# *A brief introduction to UML use case modeling*

**Professor Hossein Saiedian**

EECS 448: Software Engineering

Fall 2022

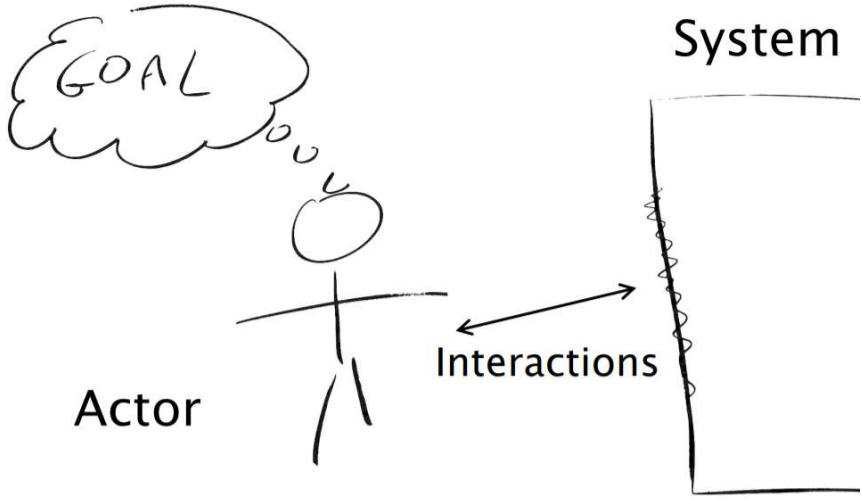
KU THE UNIVERSITY OF KANSAS

# Use cases modeling

- Objective: Discover and document the functional requirements of the software product (or system)
  - Use cases capture interactions between the system actors to achieve user goals
  - Use cases capture who (actor) does what (interaction) with the system, with what purpose (goal) without dealing with system internals
  - A complete set of use cases specifies all the different ways to achieve a goal with the system, and thus defines all behavior required of the system
- Designed to be understood by non-technical parties

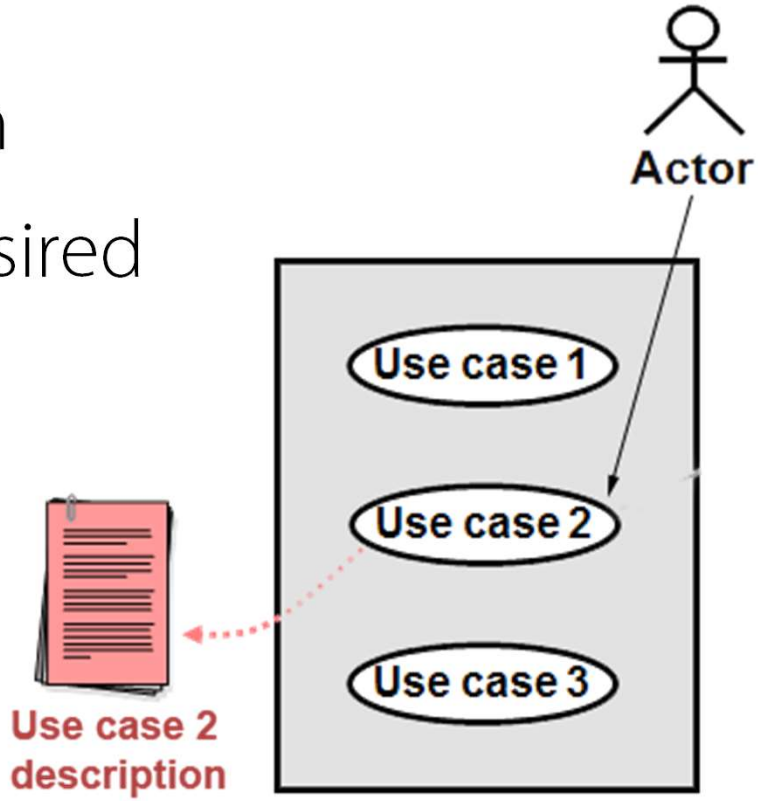
# Use case modeling at a glance

- A use case is a sequence of transactions performed by a system, which yields an observable result of value for a particular actor
  - The system being used (treated as a black-box)
  - The *actor* involved (type of user that interacts with the system)
  - The functional goal that the actor achieves using the system

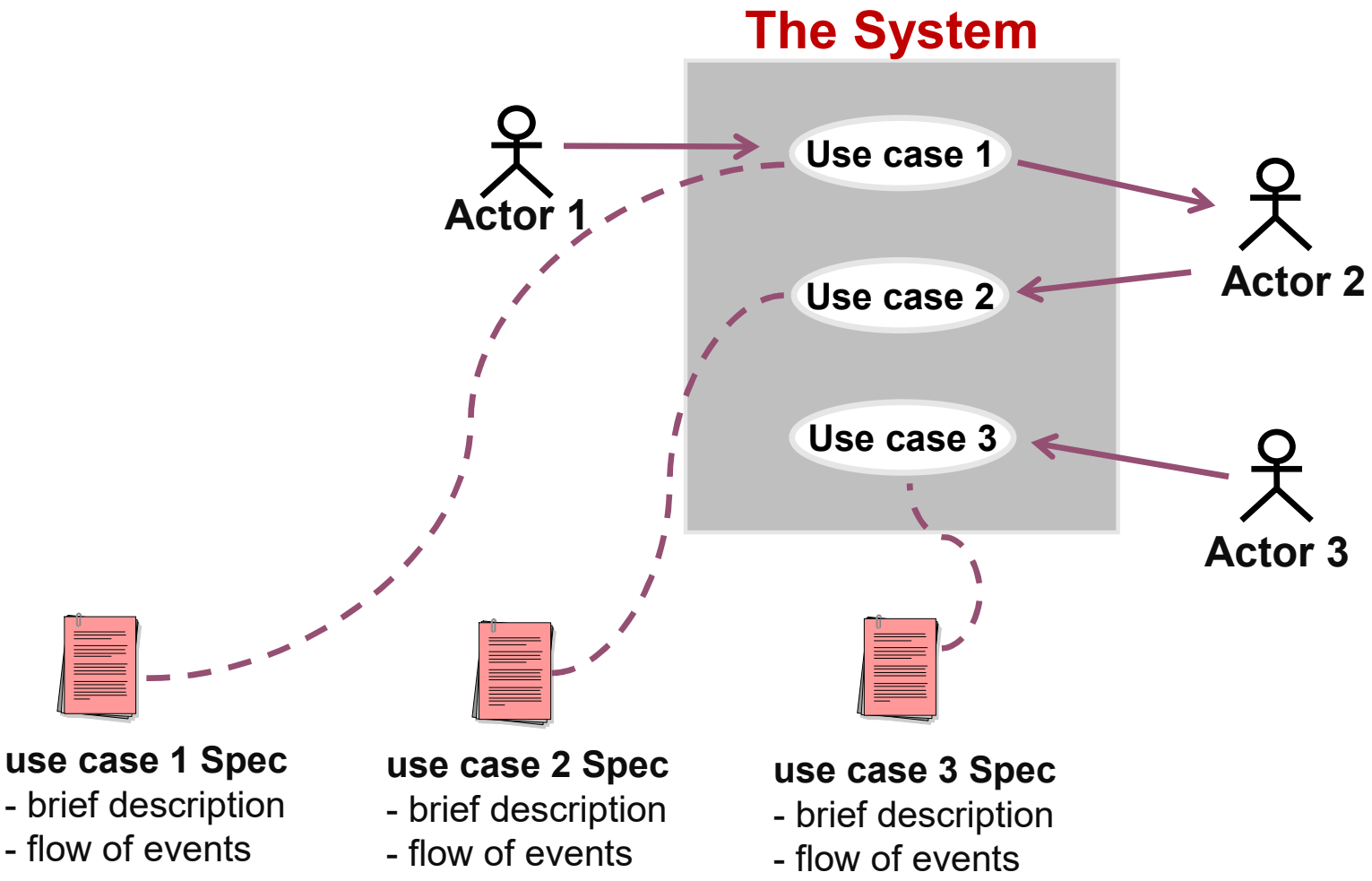


# Use case modeling

- Identifies who or what interacts with the system
  - Connects the stakeholder needs to the software requirements
- Defines clear boundaries of a system
- Captures and communicates the desired behavior of a system



# A use case model



# Definitions: Actor and use case

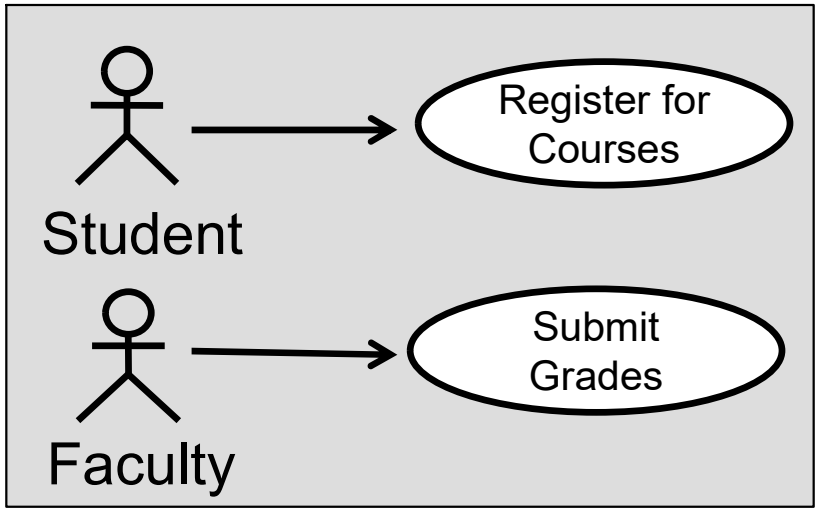
- **Actor**: someone/something outside the system, acting in a role that interacts with the system
  - Who/what uses the system
- **Use case**: a sequence of actions performed by a system that yields an observable result of value to an actor (achieves a goal)

# Actors represent roles

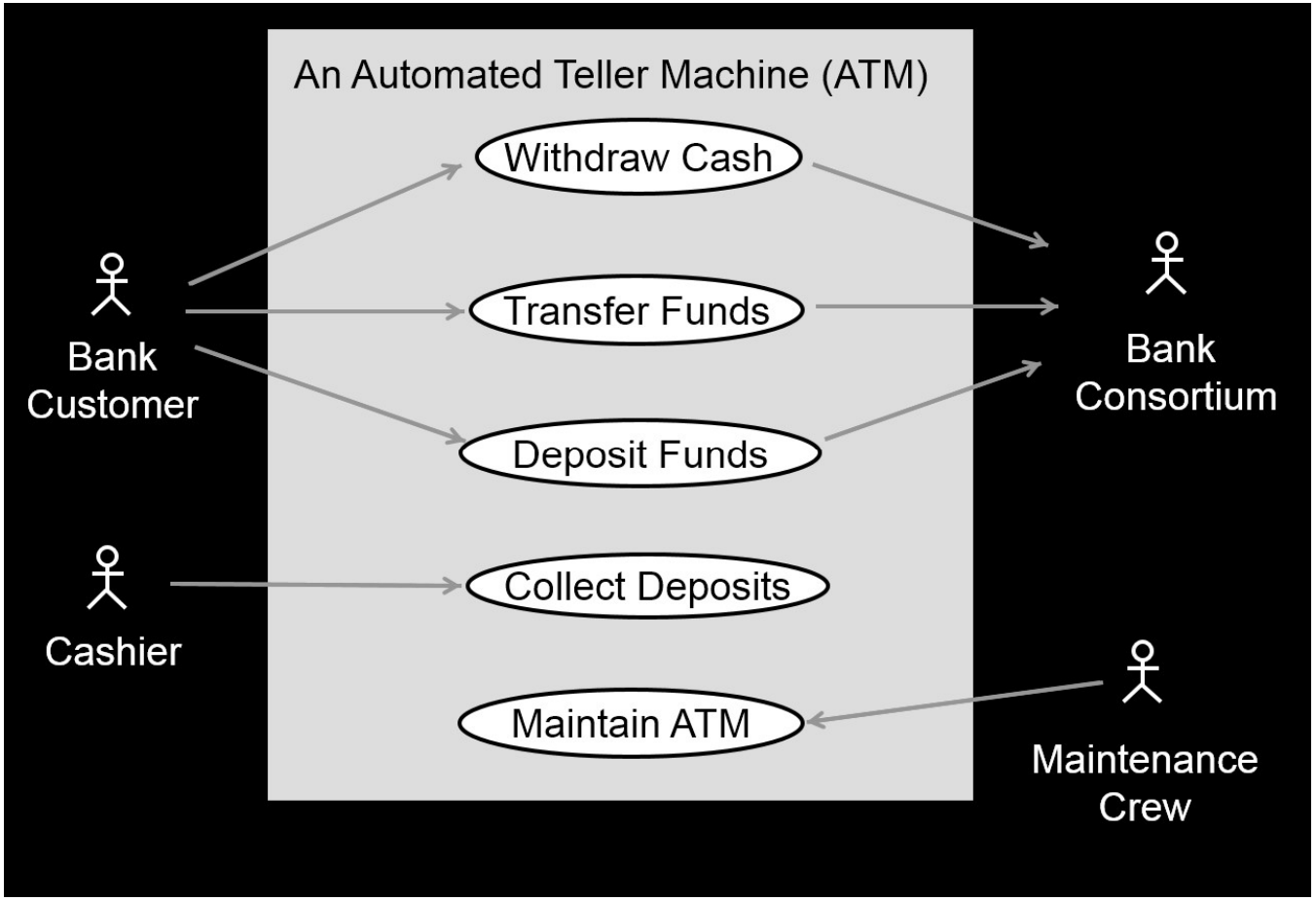
- John: an EECS Ph.D. student and an CS faculty
- Sara: a science undergraduate student

John and Sara act as a student

John acts as a student



# A banking example



# How to find actors

- Look for external entities that interact with the system
  - Which persons interact with the system (directly or indirectly)? Don't forget maintenance staff!
  - Will the system need to interact with other systems or existing legacy systems?
  - Are there any other hardware or software devices that interact with the system?

- Primary and secondary (supporting) actors
  - A primary actor has a goal and initiates an interaction

# Use cases

- Use cases describe the sequence of interactions between actors and the system
  - To deliver the service that satisfies the goal
  - Use cases include possible variants of this sequence
  - Either successful or fail

# Goals

- The use case describes only what is the relationship of the actor to the system
  - The goal must be of value to the (primary) actor:
  - "Withdraw cash" is goal
  - "Enter PIN code" is not a goal

- As a <**actor type**> I want <**to do something**> so that <**some value is created**>

*bank customer*

*to perform a withdrawal*

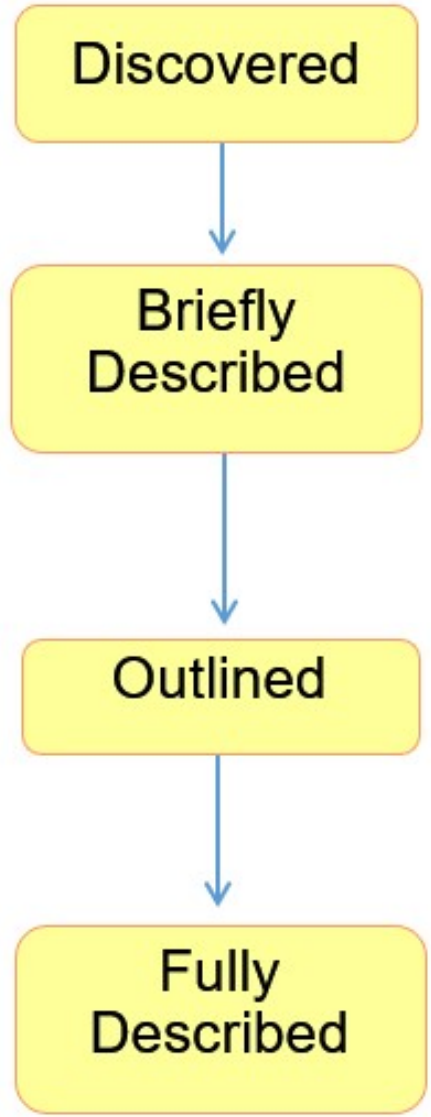
*I get some cash*

# Benefits of use case modeling

- Gives context for requirements
  - Put system requirements in logical sequences
  - Illustrate why the system is needed
  - Help verify that all requirements are captured
- Easy to understand
  - Uses terminology that customers and users understand
  - Tell concrete stories of system use
  - Verify stakeholder understanding
- Facilitate agreement with customers

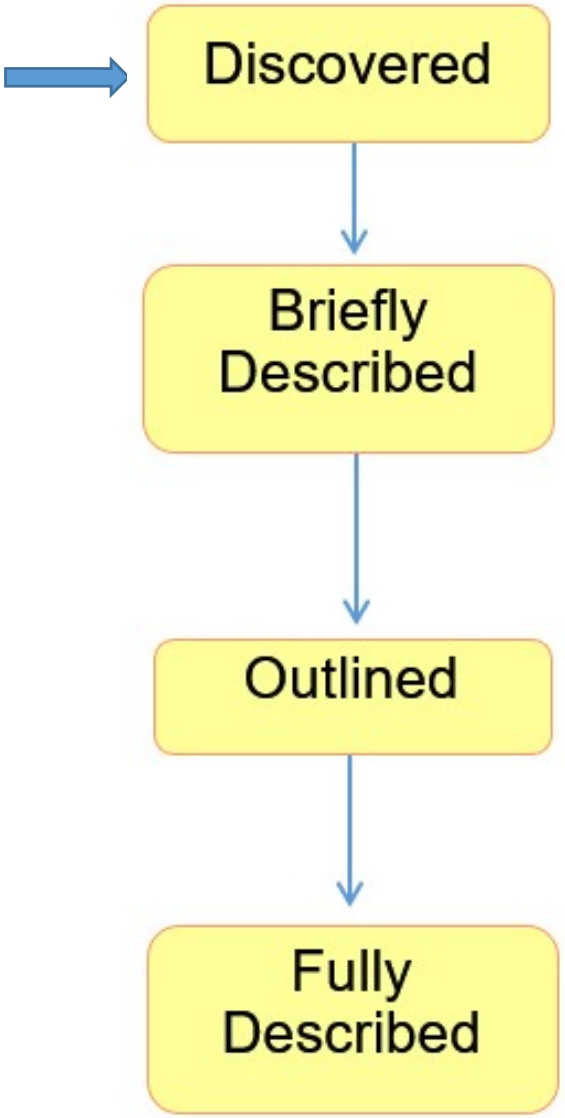
# Lifecycle of a use case

1. Identify/discover a use case
2. Provide a brief description for the use case
3. Provide a step-by-step flow (outline) of the events for the use case
4. Provide a detailed description for the use cas



# 1. Identify/discover use cases

- Examples of potential uses cases discovered for a course registration system
  - Register for Courses
  - Withdraw from Courses
  - Print Transcript
  - Close Registration
  - …



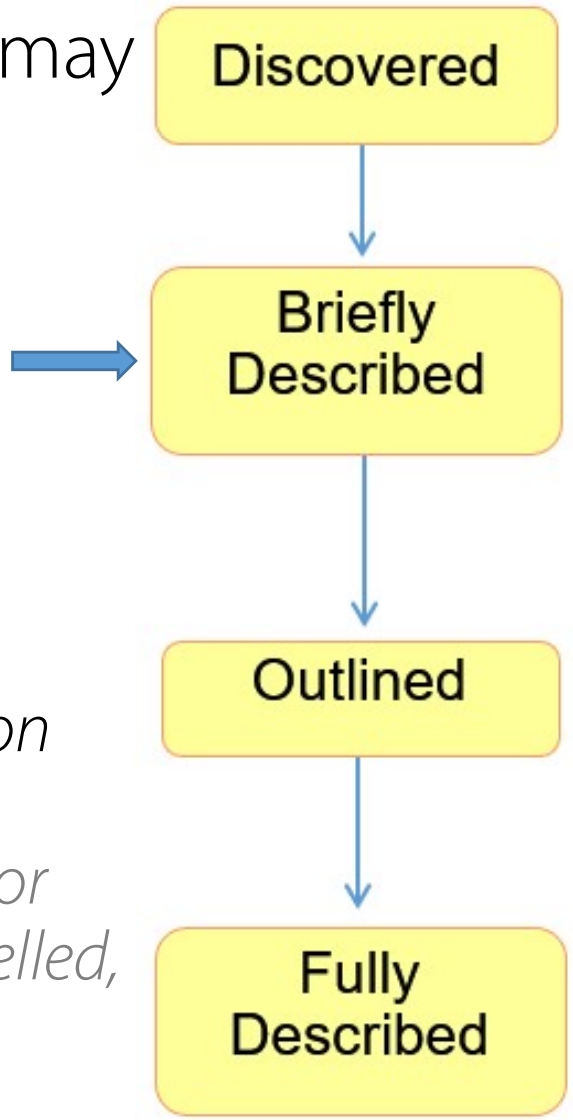
# 2. Provide a brief description

- For Register for Courses a brief description may be as follows:

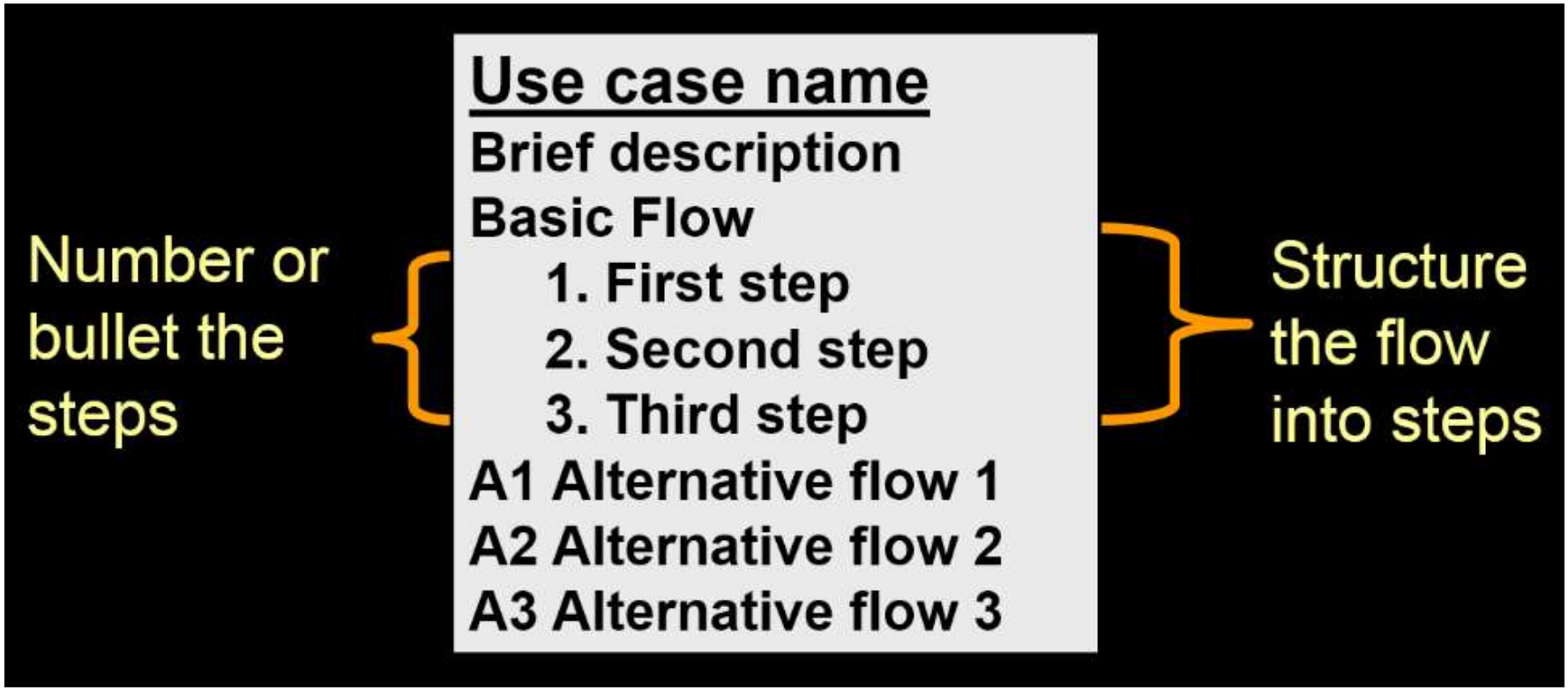
  *This use case allows a Student to register for course offerings in the current semester.*

- For Close Registration use case, a brief description may be as follows:

  *This use case allows a Registrar to close the registration process. Course offerings that do not have enough students are cancelled. The Billing System is notified for each student in each course offering that is not cancelled, so the student can be billed for the course offering.*

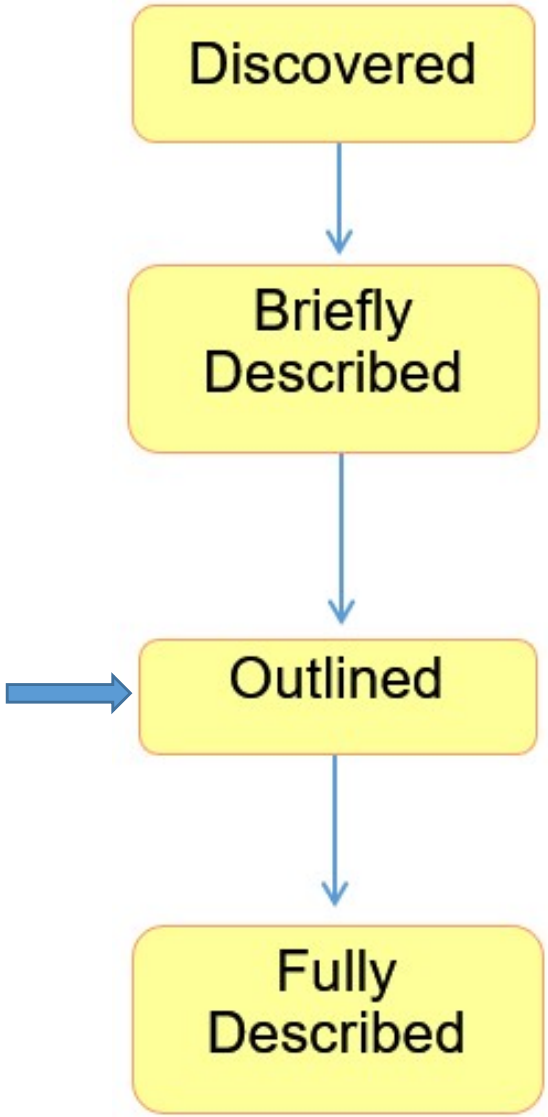


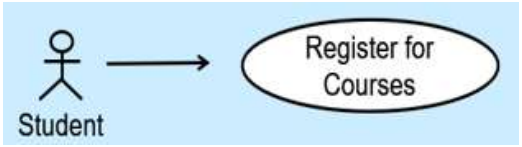
# 3. Provide a step-by-step flow of the events in a use case



# 3. Provide a step-by-step flow of the events in a use case

- A possible outline of Register for a Course
  1. Log on to system
  2. Approve log on
  3. Enter subject in search
  4. Get course list
  5. Display course list
  6. Select courses
  7. Confirm availability
  8. Enroll
  9. Display final schedule



# 4. Provide a detailed description

**Get Quote**

**1.1 Basic Flow**

**1. Customer Logs On**

The use case starts when the Trading Customer logs on.
The system validates the customer id and password.
The system presents a list of available functions.

**2. Customer Selects "Get Quote" Function**

The Trading Customer chooses to get a quote.
The system displays the list of trading symbols and names of securities.

**3. Customer Selects Security**

The Trading Customer selects from the list of securities or enters the trading symbol for a security.

**4. System Gets Quote from Quote System**

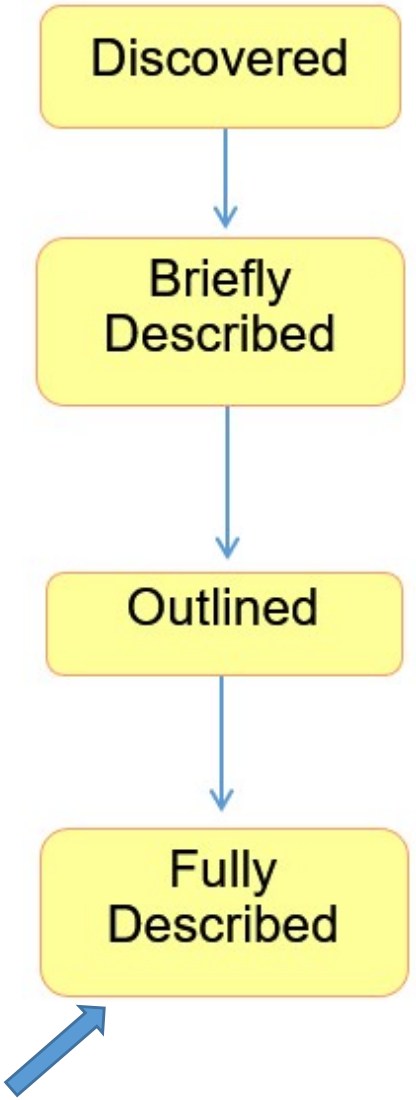
The system sends the trading symbol to the Quote System, and receives the Quote System Response.
The system presents the corresponding Quote Display (see fields to display in Supplementary Specifications) to the Trading Customer.

**Structure the flow into steps**

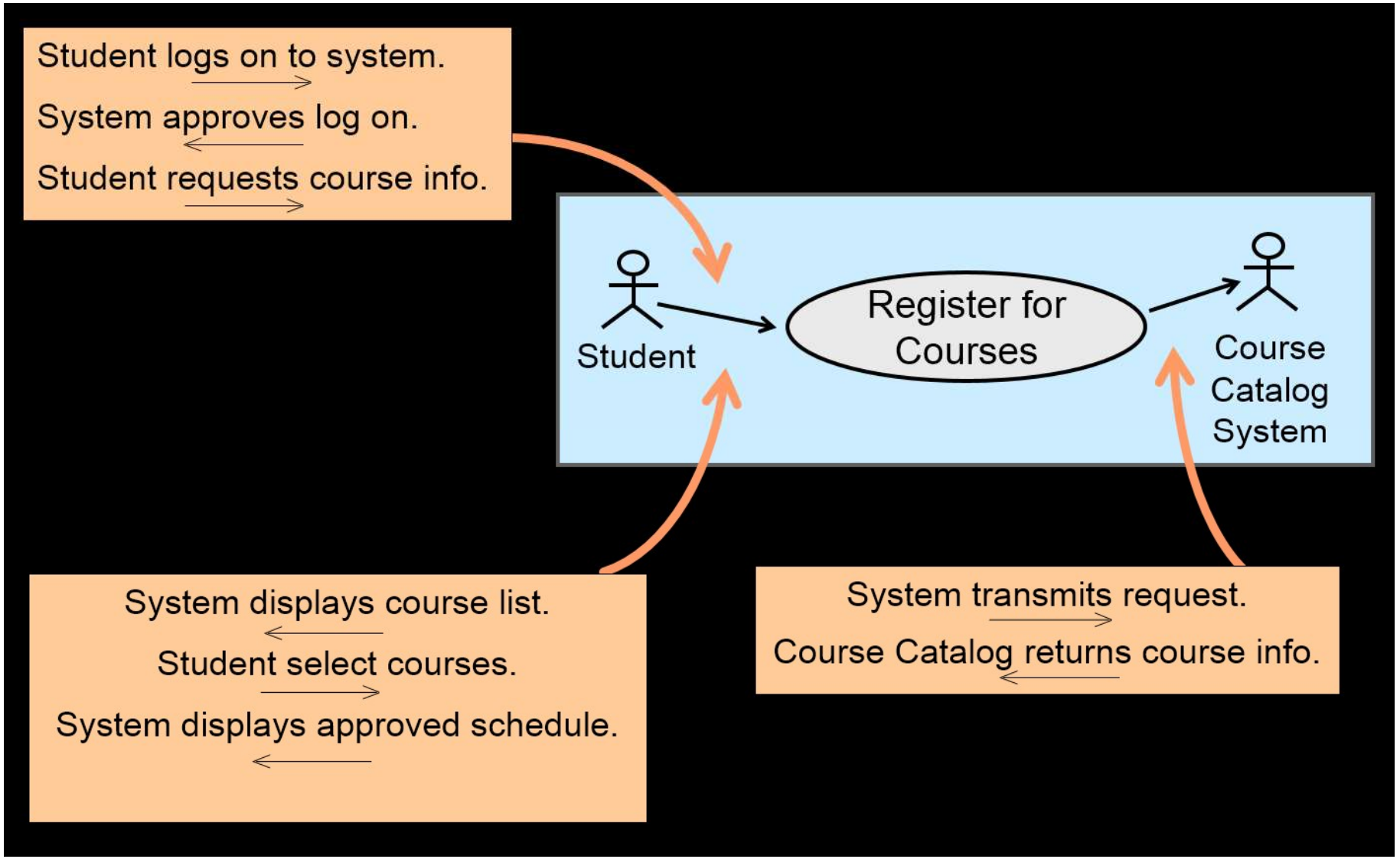
**Number and title each step**

**Make each step a roundtrip of events**

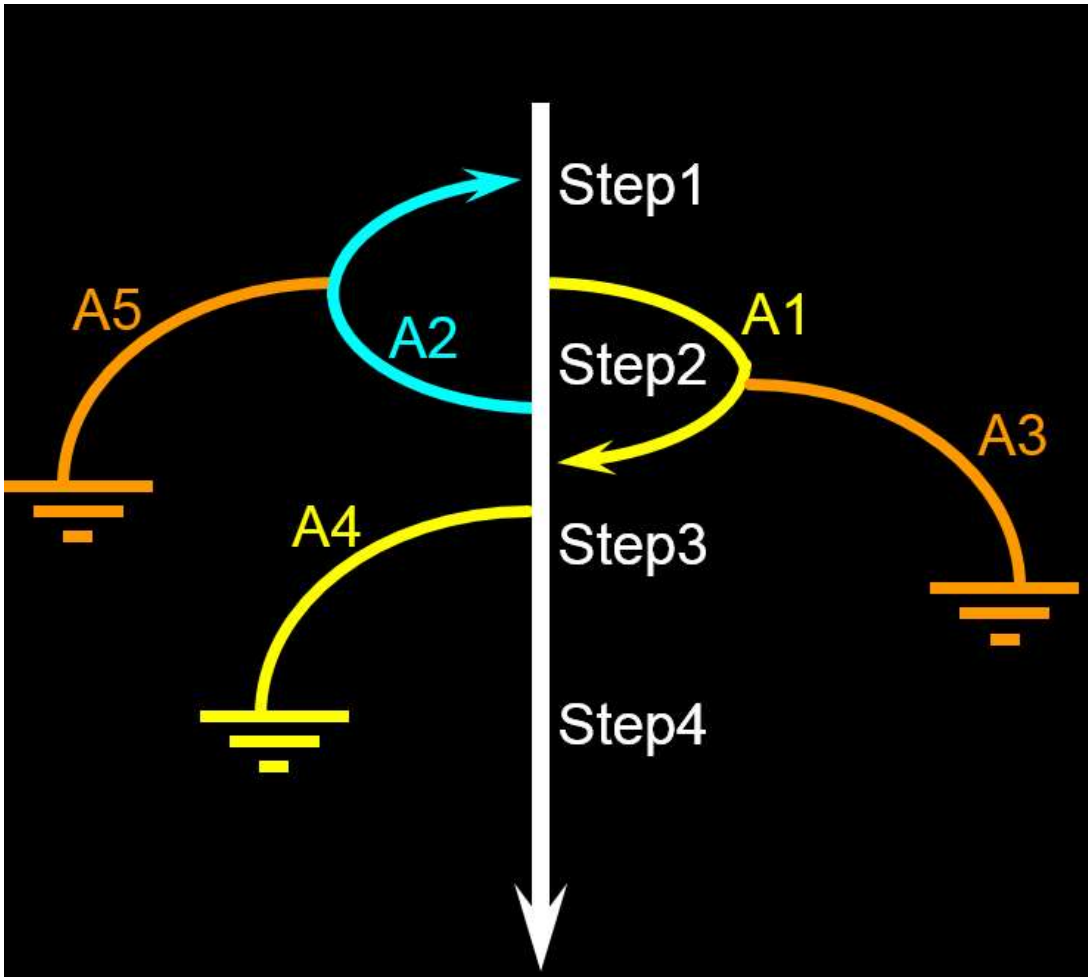
**Describe steps (completely and concisely)**



# An arrow is a whole dialog



# Alternate courses of action




<Use-Case Name>
1. Brief Description
2. Flow of Events
    2.1 Basic Flow
        Step 1
        Step 2
        Step 3
        Step 4
    2.2 Alternative Flows
        2.2.1 A1 …
        2.2.2 A2 …
        2.2.3 A3 …
        2.2.4 A4 …
        2.2.5 A5 …

# Alternate courses of action: an example

**Basic Flow**

1. Customer logs on.
2. Customer chooses to get a quote.
3. Customer selects stock trading symbol.
4. Get desired quote from Quote System.
5. Display quote.
6. Customer gets other quotes.
7. Customer logs off.

**Alternative Flows**

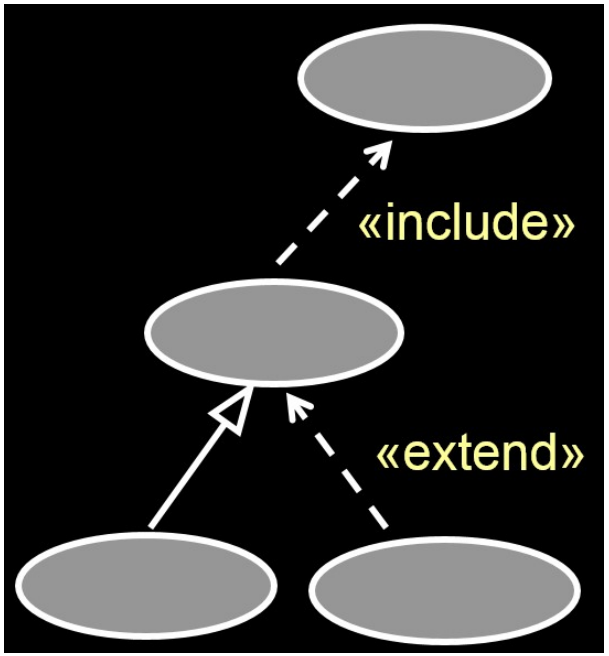
A1. Unidentified Trading Customer.

A2. Quote System Unavailable.

A3. Quit.

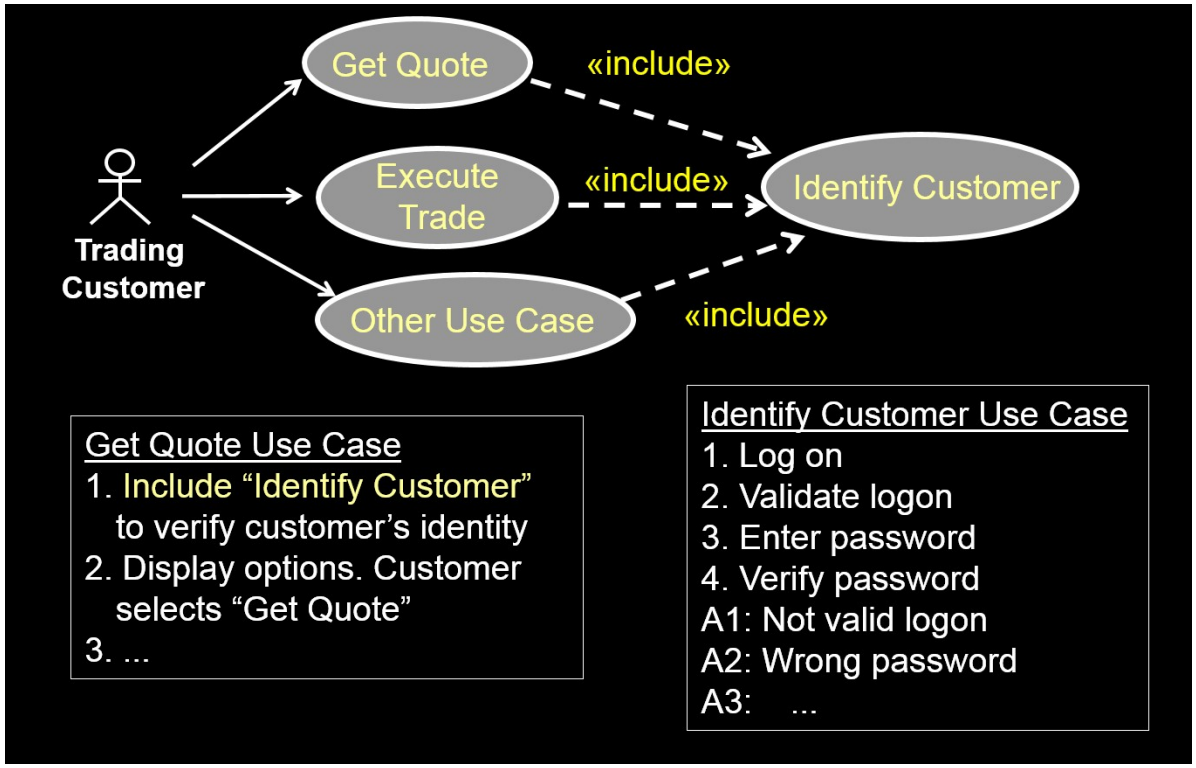
# Structuring use cases

- Factoring out parts of use cases to make new ones
  - Make it easier to understand
  - Make it easier to maintain
  - Share among many use cases
- Primary mechanisms
  - "include", "extend", generalization



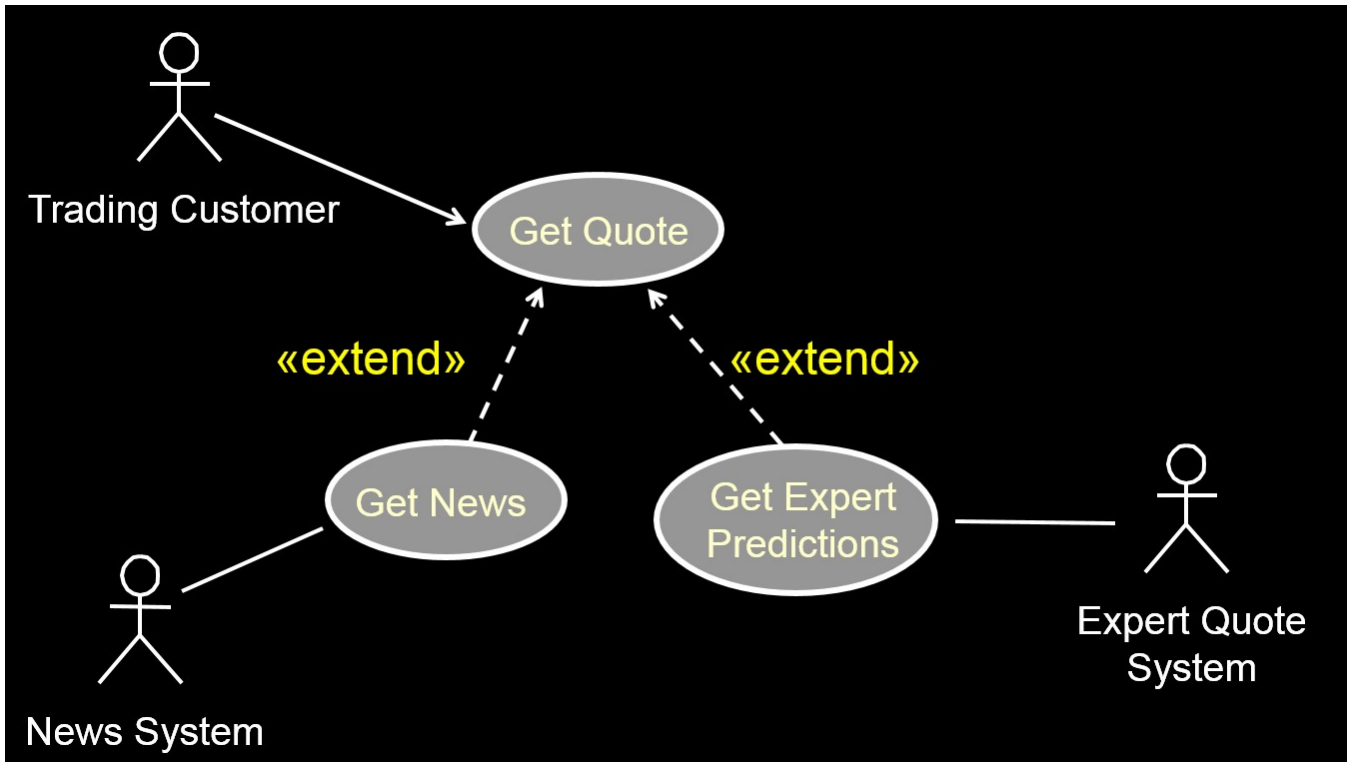
# The "include" relationship

- The behavior defined in the inclusion use case is explicitly included into the base use case

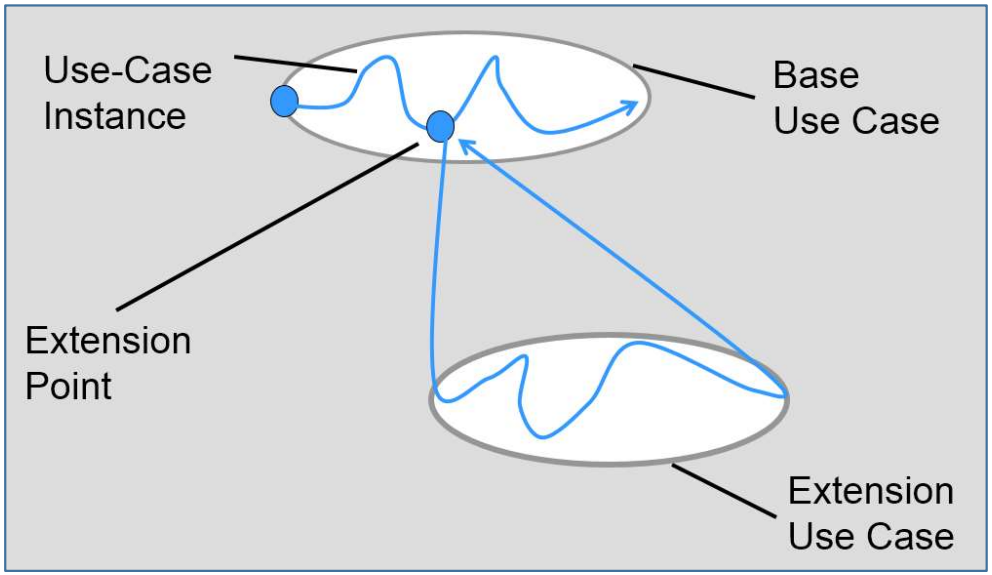


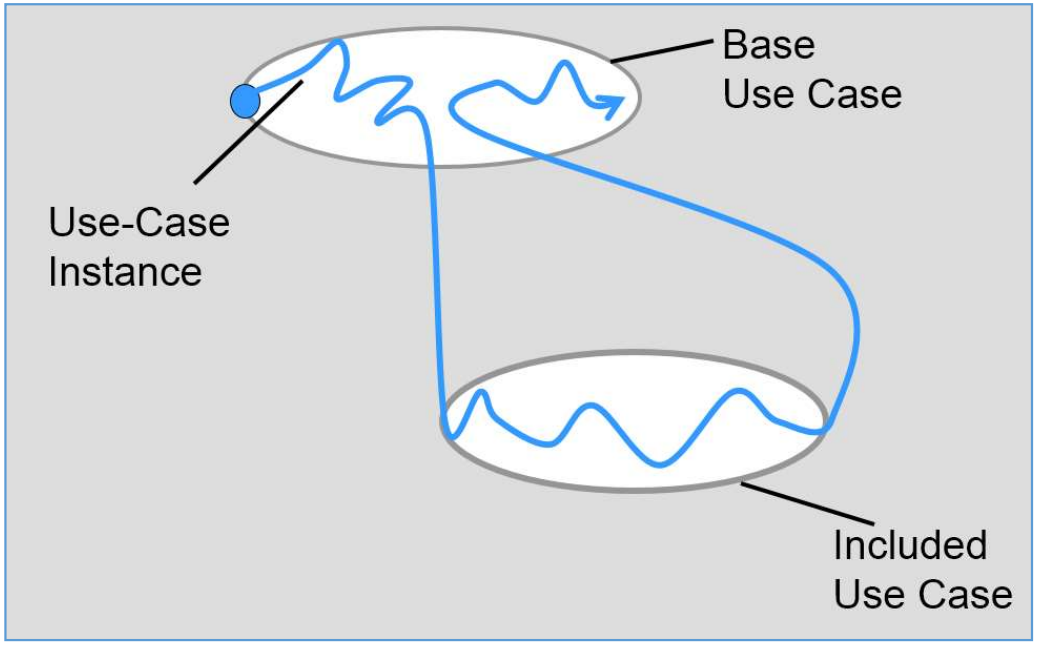
# The "extend" relationship

- Insert extension use case's behavior into base case only if the extending condition is true



# Use case relationships: "extend" vs "include"



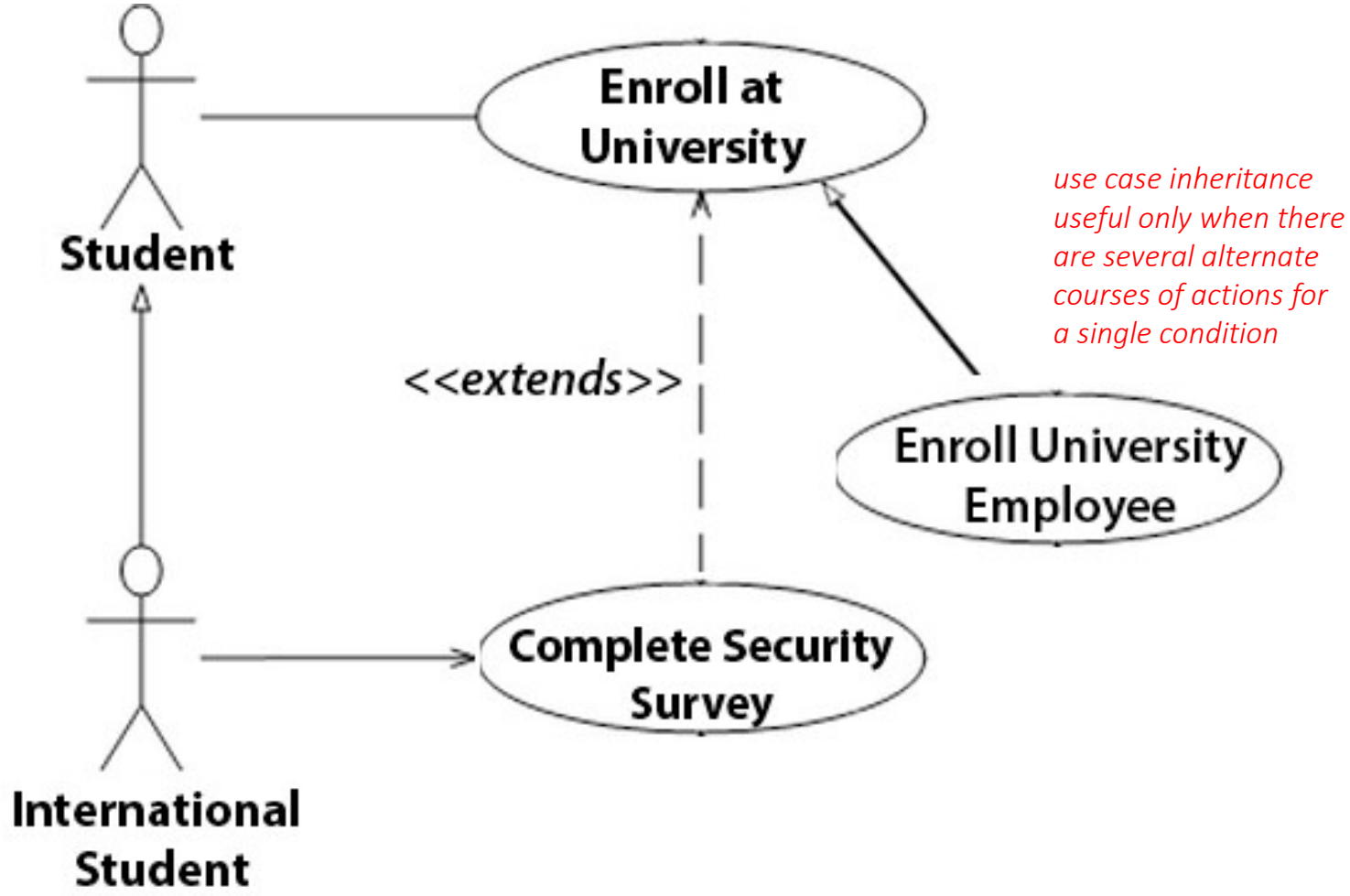



# Other use case concepts [1/3]

- Inheritance relates a specialized and a generalized element
  - A specialized use case inherits the goals and actors of its generalization, and may add more specific goals and steps for achieving them
  - A specialized actor inherits the use cases, attributes and associations of its generalization, and may add more

- In UML, generalized elements are at the arrowhead end

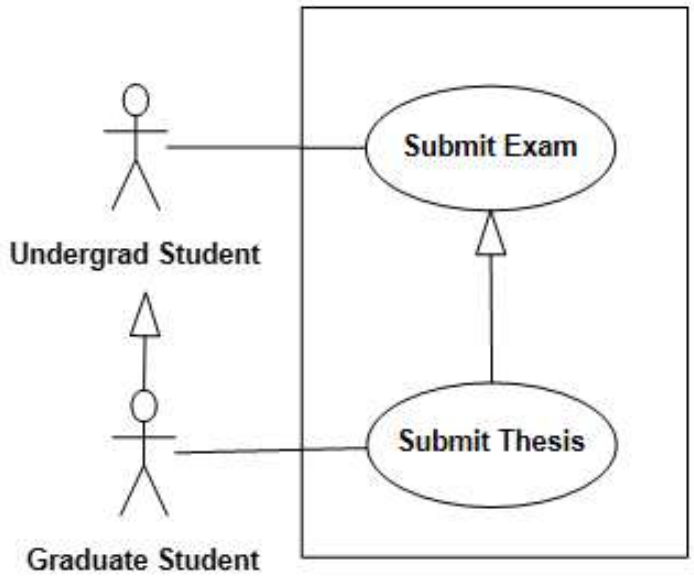
# Other use case concepts [2/3]

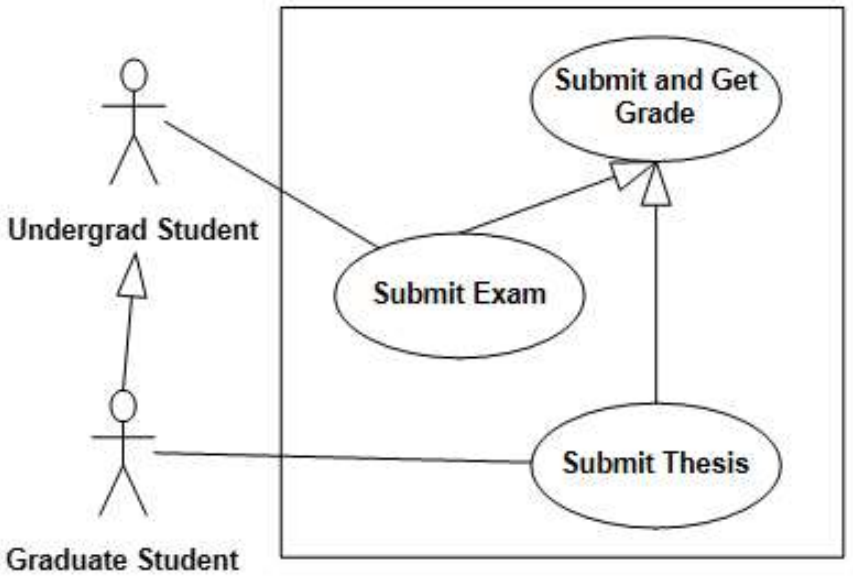



*use case inheritance useful only when there are several alternate courses of actions for a single condition*

# Put some though on specializations

- Similar to program design



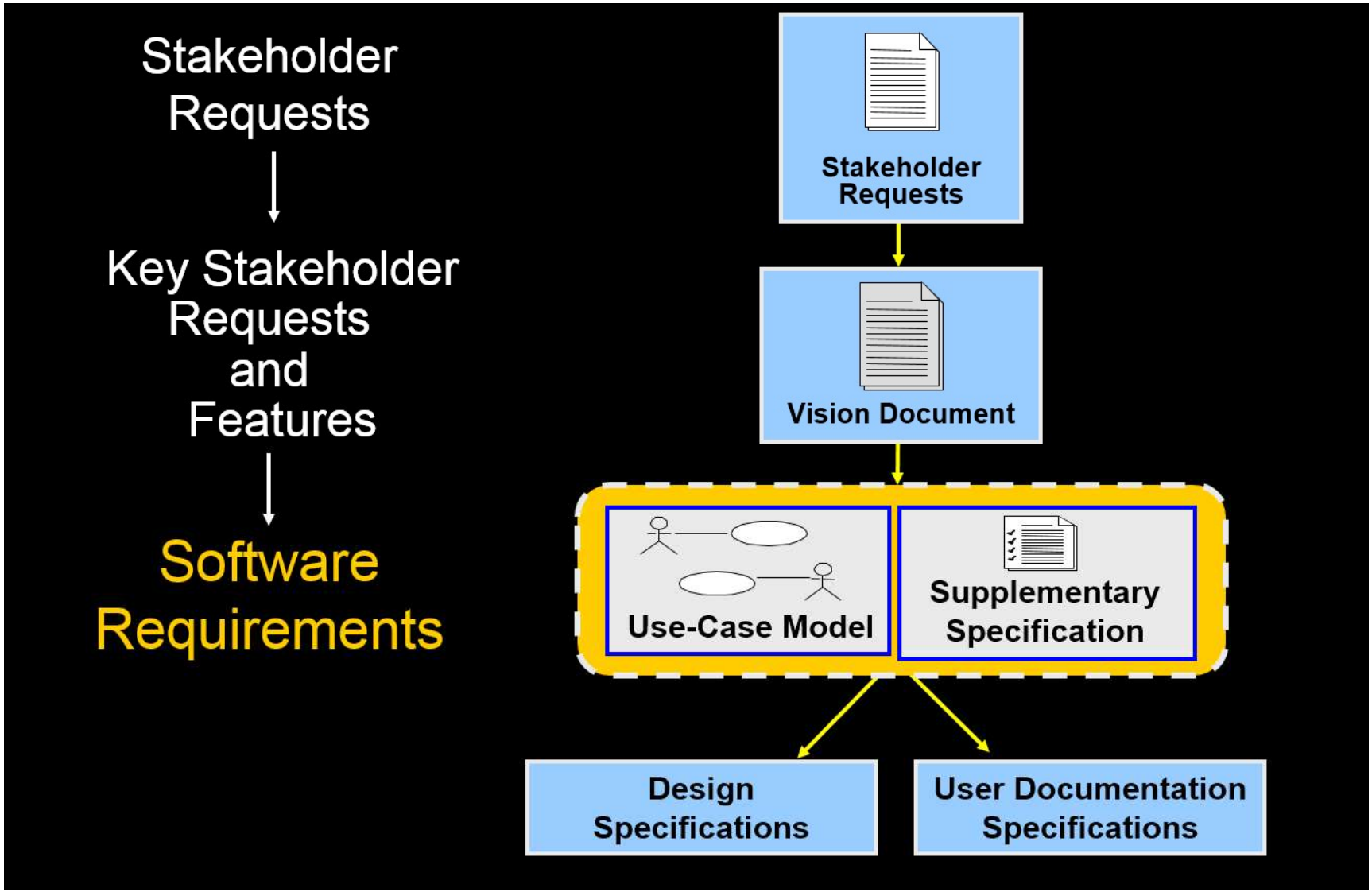

**Bad**: Undergrad can submit thesis

**Good**: Only graduate student can submit thesis

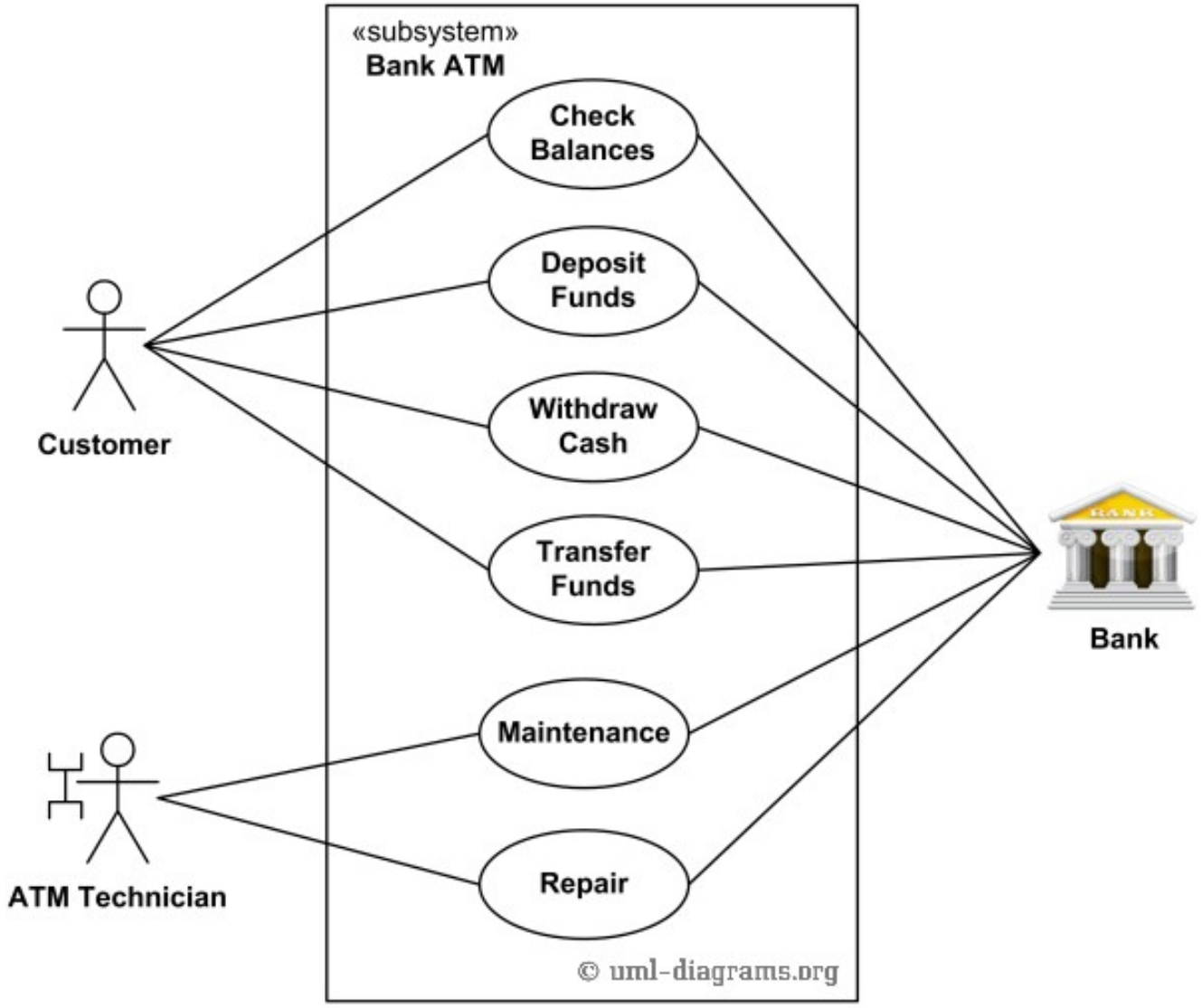
# Other use case concepts [3/3]

- Multiplicity on association links
  - 1: to state that exactly one instance of a role participates in each link
  - 1..*: to state that one or more instance of a role participate in each link
  - 0..1: to show that participation is optional
  - *: to state that zero or more instances of a role participate in the link
  - Default: 1

# Capturing requirements with use cases at a glance



# A sample example: an ATM machine



# A sample example: an ATM machine

**Discover** a use case:
Withdraw Cash

**Brief description**:
The use case allows a customer to withdraw money from their account

**Outline**:
Authenticate
Select amount
Dispense cash
Exit system

# ATM Withdraw Cash use case: Basic flow of events

- Insert Card: The use case begins when the actor Customer inserts their bank card into the card reader on the ATM. The system allocates an ATM session identifier to enable errors to be tracked and synchronized between the ATM and the Bank System.
- Read Card: The system reads the bank card information from the card.
- Authenticate Customer: Perform subflow Authenticate Customer to authenticate the use of the bank card by the individual using the machine.
- Select Withdrawal: The system displays the service options that are currently available on the machine. The Customer selects to withdraw cash.
- Select Amount: The system prompts for the amount to be withdrawn by displaying the list of standard withdrawal amounts. The Customer selects an amount to be withdrawn.
- Confirm Withdrawal: Perform subflow Assess Funds on Hand
- Eject Card: The system ejects the Customer's bank card. Perform subflow Conduct Withdrawal
- Dispense Cash: The system dispenses the requested amount of cash to the Customer. The system records a transaction log entry for the withdrawal.
- Use Case Ends: The use case ends.

# ATM Withdraw Cash use case: Alternative flow of events

- Handle No Communications with the Bank System
- Handle No Communications with the Customer's Bank
- Handle Inactive Card or Account
- Handle Stolen Bank Card
- Handle Invalid Bank Card Information
- Handle Correct PIN Not Entered
- Handle the Withdrawal of a Non-Standard Amount
- Handle Card Jam
- Handle Unreadable Bank Card
- Handle Invalid Card
- Handle Card Left Behind By Customer
- ….

# Each alternate flow is fully detailed

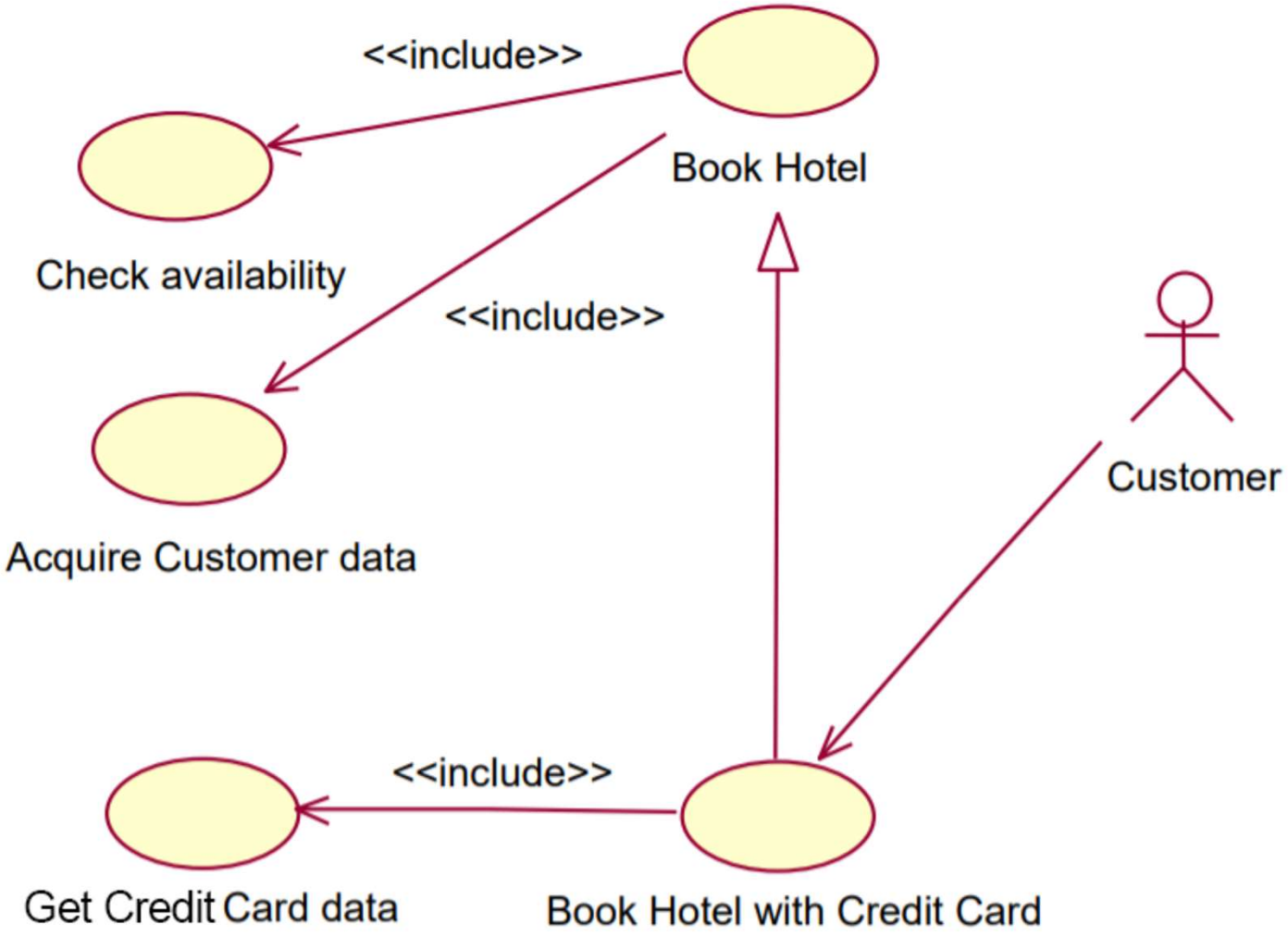
### 5.1 Customer Authentication

#### *5.1.1 Handle No Communications with the Bank System*

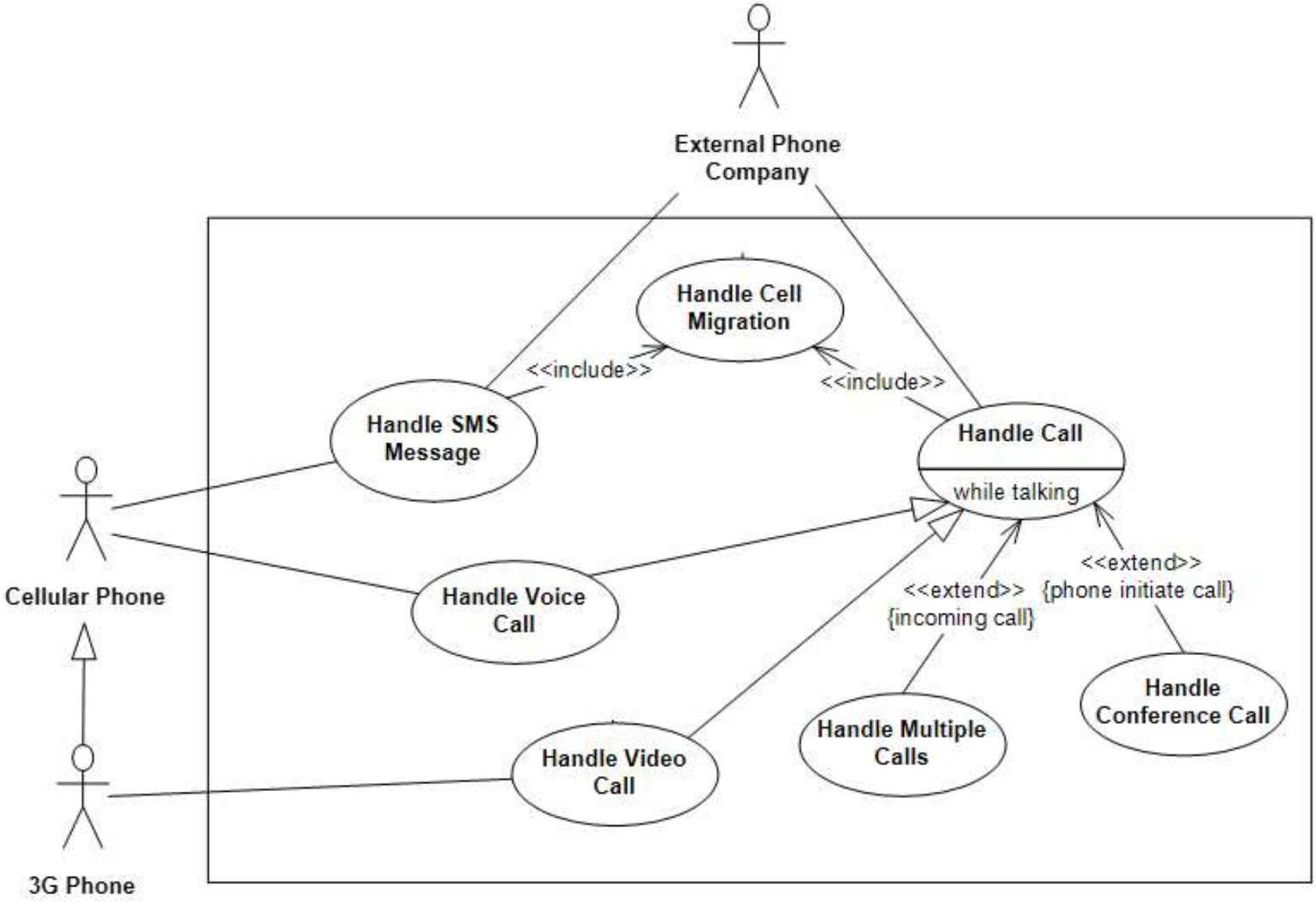
At the **Validate Card Information** step of Subflow ***Authenticate Customer*** if the Bank System cannot be contacted or does not reply within the set **communication time out period**,

1. If the communications link has failed more times than the **communication retry number,** then the authentication attempt is abandoned and Basic Flow is resumed at **Use Case Ends.**
2. The system will attempt to contact the Bank System until it has completed the number of retry attempts indicated by the **communication retry number.**
3. If communications is re-established the Basic Flow is resumed at **Authenticate Customer.**
4. If there is still no response from the Bank System the system creates an **event log** entry to record the failure of the communications link to the Bank System. The **event log** entry includes the type of failure.
5. The system sends the **event log** to the **Service Administrator** to inform them that communications with Bank System has been lost.
6. Resume the Basic Flow at **Use Case Ends**.

# An example with multiple include use cases

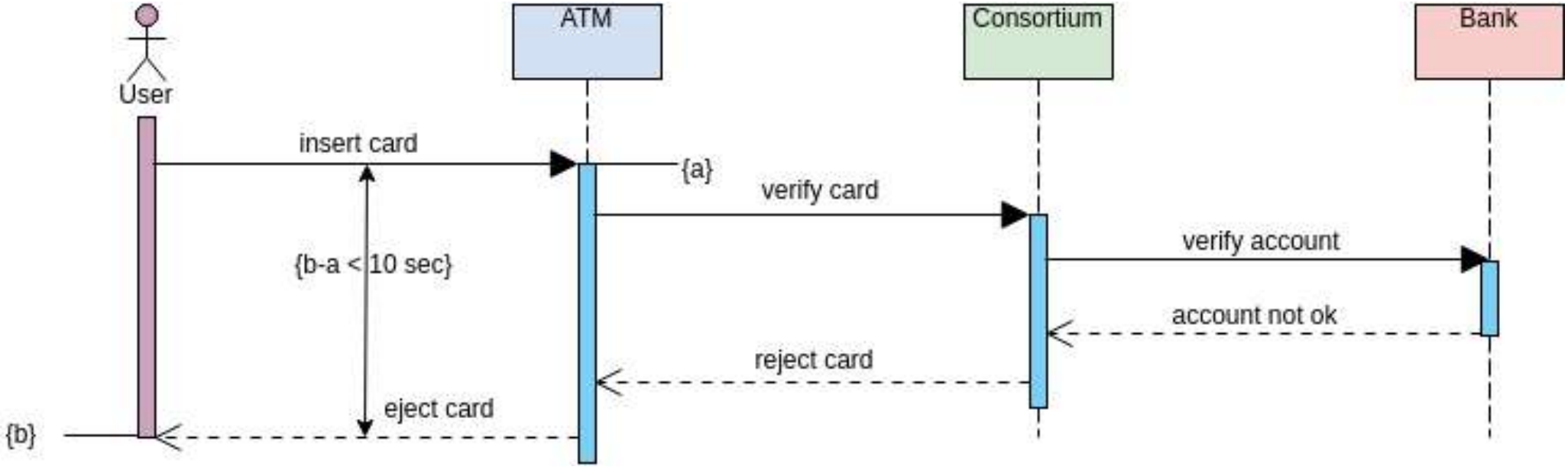


# Another example

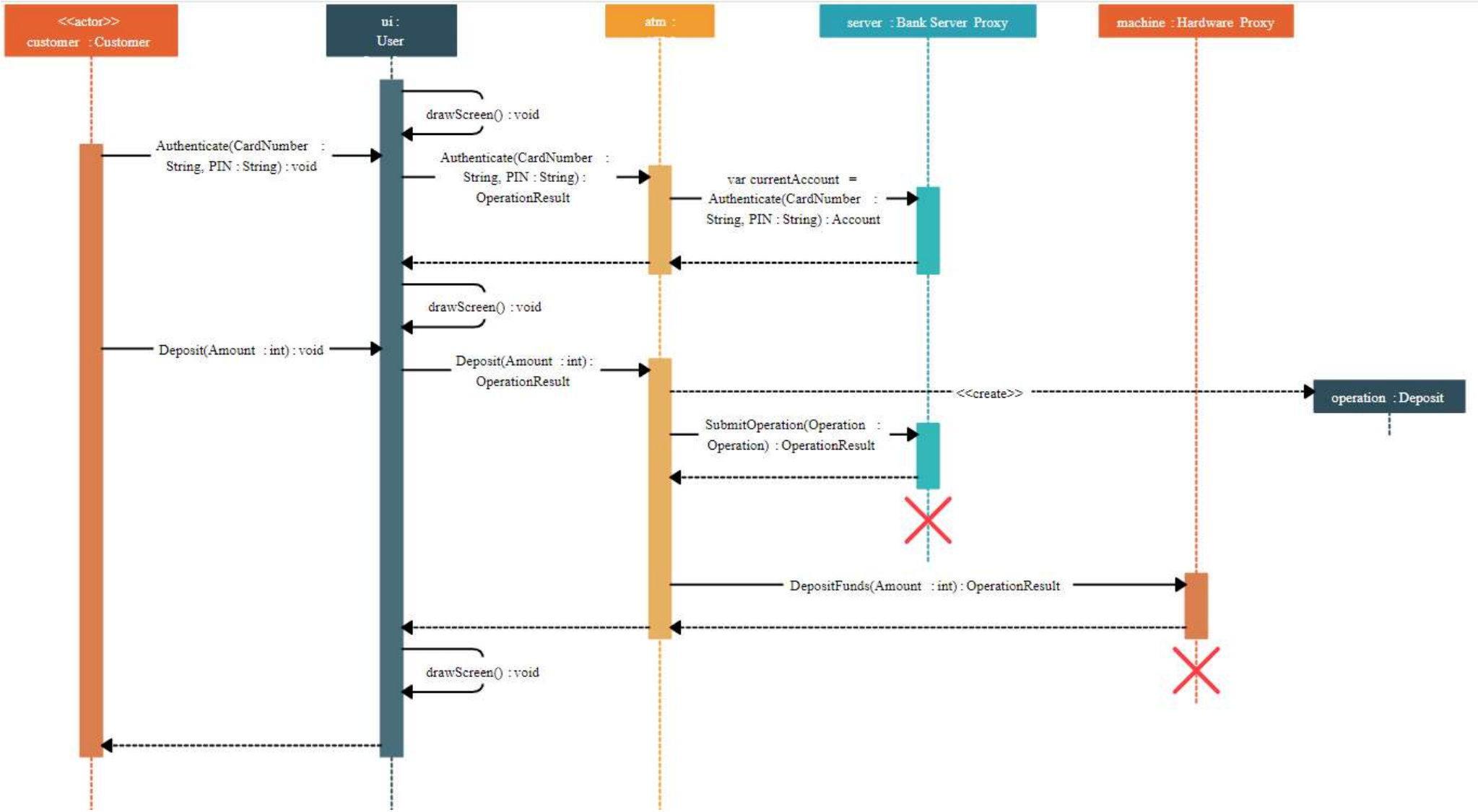


# Later we learn about UML sequence diagrams

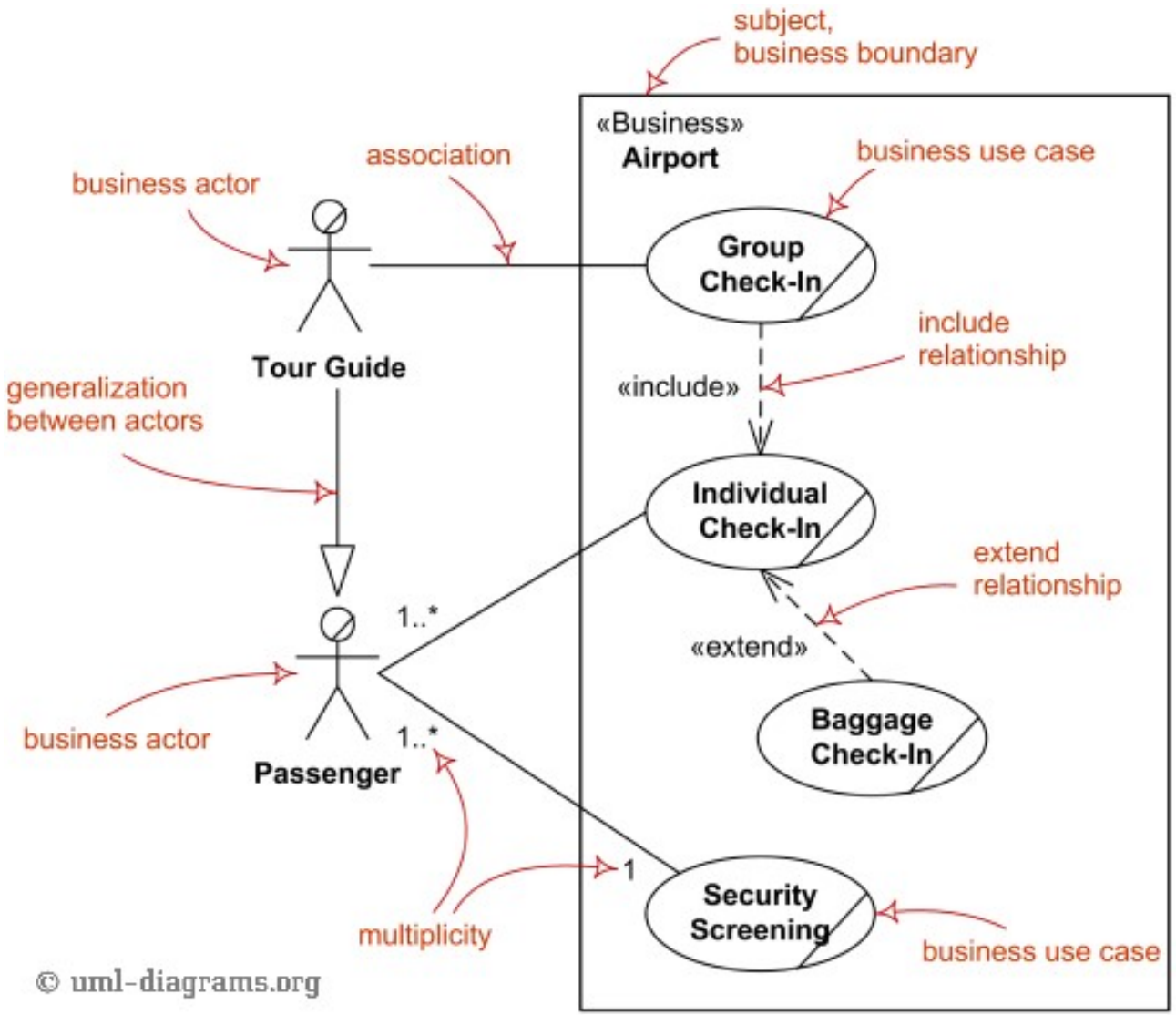
- More precisely specifying a use case behavior



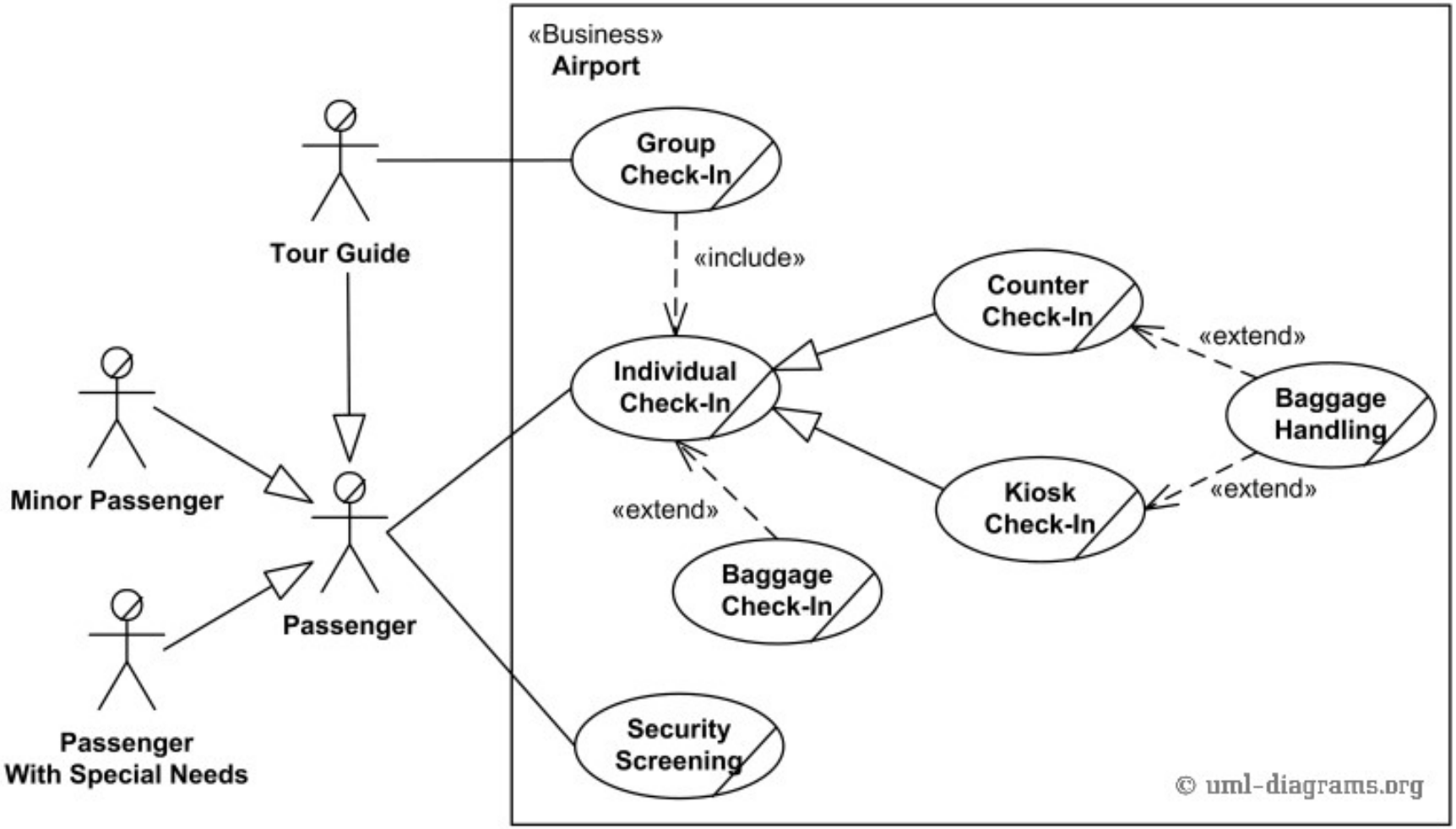
# Later we learn about UML sequence diagrams



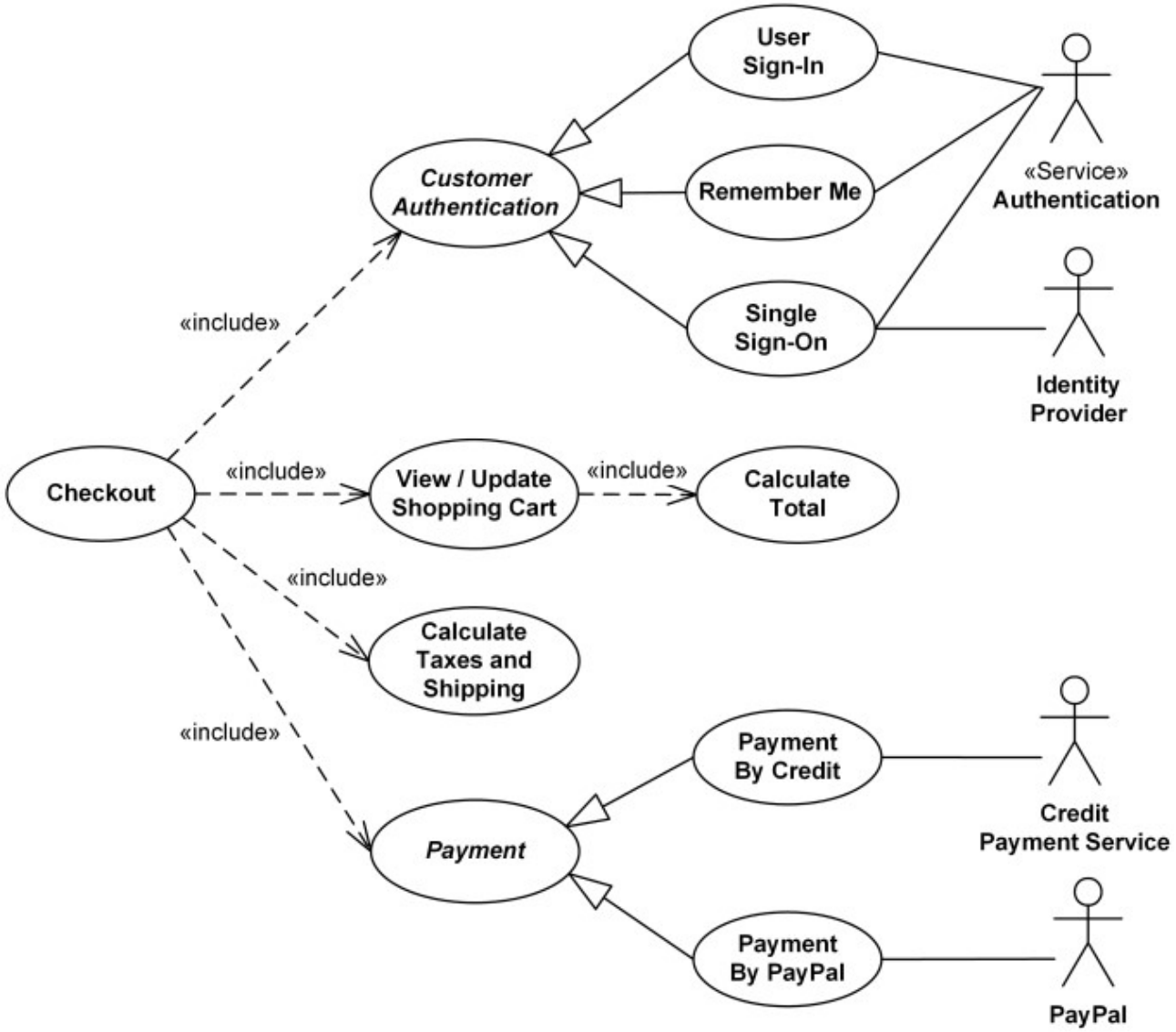
# A complete annotation



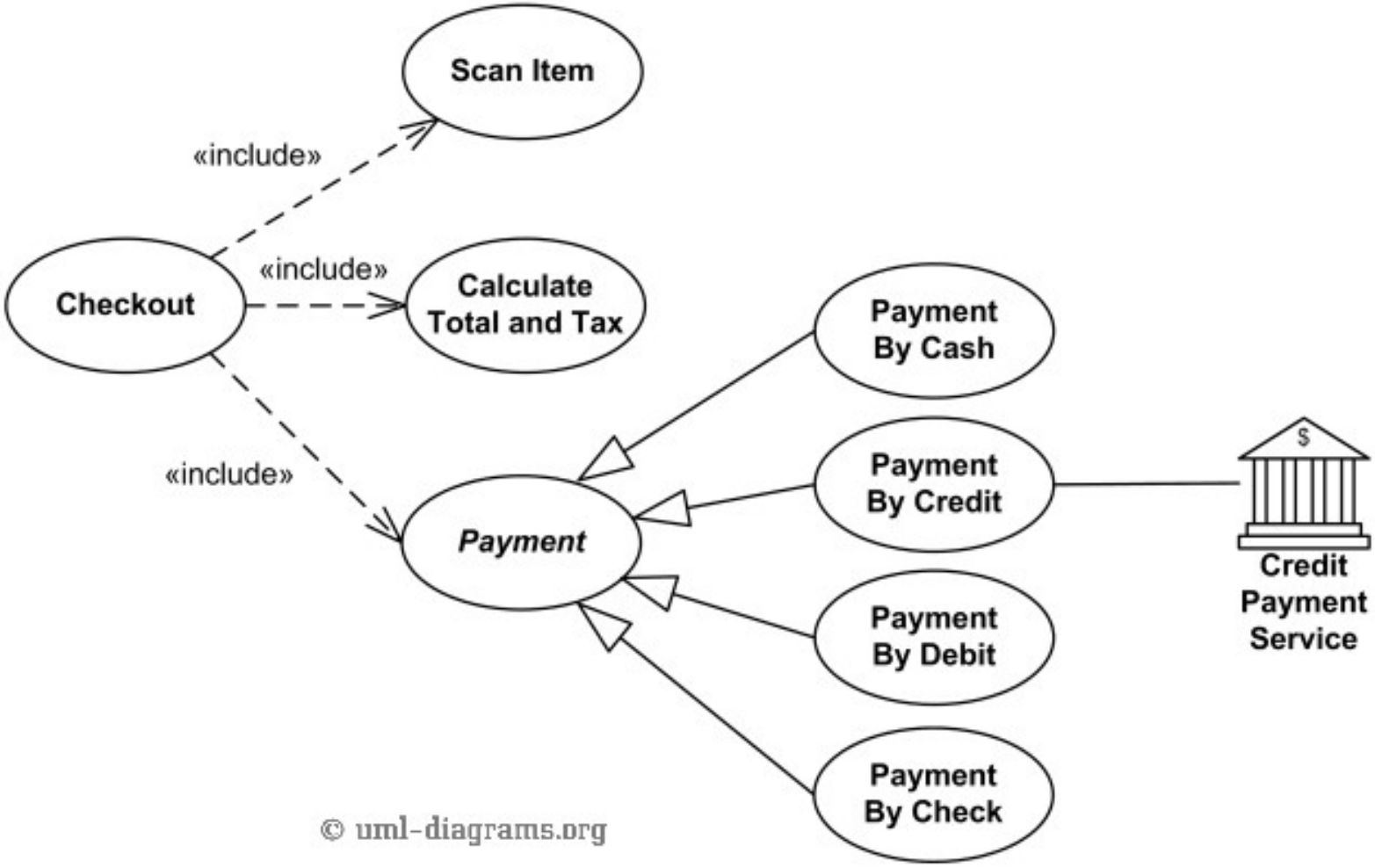
# A nice use of generalization/specialization



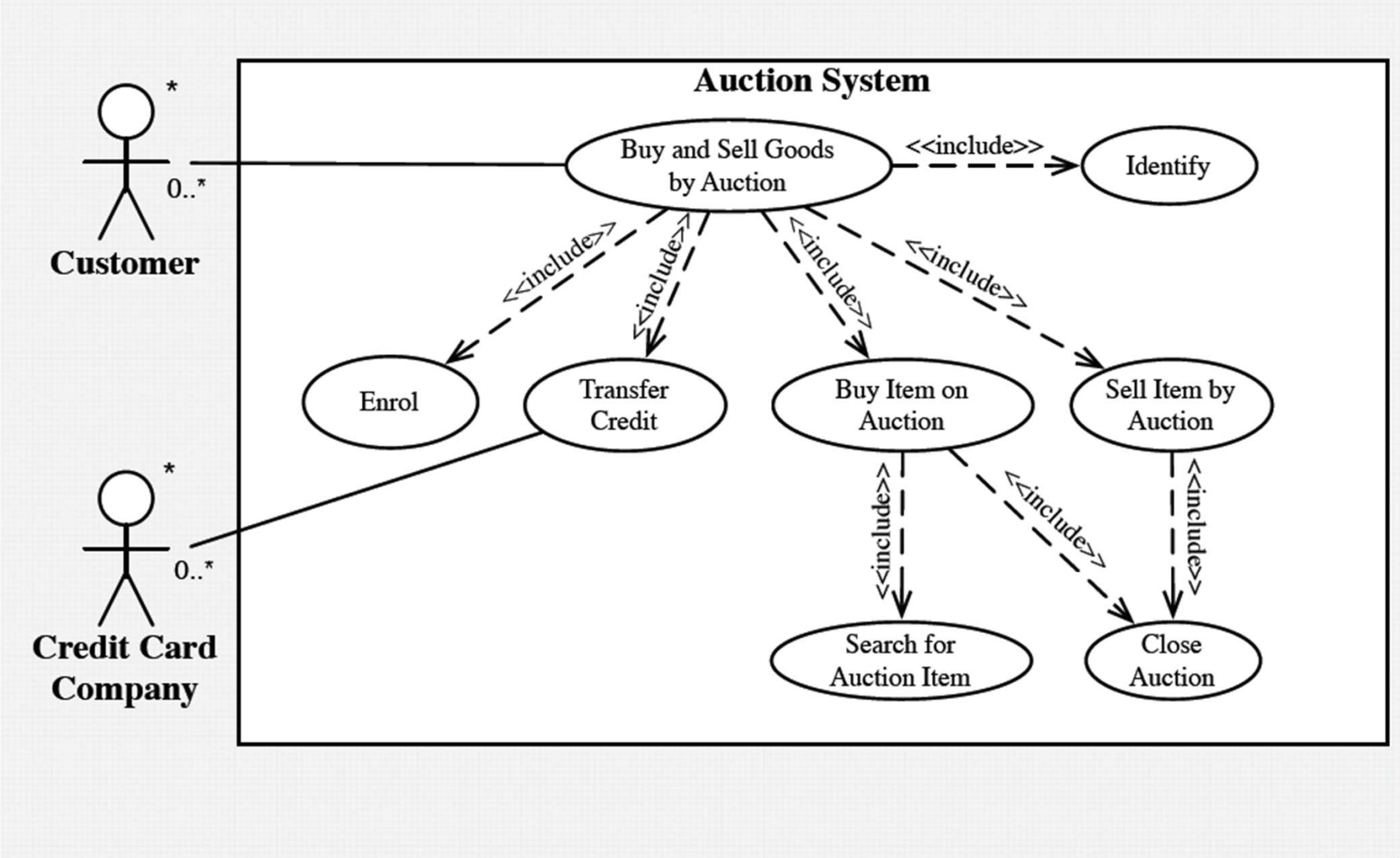
# A checkout use case



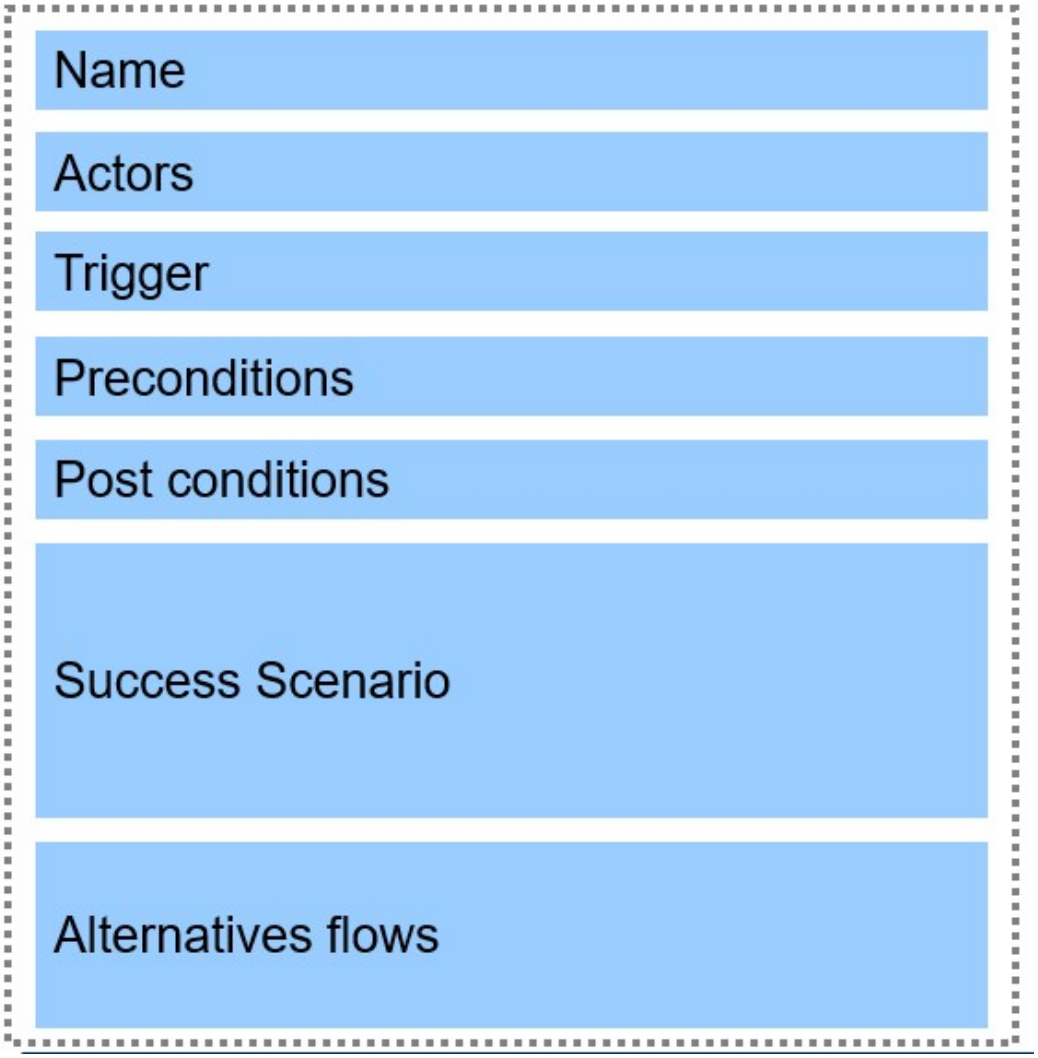
# A checkout use case



# An auction system

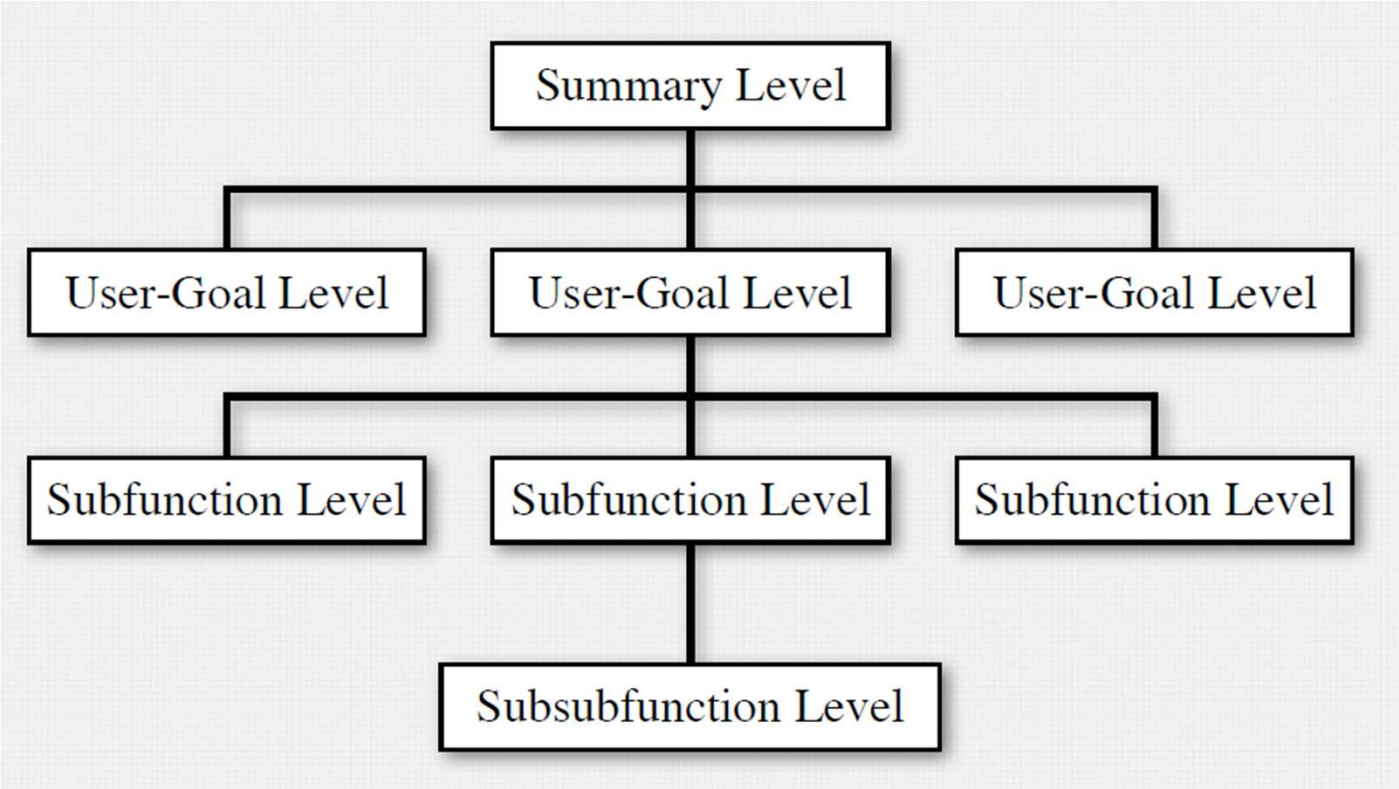


# Templates for a use case spec



# Granularity of use cases

- Summary level
- User goal level
- Subfunction level



# Summary level

- Written for either strategic or system scope; represent collections of user level goals
  - Example: "Configure Data Base"
- Provides the context for those lower-level use cases
- Can act as a table of contents for user level use cases

# User goal level

- Are usually done by one actor, in one place, at one time; the (primary) actor can normally go away happy as soon as this goal is completed
- Achieve a single, discrete, complete, meaningful, and well-defined task of interest to an actor

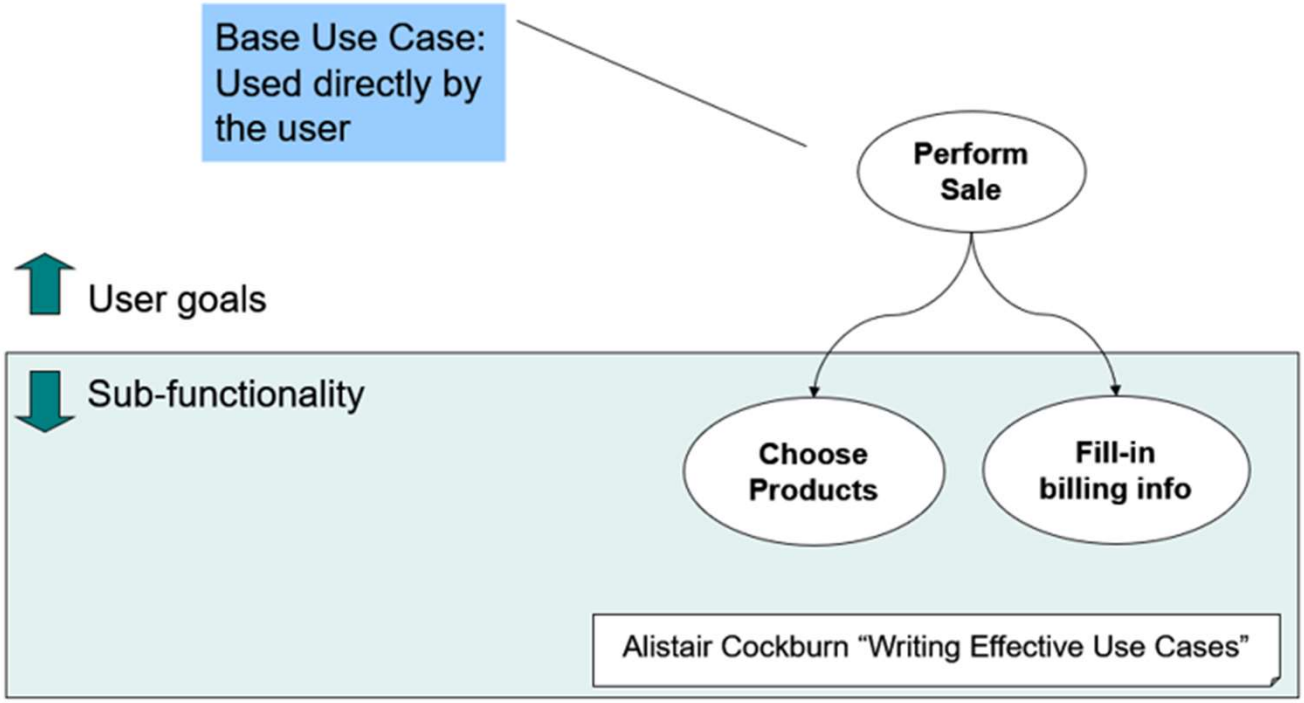
# Subfunction level

- Provide "execution support" for user-goal level use cases
  - Low-level and need to be justified, either for reasons of reuse or necessary detail
  - Often it is not clear who the primary actor of a subfunction-level use case is

# A quick example of levels

- Summary level: Configure the database
- User goal level: Create a virtual table
- Subfunction level: Log into the database

# A user goal vs a sub-function



# Summary level use case

**Use Case**: Manage Funds By Bank Account

**Scope**: Bank Accounts and Transactions System

**Level**: Summary

**Context**: The intention of the Client is to manage his/her funds by way of a bank account. Clients do not interact with the System directly; instead all interactions go through either: a Teller, a Web Client, or an ATM, which one depends also on the service.

**Multiplicity**: Many Clients may be performing transactions and queries at any one time. Each Client performs its transactions sequentially.

**Primary** Actor: Client

**Main Success Scenario**

1. Client opens an account.
2. Client identifies with system.
3. Client performs task on account: deposit money, withdraw money, transfer money, get balance.
4. Client closes his/her account.

**Extensions**

3a. System fails to identify the client; use case continues at step 2.

# User goal level use case

**Use Case**: Deposit Money

**Scope**: Bank Accounts and Transactions System

**Level**: User Goal

**Context**: The intention of the Client is to deposit money on an account. Clients do not interact with the System directly; instead, for this use case, a Client interacts via a Teller.

**Multiplicity**: Many Clients may be performing deposits at any one time. A Client only requests one deposit at a given time.

**Primary Actor:** Client

**Supporting Actor**: Teller

**Main Success Scenario**:

1. Client requests Teller to deposit money on an account, providing sum of money.
2. Teller requests System to perform a deposit, providing deposit transaction details.
3. System validates the deposit, credits account with the requested amount, records details of the transaction.
4. System informs Teller that deposit was successful.

**Extensions:**

2a. System ascertains that it was given incorrect information:

2a.1 System informs Teller about error; use case continues at step 2.

# Subfunction level use case

**Use Case**: Authenticate Client

**Scope**: Automatic Teller Machine (ATM for short)

**Level**: Sub-Function

**Context**: The intention of the Client is to identify him/herself to the System. A project (operational) constraint states that identification is made with a card and a personal identification number (PIN).

**Multiplicity**: Only one client can use the ATM for identification at a given time.

**Primary Actor**: Client

**Supporting Actors**: Card Reader, Bank Server, Numeric Keyboard

**Main Success Scenario**:

1. Client inserts card into Card Reader.
2. Card Reader informs System of card details.
3. System validates card type.
4. Client provides PIN to System using the Numeric Keyboard.
5. System requests Bank Server to verify identification information.
6. Bank Server informs System that identification information is valid.

# Use case level example

- “Jenny is standing in front of her bank's ATM. It is dark. She needs to enter her PIN to get some cash.
- What are possible summary, user goal, and subfunction level uses cases?

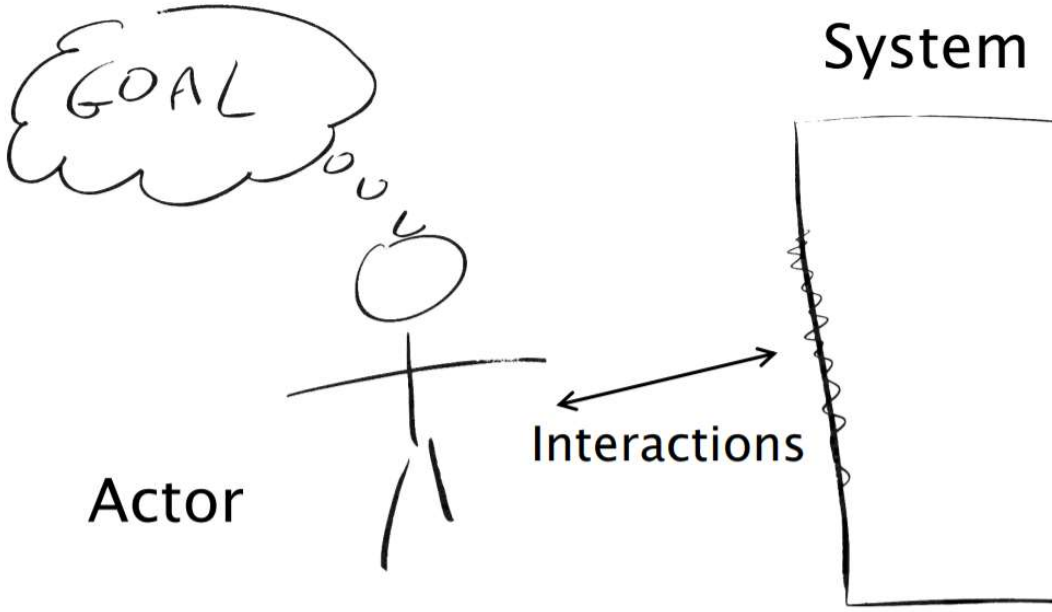
- Summary: Use the ATM
- User goal: Get money from the ATM
- Subfunction: Enter PIN

# Use cases and requirements

- Use case modeling is a means of arriving at the requirements document
- Use cases alone are not the requirements
  - Represent most of functional requirements
  - Miss external interfaces, data formats, business rules
- Use cases should be based on some form of conceptual model or glossary

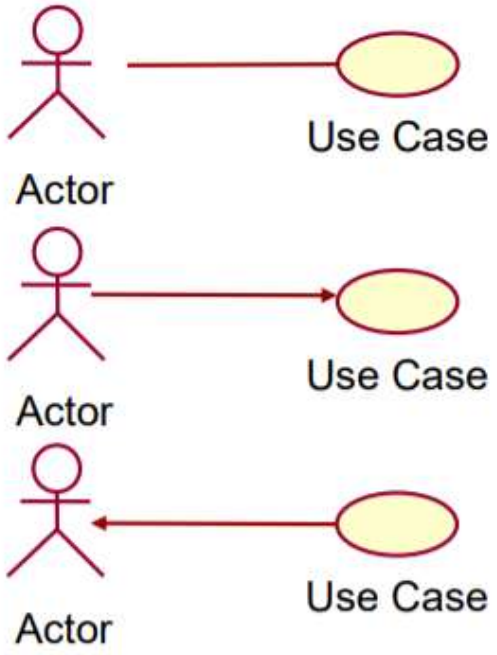
# Summary of use case modeling

- A use case describes a goal-oriented set of interactions between external actors and the system under consideration

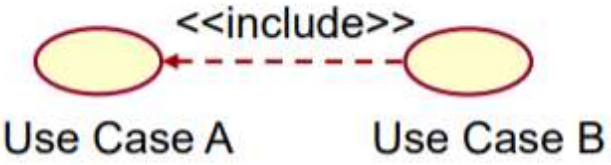


# Summary of notation

- Association model

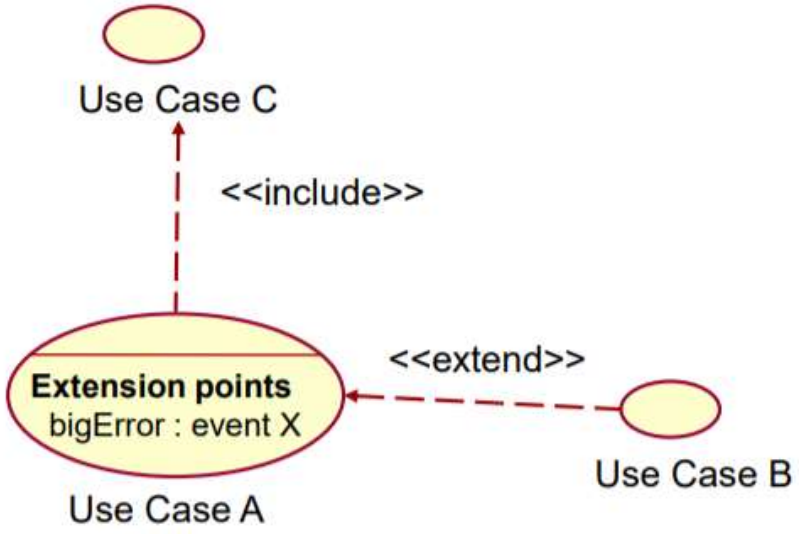



- Include



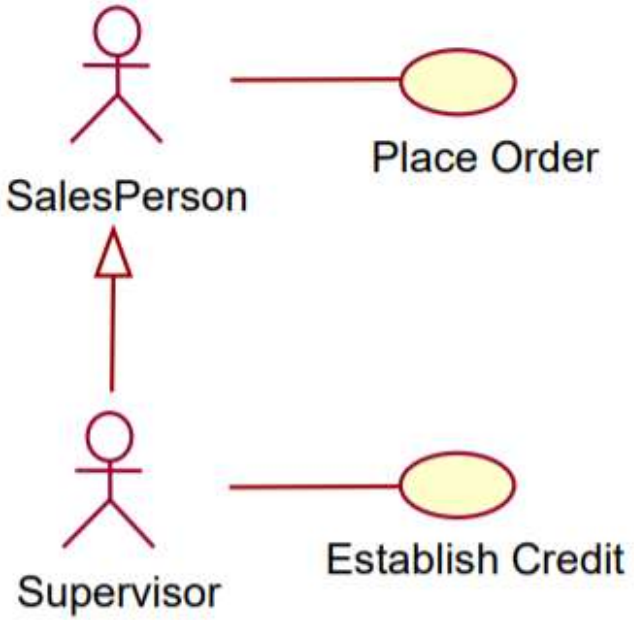
# Summary of notation

- Extension: An extend relationship from use case B to use case A indicates that an instance of use case A may be augmented by the behavior specified by B



# Summary of notation

- Actor generalization

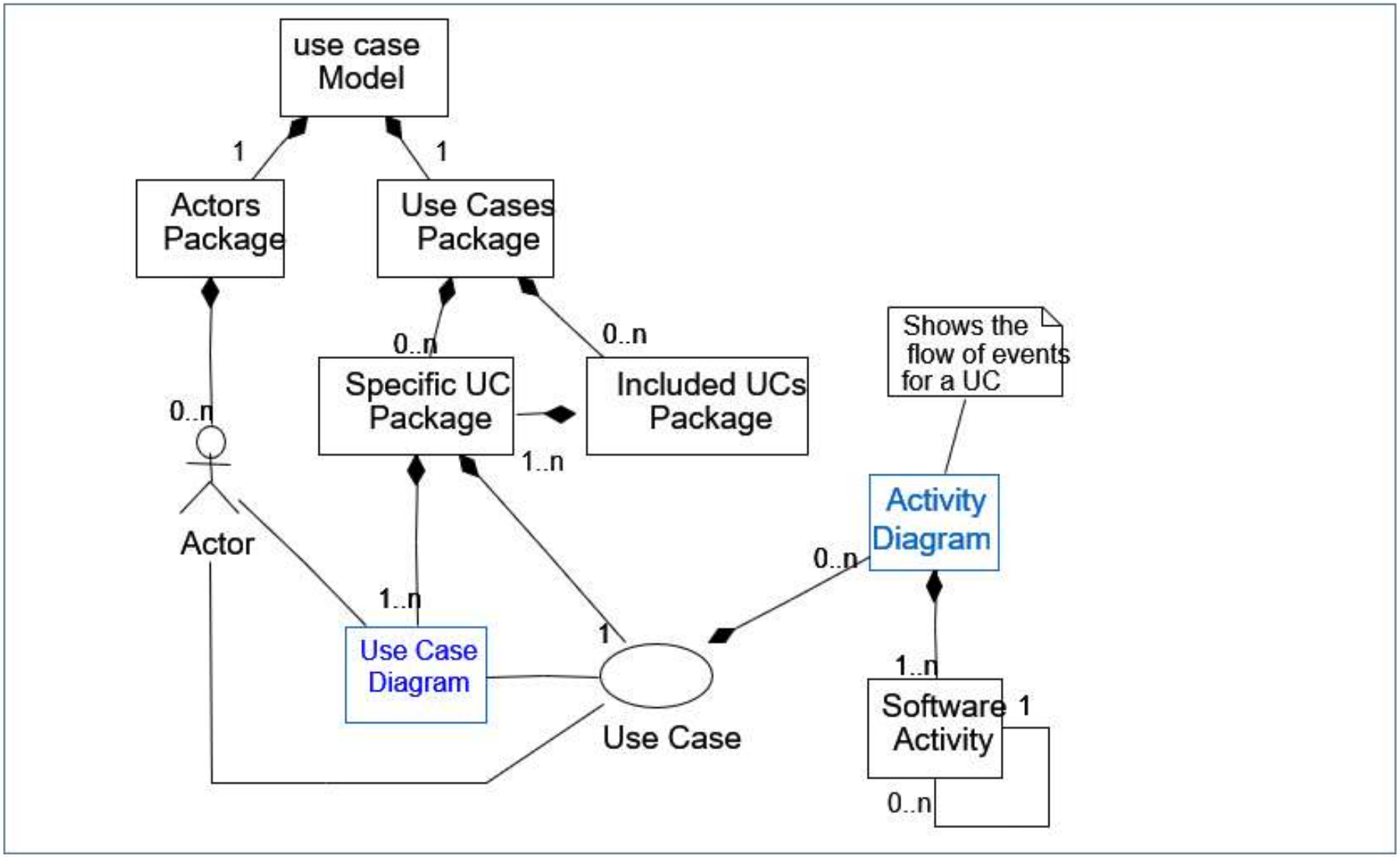



- Use case generalization

# Summary: A process of developing use cases

- Identify use cases (find actors, discover use cases)
- Provide a brief description
- Outline each use case (a step-by-step flow of the events)
- Provide detailed descriptions

# Elements of a use case model



# References


- IBM Rational UML/UP Course Resources
- Jorg Kienzle and Shane Sendall, *Requirement elicitation with use cases*
- K. Bittner and I. Spence, *Use Case Modeling*, pp. 301-330, Pearson, 2002.