UML Part 2

Ir. I Gede Made Karma, MT

1. The Use Case Model

- Basics of Use Cases
- Relationships between Use Cases
- Other Enhancements

Basics of Use Cases

- Recall the objective is to refine the model so all information is easily available
- The diagram primarily shows the relationship between actors and individual use cases

Basics of Use Cases

- Normally there is no direction to arrows
- However, some author's (Richter) show the initiating actor by using an arrowhead
- Then no arrowhead means a participating actor

Basics of Use Cases

- The supporting documentation shows:
  - description
  - expanded textural specification
  - Exceptions
  - pre and post conditions
- Sometimes the use case author finds themselves writing the same events in more than one use case
- There are two solutions to this problem

Relationships between Use Cases

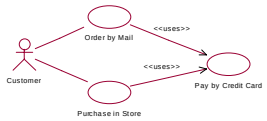
- To avoid repetition one can either use the <<uses>> or <<extends>> relationship
- Originally the primary way to avoid repetition was to invoke the use relationship
- The extends relationship was thought of sort of like inheritance
- However, use cases are not objects !

Relationships between Use Cases

- When a textural specification is repeated, one needs to be able to avoid needless repetition
- For example, if 'Order by Mail' and 'Purchase in Store' both pay by credit card
- Or, if 'Order by Web' and 'Order by Phone' both check if item is in stock

Relationships between Use Cases

- The <<uses>> relationship is the primary technique to invoke another use case
- It is simple in that all it means is two use cases include the same flow of events



Relationships between Use Cases

- The brackets << and >> are a part of the name
- The arrow head is critical as it shows the direction of inclusion

Relationships between Use Cases

The <<uses>> relationship is sometimes called the includes relationship

Relationships between Use Cases

- When a use case uses another use case conditionally it is best to invoke the extends relationship
- For example, if a purchase sometimes has a 'Special Sale Price' a new use case can be created to deal with this
- The use case for 'Special Sale Price' can not stand alone

Relationships between Use Cases

- The use case textural specification begins with an if statement
- For example:
  - If the sale is advertised, then 'Special Sale Price'

Relationships between Use Cases

- The diagram is also modified to show the point which means after step five the use case is invoked
- This is called the extension point

13 03 - Use Case Model 12/4/2007

Relationships between Use Cases

- Note the arrow goes the other way

•This is why some users treat this as inheritance

14 03 - Use Case Model 12/4/2007

Relationships between Use Cases

- Extends Relationship
- Can be thought of all the events in main use case will be executed plus sometimes the are other events added to the use case
- This added feature is why the relationship is thought of as extends
- In other words the extended use case is being invoked

15 03 - Use Case Model 12/4/2007

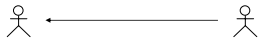
Relationships between Use Cases

- Extends Relationship
- There is a problem in that the case tools may not fully support this extension
- Rational Rose has a limited capacity to write in the bubble thus a note is used
- However, extension is not used much!

16 03 - Use Case Model 12/4/2007

Other Enhancements

- Actors can inherit
- For example an actor can be both a customer and member customer
- This is shown as



Customer Member

17 03 - Use Case Model 12/4/2007

Other Enhancements

- Actors can inherit
- This means that members have more to them then customers
- However there is no way to rigorously specify what is different
- Thus actor inheritance is not used much!

18 03 - Use Case Model 12/4/2007

2. UML Deliverables

- Definitions
- Analysis Deliverables
- Design Deliverables

Definitions

- Kinds of views
- Diagrams that emphasize a particular aspect of the system
- Examples
  - Functional – Basic functional requirements
  - Structural – System components and their relationships
  - Static – Non-temporal or system at rest
  - Dynamic – Temporal or system in motion

Definitions

- Functional view of a system
- Shows the system functionality
- Emphasizes events
- Examples –
  - Use case diagram shows static functionality
  - Activity diagram shows dynamic functionality

Definitions

- Structural view of a system
- Shows the classes and objects
- Emphasizes relationships
- Examples –
  - Class and object diagrams shows static structure
  - Interaction diagrams shows dynamic structure

Definitions

- Static view of a system
- Shows the system at rest
- Emphasizes relationships
- Examples
  - Use case diagram shows static functionality
  - Class and object diagrams shows static structure

Definitions

- Dynamic view of a system
- Shows the system in motion
- Emphasizes the temporal
- Examples:
  - Activity diagram shows dynamic functionality
  - Interaction diagrams shows dynamic structure

Analysis Deliverables

- Static view of a system
- The use case shows:
  - Shows static functionality
  - Shows the system at rest
  - Shows view from outside the system
  - Shows system state before and after use case execution
  - Shows system attributes such a performance and robustness as special requirements (optional)

25 03 - Use Case Model 12/4/2007

Analysis Deliverables

- Dynamic view of a system
- The activity diagram shows:
  - Shows dynamic functionality
  - Shows the system in motion
  - Shows view from outside the system
  - Shows the system as a flowchart

26 03 - Use Case Model 12/4/2007

Design Deliverables

- Static view of a system
- The class and object diagrams show:
  - Shows internal structure
  - Shows object oriented view
  - Shows internals
    - Attributes
    - Operations (methods)
    - Associations

27 03 - Use Case Model 12/4/2007

Design Deliverables

- There are two kinds of diagrams:
  - Class diagram – shows the classes
    - Most often used diagram
    - Some people only do this diagram
  - Object diagram – shows the objects
    - Instance values as well
    - Not used often

28 03 - Use Case Model 12/4/2007

Design Deliverables

- Dynamic view of a system
- The interaction diagram shows:
  - Shows internal structure
  - Shows object oriented view
  - Shows internals
    - Messages
    - Parameters
    - Responses

29 03 - Use Case Model 12/4/2007

Design Deliverables

- There are two kinds of interaction diagrams:
  - Collaboration diagram – spatially orientated
    - Not used in final deliverables
    - Great for sketching the system dynamics
  - Sequence diagram – temporally orientated
    - Easy to follow
    - Used in the final deliverables
- They are easily converted from one form to the other

30 03 - Use Case Model 12/4/2007

Summary

- The points are:
  - Basics of Use Cases
  - Relationships between Use Cases
  - Other Enhancements
- It is best to not complete a use case model
- 80% rule is best, that is complete it about 80%
- Avoid too much decomposition
- Avoid extends, inheritance, and interfaces

31 03 - Use Case Model 12/4/2007

Summary

- In analysis one uses:
  - use case diagrams (static functional) and
  - activity diagrams (dynamic functional)
- In design one uses:
  - class diagrams (static structural) and
  - sequence diagrams (dynamic structural)

32 03 - Use Case Model 12/4/2007