Lecture # 26

Use Case Modeling - II

Documenting Use Cases - 1

- Name
- Summary
  - Short description of use case
- Dependency (on other use cases)
- Actors
- Preconditions
  - Conditions that are true at start of use case

Documenting Use Cases - 2

- Flow of Events
  - Narrative description of basic path
- Alternatives
  - Narrative description of alternative paths
- Post-condition
  - Condition that is true at end of use case

Use Cases and Flow of Events - 1

- A use case describes *what* the system (or subsystem, class, or interface) does but it does not specify *how* it does it
- It is important that you keep clear the separation of concerns between this outside and inside view

Use Cases and Flow of Events - 2

- The behavior of a use case can be specified by describing a flow of events in text clearly enough for an outsider to understand it easily

Use Cases and Flow of Events - 3

- How and when the use case starts and ends
- When the use case interacts with the actors and what objects are changed
- The basic flow and alternative flows of behavior

Use Cases and Flow of Events - 4

- A use case's flow of events can be specified in a number of ways:
  - Informal structured text
  - Formal structured text (with pre- and post-conditions)
  - Pseudocode

Organizing Use Cases - 1

- Use cases can be organized by grouping them in packages
- You can also organize use cases by specifying generalization, include, and extend relationships among them

Organizing Use Cases - 2

- You apply these relationships in order to factor common behavior (by pulling such behavior from other use cases that it includes) and in order to factor variants (by pushing such behavior into other use cases that extend it)

Organizing Use Cases - 3

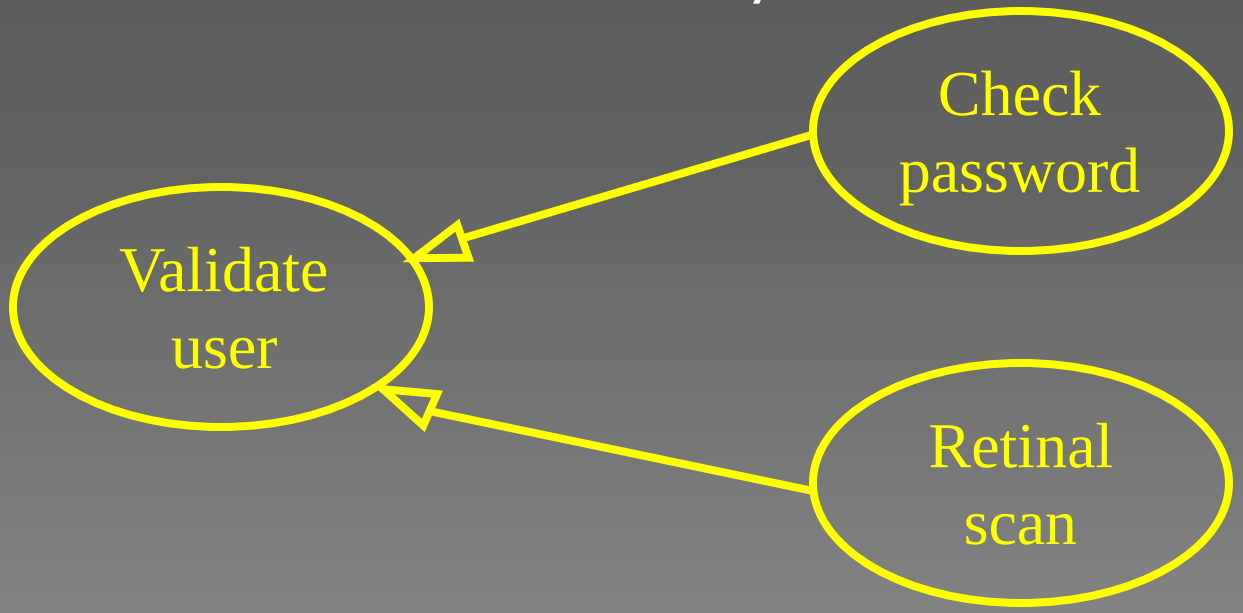
- Generalization among use cases is just like generalization among classes
- It means that the child use case inherits the behavior and meaning of the parent use case; the child may add to or override the behavior of its parent; and the child may be substituted any place the parent appears

Organizing Use Cases - 4

- You may have a use case Validate User, which is responsible for the verifying the identity of the user
- This use case can be used in many different application domains

Specialized Use Cases

- You may have two specialized children of this use case (Check password and Retinal scan)



Organizing Use Cases - 5

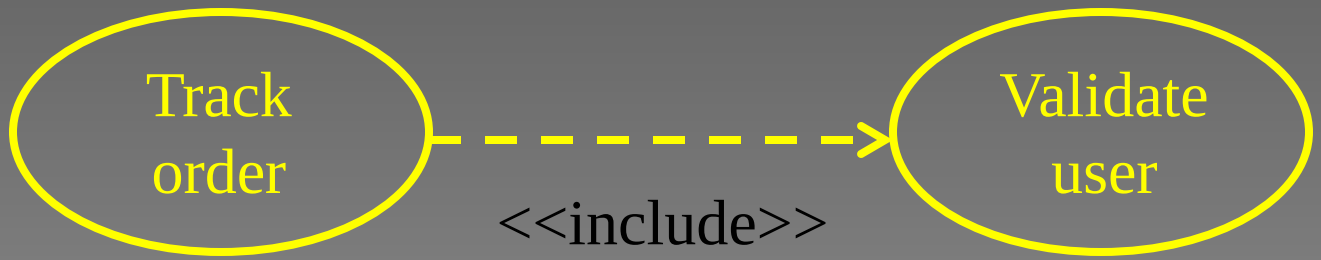
- An include relationship between use cases means that the base use case explicitly incorporates the behavior of another use case at a location specified in the base
- The included use case never stands alone, but is only instantiated as part of some larger base that includes it
- It is like the base use case

Organizing Use Cases - 6

- Include relationship is used to avoid describing the same flow of events several times, by putting the common behavior in a use case of its own
- This is an example of dependency
- Track order use case includes Validate user use case

Including a Use Case

- Obtain and verify the order number. include (Validate user). For each part in the order, query its status, then report back to the user



Organizing Use Cases - 8

- An extend relationship between use cases means that the base use case implicitly incorporates the behavior of another use case at a location specified indirectly by extending use case
- The base use case may stand alone, but under certain conditions, its behavior may be extended by another use case

Organizing Use Cases - 9

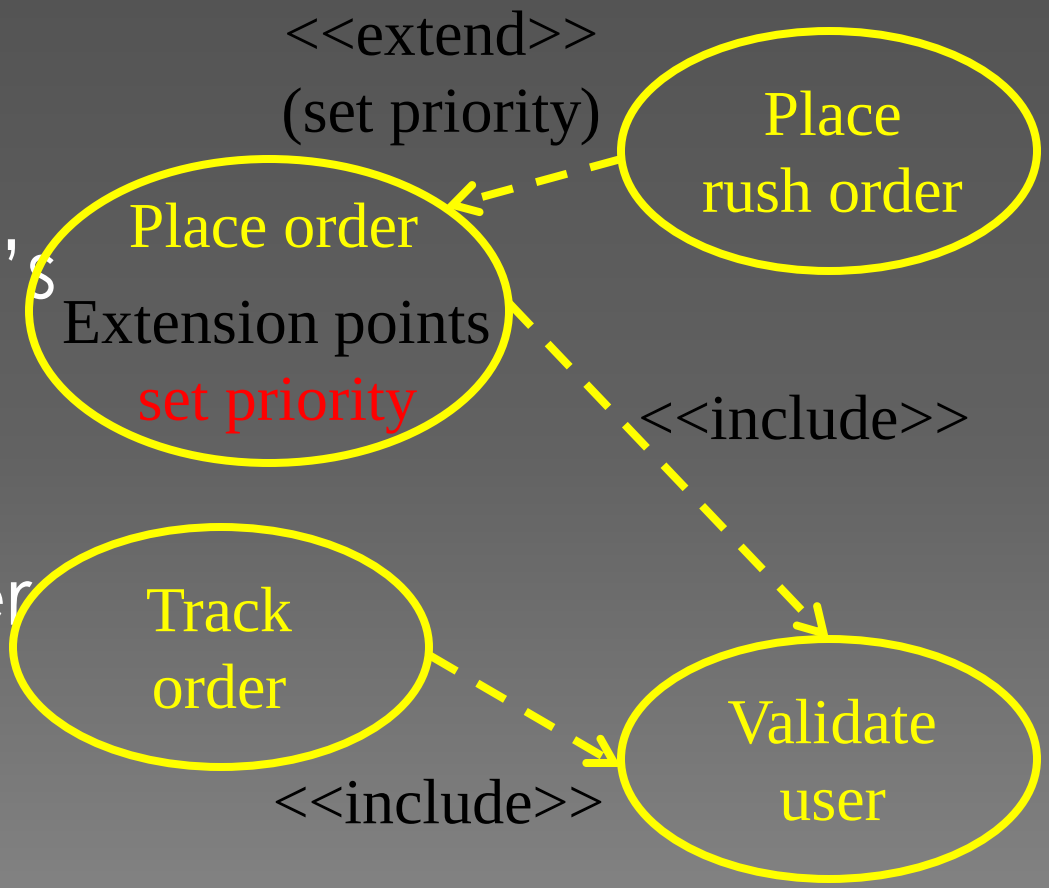
- A base use case may be extended only at certain points, called its extension points
- Extend relationship can be used to model
  - Optional behavior
  - Separate sub-flow that is executed only under given conditions
  - Several flows that may be inserted at a certain point, governed by explicit interaction with an actor

Organizing Use Cases - 10

- This is also an example of dependency
- Consider the Place order use case, which extends the Place rush order

Extending a Use Case

- include (Validate user)
- Collect the user's order items
- (set priority)
- Submit the order for processing



Organizing Use Cases - 11

- In this example, set point is an extension point
- A use case may have more than one extension points

Organizing Use Cases - 12

- Apply use cases to model the behavior of an element (system, subsystem, class)
- Outside view by domain experts
- Developers can approach the element
- Basis for testing

Guidelines for Use Cases - 1

- Identify the actors that interact with the element. Candidate actors include groups that require certain behavior to perform their tasks or that are needed directly or indirectly to perform the element's functions

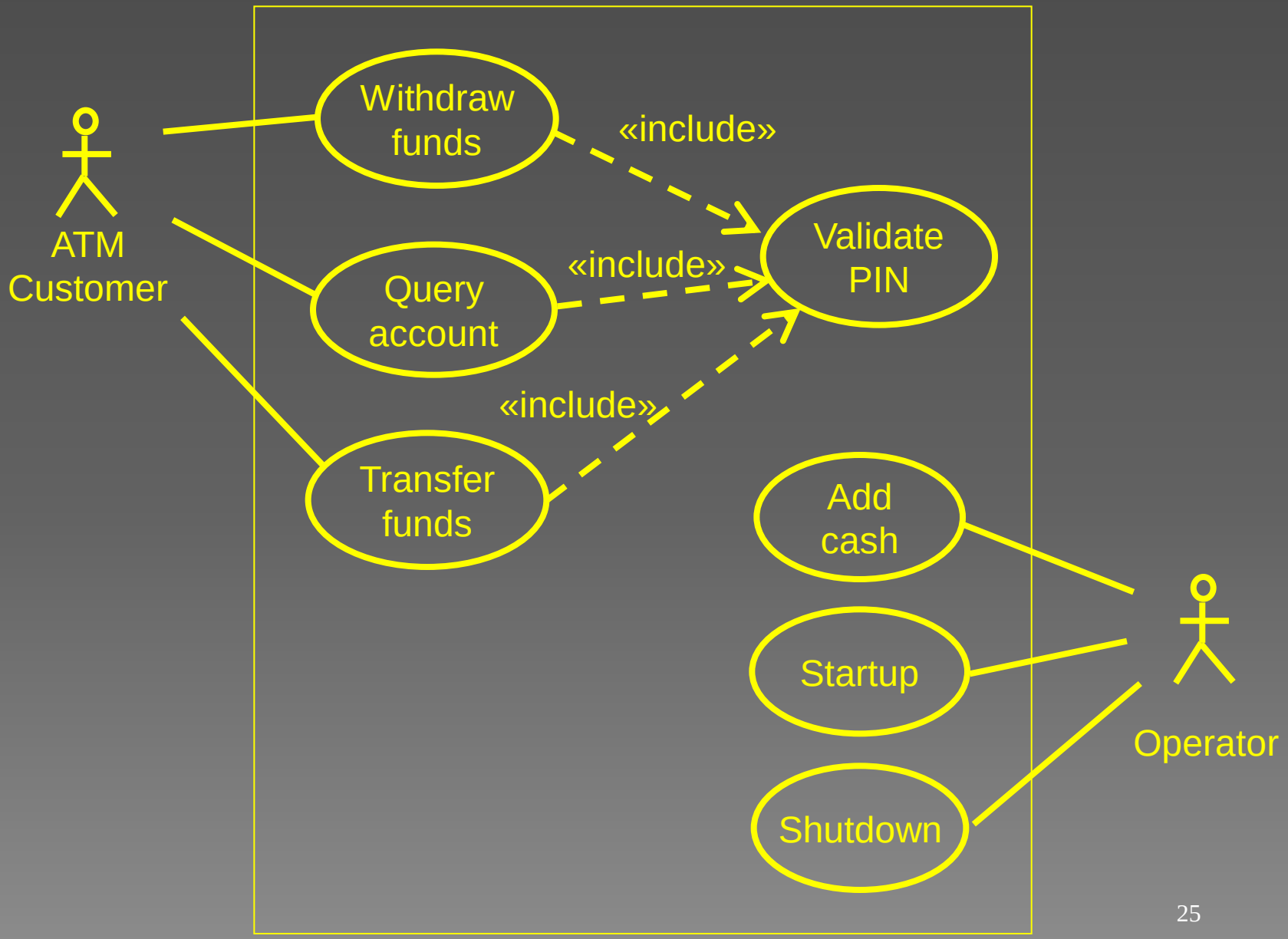
Guidelines for Use Cases - 2

- Organize actors by identifying general and more specialized roles
- For each actor, consider the primary ways in which that actor interacts with the element. Consider also interactions that change the state of the element or its environment or that involve a response to some event

Guidelines for Use Cases - 3

- Consider the exceptional ways in which each actor interacts with the element
- Organize these behaviors as use cases, applying include and extend relationships to factor common behavior and distinguish exceptional behavior

Use Case Diagram for ATM



Abstract Use Case Example - 1

- Name: Validate PIN
- Summary : System validates customer PIN
- Dependency: none
- Actors: ATM Customer
- Preconditions: ATM is idle, displaying a Welcome message.

Abstract Use Case Example - 2

- Flow of Events: Basic Path
  - 1. Customer inserts the ATM card into the Card Reader
  - 2. If the system recognizes the card, it reads the card number
  - 3. System prompt customer for PIN number
  - 4. Customer enters PIN
  - 5. System checks the expiration date and whether the card is lost or stolen

Abstract Use Case Example - 3

- Flow of Events (cont.):
  - 6. If card is valid, the system then checks whether the user-entered PIN matches the card PIN maintained by the system
  - 7. If PIN numbers match, the system checks what accounts are accessible with the ATM card
  - 8. System displays customer accounts and prompts customer for transaction type: Withdrawal, Query, or Transfer

Abstract Use Case Example - 4

- Alternatives:
  - If the system does not recognize the card, the card is ejected
  - If the system determines that the card date has expired, the card is confiscated
  - If the system determines that the card has been reported lost or stolen, the card is confiscated

Abstract Use Case Example - 5

- Alternatives (cont.):
  - If the customer-entered PIN does not match the PIN number for this card, the system re-prompts for PIN
  - If the customer enter the incorrect PIN three times, the system confiscates the card
  - If the customer enters Cancel, the system cancels the transaction and ejects the card
- Postcondition: Customer PIN has been validated

Concrete Use Case Example - 1

- Name: Withdraw Funds
- Summary : Customer withdraws a specific amount of funds from a valid bank account
- Dependency: Include Validate PIN abstract use case
- Actors: ATM Customer
- Preconditions: ATM is idle, displaying a Welcome message.

Concrete Use Case Example - 2

- Flow of Events: Basic Path
  - 1. Include Validate PIN abstract use case
  - 2. Customer selects Withdrawal, enters the amount, and selects the account number
  - 3. System checks whether customer has enough funds in the account and whether the daily limit will not be exceeded

Summary

- Extended the discussion on Use Case Modeling
- The use cases are documented using simple flow of events along with any variations
- There are some relationships between use cases which help us in extending and reusing the existing use cases