Interaction Modelling: Sequence Diagrams

Fabrizio Maria Maggi
Institute of Computer Science

(These slides are derived from the book “Object-oriented modeling and design with UML”)
Interaction Modelling: Detailing Use Cases with Scenarios

**Use Case**: Buy a beverage

**Summary**: The vending machine delivers a beverage after a customer selects and pays for it.

**Actors**: Customer

**Preconditions**: The machine is waiting for money to be inserted.

**Description**: The machine starts in the waiting state in which it displays the message “Enter coins.” A customer inserts coins into the machine. The machine displays the total value of money entered and lights up the buttons for the items that can be purchased for the money inserted. The customer pushes a button. The machine dispenses the corresponding item and makes change, if the cost of the item is less than the money inserted.

**Exceptions**: Canceled: If the customer presses the cancel button before an item has been selected, the customer’s money is returned and the machine resets to the waiting state.

Out of stock: If the customer presses a button for an out-of-stock item, the message “That item is out of stock” is displayed. The machine continues to accept coins or a selection.

Insufficient money: If the customer presses a button for an item that costs more than the money inserted, the message “You must insert $n.mn more for that item” is displayed, where $n.mn is the amount of additional money needed. The machine continues to accept coins or a selection.

No change: If the customer has inserted enough money to buy the item but the machine cannot make the correct change, the message “Cannot make correct change” is displayed and the machine continues to accept coins or a selection.

**Postconditions**: The machine is waiting for money to be inserted.

---

*Figure 7.3: Use case description. A use case brings together all the behavior relevant to a slice of system functionality.*

Interaction Modelling

- Customer
- Vending Machine
  - 1. InsertCoin()
  - 2. totalAmount
  - 3. chooseItem

Vending Machine
  - buy beverage
  - perform scheduled maintenance
  - make repairs
  - load items

- Customer
- Repair technician
- Stock clerk
Sequence Diagrams

- A sequence diagram shows the participants in an interaction and the sequence of messages among them.
- A sequence diagram shows the interaction of a system with its actors to perform a use case.
Sequence Diagrams

- Sequence diagrams show procedure calls.
- An object is activated only when it is called. Once the execution of an operation completes the control returns to the caller and the object becomes inactive.
The period of time of an object’s execution is a thin rectangle called activation or focus of control.
Procedural Sequence Diagrams

An activation shows the time period during which a call of a method is processed including the time when the called method invoke other methods.
The period of time when an object exists but is not active is shown as a dashed line.
The entire period of time when an object exists is called **lifeline**
The notation for a call is an arrow from the calling activation to the activation created by the call.
A return of a call is a dashed arrow from the bottom of the called activation to the calling activation.
An activation has a call arrow coming into its top and a return arrow leaving its bottom.
Objects A and B exist during the entire time shown in the diagram, whereas object C is created and destroyed in a smaller period of time. Therefore, its lifetime does not span the whole diagram.
Procedural Sequence Diagrams

If an object does not exist at the beginning of the sequence diagram, it must be created. UML shows creation by placing the object symbol at the head of the dashed arrow representing the call that creates the object.
A large ‘X’ marks the end of the life of an object that is destroyed during the sequence diagram. The ‘X’ is placed at the head of the call arrow that destroys the object.
During a call to a method on an object there can be another call to another method on the same object. This call is shown with an arrow from the activation rectangle to the top of an additional rectangle superimposed on the first.
An object can call its own operations (self calls)
A simple example
Advanced Sequence Diagrams: alt

bank : Bank

theCheck : Check

account : CheckingAccount

getAmount ( )

amount

getBalance ( )

balance

[Balance >= amount]

addDebitTransaction ( check, Number, amount )

storePhotoOfCheck ( theCheck )

[else]

addInsufficientFundsFee ( )

noteReturnedCheck ( theCheck )

returnCheck ( theCheck )

http://www.ibm.com
Advanced Sequence Diagrams: opt

Diagram:

```
register : RegisterOffice

ar : AccountsReceivable
drama : Class

getPastDueBalance (studentId)
pastDueBalance

[opt]
[pastDueBalance = 0]
addStudent (studentId)

getCostOfClass()
classCost

chargeForClass()
```
Advanced Sequence Diagrams: loop
Advanced Sequence Diagrams: break
Sequence Diagrams: Note for the reader

Useful material about sequence diagrams can be found on the course website:

https://courses.cs.ut.ee/MTAT.03.083/2016_fall/uploads/Main/MaterialSD
Closing the circle
From a Domain model to an Application model
From a Domain model to an Application model
The Entity-Control-Boundary Pattern

Entities
Objects representing system data, often from the domain model.

Boundaries
Objects that interface with system actors (e.g. a user or external service). Windows, screens and menus are examples of boundaries that interface with users.

Controls
Objects that mediate between boundaries and entities. These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions. It is important to understand that you may decide to implement controllers within your design as something other than objects — many controllers are simple enough to be implemented as a method of an entity or boundary class for example.

Four rules apply to their communication:

1. Actors can only talk to boundary objects.
2. Boundary objects can only talk to controllers and actors.
3. Entity objects can only talk to controllers.
4. Controllers can talk to boundary objects and entity objects, and to other controllers, but not to actors

Communication allowed:

<table>
<thead>
<tr>
<th></th>
<th>Entity</th>
<th>Boundary</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Boundary</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>