SYSC 4005 Simulation Project

Final Report

Ryan Ribeiro (100997936)

Luke Newton (100999309)

Table of Contents

1.0 Problem Formulation

The manufacturing facility produces three types of product, each requiring one or more different components. Products are assembled at workstations with each unique product type having its own workstation. Each workstation has one or more buffers for each unique component and the workstation must wait for one of each of the required component(s) before assembling the product. There are two inspectors who must inspect the components before sending the components in the appropriate buffer for a given workstation. The policy for selecting which buffer to send the components from inspector one is to pick the workstation with the shortest buffer. If there is a tie between two buffers, the priority is workstation one, then workstation two, then workstation three. If all the buffers of a given component type become full, then inspector for that component is blocked from sending any more components until a buffer has space available.

There are three different types of products, P_1, P_2, and P_3, each assembled at its own workstation W_1, W_2, and W_3 respectively. There are three different types of components, C_1, C_2, and C_3, required to assemble the products. P_1 requires one C_1, P_2 requires one C_1 and one C_2, and P_3 requires one C_1 and one C_3. There are two inspectors, I_1 and I_2, who inspect the components. I_1 inspects component C_1 and I_2 inspects components C_2 and C_3. The inspected components are placed into the workstation buffers, denoted as $B_{i, j}$, where i is the workstation number and j is the component type of the buffer. This means C_1 can be sent to $B_{1,1}$, $B_{2,1}$, or $B_{3,1}$, C_2 is sent to $B_{2,2}$, and C_3 is sent to $B_{3,3}$.

2.0 Setting of Objectives and Overall Project Plan

2.1 Objectives

The question that we want to answer is:

How should inspector 1 distribute their C_1 components?

Criteria for evaluating alternatives is the throughput of the system and idle time of each workstation and inspector. Our primary objective is to maximize the throughput of the system; other objectives are to minimize the idle time of inspectors and workstations.

The current method inspector 1 uses to place C_1 components is to place them in the buffer with the fewest components. If there is a tie, W_1 has highest priority, then W_2, then W_3. We will consider two alternatives to the current method.

First, we will alter the criteria inspector 1 uses to pick a buffer to place C1 components. Instead of picking the shortest buffer the inspector's C1 component into, we will use a fair/round-robin policy (i.e. 1st C_1 to W_1 buffer, 2nd C_1 to W_2 buffer, 3rd C_1 to W_3 buffer, 4th C_1 to W_1 buffer, etc.). This will mean that there is no need for the secondary workstation priorities, and we should see a more even number of each product being created. In the event that a buffer is full when it is the buffer's "turn" to receive a C_1, the buffer will be skipped and C_1 will be placed into the next available buffer. If all buffers for component C_1 are full, then inspector 1 is blocked from sending any more components to the buffers. Once a buffer for component C_1 becomes available, inspector 1 sends the component to that buffer and the selection policy continues from that buffer.

Second, we will change the priority of the workstations in the event of a tie when using the current criteria. The current priority when multiple buffers are equally empty prioritizes W_1, but this may lead to inspector 2 being idle for long periods of time, so we will invert the priority order to W_3, then W_2, then W_1.

Simulation is an appropriate method to answer this question because there are many different criteria and priorities to choose, and it would be too time consuming and resource intensive to actually implement a production line for each C_1 distribution method we want to compare.

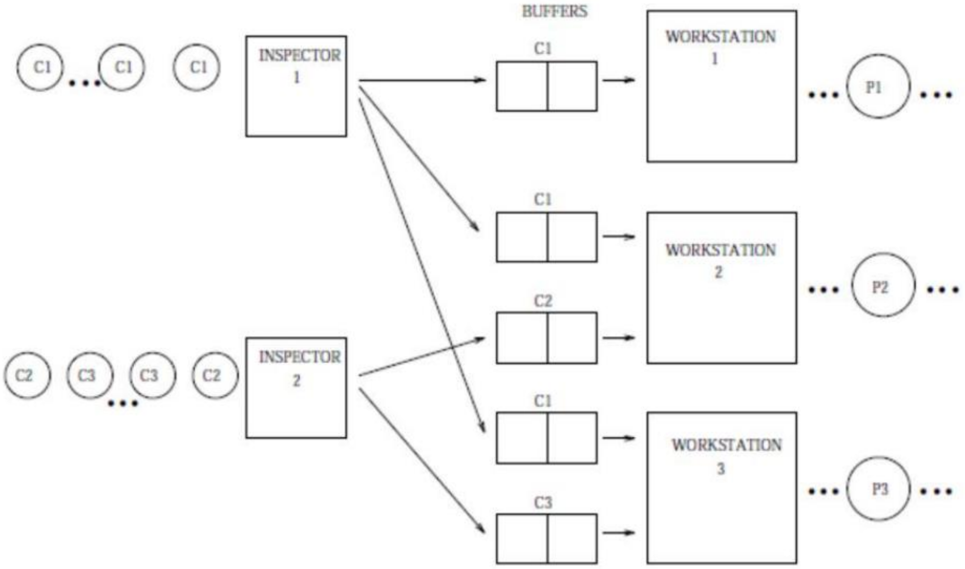
2.2 Project Plan

Our group will meet every Monday from 11:00am to 12:00pm and Tuesday 10:30 to 11:30am to work on the project. These times should work well because they are the scheduled times for the labs each group member is registered in, so we will always be free and will have access to the TA if needed. Depending on progress, additional meetings may be required to ensure completion on time.

The schedule for completing each task is not predictable as it depends on the progress we make in class, but during each weekly meeting the goal is to apply the new knowledge from the previous week's classes so that the project stays up to date.

3.0 Model Conceptualization

Below is the provided schematic of the manufacturing facility from which the entities, relationships among these entities, abstractions, and assumptions that are being made about the system will be listed.



3.1 Entities

Component: The lowest level piece of the system which is used by every other part of the manufacturing system. There is an infinite inventory of components available and taking components from the inventory happens instantaneously. There are three types of components.

Inspector: An inspector is an individual that inspects specific components for some time and then sends the component off to a buffer before starting to inspect the next component. There are two different inspectors.

Buffer: A queue which holds only a specific type of component for a given work station. At most, a queue will hold two items. There are five buffers among all workstations.

Workstation: The part of the system which takes in components as inputs to general different products as outputs. There are three workstations total and each workstation only assembles on type of product. Workstations can only begin to assemble products once they have the required components in their buffers.

Product: The final output of the system obtained from a workstation. There are three different products created and output by the system.

3.2 Entity Interactions

Components and Inspectors: Inspectors take a single component (C1 to inspector 1 and C2/C3 to inspector 2) and inspect the component for some amount of time. Once completed, the inspector will send the component to a specific buffer and instantaneously retrieve take another component for inspection.

Components and Buffers: Components are placed into buffers of the same component type and are only removed from the buffer when the workstation begins assembly.

Components and Workstations: Workstations require a specific combination of components before constructing a product out of the components.

Components and Products: Components are used in the assembly of products.

Inspectors and Buffers: Inspectors place the inspected components into a buffer of the same component type. The buffer chosen depends on the specific routing policy being used. Additionally, buffers can block inspectors from sending additional components if the buffers for that component type are full.

Workstations and Buffers: Each workstation has one buffer per component type that is used in the assembly the product associated with the specific workstation. The workstation will remove one of each of the required components from the workstation's buffer(s) in order to assemble a product.

Workstation and Product: Workstations will assemble a specific product over some amount of time once the required components exist in the workstation's buffers.

3.3 Essential Features

The essential features of this simulation are the inspectors, the buffers, and the workstations. The inspectors are essential because the amount of time taken to inspect the components directly affects how long it takes to go from component to finished product. The buffers are essential because it is important to observe whether the buffers are being filled up too often, or not often enough, to determine if there is a bottle neck in the speed of inspecting a given component. Finally, the workstations are essential for a similar reason as the inspectors in determining the total time taken to go from components to final products.

3.4 Assumptions

When an inspector gets blocked from placing a component into a buffer on account of the buffer being full, it is assumed that the inspector will keep that component and will able to instantly send that component to a buffer once a buffer becomes available. The reason for this assumption is that the component will eventually be needed, and it would be a waste of time to dispose of the component and then inspect another one later.

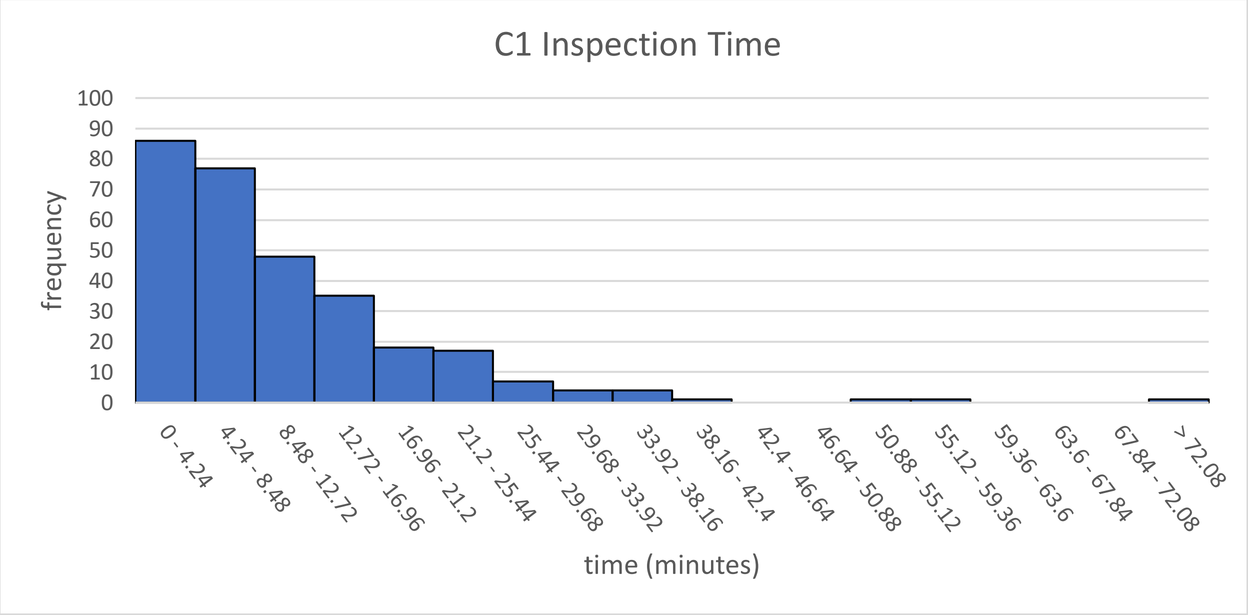
We also assume that Inspector 2, who inspects both component 2 and component 3, selects his next component to inspect uniformly at random (i.e. 50% chance to inspect component 2 next, 50% chance to inspect component 3 next).

4.0 Data Collection and Input Modelling

Data has been collected for 6 different aspects of the environment: Inspection time for C1, Inspection time for C2, Inspection time for C3, W1 assembly time, W2 assembly time, and W3 assembly time. In this section, we use this sample data to fit distributions to each environment variable so we can get random samples for the simulation.

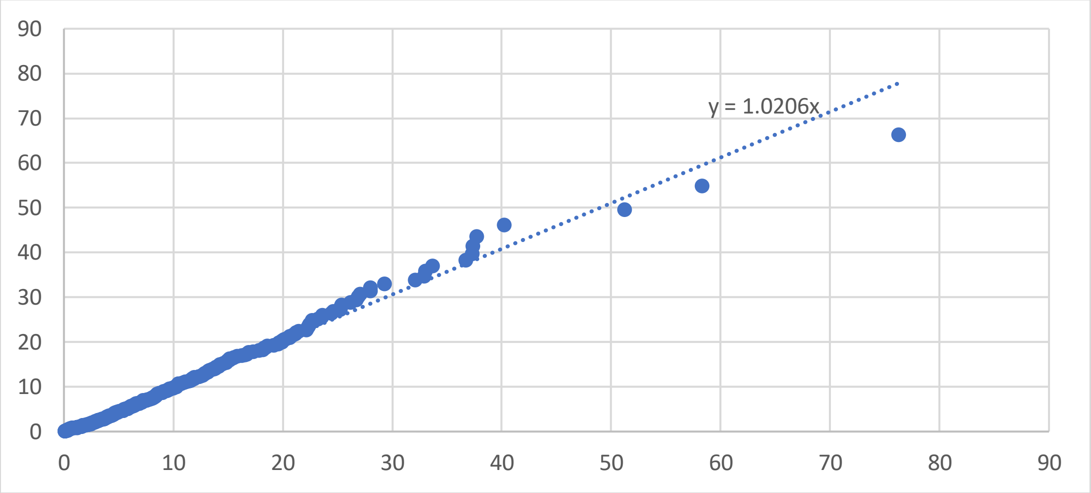
4.1 Inspection Time for C1

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins will be used to produce the histogram below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for C1 inspection time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one.

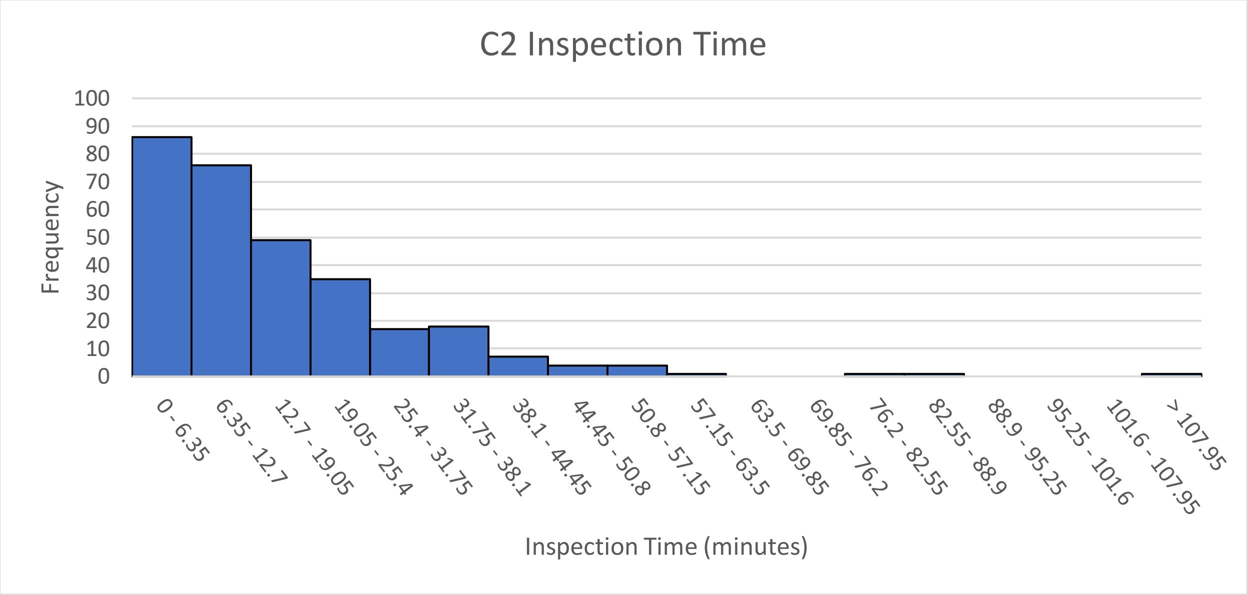
To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the inspection time for C1 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.096545

With 18 bins and one variable estimated, the assumed chi-square distribution has $18 - 1 - 1 = 16$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o{}^2{}_{0.05,\,16}$ = 26.3. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.1), $z_o{}^2$ = 15.0409. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the inspection time for C1 components follows an exponential distribution with $\lambda = 0.096545$.

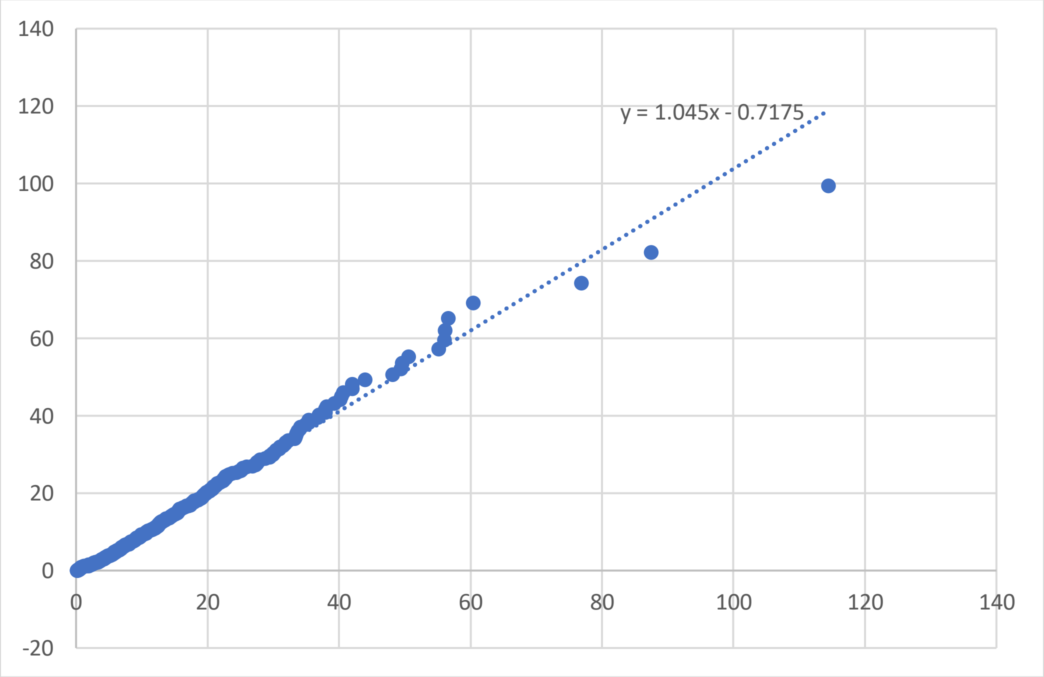
4.2 Inspection Time for C2

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins will be used to produce the histogram below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for C2 inspection time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one. A few of the largest values do not adhere to this linear trend as strongly, but this is fairly typical of the extrema in q-q plot.

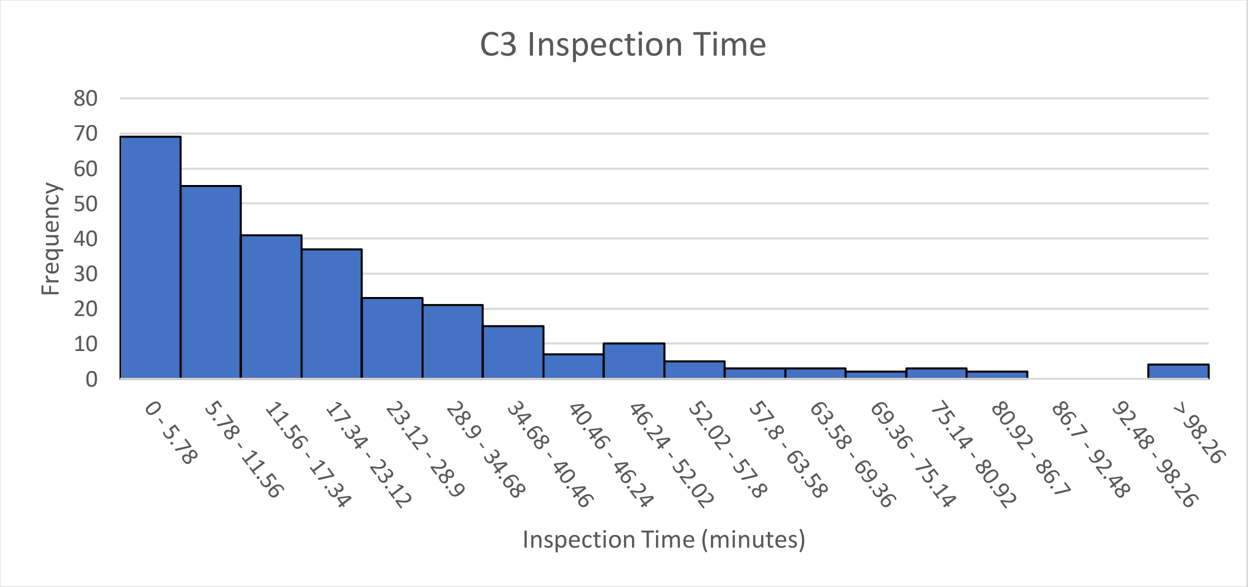
To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the inspection time for C2 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.0644

With 18 bins and one variable estimated, the assumed chi-square distribution has $18 - 1 - 1 = 16$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o{}^2_{0.05,\,16}$ = 26.3. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.2), $z_o{}^2$ = 15.663. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the inspection time for C2 components follows an exponential distribution with $\lambda =$ 0.0644.

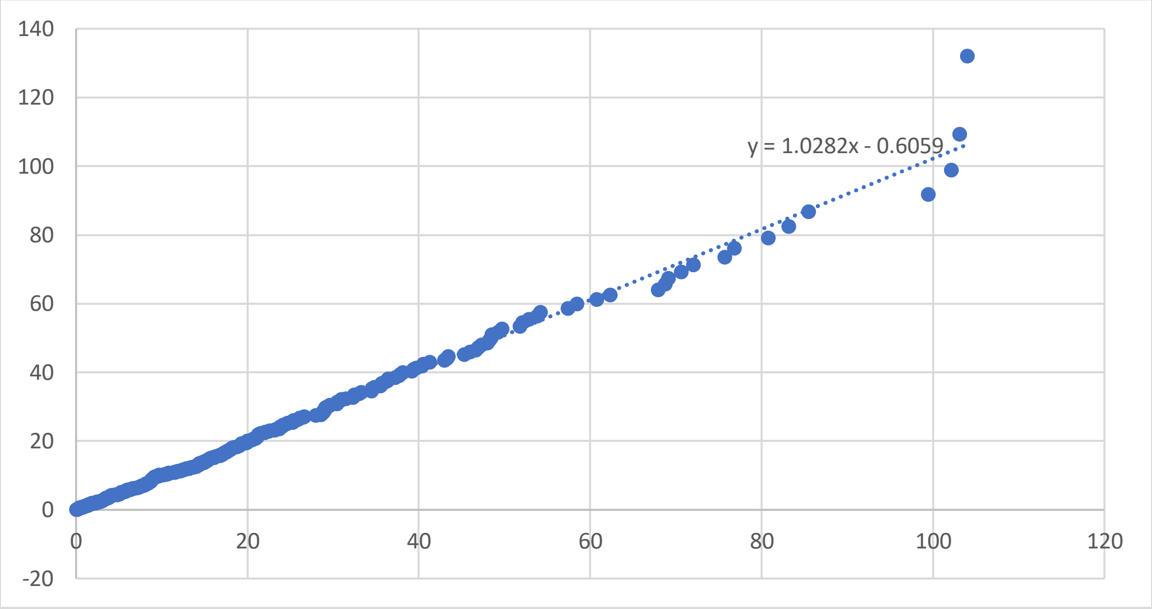
4.3 Inspection Time for C3

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins will be used to produce the histogram below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for C3 inspection time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one. A few of the largest values do not adhere to this linear trend as strongly, but this is fairly typical of the extrema in q-q plot.

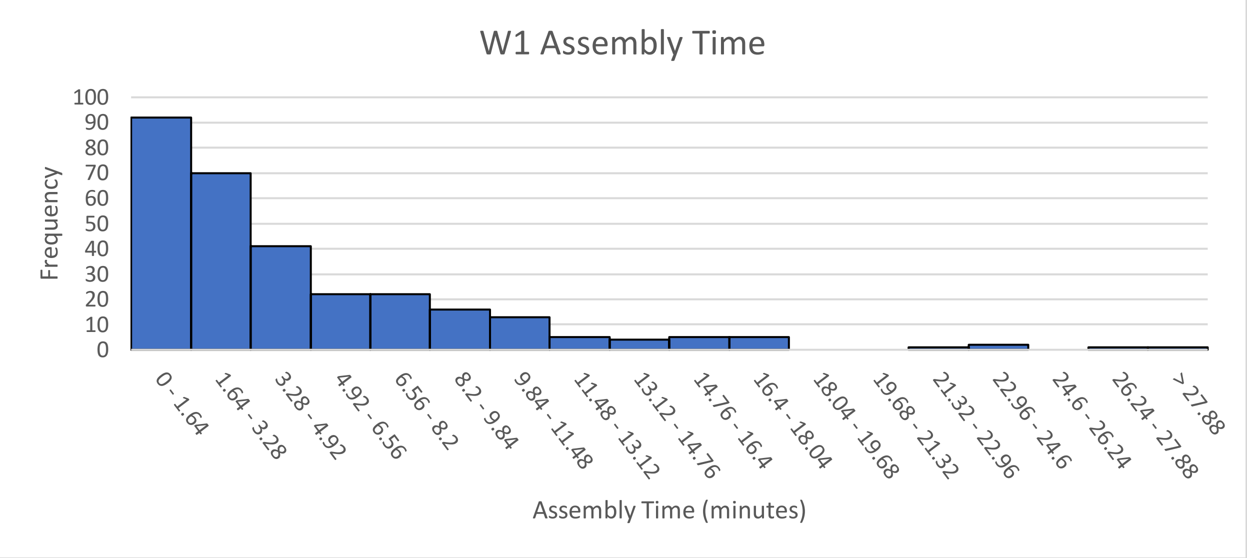
To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the inspection time for C3 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.048467

With 18 bins and one variable estimated, the assumed chi-square distribution has $18 - 1 - 1 = 16$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o{}^2{}_{0.05,\,16}$ = 26.3. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.3), $z_o{}^2$ = 17.48892. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the inspection time for C3 components follows an exponential distribution with $\lambda = 0.048467$.

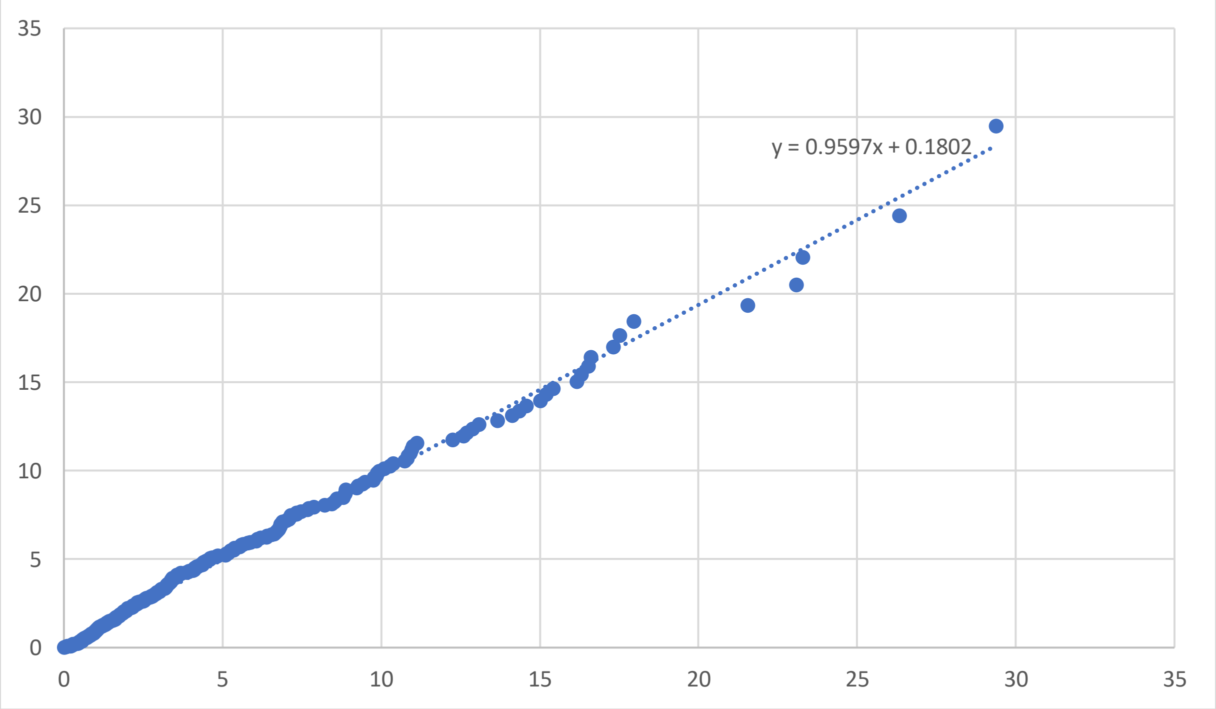
4.4 W1 Assembly Time

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins will be used to produce the histogram below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for W1 assembly time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one.

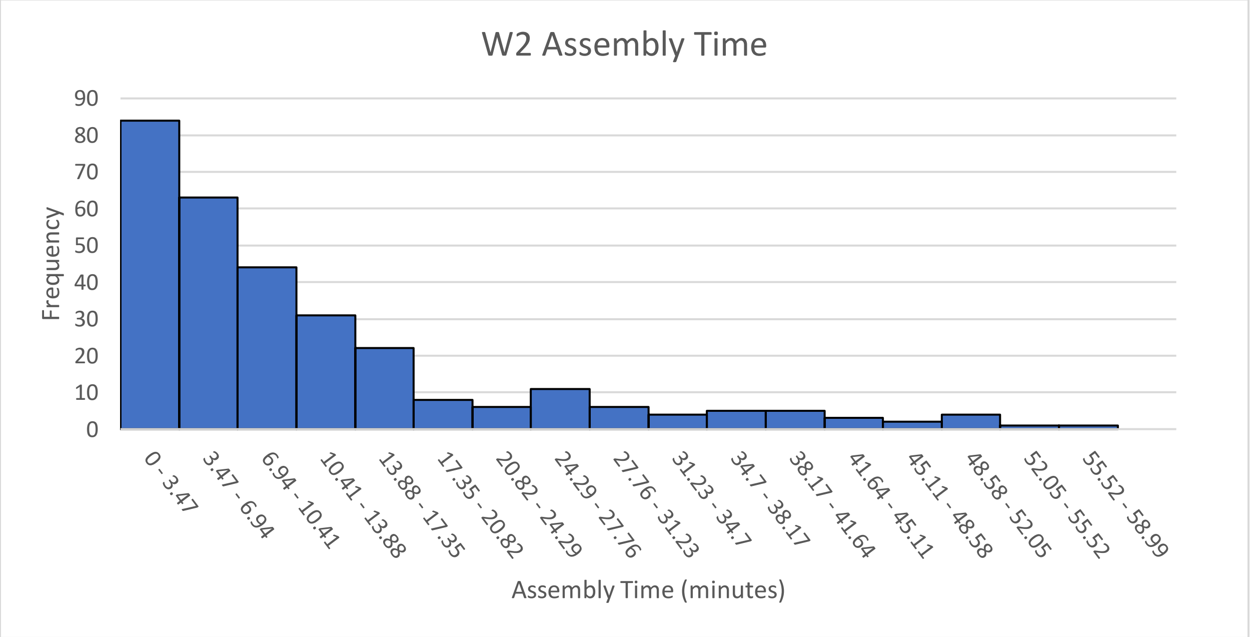
To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the assembly time for W1 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.217183

With 18 bins and one variable estimated, the assumed chi-square distribution has $18 - 1 - 1 = 16$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o^2{}_{0.05,\,16} = 26.3$. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.4), $z_o^2 = 13.52932$. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the assembly time for W1 components follows an exponential distribution with $\lambda = 0.217183$.

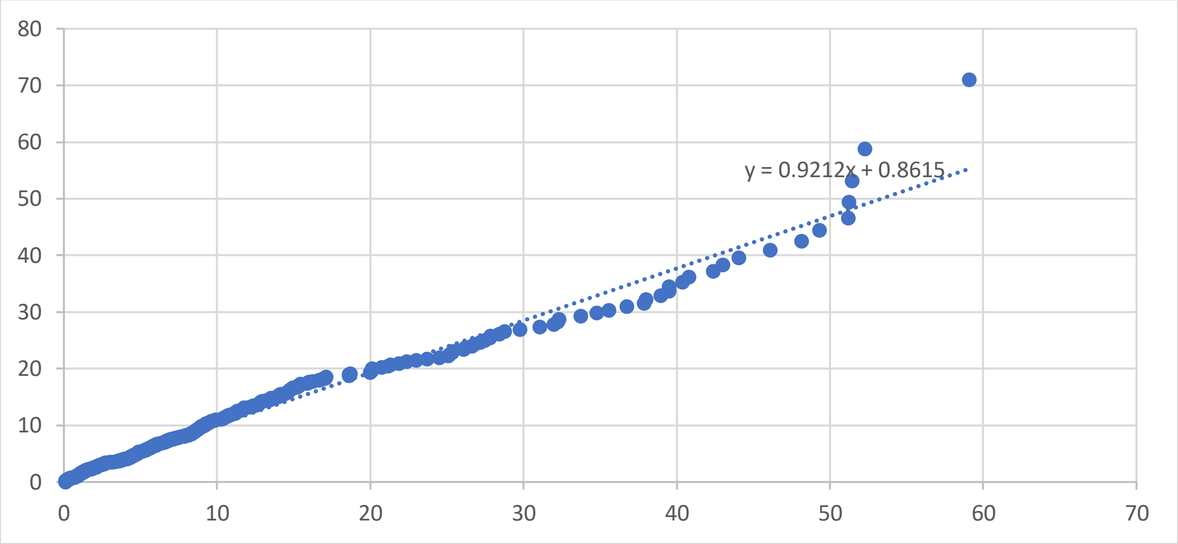
4.5 W2 Assembly Time

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins would be the standard; however, to reduce the number of empty bins, 17 bins will be used in the graph below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for W2 assembly time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one. A few of the largest values do not adhere to this linear trend as strongly, but this is fairly typical of the extrema in q-q plot.

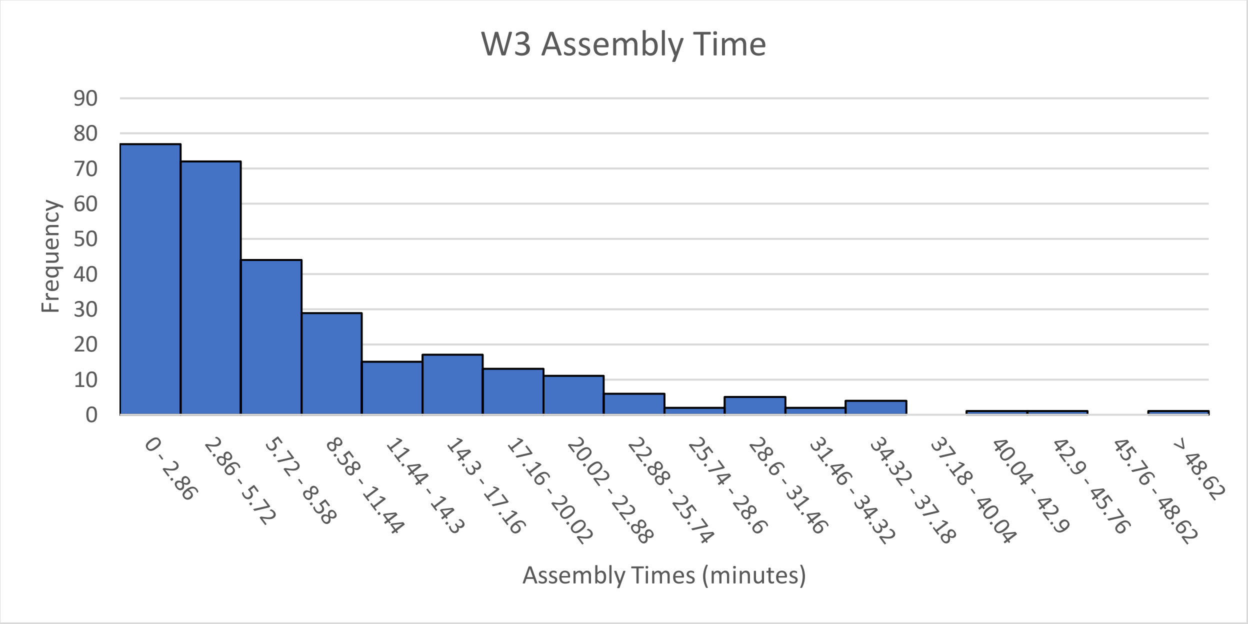
To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the assembly time for W2 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.0902

With 17 bins and one variable estimated, the assumed chi-square distribution has $17 - 1 - 1 = 15$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o{}^2{}_{0.05,\,15}$ = 25. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.5), $z_o{}^2$ = 13.64. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the assembly time for W2 components follows an exponential distribution with $\lambda = 0.0902$.

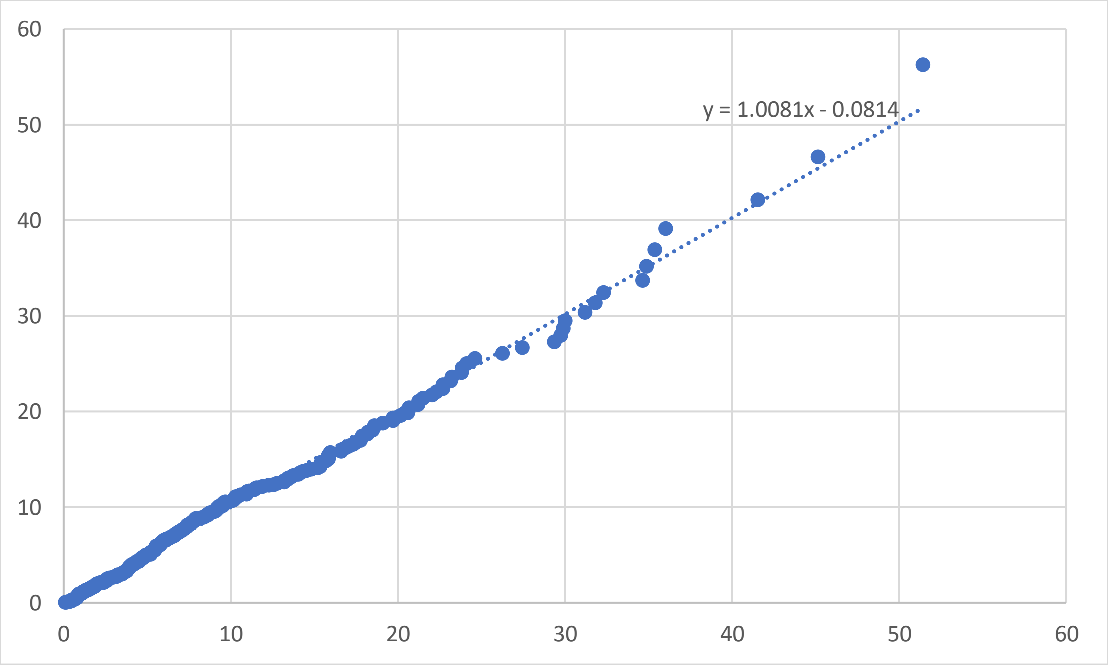
4.6 W2 Assembly Time

Given that the data represents service times and is a continuous variable, it likely follows an exponential distribution. As there are 300 data points, $\lceil\sqrt{300}\,\rceil = 18$ bins will be used to produce the histogram below:



Based on the shape of this plot, the sample data does seem to fit an exponential distribution, but quantile-quantile plot and chi-squared test must be used before we can say with certainty that this is the distribution that fits.

Using $\lambda = \frac{1}{\bar{X}}$ as the parameter estimation and the quantile function $F^{-1}(x) = \frac{-ln(1-x)}{\lambda} = -\bar{X}ln(1-x)$ yields the following quantile-quantile plot:



The linearity of this plot further confirms that an exponential distribution for W3 assembly time is a good fit, and the parameter estimation is accurate since the slope of the plot is almost exactly one.

To finalize this choice of distribution, a chi-squared test must be performed. This is an appropriate test to use in these circumstances since there are a large number of data points and there are estimated parameters.

H_0 = the assembly time for W3 follows an exponential distribution with $\lambda = \frac{1}{\bar{X}}$ =0.113693

With 18 bins and one variable estimated, the assumed chi-square distribution has $18 - 1 - 1 = 16$ degrees of freedom. Using a significance level of 0.05, the threshold value obtained from tables is $z_o{}^2{}_{0.05,\,16}$ = 26.3. By calculating expected frequencies from the hypothesized exponential distribution (see Appendix 1.6), $z_o{}^2$ = 13.68477. Since this is below the threshold value, the hypothesis can be confidently accepted, and we can say with certainty that the assembly time for W3 components follows an exponential distribution with $\lambda =$ 0.113693.

5.0 Model Translation

In this section we discuss how the simulation runs at a high level, then the specifics of each activity and some specific design choices that were made.

5.1 Choice of Simulation Language

MATLAB is the chosen language for this simulation, as it provides many benefits over other options. MATLAB is readily available on university computers, and can be downloaded for free through the university license, so the simulation will not require purchasing any specialized software tools. All team members have experience using MATLAB as well, meaning that most time can be spent on implementing the model, rather than learning the language. Most importantly, MATLAB has built in support for plotting, creating distributions, sampling from distributions, and generating random numbers; all of which will be useful functionalities for the simulation that we would otherwise need to implement ourselves.

5.2 High Level Description

The simulation starts by first initializing the model based on the results from section 4, along with the setup of the future event list (FEL) and the adding of initial events into the FEL. Next, we take the first event from the FEL, sorted in chronological order, and we process the event; this processing differs between event types. After the event is processed, we check if the FEL is empty. If it is empty, then we are finished, and we can output the statistics recorded from the simulation. If we are not finished, then we keep processing the top event from our FEL until all events are processed.

For a component C1 event, we first check if all the C1 buffers are full. If they are full, then we block inspector 1 from producing more C1s. If there is an open buffer to place a C1, then we do so for a given workstation. At that workstation, we then check if all the necessary components are available to assemble the given product for that workstation. If we are missing components, then we move on and generate the next C1 event. If the workstation does have the remaining components, then we check if there is already a product being produced in the workstation. If there is a product being produced, we generate an event for component C1 being ready as before. If there is no product at that workstation, then we unblock the workstation, remove the components from the buffers, we generate a future event to indicate that the product has been built, and then we generate an event for the next C1 being ready.

Component C2 and C3 events are handled in a similar fashion to C1 events. We first check if the corresponding workstation (W2 for C2 and W3 for C3) has a full buffer. If the buffer is full, then we block inspector two and advance to the next event in the FEL. If the buffer is not full, then we check if the respective workstations have the necessary components to assemble the product and if they are not currently assembling a product. If they have the components required and are not already assembling a product, then we unblock the workstation, remove the components, and generate a product-built event. If the workstation didn't have the components or it was assembling a product already, then we place the component into the workstation's buffer and determine the next component to inspect. If only one of the buffers for C2 or C3 are full, then we generate an event for the buffer that has space to accept another component. If both buffers are not full, then we uniformly random select C2 or C3 as the next component to generate an event for.

For the P1Built event, we start by increasing the total number of P1s built. Then, we check if the buffer for workstation 1 is empty. If the buffer is empty, we block workstation one to put it in an idle state. If it is not empty, then we want to remove a component C1 to build another P1. We then check if inspector one is currently blocked or not. If they aren't blocked, we just generate the next P1Built event. If they are blocked, we unblock them and generate the next P1Built event all the same.

P2Built and P3Built events are handled in a similar fashion. We treat them in the same way as P1Built events up until the point where we check if the inspector one is blocked. Since these products require 2 components, we also need to notify inspector 2 there is space in a buffer for a product. Inspector 2 is unblocked only if already blocked, and the component inspector 2 held when being blocked matches the component type for the product that was built.

The simulation can be run as a single, stand-alone replication by running Sim.m, or can be executed several times using the manySim.m script. If multiple replications are done, the random number streams for sampling data continue using the same seed, but do not reset the sequence produced, so different results are produced with each replication.

5.3 Design Choices

5.3.1 Calculating Idle time for Inspectors and Workstations

We calculate the idle times for the inspectors and the workstations in a similar fashion. In essence, we are using arrays to record the start and end times for when an inspector or workstation is idle. At the end of the simulation, we sum all the differences between all start and end times to get the overall idle time for a specific inspector or workstation. This approach allows us to throw out any data from an initialization phase, which we would not be able to do with a single running summation throughout the simulation. For inspectors, we take the start time at when they have inspected a component but all buffers for that component are full, and the end time when a buffer for their component is no longer full. Similarly, for workstations we take the start time at when the one or both buffer for that workstation are empty and a product is not in production, and we record the end time at the time the workstation begins to assemble an item.

5.3.2 Unblocking Inspectors

We handle the unblocking of inspectors in our product-built events. We have decided to do this because an inspector is only blocked when they have nowhere to place a component. As a result, an inspector can only be unblocked if the components are consumed and used in the assembly of a product. Therefore, we determine which inspector should be unblocked depending on the type of product being assembled, and we handle this when the product is built.

5.3.3 Multiple Random Number Streams for Sampling from Distributions

During the initialization phase, independent random number streams are created for sampling from each of the 6 distributions used for service times, as well as another for inspector 2's next component to inspect, for a total of 7 independent streams. This is done so that the sample values from different distributions are more independent, and also allows for much easier testing since the same sequences can be used for each simulation run.

5.3.4 Verbose Mode

We made the decision to include a verbose mode for the simulation. This is controlled by a boolean value at the beginning on the Sim.m script and allows detailed information about the execution to be displayed in the command window. We chose to do this because it allows us to see the order that events occur and are processed so we can ensure the simulation runs properly without doing much formal testing (which will be in the next deliverable).

5.3.5 Defining Objects for Event and Future Event List

While most of the simulation is written in a procedural style, we chose to also create specific objects for events and for the future event list. Making an object for the future event list was helpful for the design of the program, because it allows us to create a customized list with capabilities beyond what a normal MATLAB vector could do and allows us to encapsulate all the functionalities related to the future event list in one place. Since we created this custom list object, it was also necessary to create an object for an event so we can group all the information for a single event together and sort them all.

5.4 Alternative Design Description

The proposed alternative design (description here) appears in the function to process C1Ready events. When the activity reaches the point where the inspector must decide which buffer to place the new component one into, two global booleans called alternativeStrategy and alternativePriority are checked. These boolean variables are set at the top of Sim.m, along with other program control variables.

If alternativeStrategy is true, then we use the alternative round-robin strategy for place component 1's, otherwise the components are placed in the shortest buffer. If alternativePriority is true, then when two buffers have the same length, the order of priority for choosing a buffer is workstation 3, then 2, then 1; otherwise the default priority of workstation 1, then 2, then 3 is used. Note that if both alternativeStrategy and alternativePriority are true, only the alternative strategy is used, because a round-robin approach does not require a tie-breaking decision.

6.0 Verification

In this section we will be looking at different tests and checks we have implemented in our simulation that we use to ensure that the output of our system matches what we would expect given the implementation.

6.1 Clock Moves Forward in Time

One of the requirements of our simulation is that we are only progressing forward in clock time, i.e. we are not trying to process an event before we arrive at the event's start time. We are verifying this by having a condition in the processEvent() function of Sim.m which checks that the clock time is not after the start time of the event. If the check fails, we are displaying an error message and stopping the simulation. After we check this, we set the current clock time to be equal to that of the event in order to represent the progression of time.

6.2 Buffers Do Not Exceed Bounds

We need to make sure that the size of each buffer is always between zero and two inclusive at the end of the simulation, but also during all points of the simulation. We are checking for this in the processEvent() function in the Sim.m file by calling a function called checkBoundries(). The checkBoundries() function is located in Sim.m, and it checks that the passed buffer is within the range [0, 2]. If the buffer is outside of the acceptable range, then an error is thrown; otherwise, nothing happens as the system is functioning correctly.

6.3 Verbose Mode

Stepping through the simulation line by line during execution would be too difficult and time consuming of a method to check that the simulation is working correctly. Instead of doing this, we implemented a verbose mode which will output details about what is happening to various variables during the simulation to the console. When set to true, verbose mode will output several pieces of information every time an event is processed. Currently, verbose mode provides information on the FEL, the clock time, which event is being processed, if a component is being added to a buffer and where, if any products begin production, and the size of each buffer. These outputs let us monitor the state of the system after every processed event.

6.4 Flow Charts

In appendices 2.1 to 2.6 we have the diagrams that we originally used to design our simulation system. These flow charts were initially designed to help understand the flow of our simulation; however, they now serve a second purpose as a means of verifying that our system runs our simulation correctly. We can trace our way through the flow charts and determine what pieces of code correspond to a given event in the flow chart to ensure that all possible scenarios are being covered.

6.5 Products Produced vs. Components Inspected

After our simulation has finished, we generate a text file that contains the results of our simulation. These results include the number of each product produced, the number of each component used or left in a buffer, the number of each component inspected, and a few other pieces of information regarding idle time and average components in buffer. The key piece of information for this part is the information about the number of components inspected and the number of components used. We want to ensure that we are not using more components than we create, or that we are creating components but never using them. Either case would be an indicator that we are not handling the components correctly, so we have to check for any differences. To do this, we check the following formulas:

$$\begin{aligned}(\text{\# C1 Inspected}) \overset{?}{=} {} & (\text{total \# products produced}) + (\text{size of worksation } 1'\text{s C1 buffer}) \\ & +(\text{size of worksation } 2'\text{s C1 buffer}) + (\text{size of worksation } 3'\text{s C1 buffer}) \\ & +(\text{\# of products currently under production})\end{aligned}$$

$$(\text{\#C2 Inspected}) \overset{?}{=} (\text{\#P2 produced}) + (\text{size of worksation } 2'\text{s C2 buffer}) + (\text{any P2 in production})$$

$$(\text{\#C3 Inspected}) \overset{?}{=} (\text{\#P3 produced}) + (\text{size of worksation } 3'\text{s C3 buffer}) + (\text{any P3 in production})$$

If all three of these equations are satisfied, then we can say that components move through the simulation correctly.

6.6 Little’s Law

Normally, the purpose of Little’s Law would be to allow us to calculate the average number of components in each buffer using the effective arrival rate and the average time a component spends in a buffer. Calculating the average time a component spends in the system would require keeping track of all components as separate entities, which our simulation is not capable of doing without major reworking. Instead, we find the average number of each buffer by recording the number of items in each buffer every time an event is processed, and then we divide it by the total number of events processed. We can do this for every buffer, and it will give us the same result as if we used Little’s Law.

The reason we are looking for average sizes is because we want to ensure that the values are within an appropriate range. The average components in any buffer should be between zero and two, but given that workstation 1 is the preferred buffer for component one placement and has the lowest average service time (see here), it is expected that the workstations will have component 1 buffers with small average sizes and fuller buffers for components 2 and 3. 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.

7.0 Validation

The validation section looks at different ways that we can determine that the results of our system are reasonable and expected. As an example, we would be looking to see if the number of each product produced makes sense and matches our expectations.

7.1 Sanity Checking Our Inputs and Outputs

7.1.1 Making Sense of the Outputs

After every simulation we print out a series of results to a file so that we can go though and rationalize if the outputs are reasonable. For example, we record the number of products produced and then determine if that value make sense. The first thing we check is if it seems too high/low for the given simulation time (e.g. it would be unreasonable to have 100s of each product in a 10-minute simulation). We then check if the distribution of products produced makes sense for the strategy being used. When we use the default, we expect significantly more product ones, and then a split between product two and three. When we use the alternative, we expect less of product one, and a higher, but roughly equal, amount of product two and three.

In addition to the number of products, we are also looking to see if the idle times of the inspectors and workstations makes sense. We check that inspector one has zero idle time, or very little, because it would make sense for them to always be inspecting components since component one is required by all workstations. Similarly, we are checking that the other idle times make sense given the amount of each product produced. We would expect the idle time for a workstation to be very high if the number of products for that workstation was low.

The bottom line is that we can view the results we generate from a simulation and we can determine if the values recorded are rational. By doing so, we are able to determine if the simulation is producing values that would be expected of it, and thus determine if we have constructed the system correctly.

7.1.2 Sensitivity Analysis

For our simulation there is not much need for sensitivity analysis, nor does it benefit us from extensively testing this. There are only two values that are relevant for sensitivity analysis in our simulation: seed and simulation time. Performing sensitivity analysis on the seed is completely pointless as different seeds provided different random values, but all seeds provide random values that we cannot predict.

As for the simulation time, we can see some variance in our results depending on what value we use. We expect to see that larger simulation times produce a larger number of products while smaller times produce less products, which is indeed the case. We would also expect a lot more variance in the amount of products produced for smaller simulation times, since we do not run for long enough to reach a steady state. As we increase the simulation time, we can see that the results start to average out and reflect the outputs we would expect.

7.2 Real-World Output Data

A good way to validate that our simulation is accurate would be to use historical data from the real-world system and compare it to the results of our simulation. Historical data allows us to see what the actual results should look like, regardless of what we think they should be. Currently, we are inspecting the results and we have to rationalize whether or not they make sense. This method isn't bad, but it hinges on us being able to accurately estimate what the final outputs should be. Without access to any real-world output data, we cannot determine with absolutely certainty that the results of our system are accurate or expected.

7.3 Real-World Input Data

In lieu of any real-world output data, the best we can do is compare the system output from known input to the system output from random input. As such, we decided to implement the option to use pre-generated input data in place of randomly generated values. This allows the program to read in service times from a file and use those values instead of random sampling from a distribution. Since we analyzed the data in section 4, we know some information such as the average inspection times for each component and product. Using this information, we can assess compare it with the results of our simulation and determine if they make sense.

For example, in section 4 we determined that our data had an average inspection time of 10.35 min and 15.54 min for inspector one and two respectively. Knowing this, we can draw the conclusion that we will end up with more P1s than P2s or P3s. When we look at the results of our simulation, using the milestone one data as our input, we find exactly that. The number of P1s produced is much larger than the number of P2s or P3s. This finding helps to further validate that the simulation we constructed is correct and that our results are accurate. Additionally, when we run the simulation for equal lengths using random inputs and known inputs, the number of products produced are very similar.

Note that there is only enough input data in the input files to run ~3000 minutes of simulation. If using the readInFilesMode, you cannot run a simulation longer than 3000 minutes.

8.0 Production Runs and Analysis

To exactly replicate the results discussed in this section using the simulation, enter the seed as 5437, use the default strategy (alternativeStrategy = false and alternativePriority = false in Sim.m), and set a run time of 1000 minutes.

8.1 Identifying Steady State and Initialization Phase Length

Since each replication of the simulation begins with empty buffers for all workstations and no components immediately being inspected, there is a bias in the system output which pulls down the average values we use for results. The longer a replication is run, the less of an impact this bias has, so it would be possible to just run the simulation for a long period of time to remove the bias from our results, but it would be better to determine a point at which the system reaches a steady state so we can remove the bias completely.

This can be a difficult task, since the output is random so inherently it is not that steady. To determine where steady state begins, we plotted the size of each buffer against simulation time, as well as the frequency of event occurrences against time. These plots can be seen in Appendices 3.1 to 3.11 and seem to suggest an initialization phase of about 225 minutes. Of course, some of the results tell us very little about where steady state occurs (e.g. 3.4 or 3.11), but 225 minutes is about the point where the buffers for component 2s and 3s fill, and the point where the amount of C1 inspected and P1 built become more consistent.

8.2 Identifying the Required Number of Replications

In order to obtain usable results, we need to perform several independent replications of the simulation. We achieve this with our manySim.m script, which runs many replications that each use a separate FEL, along with resetting all state variables at the beginning of each replication. Each replication uses the same random number streams initialized from the same seed, but since the seed is only reset once at the beginning of the first replication, we will get different results in each replication.

In the project requirements it is stated that we want 95% confidence intervals with a width that does not exceed 20% of the estimated value. In order to do this, run manySim.m and set the control variable “calculateReplicationsRequired” to true. This will execute 10 replications, after which each confidence interval is calculated from the data from each replication; if any confidence intervals are too wide, another replication is performed. Due to the random nature of the output, depending on the seed it could take 2 replications or 50 to get an appropriate width, so we chose 10 as a benchmark to ensure that there is always many independent replications to obtain data from.

8.3 Simulation Results

To obtain these simulations results, we ran the simulation with seed 5437, set each replication to simulate 3000 minutes, and continued to run replications until all confidence intervals were less than 20% the width of their estimated values. For inspector 1, the default policy of selecting the shortest buffer for C1’s was used, along with the default priority of W1, then W2, then W3. To reproduce this with the simulation program set the following control variables:

In manySim.m:

```
calculateReplicationsRequired = true;
seed = 5437;
compareAlternateDesigns = false;
alternativeStrategy = false;
alternativePriority = false;
```

In Sim.m:

```
maxSimulationTime = 3000;
readInFilesMode = false;
removeInitializationBias = true;
initializationPhaseLength = 225;
```

Statistics were collected for the number of each component inspected, the number of each product produced, the sizes of each workstation buffer, and the proportion of time that each inspector and workstation is idle:

Statistic	Estimated mean	Sample variance	95% confidence interval
P1 Produced	234.000	210.667	[223.617 244.383]
P2 Produced	28.700	37.789	[24.302 33.097]
P3 Produced	5.000	7.555	[3.034 6.966]
Total Products Produced	267.700	229.344	[256.866 278.533]
C1 Inspected	267.700	225.567	[256.956 278.443]
C2 Inspected	28.400	38.933	[23.936 32.863]
C3 Inspected	5.000	7.555	[3.033 6.966]
Workstation 1 C1 queue size	0.245	0.000645	[0.226 0.263]
Workstation 2 C1 queue size	0.0961	0.00321	[0.0556 0.136]
Workstation 2 C2 queue size	1.668	0.0445	[1.517 1.819]
Workstation 3 C1 queue size	0.0110	0.0036	[0.00313 0.0189]
Workstation 3 C3 queue size	1.904	0.00397	[1.859 1.949]
Proportion of Time Inspector 1 Idle	0	0	[0 0]
Proportion of Time Inspector 2 Idle	0.860	0.00176	[0.830 0.890]
Proportion of Time Workstation 1 Idle	0.556	0.000385	[0.542 0.570]
Proportion of Time Workstation 2 Idle	0.833	0.000948	[0.811 0.855]
Proportion of Time Workstation 3 Idle	0.908	0.000143	[0.899 0.916]

These results tell us that the production line system is heavily skewed towards inspector 1's role. In every simulation inspector 1 is never idle, while inspector 2 is idle about 86% of the time. This is because inspector 1 prefers to place components into workstation 1, which on average can produce a P1 faster than the inspector can inspect another C1. The result is the system is always produced many P1 while very few P2 and P3 are created as most C1 go towards creating P1s. By having an alternative design which more fairly distributes inspected C1s, the system should have a more even distribution of products and overall higher product output.

9.0 Alternative Design Comparison

The discussed results were obtained by running 10 replications for 3000 minutes each, with a seed value 5437. To reproduce this with the simulation program set the following control variables:

In manySim.m:

```
seed = 5437;
numberOfReplications = 10;
compareAlternateDesigns = true;
```

In Sim.m:

```
maxSimulationTime = 3000;
readInFilesMode = false;
removeInitializationBias = true;
initializationPhaseLength = 225;
```

Note that comparing alternate designs will run numberOfReplications times 3. For numberOfReplications = 10, this is 30 replications and will take ~1.5 minutes to execute. Setting verbose mode to true prints a lot of information about each replication, which takes some time, so printing this data for 30 (or more) replications will take a very long time to execute!

9.1 Identifying Required Number of Replications

Comparing Designs has different requirements for replications than the previous section. In Production Runs and Analysis, we were only concerned with ensuring that confidence intervals had a width of 20% of the estimated value; since we are comparing the difference of values here, we also need to ensure that confidence intervals are entirely positive or negative so we can say for sure whether the true difference between the systems is positive or negative for each statistic.

Of course, if the values are approximately equal for the two systems, then it would take far too many replications for the confidence intervals to not contain zero, so we will impose the following rules as well to determine when we have enough replications:

1. For the number of products made and components inspected, a confidence interval that does not exceed 5 or -5 is considered equal (corresponding to less than 5 product/component difference between systems)
2. For the proportion of time spent idle for inspectors and workstations, a confidence interval that does not exceed 0.01 or -0.01 is considered equal (corresponding to less than 1% difference in proportion of time spent idle between systems)

In the case of our system, these rules all held after 10 replications, so no additional replications were required after the results obtained in section 8.3.

9.2 Simulation Results from Alternative C1 Placement Strategy

Statistic	Estimated mean	Sample variance	95% confidence interval
P1 Produced	121.400	281.156	[109.620 133.180]
P2 Produced	71.1000	60.3222	[65.5440 76.6550]
P3 Produced	74.6000	32.2667	[70.5365 78.6635]
Total Products Produced	267.100	227.211	[256.317 277.883]
C1 Inspected	267.300	219.567	[256.700 277.900]
C2 Inspected	71.2000	60.8444	[65.6200 76.7800]
C3 Inspected	75.1000	32.3222	[71.0330 79.1700]
Workstation 1 C1 queue size	0.07140	0.00197	[0.03962 0.10317]
Workstation 2 C1 queue size	0.95820	0.07984	[0.75607 1.16033]
Workstation 2 C2 queue size	0.43631	0.06890	[0.24854 0.62408]
Workstation 3 C1 queue size	0.92130	0.07591	[0.72420 1.11839]
Workstation 3 C3 queue size	0.42251	0.05272	[0.25826 0.58677]
Proportion of Time Inspector 1 Idle	0.00261	0.00001	[0.00045 0.00477]
Proportion of Time Inspector 2 Idle	0.11308	0.00872	[0.04629 0.17987]
Proportion of Time Workstation 1 Idle	0.73860	0.00137	[0.71211 0.76508]
Proportion of Time Workstation 2 Idle	0.67350	0.00156	[0.64525 0.70176]
Proportion of Time Workstation 3 Idle	0.69004	0.00094	[0.66809 0.71198]

Here we can observe that the alternative C1 placement strategy results in a more balanced distribution between the number of product P1s, P2s, and P3s produced. Additionally, we can see that Inspector 1's idle time is almost zero because every product requires a C1 component. However, Inspector 2 has a very low idle time too, indicating that they and not being blocked and that components C2 and C3 and being used, which we know is true based on the number of product P2 and P3 being produced.

9.3 Simulation Results from Alternative C1 Placement Priority

Statistic	Estimated mean	Sample variance	95% confidence interval
P1 Produced	117.200	384.400	[103.175 131.225]
P2 Produced	65.3000	74.0111	[59.1458 71.4542]
P3 Produced	85.0000	57.3333	[79.8534 90.4166]
Total Products Produced	267.500	245.167	[256.230 278.701]
C1 Inspected	267.300	236.233	[256.305 278.295]
C2 Inspected	65.9000	71.6556	[59.8445 71.9555]
C3 Inspected	85.0000	57.5556	[79.5730 90.4271]
Workstation 1 C1 queue size	0.10028	0.00150	[0.07258 0.12798]
Workstation 2 C1 queue size	0.67757	0.00469	[0.62857 0.72657]
Workstation 2 C2 queue size	0.65153	0.01579	[0.26164 0.44142]
Workstation 3 C1 queue size	0.81606	0.00491	[0.76594 0.86619]
Workstation 3 C3 queue size	0.22394	0.00760	[0.16158 0.28630]
Proportion of Time Inspector 1 Idle	0.00102	0.00000	[-0.00008 0.00212]
Proportion of Time Inspector 2 Idle	0.05228	0.00275	[0.01475 0.08980]
Proportion of Time Workstation 1 Idle	0.74610	0.00189	[0.71507 0.77713]
Proportion of Time Workstation 2 Idle	0.69800	0.00151	[0.67025 0.72576]
Proportion of Time Workstation 3 Idle	0.65637	0.00122	[0.63143 0.68132]

Here we can observe that the alternative C1 placement priority results in a more balanced distribution among the three products, with slightly more product P3 produced compared to P2. This is expected as our priority is workstation W3>W2>W1, so we would expect more product P3s over P2s. We also note that the idle time of Inspector 1 is still nearly zero, as expected, and that the idle time for Inspector 2 is approaching zero as well. This indicates to us that our alternative placement priority is effective at reducing the amount of idle time among the inspectors.

9.4 Results from Comparing Original Design to Alternative Strategy

Statistic	Estimated mean	95% confidence interval
P1 Produced	112.600	[104.820 120.380]
P2 Produced	-42.4000	[-47.6481 -37.1519]
P3 Produced	-69.6000	[-73.5500 -65.6500]
Total Products Produced	0.60000	[-0.36566 1.56566]
C1 Inspected	0.40000	[-0.62285 1.42285]
C2 Inspected	-42.8000	[-48.2771 -37.3229]
C3 Inspected	-70.1000	[-74.1810 -66.0191]
Workstation 1 C1 queue size	0.17365	[0.13796 0.20934]
Workstation 2 C1 queue size	-0.86206	[-1.09475 -0.62938]
Workstation 2 C2 queue size	1.23236	[0.91719 1.54753]
Workstation 3 C1 queue size	-0.91028	[-1.11047 -0.71008]
Workstation 3 C3 queue size	1.48161	[1.28824 1.67498]
Proportion of Time Inspector 1 Idle	-0.00261	[-0.00477 -0.00045]
Proportion of Time Inspector 2 Idle	0.74695	[0.65358 0.84033]
Proportion of Time Workstation 1 Idle	-0.18205	[-0.21282 -0.15127]
Proportion of Time Workstation 2 Idle	0.15956	[0.13016 0.18895]
Proportion of Time Workstation 3 Idle	0.21794	[0.19372 0.24216]

A negative value in the table above indicated that the value for the alternative strategy design is higher, while positive means that the default design value is higher. In this case, we observe that the alternative strategy we implemented produces a higher amount of produce P2 and P3, but a lower amount of product P1. Despite this, we observe that both the original design and the alternative strategy produce almost the same amount of total products, with less than one product difference between the two designs on average.

9.5 Results from Comparing Original Design to Alternative Priority

Statistic	Estimated mean	95% confidence interval
P1 Produced	116.800	[109.045 124.555]
P2 Produced	-36.6000	[-42.1020 -31.0980]
P3 Produced	-80.0000	[-84.6970 -75.3030]
Total Products Produced	0.20000	[-1.14043 1.54043]
C1 Inspected	0.40000	[-0.36897 1.16897]
C2 Inspected	-37.5000	[-42.7811 -32.2189]
C3 Inspected	-80.000	[-84.6970 -75.3030]
Workstation 1 C1 queue size	0.14478	[0.12000 0.16954]
Workstation 2 C1 queue size	-0.58143	[-0.66341 -0.499460]
Workstation 2 C2 queue size	1.31714	[1.12553 1.50875]
Workstation 3 C1 queue size	-0.80504	[-0.86074 -0.74934]
Workstation 3 C3 queue size	1.68018	[1.58778 1.77258]
Proportion of Time Inspector 1 Idle	-0.00102	[-0.00212 0.00008]
Proportion of Time Inspector 2 Idle	0.80775	[0.75543 0.86008]
Proportion of Time Workstation 1 Idle	-0.18955	[-0.22516 -0.15394]
Proportion of Time Workstation 2 Idle	0.13506	[0.10716 0.16295]
Proportion of Time Workstation 3 Idle	0.25160	[0.23158 0.27162]

A negative value in the table above indicated that the value for the alternative priority design is higher, while positive means that the default design value is higher. In this case, we observe that the alternative priority we implemented produces a higher amount of produce P2 and P3 — with P3 being a little over double P2 — but a lower amount of product P1. Despite this, we observe that both the original design and the alternative priority produce almost the same amount of total products, with less than one product difference between the two on average.

10.0 Conclusion

In section 8.3 we ran the simulation with the original design configurations and displayed the outputs. For starters, nearly 87% of the total products produced are of type P1. Additionally, Inspector 1 is never idle and Inspector 2 is idle for approximately 86% of the time. From these two points, we can conclude that the original design creates a system that is heavily skewed towards the role that Inspector 1 plays. This makes sense as the original design places the highest priority on Workstation 1. Additionally, product P1 has the shortest assembly time of all three products. This results in far more product P1s being produced and more idle time for Inspector 2. To further support this, we can see that the queue size for component C2 and C3 is fairly high and the queue sizes for any C1 component is nearly zero. This indicates that we are not waiting on C1 — and by extension Inspector 1 — for Workstations W2 and W3, but rather we are always waiting on Inspector 2.

In section 9.2 we ran the simulation with our alternative placement strategy in effect and displayed the outputs. In comparison to the original design, the alternative placement strategy resulted in a more even distribution of product P1, P2, and P3 produced. Additionally, the idle time of Inspector 2 dropped from 86% in section 8.3 to approximately 11%. Regarding idle times, we can see that workstation W2 and W3 are idle for about 67% and 69% respectively, which is a decent amount lower than the original design's 83% and 91% respectively. However, workstation W1 is idle about 74% of the time, which is much higher than the 56% in the original design. In summary, using the alternative placement strategy results in a more even distribution of products produced, less idle time for Inspector 2, less idle time for workstation W2 and W3, more idle time for workstation W1, and the same total number of products produced.

In section 9.3 we ran the simulation with our alternative placement priority in effect and displayed the outputs. The alternative placement priority that we implemented differs from the original design in that it prioritizes workstation W3, then W2, and then W1 in the event that all three workstations have the same number of component C1s. We observed a drop in the amount of time Inspector 2 is idle from 86% in the original design to 75% using our alternative. Similarly, we observed a drop in the workstation idle times for W2 and W3 from 83% and 91% in the original design to 70% and 66% using our alternative. The idle time for workstation W1 actually increased from 56% to 75% due to there being less component C1s being sent to workstation W1. In summary, using the alternative placement priority results in more product P2s and P3s being produced, less product P1s being produced, a little less idle time for Inspector 2, less idle time for workstation W1, and the same total number of products produced.

The question originally posed in the project objectives is:

How should inspector 1 distribute their C1 components?

with our goals being maximizing products produced. Both alternative designs produced on average the same total number of products as the original design, so ultimately which design to use depends on the distribution of products needed from the system. The original configuration heavily favors P1, so if mainly P1's are required then inspector 1 should distribute their C1 components according to the original policy. If a more balanced distribution of products is required, then inspector 1 should shift their C1 distribution policy to either of the alternative designs, since both cause similar changes in the distribution of products. It should be noted that if the distribution of products does not matter, and only the total produced is important, then the default design should be used since it does produce the same number of products as the alternative while have much higher idle times, meaning the same throughput is achieved for less work.

Appendix

Appendix 1.1: Spreadsheet Data for C1 Inspection Time

number of points	300
number of bins	18
max value	76.284
min value	0.087
data range	76.197
bin width	4.233167
sample mean	10.08446
sample variance	92.5225

bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X_0^2 $\frac{(O_i - E_i)^2}{E_i}$
1	0	4.24	0 - 4.24	86	101	2.227723
2	4.24	8.48	4.24 - 8.48	77	67	1.492537
3	8.48	12.72	8.48 - 12.72	48	45	0.2
4	12.72	16.96	12.72 - 16.96	35	30	0.833333
5	16.96	21.2	16.96 - 21.2	18	20	0.2
6	21.2	25.44	21.2 - 25.44	17	14	0.642857
7	25.44	29.68	25.44 - 29.68	7	9	0.444444
8	29.68	33.92	29.68 - 33.92	4	6	0.666667
9	33.92	38.16	33.92 - 38.16	4	4	0
10	38.16	42.4	38.16 - 42.4	1	3	1.333333
11	42.4	46.64	42.4 - 46.64	0	2	2
12	46.64	50.88	46.64 - 50.88	0	2	2
13	50.88	55.12	50.88 - 55.12	1	1	0
14	55.12	59.36	55.12 - 59.36	1	1	0
15	59.36	63.6	59.36 - 63.6	0	1	1
16	63.6	67.84	63.6 - 67.84	0	1	1
17	67.84	72.08	67.84 - 72.08	0	1	1
18	72.08	76.284	> 72.08	1	1	0
				300		15.0409

Appendix 1.2: Spreadsheet Data for C2 Inspection Time

number of points	300
number of bins	18
max value	114.426
min value	0.13
data range	114.296
bin width	6.349778
sample mean	15.5369
sample variance	215.6443

bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X_0^2 $\frac{(O_i - E_i)^2}{E_i}$
1	0	6.35	0 - 6.35	86	101	2.227723
2	6.35	12.7	6.35 - 12.7	76	67	1.208955
3	12.7	19.05	12.7 - 19.05	49	45	0.355556
4	19.05	25.4	19.05 - 25.4	35	30	0.833333
5	25.4	31.75	25.4 - 31.75	17	20	0.45
6	31.75	38.1	31.75 - 38.1	18	14	1.142857
7	38.1	44.45	38.1 - 44.45	7	9	0.444444
8	44.45	50.8	44.45 - 50.8	4	6	0.666667
9	50.8	57.15	50.8 - 57.15	4	4	0
10	57.15	63.5	57.15 - 63.5	1	3	1.333333
11	63.5	69.85	63.5 - 69.85	0	2	2
12	69.85	76.2	69.85 - 76.2	0	2	2
13	76.2	82.55	76.2 - 82.55	1	1	0
14	82.55	88.9	82.55 - 88.9	1	1	0
15	88.9	95.25	88.9 - 95.25	0	1	1
16	95.25	101.6	95.25 - 101.6	0	1	1
17	101.6	107.95	101.6 - 107.95	0	1	1
18	107.95	114.426	> 107.95	1	1	0
				300		15.66287

Appendix 1.3: Spreadsheet Data for C3 Inspection Time

number of points	300
number of bins	18
max value	104.019
min value	0.031
data range	103.988
bin width	5.777111
sample mean	20.63276
sample variance	394.3862

bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X_0^2 $\frac{(O_i - E_i)^2}{E_i}$
1	0	5.78	0 - 5.78	69	74	0.337838
2	5.78	11.56	5.78 - 11.56	55	56	0.017857
3	11.56	17.34	11.56 - 17.34	41	42	0.02381
4	17.34	23.12	17.34 - 23.12	37	32	0.78125
5	23.12	28.9	23.12 - 28.9	23	24	0.041667
6	28.9	34.68	28.9 - 34.68	21	19	0.210526
7	34.68	40.46	34.68 - 40.46	15	14	0.071429
8	40.46	46.24	40.46 - 46.24	7	11	1.454545
9	46.24	52.02	46.24 - 52.02	10	8	0.5
10	52.02	57.8	52.02 - 57.8	5	6	0.166667
11	57.8	63.58	57.8 - 63.58	3	5	0.8
12	63.58	69.36	63.58 - 69.36	3	4	0.25
13	69.36	75.14	69.36 - 75.14	2	3	0.333333
14	75.14	80.92	75.14 - 80.92	3	2	0.5
15	80.92	86.7	80.92 - 86.7	2	2	0
16	86.7	92.48	86.7 - 92.48	0	2	2
17	92.48	98.26	92.48 - 98.26	0	1	1
18	98.26	104.019	> 98.26	4	1	9
				300		17.48892

Appendix 1.4: Spreadsheet Data for W1 Assembly Time

number of points	300
number of bins	18
max value	29.375
min value	0.007
data range	29.368
bin width	1.631556
sample mean	4.604417
sample variance	22.62537

bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X$_0^2$ $\frac{(O_i - E_i)^2}{E_i}$
1	0	1.64	0 - 1.64	92	90	0.044444
2	1.64	3.28	1.64 - 3.28	70	63	0.777778
3	3.28	4.92	3.28 - 4.92	41	45	0.355556
4	4.92	6.56	4.92 - 6.56	22	31	2.612903
5	6.56	8.2	6.56 - 8.2	22	22	0
6	8.2	9.84	8.2 - 9.84	16	16	0
7	9.84	11.48	9.84 - 11.48	13	11	0.363636
8	11.48	13.12	11.48 - 13.12	5	8	1.125
9	13.12	14.76	13.12 - 14.76	4	6	0.666667
10	14.76	16.4	14.76 - 16.4	5	4	0.25
11	16.4	18.04	16.4 - 18.04	5	3	1.333333
12	18.04	19.68	18.04 - 19.68	0	2	2
13	19.68	21.32	19.68 - 21.32	0	2	2
14	21.32	22.96	21.32 - 22.96	1	1	0
15	22.96	24.6	22.96 - 24.6	2	1	1
16	24.6	26.24	24.6 - 26.24	0	1	1
17	26.24	27.88	26.24 - 27.88	1	1	0
18	27.88	29.375	> 27.88	1	1	0
				300		13.52932

Appendix 1.5: Spreadsheet Data for W2 Assembly Time

number of points	300
number of bins	17
max value	59.078
min value	0.091
data range	58.987
bin width	3.469824
sample mean	11.09261
sample variance	140.3376

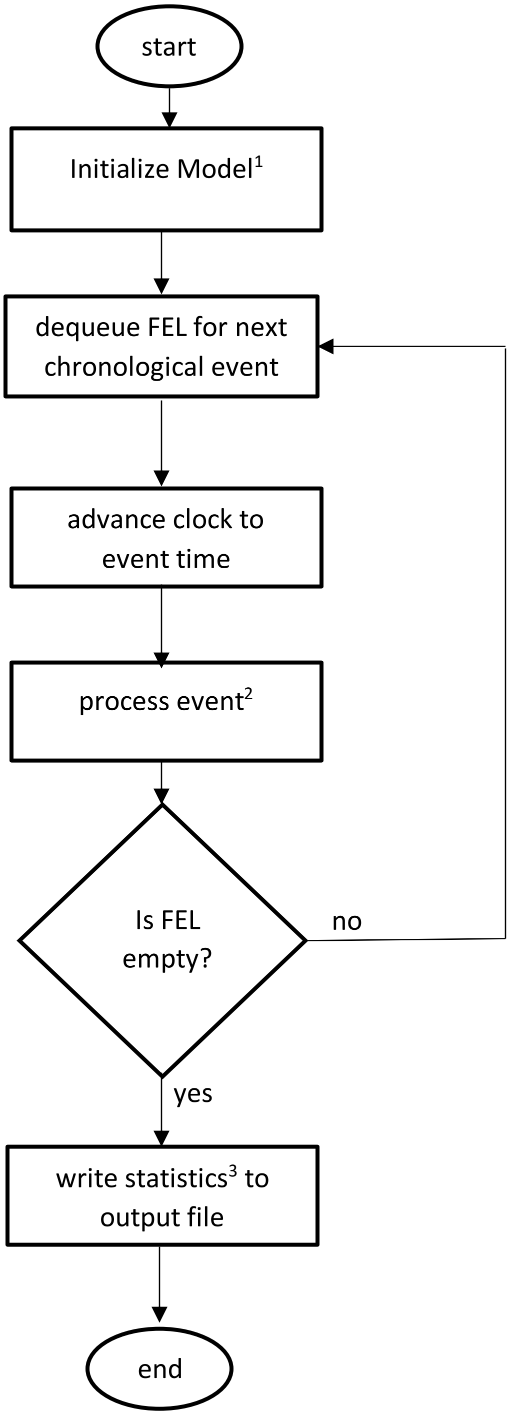
bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X_0^2 $\frac{(O_i - E_i)^2}{E_i}$
1	0	3.47	0 - 3.47	84	81	0.111111
2	3.47	6.94	3.47 - 6.94	63	59	0.271186
3	6.94	10.41	6.94 - 10.41	44	44	0
4	10.41	13.88	10.41 - 13.88	31	32	0.03125
5	13.88	17.35	13.88 - 17.35	22	24	0.166667
6	17.35	20.82	17.35 - 20.82	8	17	4.764706
7	20.82	24.29	20.82 - 24.29	6	13	3.769231
8	24.29	27.76	24.29 - 27.76	11	10	0.1
9	27.76	31.23	27.76 - 31.23	6	7	0.142857
10	31.23	34.7	31.23 - 34.7	4	5	0.2
11	34.7	38.17	34.7 - 38.17	5	4	0.25
12	38.17	41.64	38.17 - 41.64	5	3	1.333333
13	41.64	45.11	41.64 - 45.11	3	2	0.5
14	45.11	48.58	45.11 - 48.58	2	2	0
15	48.58	52.05	48.58 - 52.05	4	2	2
16	52.05	55.52	52.05 - 55.52	1	1	0
17	55.52	58.99	55.52 - 58.99	1	1	0
				300		13.64034

Appendix 1.6: Spreadsheet Data for W3 Assembly Time

number of points	300
number of bins	18
max value	51.418
min value	0.102
data range	51.316
bin width	2.850889
sample mean	8.79558
sample variance	74.82545

bin	bin lower range	bin upper range	range	observed frequency O_i	expected frequency E_i	X_0^2 $\frac{(O_i - E_i)^2}{E_i}$
1	0	2.86	0 - 2.86	77	84	0.583333
2	2.86	5.72	2.86 - 5.72	72	61	1.983607
3	5.72	8.58	5.72 - 8.58	44	44	0
4	8.58	11.44	8.58 - 11.44	29	32	0.28125
5	11.44	14.3	11.44 - 14.3	15	23	2.782609
6	14.3	17.16	14.3 - 17.16	17	17	0
7	17.16	20.02	17.16 - 20.02	13	12	0.083333
8	20.02	22.88	20.02 - 22.88	11	9	0.444444
9	22.88	25.74	22.88 - 25.74	6	7	0.142857
10	25.74	28.6	25.74 - 28.6	2	5	1.8
11	28.6	31.46	28.6 - 31.46	5	4	0.25
12	31.46	34.32	31.46 - 34.32	2	3	0.333333
13	34.32	37.18	34.32 - 37.18	4	2	2
14	37.18	40.04	37.18 - 40.04	0	2	2
15	40.04	42.9	40.04 - 42.9	1	1	0
16	42.9	45.76	42.9 - 45.76	1	1	0
17	45.76	48.62	45.76 - 48.62	0	1	1
18	48.62	51.418	> 48.62	1	1	0
				300		13.68477

Appendix 2.1: Flowchart for Progression of One Replication

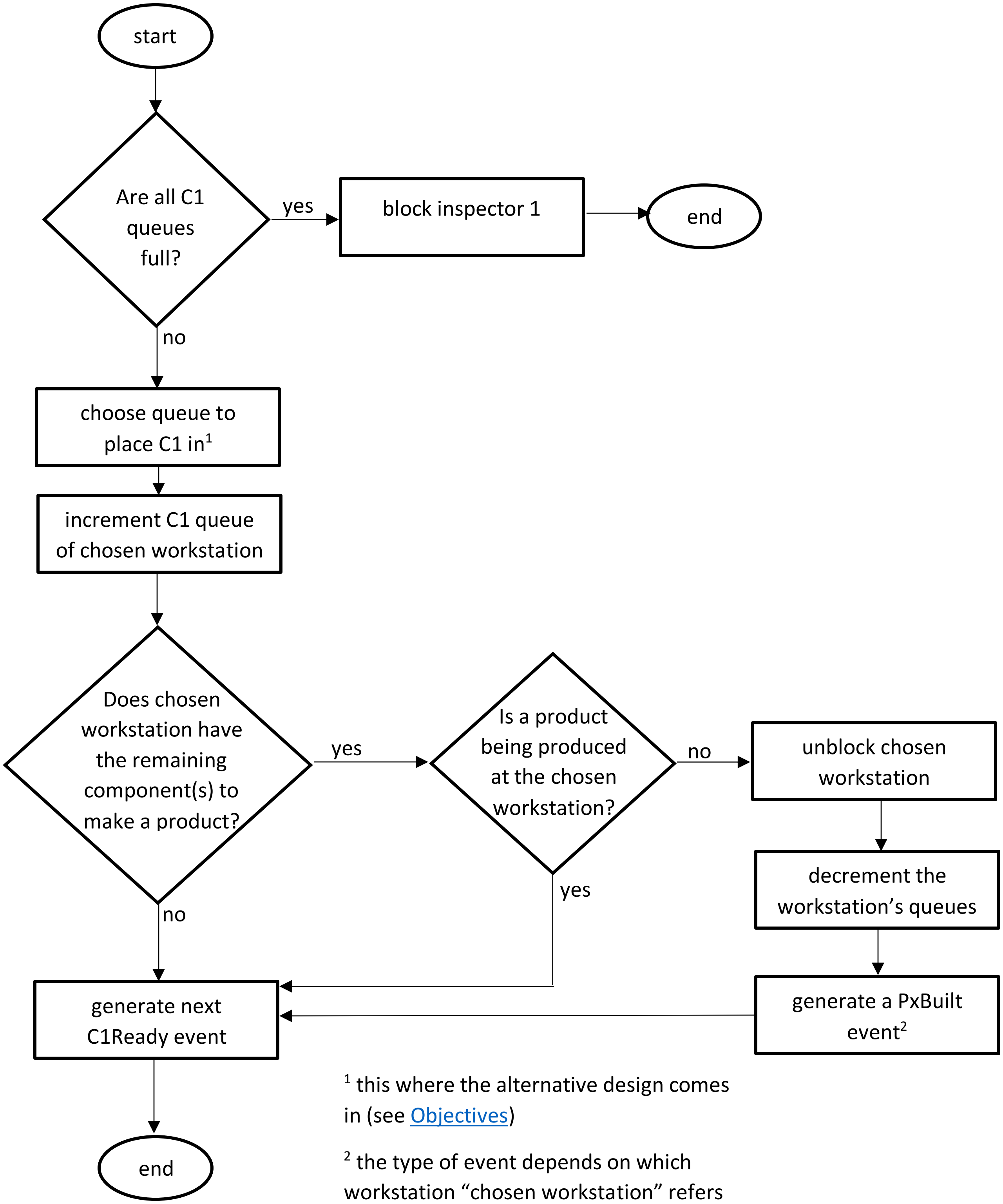

start
Initialize Model[1]
dequeue FEL for next chronological event
advance clock to event time
process event[2]
Is FEL empty?
no
yes
write statistics[3] to output file
end

[1] initializing model includes creating distributions, creating the future event list (FEL) with initial event(s), and setting default value for statistics

[2] dependent on type of event, elaborated on in next 6 flowcharts

[3] statistics of interest are simulation time, number of each product produced, and the amount of time each inspector/ workstation spends idle

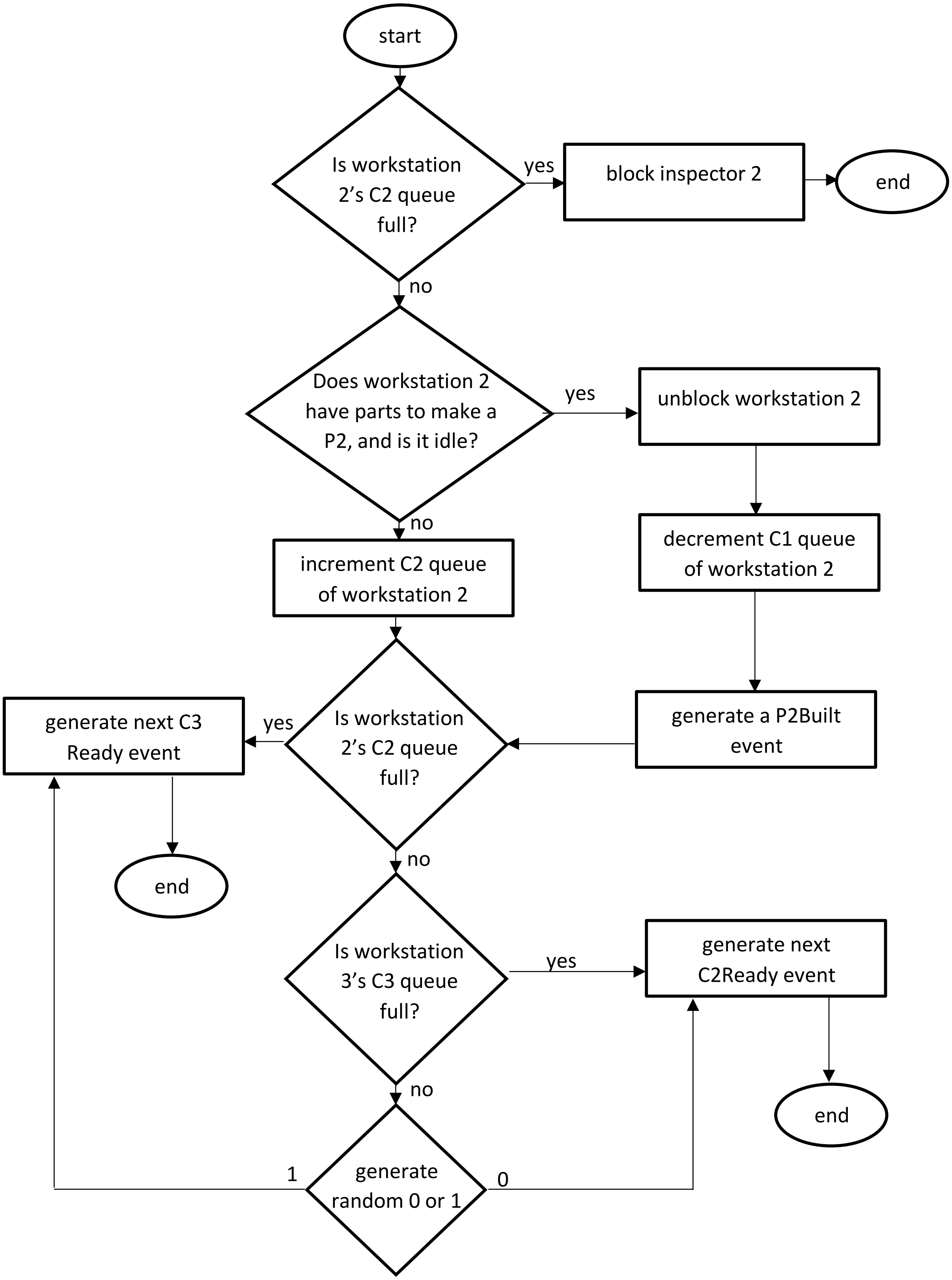
Appendix 2.2: Flowchart for C1Ready Event


start
Are all C1 queues full?
yes
block inspector 1
end
no
choose queue to place C1 in[1]
increment C1 queue of chosen workstation
Does chosen workstation have the remaining component(s) to make a product?
yes
Is a product being produced at the chosen workstation?
no
unblock chosen workstation
decrement the workstation's queues
generate a PxBuilt event[2]
yes
no
generate next C1Ready event
end

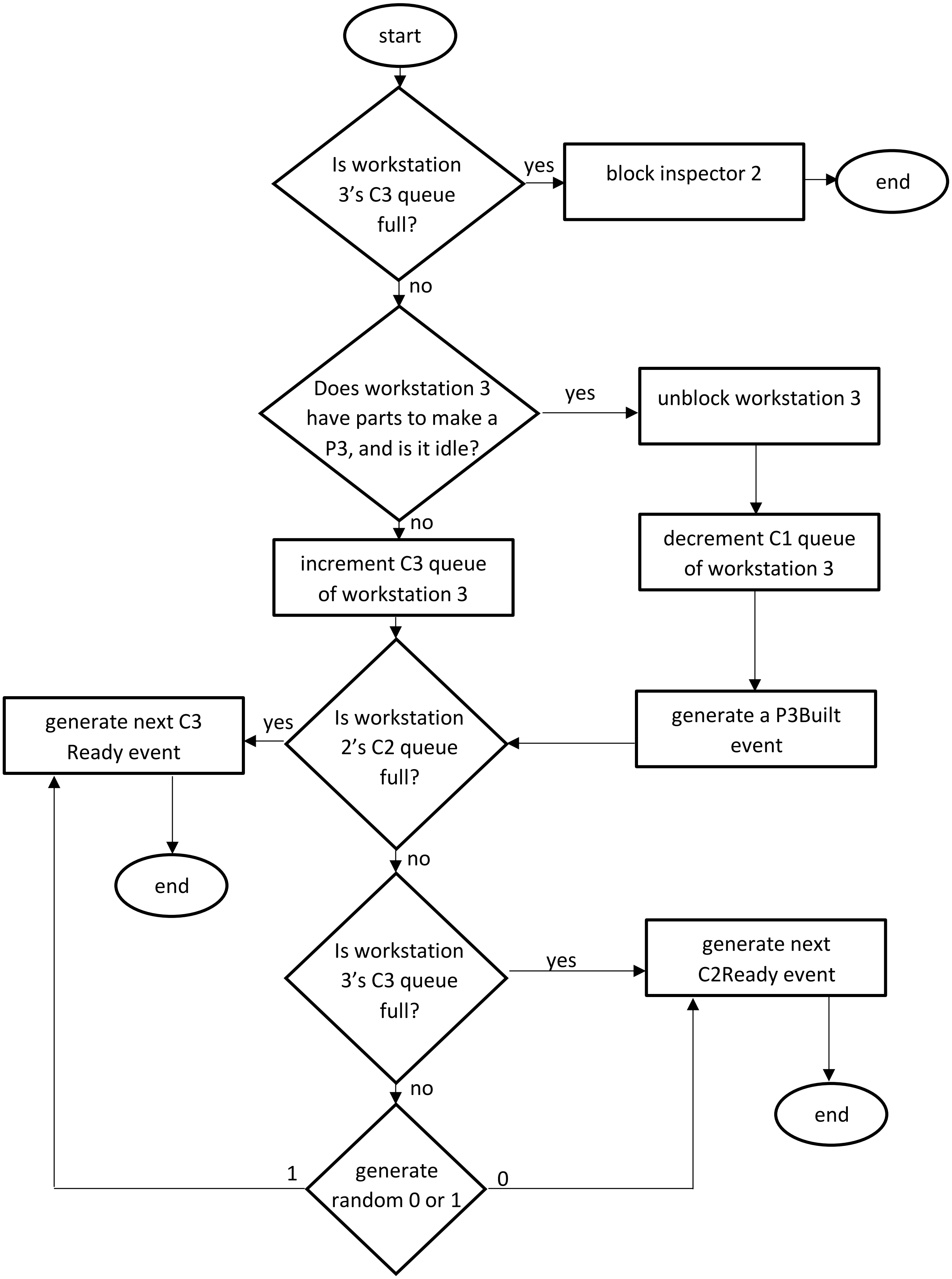
[1] this where the alternative design comes in (see Objectives)

[2] the type of event depends on which workstation "chosen workstation" refers to (P1Built for workstation 1, P2Built for workstation 2, P3Built for workstation 3)

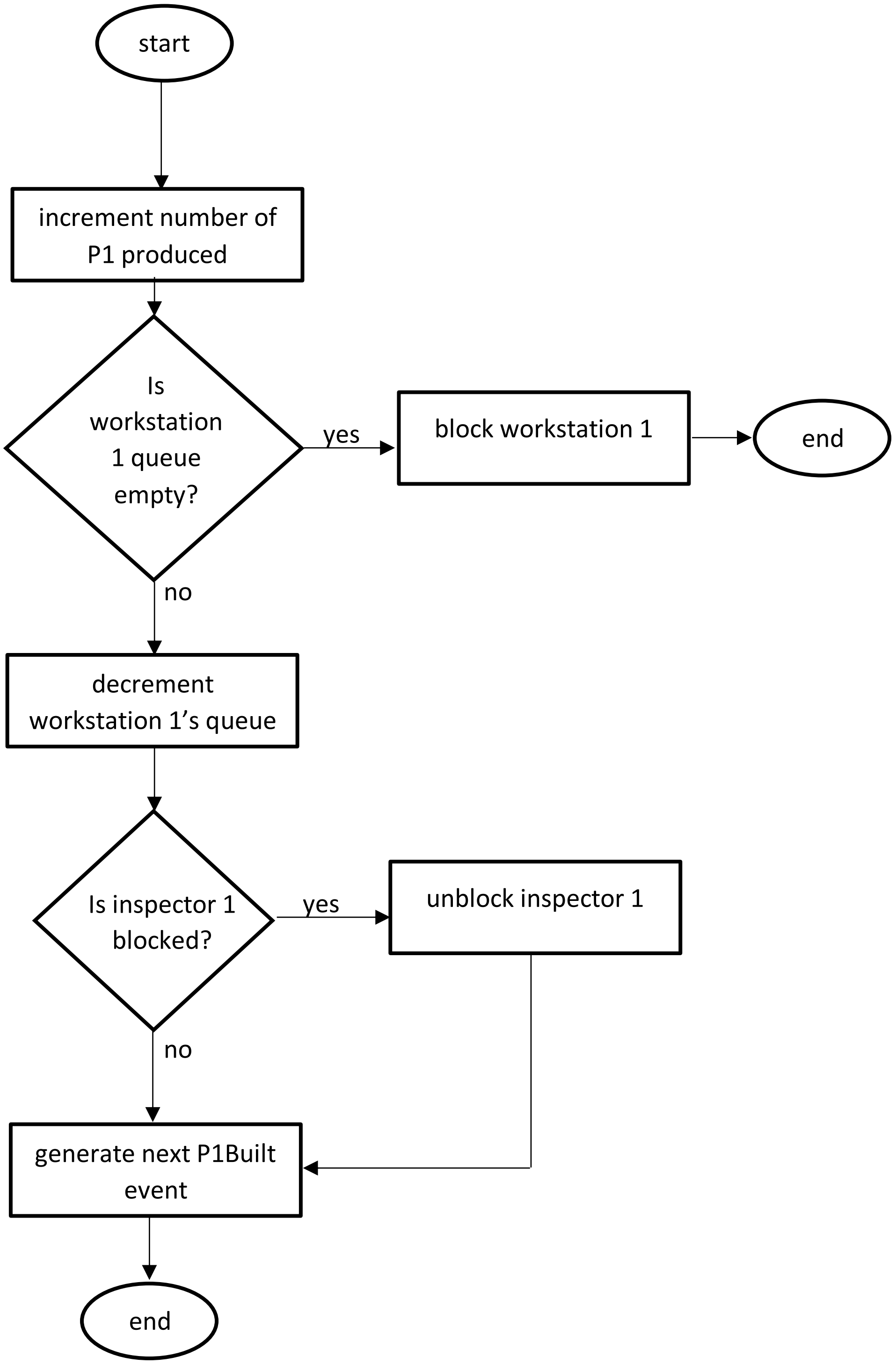
Appendix 2.3: Flowchart for C2Ready Event


start
Is workstation 2's C2 queue full?
yes
block inspector 2
end
no
Does workstation 2 have parts to make a P2, and is it idle?
yes
unblock workstation 2
decrement C1 queue of workstation 2
generate a P2Built event
no
increment C2 queue of workstation 2
Is workstation 2's C2 queue full?
yes
generate next C3 Ready event
end
no
Is workstation 3's C3 queue full?
yes
generate next C2Ready event
end
no
generate random 0 or 1
1
0

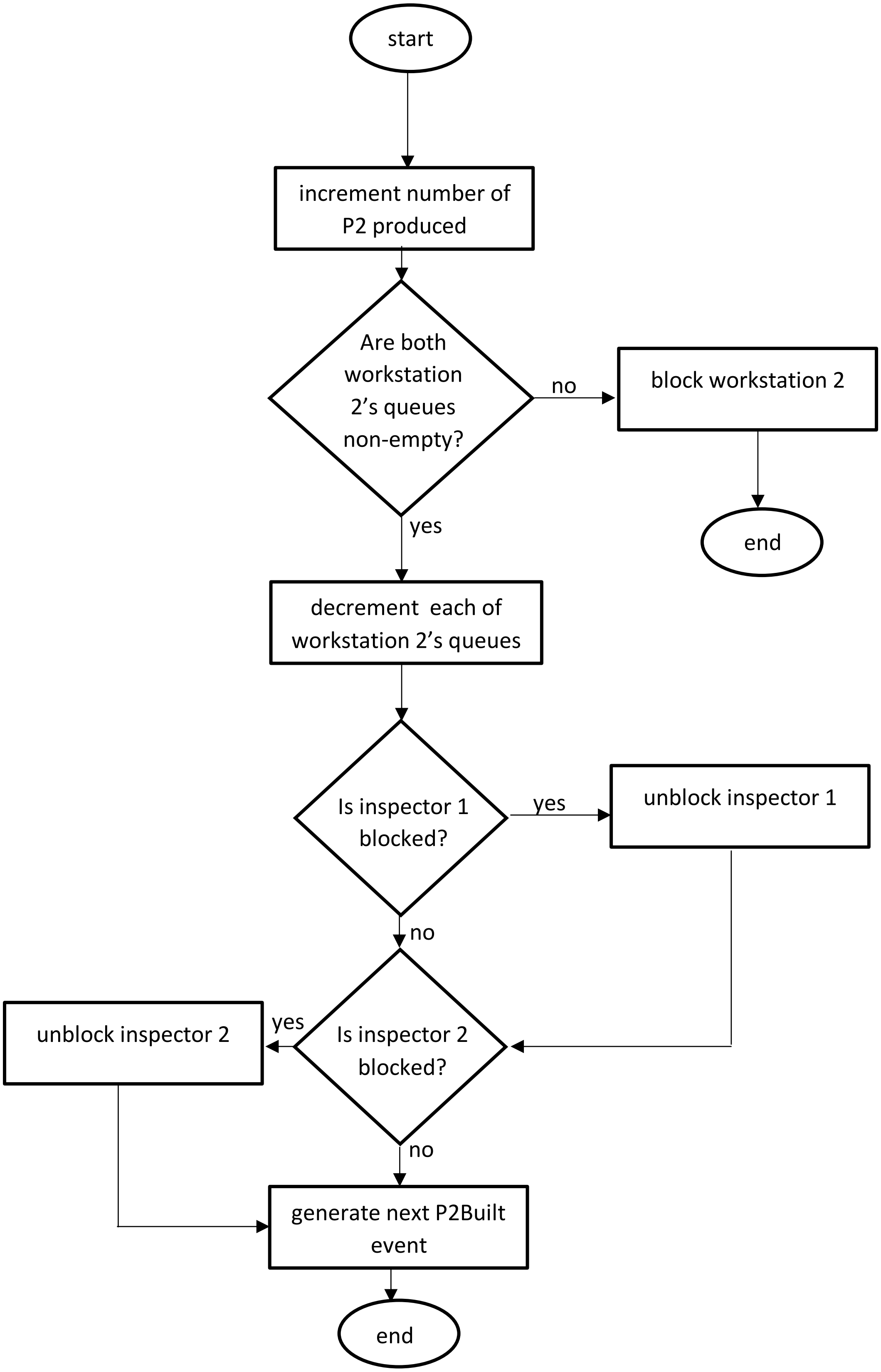
Appendix 2.4: Flowchart for C3Ready Event


start
Is workstation 3's C3 queue full?
yes
block inspector 2
end
no
Does workstation 3 have parts to make a P3, and is it idle?
yes
unblock workstation 3
decrement C1 queue of workstation 3
generate a P3Built event
no
increment C3 queue of workstation 3
Is workstation 2's C2 queue full?
yes
generate next C3 Ready event
end
no
Is workstation 3's C3 queue full?
yes
generate next C2Ready event
end
no
generate random 0 or 1
1
0

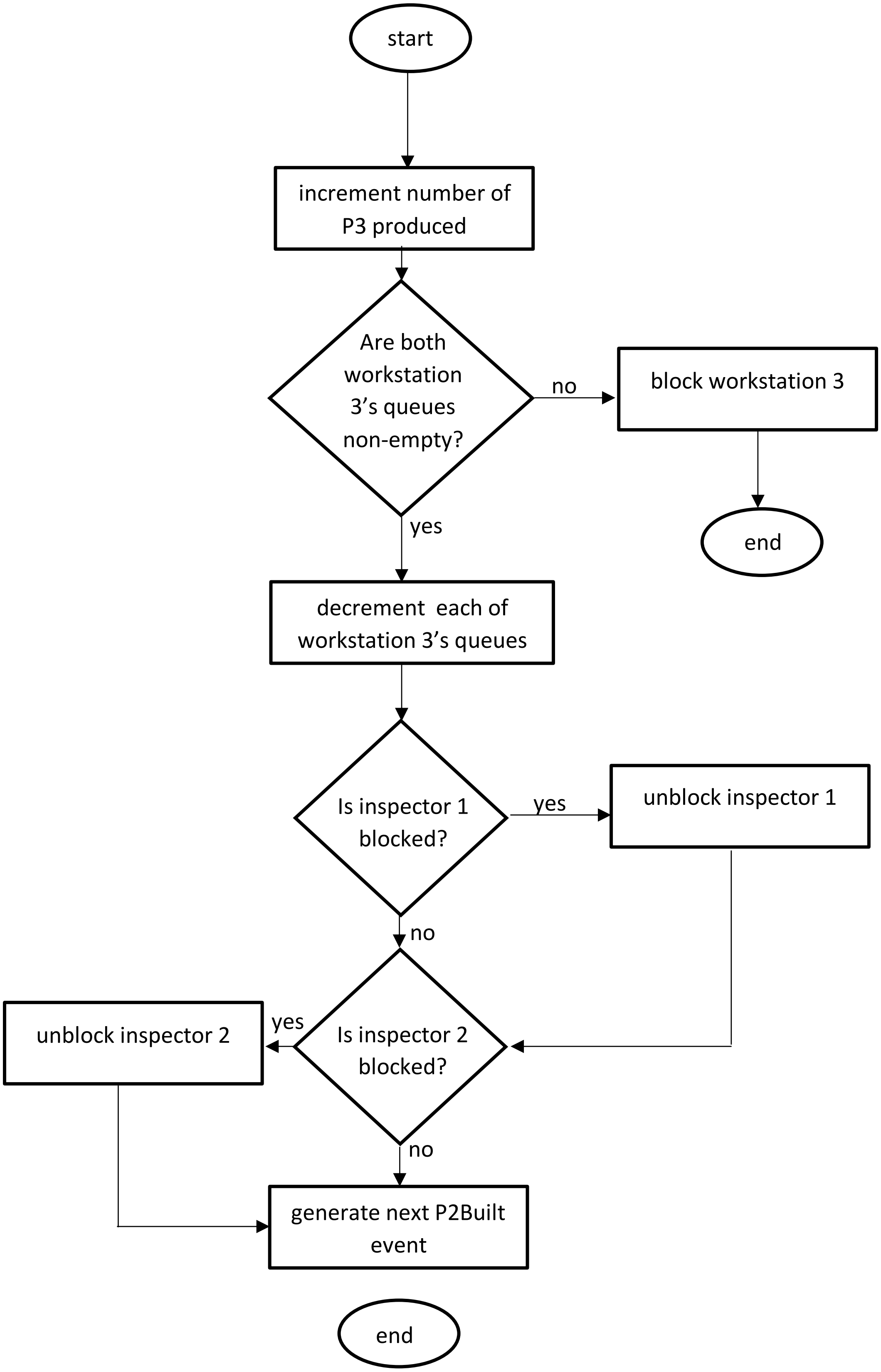
Appendix 2.5: Flowchart for P1Built Event


start
increment number of P1 produced
Is workstation 1 queue empty?
yes
block workstation 1
end
no
decrement workstation 1’s queue
Is inspector 1 blocked?
yes
unblock inspector 1
no
generate next P1Built event
end

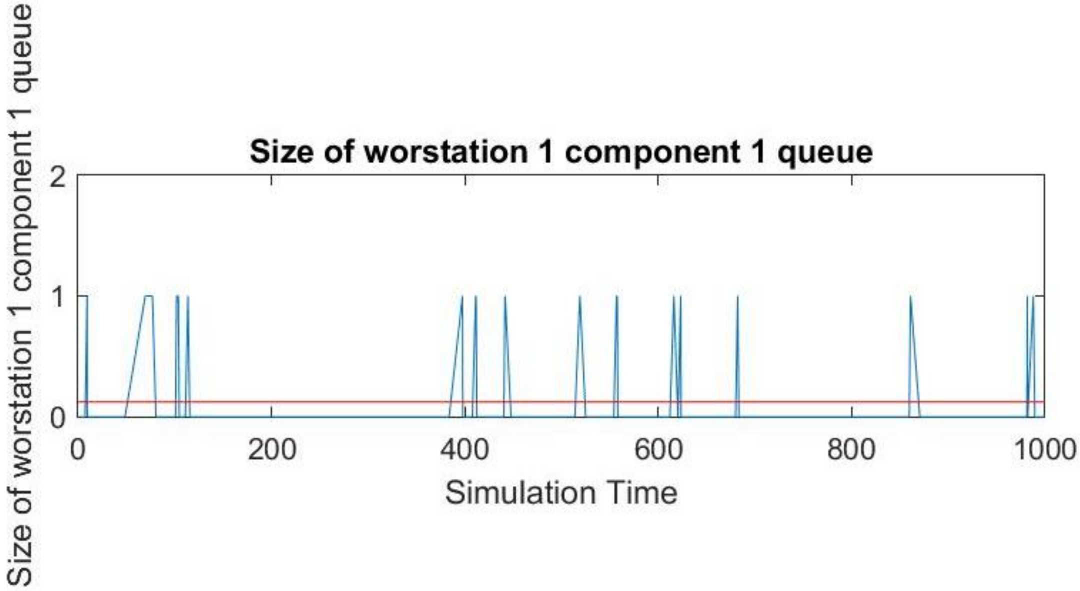
Appendix 2.5: Flowchart for P2Built Event


start
increment number of P2 produced
Are both workstation 2's queues non-empty?
no
block workstation 2
end
yes
decrement each of workstation 2's queues
Is inspector 1 blocked?
yes
unblock inspector 1
no
Is inspector 2 blocked?
yes
unblock inspector 2
no
generate next P2Built event
end

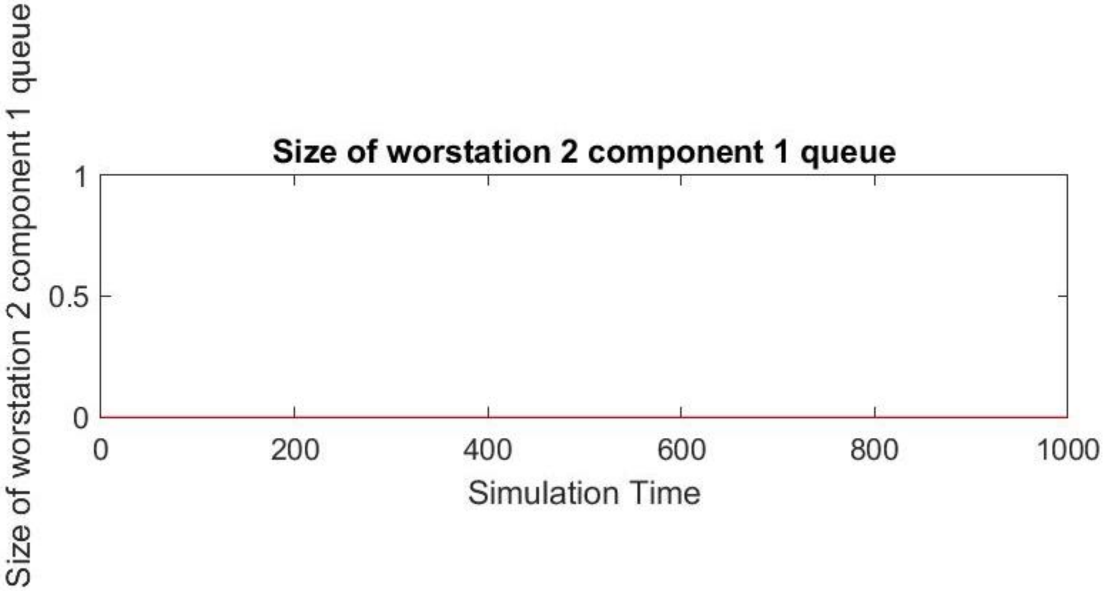
Appendix 2.6: Flowchart for P3Built Event


start
increment number of P3 produced
Are both workstation 3's queues non-empty?
no
block workstation 3
end
yes
decrement each of workstation 3's queues
Is inspector 1 blocked?
yes
unblock inspector 1
no
Is inspector 2 blocked?
yes
unblock inspector 2
no
generate next P2Built event
end

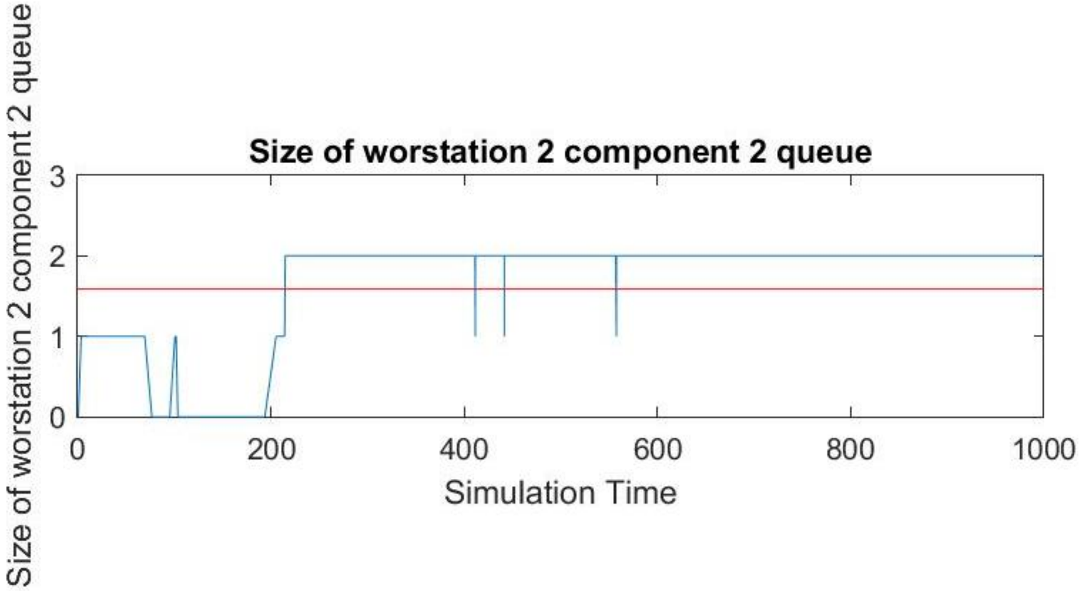
Appendix 3.1: Size of Workstation 1's Component 1 Queue over one Replication



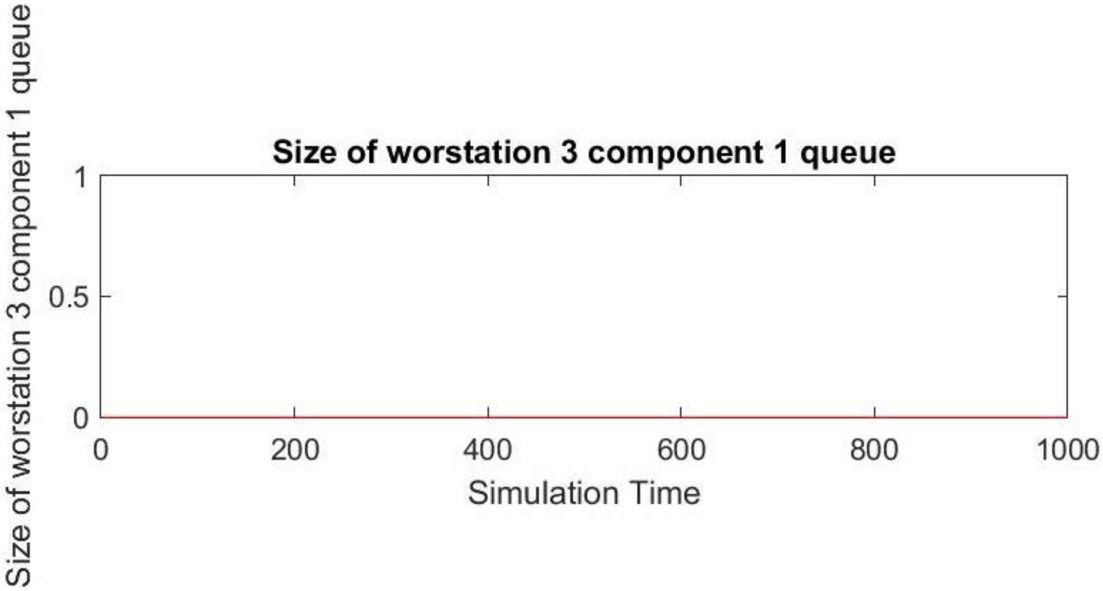
Appendix 3.2: Size of Workstation 2's Component 1 Queue over one Replication



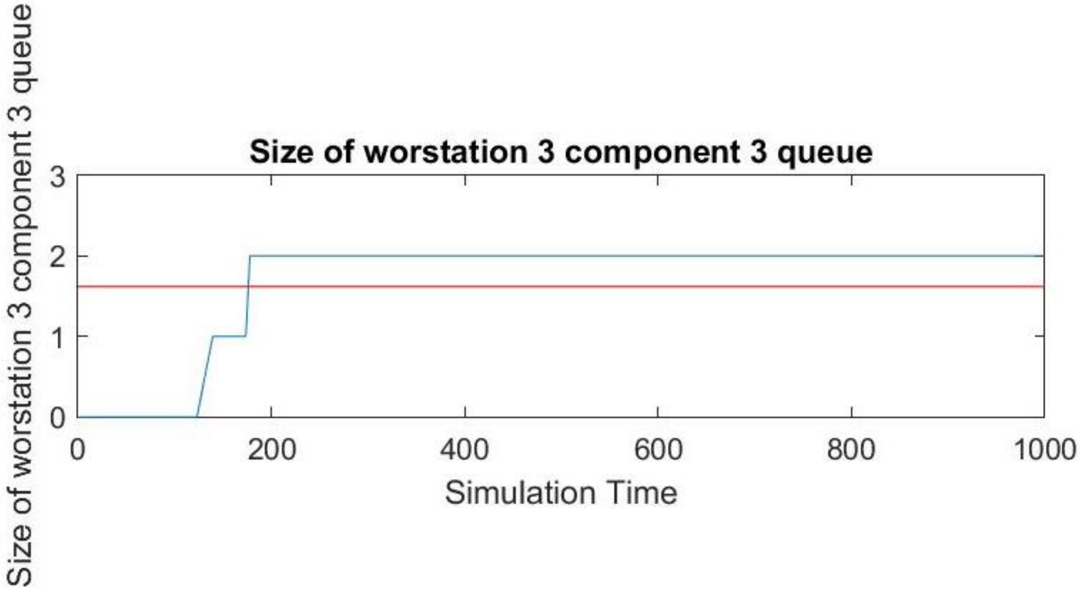
Appendix 3.3: Size of Workstation 2’s Component 2 Queue over one Replication



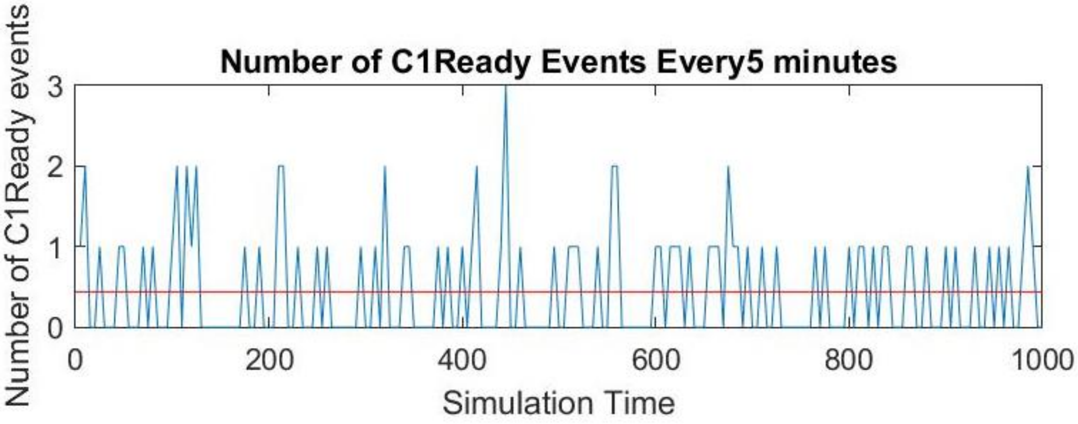
Appendix 3.4: Size of Workstation 3’s Component 1 Queue over one Replication



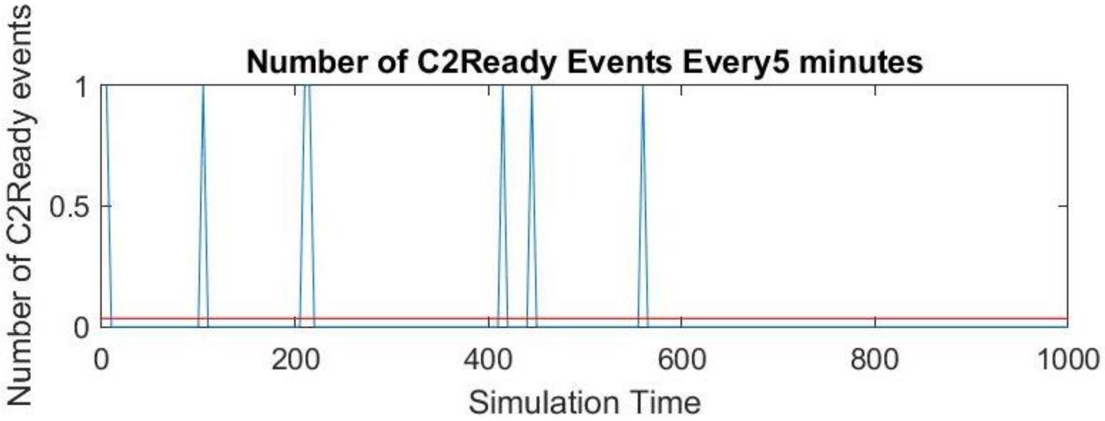
Appendix 3.5: Size of Workstation 3's Component 3 Queue over one Replication



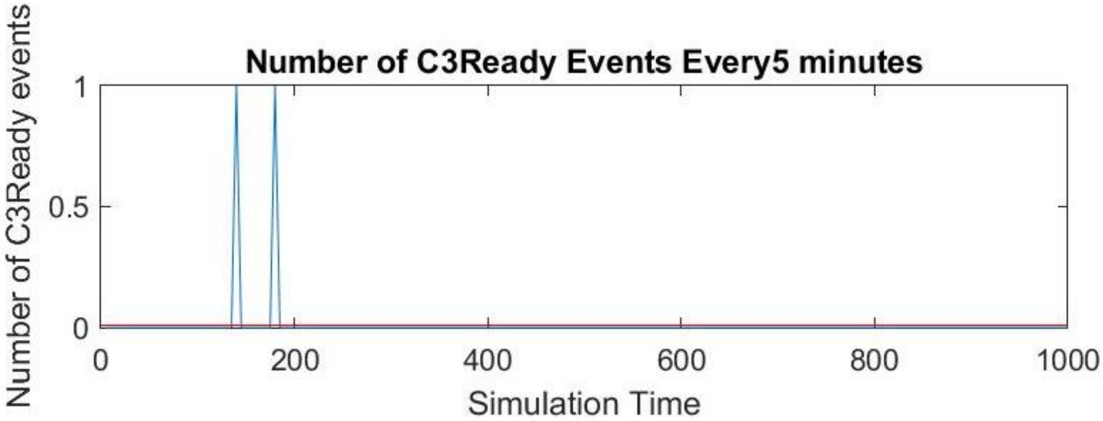
Appendix 3.6: Number of C1Ready Events Every 5 Minutes over one Replication



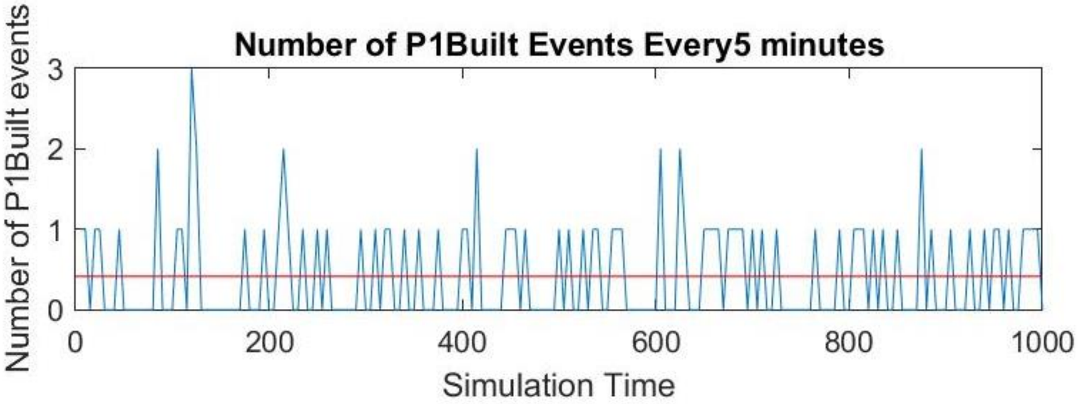
Appendix 3.7: Number of C2Ready Events Every 5 Minutes over one Replication



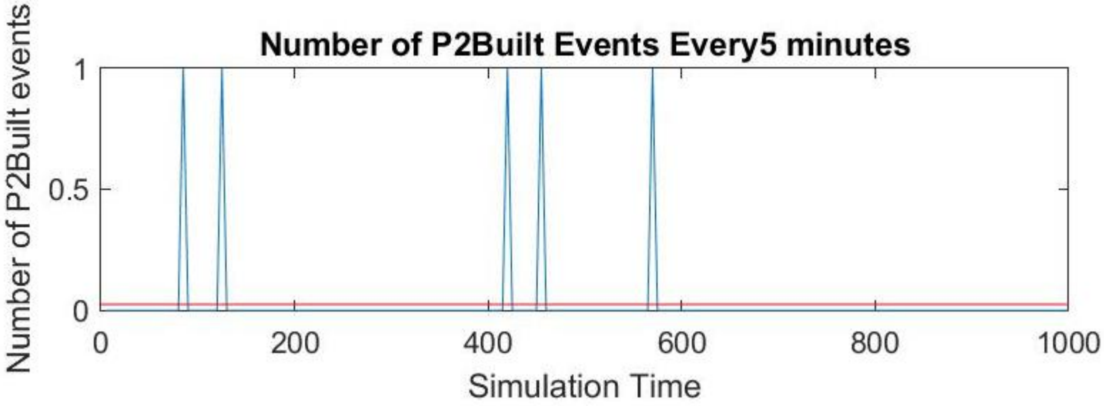
Appendix 3.8: Number of C3Ready Events Every 5 Minutes over one Replication



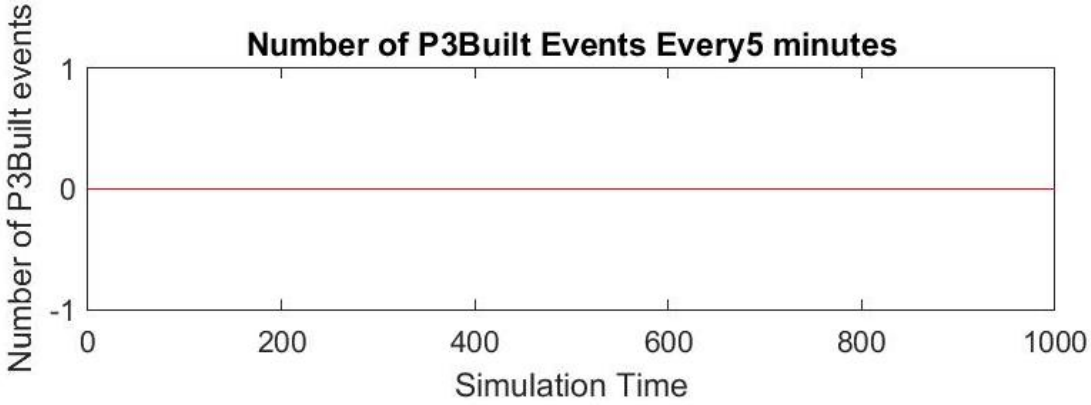
Appendix 3.9: Number of P1Built Events Every 5 Minutes over one Replication



Appendix 3.10: Number of P2Built Events Every 5 Minutes over one Replication



Appendix 3.11: Number of P3Built Events Every 5 Minutes over one Replication


Number of P3Built Events Every5 minutes
Number of P3Built events
Simulation Time