SYSC 4005
1st Deliverable: Milestone #1
Date: February 28, 2021
Group: Simulating Discretely

Authors:

Archit Bhatia - 101065674
Ross Matthew - 101079586
Jon Menard - 101086242

Problem Formulation

Using a complete simulation study our team is going to assess the throughput and probability that the inspectors will remain blocked and possibly improve the policy that inspector 1 follows when delivering C1 components to the different workstation for the manufacturing facility bases on the observed historical data provided by the inspectors and workstations.

Setting of Objectives and Overall Project Plan (see step 1)

Goals for February 14th - 28th

1. Determine the Project Goals and Objectives
2. Conceptualize the project by determining the systems states and environment variables.
3. Create the project's UML
4. Using java to implement the conceptual model a computation model will be created. Buffer, Inspector, and Workstation will each be implemented as their own classes. C1,C2 and C3 will be implemented as Enums. The states of each class will also be Enums.

Goals for February 28th - March 14:

1. Configure input and output files for running the simulation. Input files will determine the time it takes for the Inspector or Workstation to complete a task. The output file will record the states for each workstation, buffer, and inspector every second.
2. Begin preliminary analysis of the output data using histograms. Evaluate the distributions with Q-Q plots. Perform chi-square test for each distribution

Goals for March 14th - 28th

1. Verify the model is working correctly and validate the simulation model was built right.
2. Run the simulation in production to gain final output values for analysis. The simulation will be replicated multiple times.
3. Determine the confidence interval for the data being analyzed.

Goals for March 28th - April 11

1. Describe alternative design solutions.
2. Compare the initial design with the alternative designs
3. Write a conclusion and final report

Conceptual Model

A conceptual model for this system is illustrated with the current states of the Inspectors (blocked, inspecting), Buffers (empty, partial, full), and WorkStations (waiting, working). These states completely describe the system and its status in real-time.

Model Translation

Java was chosen as the simulation language because of its object-oriented functionality making it easier to reuse functions for similar objects with the use of inheritance. Additionally, Java provides a mutli-threaded functionality which may be beneficial for this simulation if we decide to implement each object to continuously run at the same time. Java also has useful library functions for I/O operations which will be utilized in this project. It is for these reasons Java is an appropriate language to use for this project.

From the conceptual model, the states for each class need to be determined, as such the following is proposed:

- The states for the inspector are: inspecting, blocked.
- The states for the buffers are: empty, partial, full.
- The states for the workstation are: waiting, working.

The model will be implemented in Java. Inspector, Buffer, and Workstation will each be their own class objects. Inspector and Workstation will implement an abstract class called SimulationObject. SimulationObject will contain the necessary methods to read inputs from a .dat file. Inspector will have two subclasses called Inspector1 and Inspector2 that will implement the required inspector functionalities. Each inspector will have a different getComponent method that is designed based on the components the respective inspector will be inspecting (i.e. C1, C2, or C3).

Similarly, WorkStation will also have two subclasses called Workstation2 and Workstation3. WorkStation provides all of the base functionalities of a workstation which are required for all workstations. Workstation will have a method called getComponents which will retrieve the required components from its available buffers. Workstation2 and Workstation3 will override the getComponents method, to retrieve the components which correspond to their requirements. Workstations and Inspectors will contain an ArrayList with the buffer they can access.

The buffers will store the component they hold, and how many it currently has. A main class called Simulator will be responsible for instantiating the class objects and setting up the simulation. It will also be responsible for polling each instantiated object every second to retrieve the objects' states. The Simulator will output the states into a .csv file for analysis later.

The UML diagram for the simulation is found below:

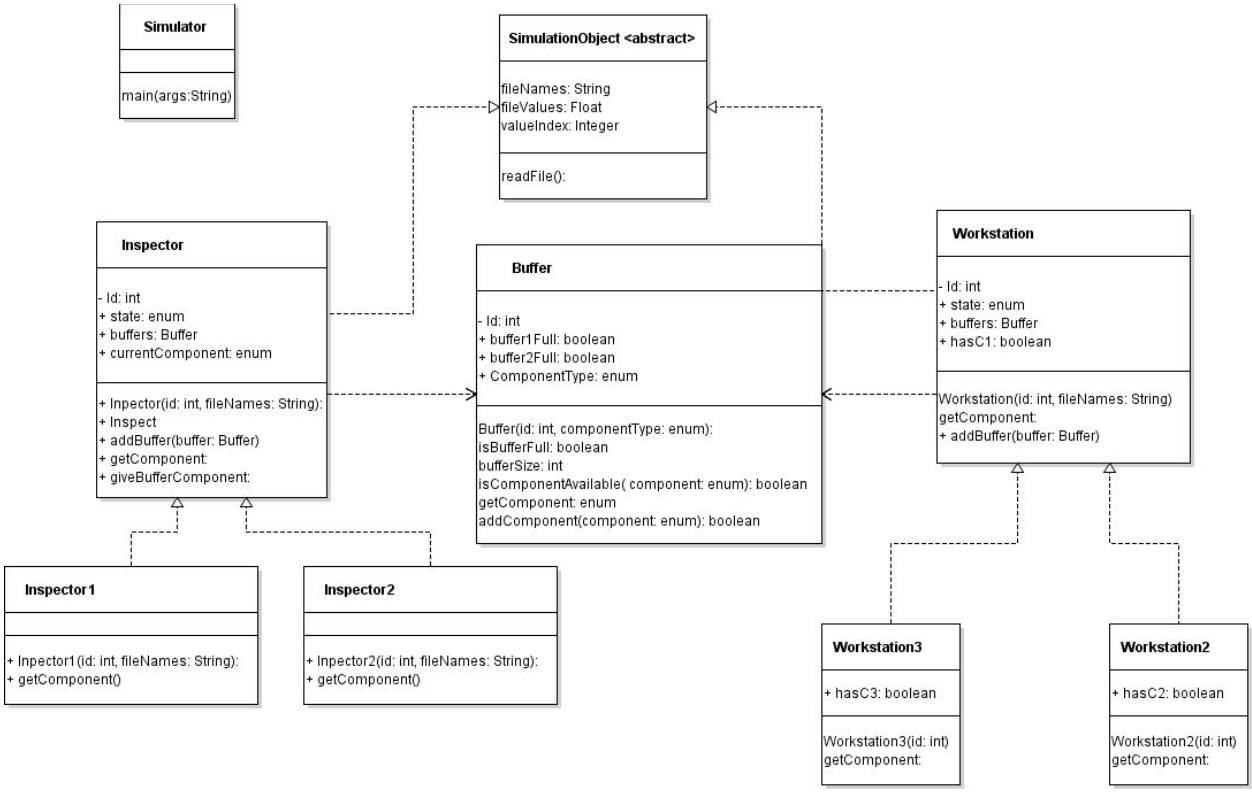

Figure 1: UML diagram for the simulation

The source code for the UML diagram can be found at: https://github.com/jonmenard007/SYSC-4005