Purpose

- Develops a common framework for the modeling of complex systems.
- Covers the basic blocks for all discrete-event simulation models.
- Introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages:
  - These concepts and methodologies are not tied to any particular simulation package.
Dump-Truck Example

• Six dump trucks are used to haul coal from the entrance of a small mine to the railroad:
  – Each truck is loaded by one of two loaders.
  – After loading, the truck immediately moves to the scale to be weighed.
  – The loaders and the scale have a FCFS waiting line (or queue) for trucks.
  – After being weighed, a truck begins a travel time (during which the truck unloads) and returns to the loader queue.
## Dump-Truck Example

- The distribution of loading time, weighing time and travel time are:

<table>
<thead>
<tr>
<th>Loading Time</th>
<th>Probability</th>
<th>Cumulative Probability</th>
<th>Random Digit Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.3</td>
<td>0.3</td>
<td>1 - 3</td>
</tr>
<tr>
<td>10</td>
<td>0.5</td>
<td>0.8</td>
<td>4 - 8</td>
</tr>
<tr>
<td>15</td>
<td>0.2</td>
<td>1</td>
<td>9 - 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weighing Time</th>
<th>Probability</th>
<th>Cumulative Probability</th>
<th>Random Digit Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.7</td>
<td>0.7</td>
<td>1 - 7</td>
</tr>
<tr>
<td>16</td>
<td>0.3</td>
<td>1</td>
<td>8 - 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Travel Time</th>
<th>Probability</th>
<th>Cumulative Probability</th>
<th>Random Digit Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0.4</td>
<td>0.4</td>
<td>1 - 4</td>
</tr>
<tr>
<td>60</td>
<td>0.3</td>
<td>0.7</td>
<td>5 - 7</td>
</tr>
<tr>
<td>80</td>
<td>0.2</td>
<td>0.9</td>
<td>8 - 9</td>
</tr>
<tr>
<td>100</td>
<td>0.1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Dump-Truck Example

- Purpose: to estimate the loader and scale utilizations (% of time busy).
- The model has the following components:
  - System state \([LQ(t), L(t), WQ(t), Q(t)]\), where:
    * \(LQ(t)\) is the number of trucks in loader queue.
    * \(L(t)\) is the number of trucks (0, 1, or 2) being loaded.
    * \(WQ(t)\) is the number of trucks in the weigh queue.
    * \(W(t)\) is the number of trucks being weighed.
  - Event notices:
    * \((ALQ, t, DTi)\), dump truck \(i\) arrives at loader queue \((ALQ)\) at time \(t\).
    * \((EL, t, DTi)\), dump truck \(i\) ends loading \((EL)\) at time \(t\).
    * \((EW, r, DTi)\), dump truck \(i\) ends weighing \((EQ)\) at time \(t\).
  - Entities: The six dump trucks \((DT1, \ldots, DT6)\)
Dump-Truck Example

- The model has the following components cont.:
  - Lists:
    * Loader queue, all trucks waiting to begin loading, ordered in a FCFS basis.
    * Weigh queue, all trucks waiting to be weighted, ordered on a FCFS basis.
  - Activities: Loading time, weighing time, and travel time.
  - Delay: Delay at loader queue, and delay at scale.
Dump-Truck Example

• When an end-loading ($EL$) event occurs, say for truck $j$ at time $t$, other events are triggered:
  
  – If the scale is idle ($W(t) = 0$), truck $j$ begins weighing and an end-weighing event ($EW$) is scheduled on the FEL. Otherwise, truck $j$ joins the weigh queue.
  
  – If there is another truck waiting for a loader, it will be removed from the loader queue and begin loading by the scheduling of an end-loading event ($EL$) on the FEL.
  
  – Both this logic for the occurrence of the end-loading event and the appropriate logic for the other two events should be incorporated into an event diagram.
## Dump-Truck Example

<table>
<thead>
<tr>
<th>Clock t</th>
<th>LQ(t)</th>
<th>System State</th>
<th>WQ(t)</th>
<th>W(t)</th>
<th>Loader Queue</th>
<th>Weigh Queue</th>
<th>Future Event List</th>
<th>Cumulative Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>DT4</td>
<td>DT5</td>
<td>(EL, 5, DT3)</td>
<td>0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT5</td>
<td>DT6</td>
<td>(EL, 10, DT2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(EW, 12, DT1)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>DT5</td>
<td>DT3</td>
<td>(EL, 10, DT2)</td>
<td>10 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT5</td>
<td>DT6</td>
<td>(EL, 5+5, DT4)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(EW, 12, DT1)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>DT6</td>
<td>DT3</td>
<td>(EL, 10, DT4)</td>
<td>20 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT6</td>
<td>DT2</td>
<td>(EW, 12, DT1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(EL, 10+10, DT5)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>DT3</td>
<td>DT2</td>
<td>(EW, 12, DT1)</td>
<td>20 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT3</td>
<td>DT4</td>
<td>(EL, 20, DT5)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(EW, 10+15, DT6)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>DT2</td>
<td>DT4</td>
<td>(EL, 20, DT5)</td>
<td>24 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT2</td>
<td></td>
<td>(EW, 12+12, DT3)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(EL, 25, DT6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 12+60, DT1)</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>DT2</td>
<td>DT4</td>
<td>(EW, 24, DT5)</td>
<td>40 20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT2</td>
<td>DT5</td>
<td>(EL, 25, DT6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 72, DT1)</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>DT4</td>
<td>DT5</td>
<td>(EL, 25, DT6)</td>
<td>44 24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT4</td>
<td></td>
<td>(EW, 24+12, DT2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 72, DT1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 24+100, DT3)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>DT4</td>
<td>DT5</td>
<td>(EW, 36, DT2)</td>
<td>45 25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DT5</td>
<td>DT6</td>
<td>(ALQ, 72, DT1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 124, DT3)</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>DT5</td>
<td>DT6</td>
<td>(EW, 36+16, DT4)</td>
<td>45 36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 72, DT1)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 36+40, DT2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(ALQ, 124, DT3)</td>
<td></td>
</tr>
</tbody>
</table>
Dump-Truck Example

- On the previous slide:
  - At time 0, 5 trucks at the loaders and 1 is at the scale.
  - The imminent event is an $EL$ event with time 5, hence clock is advanced to time $t = 5$.
  - Truck 3 joins the weigh queue (because the scale is occupied).
  - Truck 4 begins to load, schedule an $EL$ event for future time 10.
Dump-Truck Example

- Two cumulative statistics are maintained:
  - $B_L$ is total busy time of both loaders from time 0 to time $t$.
  - $B_S$ is total busy time of the scale from time 0 to time $t$.
    * Both loaders are busy from time 0 to time 20, so $B_L = 40$ at time 20.
    * From time 20 to time 24, only one loader is busy, thus $B_L$ increases by only 4 minutes over the time interval $[20, 24]$.
    * From time 25 to time 36, both loaders are idle ($L(25) = 0$) so $B_L$ does not change.
Dump-Truck Example

• Under the activity-scanning approach, the conditions for beginning each activity are:
  – Activity: Load time. Condition: Truck is at front of loader queue, and at least one loader is idle.
  – Activity: Weighing time. Condition: Truck is at front of weigh queue, and weigh scale is idle.
  – Activity: Travel time. Condition: Truck has just completed a weighing.
Using process-interaction approach, we view the model from the viewpoint of one dumper truck and its ‘life cycle’.

Considering a life cycle as beginning at the loader queue, we can picture a dump-truck process as:

\[ \text{Delay} \quad \text{Loading} \quad \text{Delay} \quad \text{Weigh} \quad \text{Travel time} \]

\[ ALQ \quad EL \quad EW \quad ALQ \]
Lists: Basic Properties and Operations

- Lists are a set of ordered or ranked records.
  - In summation, each record represents one entity or one event notice.
  - They have top or head (first item) and bottom or tail.
  - Ways to traverse the list (to find the second, third etc. items on the list) are necessary.

- An entity identifier and its attributes are ‘fields’ in the entity record.

- Each record on a list has a field that holds a ‘next pointer’ that points to the next record on the list.
Lists: Basic Properties and Operations

- The main operations on a list are:
  - Removing a record from the top of the list.
  - Removing a record from any location on the list.
  - Adding an entity record to the top or bottom of the list.
  - Adding a record at an arbitrary position in the list, specified by the ranking rule.
In the event-scheduling approach, when time is advanced and the imminent event is due to be executed:

- First, the removal operation takes place.
- If an arbitrary event is being canceled, or an entity is removed from a list based on some of its attributes to being an activity, then the second removal operation is performed.
- If a queue has the ranking rule ‘earliest’ due date first, then, upon arrival at the queue, an entity must be added to the list determined by the due-date ranking rule.
Lists: Basic Properties and Operations

For simulation on a computer:

- All records are stored in arrays: arrays hold successive records in contiguous locations in computer memory, referenced by array index.
- All entities and event notices are represented by structure (as in C) or classes (as in Java) allocated from RAM as needed, and tracked by pointers to a record or structure.
Using Arrays

• The array method of list storage is typical of FORTRAN.

• Most modern simulation packages do not use arrays for list storage, but rather use dynamically allocated records.

• Advantages of arrays:
  – Any specified record (say the $i$-th record) can be retrieved quickly without searching, merely by referencing $R(i)$.

• Disadvantages of arrays:
  – The list is rearranged when items are added to the middle of a list.
  – Have a fixed size which is determined at compile time.
Using Arrays

- Two basic methods for keeping track of record ranking in a list:
  - Method 1: Store the first record in $R(1)$ and the second in $R(2)$ and so on:
    * Extremely inefficient.
    * For example, adding a record in position 41 in a list of 100 items requires that the last 60 records be physically moved down one array position to make space for the new record.
  - Method 2: Use head pointer:
    * A variable that points to the record at the top of the list, denoted as ‘headptr’
    * For example, if the record in position $R(1)$ were the record at the top of the list, then headptr would have value 1.
Using Arrays

• Recall the dump-truck problem: At clock time 10, there is a waiting line of 3 dump trucks occurred at the weigh queue, specifically, $DT_3$, $DT_2$ and $DT_4$ (in this order).

• Suppose the model is tracking one attribute of each dump truck: its arrival time at the weigh queue, updated each time it arrives.

• Suppose that the entities are stored in records in an array dimensioned from 1 to 6, one record for each dump truck.
  – Each entity is represented by a record with 3 fields:
    * The first is an entity identifier.
    * The second is the arrival time at the weigh queue.
    * The last is a pointer field to ‘point to’ the next record.
    * ($DT_i$, arrival time at weigh queue, next index)
Dump-Truck Example

• At time 0, the records would be initialized as follows:

\[ R(1) = [DT1, 0, 0], R(2) = [DT2, 0, 0], R(3) = [DT3, 0, 0] \]
\[ R(4) = [DT4, 0, 0], R(5) = [DT5, 0, 0], R(6) = [DT6, 0, 0] \]

• At clock time 10, the list of entities in the weigh queue would be defined by headptr=3 and:

\[ R(1) = [DT1, 0, 0], R(2) = [DT2, 10, 4], R(3) = [DT3, 5, 2] \]
\[ R(4) = [DT4, 10, 0], R(5) = [DT5, 0, 0], R(6) = [DT6, 0, 0] \]

• To traverse the list, start with the head pointer, go to that record, retrieve that record’s next point, and proceed. To create the list in its logical order, for example headptr=3

\[ R(3) = [DT3, 5, 2], R(2) = [DT2, 10, 4], R(4) = [DT3, 10, 0] \]

– The zero entry for \( R(4) \) indicates that \( DT4 \) is at the end of the list.
Dump-Truck Example

- At clock time 12, dump truck $DT_3$ begins weighing and thus leaves the weigh queue.
  - To remove the $DT_3$ entity record from the top of the list, update the head pointer by setting it equal to the next pointer value at the top of the list, \( \text{i.e., headptr} = R(\text{headptr}, \text{next}) \).
  - In this example, we get headptr$=R(3, \text{next})=2$. Hence dump truck $DT_2$ in $R(2)$ is now at the top of the list.
Using Dynamic Allocation and Linked Lists

• Used in procedural languages such as C++ and Java, and in most simulation languages.

• Entity records are dynamically created when an entity is created.

• Event notice records are dynamically created whenever an event is scheduled on the future event list.

• Basic mechanics:
  – The languages maintain a linked list of free chunks of computer memory and allocate a chunk of desired size upon request to running programs.
  – When an entity exists from the simulated system, and also after an event occurs and the event notice is no longer needed, the corresponding records are free.
Using Dynamic Allocation and Linked Lists

- A record is referenced by a pointer instead of by an array index.
  - A pointer to a record can be thought of as the physical or logical address in computer memory of the record.
  - If the third item on the list is needed, we need to traverse the list, counting items until we reach the record.

- Notation for records:
  - Entities: (ID, attributes, next pointer).
  - Event notices: (event type, event time, other data, next pointer).

- List types:
  - Singly-linked lists: one-way linkage from the head of the list to its tail.
  - Doubly-linked lists: records have two pointer fields, one for the next record and one for the previous record.
Dump-Truck Example

- Event notices in the dump truck problem are expanded to include a pointer to the next event notice on the future event list, and can be represented by (event type, event time, $DT_i$, nextptr):
  - Keep in mind that records may be stored anywhere in computer memory.

- For example, $[EL, 10, DT3, nextptr]$
  - At future event list at clock time 10, the future event list as a linked list:
Summary

• Introduced the major concepts and building blocks in simulation:
  – Entities and attributes.
  – Events and activities.

• Three major world views:
  – Event-scheduling.
  – Process interaction.
  – Activity scanning.

• Basic notions of list processing are introduced to gain understanding of important underlying methodologies.