Behavioral models describe the internal dynamic aspects of an IS that supports the business processes in an organization.

- During the analysis phase, behavioral models describe what the internal logic of the processes is without specifying how the processes are to be implemented.
- In the design and implementation phases, the detailed design of the operations contained in the object is fully specified.

In this lecture, three UML 2.0 diagrams that are used in behavioral modeling will be described:
- sequence diagrams, communication diagrams, behavioral state machines.

Objectives

- Understand the rules and style guidelines for sequence and communication diagrams and behavioral state machines.
- Understand the processes used to create sequence and communication diagrams and behavioral state machines.
- Be able to create sequence and communication diagrams and behavioral state machines.
- Understand the relationship between the behavioral models and the structural and functional models.

Systems analysts

- use functional models
  - to describe the external behavioral view of an information system,
- use structural models
  - to depict the static view of an information system,
- use behavioral models
  - to represent the internal behavior or dynamic view of an information system.

There are two types of behavioral models:

- behavioral models that are used to represent the underlying details of a business process portrayed by a use case model.
  - In UML, interaction diagrams (sequence and communication) are used for this type of behavioral model.
  - The behavioral model that is used to represent the changes that occur in the underlying data.
  - UML uses behavioral state machines for this.
- During the analysis phase, analysts use behavioral models to capture a basic understanding of the dynamic aspects of the underlying business process.
- Traditionally, behavioral models have been used primarily during the design phase where analysts refine the behavioral models to include implementation details.

BEHAVIORAL MODELS

Structural model describes the structure of the data that supports the business processes in an organization.

During the analysis phase, the structural model presents the logical organization of data without indicating how the data are stored, created, or manipulated so that analysts can focus on the business without being distracted by technical details.

Later, during the design phase, the structural model is updated to reflect exactly how the data will be stored in databases and files.
• Primary purposes of behavioral models
  – to show how the underlying objects in the problem domain will collaborate to support each of the use cases.
• Structural models
  – represent the objects and the relationships between them.
• Behavioral models
  – depict the internal view of the business process that a use case describes.
• The process can be shown by the interaction
  – taking place between the objects that collaborate to support a use case through the use of interaction diagrams.
• It is also possible to show the effect that the set of use cases that make up the system has on the objects in the system through the use of behavioral state machines.

In this lecture we will describe
• interaction diagrams
• behavioral state machines
• their uses.

Differences between the class diagrams and the interaction diagrams:
• class diagrams
  – describes structure
  – the modeling focus is at the class level.
• interaction diagrams
  – describes behavior.
  – modeling focus is at the object level.
• Next we review objects, operations, and messages and we cover the two different diagrams
  – sequence diagrams
  – communication (collaboration) diagrams
that can be used to model the interactions that take place between the objects in an information system.

Interaction Diagram Components

• object
  – an instantiation of a class.
  • an actual person, place, event, or thing about which we want to capture information.
    – If we were building an appointment system for a doctor’s office,
      classes might include doctor, patient, and appointment. The specific patients like Jim Maloney, Mary Wilson, and Theresa Marks are considered objects— that is, instances of the patient class.
  – Each object has attributes
    • describing information about the object,
      – such as a patient’s name, birth date, address, and phone number.
• Each object also has behaviors.
• In the development of the evolving system, the behaviors are described by operations.

• Creating behavioral models
  – an iterative process
    – iterating over the individual behavioral models (e.g., interaction (sequence and communication) diagrams and behavioral state machines)
    – iterating over the functional and structural models.
• As the behavioral models are created, making changes to the functional and structural models may be required.

• Operation
  – an action that an object can perform.
  – For example,
    • an appointment object likely can schedule a new appointment, delete an appointment, and locate the next available appointment.
  – Each object also can send and receive messages.
• Messages
  – information sent to objects to tell an object to execute one of its behaviors.
  – a message is a function or procedure call from one object to another object.
  – For example,
    • if a patient is new to the doctor’s office, the system will send an insert message to the application. The patient object will receive the instruction (the message) and do what it needs to do to go about inserting the new patient into the system (the behavior).
Sequence Diagrams

• illustrate the objects that participate in a use case
• show the messages that pass between them over time for one use case.
• A sequence diagram
  – a dynamic model
    • showing the explicit sequence of messages that are passed between objects in a defined interaction.
• emphasize the time-based ordering of the activity that takes place among a set of objects.
• very helpful for understanding real-time specifications and complex use cases.

• be a generic sequence diagram that shows all possible scenarios for a use case,
  – but usually each analyst develops a set of instance sequence diagrams.
    • each of which depicts a single scenario within the use case.
• If you are interested in understanding the flow of control of a scenario by time, you should use a sequence diagram to depict this information.
• The diagrams are used throughout both the analysis and design phases.
  – However, the design diagrams are very implementation-specific, often including database objects or specific GUI components as the classes.

Elements of a Sequence Diagram

• Next figure shows an instance sequence diagram
  – depicting the objects and messages for the Make Appointment use case,
    • which describes the process by which a patient creates a new appointment, cancels, or reschedules an appointment for the doctor's office appointment system.
• In this specific instance, the make appointment process is portrayed.

Elements of a Sequence Diagram

• Actors and objects that participate in the sequence are placed across the top of the diagram using actor symbols from the use case diagram, and object symbols from the object diagram.
• Notice that the actors and objects in the previous figure are aPatient, aReceptionist, Patients, UnpaidBills, Appointments, and anAppt.
• They are not placed in any particular order, although it is nice to organize them in some logical way, such as the order in which they participate in the sequence.
• For each of the objects, the name of the class that they are an instance of is given after the object's name (e.g., Patients:List means that Patients is an instance of the List class that contains individual patient objects).

Sequence Diagram Syntax

An Actor:
• Is a person or system that derives benefit from and is external to the system
• Participates in a sequence by sending and/or receiving messages
• Are placed across the top of the diagram
• Is depicted as either a stick figure (default) or if a non-human actor is involved, as
• a rectangle with <<actor>> in it (alternative)

Sequence Diagram Syntax

An Object:
• Participates in a sequence by sending and/or receiving messages
• Are placed across the top of the diagram
A Lifeline:
• Denotes the life of an object during a sequence
• Contains an “X” at the point at which the class no longer interacts
An Execution Occurrence:
• Is a long narrow rectangle placed atop a lifetime
• Denotes when an object is sending or receiving messages
**Sequence Diagram Syntax**

### A Message:
- Conveys information from one object to another one.
- An operation call is labeled with the message being sent and a solid arrow, while a return is labeled with the value being returned and shown as a dashed arrow.

### Object Destruction:
- An X is placed at the end of an object’s lifeline to show that it is going out of existence.

### A Frame:
- Indicates the context of the sequence diagram.

---

**Sequence Diagram Syntax**

- A dotted line runs vertically below each actor and object to denote the lifetime of the actor/objects over time.
- Sometimes an object creates a temporary object, and in this case, an X is placed at the end of the lifetime at the point the object is destroyed.
  - For example, think about a shopping cart object for a Web commerce application.
  - The shopping cart is used for temporarily capturing line items for an order, but once the order is confirmed, the shopping cart is no longer needed.
  - In this case, an X would be located at the point at which the shopping cart object is destroyed.
- When objects continue to exist in the system after they are used in the sequence diagram, then the lifeline continues to the bottom of the diagram.

---

**Sequence Diagram Syntax**

- A thin rectangular box, called the execution occurrence, is overlaid onto the lifeline to show when the classes are sending and receiving messages.
- A message is a communication between objects that conveys information with the expectation that activity will ensue:
  - There are many different types of messages that can be portrayed on a sequence diagram.
  - However, in the case of using sequence diagrams to model use cases, two types of messages are typically used:
    - Operation Call
    - Return Call

---

**Sequence Diagram Syntax**

- At times a message is sent only if a condition is met.
  - In those cases, the condition is placed between a set of [ ].
  - The condition is placed in front of the message name.
    - However, when using a sequence diagram to model a specific scenario, conditions typically are not shown on any single sequence diagram.
    - Instead, conditions are implied only through the existence of different sequence diagrams.
- There are times that a message is repeated.
  - This is designated with an * in front of the message name.
- An object can send a message to itself.
  - This is known as self-delegation.
- Sometimes, an object will create another object.
  - This is shown by the message being sent directly to an object instead of its lifeline.

---

**Building a Sequence Diagram**

1. Determine the context of the sequence diagram.
   - The context of the diagram can be:
     - a system,
     - a use case,
     - a scenario of a use case,
     - an operation of a class.
   - The context of the diagram is depicted as a labeled frame around the diagram.
   - Most commonly, it is one use case scenario.
2. Identify the participating objects

- The objects are identified during the development of the structural model.
- These are the classes on which the objects of the sequence diagram for this scenario will be based.
- However, during this process, it is likely that new classes, and hence new objects, will be uncovered.
- Because the behavioral modeling process is iterative, usually, the sequence diagrams are revised multiple times during the structural and behavioral modeling processes.

3. Set the lifeline for each object

- draw a vertical dotted line below each class to represent the class’s existence during the sequence.
- An X should be placed below the object at the point on the lifeline where the object goes out of existence.

4. Add messages

- done by drawing arrows to represent the messages being passed from object to object, with the arrow pointing in the message’s transmission direction.
- The arrows should be placed in order from the first message (at the top) to the last (at the bottom) to show time sequence.
- Any parameters passed along with the messages should be placed in parentheses next to the message’s name.
- If a message is expected to be returned as a response to a message, then the return message is not explicitly shown on the diagram.

5. Place the focus of control on each object’s lifeline

- by drawing a narrow rectangle box over top of the lifelines to represent when the classes are sending and receiving messages.

6. Validate the sequence diagram

- to guarantee that the sequence diagram completely represents the underlying process(es).
- This is done by guaranteeing that the diagram depicts all of the steps in the process.

Example: CD Selections

Normal Flow of Events:

1. Customer submits a search request to the system.
2. The system provides the customer a list of recommended CDs.
3. The customer chooses one of the CDs to find additional information.
4. The system provides the customer with basic information & CD Reviews.
5. The customer calls the maintain order use case.
6. The customer iterates over 3 through 5 until finished shopping.
7. The customer executes the checkout use case.
8. The customer leaves the website.

Sequence Diagram for the Places Order Use Case

Communication Diagrams

- provide a view of the dynamic aspects of an OO system.
- very useful to show process patterns - i.e., patterns of activity that occur over a set of collaborating classes.
- A communication diagram
- an object diagram that shows message passing relationships instead of aggregation or generalization associations.
- Communication diagrams
- emphasize the flow of messages through a set of objects
- Sequence diagrams
- focus on the time ordering of the messages being passed.
• Therefore, if you are interested in understanding the flow of control over a set of collaborating objects, you should use a communication diagram. If you are interested in the time ordering of the messages, you should use a sequence diagram. In some cases, you will want to use both to more fully understand the dynamic activity of the system.

Communication Diagrams

Communication Diagram Syntax

An Actor:
• Is a person or system that derives benefit from and is external to the system
• Participates in a collaboration by sending and/or receiving messages

An Object:
• Participates in a collaboration by sending and/or receiving messages
• placed across the top of the diagram

Communication Diagram Syntax

An Association:
• Shows an association between actors and/or objects
• Messages are sent over associations

A Message:
• Conveys information from one object to another one
• Direction is shown using an arrowhead
• Sequence is shown by a sequence number

A Frame:
• Indicates the context of the communication diagram

Communication Diagram Syntax

Example Communication Diagram

Building a Communication Diagram

1. Set the context.
• the context of the diagram
– can be
  • a system, a use case, a scenario of a use case, an operation of a class.
– depicted as a labeled frame around the diagram

2. Identify which objects (actors) and the associations between the objects participate in the collaboration.
• the objects that participate in the collaboration are instances of the classes identified during the development of the structural model

Building a Communication Diagram

• Since the underlying development process is iterative and incremental, it is likely that additional objects, and hence classes, may be discovered.
• One useful technique to identify potential collaborations is CRUD analysis.
– a matrix in which each interaction among objects is labeled with a letter for the type of interaction:
  • C for Create,
  • R for Read or Reference,
  • U for update,
  • D for Delete.
• In an OO approach, a class-by-class matrix is used.
– Each cell in the matrix represents the interaction between instances of the classes.

Building a Communication Diagram
For example in the previous figure, an instance of SearchReq created an instance of CDList.
- As such, assuming a Row:Column ordering, a “C” would be placed in the cell SearchReq:CDList.
- Also, an instance of CD references an instance of MktInfo. In this case, an “R” would be placed in the CD:MktInfo cell.

The following figure shows the CRUD matrix based on the Place Order use case.

<table>
<thead>
<tr>
<th>Customer</th>
<th>SearchReq</th>
<th>CDList</th>
<th>CD</th>
<th>MktInfo</th>
<th>Review</th>
<th>Artist Info</th>
<th>Sample Clip</th>
<th>Shopping Cart</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SearchReq</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CDList</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MktInfo</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Review</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Artist Info</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sample Clip</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shopping Cart</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Order</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Once a CRUD matrix is completed for the entire system, the matrix can be scanned quickly to ensure that each and every class can be instantiated.

Furthermore, each type of interaction can be validated for each class.
- For example, if a class only represents temporary objects, then the column in the matrix should have a D in it somewhere.
- Otherwise, the instances of the class will never be deleted.

3. Lay out the objects (actors) and their associations on the communication diagram.
- By placing them together based on the associations that they have with the other objects in the collaboration.
- By focusing on the associations between the objects (actors) and minimizing the number of associations that cross over one another, the understandability of the diagram can be increased.

4. Add messages to the associations between the objects.
- By adding the name of the message(s) to the association link between the objects and an arrow showing the direction of the message being sent.
- Each message has a sequence number associated with it to portray the time-based ordering of the message.

5. Validate the communication diagram.
- The purpose of this step is to guarantee that the communication diagram faithfully portrays the underlying process(es).
- This is done by ensuring that all steps in the process are depicted on the diagram.

Example: CD Selections

- Communication Diagram for the Place Order Use Case

BSMs are used to further define complex objects to help simplify the design of algorithms for their methods.
- CRUD analysis can be used to identify complex objects.
  - The more (C)reate, (U)pdate or (D)elete entries in the column associated with a class, the more likely the instances of the class will have a complex life-cycle.
    - As such, these objects are candidates for state modeling with a BSM.
- In comparison to the interaction diagrams,
  - BSMs should be used
    - If you are interested in understanding the dynamic aspects of a single class and how its instances evolve over time,
    - Not with how a particular use case scenario is executed over a set of classes.

The behavioral state machine (BSM)
- A dynamic model showing
  - The different states of the object
  - What events cause the object to change from one state to another, along with its responses and actions.
Elements of a Behavioral State Machine

- States
  - State of an object
    - defined by the value of its attributes and its relationships with other objects at a particular point in time.
    - For example, a patient might have a state of “new” or “current” or “former.”
  - The attributes or properties of an object affect the state that it is in;
    - however, not all attributes or attribute changes will make a difference.
    - For example, a patient’s address make very little difference as to changes in a patient’s state.
    - However, if states were based on a patient’s geographic location (e.g., in-town patients were treated differently than out-of-town patients), changes to the patient’s address would influence state changes.

- Events
  - An event
    - something that takes place at a certain point in time and changes a value that describes an object,
    - which, in turn, changes the object’s state.
    - It can be
      - a designated condition becoming true,
      - the receipt of the call for a method by an object,
      - the passage of a designated period of time.
  - The state of the object determines exactly what the response will be.

- Transitions
  - A transition
    - a relationship that represents the movement of an object from one state to another state.
    - Some transitions will have a guard condition.
      - A guard condition
        - a Boolean expression that includes attribute values, which allows a transition to occur only if the condition is true.
    - An object typically moves from one state to another
      - based on the outcome of an action that is triggered by an event.

- Actions
  - An action
    - an atomic, nondecomposable process than cannot be interrupted.
    - From a practical perspective, actions
      - take zero time,
      - associated with a transition.
  - An activity
    - a nonatomic, decomposable process that can be interrupted.
    - Activities
      - take a long period of time to complete,
      - can be started and stopped by an action,
      - associated with states.

- Activities

Elements of a Behavioral State Machine

An example of a behavioral state machine representing the patient class in the context of a hospital environment.

- a patient enters a hospital and is admitted after checking in.
  - If a doctor finds the patient to be healthy, (s)he is released, and is no longer considered a patient after two weeks elapses.
  - If a patient is found to be unhealthy, (s)he remains under observation until the diagnosis changes.
A State:
- Is shown as a rectangle with rounded corners
- Has a name that represents the state of an object

An Initial State:
- Is shown as a small filled-in circle
- Represents the point at which an object begins to exist

A Final State:
- Is shown as a circle surrounding a small solid filled circle (bull’s-eye)
- Represents the completion of activity

An Event:
- Is a noteworthy occurrence that triggers a change in state
- Can be a designated condition becoming true, the receipt of an explicit signal
- from one object to another, or the passage of a designated period of time
- Is used to label a transition

A Transition:
- Indicates that an object in the first state will enter the second state
- Is triggered by the occurrence of the event labeling the transition
- Is shown as a solid arrow from one state to another, labeled by the event name

A Frame:
- Indicates the context of the behavioral state machine

1. Determine the context of the behavioral state machine
   - The context of a behavioral state machine is usually a class.
   - However, it also could be a set of classes, a subsystem, or an entire system

2. Identify the initial final, and stable states of the object
   - This includes establishing the boundaries of the existence of an object by identifying the initial and final states of an order object.
   - We also must identify the stable states of an object.
   - The information necessary to perform this is gleaned from:
     - reading the use case descriptions,
     - talking with users
     - relying on the requirements gathering techniques

3. Determine the order in which the object will pass through stable states
   - Using this sequence, the states are placed onto the behavioral state machine, in a left-to-right order.

4. Identify the events, actions, and guard conditions associated with the transitions
   - The events are the triggers that cause an object to move from one state to the next state:
     - an event causes an action to execute that changes the value(s) of an object’s attribute(s) in a significant manner:
       - actions are typically operations contained within the object.
       - guard conditions can model a set of test conditions that must be met for the transition to occur.
     - At this point in the process, the transitions are drawn between the relevant states and labeled with the event, action, or guard condition.

5. Validate the behavioral state machine
   - by making sure that:
     - each state is reachable
     - it is possible to leave all states except for final states.
   - If an identified state is not reachable:
     - either a transition is missing or the state was identified in error.
   - Furthermore, only final states can be a dead end from the perspective of an object-life cycle.
The Life of an Order:

1. The customer creates an order on the Web
2. The customer submits the order once he or she is finished
3. The credit authorization needs to be approved for the order to be accepted
4. If denied, the order is returned to the customer for changes or deletion
5. If accepted, the order is placed
6. The order is shipped to the customer
7. The customer receives the order
8. The order is closed

Example: CD Selections

• Behavioral State Machine for the Order Class

Your Turn

• What distinguishes the sequence diagram, the collaboration diagram, and the behavioral state machine diagram?
• For what sort of new applications might you need to develop all of these?
• Are there any new applications that would not need all of these diagrams for full development?

Summary

• Sequence diagrams illustrate the classes that participate in a use case and the messages that pass between them.
• Collaboration diagrams provide a dynamic view of the object-oriented system and accentuate message passing between collaborating actors and objects.
• Behavioral State Machine diagrams show the different states that a single class passes through in response to events.

Expanding the Domain

• Each year the Association for Computing Machinery (ACM) sponsors a conference on object oriented programming.
• For details about future conferences and other ACM programs check:
  • http://ooopsla.acm.org
  • http://www.acm.org