SYSC 5001W: Project deliverable 2

Qiguang Chu
300042722

University of Ottawa — March 29, 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)

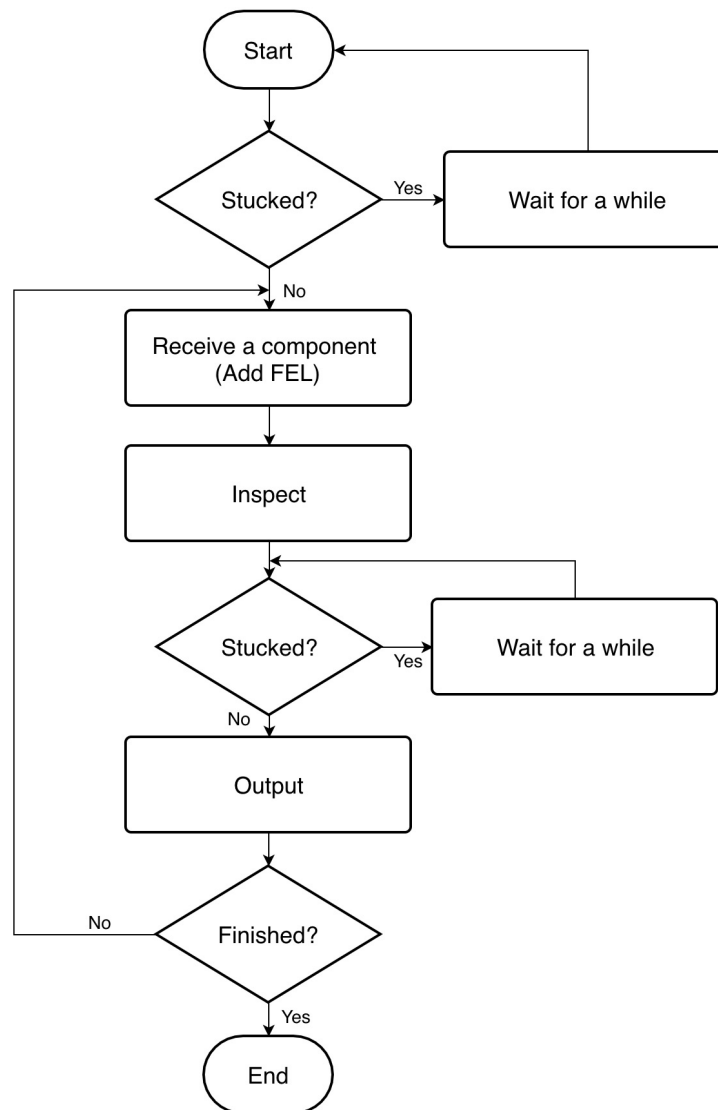| CLOCK	ETYPE	NCUST	STATUS
0	Start	0	0
7.46	Arrival	1	0
8.31	Output	0	0
22.65	Arrival	1	0
34.56	Output	0	0

Figure 1: Simulation Trace

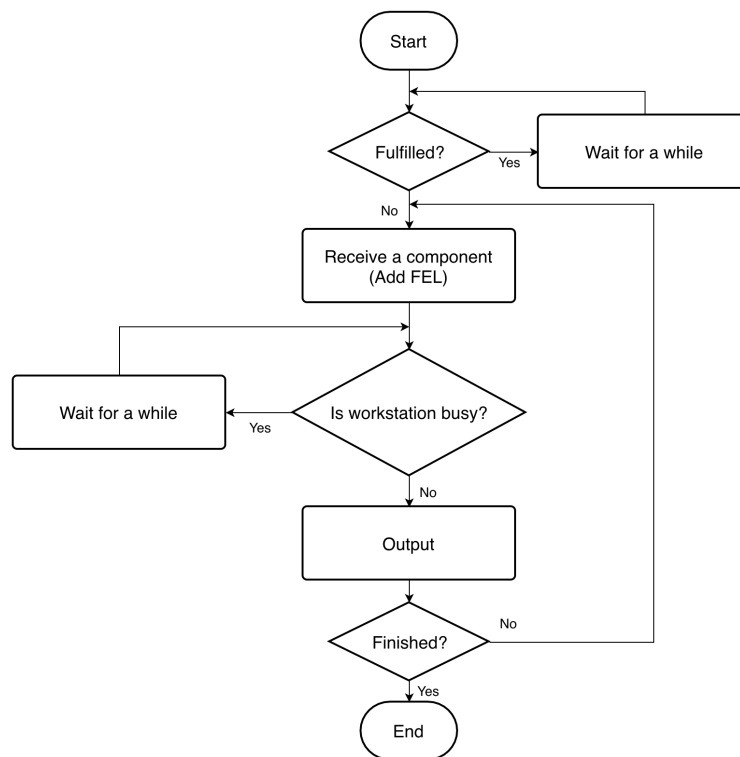Figure 2: Work flow of Inspectors
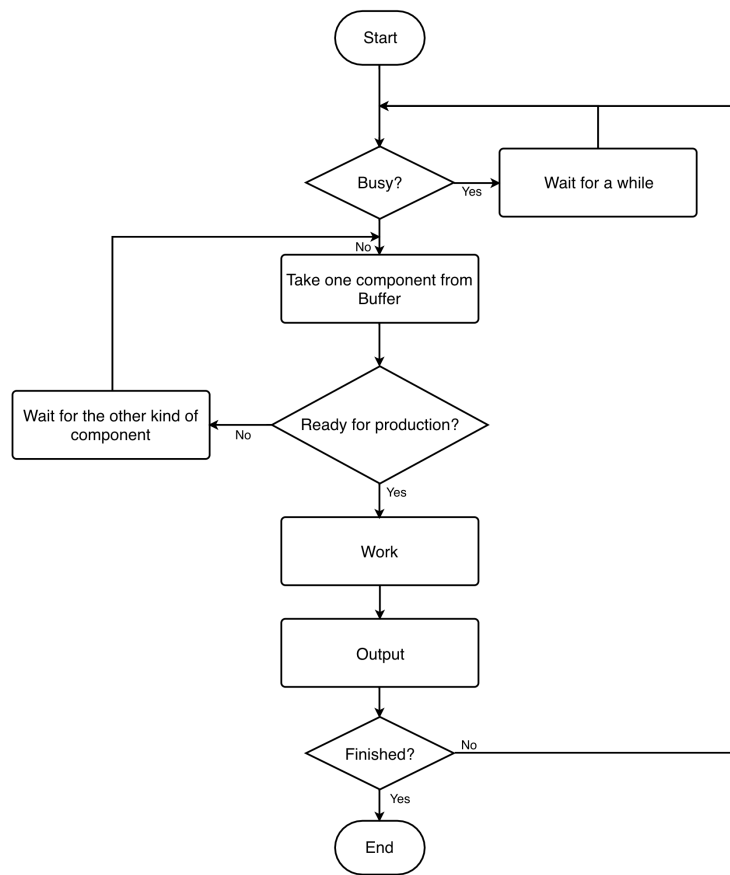
Figure 3: Work flow of Buffer

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 mean line to denote the average number of gears in buffer in this period.

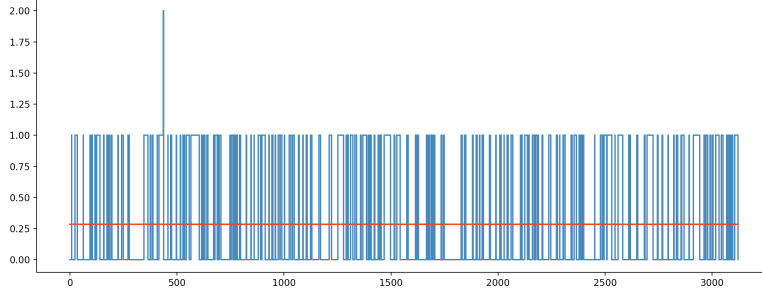Component 1 buffer of Workstation 1:

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

Verification: $L = 0.286, \lambda = 0.06, w = 4.723, T = 3135.49, N = 189$

$$\lambda w = 0.06 * 4.723 = 0.283 \approx 0.286 = L \tag{2}$$

A little error is acceptable because the formulation is under the condition that $T \to \infty, N \to \infty$. This queue pass the Little's law verification.
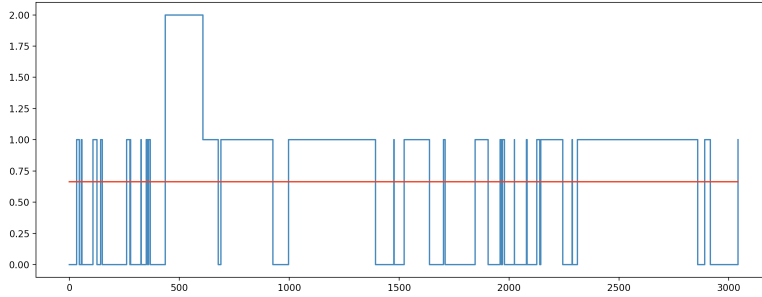

Figure 6: The number of components in Component 1 buffer of Workstation 2

Component 1 buffer of Workstation 2:
Verification: $L = 0.664, \lambda = 0.02, w = 28.034, T = 3135.49, N = 36$

$$\lambda w = 0.02 * 28.034 = 0.561 \approx 0.664 = L \tag{3}$$

This queue pass the Little's law verification.
Component 2 buffer of Workstation 2:
Verification: $L = 1.037, \lambda = 0.02, w = 49.413, T = 3135.49, N = 33$

$$\lambda w = 0.02 * 49.413 = 0.988 \approx 1.037 = L \tag{4}$$

This queue pass the Little's law verification.
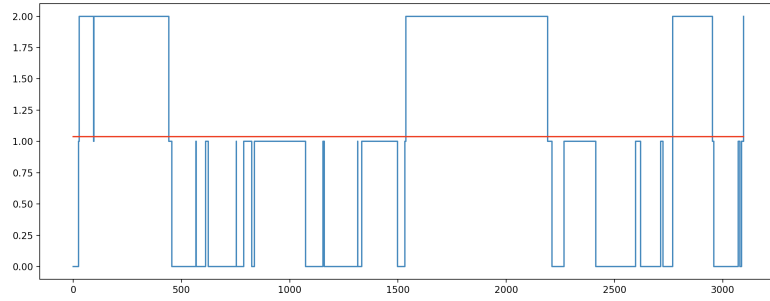Component 1 buffer of Workstation 3:

6

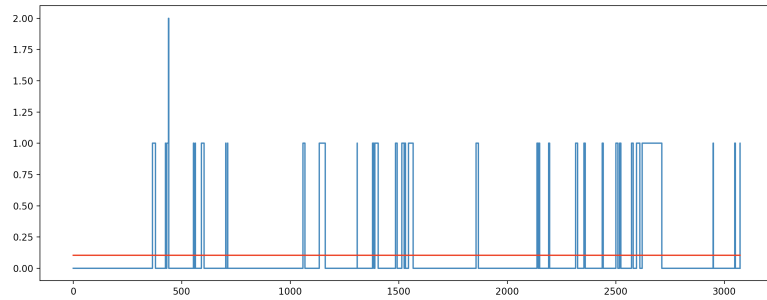Figure 7: The number of components in Component 2 buffer of Workstation 2

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

Verification: $L = 0.105, \lambda = 0.023, w = 4.598, T = 3135.49, N = 35$

$$\lambda w = 0.023 * 4.598 = 0.106 \approx 0.105 = L \qquad (5)$$
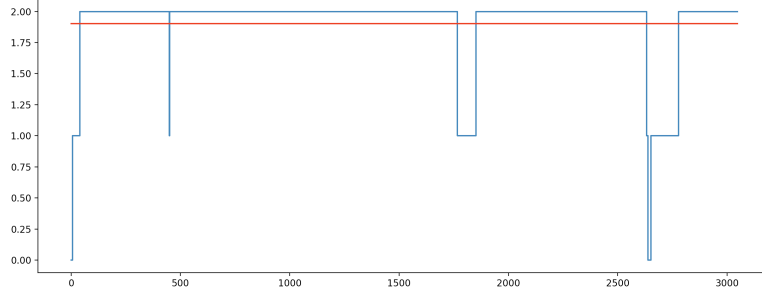
This queue pass the Little's law verification.

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

Component 3 buffer of Workstation 3:
Verification: $L = 1.903, \lambda = 0.013, w = 145.029, T = 3135.49, N = 20$

$$\lambda w = 0.013 * 145.029 = 1.885 \approx 1.903 = L \qquad (6)$$

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

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.

8

1.2.2 Model Assumptions

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 experiment data for us and these points are representative. For the structure part, I used Python language to build a system as

1.2.3 Input-Output Transformation

2 Production Runs and Analysis