K.H.U. Faculty of Engineering

**CMPE412 Computer Simulation**

# Manufacturing System

Instructor: Assoc. Prof. Doğan Çörüş

Ahmet Kemal Cabbar – 20191701008 (CE)

**Faculty of Engineering and Natural Sciences**

**Kadir Has University**

**June 2024**

# Table of Contents

# 1 INTRODUCTION

In the modern industrial environment, the effectiveness and efficiency of production processes are critical to achieving competitive advantage. Correctly designing and managing the production line makes a big difference in reducing costs, ensuring quality control and shortening delivery times. In this context, simulation of the manufacturing process [2] offers a powerful tool to detect potential disruptions in advance and implement process improvements.

This project aims to model a production line using the SimPy [1] library to simulate a production line. SimPy [1] is an event-based simulation library that makes it easy to model the behavior of processes over time. In this project, the various stages of a particular production line, the processes taking place at each stage, labor requirements and possible machine malfunctions will be considered.

# 2 METHODOLOGY

## 2.1 Production Line Processes and Challenges

First, the project was simulated using the Python programming language. For this reason, I started by integrating the libraries we will use for simulation.

```
import simpy
import random
import pandas as pd
```

- **simpy:** This module is an event-based simulation library and is used to perform factory production line simulation.
- **random**: Used to generate random numbers, for example to simulate machine malfunctions.
- **pandas**: Data analysis library, used to store and analyze simulation results in tabular form.

A production line represents the successive stages a product goes through, starting as raw material and becoming the final product. The production line considered in this project consists of five main stages:

- Loading: Loading of raw materials to the production line.
- Machining: Turning raw materials into specific parts.
- Assembly: Combining parts to turn them into semi-finished or final products.
- Inspecting: Products passing through quality control processes.
- Packaging: Packaging the final products and making them ready for shipment.

At each stage, a certain number of workers work and operations are carried out within certain periods of time. In addition, each process has a certain risk of failure, in which case the production process is interrupted and maintenance is required.

I defined a dictionary structure that describes the processing times (in minutes) of each stage.

```
PROCESSING_TIMES = {
    'loading': 5,
    'machining': 10,
    'assembling': 8,
    'inspecting': 6,
    'packaging': 4
}
```

I defined a variable that defines the maintenance time when the machine breaks down and assigned this time to 3 minutes. Then, I defined the probability of the machine malfunctioning in each operation and assigned this probability to 10 percent (0.1 = 10%).

```
MAINTENANCE_TIME = 3
BREAKDOWN_RATE = 0.1
```

I defined the number of workers at each stage using the Python dictionary data structure.

```
NUM_WORKERS = {
    'loading': 2,
    'machining': 3,
    'assembling': 4,
    'inspecting': 2,
    'packaging': 3
}
```

I defined the length of a shift (8 hours), the total duration of the simulation (100 minutes), and the number of product types to be produced as follows:

```
SHIFT_LENGTH = 8
SIMULATION_TIME = 100
NUM_PRODUCTS = 2
```

### 2.2 Project Goals

The main goals of this project are:

- Modeling the processing time and labor requirements of each stage in the production line.

- Simulating possible machine malfunctions and maintenance processes.
- To evaluate the overall performance and efficiency of the production line.
- To examine the differences in the production processes of different product types.

### 2.3 Simulation Environment and Method

The simulation environment was created using the SimPy library. The simulation simulates the operations and possible malfunctions of the production line over a period of time. A certain number of parts are produced for each product type and the progress of each part on the production line is recorded.

The main components of the simulation are:

#### 2.3.1 Production Line Class

The class (ManufacturingLine) that models each stage and maintenance team in the production line.

```
class ManufacturingLine:
    def _init_(self, env):
        self.env = env
        self.stages = {
            stage: simpy.Resource(env, capacity=NUM_WORKERS[stage]) for stage in NUM_WORKERS
        }
        self.repair_team = simpy.Resource(env, capacity=2)
        self.data = []
```

- **__init__** method: Initializes the production line and creates simpy resources for each stage.
- **self.repair_team**: Defines the repair team (capacity of 2 people).
- **self.data**: A list to store simulation data.

#### 2.3.2 Part Processing

The function (process_part) that manages the passage of each part through the stages in the production line.

```
def process_part(self, part, product_type):
    stages = ['loading', 'machining', 'assembling', 'inspecting', 'packaging']
    for stage in stages:
        processing_time = PROCESSING_TIMES[stage] * (1 if product_type == 1 else 1.2)
        with self.stages[stage].request() as request:
            start_time = self.env.now
            yield request
            try:
                yield self.env.timeout(processing_time)
                if random.random() < BREAKDOWN_RATE:
                    yield self.env.process(self.repair_machine(stage))
                finish_time = self.env.now
                self.data.append({
                    'Part': part,
                    'Stage': stage,
                    'Start Time': start_time,
                    'Finish Time': finish_time,
                    'Duration': finish_time - start_time,
                    'Product Type': product_type
                })
            except simpy.Interrupt:
                # Handle a breakdown during processing
                yield self.env.process(self.repair_machine(stage))
```

- **stages**: List of production stages.
- For each stage, it determines the processing time and resource demands.
- The machine checks the malfunction status and initiates the repair process if necessary.
- Records simulation data (start time, end time, duration and product type).

### 2.3.3 Machine Repair

Function (repair_machine) that models machine malfunctions and maintenance processes.

```
def repair_machine(self, stage):
    with self.repair_team.request() as repair:
        start_repair = self.env.now
        yield repair
```

```
            yield self.env.timeout(MAINTENANCE_TIME)
```

### 2.3.4 Part Manufacturer

The function (part_manufacturer) that starts the processing of each part on the production line.

```
def part_manufacturer(env, line, part_id, product_type):
    yield env.process(line.process_part(part_id, product_type))
```

### 2.3.5 Setup and Simulation

Starting the simulation and collecting data.

```
def setup(env, num_parts_per_type):
    line = ManufacturingLine(env)
    for product_type in range(1, NUM_PRODUCTS + 1):
        for i in range(num_parts_per_type):
            env.process(part_manufacturer(env, line, f"Part_{i + 1}", product_type))

    yield env.timeout(SIMULATION_TIME)
    df = pd.DataFrame(line.data)
    return df
```

- **line**: Creates the production line.
- Starts production of a certain number of parts for each product type.
- It waits for the simulation time and then returns the data as a DataFrame.

```
env = simpy.Environment()
process = env.process(setup(env, 5))
env.run()
df = process.value
print(df)
```

- Creates the simulation environment (**env**).
- It starts the **setup** function and simulates the process.
- Runs the simulation (**env.run()**).

- It takes the simulation data as DataFrame (**df = process.value**) and prints it to the screen (**print(df)**).

### 2.4 Expected Outputs and Evaluation

At the end of the simulation, the stages, processing times and possible interruptions in the production process of each part will be recorded. These data will be analyzed using the pandas library and the simulation results will be evaluated. The results obtained will provide valuable information on which stages can be improved to increase the efficiency of the production line and where possible bottlenecks are.

As a result, this project provides detailed simulation modeling to enable better understanding and management of production line processes. This type of simulation study could be an important step in optimizing real-world production processes and reducing operating costs.

# 3 CONCLUSION

Within the scope of this project, a comprehensive simulation study was carried out using the SimPy library to model and simulate various stages of a production line. The performance and efficiency of the production line were analyzed in detail, taking into account various factors such as operations encountered in the production process, labor requirements and possible machine malfunctions. Simulation results revealed bottlenecks and improvement potentials at different stages of the production line. The data obtained provides important insights for optimizing production processes, managing maintenance and repair processes more effectively, and increasing overall operational efficiency. This type of simulation study can be used as a valuable tool in making strategic decisions in the fields of production management and industrial engineering and can provide significant benefits for businesses seeking to gain a competitive advantage.

# 4 REFERENCES

[1] Oujezsky, V., & Horvath, T. (2016). Case Study and Comparison of SimPy 3 and OMNeT+ plus Simulation. In N. Herencsar (Ed.), *2016 39TH INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS AND SIGNAL PROCESSING (TSP)* (pp. 15–19). IEEE Reg 8; IEEE Austria Sect; IEEE Czechoslovakia Sect; IEEE Czechoslovakia Sect SP CAS COM Joint Chapter; IEEE Croatia Sect, Communicat Chapter; Brno Univ Technol, Dept Telecommunicat; Budapest Univ Technol & Econ, Dept Telecommunicat & Media Informat; Karadeniz Tech Univ, Dept Elect & Elect Engn; W Pomeranian Univ Technol, Fac Elect Engn; VSB Tech Univ Ostrava, Dept Telecommunicat; Slovak Univ Technol, Inst Telecommunicat; Univ Ljubljana, Lab Telecommunicat; Czech Tech Univ Prague, Dept Telecommunicat Engn; Univ Osijek, Fac Elect Engn; Tech Univ Sofia, Fac Telecommunicat; Seikei Univ, Grad Sch & Fac Sci & Technol, Informat Networking Lab.

[2] Fujii, S., & Sugimura, N. (1996). Simulation for manufacturing systems. *INTERNATIONAL JOURNAL OF THE JAPAN SOCIETY FOR PRECISION ENGINEERING*, *30*(3), 195–199.