

DA 507

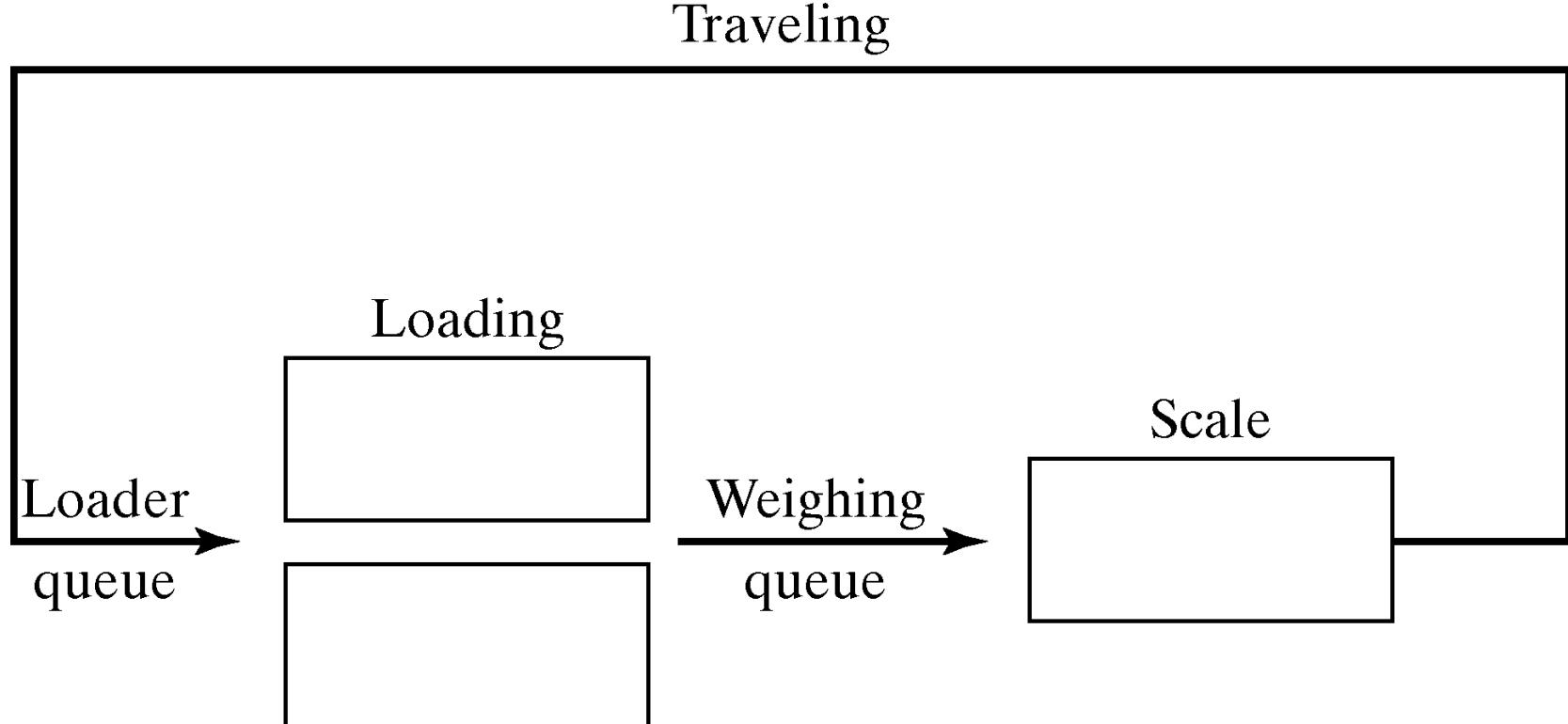
modeling and optimization

Week 5 – Lecture 1

Fundemantals of Simulation Models

Güvenç Şahin

Dump Truck System Example



What is Simulation?

- According to Robert E. Shannon (1975):

Simulation is “**the process of designing a model of a real system and conducting experiments with this model**

- for the purpose either of understanding the behavior of the system or
- of evaluating various strategies for the operation of the system”.

What is Simulation? (cont'd.)

Simulation: methods and applications to imitate or mimic systems that are in existence or not in existence, usually via computer.

The existing or non-existing systems can be investigated by modeling them.

Helps us to understand and analyze **complex systems**.

Simulation models are *descriptive*, not *prescriptive*. They tell how a system works under given conditions; not how to arrange the conditions to make the system work best.

What is the purpose?

- Simulation enables the study and experimentation with the internal interactions of a complex system, or of a subsystem within a complex system.
- Simulation can be used to experiment with new designs or policies prior to implementation, so as to prepare for what may happen.
- It's often impossible to work with the real system, since it would be disruptive, expensive, or dangerous. We prefer to simulate the system to evaluate alternative designs or policies.

What is the purpose?

- By changing simulation inputs and observing the resulting outputs, valuable insights may be obtained
 - which variables are most important and how the variables interact.
- The knowledge gained in designing a simulation model may be of great value toward suggesting improvements in the system under investigation.
- Experiment to *estimate* something hard to compute exactly.
- Simulation can be used to verify analytic solutions.

Systems

- *System* – facility or process, actual or planned
- A system is defined as a group of objects that are joined together in some interaction or interdependence toward the accomplishment of some purpose.
 - Examples
 - Manufacturing facility (with machines, people, transport devices, conveyor belts, and storage space)
 - Bank operation (customers, servers, and facilities like teller windows, automated teller machines, loan desks, and safety deposit boxes)
 - Airport operations (passengers, security, planes, crews, baggage)
 - Hospital facilities (emergency room, personnel, operating room, equipment)
 - Computer network (servers, clients, disk drives, printers, networking capabilities, and operators)
 - Freeway system (road segments, interchanges, controls, and traffic)
 - Business process (insurance office)
 - Emergency-response system

Models

- **Model** – set of assumptions/approximations about how the system works
 - Study the model instead of the real system ... usually much easier, faster, cheaper, safer
 - Can try wide-ranging ideas with the model
 - Make your mistakes on the computer where they *don't* count, rather than for real where they *do* count
 - Model *validity* (for any kind of model, not just for simulation)
 - Care in building to mimic reality faithfully
 - Level of detail
 - Is the model good enough to represent the real system?
 - Can get almost same conclusions from the model as you would from the real system?

Types of Models

- *Physical (iconic)* models

Physical replica or scale model of the system

- Tabletop material-handling models (miniature versions of the facility)
- Mock-ups of fast-food restaurants
- Flight simulators

- *Logical (mathematical)* models

- Approximations and assumptions about a system's operation
- Often represented via computer program in appropriate software
- Exercise the program to try things, get results, learn about the model behavior

Appropriate Logical Model?

- If model is simple enough, use traditional mathematical analysis and try to obtain exact results
 - Queueing theory
 - Requires restrictive assumptions about the model
 - Popular, simple model: *M/M/1 queue*
 - Interarrival times ~ exponential
 - Service times ~ exponential, independent of interarrivals
 - Must have $E(\text{service time}) < E(\text{interarrival time})$
 - Can obtain exact analytic results for the steady-state (long-run) ; e.g., average waiting time in queue is
$$\frac{\mu_S^2}{\mu_A - \mu_S}, \quad \mu_A = E(\text{interarrival time})$$
$$\mu_S = E(\text{service time})$$
 - Problems: validity, estimating means, time frame
 - Linear programming (solving optimization models)

Appropriate Logical Model? (cont'd.)

- But complex systems can seldom be *validly* represented by a simple analytic model
 - Danger of over-simplifying assumptions ...working on the wrong problem
- Often, a complex system requires a complex model, and analytical methods don't apply ... what to do?
 - Employ simulation

Advantages of Simulation

- Flexibility to model things as they are (even if the system is messy and complicated)
- Allows to consider uncertainty and nonstationarity in modeling
 - Allows us not to ignore system variability
 - Improves model validity
- Advances in simulation software

Disadvantages of Simulation

- Don't get exact answers, only approximations, estimates
 - Also true for many other methods
 - However, an over-simplified model with non-random results will probably not be a valid representation of the system
- Get random output (*RIRO, random-in random-out*) from stochastic simulations
 - It can be hard to distinguish whether an observation is a result of system interrelationships or of randomness
 - It is crucial to utilize statistical methods and analyze the simulation experiments

Different Kinds of Simulation

- Static vs. *Dynamic*
 - Does time have a role in the model?
 - Dynamic simulation models represents systems as they change over time.
- Continuous vs. *Discrete*
 - Can the “state” change continuously or only at discrete points in time?
- Deterministic vs. *Stochastic*
 - Is everything for sure or is there uncertainty?
- Most operational models:
 - *Dynamic, Discrete, Stochastic*

Classification of Simulation Models

1. *Static* versus *dynamic* models.

- In a **static model**, time plays no role.

Most of these models are called *Monte Carlo* models.

Example: Determining the order quantity of a product for a single period (with random demand).

For a given order quantity we can generate the realizations of the random demand and then estimate the expected net profit (or distribution of the profit).

- A **dynamic simulation** models a sequence of events that occur over time.

Example: The analysis of a bank queue as it evolves over time.

Classification of Simulation Models (cont'd.)

2. *Discrete* versus *continuous* models.

- **Discrete-event simulation:** The state of the system changes **only at a discrete set of points in time** as a result of the occurrence of discrete events.

For example, in a queueing system where the state of the system is the number of customers in the system.

- **Continuous simulation:** The changes in **state of the system occur continuously over time.**

For example, if we are interested in an airplane in flight, then state is defined as the current position of the airplane which changes continuously over time.

Continuous simulation typically require differential equations.

Classification of Simulation Models (cont'd.)

3. Whether or not the model has a stochastic (or random) aspect:

- *Deterministic simulations* involve no random input.
Rerunning a simulation will not change the outcome.
- *Stochastic simulations* include randomness (at least some inputs are random).
 - Multiple runs of the same model may generate different values.
 - Randomness forces us to generate many outcomes to see the range of possibilities and obtain better estimates.
 - The question of whether to generate 10 or 1000 or 100,000 outcomes is primarily a statistical question (as we will discuss later).

Steps of a Simulation Study

- Understand the system
- Problem formulation
- Setting objectives and overall project plan
- Model conceptualisation (Formulate the model representation)
- Data collection
- Model translation
- Verification
- Validation
- Experimental design
- Production runs and output analysis
- Documentation and reporting

Using Computers to Simulate

- General-purpose languages (FORTRAN, C, C++, Java, Matlab, others)
 - Tedious, low-level, error-prone, but almost complete flexibility
- Simulation languages
 - GPSS, SLX, *SIMAN* (*on which Arena is based, and is included in Arena*)
 - Popular and effective, are still in use
- High-level simulators
 - Very easy, graphical interface
 - Domain-restricted (manufacturing, communications)
 - Limited flexibility — model validity?

Simulating the System

- Individual operations (arrivals, service by the drill press, etc.) will occur as they would as in reality
- Movements, changes occur at the right “time” in the right order
- Different pieces interact and have right effects on each other

- Bookkeeping to get output performance measures
- Simulation software keeps track of things for you

A simple single-serve queue: Grocery Exm

- A small grocery store has one checkout counter
- Customers arrive at this checkout counter at random time fro 1 to 8 minutes apart
- Each possible value of interarrival time has the same probability of occurrence
- Service time varies from 1 to 6 minutes, with probabilities shown in table 6
- Problem is to analyze the system by simulating the arrival and service of 100 customers
- A set of uniformly distributed random numbers is needed to generate the arrivals at the checkout counter
- These random numbers have the following properties:
 - It is uniformly distributed between 0 and 1
 - Successive random numbers are independent

A simple single-serve queue: Grocery Exm

- Arrival Event characterized by the distribution of time between arrivals

<i>Time between Arrivals (minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>
1	0.125	0.125
2	0.125	0.250
3	0.125	0.375
4	0.125	0.500
5	0.125	0.625
6	0.125	0.750
7	0.125	0.875
8	0.125	1.000

A simple single-serve queue: Grocery Exm

Time between Arrivals

<i>Customer Number</i>	<i>Time between Arrivals (Minutes)</i>	<i>Customer Number</i>	<i>Time between Arrivals (Minutes)</i>
1	-	11	4
2	1	12	4
3	1	13	7
4	6	14	6
5	3	15	3
6	7	16	8
7	5	17	8
8	2	18	2
9	4	:	:
10	1	100	5

A simple single-serve queue: Grocery Exm

- Departure event controlled by the distribution of service time

<i>Service Time (minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>
1	0.10	0.10
2	0.20	0.30
3	0.30	0.60
4	0.25	0.85
5	0.10	0.95
6	0.05	1.00

A simple single-serve queue: Grocery Exm

<i>Customer</i>	<i>Service Time (Minutes)</i>	<i>Customer</i>	<i>Service Time (Minutes)</i>
1	4	11	5
2	2	12	3
3	5	13	4
4	4	14	5
5	1	15	3
6	5	16	2
7	4	17	4
8	1	18	3
9	4	:	:
10	3	100	2

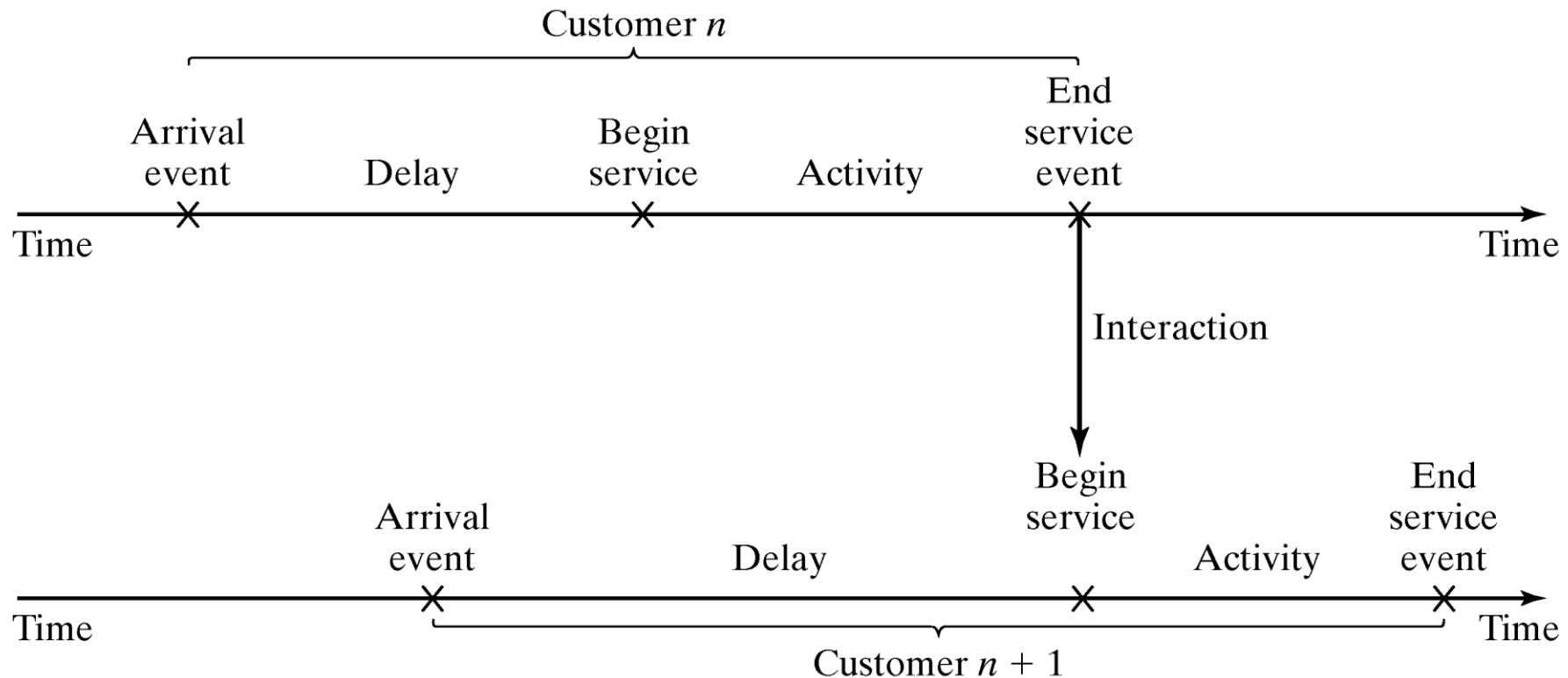
A simple single-serve queue: Grocery Exm

- Prepare the table basen on the events, system state, and statistics
- Initialize the table by filling cells for the 1st customer:
arrives at time 0; service begins immediately and finishes at time 4;
customer stays in the system for 4 minutes
- Subsequent rows in table are based on the random numbers for
interarrival time, service time, and the completion time of the previous
customer
example: 2nd customer arrives at time 1; service could not begin until
time 4; customer waits in queue for 3 minutes; as result, stays in the
system for 5 minutes
- Totals are calculated to compute summary statistics

A simple single-serve queue: Grocery Exm

<i>Customer</i>	<i>Interarrival time (minutes)</i>	<i>Arrival Time</i>	<i>Service Time (minutes)</i>	<i>Time Service begins</i>	<i>Waiting time in Queue</i>	<i>Time Service ends</i>	<i>Time Customer in system</i>	<i>Idle Time of Server</i>
1	-	0	4	0	0	4	4	-
2								0
3								0
4								0
5								0
6								2
7								0
8								0
9								1
10								0
11								0
12								0
13								1
14								2
15								0
16								3
17								6
18								0
19								2
20								3
:								:
100								0
Total	415	415	317	416	174	418	491	101

A simple single-serve queue: Grocery Exm



A simple single-serve queue: Grocery Exm

<i>Customer</i>	<i>Interarrival time (minutes)</i>	<i>Arrival Time</i>	<i>Service Time (minutes)</i>	<i>Time Service begins</i>	<i>Waiting time in Queue</i>	<i>Time Service ends</i>	<i>Time Customer in system</i>	<i>Idle Time of Server</i>
1	-	0	4	0	0	4	4	-
2	1	1	2	4	3	6	5	0
3	1	2	5	6	4	11	9	0
4	6	8	4	11	3	15	7	0
5	3	11	1	15	4	16	5	0
6	7	18	5	18	0	23	5	2
7	5	23	4	23	0	27	4	0
8	2	25	1	27	2	28	3	0
9	4	29	4	29	0	33	4	1
10	1	30	3	33	3	36	6	0
11	4	34	5	36	2	41	7	0
12	4	38	3	41	3	44	6	0
13	7	45	4	45	0	49	4	1
14	6	51	5	51	0	56	5	2
15	3	54	3	56	2	59	5	0
16	8	62	2	62	0	64	2	3
17	8	70	4	70	0	74	4	6
18	2	72	3	74	2	77	5	0
19	7	79	1	79	0	80	1	2
20	4	83	2	83	0	85	2	3
:	:	:	:	:	:	:	:	:
100	5	415	2	416	1	418	3	0
Total	415		317		174		491	101

A simple single-serve queue: Grocery Exm

- Results/findings from the simulation table

[All time figures are in minutes]

$$\text{Average waiting time} = \frac{\text{total time customers wait in queue}}{\text{total number of customers}} = \frac{174}{100} = 1.74$$

Probability that customer has to wait in queue

$$= \frac{\text{number of customers who wait}}{\text{total number of customers}} = \frac{46}{100} = 0.46$$

$$\text{Proportion of idle time of server} = \frac{\text{total idle time of server}}{\text{total run time of simulation}} = \frac{101}{418} = 0.24$$

$$\text{Average service time} = \frac{\text{total service time}}{\text{total no. of customers}} = \frac{317}{100} = 3.17$$

[compare to expected service time given by equation

$$\begin{aligned} E(s) &= \sum s p(s), \quad s = 0 \text{ to } \infty \\ &= 1(0.1) + 2(0.2) + 3(0.3) + 4(0.25) + 5(0.1) + 6(0.05) = 3.2 \end{aligned}$$

A simple single-serve queue: Grocery Exm

- Results/findings from the simulation table

$$\text{Average time between arrivals} = \frac{\text{sum of all times between arrivals}}{\text{number of arrivals} - 1}$$
$$= 415 / 99 = 4.19$$

[compare the above time with expected time between arrivals by finding mean of discrete uniform distribution whose endpoints are $a = 1$ & $b = 8$

$$\text{Mean } E(A) = (a + b) / 2 = (1 + 8)/2 = 4.5$$

$$\text{Average waiting time of } = \frac{\text{total time customers wait in queue}}{\text{those who wait}}$$
$$= 174 / 54 = 3.22$$

$$\text{Average time customer spends in system} = \frac{\text{total time customers spend in system}}{\text{total number of customers}}$$
$$= 491 / 100 = 4.91$$

[another way to find the same is to add average time customers spends waiting in queue and average time customers spends in service

$$= 1.74 + 3.17 = 4.91$$

A simple single-serve queue: Grocery Exm

- Conclusions from the simulation table
 - A longer simulation would increase the accuracy of the findings
 - About half of the customers had to wait; however, average waiting time not excessive
 - Server does not have undue amount of idle time

Discrete-Event Simulation: Components

- Entities
- Attributes
- Activities and Events
- Resources
- Global Variables
- A Random Number Generator
- A Calendar
- System State Variables
- Statistics Collectors

Discrete-Event Simulation: Components

- *Entities*
 - An object of interest in the system
 - “Players” that move around, change status, affect and are affected by other entities
 - *Dynamic objects* — get created (randomly or according to a schedule), move around, leave (maybe)
 - Usually represent “real” things
 - Our model: entities are the parts
 - Can have “fake” entities for modeling “tricks”
 - Breakdown demon, break angel

Discrete-Event Simulation: Components

- *Entities*
 - Can have different types of entities concurrently
Usually, identifying the types of entities is the first thing to do in building a model
 - Without entities, nothing would happen in a simulation
One stopping condition for a simulation model is the condition where there are no active entities in the system

Discrete-Event Simulation: Components

- *Attributes*
 - Characteristics of all entities: describe, differentiate
 - Attributes are characteristics of a given entity that are specific to that entity. For example:
 - Time of arrival
 - Due date
 - Routing of a job in a job shop
 - Priority of a waiting customer
 - Attribute value tied to a specific entity
 - Like “local” (to entities) variables
 - Some automatic in Arena, some you define

Discrete-Event Simulation: Components

- *Events*
 - Conditions that occur at a point in time which cause a change in the state of the system
 - Instantaneous occurrence that changes the state of a system
 - Examples: Arrival or departure of a customer
- *Activities*
 - An activity represents a time period of **specified length**
 - Example: a service time or interarrival time
 - In general, the duration of an activity is either constant or is randomly generated (but known exactly at the beginning of the activity!)
 - The completion of an activity is an event, called a **primary event**
 - At the simulated instant that an activity duration begins, an event notice is created having an event time equal to the activity's completion time
 - The event notice is placed on the calendar (**future event list**)

An entity *interacts* with activities. Entities *interacting with* activities *create events*.

Discrete-Event Simulation: Components

- *The Event Calendar (Future event list)*
 - List of events that are scheduled to occur in the future
 - In every simulation, there is only one calendar of future events and it is ordered by the earliest scheduled time first
 - At any given point in time, every event that has already been scheduled to occur in the future is held on the calendar
 - Only primary events appear on future event list
 - List of event *records*:
 - [Entity No., Event Time, Event Type]
 - Keep *ranked* in increasing order on Event Time
 - Next event always in top record
 - Initially, schedule first Arrival, The End (for our example)

Discrete-Event Simulation: Components

- *Delay*
 - A duration of time of unspecified length, which is not known until it ends
 - In contrast to an activity, a delay's duration is not specified by the modeler ahead of time, but rather determined by system conditions
 - Also called conditional wait.
 - The completion of a delay is sometimes called **secondary event** (no event notice!). Example: a part leaves the queue (when the machine becomes available due to the primary event of “other part’s departure”).

Examples:

- A customer’s delay in a first-in-first-out waiting line, which depends on the number and duration of service of other customers ahead in the line as well as the availability of servers.
- A customer’s delay in a last-in-first-out waiting line, which depends on future arrivals

Discrete-Event Simulation: Components

- *Queues*

- Place for entities to wait when they can't move on for an unspecified period of time (maybe since the resource they want to **seize** is not available)
- Queues are most commonly used for waiting in line for a resource or storing material that will be taken out of the queue when the right conditions exist
- Have names, often tied to a corresponding resource
- Can have a finite capacity to model limited space — have to model what to do if an entity shows up to a queue that's already full
- Usually watch the length of a queue, waiting time in it

Discrete-Event Simulation: Components

- *Resources*
 - What entities compete for
 - People (workers)
 - Equipment (machines, nodes in a communication network)
 - Space (traffic intersections)
 - Entity *seizes* a resource, uses it, *releases* it
 - Think of a *resource being assigned to an entity*, rather than an entity “belonging to” a resource
 - “A” resource can have several *units* of capacity
 - Seats at a table in a restaurant
 - Identical ticketing agents at an airline counter
 - Number of available units of resource can be changed during the simulation

Discrete-Event Simulation: Components

- **Global Variables**
 - Reflects a characteristic of the whole model, not of specific entities
 - Available to the entire model at all times
 - Examples:
 - Travel time between all station pairs
 - The length of the line allowed at a banch brank
 - Number of parts in system
 - Simulation clock
 - There's only one copy for the whole model
 - Not tied to entities
 - May change during the simulation run

Discrete-Event Simulation: Components

- *Random Number Generators*
 - Every simulation model has a random number generator
 - The random number generator is a software routine that generates a random number between 0 and 1 that is used in sampling random distributions.
 - For example, let us assume that you have determined that a given process delay is uniformly distributed between 10 minutes and 20 minutes. Then every time an entity went through that process, the random number generator would generate a number between 0 and 1 and evaluate the uniform distribution formula that has a minimum of 10 and a maximum of 20.

Discrete-Event Simulation: Components

- *System State Variables*
 - Describe current status
 - There can be several system state variables
 - Examples:
 - Server status $B(t) = 1$ for busy, 0 for idle
 - Number of customers in queue $Q(t)$
 - The one system state variable that every simulation package has is the current time of the simulation.
 - The simulation current time (clock) variable is updated every time a record is taken from the calendar.

Discrete-Event Simulation: Components

- **Statistics accumulators (collectors)**
 - Part of the simulation that collects statistics on certain states (such as the state of a resources), or the value of global variables, or certain performance statistics based on attributes of the entity
 - Variables that “watch” what’s happening
 - Depend on output performance measures desired
 - “Passive” in model — don’t participate, just watch
 - At the end of simulation, they are used to compute final output performance measures

Discrete-Event Simulation: Components

- Types of statistics accumulators (collectors)
 - Counts are very straightforward, they count.
 - Time-persistent (time-weighted) statistical collectors give the time averaged values of different variables in the simulation.
 - Examples: The number of busy resources that we have in the model and, the number of entities in each of the queues that serve the resources.
 - Tally statistics are collected one observation at a time without regard to the amount of time between observations.
 - Examples: The amount of time that an entity stays in the system and the waiting of a customer at a particular queue.

Discrete-Event Simulation: Components

- Statistical accumulators for the simple processing system
 - Number of parts produced so far
 - Total of the waiting times spent in queue so far
 - No. of parts that have gone through the queue
 - Max time in queue we've seen so far
 - Total time spent in system
 - Max time spent in system we've seen so far
 - Area so far under queue-length curve $Q(t)$
 - Max of $Q(t)$ so far
 - Area so far under server-busy curve $B(t)$

2-server queue: Call Center Problem

- Consider a computer technical support center where personnel takes calls and provide service
- The time between calls range from 1 to 4 minutes.
- There are two technical support people – Abel and Baker
 - Able is more experienced and can provide faster service than Baker
 - their service times are random from 2 to 5 and 3 to 6
 - service rule rule: Able gets the call if both of them are idle
- To estimate the system measures of performance, a simulation of the first 100 callers is made
- *Problem: Find out how well the service rule is working*

2-server queue: Call Center Problem

interarrival times

<i>Time between Arrivals (minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>
1	0.25	0.25
2	0.40	0.65
3	0.20	0.85
4	0.15	1.00

2-server queue: Call Center Problem

Able's service time

<i>Service Time (minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>
2	0.30	0.30
3	0.28	0.58
4	0.25	0.83
5	0.17	1.00

Baker's service time

<i>Service Time (minutes)</i>	<i>Probability</i>	<i>Cumulative Probability</i>
3	0.35	0.35
4	0.25	0.60
5	0.20	0.80
6	0.20	1.00

2-server queue: Call Center Problem

The simulation proceeds in accordance with the following steps

Step 1: For caller k , generate an interarrival time A_k ; arrival time of Caller k as $T_k = T_{k-1} + A_k$

Step 2: If Able is idle, Caller k begins service with Able at the current time T_{now}

- Able's service completion time: $T_{fin,A} = T_{now} + T_{svc,A}$
- Caller k 's time in system: $T_{sys} = T_{fin,A} - T_k$
- Since Able is idle, Caller k 's delay $T_{wait} = 0$

– If Able is busy and Baker is idle

- service completion time, $T_{fin,B} = T_{now} + T_{svc,B}$
- Caller k 's time in system, T_{sys} , is given by $T_{sys} = T_{fin,B} - T_k$

– Since Baker is idle, Caller k 's delay $T_{wait} = 0$

2-server queue: Call Center Problem

Step 3: If Able and Baker are both busy, then calculate the time at which the first one becomes available: Caller k begins service at

$$T_{beg} = \min(T_{fin,A}, T_{fin,B})$$

- when service for Caller k begins, set $T_{now} = T_{beg}$
- compute $T_{fin,A}$ or $T_{fin,B}$ as in Step 2
- Caller k 's time in system,

$$T_{sys} = T_{fin,A} - T_k \text{ or } T_{sys} = T_{fin,B} - T_k$$

2-server queue: Call Center Problem

- Caller 1 arrives at clock time 0 to get simulation started; Able is idle, so Caller 1 begins service with Able at clock time 0.
- The service time, 2 minutes, is generated from Able's service time distribution. Thus, Caller 1 completes service at clock time 2 minutes and was not delayed.
- An interarrival time of 2 minutes is generated from the given distribution. So, the arrival of Caller 2 is at clock time 2 minutes.

2-server queue: Call Center Problem

A generic simulation framework

