SIMULATION PROJECT | 2021

Laptop Manufacturing Simulation

Gvero Tom, Zielinski Maciej

Contents

1 Introduction

In this report we will be attempting to model a simplified laptop manufacturing process using Simul8 in order to analyse the production line and suggest possible improvements which can be made with regards to its efficiency. We have been provided with a summary of how each stage of the process works. Each stage has a duration which is randomly distributed according to some probability distribution, however not all of these are given. Our first job is to carry out input analysis on some historical time data of these processes and work out how they are distributed. This will allow us to model our system as accurately as possible. We then move on to detail how we go about implementing our model in Simul8 based on the constraints outlined in the project brief. Each process in the chain uses a certain number of resources and workers, all of which need to be kept track of, whilst some processes need to wait for others to finish before they begin. All of these things add complexity to the model and will be considered below. Finally we will analyse the results produced by our model and try to use them to give helpful recommendations as to how the overall process can be improved.

2 Input Analysis

Using the data provided, we will be attempting to estimate the distributions of task durations using a variety of statistical methods, the details of which can be seen below for each missing distribution. We have data for the last 200 realisations of each of the values we are trying to estimate the distribution of.

2.1 Inter-arrival Time

We are given that the inter-arrival times for laptop orders are exponentially distributed, however, we are unsure of the rate. First, we use maximum likelihood estimation to obtain an estimate for the rate in terms of our observed data. For an exponentially distributed random variable we know this estimate to be:

$$\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} x_i}$$

for n data points x_i. Substituting in our values for n and x_i we arrive at $\hat{\lambda} = 0.157776$. In order to determine the goodness of fit of this exponential distribution to our data we will carry out a Kolmogorov-Smirnov test. We obtain a test statistic of 0.560252 which is smaller than our 95% critical value of 1.094 meaning that we cannot reject the null hypothesis and can conclude that this distribution is reasonable.

2.2 Initial Phase

For this initial phase we are given no information about the distribution so need to estimate it from scratch. Looking at the histogram of the time data in Figure 1 overleaf, it would be reasonable to assume that this data is uniformly distributed, so we will proceed under this assumption. We will take the lower and upper bounds of our uniform distribution to be the minimum and maximum of the dataset respectively. This leads us to assume that the data is distributed Uniform$(3.29296, 5.97324)$. In order to test this hypothesis we will again use the Kolmogorov-Smirnov test. In this instance, we achieve a test statistic of 0.601197 which is smaller than our critical value of 1.358, thus there is no evidence to reject the null hypothesis and thus this distribution is acceptable.

2.3 Placing the Keyboard and Mouse

Again, here we have no distribution given so we will need to estimate. Looking at the histogram for the data in Figure 2, it would be reasonable to assume a normal distribution. We will take

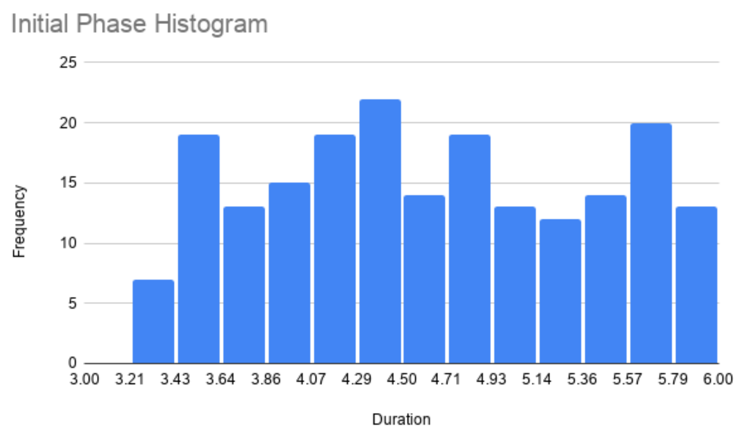

Figure 1. Histogram for initial phase data

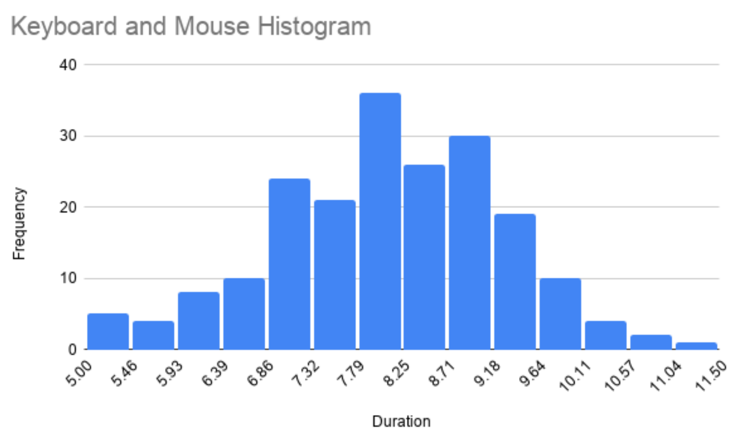

Figure 2. Histogram for keyboard and mouse data

the mean and variance of the data and use these as our parameter estimates for the normal distribution. As such, we estimate that the time taken to place the keyboard and mouse is distributed approximately $\mathcal{N}(8.105474, 1.430553)$. In order to test the goodness of fit of this distribution to our data we will use the Chi-squared test. This gives us a Chi-squared test statistic of 6.1, which is smaller than our 95% critical value of 14.067 so we cannot reject our null hypothesis, thus this distribution is appropriate.

2.4 Assembling the Case

We will again need to estimate this distribution fully from the data. Looking at the histogram in Figure 3 we see that, much like the last process, this data appears to be normally distributed.

We will proceed in the same way as before, using the sample mean and variance as our estimates for the distribution parameters. This gives us an estimated distribution of $\mathcal{N}(3.264081, 1.46102)$. We will also analyse the goodness of fit using the Chi-squared test. We get a Chi-squared test statistic of 6 which is smaller than our 95% critical value of 14.067 thus we cannot reject the null hypothesis and this is a reasonable distribution for the duration of this process.

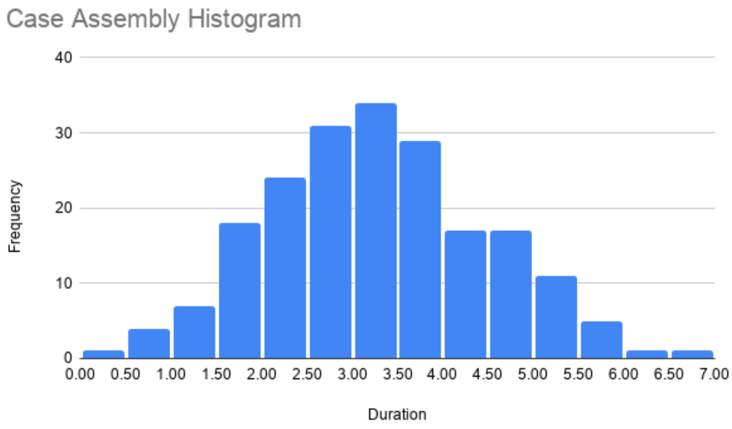

Figure 3. Histogram for case assembly data

3 Model Building

Now we have carried out the required input analysis, we have a good idea of how the durations of each component of the process behave. We can now begin to put the pieces together and build a model of the entire system in Simul8. In this report we will just cover the non-trivial parts of the model that require some explanation, but the whole thing will be available in an accompanying Simul8 file.

3.1 Arrival Process

We have modelled our arrival process as a collection of activities as seen below. Firstly, in our start point we define our inter-arrival distribution as detailed in our input analysis and assign an order size label to our work item, detailing the size of the order which is distributed as described in the project brief. We then move to a routing decision activity, which instantly terminates orders if the global number of laptops in the system plus the order size exceeds 30. If an order is accepted, we move to a dummy activity which assigns each accepted order a unique order number, so we can keep track of all laptops that correspond to a given order. We also time-stamp each order to give us an indication of the start time of its production - this will be useful in calculating the cycle time later on. Finally, we move to the "order_accepted" activity, which updates the global variable of number of laptops in the system, by adding to it the number of laptops in this particular order. It also batches into the number of work items corresponding to 2 times the order size, because there are two process routes which run simultaneously for each laptop.

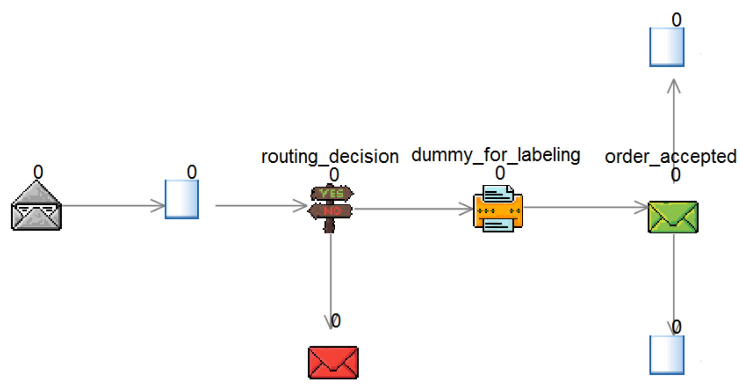

Figure 4. Simul8 screenshot: Arrival Process

3.2 Power Assembly Queue

Since there is space for only three items in front of power assembly, we set the capacity for the queue to 3. This way the "initial_phase" activity cannot process any more work items until there is space in the queue.

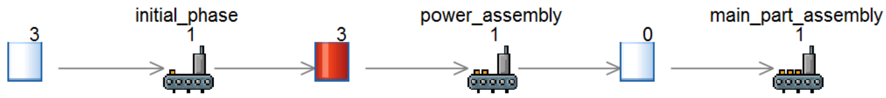

Figure 5. Simul8 screenshot: Power Assembly Queue

3.3 Cutting Aluminium Plates

Given the nature of this activity, we were unable to replicate it [1] and so had to create two identical activities to account for the fact that we have two cut and roll machines that can be used simultaneously. The activities can only go ahead if we have aluminium available. To account for this we have visual logic "Before Selecting" in routing in which blocks the current routing if the global variable of aluminium inventory is less than 1. If there is aluminium available, then our second piece of visual logic which occurs "After Loading Work" takes 1 away from our global aluminium inventory variable, to account for 1 piece of aluminium being used up for this laptop. We also batch into 2 work items on routing out to account for the 2 metal plates which have been created.

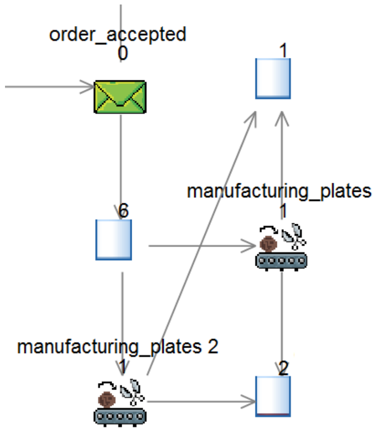

Figure 6. Simul8 screenshot: Aluminium Plates Cutting

3.4 Cutting (for the keyboard and mouse)

We can see below our set-up for the cutting of the keyboard and mouse. In the routing out options of the main activity we set our work item to move to the next stage with 80% probability and to our quality check activity with 20% probability. In quality check, we introduce a new label in the routing in "After Loading Work" section, which takes a value of 1 to indicate that this particular work item has been quality checked, before sending it back to the cutting queue. We have defined some visual logic in the "Actions" area of the main activity to send the current work item straight to the next queue if it has a label indicating it has been quality checked. This ensures that no item can be quality checked more than once.

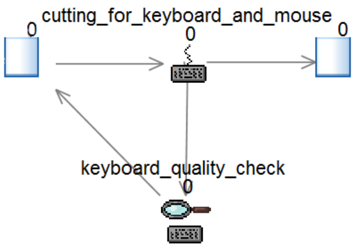

Figure 7. Simul8 screenshot: Cutting

3.5 Main Assembly

To start the main assembly process we first need to collect together all the work items from the preceeding processes. We do this using the activity "waiting_for_all_parts" which has the "collect" discipline selected on routing in, collecting one work item from each of the three routes into it, matching based on order number and assembling to one work item. This work item is then sent to a dummy activity, which batches it into two work items, so that the two processes of the main assembly can begin simultaneously. Given that we have 4 workbenches on which this assembly takes place, we set replicate to 4 for each of the concerned activities. Since placing the motherboard activity collects items we were unable to replicate this activity as described in [1]. So instead we had to create 4 separate activities as seen in the figure below. The same type 1 worker has to assemble the case and then place the motherboard, so in the resource detail menu of the case assembly activity, we set the option to "require here, but do not release the resource". Then, on each of the four "placing_motherboard" activities, we set the option to "only release the resource here". This causes the same resource to be used for both activities. In placing the motherboard we also need to assemble the two work items that feed into it, matching again by order number. For the hard disk assembly, we need to consider the limited availability of the hard discs themselves. We have some visual logic on the routing in for this activity which blocks the routing before selecting if the global variable of hard disc inventory is less than 1. After loading the work item, we then have another piece of visual logic which takes 1 away from the global hard disc inventory value to indicate that a hard disc has been used up.

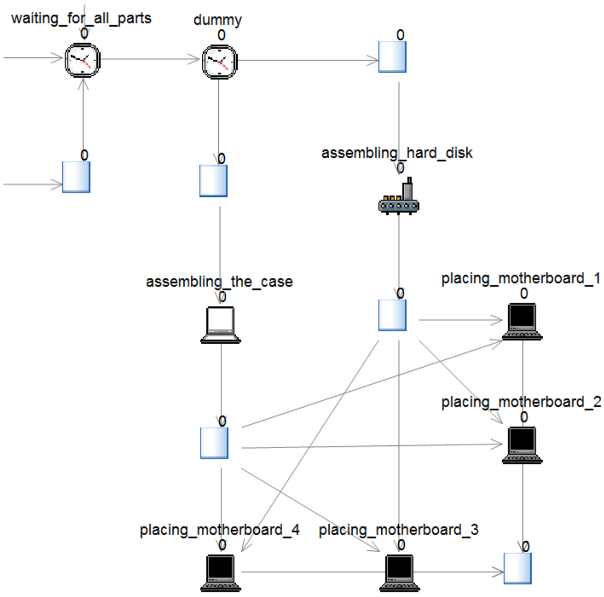

Figure 8. Simul8 screenshot: Main Assembly

3.6 Quality Assurance

When routing in to the quality check, we start with visual logic after loading the work item to remove the concerned laptop from the global number of laptops in the system, as we consider the manufacturing process complete here. Given that we have two type 4 workers that are required by this activity, we set replicate to 2. On exit, we define two new labels for the departure time of the laptop from the system and the cycle time which is just the difference between the start time and departure time. After this activity we have a dummy activity to add the cycle time to a new global variable which counts the total cycle time of all laptops that have passed through the system. We also define another new global variable to count the total number of laptops we have processed.

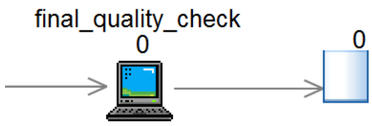

Figure 9. Simul8 screenshot: Quality Check

3.7 Boxing

After all the laptops within the order are finished and we assured their quality, we wish to box laptops based on the order size and the order number. Even though in our system the laptops are identical, we do not allow laptops from different orders to be mixed together. This could be important when simulating more complex systems where we allow for laptop personalization and keeping track of the order is essential. Nevertheless, once the laptop is checked for quality it enters the queue for boxing. The boxing activity checks the order size label and assembles laptops into one box based on the spreadsheet shown below. Here we used the Label/Sheet option that enabled us to define how many work items will be collected, based on another work item's label value as described in [2]. This way, when the work item enters the boxing activity and its order size label is $x \in \{1,2,3,4,5\}$ then the activity collects corresponding number of work items based on the spreadsheet. We also use the match option based on order number label to ensure that laptops only from the same order are packed together. Since the activity requires one type 5 worker, we created another copy of that activity. After the order is packed, it is put on a track and shipped to consumers. Here we assume that some other department keeps track of shipping orders to the actual clients.

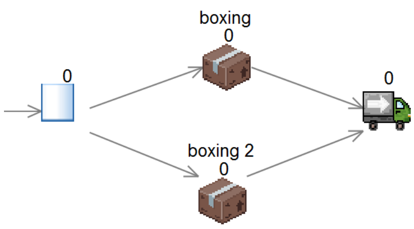

Figure 10. Simul8 screenshot: Boxing

lbl_OrderSize	Queue for boxing
1	1
2	2
3	3
4	4
5	5

Figure 11. Spreadsheet for Label/Sheet option

3.8 Raw Material Inventory

We have two raw materials with limited availability, the supplies of which need to be replenished when they fall below a certain level. To achieve this we create two small "sub-models" which each have a start point limited to just one work item. This work item represents the observer that comes and checks the stock of our materials. In each case we have visual logic on routing into the activity which blocks the routing for as long as the inventory levels are above the given threshold. We then have routing out logic on both activities to replenish the inventories, adding the designated number of new materials to the current global inventory level variables. Each of these activities has a fixed duration, equal to the length of time it takes the ordered material to arrive, according to the brief. Once this activity has been completed, the observer returns to the queue, ready to order more of a material once it starts to run out.

Figure 12. Simul8 screenshot: Material Inventory

3.9 Overview

To conclude this section of the report, here we see a screenshot of our entire model in Simul8 so that you can see how each of these processes explained above fit into the overall picture, alongside all of the trivial processes that we have omitted. Note that the sub-system in the bottom left of figure 13 relates to the recording of performance measures and will be explained later on.

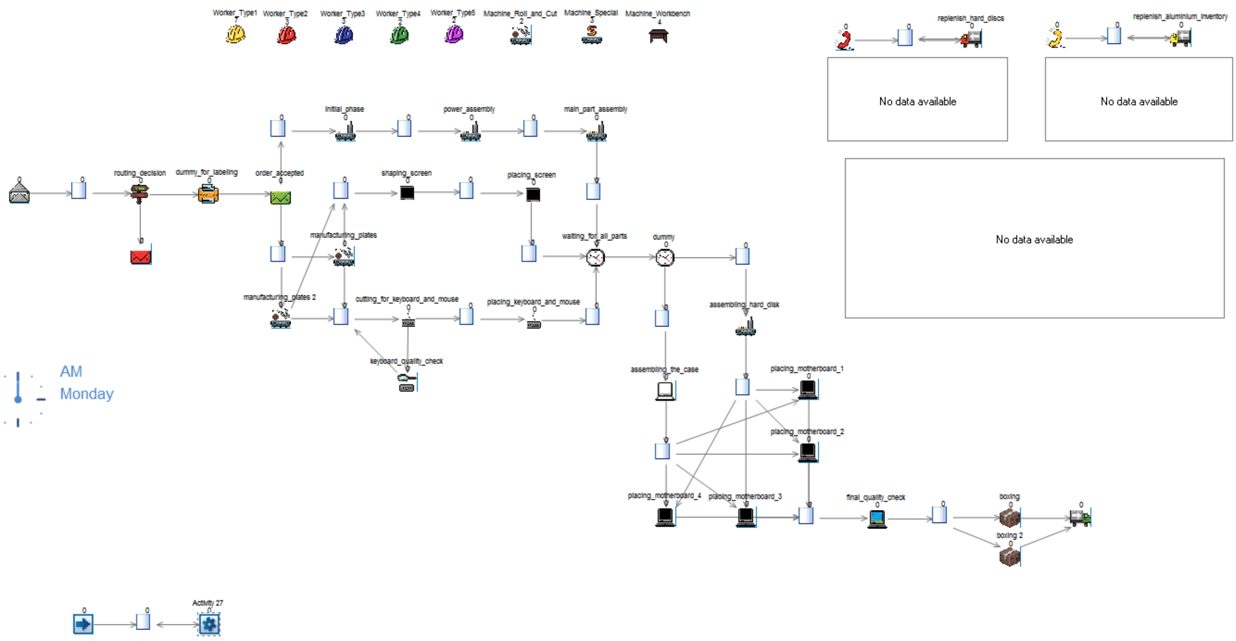

Figure 13. Simul8 screenshot: Full Model

4 Output Analysis

To analyse the functionality of our system, we will need to record some performance measures. We will start with the average cycle time of a laptop. As mentioned before, we start by time-stamping each laptop with a start time label as they begin the production process. We then time-stamp them again with an end time label when they leave quality check. We consider this to be the end of the production process, as beyond this point laptops sit around in the boxing queue waiting for other laptops in their order before they can be boxed. This means if we record the end time any later than quality check, all laptops will have a cycle time equal to the slowest laptop in the order which won't give us particularly interesting results. At this stage, we also assign each laptop a cycle time label which is just the difference between its start and end times.

In order to carry out output analysis on a steady-state system we first need to calculate a warm-up time to ensure we are taking performance measures from a system which is in a stable state. We do this by looking at the moving average of the last 10 laptops in the system. To record this, we use some visual logic when a laptop exits the quality check stage. This effectively ensures that only the previous 10 laptops are included in the average cycle time calculation, calculating a new global average cycle time variable which we can plot.

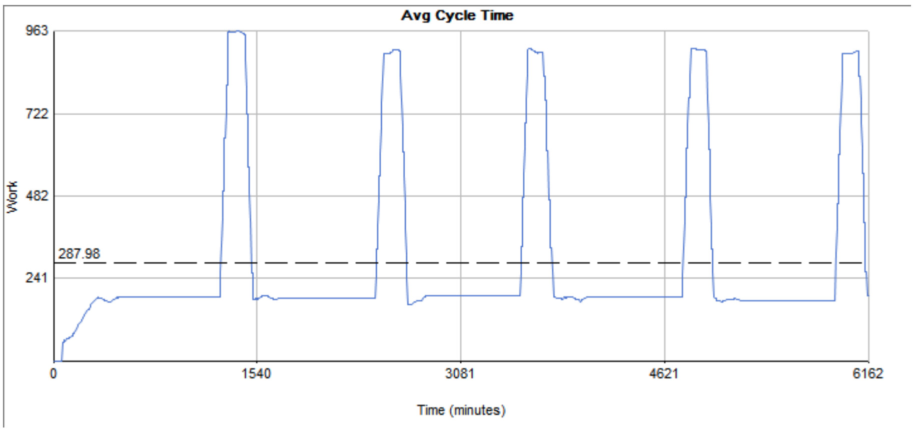

Figure 14. Moving Average Cycle Time Graph

Looking at the graph in Figure 14 of how the average cycle time of the last 10 laptops to pass through our system changes over time, we see that we reach a repetitive state after around 600 minutes, so this is the value we will use for our warm up time. We then run an initial trial using the batch means method to take the average cycle time of batches 1200 minutes long. To implement this in Simul8 we had to create a new start point, queue and activity in an isolated system separate from our model. We limit this system to just one work item arriving after 600 minutes - this allows for the warm-up period to pass. Then, after 600 minutes, our work item moves through the queue into the activity, which has a fixed duration of 1200 minutes (2 times our warm-up period). At the end of this period, we have some "On Exit" visual logic which records our average cycle time for that 1200 minute period in a spreadsheet. It also sets the global variables of total cycle time and number of laptops that have left the system to 0 so that we can start the next period of recording. The work item then returns back to the queue to be routed into the activity again to start the next 1200 minute recording period.

We started with 50 batches to help calculate how many batches we would need in total to achieve our desired confidence interval with half-width not greater than 1% of the true average cycle time. We needed to calculate

$$m = \min\{n : n \geq \frac{t^2_{n-1,1-\alpha/2} S^2_{50}}{(\bar{X}\epsilon)^2}\},$$

where $\bar{X}$ is our sample mean, S_{50} is the standard deviation of our 50 batches and ϵ is 0.01 (referring to our 1% half-width). This value of m gives us the number of batches, which turned out to be 1523. Changing the random number seed in Simul8 to ensure an independent sample, we carried out another trial of sufficient length to yield this number of batches. We obtained a sample mean of 530.85 and a sample standard deviation of 104.56, which along with our t-value of 1.96 could be combined to calculate our confidence interval of (525.60,536.12).

What we do notice is that these values for average cycle time seem a lot higher than we would expect when looking at the lengths of each of the individual processes. Looking back at Figure 14 we see that there are repetitive spikes in the average cycle time as we move through time. These will naturally skew the average cycle time, explaining why these values are so large. In the next section we will look to address these by suggesting some improvements to the system.

5 Recommendations

In this section we will investigate several possibilities for improving the efficiency of the system in order to provide helpful recommendations to the operators.

5.1 Material Order Threshold

When we look closer at why the large spikes in cycle time occur, we see that they appear to coincide with low supply of the two exhaustible materials in our system, hard discs and aluminium. The inventory level of both of these materials is monitored as the system runs, and more are ordered when the inventory level falls below a certain value. The issue is that there is a lengthy delay in the arrival of these materials once they have been ordered. Both are ordered once there are less than 10 of them in the system and take 15 hours to arrive. We will investigate how raising the threshold of inventory at which we order more of each material will affect the average cycle time of a laptop. The thinking here is that this will reduce, or even eliminate the large period of time in which the whole process is forced to stop due to there being a shortage of either of these materials. We set up a new system, with the only change from our original being that new hard discs and aluminium was ordered when the inventory level of each fell below 20, rather than 10 as we had before. We took samples as in the previous section, using mean batching, to get 50 average cycle lengths for each system. We then carried out a paired t-test to assess whether our new system was an improvement. To do this, we calculate the difference between each pair of cycle lengths, calculate the mean ($\bar{D}$) and standard deviation (S) of these differences and then arrive at a confidence interval of the form $\bar{D} \pm \frac{t_{n-1,1-\alpha/2}S}{\sqrt{n}}$. where n is the number of samples and α is our confidence level. We carried this out at a confidence level of 95%, giving us a confidence interval of (30.45,115.64). Since this interval does not contain zero, we can conclude that these two systems are significantly different in performance with 95% confidence, thus increasing the inventory threshold at which we order more of a given material results in a reduction of cycle time. The main point here is to decrease the period of time that we are without either of these materials. An equally effective action would be to find a supplier that delivers these items more quickly than the current one does.

5.2 Quality Checkers

When running our simulation, it's apparent that there is a frequent bottleneck at the final quality check stage. Running the simulation for 1 year and looking at the performance measures provided by Simul8, we see that the average wait time in the queue before this stage is almost 80 minutes.

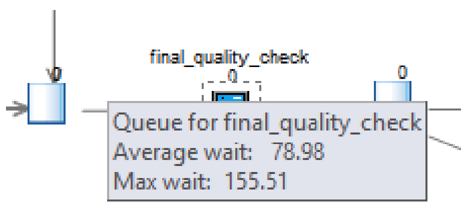

Figure 15. Simul8 screenshot: Quality Check Queue Time

This is far in excess of every other queue in the system, with the exception of those we dealt with in the previous part. Looking deeper at this process we see that it is carried out by Type 4 workers, of which there are only two. We investigated the effect of employing two more Type 4 workers, to see if this would benefit the process and if it is something the operators should consider. An immediate benefit of this is that the average wait time in this queue plummets to just 1.78 minutes. We would expect this to have a positive impact on the system as a whole however when we look at the new average cycle time of a laptop it appears to make no difference. If storing these items before they are quality checked is inconvenient or even costly for the factory operators, then this problem can be solved by employing extra staff who can carry out quality checks, or even training current staff members to do that job also.

5.3 Wasted Orders

Another issue we notice with the system is that a massive proportion of all orders made are rejected. This is due to the limit we have on how many laptops can be in the production line at any one time. We decided to look at the average number of laptops produced per minute, first with the original limit of 30 laptops in the system, and then again with this limit doubled to 60. Again, using a sample of 50 average productions per minute in each case, we carried out a paired t-test, which yielded a 95% confidence interval of (-0.010,0.002). This interval does contain zero so we can conclude that there is not a significant difference between these two systems with 95% confidence, in terms of how many laptops they can produce per minute. This suggests that loosening this constraint on how many laptops we can process at any one time does not have a negative impact on the operation of the production line. Allowing more orders to be accepted could lead to increased customer satisfaction, and in turn lead to larger long-term profits, as more customers will be able to be serviced than before without negatively impacting operations.

6 Conclusion

In this report, we began by estimating duration distributions of various processes based on historical data. We used maximum likelihood estimation, among other methods, to estimate these distributions, before evaluating their validity using goodness of fit tests, specifically, the Kolmogorov-Smirnov and Chi-squared tests. These distributions allowed us to build a comprehensive model of the laptop production process in Simul8 which we could use to run trials and evaluate various aspects using a number of performance measures. From this, we were able to recommend that the operators of the laptop factory ensure that they replenish the stocks of their two limited-supply materials more quickly, to ensure that the overall efficiency of the process can increase. Furthermore, employing additional "quality checkers" (Type 4 workers) could have a positive impact if storing products before that are quality checked is a costly or troublesome ordeal. Finally, we highlight the benefits of allowing there to be more laptops in the system at any one time before orders start being rejected. This will have a positive impact on customer satisfaction and ultimately profits, without negatively impacting the efficiency of the factory.

References

[1] *Simul8 Support: Replicate*. Accessed: 08/04/2021. URL: https://www.simul8.com/support/help/doku.php?id=features:replicate.

[2] *Simul8 Support: Collect*. Accessed: 08/04/2021. URL: https://www.simul8.com/support/help/doku.php?id=features:collect.