SYSC 4005
Term Project
Date: April 11, 2021
Group: Simulating Discretely

Authors:

Archit Bhatia - 101065674
Ross Matthew - 101079586
Jon Menard - 101086242

Table of Contents

Chapter 1: Deliverable 1

1.1 Problem Formulation

Using a complete simulation study our team is going to assess the throughput and probability that the inspectors will remain blocked and possibly improve the policy that inspector1 follows when delivering C1 components to the different workstation for the manufacturing facility based on the observed historical data provided by the inspectors and workstations.

1.2 Setting of Objectives and Overall Project Plan (see step 1)

Goals for February 14th - 28th

1. Determine the Project Goals and Objectives.
2. Conceptualize the project by determining the systems states and environment variables.
3. Create the project's UML.
4. Using java to implement the conceptual model a computation model will be created. Buffer, Inspector, and Workstation will each be implemented as their own classes. C1,C2 and C3 will be implemented as Enums. The states of each class will also be Enums.

Goals for February 28th - March 14:

1. Configure input and output files for running the simulation. Input files will determine the time it takes for the Inspector or Workstation to complete a task. The output file will record the states for each workstation, buffer, and inspector every second.
2. Begin preliminary analysis of the output data using histograms. Evaluate the distributions with Q-Q plots. Perform chi-square test for each distribution.

Goals for March 14th - 28th

1. Verify the model is working correctly and validate the simulation model was built right.
2. Run the simulation in production to gain final output values for analysis. The simulation will be replicated multiple times.
3. Determine the confidence interval for the data being analyzed.

Goals for March 28th - April 11

1. Describe alternative design solutions.
2. Compare the initial design with the alternative designs.
3. Write a conclusion and final report.

1.3 Conceptual Model

A conceptual model for this system is illustrated with the current states of the Inspectors (blocked, inspecting), Buffers (empty, partial, full), and WorkStations (waiting, working). These states completely describe the system and its status in real-time.

1.4 Model Translation

Java was chosen as the simulation language because of its object-oriented functionality making it easier to reuse functions for similar objects with the use of inheritance. Additionally, Java provides a multi-threaded functionality which may be beneficial for this simulation if we decide to implement each object to continuously run at the same time. Java also has useful library functions for I/O operations which will be utilized in this project. It is for these reasons Java is an appropriate language to use for this project.

From the conceptual model, the states for each class need to be determined, as such the following is proposed:

- The states for the inspector are: inspecting, blocked.
- The states for the buffers are: empty, partial, full.
- The states for the workstation are: waiting, working.

The model will be implemented in Java. Inspector, Buffer, and Workstation will each be their own class objects. Inspector and Workstation will implement an abstract class called

SimulationObject. SimulationObject will contain the necessary methods to read inputs from a .dat file. Inspector will have two subclasses called Inspector1 and Inspector2 that will implement the required inspector functionalities. Each inspector will have a different getComponent method that is designed based on the components the respective inspector will be inspecting (i.e. C1, C2, or C3). Similarly, WorkStation will also have two subclasses called Workstation2 and Workstation3. WorkStation provides all of the base functionalities of a workstation which are required for all workstations. Workstation will have a method called getComponents which will retrieve the required components from its available buffers. Workstation2 and Workstation3 will override the getComponents method, to retrieve the components which correspond to their requirements. Workstations and Inspectors will contain an ArrayList with the buffer they can access.

The buffers will store the component they hold, and how many it currently has. A main class called Simulator will be responsible for instantiating the class objects and setting up the simulation. It will also be responsible for polling each instantiated object every second to retrieve the objects' states. The Simulator will output the states into a .csv file for analysis later.

1.5 The UML diagram for the simulation is found below:

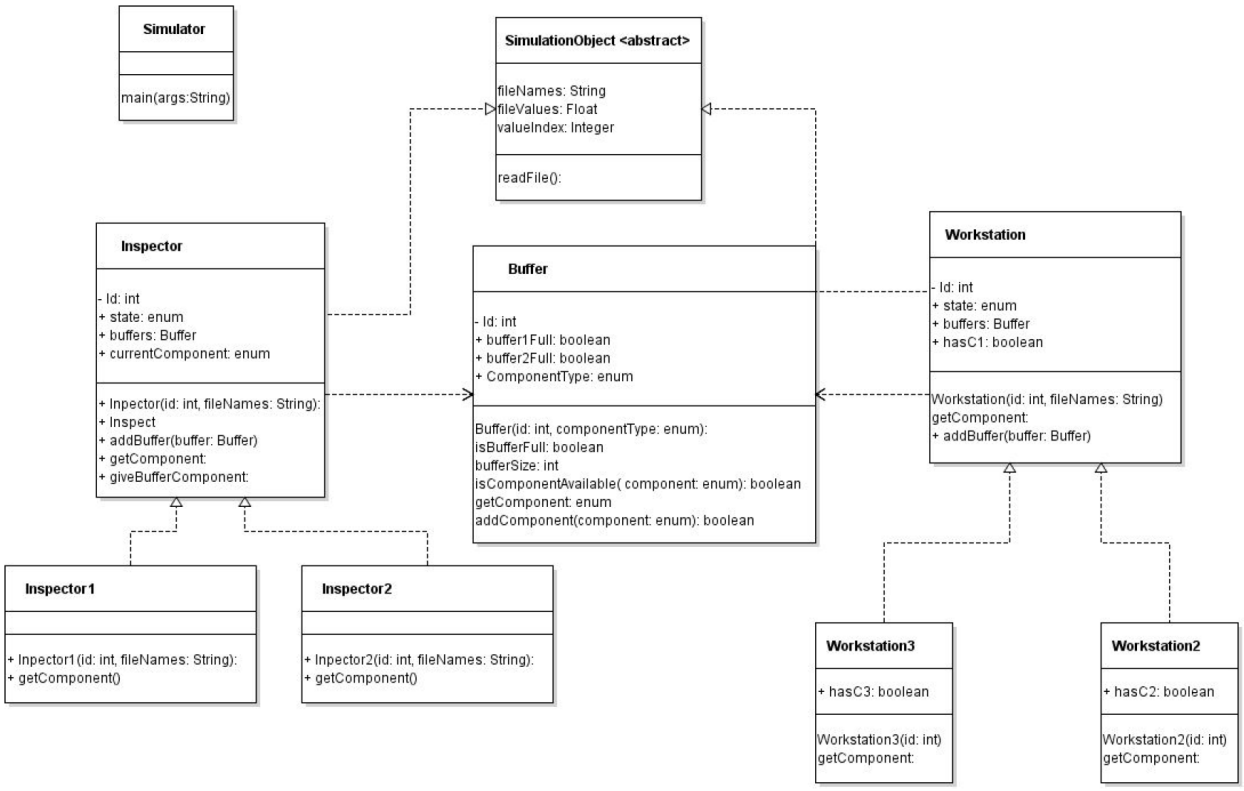


Figure 1.1: UML diagram for the simulation

The source code for the UML diagram can be found at: https://github.com/jonmenard007/SYSC-4005

Chapter 2: Deliverable 2

2.1 Data Collection and Input Modeling

The data collected and input modeling formatted are located in the appendix and will be summarized in this section.

Using Excel and Java's formula for random numbers, we generated 1000 different numbers for 6 different objects (the section below will explain how).

8		Inpsector 1	Inspector 2	Inspector 2 C2 C3 Prob	Workstation 1	Workstation 2	Workstation 3
9	Seed	243	98743453	1789343	392575424	24353	87654
10	X(0)	243	98743453	1789343	392575424	24353	87654
11	RN's						
12	1	0.02177	0.57025	0.29173	0.43020	0.18157	0.85216
13	2	0.62616	0.85116	0.08520	0.15324	0.25939	0.25099
14	3	0.41232	0.35513	0.09979	0.09890	0.26199	0.53152
15	4	0.90882	0.94388	0.04612	0.39023	0.27911	0.88304
16	5	0.83571	0.42960	0.94188	0.56299	0.19210	0.50305
1000	989	0.50124	0.04581	0.68694	0.18426	0.54000	0.68628
1001	990	0.40816	0.10233	0.08382	0.16610	0.11401	0.70442
1002	991	0.79384	0.07535	0.70324	0.04626	0.10244	0.92416
1003	992	0.73933	0.69730	0.39138	0.65530	0.82610	0.27446
1004	993	0.57961	0.80215	0.13311	0.27496	0.00228	0.88641
1005	994	0.32539	0.29422	0.02853	0.36471	0.56506	0.42530
1006	995	0.64760	0.74185	0.73827	0.75707	0.07937	0.28201
1007	996	0.17415	0.09622	0.02261	0.94668	0.87383	0.43626
1008	997	0.84938	0.03506	0.49920	0.59808	0.51730	0.55432
1009	998	0.10591	0.43143	0.24471	0.89730	0.76415	0.63350
1010	999	0.59901	0.44742	0.36510	0.87218	0.59341	0.25607
1011	1000	0.25349	0.75296	0.15969	0.30642	0.46648	0.26506

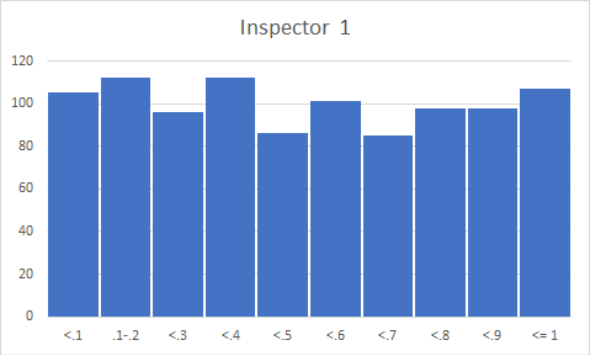
Figure 2.1: Beginning and end of the set of 1000 random numbers which were generated for each object.

The random numbers generated had a lower bound of 0 and an upper bound of 1. A k-value selected was 10 resulting in a bin size of 0.10.

Bin	<.1	<.2	<.3	<.4	<.5	<.6	<.7	<.8	<.9	<= 1	Total
Inspector 1	105	112	96	112	86	101	85	98	98	107	1000
Inspector 2	124	107	109	85	92	99	82	92	106	104	1000
Inspector 2 C2 C3 Prob	108	120	99	95	84	94	99	110	90	101	1000
Workstation 1	98	100	96	111	109	108	99	101	101	77	1000
Workstation 2	114	115	106	102	91	92	89	104	103	84	1000
Workstation 3	92	125	93	102	90	107	93	108	93	97	1000

Table 2.1: Distribution of the random generated values for each object

From the bins the following histograms were created for each LGC and the distributions were hypothesised.



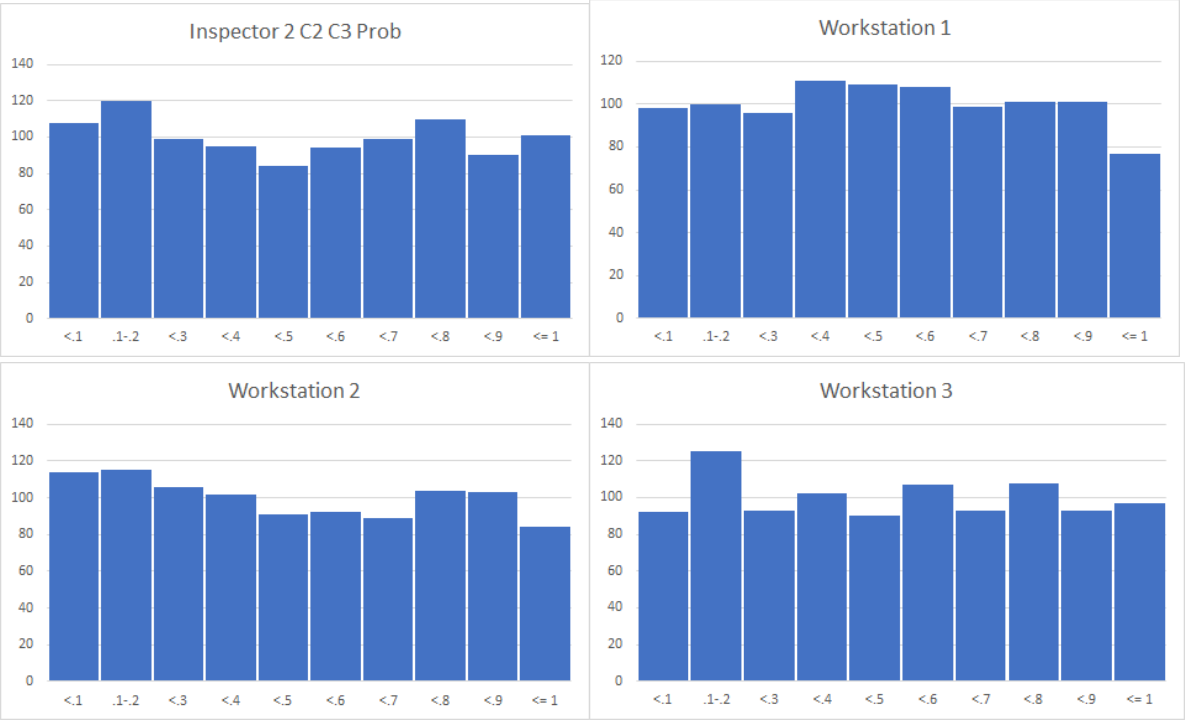


Figure 2.2: Histogram plots show the distribution of each of the objects randomly generated values

Each histogram does not only look similar to each other, but also replicates a uniform distribution. Each dataset was tested to determine the probability of distribution. The probability of each bin was calculated using the equation for a uniform distribution.

$$Uniform\ Distribution\ Equation\ = \frac{a - x}{b - a}$$

$$where\ a\ = 0\ and\ b\ = 1, represents\ the\ upper\ and\ lower\ bound\ values\ of\ the\ bins.$$

The probability of each bin was multiplied against the total sample number to determine the expected frequency.

$$E_i\ = n * p_i$$

$$where\ n\ = 1000\ and\ p_i\ is\ the\ probability\ for\ each\ bin.$$

To determine the calculated chi-squared value, the amount of occurrence for each bin was subtracted by the expected frequency and was then squared. This was then divided by the expected frequency as shown below.

$$X_0^2\ = (O_i\ - E_i)^2\ /\ E_i$$

The values for each bin were summed together to calculate the total chi-squared number. This was done for each of the 6 LGCs.

Chapter 3: Deliverable 3

3.1 Verification

To verify our model, our group first started by getting a fellow student to review our model. We walked them through the steps we took, and why we chose our design. They were able to agree without design as well as confirm our choices and did not recommend any changes.

3.2 Flow Diagram

The creation of flow diagrams for the major components in the system was used to verify that the functionalities were as we presumed and helped to give a better understanding of the processes occurring in the system.

3.2.1 Workstation Flow Diagram

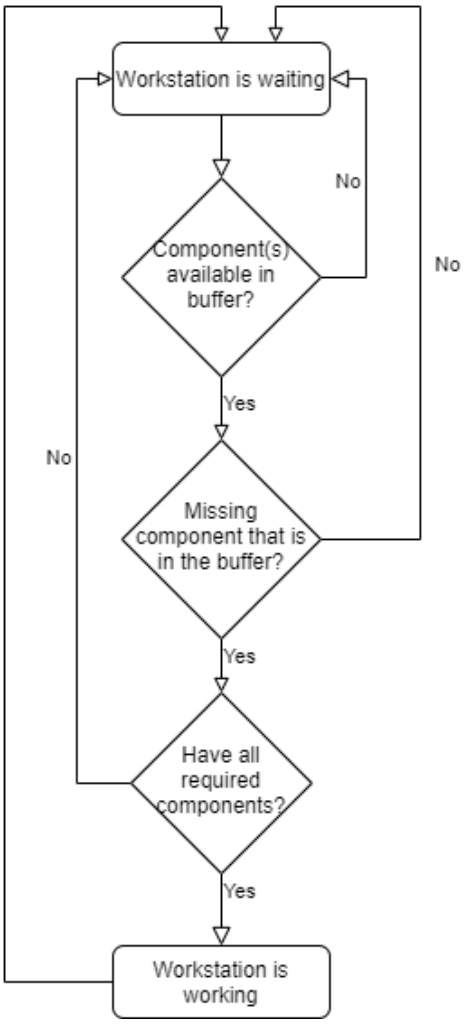


Figure 3.1: Flow diagram representation of the functionalities of the workstations in the system.

3.2.2 Inspector Flow Diagram

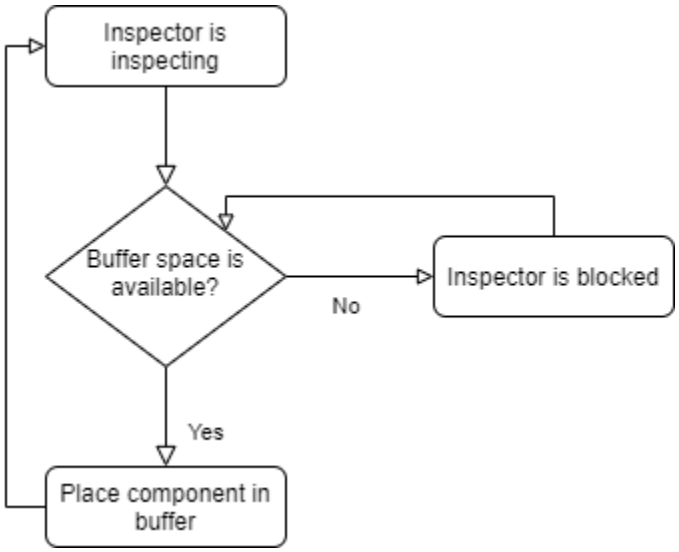


Figure 3.2: Flow diagram representation of the functionalities of the inspectors in the system.

3.3 Input Parameters

After going through the model design we began doing test runs to examine the model outputs and inputs. The inputs were written to a file for each simulation class (inspectors and workstations). The inputs, when compared to the samples provided, followed a similar distribution. When inspecting the outputs, we concluded the model was behaving as expected.

We went through the entire model one more time, properly documenting as we checked the components. This acted as our 5th and final check to verify the model was built correctly.

3.4 Validation

3.4.1 Face Validity

From the observed data that was produced we are able to confirm that all the inputs and outputs matched up as expected. By manually calculating what they should be for a few cycles, we confirmed the model was working correctly.

3.4.2 Validate Model Assumptions

Structural Assumption: We assumed the model ran for 2 hours since the data provided showed the total run times were between 1 hour and 2 hours. Therefore, each of our trials ran for 7200 seconds equivalent to 2 hours.

Data Assumption: By analyzing the sample data provided in the .dat files, we found the mean time for the inputs of each simulation object (Inspector1, Inspector2, Workstation1, Ect.). The average for the inputs were the following:

	Insp 1	Insp 2 C2	Insp 2 C3	WS1	WS 2	WS 3	Total
Mean	10.357	15.53690	20.63275	4.6044166	11.092606	8.79558	11.8367
Lambda	0.0965	0.064362	0.048466	0.2171827	0.0901501	0.1136934	0.084483

Table 3.1: Average input mean and lambda for the sample data

Additionally, when plotting the provided inputs, we discovered they all followed an exponential distribution.

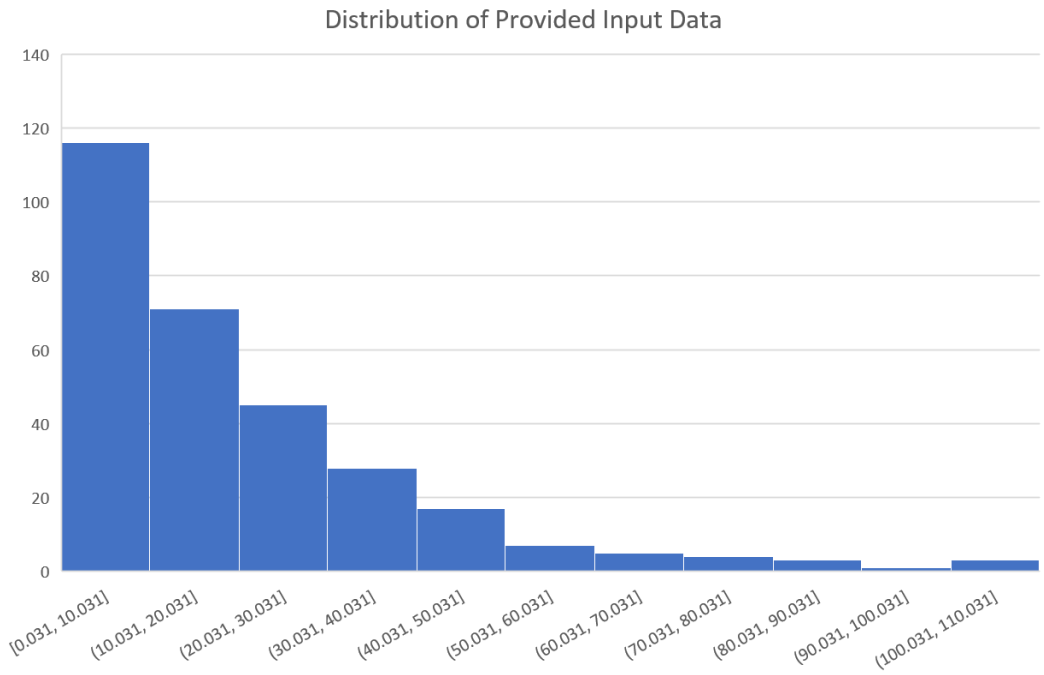


Figure 3.3: Distribution plot for the provided input data

Using the mean for each provided input data, and Equation 1 for the distribution of an exponential equation. We created the proper formulas for generating the inputs for each simulation model.

$$x_i \simeq \frac{-1}{\lambda} ln(r_i) \qquad (1)$$

$\lambda$is equal to 1/the mean
$r_i$is a generated random value

This formula was used every time a simulation object needed a new time for either inspecting or working.

```
/**
 * Gets the next random number.
 *
 * @return the random number
 */
public float getRandomNumber() {
    float randomTime = this.randomNumber.nextFloat() ;
    randomTime = (float) ((-1 / lambda) * (Math.log(randomTime)));
    return randomTime;
}
```

Figure 3.4: Code used to implement the random number equation

To validate the inputs we generated were correct we ran 5 trials and gathered data from all the inputs generated. Shown in Table 2, the mean value of the generated inputs for each simulation object.

	Insp 1	Insp 2 C2	Insp 2 C3	WS1	WS 2	WS 3
1	10.27527	14.54219	16.83981	4.703893764	10.28869	10.0869
2	10.51939	16.73446	20.24589	4.591713762	11.41955	9.840753
3	10.3907	20.36956	22.96274	4.604693053	8.676378	12.38443
4	10.50459	14.76534	21.02882	4.145667131	11.20464	7.966175
5	10.21949	11.50723	18.98501	4.544909726	9.677508	8.101089

Table 3.2: Shows the trials ran to validate the inputs and the mean values that were generated for each simulation

Using a 2-tailed T-test, and a confidence interval of 95%, we compared the generated inputs to the provided ones.

2-tailed T-test Inspector 1 Example

Inspector 1			
Replication	Rn Generated	Total Time	Average Rn value
1	1430	7200	10.27526808
2	1410	7200	10.51938616
3	1309	7200	10.39070304
4	1339	7200	10.50458511
5	1406	7200	10.21948792
Sample Mean			10.38188606
Sample Variance			0.133958598

Table 3.3: Comparison of generated inputs to ones provided using 2-Tailed T-test for inspector 1

H_0	E(Y2) =	10.35791	95 Confidence Interval
H_1	E(Y2) !=	10.35791	
$\underline{y}$	10.38188606		$t_0 = (\underline{y} - \underline{x})/(S/\sqrt{N})$

S	0.133958598		0.400213978
$t_{0.025,4}$	2.776		
0.400213978	<	2.776	Fail to reject H_0

Table 3.4: Hypothesis testing to verify validation input generation

For all 5 simulation objects, we failed to reject the H_0 hypothesis proving our inputs generation was valid.

**Since there is no output data available we are not able to do the error evaluation tests

3.5 Production Runs and Analysis

As mentioned above, each of our simulations ran for a simulated time of 7200 seconds (2 hours).

The confidence interval we will be using is 95%, as mentioned in the project outline.

We ran each simulation 5 times and gathered the results from each. To analyze the data, we compared the time each simulation object was blocked in each trial.

$t_{0.025,4}$	2.776

Time Blocked

	Inspector 1	Inspector 2	Workstation 1	Workstation 2	Workstation 3
1	0	5826.24	1731.09	6587.3	6751.25
2	0	5855.25	1669.18	6835.95	6804.94
3	0	5582.41	1757.13	6790.75	6734.89
4	0	5996.42	1583.87	6806.35	6882.15

5	0	6120.21	1682.54	6723.52	6915.09
W_Q	0	5876.106	1684.762	6748.774	6817.664
SE	0	202.0155641	66.72769792	99.23236937	79.17468712
$t_{0.025,4} \times \frac{SE}{\sqrt{N}}$	0	250.7952404	82.84009757	123.1935076	98.29259828
	0 ± 0	5876.106 $\pm$250.7952404	1684.762 $\pm$ 82.84009757	6748.774 $\pm$ 123.1935076	6817.664 $\pm$ 98.29259828

Table 3.5: Average time blocked for each simulation run

Since all 5 simulation objects have a plus or minus range less than 5%, we can say with 95% certainty that the number of simulation replications we have chosen provides adequate data for proper analysis.

Chapter 4: Deliverable 4

4.1 An Alternative Operating Policy

The alternative operating policy our team proposes is to change how Inspector 1 delivers component C1 to its buffers. Currently, C1 is distributed to the buffer with the least amount of components and in the case of a tie, buffer 1 has the highest priority, and buffer 3 has the lowest. For our alternative operating policy, we want to change Inspector 1 to have the following distribution plan (**Figure 4.1**).

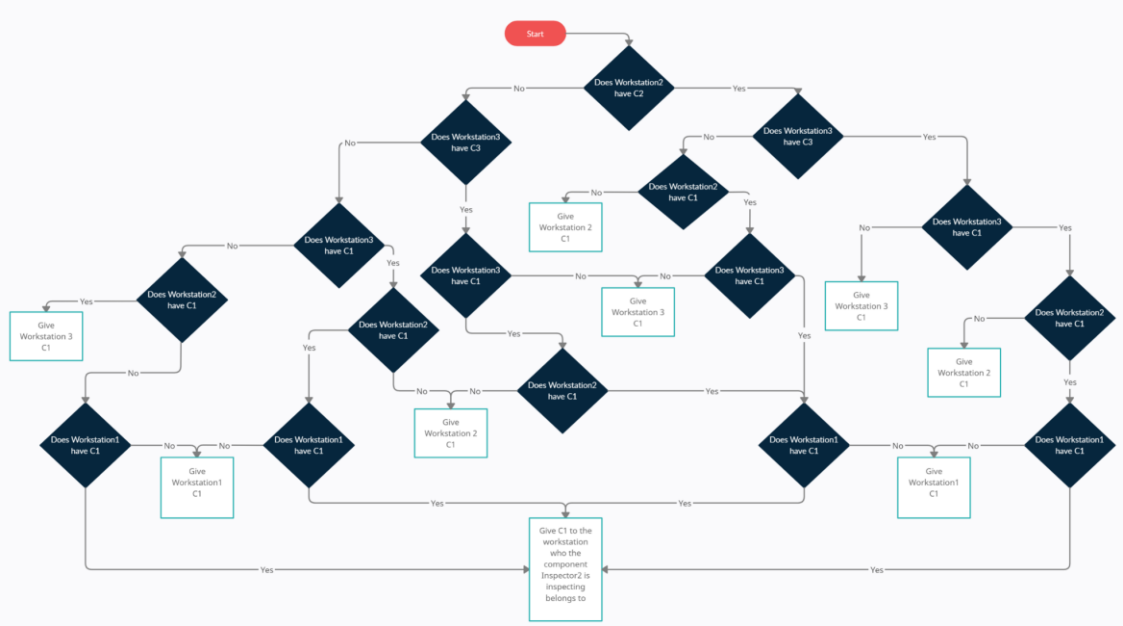


Figure 4.1: Flowchart showing the logic used in the newly designed algorithm

Inspector 1 will first check to see if either Workstation 2 or Workstation 3 has received the other part needed to construct their component from Inspector 2. If they do, and they are waiting to receive a C1 then Inspector 1 will give a C1 component so the product can be assembled. Workstation 2 will take priority over Workstation 3 if both are found to have their required component that is not C1. In the case, both Workstation 2 and 3 are both missing the other component (C2/C3) but both have C1, then Workstation 1 will get the C1 component from Inspector 1. If all three workstations already have a C1 component, then Inspector 1 will check to see what component Inspector 2 is inspecting, and give the C1 component to the corresponding workstation that matches the type. If that workstation is full then Workstation 1 will receive the C1 component.

We have designed our algorithm to maximize throughput and minimize the probability an inspector is blocked. In the original algorithm, Inspector 2 was blocked over 70% of the time, while Inspector 1 was never blocked. Our algorithm focuses on ensuring Inspector 2 is minimally blocked and is utilized as close to 100% of the time as possible. By achieving higher inspector utilization, the product throughput will increase as well.

The alternative operating policy is implemented in our Java codebase in Inspector 1's giveBufferComponent() method. The function checks Workstations 2 and 3 to determine if they have a C1 component, and/or their other required component. The function also checks to see if Workstation 1 has any C1 components. With those results, the function iterates through Figure 4.1's flowchart using if-else statements.

4.2 Comparing Alternative Operating Policies

To compare the two operating policies, 25 trials were run for each policy.The probability of Inspector 1 and Inspector 2 being blocked, as well as the throughput of items created per second were used to measure the performance of each policy. Running 25 trials resulted in the following metrics.

Original	Insp1 Blocked Probability	Insp2 Blocked Probability	Items Made	Throughput
Y1	0	0.783571	693.8	0.096464328
S^2	0	0.001558	503.583	9.80226E-06

Alternative	Insp1 Blocked Probability	Insp2 Blocked Probability	Items Made	Throughput
Y2	0.031772452	0.01853771	1194.36	0.165981138
S^2	3.49122E-05	0.00021385	1159.99	2.25562E-05

By comparing the two policies using independent sampling with equal variances, we resulted in the following confidence intervals.

	Insp1 Blocked Probability	Insp2 Blocked Probability	Items Made	Throughput
T,0.025,48	2.011			
Y1-Y2	-0.031772452	0.7650	-500.56	-0.06951681
S^2_p	1.74561E-05	0.00088771	831.7867	1.61792E-05
SE(Y1-Y2)	4.93733E-06	0.00025108	235.2648	4.57618E-06
±	9.92898E-06	0.00050493	473.1175	9.2027E-06
CI	-0.03±0.00001	0.77±0.0005	-501±473	-0.070±0.00001
	Fully < 0	Fully > 0	Fully < 0	Fully < 0
Conclusion	With 95% confidence, Inspector1 blocked probability is greater in the alternative policy	With 95% confidence, Inspector2 blocked probability is smaller in the alternative policy	With 95% confidence, items made is greater in the alternative policy	With 95% confidence, the item throughput is greater in the alternative policy

By analyzing this chart, it shows that with 95% confidence the alternative policy has a higher probability of being blocked for Inspector 1, a low probability of being blocked for Inspector 2, a higher number of items made, and higher throughput of items per second. While the Inspector 1 probability of being blocked is higher, it is only a difference of 0.03%, which is negligible

4.3 Discussing Results

In the original policy being used the probability of Inspector 1 and 2 being blocked were 0 and 0.783571 respectively. The alternative policy which our group designed gives Inspector 1 and 2 a 0.031772452 and 0.01853771 probability of being blocked, respectively. While Inspector 1's probability of being in a blocked state does increase slightly it is offset by the large decrease that Inspector 2's probability of being blocked experiences. By having low probability values for each of the Inspectors being in a blocked state the throughput of the system is greatly increased as more products are able to be created by the workstations. The throughput value nearly doubles between policies from 0.096464328 to 0.165981138. This shows that the policy changes which were made had a large positive impact on the performance of the system.

4.4 Conclusion

Our group believes that implementing the new policy for the facility would be a positive change and result in increased production efficiency, as shown when comparing policies in **Section 4.2**. When maximizing the efficiency of a process the throughput and utilization of components in the system need to be as close to 1 (constantly working) as possible and this is achieved with our alternative policy.

Milestone 4 Appendix

ORIG	I1 BT	I2 BT	TT	I1 BP	I2 BP	WS1	WS2	WS3	TM	TM/TT
1	0	5296.6	7190	0	0.73667374	617	40	41	698	0.097080087
2	0	5448.1	7188	0	0.75798382	627	40	50	717	0.099754301
3	0	5180.5	7190	0	0.72056025	608	44	48	700	0.097362857
4	0	5904.1	7193	0	0.82085819	621	46	29	696	0.096766524
5	0	5766.2	7200	0	0.80090255	640	40	27	707	0.098200036
6	0	5621.8	7197	0	0.78107941	595	53	37	685	0.095172596
7	0	5159.2	7188	0	0.71773344	652	49	50	751	0.104477217
8	0	5572.2	7198	0	0.77414949	584	47	33	664	0.092250794
9	0	5790.5	7181	0	0.80637664	609	26	31	666	0.092746035
10	0	5605.9	7199	0	0.77866057	614	49	37	700	0.097231
11	0	6230.4	7197	0	0.8656474	609	32	25	666	0.092534315
12	0	5938.4	7187	0	0.82626602	622	40	32	694	0.096562164
13	0	6044.5	7193	0	0.84028932	577	42	28	647	0.08994456
14	0	5552.4	7183	0	0.77294824	620	46	51	717	0.099814293
15	0	5766.6	7195	0	0.80147519	623	45	41	709	0.098540379
16	0	5917.4	7191	0	0.82293386	619	41	34	694	0.096515029
17	0	5587.5	7197	0	0.77637332	597	43	44	684	0.09504026
18	0	5708.3	7198	0	0.79308879	636	42	32	710	0.098644267
19	0	5380.8	7187	0	0.7486872	618	44	50	712	0.099068588
20	0	5158.1	7199	0	0.71654488	616	56	40	712	0.098908121
21	0	5927.3	7187	0	0.82466431	605	32	33	670	0.09321778
22	0	5738.4	7192	0	0.79791095	584	37	43	664	0.09232763
23	0	5284.8	7197	0	0.73427943	600	37	53	690	0.095869817
24	0	5681.3	7184	0	0.79084804	632	40	28	700	0.097441194
25	0	5632.6	7198	0	0.78253005	609	42	41	692	0.09613835
Y1	0	5635.8	7192	0	0.78357861	613.36	42.1	38.32	693.8	0.096464328
S^2	0	80822	31.5	0	0.00156158	315.74	42.4	73.98	503.5833	9.80226E-06

ALT	I1 BT	I2 BT	TT	I1 BP	I2 BP	WS1	WS2	WS3	TM	TM/TT
1	224.2	222.9	7194	0.031169246	0.03098437	824	206	179	1209	0.168057882
2	271.7	215.64	7198	0.037741275	0.02995741	773	199	178	1150	0.159761719
3	270.2	112.23	7200	0.037535921	0.015588	812	190	203	1205	0.167366458
4	248.3	41.67	7199	0.034483046	0.00578815	868	189	195	1252	0.173908454
5	188.8	179.15	7198	0.026232943	0.02488948	757	193	183	1133	0.157408771
6	210.8	100.31	7200	0.029276423	0.01393262	806	193	187	1186	0.16473023
7	167.4	168.45	7198	0.023249643	0.02340107	787	191	224	1202	0.166981783
8	215.2	62.8	7198	0.029895389	0.00872452	859	159	199	1217	0.169072394
9	298.5	109.58	7200	0.041460176	0.01522012	833	167	200	1200	0.166674074
10	219	467.35	7198	0.030422714	0.06492559	755	201	200	1156	0.160594812
11	221.1	13.15	7197	0.030722846	0.00182709	783	188	207	1178	0.163673625
12	223.6	294.78	7199	0.03105289	0.04094548	736	189	200	1125	0.156264541
13	150.7	45.78	7187	0.020968824	0.00636996	776	197	180	1153	0.160431677
14	189.4	3.31	7182	0.026372106	0.00046086	816	175	192	1183	0.164712533
15	274.9	77.03	7198	0.038187315	0.01070167	798	189	207	1194	0.165880794
16	268.7	27.82	7198	0.037337653	0.0038652	817	181	188	1186	0.164778064
17	257.1	23.81	7192	0.035740355	0.00331042	864	186	182	1232	0.171291205
18	155.2	240.13	7198	0.0215537	0.03335927	787	222	192	1201	0.16684495
19	192.4	168.12	7199	0.026728044	0.02335266	830	197	200	1227	0.170436077
20	262	95.07	7199	0.03639033	0.01320519	753	184	211	1148	0.159456847
21	245.1	226.95	7182	0.034126366	0.03159926	819	217	193	1229	0.171119153
22	190	64.89	7200	0.02638794	0.00901265	813	216	197	1226	0.170280616
23	246	109.08	7187	0.034226627	0.01517844	817	192	207	1216	0.169205915
24	308.3	156.78	7193	0.042861115	0.02179619	820	203	207	1230	0.170999583
25	217.3	108.33	7199	0.030188417	0.01504698	800	205	216	1221	0.169596289
Y2	228.6	133.4	7196	0.031772452	0.01853771	804.12	193	197.1	1194.36	0.165981138
S^2	1809	11077	30.4	3.49122E-05	0.00021385	1183.4	209	142	1159.99	2.25562E-05

Milestone 3-1 Appendices