Discrete Event Simulations (Part 6)

Dr.Çağatay ÜNDEĞER

Öğretim Görevlisi
Bilkent Üniversitesi Bilgisayar Mühendisliği Bölümü

e-mail : cagatay@undeger.com
cagatay@cs.bilkent.edu.tr

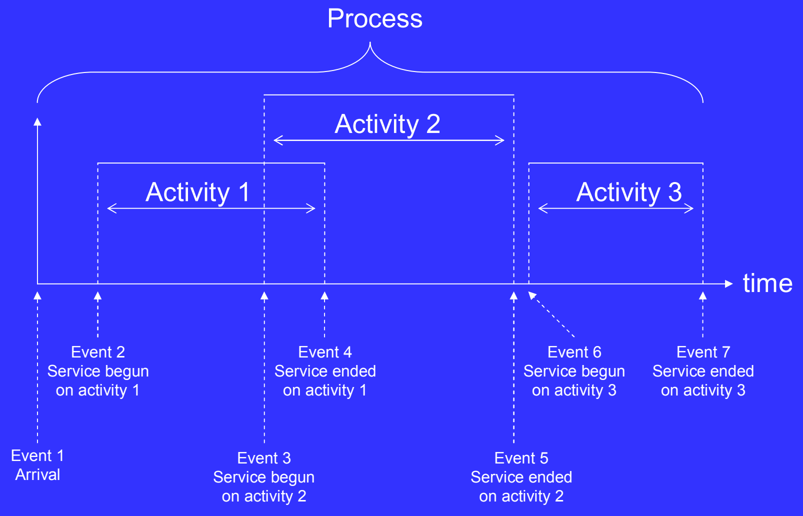
Discrete Event Simulations (Outline)

- Introduction
  - Next-Event Time Advance
  - Arithmetic and Logical Relationships
  - Discrete-Event Modeling Approaches
- Event-Scheduling Approach
- Process-Interaction Approach
- Processes and Resources

Terminology

- Event:
  - A change in state of a system.
- Activity:
  - A pair of events, one initiating and other completing an operation that transforms the state of an entity.
- Process:
  - A collection of events ordered in time, which involves a set of activities.

Terminology



Discrete Event Simulation

- Concerns modeling of a system on a digital computer,
  - Where state changes can be represented by a collection of discrete events.

Next Event Time Advance

- In a discrete-event simulation,
  - A state change implies an event occurs.
  - The states of entities remain constant between successive events.
  - No need to simulate this inactive time in the model.
  - Next-event approach is commonly used to advance time.

Next Event Time Advance

- After all state changes are made at the time corresponding to a particular event,
  - Simulated time is advanced to the time of the next event, and
  - That next event is executed.
- This procedure is repeated until simulation is ended.
- This enables skipping over inactive time, whose passage in real world we are forced to live.

Arithmetic and Logical Relationships

- Two different relationships play significant roles in a simulation system:
  - Mathematical relationships
  - Logical relationships

Arithmetic and Logical Relationships

- Mathematical relationships:
  - Exits between events and attributes associated with entities.
  - For example:
    - Assume that Q is the number of bullets in a gun, and C is the total capacility,
    - If a gun is fired, then Q ← Q – 1
    - If a gun is reloaded, then Q ← C

Arithmetic and Logical Relationships

- Logical relationships:
  - Describes logical relationships in a system.
  - We represent logical behaviors of the system by defining actions that will be taken according to some conditions.
  - For example,
    - In a job processing server, when current job is completed,
      - If new jobs exist in the queue, takes it according to a rule and starts doing,
      - If no job is waiting, remains idle.

A Sample Model: Airline Reservation

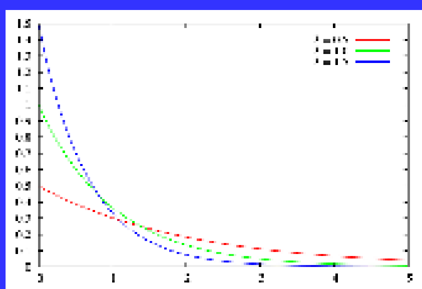
- All customers call on the phone.
- Customers are willing to wait indefinitely for service.
- No restriction exists on the number of customers that can wait on phone at one time.
- If all reservationists are busy when a call occurs then
  - A 9-second recorded message asks the customer to wait.

A Sample Model: Airline Reservation

- The airline office has *m* reservationists.
- Calls occurs at a rate λ and follows exponential distribution.

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & , x \geq 0, \\ 0 & , x < 0. \end{cases}$$

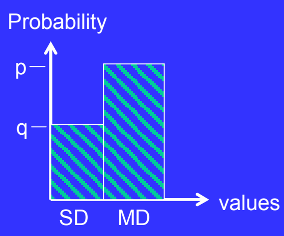
Exponential Distribution



A Sample Model: Airline Reservation

- Customers can either request:
  - A multiple-destination service with probability *p,*
  - A single-destination service with probability *q = 1-p,*

Bernoulli Distribution

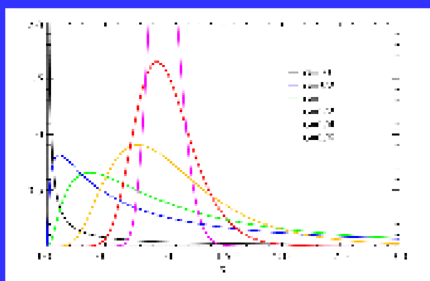


A Sample Model: Airline Reservation

- Multiple-destination service times are random variables that follow Log-Normal distribution with mean *μ*, and standard deviation *σ*.

$$f(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln(x)-\mu)^2}{2\sigma^2}}$$

Log-Normal Distribution

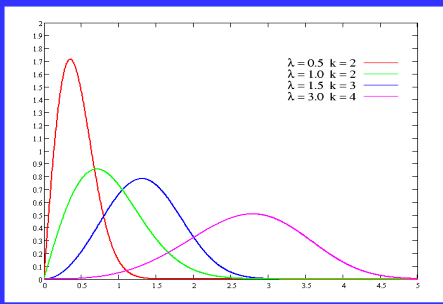


A Sample Model: Airline Reservation

- Single-destination service times are random variables that follow Log-Normal distribution with shape *k*, and scale *λ*.

$$f(x;k,\lambda) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}$$

Weibull Distribution



A Sample Model: Airline Reservation

- Probability distribution of service time for an arbitrarily selected caller is a compound density function:

$f(t) = p$ log-normal$(\mu,\sigma) + (p-1)$ weibull(k,λ)

A Sample Model: Airline Reservation

- The purpose of simulation is:
  - To compare performance of the system with different number of reservationist (*m*) while all the other parameters are fixed.

A Sample Model: Airline Reservation

- Next we need to design implementation details of the model.
- Therefore we define:
  - Entities,
  - Attributes of entities,
  - Probable attribute values,
  - Events, and
  - Event flow.

A Sample Model: Airline Reservation

- Entities, attributes & values in our model are:
  - Customer (Entity):
    - Reservation Type (Attribute):
      - Single-Destination (Value)
      - Multiple-Destination (Value)
    - Status (Attribute):
      - Listening 9-seconds message (Value)
      - Waiting for service (Value)
      - Getting service (Value)
  - Reservationist (Entity):
    - Status (Attribute):
      - Idle (Value)
      - Busy (Value)

A Sample Model: Airline Reservation

- Elementary events in our model are:
  1) Call (customer)
  2) Hang up (customer or reservationist)
  3) Begin service (customer or reservationist)
  4) End service (customer or reservationist)
  5) Begin 9-seconds recorded message (customer)
  6) End 9-seconds recorded message (customer)
  7) File in waiting line (customer)
  8) Remove customer from waiting line (reservationist)
  9) Become busy (reservationist)
  10) Become idle (reservationist)

A Sample Model: Airline Reservation

- How to model event flow (behaviour of the system) in our model depends on which discrete event modeling approach is used.
- Then this behaviour model will be transformed into a computer program.

Discrete-Event Modeling Approaches

- Alternative ways to build discrete-event models:
  - Event-Scheduling Approach:
  - Activity-Scanning Approach
  - Process-Interaction Approach

Discrete-Event Modeling Approaches

- Event-Scheduling Approach:
  - Emphasizes a detailed description of the steps that occur when an individual event takes place.
  - Each type of event naturally has a distinct set of steps associated with it.
    - For example,
      - When a “service ended” event happens,
        - » This may triger a “service begin” event for the next job.

CS-503 23

Discrete-Event Modeling Approaches

- Activity-Scanning Approach:
  - Emphasizes a review of all activities in a simulation to determine which can be begun or terminated at each advance of the clock.
  - A logically equivalent approach to others, but it is computationally less efficient for most simulation scenarios.
  - Therefore, it is not widely used.

CS-503 24

Discrete-Event Modeling Approaches

- Process-Interaction Approach:
  - Emphasizes the progress of an entity through a system from its arrival event to its departure event.
  - It is easier to define a model with processes rather than events.
  - Therefore, it is widely used
    - Although event scheduling approach has more programming control and flexibility.

Event-Scheduling Approach

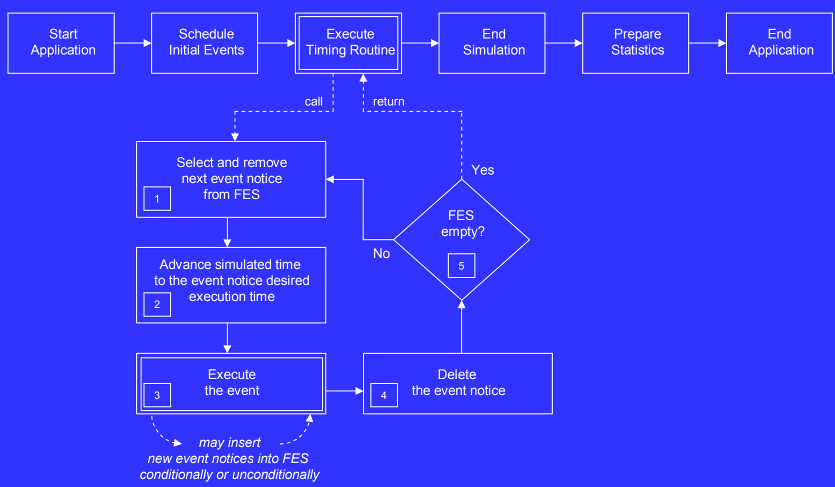
- Every discrete simulation has a set of state variables.
- These state variables change in time by the effect of events.
- Each event signals a change in one or more of the items and resources in a system.
- Event-scheduling approach is built on this concept.

Event-Scheduling Approach (Terminology)

- Event notice:
  - A record describing when a future event is to be executed.
- Future event set (FES):
  - A linked list containing event notices ordered by desired execution time.
- Timing routine:
  - The procedure for maintaining the future event set and advancing simulated time.

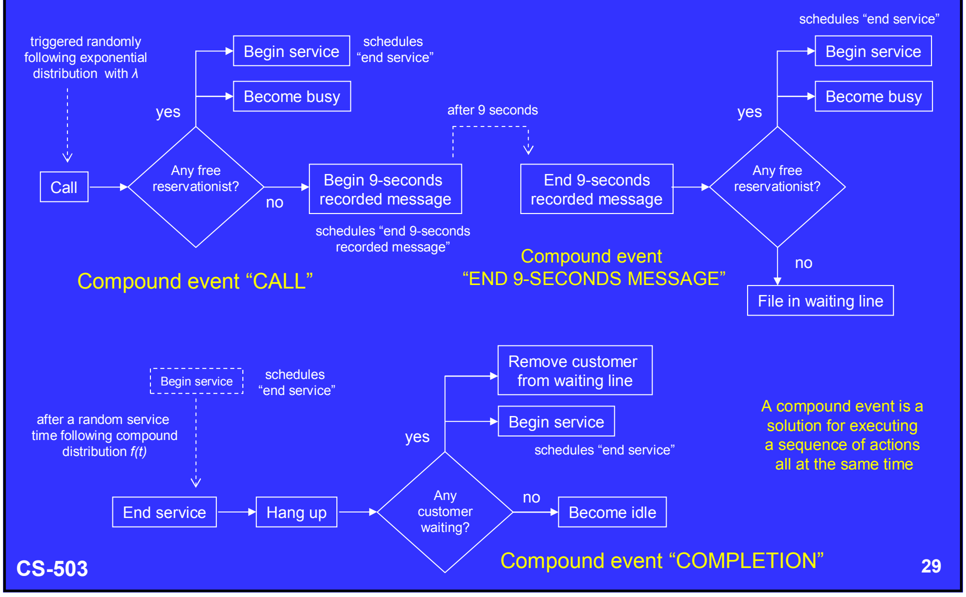
CS-503 27

Event-Scheduling Approach (Timing Routine)



CS-503 28

Event-Scheduling Approach (Airline Reservation Sample)



Event-Scheduling Approach (Airline Reservation Sample)

- System state variables:
  - Remaining time until next call *(S)*
  - Number of busy reservationists *(i)*
  - Number of customers listening to recorded message *(j)*
  - For each caller *m* listening to message :
    - Remaining time until completion of message $(T^c{}_m)$
  - For each busy reservationist *m* :
    - Remaining time until completion of service $(T^r{}_m)$

Event-Scheduling Approach (Airline Reservation Sample)

- System state variables:
  - Remaining time until next call *(S)*
  - Number of busy reservationists *(i)*
  - Number of customers listening to recorded message *(j)*
  - For each caller *m* listening to message :
    - Remaining time until completion of message (T^c_m)
  - For each busy reservationist *m* :
    - Remaining time until completion of service (T^r_m)

- Next event time = min $(S, T^c_1..T^c_j, T^r_1...T^r_i)$

may set priority to event types in order to guide selection from a set of events sharing the same time

Process-Interaction Approach

- Although produces identical results,
  - Differs substantially from the event-scheduling approach.
- Instead of only scheduling events independent of processes that create them,
  - Allows model representation (process).

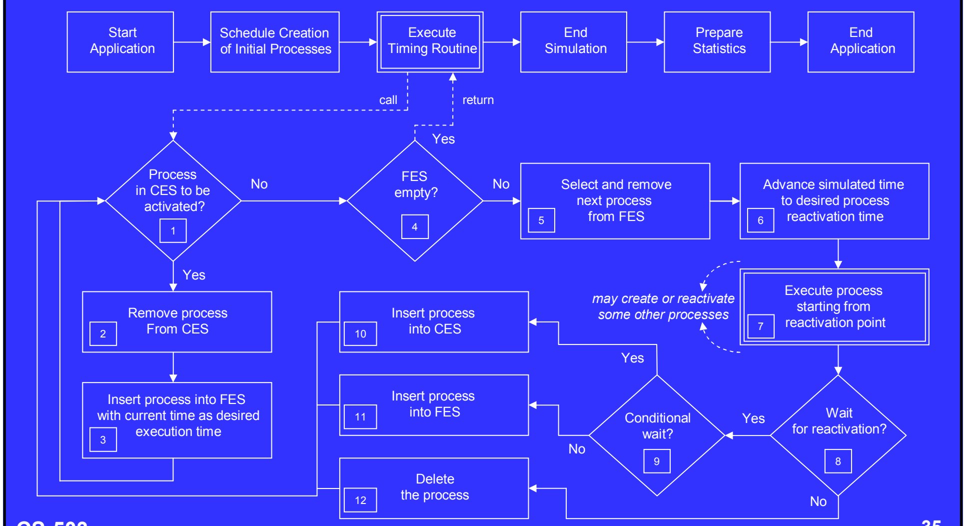
Process-Interaction Approach (Conceptual Simplicity vs Programming Control)

- Many well-known discrete-event simulation languages such as GPSS, SIMAN, SIMSCRIPT and SLAM are based on that approach.
- Appears conceptually simpler to a model builder.
- This simplicity usually comes at the expense of less programming control and flexibility than event-scheduling approach offers.

Process-Interaction Approach (Terminology)

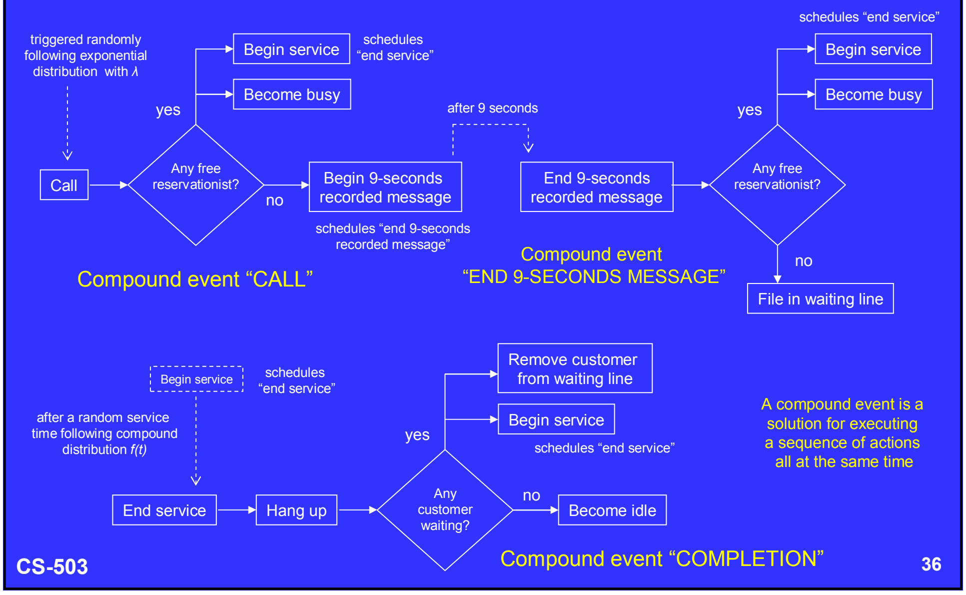
- Future event set (FES):
  - A linked list containing event notices associated with deactivated processes ordered by desired reactivation time.
- Conditional-Event Set (CES):
  - A linked list containing processes deactivated, reactivation of which depends on some conditions.
  - These processes are conditionally reactivated by other processes.

Process-Interaction Approach (Timing Routine)



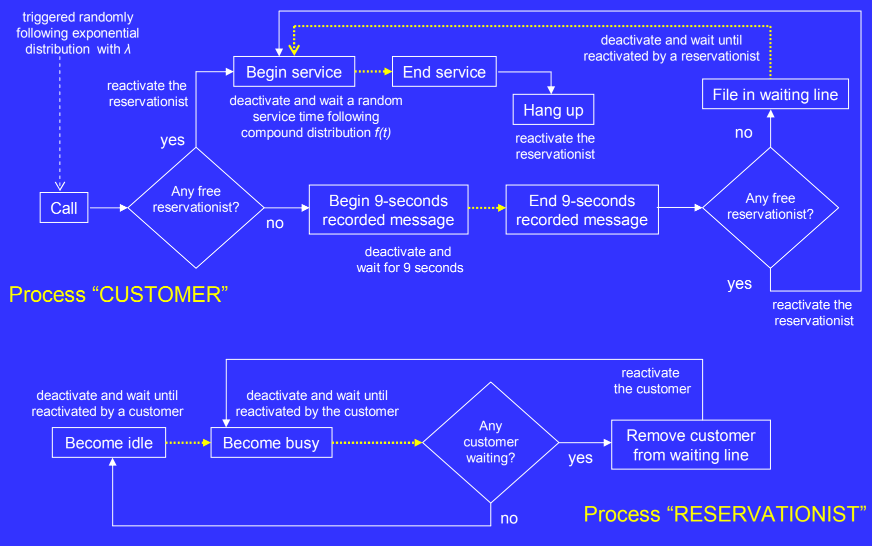
CS-503 35

Event-Scheduling Approach (Previous: Airline Reservation Sample)



CS-503 36

Process-Interaction Approach (New: Airline Reservation Sample)



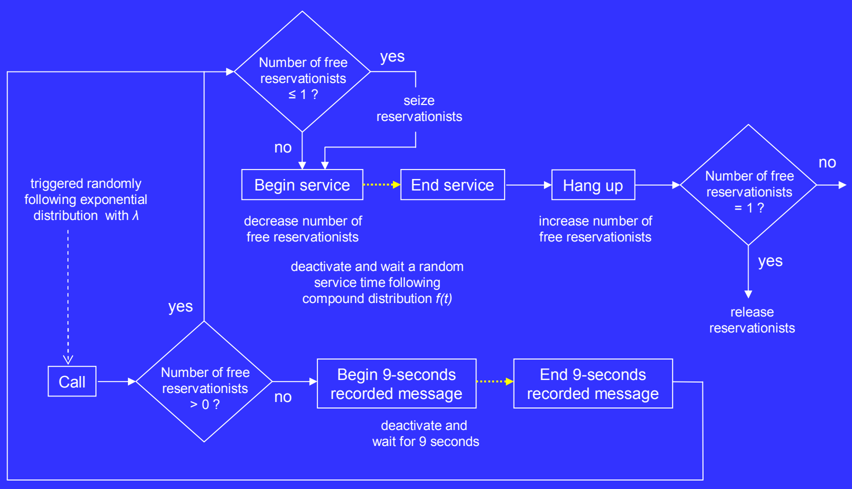
Processes and Resources

- In the previous sample,
  - Customers provide the stimulus to action,
  - Reservationists respond to the demands on a queue.
- So we see that,
  - Customer processes are active, but
  - Resevationist processes are passive.

Processes and Resources

- It is possible to model these passive processes as resources (first introduced by GPSS).
- Most simulation programming languages have:
  - Seize command (activates a resource) and
  - Release command (free a resource).
- By using passive resources,
  - We may not require to model a reservationist explicitly.

Processes and Resources (Airline Reservation Sample)



Process “CUSTOMER”