SYSC 4005 L2

Discrete Simulation/Modelling

Winter 2021

Milestone #1

Group Members: Julian Mendoza (101067270)

Hamna Nimra Manzoor (101057437)

Colin Wallace (100872525)

Due: February 28th, 2021

TABLE OF CONTENTS

Deliverable 1 2-9

Problem Formulation 3

Setting of objectives and overall project plan 4

model Conceptualization 5

Model translation 8

**Problem Formulation**

Our system consists of a manufacturing facility that assembles three different products labelled P1, P2 and P3. There are three different components which are used to create a product which are labelled C1, C2 and C3. P1 consists of one C1. P2 consists of one C1 and one C2. P3 Consists of one C1 and one C3.

In the manufacturing facility, there are two inspectors which clean, and repair components labelled I1 and I2. I1 works with C1 and I2 works with C2 and C3. I2 randomly chooses which component its working on and there is an infinite number of components which are always immediately available.

There are three workstations labelled W1, W2 and W3, each assemble products P1, P2, and P3, respectively. Once the components pass inspection, inspectors distribute the components to an available workstation which create their respective product. Each workstation has a buffer capacity of two components, with one buffer available for each of the component types needed. When the required components are in queue and the workstation is not busy, production of the product begins immediately. If all the workstation’s buffers for a specific type of component are full, the inspector’s component is considered block until there is an available slot.

Inspector 1 has a routing policy to send out its component to the buffer with the smallest number of components in waiting. In the event of a tie, W1 has the highest priority followed by W2 then W3.

**Setting of Objectives and Overall Project Plan**

The purpose of this simulation is to determine the throughput per unit time and the probability that the inspectors remain blocked. Additionally, we would like to improve the policy that inspector 1 follows to increase production of the products or decrease the amount of time that the inspectors are blocked. To complete all deliverables in the given 8 weeks, the following objective plan will be followed.

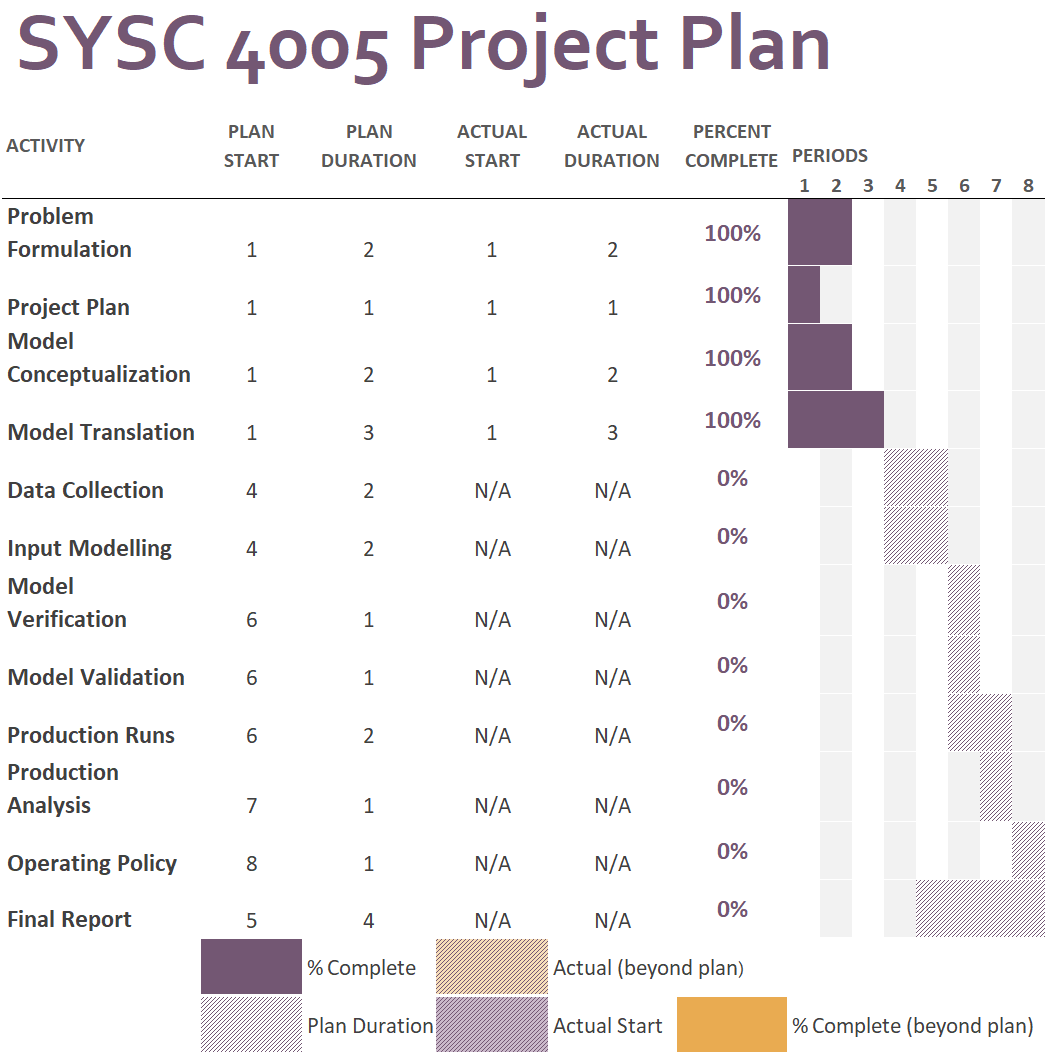


Figure 1: Group Project Plan

Using the given timeline above, the first objective to be considered is statistically analyzing the random aspects of the model and determining the estimated values of the throughput of each product, as well as the probability of an inspector experiencing backpressure from the system, with 95% confidence. Once this data is collected, it will be compared to simulated values using our Java program that has gone under verification and validation.

**Model Conceptualization**

We are given a schematic illustration of the manufacturing facility in its present mode of operation:

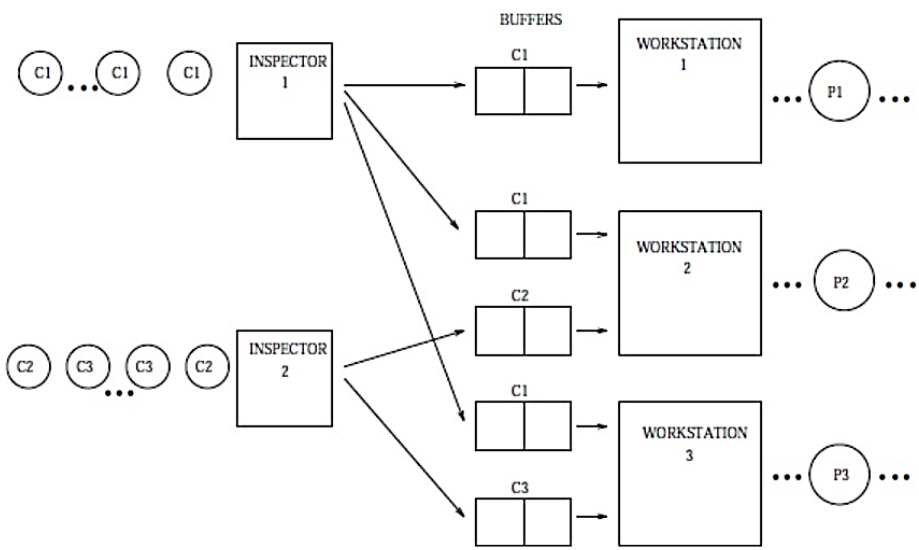


Figure 1: Schematic illustration of the manufacturing facility

**Entities:**

1. Components (C1, C2 and C3)

Attributes: The raw materials required to produce a finished product. There is an infinite supply of components. C1 creates P1. The combination of C1 and C2 creates P2, the combination of C1 and C3 creates P3.

1. Inspectors (I1 and I2)

Attributes: Clean and repair components. Each inspector will have their own probability distribution for the time it takes to inspect a component. I1 only inspects C1. I2 will inspect both C2 and C3.

1. Buffers (BW1C1, BW2C1, BW2C2, BW3C1, BW3C3)

Attributes: Hold a maximum of one component until the respective workstation is available and the respective components are received to create the product.

1. Workstations (W1, W2, W3)

Attributes: Begin production when the required components are given. Each workstation will have their own probability distribution for the time it takes to create a product. Workstations contain buffers.

1. Products (P1, P2, P3)

Attributes: The finished product of the system. Statistics are modelled based of the amount of time required to produce a product.

**Activities:**

Inspection time: The amount of time required by an inspector to clean and repair a component.

Production time: The amount of time required to produce a product from the arrival of the workstation.

**State Variables:**

Inspectors

**INSPECTOR\_BUSY** – An inspector will remain busy until it is blocked.

Buffers

**BUFFER\_FULL** – A buffer will become full if it has a component in it.

Workstations

**WORKSTATION\_BUSY** – A workstation will become busy once given the respective components.

**Events:**

1. Inspection starts (**EIs**): If an inspector is free, it immediately begins to clean and repair components. This event is triggered when the inspection of the previous component is finished, and the inspector is not blocked, or when the simulation begins.
2. Inspector blocked (**EIb**): Once the inspection is finished, the inspector will follow a routing policy. If the routing policy does not allow the component to be passed, the inspector will trigger the event.
3. Workstation start (**EWs**): Once the workstation has the respective components in its buffers and is not currently busy, it will trigger this event.

**Event Notices:**

1. Inspection (EIs,t,Ci, Ii)
2. Blocked (EIb,t,Ci,Ii)
3. Production (EWs,t,Wi,Ci)

**Essential features:**

There are three essential features that need to be monitored during the simulation. The inspectors, buffers, and workstations. Inspectors are given a simulated amount of time to clean and repair components. These inspectors need to follow a routing policy to determine which workstation will acquire the component. Buffers are essential for determining where the component will go based off the routing policy. Workstations are given a simulated amount of time to complete the production of a product once the respective components are given.

**Interactions:**

Components and Inspectors

Components will be given to inspectors to be repaired and cleaned. Components are labelled C1, C2 and C3. C1 components are handled by inspector I1. C2 and C3 components are handled by inspector I2. C2 and C3 are randomly given to the inspector. Inspector will place the components into a buffer.

Inspectors and Buffers

To follow the routing policy, inspectors will have to determine if a buffer is full. Inspector I1 will have to check the buffers corresponding to the component in a priority sequence. If the buffer with the highest priority is full, proceed to the next one, until all buffers are checked. If all buffers are full the inspector will be blocked.

Components and Buffers

Each component will be routed to a corresponding buffer. A buffer can have a maximum of one component.

Components and Workstations

A component will leave the buffer and enter a workstation that it is suited for in order produce the corresponding product. Workstation W1 uses component C1. Workstation W2 uses component C1 and C2. Workstation W3 uses component C1 and C3.

Workstations and Products

Workstations are responsible for creating the finished products. A workstation is given a simulated amount of time to create a finished product.

**Assumptions:**

1. There is an infinite number of components.
2. There is no delay in transferring components. A component will immediately transfer to the desired destination when required.
3. The inspector will hold on to the component when blocked. The inspector will not attempt to inspect another component or destroy the component in the process.

**Model Translation**

Our team has decided to create the simulation using Java. Java has built-in features that our team can use to create a simulation with ease. Due to the Java being an object-oriented language, we can use this to our advantage in designing our simulation.

The Model Translation section will be continuously updated. For this milestone, we simply described the functionality of each class, provided a skeleton of the code, and created the UML for the system in its current state.

**SimModel.java**

The SimModel class will be the driver class of the simulation. It will provide the necessary utility functions such as scheduling and processing events, handling files, and generating reports.

- init() : initializes all variables required for the simulation. Puts the entities in their necessary state for starting the simulation.

- start() : runs the simulation. It will consistently poll from the event list and check if the lifecycle of the simulation is over.

- processInspection(event: SimEvent): handles inspection events. This function implements the logic for the routing policy. It will generate the next inspection event for the Inspector if applicable otherwise it will generate a blocked event.

- processBlocked(event: SimEvent): handles blocked events. This function will be used to determine the amount of time the process is blocked.

- processProduction(event: SimEvent): handles events related to the workstation. This function will be used to schedule the next production event and increment the necessary production.

- createEvent(type: SimEvent.event, entity: Object): utility function used to create new events.

- generateStatistics(filename:String): utility function for computing statistics for input modelling.

- generateReport(): Outputs the necessary statistics for the simulation

**Inspector.java**

Class representing the inspector. This class contains the enum for components.

**WorkStation.java**

Class representing the workstation.

**SimEvent.java**

Class representing an event. This class contains the associated class involved, the enum for the events, and the time of service.

**Simulation model logic:**

* 1. Initialize all components
  2. Input modeling
     1. Manually analyze data and determine best distribution fit –
     2. File handler - get data and compute statistics
  3. Distribute inputs to required components
  4. Run simulation
     1. while the simulation has not ended and there is still an event in the event list,

get the event

advance simulation clock

simulate event and schedule next event

* 1. Generate a report

**System UML:**

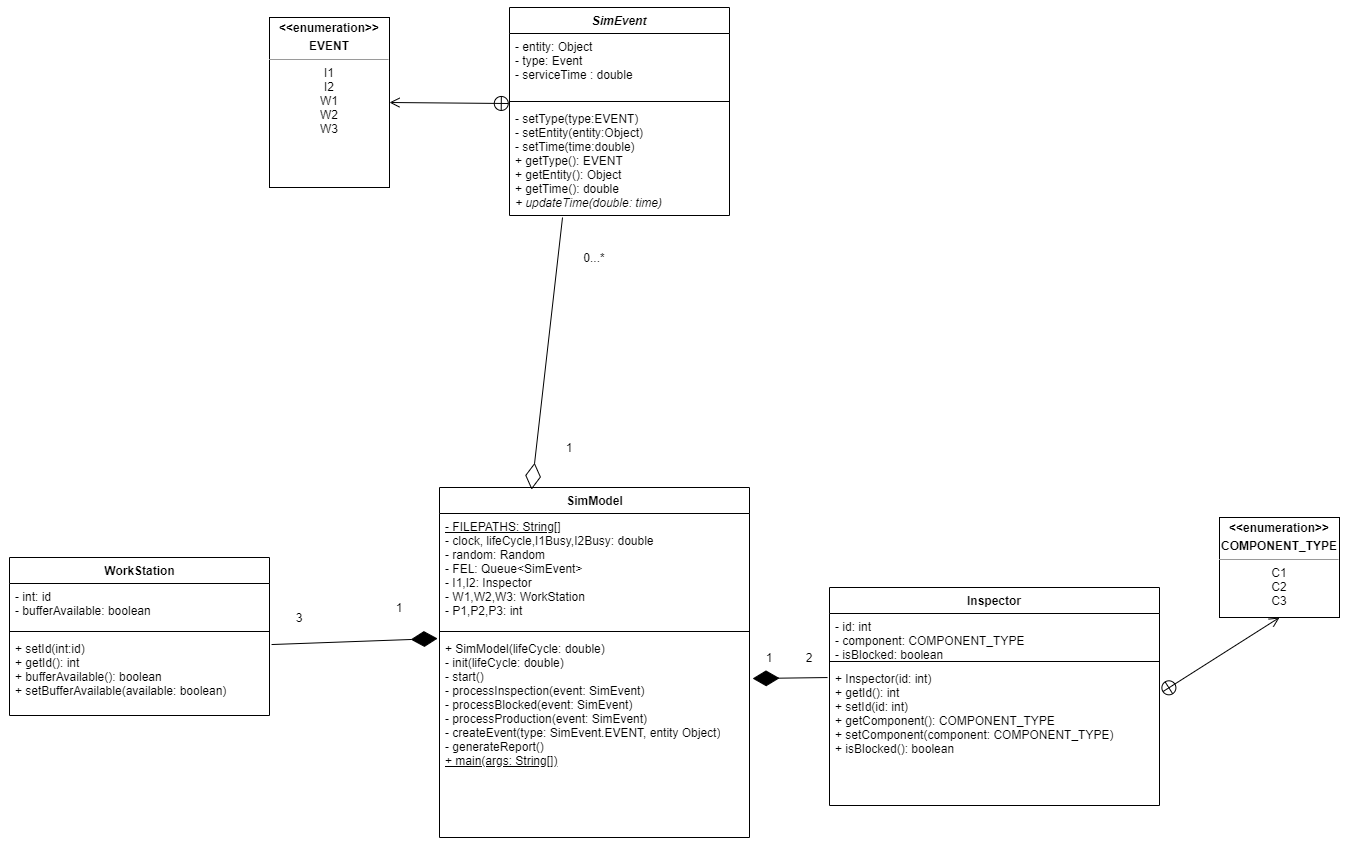


Figure 3: UML diagram of the system.