SYSC 4005 Project Report Milestone 1

Momin Mushtaha – 101114546

Mohamed Abdalla - 101115622

Contents

Problem Formulation:

- Conduct a complete simulation study on a manufacturing facility, and Specific the requirements.
- Provide at least one recommendation for an alternative operating policy in the facility.

The facility consists of:

1. Three component types (C1, C2, C3).
    - Each component type has an infinite number of components (do not worry about the supply it is unlimited).
2. Two inspectors (ins1, ins2).
    - Each inspector inspects a pre-defined set of components:
        - Ins1 inspects C1 only.
        - Ins2 inspects C2 and C3.
            - No order for inspecting the components is defined for the ins2
3. Three buffers (buf11, buf12, buf22, buf13, buf33) corresponding to the components every workstation needs.
   ex. W3 needs C1 and C3 and there corresponding buffers are buf13 and buf33.
    - Each of which can store up to two components.
        - If a buffer is full, the inspector that puts the inspected components into the buffer will be blocked.
        - Once the buffer is not full anymore, the inspector will be unblocked.
4. Three workstations (W1, W2, W3) that will use the components stored in the buffers to assemble a product.
    - Each workstation creates a product from a pre-defined combination of component types:
        - W1 will get one C1 component to assemble P1.
        - W2 will get one C1 and one C2 to assemble P2.
        - W3 will get one C1 and one C3 to assemble P3.
5. Three products (P1, P2, P3)

- One of the factors that can halt flow of components is a full buffer , which is typically caused by a delay in a workstation's assembly of a product.
- Another one would be because the workstation does not have all the component types it needs to use to start the assembly.
- Another delay would be because only 1 inspector (ins2) is handling the inspection of 2 components, C2 and C3.
    - **low inspected component supply for C1 and C2.**
- Another one is that component 1 is used in the assembly of all the products. This will increase the probability of its buffer being empty when a workstation needs a component from it.
    - **C1 is in high demand.**

Project Plan:

1. Build an implementation that can be used to simulate the system with the given historical data.
2. Conduct a simulation study using the collected data to assess the performance of the facility.
3. Analyze system throughput, “busy” state probabilities, average buffer occupancy, “blocked” inspector state probabilities.
4. Identify improvements needed to enhance system performance.
5. Make system adjustments and analyze the new simulation to quantify the improvement level.

Model Conceptualization:

A simulation model will be used to conceptualize the manufacturing facility and its operation. The model will analyze the two inspectors, the three workstations, and the buffers used to store components. Historical data is provided in ‘.dat.’ files. This data will be used to conceptualize the system’s performance. The goal of this project is to simulate the operation of the facility and collect data about throughput/product output per unit time, the probability that each workstation is “busy,” the average buffer occupancy of each buffer, and the probability of each inspector having “blocked” status. Using the simulation, the data supplied will be analyzed to identify improvements to the system’s routing policy. All modifications to the system will be documented and analyzed to determine the scale of efficacy.

Model Translation:

Choice of Simulation Language:

The choice of simulation language will be Java. Firstly, in terms of personal experience, Java is the language we are most confident and comfortable with. Java classes allow us to divide the tasks of the system in an organized fashion. The Inspectors and workstations can each have their own classes to avoid too much coupling. Java also allows us to easily scan the data in the given files and save historical data. The data can be simulated in many ways using external libraries or using simple terminal prints.

Implementation Description:

Code does not work; the following is an explanation of how it should've worked.

Classes are divided into mini directories: Buffer_Product, Inspectors, and Workstations. There is a 'main' class that takes care of the simulation calls.

1. Main Class (main.java)
    - Implementation Starter.
    - Reads the '.dat' files and calculates average time values for each file.
    - Initializes inspector and workstation threads with their respective average processing times.

The main file starts off by calculating the average values of the given historical data. This is done by scanning the '.dat' files and reading them line by line. Each line is then parsed into a Double.
A counter increment with each line, the total of the values is incremented with each new value, Average is calculated. Using the average values, threads of each component are issued

2. Inspectors Classes (inspector1.java and inspector2.java)
    - Infinitely loops, sends product to buffer
        i. Inspector 1 sends a general request to the Buffer class; the buffer then assigns the proper buffer allocation. (Currently sends to all respective buffers with no priority)
        ii. Inspector 2 uses a random Boolean to decide which Buffer to send to.

The inspectors are threads. They run infinitely sending components to the buffer. The c components are not treated as objects for now. The inspectors are sending instructions for the buffer to read simulating the reception of components. Inspector 1 sends to all their respective buffers. Inspector 2 uses a random Boolean to determine if it will send to buffer C2 or to Buffer C3. To simulate the inspection time, the threads sleep the duration of the average historical inspection time.

3. Buffer
    - Each Buffer has a data structure to act as the queue.
        i. IF the queue is full, the inspector waits.
        ii. Else, request is taken into queue.
    - Buffer sends the product to respective workstations after processing.

The Buffer is the intermediary between the workstation and the inspector. Each buffer has its own ArrayList of maximum size 2. Whenever the buffer receives an 'Iput' request (inspector gives to buffer), the buffer checks the instruction sent by the inspector in the componentID. Using the componentID, the component is sent to the right buffer. If the buffer queue is full, the inspector will wait until a space is available. This process is timed.

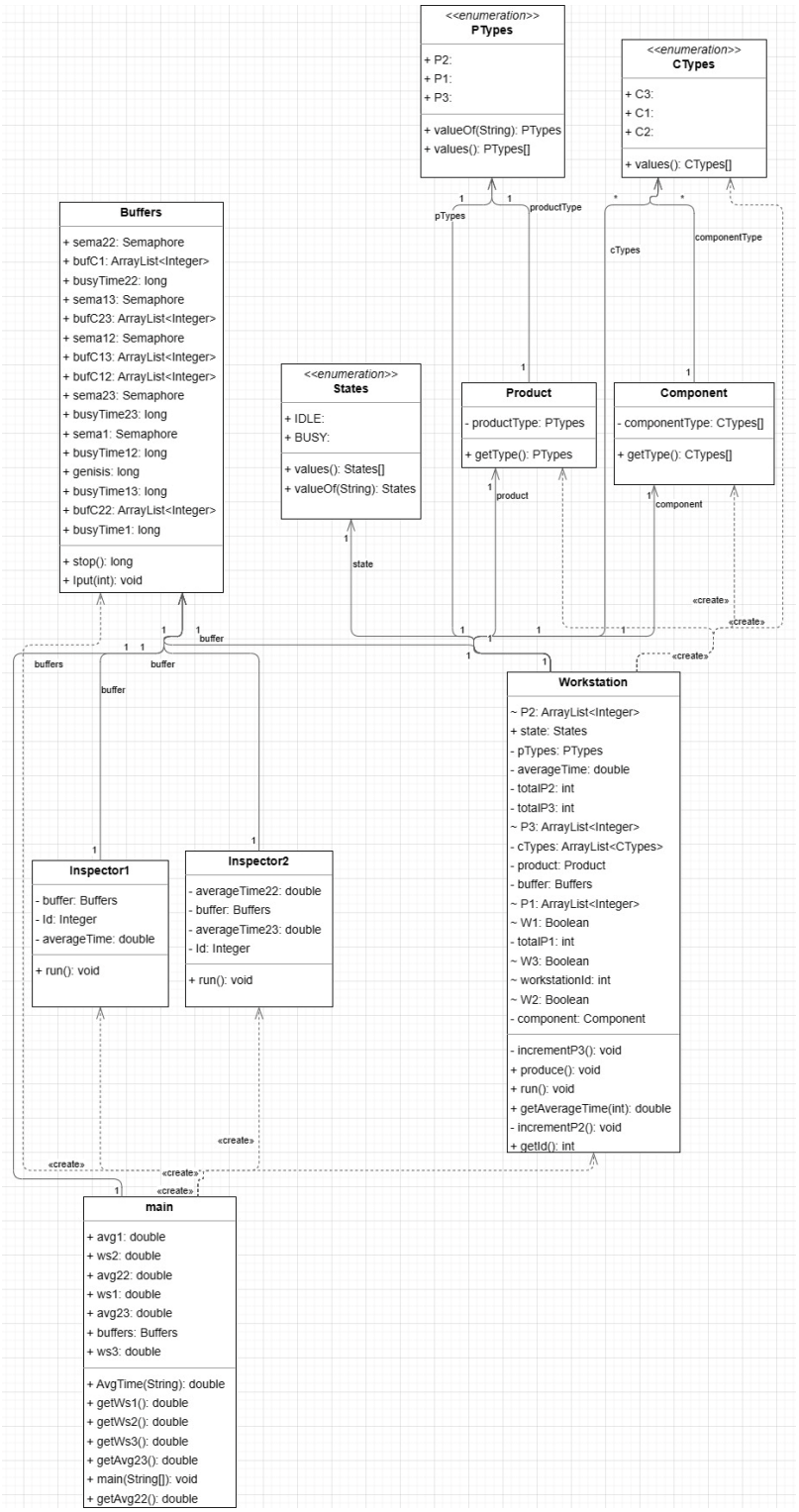
When the Buffer receives a 'Wget' request (workstation asks component), it checks which station has issued the request. Workstation1 will wait for its 1 buffer to have a value and will then take. As for Workstations 2 and 3, they wait for both components needed (Will be changed to more accurately fit system).

4. Workstation
    - Receives product from Buffer and goes through processing time.
    - Outputs a finished product
        i. Workstations 2 and 3 wait for a pair before outputting the finished product.

Thread. Infinitely runs and requests components from Buffer. The thread sleeps to simulate workstation.

Diagrams:

Class UML Diagram:



Sequence Diagram: