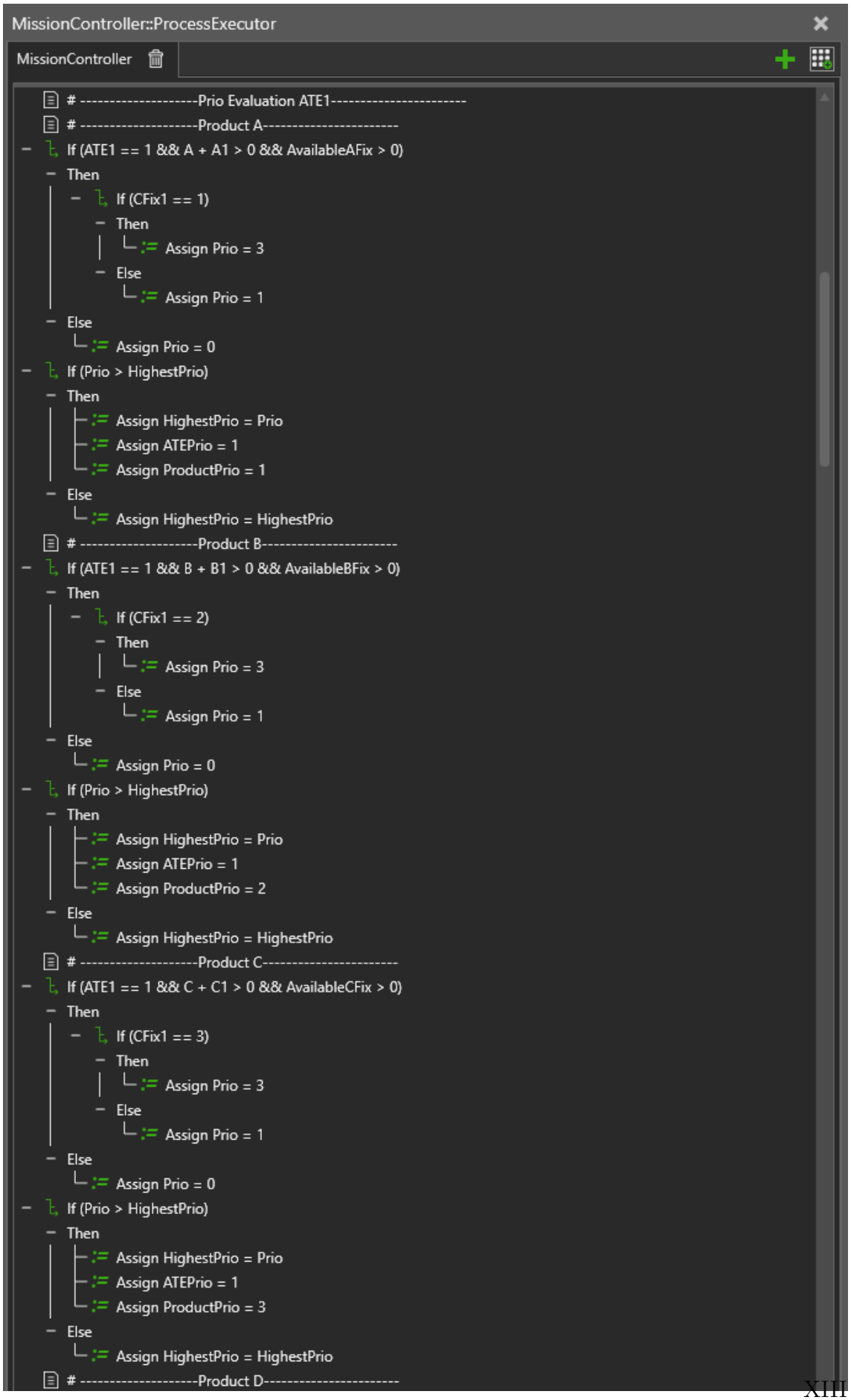


Figure A.12: Mission Controller code Layout 2 - Part 2

A.1.4 Repair station codes

The code for the repair stations for layout 1 and layout 2 are almost identical. The only difference is that in layout 2 the products in all the stacks at ATE1, ATE2 and ATE3 are summed up while in layout 1 there is only one stack to take into account.

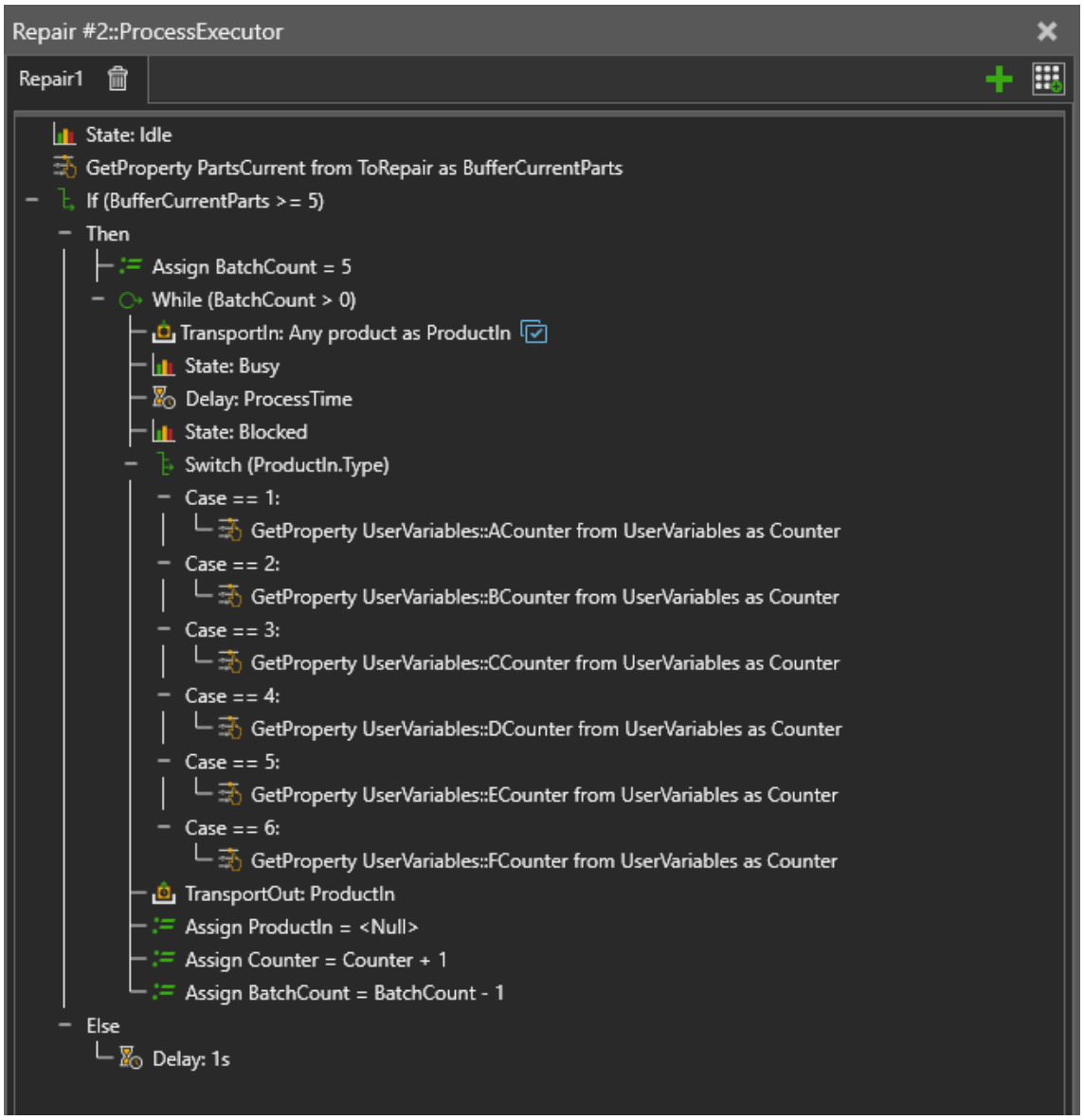


Figure A.13: Repair station code Layout 1

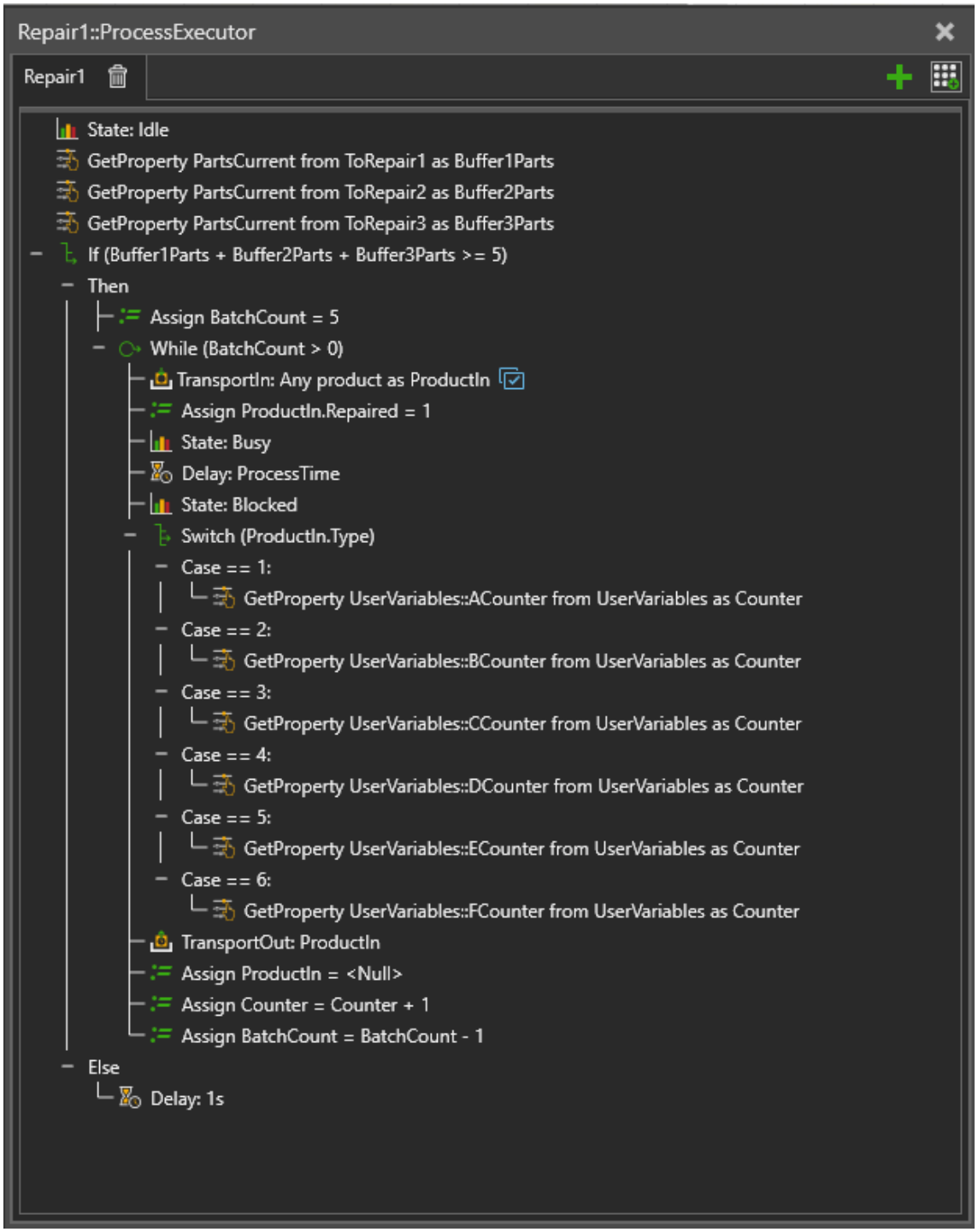


Figure A.14: Repair station code Layout 2

A.1.5 Product list and product properties

Figure A.15 shows a list of the products and the product properties for product A and B.

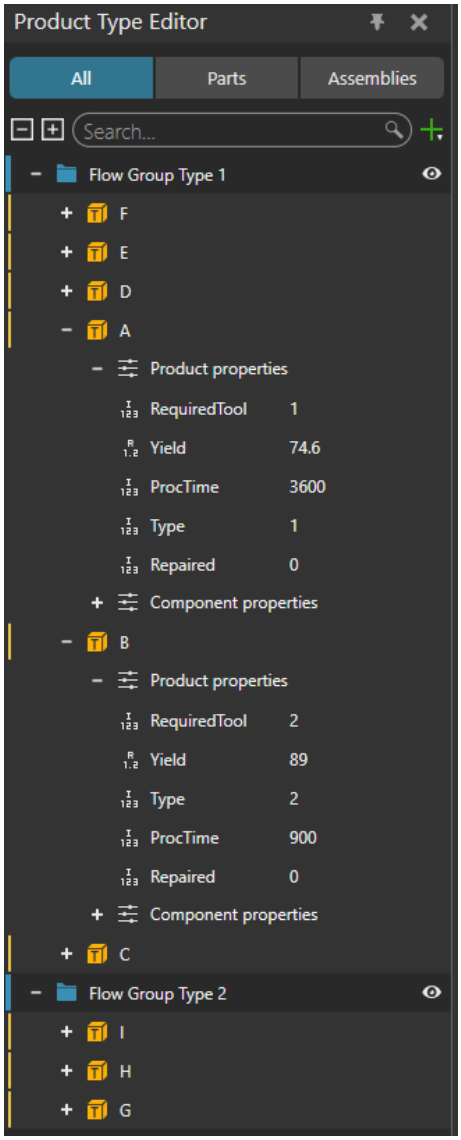


Figure A.15: List of products and their properties

A.2 Tecnomatix Plant Simulation Layout 1

This section of the appendix contains all the methods used in Tecnomatix Plant Simulation to program the simulation in Layout 1.

A.2.1 Init_X

```
--INIT_X-------------------------------------------------------------------------
--Init code, connected to AGVpool2 - Controls AGV in area X

-- 1. Remove finished part from ATE machines
-- 2. Move repaired parts back to ATE
-- 3. Fill machines from Sources
-- 4. Idle if nothing else happened
-----------------------------------------------------------------------------------

--The controller waits until an AGV becomes available, then assigns it to agv.
waituntil AGVPool2.NumIdleAGVs > 0
var agv := AGVPool2.getIdleAGV

--Creates arrays that system can loop through
var Sources := [SourceA, SourceB, SourceC, SourceD, SourceE, SourceF]
var sourceMarkers := [MarkerSourceA, MarkerSourceB, MarkerSourceC, MarkerSourceD, MarkerSourceE,
     MarkerSourceF]
var ATEs := [ATE1, ATE2, ATE3]
var markers := [MarkerATE1, MarkerATE2, MarkerATE3]

--Variables
var taskDone : boolean := false -- limits robot to one task per loop
var emptyATE : boolean := false -- checks for empty machine
var targetATE := void -- stores chosen machine
var partType : integer -- stores current product type number
var idx : integer -- marker for specific machine, used when selecting markers

while true --Main control loop, runs continously during simulation

  --Resets loop variables to avoid simulatoin freezing
    --(partType & idx get overwrittten so no need to reset)
  taskDone := false
    emptyATE := false
    targetATE := void

-----------------------------------------------------------------------------------
-- STOP AGVs AFTER 8h
  --> This section is toggled when robots are to be active for 24 hours
-----------------------------------------------------------------------------------
  if eventController.simTime >= 8:00:00 --checks is simulation time has reached 8 hours

    if agv.name = ".UserObjects.Transporter2:1"--checks fro correct AGV
        agv.setRoute([IdleMarker2]) --Sends AGV to idle marker
        waituntil agv.destinationWasReached
    end

    wait 1
    continue --Skips remaining logic

  end
-----------------------------------------------------------------------------------



-----------------------------------------------------------------------------------
-- Check if any ATE is empty
-----------------------------------------------------------------------------------
for var i := 1 to ATEs.dim
    if ATEs[i].Empty
        emptyATE := true
```

```
        exitloop
    end
next

-------------------------------------------------------------------------------
-- 1. Remove finished part from ATE machines
-------------------------------------------------------------------------------

for var i := 1 to ATEs.dim

    var station := ATEs[i] --Stores current machine

  --Checks if machine contains MU & if finished
    if station.cont != void and station.cont.finished

        agv.setRoute([markers[i]])
        waituntil agv.destinationWasReached

        var mu := station.cont --Stores curent MU
        mu.move(agv) --Load part onto AGV

    ------------------------------------------
    -- PASS & FAIL logic
    ------------------------------------------
        if mu.Quality = 1
            agv.setRoute([FAILMarker2])
            waituntil agv.destinationWasReached
            mu.move(FAIL_X)
        else
            agv.setRoute([PASSMarker2])
            waituntil agv.destinationWasReached
            mu.move(PASS_X)
        end

        taskDone := true
        exitloop

    end

next

-------------------------------------------------------------------------------
-- 2. Move repaired parts back to ATE
  --only runs if:
    --AGV doesnt ahve task assigned
    --Reapair station contains MU
    --AGV empty
    --Machine available
-------------------------------------------------------------------------------

if not taskDone and FromRepairX.cont != void and agv.NumMU = 0 and emptyATE

  var repairMU := FromRepairX.cont

  --Starts by assuming transport is allowed, then checks for available fixtures
      --if no fixtures are available -> task is blocked
  var allow : boolean := true

  if repairMU.Class = .UserObjects.PartA and FixtureA <= 0
      allow := false
  end

  if repairMU.Class = .UserObjects.PartB and FixtureB <= 0
      allow := false
  end

  if repairMU.Class = .UserObjects.PartC and FixtureC <= 0
      allow := false
  end

```

```
129  if repairMU.Class = .UserObjects.PartD and FixtureD <= 0
130      allow := false
131  end
132
133  if repairMU.Class = .UserObjects.PartE and FixtureE <= 0
134      allow := false
135  end
136
137  if repairMU.Class = .UserObjects.PartF and FixtureF <= 0
138      allow := false
139  end
140
141  --Reserves fixture that is being used
142  if allow
143
144      if repairMU.Class = .UserObjects.PartA
145          FixtureA -= 1
146      end
147
148      if repairMU.Class = .UserObjects.PartB
149          FixtureB -= 1
150      end
151
152      if repairMU.Class = .UserObjects.PartC
153          FixtureC -= 1
154      end
155
156    if repairMU.Class = .UserObjects.PartD
157          FixtureD -= 1
158      end
159
160      if repairMU.Class = .UserObjects.PartE
161          FixtureE -= 1
162      end
163
164      if repairMU.Class = .UserObjects.PartF
165          FixtureF -= 1
166      end
167
168      agv.setRoute([FromFAIL2])
169      waituntil agv.destinationWasReached
170
171      repairMU.move(agv)
172
173    --------------------------------
174    -- choose best ATE
175    --------------------------------
176
177    partType := repairMU.ProductType --Stores product type
178
179    -- Avoid changeovers, chose machine that has processed same product if possible
180    for var i := 1 to ATEs.dim
181        if ATEs[i].Empty and ATEs[i].lastProductType = partType
182            targetATE := ATEs[i]
183           exitloop
184        end
185    next
186
187    -- If not possible, choose any empty ATE
188    if targetATE = void
189        for var i := 1 to ATEs.dim
190            if ATEs[i].Empty
191                targetATE := ATEs[i]
192                exitloop
193            end
194        next
195    end
196
197  --------------------------------
198    -- deliver part to ATE
```

```
    ------------------------------

    if targetATE /= void --This means if a valid ATE has been found

        idx := ATEs.find(targetATE) -- finds selected machine

        agv.setRoute([markers[idx]]) --sends AGV to corect ATE marker
        waituntil agv.destinationWasReached

        agv.cont.move(targetATE)

        taskDone := true

    end

end

end


-------------------------------------------------------------------------------
-- 3. Fill machines from Sources
  --only runs if:
      --AGV doesnt have task assigned
      --Machine available
      --AGV empty
-------------------------------------------------------------------------------
if not taskDone and emptyATE and agv.NumMU = 0

  for var b := 1 to Sources.dim -- checks Sources

      if Sources[b].cont != void --checks if sources have a part

          var SourceMU := Sources[b].cont --gets part from source

          ------------------------------
          -- fixture check
          ------------------------------

          if SourceMU.Class = .UserObjects.PartA
              if FixtureA <= 0
                  continue
              end
              FixtureA -= 1
          end

          if SourceMU.Class = .UserObjects.PartB
              if FixtureB <= 0
                  continue
              end
              FixtureB -= 1
          end

          if SourceMU.Class = .UserObjects.PartC
              if FixtureC <= 0
                  continue
              end
              FixtureC -= 1
          end


      if SourceMU.Class = .UserObjects.PartD
              if FixtureD <= 0
                  continue
              end
              FixtureD -= 1
          end

          if SourceMU.Class = .UserObjects.PartE
              if FixtureE <= 0
```

```
269                 continue
270             end
271             FixtureE -= 1
272         end
273
274         if SourceMU.Class = .UserObjects.PartF
275             if FixtureF <= 0
276                 continue
277             end
278             FixtureF -= 1
279         end
280
281
282         agv.setRoute([sourceMarkers[b]])
283         waituntil agv.destinationWasReached
284
285         SourceMU.move(agv)
286
287
288         --------------------------------
289         -- choose best ATE
290         --------------------------------
291
292
293         partType := SourceMU.ProductType
294
295
296         -- Avoid changeovers, chose machine that has processed same product if possible
297         for var i := 1 to ATEs.dim
298             if ATEs[i].Empty and ATEs[i].lastProductType = partType
299                 targetATE := ATEs[i]
300                 exitloop
301             end
302         next
303
304
305         -- If not possible, choose any empty ATE
306         if targetATE = void
307             for var i := 1 to ATEs.dim
308                 if ATEs[i].Empty
309                     targetATE := ATEs[i]
310                     exitloop
311                 end
312             next
313         end
314
315     --------------------------------
316       -- deliver part to ATE
317       --------------------------------
318
319         if targetATE /= void
320
321             idx := ATEs.find(targetATE)
322
323             agv.setRoute([markers[idx]])
324             waituntil agv.destinationWasReached
325
326       agv.cont.move(targetATE);
327
328
329             taskDone := true
330
331         end
332
333
334         if taskDone
335             exitloop
336         end
337
338     end
```

```
339
340   next
341
342 end
343
344
345 ------------------------------------------------------------------------------------
346 -- 4. Idle if nothing else happened
347   --only runs if:
348     --AGV doesnt have task assigned
349 ------------------------------------------------------------------------------------
350 if not taskDone
351
352   agv.setRoute([IdleMarker2])
353   waituntil agv.destinationWasReached
354
355   wait 1 --Helps prevent freezing and debugger appearing
356
357   end
358
359 end
```

A.2.2 Init_Y

```
1  --INIT_Y------------------------------------------------------------------------
2  --Init code, connected to AGVpool - Controls AGV in area Y
3
4  -- 1. Remove finished part from ATE machines
5  -- 2. Move repaired parts back to ATE
6  -- 3. Fill machines from Sources
7  -- 4. Idle if nothing else happened
8  ------------------------------------------------------------------------------------
9
10 --The controller waits until an AGV becomes available, then assigns it to agv.
11 waituntil AGVPool.NumIdleAGVs > 0
12 var agv := AGVPool.getIdleAGV
13
14 --Creates arrays that system can loop through
15 var Sources := [SourceG, SourceH, SourceI]
16 var sourceMarkers := [MarkerSourceG, MarkerSourceH, MarkerSourceI]
17 var ATEs := [ATE4, ATE5, ATE6]
18 var markers := [MarkerATE4, MarkerATE5, MarkerATE6]
19
20 --Variables
21 var taskDone : boolean := false
22 var emptyATE : boolean := false
23 var targetATE := void
24 var partType : integer
25 var idx : integer
26
27 while true
28
29   taskDone := false
30     emptyATE := false
31     targetATE := void
32
33 ------------------------------------------------------------------------------------
34 -- STOP AGVs AFTER 8h
35 ------------------------------------------------------------------------------------
36 if eventController.simTime >= 8:00:00
37
38     if agv.name = ".UserObjects.Transporter:1"
39         agv.setRoute([IdleMarker])
40         waituntil agv.destinationWasReached
41     end
42
43     wait 1
```

```
    continue

end
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
-- Check if any ATE is empty
-------------------------------------------------------------------------------

    for var i := 1 to ATEs.dim
        if ATEs[i].Empty
            emptyATE := true
            exitloop
        end
    next


-------------------------------------------------------------------------------
-- 1. Remove finished part from ATE machines
-------------------------------------------------------------------------------

    for var i := 1 to ATEs.dim

        var station := ATEs[i]

        if station.cont != void and station.cont.finished

            agv.setRoute([markers[i]])
            waituntil agv.destinationWasReached

            var mu := station.cont
            mu.move(agv)

      ------------------------------------------
      -- PASS & FAIL logic
      ------------------------------------------

            if mu.Quality = 1
                agv.setRoute([FAILMarker])
                waituntil agv.destinationWasReached
                mu.move(FAIL_Y)
            else
                agv.setRoute([PASSMarker])
                waituntil agv.destinationWasReached
                mu.move(PASS_Y)
            end

            taskDone := true
            exitloop

        end

    next



-----------------------------------------------
-- 2. Move repaired parts back to ATE
-----------------------------------------------

if not taskDone and FromRepairY.cont != void and agv.NumMU = 0 and emptyATE

    var repairMU := FromRepairY.cont

    var allow : boolean := true

    if repairMU.Class = .UserObjects.PartG and FixtureG <= 0
        allow := false
    end

```

```
    if repairMU.Class = .UserObjects.PartH and FixtureH <= 0
        allow := false
    end

    if repairMU.Class = .UserObjects.PartI and FixtureI <= 0
        allow := false
    end


    if allow

        if repairMU.Class = .UserObjects.PartG
            FixtureG -= 1
        end

        if repairMU.Class = .UserObjects.PartH
            FixtureH -= 1
        end

        if repairMU.Class = .UserObjects.PartI
            FixtureI -= 1
        end

        agv.setRoute([FromFAIL])
        waituntil agv.destinationWasReached

        repairMU.move(agv)

        ------------------------------
        -- choose best ATE
        ------------------------------

        partType := repairMU.ProductType

        -- Avoid changeovers, chose machine that has processed same product if possible
        for var i := 1 to ATEs.dim

      ---------------------------------------------------------------------
      -- PartH is NOT allowed in ATE4
      --(if current part is H and the current ATE in the loop is ATE4,
      --then skip ATE4 and use another ATE))

            if repairMU.Class = .UserObjects.PartH and ATEs[i] = ATE4
                continue
            end
      ---------------------------------------------------------------------

            if ATEs[i].Empty and ATEs[i].lastProductType = partType
                targetATE := ATEs[i]
               exitloop
            end
        next

        -- If not possible, choose any empty ATE
        if targetATE = void
            for var i := 1 to ATEs.dim

        ---------------------------------------------------------------
        -- PartH is NOT allowed in ATE4

                if repairMU.Class = .UserObjects.PartH and ATEs[i] = ATE4
                    continue
                end
        ---------------------------------------------------------------

                if ATEs[i].Empty
                    targetATE := ATEs[i]
                    exitloop
                end
            next
```

```
        end

    ------------------------------
      -- deliver part to ATE
      ------------------------------


        if targetATE /= void

            idx := ATEs.find(targetATE) -- deliver to the selected machine

            agv.setRoute([markers[idx]])
            waituntil agv.destinationWasReached

            agv.cont.move(targetATE)

            taskDone := true

        end

    end

end



-----------------------------------------------
-- 3. Fill machine from Sources
-----------------------------------------------

if not taskDone and emptyATE and agv.NumMU = 0

    for var b := 1 to Sources.dim

        if Sources[b].cont != void

            var SourceMU := Sources[b].cont

            ----------------------------
            -- fixture check
            ----------------------------

            if SourceMU.Class = .UserObjects.PartG
                if FixtureG <= 0
                    continue
                end
                FixtureG -= 1
            end

            if SourceMU.Class = .UserObjects.PartH
                if FixtureH <= 0
                    continue
                end
                FixtureH -= 1
            end

            if SourceMU.Class = .UserObjects.PartI
                if FixtureI <= 0
                    continue
                end
                FixtureI -= 1
            end


            agv.setRoute([sourceMarkers[b]])
            waituntil agv.destinationWasReached

            SourceMU.move(agv)


```

```
254             -------------------------------
255             -- choose best ATE
256             -------------------------------
257
258
259             partType := SourceMU.ProductType
260
261             -- Avoid changeovers, chose machine that has processed same product if possible
262             for var i := 1 to ATEs.dim
263
264         ----------------------------------------------------------------
265         -- PartH is NOT allowed to ATE4
266               if SourceMU.Class = .UserObjects.PartH and ATEs[i] = ATE4
267                   continue
268               end
269         ----------------------------------------------------------------
270
271                 if ATEs[i].Empty and ATEs[i].lastProductType = partType
272                     targetATE := ATEs[i]
273                     exitloop
274                 end
275             next
276
277
278             -- If not possible, choose any empty ATE
279             if targetATE = void
280                 for var i := 1 to ATEs.dim
281
282           ----------------------------------------------------------------
283            -- PartH is NOT allowed to ATE4
284                 if SourceMU.Class = .UserObjects.PartH and ATEs[i] = ATE4
285                     continue
286                 end
287           ----------------------------------------------------------------
288
289                     if ATEs[i].Empty
290                         targetATE := ATEs[i]
291                         exitloop
292                     end
293                 next
294             end
295
296       -------------------------------
297         -- deliver part to ATE
298         -------------------------------
299
300             if targetATE /= void
301
302                 idx := ATEs.find(targetATE)
303
304                 agv.setRoute([markers[idx]])
305                 waituntil agv.destinationWasReached
306
307                 agv.cont.move(targetATE)
308
309                 taskDone := true
310
311             end
312
313
314             if taskDone
315                 exitloop
316             end
317
318         end
319
320     next
321
322 end
323
```

```
324
325
326 ------------------------------------------------
327 -- 4. Idle if nothing else happened
328 ------------------------------------------------
329
330 if not taskDone
331
332     agv.setRoute([IdleMarker])
333     waituntil agv.destinationWasReached
334
335     wait 1
336
337 end
338
339 end
```

A.2.3 ATEMethod

```
1  --ATEMethod-----------------------------------------------------------
2    --This method is connected to the Exit Control in each ATE.
3    --Therefore it runs every time a part exits the machine.
4    --The purpose of this method is to:
5      --Determine whether product PASSES or FAILS the quality check
6      --Returns fixture that has been used for the product
7  ------------------------------------------------------------------------
8
9  -- Quality check
10 var r : real := z_uniform(0,1)
11
12 if r > @.Yield
13     @.Quality := 1   -- FAIL_Y
14 else
15     @.Quality := 2   -- PASS_Y
16 end
17
18
19 -- Return fixture
20 if @.Class = .UserObjects.PartA
21     FixtureA += 1
22 end
23
24 if @.Class = .UserObjects.PartB
25     FixtureB += 1
26 end
27
28 if @.Class = .UserObjects.PartC
29     FixtureC += 1
30 end
31
32 if @.Class = .UserObjects.PartD
33     FixtureD += 1
34 end
35
36 if @.Class = .UserObjects.PartE
37     FixtureE += 1
38 end
39
40 if @.Class = .UserObjects.PartF
41     FixtureF += 1
42 end
43
44 if @.Class = .UserObjects.PartG
45     FixtureG += 1
46 end
47
48 if @.Class = .UserObjects.PartH
```

```
    FixtureH += 1
end

if @.Class = .UserObjects.PartI
    FixtureI += 1
end
```

A.2.4 MethodEntranceATE

```
--MethodEntranceATE-------------------------------------------------------------
  --This method is connected to the Entrance Control in each ATE.
  --Therefore it runs every time a part enters the machine.
  --The purpose of this method is to set the proessing time for the ATE.
--------------------------------------------------------------------------------

var mu := @

-- Get processing time from the part
var baseProcTime : time := mu.ProcessingTime

-- Set machine processing time
?.procTime := baseProcTime
```

A.2.5 Observer code FAIL buffer

```
param newValue: integer

-- Observer code

if ?.NumMU =5 then
    ?.cont.move;
    Unload := true;
elseif ?.NumMU =0 then
    Unload:=false;
end;
```

A.2.6 ExitMethodFAIL

```
--ExitMethodFAIL----------------------------------------------------------------------------
  --This method is connected to the Exit Control in both FAIL buffers.
  --The purpose of this method is tochecks whether unloading is allowed.
  --Unloading will be allowed when there are 5 parts in the buffer.
    --This method works in combination with an observer code placed on each FAIL buffer.
-----------------------------------------------------------------------------------------

if Unload then
    @.move;
end;
```

A.2.7 Counting

```
--Counting-----------------------------------------------
  --This methods counts the output per product type.
------------------------------------------------------

if @.name = "PartA"
    PartA_Count += 1
end
```

```

if @.name = "PartB"
    PartB_Count += 1
end

if @.name = "PartC"
    PartC_Count += 1
end

if @.name = "PartD"
    PartD_Count += 1
end

if @.name = "PartE"
    PartE_Count += 1
end

if @.name = "PartF"
    PartF_Count += 1
end

if @.name = "PartG"
    PartG_Count += 1
end

if @.name = "PartH"
    PartH_Count += 1
end

if @.name = "PartI"
    PartI_Count += 1
end
```

A.2.8 EndSim

```
--END SIM-----------------------------------------------
  --This method prints:
    --distance travelled by each AGV
    --output per product
------------------------------------------------------------

--Distance AGVs
var dist1 : real
var dist2 : real

dist1 := .UserObjects.Transporter:1.statTraveledDistance
dist2 := .UserObjects.Transporter2:1.statTraveledDistance

print("Transporter1 distance: " + to_str(dist1))
print("Transporter2 distance: " + to_str(dist2))

print("Total distance: " + to_str(dist1 + dist2))

--Print output per product
print("PartA produced: " + to_str(.Models.Model.PartA_Count))
print("PartB produced: " + to_str(.Models.Model.PartB_Count))
print("PartC produced: " + to_str(.Models.Model.PartC_Count))
print("PartD produced: " + to_str(.Models.Model.PartD_Count))
print("PartE produced: " + to_str(.Models.Model.PartE_Count))
print("PartF produced: " + to_str(.Models.Model.PartF_Count))

print("PartG produced: " + to_str(.Models.Model.PartG_Count))
print("PartH produced: " + to_str(.Models.Model.PartH_Count))
print("PartI produced: " + to_str(.Models.Model.PartI_Count))
```

A.3 Tecnomatix Plant Simulation Layout 2

This section of the appendix contains all the methods used in Tecnomatix Plant Simulation to program the simulation in Layout 2.

A.3.1 Init

```
--INIT-------------------------------------------------------------------
  --Start ONE independent process for every AGV
-------------------------------------------------------------------------

for var i := 1 to AGVPool.numMU --Loop through all AGVs in the pool
    var agv := AGVPool.mu(i) -- Access the AGV by its index in the pool

  --Start a new parallel process (call chain)

  --&ControlAGV = reference to AGV logic method
  --executeNewCallchain(agv) = run it independently

    &ControlAGV.executeNewCallchain(agv)
  --Each AGV gets its own controller loop all running simultaneously

next
```

A.3.2 ControlAGV

```
-----CONTROL AGV ----------------------------------------------------
  -- 1.Remove finished parts & refill immediately
      -- Refill from repair ATE13
      -- Refill from repair ATE46
      -- Refill from source next to same ATE
  -- 2.Feed other empty ATEs from their sources
  -- 3.Idle
-------------------------------------------------------------------

--AGV Object
param agv : object --This works for any AGV.

--Creates arrays that system can loop through
var ATEs := [ATE1, ATE2, ATE3, ATE4, ATE5, ATE6]
var FAILStations := [FAIL1, FAIL2, FAIL3, FAIL4, FAIL5, FAIL6]
var PASSStations := [PASS1, PASS2, PASS3, PASS4, PASS5, PASS6]
var Sources := [SourceA, SourceB, SourceC, SourceD, SourceE, SourceF]

--Variables
var mu : object
var taskFound : boolean

-------------------------------------------------------------------
-- 1. INITIALISATION
  --Runs once per AGV
  --Sets the initial location of the ATEs.
    --This is to make sure the routing table works later.
-------------------------------------------------------------------
if agv.initialized = void --Checks if the AGV has been initialised
    agv.initialized := 1 --Prevents this from running again

    agv.CurrentATE := ATE0 --Defines starting position for AGV
    agv.move(ATE0)
    agv.location := ATE0
end

-------------------------------------------------------------------
```

```
-- 2. MAIN LOOP (8H)
-------------------------------------------------------------------
while true

  taskFound := false

    ------------------------------------------------------------
    -- STOP AGVs AFTER 8h
    --> This section is toggled when robots are to be active for 24 hours
    ------------------------------------------------------------

  --Check simulation time has run for 8h, creates boolean var
  var shiftEnded := eventController.simTime >= 8:00:00

  --Before 8h, shiftEnded = false
    if shiftEnded

    --If AGV not at ATE0, get route to ATE0
        if agv.location /= ATE0

            agv.setRoute(getRoute(agv.CurrentATE, 0))
            agv.CurrentATE := 0

            waituntil agv.destinationWasReached

        end

        wait 1
        continue --Skip following logic in the loop

    end


    ------------------------------------------------------------
    -- 1. REMOVE FINISHED PARTS & REFILL IMMEDIATELY
    --loops through all ATEs
    --checks whether machine is ready
    ------------------------------------------------------------
    for var i := 1 to ATEs.dim

    --Checks if machine empty, if product finished and if product reserved
        if ATEs[i].cont /= void and ATEs[i].cont.finished and not ATEs[i].cont.reserved

            mu := ATEs[i].cont --Stores product
            mu.reserved := true --Reserves product

            agv.setRoute(getRoute(agv.CurrentATE, i)) --Gets route from DataTable
            waituntil agv.destinationWasReached

            agv.CurrentATE := i --Update ATE location

      --Pick up product
            if ATEs[i].cont = mu
                mu.move(agv)
            end


            ------------------------------------------------------
            -- PASS or FAIL
            ------------------------------------------------------
            if mu.Quality = 1
                mu.move(FAILStations[i])
            else
                mu.move(PASSStations[i])
            end

            mu.reserved := false --Remove reservation


            ------------------------------------------------------------------
```

```
            -- I. REFILL FROM REPAIR ATE1-3
        --Refills ATE using repaired parts that match current part in ATE
            ------------------------------------------------------------------

      --Checks the following:
          --if machine is connected to relevant repair area
          --if FromRepair buffer contains a product
          --if AGV is empty
         if i >= 1 and i <= 3 and FromRepair.cont /= void and agv.NumMU = 0

                var repairMU := FromRepair.cont --Stores repaired MU
                var allow : boolean := true --refill is allowed

        --Checks if repaired product is same type as previous product type
                if ATEs[i].lastProductType /= 0 and repairMU.ProductType /= ATEs[i].lastProductType

                    allow := false --Refill not allowed if product is not a match

                end

        --If product is a match, part is retrieved from repair and placed in ATE
                if allow

          --Reserves machine for refill from repair
          ATEs[i].reservedForRepair := true

                    agv.setRoute([FromFAIL])
                    waituntil agv.destinationWasReached

                    repairMU.move(agv)
                    waituntil agv.cont = repairMU

                    agv.setRoute(getRoute(agv.CurrentATE, i))
                    waituntil agv.destinationWasReached

                    repairMU.move(ATEs[i])

                    taskFound := true
                    exitloop

                end

            end


            ------------------------------------------------------------------
            -- II. REFILL FROM REPAIR ATE4-6
         --This section is alsomst identical to  the previous repair flow
         --The change is that another repair area is used for ATE4-ATE6
            ------------------------------------------------------------------
            if i >= 4 and i <= 6 and FromRepair2.cont /= void and agv.NumMU = 0

                var repairMU2 := FromRepair2.cont
                var allow2 : boolean := true

                if ATEs[i].lastProductType /= 0 and repairMU2.ProductType /= ATEs[i].lastProductType

                    allow2 := false

                end


                if allow2

                    ATEs[i].reservedForRepair := true

                    agv.setRoute([FromFAIL2])
                    waituntil agv.destinationWasReached

                    repairMU2.move(agv)
```

```
178                     waituntil agv.cont = repairMU2
179
180                     agv.setRoute(getRoute(agv.CurrentATE, i))
181                     waituntil agv.destinationWasReached
182
183                     repairMU2.move(ATEs[i])
184
185                     taskFound := true
186                     exitloop
187
188                 end
189
190             end
191
192
193             ------------------------------------------------------------------
194             -- III. REFILL FROM SOURCE NEXT TO SAME ATE
195         --Happens if there are no repair products that can refill machine
196             ------------------------------------------------------------------
197             if not taskFound
198
199                 var refillMU : object := void --No refill part selected
200
201         --Match source to machine
202         var c := i
203
204                 if Sources[c].cont /= void --Check source content
205                     refillMU := Sources[c].cont --Store source part
206                 end
207
208
209                 if refillMU /= void
210                     agv.setRoute(getRoute(agv.CurrentATE, c)) -- Get route to source
211                     waituntil agv.destinationWasReached
212
213                     refillMU.move(agv) --Load product
214
215                     agv.setRoute(getRoute(agv.CurrentATE, i)) -- Get route to machine
216                     waituntil agv.destinationWasReached
217
218                     refillMU.move(ATEs[i]) -- Deliver product
219
220                     taskFound := true
221                     exitloop
222
223                 end
224
225             end
226
227
228             taskFound := true
229             exitloop
230
231         end
232
233     next
234
235
236
237     ------------------------------------------------------------------
238     -- 2. FEED OTHER EMPTY ATEs FROM THEIR SOURCES
239     --This happens anfter the previous options have been exhausted
240     --Here the ATEs are searched in order to find which are empty
241     --The empty ones are refilled from their sources
242     ------------------------------------------------------------------
243     if not taskFound
244
245         for var b := 1 to Sources.dim
246
247       --Conditions:
```

```
248         -- Does buffer contain part?
249         -- Is machine empty?
250         -- Is part reserved?
251         -- Is machine reserved for refill?
252             if Sources[b].cont /= void and ATEs[b].Empty and not Sources[b].cont.reserved and not
    ATEs[b].reservedForRepair
253 
254                 mu := Sources[b].cont
255                 mu.reserved := true --Reserve product
256 
257                 agv.setRoute(getRoute(agv.CurrentATE, b))
258                 waituntil agv.destinationWasReached
259 
260                 agv.CurrentATE := b --Update AGV location
261 
262                 if Sources[b].cont = mu
263                     mu.move(agv)
264                 end
265 
266                 if agv.cont /= void
267                     agv.cont.move(ATEs[b])
268                 end
269 
270                 mu.reserved := false --Remove reservation
271 
272                 taskFound := true
273                 exitloop
274 
275             end
276 
277         next
278 
279     end
280 
281 
282     ------------------------------------------------------------------
283     -- 3. IDLE
284     -- Happens if no tasks exist
285     ------------------------------------------------------------------
286     if not taskFound
287 
288         if agv.location /= ATE0
289 
290       -- Get route back to ATE0
291             agv.setRoute(getRoute(agv.CurrentATE, 0))
292 
293             agv.CurrentATE := 0 --Update location
294 
295             waituntil agv.destinationWasReached
296 
297         end
298 
299         wait 1
300 
301     end
302 
303 end
```

A.3.3 getRoute

```
1 --GET ROUTE-----------------------------------------------------------------------------
2   -- This method looks up the correct path for the AGV in the routing table (DataTable)
3   --Then the path is converted into an array the AGV can follow
4 ------------------------------------------------------------------------------------------
5 
6 -- Returns route object array between two ATEs
7 param fromIdx : integer, toIdx : integer -> object[]
```

```

var route : object[]

var routeName :=
    "ATE" + to_str(fromIdx) + "-ATE" + to_str(toIdx)

var row : integer := 0 --Stores which table row has correct route

-- Find mathching row
  --Loops through all rows, searcing for route name
for var r := 1 to DataTable.YDim

    if to_str(DataTable[0,r]) = routeName

        row := r
        exitloop

    end

next


--Build array for AGV to follow
if row > 0

    for var c := 1 to DataTable.XDim

        if DataTable[c,row] /= void

            route.append(DataTable[c,row])

        end

    next

end

return route
```

A.3.4 ATEMethod

```
--ATEMethod-----------------------------------------------------------
  --This method is connected to the Exit Control in each ATE.
  --Therefore it runs every time a part exits the machine.
  --The purpose of this method is to:
    --Determine whether product PASSES or FAILS the quality check
    --Returns fixture that has been used for the product
-------------------------------------------------------------------------

-- Quality check
var r : real := z_uniform(0,1)

if r > @.Yield
    @.Quality := 1   -- FAIL_Y
else
    @.Quality := 2   -- PASS_Y
end


-- Return fixture
if @.Class = .UserObjects.PartA
    FixtureA += 1
end

if @.Class = .UserObjects.PartB
    FixtureB += 1
end
```

```
27
28 if @.Class = .UserObjects.PartC
29     FixtureC += 1
30 end
31
32 if @.Class = .UserObjects.PartD
33     FixtureD += 1
34 end
35
36 if @.Class = .UserObjects.PartE
37     FixtureE += 1
38 end
39
40 if @.Class = .UserObjects.PartF
41     FixtureF += 1
42 end
43
44 if @.Class = .UserObjects.PartG
45     FixtureG += 1
46 end
47
48 if @.Class = .UserObjects.PartH
49     FixtureH += 1
50 end
51
52 if @.Class = .UserObjects.PartI
53     FixtureI += 1
54 end
```

A.3.5 MethodEntranceATE

```
1  --MethodEntranceATE-----------------------------------------------------------
2    --This method is connected to the Entrance Control in each ATE.
3    --Therefore it runs every time a part enters the machine.
4    --The purpose of this method is to:
5      --set the proessing time for the ATE
6      --reserve fixture
7  ------------------------------------------------------------------------------
8
9  var mu := @
10
11 -- Get processing time from the part
12 var baseProcTime : time := mu.ProcessingTime
13
14 --Reserve fixture
15 if mu.Class = .UserObjects.PartA
16     FixtureA -= 1
17 elseif mu.Class = .UserObjects.PartB
18     FixtureB -= 1
19 elseif mu.Class = .UserObjects.PartC
20     FixtureC -= 1
21 elseif mu.Class = .UserObjects.PartD
22     FixtureD -= 1
23 elseif mu.Class = .UserObjects.PartE
24     FixtureE -= 1
25 elseif mu.Class = .UserObjects.PartF
26     FixtureF -= 1
27 elseif mu.Class = .UserObjects.PartG
28     FixtureG -= 1
29 elseif mu.Class = .UserObjects.PartH
30     FixtureH -= 1
31 elseif mu.Class = .UserObjects.PartI
32     FixtureI -= 1
33 end
34
35 -- Set machine processing time
36 ?.procTime := baseProcTime
```

A.3.6 Observer code FAIL buffer

```
param newValue: integer

-- Observer code

if ?.NumMU =5 then
    ?.cont.move;
    Unload := true;
elseif ?.NumMU =0 then
    Unload:=false;
end;
```

A.3.7 ExitMethod

```
--ExitMethod-----------------------------------------------------------------------------
  --This method is connected to the Exit Control in both FAIL buffers.
  --The purpose of this method is tochecks whether unloading is allowed.
  --Unloading will be allowed when there are 5 parts in the buffer.
    --This method works in combination with an observer code placed on each FAIL buffer.
------------------------------------------------------------------------------------------

if Unload then
    @.move;
end;
```

A.3.8 Counting

```
--Counting-------------------------------------------
  --This methods counts the output per product type.
-----------------------------------------------------

if @.name = "PartA"
    PartA_Count += 1
end

if @.name = "PartB"
    PartB_Count += 1
end

if @.name = "PartC"
    PartC_Count += 1
end

if @.name = "PartD"
    PartD_Count += 1
end

if @.name = "PartE"
    PartE_Count += 1
end

if @.name = "PartF"
    PartF_Count += 1
end

if @.name = "PartG"
    PartG_Count += 1
end

if @.name = "PartH"
    PartH_Count += 1
end

```

```
if @.name = "PartI"
    PartI_Count += 1
end
```

A.3.9 EndSim

```
--END SIM-----------------------------------------------------
  --This method prints:
    --distance travelled by each AGV
    --output per product
-------------------------------------------------------------

--Distance AGVs
var dist1 : real
var dist2 : real

dist1 := .UserObjects.Transporter:1.statTraveledDistance
dist2 := .UserObjects.Transporter:2.statTraveledDistance

print("Transporter1 distance: " + to_str(dist1))
print("Transporter2 distance: " + to_str(dist2))

print("Total distance: " + to_str(dist1 + dist2))

--Print output per product
print("PartA produced: " + to_str(.Models.Model.PartA_Count))
print("PartB produced: " + to_str(.Models.Model.PartB_Count))
print("PartC produced: " + to_str(.Models.Model.PartC_Count))
print("PartD produced: " + to_str(.Models.Model.PartD_Count))
print("PartE produced: " + to_str(.Models.Model.PartE_Count))
print("PartF produced: " + to_str(.Models.Model.PartF_Count))

print("PartG produced: " + to_str(.Models.Model.PartG_Count))
print("PartH produced: " + to_str(.Models.Model.PartH_Count))
print("PartI produced: " + to_str(.Models.Model.PartI_Count))
```

A.3.10 DataTable

	0	1	2	3	4	5	6	7	8
1	ATE1-ATE1	Marker1							
2	ATE1-ATE2	MarkerNav1	MarkerNav2	Marker2					
3	ATE1-ATE3	MarkerNav1	MarkerNav2	MarkerNav3	Marker3				
4	ATE1-ATE4	MarkerNav1	MarkerNav2	MarkerNav3	MarkerNav4	Marker4			
5	ATE1-ATE5	MarkerNav1	MarkerNav2	MarkerNav3	MarkerNav4	MarkerNav5	Marker5		
6	ATE1-ATE6	MarkerNav1	MarkerNav2	MarkerNav3	MarkerNav4	MarkerNav5	MarkerNav6	MarkerNav7	Marker6
7	ATE2-ATE1	MarkerNav2	MarkerNav1	Marker1					
8	ATE2-ATE2	Marker2							
9	ATE2-ATE3	MarkerNav2	MarkerNav3	Marker3					
10	ATE2-ATE4	MarkerNav2	MarkerNav3	MarkerNav4	Marker4				
11	ATE2-ATE5	MarkerNav2	MarkerNav3	MarkerNav4	MarkerNav5	Marker5			
12	ATE2-ATE6	MarkerNav2	MarkerNav3	MarkerNav4	MarkerNav5	MarkerNav6	MarkerNav7	Marker6	
13	ATE3-ATE1	MarkerNav3	MarkerNav2	MarkerNav1	Marker1				
14	ATE3-ATE2	MarkerNav3	MarkerNav2	Marker2					
15	ATE3-ATE3	Marker3							
16	ATE3-ATE4	MarkerNav3	MarkerNav4	Marker4					
17	ATE3-ATE5	MarkerNav3	MarkerNav4	MarkerNav5	Marker5				
18	ATE3-ATE6	MarkerNav3	MarkerNav4	MarkerNav5	MarkerNav6	MarkerNav7	Marker6		
19	ATE4-ATE1	MarkerNav4	MarkerNav3	MarkerNav2	MarkerNav1	Marker1			
20	ATE4-ATE2	MarkerNav4	MarkerNav3	MarkerNav2	Marker2				
21	ATE4-ATE3	MarkerNav4	MarkerNav3	Marker3					
22	ATE4-ATE4	Marker4							
23	ATE4-ATE5	MarkerNav4	MarkerNav5	Marker5					
24	ATE4-ATE6	MarkerNav4	MarkerNav5	MarkerNav6	MarkerNav7	Marker6			
25	ATE5-ATE1	MarkerNav5	MarkerNav4	MarkerNav3	MarkerNav2	MarkerNav1	Marker1		
26	ATE5-ATE2	MarkerNav5	MarkerNav4	MarkerNav3	MarkerNav2	Marker2			
27	ATE5-ATE3	MarkerNav5	MarkerNav4	MarkerNav3	Marker3				
28	ATE5-ATE4	MarkerNav5	MarkerNav4	Marker4					
29	ATE5-ATE5	Marker5							
30	ATE5-ATE6	MarkerNav5	MarkerNav6	MarkerNav7	Marker6				
31	ATE6-ATE1	MarkerNav7	MarkerNav6	MarkerNav5	MarkerNav4	MarkerNav3	MarkerNav2	MarkerNav1	Marker1
32	ATE6-ATE2	MarkerNav7	MarkerNav6	MarkerNav5	MarkerNav4	MarkerNav3	MarkerNav2	Marker2	
33	ATE6-ATE3	MarkerNav7	MarkerNav6	MarkerNav5	MarkerNav4	MarkerNav3	Marker3		
34	ATE6-ATE4	MarkerNav7	MarkerNav6	MarkerNav5	MarkerNav4	Marker4			
35	ATE6-ATE5	MarkerNav7	MarkerNav6	MarkerNav5	Marker5				
36	ATE6-ATE6	Marker6							
37	ATE1-ATE0	MarkerNav1	MarkerNav2	MarkerNav3	MarkerNav4	ATE0			
38	ATE2-ATE0	MarkerNav2	MarkerNav3	MarkerNav4	ATE0				
39	ATE3-ATE0	MarkerNav3	MarkerNav4	ATE0					
40	ATE4-ATE0	MarkerNav4	ATE0						
41	ATE5-ATE0	MarkerNav5	NavIdle	ATE0					
42	ATE6-ATE0	MarkerNav7	MarkerNav6	MarkerNav5	NavIdle	ATE0			
43	ATE0-ATE1	MarkerNav4	MarkerNav3	MarkerNav2	MarkerNav1	Marker1			
44	ATE0-ATE2	MarkerNav4	MarkerNav3	MarkerNav2	Marker2				
45	ATE0-ATE3	MarkerNav4	MarkerNav3	Marker3					
46	ATE0-ATE4	Marker4							
47	ATE0-ATE5	NavIdle	MarkerNav5	Marker5					
48	ATE0-ATE6	NavIdle	MarkerNav5	MarkerNav6	MarkerNav7	Marker6			
49	ATE0-ATE0	ATE0							
50	ATE1-ATE1	Marker1							
51	ATE2-ATE2	Marker2							
52	ATE3-ATE3	Marker3							
53	ATE4-ATE4	Marker4							
54	ATE5-ATE5	Marker5							
55	ATE6-ATE6	Marker6							

Table A.1: Routing table for paths in Layout 2, TPS

A.4 Specific simulation output data

A.4.1 Visual Components

ATEs	Working	Changeover	Idle Time	Blocked
ATE1	29.26%	4.17%	0.37%	66.20%
ATE2	29.26%	4.17%	0.46%	66.11%
ATE3	29.37%	4.17%	1.81%	64.65%
ATE4	29.26%	4.17%	0.36%	66.21%
ATE5	29.26%	4.17%	0.43%	66.14%
ATE6	52.97%	4.17%	0.15%	42.71%

(a)

ATEs	Working	Changeover	Idle Time	Blocked
ATE1	85.38%	12.57%	1.54%	0.51%
ATE2	89.95%	8.34%	1.32%	0.39%
ATE3	83.26%	12.51%	3.58%	0.65%
ATE4	75.21%	23.38%	1.13%	0.28%
ATE5	90.43%	8.34%	1.02%	0.21%
ATE6	95.31%	4.17%	0.46%	0.06%

(b)

Table A.2: Exact ATE utilisation percentages for Layout 1

ATEs	Working	Changeover	Idle Time	Blocked
ATE1	29.64%	4.17%	0.37%	65.82%
ATE2	29.40%	4.17%	0.31%	66.12%
ATE3	29.40%	4.17%	0.43%	66%
ATE4	29.40%	4.17%	0.33%	66.10%
ATE5	29.39%	4.17%	0.40%	66.04%
ATE6	53.00%	4.17%	0.15%	42.68%

(a)

ATEs	Working	Changeover	Idle Time	Blocked
ATE1	85.49%	12.51%	1.08%	0.92%
ATE2	90.43%	8.34%	0.79%	0.44%
ATE3	86.24%	12.51%	0.87%	0.38%
ATE4	90.33%	8.34%	1.10%	0.23%
ATE5	90.85%	8.34%	0.59%	0.22%
ATE6	95.56%	4.17%	0.21%	0.06%

(b)

Table A.3: Exact ATE utilisation percentages for Layout 2

Products	Passed	To repair/ Repairing	Processing
A	9	3	2
B	15	0	1
C	0	0	0
D	0	0	0
E	0	0	0
F	0	0	0
Total	24	3	3
-	-	-	-
G	8	1	2
H	1	0	1
I	0	0	0
Total	9	1	3

(a)

Products	Passed	To repair/ Repairing	Processing
A	18	8	0
B	24	2	0
C	16	0	0
D	7	0	1
E	19	5	2
F	0	0	0
Total	84	15	3
-	-	-	-
G	25	5	0
H	3	1	2
I	2	1	1
Total	30	7	3

(b)

Table A.4: Exact ATE product output for Layout 1

Products	Passed	To repair/ Repairing	Processing
A	12	0	2
B	0	0	0
C	9	0	2
D	0	0	0
E	0	0	0
F	0	0	0
Total	21	0	4
-	-	-	-
G	8	1	2
H	1	0	1
I	0	0	0
Total	9	1	3

(a)

Products	Passed	To repair/ Repairing	Processing
A	18	7	0
B	24	3	0
C	12	0	1
D	8	4	1
E	9	7	1
F	1	0	1
Total	72	21	4
-	-	-	-
G	23	6	1
H	3	1	2
I	7	6	1
Total	33	13	4

(b)

Table A.5: Exact ATE product output for Layout 2

Travel distance (km)	Cobot 1	Cobot 2	Total
Layout 1	4.293	1.321	5.614
Layout 2	2.017	3.250	5.267

Table A.6: Travel distance in km for each cobot during 24 hours

A.4.2 Tecnomatix Plant Simulation

Layout 1, 8h shift				
ATEs	Working	Changeover	Idle Time	Blocked
ATE1	29.17%	4.17%	0.17%	66.50%
ATE2	29.17%	4.17%	0.17%	66.50%
ATE3	29.27%	8.33%	0.48%	61.92%
ATE4	29.17%	4.17%	0.15%	66.51%
ATE5	29.17%	4.17%	0.15%	66.51%
ATE6	52.92%	4.17%	0.06%	42.85%

(a)

Layout 1 - 24h shift				
ATEs	Working	Changeover	Idle Time	Blocked
ATE1	82.37%	16.92%	0.54%	0.17%
ATE2	82.12%	17.28%	0.45%	0.15%
ATE3	76.81%	22.26%	0.79%	0.14%
ATE4	80.74%	18.54%	0.49%	0.24%
ATE5	91.11%	8.33%	0.37%	0.18%
ATE6	95.71%	4.17%	0.11%	0.01%

(b)

Table A.7: Exact ATE utilisation percentages for Layout 1

Layout 2 - 8h shift				
ATEs	Working	Changeover	Idle Time	Blocked
ATE1	29.17%	4.17%	0.03%	66.64%
ATE2	29.17%	4.17%	0.02%	66.64%
ATE3	29.17%	4.17%	0.04%	66.63%
ATE4	29.17%	4.17%	0.05%	66.62%
ATE5	29.17%	4.17%	0.05%	66.61%
ATE6	52.92%	4.17%	0.07%	42.85%

(a)

Layout 2 - 24h shift				
ATEs	Working	Changeover	Idle Time	Blocked
ATE1	83.32%	15.97%	0.04%	0.67%
ATE2	81.02%	11.19%	7.03%	0.76%
ATE3	78.63%	18.65%	2.38%	0.34%
ATE4	82.16%	14.85%	2.78%	0.22%
ATE5	91.35%	8.33%	0.07%	0.25%
ATE6	92.78%	7.08%	0.07%	0.07%

(b)

Table A.8: Exact ATE utilisation percentages for Layout 2

Output per product				
Part	**Layoutt 1 (8h)**	**Layout 1 (24h)**	**Layout 2 (8h)**	**Layout 2 (24h)**
A	9.55	20.30	8.85	21.30
B	23.25	24.95	0.00	21.25
C	0.00	15.55	12.80	15.85
D	0.00	6.20	0.00	7.95
E	0.00	10.95	0.00	6.70
F	0.00	0.00	0.00	0.35
Area X	**32.80**	**77.95**	**21.65**	**73.40**
G	9.20	24.75	9.40	24.75
H	0.90	3.25	0.75	2.95
I	0.00	2.95	0.00	4.75
Area Y	**10.10**	**30.95**	**10.15**	**32.45**
Total	**42.90**	**108.90**	**31.80**	**105.85**

Table A.9: Product output

Travel Distance (km)			
Layout	**Cobot 1**	**Cobot 2**	**Total**
1	2.30	1.17	3.47
1	2.02	1.86	3.88

Table A.10: Travel distance in km for each cobot during 24 hours

Working Time (h)				
ATE	**Layoutt 1 (8h)**	**Layout 1 (24h)**	**Layout 2 (8h)**	**Layout 2 (24h)**
ATE1	7	19.77	7	20
ATE2	7	19.71	7	19.45
ATE3	7.025	18.43	7	18.87
ATE4	7	19.38	7	19.72
ATE5	7	21.87	7	21.92
ATE6	12.7	22.97	12.7	22.27
Total	**47.725**	**122.13**	**47.7**	**122.23**

Table A.11: Total working time in hours per ATE

A.5 Experimental manager

This section contains the detailed results from the experimental manager report in Tecnomatix Plant Simulation.

A.5.1 Layout 1, 8h shift

Layout 1, 8h shift						
Output Name	**Output Value**	**Standard Deviation**	**Minimum**	**Maximum**	**Left Interval Bound**	**Right Interval Bound**
Output area X	32.8	1.5424	30	36	32.0781	33.5219
Output area Y	10.1	1.5526	7	13	9.3734	10.8266
ATE1 Working	0.2917	0	-	-	-	-
ATE1 Blocked	0.665	0.0004	0.6646	0.6658	0.6648	0.6651
ATE1 Idle	0.0017	0.0004	0.0009	0.0021	0.0015	0.0019
To repair area X	5.3	1.689	2	9	4.5095	6.0905
To repair area Y	2.9	1.5526	0	6	2.1734	3.6266
ATE2 Blocked	0.665	0.0002	0.6647	0.6654	0.6649	0.6651
ATE2 Idle	0.0017	0.0002	0.0012	0.0019	0.0016	0.0018
ATE2 Working	0.2917	0	-	-	-	-
ATE3 Blocked	0.6192	0.0033	0.6095	0.6203	0.6176	0.6207
ATE3 Idle	0.0048	9.40E-05	0.0047	0.0051	0.0048	0.0048
ATE3 Working	0.2927	0.0032	0.2917	0.3021	0.2912	0.2942
ATE4 Blocked	0.6651	6.59E-05	0.665	0.6653	0.6651	0.6652
ATE4 Idle	0.0015	6.59E-05	0.0014	0.0017	0.0015	0.0016
ATE4 Working	0.2917	0	-	-	-	-
ATE5 Blocked	0.6651	5.49E-05	0.6651	0.6653	0.6651	0.6652
ATE5 Idle	0.0015	5.49E-05	0.0014	0.0016	0.0015	0.0015
ATE5 Working	0.2917	0	-	-	-	-
ATE6 Blocked	0.4285	2.08E-05	0.4285	0.4286	0.4285	0.4285
ATE6 Idle	0.0006	2.08E-05	0.0006	0.0006	0.0006	0.0006
ATE6 Working	0.5292	0	-	-	-	-
ATE1 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE2 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE3 Changeover	0.0833	2.70E-09	0.0833	0.0833	0.0833	0.0833
ATE4 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE5 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE6 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE1 Working Time	07:00:00	0	-	-	-	-
ATE2 Working Time	07:00:00	0	-	-	-	-
ATE3 Working Time	07:01:30	04:37.01	07:00:00	07:15:00	06:59:20	07:03:40
ATE4 Working Time	07:00:00	0	-	-	-	-
ATE5 Working Time	07:00:00	0	-	-	-	-
ATE6 Working Time	12:42:00	0	-	-	-	-

Table A.12: Detailed report Layout 1, 8h shift

A.5.2 Layout 1, 24h shift

Layout 1, 24h shift						
Output Name	**Output Value**	**Standard Deviation**	**Minimum**	**Maximum**	**Left Interval Bound**	**Right Interval Bound**
Output area X	78.1	4.8979	67	85	75.8077	80.3923
Output area Y	30.95	2.7429	27	38	29.6663	32.2337
ATE1 Working	0.8237	0.0514	0.7422	0.9097	0.7996	0.8478
ATE1 Blocked	0.0017	0.0003	0.0013	0.0022	0.0015	0.0018
ATE1 Idle	0.0054	0.0009	0.0034	0.0067	0.005	0.0059
To repair area X	15.55	3.5165	9	21	13.9042	17.1958
To repair area Y	11.8	2.1176	9	16	10.8089	12.7911
ATE2 Blocked	0.0015	0.0002	0.0013	0.0018	0.0014	0.0016
ATE2 Idle	0.0045	0.0007	0.0036	0.006	0.0041	0.0048
ATE2 Working	0.8212	0.0457	0.7432	0.9101	0.7998	0.8426
ATE3 Blocked	0.0014	0.0001	0.0012	0.0017	0.0014	0.0015
ATE3 Idle	0.0079	0.0006	0.007	0.0089	0.0076	0.0082
ATE3 Working	0.7681	0.0629	0.6575	0.8645	0.7386	0.7975
ATE4 Blocked	0.0024	0.0003	0.002	0.0029	0.0023	0.0025
ATE4 Idle	0.0049	0.0004	0.0043	0.0055	0.0047	0.0051
ATE4 Working	0.8074	0.0275	0.7496	0.8454	0.7945	0.8203
ATE5 Blocked	0.0018	4.95E-05	0.0017	0.0019	0.0018	0.0019
ATE5 Idle	0.0037	0.0001	0.0035	0.0039	0.0036	0.0037
ATE5 Working	0.9111	0.0001	0.911	0.9113	0.9111	0.9112
ATE6 Blocked	8.06E-05	5.78E-06	6.93E-05	9.27E-05	7.93E-05	8.33E-05
ATE6 Idle	0.0011	4.09E-05	0.001	0.0012	0.0011	0.0012
ATE6 Working	0.9571	3.97E-05	0.9571	0.9572	0.9571	0.9571
ATE1 Changeover	0.1692	0.0515	0.0833	0.25	0.1451	0.1933
ATE2 Changeover	0.1728	0.0459	0.0833	0.25	0.1514	0.1943
ATE3 Changeover	0.2226	0.0631	0.125	0.3333	0.1931	0.2522
ATE4 Changeover	0.1853	0.0281	0.1462	0.244	0.1722	0.1985
ATE5 Changeover	0.0833	2.70E-09	0.0833	0.0833	0.0833	0.0833
ATE6 Changeover	0.0417	1.35E-09	0.0417	0.0417	0.0417	0.0417
ATE1 Working Time	19:46:08	01:14:04	17:48:43	21:49:55	19:11:28	20:20:48
ATE2 Working Time	19:42:30	01:05:53	17:50:15	21:50:32	19:11:40	20:13:20
ATE3 Working Time	18:26:00	01:30:37	15:46:48	20:44:54	17:43:36	19:08:25
ATE4 Working Time	19:22:37	39:40.26	17:59:24	20:17:24	19:04:03	19:41:11
ATE5 Working Time	21:52:02	8.9838	21:51:47	21:52:15	21:51:57	21:52:06
ATE6 Working Time	22:58:14	3.4267	22:58:10	22:58:23	22:58:13	22:58:16

Table A.13: Detailed report Layout 1, 24h shift

A.5.3 Layout 2, 8h shift

Layout 2, 8h shift						
Output Name	**Output Value**	**Standard Deviation**	**Minimum**	**Maximum**	**Left Interval Bound**	**Right Interval Bound**
Output ATE1	12.8	0.4103913408	12	13	12.60793094	12.99206906
Output ATE2	4.7	1.128576187	2	6	4.171810085	5.228189915
Output ATE3	4.15	0.8750939799	3	6	3.74044341	4.55955659
Output ATE4	4.6	1.142481141	1	6	4.065302367	5.134697633
Output ATE5	4.75	1.019545823	3	6	4.272837867	5.227162133
Output ATE6	0.8	0.4103913408	0	1	0.6079309401	0.992069059
To repair ATE1	0.2	0.4103913408	0	1	0.007930940147	0.392069059
To repair ATE2	1.3	1.128576187	0	4	0.7718100854	1.828189915
To repair ATE3	1.85	0.8750939799	0	3	1.44044341	2.25955659
To repair ATE4	1.4	1.142481141	0	5	0.8653023667	1.934697633
To repair ATE5	1.25	1.019545823	0	3	0.7728378669	1.727162133
To repair ATE6	0.2	0.4103913408	0	1	0.007930940147	0.392069059
ATE1 Blocked	0.6664047783	0	-	-	-	-
ATE1 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE1 Idle	0.0002618883484	0	-	-	-	-
ATE1 Working	0.2916666667	0	-	-	-	-
ATE2 Blocked	0.6664426615	1.37E-08	0.6664426615	0.6664426615	0.6664426551	0.6664426679
ATE2 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE2 Idle	0.0002240051814	0	-	-	-	-
ATE2 Working	0.2916666667	0	-	-	-	-
ATE3 Blocked	0.6663069032	0	-	-	-	-
ATE3 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE3 Idle	0.0003597634873	0	-	-	-	-
ATE3 Working	0.2916666667	0	-	-	-	-
ATE4 Blocked	0.6662018463	1.67E-08	0.6662018463	0.6662018463	0.6662018385	0.6662018541
ATE4 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE4 Idle	0.0004648203654	1.34E-11	0.0004648203654	0.0004648203654	0.0004648203591	0.0004648203716
ATE4 Working	0.2916666667	0	-	-	-	-
ATE5 Blocked	0.6661277849	0	-	-	-	-
ATE5 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE5 Idle	0.0005388817358	0	-	-	-	-
ATE5 Working	0.2916666667	0	-	-	-	-
ATE6 Blocked	0.4285083527	4.83E-09	0.4285083527	0.4285083527	0.4285083504	0.428508355
ATE6 Changeover	0.04166666667	1.35E-09	0.04166666667	0.04166666667	0.04166666603	0.0416666673
ATE6 Idle	0.0006583139578	0	-	-	-	-
ATE6 Working	0.5291666667	0	0	0	0	0
ATE1 Working Time	07:00:00	0	-	-	-	-
ATE2 Working Time	07:00:00	0	-	-	-	-
ATE3 Working Time	07:00:00	0	-	-	-	-
ATE4 Working Time	07:00:00	0	-	-	-	-
ATE5 Working Time	07:00:00	0	-	-	-	-
ATE6 Working Time	12:42:00	0	-	-	-	-

Table A.14: Detailed report Layout 2, 8h shift

A.5.4 Layout 2, 24h shift

Layout 2, 24h shift						
Output Name	**Output Value**	**Standard Deviation**	**Minimum**	**Maximum**	**Left Interval Bound**	**Right Interval Bound**
Output ATE1	24.6	2.458069418	18	28	23.4495881	25.7504119
Output ATE2	32.3	4.824280694	20	38	30.04216713	34.55783287
Output ATE3	16.7	3.771081096	6	22	14.93507972	18.46492028
Output ATE4	17.2	3.088177797	12	23	15.7546883	18.6453117
Output ATE5	12.8	1.576137851	10	15	12.06234478	13.53765522
Output ATE6	2.65	0.8127277009	1	4	2.269631727	3.030368273
To repair ATE1	3.35	1.460893742	1	6	2.666280682	4.033719318
To repair ATE2	5.2	2.525657809	1	11	4.017955759	6.382044241
To repair ATE3	6.15	1.531253357	2	8	5.433351369	6.866648631
To repair ATE4	9.85	2.906888371	4	16	8.489534364	11.21046564
To repair ATE5	3.3	1.592746717	1	6	2.55457159	4.04542841
To repair ATE6	0.7	0.8013147092	0	2	0.3249731719	1.075026828
ATE1 Blocked	0.006661788298	0.0004038785221	0.005476665065	0.006941775052	0.006472767332	0.006850809265
ATE1 Changeover	0.1597303029	0.05983513862	0.125	0.3333333333	0.131726596	0.1877340098
ATE1 Idle	0.0003882014492	0.0001979588793	0.0002618883484	0.0008934538523	0.0002955538417	0.0004808490566
ATE1 Working	0.8332197074	0.05963637209	0.6602965477	0.8678445395	0.8053090261	0.8611303887
ATE2 Blocked	0.007606668455	0.0008598137341	0.004824808139	0.008098600609	0.007204263241	0.00800907367
ATE2 Changeover	0.1118959126	0.05434696983	0.08333333333	0.2441382548	0.08646074778	0.1373310775
ATE2 Idle	0.07026967002	0.04226580384	0.0004949824611	0.0963**2**108357	0.05048866492	0.09005067512
ATE2 Working	0.8102277489	0.02285426007	0.75	0.8541666667	0.7995316259	0.8209238719
ATE3 Blocked	0.003390116752	0.0006184703652	0.001784029676	0.00416**8**103654	0.003100663711	0.003679569793
ATE3 Changeover	0.1864709165	0.06131214301	0.04166666667	0.2916666667	0.1577759503	0.2151658828
ATE3 Idle	0.02382577435	0.101842494	0.0003597634873	0.4565018265	-0.02383798005	0.07148952875
ATE3 Working	0.7863131924	0.0844177354	0.5000474771	0.870496417	0.746804476	0.8258219087
ATE4 Blocked	0.00216203544	0.0001596587438	0.001852701843	0.002366284228	0.002087312848	0.002236758032
ATE4 Changeover	0.1484547054	0.05433458023	0.08333333333	0.2083333333	0.1230253391	0.1738840717
ATE4 Idle	0.0277980506	0.03819407985	0.0008186489351	0.08311750788	0.00992267098	0.04567343021
ATE4 Working	0.8215852086	0.0276142768	0.7885269545	0.872**9**166667	0.8086613292	0.8345090879
ATE5 Blocked	0.002478261286	0.0002046511362	0.002322457601	0.002959554809	0.002382481606	0.002574040967
ATE5 Changeover	0.08333333333	2.42E-09	0.08333333333	0.08333333333	0.0833333322	0.08333333446
ATE5 Idle	0.0006525720083	0.0001564940237	0.0005388817358	0.0009936428257	0.0005793305506	0.0007258134659
ATE5 Working	0.9135358334	0.0002661430839	0.9129494326	0.9137983664	0.9134112746	0.9136603922
ATE6 Blocked	0.0006794226323	9.90E-05	0.0006023570532	0.0008183718747	0.0006330676923	0.0007257775723
ATE6 Changeover	0.07083333333	0.04078004041	0.04166666667	0.125	0.05174768692	0.08991897975
ATE6 Idle	0.0006909871511	4.57E-05	0.0006583139578	0.0007516659386	0.0006696069563	0.0007123673458
ATE6 Working	0.9277962569	0.0409246487	0.8734299622	0.9570726623	0.9086429317	0.9469495821
ATE1 Working Time	19:59:50	01:25:53	15:50:50	20:49:42	19:19:39	20:40:02
ATE2 Working Time	19:26:44	32:54.6081	18:00:00	20:30:00	19:11:20	19:42:08
ATE3 Working Time	18:52:17	02:01:34	12:00:04	20:53:31	17:55:24	19:49:11
ATE4 Working Time	19:43:05	39:45.8735	18:55:29	20:57:00	19:24:28	20:01:42
ATE5 Working Time	21:55:29	22.9948	21:54:39	21:55:52	21:55:19	21:55:40
ATE6 Working Time	22:16:02	58:55.8896	20:57:44	22:58:11	21:48:27	22:43:36

Table A.15: Detailed report Layout 2, 24h shift

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY