SYSC 5001W: Project deliverable 2

Qiguang Chu
300042722

University of Ottawa — April 1, 2020

1 Model Verification and Validation

The goal of the validation is to produce a model that represents true system behavior closely enough for the model to be used as a substitute for the actual system.

1.1 Verification of Simulation Models

Verification is concerned with building the model correctly. It proceeds by the comparison of the conceptual model to the computer representation what implements that conception.

1.1.1 Flow Diagram

I made a flow diagram that includes each logically possible action a system can take when an event occurs.

CLOCK = Simulation Clock

ETYPE = Event Type

NCUST = Number of Customer in the WorkStation

STATUS = Status of WorkStation (1-busy 0-idle)

CLOCK	ETYPE	NCUST	STATUS
0	Start	0	0
7.46	Arrival	1	0
8.31	Output	0	0
22.65	Arrival	1	0
34.56	Output	0	0

Figure 1: Simulation Trace

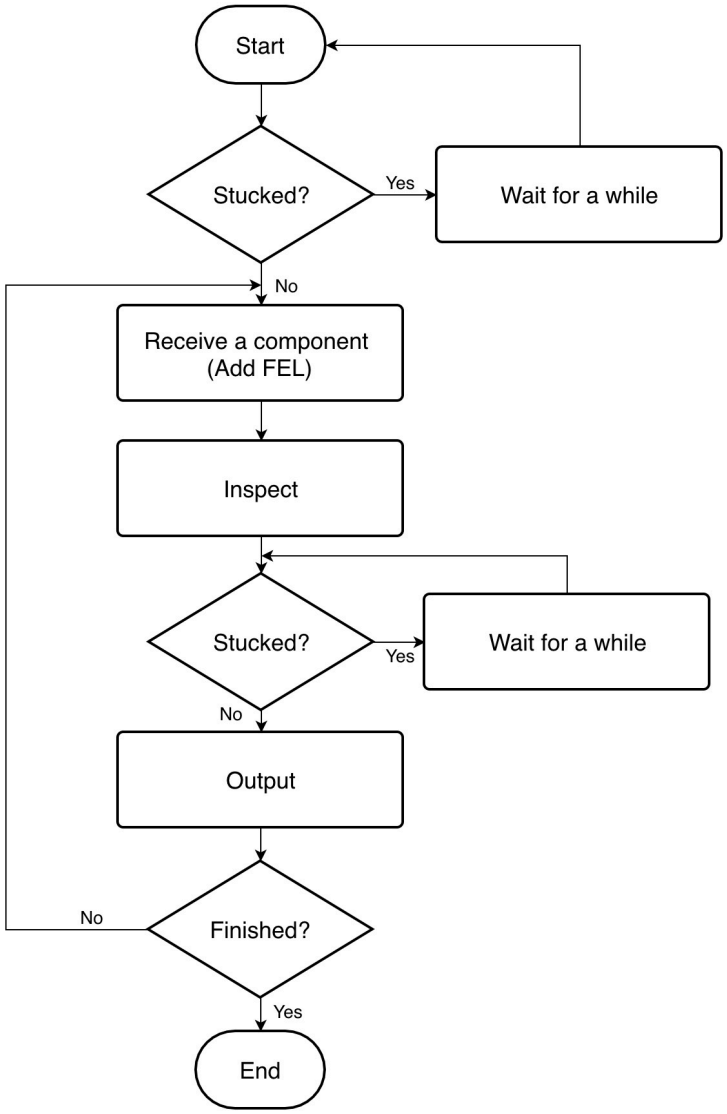

Figure 2: Work flow of Inspectors

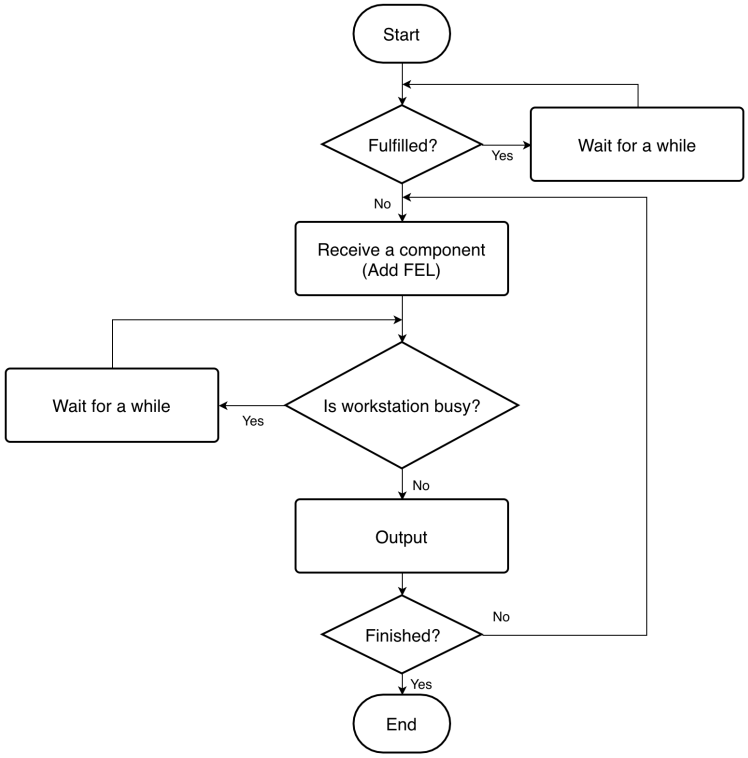

Figure 3: Work flow of Buffer

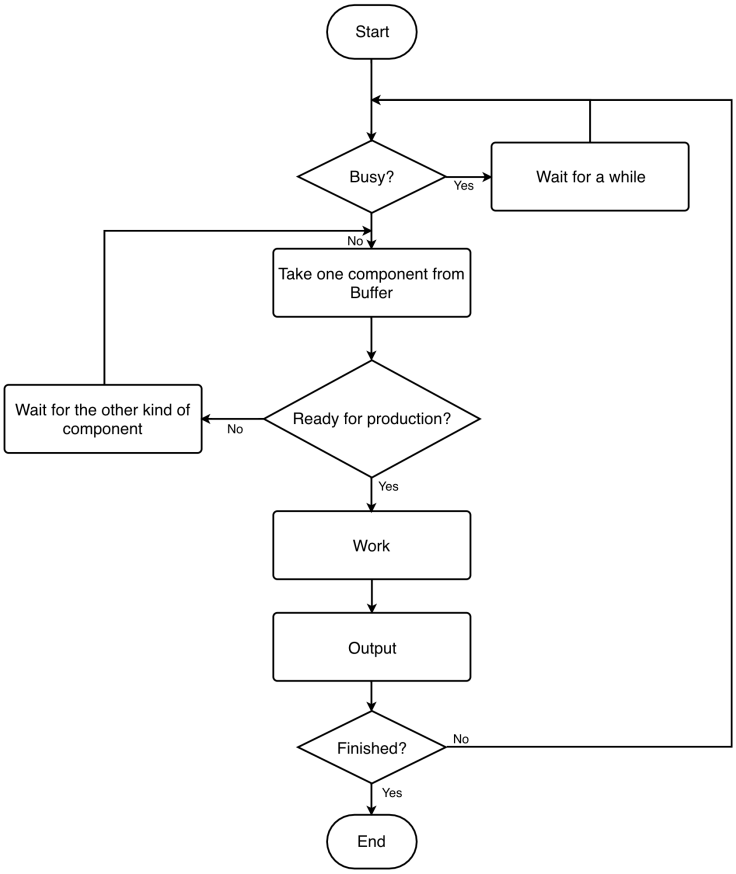

Figure 4: Work flow of Workstation

This simulation trace table indicates how the Workstation 1 work in this process. It just wait for Component 1, deal with it and then output. If the station is busy and a new one comes, the new arrival will stay at the buffer for now.

1.1.2 Model Output

The output of model is reasonable because it tracks each component of the whole process and the total time it costs is similar to the real time cost. For each object in the process, I can extract what it is doing at a certain point in time.

1.1.3 Sequence of dealing events

My code follows the rule that proceed events as time goes. The primary key of every event happened in this process is the clock. When we know that something is certain to happen, we put it in the FEL. Then we sort it according to the time of the occurrence of these things. The earlier event comes out first and latter event comes out the last.

1.1.4 Little's Law

Little's Law works for any blackbox, where the blackbox can be a complex system, or any subunit such as a single waiting line or a single server, a waiting line plus a sever, etc.

For each queue system in this project, we do a Little's law verification.

$$\hat{L} = \hat{\lambda}\hat{w} \tag{1}$$

$\hat{L}$ means average number of components in the queue. $\hat{\lambda}$ means the arrival rate of components. $\hat{w}$ means average working time for each component.

For Inspector system, consider the supply of the component for each inspector is sufficient, which means theres always a component waiting outside and there's always existing one component in each inspector. It's easy to calculate that the average dealing time for each inspector but things are little different here. If the corresponding workstation has a full buffer already and can't hold more components, the component must stay in the inspector and the inspector stop receiving new components. Here we have $L = \hat{L} = 1, \hat{w} = 10.36, \hat{\lambda} = 0.10$. It follows Little's law. By checking to see if these averages are in range, we can verify quickly if the system is wrong before we even look at any other part of the simulation. Here we have the number of each component in each buffer of the specific workstation.

The x axle on the diagram shows the time line of the whole process of production, while the y axle indicates how many kind of this components in the buffer at this time. I also draw a red line to denote the average number of gears in buffer in this period. It has to be mentioned that I ignored the case that the workstation receive the component and then immediately use it to produce.

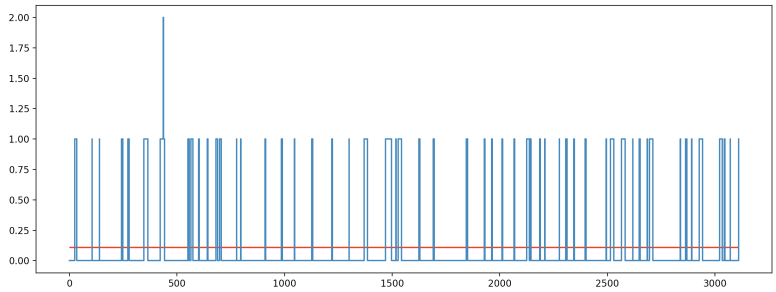

Figure 5: The number of components in Component 1 buffer of Workstation 1

That is because the time component stay in buffer is extremely slow and we can see it as no change in the buffer. By the way, in order to get rid of error, we must promise all buffers of workstations are empty when the simulation ends, which is as same as the beginning. The time point in this simulation is T = 2713.501

Component 1 buffer of Workstation 1:
Verification: $L = 0.109, \lambda = 0.037, w = 2.979, T = 2713.501$

$$\lambda w = 0.037 * 2.979 = 0.109 = L \tag{2}$$

This queue pass the Little's law verification.

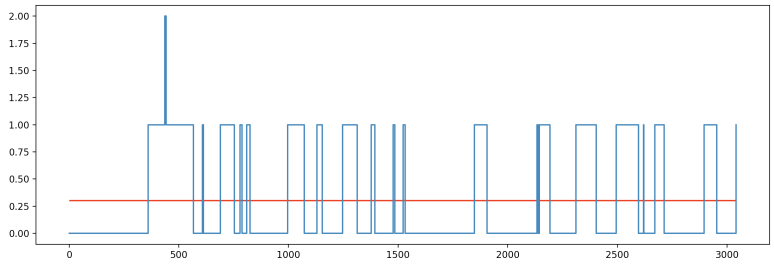

Figure 6: The number of components in Component 1 buffer of Workstation 2

Component 1 buffer of Workstation 2:
Verification: $L = 0.301, \lambda = 0.014, w = 21.773, T = 2713.501$

$$\lambda w = 0.02 * 28.034 = 0.301 = L \tag{3}$$

This queue pass the Little's law verification.
Component 2 buffer of Workstation 2:
Verification: $L = 0.751, \lambda = 0.020, w = 37.372, T = 2713.501$

$$\lambda w = 0.020 * 37.372 = 0.751 = L \tag{4}$$

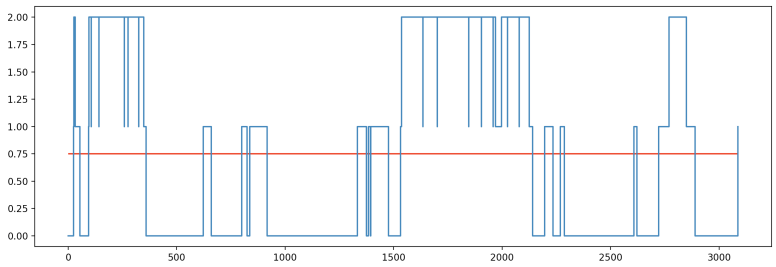

Figure 7: The number of components in Component 2 buffer of Workstation 2

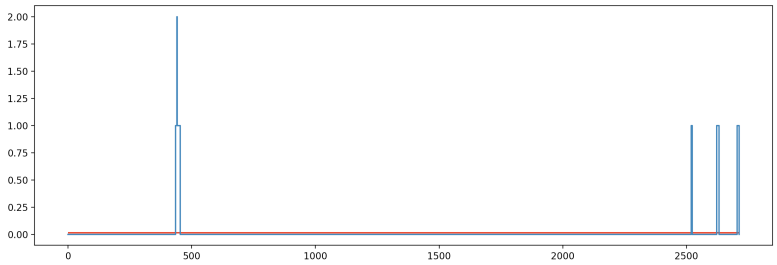

Figure 8: The number of components in Component 1 buffer of Workstation 3

This queue pass the Little's law verification.
Component 1 buffer of Workstation 3:
Verification: $L = 0.015, \lambda = 0.004, w = 3.676, T = 2713.501$

$$\lambda w = 0.004 * 3.676 = 0.015 = L \tag{5}$$

This queue pass the Little's law verification.
Component 3 buffer of Workstation 3:
Verification: $L = 1.556, \lambda = 0.024, w = 64.972, T = 2713.501$

$$\lambda w = 0.024 * 64.972 = 1.556 = L \tag{6}$$

This queue pass the Little's law verification.
So, all queues of this model pass the Little's law verification.

1.2 Validation

Validation Process
Step 1: Build a model that has high face validity
Step 2: Validate model assumptions
Step 3: Validate input-output transformations

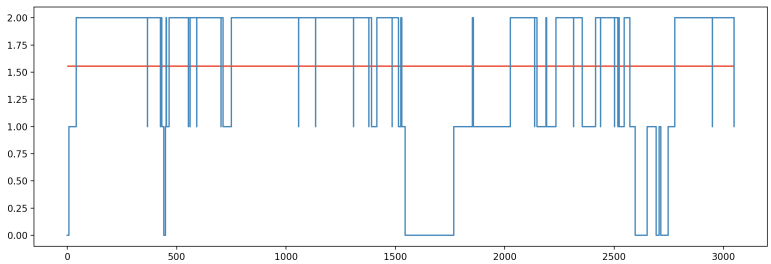

Figure 9: The number of components in Component 3 buffer of Workstation 3

1.2.1 Face Validity

The model appears reasonable. Potential users and knowledgeable people can also evaluate model output for reasonableness and can aid in identifying model deficiencies. The model is able to reflect the reality queuing systems that have events like arrival, output and block. Inspectors distribute components to correct workstations while each station prepares a buffer for each component they receive. Each buffer can temporarily store 2 components at most and it would close itself after that. The workstation gets all components it needs to produce, it takes out one component from its buffer and starts working. This model can deal with what the real scenario describes and it appears reasonable, which means I reckon it passes the face validity.

1.2.2 Model Assumptions

The model assumption consists of two parts: structure assumption and data assumption. About the data assumption, we have mentioned that in the last deliverable that the data we simulate comes from the data points generator, which is validated as that can produce similar data points to reality recorded samples. Now that the generator passes Chi-square validation, we can use it to generate experimental data for us and these points are representative. For the structure part, I used Python language to build a system as what the question depicts. Each inspector and workstation is an object and python is an object-orientated program language so it's easy to achieve these functions.

1.2.3 Input-Output Transformation

2 Production Runs and Analysis

2.1 The Length of Time in Simulation

The length of time we record in the simulation is worth to talk. Although in workstation 1 the workstation many continuously working because it just need one component, the other two stations may stop working until the simulation

ends. That is because the other two workstations needs two different kinds of gears and C2 and C3 are both provided by inspector 2 and also need C1 to start working. Of course, we can imagine in this experiment the number of C1 is not enough for both three workstations to finish their products, which means the last two components stay in the buffer for much long time until the simulation ends. So we have to arrange a good time length for them, which is a time point that no components in all workstations' buffers. That would be helpful for us to calculate the output data points. The time point is $T = 2713.501$

2.2 Measures of Performance

Obviously, it's a discrete time data model, which we assign $[Y_1, Y_2, ..., Y_n]$ to output. The point estimation for discrete time data is:

$$\hat{\theta} = \frac{1}{n} \sum_{i=1}^{n} Y_i \tag{7}$$

We hope that $E(\hat{\theta}) = \theta$ The output of the system is gettable and we have confidence to evaluate it.

To make sure that what I got is in static state, I used moving average and batch average to make sure this state is representative. I made experiment 10 times longer than the transit state to make sure the state is the static state.

Based on output distributions, suppose an output Y follows the normal distribution with mean θ, variance δ^2 (both unknown). Let Y_i be the average within the i^{th} replication of the simulation (its mathematical expectation is θ). Sample variance across R replications:

$$S^2 = \frac{1}{R-1} \sum_{i=1}^{R} (Y_i - \bar{Y}) \tag{8}$$

In this experiment, I did 10 times replications to estimate the confidence interval of the outputs. As the formulation, the confidence interval would be the normally distributed.

$$\bar{Y} \pm t_{\alpha/2,R-1} \frac{S}{\sqrt{R}} \tag{9}$$

2.3 Required Number of Replications

We need to perform several independent replications of the simulation in order to obtain usable results.

In the project requirements, we stated that there must be a 0.95 confidence interval, and its width should not exceed 0.2 of the estimated value. This will perform 10 copies and then calculate each confidence interval based on the data from each copy.