# Software Engineering I CSC-382


**Lecture 6**

# Software Engineering I CS-382

- Lecture 6
- What we will cover: (Requirements Engineering)
  - More on Use Cases

## Recall Use-Cases

"[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)." Ivar Jacobson

- **Actors** represent roles people or devices play as the system functions
- Actors have two common features:
  1. External to the application
  2. They take initiative, stimulate and interact with our system

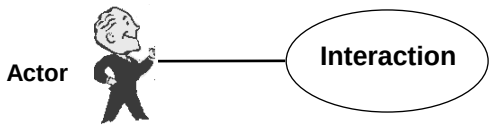


## Recall Use-Cases II

- Three types of actors:
  1. Users
  2. Administrators
  3. External Programs and Devices

## Recall the Two Formats of the Use-Case

•**Use Case Diagram:**




•**The Use Case Text Description:**

•**Define using the Use Case descriptions:**

| | |
|---|---|
| **Use Case Name:** | **Primary Actor:** |
| **Data:** | |
| **Stimulus:** | |
| **Response:** | |
| **Comments:** | |



## Recall Some Guidelines to Writing Use Cases

1. Use Simple Grammar
   - Subject…verb…direct object…prepositional phrase
   - E.g. "the system deducts amount from the user account balance
2. Show Clearly 'Who Has the Ball'
   - Picture a ping-pong game with the action going back and forth
   - The ball is the message or data that is being passed around the system

**Cockburn**

## Recall Some More Guidelines to Writing Use Cases II

3. Write from a 'Bird's Eye View'
   - Rather than write get ATM card and Pin…
   - The customer inserts the ATM card and enters PIN
4. Show the Process Moving Forward
   - Amount of progress relates to how abstract we are making the Use Case
   - Initially in the design we should stay at a high level to keep the length of the Use Case manageable (I like no more than 10 steps)



## Recall Some More Guidelines to Writing Use Cases III

5. Show the Actors Intent not the Movements
   - Do not describe the interface details
   - Stay at the level of stating what data to enter, rather than specifics such as "user hits enter"..

## Recall Some More Guidelines to Writing Use Cases IV

6. Include a 'Reasonable' Set of Actions
   - Jacobson views each step Use Cases as a Transaction, which entails:
     1. Actor sends Request
     2. System validates Request
     3. System alters its internal state
     4. System responds to actor with result
   - This is a lot to put on 1 line (hard to read), but remembering these sub-steps allows us to write better Use Cases (we won't miss steps)



## Example Template for Use-Cases

**Use Case Name:** **Primary Actor:**

**Goal in Context:**

**Pre-conditions:**

**Trigger:**

**Scenario:**

**Exceptions:**

**Priority:** **Availability:** **Frequency of Use:**

**Channel to Actor:** **Secondary Actors:** **Channels to 2nd Actors:**

**Open Issues:**

- **Note: You do not need to fill all of the fields out at once.**
- **Choose the critical ones and the others can be ignored or filled out as the requirements analysis continues.**

# Simpler Template for Use-Cases

**Use Case Name:** **Primary Actor:**

**Data:**

**Stimulus:**

**Response:**

**Comments:**

*From Sommerville



# Even Simpler Template for Use-Cases

**Use Case Name:**

**Primary Actor:**

**Description of Usage:**

**Feel free to customize your own !**

# Example Template for Use-Cases

**Use Case Name: Process Sale**

**Primary Actor: Cashier**

**Description of Usage:**

**Main Success Scenario (or Basic Flow):**

1. Customer arrives at POS checkout with goods and/or services to purchase.
2. Cashier starts a new sale.
3. Cashier enters item identifier.
4. System records sale line item and presents item description, price, and running total. Price calculated from a set of price rules.

*Cashier repeats steps 3-4 until indicates done.*

5. System presents total with taxes calculated.
6. Cashier tells Customer the total, and asks for payment.
7. Customer pays and System handles payment.
8. System logs completed sale and sends sale and payment information to the external Accounting system (for accounting and commissions) and Inventory system (to update inventory).
9. System presents receipt.
10. Customer leaves with receipt and goods (if any).



# One Process for Filling in Use Cases

1. Brainstorm list of all the actors and their goals
   - This provides the highest level of precision
2. For each Use Case sketch the main success scenario
   - Defines the steps in the actor-system interaction to reach the goal
   - It is the next finer level of detail

## One Process for Filling in Use Cases II

3. Brainstorm on all possible failure conditions
   - Generate as many as possible prior to thinking how to handle them

4. Describe how the system will handle the failure conditions
   - This part of the Use Case often shows the most interesting aspects of the system
   - These are called the Extensions (mentioned earlier) so they are a little more advanced.
     - Lets focus on the primary paths first



## Start with Actors and Work Down in Detail

| | | | | |
|---|---|---|---|---|
| Actor | Goal | Success Action | Failure Condition | Recovery Action |
| | | | Failure Condition | Recovery Action |
| | | Success Action | Failure Condition | Recovery Action |
| | | | Failure Condition | Recovery Action |
| | Goal | Success Action | Failure Condition | Recovery Action |
| | | | Failure Condition | Recovery Action |
| | | Success Action | Failure Condition | Recovery Action |
| | | | Failure Condition | Recovery Action |

Cockburn

## Deriving the Hierarchy of Use Cases

- Each Use Case describes a goal the actor wants to reach
  - The Use Case name is this primary goal
- Each Use Case collects all the scenarios that result in either success or failure towards reaching the goal



## Deriving the Hierarchy of Use Cases II

- The Use Case scenario describes a list of steps that will lead to satisfying that goal
  - Each step can be viewed as a sub-goal of the actors/sub-actors defined on that line
  - Some of these steps can then be expanded into your next finer detail of Use Cases

## Actions in the Use Case

- The system must satisfy the goal of the Actor through its Actions
- There are three types of actions defined in a Use Case
  1. Interaction between two actors (info passed back and forth)
  2. Validation to protect the stakeholders (e.g. validate password)
  3. A n internal state change of the system



## Actions in the Use Case II

- The actions are initiated thru one of three triggers:
  1. Primary actor initiates the Use Case
  2. Primary actor uses an intermediary to initiate interaction
  3. Time or state-based initiation

## Actors in the Use Case

- Recall there are two types of Actors: Primary and Secondary
  - Secondary actors are there to provide a service to the system to allow it to help the primary actor reach her goal
  - These can be things like printers, etc.



## Actors in the Use Case II

- Sometimes it is helpful to characterize the primary actors
  - The actors define a category of person that will be interacting with the system
  - Helpful to create a User Profile Table

| Actor Name | Profile: Background and Skills |
|---|---|
| Customer | Person off street, no computer experience,… |
| Accountant | Computer savvy, wants to customize system,… |

## Describing the Actions at the Right Level

- Common to add too much detail into the action sequence in the Use Case
- Hints for managing this detail are:
  - Try not to keep number of steps between 3 and 10
  - What to do if you have too many steps?



## If You Have Too Many Steps Ask

- What does the actor really want from the system?
- Why is the actor going thru these actions ?
- These questions can help you merge actions together
  - E.g. replace:
    - User enters login screen
    - User enters ID
    - User enters password
  - With:
    - System authenticates user

## Recap: Nice Recipe for Generating Use Cases

1. Define the boundaries of the system (Context Diagram)
2. Brainstorm and list the primary actors
3. Brainstorm and list the primary actors' goals
4. Write the top most Use Cases for the system
5. Write the main success scenarios for each Use Case



## Recap: Nice Recipe for Generating Use Cases II

6. Brainstorm on failure conditions and alternate success paths
7. Describe these failure scenarios (in Extension section or maybe separate Use Cases if complicated)
8. Edit all use cases (expand, combine, break apart,…)

## Graphical versus Text Use Cases

- Clearly text Use Cases can hold significantly more information
  - Can be harder to derive the big picture from them



## Graphical versus Text Use Cases II

- Graphical Use Cases are good for providing this higher level
  - Use them as a Table of Contents or Index for all the text Use Cases
  - Can use them as a Context Diagram for the system (Note they define the various external actors just like a context diagram
  - Are good for capturing Big Picture to show how all the other Use Cases relate to each other.

# In-class Requirements Engineering Exercise: SafeHome



# Recall: SafeHome from Users Perspective

*How a Project Starts*

**The scene:** Meeting room at CPI Corporation, a (fictional) company that makes consumer products for home and commercial use.

**The players:** Mal Golden, senior manager, product development; Lisa Perez, marketing manager; Lee Warren, engineering manager; Joe Camalleri, executive VP, business development.

**The conversation:**

**Joe:** Okay, Lee, what's this I hear about your folks developing a what? A generic universal wireless box?

**Lee:** It's pretty cool, about the size of a small matchbook. We can attach it to sensors of all kinds, a digital camera, just about anything. Using the 802.11b wireless protocol. It allows us to access the device's output without wires. We think it'll lead to a whole new generation of products.

**Joe:** You agree, Mal?

**Mal:** I do. In fact, with sales as flat as they've been this year, we need something new. Lisa and I have been doing a little market research, and we think we've got a line of products that could be big.

**Joe:** How big. . . , bottom-line big?

**Mal: (avoiding a direct commitment):** Tell him about our idea, Lisa.

**Lisa:** It's a whole new generation of what we call "home management products." We call 'em *SafeHome*. They use the new wireless interface, provide homeowners or small business people with a system that's controlled by their PC—home security, home surveillance, appliance and device control. You know, turn down the home air conditioner while you're driving home, that sort of thing.

**Lee: (jumping in)** Engineering's done a technical feasibility study of this idea, Joe. It's doable at low manufacturing cost. Most hardware is off the shelf. Software is an issue, but it's nothing that we can't do.

**Joe:** Interesting. Now, I asked about the bottom line.

**Mal:** PCs have penetrated 60 percent of all households in the USA. If we could price this thing right, it could be a killer-App. Nobody else has our wireless box—it's proprietary. We'll have a two-year jump on the competition. Revenue? Maybe as much as $30–40 million in the second year.

**Joe (smiling):** Let's take this to the next level. I'm interested.

# Can we Find Some Actors in This ?

**Recall only going to focus on home security first**

Home security functions:

- Standard window/door/motion sensor monitoring for unauthorized access (break-ins).
- Monitoring for fire, smoke, and CO levels.
- Monitoring for water levels in basement (e.g., flood or broken water heater).
- Monitoring for outside movement.
- Change security setting via the Internet.



# Can we Find Some Actors in This ?

Home surveillance functions:

- Connect to one or more video cameras placed inside/outside house.
- Control pan/zoom for cameras.
- Define camera monitoring zones.
- Display camera views on PC.
- Access camera views via the Internet.
- Selectively record camera output digitally.
- Replay camera output.

# In-class Text Use-Cases

**Use Case Name:**

**Primary Actor:**

**Description of Usage:**



# In-class Text Use-Cases

**Use Case Name:** **Primary Actor:**

**Data:**

**Stimulus:**

**Response:**

**Comments:**

# For Next Class

- Continue to Study Chapter 7 Sections 7.6-7.9 in Pressman (Requirements Engineering)