CMPE 412 – COMPUTER SIMULATION

PROJECT 2 – MANUFACTURING SYSTEM

Submitted by:

Emircan Durmuş

Project Advisor:

Assist. Prof. Doğan Çörüş

Faculty of Engineering and Natural Sciences

Kadir Has University

Spring 2024

Python programming language was used to perform this simulation. SimPy, a Python library used to create event-based simulations, was also used. The input variables in the simulation are as follows.

```
#---------------START OF CHANGING INPUT VARIABLES---------------
numberOfCNCMachines = 5
numberOfMoldingMachines = 5
numberOfAssemblyStations = 5
numberOfInspectionStations = 5
numberOfPackagingStations = 5
lengthOfWorkShift = 480
totalShift = lengthOfWorkShift * 3
simulationTime = totalShift * 10

loadingTime = 2
machiningTime = 7
moldingTime = 5
assemblingTime = 6
inspectingTime = 2
packagingTime = 3

CNCFailRate = 0.001
CNCRepairTime = 30
moldingFailRate = 0.00125
moldingRepairTime = 25
assemblyFailRate = 0.00083
assemblyRepairTime = 20
#---------------END OF CHANGING INPUT VARIABLES---------------
```

These input variables can be changed by the user as desired to obtain different results in the simulation.

```
#---------------RAW MATERIAL LOAD FUNCTION---------------
def loading(env, name, rawMaterialQueue):
    while True:
        yield env.timeout(loadingTime)
        print(f'{name} Raw materials loaded at {env.now:.2f}')
        yield rawMaterialQueue.put(1)
```

The function named “loading” shown above ensures that raw material is loaded into the production line at certain intervals. The loadingTime previously determined by the user is used and a raw material is added to the rawMaterialQueue.

```
#--------------MACHINING FUNCTION--------------
def machining(env, name, rawMaterialQueue, cncMachines, machiningOutput):
    while True:
        yield rawMaterialQueue.get()
        with cncMachines.request() as request:
            yield request
            start_time = env.now
            yield env.timeout(machiningTime)
            machiningTimes.append(env.now - start_time)
            print(f'{name} Machining finished at {env.now:.2f}')
            yield machiningOutput.put(1)
```

The “machining” function shown above sends the raw material previously added to the rawMaterialQueue to the CNC machines. The machiningTime previously specified by the user is used and added to the machiningOutput.

```
#--------------MOLDING FUNCTION--------------
def molding(env, name, machiningOutput, moldingMachines, moldingOutput):
    while True:
        yield machiningOutput.get()
        with moldingMachines.request() as request:
            yield request
            start_time = env.now
            yield env.timeout(moldingTime)
            moldingTimes.append(env.now - start_time)
            print(f'{name} Molding finished at {env.now:.2f}')
            yield moldingOutput.put(1)
```

The “molding” function shown above takes the processed material in the queue called machiningOutput and performs molding within the “moldingTime” period. The new material that emerges is added to the queue called “moldingOutput”.

```
#---------------ASSEMBLING FUNCTION---------------
def assembling(env, name, moldingOutput, assemblyStations, assemblyOutput):
    while True:
        yield moldingOutput.get()
        with assemblyStations.request() as request:
            yield request
            start_time = env.now
            yield env.timeout(assemblingTime)
            assemblyTimes.append(env.now - start_time)
            print(f'{name} Assembling finished at {env.now:.2f}')
            yield assemblyOutput.put(1)
```

The function named assembling shown above, like the other functions, assembles the item it takes from the “moldingOutput” queue for the “assemblingTime” period and adds it to the queue named “assemblyOutput”.

```
#---------------INSPECTING FUNCTION---------------
def inspecting(env, name, assemblyOutput, inspectionStations, inspectionOutput):
    while True:
        yield assemblyOutput.get()
        with inspectionStations.request() as request:
            yield request
            start_time = env.now
            yield env.timeout(inspectingTime)
            inspectingTimes.append(env.now - start_time)
            print(f'{name} Inspecting finished at {env.now:.2f}')
            yield inspectionOutput.put(1)
```

The function named “inspecting” shown above performs the inspection in the same logic as the other functions and puts the inspected product into a new queue.

```
#---------------PACKAGING FUNCTION---------------
def packaging(env, name, inspectionOutput, packagingStations):
    global totalFinished
    while True:
        yield inspectionOutput.get()
        with packagingStations.request() as request:
            yield request
            start_time = env.now
            yield env.timeout(packagingTime)
            packagingTimes.append(env.now - start_time)
```

```
            totalFinished += 1
            print(f'{name} Packaging finished at {env.now:.2f}')
```

Finally, the “packaging” function packages the product for a predetermined period of time called “packagingTime” and increases “totalFinished” by one for each product that is packaged.

```
#--------------FUNCTION FOR SIMULATING THE FAILURE OF MACHINES--------------
def machineFail(env, name, machines, failRate, repairTime):
    while True:
        yield env.timeout(random.expovariate(failRate))
        with machines.request() as request:
            yield request
            print(f'{name} Machine failed at {env.now:.2f}')
            yield env.timeout(repairTime)
            print(f'{name} Repaired at {env.now:.2f}')
```

This function called “machineFail” is used to add machine failures that are likely to occur in real life to the simulation. This function works according to the failure probability determined for each machine and if the relevant machine is broken, the relevant machine is repaired for a predetermined period of time.

At the end of the simulation, the user is provided with values such as the total number of products produced, thanks to the "totalFinished" in the "packaging" function, and the average completion time of each process, thanks to the process times recorded separately in each function.