Object-Oriented Technology and UML

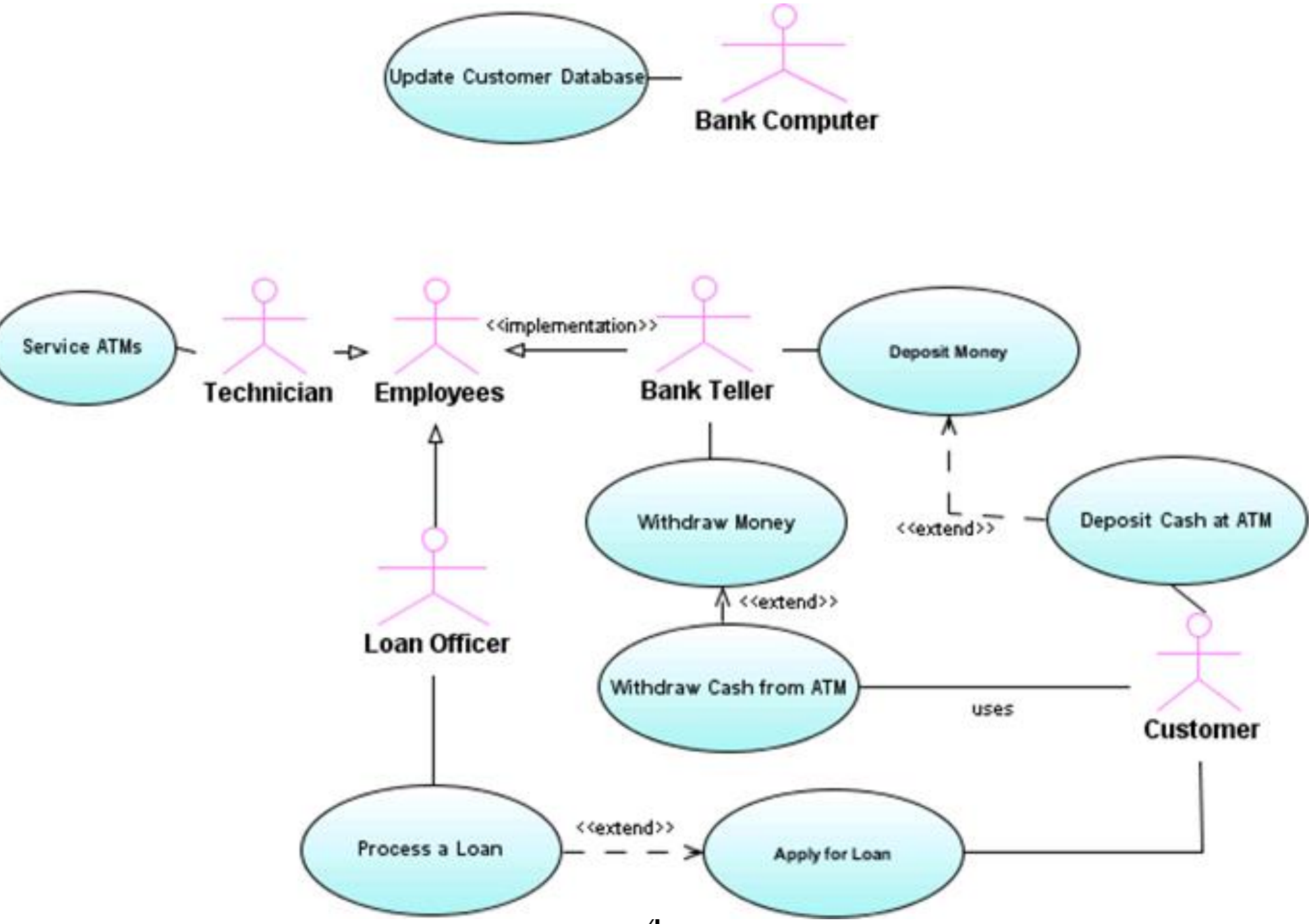
Use Case Modeling

Topics

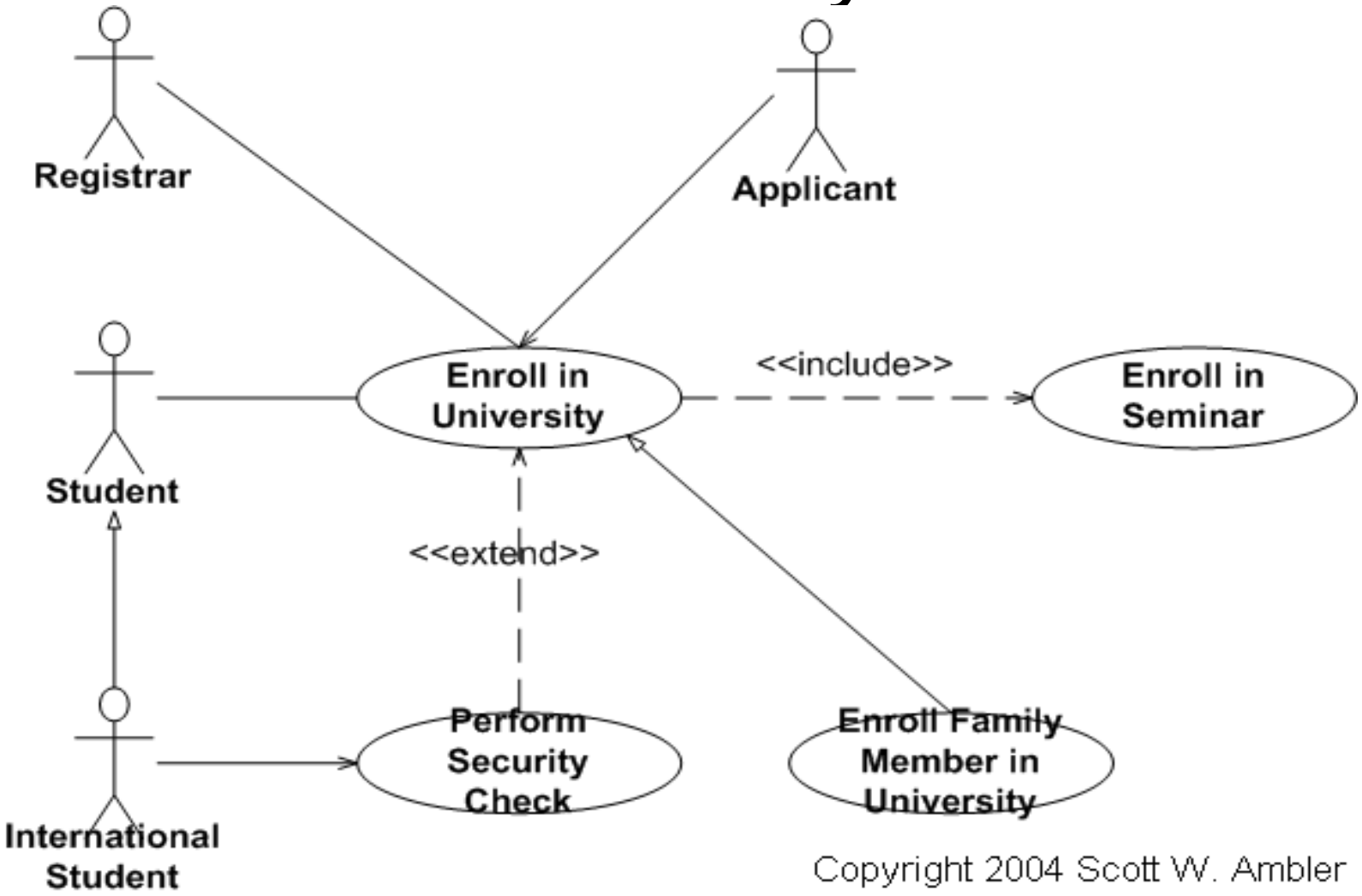
- Use case
- Use case diagram
- Use case modeling process
- Examples
- Styles of use case modeling

The Important Points

- Requirements technology
- Use case
- Use case diagram
- Actor
- Scenario
- The relationship between use cases
- Description of the use case


Update Customer Database
Bank Computer
Service ATMs
Technician
Employees
<<implementation>>
Bank Teller
Deposit Money
Withdraw Money
<<extend>>
Deposit Cash at ATM
Loan Officer
<<extend>>
Withdraw Cash from ATM
uses
Customer
Process a Loan
<<extend>>
Apply for Loan

Example of management system in university



Requirements technology

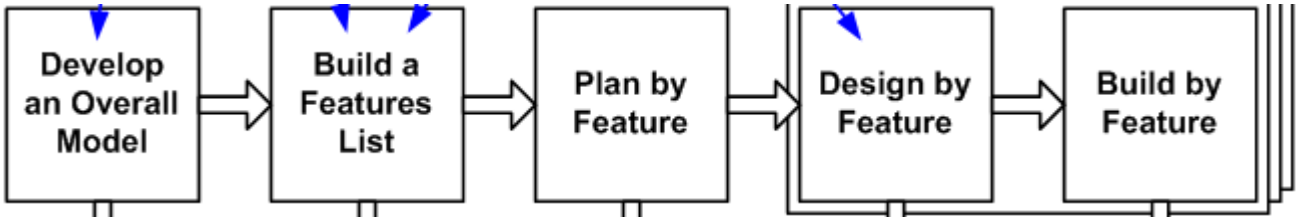
- Approximately 25 percent of failed projects due to the problems in requirements
- Two types of Requirements
  - Functional Requirements
    - The functions that the software should be able to perform
  - Non-Functional Requirements
    - Specific attributes or constraints in a system
- Three types of requirements technology
  - User story in extreme programming (XP)
  - Feature description in feature-driven development (FDD)
  - Use case in rational unified process (RUP)

User Story In Extreme Programming

- User stories
  - are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system
- A simple template
  - As a <type of user>, I want <some goal> so that <some reason>
- Examples
  - As a non-administrative user, I want to modify my own schedules but not the schedules of other users
  - Starting application with last edit
    - The application begins by bringing up the last document the user was working with
    - As a user, I want to start an application with the last edit

Feature description in feature-driven development (FDD)

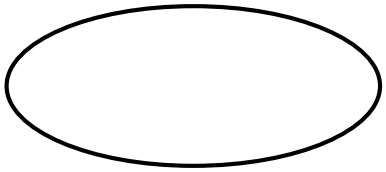
- FDD: Feature Driven Development



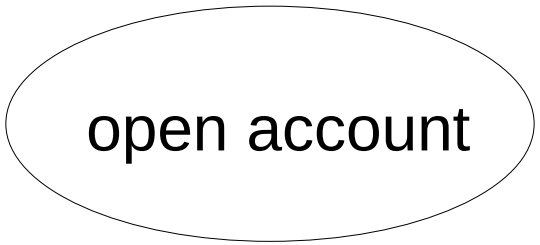
- Feature description
  - <action> the <result> <by|for|of|to> a(n) <object>
  - Calculate the total of a sale
  - Validate the password of a user
  - Retrieve the balance of a bank account

Use Case In Rational Unified Process (RUP)

- Use case is the basic characteristics of RUP
- Use case
  - A use case is a description of sequences of actions that a system performs that yield observable results of value to a particular actor
  - Use cases are denoted as ellipses or ovals



PurchaseTicket



Use Case Driven Software Development Process

- Use Case Model
  - Use case diagram…
- Analysis Model
  - Class diagram, Package diagram…
- Design Model
  - Class diagram, Package diagram, Object diagram, Component diagram…
- Deployment Model
  - Component diagram, Deployment diagram…
- Test Model
  - Construct the test model according to functions described in the use cases

Use Case

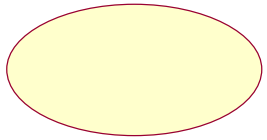
- Definition
  - Use cases are a means to capture the requirements of systems, i.e., what systems are supposed to do. A Use Case is a specification of behavior
  - Use cases define interactions between external actors and the system to attain particular goals

Use Case

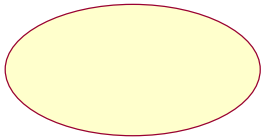
- Name
  - Every use case must have a name that distinguishes it from other use cases
  - A name is a textual string
  - In practice, use case names are short active verb phrases naming some behavior found in the vocabulary of the system you are modeling

Example of Use Case

- Word processing program
  - Set text in bold
  - Create index



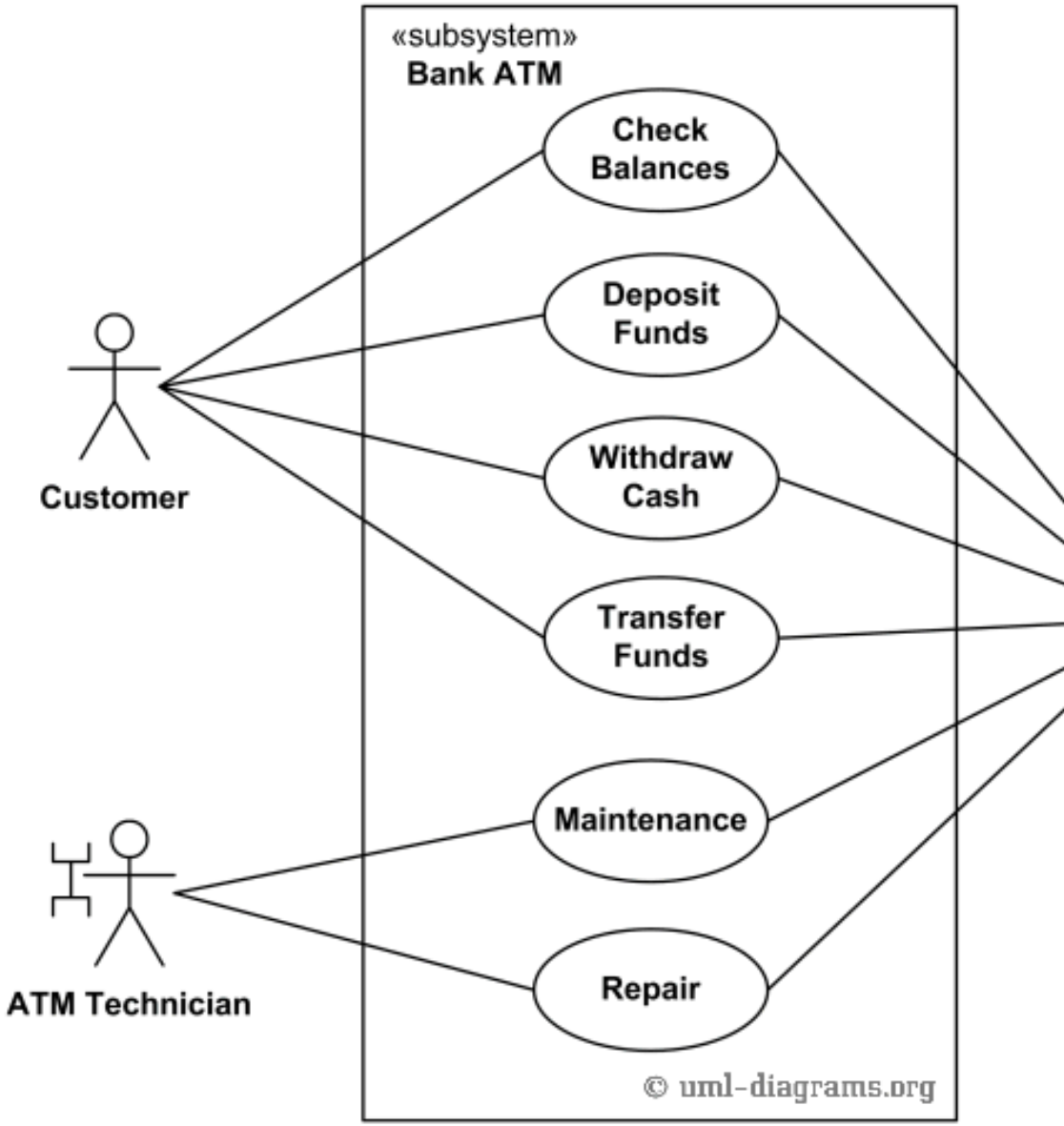
Set text in bold



Create index

Example of Use Case

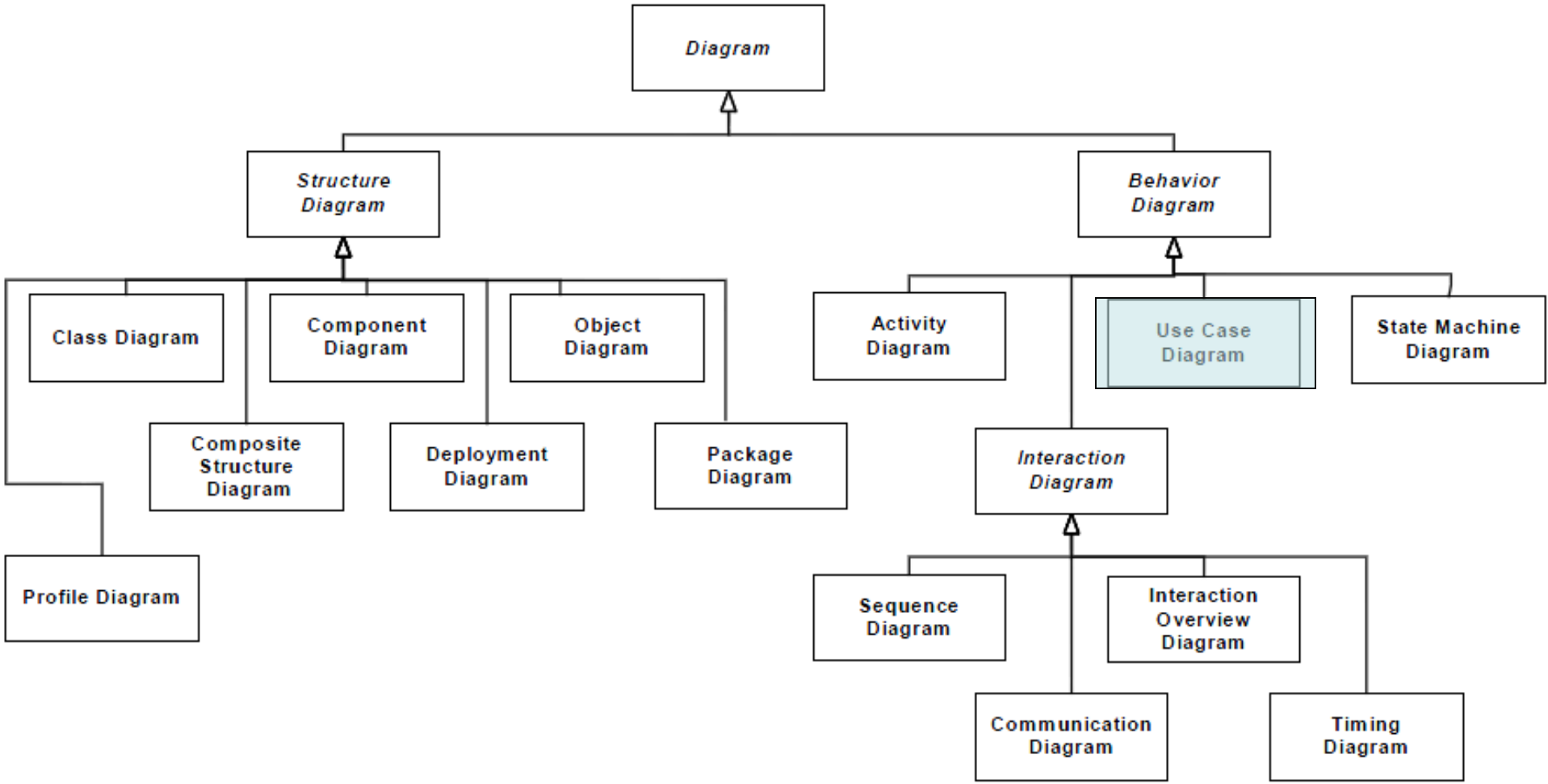
- Banking System



Use Case Based Requirement Analysis

- Use cases describe the system information from the perspective of using the system
- Use cases describe some visible requirements raised by users which correspond to specific user goals
- Use cases are descriptions of system behaviors, which belongs to the dynamic model of system

Diagrams of UML2.X



Considerations of using use case

- Use cases describe the visible requirements raised by users. Granularity of use case can be very flexible. A use case represents a specific goal
- A well-structured use case show only those use cases that are important to understand the behavior of the system or the part of the system in its context

Alistair Cockburn Requirements Outline

- Purpose and scope
- Terms / glossary
- **Use cases**
- Technology used
- Others
  - Participants, dependencies
  - Business rules / constraints
  - Performance demands
  - Security (now a hot topic), documentation
- Human issues: legal, political, organizational, training

Discussion about the use cases

- Use case analysis is a part of object oriented analysis and design?

- Anyway, use case diagram is a part of UML, determination of use cases is often the first step of object oriented development process

Use Cases and Collaborations

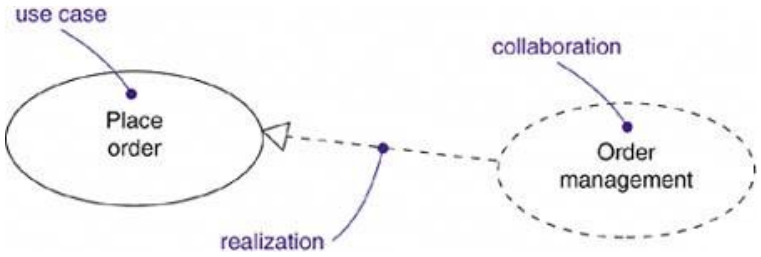
- A use case captures the intended behavior of the system (or subsystem, class, or interface) you are developing, without having to specify how that behavior is implemented
- Use cases are implementation-independent description of system functions
- Ultimately, however, you have to implement your use cases, and you do so by creating a society of classes and other elements that work together to implement the behavior of this use case. This society of elements, including both its static and dynamic structure, is modeled in the UML as a collaboration
- In general, every use case should be realized by one or more collaborations

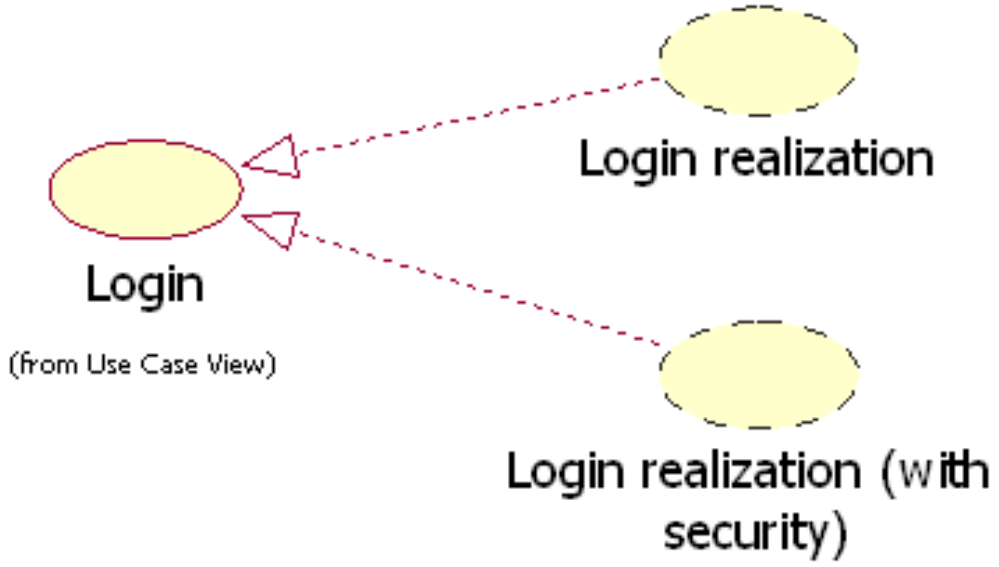
Use Cases and Collaborations

- A collaboration is a society of classes, interfaces, and other elements that work together to provide some cooperative behavior that's bigger than the sum of all its parts
- A collaboration is also the specification of how an element, such as a classifier (including a class, interface, component, node, or use case) or an operation, is realized by a set of classifiers and associations playing specific roles used in a specific way

Use Cases and Collaborations

- Graphically, a collaboration is rendered as an ellipse with dashed lines





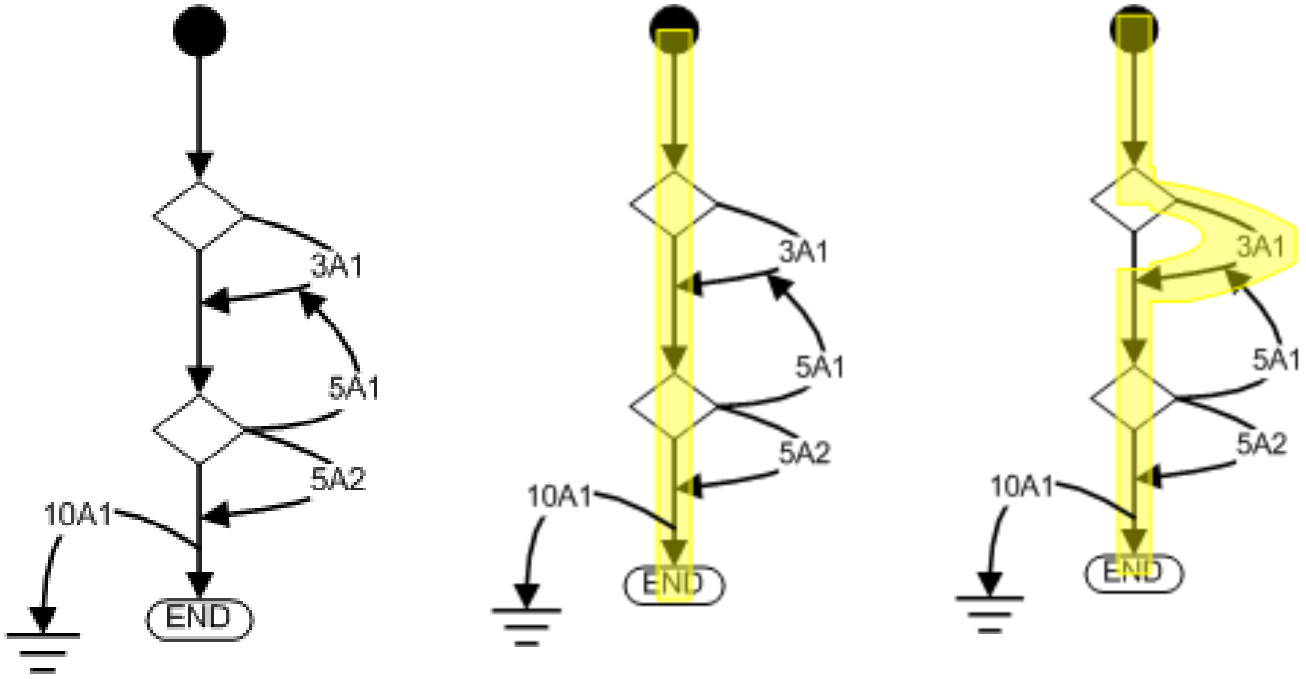
Use Cases and Scenarios

- It is desirable to separate main versus alternative flows because a use case describes a set of sequences, not just a single sequence, and it would be impossible to express all the details of an interesting use case in just one sequence
- Each sequence is called a scenario
- A scenario is a specific sequence of actions that illustrates behavior

Use Cases and Scenarios

- Scenarios are to use cases as instances are to classes, meaning that a scenario is basically one instance of a use case
- A modestly complex system might have a few dozen use cases that capture its behavior, and each use case might expand out to several dozen scenarios
- For each use case, you'll find primary scenarios (which define essential sequences) and secondary scenarios (which define alternative sequences)

Use Cases and Scenarios



Use Cases and Scenarios

- There is no predefined UML notation for expressing use case scenarios

Use Cases and Scenarios

- Basic Idea
  - If Use Case = Human Life, then Scenario = My Own Life.
- Basic Idea
  - If Use Case = Do Assignment, then Scenario = My Own Way to Do My Assignment.

Elements of Use Case Diagram

- Use case
- Actor
- Relation

Actor

- An actor represents a coherent set of roles that users of use cases play when interacting with these use cases
- Typically, an actor represents a role that a human, a hardware device, or even another system plays with a system

Possible Actors in Banking System

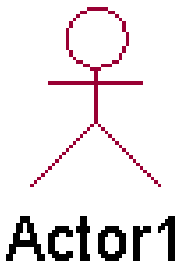
- Customer
- LoanOfficer
- MailSystem

Actor

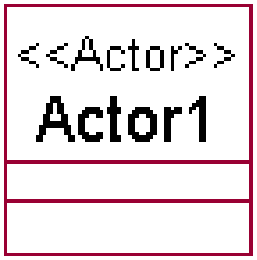
- An instance of an actor, therefore, represents an individual interacting with the system in a specific way
- Although you'll use actors in your models, actors are not actually part of the software application. They live outside the application within the surrounding environment

Actor

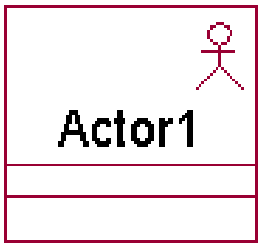
- Actors are rendered as stick figures
  - You can use the UML's extensibility mechanisms to stereotype an actor in order to provide a different icon that might offer a better visual cue for your purposes



Icon



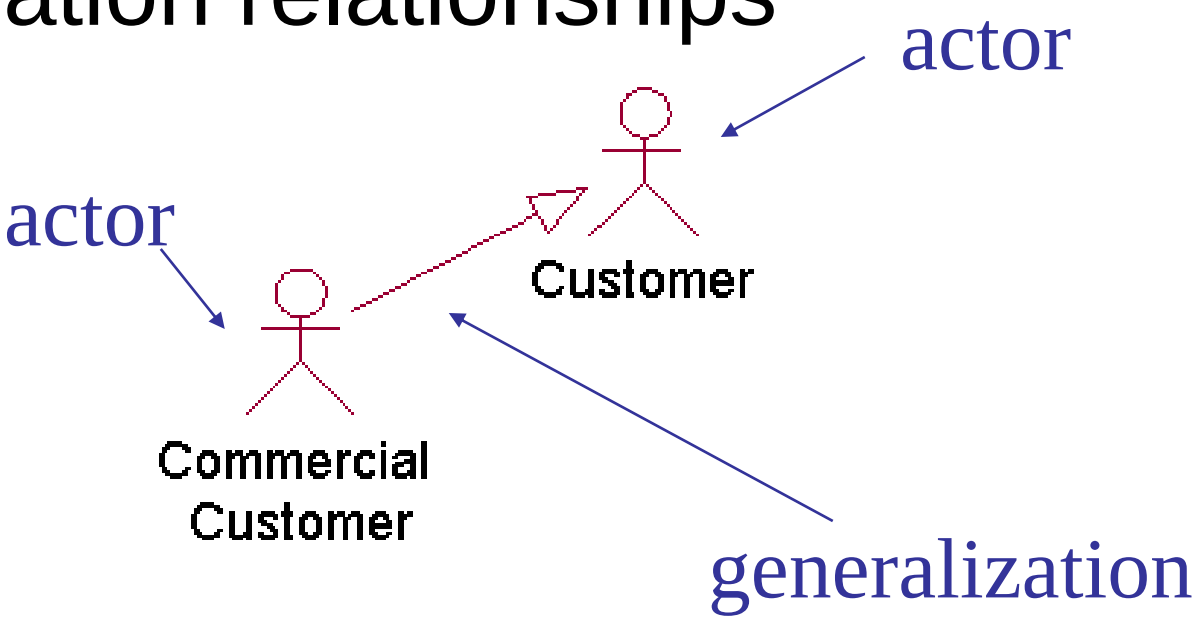
Label



Decoration

Generalization Relationships Between Actors

- You can define general kinds of actors (such as Customer) and specialize them (such as CommercialCustomer) using generalization relationships



Useful Questions in Finding Actors

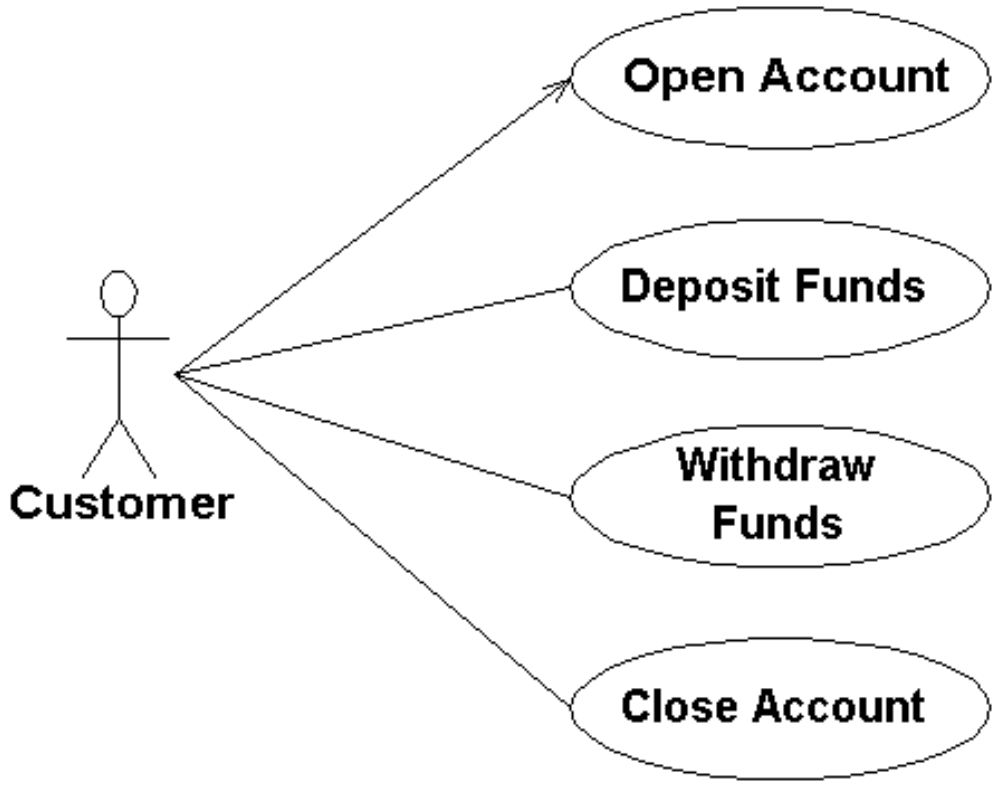
- Who will supply, use, or remove information?
- Who will use this functionality?
- Who is interested in a certain requirement?
- Where in the organization is the system used?
- Who will support and maintain the system?
- What are the system's external resources?
- What other systems will need to interact with this one?

Relationships between Actors And Use Cases

- Actors may be connected to use cases only by association
- An association between an actor and a use case indicates that the actor and the use case communicate with one another, each one possibly sending and receiving messages

Relationships between Actors And Use Cases

- Example



Relationships between Use Cases

- Relationships
  - Generalization
  - Include
  - Extend

Generalization

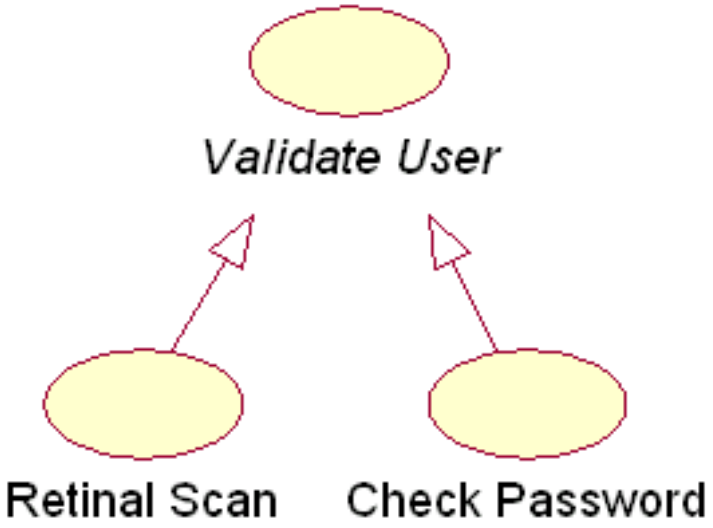
- Generalization among use cases is just like generalization among classes
  - Here it means that the child use case inherits the behavior and meaning of the parent use case
  - The child may add to or override the behavior of its parent; and the child may be substituted any place the parent appears

Generalization

- 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

Generalization

- Generalization among use cases is rendered as a solid directed line with a large triangular arrowhead, just like generalization among classes
- Example

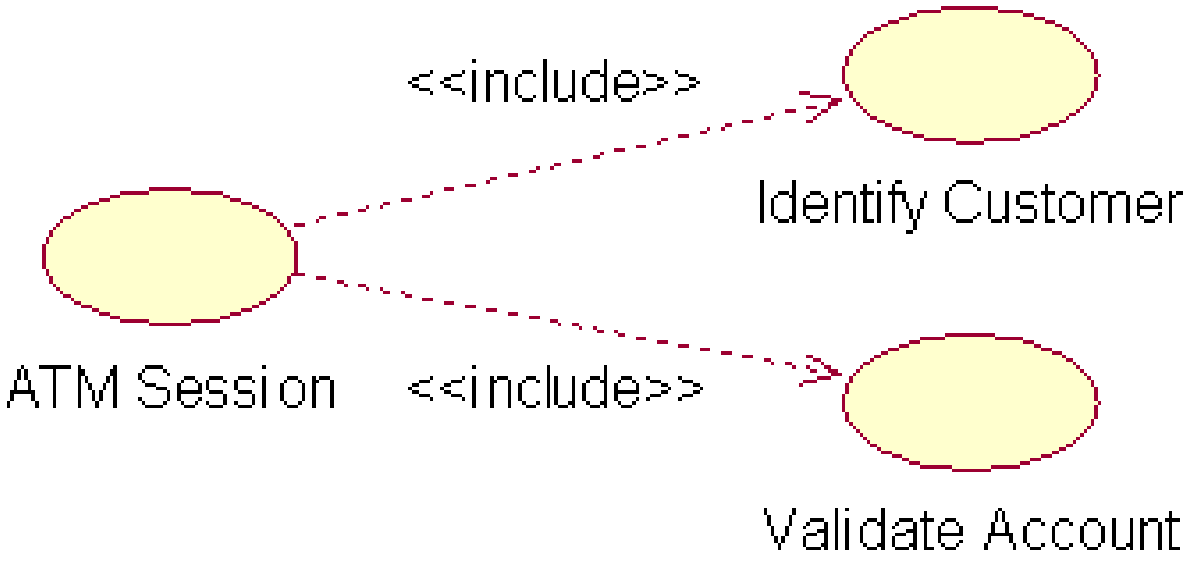


Include

- 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

Include

- An include relationship is rendered as a dependency, stereotyped as include
- Example



Include

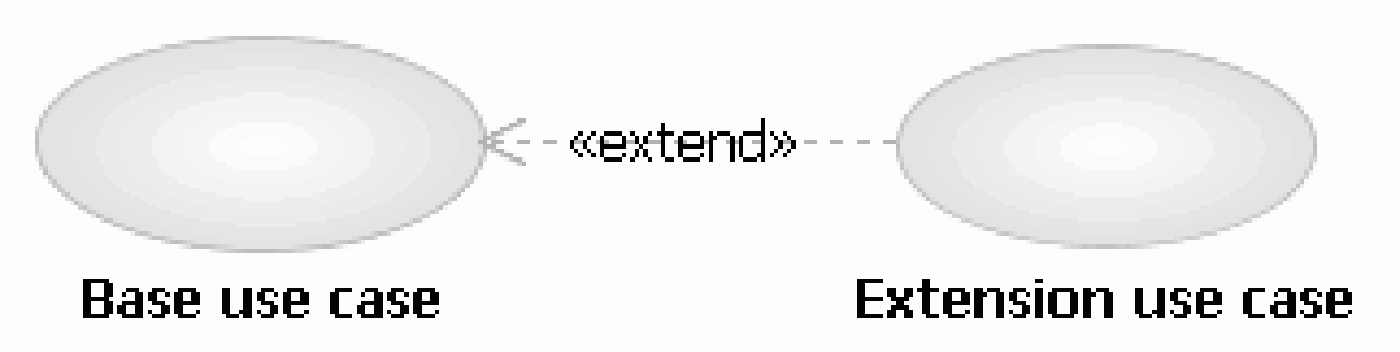
- The direction of the arrow
  - From the base use case to the included use case

Extend

- 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 the extending use case
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case
- This base use case may be extended only at certain points called, not surprisingly, its extension points

Extend

- An extend relationship is rendered as a dependency, stereotyped as extend

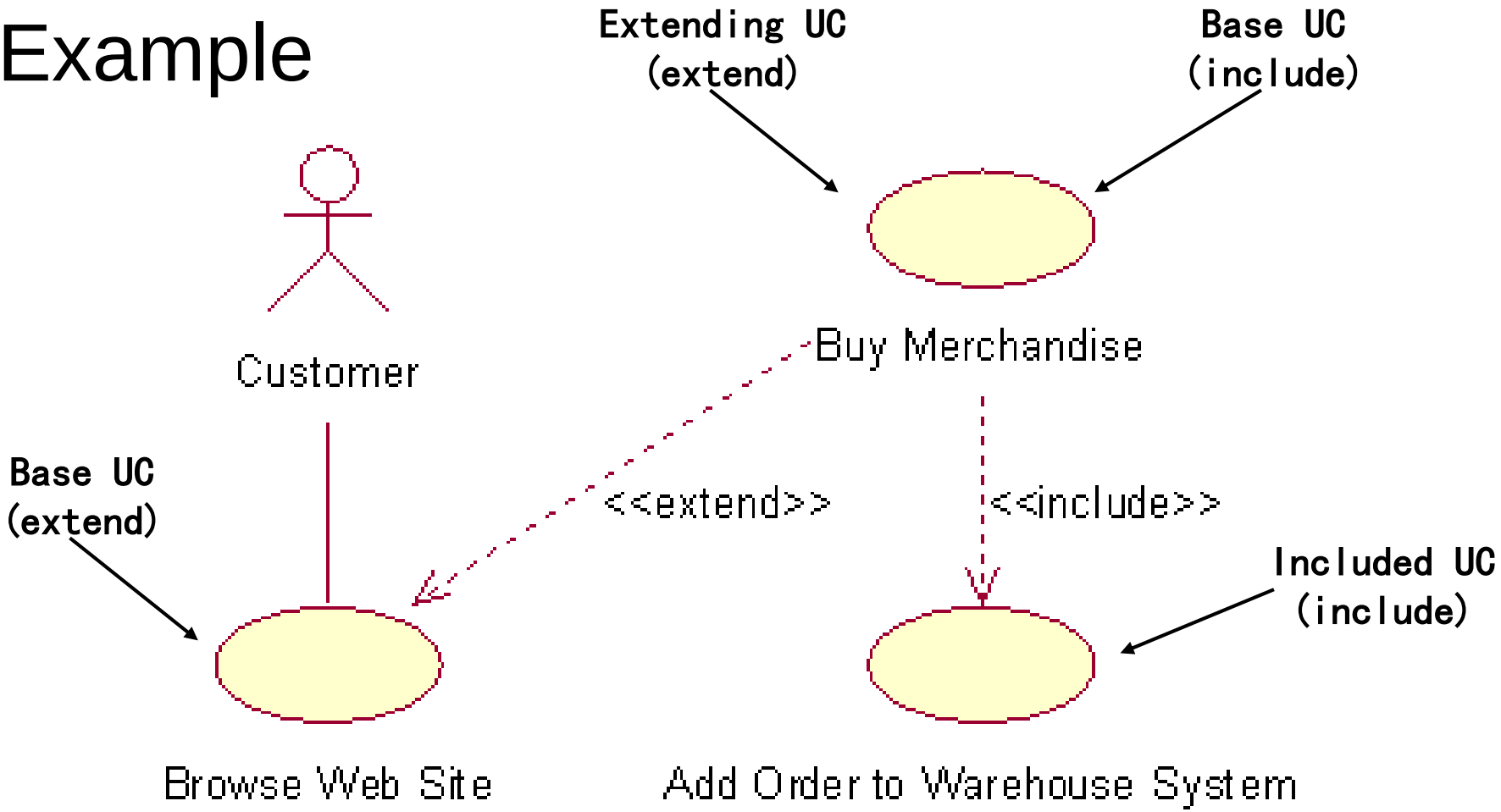


Extend

- Example

Relationships between Use Cases

- Example



Relationships between Use Cases

- “is a”
  - Genernalization
  - Extend
- “has a”
  - include
- Difference of base use case between include and extend relations

Relationships among Actors And Use Cases

Relation	Description	Symbol
Association	Between actor and use case	
Generalization	Between actors, between use cases	
Include	Between use cases	
Extend	Between use cases	

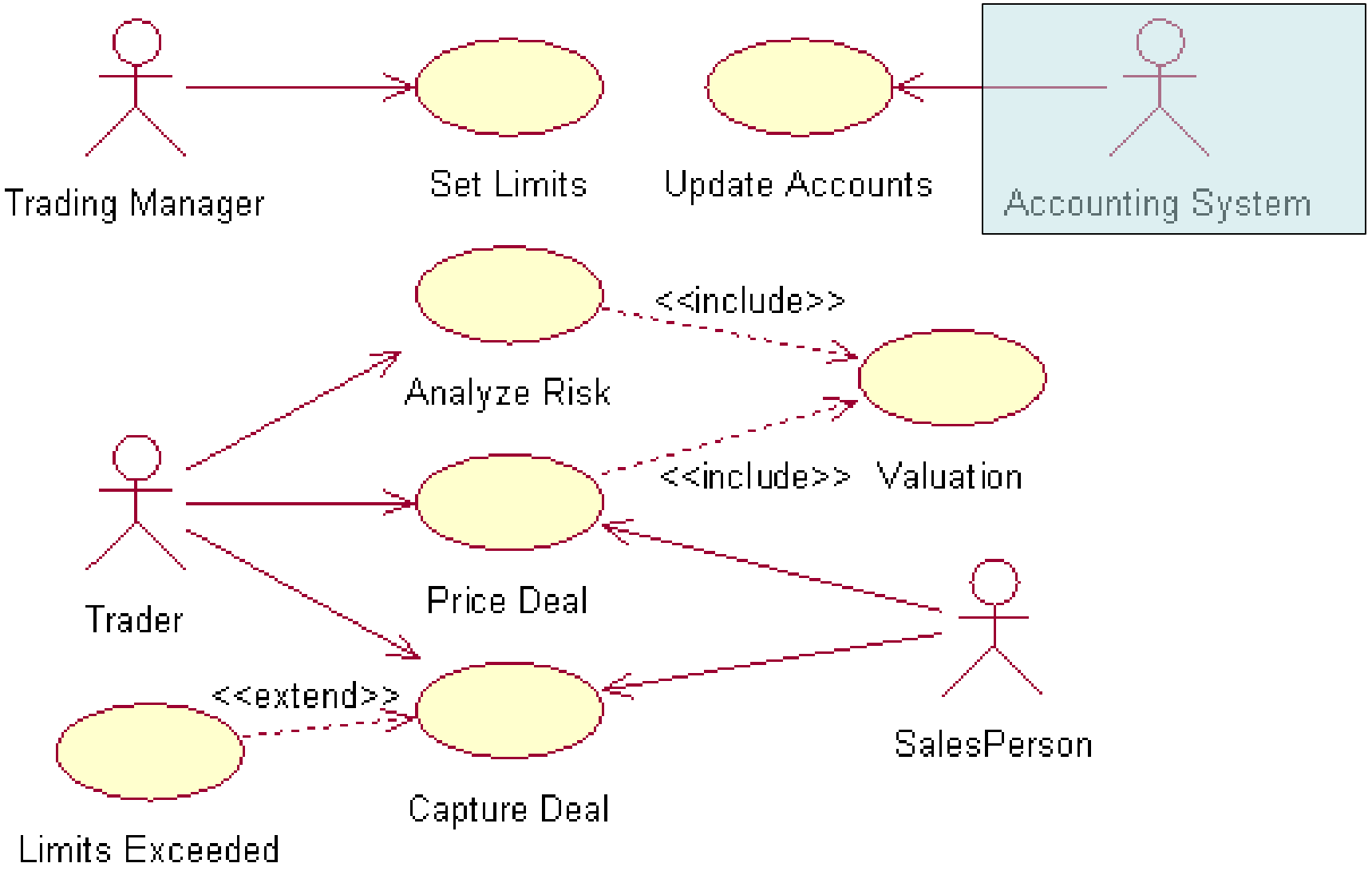
Relationships in Use Case Diagram

- Relationship, association, generalization, dependency
  - Association, generalization, and dependency are relationships
  - Include, extend are special forms of dependency

Use Case Diagram

- Use Case Diagram
  - Use case diagrams are one of the diagrams in the UML for modeling the dynamic aspects of systems
  - Use case diagrams are central to modeling the behavior of a system, a subsystem, or a class
  - Each one shows a set of use cases and actors and their relationships
  - In UML, a use case model is usually described by several use case diagrams
  - Elements
    - Use case
    - Actor
    - Relationship

Example of Use Case Diagram



Example of Use Case Diagram

Use Case Specification

- A use case specification provides textual detail for a use case

Fragment of Use Case Specification

- The Withdraw Cash use case for an Automated Teller Machine
  - Basic Flow of Events
    - 1. The use case begins when Bank Customer inserts their Bank Card.
    - 2. Use Case: Validate User is performed.
    - 3. The ATM displays the different alternatives that are available on this unit. [See Supporting Requirement SR-xxx for list of alternatives]. In this case the Bank Customer always selects "Withdraw Cash".
    - …
    - 11. The receipt is printed.
    - 12. The use case ends successfully.
  - Alternative Flows (Invalid User)
    - If in step 2 of the basic flow Bank Customer the use case: Validate User does not complete this successfully, then
    - 1. The use case ends with a failure
  - Alternative Flows (Wrong account)
    - If in step 8 of the basic flow the account selected by the Bank Customer is not associated with this bank card, then
    - 1. The ATM shall display the message "Invalid Account – please try again".
    - 2. The use case resumes at step 4.
  - …

Use Case Specification

- A use case specification is a document used to capture the specific details of a use case
- Use case specifications provide a way to capture the functional requirements of a system
- Use case specifications provide a means of organizing all of the different scenarios that exist. They add detail beyond what is shown in a use case diagram. They are a useful tool in communicating with project stakeholders, system users, business analysts, and developers

A Template Of Use Case Specification

- Use case name
  - States the use case name
- Brief description
  - Describes the role and purpose of the use case
- Flow of events
  - Presents the basic flow and alternative flows

A Template Of Use Case Specification

- Basic flow
  - Describes the ideal, primary behavior of the system
- Alternative flows
  - Describes exceptions or deviations from the basic flow, such as how the system behaves when the actor enters an incorrect user ID and the user authentication fails

A Template Of Use Case Specification

- Special requirements
  - A nonfunctional requirement that is specific to a use case but is not specified in the text of the use case flow of events
  - Examples of special requirements include: legal and regulatory requirements; application standards; quality attributes of the system, including usability, reliability, performance, and supportability; operating systems and environments; compatibility requirements; and design constraints

A Template Of Use Case Specification

- Preconditions
  - A state of the system that must be present before a use case is performed
- Post conditions
  - A list of possible states for the system immediately after a use case is finished
- Extension points
  - A point in the use-case flow of events at which another use case is referenced

Example: Withdraw Cash use case for an Automated Teller Machine

- 1 Brief Description
  - This use case describes how the Bank Customer uses the ATM to withdraw money to his/her bank account.
- 2 Actors
  - 2.1 Bank Customer
  - 2.2 Bank
- 3 Preconditions
  - There is an active network connection to the Bank.
  - The ATM has cash available.

Example: Withdraw Cash use case for an Automated Teller Machine

- 4 Basic Flow of Events
  - 1. The use case begins when Bank Customer inserts their Bank Card.
  - 2. Use Case: Validate User is performed.
  - 3. The ATM displays the different alternatives that are available on this unit. [See Supporting Requirement SR-xxx for list of alternatives]. In this case the Bank Customer always selects “Withdraw Cash”.
  - 4. The ATM prompts for an account. See Supporting Requirement SR-yyy for account types that shall be supported.
  - 5. The Bank Customer selects an account.
  - 6. The ATM prompts for an amount.
  - 7. The Bank Customer enters an amount.
  - 8. Card ID, PIN, amount and account is sent to Bank as a transaction. The Bank Consortium replies with a go/no go reply telling if the transaction is ok.
  - 9. Then money is dispensed.
  - 10. The Bank Card is returned.
  - 11. The receipt is printed.
  - 12. The use case ends successfully.

Example: Withdraw Cash use case for an Automated Teller Machine

- 5 Alternative Flows
  - 5.1 Invalid User…
  - 5.2 Wrong account…
  - 5.3 Wrong amount …
  - 5.4 Amount Exceeds Withdrawal Limit …
  - 5.5 Amount Exceeds Daily Withdrawal Limit …
  - 5.6 Insufficient Cash …
  - 5.7 No Response from Bank …
  - 5.8 Money Not Removed …
  - 5.9 Quit …

Example: Withdraw Cash use case for an Automated Teller Machine

- 6 Key Scenarios
  - 6.1 No Response from Bank
- 7 Post-conditions
  - 7.1 Successful Completion
    - The user has received their cash and the internal logs have been updated.
  - 7.2 Failure Condition
    - The logs have been updated accordingly.

Example: Withdraw Cash use case for an Automated Teller Machine

- 8 Special Requirements
  - The ATM shall dispense cash in multiples of $20.
  - The maximum individual withdrawal is $500.
  - The ATM shall keep a log, including date and time, of all complete and incomplete transactions with the Bank.

Some mistake in Use Case Specification

- Only describe the behavior of the system, there is no description of the behavior of the actors
- Only describe the behavior of the actors, there is no description of the behavior of the system
- Description is too long

The way to find use cases

- To consider what each actor requires of the system.
- For each actor, human or not, ask yourself the following questions
  - What are the primary tasks the actor wants the system to perform?
  - Will the actor create, store, change, remove, or read data in the system?
  - Will the actor need to inform the system about sudden, external changes?
  - Does the actor need to be informed about certain occurrences in the system?
  - Will the actor perform a system start-up or shutdown?

Modeling the Requirements of a System

- Establish the context of the system by identifying the actors
- For each actor, consider the behavior that each expects or requires the system to provide
- Name these common behaviors as use cases
- Factor common behavior into use cases that are used by others, factor variant behavior into new use cases that extend more main line flows
- Model these use cases, actors, and their relationships in a use case diagram
- Adorn these elements with any constraints that assert nonfunctional requirements; you may have to attach some of these to the whole system

identifying the actors

identifying the use cases

Draw a use case diagram, refining the description

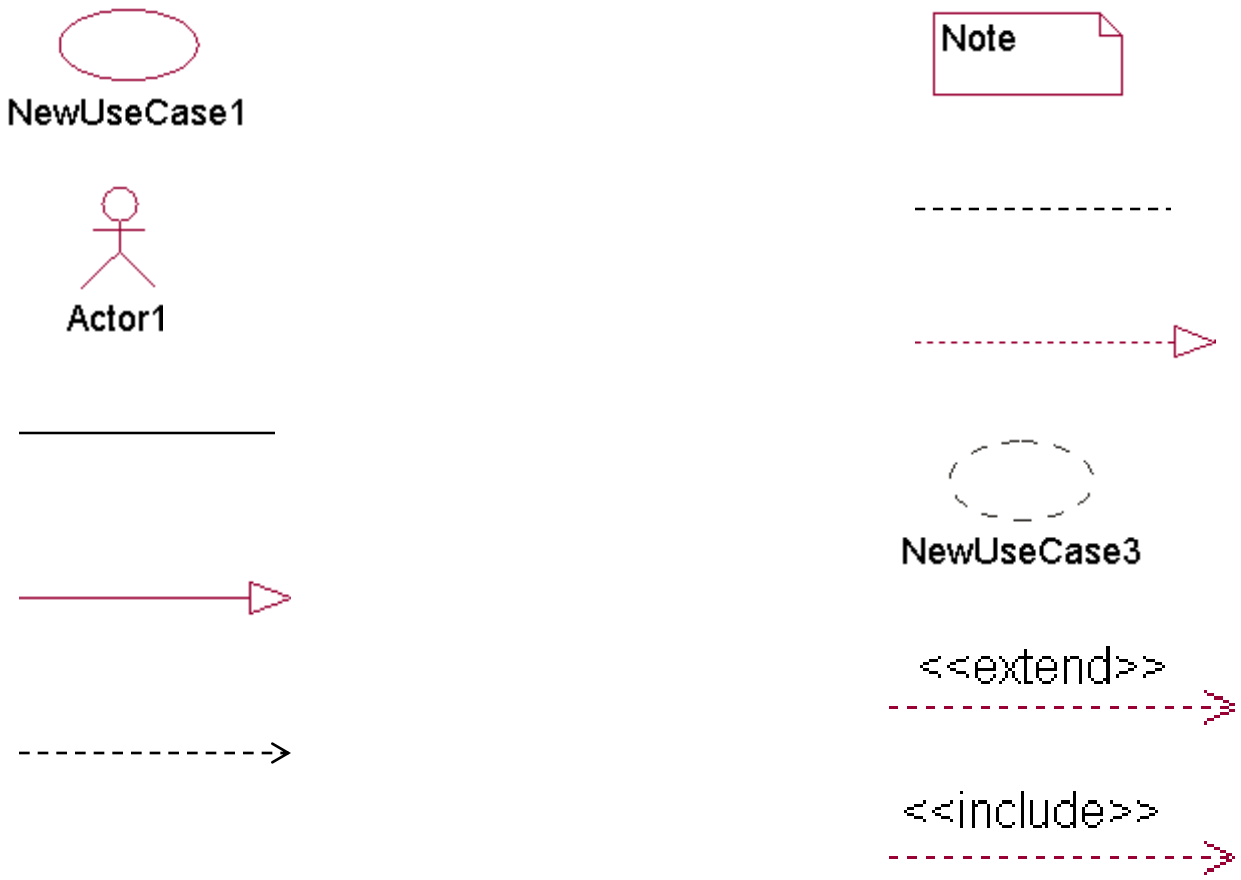
FAQ in Use Case Modeling

- Granularity of use cases
  - There is no conclusion

FAQ in Use Case Modeling

- Decomposition and merge of use cases

The Main Symbols in Use Case Diagram

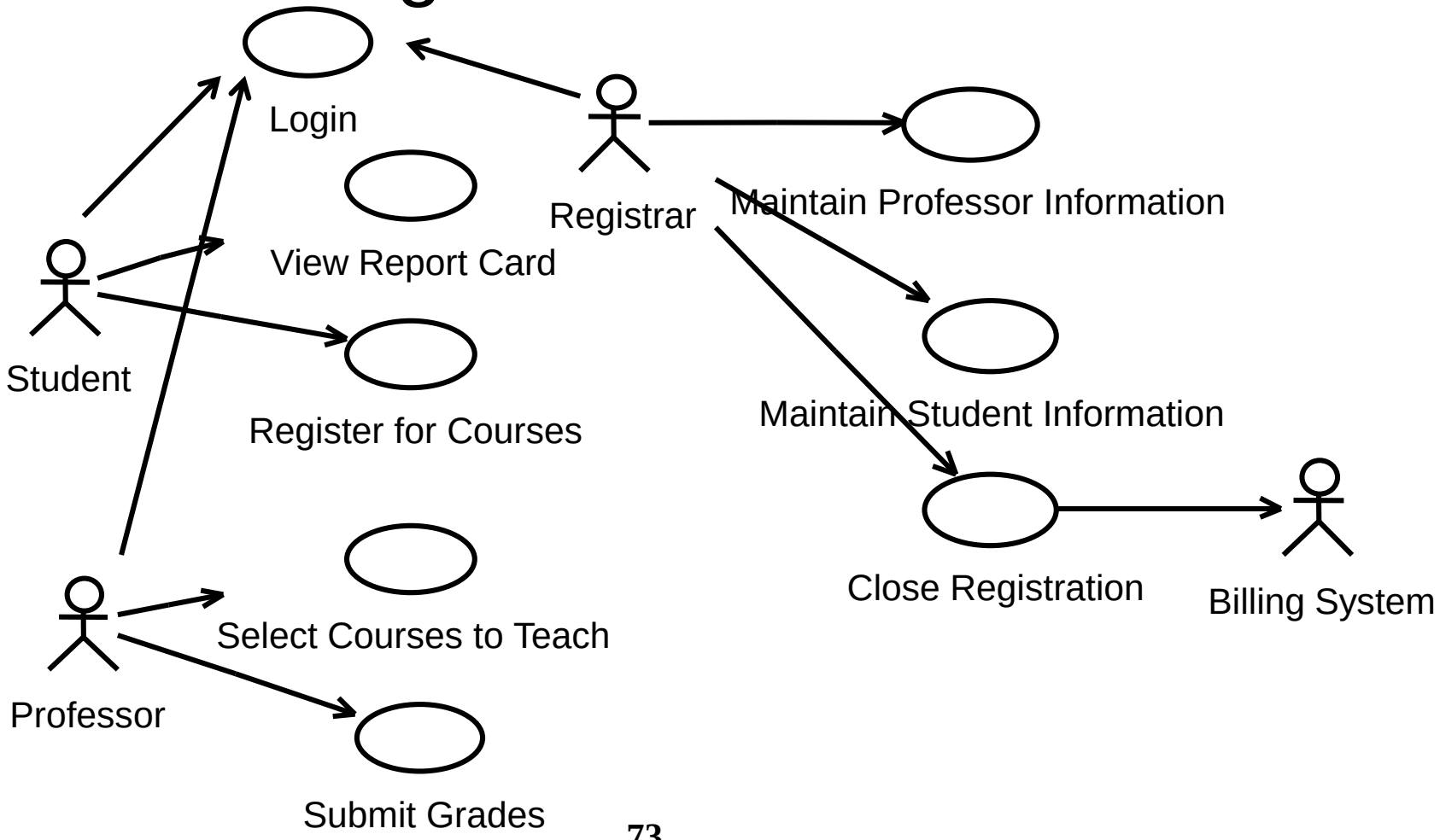


Example: Course Registration System

- Actors
  - ...
- Use cases
  - ...

Example: Course Registration System

- Use case diagram

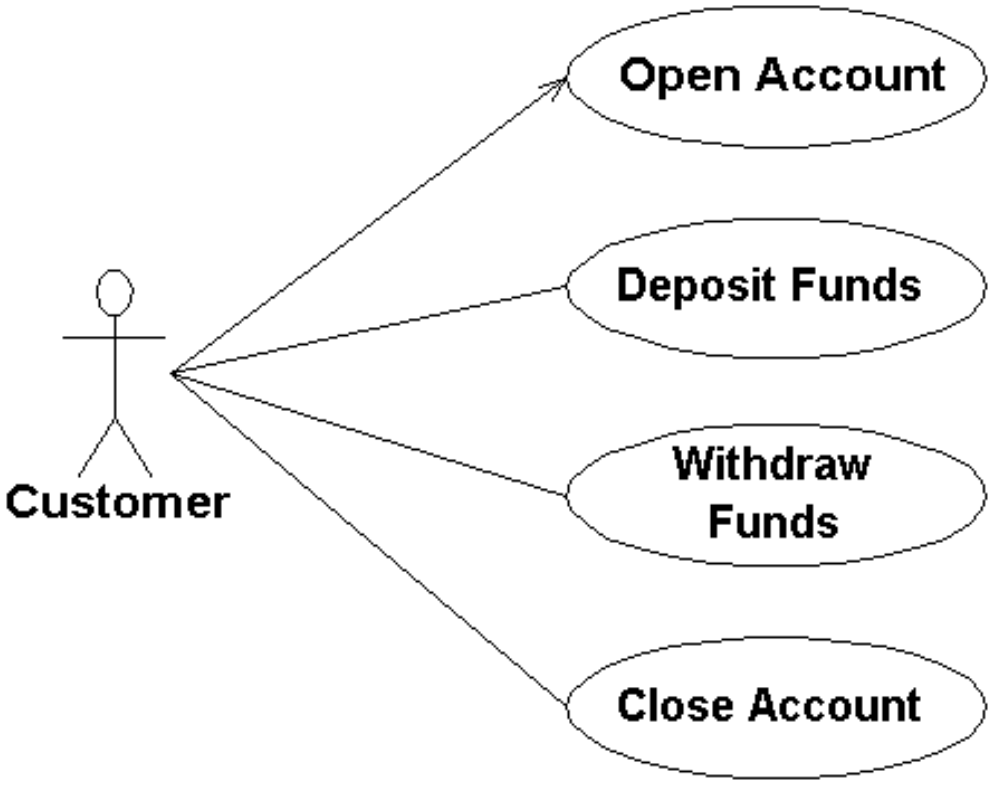


Style of UML Use Case Diagrams

- Begin Use-Case Names with a Strong Verb
  - Use-case names beginning with weak verbs such as “process,” “perform,” and “do” are often problematic
  - Good use-case names
    - Withdraw Funds
    - Register Student in Seminar
    - Deliver Shipment

Style of UML Use Case Diagrams

- Imply Timing Considerations by Stacking Use Cases

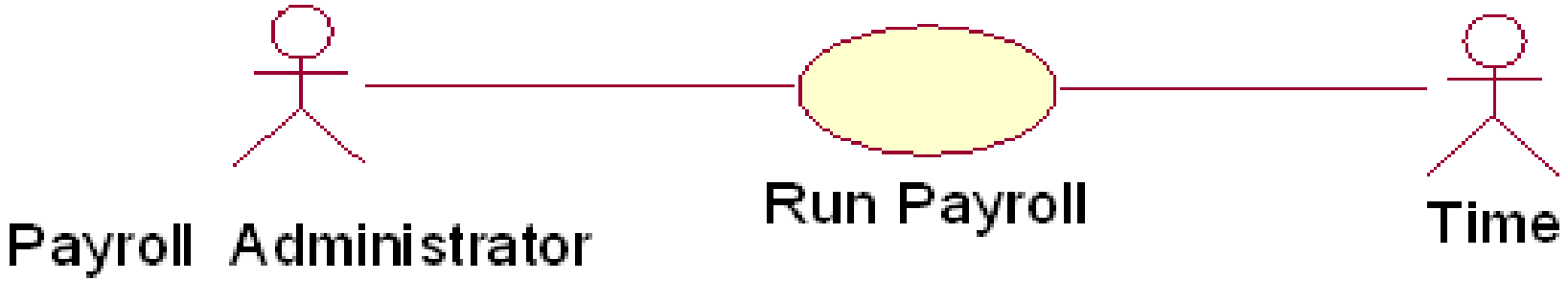


Style of UML Use Case Diagrams

- Name Actors with Singular, Domain-Relevant Nouns
  - An actor should have a name that accurately reflects its role within your model
  - Actor names are usually singular nouns such as Grade Administrator, Customer, and Payment Processor

Style of UML Use Case Diagrams

- Place Your Primary Actor(s) in the Top Left Corner of the Diagram
- Draw Actors on the Outside Edges of a Use Case Diagram
- Name Actors with Singular, Domain-Relevant Nouns



Style of UML Use Case Diagrams

- Do Not Apply <<uses>>, <<includes>>, or <<extends>>
  - All three of these stereotypes were supported by earlier versions of the UML but over time they have been replaced