TIES481 – Simulation Assignment 2

Process-Based Hospital Simulation Using SimPy

Mohammad Ghafouri Varzaneh & Bahous Amine

November 16, 2025

1. Introduction

This document describes the implementation of a process-based simulation model for a hospital surgery unit, following the requirements of Assignment 2 of TIES481. The model is implemented in Python using the SimPy discrete-event simulation framework.

The objective is to model the flow of identical patients through three capacity-limited stages:

- Preparation (3 units)
- Operating theatre (O units, where O = 1 in this assignment)
- Recovery (3 beds)

All patient processing times and interarrival times follow exponential distributions.

2. System Description

Each patient undergoes the following sequence:

$$\text{Arrival} \rightarrow \text{Preparation} \rightarrow \text{Surgery} \rightarrow \text{Recovery} \rightarrow \text{Exit}$$

If a stage is fully occupied, the patient waits in the corresponding queue:

- Entrance queue (before preparation)
- OR queue (before surgery)
- Recovery queue (before recovery)

2.1 Stochastic Modeling

All random times follow Exponential($1/\mu$):

Process	Mean Time (min)
Interarrival	25
Preparation	40
Surgery	20
Recovery	40

These are generated using Python's `random.expovariate()`.

3. Model Implementation

The simulation is implemented using SimPy's process-oriented paradigm.

3.1 Components

- **Config class**: contains all model parameters.
- **Patient class**: each patient stores individual preparation, surgery, and recovery times.
- **SimPy Resources**:
  - `prep = simpy.Resource(env, capacity=3)`
  - `or = simpy.Resource(env, capacity=cfg.O)`
  - `rec = simpy.Resource(env, capacity=3)`
- **Processes:**
  - `patient_generator`: generates patients with exponentially distributed inter-arrival times.
  - `patient_flow`: the full patient lifecycle.
  - `monitor`: periodically records queue length and OR status.

3.2 Patient Lifecycle

The main process executed for each patient is:

1. Wait for and seize a preparation unit.
2. After preparation, request the operating theatre.
3. During surgery, the OR remains occupied until a recovery bed is available.

4. After surgery, request a recovery bed.

5. Once a bed is obtained, the OR is released.

6. Undergo recovery and then exit.

This logic ensures correct blocking behavior: the OR is not released until the patient transfers to recovery.

3.3 Monitoring

A dedicated monitoring process executes every 5 minutes:

- records entrance queue length,
- records whether the OR is busy.

This enables the computation of approximate OR utilization and average entrance queue length.

4. Output Metrics

The following quantities are collected:

- **Throughput time**: time from arrival to the end of recovery.
- **Entrance queue length**: time-averaged length of the preparation queue.
- **OR utilization**: computed in two ways:
  - exact (integrating OR busy periods),
  - monitored (sampling every 5 minutes).
- **Number of completed patients**.

5. Simulation Parameters

The baseline scenario uses:

- $P = 3$ preparation units
- $O = 1$ operating theatre unit
- $R = 3$ recovery beds
- Simulation horizon: 20 000 minutes
- Monitoring interval: 5 minutes

6. Sample Results

A representative run (seed = 123) yields:

```
simulation_time: 20000 minutes
n_completed: 555 patients
avg_throughput_time: 3387.58 minutes
avg_entrance_queue_length: 138.99 patients
or_utilization_exact: 0.55 (fraction)
or_utilization_monitored: 0.55 (fraction)
```

Interpretation

- The OR is the bottleneck of the system.
- Long entrance queues form due to congestion at the OR.
- Throughput time increases significantly due to exponential variability and waiting at bottlenecks.

7. Modifiability

The entire model is parameterized and easy to extend:

- Change the number of units (P, O, R)
- Change mean service or arrival rates
- Add patient types (needed for Assignment 3)
- Run replications or sensitivity studies

8. Conclusion

The implemented SimPy model satisfies all requirements of Assignment 2. It correctly represents patient flow through preparation, surgery, and recovery stages, handles blocking and queueing behavior, and provides all required performance metrics.

This model forms the basis for future extensions, including multiple patient types and experimental analysis.