SYSC 4005 A

Winter 2023

Project Deliverable #1

Completed By:

Favour Olotu	101130753
Ray Agha	101108060
Joseph Anyia	101117261

Due: February 10, 2023

1. Problem Formulation

A simulation study is to be conducted in order to assess the performance of this manufacturing facility, partly based on observed historical data of the inspectors' and workstations' service times given in units of minutes. An additional objective is to possibly improve the policy that Inspector 1 follows when delivering C1 components to the different workstations, in order to increase throughput and/or decrease the inspectors “blocked" time.

The manufacturing process creates 3 products which shall be referred to as P1, P2, and P3. The products are created on corresponding workstations referred to as W1, W2, W3 with components referred to as C1, C2, and C3. The product composition is as follows: · P1 is composed of one C1 · P2 is composed of one C1 and one C2 · P3 is composed of one C1 and one C3 Creating products involves two inspectors, referred to as I1 and I2, that inspect, clean, and repair components before they are sent to workstation buffers. The supply of components is infinite, obtaining a component to inspect happens instantaneously, however, it takes a period of time to complete an inspection, and this time period will follow a specified distribution.

Inspector I1 is responsible for inspecting component C1, and inspector I2 is responsible for inspecting components C2 and C3. For now, inspector I2 will randomly choose to select either C2 or C3 for inspection without any consideration for other production conditions. Once an inspection is complete, the components are sent to one of the three workstations W1, W2, and W3, which each assemble products P1, P2, P3 respectively.

Each workstation has a buffer with a capacity of two components for each of the components it requires to assemble its respective product and removes its needed components from the buffers to begin assembly. A workstation can only assemble one product at a time, and assembly may begin as soon as all of its component buffers have a component to take. Each workstation will have assembly times that will follow a specified workstation-specific distribution. When an inspector completes an inspection, the inspector will try to place the component on the workstation buffer if it has an opening for that component type. If all buffers have an equal amount of freespace, the buffer corresponding to the lowest workstation number will be chosen.

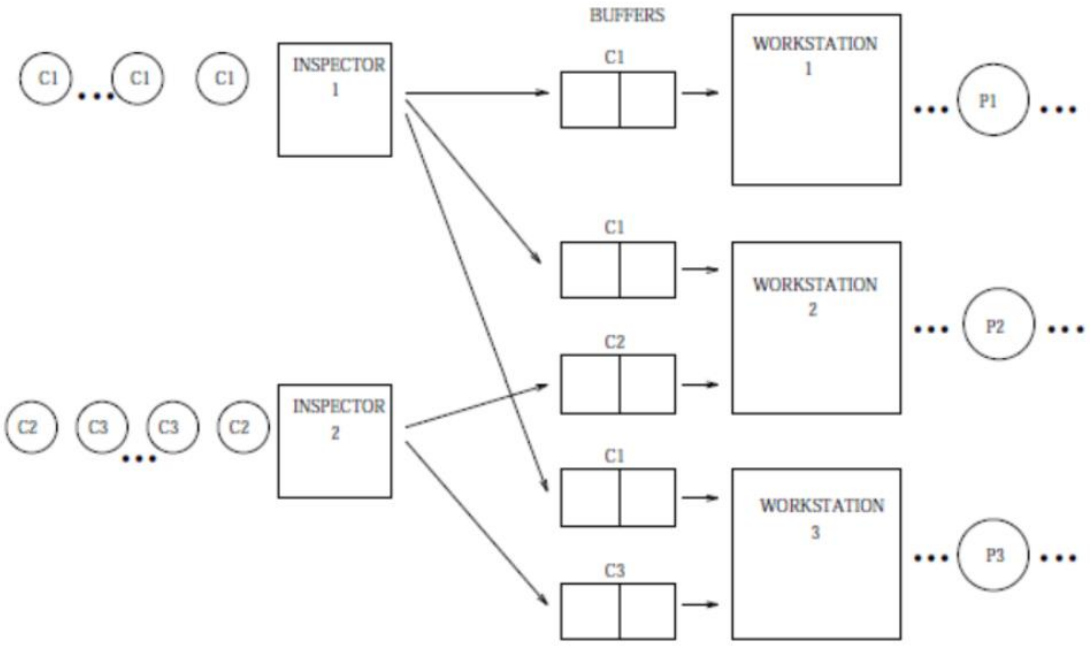

Figure 1: Schematic illustration of the manufacturing facility.

In the present mode of operation, Inspector 1 routes components C1 to the buffer with the smallest number of components in waiting (i.e., a routing policy according to the shortest queue). In case of a tie, W1 has the highest and W3 the lowest priority.

2. Setting of objectives

The following are the objectives of this project:

1. Study the performance of the manufacturing factory
2. Increase the production protocol for inspector 1

The idea behind the simulation is to use inputs to generate probable output. In this project, the inputs used are the service times of both the inspectors and the workstation. The input data are units of time. (seconds or minutes converted to seconds). The outputs for the simulation are as follows;

I. Product per unit time: This is the number of products produced over a specified period of time. The accounts for all the products from all three workstation.
II. Probability a workstation is busy:
III. Average buffer occupancy of each buffer: The output value represents the average occupancy for each buffer used in the system.
IV. Probability that an inspector is blocked or idle: This would be a value expressed in percentage.

3. Model Conceptualization

This problem will be modelled using simulation software. The Simulation will use the object-oriented programming paradigm to create a clear separation of concern between distinct entities of the system. Random variable generators are used for simulating the random inputs of components for each inspector. The first Input will be all components 1, which will be connected to inspector 1. The second Input will be a random assortment of component 2 and component 3 connecting to inspector 2.

Each inspector will have additional inputs from data files with inspection times for each component. Inspector 1 will input times for component 1 and inspector two for components 2 and 3. The first inspector will output component 1 to three different buffers of component 1 that feed into each workstation. The inspector will prioritize the most open buffer, followed by the workstations in order from 1 to 3. The second inspector will give component 2 to a buffer on workstation 2 and component 3 a buffer on workstation 3.

Each workstation will have additional Input from a specific data file for time to assemble the final part. Once the workstations have taken the inputted time, they can export the final product and add more components. Within the functions for the inspectors, there will be additional coding to record the time it takes for them to complete their processing time to determine if they become blocked. The primary function will also require code to record the system's throughput.

These processes will be initiated in a main Simulation class that will track the timings of events, translate these timings into metrics of interest, and then process statistics from these metrics accordingly.

4. Model Translation

The choice of simulation language for this project would be the python programming language. The python language provides libraries such as “numPy”, “simPy”and data structures compatible with expressing the problem as a model.

Inspector Module:

The Inspector module contains two class implementations, one for each inspector. Each inspector class will calculate the delay time based on the data file corresponding to each component. Inspector1 has an “__init__” method to initialize the class with the mean of component 1. After this is the “generate_mean_from_data” method that calculates the mean from the component's data set. The last method of Inspector1 is “generate_inspect_time” and this uses the mean to generate a random delay within a distribution of the mean. This uses numpy.random.exponential with the mean as an input. Inspector2 has an identical structure and functionality as Inspector1. The only difference is that since Inspector2 works with two components, there is an added functionality for randomly selecting the component to generate a random delay.

Workstation Module:

The Workstation module encapsulates the logic for the Workstation entity represented as a class. The Workstation class receives components and calculates a delay based on the imputed distribution of data. The Class is set up with the __init__ file to initialize the mean of the production times and the product. The Next method is “generate_mean_from_data,” which will see what product is being worked on, select the corresponding data file, and calculate the mean for random number generation. Finally, the “get_delay_time” method generates a product assembly delay time from the mean.

Buffer Module:

The Buffer module contains two class implementations a “Buffer” class and the “Buffer_Manager” class. The Buffer implements the buffer object where components are stored and waiting to be moved into the workstation. The Buffer has an “add_to_buffer” function which attempts to add a

component to the buffer object and returns true if successful and false otherwise. It also has a "remove_from_buffer" to remove components from the Buffer.
The Buffer_Manager class is a controller for all possible buffers, as described in the problem statement. It routes components to the appropriate Buffer and pushes components out of the Buffer to the workstations. The class consists of a constructor that initializes all the Component_Buffer objects needed with their associated component product pairs represented as a tuple. The "attempt_to_add_to_buffer" function takes in a Component as a parameter and attempts to find a buffer to send the component to. It returns True and the product if successful, and False and None otherwise. The "assemble_product" method takes a product as a parameter. It takes the needed components off the workstation buffer to assemble the product. It returns true if the needed components were provided from the buffers and false otherwise.

Simulation Module:

The Simulation module serves as the main file. It contains the Simulation class and the main script that runs the simulation. The simulation consists of a single event list to manage events until a maximum production number has been reached. The constructor method "__init__" is for setting up the necessary components for the simulation, i.e. inspectors, workstations and buffer manager. The "get_next_event" method identifies the closest event in the future event list based on the time it was added. It then returns and removes the event from the list to advance the simulation. The "schedule_add_to_buffer" method initiates an inspection process for a given inspector. It adds the estimated inspection completion time to the future event list. The inspection time is estimated using the inspection time generation mechanism from the inspector module. The "add_to_buffer" method processes an attempt to add an inspected component to a buffer; if a buffer can accept the component, then the construction of a product will be immediately scheduled by adding a corresponding event to the future event list; else, nothing happens. The "unbuffer_workstation" method deals with product assembly in the corresponding workstation. It will try to retrieve the needed components from the buffer. If successful, a new event will be added to the future event list for the completion of the product assembly. The "product_assembled" method deals with keeping track of the products assembled.

UML Class Diagram: