Objective
- Learn the basics of discrete-event simulation
- We will focus on a simple single-server service center. Possible examples:
  - Convenience store: customers, cashiers
  - Airport: runways, airplanes

What is Discrete-Event Simulation (DES)?
- Modeling of a system as it evolves over time by a representation where the state variables change instantaneously at separated points in time
  - More precisely, state can change at only a countable number of points in time
  - These points in time are when events occur
- What is an event? Instantaneous occurrence that may change the state of the system
  - Arrival of a customer
  - Service completion (and departure) of a customer
  - End of simulation (a “fake” event)
Single-Server Service Center

- Performance measures of interest:
  - Average customer waiting time
  - Average number of customers in queue
  - Average server utilization
- How do we simulate this system and obtain measures of interest?
  - Need to simulate "time" ...

Time Advance Mechanism

- Simulation clock: Variable that keeps the current value of (simulated) time in the model
  - Must decide on, be consistent about, time units
  - Usually no relation between simulated time and (real) time needed to run a model on a computer
- Two approaches for time advance
  - Fixed-increment time advance
  - Next-event time advance

Fixed-Increment Time Advance

- Events occur at a fixed increment
- Events occurring between time increments must be moved to an increment boundary
- Simple to implement, but not an accurate realization of occurrence of events
Next-event Time Advance

- Initialize simulation clock to 0
- Determine times of occurrence of future events - event list
- Clock advances to next (most imminent) event, which is executed
  - Event execution may involve updating event list
- Continue until stopping rule is satisfied (must be explicitly stated)
- Clock "jumps" from one event time to the next, and doesn't "exist" for times between successive events ... periods of inactivity are ignored

Next-event time-Advance (cont'd.)

- Consider the single-server service center example
  - $t_i$ = time of arrival of $i$th customer ($t_0 = 0$)
  - $A_i = t_i - t_{i-1} =$ interarrival time between $(i-1)$st and $i$th customers
  - $S_i =$ time spent serving the $i$th customer
  - $D_i =$ delay in queue of $i$th customer
  - $C_i = t_i + D_i + S_i =$ time $i$th customer completes service and departs

Want to write a next-event program?

- Determine the events and understand what happens when it occurs
- Generate events and keep a time-ordered list of the events
- Based on the time-ordered list, we will have to schedule events
- Track information of interest during the simulation
- Finally, generate report when simulation ends
Components of a DES Program

- Simulation clock - current value of simulated time
- System state - variables to describe state
  - Server status, number in queue, arrival times, etc
- Event list - times of future events for each event type
- Statistical counters - to accumulate performance measures
  - Waiting time in queue, server utilization, ...

Components of a DES Program (cont'd.)

- Initialization routine
  - Start simulation at time 0
- Timing routine
  - Determines next event time, type; advances clock
- Event routines
  - Carry out logic for each event type
- Library routines
  - Utility routines to generate random variates, etc.
- Report generator
- Main program
  - Ties routines together, executes them in the correct order

Organization of a DES Program
**DES Program Flow**

- **Start**
  - Initialization Routine: 1. Set clock = 0, 2. initialize state & counters, 3. Initialize event list, 4. Return to main program
  - Main Routine

---

**DES Program Flow**

- **Main Routine**
  - Timing Routine: 1. Determine next event type i, 2. Advance simulation clock, 3. Return to main program

---

**DES Program Flow**

- **Main Routine**
  - Event routine i: 1. update system state, 2. update counters, 3. generate future events & add them to the event list
  - Library Routine
### Hand Simulation of a Single Server Service Center

- Interarrival times (all times are in minutes): 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
- Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...
- $n = 6$ delays in queue desired
- "Hand" simulation:
  - Display system, state variables, clock, event list, statistical counters ... all after execution of each event
  - Use above lists of interarrival, service times to "drive" simulation
  - Stop when number of delays hits $n = 6$, compute output performance measures

### Performance Measures

- "Expected" average delay in queue (excluding service time) of the $n$ customers completing their delays:
  \[ \hat{d}(n) = \frac{1}{n} \sum_{i=1}^{n} D_i \]

- Expected average number of customers in queue (excluding any in service): A continuous-time average
  \[ \hat{q}(n) = \frac{1}{T(n)} \int_0^T \hat{Q}(t) \, dt = \frac{1}{T(n)} \sum_{i=1}^{n} i \hat{T}_i \]

- Expected utilization (proportion of time busy) of the server: Another continuous-time average
  \[ \hat{u}(n) = \frac{1}{T(n)} \int_0^T \hat{B}(t) \]
Time = 0

Interarrival times: 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...

Time = 0.4

Interarrival times: 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 0.7, 0.2, 1.1, 3.7, 0.6, ...

Time = 1.6

Interarrival times: 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 0.7, 0.2, 1.1, 3.7, 0.6, ...
Time = 2.1

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...

Time = 2.4

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...

Time = 3.1

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...
Time = 3.3

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Time = 3.8

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …

Time = 4.0

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, …
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, …
**Time = 4.9**

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...

**Time = 5.6**

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...

**Time = 5.8**

Interarrival times: 0.4, 1.2, 0.5, 1.7, 0.2, 1.6, 0.2, 1.4, 1.9, ...
Service times: 2.0, 0.7, 0.2, 1.1, 3.7, 0.6, ...
Interarrival times: \( x, x, x, x, x, x, x, x, x, \ldots \)
Service times: \( x, x, x, x, 0, 6, \ldots \)

Interarrival times: \( x, x, x, x, x, x, x, x, x, \ldots \)
Service times: \( x, x, x, x, x, x, \ldots \)

Final output performance measures:
- Average delay in queue = \( 5.7/6 = 0.95 \) min./customers
- Time-average number in queue = \( 9.9/8.6 = 1.15 \) customers
- Server utilization = \( 7.7/8.6 = 0.90 \) (dimensionless)

**DES Programming Issues**
- Program termination rules
  - Number of events, total time
- Client balking - leave the queue without service
- Time breaking during event scheduling
  - Choose departure before arrival
  - Two arrivals at the same time? Randomly choose one
  - Other defined rules