


Use Cases and Scenarios

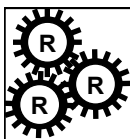
CSE870: Advanced Software Engineering: UML-- Use Cases 1



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

CSE870: Advanced Software Engineering: UML-- Use Cases 2

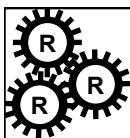


What is a Use-Case

- 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)?

CSE870: Advanced Software Engineering: UML-- Use Cases

3

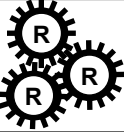


User Goals versus User Interactions

- 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

CSE870: Advanced Software Engineering: UML-- 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

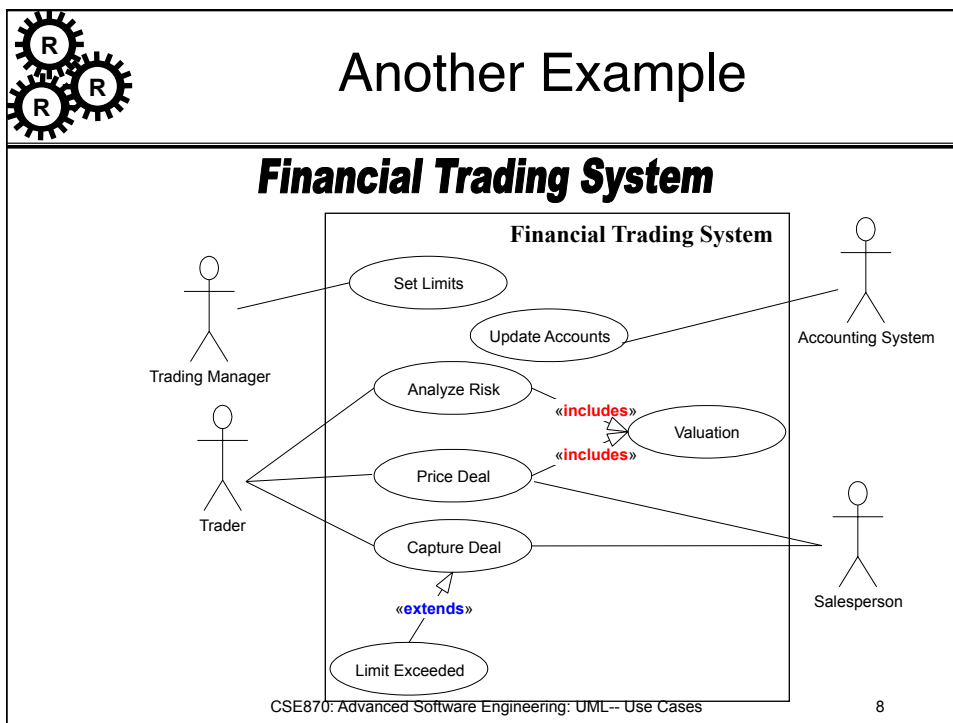
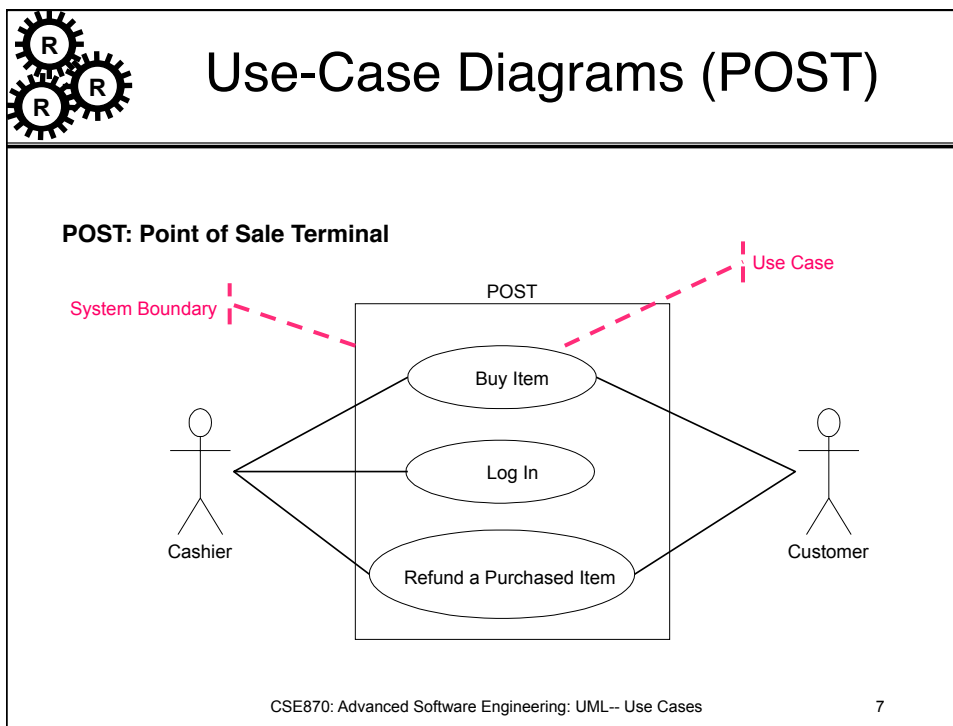
CSE870: Advanced Software Engineering: UML-- Use Cases 5




Point of Sale Terminal (POST)

- Computerized system to record sales and handle payments
- Typically used in a retail store
- Includes HW components, such as computer and bar code scanner
- Software to run the system
- Goals of system:
 - Increase checkout automation
 - fast and accurate sales analysis
 - automatic inventory control

CSE870: Advanced Software Engineering: UML-- Use Cases 6






Includes and Extends

- Includes/Uses
 - 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 much easier to understand

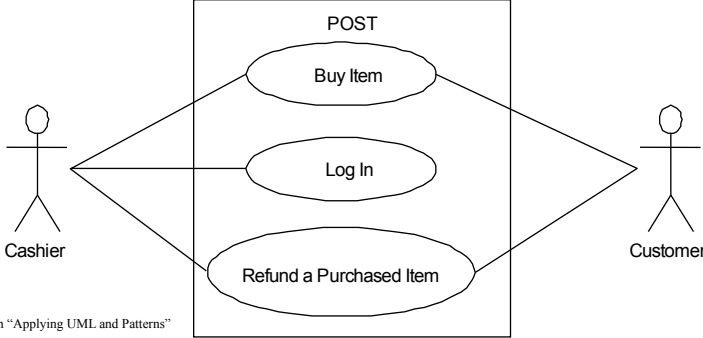
CSE870: Advanced Software Engineering: UML-- Use Cases

9



Setting the System Boundary

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



```

graph LR
    subgraph SystemBoundary [POST]
        UC1([Buy Item])
        UC2([Log In])
        UC3([Refund a Purchased Item])
    end
    Cashier((Cashier)) --- UC1
    Cashier --- UC2
    Cashier --- UC3
    Customer((Customer)) --- UC1
    Customer --- UC3
  
```

Adapted from Larman "Applying UML and Patterns"

CSE870: Advanced Software Engineering: UML-- Use Cases

12

A Different Boundary

- Let us view the whole store as our system

The diagram shows a rectangular box labeled 'Store'. Inside the box are two ovals representing use cases: 'Buy Item' and 'Refund a Purchased Item'. To the right of the box is a stick figure actor labeled 'Customer'. Two lines connect the 'Customer' actor to the 'Buy Item' and 'Refund a Purchased Item' use cases, indicating that the customer can interact with both.

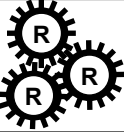
Adapted from Larman "Applying UML and Patterns"

CSE870: Advanced Software Engineering: UML-- Use Cases 13

Partial POST

The diagram shows a rectangular box labeled 'POST'. Inside the box are six ovals representing use cases: 'Buy Item', 'Log In', 'Refund a Purchased Item', 'Start Up', 'Manage Users', and 'And a Lot More'. Outside the box are four stick figure actors: 'Cashier' (top left), 'Customer' (top right), 'System Administrator' (bottom left), and 'Manager' (bottom right). Lines connect the actors to the use cases: 'Cashier' connects to 'Buy Item', 'Log In', and 'Refund a Purchased Item'; 'Customer' connects to 'Buy Item', 'Log In', and 'Refund a Purchased Item'; 'System Administrator' connects to 'Manage Users'; and 'Manager' connects to 'Start Up'.


CSE870: Advanced Software Engineering: UML-- Use Cases 14



POST Use-Case

Use case:	Buy Item
Actors:	Customer (initiator), Cashier
Type:	Primary
Description:	The Customer arrives at the checkout with items to purchase. The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items

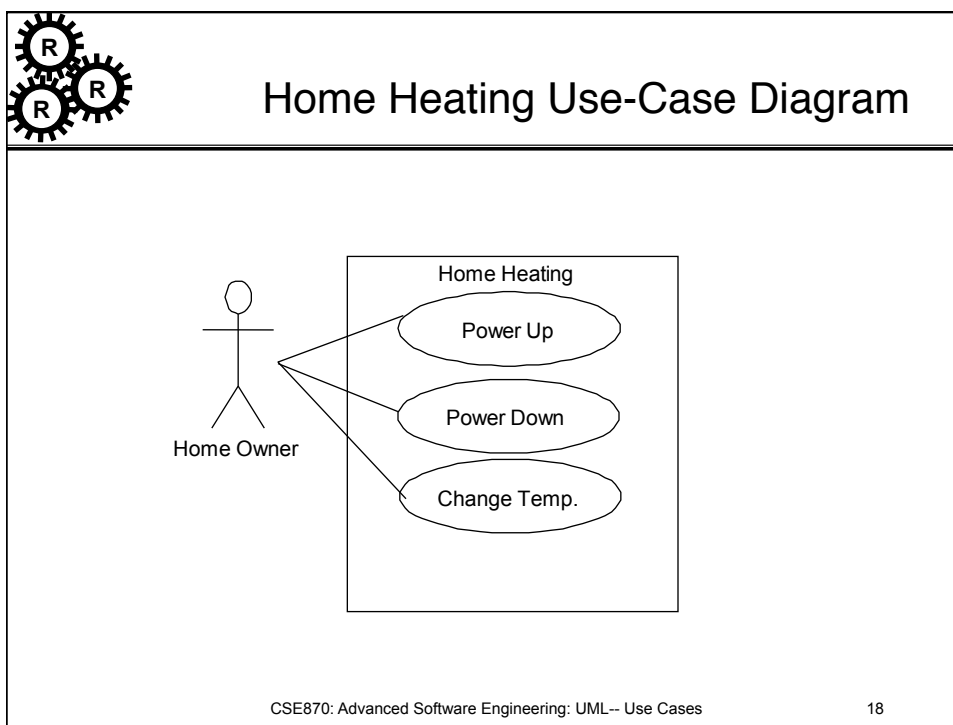
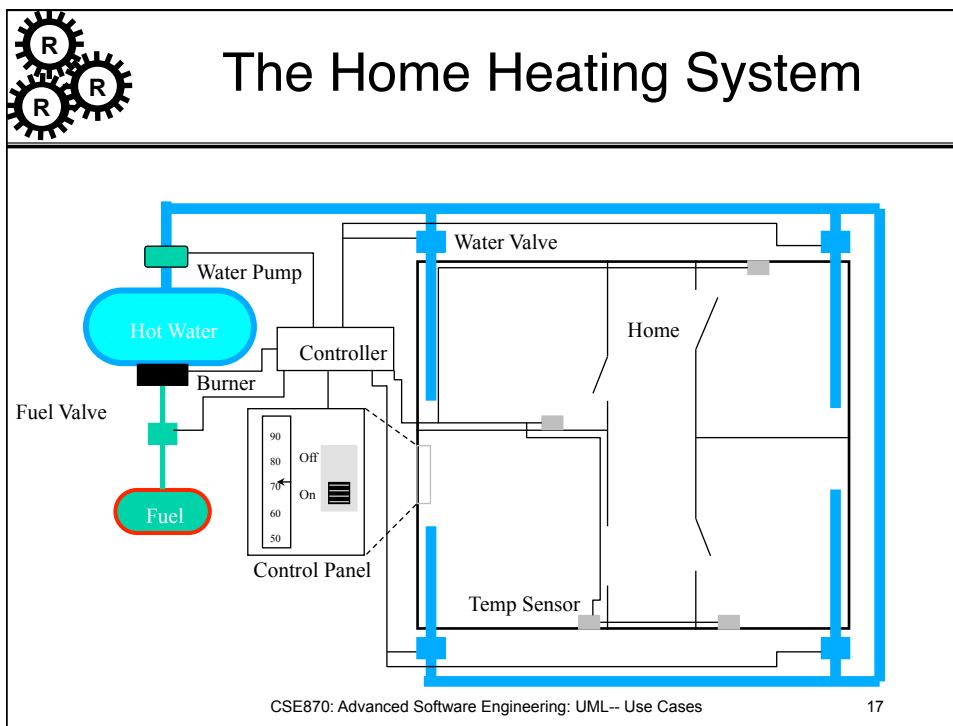
CSE870: Advanced Software Engineering: UML-- Use Cases 15




POST Expanded Use-Case

Use case:	Buy Item
Actors:	Customer (initiator), Cashier
Type:	Primary and essential
Description:	The Customer arrives at the checkout with items to purchase. The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items.
Cross Ref.:	Requirements XX, YY, and ZZ
Use-Cases:	Cashier must have completed the <i>Log In</i> use-case

CSE870: Advanced Software Engineering: UML-- Use Cases 16





Home Heating Use-Cases

Use case: **Power Up**

Actors: Home Owner (initiator)

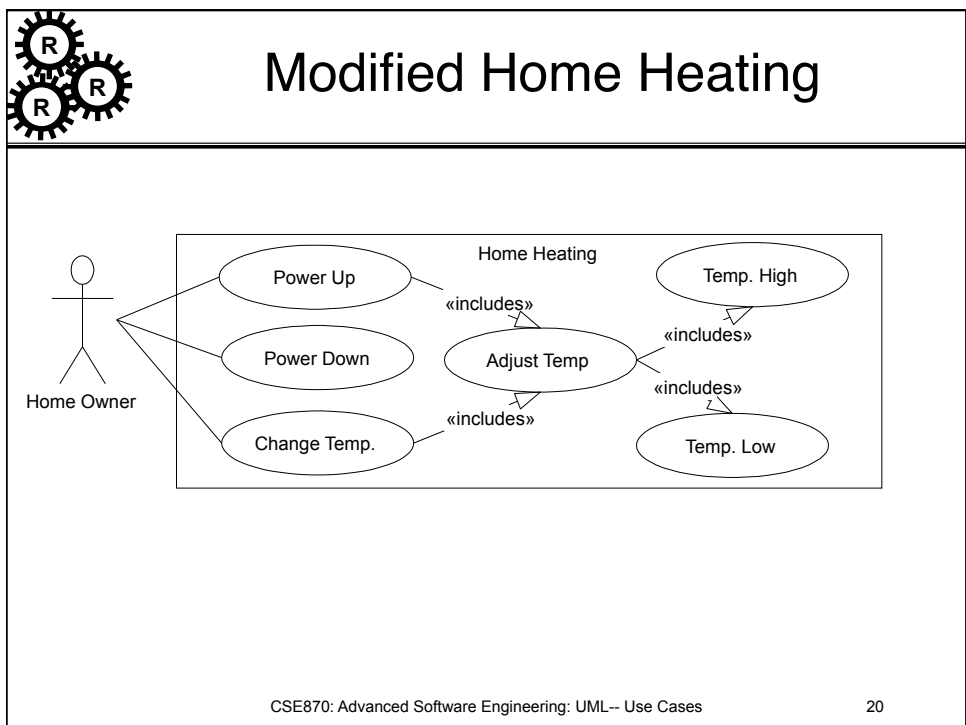
Type: Primary and essential

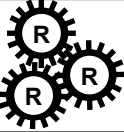
Description: The Home Owner turns the power on. Each room is temperature checked. If a room is below the the desired temperature the valve for the room is opened, the water pump started, the fuel valve opened, and the burner ignited. If the temperature in all rooms is above the desired temperature, no actions are taken.

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: None

CSE870: Advanced Software Engineering: UML-- Use Cases 19






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.

CSE870: Advanced Software Engineering: UML-- Use Cases 21

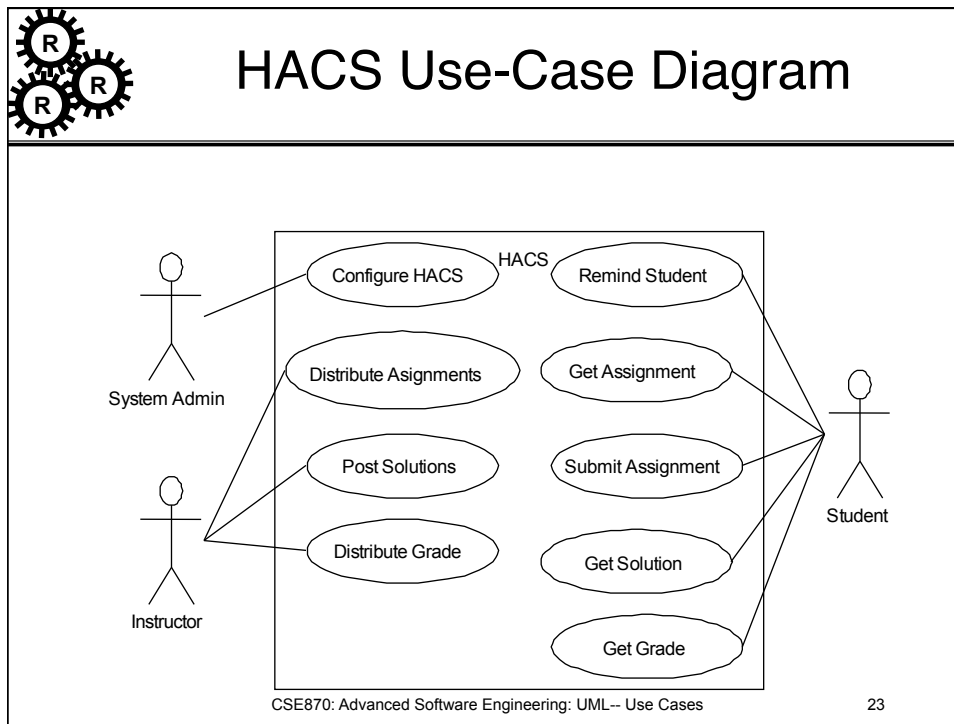


In-class exercise

Work in Pairs

- Create use case diagram
- Have an example of <<includes>> relationship
- Bonus: have an example of <<extends>> relationship
- Remember to include key elements of UC diagram.

CSE870: Advanced Software Engineering: UML-- Use Cases 22



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

CSE870: Advanced Software Engineering: UML-- Use Cases 24

