System modeling 12th project report

Project 2: Factory Production Line

Names:

Hussein Galal 221001810

Marwan Elsayed 221003166

Supervised by: Prof. Dr. Khaled Mahar

Teaching Assistant: Eng. Sara Mohamed

Course name: System modeling and simulation

Course code: CCS3003

🏭 Factory Simulation Engine: Enhanced Edition

Version: 2.0

Python: 3.8+

SimPy: 4.0+

> A state-of-the-art discrete-event simulation of manufacturing systems with comprehensive queueing theory metrics and flexible termination conditions.

**📋 Overview**

This repository contains two versions of a factory production line simulation:

- mainVersion.py: The baseline simulation model

- enhancedVersion.py: An improved version with advanced features

Both simulations model a three-machine production line (M1, M2, M3) processing products with:

- Random interarrival times

- Variable processing times

- Machine failures and repairs

- Queue behavior

The enhanced version significantly improves the simulation's flexibility, realism, and analytical capabilities while maintaining the core structure of the original model.

**✨ Key Enhancements**

Comparison of features between mainVersion.py and enhancedVersion.py:

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | mainVersion.py | enhancedVersion.py | Impact |
| Simulation Termination | Fixed time (200 units) | Flexible: runs until all products are processed | Ensures complete processing and improves flexibility |
| Number of Products | Hardcoded (20) | Configurable (default 100) | Enables scalable, user-defined simulations |
| Machine Failure Logic | Machines fail anytime | Machines fail only when busy | More realistic failure model based on machine operation |
| Machine State Tracking | Basic (broken, queue length) | Enhanced (busy/idle states, idle time) | Precise state metrics supporting advanced analysis |
| Queue Monitoring | Event-based queue length | Continuous queue sampling | More accurate average queue length calculation |
| Queueing Metrics | Limited (utilization, downtime) | Comprehensive (probability of wait, queue length, etc.) | Deeper system analysis capabilities |
| Interarrival Tracking | None | Tracks all interarrival times | Enables key queueing theory metrics |
| Active Products | None | Tracks products in system | Enables precise termination and load monitoring |

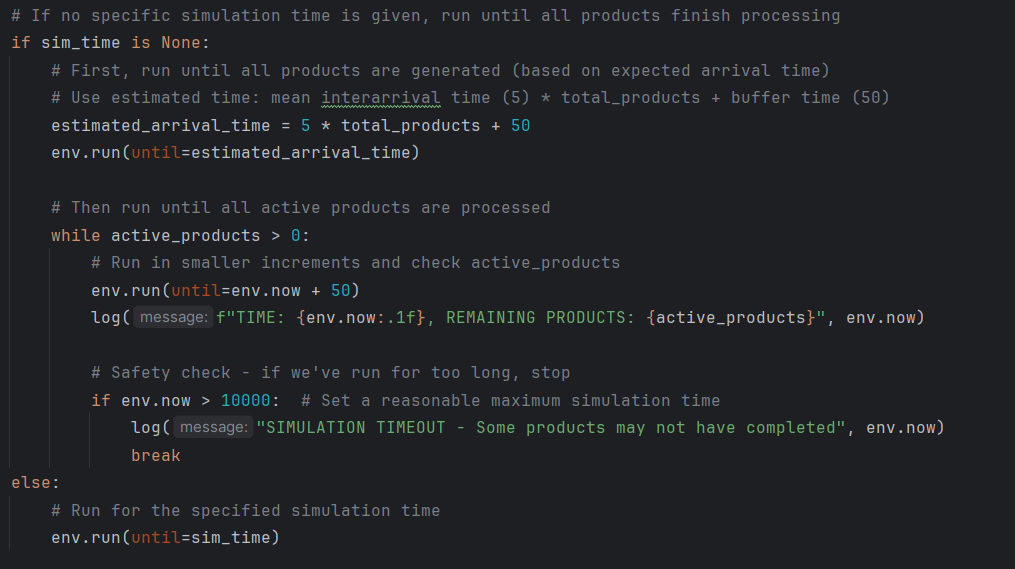
**🔍 Detailed Improvements**

**1. Flexible Simulation Termination**

Instead of running for a fixed time period that might be too short or too long, the enhanced version runs until all specified products have been processed:  
```python  
# Enhanced version  
def run\_simulation(sim\_time=None, total\_products=100):  
 # ...initialization...  
 env.process(product\_arrival(env, machines, total\_products))  
  
 if sim\_time is None:  
 estimated\_arrival\_time = 5 \* total\_products + 50  
 env.run(until=estimated\_arrival\_time)  
 while active\_products > 0:  
 env.run(until=env.now + 50)  
 log(f"TIME: {env.now:.1f}, REMAINING PRODUCTS: {active\_products}", env.now)  
 if env.now > 10000: # Safety timeout  
 log("SIMULATION TIMEOUT", env.now)  
 break  
 else:  
 env.run(until=sim\_time)  
 return machines, env.now  
```

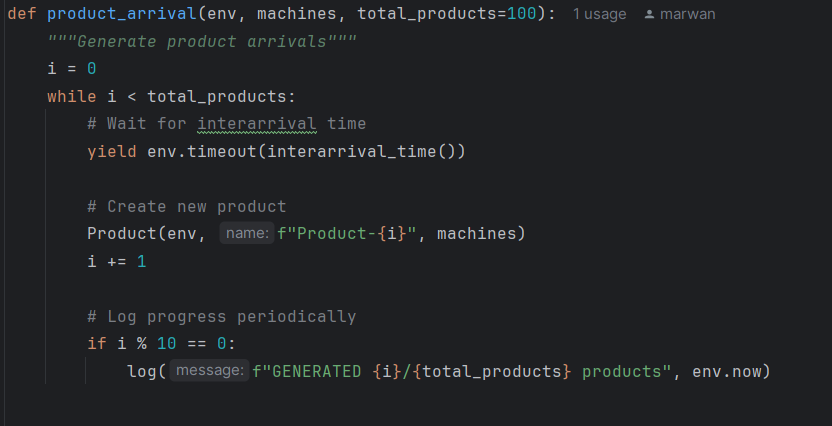
A screenshot of a computer program

AI-generated content may be incorrect.



**2. Configurable Number of Products**

```python  
# Enhanced version  
def product\_arrival(env, machines, total\_products=100):  
 i = 0  
 while i < total\_products:  
 yield env.timeout(interarrival\_time())  
 Product(env, f"Product-{i}", machines)  
 i += 1  
 if i % 10 == 0:  
 log(f"GENERATED {i}/{total\_products} products", env.now)  
```



**3. Enhanced Machine Failure Logic**

```python  
# Enhanced version  
def break\_machine(self):  
 while True:  
 yield self.env.timeout(failure\_time())  
 if not self.broken and self.busy: # Only fail during operation  
 self.broken = True  
 # ...repair process...  
 self.broken = False  
```

A computer screen shot of a program code

AI-generated content may be incorrect.

**4. Improved Machine State Tracking**

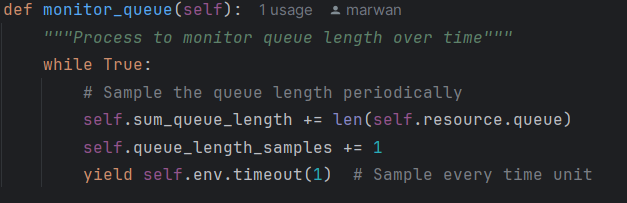
```python  
# Enhanced version  
def update\_busy\_status(self, new\_busy\_state):  
 current\_time = self.env.now  
 if self.busy and not new\_busy\_state: # Becoming idle  
 self.busy = False  
 elif not self.busy and new\_busy\_state: # Becoming busy  
 self.total\_idle\_time += current\_time - self.last\_busy\_state\_change  
 self.busy = True  
 self.last\_busy\_state\_change = current\_time  
```

A computer screen shot of a program

AI-generated content may be incorrect.

**5. Queue Monitoring Process**

```python  
# Enhanced version  
def monitor\_queue(self):  
 while True:  
 self.sum\_queue\_length += len(self.resource.queue)  
 self.queue\_length\_samples += 1  
 yield self.env.timeout(1)  
```



**6. Enhanced Queueing Theory Metrics**

```python  
# Enhanced version - additional metrics  
system\_queueing\_metrics = {  
 'Average Waiting Time': total\_system\_wait / total\_products if total\_products > 0 else 0,  
 'Probability of Wait': products\_that\_waited / total\_products if total\_products > 0 else 0,  
 'Average Queue Length': sum(m.sum\_queue\_length for m in machines) / sum(m.queue\_length\_samples for m in machines) if sum(m.queue\_length\_samples for m in machines) > 0 else 0,  
 'Average Service Time': product\_df['Total Service Time'].mean(),  
 'Average Interarrival Time': avg\_interarrival\_time,  
 'Average Time in System': product\_df['Lead Time'].mean(),  
 'Server Utilization': 1 - (sum(m.total\_idle\_time for m in machines) / (sim\_time \* len(machines)))  
}  
```

A computer code on a black background

AI-generated content may be incorrect.

**7. Interarrival Time Tracking**

```python  
# Enhanced version  
def \_\_init\_\_(self, env, name, machines):  
 # ...initialization...  
 if len(arrival\_times) > 0:  
 interarrival\_times.append(self.start\_time - arrival\_times[-1])  
 arrival\_times.append(self.start\_time)  
 # ...continue with processing...  
```

A screen shot of a computer program

AI-generated content may be incorrect.

**8. Active Products Tracking**

```python  
# Enhanced version  
def \_\_init\_\_(self, env, name, machines):  
 # ...initialization...  
 global active\_products  
 active\_products += 1  
 # ...  
  
def process(self):  
 # ...processing code...  
 global active\_products  
 active\_products -= 1  
```

A screen shot of a computer

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

**💡 Benefits**

The enhanced simulation provides several advantages:

- Greater Realism: Machines fail only during operation

- Flexible Configuration: Control product counts and simulation termination

- Richer Analytics: Comprehensive queueing theory metrics

- Better Monitoring: Continuous queue monitoring and state tracking

- Precise Control: Simulation terminates when all products complete processing

**📊 Simulation Outputs**

- Detailed event logs

A screenshot of a computer

AI-generated content may be incorrect.

- Product metrics (waiting times, processing times, lead times)

A graph of a bar chart

AI-generated content may be incorrect.

- Machine metrics (utilization, availability, downtime)

A screenshot of a computer

AI-generated content may be incorrect.

- System queueing metrics (probabilities, average times)

A screenshot of a computer

AI-generated content may be incorrect.

- Bottleneck identification

A screenshot of a computer

AI-generated content may be incorrect.

**🚀 Getting Started**

```python  
from enhancedVersion import run\_simulation, analyze\_results  
  
# Run simulation with 200 products  
machines, sim\_time = run\_simulation(total\_products=200)  
  
# Analyze results  
results = analyze\_results(machines, sim\_time)  
  
# Access metrics  
print(f"Average Lead Time: {results['system\_metrics']['Average Lead Time']}")  
print(f"System Efficiency: {results['system\_metrics']['System Efficiency (%)']}")  
print(f"Bottleneck: {results['bottleneck']}")  
```

**🔬 Conclusion**

The enhancements in enhancedVersion.py transform a basic simulation into a powerful analytical tool for production line optimization. The improved metrics, flexible configuration, and realistic modeling make it suitable for complex manufacturing system analysis, bottleneck identification, and performance optimization.