SYSC 4005 Milestone 3

Group # 11

YouHeng Zhou – 101078065

Imran Latif – 101073950

Leenesh Kumar - 101115874

Date of Submission: April 7th, 2023

1. Model Verification

In this section we are performing model verification. Model verification is done to ensure that the system we are building makes sense, and we have done so using a variety of ways such as checking our component item counts in our buffers cannot be more than 2 or less than 0, our product creation throughput cannot be negative, and the total count of products can't be more than the number of components inspected.

We will be able to get these tests completed using the data we gather from our simulator, such as the csv file we generate from running our simulator below, which produces information relevant to the simulator's states, such as product throughputs, inspector/workstation blocked/idle times, buffer average occupancies at different state of time, components used, and the total products produced.

Since the csv file has over 4000 rows, we included only the heading and the first three rows from our csv file below:

arrival_time (minutes)	event_type	p1 throughput	p2 throughput	p3 throughput	ins1 blocked time	ins2 blocked time	ws1 idle time	ws2 idle time	ws3 idle time	c1_ws1 occp	c1_ws2 occp
0	ins2 started for c3	0	0	0	-27.983	0	0	0	0	0	0
15.413	ins2 end produce to c3_ws3	0	0	0	-12.57	0	15.413	15.413	15.413	0	0
15.413	ins2 started for c3	0	0	0	-12.57	0	15.413	15.413	15.413	0	0

Table 1a: First three rows of Output Data from Simulator

c1_ws3 occp	c2_ws2 occp	c3_ws3 occp	c1 used	c2 used	c3 used	ins1 state	ins2 state	c1_ws1 capacity	c1_ws2 capacity	c3_ws3 capacity	c2_ws2 capacity	c3_ws3 capacity	ws1 state	ws2 state	ws3 state	p1 produced	p2 produced	p3 produced
0	0	0	1	0	1	1	3	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	2	1	3	0	0	0	0	1	0	0	0	0	0	0

Table 1b: First three rows of Output Data from Simulator

From reading our csv file, we can gather that our buffer's occupancies are not over 2 and less than 0, because they are all within the range of 0 or 1 or 2 for the columns of c1_ws1 capacity, c1_ws2 capacity, c1_ws3 capacity, c2_ws2 capacity, c3_ws3 capacity, which are the occupancies for all of our buffers between the inspectors and the workstations.

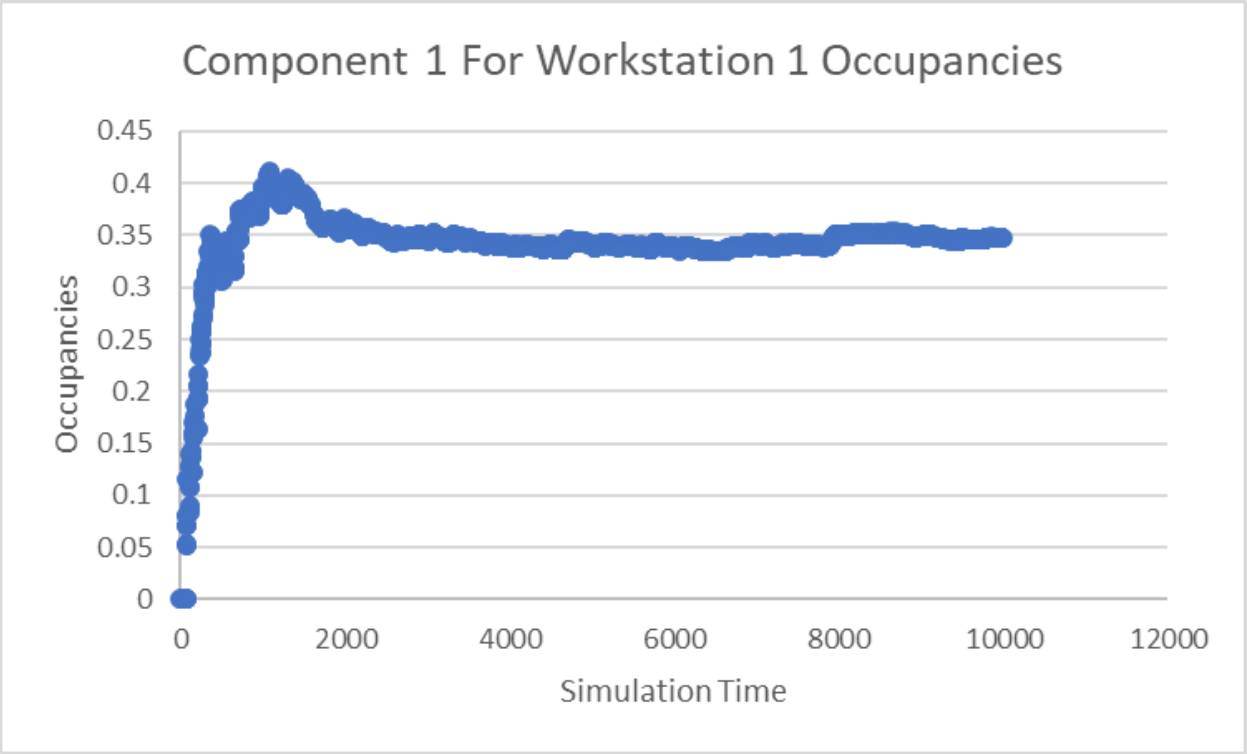


Figure1: Component 1 For Workstation 1 Occupancies

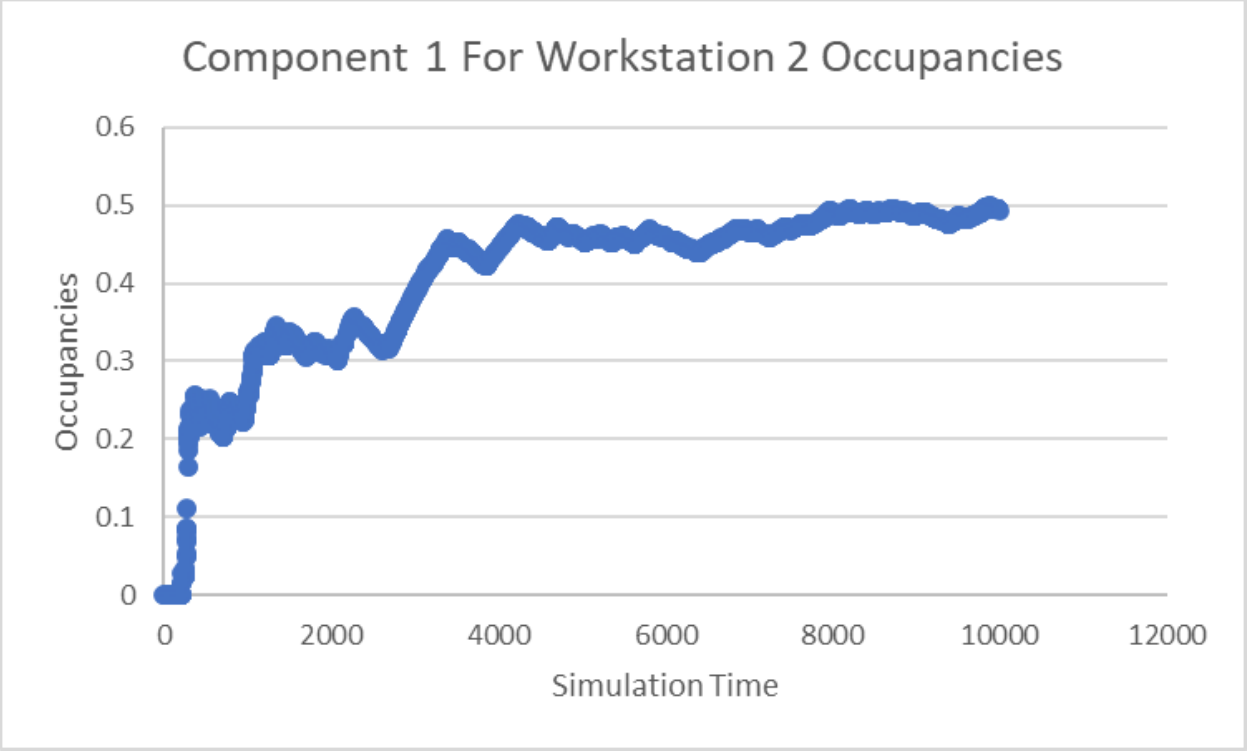


Figure 2: Component 1 for Workstation 2 Occupancies

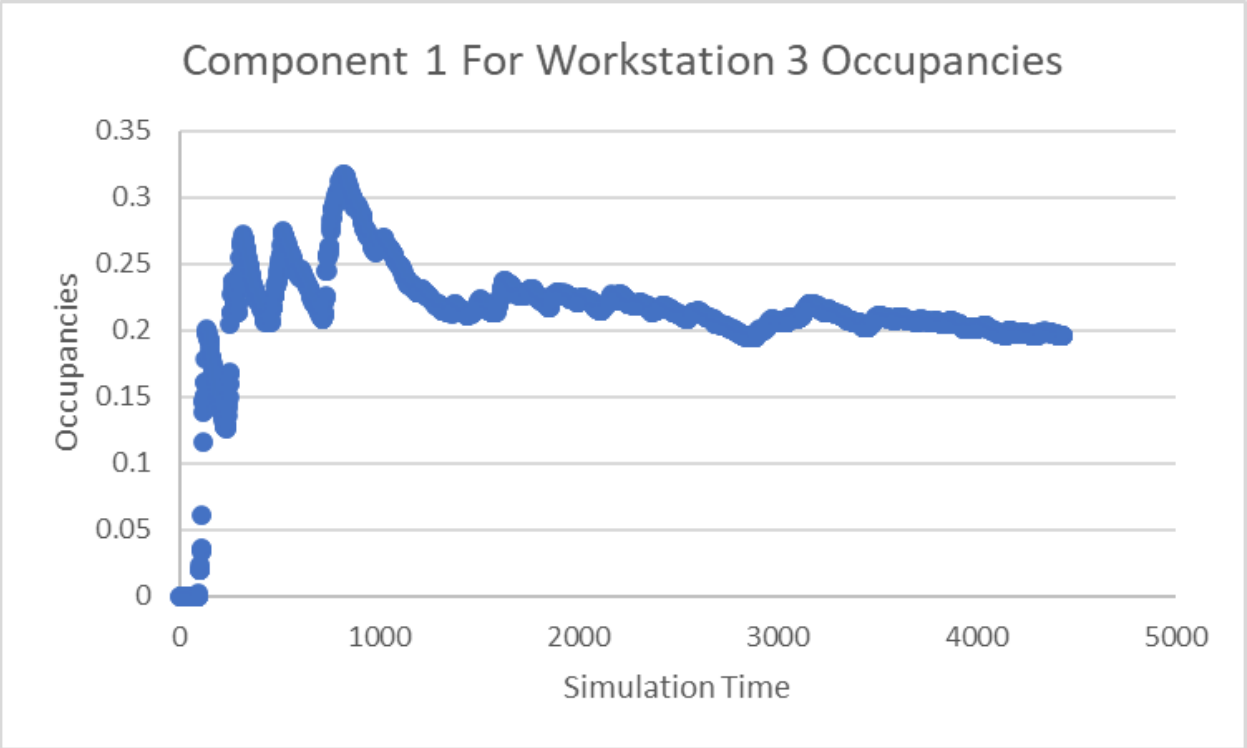


Figure 3: Component 1 for Workstation 3 Occupancies

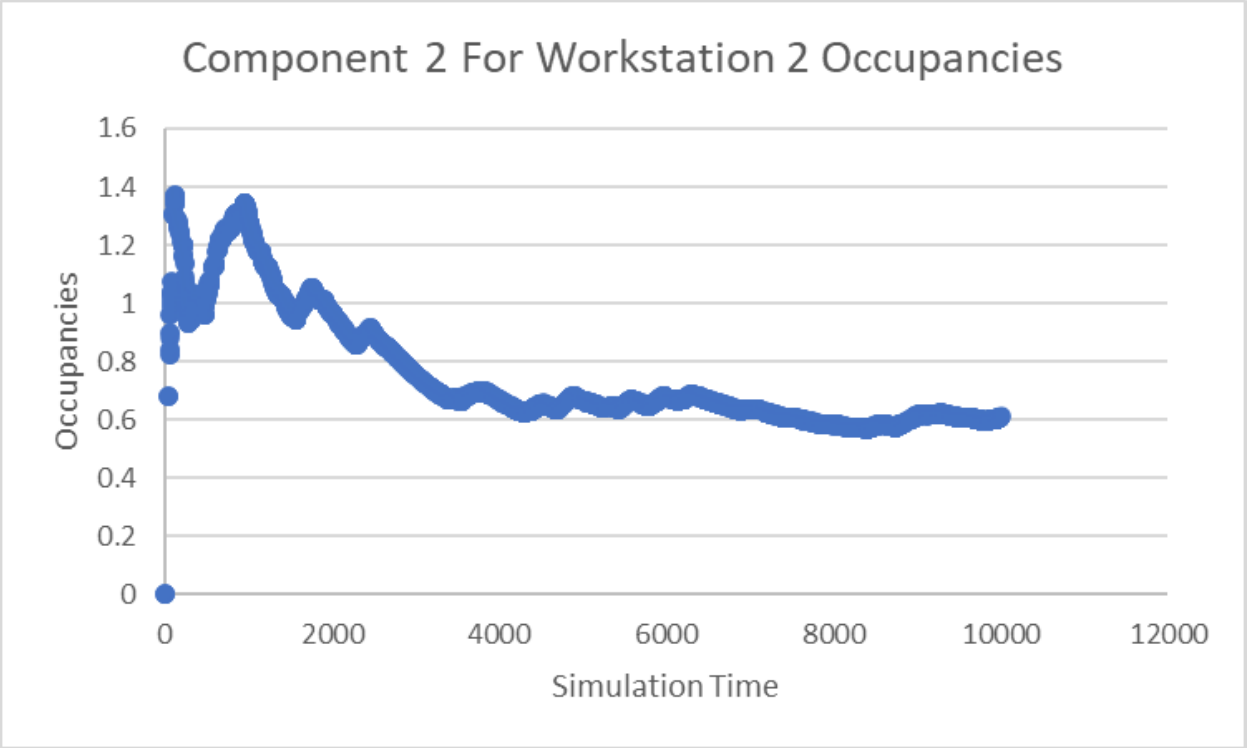


Figure 4: Component 2 for Workstation 2 Occupancies

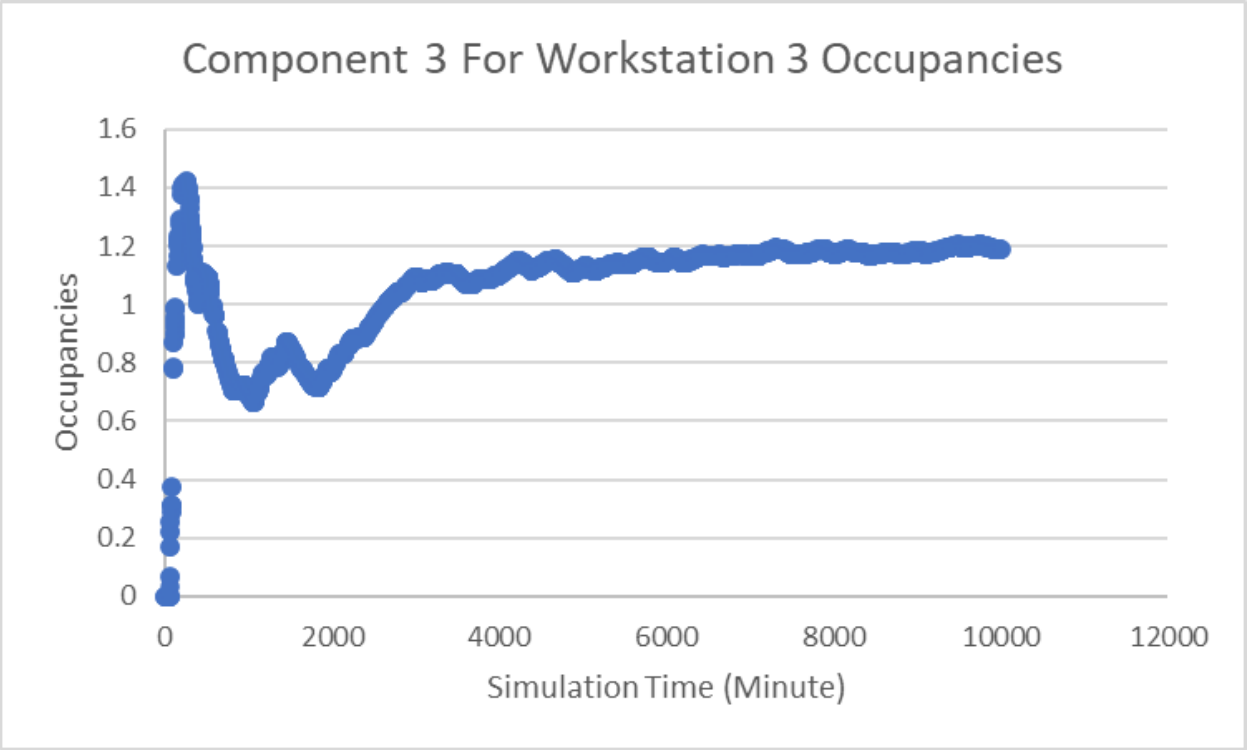


Figure 5: Component 3 for Workstation 3 Occupancies

And as we can see, none of the component buffer occupancies ever exceed 2 and become less than 0, which verifies that our simulation makes sense for our component buffers.

Next, we can check that our throughput are not negative for the entire duration of the simulation:

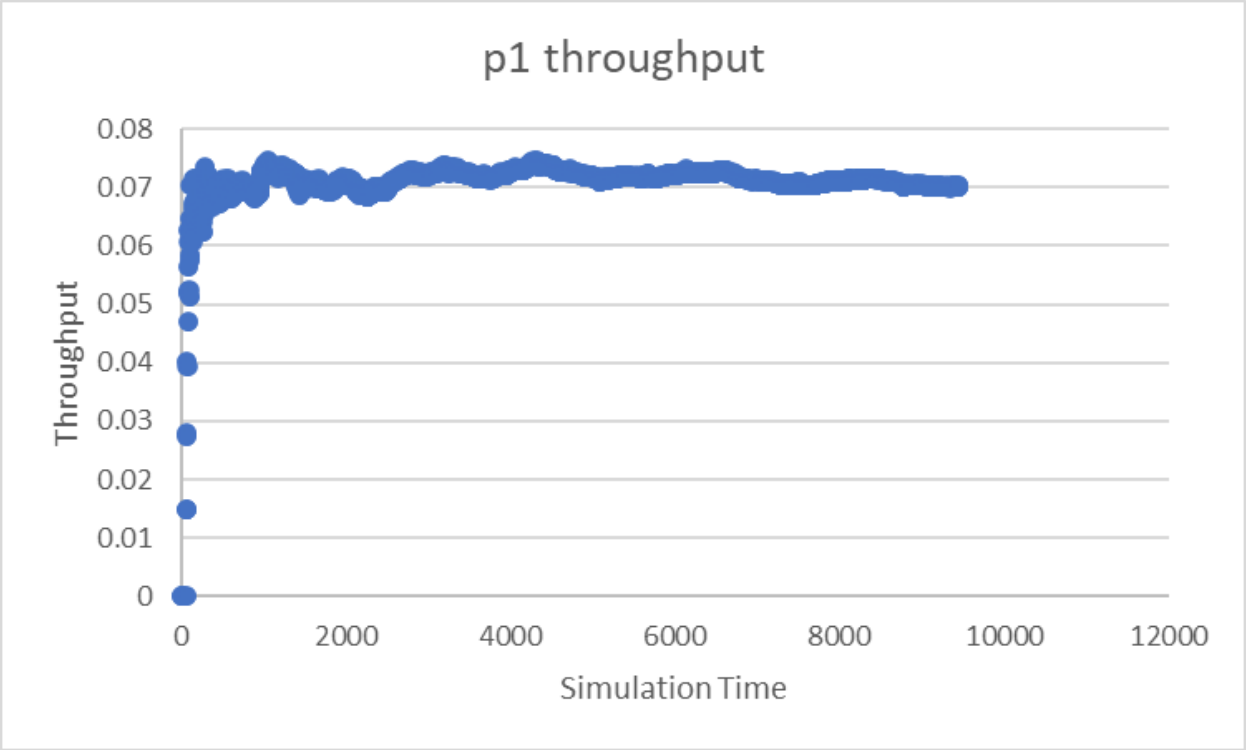


Figure 6: P1 Throughput

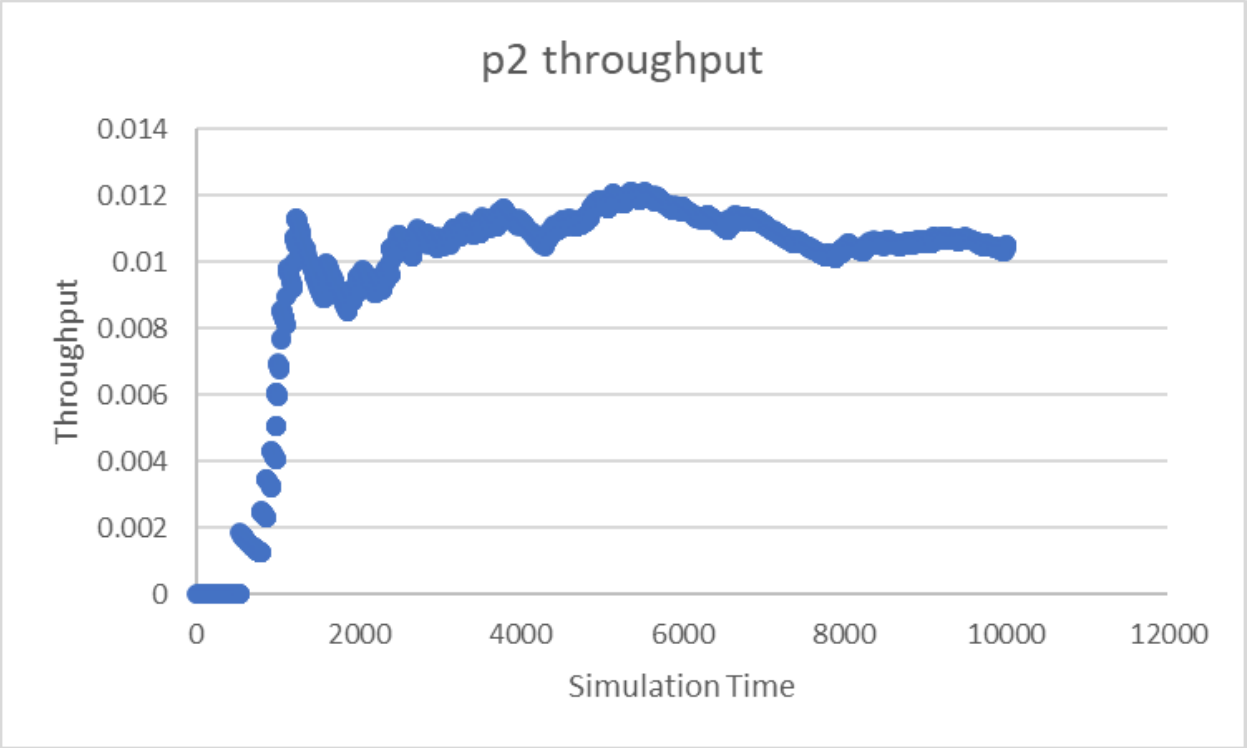


Figure 7: P2 Throughput

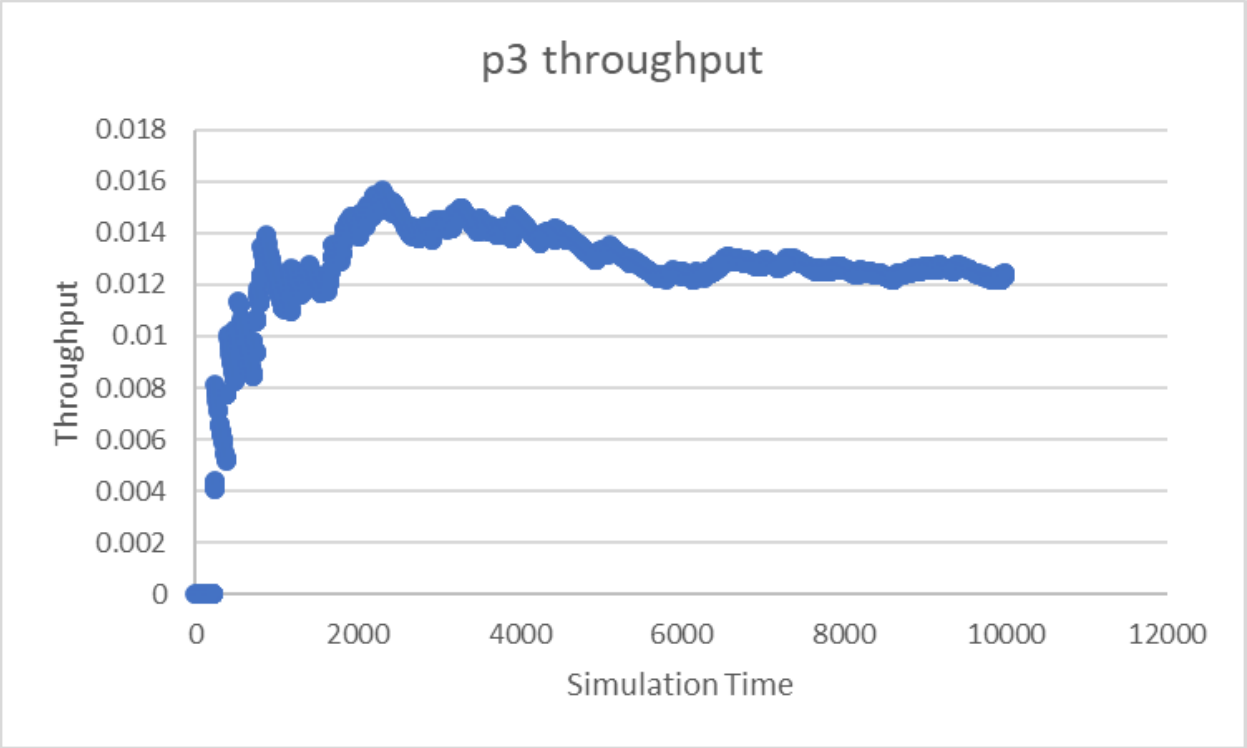


Figure 8: P3 Throughput

From the 3 graphs above, we can tell that for a simulation time that lasts for 10000 minutes, our throughput rate never goes below 0, which confirms that our simulation makes sense for product throughput.

Next, we have created a flow chart to verify our model through depicting the states and events flow in our simulation, which we changed to use FEL from using threads in Milestone 1.

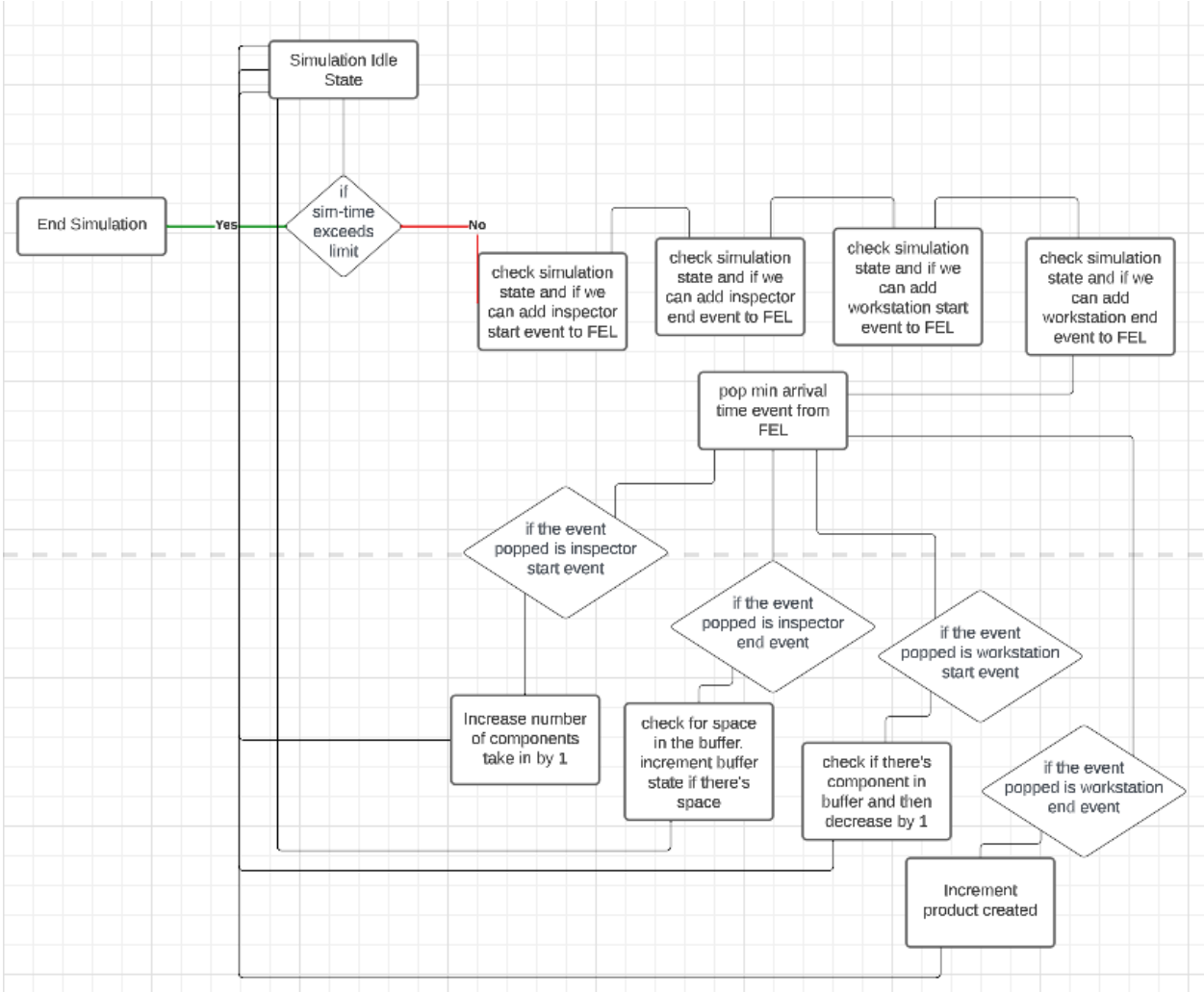


Figure 9: State Flow Chart

To briefly explain our states in the FEL implementation simulation, we first check the simulation states, that is information such as whether the inspector is idle or working, whether the workstation is idle or working, whether the buffer has components in it, and we create events based on those states.

Those events are then placed in a priority queue, which is our FEL, and are sorted by the event's arrival times.

After which, when the min arrival time event is popped from the FEL, we check for the state to see if we can change it, whether that is to increase the buffer's capacity by 1, or decrease the components in the buffer to let the workstation create the next new product.

Different Validation alternatives

There are several other methods that can be used in order to determine if the set of output data is valid, as in are we building our model that reflects the historical data. Some ways we can do this include using little's law at multiple levels of granularity including at the facility's level and at the level of the different buffers, which we will do in the next section.

We can also perform input output analysis with confidence levels using t-table stat calculations to see if the historical value from the project descriptions are within our range of confidence level, which we set the alpha level to 0.05, which is 95% confidence level.

If the historical data falls within the range of our confidence level, our model would be valid, and we can confirm we are building the right model that adheres to the historical values.

2. Little's Law Analysis

We can use Little's law to further verify that our model makes sense. Little's law states that L, the average number of items in the system, equals to lambda times W.

Where lambda is the average rate at which items arrive to the system.

And W is the average time that an item spends in the system.

For our system, we would perform Little's law analysis for two levels, which are at the facility's level for each of the input component types, and at the buffer level for each of the component buffers.

To accomplish this, we will find the time interval of how long each component in the buffer and each component in the facility stays in the buffer and in the facility.

```
# find prev_sim_time for average buffer occupancy calculations
prev_sim_time = sim_time

# pop min event from fel
sim_time, event_type = heapq.heappop(fel)

# calculate average buffer occupancies
time_interval = sim_time - prev_sim_time

# time interval of component in the buffer
c1_ws1_s += c1_ws1 * time_interval
c1_ws2_s += c1_ws2 * time_interval
c1_ws3_s += c1_ws3 * time_interval
c2_ws2_s += c2_ws2 * time_interval
c3_ws3_s += c3_ws3 * time_interval

# time interval of component in the facility
c1_l += c1_s * time_interval
c2_l += c2_s * time_interval
c3_l += c3_s * time_interval

if not sim_time == 0:
    c1_ws1_o = Decimal(c1_ws1_s) / Decimal(sim_time)
    c1_ws2_o = Decimal(c1_ws2_s) / Decimal(sim_time)
    c1_ws3_o = Decimal(c1_ws3_s) / Decimal(sim_time)
    c2_ws2_o = Decimal(c2_ws2_s) / Decimal(sim_time)
    c3_ws3_o = Decimal(c3_ws3_s) / Decimal(sim_time)

    c1_o = Decimal(c1_l) / Decimal(sim_time)
    c2_o = Decimal(c2_l) / Decimal(sim_time)
    c3_o = Decimal(c3_l) / Decimal(sim_time)
```

Figure 10: code snippet for finding the time interval for finding L value

And following getting our L value, we can find our lambda value, which is our take-in rate for components, and W, which is the time the component spends in the facility.

```
# total components little's law
print('facility littles law')
print(f'L c1: {c1_o}')
print(f'L c2: {c2_o}')
print(f'L c3: {c3_o}')

print(f'lambda c1 take in rate: {c1_s/sim_time}')
print(f'lambda c2 take in rate: {c2_s/sim_time}')
print(f'lambda c3 take in rate: {c3_s/sim_time}')

print(f'W c1: {c1_l/c1_s}')
print(f'W c2: {c2_l/c2_s}')
print(f'W c3: {c3_l/c3_s}')
```

Figure 11: Code snippet for finding lambda

Here is Little's law analysis for all the components and products in our system at the facility level:

```
facility littles law
L c1: 4.98941134562335290838401980 1
L c2: 1.76318654374846854884712946 9
L c3: 2.51803982313690390899703619 7
lambda c1 take in rate: 0.00059993023739015051025077784 20
lambda c2 take in rate: 0.00039995349159343367350051856 13
lambda c3 take in rate: 0.00009998837289835841837512964033
W c1: 8316.652561685445882708533073
W c2: 4408.47893769810524869072632
W c3: 25183.32632231726609825517296
```

Figure 12: code output snippet for L, lambda, and W for each of the component types

Component Type:	Component 1
L	4.98941134562335290838401980 1
Lambda	0.00059993023739015051025077784 20
W	8316.652561685445882708533073

Here is our calculations for verifying Component 1's L value:

$$L = \lambda * \hat{w}$$

$$L = 0.00059993023739015051025077784 20 * 8316.652561685445882708533073$$

$$L = 4.98941135$$

Component Type:	Component 2
L	1.763186543748468548847129469
Lambda	0.000399953491593433673500518 5613
W	4408.4789376981052486907263 2

Here is our calculations for verifying Component 2's L value:

$$L = \lambda * \hat{w}$$

$$L = 0.00039995349159343367350051856 13 * 4408.4789376981052486907263 2$$

$$L = 1.76318654$$

Component Type:	Component 3
L	2.518039823136903908997036197
Lambda	0.000099988372898358418375129640 33
W	25183.32632231726609825517296

Here is our calculations for verifying Component 3's L value:

$$L = \lambda * \hat{w}$$

$$L = 0.00009998837289835841837512964033 * 25183.32632231726609825517296$$

$$L = 2.51803982$$

And as we can see from the calculations above, the L values of each of the components in the facility level matches up with the Lambda multiplied by the W values respectively for each of the components, which means little's law holds true and our simulation is verified to make sense for each of the components at the facility level.

Next, we will verify little's law for each of our 5 buffers in our simulation.

```
buffers littles law
L c1_ws1 average occupancy: 0.35835980239607281013184554 63
L c1_ws2 average occupancy: 0.62895117821346786255228135 33
L c2_ws2 average occupancy: 0.62797667965701560315689400 40
L c1_ws3 average occupancy: 0.19620254055960413770146822 61
L c3_ws3 average occupancy: 1.21307386284429053405566198 9
lambda c1_ws1: 0.074479252701779870276777337 07
lambda c1_ws2: 0.013796156876302848453953385 93
lambda c2_ws2: 0.013796156876302848453953385 93
lambda c1_ws3: 0.011396825245641483505439753 59
lambda c3_ws3: 0.011396825245641483505439753 59
W c1_ws1: 4.81153864192180456071524642 8
W c1_ws2: 45.588868251693642475566420 46
W c2_ws2: 45.518232743182854246918516 11
W c1_ws3: 17.215543480816148157174255 77
W c3_ws3: 106.43963004593840001940671 18
```

Figure 13. code output snippet for L, lambda, and W for each of the buffers

Buffer Name:	C1 for Workstation 1
L	0.35835980239607281013184554 63
Lambda	0.074479252701779870276777337 07
W	4.81153864192180456071524642 8

$$L = \lambda * \hat{w}$$

$$L = 0.074479252701779870276777337\ 07 * 4.81153864192180456071524642 8$$

$$L = 0.358359802$$

Buffer Name:	C1 for Workstation 2
L	0.628951178213467862552281353
Lambda	0.013796156876302848453953385 93
W	45.588868251693642475566420 46

$$L = \lambda * \hat{w}$$

$$L = 0.013796156876302848453953385 93 * 45.588868251693642475566420 46$$

$$L = 0.628951178$$

Buffer Name:	C2 for Workstation 2
L	0.627976679657015603156894004 0
Lambda	0.013796156876302848453953385 93
W	45.518232743182854246918516 11

$$L = \lambda * \hat{w}$$

$$L = 0.013796156876302848453953385 93 * 45.518232743182854246918516 11$$

$$L = 0.62797668$$

Buffer Name:	C1 for Workstation 3
L	0.19620254055960413770146822 61
Lambda	0.011396825245641483505439753 59
W	17.215543480816148157174255 77

$$L = \lambda * \hat{w}$$

$$L = 0.011396825245641483505439753 59 * 17.215543480816148157174255 77$$

$$L = 0.196202541$$

Buffer Name:	C3 for Workstation 3
L	1.213073862844290534055661989
Lambda	0.011396825245641483505439753 59
W	106.439630045938400019406711 8

$$L = \lambda * \hat{w}$$

$$L = 0.011396825245641483505439753 59 * 106.439630045938400019406711 8$$

$$L = 1.21307386$$

As we can see from the calculations above, each of our buffer's Lambda multiply W calculations equates to the found L value, which means our little's law analysis passes for both the facility level for each of the component types and for the granularity level of each of the buffers.

3. Finding Number of Replications Using OC Curve

We want to find the number of replications that are enough while also minimizing our type 1 and type 2 errors.

In order to start showing we have enough replications to minimize our type 1 and type 2 errors, we must first provide our hypothesis.

Our null hypothesis is that the simulation data of the buffer occupancies matches the historical buffer occupancies value provided in the project description file.

Our alternative hypothesis states that the buffer occupancy simulation data is different from the historical values of the system.

In order to determine the number of replications that are required for our system, we first define that we want our alpha level to be 0.05 and we want a 95% confidence interval level for our system. In addition, we want a beta value of less than or equal to 0.20, which means we want type 1 error to happen less than 5% of the time, and type 2 error to happen less than 20% of the time.

We will be using the OC curve for the 0.05 alpha value for finding this number of replication values.

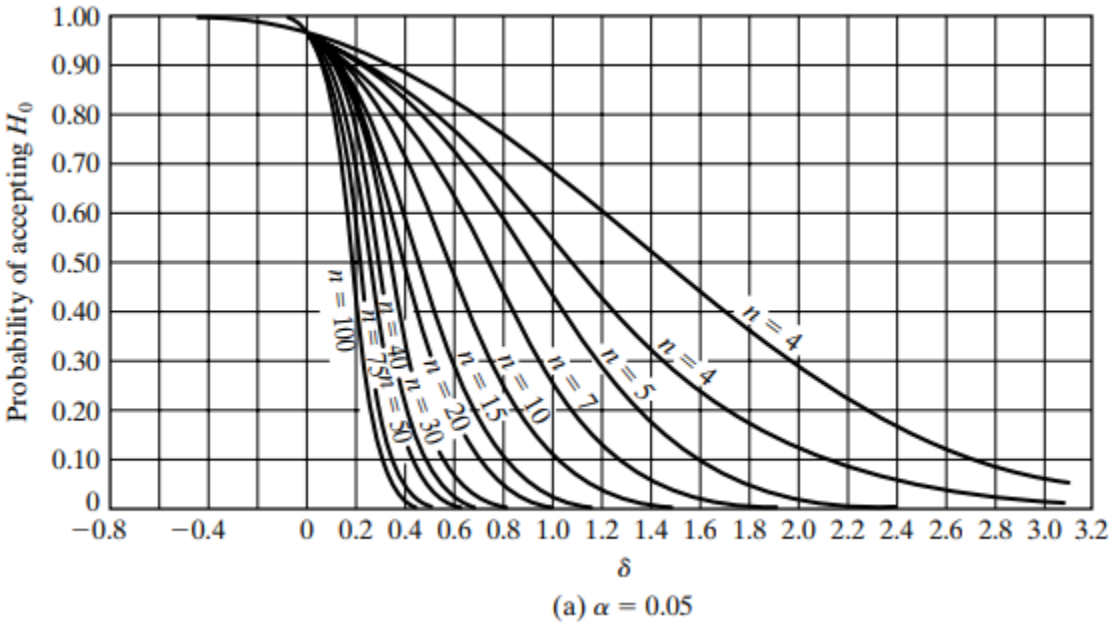


Figure 14: OC Curve for 0.05 alpha level [1, p. 550]

To use the OC Curve for 0.05 alpha level, we have to find the probability of accepting H0 on the y axis and the delta value on the axis.

First, for finding the delta value, we have to use the following equation:

$$\delta = \frac{|E(Y_2) - \mu_0|}{\sigma}$$

Figure 15: Delta equation [1, p. 404]

Our beta value would depend on the delta equation, where we find the curve that matches our given delta value and the given number of replications we used to derive the number we got from our delta equation.

Below, we will explain what the values in the delta equation are:

First, for the numerator of our delta equation, which is the critical difference, we can define a number for the critical difference found by using the following equation:

$$CD = |E(y_2) - \mu_0| < \varepsilon$$

Where $E(y_2)$ is the average buffer occupancies per the replications we did for our model and μ_0 is given by the project description as the historical buffer occupancies data. And we want to set this value to be reasonable, which was suggested to be 0.1 for the project.

Next, we will find the standard deviation of our sample replications for our denominator, and once we divide the two numbers, we get our delta value.

First, we will perform 5 sample replications to get our sample mean and sample standard deviation, and they are as follows:

Replications	Component 1 for Workstation 1 Occupancy per Minute	Component 1 for Workstation 2 Occupancy per Minute	Component 2 for Workstation 2 Occupancy per Minute	Component 1 for Workstation 3 Occupancy per Minute	Component 3 for Workstation 3 Occupancy per Minute
1	0.3875912841	0.6682245149	0.6191603362	0.2681439167	1.02913489
2	0.3266175605	0.5937144283	0.568270981	0.1737639666	1.27134742
3	0.3232813405	0.4618622212	0.8551631523	0.2359843353	1.058653018
4	0.3980589057	0.574352931	0.7387385194	0.2806613754	1.028737316
5	0.3515240344	0.4590412964	0.821799383	0.2909379517	0.9939913241

Below, we use the follow equation for finding the standard mean for each of the buffers for 5 replications:

$$(\overline{x}) = \frac{\sum_{i=1}^{n} x_i}{n}$$

and the following equation for finding the sample standard deviation for each of the buffers for 5 replications:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

Replications	Component 1 for Workstation 1 Occupancy per Minute	Component 1 for Workstation 2 Occupancy per Minute	Component 2 for Workstation 2 Occupancy per Minute	Component 1 for Workstation 3 Occupancy per Minute	Component 3 for Workstation 3 Occupancy per Minute
1	0.3875912841	0.6682245149	0.6191603362	0.2681439167	1.02913489
2	0.3266175605	0.5937144283	0.568270981	0.1737639666	1.27134742
3	0.3232813405	0.4618622212	0.8551631523	0.2359843353	1.058653018
4	0.3980589057	0.574352931	0.7387385194	0.2806613754	1.028737316
5	0.3515240344	0.4590412964	0.821799383	0.2909379517	0.9939913241
Mean:	0.35741462504	0.55143907836	3.6031323719	1.2494915457	5.3818639681
Standard Deviation:	0.03431773469 4506	0.09015604734 7067	0.12467126260 632	0.04731161983 1764	0.11137365057 187

Next, we find the critical difference value, which is the difference between our sample mean and the historical value given in the project description, and that it should be less than 0.1.

Replications	Component 1 for Workstation 1 Occupancy per Minute	Component 1 for Workstation 2 Occupancy per Minute	Component 2 for Workstation 2 Occupancy per Minute	Component 1 for Workstation 3 Occupancy per Minute	Component 3 for Workstation 3 Occupancy per Minute
1	0.3875912841	0.6682245149	0.6191603362	0.2681439167	1.02913489
2	0.3266175605	0.5937144283	0.568270981	0.1737639666	1.27134742
3	0.3232813405	0.4618622212	0.8551631523	0.2359843353	1.058653018
4	0.3980589057	0.574352931	0.7387385194	0.2806613754	1.028737316
5	0.3515240344	0.4590412964	0.821799383	0.2909379517	0.9939913241
Mean:	0.35741462504	0.55143907836	0.72062647438	0.24989830927	1.07637279362
Difference Between Mean And Historical Value:	0.077414625	0.141439078	0.120626474	0.0701016907	0.673627206
Standard Deviation:	0.03431773469 4506	0.09015604734 7067	0.12467126260 632	0.04731161983 1764	0.11137365057 187

Next, once we have the difference between our sample mean and the historical value, which is our critical difference, and the sample standard deviation values, we can find the delta values for each of our buffers.

$$\delta = \frac{CD}{\hat{\sigma}}$$

For Component 1 for Workstation 1 buffer:

$$\delta = \frac{0.077414625}{0.034317734694506}$$
$$\delta = 2.25581979$$

For Component 1 for Workstation 2 buffer:

$$\delta = \frac{0.141439078}{0.090156047347067}$$
$$\delta = 1.56882519$$

For Component 2 for Workstation 2 buffer:

$$\delta = \frac{0.120626474}{0.12467126260632}$$
$$\delta = 0.967556368$$

For Component 1 for Workstation 3 buffer:

$$\delta = \frac{0.0701016907}{0.047311619831764}$$
$$\delta = 1.48170134$$

For Component 3 for Workstation 3 buffer:

$$\delta = \frac{0.673627206}{0.11137365057187}$$
$$\delta = 6.04835347$$

Next, we can finally use the Operating Characteristic Curve for alpha = 0.05 to find the minimum number of replications we need while minimizing type 1 and type 2 errors.

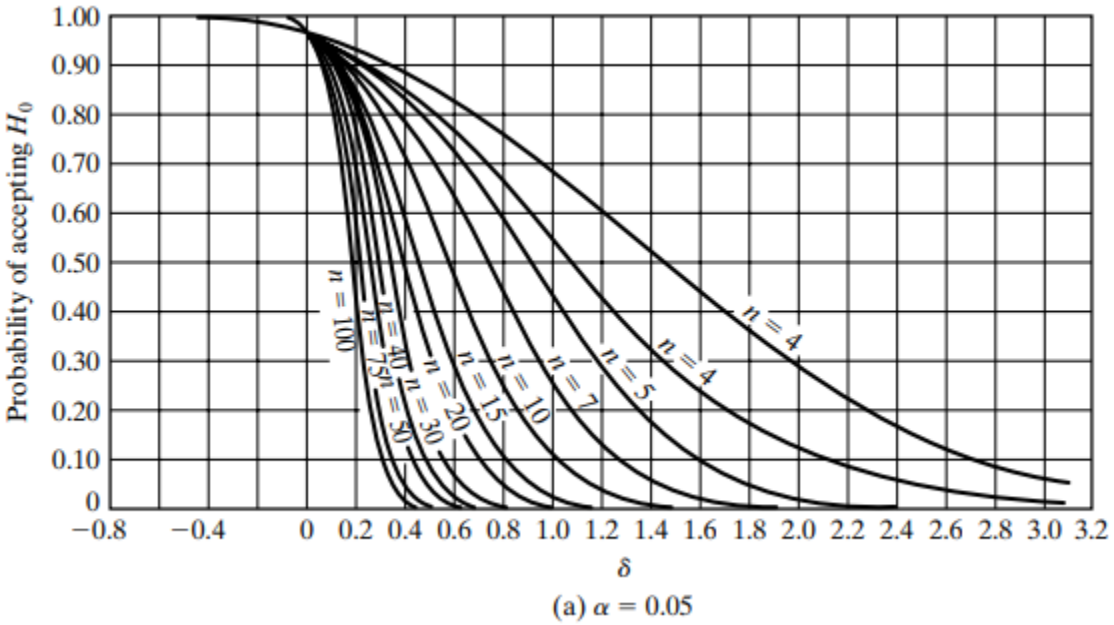


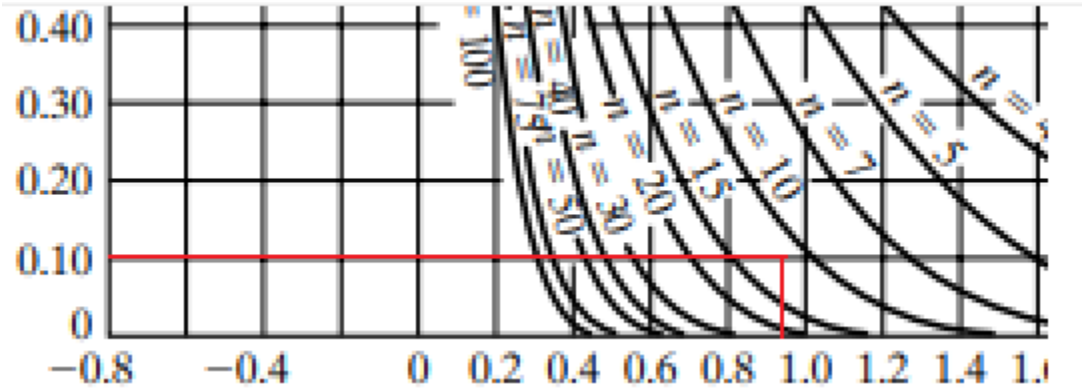
Figure 16: OC Curve for 0.05 alpha level [1, p. 550]

For our y axis, we use our beta value, which we were recommended for this project to use 0.1.

For our x axis, which is our delta variable, we have picked $\delta = 0.967556368$ from Component 2 for Workstation 2 buffer. And this is because for the lower the delta value, the more replications we need to minimize our type 2 error. For example, for Component 1 for

Workstation 1 buffer, our delta value is 2.25581979, and with a beta value of less than or equal to 0.1, we would only need 4 replications.

But for 0.967556368 from Component 2 for Workstation 2 as our x-axis, and with 0.1 as our beta value, we find that we need around 12 replications, as shown by our graph below:



With the lines touching between n = 15, and n = 10, we can therefore determine that the number of replications to minimize our type 1 and type 2 errors with an alpha value of 0.05 and a beta value of 0.1 and below for our system is with 12 replications.

4. Finding Initialization time

After getting our replication number to minimize our type 1 and type 2 values as 12 replications, we can use ensemble averages to plot our simulation data on a throughput vs time graph. Each graph will feature 12 sets of throughput vs. simulation time data from our simulation.

In addition, the reason we chose the number of 12 replications is from the previous part of the report, where we want to control type 1 and type 2 error to an alpha value of 0.05 and beta value of 0.1 respectively.

And when we found our delta value, comparing the critical difference between our sample mean versus the historical value of 5 buffer occupancies, we found that the minimum number of replications we need for an alpha value of 0.05 and beta value of 0.1 is for 12 replications.

Below, we will display the result of running our simulation for 12 times for Component 1, Component 2, Component 3's throughput versus the simulation time in minutes:

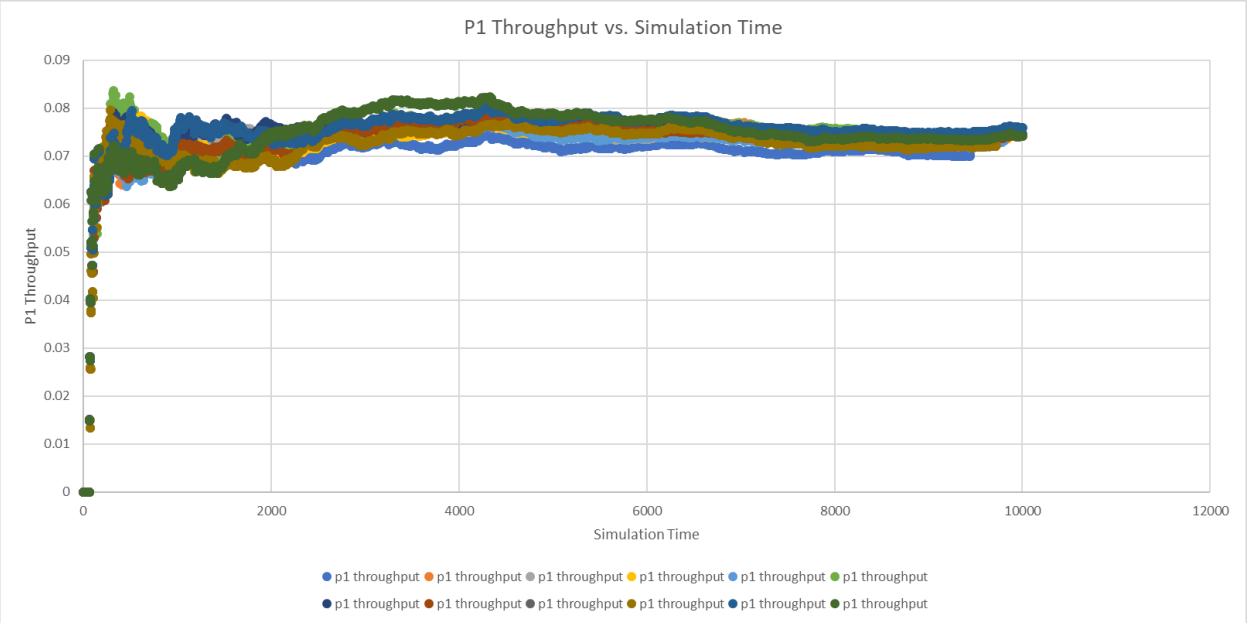


Figure 17: Product 1's throughput vs simulation time

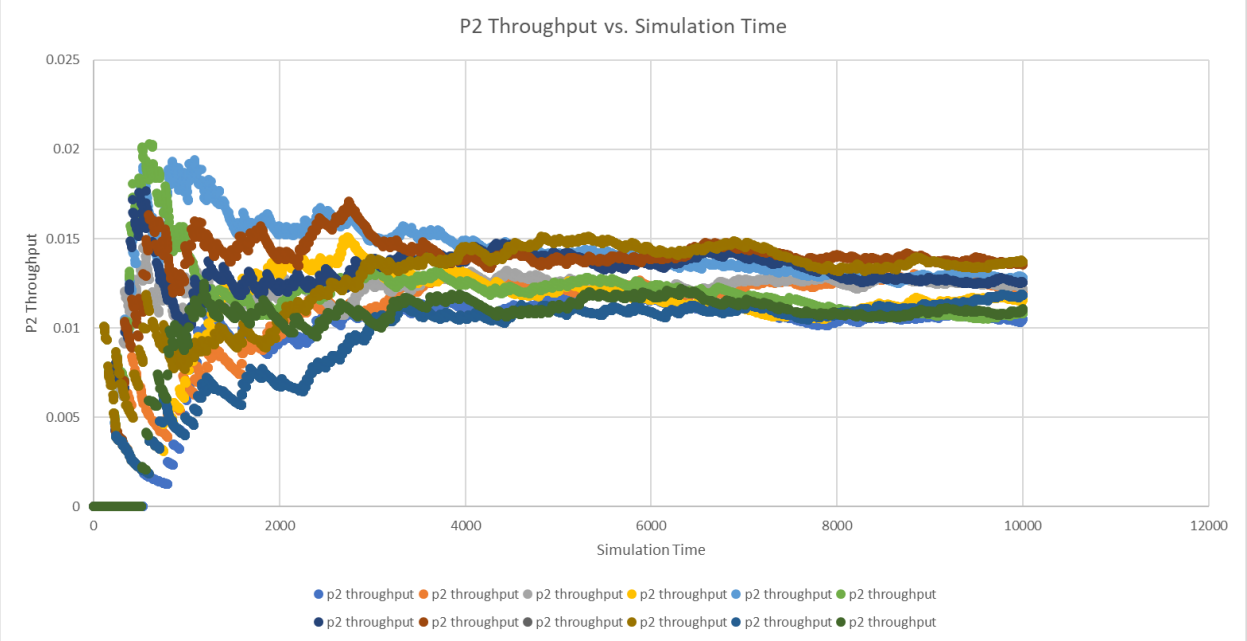


Figure 18: Product 2's throughput vs simulation time

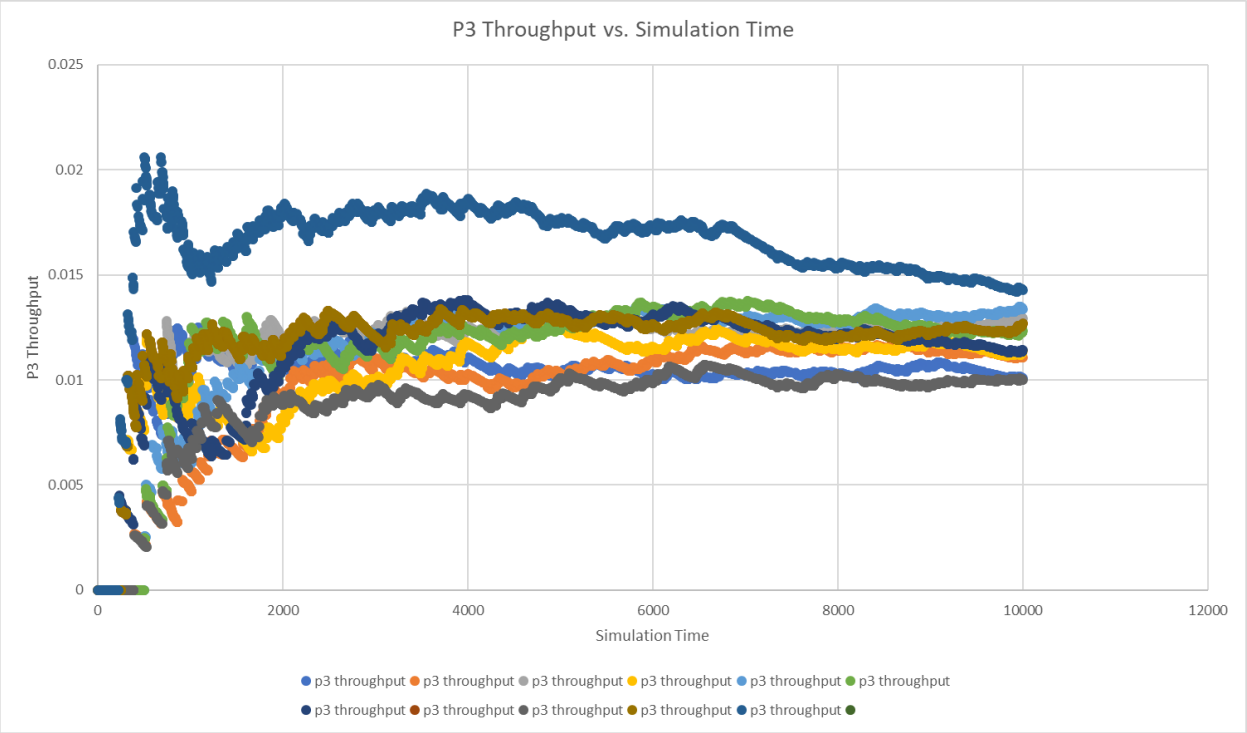


Figure 19: Product 3's throughput vs arrival time

For the first chart showing the relation between the throughput of product 1 and the simulation time from when that metric is captured. We can see that it reaches a steady state around when time is equal to 1000, on a scale from 0 to 10000. This represents that T_0 is 1000, and that T_E is 9000, for a combined T_E and T_0 totalling 10000. This shows that the initialization bias lasts for approximately 10% of the life cycle, so we are to discard that portion, leaving us with 90%% of the lifecycle where the system has reached steady state.

For the second chart showing the relation between the throughput of product 2 and the arrival time in the queue. We can see that it reaches a steady state around when time is equal to 3000, on a scale from 0 to 10000. This represents that T_0 is 3000, and that T_E is 7000, for a combined T_E and T_0 totalling 10000. This shows that the initialization bias lasts for 30% of the life cycle, so we are to discard that portion, leaving us with 70% of the lifecycle where the system has reached steady state.

For the third chart showing the relation between the throughput of product 3 and the arrival time in the queue. We can see that it reaches a steady state around when time is equal to 2000, on a scale from 0 to 10000. This represents that T_0 is 2000, and that T_E is 8000, for a combined T_E and T_0 totalling 10000. This shows that the initialization bias lasts for 20% of the

life cycle, so we are to discard that portion, leaving us with 80% of the lifecycle where the system has reached steady state.

Below there has been included a subset of the first 20 rows of each table used to create one of the figures above, to showcase the data chart was created from.

arrival_time (minutes)	p1 throughput
0	0
15.413	0
15.413	0
27.983	0
27.983	0
27.983	0
39.232	0
39.667	0
39.667	0
40.7	0.02457002457
40.7	0.02457002457
45.009	0.02221777867
45.009	0.02221777867
48.062	0.04161291665
48.062	0.04161291665
67.719	0.02953380883
67.719	0.02953380883
74.339	0.04035566795
74.339	0.04035566795

Table 2: Sample Snippet of Data For Product 1’s Throughput chart

arrival_time (minutes)	p2 throughput
231.571	0
231.571	0
235.629	0
235.629	0
241.212	0.004145730727
241.212	0.004145730727
241.749	0.004136521764
241.749	0.004136521764
242.749	0.00411948144
242.749	0.00411948144
246.605	0.004055067821
246.605	0.004055067821
247.542	0.004039718512
247.542	0.004039718512
252.974	0.003952975405
252.974	0.003952975405
254.162	0.00393449847
254.162	0.00393449847
259.52	0.003853267571
259.52	0.003853267571
262.327	0.003812036123

Table 3: Sample Snippet of Data For Product 2’s Throughput Chart

arrival_time (minutes)	p3 throughput
131.36	0
137.808	0
137.808	0
138.358	0
138.358	0
141.555	0
141.555	0
141.917	0
141.917	0
142.074	0.007038585526
142.074	0.007038585526
142.634	0.007010951106
142.634	0.007010951106
144.854	0.006903502837
144.854	0.006903502837
146.059	0.006846548313
146.059	0.006846548313
146.313	0.006834662675
146.313	0.006834662675
146.313	0.006834662675
148.142	0.006750280137

Table 4: Sample Snippet of Data For Product 3's Throughput Chart

5. Finding Confidence Interval and Input-Output Validation Using Historical Data

For this part, we will find the confidence interval for each of our measured average buffer occupancy and compare them with the given buffer occupancies from the project description.

$$CI = \bar{Y} \pm t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$$

In this equation, $\bar{Y}$ represents the sample mean. S represents the sample variance, and r represents the number of replications. We are required to have a confidence interval that is not larger than 10% of the estimated means in order to control the number of replications calculated.

And here's the sample equations for the confidence interval calculation.

$$CI = \bar{Y} \pm t * (x)$$

$$CI = \bar{Y} \pm y$$

$$CI = (w, v)$$

$$Best\ case: |v - \mu_0| < 1$$

$$Best\ case: |v - \mu_0| < 1$$

$$Worst\ case: |w - \mu_0| > 1$$

$$Worst\ case: |w - \mu_0| > 1$$

According to the best and worst case scenario calculations we are required to keep or reject the results for the system overall.

First, we want to use the new number of replications we have, which is 12, to find the sample average and the sample variance:

Replications	Component 1 for Workstation 1 Occupancy per Minute	Component 1 for Workstation 2 Occupancy per Minute	Component 2 for Workstation 2 Occupancy per Minute	Component 1 for Workstation 3 Occupancy per Minute	Component 3 for Workstation 3 Occupancy per Minute
1	0.3875912841	0.6682245149	0.6191603362	0.2681439167	1.02913489
2	0.3266175605	0.5937144283	0.568270981	0.1737639666	1.27134742
3	0.3232813405	0.4618622212	0.8551631523	0.2359843353	1.058653018
4	0.3980589057	0.574352931	0.7387385194	0.2806613754	1.028737316
5	0.3515240344	0.4590412964	0.821799383	0.2909379517	0.9939913241
6	0.3358016687	0.5677421434	0.6443263263	0.2058388192	1.179231903
7	0.3497646652	0.6521626418	0.4661118224	0.1973508942	1.328294356
8	0.3485011239	0.5558626517	0.7434008946	0.227697268	1.157459857
9	0.3607846733	0.6548999624	0.5249083916	0.2852082674	1.187339498
10	0.3582351265	0.5855258365	0.6452716632	0.1773725026	1.10721454
11	0.3612647344	0.5161272725	0.7063588662	0.2166947466	1.106169398
12	0.3797019932	0.6295617305	0.6513530787	0.2554245251	1.143914739

Below, we use the follow equation for finding the standard mean for each of the buffers for 12 replications:

$$(\overline{x}) = \frac{\sum_{i=1}^{n} x_i}{n}$$

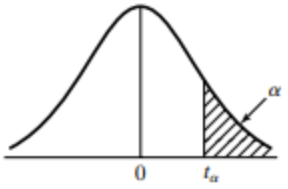
and the following equation for finding the sample standard deviation for each of the buffers for 12 replications:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}$$

That will give us the values as follows:

Replications	Component 1 for Workstation 1 Occupancy per Minute	Component 1 for Workstation 2 Occupancy per Minute	Component 2 for Workstation 2 Occupancy per Minute	Component 1 for Workstation 3 Occupancy per Minute	Component 3 for Workstation 3 Occupancy per Minute
1	0.3875912841	0.6682245149	0.6191603362	0.2681439167	1.02913489
2	0.3266175605	0.5937144283	0.568270981	0.1737639666	1.27134742
3	0.3232813405	0.4618622212	0.8551631523	0.2359843353	1.058653018
4	0.3980589057	0.574352931	0.7387385194	0.2806613754	1.028737316
5	0.3515240344	0.4590412964	0.821799383	0.2909379517	0.9939913241
6	0.3358016687	0.5677421434	0.6443263263	0.2058388192	1.179231903
7	0.3497646652	0.6521626418	0.4661118224	0.1973508942	1.328294356
8	0.3485011239	0.5558626517	0.7434008946	0.227697268	1.157459857
9	0.3607846733	0.6548999624	0.5249083916	0.2852082674	1.187339498
10	0.3582351265	0.5855258365	0.6452716632	0.1773725026	1.10721454
11	0.3612647344	0.5161272725	0.7063588662	0.2166947466	1.106169398
12	0.3797019932	0.6295617305	0.6513530787	0.2554245251	1.143914739
Sample Mean:	0.35676059253	0.57658980255	0.665405284575	0.2345898808	1.132624022
Sample Standard Deviation:	0.023030350321809	0.070432735510084	0.11018299763379	0.0415348033846	0.10044451613069

And for finding out T value, we first find the one tailed alpha value which is 0.05/2 which gives us 0.025.



v	$t_{0.005}$	$t_{0.01}$	$t_{0.025}$	$t_{0.05}$	$t_{0.10}$
1	63.66	31.82	12.71	6.31	3.08
2	9.92	6.92	4.30	2.92	1.89
3	5.84	4.54	3.18	2.35	1.64
4	4.60	3.75	2.78	2.13	1.53
5	4.03	3.36	2.57	2.02	1.48
6	3.71	3.14	2.45	1.94	1.44
7	3.50	3.00	2.36	1.90	1.42
8	3.36	2.90	2.31	1.86	1.40
9	3.25	2.82	2.26	1.83	1.38
10	3.17	2.76	2.23	1.81	1.37
11	3.11	2.72	2.20	1.80	1.36
12	3.06	2.68	2.18	1.78	1.36
13	3.01	2.65	2.16	1.77	1.35
14	2.98	2.62	2.14	1.76	1.34
15	2.95	2.60	2.13	1.75	1.34
16	2.92	2.58	2.12	1.75	1.34
17	2.90	2.57	2.11	1.74	1.33
18	2.88	2.55	2.10	1.73	1.33
19	2.86	2.54	2.09	1.73	1.33
20	2.84	2.53	2.09	1.72	1.32
21	2.83	2.52	2.08	1.72	1.32
22	2.82	2.51	2.07	1.72	1.32
23	2.81	2.50	2.07	1.71	1.32
24	2.80	2.49	2.06	1.71	1.32
25	2.79	2.48	2.06	1.71	1.32
26	2.78	2.48	2.06	1.71	1.32
27	2.77	2.47	2.05	1.70	1.31
28	2.76	2.47	2.05	1.70	1.31
29	2.76	2.46	2.04	1.70	1.31
30	2.75	2.46	2.04	1.70	1.31
40	2.70	2.42	2.02	1.68	1.30
60	2.66	2.39	2.00	1.67	1.30
120	2.62	2.36	1.98	1.66	1.29
∞	2.58	2.33	1.96	1.645	1.28

Source: Robert E. Shannon, *Systems Simulation: The Art and Science*, © 1975, p. 372.
Reprinted by permission of Prentice Hall, Upper Saddle River, NJ.

Figure 20. T-table [1, p. 544]

And since our original equation states: $t_{\frac{\alpha}{2}, r-1}$, we want to have r-1, which is 12 - 1 and gives us 11. With alpha as 0.025, we find our t value as 2.2.

6. Input-Output Validation Using Historical Data

Next, we want to find the CI of the full equation, which is our sample mean, plus or minus the t value we found as 2.2, multiplied by the sample standard deviation divided by square root of the number of replications we picked as 12:

$$CI = \bar{Y} \pm t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$$

Occupancies	Average Occupancies per Minute	+ $t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$ (+ ε, margin of error)	- $t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$ (- ε, margin of error)	Historical Average Occupancies
Component 1 for Workstation 1 Occupancy per Minute	0.3567605925	0.4639603045	0.2495608806	0.28
Component 1 for Workstation 2 Occupancy per Minute	0.5765898026	0.764059304	0.3891203011	0.41
Component 2 for Workstation 2 Occupancy per Minute	0.6654052846	0.8998824947	0.4309280744	0.6
Component 1 for Workstation 3 Occupancy per Minute	0.2345898808	0.378552396	0.09062736556	0.32
Component 3 for Workstation 3 Occupancy per Minute	1.132624022	1.356499467	0.9087485771	1.75

From the table above, using the equation of $\pm t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$ with regards to $\bar{Y}$, we can calculate the epsilon value that represents the margin of error or the width of the confidence interval around our sample mean, which is our average occupancies per minute per 12 replications.

We can show a mock calculation below for how we got the epsilon values for our table for the buffer for Component 1 for Workstation 1's Occupancy per minute:

$$CI = \bar{Y} \pm t_{\frac{\alpha}{2}, r-1} * (\frac{S}{\sqrt{r}})$$

$$CI = \bar{Y} \pm 2.447 * 0.02303035032 / \sqrt{12}$$

$$CI = \bar{Y} + 0.107199712$$

$$CI = \bar{Y} - 0.107199712$$

CI = 0.4639603045 for range (high)

CI = 0.2495608806 for range (low)

And from the calculations from the table above, we can see that for each of the buffer occupancies except for the very last buffer, they are within the ranges of the t-table derived confidence intervals of epsilon that we calculated.

Therefore, we cannot accept within 95% confidence interval, from alpha = 0.05, that the values from our model are all valid when compared with the historical average occupancies data.

And we will continue to investigate if there's an alternate routing strategy if we can find our facility's component 3's buffer results to match the historical data in the next milestone.

Bibliography

[1] J. Banks, *Discrete-event system simulation*, 2014th ed. Harlow: Pearson, 2014.