Associations, contracts and UML interaction diagrams

Week 4 lecture
Aug 17, 2005

Agenda
- Refine Domain Model
  - More on Associations
- System sequence diagram
- Operation Contract
- On to Object design
  - Interaction diagrams
    - Sequence diagrams
    - Communication diagrams

Refine Domain model:
Aggregation vs. composition

- An aggregation is a special form or association
  - The classes relate to each other as a whole relates to its parts

- Composition is a strong kind of whole-part aggregation

Association, aggregation and composition

- Unconfirmed characteristics of composition
  - Cascading delete: when a composite object is deleted, the component objects get deleted too.
  - Aggregation is a placebo, hardly any semantic defined for aggregation in UML.
  - Whenever in doubt, use plain association or composition

Qualified association

- A qualified association has a qualifier that is used to select an object (or objects) from a larger set of related objects

Reflexive association

- A conceptual class may have association with itself
  - Eg. A Directory can contain other directories
  - Eg. A node in a liked list is linked to another node

Refine Domain model: Identifying Association classes

- Association classes
  - Associations can have attributes and operations attached them
  - Use association class to represent that kind of association

A scenario

- Authorization domain requirements
  - Authorization service assign a merchant ID to each store for identification
  - A payment authorization request from the store to an authorization service needs the merchant ID that identifies the store to the service
  - A store has a different merchant ID for each service

The problem

- A Store can have more than one value for merchantID
- An Authorization Service can have more than one value for merchantID as well

Modeling principle

- In a domain model, if a class C can simultaneously have many values for the same kind of attribute A, do not place attribute A in C. Place attribute A in another class that is associated with C.
- A Student may choose many unit of studies. From each unit of study, the student receives a mark. Place mark in a separate class Mark and associate it with Student and UnitOfStudy.

Two types of modeling

- Association class add an extra constraint
  - There can be only one instance of the association class between any two participating objects.
Guidelines

- An attribute is related to an association.
- Instances of the association class have a life-time dependency on the association.
- There is a many-to-many association between two concepts and information associated with the association itself.


What is System Sequence Diagram

- A diagram that shows, for ONE particular scenario of a use case, the events that external actors generate, their order, and INTER-system events. (not detailed method calls between objects)
- Describe what a system does without explaining why

Use Cases and SSDs

Choosing events and operation name

- System events should be expressed at the abstract level of intention rather than in terms of the physical input device
Iterative and Evolutionary SSDs

- Do not create SSDs for all scenarios (remember agile style)
- SSDs are part of the Use-Case Model
  - Visualization of the interactions implied in the use cases scenarios

Operation Contract

Example Contract

Contract CO2: enterItem

Operation: enterItem(itemID: Integer, quantity: Integer)
Cross References: Use Cases, Process Sale
Preconditions: This is a sale underway.
Postconditions: A SaleItem instance is created (instance creation).
- It is associated with the current Sale (association formed).
- Its quantity becomes quantity (attribute modification).
- It is associated with a ProductSpecification, based on itemID (association formed).

Contract Definition

Contract procedure

- To make contract
  - Identify System Operations from SSD
  - For system operations that are complex or subtle in their results or are not clearly express in use cases, construct a contract
  - To describe the post conditions, use the following categories
    - Instance creation and deletion
    - Attribute change of value
    - Associations (to be precise, UML links) formed and broken

System operations
Process Sale: makeNewSale

Contract C01: makeNewSale

Operation: makeNewSale
Preconditions: none
Postconditions:
- A Sale instance s was created (instance creation).
- It was associated with the Register (association formed).
- Attributes of s were initialized.

[Larman, 2002]

Object design introduction

- How do developers design objects
  - Code
  - Design-while-coding
  - Draw, then code
  - Only draw
- Agile modeling: reduce drawing overhead and model to understand and communicate
  - Modeling with others
  - Creating several models in parallel
  - Using an internal wiki (www.twiki.org)
- How much time spent drawing UML before coding?
  - For a three-week timeboxed iteration, spend a few hours or at most one day near the start of the iteration drawing UML for the hard, creative parts of the detailed object design.

Static vs. dynamic modeling

- Dynamic models help design the logic, the behavior of the code or the method bodies
  - Sequence diagram, communication diagram
- Static models help design the definition of packages, class names, attributes, and method signatures
  - UML class diagram

Interaction diagrams

- Generalization of two more specialized UML diagram types
  - Sequence diagrams
  - Communication diagrams
- Illustrate how objects collaborate via messages and methods

Sequence Vs. Communication diagram

- Sequence diagrams illustrate interactions in a kind of fence format
  - Easier to see the call sequence
  - Large set of detailed notation options
- Communication diagrams illustrate interactions in a graph or network format
  - More space-efficient

Common notation: participants – lifeline box

- Metrics (e.g., time, count, total, etc.) are indicated using text within the lifeline box.
Sequence diagram notation

- Lifeline boxes and lifelines
- Messages
  - Synchronous message
  - Found message
- Focus of control and execution specification bars

Illustrating reply or returns
- Using the message syntax `returns = message(parameters);`
- Using a reply message line
  - Message to "self" or "this"

```
public class Register{
  public void doX(){
    …
    clear();
    …
  }
  public void clear(){
    …
  }
}
```

Creation of instance
- In implementation: invoke the `new` operator and call the constructor

Object lifelines and object destruction
- In language that does not have garbage collection

```
makePayment(cashTendered)
```

Diagram frames in UML sequence diagrams
- Support conditional and looping construct

```
A
```

```
B
```

Conditional messages
- `public class Foo{
  Bar bar = new Bar();
  public void m1(){
    if (color.equals("red")) {
      bar.x();
      bar.y();
    }
  }
}
```

Common frame operators

<table>
<thead>
<tr>
<th>Frame operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>Alternative fragment for mutual exclusion conditional logic expressed in the guards</td>
</tr>
<tr>
<td>Loop</td>
<td>Loop fragment while guard is true</td>
</tr>
<tr>
<td>Opt</td>
<td>Optional fragment that executes if guard is true</td>
</tr>
<tr>
<td>Par</td>
<td>Parallel fragments that execute in parallel</td>
</tr>
<tr>
<td>Region</td>
<td>Critical region within which only one thread can run</td>
</tr>
</tbody>
</table>
Sequence diagram notation

- Mutually exclusive conditional messages
  ```java
  public class A {
    B b = new B();
    C c = new C();
    public void doX() {
      if (x < 10)
        b.calculate();
      else
        c.calculate();
    }
  }
  ```

Sequence diagram notation

- Iteration over a collection
  ```java
  public class Sale {
    private List<SaleLineItem> lineItems = new ArrayList<>();
    public Money getTotal() {
      Money total = new Money();
      Money subtotal = null;
      for (SaleLineItem lineItem : lineItems) {
        subtotal = lineItem.getSubtotal();
        total.add(subtotal);
      }
    }
  }
  ```

Sequence diagram notation

- Nesting of Frames

Sequence diagram notation

- Relate interaction diagrams
  - An interaction occurrence (an interaction use) is a reference to an interaction within another interaction.
  - Two related frames
    - A frame around an entire sequence diagram, labeled with the tag `sd` and a name.
    - A frame tagged `ref`, called a reference, that refers to another named sequence diagram;
Sequence diagram notation

- Messages to classes to invoke static methods
  ```java
  public class Foo
  {
  public void doX()
  {
  Locale[] locales = Calendar.getAvailableLocales();
  ..
  }
  }
  ```

- Polymorphic messages and cases
  ```java
  public class Foo
  {
  public void doX()
  {
  ..
  }
  }
  ```

Sequence diagram notation

- Asynchronous and synchronous calls
  - Call does not wait for a response
  - Asynchronous calls are used in multi-thread application
  - Show as a stick arrow message
  - It is common for modeler to use stick arrow to represent synchronous message
  - Do not assume the shape of the arrow is correct

- Links
  - A connection path between two objects
  - Messages flow along link
  - Messages are represented with a message expression and small arrow indicating the direction of the message
  - A sequence number is added to show the sequential order of messages in the current thread of control
  - Many messages may flow along a same single link
collaboration diagram notation (cont)

- Message to "self" or "this"
- Creation of instances
  - Use a message named create for this purpose
  - create message may have parameters
    - Indicates a constructor call with parameters in Java

Basic collaboration diagram notation (cont)

- Conditional messages
  - A conditional message is shown by following a sequence number with a conditional clause in square brackets. The message is only sent if the clause evaluates to true
  - Attach a letter to the sequence number to express mutually exclusive message

Communication diagram notation (cont)

- Iteration or Looping
  - Use a "*" to signify iteration
Communication diagram notation (cont)

- Messages to a Class Object

  - message to class, or a static method call

  ![Diagram](image1)

- Polymorphic messages and cases

  - Polymorphic message

  ![Diagram](image2)

Communication diagram notation (cont)

- Asynchronous and Synchronous calls

  - Asynchronous message

  ![Diagram](image3)