CMPE412 - Computer Simulation

Project 2 - Manufacturing System

Submitted by:

20201701021 - Begüm Doğan

20191701011 - Kazım Emre Yılmazcan

Project Supervisors:

Dr. Doğan Çörüş

Faculty of Engineering and Natural Sciences

Kadir Has University

Spring 2024

TABLE OF CONTENTS

1 INTRODUCTION

In this project, we aimed to create a discrete-event simulation to create a model of a high-volume manufacturing line. Our focus is optimizing throughput for a single product line by identifying and handling production bottlenecks and analyzing the impacts of operational variables through scenario analysis. A bonus objective is implemented for multi-product manufacturing in the facility where the simulation is aimed to handle multiple product types, each with unique manufacturing requirements.

2 DESIGN

2.1 System Description

The production line is divided into several different stages like raw material handling, machining, assembly, quality control, and packaging. Each stage includes specific machinery and labor requirements, operating across multiple shifts.

2.2 Objectives

- Optimizing the throughput for a single product line
- Identifying and eliminating production bottlenecks
- Analyzing the impacts of operational variables
- A bonus objective where the simulation to handle multiple product manifacturing, analyzing additional complexity and resource allocation challenges.

2.3 Tools and Programming Requirements

We used Python language for this project. We utilized SimPy library for discrete-event simulation.

3 IMPLEMENTATION

Our implementation of the code consists of several classes and functions to model the manufacturing process, manage the raw materials, and simulate the production process.

3.1 Product Class

This class initializes products with their unique IDs and types, setting the processing times for each stage of the manufacturing based on the product type.

```
4   class Product:
5       def __init__(self, product_id, product_type):
6           self.product_id = product_id
7           self.product_type = product_type
8           self.processing_times = self.set_processing_times()
9
10      def set_processing_times(self):
11          if self.product_type == 'TypeA':
12              return {
13                  'raw_material': random.uniform(1, 3),
14                  'machining': random.uniform(4, 6),
15                  'assembly': random.uniform(2, 4),
16                  'quality_control': random.uniform(1, 3),
17                  'packaging': random.uniform(1, 3)
18              }
19          elif self.product_type == 'TypeB':
20              return {
21                  'raw_material': random.uniform(2, 4),
22                  'machining': random.uniform(3, 5),
23                  'assembly': random.uniform(3, 5),
24                  'quality_control': random.uniform(2, 4),
25                  'packaging': random.uniform(2, 4)
26              }
27          else:
28              raise ValueError('Unknown product type')
```

This class has the attributes and behaviors of a product, including its unique ID (product_id), type (product_type), and processing times for each stage of production (processing_times). The processing times are assigned by set_processing_times method based on the product type, using random values within specified ranges that we initialized to simulate variability.

3.2 ManufacturingSystem Class

The ManufacturingSystem class initializes resources for each step of the production and it tracks total processing times.

```
class ManufacturingSystem:
    def __init__(self, env):
        self.env = env
        self.raw_material_handling = simpy.Resource(env, capacity=1)
        self.machining = simpy.Resource(env, capacity=1)
        self.assembly = simpy.Resource(env, capacity=1)
        self.quality_control = simpy.Resource(env, capacity=1)
        self.packaging = simpy.Resource(env, capacity=1)
        self.total_times = {
            'TypeA': {
                'raw_material': 0,
                'machining': 0,
                'assembly': 0,
                'quality_control': 0,
                'packaging': 0
            },
            'TypeB': {
                'raw_material': 0,
                'machining': 0,
                'assembly': 0,
                'quality_control': 0,
                'packaging': 0
            }
        }
```

This class utilizes the resources like raw materials, machine availability etc. that are required for each production stage using SimPy's Resource class, which allows for the simulation of resource constraints. The total_times dictionary tracks the cumulative processing times for each product type and stage.

3.3 Process Functions

Each process function simulates the production stage, and considering the potential machine failures during machining.

```
55      def raw_material_process(self, product):
56          processing_time = product.processing_times['raw_material']
57          yield self.env.timeout(processing_time)
58          self.total_times[product.product_type]['raw_material'] += processing_time
59          print(f'Product {product.product_id} ({product.product_type}): Raw material handling complete at {self.env.now}')
60
61      def machining_process(self, product):
62          processing_time = product.processing_times['machining']
63          if random.random() < 0.1:  # 10% chance of machine failure
64              repair_time = random.uniform(1, 3)
65              print(f'Product {product.product_id} ({product.product_type}): Machining failure, repair time {repair_time} at {self.env.now}')
66              yield self.env.timeout(repair_time)
67          yield self.env.timeout(processing_time)
68          self.total_times[product.product_type]['machining'] += processing_time
69          print(f'Product {product.product_id} ({product.product_type}): Machining complete at {self.env.now}')
```

These functions simulate the processing time for each step by a timeout event in SimPy. The machining_process includes a random chance of machine failure in the simulation which can also happen in real-world.

3.4 Main Simulation Function

The main simulation function creates a random number of products and processes each through the manufacturing steps.

```
89  def process_product(env, product, system):
90      print(f'Product {product.product_id} ({product.product_type}): Arrived at {env.now}')
91      with system.raw_material_handling.request() as request:
92          yield request
93          yield env.process(system.raw_material_process(product))
94
95      with system.machining.request() as request:
96          yield request
97          yield env.process(system.machining_process(product))
98
99      with system.assembly.request() as request:
100         yield request
101         yield env.process(system.assembly_process(product))
102
103     with system.quality_control.request() as request:
104         yield request
105         yield env.process(system.quality_control_process(product))
106
107     with system.packaging.request() as request:
108         yield request
109         yield env.process(system.packaging_process(product))
110
111     print(f'Product {product.product_id} ({product.product_type}): Finished at {env.now}')
```

This function orchestrates the processing of each product through the entire production line, ensuring that each stage is completed in sequence. Resource requests are made for each stage, and the corresponding process function is called to simulate the processing time.

3.5 Main Simulation Execution

The main simulation generates a random number of products and processes each through the manufacturing stages.

```
env = simpy.Environment()
system = ManufacturingSystem(env)

# Generate a random number of products between 20 and 30
num_products = random.randint(20, 30)
product_types = ['TypeA', 'TypeB']
products = [Product(i, random.choice(product_types)) for i in range(num_products)]

for product in products:
    env.process(process_product(env, product, system))

env.run()

# Output total processing times
print("\nTotal processing times:")
for product_type, times in system.total_times.items():
    total_time = sum(times.values())
    print(f"\n{product_type}:")
    for stage, time in times.items():
        print(f"  {stage}: {time:.2f} min")
    print(f"  Total: {total_time:.2f} min")
```

This sets up the simulation environment, creates a list of products with random types, and initializes the simulation by processing each product through the manifacturing. After the simulation is done, the total processing times for each product and stage are printed via our code.

4 FINDINGS

During the simulation, our key findings include:

1. Processing Times: Each product type has distinct processing times throughout stages.
2. Machine Failures: Our simulation considers potential machine failures during the process and time for maintenance and repair protocols to be considered.
3. Throughput: The total processing times gave us insight into the efficiency and bottlenecks through the production steps. The figures of our outputs displayed below.

```
Product 18 (TypeB): Arrived at 0
Product 19 (TypeB): Arrived at 0
Product 20 (TypeB): Arrived at 0
Product 21 (TypeB): Arrived at 0
Product 0 (TypeA): Raw material handling complete at 2.246121907266959
Product 1 (TypeA): Raw material handling complete at 3.8765688900615363
Product 2 (TypeB): Raw material handling complete at 7.489124676927539
Product 0 (TypeA): Machining complete at 8.180882049879965
Product 3 (TypeA): Raw material handling complete at 9.981942635859784
Product 0 (TypeA): Assembly complete at 11.359005432932506
Product 4 (TypeA): Raw material handling complete at 11.739146285619364
Product 1 (TypeA): Machining complete at 13.198422700800705
```

```
Product 20 (TypeB): Assembly complete at 105.44227126918928
Product 19 (TypeB): Packaging complete at 106.71700970598472
Product 19 (TypeB): Finished at 106.71700970598472
Product 20 (TypeB): Quality control complete at 107.85425475320764
Product 21 (TypeB): Assembly complete at 108.51362808299466
Product 21 (TypeB): Quality control complete at 110.99652800516144
Product 20 (TypeB): Packaging complete at 111.61204664723167
Product 20 (TypeB): Finished at 111.61204664723167
Product 21 (TypeB): Packaging complete at 114.19867652309588
Product 21 (TypeB): Finished at 114.19867652309588
```

```
Total processing times:

TypeA:
  raw_material: 20.14 min
  machining: 51.26 min
  assembly: 29.54 min
  quality_control: 19.79 min
  packaging: 18.94 min
  Total: 139.67 min

TypeB:
  raw_material: 39.34 min
  machining: 48.21 min
  assembly: 48.11 min
  quality_control: 38.67 min
  packaging: 37.44 min
  Total: 211.76 min
```

5 SUMMARY

In this project we created a discrete-event simulation to simulate a high-volume manufacturing line. Optimizing throughput and handle the bottlenecks, the simulation of the production process is finalized. The bonus challenge to challenge our simulation to handle multiple product manufacturing, created a complexity for our system and resource allocation considerations for us. The findings show the importance of efficient resource handling and possible pitfalls that might arise in the real-world scenarios considered in simulation environment.