

# Use Cases and Scenarios

05-Use-Cases 1

## We Will Cover

- What is a use-case
  - Use-case versus user interaction
- Use-Case diagrams
  - The constructs in the use-case diagrams
- Capturing the use-case
  - High-level use-case
  - Extended use-case
  - Difference between use case and scenario

05-Use-Cases 2

## What is a Use-Case

- A use-case captures some user visible function
- This may be a large or small function
  - Depends on the level of detail in your modeling effort
- A use-case achieves a discrete goal for the user
- Examples
  - Format a document
  - Request an elevator
- How are the use cases found (captured or elicited)?

05-Use-Cases 3

## User Goals versus User Interactions

- Consider the following when formatting a document
  - ◆ Define a style
  - ◆ Change a style
  - ◆ Copy a style from one document to the next
  - versus
    - ◆ Format a document
    - ◆ Ensure consistent formatting of two documents
- The latter is a user goal
  - Something the user wants to achieve
- The former are user interactions
  - Things the user does to the system to achieve the goal

05-Use-Cases 4

## Goals and Interactions

- There is a place for both goals and interactions
- Understand what the system shall do
  - Capture the user goals
- Understand how the user will achieve the goals
  - Capture user interactions
  - Sequences of user interactions
- Thus, start with the user goals and then refine the user goals into several (many) user interactions

05-Use-Cases 5

## Use-Case Diagrams (POST)

POST: Point of Sale Terminal

Adapted from Larman "Applying UML and Patterns"

05-Use-Cases 6

## Another Example

### Financial Trading System

Adapted from Fowler "UML, Distilled"

05-Use-Cases 7

## Includes and Extends

- Includes
  - You have a piece of behavior that is similar across many use cases
  - Break this out as a separate use-case and let the other ones "include" it
  - Examples include
    - Valuation
    - Validate user interaction
    - Sanity check on sensor inputs
    - Check for proper authorization
- Extends
  - A use-case is similar to another one but does a little bit more
  - Put the normal behavior in one use-case and the exceptional behavior somewhere else
    - Capture the normal behavior
    - Try to figure out what can go wrong in each step
    - Capture the exceptional cases in separate use-cases
  - Makes it a **lot** easier to understand

05-Use-Cases 8

## Includes

- You have a piece of behavior that is similar across many use cases
- Break this out as a separate use-case and let the other ones "include" it
- Examples include
  - Valuation
  - Validate user interaction
  - Sanity check on sensor inputs
  - Check for proper authorization

05-Use-Cases 9

## Extends

- A use-case is similar to another one but does a little bit more
- Put the normal behavior in one use-case and the exceptional behavior somewhere else
  - Capture the normal behavior
  - Try to figure out what can go wrong in each step
  - Capture the exceptional cases in separate use-cases
- Makes it a lot easier to understand

05-Use-Cases 10

## Setting the System Boundary

- The system boundary will affect your actors and use-cases

Adapted from Larman "Applying UML and Patterns"

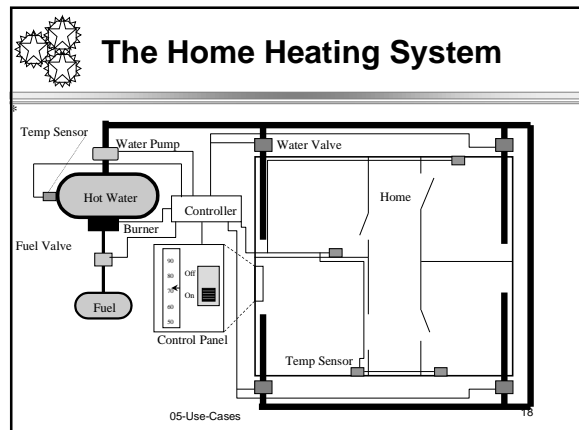
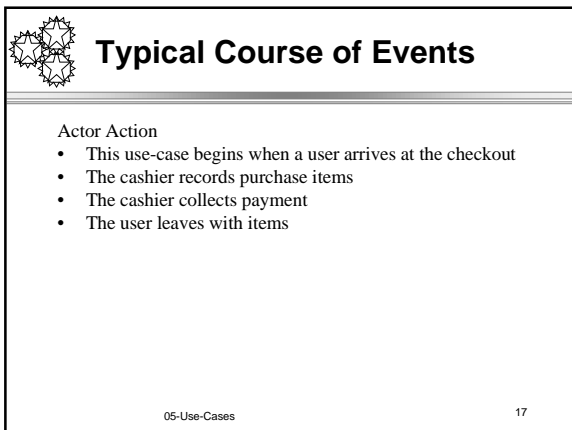
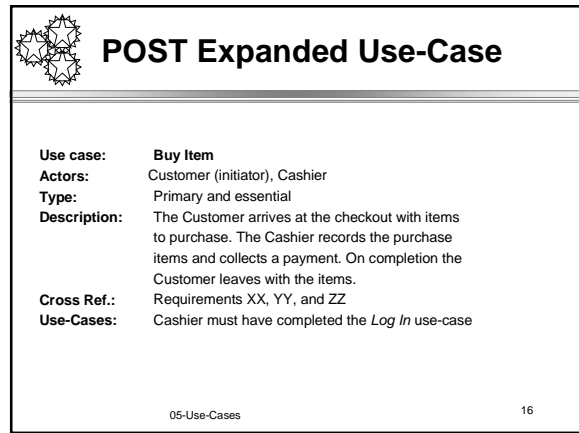
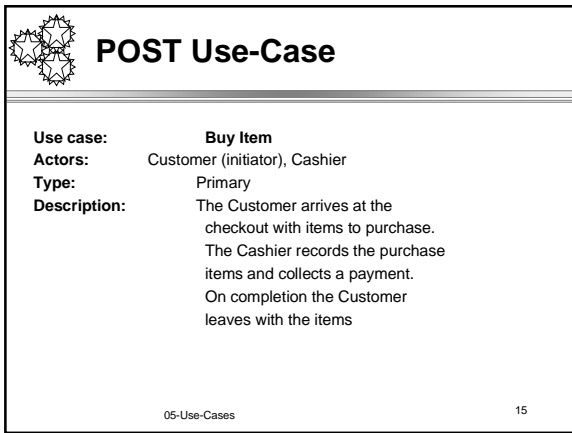
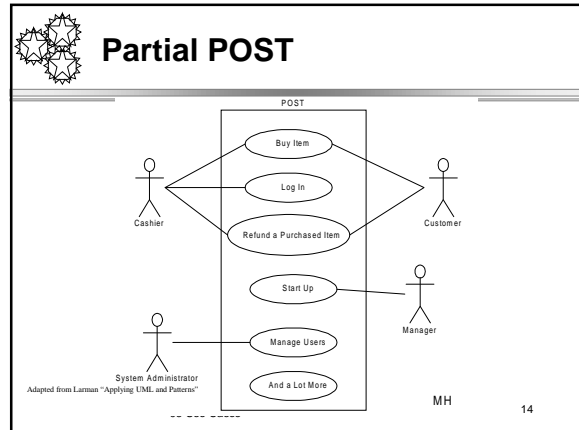
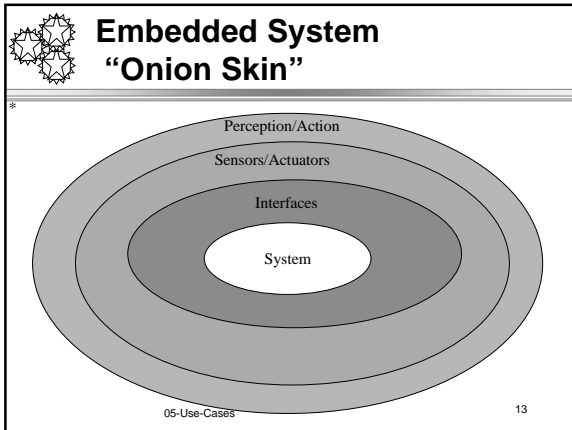
05-Use-Cases

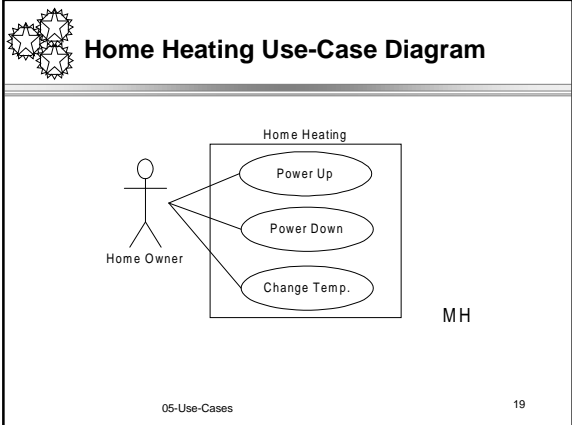
## A Different Boundary

- Let us view the whole store as our system

Adapted from Larman "Applying UML and Patterns"

05-Use-Cases 12

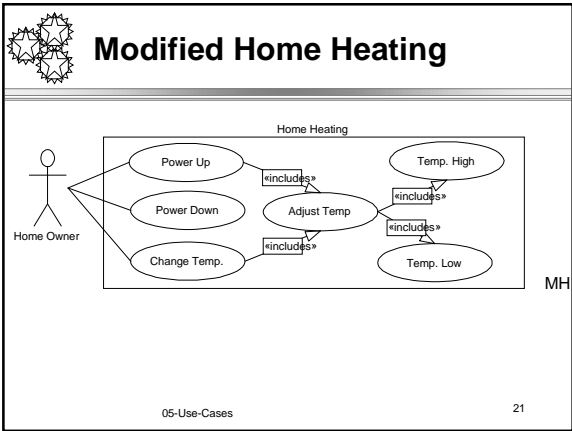




### Home Heating Use-Cases

**Use case:** Power Up  
**Actors:** Home Owner (initiator)  
**Type:** Primary and essential  
**Description:** The Home Owner turns the power on. Each room is temperature checked. If a room is below the desired temperature the valve for the room is opened, the water pump started. If the water temp falls below threshold, the fuel valve is opened, and the burner ignited. If the temperature in all rooms is above the desired temperature, no actions are taken.  
**Cross Ref.:** Requirements XX, YY, and ZZ  
**Use-Cases:** None

05-Use-Cases 20



### Modified: Home Heating Use-Cases

\*  
**Use case:** Power Up  
**Actors:** Home Owner (initiator)  
**Type:** Primary and essential  
**Description:** The Home Owner turns the power on.  
**Perform Adjust Temp.** If the temperature in all rooms is above the desired temperature, no actions are taken.  
**Cross Ref.:** Requirements XX, YY, and ZZ  
**Use-Cases:** Perform Adjust Temp

05-Use-Cases 22

### Modified: Home Heating Use-Cases

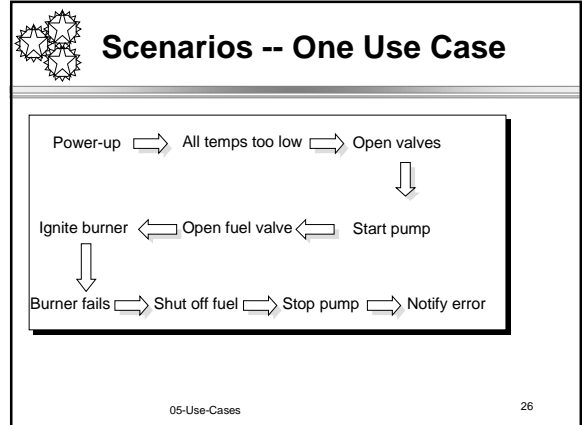
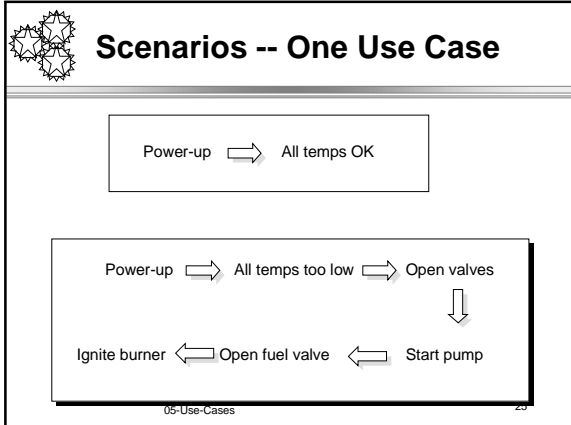
\*  
**Use case:** Adjust Temp  
**Actors:** System (initiator)  
**Type:** Secondary and essential  
**Description:** Check the temperature in each room. For each room:  
 Below target: **Perform Temp Low**  
 Above target: **Perform Temp High**  
**Cross Ref.:** Requirements XX, YY, and ZZ  
**Use-Cases:** Temp Low, Temp High

05-Use-Cases 23

### Modified: Home Heating Use-Cases

\*  
**Use case:** Temp Low  
**Actors:** System (initiator)  
**Type:** Secondary and essential  
**Description:** Open room valve, start pump if not started.  
 If water temp falls below threshold, open fuel valve and ignite burner.  
**Cross Ref.:** Requirements XX, YY, and ZZ  
**Use-Cases:** None

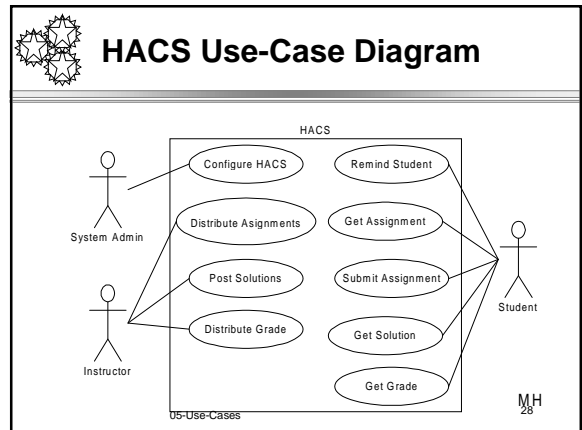
05-Use-Cases 24



## HACS

- Homework assignment and collection are an integral part of any educational system. Today, this task is performed manually. What we want the homework assignment distribution and collection system (HACS for short) to do is to automate this process.
- HACS will be used by the instructor to distribute the homework assignments, review the students' solutions, distribute suggested solution, and distribute student grades on each assignment.
- HACS shall also help the students by automatically distributing the assignments to the students, provide a facility where the students can submit their solutions, remind the students when an assignment is almost due, remind the students when an assignment is overdue.

05-Use-Cases 27



## HACS Use-Cases

**Use case:** **Distribute Assignments**

**Actors:** Instructor (initiator)

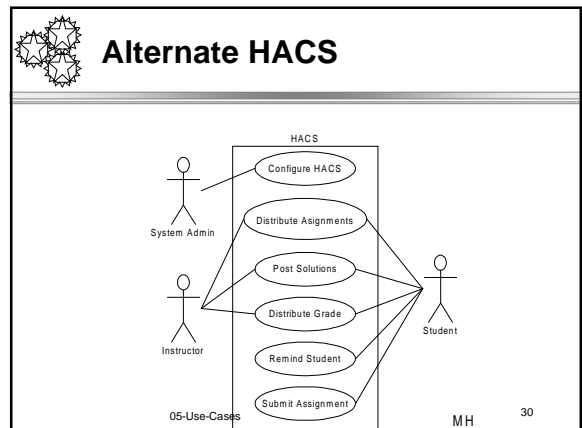
**Type:** Primary and essential

**Description:** The Instructor completes an assignment and submits it to the system. The instructor will also submit the due date and the class the assignment is assigned for.

**Cross Ref.:** Requirements XX, YY, and ZZ

**Use-Cases:** *Configure HACS* must be done before any user (Instructor or Student) can use HACS

05-Use-Cases 29





## Alternate HACS Use-Cases

**Use case:** Distribute Assignments  
**Actors:** Instructor (initiator), Student  
**Type:** Primary and essential  
**Description:** The Instructor completes an assignment and submits it to the system. The instructor will also submit the delivery date, due date, and the class the assignment is assigned for. The system will at the due date mail the assignment to the student.  
**Cross Ref.:** Requirements XX, YY, and ZZ  
**Use-Cases:** *Configure HACS* must be done before any user (Instructor or Student) can use HACS

05-Use-Cases

31



## When to use Use-Cases

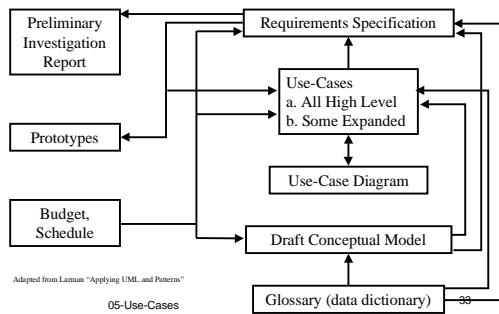
- In short, always!!!
- Requirements is the toughest part of software development
  - Use-Cases is a powerful tool to understand
    - Who your users are (including interacting systems)
    - What functions the system shall provide
    - How these functions work at a high level
- Spend adequate time on requirements and in the elaboration phase

05-Use-Cases

32



## How it Fits Together



05-Use-Cases

33