REQUIREMENTS & USE CASES

CS-463
Object Oriented Analysis & Design
Session 2

Agenda
- Requirements
- Use Case Analysis - Basic definitions
- Developing a Use Case model
- Writing Use Case - guidelines
- Use case Benefits, Drawbacks
- Exercise.

Understanding Requirements
- “Capabilities and conditions to which the system and project must conform” [Jacobson et al., 1999]
- Challenges: {find, communicate, record, manage} the requirements
- Requirements always change, so effective management is critical

Factors that Challenge Projects

The FURPS+ Model
- Functional – features, capabilities, security
- Usability – human factors, help, documentation
- Reliability – failure frequency, recoverability
- Performance – response times, throughput, accuracy, availability, resource utilization
- Supportability – adaptability, maintainability, internationalization, configurability

[Standish 94; Larman, 2002]
The FURPS+ Model [2]

- Implementation – resource limitations, languages/tools, hardware
- Interface – with legacy systems
- Operations – sysop management
- Packaging – delivery, installation
- Legal – licensing, etc.

Use FURPS+ as a global checklist when identifying requirements for a system you are designing.

Other Terminology

- Quality Attributes, or “-ilities”
  - Usability, reliability, supportability, performance (non-functional)
- Functional vs. Non-Functional
  - Behavioral features vs. everything else

Documenting Requirements

- Primarily, in the use case model (functional requirements)
- Also, Supplementary Specifications (other requirements)

Choosing an appropriate architecture often depends on a clear understanding of non-functional requirements.

Use Case Analysis

- Use Case (Stories of using the system) is a technique to understand and record functional requirements

  - The idea was first introduced by Ivar Jacobson in 1986.

Use Case Analysis...

- Goals & Stories
  - Customers and end users have goals
  - Expectations from the system
  - The “What” of our system.
  - Ideas

Use Case Analysis...

- Basic definitions

  - What is a Use Case?
    - A sequence of actions a system performs that yields a valuable result for a particular actor.
    - E.g. register for a course
    - Process Sale

  - What is an Actor?
    - A user or outside system that interacts with the system being designed in order to obtain some value from that interaction
    - An ACTOR is anything that exchanges data with the system. An actor can be a user, external hardware, or another system.
    - e.g. Student, Clerk
**Use Case Analysis**

**Basic definitions**

- **What is a Scenario?**
  Specific sequence of actions and interactions between actor(s) and system (= one story; success or failure). Also known as an instance of a use case.
- **Use Case** is a collection of related success and failure scenarios describing the actors attempts to support a specific goal.
- The RUP defines the use cases with in the requirements discipline.

**Use Case Analysis**

- Use Cases Describe Function not Form
- Use Cases describe **WHAT** the system will do, but never describe **HOW** it will be done.
- Use Cases are Analysis Products, not Design Products.

**Use Case Model**

- Individual use case describes how a particular actor interacts with the system to achieve a result of value to the specific actor.
- The set of **all use cases** together describes the complete behavior of the system.
- The complete set of use cases, actors, and their interactions constitutes the use-case model for the system.

**Use Case Model**

- **EXAMPLE SYSTEM**
- Develop a use case model for the ATM systems used by banks

**Building the Use-Case Model**

**A Simple Recipe**

- Identify System Boundary (High level Scoping)
- **Step 1:** Identify and Describe the Actors
- **Step 2:** Identify Use cases and write Brief Descriptions
- **Step 3:** Identify the Actor and Use-Case Relationships
- **Step 4:** Outline the Individual Use Cases
- **Step 5:** Refine the Use Cases

**Building the Use-Case Model**

**Step 1: Identify and Describe the Actors**

- Identify who is going to be using the system directly - e.g., hitting keys on the keyboard or automatically interfaces with the system. These are the Actors. A general guideline to find actors is given below.
  - Who uses the system?
  - Who gets info/output from the system?
  - Who provides info/input to the system?
  - Where in the company is the system used?
  - Who supports the and maintain the system?
  - What other systems use this system?
  - Who evaluates system logs, performance?
  - Who does system administration?
  - Is Time an actor?
  - Who starts & stops the system?
Building the Use-Case Model...

Step 1: Identify and Describe the Actors...

- ATM Example - Potential Actors
  - Customer
  - Administrator
  - Bank Software

Building the Use-Case Model...

Step 2: Identify the Use Cases and Write a Brief Description

- Pick one of the Actors, identified in the previous step and see what that Actor wants to do with the system (Goal of the actor)
- Raise the level of goal to an appropriate point and then each of these goals will become a Use Case
- Use EBP Guide line (or the Boss test ☺)

Building the Use-Case Model...

Step 2: Identify the Use Cases and Write a Brief Description...

- To identify the use cases, ask the following for each actor:
  - What will the actor use the system for?
  - Will the actor create, store, change, remove, or read data in the system?
  - Will the actor need to inform the system about external events or changes?
  - Will the actor need to be informed about certain occurrences in the system?

Building the Use-Case Model...

Step 2: Identify the Use Cases and Write a Brief Description...

- Some Guidelines
  - Make an actor goal list
  - Do event analysis
  - Name the use case similar to the user goal
  - The name of the use case should start with a verb.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cashier</td>
<td>Process sales, Process rentals, Handle returns...</td>
</tr>
<tr>
<td>Manager</td>
<td>Start up, Shut down</td>
</tr>
<tr>
<td>System Admin</td>
<td>Add user, Modify user</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>External Event</th>
<th>From Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter sale line item</td>
<td>Cashier</td>
<td>Process Sale</td>
</tr>
</tbody>
</table>

Building the Use-Case Model...

Step 2: Identify Use Cases and Write Brief Descriptions...

- Typically, the use-case name is a few words or a short phrase that starts with an action verb and communicates what the actor achieves with the use case.

- Repeat Step 2 for all primary actors to identify a reasonable number of use cases and then do the refining (step 3,4,5)
Building the Use-Case Model…

Step 1: Identify Use Cases and Write Brief Descriptions.

- ATM - Example
  - Customer
    - Withdraw cash
    - Check Balance
    - Transfer funds
    - Deposit
  - Administrator
    - Startup, shut down, Insert Cash

Step 2: Identify Use Cases and Write Brief Descriptions.

- Only one actor can initiate a use case
- However, many actors can be involved in a use case.
- Each use case is analyzed to see what actors interact with it and
- Each actor’s behavior is reviewed to make sure that all of the results he needs to see are achieved by the system.
- Diagrams can be used as an aid.

Step 3: Identify Actors and Use-Case Relationships

- To understand the system more think about the flow of events (basic and alternatives) for each use case.
  - Basic flow: the most common path from start to finish through the system
  - Alternative flows: other possible paths based on regular or exceptional circumstances.

Step 4: Outline the Individual Use Cases

- To do that ask the following questions:
  - Basic flow:
    - What events start the use case?
    - How does the use case end?
    - How does the use case repeat some behavior?
  - Alternative Flows:
    - Are there optional situations in the use case?
    - What odd cases might happen?
    - What variants might happen?
    - What may go wrong? etc

Step 5: Refine the Use Cases

- When to refine and think about the next level of details?
- Depends on
  - All alternative flows including exception conditions
  - Pre- and post-conditions

[Use Case Diagram] [Larman, 2002]
USE CASE- Guidelines

- What is a proper, valid use case? Granularity level
- For Course registration application which one of the following is a proper use case
  - Verify password
  - log in
  - Add a course

USE CASE- Guidelines...

- EBP Guideline
  - For requirement analysis for a computer application focus on use case at the elementary business process (EBP)
  
  *Now what does this French means?*

USE CASE- Guidelines...

- A common use case mistake is defining use cases at too low level or sub function level
- An EBP level use case is known as a user goal level use case. i.e. user wants to do something that adds business value.
- It is not necessary that 100% of your use cases will follow the EBP guideline, but a predominant portion of your use cases will

USE CASE- Guidelines...

- Use Cases & Goals
  - Actors has goals (record sale, add course, issue book) and they use computer application to achieve that.
  - EBP is therefore a user level goal
  - While figuring out use cases try to raise the level of a goal to a user level goal.

USE CASE- Guidelines...

- Raising the Goal level - Course registration system...
  - Analyst: what u want to do with the system
  - PCO: log in etc. (a sub function)
  - Analyst: why would you want to login ?
    - (raising the level of goal)
  - PCO: to add course, find student's GPA
    - (That's what user want to do)
  - Analyst: why would you want to do that ?
  - PCO: To run the students registration process smoothly
    - (More of an enterprise level goal and not a user level goal)
USE CASE Guidelines

Reasonable EBP Violations...

- If there are too many use cases. Sometimes the CRUD (create, retrieve, update, delete) cases are combined to form one use case (Manage User, Maintain Engine). Although they represent four different user goals.
- Sometimes a sub function is repeated in several base use cases.
  - Log in, Pay by credit

Though the user goal or business process is somewhat higher (login to add course, recording a sale and payment is by credit) yet the common sub function is often separated and shown as a separate use case (will be discussed in usecase relationships).

USE CASE Guidelines...

Use Cases Types & Formats

- Black Box
  - Treats the system as a black box
  - No information about the internal working of the system
  - Focus on what and not how
  - E.g.
    - The system records lending of the book (best)
    - The system saves the data of the book lender into a file or database (bad)
    - The system issues the SQL insert statement. (worst)

USE CASE Guidelines...

Essential Vs. Concrete (Real) Use Cases

- Essential use cases defer the details of the UI and focus on the intentions of the actors, and responsibilities of the system. While Concrete (AKA Real) do not.
  - Essential: “The AccountHolder identifies themselves to the ATM”
  - Real: “The AccountHolder inserts their card in the reader. Window A is displayed. They enter their PIN on the numeric keypad, ...”
- As we move from analysis to design, we are more inclined to move from essential to concrete use case descriptions.
- What we do practically in our industry and why?

USE CASE Guidelines...

Use Case Description Level

- Essential - A general description of the business process. Do not include technology information. Use the 100 year rule where the information would be understood 100 years in the past and the future.
- Real - Design oriented, shows reports, examples. Uses technological descriptions. Real use cases are undesirable during analysis and should only be used during analysis for specific reasons. Real use cases are handy for requirements gathering.

USE CASE Guidelines...

Use Case Formality Style

- Brief
  - One paragraph description, usually of a happy day scenario
- Casual
  - Multiple paragraphs describing success as well as alternate course of action
- Fully dressed
  - Two column or a single column format with everything explained

Example (brief format)

Process Sale: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running Total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a Receipt from the system and then leaves with the items.

Usually, writing the stories is more important than diagramming a use case model in UML.
Example (casual format)

Handle Returns

Main Success Scenario: A customer arrives at a checkout with items to return. The cashier uses the POS system to record each returned item...

Alternate Scenarios:
If they paid by credit, and the reimbursement transaction to their credit account is rejected, inform the customer and pay them with cash.
If the item identifier is not found in the system, notify the cashier and suggest manual entry of the identifier code
If the system detects failure to communicate with the external accounting system... (etc.)

Example of a fully dressed use case

Stakeholders and Interests
- Important: defines what the use case covers (“all and only that which satisfies the stakeholders’ interests”)
- Preconditions
  - What must always be true before beginning a scenario in the use case (e.g., “user is already logged in”)

Success Guarantees (Postconditions)
- What must be true on successful completion of the use case; should meet needs of all stakeholders

Basic Flow
- Records steps: interactions between actors; a validation (by system); a state change (by system)

Extensions (Alternative Flows)
- Scenario branches (success/failure)
- Longer/more complex than basic flow
- Branches indicated by letter following basic flow step number, e.g. “3a”
- Two parts: condition, handling

Special Requirements
- Non-functional considerations (e.g., performance expectations)

Technology & Data Variations
- Non-functional constraints expressed by the stakeholders (e.g., “must support a card reader”)
USE CASE - Guidelines...

Alternate Actor Notation

Some UML alternatives to lifetime external access that are superior to composite contexts:
- The class box style can be used for graphical contexts of a state machine. Using the compound actor's viewpoint conceptual convention.

Alternate Actor Notation

USE CASE - Guidelines...

Feature Lists: “the system shall...”
- More traditional approach
- Less preferable for user-oriented s/w
- Only mention function, not flow
- If flows are written elsewhere, then redundant = hard to maintain in parallel
- High-level feature lists are okay
- Include in Vision document
- "executive summary of use cases"

Requirements Effort in Early Iterations

[ Larman, 2002 ]

UP Artifacts and Timing

[ Larman, 2002 ]

Use Case Refinement Process

[ Larman, 2002 ]
Use Case Model serve as an Input/bases for
- for the cost and effort estimation. (UCP)
- the delivery contract in a particular release. (RUP is Use Case Driven)
- the functional testing process and UAT, helping to assure that the system actually does what it was intended to do
- a reference point to trace back the requirements, hence make it difficult for requirements to fall through the cracks
- the user documentation, conveniently organized in a step-by-step format.

Benefits of Use Cases
- Use cases are the primary vehicle for requirements capture in RUP
- Use cases are described using the language of the customer (language of the domain which is defined in the glossary) therefore act as an easily-understood communication mechanism
- Use cases provide a concise summary of what the system should do at an abstract (low modification cost) level.

Benefits of Use Cases
- They are relatively easy to write and easier to read.
- They force developers to think through the working of a system from the perspective of a user.
- They engage the users in the requirements process:
  - helping them understand the system that is being proposed.
  - giving them a way to communicate and document their needs.
- They give context for the requirements of the system:
  - One can understand why a requirement is what it is
  - as well as how the system meets its objectives.

The Benefits of Use Cases
- They provide an ordering mechanism for requirements:
  - one can tell what has to happen before the next thing happens, and so on.
- In most circumstances, development team is also involved in writing use cases. That means not only that they actually understand requirements better but also that the developers know they are responsible for determining them.
- It is a critical tool in the analysis process, helping us understand what the system needs to do and how it might go about doing it.

Difficulties with Use Cases
- As functional decompositions, it is often difficult to make the transition from functional description to object description to class design
- Reuse at the class level can be hindered by each developer “taking a Use Case and running with it”. Since UCs do not talk about classes, developers often wind up in a vacuum during object analysis, and can often wind up doing things their own way, making reuse difficult
- Use Cases make stating non-functional requirements difficult (where do you say that X must execute at Y/sec?)
- Testing functionality is straightforward, but unit testing the particular implementations and non-functional requirements is not obvious

KEY POINTS
- use cases are not diagrams, they are text. Focusing on secondary-value UML use case diagrams rather than the important use case text is a common mistake for use case novices.
- Write usecases at the user goal level. Use Cases describe WHAT the system will do, but never describe HOW it will be done. Use Cases are Analysis Products, not Design Products.
- In RUP
  - Identify most of the use cases in inception
  - For 10/20% high risk or high priority use cases write fully dressed versions
  - For the remaining majority write brief versions
  - Define the majority chunk of remaining use cases during elaboration.
Questions and Discussion

Assignments
- Reading assignment
  - Chapter 1-6 of Craig Larman, “Applying UML with Patterns” with great emphasize on chap. 6
- Written assignment
  - Identify the use cases for the given case study

In class Exercise
- Library Software project
  - Scoping of system
  - Identification of use cases & development of use case model
  - Effort Estimation using use cases
  - Functional Testing using use cases.