SYSC 4005 Project

Deliverable 2

March 11, 2022

Group 12

Ben Baggs 101122318

Erica Oliver 101114313

Wintana Yosief 101102204

1. Project Formulation

The project involves a manufacturing facility that assembles three types of products, P1, P2, and P3 that consist of one or more types of components. There are three types of components, C1, C2, and C3. Product P1 has only one component C1, Product P2 has components C1 and C2, and Product P3 has components C1 and C3.

Two inspectors are responsible for the cleaning and repairing of the components. Inspector 1 only works on C1 components, whereas Inspector 2 works on C2 and C3 components in random order. There is an infinite supply of components available, therefore the inspectors will not have to wait for the components.

The manufacturing facility also has three workstations, W1, W2, and W3 that assemble products P1, P2, and P3 respectively. Once the components pass inspection, the components are sent to the appropriate workstation. The workstations have two buffers available for each component type required, each with a buffer capacity of two components. For a product to begin the process of being assembled, all required component types must be available. In the case that all workstation buffers for a specific component type are full, the inspector that had completed the inspection of a component of that type is considered "blocked" until there is an available buffer.

Currently, Inspector 1 sends C1 components to the buffer with the least number of components in waiting. If there is a tie, the workstation, W1 has the highest priority, and workstation, W2 has the lowest priority.

2. Setting of Objectives and Overall Project Plan

The objective of this project is to assess the performance of the manufacturing facility and attempt to improve the policy that Inspector 1 follows to direct Component 1 in order to increase throughput and/or decrease the time the inspectors are blocked.

The project shall consist of four deliverables. The first deliverable (this one) contains the project formulation, setting of objectives and overall project plan, the model conceptualization, and the model translation. In the second deliverable, data will be collected and modelled, histograms representing that data will be provided, and the report will explain how the input is generated. The third deliverable will be the verification and validation process and an analysis of production. The fourth and final deliverable will include an alternative operating policy, the project conclusion, and a final report. Due dates have yet to be set for deliverables 2-4.

3. Model Conceptualization

There are four entity types in the simulation. These entities are as follows:

Inspector: This represents an individual inspector, and can either be blocked or unblocked at any point during the simulation.

Buffer: This represents an individual buffer associated with a specific inspector, component type, and workstation. This buffer can have either be empty or filled to a capacity of 1 or 2.

Workstation: This represents an individual workstation. Each workstation takes components from associated buffers and assembles a specified product.

Component: This represents the type of an individual component.

There are five events in the simulation model. See the submitted file "Event_Flowcharts.pdf" for a diagram representing each event that occurs in the system. These events are as follows:

Buffer Fill Event (BFE): This event represents an inspector placing a component into a buffer. The parameters of this event are time, and a buffer. No other entities need to be specified since each individual buffer only interacts with a single inspector and workstation, and only ever holds one type of component. Therefore, these entities can all be inferred during the simulation. This event assumes that the buffer is not already full and will only be created when that condition is true. Whenever this event occurs, the capacity of this buffer will be increased by one. If there is a workstation that is not busy that now has the required components to assemble a product, then a Begin Assembly Event will be created to take place immediately for that workstation. At the end of every Buffer Fill Event, a Begin Inspection Event will be created to take place immediately for the inspector associated with the buffer that was just filled, as that inspector will now be free to start inspecting another component.

Begin Inspection Event (BIE): This event represents an inspector beginning to inspect a new component. The parameters of this event are time, an inspector, and a component. When this event occurs, an inspection time will be generated, and a Finish Inspection Event will be created after this inspection time, with the same inspector and component.

Finish Inspection Event (FIE): This event represents an inspector finishing inspecting a component. The parameters of this event are time, an inspector, and a component. When this event occurs, the capacity of the buffer associated with the specified inspector and component will be checked. If the buffer is not full, then a Buffer Fill Event will be created to take place immediately for that buffer. If it is full, then the inspector will become blocked.

Begin Assembly Event (BAE): This event represents a workstation taking components from its buffers and beginning to assemble a product. The parameters of this event are time, and a workstation. This event assumes that the components required to assemble a product are present in the buffers associated with the specified workstation and will only be created when that condition is true. When this event occurs, the workstation will take components from its associated buffers, generate an assembly time, mark itself as busy, and then create a Finish Assembly Event after this assembly time and with the same workstation. Because there are now open spaces in the buffers, any inspectors who were blocked and waiting to put components in the buffers that were just drawn from are unblocked. In such a case, a Buffer Fill Event is created to take place immediately, which represents the inspector placing a component in the newly free buffer.

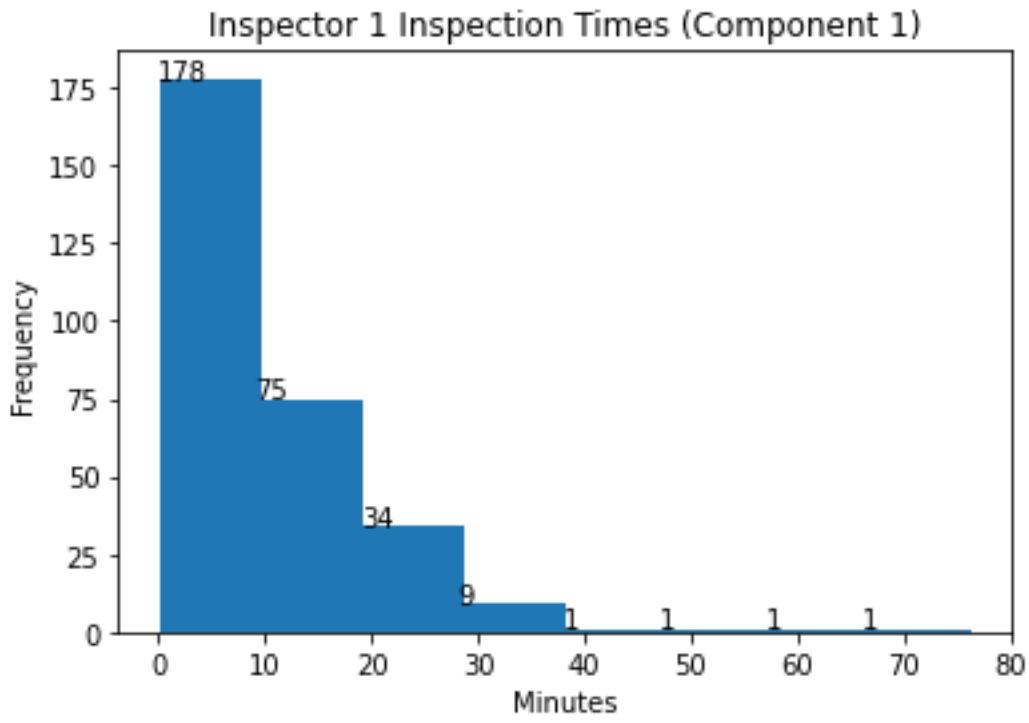
Finish Assembly Event (FAE): This event represents a workstation finishing the assembly of a product. The parameters of this event are time, and a workstation. When this event occurs, the buffers associated with the specified workstation will be checked to see whether they contain the components required to assemble a new product. If so, then a Begin Assembly Event will be created to take place immediately for the specified workstation.

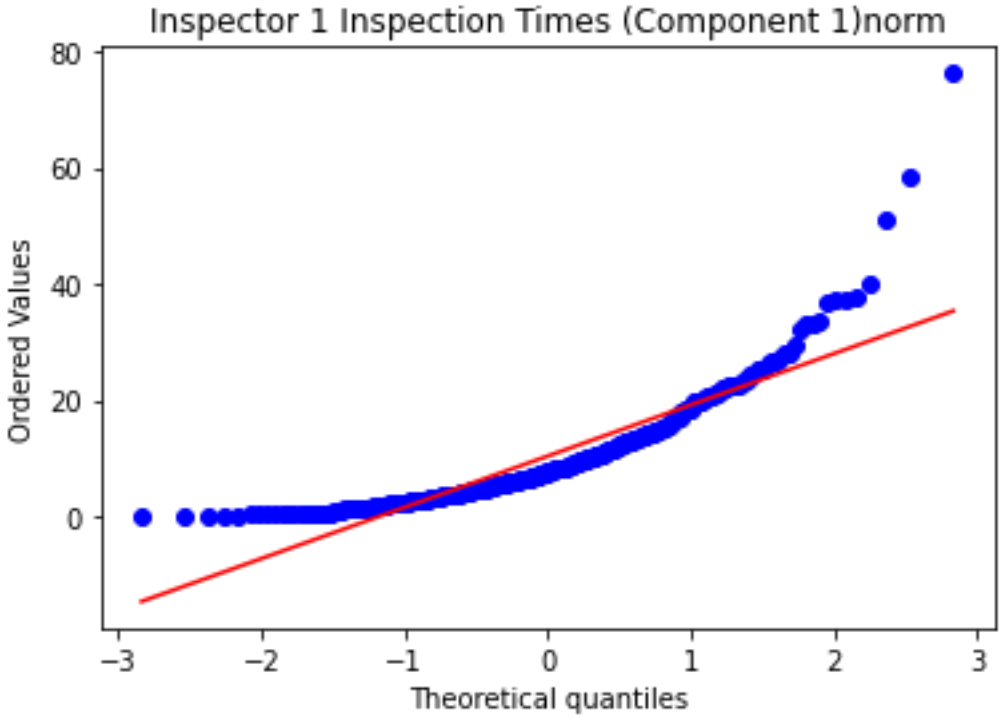
The time value associated with each event represents the time at which it occurs during the simulation.

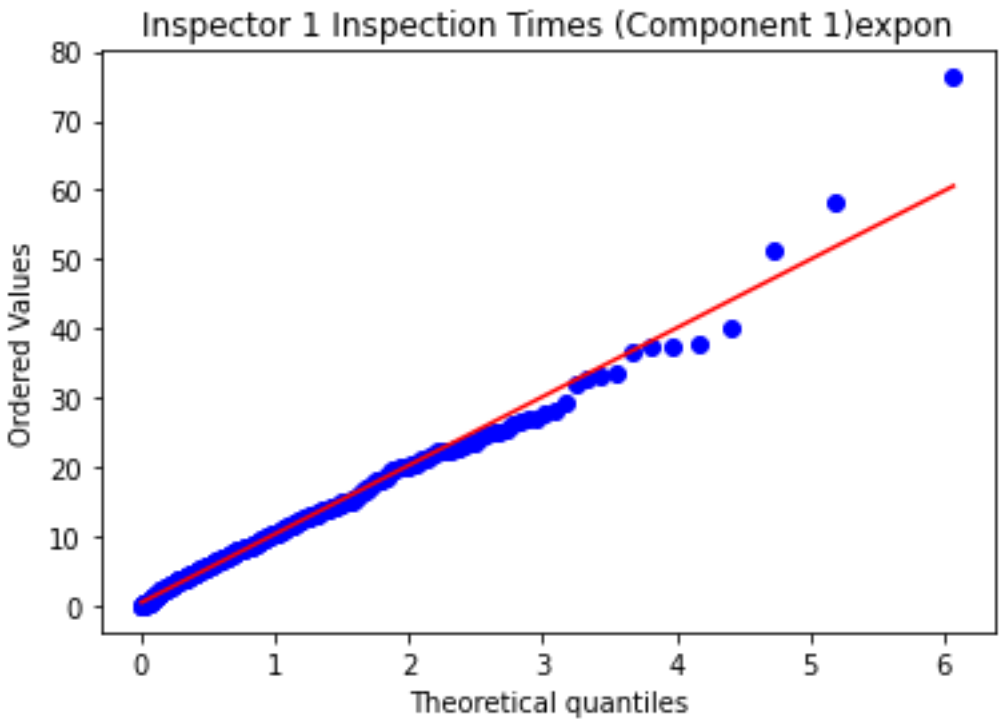
4. Data Collection and Input Modelling

Data was collected before the beginning of the project and provided as six ".dat" files. Each file contains a list of historical data representing either inspection time for an inspector or processing time for a workstation measured in minutes. Histograms of this data were plotted, as well as Q-Q plots, and a chi-square goodness-of fit test was performed to figure out if the data represents a normal or exponential distribution. From these diagrams, we can see that all six datasets represent an exponential distributions. The MLE estimator for the lambda parameter of an exponential distribution is equal to the inverse of the mean of the dataset. To confirm this, we conduct the chi-squared test comparing each dataset with an exponential distribution with the corresponding lambda value, and since all the calculations are below the threshold at the 0.05 significance level, we can conclude that the distributions are exponential with lambda values equal to the inverses of the mean of the datasets. These calculations can be found in "chi_square_tests.xlsx".

The diagrams below show the histogram and Q-Q diagrams for normal and exponential distributions for Component 1 only. The diagrams for the other components and the products can be found in the "diagrams" folder.


Inspector 1 Inspection Times (Component 1)
Frequency
Minutes
178
75
34
9
1
1
1
1
0
25
50
100
125
150
175
0
10
20
30
40
50
60
70
80


Inspector 1 Inspection Times (Component 1)norm
Ordered Values
Theoretical quantiles
80
60
40
20
0
−3
−2
−1
0
1
2
3



To generate input, the Linear Congruential Method will be used to generate random numbers then the inverse transform technique will be used to convert them into an exponential distribution. The Linear Congruential Method generates a sequence of integers, X_1, X_2,… between 0 and m-1 using the following recursive relationship: $X_{i+1} = (aX_i + c)\ mod\ m$, i = 0,1,2… where a is the multiplier, c is the increment, and m is the modulus.

The a, c, initial x, and m values chosen for the linear congruential generator are 289, 321, 65536, and 0 respectively. This is because 289 and 321 are coprime and 65536 is a power of two.

5. Model Translation

See the submitted file “sim.py” for the source code of the simulation. Python was used to implement the simulation. It was chosen because every member of the project group was experienced in using it, in addition to there being several existing libraries that may be useful in the implementation of future deliverables.

There are four classes that represent entities in the simulation. These are “Inspector”, “Component”, “Buffer”, and “Workstation”. Since each individual buffer in the simulation will only receive components from one inspector and only be used by one workstation, each buffer instance stores a reference to a single inspector and a single workstation. The instances of these entity classes and the associations between them are hardcoded and created before the simulation runs.

The five event types are also implemented as classes, with each containing two commonly named functions, “desc()” and “execute()”. These two functions return a description of the individual event instance, and perform the actions of the event respectively. Each event instance contains an associated time value in minutes, as well as references to the entities that it involves.

The future event list (FEL) is implemented as a list containing instances of the five event classes. The FEL is sorted so that events with the earliest time values are at the front of the list. Whenever a new event is placed into the FEL, it will be placed in the appropriate position based on its associated time value.

The inspection and product assembly times included in the provided files are loaded and stored as tables of numbers. During the simulation, these numbers will be retrieved sequentially and one at a time whenever they need to be “generated” for an event. When the end of a table is reached, the first number will be reused and the whole table accessed sequentially again.

Upon running the simulation Python file, the user will be prompted to enter the time that the simulation will run for in minutes. The entered value must be a positive integer value. The simulation will then begin immediately.

Once the simulation begins, the first event in the FEL will be removed and a description of it will be printed to the simulation output file. The event will then be executed, modifying the entities and creating new events to put in the FEL as necessary. Once an event is executed, the FEL will take the next event at the front of the list and repeat the steps described above. This will continue until either the FEL is empty, or the time values of events exceed the stopping time defined by the user. Because inspectors have an infinite inventory of available components, new events will always be created and the FEL will never be emptied. Therefore, the simulation will only ever end once the stopping time is reached.

The user will be shown a message to indicate when the simulation is finished. Some statistics gathered from the simulation will also be displayed and recorded in the simulation output file. A complete record of the simulation will be stored in the simulation output file, named “simulation_output.txt”.