Interaction Models

• The purpose of building interaction models is to assign responsibilities to objects in the conceptual model

• There are two equivalent interaction models: collaboration diagram and sequence diagram
Comparison

• Collaboration and sequence diagrams are semantically equivalent.
• Collaboration is good at showing objects are statically connected.
  – It is similar to the conceptual diagram derived during analysis and class diagram derived during design.
• Sequence diagram is good at showing the order in which things occur.
• Both models quickly lose their advantage for expressing complex conditional and looping logic.
Diagramming interaction

To make interaction diagrams:

1. Create a separate diagram for each system operation under development in the current iterative step.
   - For each system event, make a diagram with it as the starting message.

2. If the diagram gets complex, split it into smaller diagrams.

3. Using the contract responsibilities and post-conditions, and use case description as a starting point, design a system of interacting objects to fulfill the tasks. Apply the GRASP and other patterns to develop a good design.
Artifacts

• Artifacts at hand:
  – Use cases
  – System events and contracts
  – Conceptual model

• Artifacts sought for:
  – Interaction diagrams
  – Class diagram (adding operations to the objects)
Use case: buy items

Typical Course of Events

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This use case begins when a Customer arrives at a POST checkout with items to purchase.</td>
<td></td>
</tr>
<tr>
<td>2. The Cashier records the universal product code (UPC) from each item. If there is more than one of the same item, the Cashier can enter the quantity as well.</td>
<td></td>
</tr>
<tr>
<td>3. Determines the item price and adds the item information to the running sales transaction.</td>
<td></td>
</tr>
<tr>
<td>4. On completion of item entry, the Cashier indicates to the POST that item entry is complete.</td>
<td></td>
</tr>
<tr>
<td>5. Calculates and presents the sale total.</td>
<td></td>
</tr>
<tr>
<td>6. The Cashier tells the Customer the total.</td>
<td></td>
</tr>
<tr>
<td>7. The Customer gives a cash payment—the “cash tendered”—possibly greater than the sale total.</td>
<td></td>
</tr>
</tbody>
</table>
Use case: buy items (cont.)

<table>
<thead>
<tr>
<th>Actor Action</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>8. The Cashier records the cash received amount.</td>
<td>9. Shows the balance due back to the Customer.</td>
</tr>
<tr>
<td>9. Shows the balance due back to the Customer.</td>
<td>Generates a receipt.</td>
</tr>
<tr>
<td>10. The Cashier deposits the cash received and extracts the balance owing. The Cashier gives the balance owing and the printed receipt to the Customer.</td>
<td>11. Logs completed sale.</td>
</tr>
<tr>
<td>12. The Customer leaves with the items purchased.</td>
<td></td>
</tr>
</tbody>
</table>
Conceptual model

Figure 10.7 A point-of-sale conceptual model.
### Contract: enterItem

**Name:** enterItem  
(upc : number,  
quantity : integer)

**Responsibilities:** Enter (record) sale of an item and add it to the sale.  
Display the item description and price.

**Type:** System

**Cross References:** System Functions: RI.1, RI.3, RI.9

**Notes:** Use Cases: Buy Items

**Exceptions:** If the UPC is not valid, indicate that it was an error.

**Output:**

**Pre-conditions:** UPC is known to the system

**Post-conditions:**

- If a new sale, a Sale was created (instance creation).
- If a new sale, the new Sale was associated with the POST (association formed).
- A SalesLineItem was created (instance creation).
- The SalesLineItem was associated with the Sale (association formed).  
SalesLineItem.quantity was set to quantity (attribute modification).
- The SalesLineItem was associated with a ProductSpecification, based on  
UPC match (association formed).
Where to start

• First, who should be responsible for handing the system event?
  – System is an obvious choice, but it’s too vague
  – Scan the conceptual model and pick one. The one you chose is responsible for controlling the interaction among objects, it is called a controller.
Controller: enterItem

Figure 19.5 Applying the GRASP Controller pattern.
Carrying on

• Second, ask “what is created?” and “how is the created object related to existing ones?”
  – At least certain objects need to be created (if not already created) in order to carry on the collaboration.
  – Scan the post-conditions part of the contract, fish out the initial objects to create.
  – Note that the created objects should not be floating around; they should associated to some existing objects.
  – In this example, a Sale is created; Sale was associated with POST.

• This step is repeated until we have all the objects in place to execute the responsibilities stated in contract
Creation and association formed

Figure 19.6 Sale creation.
Carrying on

• Third, have the controller initiate the collaboration to carry out the responsibilities.
  – Scan the responsibilities section in contract
  – Consider the messages sequence that goes on
  – Put effort in naming the messages sent between objects
Creation and association formed

Figure 19.7 The enterItem collaboration diagram.
Further Readings

- Larman 1st ed., Chapters 19
- Fowler/Distilled, Chapter 4.