Step 1 – Conceptual (Pen & Paper) Model: Surgical Unit with Emergency Cases

System Description

The system models the flow of patients through a surgical unit consisting the three main steps: **preparation → surgery → Recovery.**

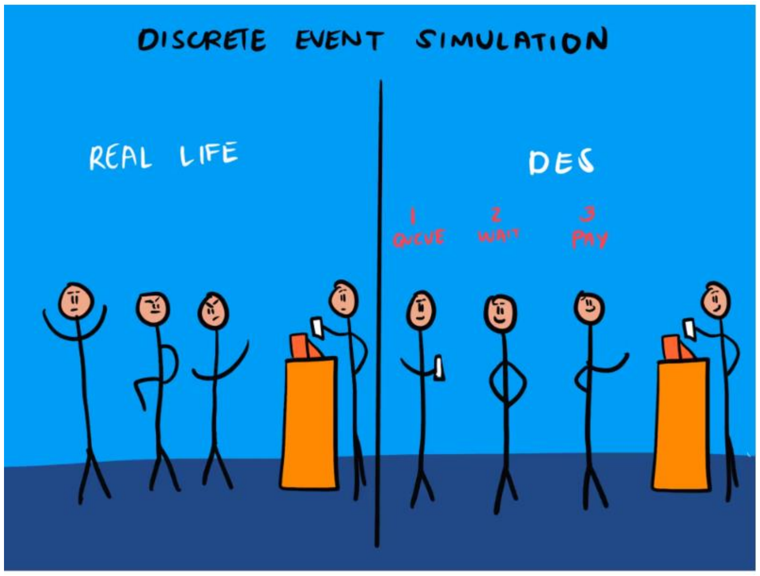
Both the preparation and recovery areas have limited **capacity,** and the surgery room becomes **blocked** when recovery beds are full. The model also incorporates **staff shifts,** which affect the availability of preparation, surgery, and recovery resources over time.

1. Event Based Simulation

Introduction

A discrete-event simulation (DES) models the operations of a system as a (discrete) sequence of events in time. Each event occurs at a particular instance in time and marks a change of state in the system.[2]

Event simulation, particularly discrete-event simulation, is a fundamental technique in modeling and analyzing complex systems by representing system behavior as sequences of discrete events occurring at specific points in virtual time.



Core Concepts and Terminology

Discrete-event simulation uses a discrete-state model pf the system for simulation (is like a system in certain “boxes” or “states”, and it jumps from one box to another. It doesn’t change smoothly – it changes in steps.[2]

Entity: A distinct object or component within the system that interacts with other components or evolves over time. It can be physical objects (e.g., machines, people) or conceptual items (e.g., orders, tasks)[2]

State: A collection of attributes representing the system's entities at a specific point in time. The state of the system changes. [2]

Event: An occurrence at a specific point in time that may alter the system's state. For example, this could be an equipment failure, a new request for service, or a traffic light change. Events are the primary drivers of state changes in discrete-choice simulations.[2]

Queue: A waiting line where entities temporarily reside until specific conditions are met or a resource becomes available. This could be a literal line that patients await their next medication injection while at a hospital. Queues are often used to model delays or bottlenecks in the system.[2]

How does discrete event simulation work?

If you're wondering how to build a discrete event simulation model, the process typically starts with identifying entities, events, and system logic.

1. Define the entities (e.g., people, products, patients).
2. Specify the process flow using logic and rules.
3. Assign resources and constraints.
4. Run the simulation to mimic the flow of events over time.
5. Analyze the output to find bottlenecks and optimize performance.

Event-Based Conceptual Model of the Surgical Unit (with Doctor Availability Scheduling)

System overview

The system models patient flow through preparation, surgery, and recovery phases. Limited resources include preparation places, recovery beds, and the operation theatre. A new twist feature doctor availability restricts when surgeries can begin, simulation shift-based scheduling.

Entities and resources

Entity	**Description**
Patient	The main entity moving through preparation, surgery, and recovery.
Doctor	Represents human resource availability (on shift/ of shift)
PrepPlaces	Limited number of preparation spots.
SurgeryRoom	The operation theatre, used only when both room and doctor are available
RecoveryBeds	Limited number of recovery beds

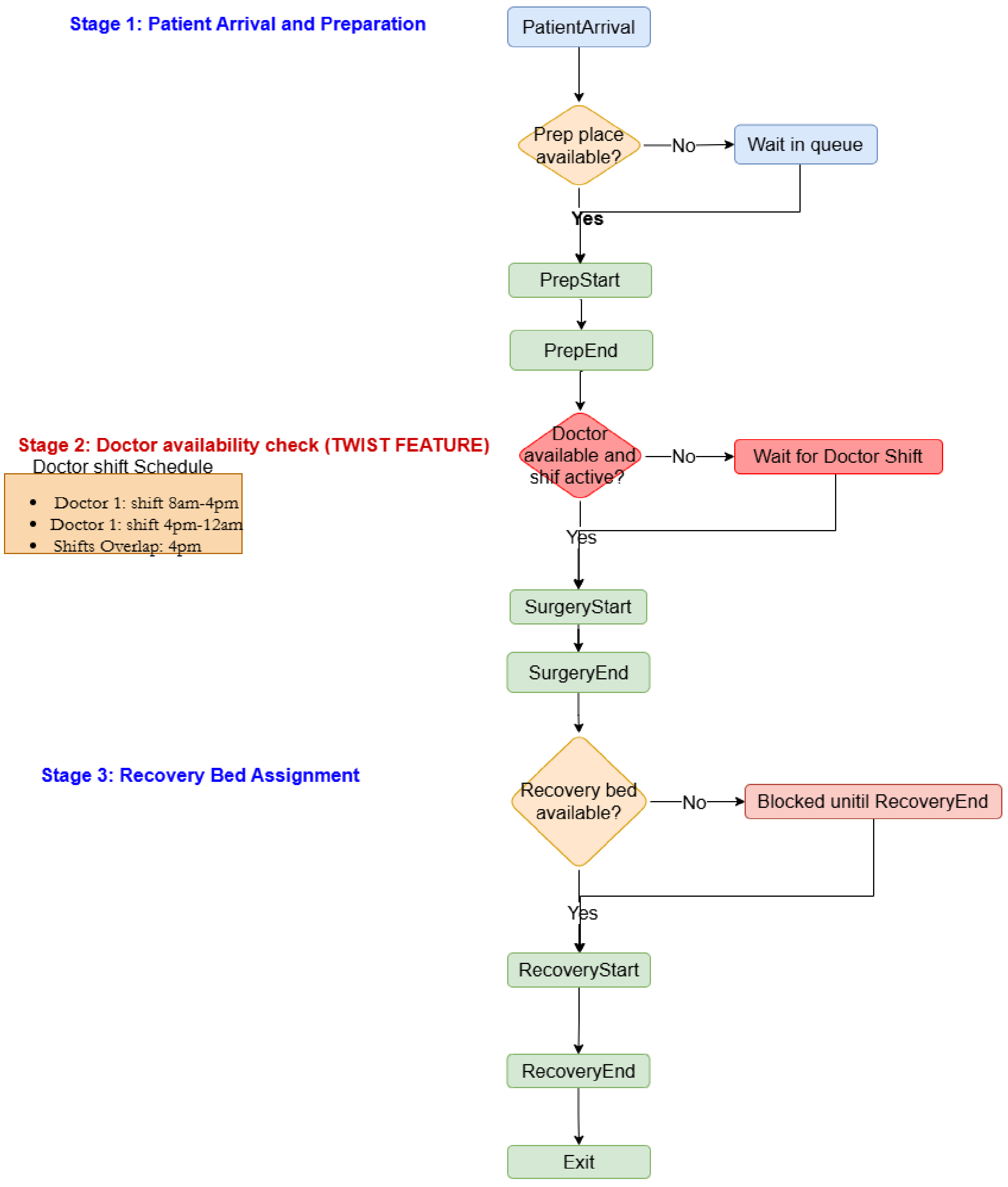
State variables

Variable	Description
n_prep_busy	Number of prep places currently occupied
n_recovery_busy	Number of recovery beds occupied
surgery_state	0 = idle, 1 = in use, 2 = blocked
doctor_state	0 = off shift, 1= available, 3 = busy
queus	Patients waiting for prep or surgery
clock	Simulation time

Discrete Events

Event	Description	State change	Triggers
PatientArrival	New patient enters system	Adds to prep queue or stats prep	Next PatientArrival, possibly PrepStart
PrepStart/ PrepEnd	Start/finnish prep	Occupies or frees prep place	SurgeryStart (if doctor available)
SurgeryStart	Starts surgery when doctor and theatre available	Sets surgery and doctor busy	SurgeryEnd
SurgeryEnd	Finnishes operation	Frees doctor (if still shift), may block if recovery full	RecoveryStart or Blocked state
RecoveryStart/ RecoveryEnd	Occupy or release recovery beds	Changes in recovery occupancy	If blocked, triggers SurgeryResume
DoctorShiftStart	Doctor becomes available for surgeries	Sets dector_state = available	May trigger SurgeryStart for waiting patient
DoctorShiftEnd	Doctor leaves; ongoing surgery finishes, but no new surgry starts	Sets doctor_stat = off shift	Wait for next DoctorShiftStart

Event Flow Diagram (Conceptual)



Event Flow and State transitions

1. PatientArrival → Resource Check (Prep)
   if prep place available: trigger PrepStart. Otherwise: queue until prep ends elsewhere.
2. PrepEnd → Resource Check (Doctor)
   TWIST: if doctor unavailable/off-shift: enter "Wait for DoctorShift" state. Otherwise trigger SurgeryStart
3. SurgeryEnd → Resource Check (Recovery)Key Outputs
   If recovery bed available: trigger RecoveryStart. Otherwise: surgery room remains blocked until bed frees.
4. RecoveryEnd → Exit
   Patient leaves system. Recovery bed becomes available for next patient

Metrics	**Description**
Throughput Time	Total time per patient from arrival to exit
Surgery utilization	Fraction of time theatre is in use (when doctor available)
Doctor idle time	Fraction of time doctor us on shift but not operation
Blocking probability	Probability surgery room is blocked because recovery full
Queue lengths	Number of patients waiting for surgery when doctor off-shift

Summary

In this event-based model, each event modifies the system's state by allocating or releasing limited resources such as preparation places, the operation theatre, and doctors.

The additional availability of doctor introduces scheduling constraints that influence when surgeries can start, potentially increasing patient waiting times or idle operating-room periods.

1. *Stewart Robinson (2004). Simulation – The practice of model development and use. Wiley.*
2. *SoftwareSim. (2022, March 12). A gentle introduction to discrete event simulation. Retrieved December 3, 2024, from* ***https://softwaresim.com/blog/a-gentle-introduction-to-discrete-event-simulation/***

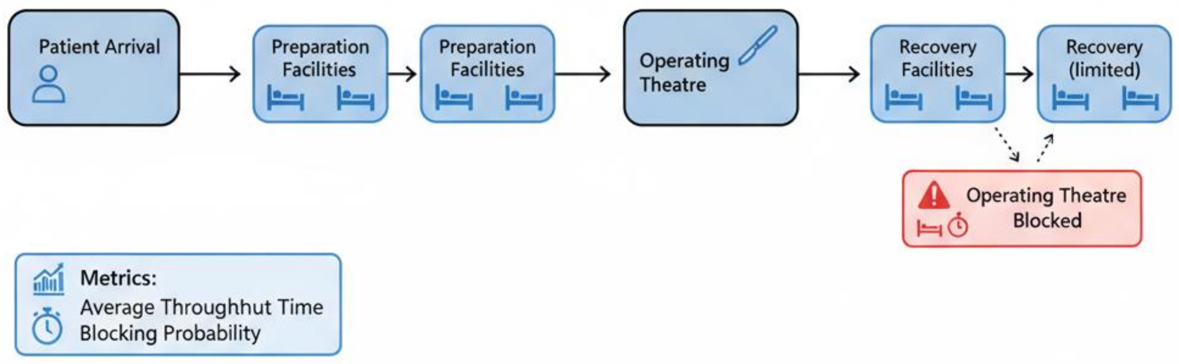
1. Process Modelling Goal

To model a continuous stream of patients that go through ***preparation → operation → recovery.*** Main performance measures:

- Average throughput time per patient (arrival → departure).
- Probability (and fraction of time) the operating theatre is blocked because no recovery bed is available.

Additional feature: Staff shifts / time-of-day effects that change effective capacities and service rates over time (e.g., fewer prep staff at night, slower service during shift handovers). This creates time-varying resource availability and arrival rates.

Modelling paradigm: Process-based — each patient is an active process following a lifecycle, requesting and releasing resources (pools) as needed. Resources have capacities that can change over time (shifts).



2. System components and variables

2.1 Resources (shared)

Name	Type	Description / role	Initial capacity (example)
PrepPool	Resource pool	Preparation stations; several identical units	3
Theatre	Resource	Operating theatre; single unit	1
RecoveryPool	Resource pool	Recovery beds/rooms; limited units	2

2.2 Entities

- **Patient** — active process. Attributes: id, arrival_time, type (optional), start_prep, end_prep, start_op, end_op, start_rec, end_rec, departure_time, priority_flag (if needed).

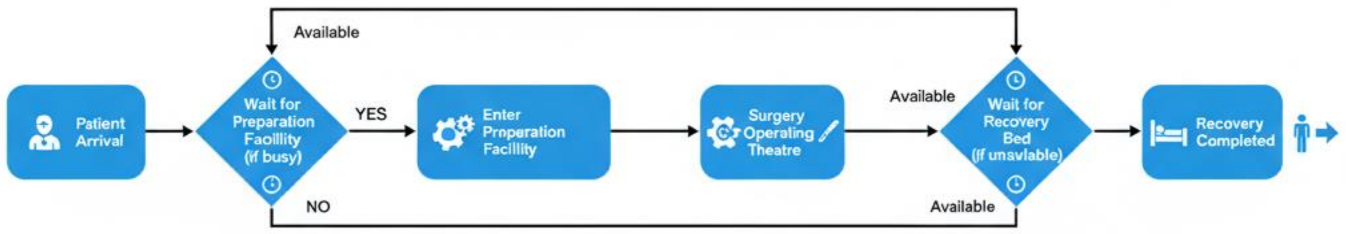
2.3 Time-varying elements (for shifts/time-of-day)

- capacity_prep(t): number of active prep units at time t (e.g., 3 daytime, 2 night).
- service_rate_prep(t): average preparation time distribution parameters depending on time of day.
- capacity_recovery(t): may be static or affected by staff availability (e.g., elective closures).
- arrival_rate(t): patient arrival intensity (e.g., higher daytime, lower night).

2.4 State variables & counters

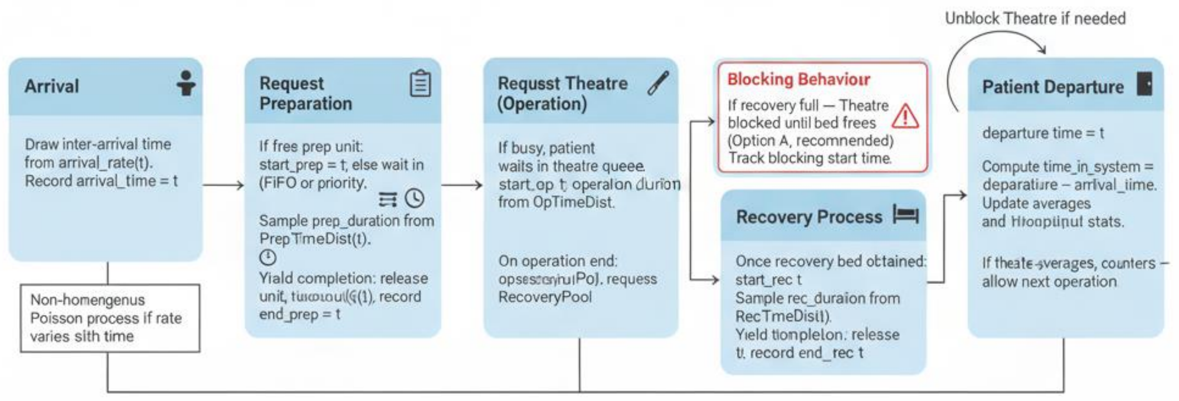
- num_in_prep_queue, num_in_theatre_queue, num_in_recovery_queue
- num_busy_prep, theatre_busy (0/1), num_busy_recovery
- total_patients_arrived, total_patients_departed
- sum_time_in_system (for mean throughput)
- theatre_blocked_time (cumulative time theatre is blocked)
- num_blocking_events (count of times theatre attempted to finish operation but no recovery bed)
- Time stamps for events, per-patient records (for distributions)

3. Patient process: lifecycle sketch


Available
Patient Arrival
Wait for Preparation Faoillity (if busy)
YES
Enter Proparation Facillity
Surgery Operating Theatre
Available
Wait for Recovery Bed (if unavlable)
Recovery Completed
NO
Available

Each patient is a process that performs the following ordered sequence. For clarity, each action specifies what resource it requests, what it does while holding it, and what it does on release.

Lifecycle steps (detailed)


Unblock Theatre if needed
Arrival
Draw inter-arrival time from arrival_rate(t). Record arrival_time = t
Non-homengenus Poisson process if rate varies slth time
Request Preparation
If free prep unit: start_prep = t; else wait in (FIFO or priority.
Sample prep_duration from PrepTimeDist(t).
Yiald completion: release unit, record end_prep = t
Requst Theatre (Operation)
If busy, patient waits in theatre queee. start_op t; operalon durion from OpTimeDist.
On operation end: request RecoveryPool
Blocking Behavioır
If recovery full — Theatre blocked until bed frees (Option A, reconmended) Track blocking start time.
Recovery Process
Once recovery bed obtained: start_rec t Sample rec_duraion from RecTmeDist(). Yield tionpleton: release t, record end_rec t
Patient Departure
departure time = t
Compute time_in_system = deparatiure – arrival_time. Update averages and Hospital stats.
If theate averages, counters – allow next operation

1. **Arrival**
    - Draw inter-arrival time from arrival_rate(t) (nonhomogeneous Poisson process if arrival_rate varies by time).
    - Record arrival_time = t.
2. **Request preparation**
    - yield PrepPool.request() — if a free prep unit exists, patient starts immediately; otherwise queues (FIFO or priority).
    - When granted: start_prep = t.
    - Preparation duration sampled from PrepTimeDist(t) (may depend on time of day or staff skill).
    - yield timeout(prep_duration).
    - On completion: release prep unit (PrepPool.release()), record end_prep = t.
3. **Request theatre (operation)**
    - yield Theatre.request(). If theatre busy, patient waits in theatre queue.
    - When granted: start_op = t.
    - Operation duration sampled from OpTimeDist (could be constant or stochastic).
    - yield timeout(op_duration).
    - On **Operation End**:
        - Attempt to request RecoveryPool for transfer.
        - If a recovery bed is available immediately:

        - start_rec = t; release Theatre (theatre becomes free); proceed to recovery stage.
    - **If all recovery beds are occupied**:
        - **Blocking behaviour:** Option A (recommended): patient *remains in Theatre* holding the theatre resource until a recovery bed is free. Mark theatre as **blocked**. Track blocking start time.
        - Option B (alternate): patient is moved to a theatre-adjacent queue / holding area (still occupying operating theatre or special holding resource). Implementation uses whichever matches system semantics — in this assignment we use Option A (theatre held until recovery bed free), because the task explicitly measures theatre blocked time.

4. **Recovery**
    - Once a recovery bed is obtained (either immediately after operation, or when one becomes free and theatre is unblocked): start_rec = t.
    - Recovery duration sampled from RecTimeDist (may vary by surgery or by time of day).
    - yield timeout(rec_duration).
    - On completion: release RecoveryPool, end_rec = t.
5. **Departure**
    - departure_time = t.
    - Update statistics: time_in_system = departure_time - arrival_time; increment counters, add to sums for averages. If this release unblocks theatre and theatre is waiting, allow theatre to start next patient.

Notes on blocking

- When theatre finishes operation but cannot release patient to recovery, theatre remains occupied by the patient and cannot start another operation. The model must:
    - Record blocked_start when theatre cannot hand off patient.
    - Record blocked_end when recovery bed becomes available and transfer occurs.
    - Update theatre_blocked_time += blocked_end - blocked_start.
    - Increase num_blocking_events by 1.

5. Essential data to collect & monitoring plan

Per-patient data (record for each patient)

- id, arrival_time, start_prep, end_prep, start_op, end_op, start_rec, end_rec, departure_time
- blocked_flag (true if operation end had to wait), blocked_duration (if any)
- patient_type (optional — e.g., elective vs emergency)

System-level time series (sampled at intervals or updated on events)

- num_waiting_prep(t), num_busy_prep(t)
- num_waiting_theatre(t), theatre_busy(t), theatre_blocked(t)
- num_busy_recovery(t), num_waiting_recovery(t)
- capacity_prep(t), capacity_recovery(t) (to verify shifts)

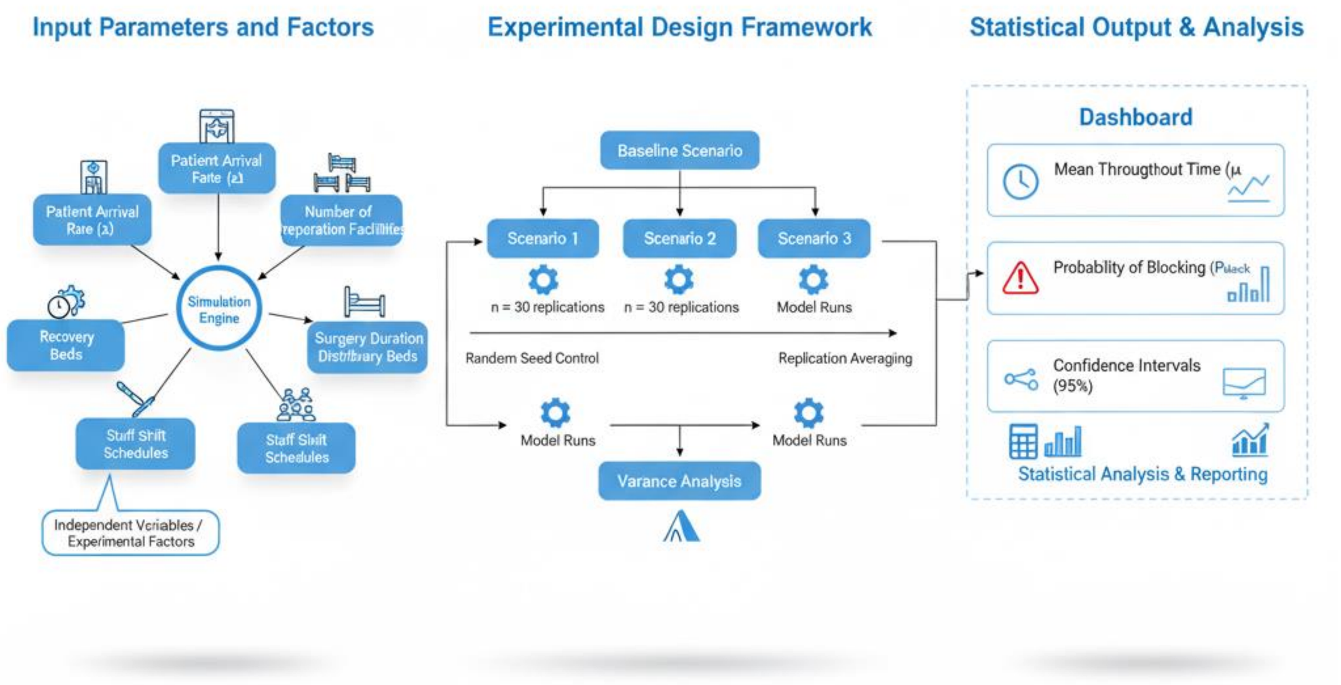
Aggregated metrics (computed after replication)

- **Average throughput time** = mean of departure_time - arrival_time.
- **Blocking probability**:
  - Option 1: fraction of completed operations that experienced blocking = num_blocking_events / total_operations.
  - Option 2: fraction of simulation time theatre is blocked = theatre_blocked_time / simulated_time.
  - Both are useful: Option 1 is event-based; Option 2 measures utilization impact.
- **Queue statistics**: average and max queue lengths per queue.
- **Resource utilization**: util_prep = total_busy_time_prep / (sim_time * nominal_prep_capacity) — care with time-varying capacity; compute utilization per shift or as integral over time: $\int busy(t)\,dt / \int capacity(t)\,dt$.
- **Delay distributions**: histograms of waiting times for prep, theatre, and recovery.
- **Time-of-day breakdowns**: compute metrics separately for day and night periods.

Logging for reproducibility

- Recording random seeds and replication IDs, shift schedule, distribution parameters, and run length.

6. Suggested statistical settings and experimental design



6.1 Warm-up and run length

- Use a **warm-up period** to avoid bias from initial empty system (common in open arrival processes). Example: discard first 8 hours (or until system reaches steady-state for stationary parts). For time-varying (day/night) experiments, prefer **multiple repeating cycles** (e.g., simulate several 24-hour days) and discard the first day.
- **Run length**: simulate N_days (e.g., 30 days) per replication to capture daily patterns, or simulate a continuous 720 hours if longer horizon desired.

6.2 Replications

- Use multiple independent replications (e.g., 30–50) with different RNG seeds to construct confidence intervals.
- For each metric compute sample mean and 95% CI.

6.3 Scenario experiments

Run these scenarios and compare outcomes:

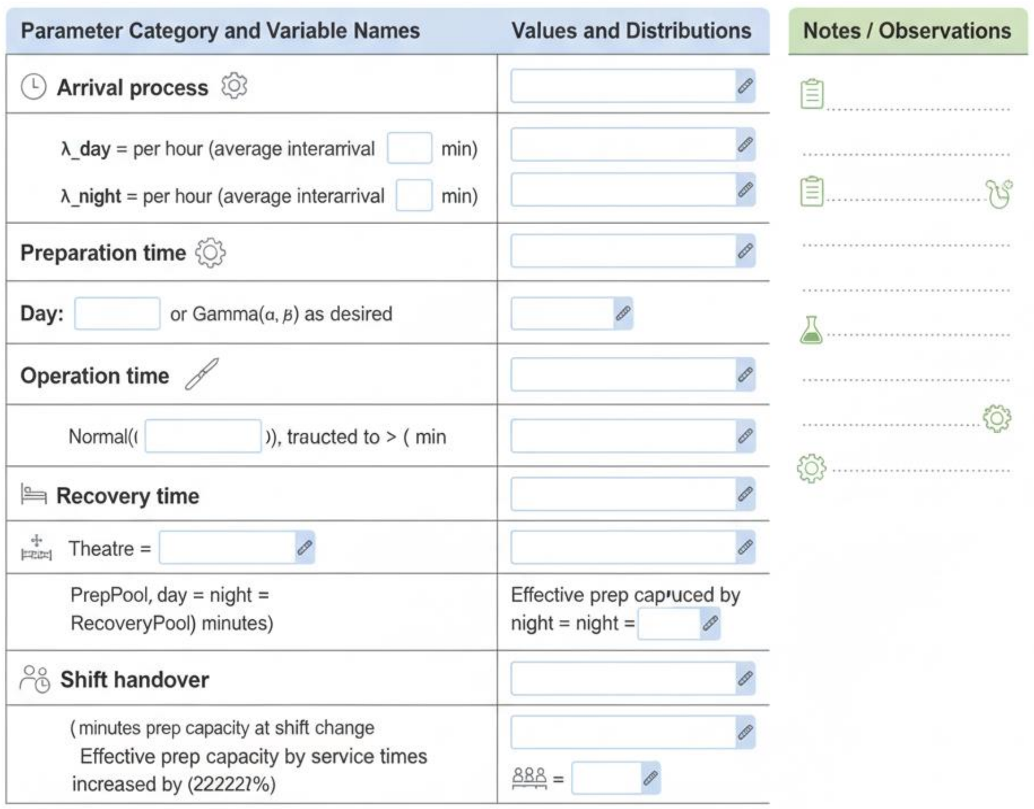
1. **Baseline**: fixed capacities, no shifts, constant arrival rate (for sanity check).
2. **Time-of-day arrivals**: time-varying arrivals but fixed capacities.

3. **Shifts**: time-varying capacities + time-varying service performance (the proposed feature).
4. **Sensitivity**: vary recovery capacity (e.g., 1,2,3 beds) to see blocking sensitivity.
5. **Handover effect**: include short handover slowdowns and measure their effect.

6.4 Key comparisons

- Throughput time (mean and CI) across scenarios.
- Blocking probability by scenario.
- Peak queue length and utilization during day vs night.
- Impact of staff reductions on theatre blocked time.

6. Example parameter set (fillable — use for experiments)



Parameter Category and Variable Names	Values and Distributions	Notes / Observations
Arrival process		
λ_day = per hour (average interarrival min)		
λ_night = per hour (average interarrival min)		
Preparation time		
Day: or Gamma(α, β) as desired		
Operation time		
Normal(( )), traucted to > (min		
Recovery time		
Theatre =		
PrepPool, day = night = RecoveryPool) minutes)	Effective prep cap'uced by night = night =	
Shift handover		
(minutes prep capacity at shift change Effective prep capacity by service times increased by (22222?%)	888 =	

- Arrival process:
  - λ_day = 6 per hour (average interarrival 10 min)
  - λ_night = 2 per hour (avg interarrival 30 min)
- Preparation time:
  - Day: Exp(mean=20 minutes) or Gamma(α,β) as desired
  - Night: Exp(mean=25 minutes)
- Operation time:
  - Normal(mean=60 min, sd=10 min) truncated to >20 min
- Recovery time:
  - Exp(mean=120 minutes)
- Capacities:
  - PrepPool: day=3, night=2
  - Theatre: 1
  - RecoveryPool: day=2, night=1
- Shift handover:
  - 10 minutes at shift change where effective prep capacity reduced by 1 or service times increased by 20%.