Lecture 2:
Requirements Modeling

H. Gomaa, “Chapter 6, Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures,” Cambridge University Press, February 2011

Steps in Using COMET

1 Develop Software Requirements Model
   – Develop Use Case Model (Chapter 6)

2 Develop Software Analysis Model
   – Develop static model of problem domain (Chapter 7)
   – Structure system into objects (Chapter 8)
   – Develop statecharts for state dependent objects (Chapter 10)
   – Develop object interaction diagrams for each use case (Chapter 9, 11)

3 Develop Software Design Model

Requirements Specification

- The process of defining requirements (called requirements engineering)
  - Identifying the requirements that a customer requires from a system
  - Specifying requirements
  - Validating requirements
- Requirements
  - Functional requirements
  - Non-functional requirements

CS5373 3

3

Functional Requirements

- Services the system should provide
  - Functions of systems
  - Positive requirements
  - e.g., Healthcare System
    - A user shall be able to search the appointments for all clinics.
- May state what the system should not do
  - Negative requirements

CS5373 4

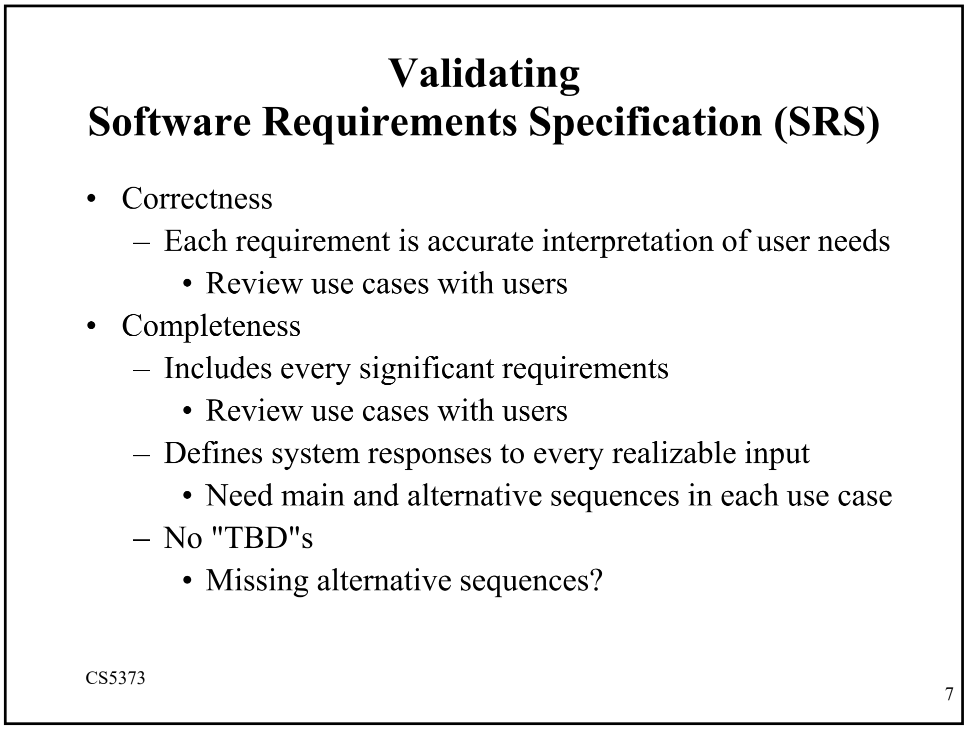
4

Non-functional requirements

- Non-functional requirements
  - Constraints on the services (or functions) offered by the system
    - E.g., reliability, response time, and security
  - Constraints on development or operation of system
    - E.g., SDLC, programming language
    - E.g., Organizational changes

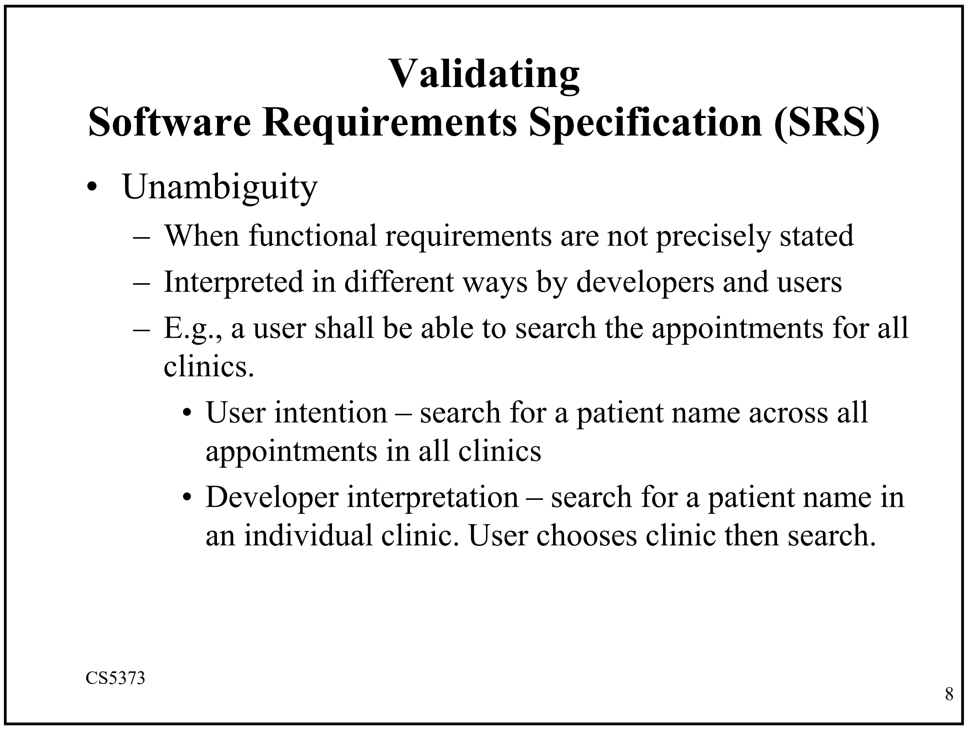
Non-functional requirements

- May be more critical than functional requirements
  - Often apply to the system as a whole rather than individual services
  - System useless if not met
- Non-functional requirement may generate functional requirements
  - E.g., security requirement



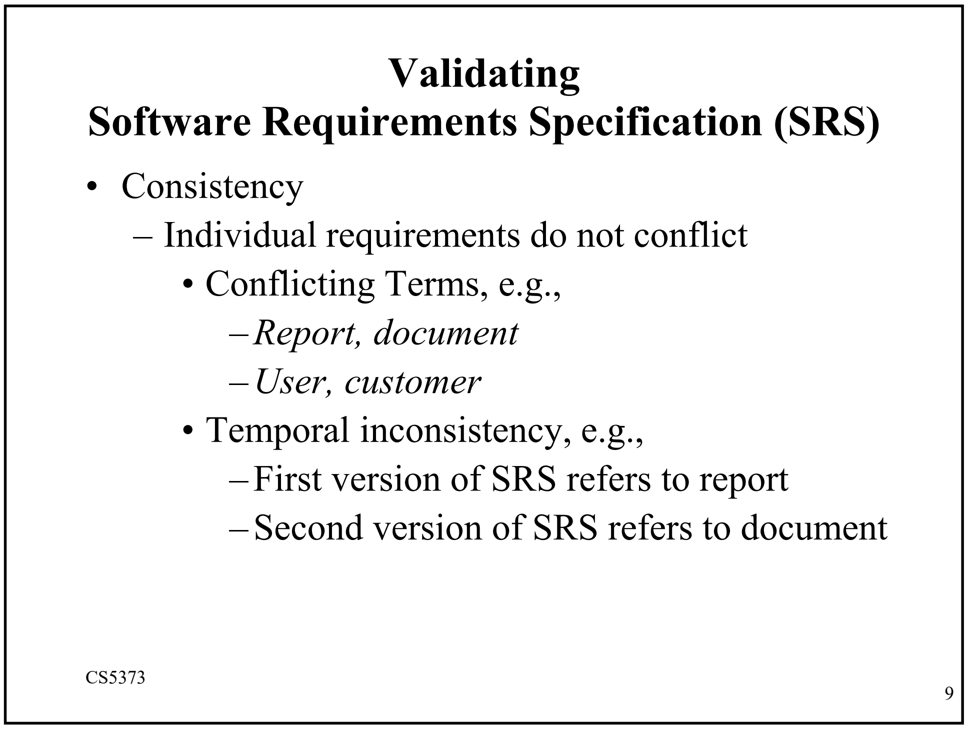
Validating Software Requirements Specification (SRS)

- Correctness
  - Each requirement is accurate interpretation of user needs
    - Review use cases with users
- Completeness
  - Includes every significant requirements
    - Review use cases with users
  - Defines system responses to every realizable input
    - Need main and alternative sequences in each use case
  - No "TBD"s
    - Missing alternative sequences?



Validating Software Requirements Specification (SRS)

- Unambiguity
  - When functional requirements are not precisely stated
  - Interpreted in different ways by developers and users
  - E.g., a user shall be able to search the appointments for all clinics.
    - User intention – search for a patient name across all appointments in all clinics
    - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.



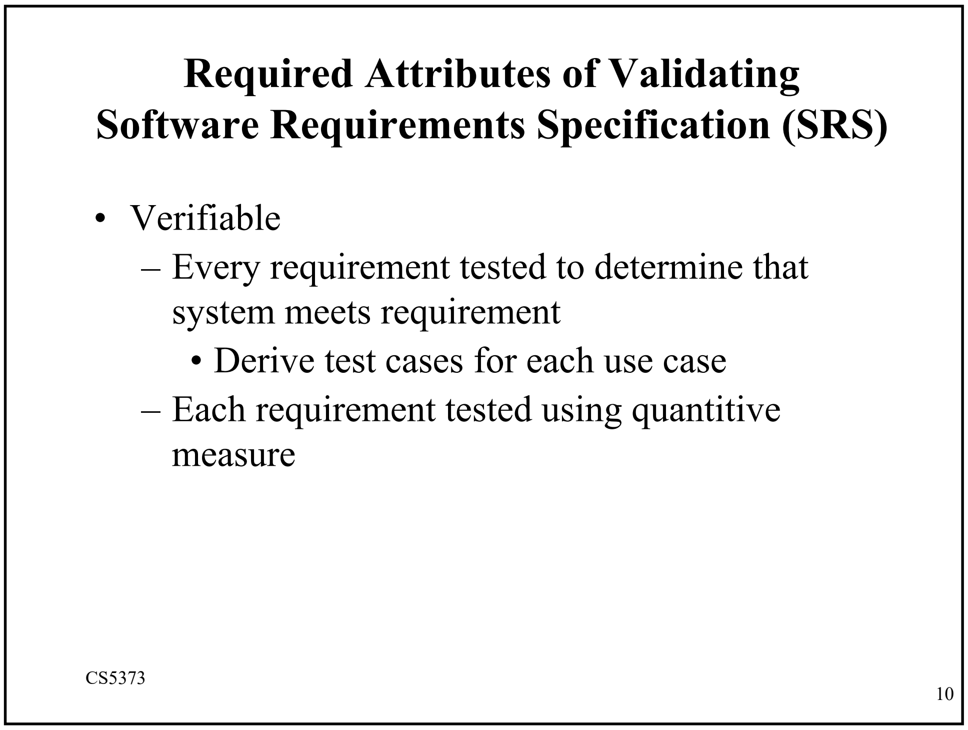
Validating Software Requirements Specification (SRS)

- Consistency
  - – Individual requirements do not conflict
    - Conflicting Terms, e.g.,
      - – *Report, document*
      - – *User, customer*
    - Temporal inconsistency, e.g.,
      - – First version of SRS refers to report
      - – Second version of SRS refers to document

CS5373

9

9



Required Attributes of Validating Software Requirements Specification (SRS)

- Verifiable
  - – Every requirement tested to determine that system meets requirement
    - Derive test cases for each use case
  - – Each requirement tested using quantitive measure

CS5373

10

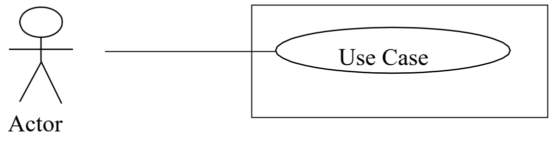
10

Use Case Modeling

Chapter 6 - *Software Modeling and Design*,
Cambridge University Press

Use Case Modeling

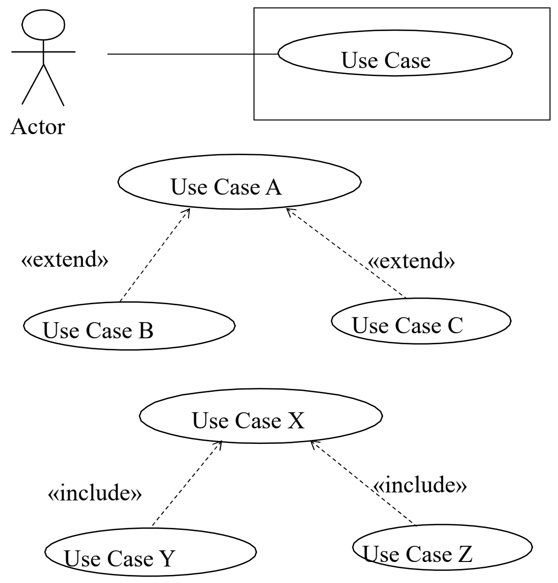
- Use Case modeling
  - Specify user’s requirements
  - Define system functional requirements in terms of Actors and Use cases



Use Case Modeling

- Initially developed in Use Case model
  - Functional requirements defined in terms of actors and use cases
  - Shows interaction between actor and system
- Use cases refined in Dynamic Model
  - Show objects participating in use case
  - Develop communication diagrams or sequence diagrams
- Use cases refined further in Design Model
- Use cases form basis of integration & system test cases
- Fig. 2.1

Figure 2.1 UML notation for use case diagram



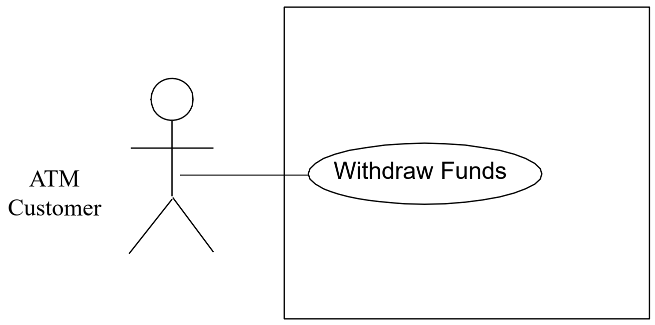
Actors

- External to system
- Interact directly with system
- System - makes decisions and not Actor
- When actor communicates with actor and bypasses system
  - Report stolen card use case
    - E.g., *ATM Customer sends message to ATM Operator*
    - E.g., *ATM customer talks over the phone to a bank clerk*

Actors

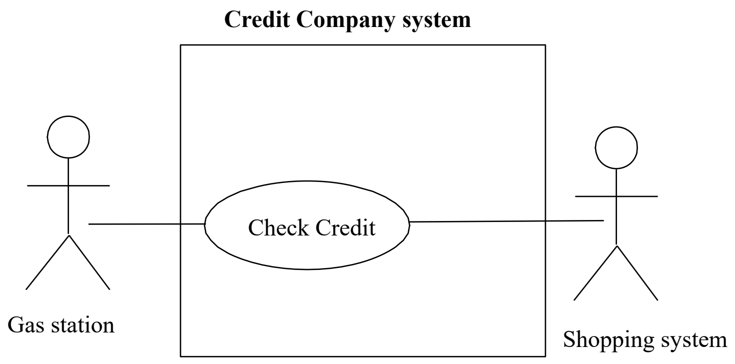
- Initiates interactions with use case
- Primary Actor
  - Starts the use case by providing input to the system
- Secondary Actor(s)
  - Participates in use case
  - Can be Primary Actor of a different use case
- An actor could be:
  - Human user
  - External system
  - Input device (sensor)
  - Timer

Figure 6.1 Example of human user actor



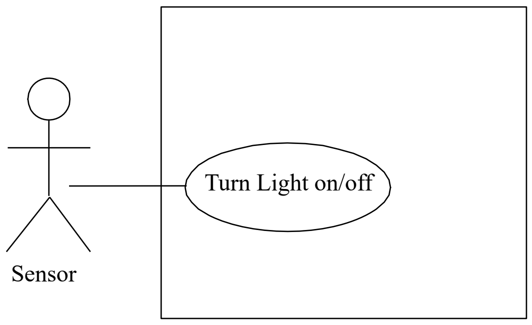
CS5373

Example of external system actor



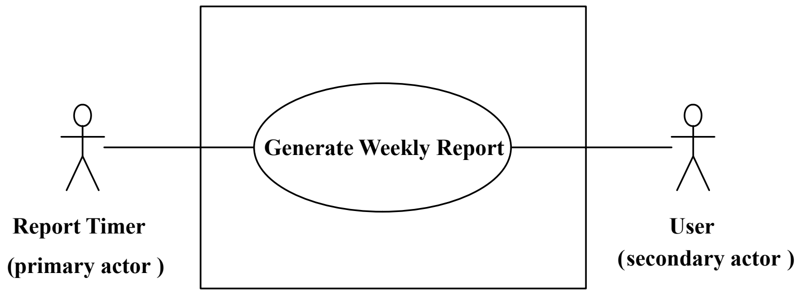
CS5373

Example of input device actor



CS5373

Figure 6.5 Example of timer actor



CS5373

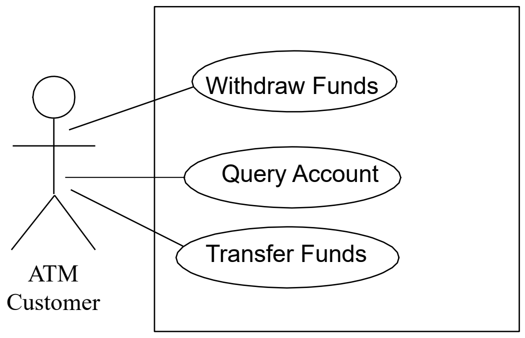
Use Cases

- Use case
  - Describes a complete sequence of interactions between actor and system
  - Starts with input from an actor
  - Use case name: Should describe an action, e.g., *Validate PIN*
- How to identify use cases?
  - Identify each actor first
  - Consider requirements of each actor who interacts with system

Use Cases

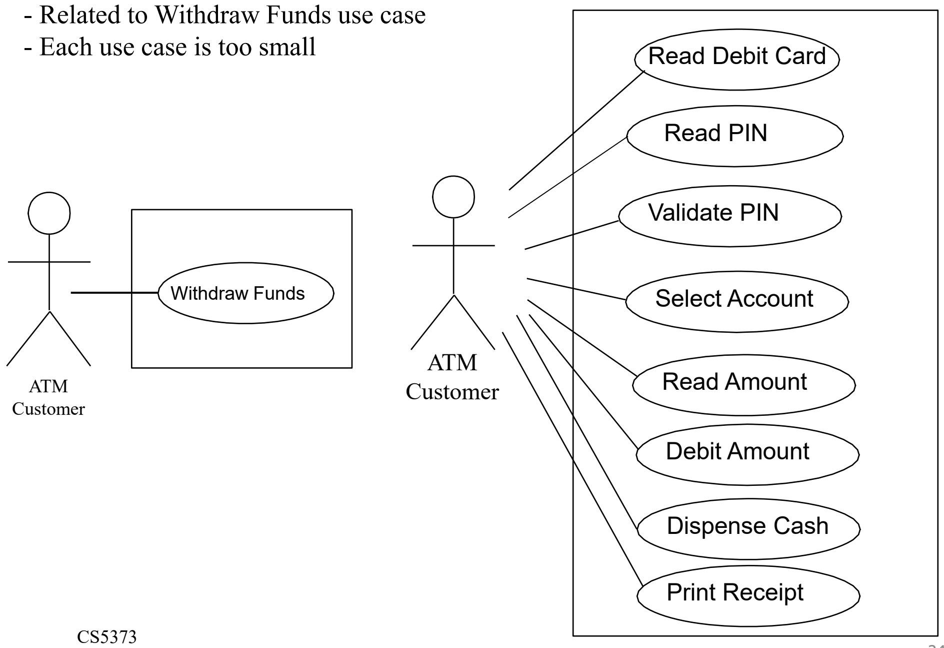
- Use case is a function provided by a system
  - Need to be big functions from actor perspective
  - A use case may require small functions for implementation
    - Small functions should not be use cases
    - Small functions make the use case model complicated
    - Each small function can be an operation (method) in class

Good Use Case Model



CS5373

Bad Use Cases in ATM system:
- Related to Withdraw Funds use case
- Each use case is too small



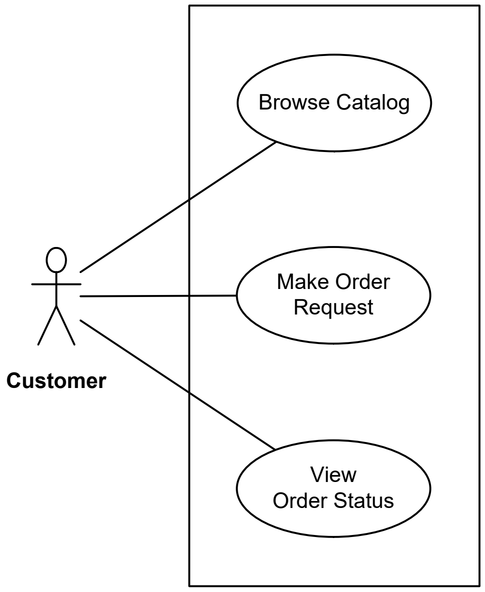
CS5373

Figure 6.8 Online Shopping System - Use Cases

- Browse Catalog
  - Customer browses catalog and selects items
  - System displays selected items and price
- Make Order Request
  - Customer makes purchase
  - Customer provides account and credit card info
- View Order Status
  - Customer views status of order

CS5373

Figure 6.8 Online Shopping System Actor and Use Cases



CS5373

Documenting Use Cases

- Name
- Summary - Short description of use case
- Dependency (on other use cases)
- Actors – primary, secondary
- Precondition(s)
  - Condition that exists before use case executes
    - e.g., *ATM is Idle, displaying a welcome message*
- Description of main sequence
  - Most common sequence
  - Sentences start with System or Actor, e.g.,
    - *Customer selects payment by credit card*
    - *System prompts Customer to swipe card*

CS5373 27

27

Documenting Use Cases

- Description of alternative sequences
  - Deviations from main sequence
    - E.g., for error handling
    - E.g., for branching out of main sequence
  - Identify step # where deviation starts
  - Rejoin main sequence?
    - Application dependent
- Nonfunctional requirements (optional)
  - Can be specified in a section separated from the use case
  - Can be specified with use case
- Postcondition
  - Condition that is true at end of use case

CS5373 28

28

Example of Use Case (Pages 81-82)

Use Case Name: Make Order Request

Summary: Customer enters an order request to purchase items from the online shopping system. The customer's credit card is checked for sufficient credit to pay for the requested catalog items.

Actor: Customer

Precondition: The customer has selected one or more catalog items.

Description of main sequence:

1. Customer provides order request and customer account id to pay for purchase.
2. System retrieves customer account information, including the customer's credit card details.
3. System checks the customer's credit card for the purchase amount and, if approved, creates a credit card purchase authorization number.
4. System creates a delivery order containing order details, customer id, and credit card authorization #.
5. System confirms approval of purchase and displays order information to customer.

Alternative sequences:

Step 2: If customer does not have account, the system creates an account.

Step 3: If the customer's credit card request is denied, the system prompts the customer to enter a different credit card number. The customer can either enter a different credit card number or cancel the order.

Postcondition: System has created a delivery order for the customer.

Use Case Relationships

- Extend and Include relationships
  - Avoid complexity of use case modeling
  - Maximize reuse and extensibility of use cases

- **Include** relationship
  - Identify common patterns (sequences) of interactions in several use cases
  - Extract common pattern into **inclusion use case**
  - Base use cases **include** inclusion use case
  - Inclusion use case might not have a specific actor

Figure 6.7 Banking system actor and use cases

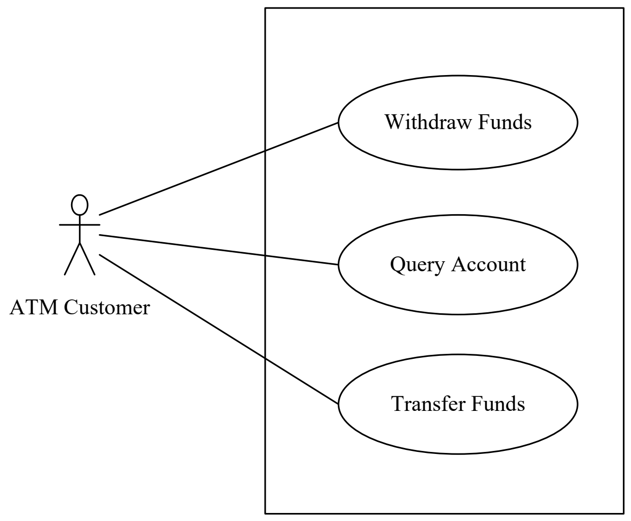
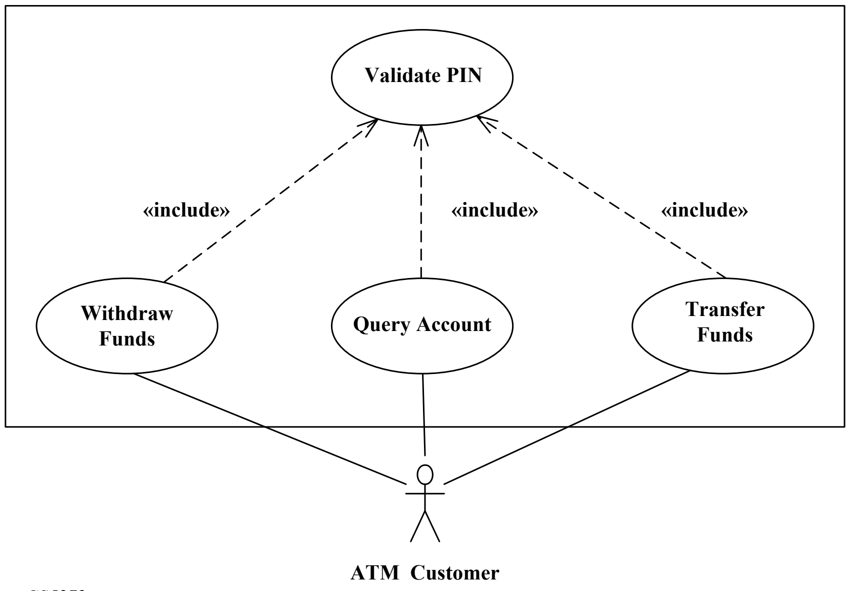


Figure 6.9 Example of inclusion use case and include relationships



Example of Inclusion Use Case (Page 374)

Use case name: Validate PIN.

Summary: System validates customer PIN.

Actor: ATM Customer.

Precondition: ATM is idle, displaying a "Welcome" message.

Description of main sequence:

1. Customer inserts the ATM card into the card reader.
2. If system recognizes the card, it reads the card number.
3. System prompts customer for PIN.
4. Customer enters PIN.
5. System checks the card's expiration date and whether the card has been reported as lost or stolen.
6. If card is valid, system then checks whether the user-entered PIN matches the card PIN maintained by the system.
7. If PIN numbers match, 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.

Example of Inclusion Use Case (Page 374)

Alternatives

Step 2: If the system does not recognize the card, the system ejects the card.

Step 5: If the system determines that the card date has expired, the system confiscates the card.

Step 5: If the system determines that the card has been reported lost or stolen, the system confiscates the card.

Step 7: If the customer entered PIN does not match the PIN number for this card, the system reprompts for the PIN.

Step 7: If the customer enters the incorrect PIN three times, the system confiscates the card

Steps 4-8: If the customer enters Cancel, the system cancels the transaction and ejects the card.

Nonfunctional requirements:
a)Security requirement: System shall encrypt ATM card number and PIN.
b)Performance requirement: System shall respond to actor inputs within 5 seconds.

Postcondition: Customer PIN has been validated.

Example of Base Use Case Page 374-375

Use Case Name: Withdraw Funds

Summary: Customer withdraws a specific amount of funds from a valid bank account.

Actor: ATM Customer

Dependency: Include `Validate PIN` use case.

Precondition: ATM is idle, displaying a Welcome message.

Description of main sequence:

1. Include `Validate PIN` 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.
4. If all checks are successful, system authorizes dispensing of cash.
5. System dispenses the cash amount.
6. System prints a receipt showing transaction number, transaction type, amount withdrawn, and account balance.
7. System ejects card.
8. System displays Welcome message.

Example of Base Use Case Page 374-375

Alternatives:

Step 3: If the system determines the account number is invalid, then it displays an error message and ejects the card.

Step 3: If the system determines there are insufficient funds in the customer's account, then it displays an apology and ejects the card.

Step 3: If the system determines the maximum allowable daily withdrawal amount has been exceeded, it displays an apology and ejects the card.

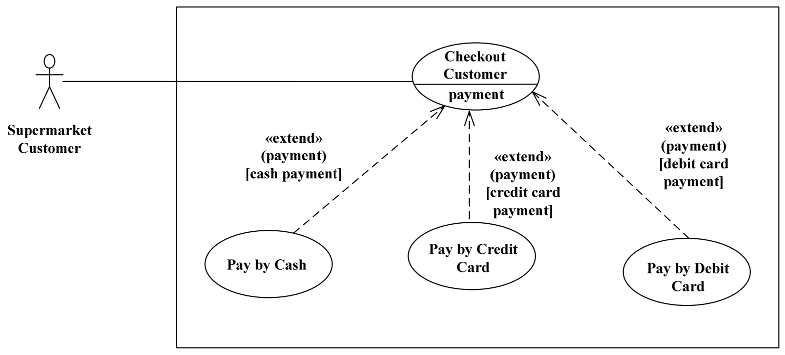
Step 4: If the ATM is out of funds, the system displays an apology, ejects the card, and shuts down the ATM.

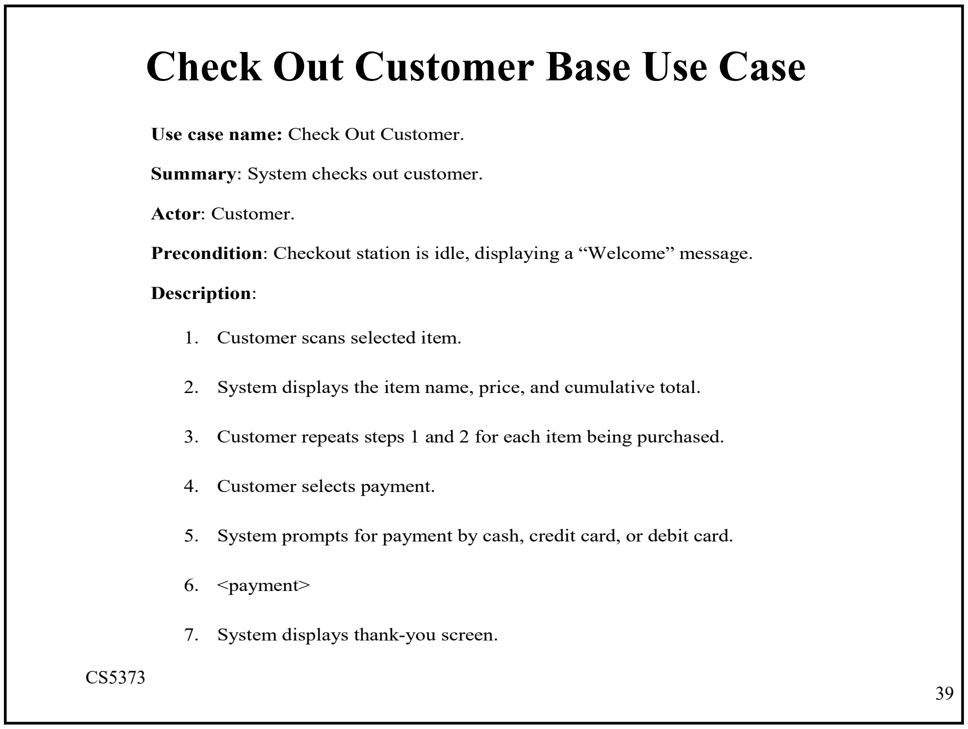
Postcondition: Customer funds have been withdrawn.

Extend relationship

- Extend relationship
  - Model alternative paths that a basic use case may take
  - A use case B extends a use case A at an extension point if the appropriate conditions hold
  - Base use case and extension use case
  - Extension point
    - The location in the base use case at which an extension can be added
    - Designated in description for the base use
  - Same use case can be extended in different ways
  - Notation

Figure 6.11 Example of extend relationship


Checkout Customer
payment
Supermarket Customer
«extend» (payment) [cash payment]
«extend» (payment) [credit card payment]
«extend» (payment) [debit card payment]
Pay by Cash
Pay by Credit Card
Pay by Debit Card



Check Out Customer Base Use Case

Use case name: Check Out Customer.

Summary: System checks out customer.

Actor: Customer.

Precondition: Checkout station is idle, displaying a “Welcome” message.

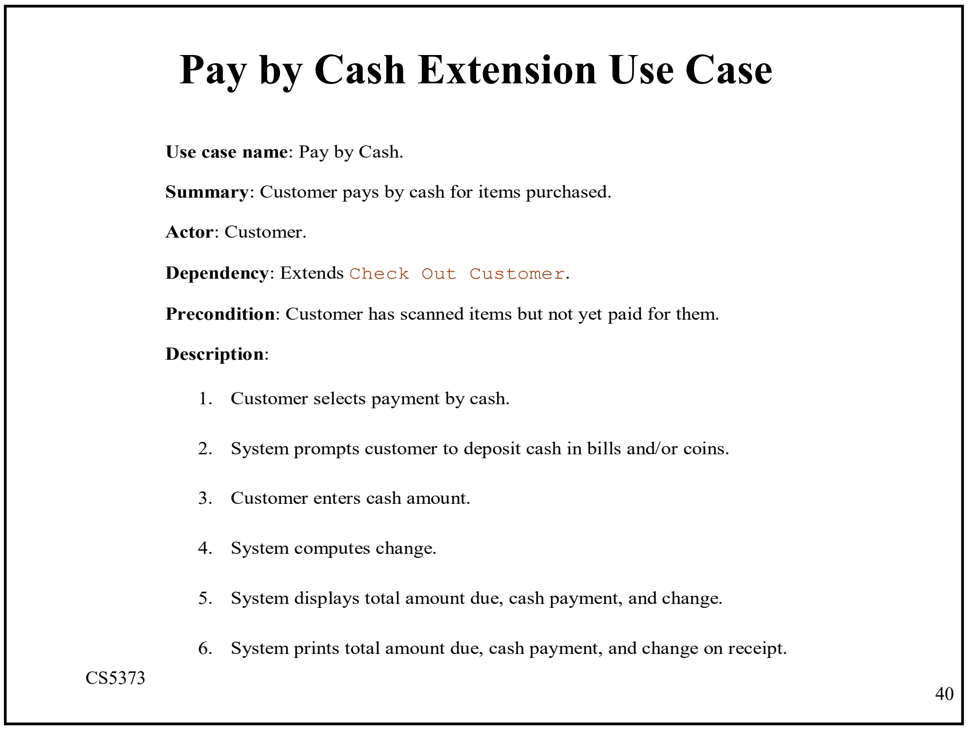
Description:

1. Customer scans selected item.
2. System displays the item name, price, and cumulative total.
3. Customer repeats steps 1 and 2 for each item being purchased.
4. Customer selects payment.
5. System prompts for payment by cash, credit card, or debit card.
6. <payment>
7. System displays thank-you screen.

CS5373

39

39



Pay by Cash Extension Use Case

Use case name: Pay by Cash.

Summary: Customer pays by cash for items purchased.

Actor: Customer.

Dependency: Extends `Check Out Customer`.

Precondition: Customer has scanned items but not yet paid for them.

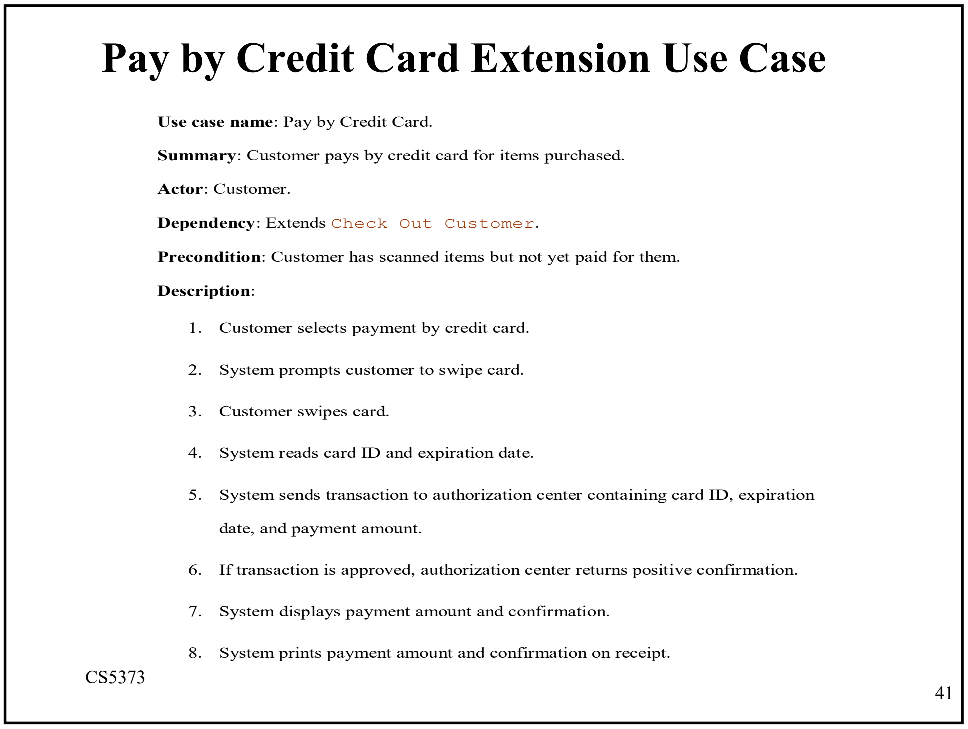
Description:

1. Customer selects payment by cash.
2. System prompts customer to deposit cash in bills and/or coins.
3. Customer enters cash amount.
4. System computes change.
5. System displays total amount due, cash payment, and change.
6. System prints total amount due, cash payment, and change on receipt.

CS5373

40

40



Pay by Credit Card Extension Use Case

Use case name: Pay by Credit Card.

Summary: Customer pays by credit card for items purchased.

Actor: Customer.

Dependency: Extends `Check Out Customer`.

Precondition: Customer has scanned items but not yet paid for them.

Description:

1. Customer selects payment by credit card.
2. System prompts customer to swipe card.
3. Customer swipes card.
4. System reads card ID and expiration date.
5. System sends transaction to authorization center containing card ID, expiration date, and payment amount.
6. If transaction is approved, authorization center returns positive confirmation.
7. System displays payment amount and confirmation.
8. System prints payment amount and confirmation on receipt.