Eindhoven University of Technology
Department of Mechanical Engineering

Inalfa Roof Systems
Department of Global Advanced Manufacturing Engineering

# Simulation Based Modeling of a High Flexible TLES Roof System Production Line

J.T van Boggelen
*Supervisors Inalfa:*
H. Bastiaanse, M. Coenen
*Supervisors TU/e:*
A.A.J. Lefeber, I.J.B.F. Adan
DC 2017.071

August 11, 2017

# Preface

This thesis contains the results of my master graduation project at the department of Mechanical Engineering at Eindhoven University of Technology. This project is performed to investigate the added value of simulation at Inalfa Roof Systems and to investigate the effects of changes in Key Performance Indicators on the performance of a production line.
For this, a research was conducted into the KPI's to research how current production lines are performing. Furthermore, a concept line was designed so a simulation model could be built for the designed experiments. These experiments are used to show the effects of KPI's on the performance of a production line.

I would like to thank Erjen Lefeber and Ivo Adan for their support during my graduation project. Furthermore, I would like to thank Alp Akay as the additional member of my graduation committee. I would like to thank Hans Bastiaanse as my supervisor at Inalfa Roof Systems for his support during a difficult time. I am grateful for the support and the profound interest that Michel Coenen showed for the research and effort he made for me. Furthermore, I would like to thank Clement Snchault for the support and knowledge he gave me about the company. Also my gratitude to Bo Zhou, who helped and supported me with the simulation software used at Inalfa Roof Systems. Finally, I would like to thank my family, friends and especially my girlfriend for the mental support, patience and interest in my research.

Enjoy reading my graduation thesis.

Jasper van Boggelen

Eindhoven, August 2017

# Abstract

With increasing technology in the engineering world, the comprehensibility of production systems gets tougher and requires more work and time to design and understand. Inalfa Roof Systems faces similar problems with more complex products and an increasing demand in quality and quantity from their customers. The aim of this project is to investigate what part simulation can have in this development and how simulation can be used to gain more insight in the performance of a production line. Therefore, Key Performance Indicators are used to indicate how the line performs and what the behaviour of the production line is. This project also aims to show the effects of changes in KPI's on the performance and behaviour of a production line.

First, the KPI's at current production lines were investigated to determine what experiments could be done on the simulation model. The KPI research showed that the targets set and the used parameters in the design phase were higher than the actual performance on the production lines. It also should that some variables used in the design phase were difficult to investigate and therefore not clear if used correctly. To show the impact of these differences, some experiments were designed. These experiments are performed on a simulation model that is based on a high flexible concept line. This concept line is designed so that most of the TLES roof systems could be built on the concept line.

The research shows that the differences in the KPI's can have a big influence on the deliverability of the roof systems. The difference in quality between target and practice can result in throughput decreases of 10-15%. Furthermore, the research shows that effects as variance in the process times of workstations can have a big effect on the performance of the production line and should not be neglected as currently is done. Even a small variance in the process times already has a significant effect on the throughput of the production line. Furthermore, the research gave insight into effects such as blocking and starvation, which are relatively easy to show using simulation. The added value of using simulation in the design phase was increasingly revealed by this research as well as the next steps into implement simulation into the design phase of new production lines.

# Contents

# Chapter 1

# Introduction

With increasing technology in the engineering world, the comprehensibility of production systems gets tougher and requires more work and time to design and understand. One method that is used to understand production systems better is to use simulation. Simulation can be used to understand the behaviour of a system or to evaluate strategies to operate a production system. Simulation is the process of designing a model which is based on a real or an imagined system which is tested by experiments or case studies. The model is usually a real production system of a system in design, but can also be a concept system that is used to experiment with interesting scenarios. At Inalfa Roof Systems the use of simulation at the process engineering department is mostly limited to experiments based on upgrades of existing production lines or troubleshooting in current production lines. This graduation project shows the possibilities for simulation in a company like Inalfa Roof Systems, such as understanding the behaviour of a complex production system and getting more insight on the design variables of the production line.

## 1.1 Inalfa Roof Systems

Inalfa Roof Systems is one of the world's biggest providers of vehicle roof systems. Inalfa designs, develops, and manufactures sunroofs and open-roof systems for the major OEM's in the automotive industry. Inalfa is an independent global player with engineering centres on three major continents. Currently, Inalfa is supplying to all premium OEM's, with a market share of 24%. With a doubling of the turnover from 500 million to 1 billion in 5 years(2011-2016) and an expected doubling to 2 billion by 2020, Inalfa is growing strong within the market. This growth is achieved with market share as well as the growing market for roof systems. Regarding FTE's the predicted increase will result in a doubling from 4000 FTE to 8000 FTE by 2020 worldwide. The products that are produced by Inalfa Roof Systems can be divided into six categories. Inslider(IS), topslider(TS), exterior slider(ES), fixed panel(FP), sun blinds and truck hatch roof systems(TVS). Some of these categories can be divided into two groups, the top and bottom loaded systems (TL and BL). For instance, a bottom loaded top slider is shortly noted as a BLTS system. Within the company, the growth of Top Loaded Exterior Slider(TLES) roof systems is seen, and the prediction is that the share of TLES roof systems will grow in the total market. Therefore, among other reasons, the focus of this research is on the TLES roof systems.

## 1.2 Technical background

The TLES roof systems are part of the exterior slider roof systems. These systems can consist of either one or two glass panels. If the roof system consists of one panel, this panel can move over the roof skin of the vehicle. If the roof system consists of two panels, the front panel will move over the rear panel and possibly the roof of the vehicle. The exterior sliders can be made as a bottom loaded or top loaded system. This refers to the assembly of the roof system with the vehicle, if the system is assembled from the top of the bottom. The Top Loaded Exterior Sliders(TLES) roof systems are the only types of roof systems that Inalfa produces, that are used in this research. Currently, there are five

TLES roof system production lines that are being used or are currently developed by Inalfa Europe. These production lines can produce multiple roof systems per line. These production lines can have different methods to assemble the roof systems. Herein three different methods can be distinguished. The first assembly method is used in the D7A, Honda/JLR and MFA2/MRA production lines. This assembly group can be considered as the 'general' roof system group.
The D7A line is an older line of which the roof systems can also be built on the Honda/JLR line. However, the D7A line has a main conveyor line from which the carriers with the roof systems can be pulled into the system. For assembly on the bottom side of the roof system, the roof system and carrier have to be turned manually. This requires extra labour and therefore time. The MFA2/MRA line has the same principles and build-up as the Honda/JLR line, however, the building space was even more limited than for the latter.
The second roof system group exists of roof systems with assembly methods similar to the VS20 line. This product group is not included in the project and therefore neglected for the feasibility further in the project.
The third roof system group is based on the stacking principle. The stacking principle is based on Design For Assembly(DFA) and is used to produce more subassemblies which are married to create the final product. This principle creates the possibility to produce the sub-assemblies independent and therefore have parallel assemblies in the production line. In Europe, the Audi/Porsche line is the only line that uses this principle. In Mexico, a new line is built for the Audi A7 and VW Jetta, which uses the same principle.

## 1.3 Problem definition

The TLES production lines have a large market share in the current "gathering" of production lines of Inalfa Europe. However for only 2 of the 5 currently running TLES production lines a simulation was made. Also, these simulations were only made after the lines were already built and showed issues with the throughput and bottleneck stations. The simulations also showed the results of the Key Performance Indicators (KPI's), that are used to determine how good or bad a production line is running. Some of these problems could have been shown by simulation in the design phase of the line as these problems were not dependent on the quality of the build of the production line but on the behaviour and controls of the line.
In addition, during the design phase of a production line, a set of assumptions is used to design the line. Of these assumptions, the Quality, Performance and Availability are the most important. These variables are used to determine the cycle time of the designed production line. Currently, these values are set to the standard of a "World Class" company, which results in a lower required cycle time that the production line needs to operate. However, within the company, the consent exists that the standard is set too high and that in practice this is barely reached. Next to this, the neglection of production process as changeover can lead to problems during production on a production line.

### 1.3.1 Research Question

Due to increasing complexity of production lines and therefore less room for common sense and ad-hoc solutions, results to more complications and overlooked problems when a production line is designed and build. Therefore, increasing the work done by simulation can decrease the problems during the build and production of a production line. However, this research is not based on the introduction of a new line. The graduation project focuses on experimenting with a concept production line and the assumptions made in the design process. The graduation report visualises the effects of the assumptions and shows the effects on the KPI's. To experiment with the KPI's, a concept production line is created. The production line is a high flexible line and should be able to produce all roof systems that are currently produced on the Honda/JLR line in Venray and the Audi/Porsche line in Wrzesnia(Poland). Therefore, the concept line could be able to produce more than 85% of all TLES roof system types, if small adjustments are taken into account for other roof systems from the same groups. The high flexibility of this line is chosen to optimise the effect of some KPI's and to make the

research viable for most TLES roof systems. The main research question for the project is: *How to model the effect of the KPI's on a high flexible TLES production line by simulation?*
To experiment en visualise the simulation for the KPI's the three mainly tested KPI's are explained and research question regarding those KPI's are stated.

### 1.3.2 Quality

The Quality component contains the effects of the quality of production. This goes for individual workstations as well as the complete line. The number of roof systems that are tested OK at the EOL are counted towards the FTT. The FTT is the First Time Through percentage and shows the number of roof systems that have been completed on the first time tested. Roof systems that are rejected or need rework are counted against the FTT. To further improve the Quality aspect of the line the following questions are used:

- What is the targeted quality?
- How to measure this quality?
- What influence does improving the quality have on the Deliverability component?
- What countermeasures have to be taken with rework handling as consequence of Quality?

### 1.3.3 Availability

The Availability rate describes the relationship between the time there was a demand for the equipment (potential production time) and the time the equipment was available (actual production time). The availability rate can be calculated for both the individual station as well as for the complete production line. The difference between the potential and actual production time is caused by unexpected delays, such as the breakdown of machines and waiting for parts. Expected delays, such as paid breaks and training, should not be included in this difference but already be deducted from the potential production time. From this the following relevant questions can be asked:

- How does downtime (technically / organizationally) affect the availability of the line?
- What influence would the implementation of buffers have on the availability?
- How do the availability and the mean time to repair influence the Deliverability?

### 1.3.4 Flexibility

The flexibility of the line is determined by the number of different roof systems the production line can produce. These different roof systems can be produced from different roof systems groups, which have different assembly methods and sequences, or from the same roof system group but with different dimensions and features. The flexibility of the line has to be determined at the begin of designing the line as this can influence the assembly sequence. The following questions can be asked to improve the design of the line for flexibility:

- How to make the line flexible for different roof systems?
- What to do to implement different assembly methods on one line?
- What time to reserve for changeover and what influences do the different changeover times have on the behaviour of the line?
- How does the changeover depend on the batch size and stock?

Using the for these three KPI's the experiments can be determined to show the effects of the KPI's on a production line. To run these experiments a simulation model has to be build. This simulation model should be based on a high flexible TLES production line that can produce a large part of the TLES roof system types. As there is no current production line capable of this demand, a new concept production line is designed for this research. This concept line is then used to build a simulation model to run the experiments on. These experiments will show the effects of the KPI's on the production line and thereby show the use of simulation at Inalfa Roof Systems.

## 1.4 Report outline

This report shows the steps that are required to make a good simulation to experiment with and visualise the effect of the KPI's on the production line. In Chapter 2 the prior research of the KPI's is performed. To do a good simulation, it is required to have good and valid data or make assumptions that are based on reality. For the Quality KPI, an analysis is made of the FTT and STT results of current lines and compared to the targets that are set. Also for flexibility, the current TLES production lines are compared and researched for different changeover systems and flexibility systems. For the Availability KPI, the individual workstations of the lines are compared and a strategy is determined to calculate the overall production line availability. In Chapter 3 the layout of the concept production line is discussed. For this, the two current production lines that are used are compared and the basic assembly methods are compared. From these a general design is proposed, where the End Of Line (EOL) and repair sections are mentioned as the sections require extra attention. In Chapter 4 the conversion between the layout of the concept production line and the simulation model is discussed. The basic principles of the model are explained here, as well as the different sections of the concept line. Finally, the simulations methods and the required assumptions are described. With the discussed simulation the experiments can be run, whereby the results are shown in Chapter 5. First, a base result is set as a reference, that is used for comparison of the of the experiments. The experiments are based on the research done for the KPI's, and some experiments are done for case studies. Finally, the influence of variance is discussed as this is often neglected in process engineering. To finish the report, a conclusion is made, and further research and recommendations are given based on experience gathered during the graduation and based on the graduation project.

# Chapter 2

# KPI Research

In this chapter, the research of the KPI's is discussed, and implications for the simulation are proposed. In Section 2.1 the role of the KPI's is discussed as well as the used KPI's in design and why the further mentioned KPI's are researched. In Section 2.2 the measurability of the Quality KPI is discussed. Furthermore, the research is mentioned that is based on the data available at Inalfa. The same is done for the Availability KPI, that is discussed in Section 2.3. Finally, the research for the Flexibility KPI is discussed in Section 2.4. Herein the possibilities for flexible fixtures are discussed, and the changeover strategies of different production lines are discussed. The implementation of the research in the simulation and experiments is discussed as last.

## 2.1 KPI's in design

The Key Performance Indicators(KPI's) are used to track the set goals of production performance. [1, 2] As a result of this the engineers and managers can quickly have an overall view of the states of a production process and easily evaluate the performance of the process. This can be used to quickly determine the problem in the process and decrease the negative effects on the output.
Within Inalfa Roof Systems a few KPI's are used to track the performance of the line. Some of the KPI's are set as a global standard and then compared with the real situation as were other are set per production line and then used as a reference. Furthermore, a KPI is set as an indicator, rather than a target or goal. Therefore, it was important to determine how the KPI's could be measured and what information was available for the data of the KPI's on current production lines. Furthermore, it was important to determine on which KPI's more research should be done, given the limitations of time for this research. Therefore the selection of KPI's to be researched is based on the expected added value for the simulation experiments and the expected time to research and implementation in the simulation. Currently, the following KPI's are used by Inalfa Roof Systems:

- Quality
- Availability
- Cycle time
- Deliverability
- Bill of Labor
- Flexibility
- Performance

From this list, the Quality KPI is used in two forms at Inalfa Roof Systems. First as a percentage form, where the ratio between good delivered parts and scrap parts are given. The second is the ratio of good/bad tested roof systems. The first will from now on be called the scrap ratio. The scrap ratio is also used in combination with the Availability and Performance KPI's. The product of these 3

KPI's gives the Overall Equipment Effectiveness(OEE). The term OEE was first mentioned by Seiichi Nakajima [3]as part of Total Productive Maintenance concept. It is used to evaluate the utilisation effectiveness of a production process. The management of Inalfa Roof Systems has set the target for an OEE at "World Class"[4]. Whether the targets for the OEE are reasonable is not part of this project as this would require too much detailed research in the scrap ratio and Performance KPI. The scrap ratio research would include a detailed part flow analysis as were each part stays after parts are reduced to scrap. The performance KPI measures the output of the production line based on the target output determined by the ideal cycle time. The research in the performance KPI had no preference as the results of this would be difficult to use in the simulation experiments. Also, there was no priority from Inalfa Roof Systems to put more research in this KPI.

The takt time is the desired time between to products to finish[5]. This can either be desired by the producer as desired forecast or by demand by the customer. As the demand is set by the customer the takt time is based on the demand. To reach the demand the production process should run at takt time at every production step. However, due to the ineffectiveness of the production process, the required cycle at each production step should be faster than the takt time. The ratio between the desired takt time and the required takt time is the OEE, which gives the efficiency of the production process. From here the required takt time is called the OEE takt time. The OEE takt time should also be the maximum for the cycle time of the workstations. The workstations, therefore, need to be designed such that the cycle time does not exceed the OEE takt time. The cycle time is mainly used as a design variable to check whether the workstations are designed within the given OEE takttime. During production the KPI is used to compare if the workstations are working fast enough. Therefore, the cycle time is not studied further as this would go into too much detail of the actual process of the workstations.

The Deliverability KPI is the percentage of roof systems delivered to the customer on time. The target for the deliverability is to deliver all roof systems on time to the customer, resulting in a deliverability KPI with a percentage of 100. This KPI is mostly used as a result KPI to analyse the production line. This is also done in this research where the deliverability is used to compare the experiments of the other researched and simulated KPI's.

Lastly, the Bill of Labor (BOL) is the number of required man-hours to produce one product, in this case, one roof system[5]. The BOL is made for every workstation to calculate the required manual work time at that station. This is used to calculate the required number of operators to run the production line. The BOL KPI is not researched because the researched line is a concept line where it is not known how big the Bill of Labor will be. Besides, the research of the BOL would require a huge amount of time with very little influence on the simulation and no experiments to simulate. The BOL is merely a result of the design decisions.

The research of the other KPI's, Availability, Quality, and Flexibility, are discussed below as they have more subject matter.

## 2.2 Quality

The quality KPI that is researched and discussed here is not the earlier called scrap ratio, but the percentage of roof systems that is tested good or OK. These tests are performed at the End of Line(EOL). During the tests, the roof systems are checked if they meet the requirements set by the customer. If the requirements are met, the roof system is finished and send to unloading. If not met the roof system is sent to the repair stations. After the roof system is repaired, the roof can be tested at the repair station or send back to the line and tested again at the EOL, depending on the design of the line. This choice, depending on the line decision, is discussed in Chapter 3. If the roof system is tested for the second time, the same outcomes are possible as for the first time testing. Because of this retesting, a roof system could be tested many times, but in practice, only a few ($<0.1\%$) are tested more than three times. Therefore, in this research, the number of times tested is limited to three times. If tested not OK (NOK) the roof system is sent to scrap and disassembled and good

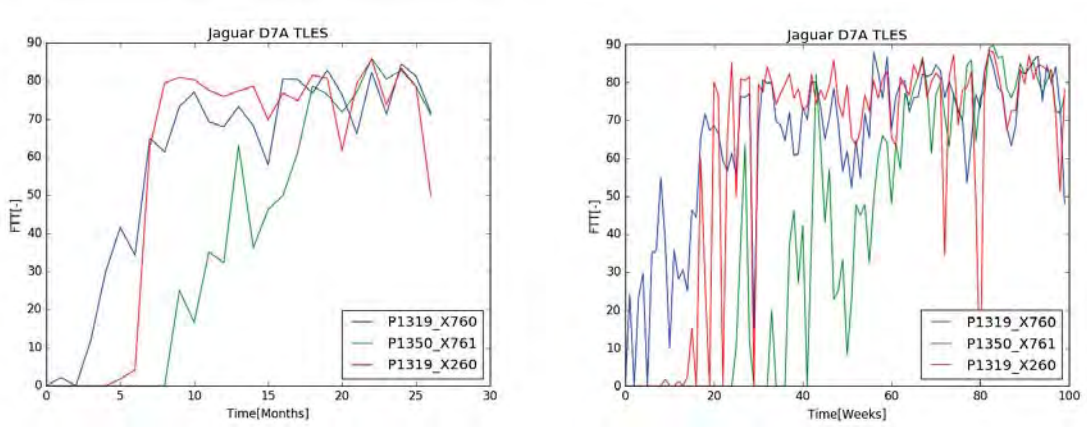
parts are reused in the process. The Quality KPI is measured for each time the roof systems are tested. The First Time Through(FTT) gives the percentage of OK tests at the first time of testing. The Second Time Through(STT) is given by the second time of testing. The third time of testing is not measured and is assumed to be equal to the STT.

The management of Inalfa Roof Systems has set the target for the FTT to 85% for the first year and 95% for subsequent years[6]. There is no global data to either confirm or reject if the targets that are set are feasible. Therefore, in this research the data is analysed from the production lines and compare this to the targets set and check the feasibility of the targets. For all production lines, the test results of the EOL are saved into databases. There are several databases for each line, which hold different information, such as station cycle times, measurement data and final test results. From this last database, the data was filtered to give information on each first test for each roof system at the production lines. The STT is not researched as there are more influences in this, such as the online repair and retesting and the more difficult filtering of the data.

### 2.2.1 Production line FTT analysis

There are two different ways to do an analysis of the FTT that both can be used but for various reasons. First, a batch average can be taken to analyse the FTT. This way would have the benefit that the FTT is calculated over a complete batch of parts and therefore the quality of that part would be the main influence on the quality and therefore the FTT. Secondly, the FTT can be averaged over a time period, which gives a better view of how the FTT develops over time. As there is no information on which batch of parts was used at what time, this method can not be used. Also, this research is focused on the global FTT analysis and not on the part analysis. Therefore the analysis is done with week and month averages. These give enough data points to do analysis but are not as fluctuating as day or shift averages.

As the production lines start up, there will be a lot of startup problems, and the operators at the line still lack the experience for the new roof systems. This also affect the FTT and STT of the production line. During the first year of production, the FTT is expected to go up and reach the target of 85% by the end of the first year. Also it is expected that the FTT will still show some variance over the weeks. At the following years, the FTT will grow to 95%. In Figure 2.1 the week and month averages of the FTT from the Start Of Production(SOP) until December 2016 are plotted for the D7A production line. As can be seen in the figures, the development of the FTT over time is fluctuating a lot. Also,



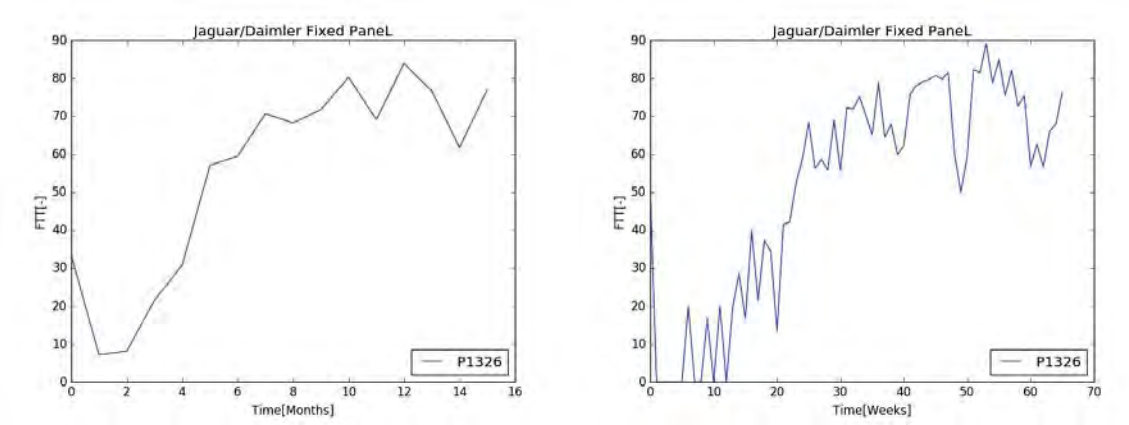

**(a)** *FTT of the D7A production line with month averages.*

**(b)** *FTT of the D7A production line with week averages.*

**Figure 2.1:** *FTT development from SOP until December 2016 of the D7A production line.*

each roof system project has a certain time where the project is started up, and the FTT is increasing. The X260 roof systems were started nine weeks after SOP, and the X761 was started 22 weeks after

SOP. Figure 2.1 also shows that the targets set by management are rarely met. If the data is checked for the D7A TLES production line, the weekly FTT only reaches 85% in 2 weeks of the 50 weeks after the first startup year and the highest FTT of a week at 88.5 % in week 85. The average over all weeks after the first year of production is only 73.6%. This is lower than the target set at the start of that year, 85% and does not even come near the 95% target set towards the end of the year. The same can be done for the other TLES production line that is working at full capacity, the Jaguar-Daimler line. On this production line, the Daimler VS20 TLES roof systems and the Jaguar Fixed panels are produced. These are produced on separate workstations, and only a part of the transportation is combined. Therefore, the TLES VS20 roof systems can be analysed independently of the Jaguar roof systems. This line began producing in September of 2015 and is still up and running. The data is analysed up to December 2016. In Figure 2.2 the week and month FTT's are given, as was done above for the Jaguar D7A line.




**(a)** *FTT of the Jaguar-Daimler production line with month averages.*

**(b)** *FTT of the Jaguar-Daimler production line with week averages.*

**Figure 2.2:** *FTT development of the Jaguar Daimler production line from SOP until December 2016.*

As can be seen in the figures, the fluctuation of the FTT is of similar scale as the fluctuation of the D7A line. Also, the ramp up phase is comparable. It is not yet clear what the stable FTT will be of the VS20 line as the production line is only 15 months in production. However, in the 12 weeks after the first year of production, the target of 85% is only met once with 89.2 %. The average is way below the target with only 71.7 %. These plots are made for all production lines, and the results are used to make an analysis on the global FTT development of the production lines.

### 2.2.2 Global FTT analysis

If the targets of the management and the results from the plots in figures 2.1 and 2.2 are compared, the differences are clear with the FTT in practice lower than the set targets. However, both the targets and the results show the same pattern with a fast increase of the FTT in the first months of production which diminishes at the end of the first year and the second year of production until a steady FTT is reached in the following years of production. This behaviour can be written as a exponential function with the form $FTT(t) = a - be^{-ct}$, with $a, b$ and $c$ variables defined by the increasing and diminishing growth of the FTT. Hereby the variable $a$ gives the steady FTT as $t$ becomes larger, and $b$ will be equal to $a$ if the function is fitted through the origin. Variable $c$ is determined by the grow rate of the FTT at the first months. The proposed function is fitted for all the production lines. Hereby the FTT of each roof system at all lines is averaged per month from SOP until four years into production or until December 2016(whichever is shorter). In Figure 2.3 this averaged FTT is plotted as well as the fitted exponential function.

As can be seen in the figure, the fitted exponential function is comparable with the data points from

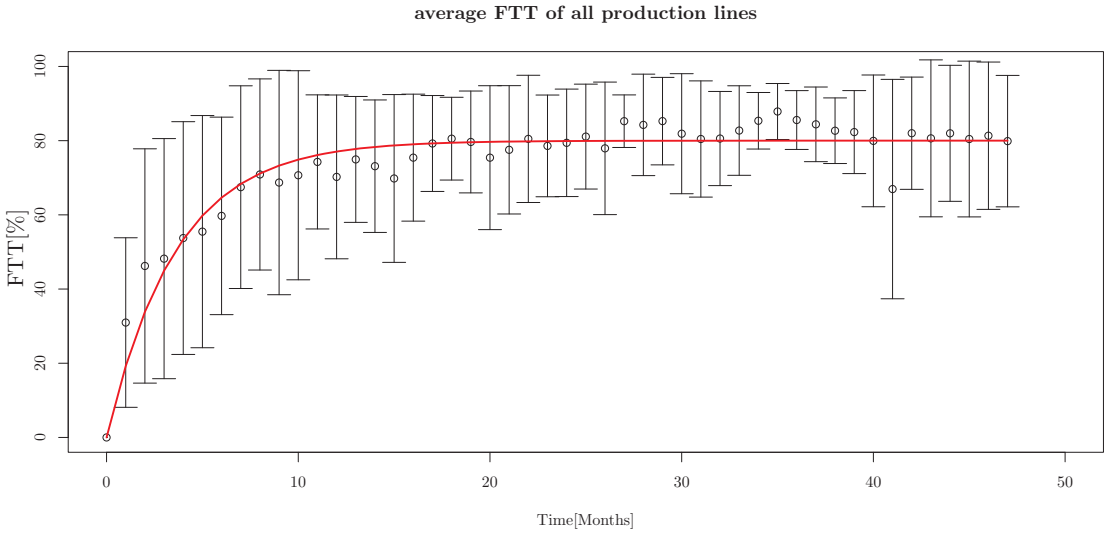
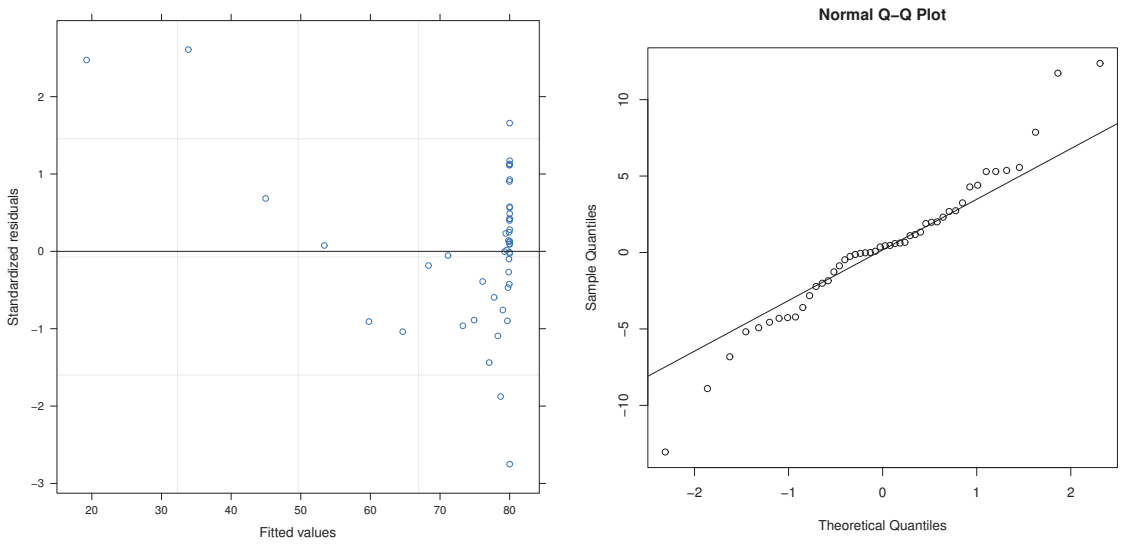



**Figure 2.3:** *FTT averages of all production lines plotted against the month since Start Of Production, fitted with exponential function.*

the averaged FTTs. The data was fitted with a non-linear least squared method using the statistics program R[7]. The residual standard error for the function was 4.692 with 46 data points. In figures 2.4a and 2.4b the fitting is checked.

As can be seen in the figures, the samples are normally distributed around the fitted function.




**(a)** *Residual plot for the fitted values of the function.*

**(b)** *QQ plot for the fitted function.*

**Figure 2.4:** *Residual plots to analyse the normality of the residuals of the fitted function*

However, for the lower values, the fitting seems to be a bit skewed. This could have multiple reasons, such as the fixed setting through the origin. Also, the later introduction of new roof systems on the line influences these lower values as most new roof systems are introduced 3 to 7 months after SOP. Although this effect is not wanted, the residues at the higher values seem to be normally distributed around the fitted distribution. Therefore, the assumption can be made that the exponential function can be used to determine the steady FTT is reached for the global production lines. From the fitting,

the value of $a$ is determined at 80.0 $\pm$ 0.8%. This is way lower that the set 95% target set by the management and still lower than the 85% target set for the first year of production.

The same analysis is done for the TLES production lines. These production lines do not have such a long production time as the other production lines, and therefore the analysis is done with week averages. The averaged FTT's are plotted in Figure 2.5.

As can be seen in the figure, the TLES averaged FTT has more variance than the global averaged

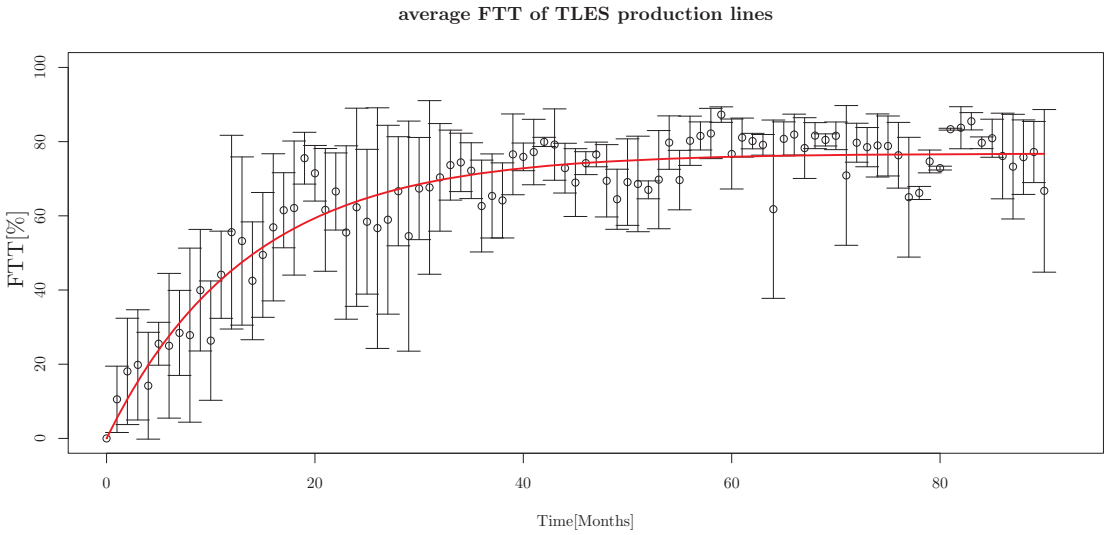


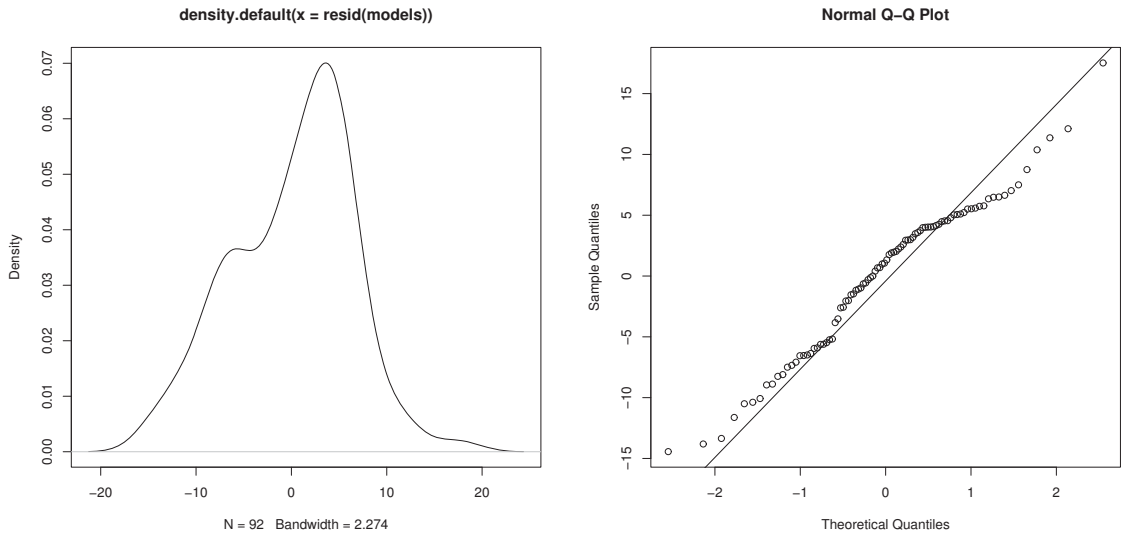

**Figure 2.5:** *FTT averages of all production lines plotted against the months since Start Of Production. Fitted with exponential function.*

FTT. This is because only four roof systems on two production lines are used compared to the 25 roof systems on 14 production lines for the global averaged FTT's. Although some data points seem very accurate due to the small error bars, this is probably caused due to coincidence that 2 or 3 FTT's are close to the same value and therefore only have a small standard deviation. This variability does also come back when the density of the residuals is plotted, and the QQ plot is made in Figure 2.6.

As can be seen in the figure, the residuals are not normally distributed but are skewed. Also, the QQ plot shows that the residuals of the fitted function are not normally distributed. So, this makes it difficult to examine the results of the fitted function. The steady FTT is set on 76.9% $\pm$ 0.97%. Although it is not proven that this value is correct or within the margin of the real value, it does show that the steady FTT is probably a bit lower than the global steady FTT.

### 2.2.3 Conclusion

The Quality of a production line is determined by the FTT of the production line. The targets set by Inalfa for the FTT are 85% after the first year and 95% in the following years of production. For TLES roof systems these targets are rarely met, with only 2 out of 50 and 1 out of 12 weeks that the average FTT is above these targets. Also, the average FTT of the global production lines does not come near the targets as the maximum of the fitted distribution is around 80%. For the FTT production lines, it was not possible to make this conclusion although it appeared to be even worse than the global FTT. The small sample size of only two production lines makes the fitting of the distribution more influenced by a bad or good week or month and therefore not possible to make a conclusion about that data.




**(a)** *Density plot for the fitted values of the function.* **(b)** *QQ plot for the fitted function.*

**Figure 2.6:** *Residual plots to analyse the normality of the residuals of the fitted function.*

### 2.2.4 Simulation implementation

To test the effect of the Quality on the line, the changes in quality is simulated on a high flexible line. The line is tested with different FTT percentages to test what the effect is on the deliverability of the production line. The experiments are done on a wide range of FTT's so that both the targets, the estimated global FTT and lower values are in the range. As no results are known for the STT, $STT = 73\%$ is used, as this is also used within the company to do the simulation of other production lines. Expected is the Deliverability to decrease with decreasing values for the FTT.

## 2.3 Availability

The Availability rate gives the percentage that the system is up and running [8]. Or as the definition is used at Inalfa, the actual production time compared to the total available times.[6]. The availability rate can be calculated using two methods. The first option is to divide the actual production time by the potential production time. The potential production time is the time that the operators should be working minus the paid breaks. The actual production time is the potential production time minus the availability losses. These losses come from the breakdown of machines, unplanned maintenance and waiting for parts or materials. This method is currently used at Inalfa to calculate the actual availability rate. The second option is to use the Mean Time To Repair(MTTR) and Mean Time Before Failure(MTBF)[8]. The MTTR gives the mean time required to repair a workstation or robot, while the MTBF gives the mean time until a workstation or robot fails and needs to be repaired. Using these two variables, the availability rate can be calculated The first and second method of calculation are shown in (2.1) and (2.2) respectively.

$$\text{Availability} = \frac{\text{Uptime}}{\text{Uptime+Downtime}} \tag{2.1}$$

$$\text{Availability} = \frac{\text{MTBF}}{\text{MTTR+MTBF}} \tag{2.2}$$

Both these equations can be used for both the production line availability as well as the workstations availability, but in practice, the first get used more for the production line and the second for the workstations.

For this project, the downtimes of the workstations were researched to determine the availability for the workstations. From this, the line availability can be determined, which later could be used as a reference for the simulation. As most of the production lines at Inalfa are one piece flow production lines, the failure of one workstation often results in a complete stop of production for the whole production line. Only in the case of parallel workstations, the production does not come to a complete standstill. If all the workstations are in series, the availability of the production line is equal to the product of the availability of each workstations, meaning that the production line is as a chain with its weakest link. The workstation with the lowest availability mostly determines the availability of the production line. If the workstations of a production line work in parallel the situation is different. In this case, two possibilities can happen. Either the failed workstations is replaceable, or the workstation is failure critical. In the first situation, the availability does not change, as the other workstation can take over the work of the failed workstation. This might change as more workstations fail, and thus resulting in the second situation. If the critical workstation is the last workstation the availability will then be 0, but if not the availability still decreases as the workstations before will start blocking, and the workstations after will start to starve. For this situation, it is very difficult to calculate the exact availability as this depends on various factors, such as the utilisation and distribution of the process times. This would mean that the availability calculation needs to be done for every situation as there is no general equation to use. The research paper of Romeu [9] gives an example for these systems with workstations with exponential failure rate, equal parallel workstations and statistically independent workstations. To make an assumption for the availability on the concept line the following example is used.
There are 2 independent parallel workstation with availability $A_i$ for each workstation with a utilisation of $\rho_i$. The Availability of the subsystem $A_ri$ is determined by (2.3).

$$A_{r2} = A_1A_2 + \left(A_1(1-A_2) + (1-A_1)A_2\right)\min\left(1, \frac{1}{\rho_1+\rho_2}\right) \tag{2.3}$$

$$\begin{aligned} A_{r3} =& A_1A_2A_3 + \left(A_1A_2(1-A_3) + A_1(1-A_2)A_3 + (1-A_1)A_2A_3\right)\min\left(1, \frac{2}{\rho_1+\rho_2+\rho_3}\right) \\ &+ \left(A_1(1-A_2)(1-A_3) + (1-A_1)A_2(1-A_3) + (1-A_1)(1-A_2)A_3\right)\min\left(1, \frac{1}{\rho_1+\rho_2+\rho_3}\right) \end{aligned}$$

The derivation of these equations for 2 and 3 parallel substations is given in Appendix C
To use the equations to calculate the production line availability the availability of the stations is needed. By analysing the current lines and their workstations the failures and repair times could be used to calculate the availability of these workstations. The availability of these workstations can then be used to make a prediction of what the availability of the workstation on the concept line is. However, the analysis of the workstations showed that the availability between similar workstations was very big. This was mainly caused by the large percentage (95+%) of one time failures.[10] After these failures were found the issue never came up again. This makes the prediction of the availability very difficult and might be impossible to prove if the prediction is reliable. Therefore a different method is used to make the predictions. This method is the result of a simulation made by a supplier of Inalfa Roof Systems and the MTBF and MTTR used by them to make the simulation.[11] The proposed numbers were analysed and approved by Inalfa Roof Systems. These values are used for general workstations based on on the complexity and automation grade of the workstations. Due to confidentiality, these values are not given.

### 2.3.1 Simulation Implementation

The availability KPI is tested with the simulation on three different cases. First, the simulation is tested with two setups for the availability of the workstations. These setups are based on the MTTR and MTBF of the supplier and show the difference between low and high automatization. The second case is testing the assumption made for the parallel workstations. These results can be used for the third case where the availability of the line is simulated by experimenting with different values of MTTR and MTTF to show the influence of these values on the Deliverability. Expected is that the Deliverability decreases with decreasing values of the availability.

## 2.4 Flexibility

The flexibility of a production line is defined by the ability to change or react with little penalty in time, effort, cost or performance [12]. Within Inalfa Roof Systems the flexibility of a production line is usually used to describe the ability to produce different roof systems and the penalty in time, effort, cost or performance, but not to react to problems in the line as there are very little buffers or flexible solutions to change the production line to absorb the problems. This research discusses the two aspects of flexibility used within Inalfa Roof Systems. The first one is the possibility to produce different roof systems and roof system types on a single production line. For the concept line, this flexibility results in the use of the roof systems build on the Audi/Porsche production line and the Honda/JLR production line. Currently, the Audi/Porsche line produces four different roof systems, while the Honda/JLR line produces five different roof systems. The five different roof systems of the Honda/JLR production line have different datum points, which are the points from which all the measurements are done, and require different fixtures. However, the assembly sequences of the roof systems are in some cases comparable. This results in 3 sets of roof systems. The X260/X760, the X761/L560 and the Honda TLGA sets. These different sets have different processing times at different stations which could lead to an other bottleneck station or robot and therefore could influence the throughput. For the Audi/Porsche line, the different set distinctions were not so clear to make as the for the Honda/JLR line. The distinction is hard to make because the Audi/Porsche line does not produce at full speed as projected in the design phase and therefore the assembly steps are still shifted between stations. Due to this reason, the choice was made to use the takt times as designed for the stations and implement those as cycle times in the concept line. The result is that the concept line should be able to produce 4.93 roof systems per time period for the roof systems of the Audi/Porsche roof systems and 5.76 roof systems per time period for the Honda/JLR roof systems. The concept line is tested in a three shift pattern to create a high volume production line. The production of the roof systems is done in batches, which are small enough that at least the demand for 400 time periods can be reached within the 400 time periods. The production of the concept line is scaled for both production lines, the exact batch sizes and production per time period are given in Chapter 3.

### 2.4.1 Changeover policy

To produce all the different roof systems, a changeover of the concept line is required. This changeover can be done on different levels and with different changeover policies. The two main parts that need to be changed during changeover are the fixtures of the carriers and the exchange of toolings in the workstations. To examine the different changeover policies, the changeovers of 3 production lines are compared. The first two production lines are the lines that are used to design the concept line. The third line is the MFA2/MRA production line. This production line is chosen as it is a production line with TLES roof systems and is different from the first two lines as it requires very little changeover. The Honda/JLR line produces five different roof systems divided into three sets as described before. However, even within a set, changeover can be required for certain stations. Mainly the big stations like the glass panel bonding and the glass setting stations determine the required time for the changeover. The changeover of fixtures of the carriers is not required as all current roof systems produced on the line have the required fixtures on the carriers. This might be necessary if new roof systems are introduced on the production line. The changeover is initiated with an empty carrier, that is sent after the last roof system of a batch, starting at the beginning of the production line.

The Audi/Porsche line is not fully operational for changeover yet. Therefore, it is required to empty the complete line first, then the grippers of the robot, the carriers and the roof holders of the two lines can be changed over. The changeover has to be done manually and takes a lot of time. Also, there is no clear estimation of what time it takes to change over. The new Audi/Porsche line that is currently being built is designed with an automated changeover. Hereby there is a fixed time or delay for changeover at the robots and stations with fixtures. Also on the carriers, the top plates need to be changed, These top plates contain the fixtures for the roof systems. The goal is to have this done within takt time. However, the possibility of a certain delay per takt is taken into account.

The MFA2/MRA line has three different roof systems that are quite similar. The datum points of the

three roof systems are the same, and therefore the fixturing can be used for the carriers and on the workstations. At some workstations, the change of tooling is required, but this can be done within takt time. Therefore, the changeover can be done within one takt time, and the only necessity is to send an empty carrier between the batches.
If the three systems are compared three different strategies come up.

1. Changeover with one takt time.
2. Changeover with a fixed time.
3. Changeover with a fixed time and a delay in takt time.

These strategies are used for different cases and affect the performance of the production line and the throughput of the line. To produce the time period demand of roof systems, the number of changeovers have to be changed depending on the strategy and required time for the changeovers. This results in a minimum batch size of production of the roof systems. The minimization of batch sizes is currently used within Inalfa to minimise the finished roof systems in stock. The costs that are required for this are not taken into account in this research, and therefore there is no research done into determining the optimal batch size to stock ratio.

### 2.4.2 Simulation Implementation

To test the Flexibility KPI, the simulation is based on the changeover strategies as mentioned above. The three strategies are tested on the concept line for the time period demand. For the second and third strategy a range of fixed times are used up to 30 minutes. This 30 minutes is the maximum fixed changeover time in the current TLES production lines and is considered as a long time of a changeover within Inalfa. For the delay in takt time, a time of 40 seconds is taken for each takt time, until the fixtures of all carriers have been changed. This happens on the first used workstation of each independent conveyor system. For each experiment, the minimum batch size is determined to keep up with the 400 time periods demand. Expected is that the batch sizes decreases as the changeover gets faster and as more changeovers can happen.

With the researches done for the different KPI's and an insight of the performances of current production lines, the experiments are made to test the effects of the KPI's on the performance of the production line. To do these experiments a simulation model has to be build. This model is based on a concept production line of which the design is explained in the next chapter.

# Chapter 3

# Layout Design

The investigated results of the KPI research from the last chapter are used to set up experiments which are also discussed in chapter 2. These experiments require a simulation model. The model should be based on a real life process or a designed process. In this chapter, the process of choosing and designing the concept production line is discussed. First, the choice for which production lines are used is explained section 3.1. In section 3.2 the general production flow is explained of some of the current production lines, and the designing of the production flow of the concept production line is discussed. With this general flow, the production per section can be decided, that is discussed in section 3.3.

## 3.1 Concept production line

To simulate the effects of the KPIs on a high flexible TLES production line, such a production line must first be chosen or designed. Currently, there is no high flexible TLES production line at Inalfa Roof Systems yet, but a few years ago an attempt was made to design such a production line. [13]. This project was not finished, and the set of roof systems that were supposed to be produced on that production line is currently produced in the US on different production lines, which made the data gathering very hard. In addition, the interest from Inalfa Roof Systems was more into the TLES roof systems produced in Europe. Therefore, the decision was made to design a new concept production line. This concept line would be able to produce 80% of current and future TLES roof systems, so only exclude the exotic TLES roof systems. The concept production line should be able to produce most roof systems, but for this research, it is not required to design the workstations and the technical details of the workstations. Therefore, the assumption is made that a basic workstation can be designed in such a manner that it can perform all basic work that is produced at any basic workstations.

## 3.2 Production Flow

The concept production line is based on the production lines that are currently used for the TLES roof systems. On this concept line, the two main assembly methods for TLES roof systems should be able to be built. The two main assembly methods are the bathtub method, where the assembly is done on both the top and bottom side of the of the roof systems, and the driveline method, where a stacking principle is used and the assembly is only done from the top side. These different methods also result in a different flow for the assembly, as the bathtub method is assembled in series, while parts of the driveline method require being built in parallel, and after that be married, and finished building in series. To create the concept production line, some of the production lines are compared. First, the comparison is made with the general assembly steps of some of the production lines. With this, the general flow layout of the concept line is made. The general assembly steps of the Honda/JLR line, which follows the bathtub assembly method, and the Audi/Porsche (original and new) lines and Jetta line, which use the driveline assembly method, are shown in Figure 3.1. Herein the difference and similarities of the assembly steps in the production lines are visible.

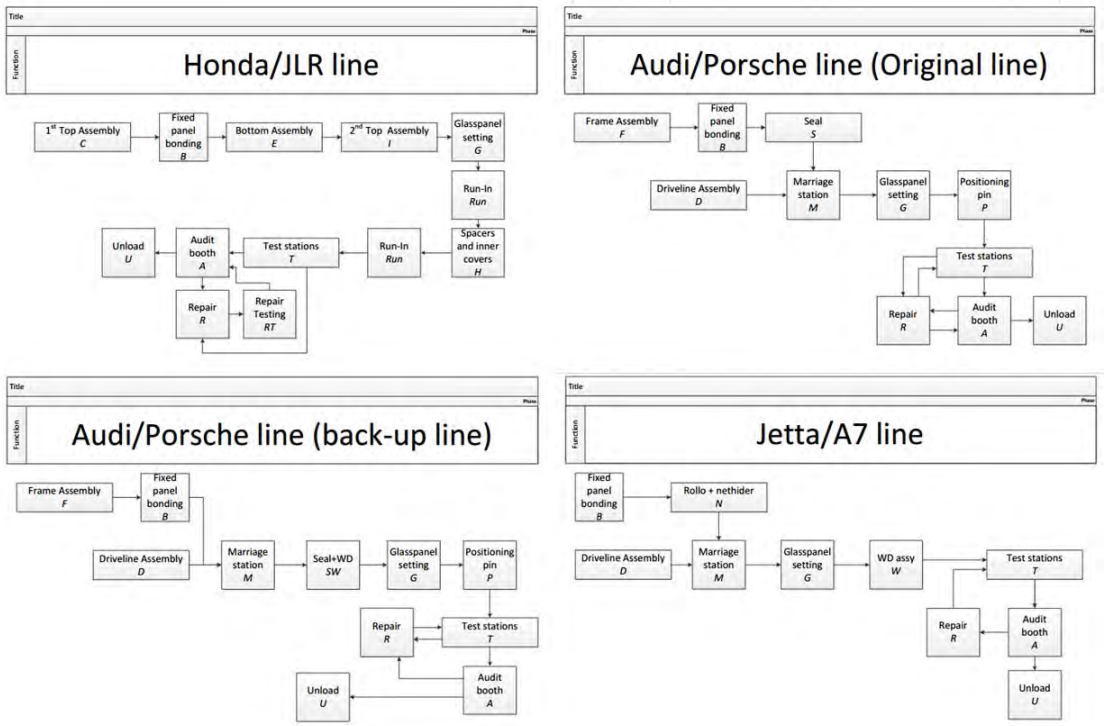



**Figure 3.1:** *General production flow of current production lines.*

As can be seen in the figure above, the production lines have some workstations in common, and all have the test stations, audit boots and repairs, were the roof systems are tested and possibly repaired. The two workstations that are in all production lines are the fixed panel bonding workstation and the glass panel setting. The End of Line(EOL) is different for each line, but they consist out of the same four elements for each line, the test stations, audit booth, repair stations and unload station. For the concept production line, only one EOL is used to test and repair all different roof systems. The other two shared workstations, fixed panel bonding and glass panel setting, are expensive workstations with expensive tooling. Therefore, these workstations are only built once in the concept production line. The other workstations and robots have to be designed around these parts. With this in notice, two concept production lines are designed, that are shown in Figure 3.2.

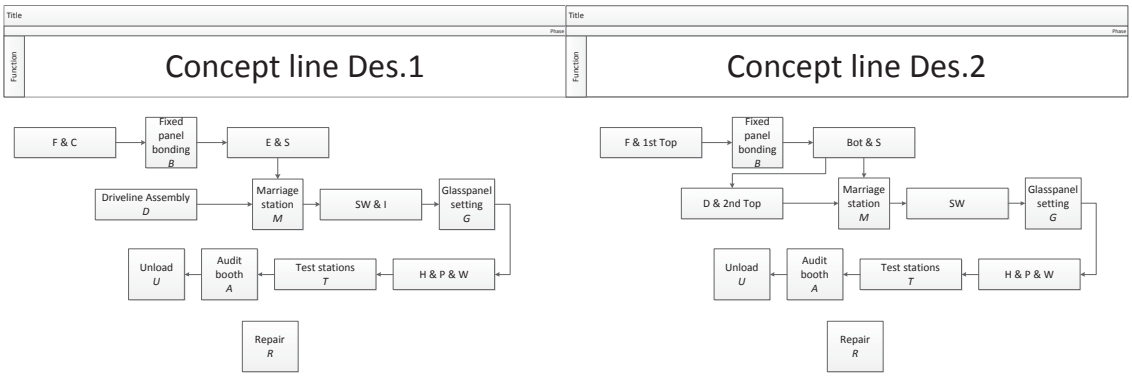



**Figure 3.2:** *General production flow of 2 proposed concept production lines. Letters represent sections of production lines of which the meaning can be seen in Figure 3.1*

As can be seen in the figure above, the two concept lines have a similar general assembly flow. The main difference is where the bottom assembly of the Honda/JLR roof systems is done. In the first design, this is placed between the marriage station, which is not used for these bathtub roof systems, and the glass panel setting. In the second design, this is done in the same section where the driveline assembly is done of the driveline roof systems. Both concept lines are able to produce the roof systems from the Audi/Porsche and the Honda/JLR line. To determine which line has a preference the lines are compared by the number of workstations needed to produce the roof system within designed cycle time. For this, the process times and designed cycle are compared for the driveline assembly section and after marriage section, and then the required number of workstations is calculated. The current process times and designed cycle times for the Audi/Porsche production line and Honda/JLR production line as shown in Table 3.1. For the first concept, the required number of workstations are 5 and 4, while for the second concept only 5 and 1 are needed. This concept was even improved by shifting the wind deflector assembly after the glass panel setting, as there was already a workstation needed for the spacers for the Honda/JLR roofs. With the general concept set, the section production can be discussed.

## 3.3 Section Production

With the general flow, the production of the sections is set. The workstation's cycle time should be designed on the actual process times of the production lines as given in Table 3.1. However, as can be seen in that table, the process times of the workstations can be much longer than the designed cycle times. For instance, the Honda/JLR line has workstation M17, which has a process time 32% higher than the designed cycle time. For the Audi/Porsche production line, this percentage is even up to 148% at the first test station. Expected is that if the simulations were done with these values for the workstations, the results would not be interesting and would mostly just suggest that the system is running with those stations as a bottleneck. Therefore, in consultation with Hans Bastiaanse [13] and later Michel Coenen [**Coenen**], the decision is made not to work with the given process times and cycle times, but choose interesting options with respect to bottleneck placement in the line and balancing of the work. As described in Section 2.4 both the production lines have four different roof systems that are produced. For both sets, the roof systems are split into four groups with different designs for the process times of the main production line. These groups are defined as follows.

1. Balanced production line with equal process times.
2. Production line with early bottleneck workstation.
3. Production line with late bottleneck workstation.
4. Production line with decreasing process times for the workstations.

Hereby the sum of the process times of all workstations on the main line is equal for all four groups, from now named Mainline BOL. The mainline BOL is equal for all roof systems with the same assembly method, meaning that the designed cycle times are different for the two assembly methods and corresponding roof systems. The designed cycle times of the EOL are set equal to 85% of the designed cycle time of the bottleneck station of the main production lines, as this is done by Inalfa for all production lines. As the concept production lines is a carrier system production line the transport of the roof systems from and to the workstation is also determining the cycle time of the workstation. Therefore, the cycle time of a basic workstation is tested.

With the design of the concept production line finished, a simulation model can be made to use for the designed experiments. How this simulation model was build and what decisions were made during the design is explained in Chapter 4. Also in this chapter, the working function of the simulated basic workstation is explained and the cycle times determined by a range of process times.

**Table 3.1:** *Table contains the process times per section and stations for the Honda/JLR line and the Audi/-Porsche line. The values are anonymized to timeunits for company confidentiality. Values are collected on 02-05-2017*

| | Honda/JLR | | | | Audi/Porsche | | | |
|---|---|---|---|---|---|---|---|---|
| Sections | Total time [time-unit] | Station | Process time [timeunit] | Designed Takt Time [timeunit] | Total time [time-unit] | Station | Process time [timeunit] | Designed Takt Time [timeunit] |
| Pre Bonding assembly | **0,303** | Load | 0,080 | 0,094 | **0,418** | 160,1 | 0,091 | 0,087 |
| | | M01 | 0,068 | 0,094 | | 160,2 | 0,090 | 0,087 |
| | | M02 | 0,077 | 0,094 | | 160,3 | 0,143 | 0,087 |
| | | M03 | 0,079 | 0,094 | | 140 | 0,093 | 0,087 |
| Fixed panel bonding | **0,069** | M04 | 0,069 | 0,094 | **0,167** | 150 | 0,167 | 0,087 |
| bottom frame assembly | **0,319** | M05 | 0,077 | 0,094 | **0,153** | 170,1 | 0,077 | 0,087 |
| | | M06 | 0,081 | 0,094 | | 170,2 | 0,077 | 0,087 |
| | | M07 | 0,077 | 0,094 | | | | |
| | | M08 | 0,084 | 0,094 | | | | |
| Driveline and top assembly | **0,364** | M10 | 0,110 | 0,094 | **0,776** | 30,1 | 0,144 | 0,087 |
| | | M11 | 0,100 | 0,094 | | 30,2 | 0,156 | 0,087 |
| | | M12 | 0,088 | 0,094 | | 30,3 | 0,178 | 0,087 |
| | | M13 | 0,066 | 0,094 | | 30,4 | 0,136 | 0,087 |
| | | | | | | 30,5 | 0,162 | 0,087 |
| Marriage station | | | | | **0,170** | 40 | 0,170 | 0,087 |
| glass panel setting | **0,277** | M14 | 0,116 | 0,094 | **0,180** | 50 | 0,000 | 0,000 |
| | | M15 | 0,096 | 0,094 | | 80 | 0,180 | 0,087 |
| | | M16 | 0,066 | 0,094 | | | | |
| Spacers, pins and WD assy | **0,097** | M18 | 0,097 | 0,094 | **0,089** | 90 | 0,089 | 0,087 |
| Run-in | **0,124** | M17 | 0,124 | 0,094 | **0,000** | 110 | Unk. | 0,087 |
| Test stations | **0,310** | M19 | 0,081 | 0,076 | **0,173** | 110A | 0,173 | 0,070 |
| | | ST20 | 0,090 | 0,076 | | 110B | Unk. | 0,070 |
| | | ST21 | 0,092 | 0,076 | | 110C | Unk. | 0,070 |
| | | ST22 | 0,048 | 0,076 | | | | |
| Audits | **0,167** | ST30/31 | 0,167 | 0,151 | | 110D | Unk. | 0,070 |
| Unload | **0,010** | ST40 | 0,010 | 0.094 | | 130 | Unk. | 0,087 |
| Total | **2,039** | | 2,039 | 2.153 | 2,126 | | 2,126 | 1,823 |

# Chapter 4

# Simulation Model

To test the experiments that were set up in Chapter 2, a simulation model needs to be built. This simulation model is based on the layout design of the concept line as proposed in Chapter 3. The simulation model is made in Siemens Tecnomatix Plant Simulation, created by Siemens PLM Software for the modelling, simulating, analysing and visualising of production processes and systems[14]. The software is currently used at Inlaid Roof Systems for simulations as it has a basic production system can easily be built with the drag-and-drop features of the software, while with the underlying programming language Simtalk complex process can be simulated and systems can be created that are more manageable and closer to real life production processes. The visual aspect of the simulation software helps the programmer to explain the experiments and results to colleagues or customers who are not familiar with the simulation software. In this chapter, the transition from the layout of the concept line to the simulation model is explained. In Section 4.1 the basic structure of the simulation model is explained and different structures are discussed. Then in Section 4.2 the basic unit is explained and the behaviour to other units.

## 4.1 Structure

The model for the concept process line consists of 3 conveyor systems which are linked through robots to transfer the roof systems. The Honda/JLR roof systems require 2 different fixtures for the carriers, one for top side assembly and one for bottom side assembly. The Audi/Porsche roof systems also require 2 different fixtures for the carriers, one for the frame assembly and one for the driveline assembly. If these are combined with the required fixtures at the different sections of the concept line, they result in 3 different fixture carrier systems. First with the top-frame fixtures, then with the bottom-frame fixtures and last the one with the top-driveline fixtures. The robots between the three carrier systems are used to transfer the roof systems and in case of the Honda/JLR roof systems turning. The robots can handle all roof systems and do not require any changeover of the grippers. The general layout of the simulation model is shown in Figure 4.1.

As can be seen in the figure above, the simulation model has three carrier systems to move the roof systems. The system on the top-right holds the frame and first top assembly section and the fixed panel bonding section. The second carrier system is on the top-left, that holds the bottom assembly and seal assembly section. Finally, the other sections are on the last carrier system at the bottom, which also holds the EOL and repair sections. Further in this chapter the different workstations and robots that are used in the simulation model are discussed, starting with the basic unit or basic workstation.

## 4.2 Basic unit

The basic unit of the simulation model consists of either one or two objects. These are the conveyor on which the carriers run, noted as either *LB...*(Line Belt) or *LR...*(Line Return) and possibly the workstation noted as *ST.....* The conveyor is noted *LB...* if the conveyor runs in the primary direction,

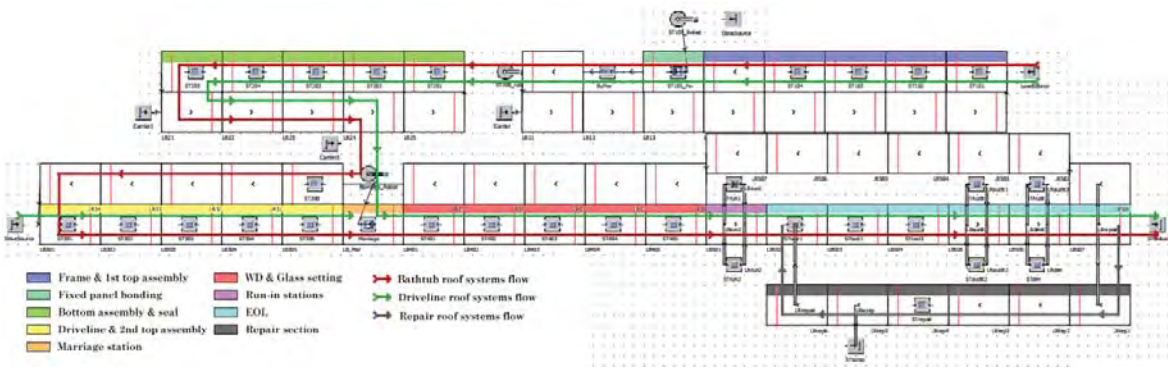



**Figure 4.1:** *Layout of the simulation model of the concept production line. Shows the sections of assembly and the flow of the roof systems through the model.*

which is the side with the most workstations, and *LR...* is used for the returning side of the conveyer belt. An example of the basic unit is shown in Figure 4.2.

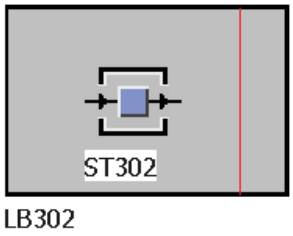



**Figure 4.2:** *Example of a basic unit with workstation ST302 and conveyor LB302.*

As can be seen in the figure above, there is a red line on the conveyor. This line represents the sensor attached to the conveyor. The sensor is required for the functioning of the basic unit. When the carrier hits the sensor, the conveyor is stopped and then a decision is made. This decision is based on the content of the carrier and whether the basic unit has a workstation or not. First, it is checked if the basic unit has a workstation. If not, the conveyor waits for a signal to start again. If it has a workstation the carrier is checked, if empty the conveyor waits for the signal to start moving again, if not empty the roof system is released for work on the workstation. This working is programmed of which the code is given in Listing 4.1.

**Listing 4.1:** *SIMTALK-Code for conveyor sensor of basic unit. Code is a part of a script for all sensor.*

```
1  @.Stopped := true;
2  ?.Stopped := true;
3
4  if ? = LB101 then
5      ...
6  ...
7  elseif not (?.ston = void) then
8      if ?.mu.empty;
9          waituntil ?.nextline.ReadyToMove prio 1;
10             ?.ReadyToMove := true;
11     else
12         @.mu.move(?.StOn);
13     end;
14 elseif  (?.ston = void) then
15     waituntil ?.nextline.readyToMove = true;
16     ?.ReadyToMove := true;
17 end;
18
```

```
19 waituntil ?.readyToMove prio 1;
20 if ? = LBscrap;
21     ?.stopped := false;
22 else
23     if not @.changeover_dummy or not (? = LB106 or ? = LB_mar);
24         ?.stopped := false;
25         if ?.empty = false;
26             @.stopped := false;
27         end;
28     end;
29 end;
```

As can be seen in the listing of code above, the code starts at a *elseif* statement, as the code is part of a bigger script. The script is used for all sensors. SIMTALK is comparable to many Java-based programming languages, but two symbols are not common for these languages. The ? and @ are variables in every script, with ? the object, such as a workstation, robot or conveyor, that calls the script, or in SIMTALK named the *Method*. The @ variable is the moving unit(MU) that triggers the *Method*. If any sensor is activated by a carrier, first the specific conveyor and carrier are stopped in line 1 and 2. Then the conveyor is checked if there are special requirements, such as conveyor *LB101*. These special requirements are discussed later in the chapter. If no special requirements exist it is checked if the unit holds a workstation, which is noted as that the conveyor has a workstation on it. This check is made in line 7 by checking the variable *STon* of object *?*. Then the carrier on the conveyor(?.mu) is checked if this is empty. If empty the conveyor waits until the next conveyor on the line is ready to receive a carrier, noted as the variable *?.nextline.ReadyToMove*. If not empty the roof system on the carrier is sent to the workstation in line 12. When a roof system enters a workstation the process time is determined depending on the station and the roof system type. When the process is finished the method *Station Exit* is started, which is partly given in Listing 4.2

**Listing 4.2:** *SIMTALK-Code for workstation of basic unit. Code is a part of a script for the station exit.*

```
1  ..
2  if ? = STtest1 or ? = STtest2 then
3      ...
4  ...
5  else
6      waituntil ?.lineon.nextline.ReadyToMove prio 1;
7      @.move(?.lineon.mu);
8      ?.lineon.readytomove := true;
9  end
10 return;
```

As can be seen in the code of the listing, the station is first checked for special requirements. If not there, the workstation waits until the next conveyor is ready to receive the carrier. Then, the roof system moves back to the carrier on the conveyor at line 7 and the conveyor receives a signal that is can start moving again. If we check Listing 4.1 again we can see that this signal is required to make the conveyor and the carrier move again. Conveyor *LBscrap* is an exception as this conveyor only hold a roof system and no carrier. The variable *changeoverdummy* is explained later in the next section.

## 4.3 Units of interest

With these methods, the basic units can be coupled to a section or simple production line. However, there are more complicated units needed to make a simulation model of the line. For a start, the carrier systems need loading stations to start the assembly. Furthermore, parallel stations, marriage stations, unloading stations and robots are required.

### 4.3.1 Loading and unloading stations

There are two types of loading stations required to model the concept production line. First, a loading station were the roof system parts are loaded from a source onto the carriers. This happens at stations

*ST101* and *ST301*. Second, a loading station were the roof systems are loaded to the carriers from a robot. This happens at stations *ST201* and *ST300*. For the first type of loading stations, the assumption is made that the source of the parts has no process time so parts are instant. The loading cycle starts when a carrier activates the sensor of a conveyor *LB101*. The method *LBsensor* is called, were the code in Listing 4.3 is programmed for the specific conveyor.

**Listing 4.3:** *SIMTALK-Code for workstation of conveyor LB101. Code is a part of a script LBsensor*

```
1  ...
2  if ? = LB101 then
3  if Ending and (WIP_bath+WIP_Drive)>0 then
4      waituntil ?.nextline.readytomove = true;
5      Empty_carrier;
6  elseif first_changeover then
7      waituntil ?.nextline.readytomove = true;
8      @.changeover_dummy := true;
9      Empty_carrier;
10     first_changeover := false;
11 else
12     ?.ReadyToLoad :=true;
13 end;
14 ...
```

In this part of the code, it shows how the loading is handled. First, the check is made if the production is coming to a stop and the production line is emptied, which would be given by the variable *Ending*. If so, the conveyor would keep sending the carriers through without loading. Lines 6 till 10 describe the process if the production line is having a changeover. This is discussed later in the next section. If none of these are true, then the conveyor gives the signal that it is ready to load. This signal is used at the exit of the *LoadSource* to send a roof system part to the station on the conveyor. The loadsource calls the method *Source Exit* every time a roof system part leaves the source. The code of this method is shown in Listing 4.4.

**Listing 4.4:** *SIMTALK-Code for source exits*

```
1  if ? = LoadSource then
2      waituntil not ?.stop_prod prio 1;
3      if first_changeover and LB101.ReadyToLoad;
4          waituntil LB101.nextline.readytomove = true prio 1;
5          Empty_carrier;
6          LB101.mu.changeover_dummy := true;
7          first_changeover  := false;
8          LB101.readyToload := false;
9      end;
10     waituntil LB101.ReadyToLoad prio 1;
11     if @.name = "A5_coupe" or  @.name = "A5_SB" or  @.name = "pana_limo" or  @.name ...
           = "pana_exe" then
12         @.Drive_concept := true;
13         WIP_Drive:=WIP_Drive+1;
14         ?.current_mu := @.name;
15     else
16         @.Drive_concept := false;
17          WIP_bath := WIP_bath+1;
18          ?.current_mu := @.name;
19     end;
20     if second_changeover = true;
21         next_batch += 1;
22     else
23         current_batch += 1;
24     end;
25     if full_changeover = true;
26         next_batch_end += 1;
27     else
28         current_batch_end +=1;
29     end;
30     LB101.ReadyToLoad := false;
```

```
31      @.Move(LB101.StOn);
32      ?.current_mu := @.name;
33
34  elseif ? = DriveSource then
35      waituntil LB301.ReadyToLoad prio 1;
36      @.Drive_concept := true;
37      LB301.ReadyToLoad := false;
38      @.Move(LB301.StOn);
39  end;
40
41  if ? = Carrier3 then
42      waituntil LR3.empty and LR45.empty prio 1;
43      @.Move(LR3)
44  end;
```

As can be seen in de listing's code, first the right source is checked. For the Loadsource the variable *stop_prod* is checked if something changed for the changeover. Lines 3 till 8 are used for the changeover and are discussed later in the next section. Then the Loadsource waits for the signal of conveyor *LB101* that it is ready to load. Then the roof system part is checked to see the type and adjust the settings of the part and upgrade the WIP level of the roof system assembly type. This WIP is not for the complete line, but the WIP between the loadsource and conveyor *LR3*. Line 20 to 29 is used for the changeover again. Then the variable to load is reset and the roof system part is moved to the loading station on the conveyor. The Drivesource waits for the signal from the *LB301* conveyor but does not adjust the WIP level, as this is done at the loading conveyor *LR3*.

As there are two different roof system assembly methods, there are 2 different flows which the roof systems can follow, as can be seen in Figure 4.1. Therefore, the decision needs to be made where the carrier is required, either at station *ST300* to receive a bathtub roof system or send the carrier to station *ST301* where the assembly of the driveline for the driveline roof systems can begin. For this decision, the WIP levels are used to check how many carriers still have to wait or have to move through *LR3*. In Figures 4.3 and 4.4 the decision diagram for conveyor *LR24* and *LR3* are shown. These diagrams show what decisions are made in the code, as the code for these conveyors is too long to put in the main text. The codes comes from the method *LBsensor*.

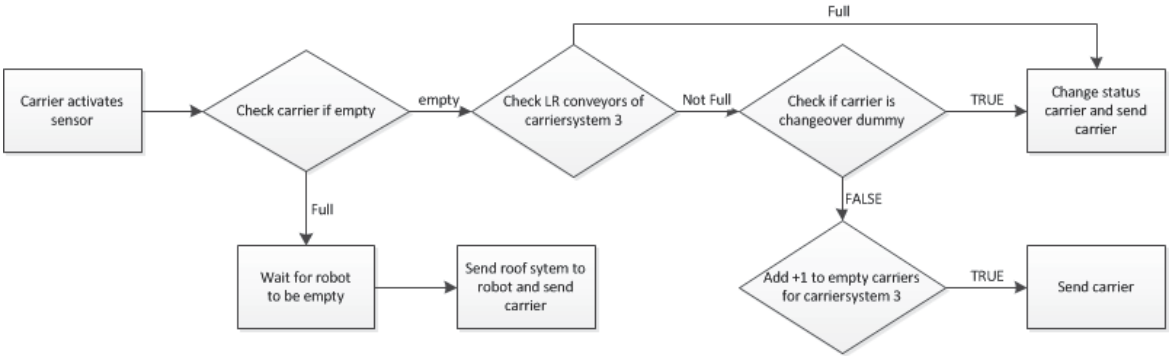



**Figure 4.3:** *Decision diagram for conveyor LR24*

As can be seen in Figure 4.3 the method is again started at the activation of the sensor. The the carrier is checked if it contains a roof system. If it does, the robot is checked to see if it is available and if the robot is ready the roof is picked up by the robot and the carrier can move to the next conveyor when possible. If the carrier is empty, carrier system 3 is checked to see if there is place to add an extra carrier to the system. If not possible the empty carrier in *LR24* is send through when possible. If it is possible, the carrier is checked whether it is a changeover dummy. For changeover dummies the system does not have to add an empty carrier to carriersystem 3, as for the driveline batches the carrierdummy is started earlier at station *LR3*, than *LR24* receives the changeover dummy. So if the changeover dummy reaches conveyor *LR24* the status of the carrier is updated and the carrier is send through when possible.

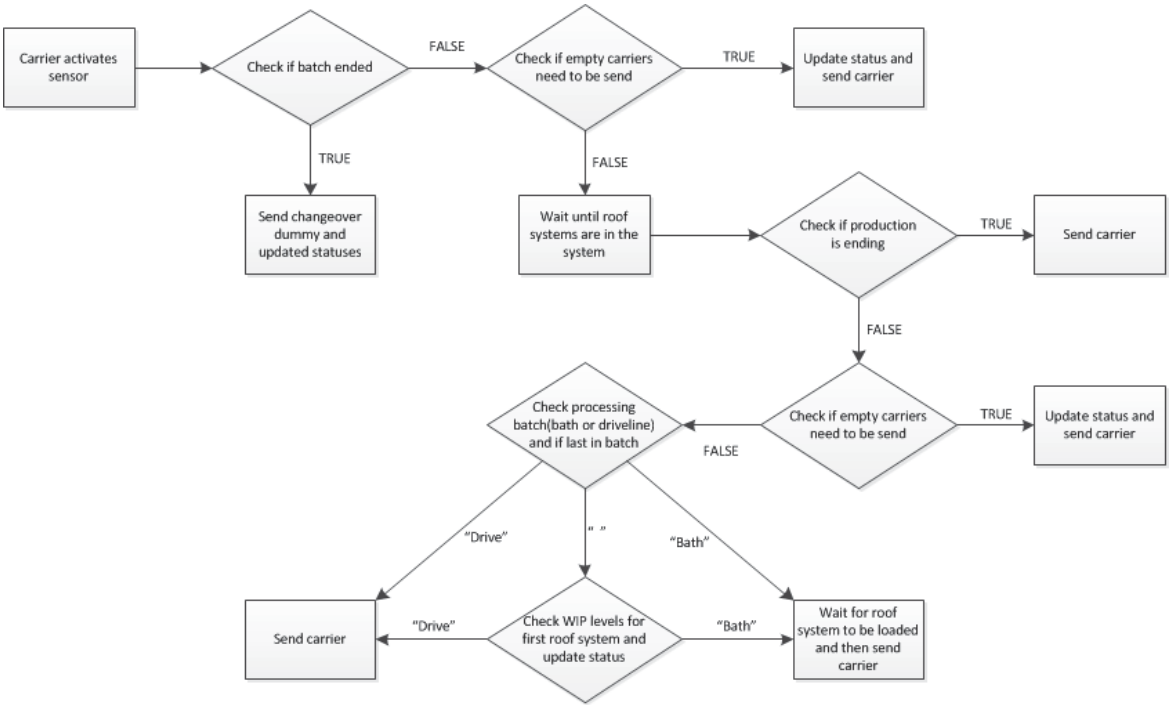



**Figure 4.4:** *Decision diagram for conveyor LR3*

The decision diagram for conveyor *LR3* is a bit bigger than for *LR24*, as can be seen in Figure 4.4. The conveyor is starting the method at activation of the sensor. First the system is checked if, either all roof systems of a bathtub roof system type have passed or if enough carriers are sent through to build the assembly of the drivelines for the driveline roof systems. If so, a carrier dummy is send and the variables are updated so a new batch can be received. If the batch is not ended the system is checked whether an empty carrier needs to be sent. Then the systems wait until there are roof systems in the system to make sure the right decisions are made. Then the system is checked whether the production is ending and the system should run empty. If so the system keeps sending carriers to make sure the EOL in not blocked by empty carriers. Due to the wait the system can have made new requests for empty carriers so this has to be checked again. If no empty carriers are required the decision can be made based on the next roof system to come into carrier system 3. First the system is checked to see which roof systems type is currently produced in the current batch. If the current batch is driveline type roof systems the carriers are sent through to make sure exactly enough driveline assemblies are made to marriage with the frame assembly. If the bathtub type roof systems are produced, the conveyor waits until a roof system is loaded onto the conveyor and then the carrier is sent through. If the production has just finished a batch and a new carrier arrives, the WIP levels of carrier systems 1 and 2 are checked to see which type of roof system is the first to be produced. The maximum WIP levels of these two carrier systems is less than 50 roof systems, therefore there can never be two different type on these systems when a new batch starts at the third carrier system.

### 4.3.2 Batchsizes and changeover

In Section 2.4 the changeover strategies are discussed. In the simulation model, a dummy carrier is used to give a signal to the workstation to change tooling and fixtures. This changeover dummy is sent after each batch that is produced. In the simulation model, there are four objects that require or update the batch sizes and statuses of the changeover. These are the loadsource and *LB101*, *LR3*, *LB504* and *LB507*. Therefore, there are 4 parameters for the status of the changeover required. These statuses are first assigned at the load station, each time a new batch starts. The method that is used, is called every time when a roof system is created in the loadsource and is shown in Listing 4.5

**Listing 4.5:** *SIMTALK-Code for source creation*

```
if ? = LoadSource then
    ?.Stop_prod := true;
    if @.name /= ?.current_mu and ?.current_mu /= "";
        waituntil next_batch_end = 0 prio 1;
        first_changeover := true;
        second_changeover := true;
        Third_changeover := true;
        Full_changeover := true;
        ?.previous_mu := ?.current_mu;
    end;
?.stop_prod:=false;
end;
```

In the listing of the method for the loadsource, the incoming MU's are checked if they are of the same type as the previous ones. If so, nothing happens and the method *Source_Exit* is called. If they are not the same, the MU is the first of a new batch and the changeover policy is started. First, the system must be cleared of the old batch, meaning there can only be two batches in the system. This is a requirement made due to the changeover policy at *LR3*, and explained later. The four statuses for changeover are updated and the name of the old batch is transferred so the information is kept until the changeover is passed completely. This is done when the variable *Full_changeover* is false again. The first changeover is done at conveyor *LB101* as can be seen in Listing 4.3. The variable *changeover_dummy* of the carrier is changed to true and the method *EmptyCarrier* is called, that sends the changeover carrier to the next station whenever possible. Then the changeover dummy moves through the system. And at every conveyor that needs changeover, the changeover is started when the dummy carrier activates the signal. The conveyor blocks new carriers entering until the changeover at that conveyor's workstation is finished. Then when conveyor *LR24* is reached with the dummy the variable *changeover_dummy* is reset to false. The dummy carrier for the third carrier system is generated independently of the dummy carrier in the first and second system. However, in the simulation, the carrier appears to continue. This is because the batches are updated and when the current batch is empty the conveyor *LR3* creates a new carrier dummy. With this in mind, the process at conveyor *LR3* can be explained, as stated in the design diagram shown in Figure 4.4 or with the code listed in Appendix A.
When a carrier activates the sensor at *LR3*, the first check is whether the previous MU was the last of the batch. If so, a dummy carrier is created to continue the changeover and the changeover variable is updated. Then the system is checked if the process should wait for an update from conveyor *LR24*. If not, the *empty_wait* variable is checked whether a new empty carrier needs to be created. Then the WIP levels are checked if any roof systems are in the production line that still needs a new carrier at the third carrier system. If this is so, at least one roof system which needs to have a driveline assembled the system can continue, noted the variable *Nexxt*. If not, it has to wait until *LR24* has a roof system that needs to be loaded at *ST300*. Then the system is checked if the production is stopped, but the carriers need to keep running to prevent blocking at the EOL. The conveyor holds the current MU assembly method in memory with the variable *Drive_Type*. This is done because both WIP levels are greater than 0 so both could be next. When the production starts the *Drive_Type* is empty and is assigned *"Drive"* or *"Bath"* , depending on which roof system comes first. If the *Drive_Type* is empty, it should only be possible to have one roof system type in the system. Therefore the requirement is made that the batches should be at least 50 roof systems. So, the WIP level is checked and *Drive_Type* is updated to the current roof system assembly method. The next roof systems that are coming in are either loaded at station *ST300* or carriers are sent through to assemble the driveline. This continues until the last roof system comes in and the *Drive_Type* is updated to empty again, the variable *Current_batch* is zero and the cycle is started again.
The transfer of the roof systems parts from one carrier system to another is done with robots. These are explained in the next section as well as the marriage station and glass bonding station that use the robot for assembly.

### 4.3.3 Robots and marriage stations

The robot that needs to do the transfer is given a sign from the conveyor that holds the roof system. This conveyor starts the corresponding method *..._Robot_Exit*. This method gives a target to the robot to where the robot needs to go and pick up the roof system part. The method *Mar_Robot_Exit* that is used by the robot between *LR24*, *LR3* and *Marriage* is shown in Listing 4.6.

**Listing 4.6:** *SIMTALK-Code used to trigger the Mar_Robot*

```
1  if LR24.mu.mu.Drive_Concept then
2      waituntil LB_mar.ReadyToLoad prio 1;
3      LB_Mar.ReadyToLoad := false;
4  elseif not LR24.mu.mu.Drive_Concept then
5      waituntil LR3.ReadyToLoad prio 1;
6      LR3.ReadyToLoad := false;
7  end;
8
9  if not Marriage_robot.resworkingand Marriage_robot.empty then
10     if LR24.Mu.Mu.Drive_concept then
11         Marriage_Robot.target := Marriage;
12         LR24.mu.mu.move(Marriage_robot);
13     else
14         Marriage_Robot.target := ST300;
15         LR24.mu.mu.move(Marriage_robot);
16     end;
17 end;
18
19 return;
```

The method is triggered when a carrier with roof system is activating the sensor on conveyor *LR24*. The method checks if the receiving station is ready and then moves the roof system part to the robot. This sends the robot to the sending conveyor to pick up the roof system part. When the part is picked up, the robot starts method *Mar_Robot_Dest*, shown in Listing 4.7, to set the destination for the robot to deliver the roof system part.

**Listing 4.7:** *SIMTALK-Code used to set the destination for the Mar_Robot*

```
1  if Marriage_robot.cont = void then
2      return;
3  end
4  if @.Drive_Concept then
5      ?.SetDestination(Marriage);
6  else
7      ?.SetDestination(ST300);
8  end;
```

The *Marriage* station sends the signal *ReadyToLoad* when the driveline assembly is at the station and the frame assembly can be married on top. The fixed panel bonding station uses the same robot principle and marriage principle. The stations after the marriage station are basic units until the two parallel run-in stations. These require a new way of control as blocking can become a problem.

### 4.3.4 Parallel stations

The simulation model has three conveyors with two parallel workstations attached. These three conveyors have the same principle for movement but have different ways of control. When a carrier enters the conveyor and activates the sensor, there are three options that can happen. The carrier can go on to the workstation on the left or on the right, or the carrier can go to the next conveyor, depending on the content of the carrier and the contents of the workstations. But this can cause blocking as a new carrier can activate the sensor and needs to go to one of the workstations on the side, but these are full and can not be emptied as the new carrier is blocking the carriers in the workstation. Therefore, the decision which carrier can move onto the conveyor and then to which workstation is going needs to be made on the conveyor before that, or with two successive parallel

workstations conveyors, two conveyors before. For the run-in station, this decision, therefore, need to be made at conveyor *LB405*. This is partly done on the conveyor and on the workstation. These are shown in listings 4.8 and 4.9

**Listing 4.8:** *SIMTALK-Code for sensor at conveyor LB405. Part of method LB_sensor*

```
...
elseif ? = LB405 then
    if not @.empty then
        @.mu.move(?.STon);
    else
        if @.changeover_dummy then;
            waituntil LBrun1.empty and STrun1.empty and LRrun1.empty and LBrun2....
                empty and STrun2.empty and LRrun2.empty prio 1;
        end;
        waituntil ?.nextline.ReadyToMove prio 1;
        ?.ReadyToMove := true;
    end;
...
```

**Listing 4.9:** *SIMTALK-Code for the station ST405. Part of method Station_Exit*

```
...
elseif ? = ST405 then
    waituntil (LBrun1.empty and STrun1.empty and LRrun1.empty and LB501.empty) or (...
        LBrun2.empty and STrun2.empty and LRrun2.empty and LB501.empty) prio 1;
    waituntil ?.lineon.nextline.ReadyToMove prio 1;
    @.move(?.lineon.mu);
    ?.lineon.readytomove := true;
...
```

If the carrier is empty and no changeover dummy, the carrier is sent through to the next conveyor, as it has to move to the conveyor after that and will not cause long time blocking. If it is a carrier dummy, the run-in stations first need to be cleared before the dummy can move further. This is needed at the audit stations to make sure the number of unfinished roof systems is lower when the changeover passes the audit stations. If the carrier is not empty the roof system is sent to the workstation *ST405*. There, after processing, the run-in stations are checked whether either one is empty and a new roof system can come in. Then it waits until the next conveyor can receive and the roof system and carrier are released to go further. When the conveyor activates the sensor at conveyor *LB501* the decision is made to move the carrier to the correct side, as shown in Listing 4.10.

**Listing 4.10:** *SIMTALK-Code for the conveyor LB501. Part of method LB_sensor*

```
...
 elseif ? = LB501 then
    if not @.empty then
        if not @.mu.Runin_OK then
            if LBrun1.empty and STrun1.empty and LRrun1.empty then
                @.move(LBrun1);
                ?.ReadyToMove := true;
            else
                waituntil LBrun2.empty and STrun2.empty and LRrun2.empty prio 1;
                @.move(LBrun2);
                ?.ReadyToMove := true;
            end;
        else
            waituntil ?.nextline.ReadyToMove and LRrepair.empty prio 1;
            ?.ReadyToMove := true;
            @.move(?.nextLine);
        end;
    else
        waituntil ?.nextline.ReadyToMove and LRrepair.empty prio 1;
        ?.ReadyToMove := true;
        @.move(?.nextLine);
```

```
22      end;
23  ...
```

The roof system is checked whether it has done the run-in already or if it still needs to be done. If needed, the stations are checked if empty and the carrier is moved to the conveyor connected to the workstation. If these are already done or the conveyor is empty, the conveyor waits for the next conveyor to be ready to receive. Also the connection conveyor *LRrepair* needs to be empty. If there are repaired roof systems are waiting to be tested again, then these roof systems have priority over the roof systems that are still on the main line. This to make sure that the repair loop is not going to fill up and block the complete production line. With these run-ins finished the roof systems are ready to be tested and checked in the EOL.

### 4.3.5 EOL

The EOL has three main parts. The test stations, the audit boots, and the dimension check. After the roof systems are run in, the roof systems go to the three test stations. These test stations determine, together with the audit booths, the FTT of the production line. The rejects of TLES roof systems come approximately for 60% from the test stations and 40% from the audit booths.[**Senechault**]. The first two test stations have the same functions as a normal basic unit. However, in the third test station, the results of the tests are determined. In Listing 4.11 the code is shown that is used for the decisions at test station 3.

**Listing 4.11:** *SIMTALK-Code for test station 3. Part of method Station_Exit*

```
1  var RandomEOL, randomTest: integer;
2
3      FTT_A := 1/7*(sqrt(21*FTT+4)+2);
4      FTT_T := 1/3*(sqrt(21*FTT+4)-2);
5      STT_A := 1/7*(sqrt(21*STT+4)+2);
6      STT_T := 1/3*(sqrt(21*STT+4)-2);
7      FTT := FTT_A*FTT_T;
8  ...
9  elseif ? = STtest3 then
10     if full_changeover and not third_changeover then
11         if @.mu = void then
12           if @.name = LoadSource.previous_mu then
13             waituntil LBaudit3.empty and STaudit3.empty and LRaudit3.empty and LB505...
                   .empty and LB506.empty prio 1;
14           else
15             waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and LB505....
                   empty) or (LBaudit2.empty and STaudit2.empty and LRaudit2.empty and ...
                   LB505.empty) prio 1;
16           end;
17        else
18           if @.mu.name = LoadSource.previous_mu or @.name = LoadSource.previous_mu ...
                 then
19              waituntil LBaudit3.empty and STaudit3.empty and LRaudit3.empty and ...
                    LB505.empty and LB506.empty prio 1;
20           else
21              waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and LB505...
                    .empty) or (LBaudit2.empty and STaudit2.empty and LRaudit2.empty ...
                    and LB505.empty) prio 1;
22           end;
23        end;
24
25        @.Testdone := true;
26        RandomEOL := Z_uniform(0,1,100);
27        if @.rejectNr = 0 then
28           if RandomEOL < FTT_T * 100 then
29              @.Test_OK := true;
30           else
31              @.Test_OK := false;
32              @.RejectNr := @.RejectNr+1;
33              @.Repair  :=true;
34           end;
```

```
35          elseif @.RejectNr = 1 then
36             if RandomEOL < STT_T * 100 then
37                @.Test_OK := true;
38             else
39                @.Test_OK := false;
40                @.RejectNr := @.RejectNr + 1;
41                @.Repair := true;
42             end;
43          elseif @.RejectNr = 2 then
44             if RandomEOL < STT_T * 100 then
45                @.Test_OK := true;
46             else
47                @.Test_OK := false;
48                @.RejectNr := @.RejectNr + 1;
49                @.Repair := true;
50             end;
51          end;
52          @.move(?.lineon.mu);
53          ?.lineon.readytomove := true;
54       else
55          waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and LB505.empty...
                ) or (LBaudit2.empty and STaudit2.empty and LRaudit2.empty and LB505....
                empty) or (LBaudit3.empty and STaudit3.empty and LRaudit3.empty and LB505...
                .empty and LB506.empty) prio 1;
56          waituntil ?.lineon.nextline.ReadyToMove and LBaudit1.empty and LBaudit2.empty...
                 and LRaudit1.empty and LRaudit2.empty prio 1;
57          @.Testdone := true;
58          RandomEOL := Z_uniform(0,1,100);
59          if @.rejectNr = 0 then
60             if RandomEOL < FTT_T * 100 then
61                @.Test_OK := true;
62             else
63                @.Test_OK := false;
64                @.RejectNr := @.RejectNr+1;
65                @.Repair  :=true;
66             end;
67          elseif @.RejectNr = 1 then
68             if RandomEOL < STT_T * 100 then
69                @.Test_OK := true;
70             else
71                @.Test_OK := false;
72                @.RejectNr := @.RejectNr + 1;
73                @.Repair := true;
74             end;
75          elseif @.RejectNr = 2 then
76             if RandomEOL < STT_T * 100 then
77                @.Test_OK := true;
78             else
79                @.Test_OK := false;
80                @.RejectNr := @.RejectNr + 1;
81                @.Repair := true;
82             end;
83          end;
84             @.move(?.lineon.mu);
85             ?.lineon.readytomove := true;
86       end;
87    ...
```

As can be seen in the listing, at the test station the changeover status is tested first. This is done to determine to which conveyor or audit booth the roof system has to go. If the changeover dummy already passed the test stations, but not all roof systems from the old batch have left the production line, due to rejection and repair, the audit stations run in a different setup. Normally all three audit booths can be used for checks on the roof system. However, the audit booths might have compatible fixtures for the two different roof systems on the production line. If the changeover dummy has passed the audit booth, some roof systems could still be on the repair loop and repair station. Therefore, one of the audit booth stays available until all of the old batch roof systems have left the production line. The checks to determine where the roof systems have to go is done at lines 4 to 16 during the

changeover and lines 48 and 49 if there is no changeover. Then the variables *Testdone* and *Test_OK* are changed depending on the outcome of the tests. The percentage of roof systems that are tested OK for the first and following passages are calculated by the global FTT in lines 4 and 6. Then, using a uniform distribution, a random number is taken and tested for the FTT and STT values. Hereby, in the long run, the percentage of roof systems that pass the test would be equal to the FTT and STT values for the test stations. Also, the number of rejections is updated so the correct percentage for STT is used the next time the roof system passes. If the tests or audit booth reject the roof system three times, the roof system is sent to scrap and leaves the production line. If the tests are OK the roof systems are sent to the audit booths, if not the roof systems go to the next conveyors and then go to the repair loop.
The audit booths use the same principles but with different values. The only difference is when a roof system is tested correctly the variable *Dim_count* is updated, so when 10 roof systems are tested correct, a roof system is sent to *STDim* to check the dimension. The assumption is made that the outcome of this test does not have influence on the production.
When a roof system enters the last conveyor *LB507* there are two options to make. Either the roof system is tested and checked OK, or not. If OK, the roof system is sent to the Unload station and the carrier is sent to the return conveyor LR501 using connection conveyor *LBreturn*. If tested NOK, the carrier with roof system is sent to the repair loop using connection conveyor *LBrepair*. If the changeover dummy activates the sensor at the conveyor, the variable is reset and the carrier is sent to *LBreturn*. The conveyor checks for each roof system if this is the last from the batch. If so the changeover is complete, and the values are reset so a new batch can be made. The code for the sensor at conveyor *LB507* is shown in Listing 4.12.

**Listing 4.12:** *SIMTALK-Code for the conveyor LB507. Part of method LB_sensor*

```
1  elseif ? = LB507 then
2          if not @.empty then
3              if  not @.mu.EOL_OK or not @.mu.Test_OK then
4                  waituntil LBrepair.readyToMove and LBrepair.empty prio 1;
5                  ?.ReadyToMove := true;
6                  @.move(LBrepair)
7              else
8                  waituntil LBreturn.readyToMove prio 1;
9                  ?.ReadyToMove := true;
10                 if @.mu.name = LoadSource.previous_mu or @.mu.name = LoadSource....
                       current_mu then
11                     if @.mu.name = LoadSource.previous_mu then
12                         current_batch_end -=1;
13                     elseif LoadSource.previous_mu= ""
14                         current_Batch_end -=1;
15                     else
16                         next_batch_end -=1;
17                     end;
18                 else
19                     if @.mu.mu.name = LoadSource.previous_mu then
20                         current_batch_end -=1;
21                     elseif LoadSource.previous_mu = ""
22                         current_Batch_end -=1;
23                     else
24                         next_batch_end -=1;
25                     end;
26                 end;
27
28                 @.mu.move(STunload)
29                 @.move(LBreturn)
30             end;
31         else
32             @.changeover_dummy := false;
33             @.move(LBreturn);
34             waituntil LR501.readyToMove prio 1;
35             ?.ReadyToMove := true;
36
37         end;
```

```
38          if current_batch_end = 0;
39              full_changeover := false;
40              current_batch_end := next_batch_end;
41              next_batch_end := 0;
42              LoadSource.previous_mu:=LoadSource.current_Mu;
43          end;
44  ...
```

### 4.3.6 Repair loop

The repair loop is used when a roof system is tested NOK or the checks in the audit booth were NOK. At the repair stations, the roof systems are repaired and the variables that are defined for the tests and audit booth are reset. If the roof systems are rejected for the third time the roof systems are sent to scrap and the carriers continue and go back to the first test station, using connection conveyor *LRrepair*.

With the repair loop, the complete line is discussed. In Appendix A all the methods are shown. To be able to produce the demand given by the customer the process times of the workstations have to be simulated to calculate the cycle times of the workstations and the production line.

## 4.4 Balancing and Throughput tests

To balance the production line, the cycle times of the workstations have to be determined. This can be done either with an analytical method or by using the simulation. The analytical method is easy to use for simple models, in this case the main production line with deterministic process times. When the system becomes more complicated, effects like processing blocking and starvation can occur, which make it difficult to calculate the cycle times. Furthermore, the simulation model is already available so to do the calculations by simulation would require less time than using the analytical method.

The process times of the robots are not calculated but determined from the robots in other production lines. The process times are determined from the robots in the VS20 production line and the Audi/Porsche production line [11]. With these working, the process times can be determined for the different workstations to reach the designed cycle time.
The cycle time experiments on the main production line are performed on the simulation model with only the tested workstation and robots having a process time. For the experiments on the EOL cycle times, the process times of all workstations and robots are used, as determined in the earlier test. This is required as the effects of starvation and blocking need to be included to test the EOL cycle times. These EOL experiments are done with an FTT of 85% and STT of 73%. These numbers are also used by Inalfa for the design of any new production line.

### 4.4.1 Main production line

The main production line has three different types of workstations, the basic unit workstations, the loading workstations and the assembly workstations as discussed in Chapter 3 and the previous section. The experiments are done with a range of process times. These experiments only needed to be done once as the process is deterministic, so each observation of the same experiment gives the same result. The experiments are run over almost 2000 timeperiods. In Figure 4.5 the results are shown for the experiments on the basic unit.

As can be seen in the figure, the cycle time of the production line is a linear function of the process time. This is no surprise as the cycle time for the system is based on the bottleneck basic unit and the cycle time of this based on the transport time of the carriers and the process time of the workstation. The throughput of the system is decreasing with longer cycle times and approaches the asymptote at zero. The designed cycle times for the Honda/JLR roof systems are met with a process time of 0.077 timeperiods and 0.093 timeperiods for the Audi/Porsche roof systems. The same experiments can be done for the loading stations and the assembly stations, for which the results are shown in figures 4.6

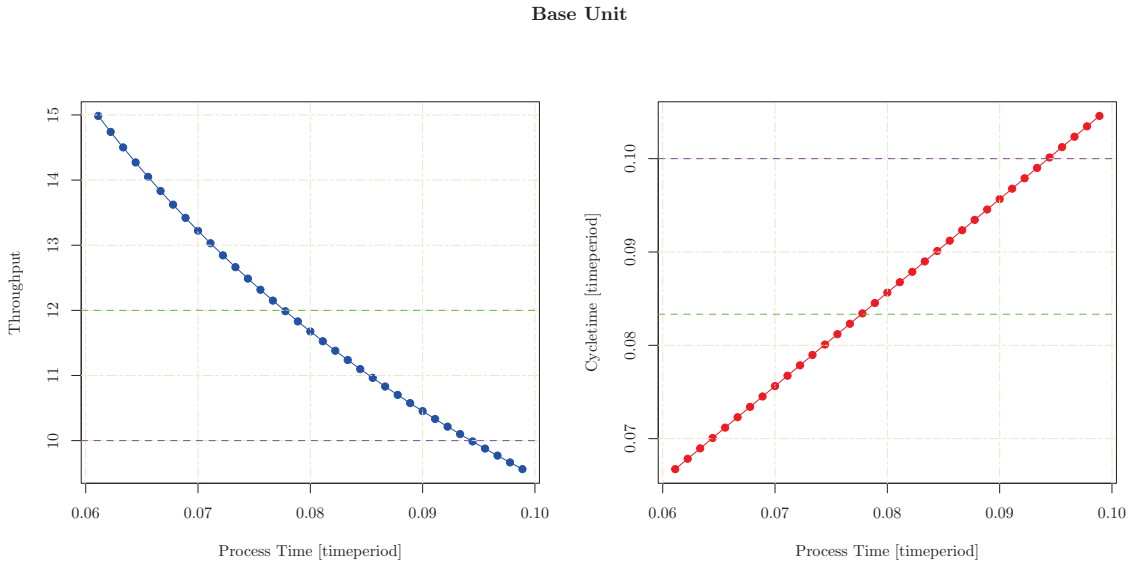



**Figure 4.5:** *The throughput and cycle time of the production line plotted against the process time of a basic unit workstation, as bottleneck of the system. The horizontal dotted lines represent the designed throughput and cycle time of the Honda/JLR(green) and Audi/Porsche roof systems(purple).*

and 4.7 respectively.

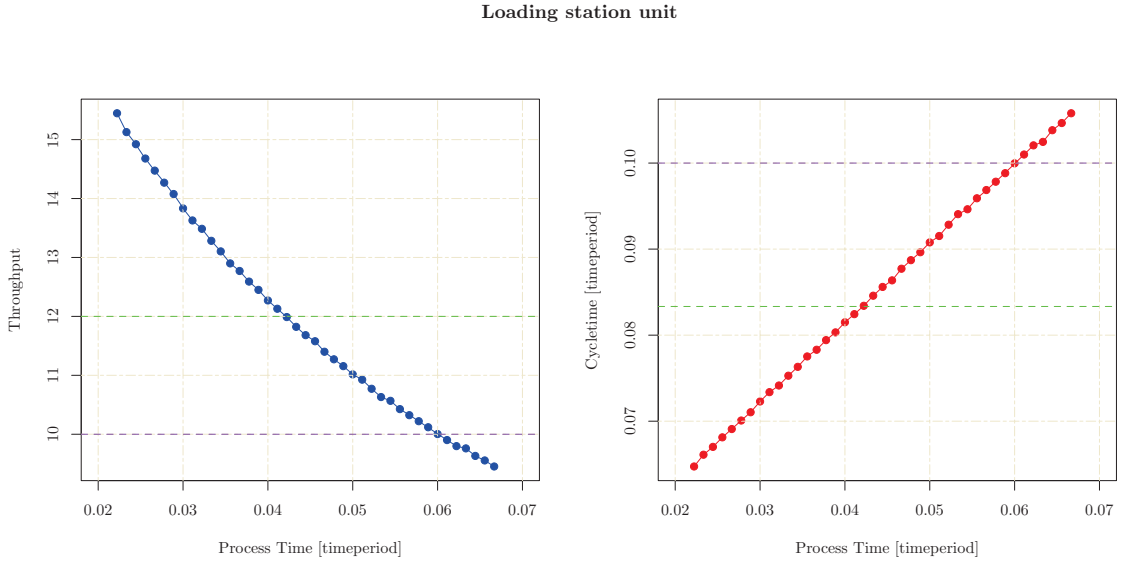



**Figure 4.6:** *The throughput and cycle time of the production line plotted against the process time of a loading workstation(ST201), as bottleneck of the system. The horizontal dotted lines represent the designed throughput and cycle time of the Honda/JLR(green) and Audi/Porsche roof systems(purple).*

As can be seen in the figures, the cycle times follow a linear function, as for the basic unit. Also, the throughputs follow the same function as for the basic model. For the loading stations, the process times need to be 0.041 timeperiods and 0.06 timeperiods for the Honda/JLR and Audi/Porsche roof systems respectively. For the assembly stations, the required process times are 0.032 timeperiods and 0.05 timeperiods respectively. With all these workstations running at the determined process times the main production line can run at the designed cycle time.

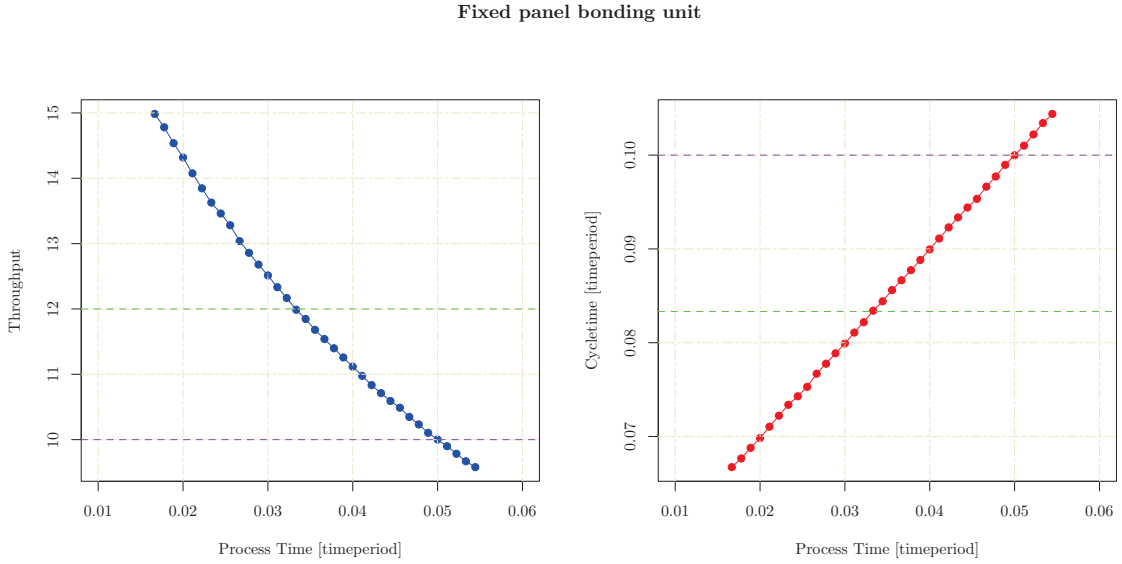



**Figure 4.7:** *The throughput and cycle time of the production line plotted against the process time of an assembly workstation(ST105_fix), as a bottleneck of the system. The horizontal dotted lines represent the designed throughput and cycle time of the Honda/JLR(green) and Audi/Porsche roof systems(purple).*

### 4.4.2 Run-in stations and EOL

The run-in stations and the EOL section are the more difficult sections to analyse. The run-in stations have parallel working stations and use the same conveyor for the entrance and exit of the roof systems. Therefore the cycle times and throughput may not be as straightforward as for the stations on the main production line. The process times for the run-in stations are determined to be equal. In Figure 4.8 the results of the cycle time experiments are shown.

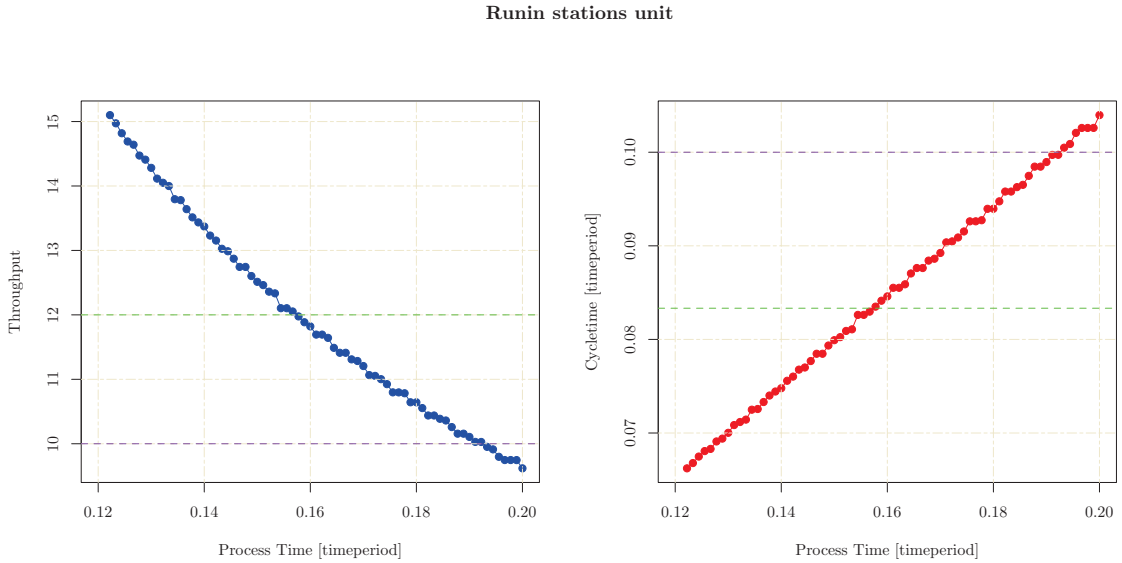



**Figure 4.8:** *The throughput and cycle time of the production line plotted against the process time of the runin stations, as a bottleneck of the system.*

As can be seen in the figure, the cycle time follows the linear trend roughly. The same goes for the

throughput. Because the experiments are run with different process times, the sequence of sending in new roof systems and getting out ready roof systems can change with small differences in process times. These changes in sequence can have a bigger effect negative or positive effect on the throughput than the small change in process times. From the experiments, the process times required to reach the designed cycle time can be determined. However, these process times are not used in the simulation and for the cycle time experiments for the EOL section. Due to blocking and starvation, the throughput of the production line is lower than the maximum throughput of the bottleneck station. This is explained on a simple example in Figure 4.9.

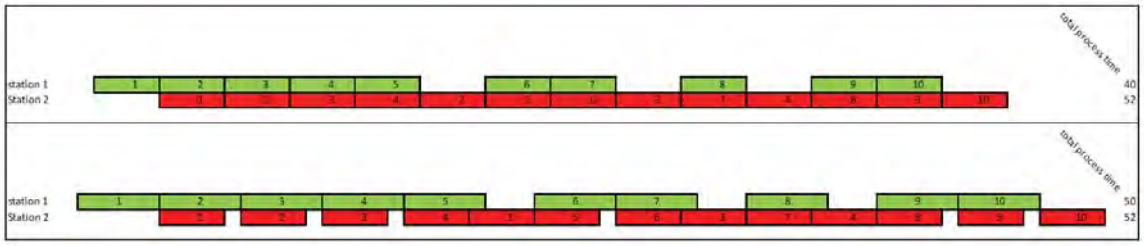



**Figure 4.9:** *Example of starvation and blocking on a simple 2 station process with rework on the second station.*

As can be seen in the figure, the bottleneck of the process is station 2 with 52 time units of production. However, due to the starvation of station 2 and the blocking at station 1, the required process time is longer. The effects of starvation and blocking become greater with cycle times of stations close to the bottleneck and with increasing variance of process times. Therefore the process times of the run-in stations have been decreased to reduce the effect of blocking and starvation. The process times for the Honda/JLR roof systems is 0.117 time periods and 0.133 time periods for the Audi/Porsche roof systems. With these times the EOL section of the production line can be tested.

The EOL section is tested with the given FTT and STT values as is used in the design of a production line at Inalfa Roof Systems. Furthermore, the experiments have to be done for both the Honda/JLR roof systems and the Audi/Porsche roof systems with the process times as determined by the experiments described earlier in this section. These experiments have different characteristics for the throughput and cycle time as the maximum throughput is already determined by the main production line bottleneck. First, the test stations are tested as these stations are most affected by starvation and blocking due to the incoming roof systems from the main production line and the repaired roof systems. The three test stations are given the same process time to balance the system. The results of the experiment are shown in Figure 4.10.

As can be seen in the figure the plot can be divided into 3 sections. First the stable part until around 0.045 timeperiods for the process time, where the main production line is the bottleneck. Secondly, the middle part where the bottleneck is determined by both the main production line and the test stations, and where the throughput is affected by starvation and blocking. Finally the third part, where the test stations are the bottleneck there is a linear function, starting around 0.05 timeperiods. As can be seen the throughput at low process times is already lower than the maximum throughput of the main production line. This is due to the repaired roof systems coming in and blocking the exits of the runin stations and therefore blocking the main production line. If transport was instant this difference in throughput would not be there. The effect is not neutralised by decreasing the process times on the main production line because the research is set up to show the effects of the KPI's at current design features and not with optimised design. The process times is chosen close to the end of part 1, where the bottlenecks of the test stations are close to the bottleneck of the main production line, at an FTT of 85%. The process time for Honda/JLR roof systems is determined at 0.046 timeperiods.

The same experiment can be done for the audit stations, where the same process times are used for

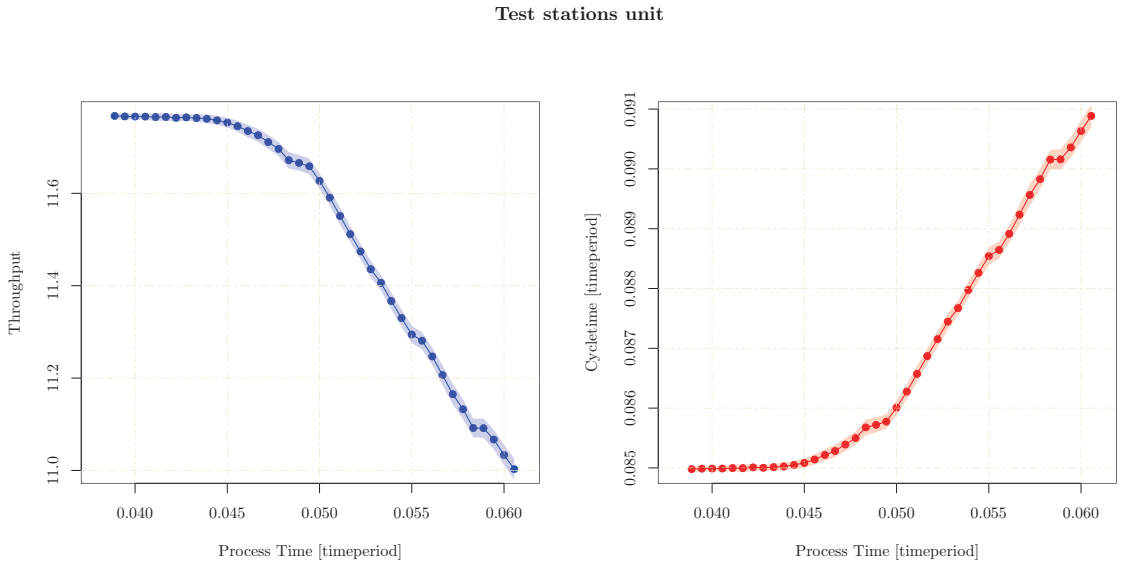



**Figure 4.10:** *The throughput and cycle time of the production line plotted against the process time of the Test stations, as a bottleneck of the system.*

the main production line and the process times for the test stations are used as determined in the last experiment. The results of this experiment are shown in Figure 4.11.

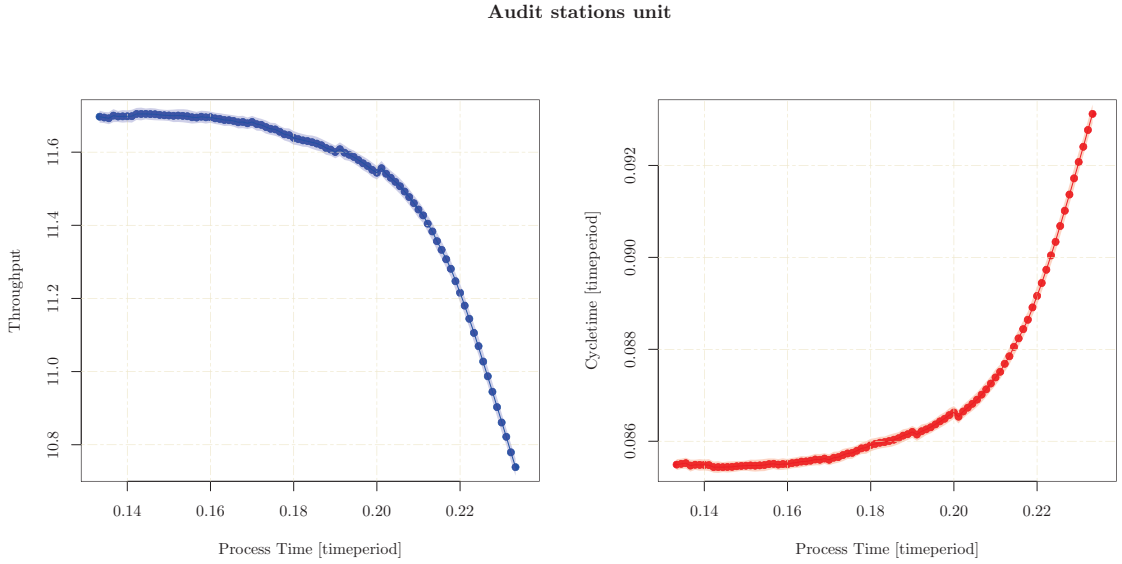



**Figure 4.11:** *The throughput and cycle time of the production line plotted against the process time of the Audit stations, as a bottleneck of the system.*

As can be seen in the figure, the results have the same characteristics as for the test stations. Furthermore, the small jumps in throughput can be explained using the same explanation as for the run-in stations, where the slightly longer process times can create a better sequence, which might decrease the starvation or blocking of a bottleneck station. From these results the process time of the audit stations is determined 0.172 timeperiods, using the same reasoning as for the test stations. These experiments are repeated for the test and audit stations for the Audi/Porsche roof systems resulting in process times of 0.057 timeperiods for the test stations and 0.206 for the audit stations.

With the process times of the workstations determined for the balanced production line, the process times of the workstations for the other roof system groups can be determined. The process times for all the workstations for all roof systems are given in Appendix B. The process times of the workstations are only changed for workstations in the main production line and workstations that could have manual working done at those stations.

## 4.5 Normal test of results

The model can now be simulated, but to test if the results are correct and not affected by something the normality of the results is tested. Therefore, 10000 observations of one experiment were made with an FTT of 85% and a batchsize of 115, which is around half of the maximum batchsize to be able to produce the required production of 400 timeperiods for each roof system type. The results of these observations are shown in a histogram in Figure 4.12 along with a QQ-plot of the results.
As can be seen in the figure, the histogram shows the frequency of the results. This shows that the

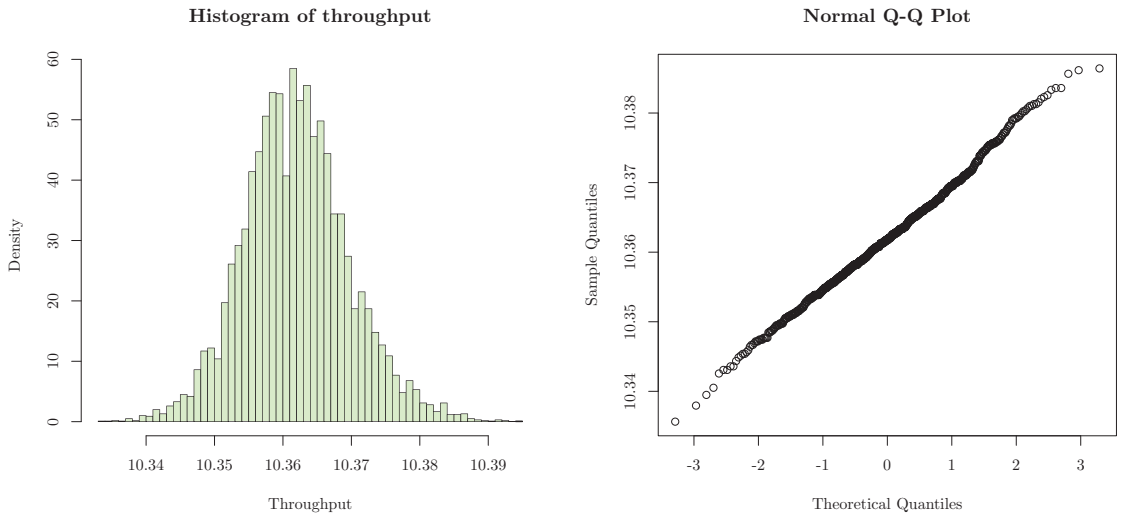



**Figure 4.12:** *Left:Histogram of results from 10000 observations of an experiment. Right:QQ-plot from 10000 observations of an experiment.*

results most likely are normally distributed. Also, the QQ-plot shows that there is no skewness or the results are not tailed. Next to this visual check of normality, the data can also be tested by a Lilliefors test. The Lilliefors test is a test based on the Kolmogorov-Smirnov test and is used to test the null hypothesis that the data is normally distributed, without specifying which normal distribution is used. The Lilliefors test is done with the results of the experiment discussed in this section. If the $p$-value of the test is smaller than 0.05, then with 95% probability it fails to reject the hypothesis. As the $p$-value of the Lilliefors test was smaller than 2.2e-16 and thereby the test fails to reject the hypothesis with high certainty and shows that the data is consistent with an assumption of a normally distributed result. For the experiments of Chapter 5 the normality of the results is assumed, based on the performed normality test on the simulation model.
The normality test concludes the design and testing of the simulation model. This simulation model is used to test the designed experiments of Chapter 2. The results of these experiments are discussed in Chapter 5

# Chapter 5

# Results

With the simulation model made and the research for the KPI's done, the experiments on the KPI's can be done. First, the basis of the experiments is tested and this basis is tested to show how long the simulation should run for. This is discussed in Section 5.1. Then the results for the Quality experiment is discussed 5.2. Here the experiments for different cycle times and FTTs are discussed and the general production is tested for different FTTs. Then the experiments for the Availability KPI are discussed in Section 5.3. The experiments for the Flexibility KPI are discussed in Section 5.4, where the three different changeover strategies are tested. Finally, for the final experiment different ranges of variance are tested and the results are discussed in Section 5.5.

## 5.1 Base result

To set the base result for reference, the time the experiments should run, first have to be determined. If the experiments are run with a short runtime the start-up phase of the process would have a bigger influence than for longer runtime experiments. However, the shorter experiments can be run more often with the same amount of simulation time which gives a better result to show the variance of the process. Longer experiments have a smaller variance compared to the shorter experiments as the influence of stochastic processes, such as the FTT determination. Therefore, the simulation was run for the base setup with three different experiment runtimes. The base setup is with the given cycle times as mentioned in the previous chapter, a changeover time of 0.667 timeperiods and an availability of 100%. The experiments were run with timeperiods of 400, 800 and 4000. The results of these experiments are shown in Figure 5.1.

As can be seen in the figure, the throughput for the tested run times is similar for all experiments. If the throughputs are compared the differences are smaller than 0.3%. However, as the figure shows, the variance in the 2000 timeperiods and 4000 timeperiods is larger than for the 20000 timeperiods experiment. The choice was made to run the experiments with a runtime of 4000 timeperiods as this gave a comparable result to the long 20000 timeperiods experiment, but requires only a fifth of the simulation time, so more observations can be made for each experiment.

With this base result, a reference can be made for the experiments for the KPI's. These KPI's are tested differently but will use the same base setup to start from.

## 5.2 Quality

To test how the Quality KPI influences the Deliverability KPI, a range of FTT values are tested and the throughput is measured. There are three experiments conducted to test this influence. First, the individual roof systems from the Audi/Porsche group are tested for various FTT values. Then the same is done for the Honda/JLR roof systems. And finally, the experiments are conducted with all roof systems at the production line produced in batches and with various FTT values.

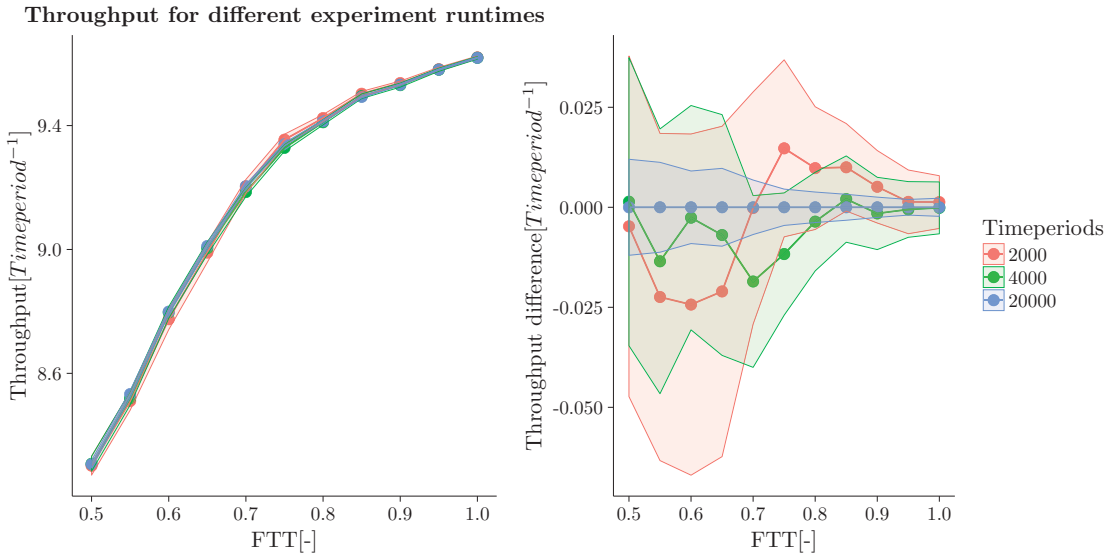



**Figure 5.1:** *Left:Throughput per timeperiod plotted against the FTT for three different experiment runtimes. Right: Difference in Throughput per timeperiod compared to the 4000 timeperiods experiment plotted against the FTT for three different experiment runtimes.*

The first Quality KPI experiment is conducted on the roof systems from the Audi/Porsche group. The experiment is run with batch sizes large enough so no changeover is required during the experiment and only one type of roof system is produced. The four different roof system each have different process times as is explained in Section 2.4 and which are given in Appendix B. In short, the A5 Coupe has balanced process times, the A5 SB has an early bottleneck, the Panamera Limo has a late bottleneck and the Panamera Executive has descending process times. The production of these four roof systems was tested for various FTT values. Within Inalfa the expectation is that the throughput is determined by the EOL bottleneck for FTT values smaller than 0.85 and the main line bottleneck for FTT values bigger than 0.85. These expectations do not take blocking and starvation into account. The results of these experiments are shown in Figure 5.2.

As can be seen in the figure, the throughput of the production line is dependent on both the FTT and the type of roof system. As expected, for low values of the FTT the throughput is almost the same as the EOL is the bottleneck and the EOL is equal for all tested Audi/Porsche roof systems. However, if the values of the FTT go towards 0.85 the slope of the throughput is not increasing anymore. This effect is caused by blocking and starvation of the bottleneck stations. Also for values of FTT over 0.85, the blocking and starvation play a role. This can be seen as the throughput of the production line is not equal for the interval from 0.85 to 1. When the results of the experiments with variance are compared it shows the increased effect of blocking and starvation. This can be seen in the smoother transition of the slope of the throughput against the values of FTT. In addition, the throughput of the experiments with variance in the process times is way lower due to the blocking and starvation effects. The variance in the process times was created by sampling the process times from a gamma distribution with a mean of the original process time and a $\theta$ of 1.0. Why this value is used is explained in Section 5.5

Also, the experiments of the roof systems show some differences. As can be seen in Figure 5.2, the roof system with the balanced process times has the biggest throughput for values of FTT over 0.85 with no variance in the process times, followed by the decreasing process times strategy. This was as expected as the balanced strategy has the lowest designed cycle time. However, if the strategies are compared with the experiments with the variance in process times the descending process times strategy has a better throughput for all values of FTT. Again the cause of this difference can be found in blocking and starvation effects. The descending strategy has lower process times at the end of the main production line and therefore reduces the possibility of starvation and less effect by

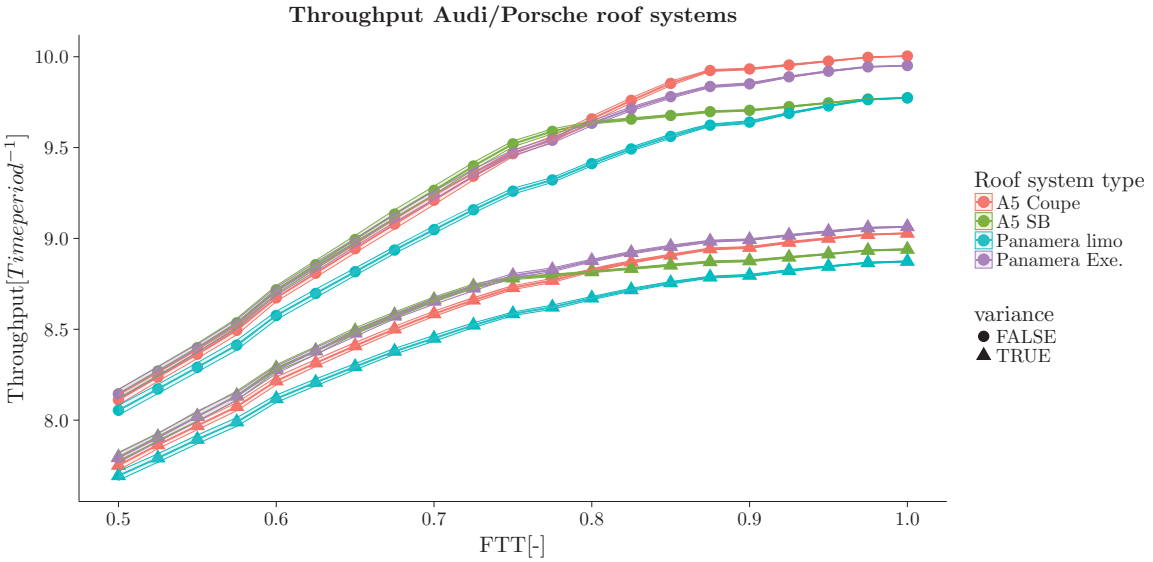



**Figure 5.2:** *Throughput of the production line for various types of roof systems plotted against the FTT. Roof systems are all based on the driveline assembly method.*

blocking as the bottleneck of the mainline is more towards the front. When the other two strategies are compared, namely the early bottleneck and late bottleneck strategy, similar results are visible. The early bottleneck strategy has a better throughput for all values of FTT than the late bottleneck strategy. Only for an FTT of 1, the throughputs are the same, as the designed cycle times are the same. For all other FTT values, the early bottleneck strategy is faster due to the decreased effects of blocking and starvation. For values of FTT between 0.6 and 0.775 the early bottleneck strategy is even the best strategy if no variance is applied on the process times.

The same experiments can be done for the Honda/JLR roof systems. Here the balanced strategy is applied to the Honda TLGA, the early bottleneck strategy to the X260, the late bottleneck strategy to the X760 roof system and the descending process times strategy is applied to the production of the L560 roof systems. The results can be different due to the different flows of production. These results are shown in Figure 5.3.

As can be seen in the figure, the balanced strategy is the fasted strategy for high values of FTT and also for the experiment with variance in the process times. The early bottleneck strategy and the descending process times strategy are faster for lower values of FTT, because of the decreased effect of blocking and starvation. For higher values of FTT, these strategies do not produce as fast as the balanced strategy, due to the higher designed cycle times. When the two different groups of roof systems are compared, the throughput trends are closely comparable for the same strategies. The only big difference can be found in the descending process time strategy. For the Audi/Porsche group, this strategy is the best performing strategy for all values of FTT when the variance is taken into account. However, for the Honda/JLR group, this strategy does not perform better than the balanced strategy for higher values of FTT with variance taken into account. No further research was done to investigate this difference. The expected reason for this is that in for the Honda/JLR roof systems the blocking and starvation of the early process in the main production line are affecting the throughput more than the blocking and starvation between the main line and the EOL. This would also explain the reason that the early bottleneck and descending process time strategy are so similar for the experiment with variance taken into account.

The third experiment is to have a full production of all roof systems. The roof systems are produced by batch, with the size of the batch determined by the ratio determined by the demand. For this test, the roof systems are produced in batch sizes so that two full rounds of batches should be equal to the

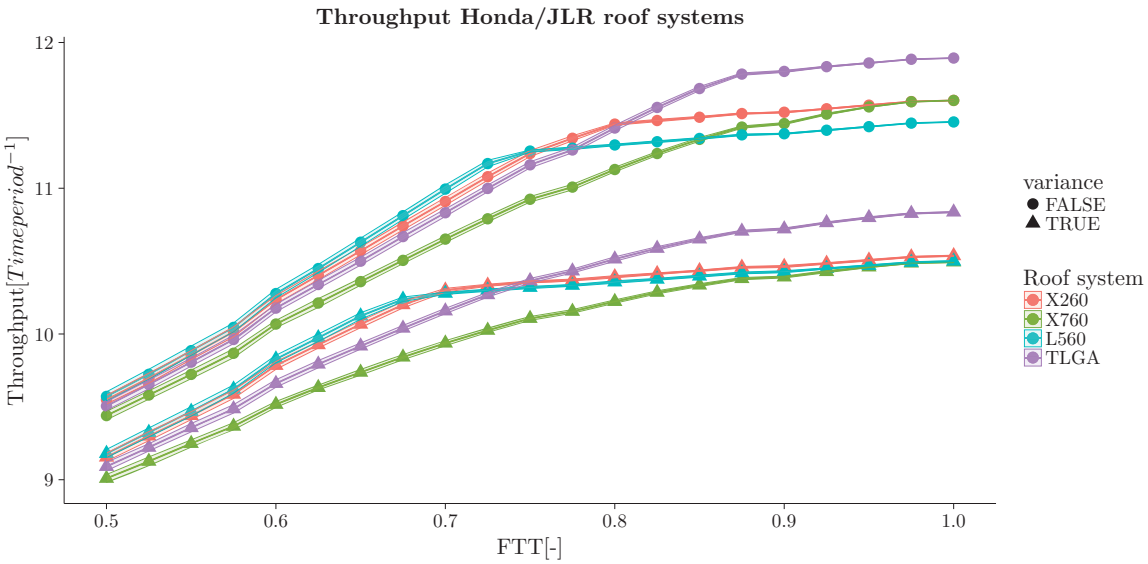



**Figure 5.3:** *Throughput of the production line for various types of roof systems plotted against the FTT. Roof systems are all based on the bathtub assembly method.*

demand of 400 time periods. Furthermore, the same settings are used as for the base experiment. The results of this experiment are shown in Figure 5.4.

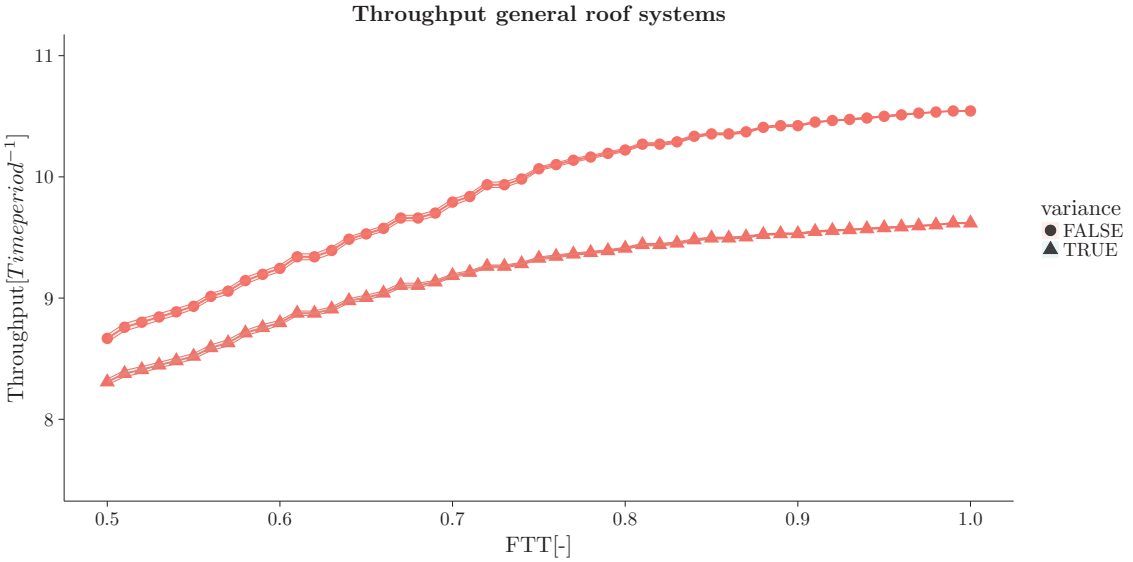



**Figure 5.4:** *Throughput of the production line for full production of all roof systems types, plotted against the FTT.*

As can be seen in the figure, the expected throughput of 10.7 is never met, not even with an FTT of 1.0. Furthermore, the figure shows that the increase of throughput at higher values of FTT is more flattened. This effect is caused due to the introduction of change over. As more changeovers happen with more production the percentage of time in changeover increases and therefore the increase of throughput with higher values of FTT is smoothed out. Furthermore, it shows that the variance has a big influence and at an FTT of 0.85 the influence of variance decreases the throughput with 8.3%.

## 5.3 Availability

To test how the Availability KPI affects the Deliverability KPI the production line is tested with various values for the availability and the MTRR as explained in Section 2.3. First the assumption made for the parallel workstations is tested, whereafter the experiment is done for the whole production line. For the first test, the parallel workstations of the run-in stations are used. In this experiment, only the run-in stations have a process time so blocking and starvation from other workstation does not have an effect on the process. The workstations are tested with various ranges of availability and MTTRs to test if the proposed availability calculation for parallel workstations also work for this simulation model and therefore are applicable to the production lines at Inalfa roof systems.

The results of this showed similar values for equal ratios of availability but different MTTR's. This is as expected as the availability is equal and there is no blocking of other stations. So to show the availability of the subsystem the throughputs of the tests are compared to the throughput of a test with an availability of 1, so the maximum throughput. This method is currently used at Inalfa to determine the availability of the production line. During the maximum throughput test utilization of the workstation is 90.2%. This is not 100% due to the transportation of the conveyors around the workstations. The result for the simulation and the theoretical values are shown in Figure 5.5.

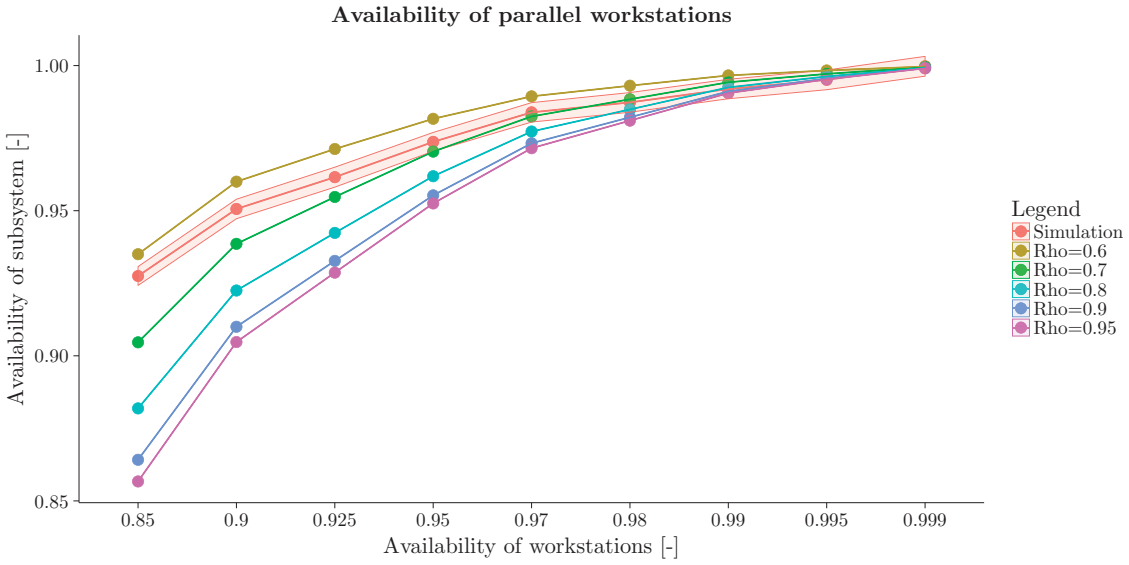



**Figure 5.5:** *Availability of the subsystem plotted against the availability of the parallel workstations for the experiments and the theoretical values.*

As can be seen in the figure, the availability of the subsystem is much higher for low availabilities of the workstation than expected from the theoretical values. However, this is not caused by the improvement of availability at lower values but by the decreased throughput at higher values of availability. Because the throughput is higher at these higher values there is more blocking at the conveyor between the workstations the throughput is suppressed. Therefore, the ratio between the throughputs at lower and higher values is skewed. This shows that it is not possible to use this simple method to calculate the availability of parallel workstations for such production lines. This shows that simulation is best used to calculate the availability of the subsystem before the proposed analytical method. A better analytical method can be used, but it is not clear how much work this would require and if it is possible.

For the second experiment, the availability of the workstations is taken equally for all workstations except for the parallel workstations, where the substituted availability is set equal. Therefore, in theory, the availability is the product of 28 workstations and substituted workstations as explained in Section 2.3. The experiment is performed with the same values as for the base experiment and FTT of 0.85. The experiment is conducted with a range of Availabilities for the whole production line, with

for each value a range of MTTRs. In theory, the throughput should be equal for the observations with the same availability but with different MTTRs. The results are shown in Figure 5.6.

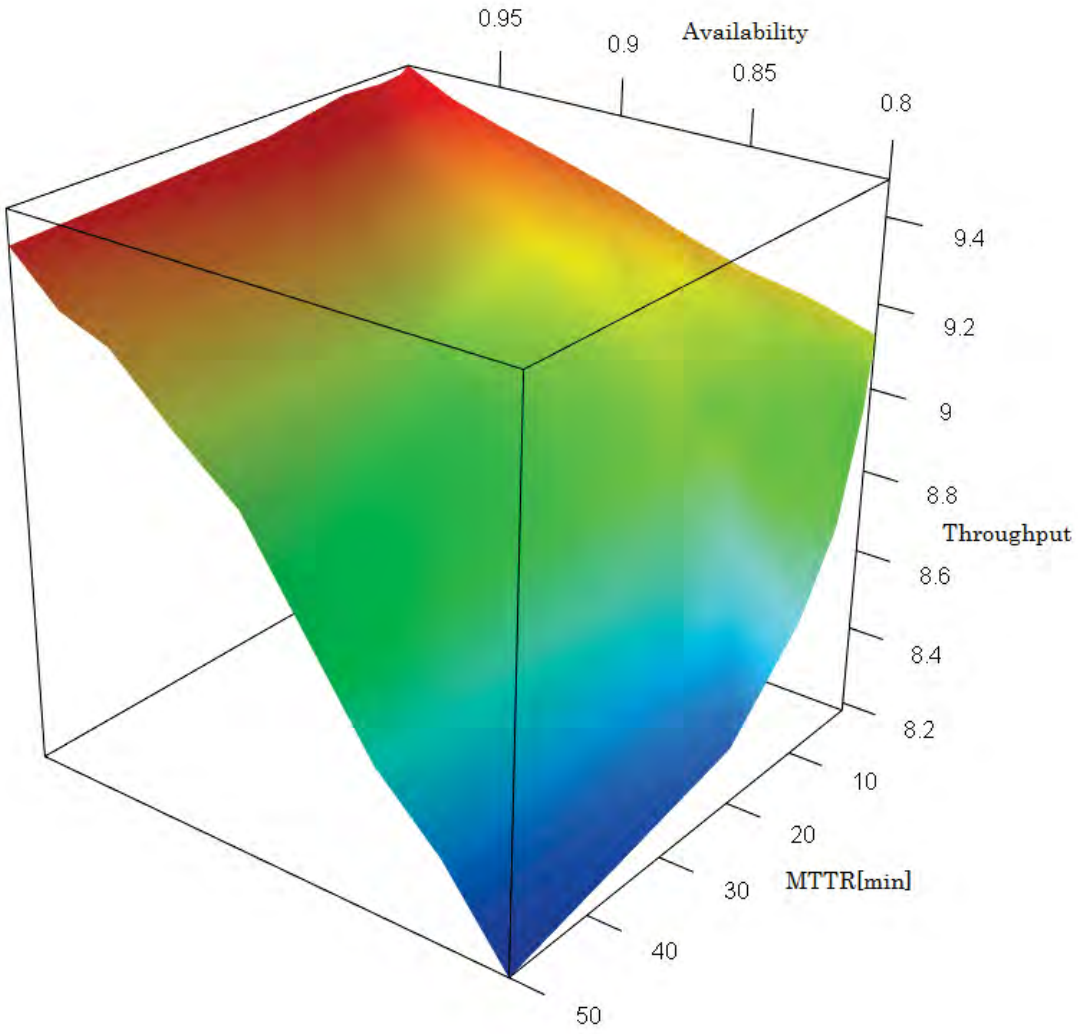



**Figure 5.6:** *3D plot of the throughput of the production line against the availability of the production line and the MTTR of the workstations.*

As can be seen in the figure, the throughput is dependent on both the Availability as well as the MTTR. In theory, the MTTR should not affect the throughput, with equal availability. However, there is a flaw in this. The assumption in the theory is made that when a single workstation is down, no production is made on the production line. In practice, if one workstation is failing and has stopped the other workstations could keep working until they fail or until they are blocked or starved. This effect becomes bigger with smaller MTTRs and therefore the availability of the production line will not decrease as much as expected by the failing of one workstation. This, however, shows the complexity of the availability calculation and how important it is to investigate how the current production lines are performing and what availability is reached on current workstations. It also shows that it is nearly impossible to calculate the availability of the production line using analytical solving methods and herein simulation can give a big advantage.

## 5.4 Flexibility

The Flexibility KPI is tested with three case studies. Herein each of the changeover strategies is tested. These strategies are explained in Section 2.4. The first strategy have a fixed changeover time while the second strategy has a fixed and delayed changeover. The third strategy only has a single empty carrier as changeover.

### 5.4.1 Case study F1

The first case study is the changeover strategy with a fixed changeover time of 0.667 and 2 time periods at the glass setting workstation(ST401). This strategy is tested for a range of batch sizes. The range is from 50, which is the lowest possible batch sizes by assumptions to 225 roof systems, the number of roof systems required to produce each roof system for a 400 time period demand. These batch sizes are for the roof system type with the smallest demand, namely the Honda TLGA. The other roof systems have batches accordingly to the demand ratio. Expected is that the throughput increases with bigger batches as there are fewer changeovers required to produce the same amount of roof systems. To decrease the influence of the differences in cycle times for the different roof systems the experiments are averaged over observations with different starting roof systems. In Figure 5.7 the results of the experiments with a 0.6667 timeperiods changeover are shown.

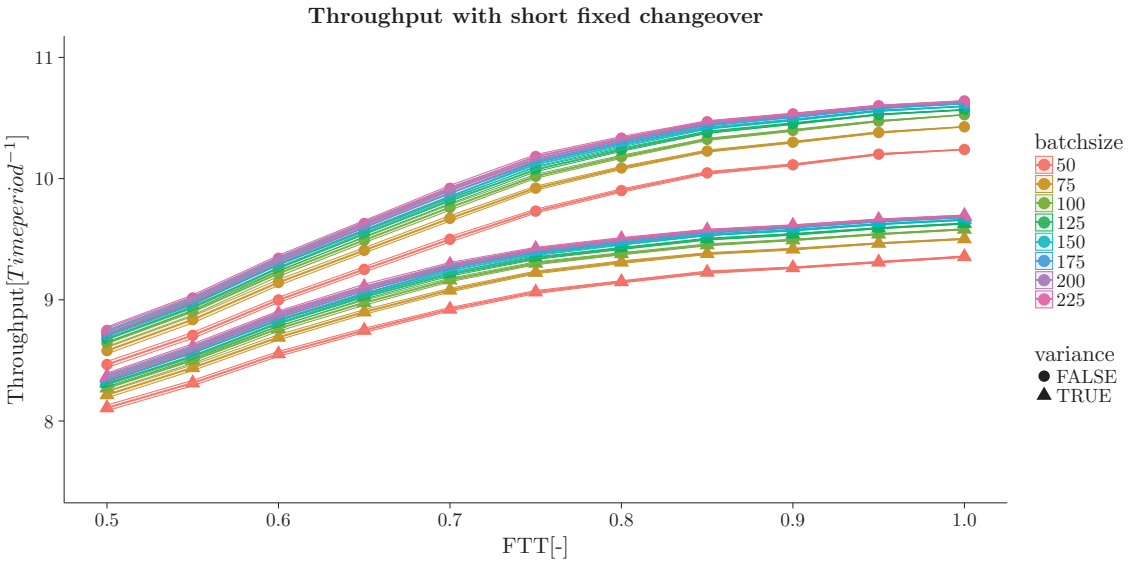



**Figure 5.7:** *Throughput of the production line plotted against the FTT for a range of batch sizes. Fixed changeover time of 0.667 timeperiods.*

As can be seen in the figure, the throughput increases with the batch sizes, as expected. This gain is only marginally for batch sizes above 100. This could be expected as the changeover is only small and has little effect on the total throughput. This might be different for the next experiment.
This experiment is done with the same settings as the first experiment, only with a longer fixed changeover time of 2 timeperiods on the glass setting workstation. Expected is that the throughput is lower for this case study compared the short changeover, as the changeover times are longer. Therefore, the differences in changeover between the batch sizes is also different as the effect by increasing the number of changeovers is bigger. The results of the experiments are shown in Figure 5.8.

As can be seen in the figure, the throughput is a lot lower compared to short fixed changeover strategy. Also, the difference in throughput between the batch sizes is a lot bigger. As predicted this is because the number of changeovers has a bigger effect when the changeovers are longer. Furthermore, two interesting effects can be seen in Figure 5.8. First, the difference between the experiments with or without the variance in the process times are not as big as for the other experiments done. In

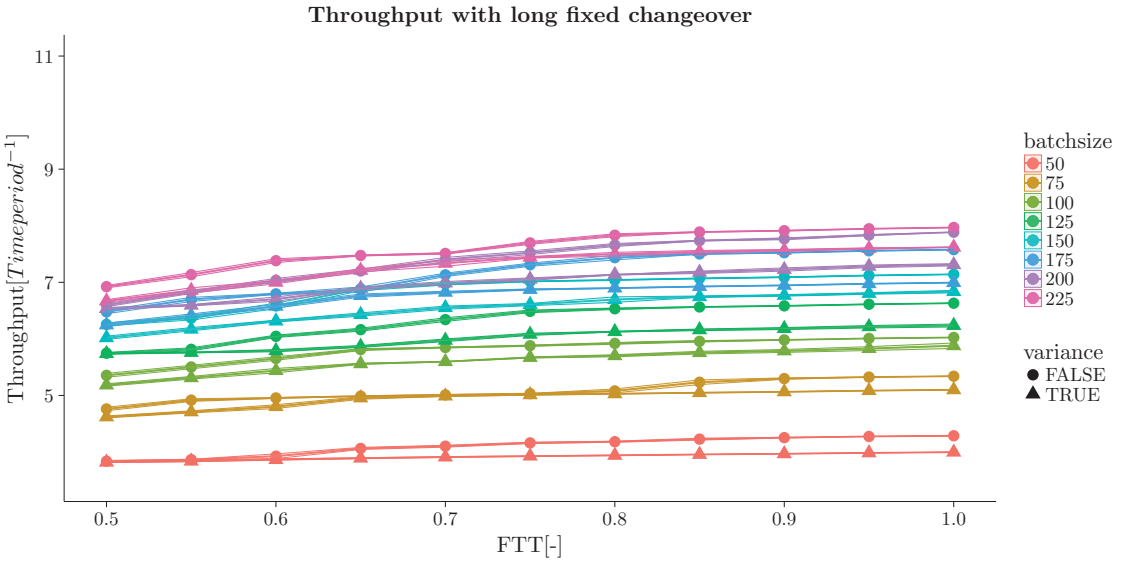



**Figure 5.8:** *Throughput of the production line plotted against the FTT for a range of batch sizes. Fixed changeover time of 2 timeperiods.*

some cases, there is almost no difference. The second effect is the slope of the throughput against the FTT. The slope is far less steep as for the other experiments for low values of FTT, especially for small batch sizes. This effect is caused by the decreased effect of blocking. When the changeover is happening, the EOL has the possibility to clear the repair loop and newly repaired roof systems without blocking the main production line. As the changeover is a lot longer than the remaining process time for the roof systems after the changeover glass setting workstation, for that period the FTT values is almost negligible for the throughput. For bigger batch sizes the ratio of this work is smaller compared to the total work of the batch and therefore, this effect is bigger for smaller batch sizes. The same phenomenon can also be used to explain the small difference in the throughput for the experiments with or without variance on the process times. For the third case study, these fixed changeover times are neglected and does the changeover take place at the beginning of the carrier systems.

### 5.4.2 Case study F2

With case study F2, the changeover strategy with changeover carriers is tested. Hereby the top plates of the carriers are exchanged which lead to extra work in the first workstation of the carrier systems. This delay is 0.044 time periods, which is about 50-65% extra process time for station ST101. This delay happens with all carriers after a changeover until all carriers in the system are exchanged. The results of the case study experiments are shown in Figure 5.9.

As can be seen in the figure, the throughput is again increasing with bigger batch sizes. Herein is the strategy comparable to the short fixed changeover strategy in case study F1. The throughput for lower values for FTT is only slightly affected by the batch size. Because for lower values of FTT the EOL is the bottleneck, the delay in the main production line does not have that much effect on the throughput. Only for higher values of FTT, it can be seen that the differences become a lot larger for smaller batch sizes as here the main production line is the bottleneck so the first stations become the bottleneck during the changeover phase. This effect is increased for smaller batch sizes as the ratio of batch size to carriers becomes much larger and therefore a bigger ratio of roof systems has a longer process time. This also effects the throughput for small batch sizes at lower FTT values as the bottleneck of the main production line is becoming slower.

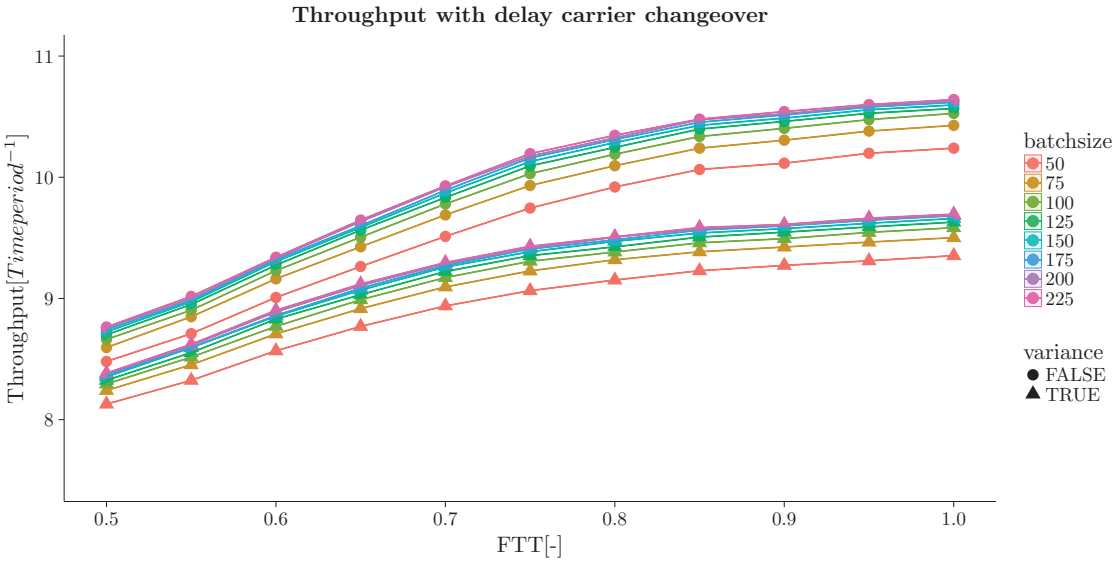



**Figure 5.9:** *Throughput of the production line plotted against the FTT for a range of batch sizes. Changeover time of 0.044 timeperiods for each carrier.*

### 5.4.3 Case study F3

Finaly in case study F3 the changeover strategy with only one takt time is tested. Here the batches are only separated by an empty carrier and no changeover times. Expected is that this strategy is the fasted strategy as no changeover of tooling is required. The results for this experiment are shown in Figure 5.10.

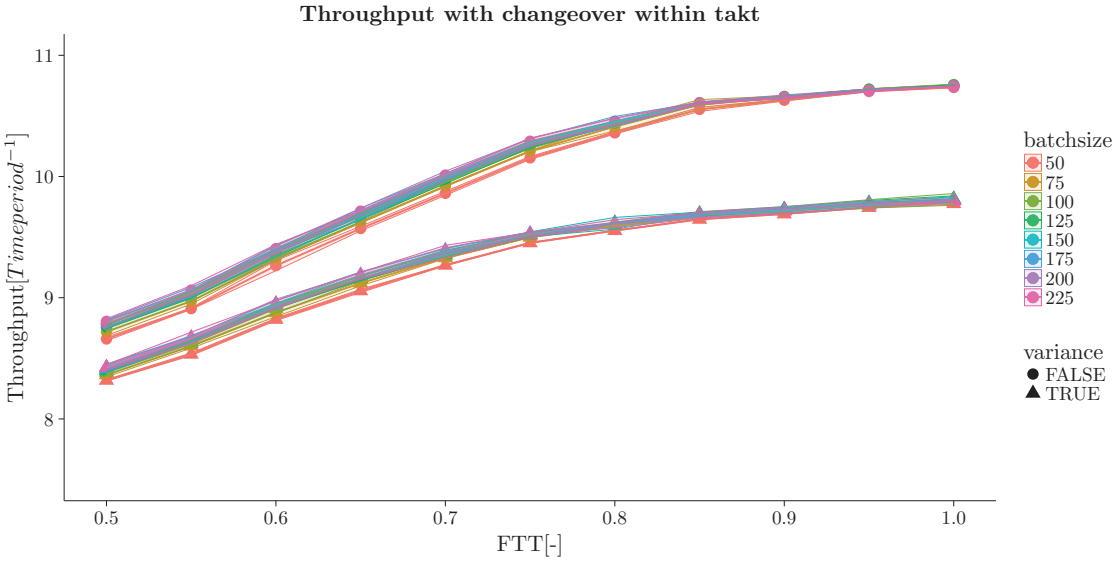



**Figure 5.10:** *Throughput of the production line plotted against the FTT for a range of batch sizes. Changeover of one takt time.*

As can can be seen in the figure, the throughput of the production line is almost equal for all batch sizes. Only for small values of FTT the smaller batches have a slightly lower throughput. This is caused due to the inefficient allocation of the roof systems during changeover in the audit stations. The system could be tested for smaller batch sizes due to the simulation model, which would be more interesting to test for this changeover strategy.

In all the Flexibility KPI case studies and all experiments of the other KPI's the experiments were conducted with process times with and without variance taken into account. And in almost all cases the difference in throughput can be called significant. Therefore, in the next section, the influence of variance is discussed.

## 5.5 Variance in process time

Currently, the variance of process times is not taken into account at Inalfa. During the research, the influence of variance came back several times and it was interesting to investigate the influence on the performance of the production line. In the previous section, the experiments where variance was accounted for the process times were sampled from a gamma distribution with $\theta$ of 1.0. However, to find the real distribution for the process times would require too much time as this data is not gathered from the current production line at this moment. By Joop van de Ende[15] information was gathered and by his opinion, the use of a $\theta$ of 1.0 would be well within the margin of the current situation. Therefore, this value is used to show what even this variance would do to the performance. To illustrate this value, some gamma distributions with a mean of 80 seconds are plotted in Figure 5.11.

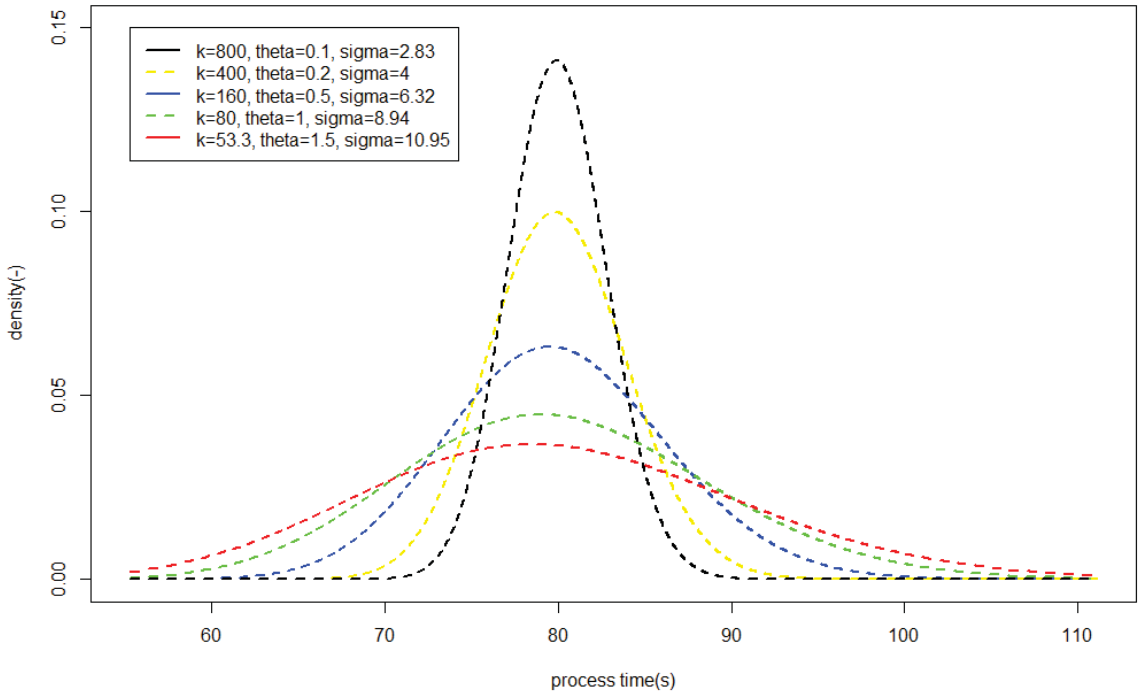



**Figure 5.11:** *Gamma distributions with means of 80 seconds and different standard deviations.*

As can be seen in the Figure, when the $\theta$ of the gamma distribution becomes greater, the spread of process times increases. For the gamma distribution with theta is 1.0 the standard deviation is 8.9 seconds, meaning that the 68% of the sampled process time has a value within 71 seconds and 89 seconds. When these gamma distributions are used for an experiment the difference in throughput can be shown for different values of variance. These results are shown in Figure 5.12.

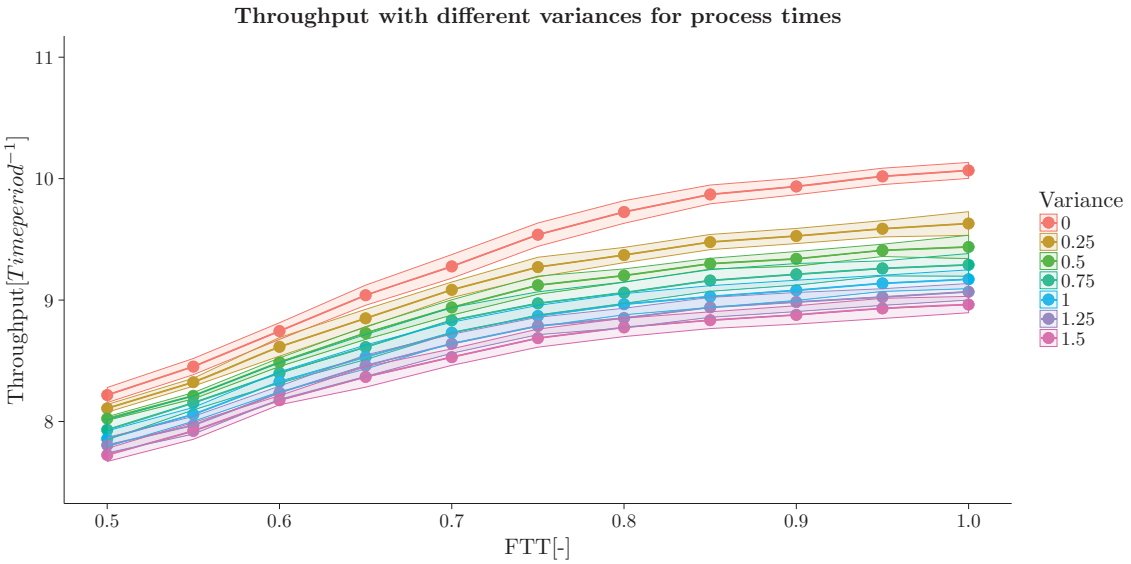



**Figure 5.12:** *Throughput of the production line plotted against the FTT for a range of variances on the process times.*

As can be seen in the figure, the throughput of the production line decreases with higher variance in the process times. This is caused by more blocking and starvation. The results show that for lower values of FTT, the difference is smaller than for higher values of FTT. Because at low values of FTT the EOL is the bottleneck of the production line the effects of blocking and starvation in the main line do not have the same effect as the whole system is blocked by the EOL. At higher values, the blocking and starvation of the main line matter more because this part of the line is now more responsible for being the bottleneck. The results show that even for low variances the difference in throughput is significant.

The results of the experiments show some interesting effects. For instance,blocking and starvation have a big effect on the throughput of the system and can not be calculated easily with an analytical method. Furthermore it shows that the availability of the system is difficult to predict with analytical methods and simulation offers a good solution. As far as the batch sizes determine the throughput is very dependent on the used changeover strategy, as the effect of the batch sizes scaled from big to almost none for the various strategies. Finally, the results showed the big impact of variance in the process times on the throughput of the system, even for small variance. This shows that variance can not be neglected as a factor in the design of a production line.

# Chapter 6

# Conclusion and Further Recommendation

This research has shown some interesting points within the company of Inalfa Roof Systems. The research into the KPI's showed that there is a lot to gain for Inalfa in collecting data to do analysis on the current production lines. The research showed that the targets set by the management are rarely met and this wonders the question whether designing a production line with the targets is the right strategy to go. For the Quality KPI the targets of 85% and 95% FTT are rarely met, and for long time production(>2 years) an average of 80% is reached.
While simulation is currently not used much at Inalfa, the possibilities in terms of manageable models of production lines and simulation tools are available and therefore using simulation to investigate the production lines could be done more. This is shown by the results and findings done in this research on the basis of simulation. The current production lines are designed for production with an FTT value of 85%, so therefore the main production line and the EOL are balanced for cycle times. However, due to the effect of blocking and starvation, the throughput of the production line is decreased the most at the best balanced point. These effects are currently neglected by the company in the design phase of the production line, while with simulation these effects of blocking and starvation can be calculated. The results of the simulation also show that the flow of the production line can influence what the best strategy of balancing the line for process times is. This can only be done with simulation for such complex production lines. Next to the effect of flow, the effect of variance in the process times is a variable which influences the choice of this strategy. The variance of process times is currently neglected by Inalfa, but as the results show, this can not be neglected as this can lead to reductions of throughput by 10-20%, dependent on the setup of the production line. This was calculated for values of variance which are within the scope of possibility for the current production lines. However, at this moment within Inalfa there is little knowledge about what the variance of the process times at the workstations really is. The complexity of the production lines can also be seen in the Availability KPI simulation results. Due to the complexity of the production line, the assumed theory of the research can not be used for analysing the availability of the production line. The only method that can be used is to use the data from current production lines and workstations to predict the availability of stations. From those results, the availability of the production line can be calculated using simulation as a tool. Currently, the changeovers of production are neglected as a factor to calculate the required cycle times for the workstations. The results of this research show that the strategy used for changeover can heavily influence the throughput of the production line. In addition, the batch sizes that are used in production can also influence the throughput. These factors are very difficult to calculate with analytical solutions and therefore the use of simulation is needed.

At Inalfa Roof Systems a lot of variables and effects are neglected at the design phase of the production line. These neglections are mostly taken into the "flex" of the production line, which is an option to be able to produce 20% more than the requested demand. This research shows the possibilities for the use of simulation to analyse current production lines and improve the design of new production lines. The improvements that would come from this would decrease the necessity for the "flex" to be as large

as it currently is. Furthermore, the use of simulation can prevent problems with flow or bottleneck workstation or robots, as can be seen in current production lines. The recommendation that can be made from this research is to introduce and implement the use of simulation for new production lines early in the design phase to improve the throughput of the production lines and decrease unwanted effects such as blocking and starvation. Furthermore, the quality of the simulation would improve if more research was done in the KPI's at current lines and more data analysis was done on current lines such as variance in process times. This would also help to better predict the deliverability of new production lines. The last recommendation that can be made is to have a research made into the implementation of buffers at the production line. During this research, the time was limited, but the necessity for the research into the implementation of buffers became clear for all involved parties. The research should mostly focus on the implementation of buffers between the main production line and the EOL, as here the blocking and starvation occur the most.

# Bibliography


[1] C. Lohman, L. Fortuin, and M. Wouters. "Designing a performance measurement system: A case study". In: *European Journal of Operational Research* 156.2 (2004), pp. 267–286.

[2] L. Fortuin. "Performance indicators: why, where and how?" In: *European journal of operational research* 34.1 (1988), pp. 1–9.

[3] S. Nakajima. "TPM tenkai". In: *Japan Institute of Plant Maintenance, Tokyo* (1982).

[4] S. Nakajima. "Introduction to TPM: Total Productive Maintenance.(Translation)". In: *Productivity Press, Inc., 1988,* (1988).

[5] D.A. Collier and J.R. Evans. *Operations management*. Cengage Learning, 2009.

[6] Author Unknown. *Labor Optimzation Work Stream Global Standard — Bol, KPIs and Budgets*. Tech. rep. Unpublished internal document. Infinity Gear.

[7] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2013. URL: `http://www.R-project.org/`.

[8] R.B. Abernethy et al. "The New Weibull Handbook Fifth Edition, Reliability and Statistical Analysis for Predicting Life, Safety, Supportability, Risk, Cost and Warranty Claims". In: *Published and distributed by Robert B. Abernethy* (2006).

[9] J.L. Romeu. "Understanding series and parallel systems reliability". In: *Selected Topics in Assurance Related Technologies (START), Department of Defense Reliability Analysis Center (DoD RAC)* 11.5 (2004).

[10] J. Engelen. *Maintenance Manager, Inalfa Roof Systems*. personal communication.

[11] B. Zhou. *Process Engineer, Inalfa Roof Systems*. personal communication.

[12] D.M. Upton. "The management of manufacturing flexibility". In: *California management review* 36.2 (1994), pp. 72–89.

[13] H. Bastiaanse. *Manager Advanced Process Engineering, Inalfa Roof Systems*. personal communication.

[14] *Tecnomatix plant simulation*. `https://www.plm.automation.siemens.com/en/products/tecnomatix/manufacturing-simulation/material-flow/plant-simulation.shtml`. Accessed: 2017-06-07.

[15] J. van de Ende. *Senior Manufacturing Engineer, Inalfa Roof Systems*. personal communication.

# Appendix A

This appendix contains all code used to make the simulation model. This is the simulation model as explained in Chapter 4. The listings shown in that chapter are parts of the complete codes as represented here.

**Listing A.1:** *SIMTALK-method for initialization. Method is started at every new simulation.*

```
1  loadsource.previous_mu:="";
2  loadsource.current_mu:="";
3  LB101.readytomove:=True;
4  LB102.readytomove:=True;
5  LB103.readytomove:=True;
6  LB104.readytomove:=True;
7  LB105.readytomove:=True;
8  LB201.readytomove:=True;
9  LB202.readytomove:=True;
10 LB203.readytomove:=True;
11 LB204.readytomove:=True;
12 LB301.readytomove:=True;
13 LB302.readytomove:=True;
14 LB303.readytomove:=True;
15 LB304.readytomove:=True;
16 LB305.readytomove:=True;
17 LB301.readytomove:=True;
18 LB302.readytomove:=True;
19 LB303.readytomove:=True;
20 LB304.readytomove:=True;
21 LB305.readytomove:=True;
22 LB401.readytomove:=True;
23 LB402.readytomove:=True;
24 LB403.readytomove:=True;
25 LB404.readytomove:=True;
26 LB405.readytomove:=True;
27 LB501.readytomove:=True;
28 LB502.readytomove:=True;
29 LB503.readytomove:=True;
30 LB504.readytomove:=True;
31 LB505.readytomove:=True;
32 LB506.readytomove:=True;
33 LB507.readytomove:=True;
34 LR41.readytomove:=True;
35 LR42.readytomove:=True;
36 LR43.readytomove:=True;
37 LR44.readytomove:=True;
38 LR45.readytomove:=True;
39 LR501.readytomove:=True;
40 LR502.readytomove:=True;
41 LR503.readytomove:=True;
42 LR504.readytomove:=True;
43 LR505.readytomove:=True;
44 LR506.readytomove:=True;
45 LR507.readytomove:=True;
46 LR31.readytomove:=True;
47 LR32.readytomove:=True;
```

```
48  LR33.readytomove:=True;
49  LR34.readytomove:=True;
50  LR3.readytomove:=True;
51  LR21.readytomove:=True;
52  LR22.readytomove:=True;
53  LR23.readytomove:=True;
54  LR24.readytomove:=True;
55  LR25.readytomove:=True;
56  LR11.readytomove:=True;
57  LR12.readytomove:=True;
58  LR13.readytomove:=True;
59  LR14.readytomove:=True;
60  LR15.readytomove:=True;
61  LR16.readytomove:=True;
62  LR17.readytomove:=True;
63  LR18.readytomove:=True;
64  LRrep1.readytomove:=True;
65  LRrep2.readytomove:=True;
66  LRrep3.readytomove:=True;
67  LRrep4.readytomove:=True;
68  LRrep5.readytomove:=True;
69  LRrep6.readytomove:=True;
70  LBrun1.backwards:=false;
71  LBrun2.backwards:=false;
72  LBrun1.readytomove:=True;
73  LBrun2.readytomove:=True;
74  LRrun1.readytomove:=True;
75  LRrun2.readytomove:=True;
76  LBaudit1.readytomove:=True;
77  LBaudit2.readytomove:=True;
78  LRaudit1.readytomove:=True;
79  LRaudit2.readytomove:=True;
80  LBaudit1.readytomove:=True;
81  LBaudit2.readytomove:=True;
82  LBaudit3.readytomove:=True;
83  LBdim.readytomove:=True;
84  LBreturn.readytomove:=True;
85  LBrepair.readytomove:=True;
86  LRrepair.readytomove:=True;
87  LB101.readytoload:=false;
88  LB_mar.readytoload:=false;
89  Ending:=false;
90  End_int:=0;
91  Planning[2,1]:=0;
92  Planning[2,2]:=0;
93  Planning[2,3]:=0;
94  Planning[2,4]:=0;
95  Planning[2,5]:=0;
96  Planning[2,6]:=0;
97  Planning[2,7]:=0;
98  Planning[2,8]:=0;
99  Planning[2,9]:=0;
100 Planning[2,10]:=0;
101 Planning[2,11]:=0;
102 Planning[2,12]:=0;
103 Planning[2,13]:=0;
104 Planning[2,14]:=0;
105 Planning[2,15]:=0;
106 Planning_drive[2,1]:=0;
107 Planning_drive[2,2]:=0;
108 Planning_drive[2,3]:=0;
109 Planning_drive[2,4]:=0;
110 Planning_drive[2,5]:=0;
111 Planning_drive[2,6]:=0;
112 Planning_drive[2,7]:=0;
113 if batch_start=1;
114 Planning[2,1]:=batch_X260*batch_size;
115 Planning[2,2]:=batch_X760*batch_size;
116 Planning[2,3]:=batch_L560*batch_size;
117 Planning[2,4]:=batch_TLGA*batch_size;
```

```
Planning[2,5]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,1]:=batch_A5C*batch_size;
Planning_drive[2,2]:=batch_A5SB*batch_size;
Planning_drive[2,3]:=batch_por_exe*batch_size;
Planning_drive[2,4]:=batch_por_limo*batch_size;
elseif batch_start=2;
Planning[2,9]:=batch_X260*batch_size;
Planning[2,2]:=batch_X760*batch_size;
Planning[2,3]:=batch_L560*batch_size;
Planning[2,4]:=batch_TLGA*batch_size;
Planning[2,5]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,1]:=batch_A5C*batch_size;
Planning_drive[2,2]:=batch_A5SB*batch_size;
Planning_drive[2,3]:=batch_por_exe*batch_size;
Planning_drive[2,4]:=batch_por_limo*batch_size;
elseif batch_start=3;
Planning[2,9]:=batch_X260*batch_size;
Planning[2,10]:=batch_X760*batch_size;
Planning[2,3]:=batch_L560*batch_size;
Planning[2,4]:=batch_TLGA*batch_size;
Planning[2,5]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,1]:=batch_A5C*batch_size;
Planning_drive[2,2]:=batch_A5SB*batch_size;
Planning_drive[2,3]:=batch_por_exe*batch_size;
Planning_drive[2,4]:=batch_por_limo*batch_size;
elseif batch_start=4;
Planning[2,9]:=batch_X260*batch_size;
Planning[2,10]:=batch_X760*batch_size;
Planning[2,11]:=batch_L560*batch_size;
Planning[2,4]:=batch_TLGA*batch_size;
Planning[2,5]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,1]:=batch_A5C*batch_size;
Planning_drive[2,2]:=batch_A5SB*batch_size;
Planning_drive[2,3]:=batch_por_exe*batch_size;
Planning_drive[2,4]:=batch_por_limo*batch_size;
elseif batch_start=5;
Planning[2,9]:=batch_X260*batch_size;
Planning[2,10]:=batch_X760*batch_size;
Planning[2,11]:=batch_L560*batch_size;
Planning[2,12]:=batch_TLGA*batch_size;
Planning[2,5]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,1]:=batch_A5C*batch_size;
Planning_drive[2,2]:=batch_A5SB*batch_size;
Planning_drive[2,3]:=batch_por_exe*batch_size;
Planning_drive[2,4]:=batch_por_limo*batch_size;
elseif batch_start=6;
Planning[2,9]:=batch_X260*batch_size;
Planning[2,10]:=batch_X760*batch_size;
Planning[2,11]:=batch_L560*batch_size;
Planning[2,12]:=batch_TLGA*batch_size;
Planning[2,13]:=batch_A5C*batch_size;
Planning[2,6]:=batch_A5SB*batch_size;
Planning[2,7]:=batch_por_exe*batch_size;
Planning[2,8]:=batch_por_limo*batch_size;
Planning_drive[2,5]:=batch_A5C*batch_size;
```

```
188 Planning_drive[2,2]:=batch_A5SB*batch_size;
189 Planning_drive[2,3]:=batch_por_exe*batch_size;
190 Planning_drive[2,4]:=batch_por_limo*batch_size;
191 elseif batch_start=7;
192 Planning[2,9]:=batch_X260*batch_size;
193 Planning[2,10]:=batch_X760*batch_size;
194 Planning[2,11]:=batch_L560*batch_size;
195 Planning[2,12]:=batch_TLGA*batch_size;
196 Planning[2,13]:=batch_A5C*batch_size;
197 Planning[2,14]:=batch_A5SB*batch_size;
198 Planning[2,7]:=batch_por_exe*batch_size;
199 Planning[2,8]:=batch_por_limo*batch_size;
200 Planning_drive[2,5]:=batch_A5C*batch_size;
201 Planning_drive[2,6]:=batch_A5SB*batch_size;
202 Planning_drive[2,3]:=batch_por_exe*batch_size;
203 Planning_drive[2,4]:=batch_por_limo*batch_size;
204 elseif batch_start=8;
205 Planning[2,9]:=batch_X260*batch_size;
206 Planning[2,10]:=batch_X760*batch_size;
207 Planning[2,11]:=batch_L560*batch_size;
208 Planning[2,12]:=batch_TLGA*batch_size;
209 Planning[2,13]:=batch_A5C*batch_size;
210 Planning[2,14]:=batch_A5SB*batch_size;
211 Planning[2,15]:=batch_por_exe*batch_size;
212 Planning[2,8]:=batch_por_limo*batch_size;
213 Planning_drive[2,5]:=batch_A5C*batch_size;
214 Planning_drive[2,6]:=batch_A5SB*batch_size;
215 Planning_drive[2,7]:=batch_por_exe*batch_size;
216 Planning_drive[2,4]:=batch_por_limo*batch_size;
217 end;
218
219 if availability = true or Avail_perc /= 100 then
220     STrun1.availability := Avail_perc2;
221     STrun1.MTTR := str_to_time(Avail_time2);
222     STrun1.failuremode :="OperatingTime";
223     STrun1.FailureActive :=true;
224     STrun2.availability := Avail_perc2;
225     STrun2.MTTR := str_to_time(Avail_time2);
226     STrun2.failuremode :="OperatingTime";
227     STrun2.FailureActive :=true;
228
229
230 end
```

**Listing A.2:** *SIMTALK-method for reset. Method is started at each reset of the simulation model.*

```
1  LB101.readytoMove := true;
2  LB102.readytoMove := true;
3  LB103.readytoMove := true;
4  LB104.readytoMove := true;
5  LB105.readytoMove := true;
6  LB106.readytoMove := true;
7  LB201.readytoMove := true;
8  LB202.readytoMove := true;
9  LB203.readytoMove := true;
10 LB204.readytoMove := true;
11 LB205.readytoMove := true;
12 LB301.readytoMove := true;
13 LB302.readytoMove := true;
14 LB303.readytoMove := true;
15 LB304.readytoMove := true;
16 LB305.readytoMove := true;
17 LB401.readytoMove := true;
18 LB402.readytoMove := true;
19 LB403.readytoMove := true;
20 LB404.readytoMove := true;
21 LB405.readytoMove := true;
22 LR21.readytoMove := true;
```

```
23 LR3.readytoMove := true;
24 LR32.readytoMove := true;
25 LB_Mar.ReadyToMove := true;
26
27 ST101.FailureActive:=false;
28 ST102.FailureActive:=false;
29 ST103.FailureActive:=false;
30 ST104.FailureActive:=false;
31 ST105.FailureActive:=false;
32 ST105_fix.FailureActive:=false;
33 ST201.FailureActive:=false;
34 ST202.FailureActive:=false;
35 ST203.FailureActive:=false;
36 ST204.FailureActive:=false;
37 ST301.FailureActive:=false;
38 ST302.FailureActive:=false;
39 ST303.FailureActive:=false;
40 ST304.FailureActive:=false;
41 ST305.FailureActive:=false;
42 ST401.FailureActive:=false;
43 ST402.FailureActive:=false;
44 ST403.FailureActive:=false;
45 ST404.FailureActive:=false;
46 ST405.FailureActive:=false;
47 STrun1.FailureActive:=false;
48 STrun2.FailureActive:=false;
49 STtest1.FailureActive:=false;
50 STtest2.FailureActive:=false;
51 STtest3.FailureActive:=false;
52 STaudit1.FailureActive:=false;
53 STaudit2.FailureActive:=false;
54 STaudit3.FailureActive:=false;
```

**Listing A.3:** *SIMTALK-method for getting the process time. Method is started when a workstation requires a process time for the roof system.*

```
1  -> time
2  var col, row    :   string;
3  row:=?.Name;
4  if @.name = "CL3" then
5      col:=@.mu.name;
6  else
7      col:=@.name;
8  end;
9  if variance = false;
10     result:= CycleTime[col, row];
11 elseif cycleTime[col,row] >0
12     --result:= z_gamma(1,cycleTime[col,row]*cycleTime[col,row]/varia,varia/cycleTime...
           [col,row],cycleTime[col,row]*0.2,cycleTime[col,row]*2);
13     result:= z_gamma(3,cycleTime[col,row]/Varia,Varia);
14 else
15     result:=0;
16 end;
```

**Listing A.4:** *SIMTALK-method for ending the production. Method is started when the production has stopped and all roof systems need to finish production.*

```
1  repeat
2      Empty_carrier;
3      waituntil LB101.readytomove = true;
4      wait 20;
5  until (WIP_bath+WIP_drive)=0
6
7  Ending:=true;
8  end_int:=1;
```

**Listing A.5:** *SIMTALK-method. for sending an empty carrier.*

```
1
2       LB101.readytomove:=true;
3       return;
```

**Listing A.6:** *SIMTALK-Code for station exit. Wheh a roofsystem or carrier leaves a workstation this method is requested.*

```
1  var RandomEOL, randomTest: integer;
2
3       FTT_A:=1/7*(sqrt(21*FTT+4)+2);
4       FTT_T:=1/3*(sqrt(21*FTT+4)-2);
5       STT_A:=1/7*(sqrt(21*STT+4)+2);
6       STT_T:=1/3*(sqrt(21*STT+4)-2);
7       FTT:=FTT_A*FTT_T;
8
9       if ? = ST101 or ? = ST102 or ? = ST103 or ? = ST104 or
10          ? = ST201 or ? = ST202 or ? = ST203 or ? = ST204 or ? = ST205 or
11          ? = ST300 or
12          ? = ST301 or ? = ST302 or ? = ST303 or ? = ST304 or ? = ST305 or
13          ? = ST401 or ? = ST402 or ? = ST403 or ? = ST404 or
14          ? = Marriage then
15          waituntil ?.lineon.nextline.ReadyToMove prio 1;
16          @.move(?.lineon.mu);
17          ?.lineon.readytomove := true;
18
19      elseif ? = STtest1 or ? = STtest2 then
20          waituntil ?.lineon.nextline.ReadyToMove prio 1;
21          @.move(?.lineon.mu);
22          ?.lineon.readytomove := true;
23
24      elseif ? = ST405
25          waituntil (LBrun1.empty and STrun1.empty and LRrun1.empty and LB501.empty) ...
                or (LBrun2.empty and STrun2.empty and LRrun2.empty and LB501.empty) prio...
                 1;
26          waituntil ?.lineon.nextline.ReadyToMove prio 1;
27          @.move(?.lineon.mu);
28          ?.lineon.readytomove := true;
29
30      elseif ? = STtest3 then
31          if full_changeover = true and third_changeover = false ;
32              if @.mu = void
33                  if @.name = LoadSource.previous_mu;
34                      waituntil LBaudit3.empty and STaudit3.empty and LRaudit3.empty ...
                            and LB505.empty and LB506.empty prio 1;
35              else
36                  waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and ...
                        LB505.empty) or (LBaudit2.empty and STaudit2.empty and LRaudit2....
                        empty and LB505.empty) prio 1;
37                  end;
38              else
39                  if @.mu.name = LoadSource.previous_mu or @.name = LoadSource....
                        previous_mu;
40                      waituntil LBaudit3.empty and STaudit3.empty and LRaudit3.empty ...
                            and LB505.empty and LB506.empty prio 1;
41              else
42                  waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and ...
                        LB505.empty) or (LBaudit2.empty and STaudit2.empty and LRaudit2....
                        empty and LB505.empty) prio 1;
43                  end;
44              end;
45
46              @.Testdone := true;
47              RandomEOL := Z_uniform(1,1,100);
48              if @.rejectNr = 0 then
49                  if RandomEOL < FTT_T * 100 then
50                      @.Test_OK := true;
51                  else
```

```
52                      @.Test_OK := false;
53                      @.RejectNr := @.RejectNr+1;
54                      @.Repair  :=true;
55                  end;
56              elseif @.RejectNr = 1 then
57                  if RandomEOL < STT_T * 100 then
58                      @.Test_OK := true;
59                  else
60                      @.Test_OK := false;
61                      @.RejectNr := @.RejectNr + 1;
62                      @.Repair := true;
63                  end;
64              elseif @.RejectNr = 2 then
65                  if RandomEOL < STT_T * 100 then
66                      @.Test_OK := true;
67                  else
68                      @.Test_OK := false;
69                      @.RejectNr := @.RejectNr + 1;
70                      @.Repair := true;
71                  end;
72              end;
73              @.move(?.lineon.mu);
74              ?.lineon.readytomove := true;
75          else
76              waituntil (LBrun1.empty and STaudit1.empty and LRaudit1.empty and LB505....
                    empty) or (LBaudit2.empty and STaudit2.empty and LRaudit2.empty and ...
                    LB505.empty) or (LBaudit3.empty and STaudit3.empty and LRaudit3....
                    empty and LB505.empty and LB506.empty) prio 1;
77              waituntil ?.lineon.nextline.ReadyToMove and LBaudit1.empty and LBaudit2....
                    empty and LRaudit1.empty and LRaudit2.empty prio 1;
78              @.Testdone := true;
79              RandomEOL := Z_uniform(1,1,100);
80              if @.rejectNr = 0 then
81                  if RandomEOL < FTT_T * 100 then
82                      @.Test_OK := true;
83                  else
84                      @.Test_OK := false;
85                      @.RejectNr := @.RejectNr+1;
86                      @.Repair  :=true;
87                  end;
88              elseif @.RejectNr = 1 then
89                  if RandomEOL < STT_T * 100 then
90                      @.Test_OK := true;
91                  else
92                      @.Test_OK := false;
93                      @.RejectNr := @.RejectNr + 1;
94                      @.Repair := true;
95                  end;
96              elseif @.RejectNr = 2 then
97                  if RandomEOL < STT_T * 100 then
98                      @.Test_OK := true;
99                  else
100                     @.Test_OK := false;
101                     @.RejectNr := @.RejectNr + 1;
102                     @.Repair := true;
103                 end;
104             end;
105             @.move(?.lineon.mu);
106             ?.lineon.readytomove := true;
107         end;
108
109
110
111
112     elseif ? = STrun1 or ? = STrun2 then
113         waituntil LRrun1.empty and LRrun2.empty and LB501.empty and LRrep6.empty and...
                LRrep5.empty prio 1;
114         @.mu.runin_OK := true;
115         @.move(?.lineon);
116         ?.lineon.readytomove := true;
```

```
117
118
119
120
121     elseif ? = STaudit1 or ? = STaudit2 then
122         waituntil LRaudit1.empty and LRaudit2.empty and LB505.empty and LBaudit1....
                empty and LBaudit2.empty prio 1;
123         @.mu.EOLdone := true;
124         RandomEOL := Z_uniform(1,1,100);
125         if @.mu.rejectNr = 0 then
126             if RandomEOL < FTT_A * 100 then
127                 @.mu.EOL_OK := true;
128                 dim_count := dim_count +1;
129             else
130                 @.mu.EOL_OK := false;
131                 @.mu.RejectNr := @.mu.RejectNr+1;
132                 @.mu.Repair  :=true;
133             end;
134         elseif @.mu.RejectNr = 1 then
135             if RandomEOL < STT_A * 100 then
136                 @.mu.EOL_OK := true;
137                 dim_count := dim_count +1;
138             else
139                 @.mu.EOL_OK := false;
140                 @.mu.RejectNr := @.mu.RejectNr + 1;
141                 @.mu.Repair := true;
142             end;
143         elseif @.mu.RejectNr = 2 then
144             if RandomEOL < STT_A * 100 then
145                 @.mu.EOL_OK := true;
146                 dim_count := dim_count +1;
147             else
148                 @.mu.EOL_OK := false;
149                 @.mu.RejectNr := @.mu.RejectNr + 1;
150                 @.mu.Repair := true;
151             end;
152
153         end;
154         @.move(?.lineon);
155         ?.lineon.readytomove := true;
156
157     elseif ? = STaudit3 then
158         waituntil LRaudit3.empty and LRdim.empty and LB506.empty and LBaudit3.empty ...
                and STdim.empty prio 1;
159         @.mu.EOLdone := true;
160         RandomEOL := Z_uniform(1,1,100);
161         if @.mu.rejectNr = 0 then
162             if RandomEOL < FTT_A * 100 then
163                 @.mu.EOL_OK := true;
164                 dim_count := dim_count +1;
165             else
166                 @.mu.EOL_OK := false;
167                 @.mu.RejectNr := @.mu.RejectNr+1;
168                 @.mu.Repair  :=true;
169             end;
170         elseif @.mu.RejectNr = 1 then
171             if RandomEOL < STT_A * 100 then
172                 @.mu.EOL_OK := true;
173                 dim_count := dim_count +1;
174             else
175                 @.mu.EOL_OK := false;
176                 @.mu.RejectNr := @.mu.RejectNr + 1;
177                 @.mu.Repair := true;
178             end;
179         elseif @.mu.RejectNr = 2 then
180             if RandomEOL < STT_A * 100 then
181                 @.mu.EOL_OK := true;
182                 dim_count := dim_count +1;
183             else
184                 @.mu.EOL_OK := false;
```

```
185                 @.mu.RejectNr := @.mu.RejectNr + 1;
186                 @.mu.Repair := true;
187             end;
188         end;
189         @.move(?.lineon);
190         ?.lineon.readytomove := true;
191 
192         elseif ? = STdim then
193         waituntil LRdim.empty and LRaudit3.empty and LB506.empty and LB507.empty ...
                prio 1;
194         @.move(?.lineon);
195         ?.lineon.readytomove := true;
196 
197 
198         elseif ? = STrepair;
199             waituntil ?.lineon.nextline.ReadyToMove prio 1;
200             @.EOLdone:=false;
201             @.repair:=false;
202             @.move(?.lineon.mu);
203             ?.lineon.readytomove := true;
204 
205 
206 elseif ? = sT105_fix then
207     @.move(?.lineon.mu);
208     waituntil buffer.full = false;
209     ?.lineon.readytomove := true;
210 end
211 return;
```

**Listing A.7:** *SIMTALK-method for conveyor entrance. Method is started when a carrier enters a conveyor.*

```
1 ?.readyToMove := false;
```

**Listing A.8:** *SIMTALK-Method for sensor activation at the conveyor.*

```
1  var ch_time : real;
2 
3      @.Stopped :=true;
4      ?.Stopped :=true;
5 
6      ls:=?.name
7 
8      if ? /= LBscrap;
9          if @.changeover_dummy = true;
10             if ? = LB106 or ? = LB_mar then
11                 m:=2;
12 
13                 ?.readytoMove := false;
14                 GetChangeTime(?);
15                 waituntil ?.nextline.readyTomove
16                 ?.stopped := false;
17                 @.stopped := false;
18                 if ? = LB106
19                     wait 0
20                 else
21                     wait 0
22                 end;
23 
24                 ?.readytoMove := true;
25                 time_change :=1;
26                 m:=1;
27             end;
28         end;
29     end;
30 
31 
32     if ? = LB101 then
```

```
33          if Ending=true and (WIP_bath+WIP_Drive)>0 then
34              waituntil ?.nextline.readytomove = true;
35              Empty_carrier;
36          elseif first_changeover then
37              waituntil ?.nextline.readytomove = true;
38              Empty_carrier;
39              @.changeover_dummy := true;
40              first_changeover := false;
41          else
42              ?.ReadyToLoad :=true;
43          end;
44
45      elseif ? = LB106 then
46          if @.empty=false then
47              if not (@.mu.name= "X260" or @.mu.name = "X760") then
48                  Robot_Exit
49                  ?.mu.mu.move(?.Ston);
50              else
51                  if @.changeover_dummy = false;
52                  waituntil buffer.full =false
53                  ?.readytoMove:=true;
54                  end;
55              end;
56          else
57              if @.changeover_dummy = false;
58              waituntil buffer.full =false
59              ?.readytoMove:=true;
60              end;
61          end;
62
63
64      elseif ? = LB107 then
65          if @.empty=false then
66              waituntil ST200_robot.empty = true;
67              ST200_Robot_Exit
68              waituntil ST200_robot.empty = false;
69              waituntil ?.nextline.ReadyToMove and ?.nextline.empty prio 1;
70              ?.readyToMove := true;
71          else
72              waituntil LB202.ReadyToMove prio 1;
73              waituntil LB201.ReadyToLoad prio 1;
74              if @.changeover_dummy = true then
75                  LB201.mu.changeover_dummy := true;
76                  @.changeover_dummy := false;
77              end;
78              ?.readyToMove:=true;
79              LB201.readyToLoad:=false;
80              waituntil ?.nextline.ReadyToMove and ?.nextline.empty prio 1;
81              LB201.readyToMove:=true;
82          end;
83
84      elseif  ? = LB201 then
85          ?.ReadyToLoad :=true;
86
87
88      elseif ? = LR24 then
89          if @.empty= true then
90              waituntil ?.nextline.ReadyToMove prio 1;
91              ?.ReadyToMove := true;
92              if LR31.empty or  LR32.empty or LR33.empty or LR34.empty then
93                  Waituntil LR3.nextline.readyToMove prio 1;
94                  if @.changeover_dummy = false then
95                      empty_car+=1;
96                      empty_wait+=1;
97                      LR3.readytomove :=true;
98                      nexxt:=true;
99                  else
100                     @.changeover_dummy := false;
101                 end;
102             end;
```

```
103
104         else
105             waituntil Marriage_robot.empty = true;
106             nexxt:=true;
107             Mar_Robot_Exit
108             waituntil Marriage_robot.empty = false;
109             waituntil ?.nextline.ReadyToMove prio 1;
110             ?.readyToMove := true;
111
112         end;
113
114     elseif ? = LR3 then
115         waituntil LR3_sensor = true prio 1;
116         if current_batch = 0 and next_batch /= 0
117             waituntil ?.nextline.readytomove = true prio 1;
118             ?.readyToMove := true;
119             @.changeover_dummy := true;
120             LR3_sensor := true;
121             second_changeover := false;
122             current_batch := next_batch;
123             next_batch := 0;
124             p:=22;
125         else
126             LR3_sensor:=false;
127             P:=7;
128             waituntil skip = false or Empty_wait = 0 prio 1;
129             P:=8;
130             if empty_wait>0 then
131                 waituntil ?.nextline.readytomove = true prio 1;
132                 ?.readyToMove := true
133                 skip := true
134                 LR3_sensor:=true;
135                 Empty_wait -=1;
136             else
137                 P:=9;
138                 if ending = false then
139                     waituntil (WIP_bath+WIP_drive+end_int)>0;
140                 end;
141                 if WIP_drive>0 then
142                     nexxt:=true;
143                 end;
144                 if end_int>0 then
145                     P:=11;
146                     waituntil skip = false prio 1;
147                     P:=12;
148                     waituntil ?.nextline.readytomove = true prio 1;
149                     ?.readyToMove := true
150                     skip := true
151                     ?.stopped := false;
152                     @.stopped := false;
153                     empty_car+=1
154                     empty_wait+=1
155                     LR3_sensor:=true;
156                     return;
157                     P:=14
158                 end;
159                 waituntil nexxt=true;
160                 P:=10;
161                 nexxt:=false;
162                 if empty_wait>0 or end_int>0 then
163                     P:=11;
164                     empty_wait-=1
165                     waituntil skip = false prio 1;
166                     P:=12;
167                     waituntil ?.nextline.readytomove = true prio 1;
168                     ?.readyToMove := true
169                     skip := true
170                     LR3_sensor:=true;
171                 else
172                     P:=13;
```

```
                    if Drive_Type = "";
                        if  second_changeover = true then
                            ?.ReadyToMove:=true;
                            second_changeover :=false;
                            LR3_sensor:=true;
                            current_batch -=1;
                        elseif  WIP_Drive>0;
                            Drive_Type:="Drive";
                            WIP_Drive:=WIP_Drive-1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToMove := true;
                            p:=1
                            LR3_sensor:=true;
                            current_batch -=1;
                        else
                            Drive_Type:="Bath";
                            WIP_Bath:=WIP_bath-1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToLoad := true;
                            p:=2
                            LR3_sensor:=true;
                            current_batch -=1;

                        end;
                    elseif Drive_Type = "Drive";
                        if WIP_Drive > 1;
                            WIP_Drive:=WIP_Drive-1;
                            current_batch -=1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToMove := true;
                            p:=3
                            LR3_sensor:=true;
                        else
                            WIP_Drive:=WIP_Drive-1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToMove := true;
                            Drive_Type := "";
                            current_batch -=1;
                            p:=4
                            LR3_sensor:=true;

                        end;
                    elseif Drive_Type = "Bath";
                        if WIP_Bath > 1;
                            WIP_Bath := WIP_Bath -1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToLoad := true;
                            current_batch -=1;
                            p:=5
                            LR3_sensor:=true;
                        else
                            WIP_Bath := WIP_Bath -1;
                            waituntil ?.nextline.ReadyToMove prio 1;
                            ?.ReadyToLoad := true;
                            current_batch -=1;
                            Drive_Type := "";
                            p:=6
                            LR3_sensor:=true;
                        end;
                    end;
                end;
                end;
                end;



    elseif ? = LB301 then
        if @.cont=Void;
            if @.changeover_dummy = true;
                waituntil ?.nextline.readytomove = true prio 1;
```

```
243                 ?.readyToMove := true;
244             elseif empty_car≤empty_wait then
245                 ?.readyToLoad := true;
246             else
247                 waituntil ?.nextline.readytomove = true prio 1;
248                 ?.readyToMove := true;
249                 empty_car-=1
250                 skip := false;
251             end;
252
253
254         else
255             waituntil ?.nextline.ReadyToMove prio 1;
256             ?.ReadyToMove :=true;
257         end;
258
259
260
261         elseif ? = LB_mar then
262             if @.empty = true;
263                 if @.changeover_dummy = false;
264                     waituntil ?.nextline.ReadyToMove prio 1;
265                     ?.ReadyToMove := true;
266                 end;
267         else
268             if @.mu.Drive_concept = true;
269                 @.mu.move(?.STon)
270                 ?.ReadyToLoad := true;
271             else
272                 waituntil ?.nextline.ReadyToMove prio 1;
273                 ?.ReadyToMove :=true;
274             end;
275         end;
276
277     elseif ? = LB405 then
278         if @.empty =false then
279             @.mu.move(?.STon);
280         else
281             if @.changeover_dummy = true;
282                 waituntil LBrun1.empty and STrun1.empty and LRrun1.empty and LBrun2....
                        empty and STrun2.empty and LRrun2.empty prio 1;
283             end;
284             waituntil ?.nextline.ReadyToMove prio 1;
285             ?.ReadyToMove := true;
286         end;
287
288
289
290     elseif ? = LB501 then
291         if @.empty = false then
292             lrtrig:=1
293             if @.mu.Runin_OK = false then
294                 lrtrig:=2
295                 if LBrun1.empty and STrun1.empty and LRrun1.empty then
296                     lrtrig:=3
297                     @.move(LBrun1);
298                     ?.ReadyToMove := true;
299                 else
300                     lrtrig:=4
301                     waituntil LBrun2.empty and STrun2.empty and LRrun2.empty  prio ...
                            1;
302                     @.move(LBrun2);
303                     ?.ReadyToMove := true;
304                     lrtrig:=5
305                 end;
306             else
307                 lrtrig:=6
308                 waituntil ?.nextline.ReadyToMove prio 1;
309                 ?.ReadyToMove := true;
310                 @.move(?.nextLine);
```

```
311                 lrtrig:=7
312             end;
313         else
314             lrtrig:=8
315             waituntil ?.nextline.ReadyToMove and LRrepair.empty prio 1;
316             ?.ReadyToMove := true;
317             @.move(?.nextLine);
318             lrtrig:=9
319         end;
320
321
322     elseif ? = LBrun1 or ? = LBrun2 or ? = LBaudit1 or ? = LBaudit2 or ? = LBaudit3 ...
            or ? = LBdim then
323         @.move(?.STon)
324         ?.readyToMove := true;
325
326
327     elseif ? = LRrun1 or ? = LRrun2 or ? = LRaudit3 or ? = LRdim then
328         waituntil ?.nextline.readytomove and ?.nextline.empty prio 1;
329         @.move(?.nextline)
330         ?.readyToMove := true;
331
332     elseif ? = LRaudit1 or ? = LRaudit2  then
333         waituntil ?.nextline.readytomove and ?.nextline.empty prio 1;
334             if Dim_count>9
335                 waituntil LBdim.empty and STdim.empty and LRdim.empty prio 1;
336             end;
337         @.move(?.nextline)
338         ?.readyToMove := true;
339
340     elseif ? = LB504 then
341         if @.empty then
342             if @.changeover_dummy = true;
343                 waituntil ?.nextline.ReadyToMove and LBaudit1.empty and LBaudit2....
                        empty and LRaudit1.empty and LRaudit2.empty and STaudit1.empty ...
                        and STaudit2.empty prio 1;
344                 Third_changeover := false;
345             end;
346             waituntil ?.nextline.ReadyToMove and LBaudit1.empty and LBaudit2.empty ...
                    and LRaudit1.empty and LRaudit2.empty prio 1;
347             ?.ReadyToMove :=true;
348             @.move(?.nextLine);
349         else
350             @.mu.move(?.StOn);
351         end;
352
353
354
355
356     elseif ? = LB505 then
357         if @.empty = false then
358             if @.mu.mu = void
359                 if (@.mu.name = LoadSource.previous_mu) and full_changeover = true ...
                        and third_changeover = false;
360                     waituntil ?.nextline.ReadyToMove and LBaudit3.empty and LBdim....
                            empty and LRaudit3.empty and LRdim.empty prio 1;
361                     ?.ReadyToMove := true;
362                     @.move(?.nextLine);
363                 elseif @.mu.EOLdone = false and @.mu.testdone = true and @.mu....
                        test_OK = true then
364                     if LBaudit1.empty and STaudit1.empty and LRaudit1.empty then
365                         @.move(LBaudit1);
366                         ?.ReadyToMove := true;
367                     elseif LBaudit2.empty and STaudit2.empty and LRaudit2.empty then
368                         waituntil LBaudit2.empty and STaudit2.empty and LRaudit2....
                                empty  prio 1;
369                         @.move(LBaudit2);
370                         ?.ReadyToMove := true;
371                     else
372                         waituntil ?.nextline.ReadyToMove and LBaudit3.empty and ...
```

```
                            LBdim.empty and LRaudit3.empty and LRdim.empty prio 1;
373                     ?.ReadyToMove := true;
374                     @.move(?.nextLine);
375                 end;
376             else
377                 waituntil ?.nextline.ReadyToMove  and LBaudit3.empty and LBdim....
                        empty and LRaudit3.empty and LRdim.empty prio 1;
378                 ?.ReadyToMove := true;
379                 @.move(?.nextLine);
380             end;
381         else
382             if (@.mu.mu.name = LoadSource.previous_mu) and full_changeover = ...
                    true and third_changeover = false;
383                 waituntil ?.nextline.ReadyToMove and LBaudit3.empty and LBdim....
                        empty and LRaudit3.empty and LRdim.empty prio 1;
384                 ?.ReadyToMove := true;
385                 @.move(?.nextLine);
386             elseif @.mu.EOLdone = false and @.mu.testdone = true and @.mu....
                    test_OK = true then
387                 if LBaudit1.empty and STaudit1.empty and LRaudit1.empty then
388                     @.move(LBaudit1);
389                     ?.ReadyToMove := true;
390                 elseif LBaudit2.empty and STaudit2.empty and LRaudit2.empty then
391                     waituntil LBaudit2.empty and STaudit2.empty and LRaudit2....
                            empty  prio 1;
392                     @.move(LBaudit2);
393                     ?.ReadyToMove := true;
394                 else
395                     waituntil ?.nextline.ReadyToMove and LBaudit3.empty and ...
                            LBdim.empty and LRaudit3.empty and LRdim.empty prio 1;
396                     ?.ReadyToMove := true;
397                     @.move(?.nextLine);
398                 end;
399             else
400                 waituntil ?.nextline.ReadyToMove  and LBaudit3.empty and LBdim....
                        empty and LRaudit3.empty and LRdim.empty prio 1;
401                 ?.ReadyToMove := true;
402                 @.move(?.nextLine);
403             end;
404         end;
405     else
406         waituntil ?.nextline.ReadyToMove  and LBaudit3.empty and LBdim.empty and...
                LRaudit3.empty and LRdim.empty prio 1;
407         ?.ReadyToMove := true;
408         @.move(?.nextLine);
409     end;
410
411 elseif ? = LB506 then
412     if @.empty = false then
413         if @.mu.EOLdone = false and @.mu.testdone = true and @.mu.test_OK = true...
                then
414             if LBaudit3.empty and STaudit3.empty and LRaudit3.empty then
415                 @.move(LBaudit3);
416                 ?.ReadyToMove := true;
417             end;
418         elseif Dim_count > 9 and @.mu.EOLdone and @.mu.EOL_OK then
419             waituntil LBdim.empty and STdim.empty and LRdim.empty prio 1;
420             @.move(LBdim)
421             ?.ReadyToMove:=true;
422             Dim_count:=0;
423
424         else
425             waituntil ?.nextline.ReadyToMove and LBreturn.empty and LBrepair....
                    empty prio 1;
426             ?.ReadyToMove := true;
427             @.move(?.nextLine);
428         end;
429     else
430         waituntil ?.nextline.ReadyToMove and LBreturn.empty and LBrepair.empty ...
                prio 1;
```

```
431             ?.ReadyToMove := true;
432             @.move(?.nextLine);
433         end;
434
435     elseif ? = LB507 then
436         if @.empty = false then
437             if  @.mu.EOL_OK = false or @.mu.Test_OK = false then
438                 waituntil LBrepair.readyToMove and LBrepair.empty prio 1;
439                 ?.ReadyToMove := true;
440                 @.move(LBrepair)
441             else
442                 waituntil LBreturn.readyToMove prio 1;
443                 ?.ReadyToMove := true;
444                 if @.mu.name = LoadSource.previous_mu or @.mu.name = LoadSource....
                        current_mu;
445                     if @.mu.name = LoadSource.previous_mu
446                         current_batch_end -=1;
447                     elseif LoadSource.previous_mu= ""
448                         current_Batch_end -=1;
449                     else
450                         next_batch_end -=1;
451                     end;
452                 else
453                     if @.mu.mu.name = LoadSource.previous_mu
454                         current_batch_end -=1;
455                     elseif LoadSource.previous_mu = ""
456                         current_Batch_end -=1;
457                     else
458                         next_batch_end -=1;
459                     end;
460                 end;
461
462                 @.mu.move(STunload)
463                 @.move(LBreturn)
464             end;
465         else
466             @.changeover_dummy := false;
467             @.move(LBreturn);
468             waituntil LR501.readyToMove prio 1;
469             ?.ReadyToMove := true;
470
471         end;
472         if current_batch_end = 0;
473             full_changeover := false;
474             current_batch_end := next_batch_end;
475             next_batch_end := 0;
476             LoadSource.previous_mu:=LoadSource.current_Mu;
477         end;
478
479
480     elseif ? = LRrep5 then
481         if @.mu.rejectnr > 2;
482             @.mu.move(LBscrap);
483             ?.ReadyToMove:= true;
484         else
485             waituntil ?.nextline.readytomove and ?.nextline.empty prio 1;
486             @.move(?.nextline)
487             ?.readyToMove := true;
488         end;
489
490     elseif ? = LRrep6 then
491         waituntil LRrepair.ReadyToMove and LRrepair.empty and LB502.empty and LRrun2...
                .empty and LRrun1.empty prio 1;
492         @.move(LRrepair);
493         ?.ReadyToMove:= true;
494
495     elseif ? = LBreturn or ? = LBrepair or ? = LRrepair then
496         waituntil ?.nextline.ReadyToMove and ?.nextline.empty prio 1;
497         ?.readyToMove := true;
498         ?.mu.move(?.nextline);
```

```
499
500     elseif ? = LBscrap then
501         if @.name = LoadSource.previous_mu or @.name = LoadSource.current_mu;
502             if @.name = LoadSource.previous_mu
503                 current_batch_end -=1;
504             elseif LoadSource.previous_mu= ""
505                 current_Batch_end -=1;
506             else
507                 next_batch_end -=1;
508             end;
509         else
510             if @.mu.name = LoadSource.previous_mu
511                 current_batch_end -=1;
512             elseif LoadSource.previous_mu = ""
513                 current_Batch_end -=1;
514             else
515                 next_batch_end -=1;
516             end;
517         end;
518         if current_batch_end = 0;
519             full_changeover := false;
520             current_batch_end := next_batch_end;
521             next_batch_end := 0;
522             LoadSource.previous_mu:=LoadSource.current_Mu;
523         end;
524         @.move(STscrap)
525         ?.readyToMove := true;
526
527     elseif ? = LR501;
528         if LR502.empty or LR503.empty or LR504.empty or LR505.empty or LR506.empty ...
                or LR506.empty or LR41.empty or LR42.empty or LR43.empty or LR44.empty ...
                or LR45.empty
529             waituntil ?.nextline.readyToMove = true;
530             ?.ReadyToMove := true;
531         elseif empty_car =0 and (LR31.empty or LR32.empty or LR33.empty or LR34....
                empty);
532             waituntil LR31.nextline.readytoMove = true;
533             LR31.readytomove:=true;
534             empty_car +=1;
535             empty_wait+=1;
536             nexxt := true;
537             waituntil ?.nextline.readyToMove = true;
538             ?.ReadyToMove := true;
539         else
540             waituntil ?.nextline.readyToMove = true;
541             ?.ReadyToMove := true;
542         end;
543
544
545
546     elseif not (?.ston = void) then
547         lrtrig:=1
548         if ?.mu.empty = true;
549             waituntil ?.nextline.ReadyToMove prio 1;
550             lrtrig:=2;
551             ?.ReadyToMove := true;
552         else
553             @.mu.move(?.StOn);
554             lrtrig:=3
555         end;
556
557     elseif  (?.ston = void) then
558         lrtrig:=4
559         waituntil ?.nextline.readyToMove = true;
560         ?.ReadyToMove := true;
561         lrtrig:=5
562     end;
563
564
565
```

```
566
567
568     waituntil ?.readyToMove prio 1;
569     if ? = LBscrap;
570         ?.stopped := false;
571     else
572         if @.changeover_dummy = false or not (? = LB106 or ? = LB_mar);
573             ?.stopped := false;
574             if ?.empty = false;
575                 @.stopped := false;
576             end;
577         end;
578     end;
```

**Listing A.9:** *SIMTALK-Code for source entrances.*

```
1
2
3  if ? = LoadSource then
4      ?.Stop_prod:=true;
5      if @.name /= ?.current_mu and ?.current_mu /= "";
6          waituntil next_batch_end = 0 prio 1;
7          first_changeover:=true;
8          second_changeover:=true;
9          Third_changeover := true;
10         Full_changeover :=true;
11         ?.previous_mu := ?.current_mu;
12     end;
13     ?.stop_prod:=false;
14
15
16 elseif ? = DriveSource then
17
18 end;
19
20 if ? = Carrier3 then
21
22 end;
```

**Listing A.10:** *SIMTALK-Code for source exit.*

```
1
2
3  if ? = LoadSource then
4          waituntil ?.stop_prod = false prio 1;
5          if first_changeover and LB101.ReadyToLoad;
6              waituntil LB101.nextline.readytomove = true prio 1;
7              Empty_carrier;
8              LB101.mu.changeover_dummy := true;
9              first_changeover := false;
10             LB101.readyToload:=false;
11         end;
12             waituntil LB101.ReadyToLoad prio 1;
13             if @.name = "A5_coupe" or  @.name = "A5_SB" or  @.name = "pana_limo" or ...
                   @.name = "pana_exe" then
14                 @.Drive_concept:=true;
15                 WIP_Drive:=WIP_Drive+1;
16                 ?.current_mu:=@.name;
17             else
18                 @.Drive_concept:=false;
19                 WIP_bath:=WIP_bath+1;
20                 ?.current_mu:=@.name;
21             end;
22             if second_changeover = true;
23                 next_batch +=1;
24             else
25                 current_batch +=1;
```

```
26              end;
27              if full_changeover = true;
28                  next_batch_end +=1;
29              else
30                  current_batch_end +=1;
31              end;
32              LB101.ReadyToLoad :=false;
33              @.Move(LB101.StOn);
34              ?.current_mu:=@.name;
35
36  elseif ? = DriveSource then
37          waituntil LB301.ReadyToLoad prio 1;
38          @.Drive_concept:=true;
39          LB301.ReadyToLoad :=false;
40          @.Move(LB301.StOn);
41
42      end;
43
44
45
46      if ? = Carrier3 then
47          waituntil LR3.empty and LR45.empty prio 1;
48          @.Move(LR3)
49      end;
```

**Listing A.11:** *SIMTALK-Code for fixed glass robot entrance.*

```
1  ?.target := ST105_fix;
```

**Listing A.12:** *SIMTALK-Code for fixed glass robot exit.*

```
1  var triggerme,  trigger : string
2  if ST105_robot.resworking = false and ST105_robot.empty = true then
3      if GlassSource.resblocked = true then
4          ST105_Robot.target:= GlassSource;
5          GlassSource.mu.move(ST105_robot);
6      end;
7  end;
8
9  return;
```

**Listing A.13:** *SIMTALK-Code for the destination of the fixed glass robot.*

```
1  if ST105_robot.cont = void then
2      return;
3  end
4  ?.SetDestination(ST105_Fix);
```

**Listing A.14:** *SIMTALK-Code for turning robot entrance.*

```
1  ?.target := ST201;
```

**Listing A.15:** *SIMTALK-Code for turning robot exit.*

```
1  var triggerme,  trigger : string
2  waituntil LB201.ReadyToLoad prio 1;
3  LB201.ReadyToLoad := false;
4  if ST200_robot.resworking = false and ST200_robot.empty = true and ST201.empty = ...
       true then
5      if LB107.mu.resblocked = true then
6          ST200_Robot.target:= ST201;
```

```
7           LB107.mu.mu.move(ST200_robot);
8       end;
9   end;
10
11  return;
```

**Listing A.16:** *SIMTALK-Code for the destination of the turning robot.*

```
1   if ST200_robot.cont = void then
2       return;
3   end
4   ?.SetDestination(ST201);
```

**Listing A.17:** *SIMTALK-Code for marriage robot entrance.*

```
1   if @.Drive_Concept = true then
2       ?.SetDestination(Marriage);
3   else
4       ?.SetDestination(ST300);
5   end;
```

**Listing A.18:** *SIMTALK-Code for marriage robot exit.*

```
1
2   var triggerme,  trigger : string
3   if LR24.mu.mu.Drive_Concept = true then
4       waituntil LB_mar.ReadyToLoad prio 1;
5       LB_Mar.ReadyToLoad := false;
6   elseif LR24.mu.mu.Drive_Concept = false then
7       waituntil LR3.ReadyToLoad prio 1;
8       LR3.ReadyToLoad := false;
9   end;
10
11  if Marriage_robot.resworking = false and Marriage_robot.empty = true then
12      if LR24.Mu.Mu.Drive_concept =true then
13          Marriage_Robot.target:= Marriage;
14          LR24.mu.mu.move(Marriage_robot);
15      else
16          Marriage_Robot.target:= ST300;
17          LR24.mu.mu.move(Marriage_robot);
18
19      end;
20  end;
21
22  return;
```

**Listing A.19:** *SIMTALK-Code for the destination of the marriage robot.*

```
1   if Marriage_robot.cont = void then
2       return;
3   end
4   if @.Drive_Concept = true then
5       ?.SetDestination(Marriage);
6   else
7       ?.SetDestination(ST300);
8   end;
```

# Appendix B

Table with process times of the different roof system types. These process times are used in the simulation model as explained in Chapter 4 and for the experiments, for which the results are shown in Chapter 5

**Table B.1:** *Process times of roof systems at the corresponding workstations as used in the simulation model for the experiments. Times are given in timeperiods.*

| | TLGA | X260 | X760 | L560 | A5_coupe | A5_SB | Pana_limo | pana_exe |
|---|---|---|---|---|---|---|---|---|
| ST101 | 0,07722 | 0,08056 | 0,07389 | 0,08167 | 0,09333 | 0,09667 | 0,09 | 0,09651 |
| ST102 | 0,07722 | 0,08056 | 0,07389 | 0,08111 | 0,09333 | 0,09667 | 0,09 | 0,09596 |
| ST103 | 0,07722 | 0,08056 | 0,07389 | 0,08056 | 0,09333 | 0,09667 | 0,09 | 0,0954 |
| ST104 | 0,07722 | 0,08056 | 0,07389 | 0,08 | 0,09333 | 0,09667 | 0,09 | 0,09484 |
| ST105_fix | 0,07722 | 0,03222 | 0,03222 | 0,03222 | 0,05 | 0,05 | 0,05 | 0,05 |
| ST201 | 0,07722 | 0,04111 | 0,04111 | 0,04111 | 0,06 | 0,06 | 0,06 | 0,06 |
| ST202 | 0,07722 | 0,07722 | 0,07722 | 0,07944 | 0,09333 | 0,09333 | 0,09333 | 0,09429 |
| ST203 | 0,07722 | 0,07722 | 0,07722 | 0,07889 | 0,09333 | 0,09333 | 0,09333 | 0,09373 |
| ST204 | 0,07722 | 0,07722 | 0,07722 | 0,07833 | 0,09333 | 0,09333 | 0,09333 | 0,09318 |
| ST205 | 0,07722 | 0,07722 | 0,07722 | 0,07778 | 0,09333 | 0,09333 | 0,09333 | 0,09262 |
| ST301 | 0,07722 | 0,07722 | 0,07722 | 0,07722 | 0,09333 | 0,09333 | 0,09333 | 0,09484 |
| ST302 | 0,07722 | 0,07722 | 0,07722 | 0,07667 | 0,09333 | 0,09333 | 0,09333 | 0,09429 |
| ST303 | 0,07722 | 0,07722 | 0,07722 | 0,07611 | 0,09333 | 0,09333 | 0,09333 | 0,09373 |
| ST304 | 0,07722 | 0,07722 | 0,07722 | 0,07556 | 0,09333 | 0,09333 | 0,09333 | 0,09318 |
| ST305 | 0,07722 | 0,07722 | 0,07722 | 0,075 | 0,09333 | 0,09333 | 0,09333 | 0,09262 |
| Marriage | 0 | 0 | 0 | 0 | 0,06444 | 0,06444 | 0,06444 | 0,06444 |
| ST401 | 0,07722 | 0,07722 | 0,07722 | 0,07444 | 0,09333 | 0,09333 | 0,09333 | 0,09207 |
| ST402 | 0,07722 | 0,07389 | 0,08056 | 0,07389 | 0,09333 | 0,09 | 0,09667 | 0,09151 |
| ST403 | 0,07722 | 0,07389 | 0,08056 | 0,07333 | 0,09333 | 0,09 | 0,09667 | 0,09096 |
| ST404 | 0,07722 | 0,07389 | 0,08056 | 0,07278 | 0,09333 | 0,09 | 0,09667 | 0,0904 |
| ST405 | 0,07722 | 0,07389 | 0,08056 | 0,07222 | 0,09333 | 0,09 | 0,09667 | 0,08984 |
| ST300 | 0,04111 | 0,04111 | 0,04111 | 0,04111 | 0,06 | 0,06 | 0,06 | 0,06 |
| STrun1 | 0,11667 | 0,11667 | 0,11667 | 0,11667 | 0,13333 | 0,13333 | 0,13333 | 0,13333 |
| STrun2 | 0,11667 | 0,11667 | 0,11667 | 0,11667 | 0,13333 | 0,13333 | 0,13333 | 0,13333 |
| STtest1 | 0,04667 | 0,04667 | 0,04667 | 0,04667 | 0,05667 | 0,05667 | 0,05667 | 0,05667 |
| STtest2 | 0,04667 | 0,04667 | 0,04667 | 0,04667 | 0,05667 | 0,05667 | 0,05667 | 0,05667 |
| STtest3 | 0,04667 | 0,04667 | 0,04667 | 0,04667 | 0,05667 | 0,05667 | 0,05667 | 0,05667 |
| STaudit1 | 0,17222 | 0,17222 | 0,17222 | 0,17222 | 0,20556 | 0,20556 | 0,20556 | 0,20556 |
| STaudit2 | 0,17222 | 0,17222 | 0,17222 | 0,17222 | 0,20556 | 0,20556 | 0,20556 | 0,20556 |
| STaudit3 | 0,17222 | 0,17222 | 0,17222 | 0,17222 | 0,20556 | 0,20556 | 0,20556 | 0,20556 |
| STdim | 0,06667 | 0,06667 | 0,06667 | 0,06667 | 0,06778 | 0,06778 | 0,06778 | 0,06778 |

# Appendix C

In this appendix the derivation of (2.3) is shown. In a parallel system with 2 identical workstation with availability $A_1$ and $A_2$, there are 4 possible states of working: Either, both workstations are available, both are unavailable, or either one of the workstations is available. If both workstations are available, the availability of the subsystem is 1. If neither of them is available the availability of the subsystem is 0. If only one of the subsystem is available the availability is determined by the utilization of the workstations. If the combined utilization is less than 1 the production can be done by one workstation and the availability of the subsystem is 1. If the combined utilization is bigger than 1 the throughput is decreased as not all production can be done by the one available workstation. Combining this with the probability of these states happening gives the following equations.

$$\begin{aligned} A_{r2} &= A_1A_2 + \Big(A_1(1-A_2) + (1-A_1)A_2\Big)\frac{1}{\rho_1+\rho_2} \quad \text{for} \quad \rho_1+\rho_2 > 1 \\ A_{r2} &= A_1A_2 + \Big(A_1(1-A_2) + (1-A_1)A_2\Big) \quad \text{for} \quad \rho_1+\rho_2 < 0 \end{aligned} \tag{C.1}$$

Combining these two results in (C.2).

$$A_{r2} = A_1A_2 + \Big(A_1(1-A_2) + (1-A_1)A_2\Big)\min\left(1, \frac{1}{\rho_1+\rho_2}\right) \tag{C.2}$$

The same can be done for a subsystem with 3 parallel workstation, only now there are 3 different possibilities, as the combined utilization can be smaller than 1, between 1 and 2 and bigger than 2. This result in three different equations.

$$\begin{aligned} A_{r3} =& A_1A_2A_3 + \Big(A_1A_2(1-A_3) + A_1(1-A_2)A_3 + (1-A_1)A_2A_3\Big)\frac{2}{\rho_1+\rho_2+\rho_3} \quad \text{for} \quad \rho_1+\rho_2+\rho_3 > 2 \\ &+ \Big(A_1(1-A_2)(1-A_3) + (1-A_1)A_2(1-A_3) + (1-A_1)(1-A_2)A_3\Big)\frac{1}{\rho_1+\rho_2+\rho_3} \\ A_{r3} =& A_1A_2A_3 + \Big(A_1A_2(1-A_3) + A_1(1-A_2)A_3 + (1-A_1)A_2A_3\Big) \quad \text{for} \quad 1 < \rho_1+\rho_2+\rho_3 < 2 \\ &+ \Big(A_1(1-A_2)(1-A_3) + (1-A_1)A_2(1-A_3) + (1-A_1)(1-A_2)A_3\Big)\frac{1}{\rho_1+\rho_2+\rho_3} \\ A_{r3} =& A_1A_2A_3 + \Big(A_1A_2(1-A_3) + A_1(1-A_2)A_3 + (1-A_1)A_2A_3\Big) \quad \text{for} \quad \rho_1+\rho_2+\rho_3 < 1 \\ &+ \Big(A_1(1-A_2)(1-A_3) + (1-A_1)A_2(1-A_3) + (1-A_1)(1-A_2)A_3\Big) \end{aligned} \tag{C.3}$$

Combining these 3 equation results in (C.4).

$$\begin{aligned} A_{r3} =& A_1A_2A_3 + \Big(A_1A_2(1-A_3) + A_1(1-A_2)A_3 + (1-A_1)A_2A_3\Big)\min\left(1, \frac{2}{\rho_1+\rho_2+\rho_3}\right) \\ &+ \Big(A_1(1-A_2)(1-A_3) + (1-A_1)A_2(1-A_3) + (1-A_1)(1-A_2)A_3\Big)\min\left(1, \frac{1}{\rho_1+\rho_2+\rho_3}\right) \end{aligned} \tag{C.4}$$