Lecture 9: Use Cases

Kenneth M. Anderson
Object-Oriented Analysis and Design
CSCI 6448 - Spring Semester, 2003

Goals for this Lecture

- Define Use Cases
- Review UML Notation for Use Cases
- Look at a variety of Use Case examples
  - from the books
    - Writing Effective Use Cases
      - by Alistair Cockburn
      - ISBN: 0-201-70225-8
    - Patterns for Effective Use Cases
      - by Steve Adolph and Paul Bramble
      - ISBN 0-201-72184-8

Use Case Terminology

- Use Case Model
  - consists of actors and use cases
- Actors
  - entities which interact with a system
  - Actors are different from users
    - An actor represents a **role** that a user can play
    - Actors are classes; Users are instances
    - Actors are unlike other objects in that their behavior is non-deterministic

Use Case Terminology

- Use Cases
  - An actor can carry out many different operations on the system
    - Each operation or task is a separate use case
  - Use cases participate in relationships with other use cases
    - They might **use or include** another use case
    - They might **extend** another use case
    - They might **generalize or specialize** another use case

Use Cases as Requirements

- The set of use case descriptions specifies the complete functional requirements of a system
- Things to remember
  - Use cases **are** requirements;
  - They are not **all** of the requirements
    - Not good for specifying user interfaces, data formats, business rules, non-functional requirements
  - They are not easy to write!
    - But there are techniques to make your job easier
    - Analogy: A good story is easy to read, but **writing** a good story is hard!

More on Use Cases

- A use case captures a contract between the stakeholders of a system about its behavior
  - The use case is initiated by the primary actor; secondary actors may come into play while the use case is executing
  - Note: actors are not restricted to human beings, other computer systems may serve as secondary actors
- The primary actor is trying to achieve a goal
  - Many things may happen; the goal can be achieved (in more than one way) or the use case may fail (also, in more than one way)
  - A use case captures all of these possible scenarios

More on Use Cases

- Use Cases are primarily a *textual* object
  - We shall review a graphical notation for use cases in a moment; this notation is useful for specifying relationships between use cases and actors
    - It is completely inappropriate, however, for specifying the details of a use case
- Writing good use cases is thus a question of style; some writing styles are more effective than others

Parts of a Use Case

- A use case can be as simple as
  - a paragraph of informal text
- to
  - template-based forms that remind developers what information to include
  - as well as supported by more formal notations
- What to use depends on the ceremony level of the project
  - high ceremony projects will tend towards formal templates
  - mid ceremony projects will use forms with some or all of the recommended fields
  - low ceremony projects will get by with paragraphs of text

Parts of a Use Case

- As recommended by Alistair Cockburn

Primary Actor	Goal in Context
Scope	Level
Stakeholders and Interests	Precondition
Minimal Guarantees	Success Guarantees
Trigger	Main Success Scenario
Extensions	Technology and Data Variations List
Priority	Releases
Response Time	Frequency of Use
Channel to Primary Actor	Secondary Actors
Channels to Secondary Actor	Open Issues

Highlights from Parts List

- Primary Actor
  - Actor that initiated use case
- Goal Level
  - Can be one of “very high summary”, “summary”, “user goal”, “subfunction”, and “too low”
  - Rule of thumb
    - a user goal is one that can be completed in one sitting at a computer
    - a summary goal is one that cannot be completed in one sitting, and may require multiple people, organizations, and systems interacting to achieve the goal

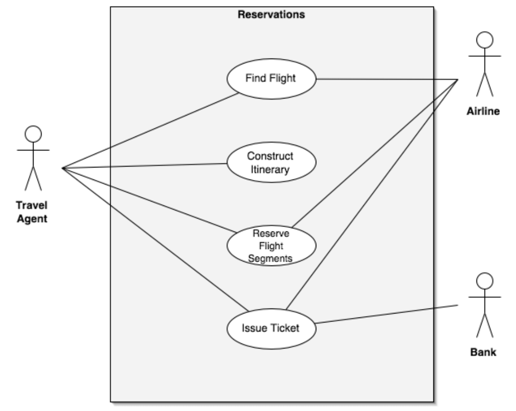
Highlights from Parts List

- Main Success Scenario
  - How is the goal accomplished successfully
- Extensions
  - How might the main success scenario be altered and
    - 1) still succeed
  - or
    - 2) fail

Lets look at some examples…

- From Alistair Cockburn’s book
  - pages 4-6 and 9-11
  - page 18
  - Screen shots of these examples are available on the class website in the “Related Materials” section

Graphical Notation



Adapted From "Patterns for Effective Use Cases" by Adolph and Bramble, © 2003; Page 107

Relationships

- A use case can include another use case within it
  - The included use case is typically referenced by name and underlined in a particular action step
  - The association is stereotyped «include»
  - See pages 191-193 of Adolph and Bramble
    - Also on class website
- Once the included use case is finished, the original use case proceeds as normal

Relationships, continued

- A use case can extend another use case
  - This typically occurs when an extension has gotten to big for a particular use case
  - An extension "interrupts" the base use case when a condition comes true
  - The association is stereotyped «extend»
  - See pages 194-195 of Adolph and Bramble
    - Also on class website
- The extending use case has the option of terminating the original use case; otherwise, the original use case proceeds as normal

Relationships, continued

- Use cases can declare that they can be extended using "extension points"
- See pages 188-189 of Adolph and Bramble for an example of the graphical notation for extension points and how they can be used textually
  - Also on class website

Relationships, continued

- The UML also allows for inheritance relationships on actors and use cases
  - There are a lot of pitfalls associated with this; so be careful
  - Example of proper use and some of the pitfalls are shown on pages 239-241 of Cockburn
    - Also on class website

Two Models of Use Cases

- Cockburn has developed two models for understanding use cases
  - Actors and Goals
  - Stakeholders and Interests
- These models can help clarify how to think about and write use cases

Stakeholders With Interests

- A use case can be viewed as a contract between stakeholders with interests
  - This model identifies what to include in a use case and what to exclude
- Not all stakeholders are present during the operation of the system; when a primary actor interacts with a system, the system must uphold the interests of the "off-stage" actors

Stakeholders/Interests Continued

- Ways to uphold stakeholder interests
  - Gather Information
    - What information do off-stage actors require to understand the actions of the primary actor
  - Running Validation Checks
    - Is the primary actor entering valid information
  - Updating Logs
    - When did the primary actor perform his actions
- Modeling stakeholder interests gives us a rule of thumb: a use cases contains all and only the behaviors related to satisfying stakeholder interests

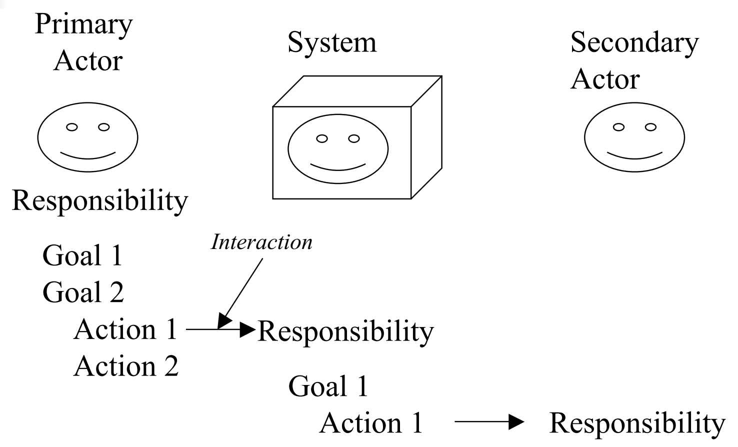
Using the model

- In writing use cases, this model recommends
  - List all Stakeholders
  - Name their interests with respect to the use case
  - State what it means to each stakeholder that the use case completes successfully
  - List what guarantees each stakeholder wants from the system
- Now, we can write actions steps
  - This brings us to the Actors and Goals model

Actors and Goals

- An actor has goals
  - To achieve a goal an actor has to take *actions*
  - Achieving a goal may require accomplishing *sub-goals*
  - Achieving sub-goals may require the support and collaboration of *secondary actors*
  - An action may call upon the *responsibilities* of a secondary actor; this sets up an *interaction* where the calling actor must wait for the secondary actor to achieve the goals associated with that responsibility

Actors and Goals Illustrated



Discussion

- Goals have sub-goals
  - avoid having too many sub-goals however
- Goals can fail
  - We must specify how to respond to failure conditions using extensions
- Actions capture Interactions
  - Writing Action Steps is critical to writing good use cases

Writing Action Steps

- Action Steps are written in one grammatical form
  - a simple action in which one actor either
    - accomplishes a task
    - or passes information to another actor
- Examples
  - User enters name and address
  - At any time, user can request the money back
  - The system verifies that the name and account are current

Action Step Guidelines

- #1: Use Simple Grammar
  - Subject…verb…direct object…prepositional phrase
    - The subject is important, see guideline 2
  - The system…deducts…the amount…from the account
  - Bad writing makes the story hard to follow
  - Complex writing makes it hard to extend an action step
    - e.g. if a step does three things, then if you extend that step, which “thing” does it extend?

Action Step Guidelines

- #2: Show Clearly “Who Has the Ball”
  - For each step, who is performing it?
    - Think of friends kicking a soccer ball
      - You can pass it to yourself
      - You can pass it to a friend
      - You can do something with the ball (e.g. perform a trick)
    - The person with the ball represents the actor
    - The ball represents a message or information being passed between actors
      - You can manipulate the information or pass it on
  - At the end of the step, who has the ball?
    - The answer should always be clear in the writing

Action Step Guidelines

- #3: Write From a Bird’s Eye View
  - Developers tend to write action steps from the system’s perspective rather than a user’s external perspective
    - e.g. “Get ATM Card and PIN” -- bad
    - rather “The customer inserts the card”
    - and “The customer enters the PIN”
  - Alternative Style
    - Customer: Inserts the Card
    - Customer: Enters the PIN

Action Step Guidelines

- #4: Show the Process Moving Forward
  - The amount of progress made in one action step varies according to the level of the use case
    - In a summary use case, each step might satisfy a goal
    - In a subfunction use case, each step may correspond to a computation by the system or data entry by the user
  - If a use case has 17 or more steps, it may indicate that the scope of each step is too small
    - Not "User hits tab key" but "User enters Name"
  - To find a slightly larger scope for a step, ask "Why is the actor doing this?" The answer is probably the scope you are looking for

Action Step Guidelines

- #5: Show the Actor's Intent, Not the Movements
  - Before
    - System asks for name; User enters name
    - System prompts for address; User enters address
    - User clicks "OK"
    - System presents user's profile
  - After
    - User enters name and address
    - System presents user's profile
- Otherwise you end up with longer, brittle, and overconstrained use cases; why?

Action Step Guidelines

- #6: Include a "Reasonable" Set of Actions
  - Ivar Jacobson's notion of a **transaction**
    - Actor sends request and data to system
    - System validates the request and data
    - System alters its internal state
    - System responds to actor with result
  - An action step can contain all four; or start with some in one step and end with the others in the subsequent step
    - See examples in lecture (page 94 of Cockburn)

Action Step Guidelines

- #7: "Validate" Do not "Check Whether"
  - Before
    - The system checks whether the password is correct
    - If it is, the system presents the available actions for the user
  - After
    - The system validates the password is correct
    - The system presents the available actions for the user
  - With "Checks" you always have to say "If true" or "If false" in the next step…not good; with validates you choose the scenario and place the alternative path in the extensions

Action Step Guidelines

- #8: Optionally Mention the Timing
  - Most steps follow directly from the previous one; Occasionally you will need to say something like:
    - At any time between steps 3 and 5, the user will…
    - As soon as the user has …, the system will …
  - Feel free to put in the timing, but only when you need to, usually the timing is obvious

Action Step Guidelines

- #9: Idiom: “User has system A kick System B”
  - Situation: you need your system (A) to fetch information from another system (B)
  - Remember to keep the user in control
    - Not: User clicks Fetch button, at which time the system fetches data from system B (see #5)
    - But: User has the system fetch data from system B
  - Ball is clearly passed from user to A to B; responsibilities are clear; and interface is not specified

Action Step Guidelines

- #10: Idom: “Do Steps x-y until Condition”
  - Situation: need to repeat a set of steps
  - If only one step needs repeating, put the repetition in the step
    - The user selects one or more products
  - If more than one step needs repeating, you can place the repetition before or after the set of steps; Cockburn recommends after in general, but before if the steps can occur in random order
    - See examples next slide

Action Step Guidelines

- Example: After
  1. Customer supplies id or email address
  2. System displays customer’s preferences
  3. User selects an item to buy
  4. System adds item to customer’s “cart”

     Customer repeats steps 3 and 4 until done
  5. Customer purchases the items in the cart

Action Step Guidelines

- Example: Before
  1. Customer logs into system
  2. System presents products and services
     Steps 3-5 can happen in any order
  3. User selects products to buy
  4. User specifies form of payment
  5. User specifies destination address
  6. User finishes shopping
  7. System processes order (of selected products with form of payment and ships to destination address)

The Writing Process

- Cockburn recommends the following process for writing use cases
  1. Name the system scope and boundaries
  2. Brainstorm and list the primary actors
  3. Brainstorm and exhaustively list user goals for the system
  4. Capture the outermost summary use cases to see who really cares
  5. Reconsider and revise the summary use cases. Add, subtract, or merge goals

The Writing Process, continued

1. Select one use case to expand
2. Capture stakeholders and interests, preconditions, and guarantees
3. Write the main success scenario (MSS)
4. Brainstorm and exhaustively list the extension conditions
5. Write the extension-handling steps
6. Extract complex flows to sub use cases; merge trivial sub use cases
7. Readjust the set: add, subtract, merge, as needed