Computer system modeling and simulation

2- Discrete-event simulation

Sosina M.
Addis Ababa institute of technology (AAiT)
2012 E.C.

Overview

- Introduction
- Discrete event simulation
- Modeling approaches
  - Event scheduling
  - Process interaction
  - Activity scanning – three phase approach

Introduction

Dynamic systems modeling

- Continuous simulation
  - Continuous system – state variables change continuously over time
  - Continuous simulation
    - Simulation of continuous system
    - continuously tracks system responses over time according to a set of equations
  - E.g., $\frac{\partial T(t)}{\partial t} = -\sigma(t)$ (Newtonian cooling model)

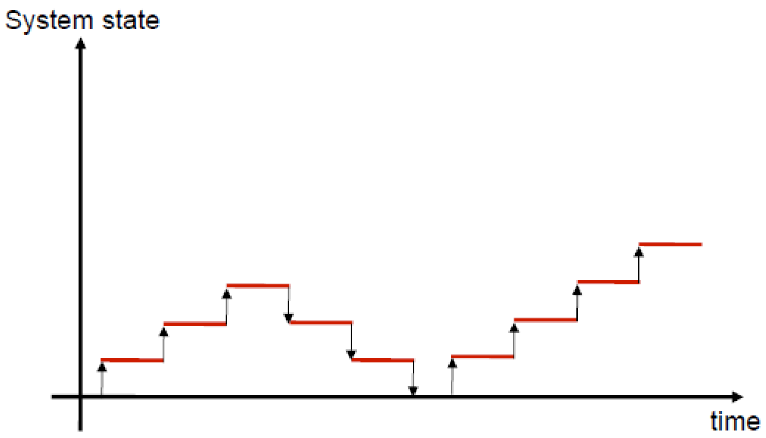
    $T(t + \Delta t) = T(t) - \sigma(t)\,\Delta t$ (approximation of differential equation)

- Discrete-event simulation

Discrete event simulation (DES)

- In discrete event simulation, a system is modeled in terms of its state at each point in time
  - System state changes at the occurrence of events
  - Time advances through discrete steps from one event to the next

- Appropriate for systems where change in system state occur only at discrete points in time



DES modeling components

- Used to describe the system dynamics
- *System state:* A collection of variables that contain all the information necessary to describe the system at any time
- *Entity:* An object in the system that requires explicit representation in the model
- *Attributes:* The properties of a given entity
- *Event:* An instantaneous occurrence that changes the state of a system
- *Event notice:* A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event
- *Event list:* A list of event notices for future events, ordered by time of occurrence; known as the future event list (FEL)

DES: modeling components

- *Activity:* A duration of time of specified length (e.g., a service time or interarrival time), which is known when it begins
  - The duration is characterized and defined by the modeler (e.g., deterministic, statistical)
- *Delay:* A duration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a last-in-first-out waiting line which, when it begins, depends on future arrivals)
  - Determined by the system condition (customer delay in a waiting line)
- *Clock*: A variable representing the simulated time

DES: model description

- Definition of model components
  - Static description
  - Dynamic relationships and interaction between the components
- Important questions
  - Event
    - ✓ How does each event affect system state, entity attributes, and set contents?
  - Activity
    - ✓ How are activities defined (i.e., deterministic, probabilistic, or some other mathematical equation)?
    - ✓ What event marks the beginning or end of each activity?
    - ✓ Can the activity begin regardless of system state, or is its beginning conditioned on the system being in a certain state? (For example, a machining “activity" cannot begin unless the machine is idle, not broken, and not in maintenance.)

DES: model description

- Delay
  - ✓ Which events trigger the beginning (and end) of each delay?
  - ✓ Under what condition does a delay begin or end?

- System state
  - ✓ What is the system state at time 0? What events should be generated at time 0 to get the simulation started?

DES: model description

- A DES proceeds by producing a sequence of system snapshots
  - Represent the evolution of the system through time

Clock	*System State*	*Entities and Attributes*	*Set 1*	*Set 2*	...	*Future Event List, FEL*	*Cumulative Statistics and Counters*
t	$(x, y, z, \ldots)$					$(3, t_1)$– Type 3 event to occur at time t_1	
						$(1, t_2)$– Type 1 event to occur at time t_2	
	⋮					⋮ ⋮	

  - Not all models contain every element

DES: Modeling approaches

- **Event scheduling**
  - Events and there effect on the system state
- **Process interaction**
  - Entities/objects and their life cycle as they flow through the system, demanding resources and queueing to wait for resource
- **Activity scanning**
  - Use a fixed time increment and a rule based approach to decide whether any activities can begin at each point in simulation time

Event scheduling

- Future event list (FES)
  - Used to advance the simulation time
  - Guarantees that all events occur in chronological order
- At any given time t, the FEL contains all previously scheduled future events + their associated time
- The FEL is ordered by event time

$$t < t_1 < t_2 < t_3 < \cdots < t_n$$

  - t is the value of clock (simulation time)
  - Event associated with time t1 – the next event that will occur (imminent event)

Event scheduling

- Steps
  - Update the system snapshot at simulation time t
  - Advance the simulation time to t1
  - Create a new system snapshot for time t1
  - At time t1 new future events may or may not be generated
    - ✓ If any, the are scheduled by creating event notices and putting then in their proper position on the FEL
- This process repeats until the simulation is over
- The sequence of actions that a simulator must perform *to advance the clock* and *build a new system snapshot* is called the **event-scheduling/time-advance algorithm**

Event scheduling algorithm

- The system snapshot at time 0 is defined by the initial conditions and the generation of exogenous events
  - The specified initial conditions define the system state at time t0

Clock	State	…	Future event list
t	(5,1,6)		$(3,t_1)$ – Type 3 event to occur at t_1
			$(1,t_2)$ – Type 1 event to occur at t_2
			$(1,t_3)$ – Type 1 event to occur at t_3
			…
			$(2,t_n)$ – Type 2 event to occur at t_n

Algorithm

1. Remove the event notice for the imminent event from FEL (event (3, t1) in the example)
2. Advance clock to imminent event time (set clock=t1)
3. Execute imminent event: update system state, change entity attributes and set membership as needed
4. Generate future events (if necessary) and place their event notice on FEL, ranked by event time
5. Update cumulative statistics and counters

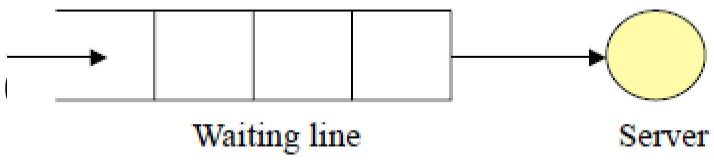
Clock	State	…	Future event list
t_1	(5,1,5)		$(1,t_2)$ – Type 1 event to occur at t_2
			$(4,t^*)$ – Type 4 event to occur at t^*
			$(1,t_3)$ – Type 1 event to occur at t_3
			…
			$(2,t_n)$ – Type 2 event to occur at t_n

Event scheduling

- → Initialize
- → Schedule the first arrival [TnextArr=CurrentTime + generated InterArrTime]
- → Insert the scheduled arrival in the FES
- → While the CurrentTime <= SimuTime
  - → Remove the first event from FES
  - → Current time=the event time
  - – If the event==arrival event
    - – Destroy the record (from FES)
    - – Number of customers in the queue++
    - – Schedule the next arrival [TnextArr=CurrentTime + generated InterArrTime]
    - – Insert the scheduled arrival in the FES and Sort FES
    - – If the server state==idle
      - – Server state=busy
      - – Schedule departure event [TnextDepart = CurrentTime + generated InterDepartTime]
      - – Insert the departure event in the FES and Sort FES
      - – Number of customers in the queue--
  - – Else If the event==departure event
    - – Destroy the record (from FES)
    - – if *Number of customers in the queue≠ 0*
      - – Server state=busy
      - – Schedule departure event [TnextDepart = CurrentTime + generated InterDepartTime]
      - – Insert the departure event in the FES and Sort FES
      - – Number of customers in the queue--
    - – Else
      - – Server state=idle

Example

- A queueing system
  - Entities
    - ✓ Client and servers
  - System state
    - ✓ Number of customers in the waiting lime at time t (LQ(t))
    - ✓ Number of customers being served at t (LS(t))
  - Events
    - ✓ Arrival (A), departure (end of service) (D), Stopping event (
  - Event notices
    - ✓ (A, t), (D, t), (E, TE)
  - Activities
    - ✓ Inter-arrival time, service time
  - Delay
    - ✓ Customer time spent in waiting time



Example

- *Initial state = [LQ(0), LS(0)]*
- ***Execution of arrival event** (at time t)*
  - Remove the event notice from the FEL
  - Clock = t
  - Calculate the interarrival time ti for the next arrival
    - *Schedule an arrival at time t+ti*
    - *create an event notice (a new arrival event)*
    - *Insert the event notice in FEL*
  - If the server is idle
    - Make the server busy
    - Determine the service time Ts
      - *Schedule the end of service at time t+Ts*
      - *create an event notice (a new departure event)*
      - *Insert the event notice in FEL*
    - Set LS(t) =1
  - Otherwise (the server is busy)
    - Increase LQ(t) by 1

Example

- ***Execution of departure event*** *(at time t)*
  - Remove the event notice from the FEL
  - Clock = t
  - *if someone is the waiting line?*
    - ✓ Determine the service time Ts
      - *Schedule the end of service at time t+Ts*
      - *create an event notice (a new departure event)*
      - *Insert the event notice in FEL*
    - ✓ reduce LQ(t) by 1
  - Otherwise
    - ✓ Set LS(t)=0

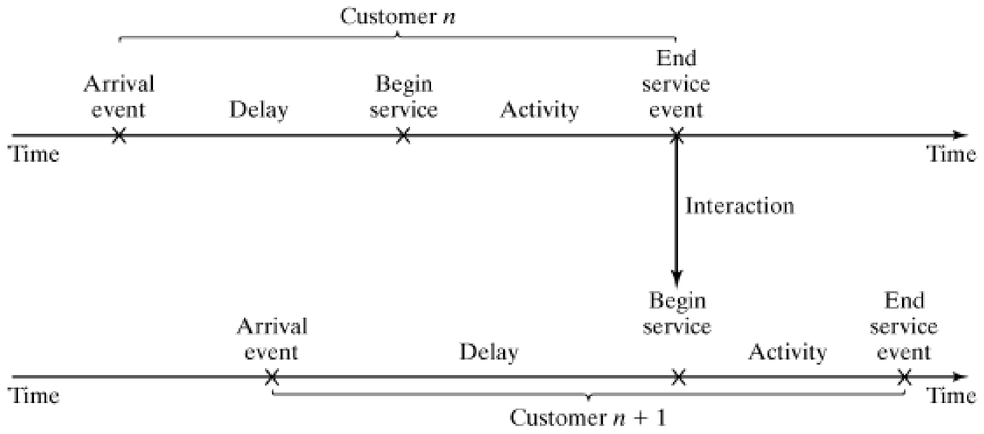
Stopping a simulation

- Stopping time defines how long the simulation will run

- There are generally two ways to stop a simulation
  - At time 0, schedule a stop simulation event at a specified future TE
    - ✓ Before simulating, it is known that the simulation will run over the time interval [0 TE]
  - The run length is determined by the simulation itself
    - ✓ TE is the time of occurrence of some specified event E(e.g.,the time of the 100th service completion)

Process interaction

- The simulation model is defined in terms of entities and their life cycle
- Process – the life cycle of one entity
  - This life cycle consists of various events and activities
- Or process is an ordered series of events related to a certain entity
- Some activities might require the use of one or more resources whose capacities are limited
  - This and other constraints cause the processes to interact
- Example – customer process



Process interaction

- Event scheduling and process interaction use a variable time advance
  - All events and system state changes occurs at one instant of simulated time
  - Then, the simulation time is advance to the time of the next imminent event on the FEL
- Process interaction modeling example - a queueing system
  - Entities – clients and servers
  - Processes
    - ✓ One process for the operations done by each client
    - ✓ One process for the operation done by the server

Process interaction

Client

- Calculate the inte-rarrival time Ti for the next client
- Schedule a client arrival at time t+Ti
- Create a record for the client and record it in the queue (list)
- Wait until his own service ends
- Destroy the record

Server

- If the queue is empty
  - ✓ Wait until a client arrives
- Otherwise
  - ✓ Select a client to be served
  - ✓ Extract the client record from the queue
  - ✓ Determine the service time Ts
  - ✓ Holds for Ts , until the end of the service (t+Ts)

Process interaction

→ Initialize

→ Schedule the first arrival [TnextArr=CurrentTime + generated InterArrTime]

%Customer process generator

- While(true)
  - Hold for TnextArr time
  - Create customer process
  - End

%Customer process

- While (true):
  - Insert the arrival event in the queue
  - Number of customers in the queue++
  - Schedule the next arrival [TnextArr=CurrentTime + generated InterArrTime]
  - If ServerState==idle
    - Activate the server process
  - ***Hold for TnextArr time***

%Server process

- While (true):
  - If Number of customers in the queue==0
    - Server state=idle
    - Passivate (wait until a customer arrives)
  - Else
    - Server state=busy
    - Schedule departure event [TnextDepart = CurrentTime + generated InterDepartTime]
    - Delete customer record from the queue
    - Number of customers in the queue--
    - ***Hold for TnextDepart time***

%Stop condition

- Wait for SimuTime
- End all processes

Activity scanning

- Use a *fixed time increment* and *a rule based approach* to decide whether any activities can begin at each point in simulated time
- Activities and conditions that allow an activity to begin
- At each clock advance
  - The conditions for each activity are checked
  - If the conditions are true
    - ✓ The corresponding activity begins
- Repeated scanning- slow runtime
- A modified version– three-phase approach

Activity scanning

- Example
- Initial conditions
  - Queue length, server state (idle, busy), arrival time (t_arrival), departure time (t_depart)

- If t>=t_arrival
  - ✓ t_arrival= t + Ti
  - ✓ If the server is idle
    - ➢ Server state=busy
    - ➢ Determine the service time Ts
    - ➢ t_depart=t+ Ts
  - ✓ If the server is busy
    - ➢ Increase queue length by 1

- If t>=t_depart
  - ✓ If queue length =0
    - ➢ Server state=idle
  - ✓ Otherwise
    - ➢ Reduce queue length by 1
    - ➢ Determine the service time Ts
    - ➢ t_depart=t+ Ts
- If t>=t_end
  - ✓ End simulation

Activity scanning

- → Initialize
- → Schedule the first arrival [TnextArr=CurrentTime + generated InterArrTime]
- → While the CurrentTime <= SimTime
  - %Check arrival events
  - → While TnextArr <= CurrentTime
    - → Inset the event in the queue
    - → Number of customers in the queue++
    - → Schedule the next arrival a
    - – If the server state== idle
      - – Server state=busy
      - – Schedule departure time [TnextDepart = ArrivalTimeOfThisArrival + generated InterDepartTime]
      - – Delete the event from the queue
      - – Number of customers in the queue--

  - %Check departure events
  - → While TnextDepart <= CurrentTime
    - – if *Number of customers in the queue≠ 0*
      - – Server state=busy
      - – Schedule departure event [TnextDepart = DepartTimeOfTheLastCustomer + generated InterDepartTime]
      - – Number of customers in the queue--
    - – Else
      - – Server state=idle
      - – Break

  - %update the time
  - → CurrentTime=CurrentTime+Δt

Three phase approach

- Combines the features of event scheduling with activity scanning
  - To allow for variable time advance and avoidance of scanning when it is not necessary
- Classification of activities
  - B activities – activities bound to occur (all primary events and unconditional activities)
    - ✓ Can be scheduled ahead of time
    - ✓ FEL contains only B type events
    - ✓ E.g., service completion is an example of a primary event
  - C activities – activities or events that are conditional upon certain condition being true
    - ✓ Scanning to learn whether any C type activities can begin
    - ✓ E.g., beginning service is a conditional event – its occurrence is triggered only on the condition that a customer is present and a server is free

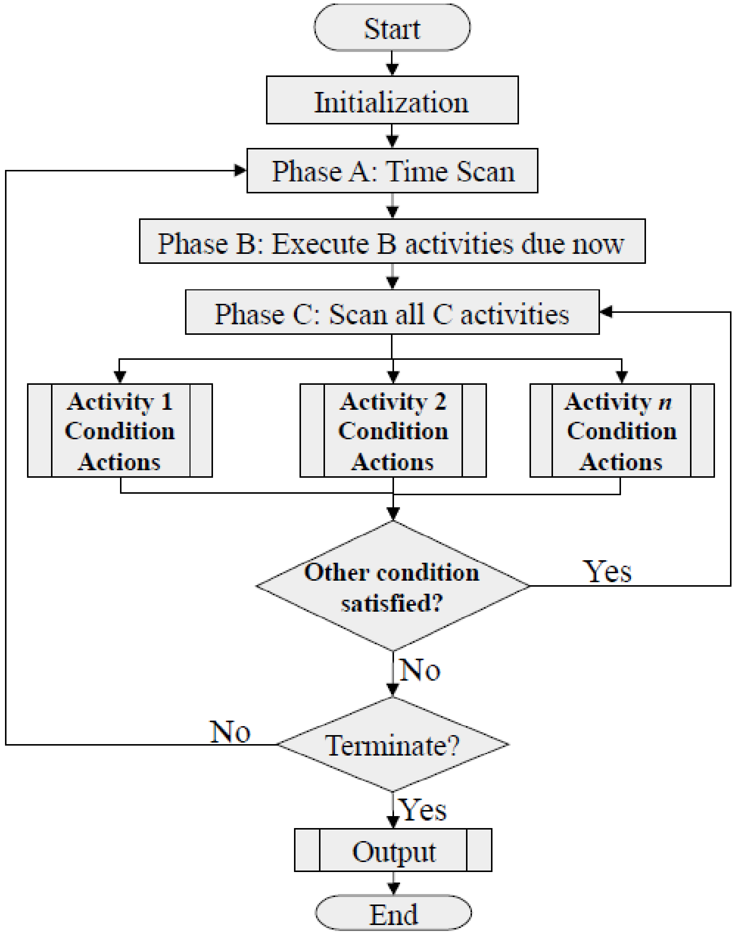
Three phase approach

- The simulation proceeds with repeated execution the 3 phases
- Phase A
  - Remove the imminent event from the FEL and advance the clock to its event time.
  - Remove from the FEL any other events that have the same event time
- Phase B
  - Execute all B-type events that were removed from the FEL (this could free a number of resources or otherwise change system state.)
- Phase C
  - Scan the conditions that trigger each C-type activity and activate any whose conditions are met.
  - Rescan until no additional C-type activities can begin and no event occur

Three phase approach

- Improves the execution efficiency of the activity scanning method

- Procedures
  - Initialization
  - Update the simulation time to the next event occurrence time
  - Execute the B activities scheduled for this simulation time
  - Processing the c activities whose condition hold



Three phase

- → Initialize
- → Schedule the first arrival [TnextArr=CurrentTime + generated InterArrTime]
- → Insert the scheduled arrival in the FES
- → While the CurrentTime <= SimuTime
  - → Remove the first event from FES
  - → Current time=the event time

  %Type B events
    - If the event==arrival event
      - Destroy the record (from FES)
      - Number of customers in the queue++
      - Schedule the next arrival [TnextArr=CurrentTime + generated InterArrTime]
      - Insert the scheduled arrival in the FES and Sort FES
    - Else If the event==departure event
      - Destroy the record (from FES)
      - Server state=idle

%Conditional activities

- if *Number of customers in the queue* $\neq 0$ & Server state==idle
  - Server state=busy
  - Schedule departure event [TnextDepart = CurrentTime + generated InterDepartTime]
  - Insert the departure event in the FES and Sort FES
  - Number of customers in the queue--