LECTURE 2

DOMAIN ANALYSIS AND REQUIREMENTS MODELING

TOPICS

Domain Analysis

- Domain Modeling

Requirements

- Functional Requirements
- Non-Functional Requirements
- Use Cases
- Use Case Diagrams
- Use Case Relationships
- Use Case Examples
- Specifications Document

Agile User Stories

DOMAIN ANALYSIS

Domain analysis is the process by which a software engineer learns background information

Domain refers to the general field in which the customer expect to be using the software

Domain analysis: gather information about the domain from experts, books, existing software and its documentation…

- The aim is to understand the problem domain independently of the particular system we intend to develop
- We do not try to draw the borderline between the system and the environment
- We focus on the concepts and the terminology of the application domain with a wider scope than the future system.

DOMAIN ANALYSIS

You are not expected to become a domain expert

- But you need to gather sufficient information to understand the problem

Benefits:

- Faster development: communicate with stakeholders more effectively
- Better understanding of users' needs
- Anticipation of extensions

DOMAIN ANALYSIS

1. **Glossary of terms defining the common terminology and concepts of the problem domain**
2. **General knowledge about the domain (basic facts widely known by experts)**
3. **Customers and users (who will or might buy the software)**
4. **Environment (will the new system interact with existing ones?)**
5. **Domain model**
   - UML class diagram to model the relationship between entities
   - Entity relationship model
6. **Tasks and procedures currently performed**
7. **Competing software**

DOMAIN MODEL

There is not a unified view in software engineering regarding what a domain model is

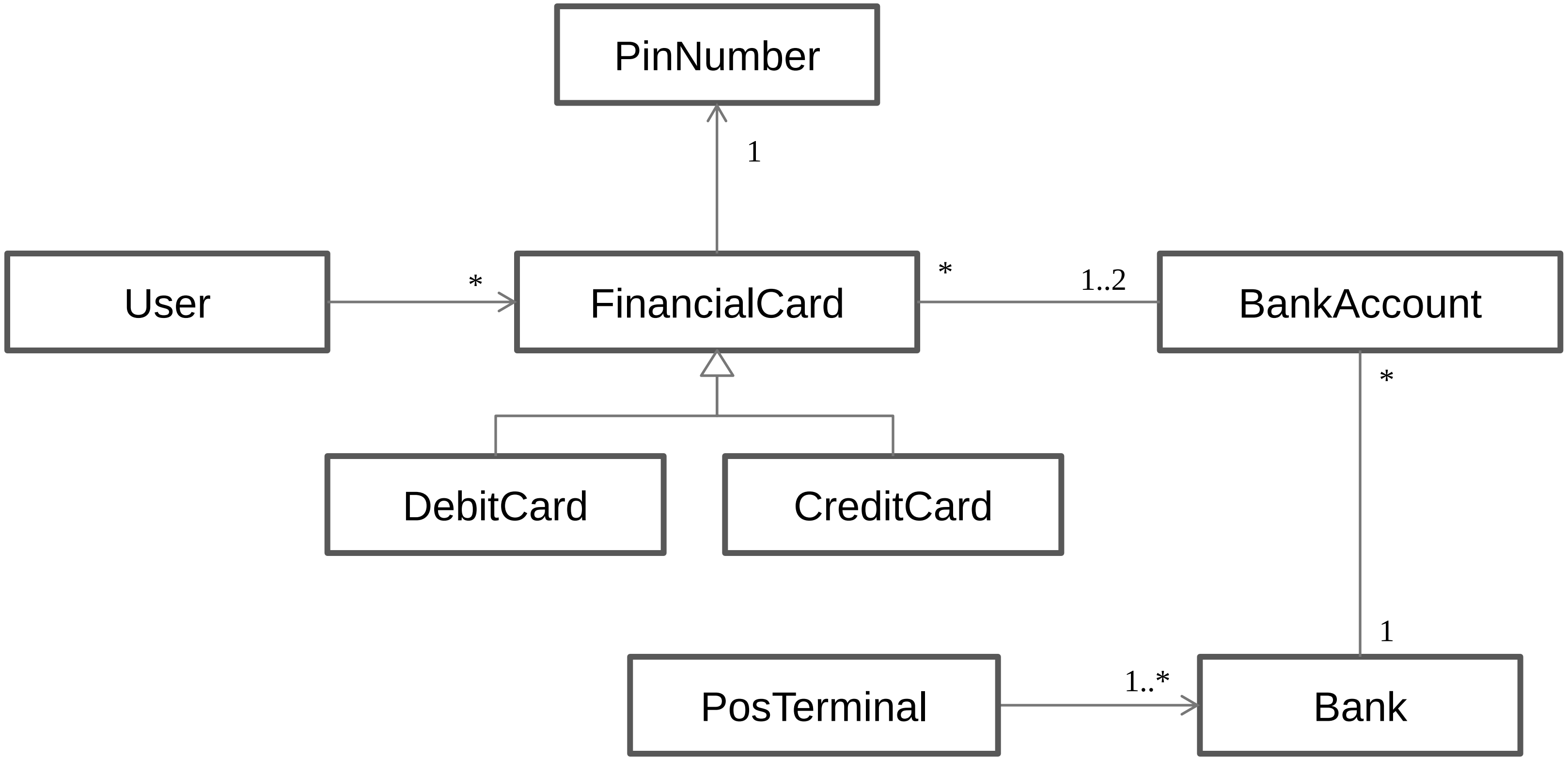
However, in general, domain modeling is the decomposition of a domain into its individual entities

It is a way to describe and model domain entities and relationship between them

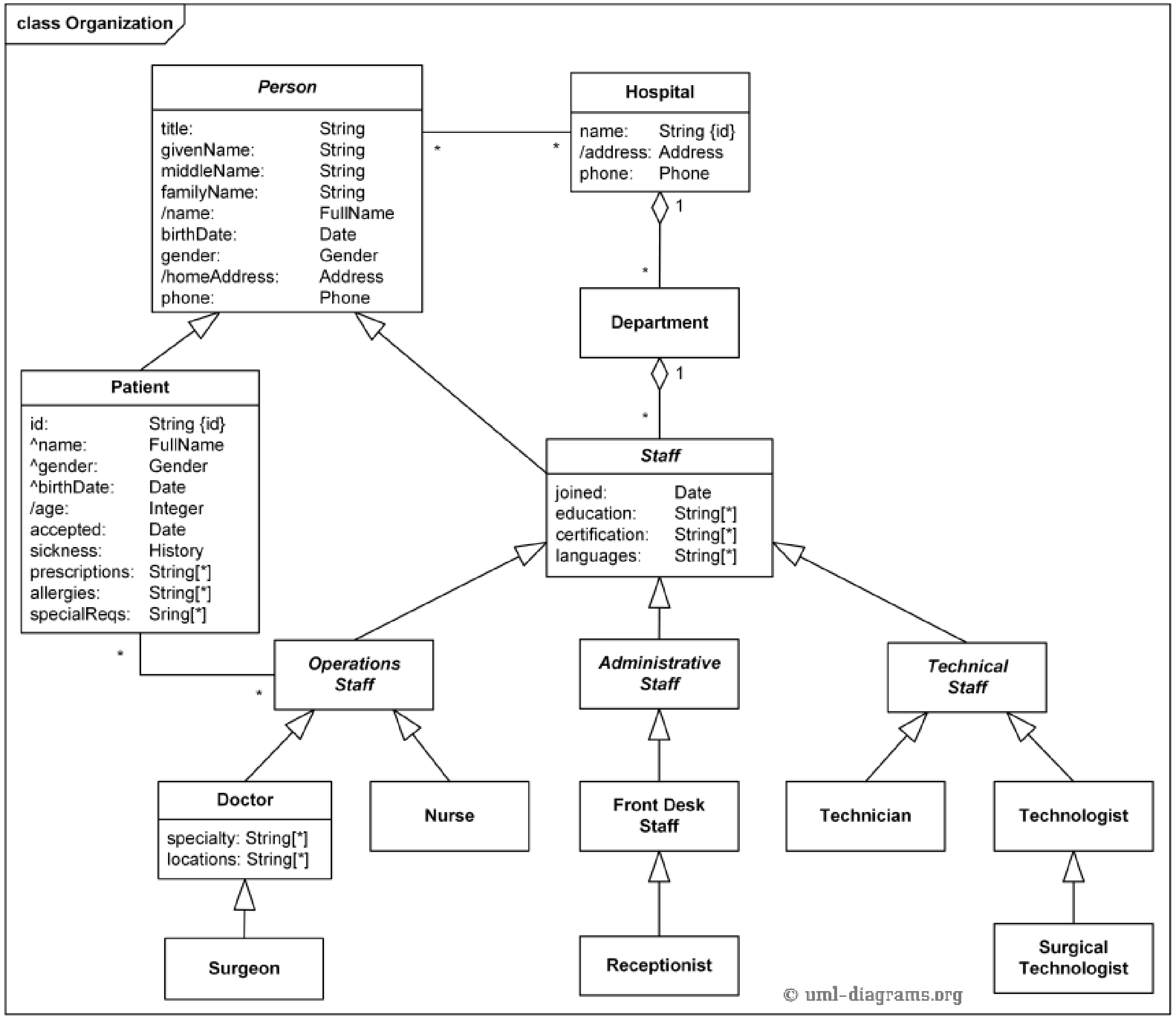
- These entities collectively describe the domain space

Typically, the domain model for all enterprises within the same domain should be somewhat the same

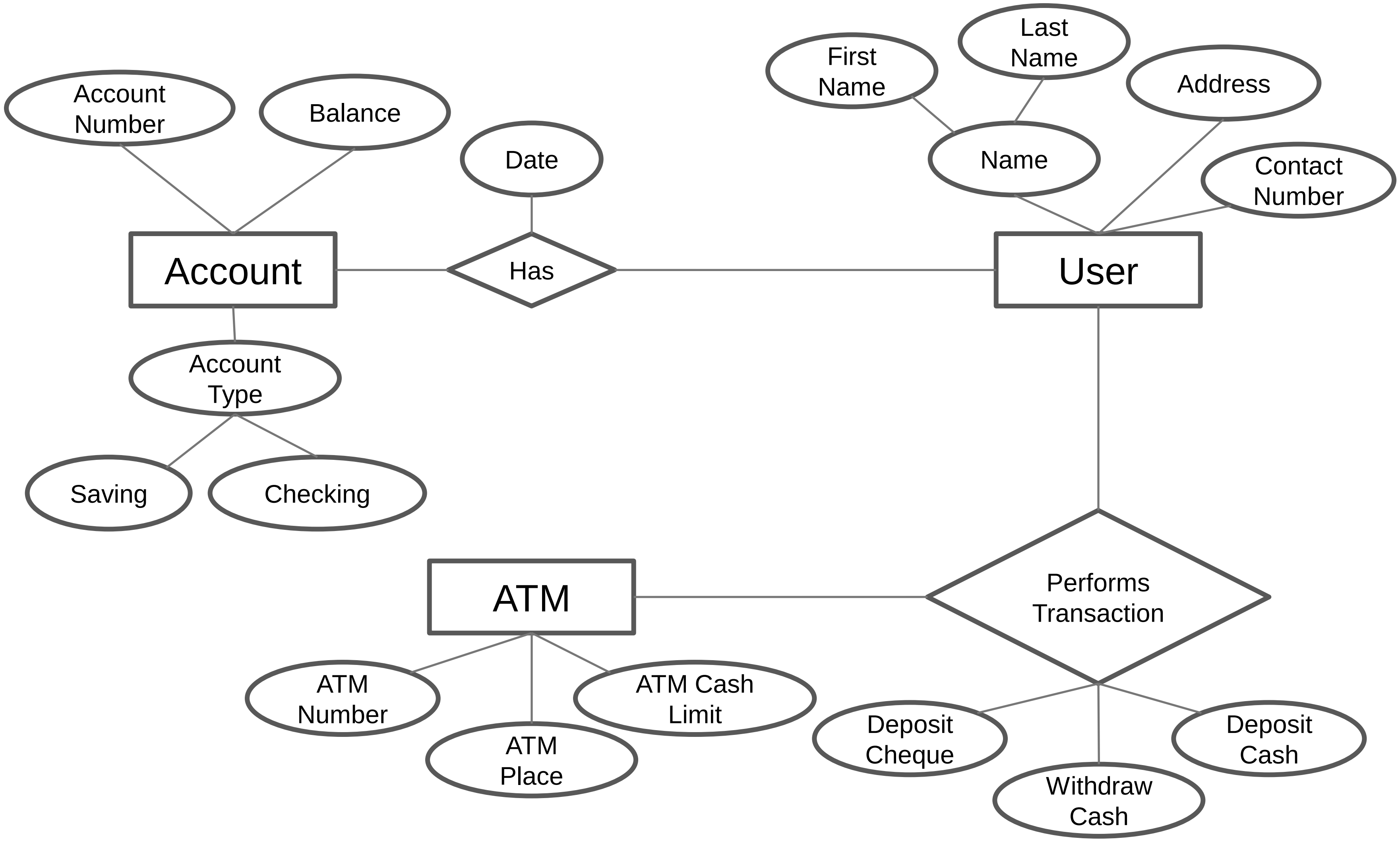
EXAMPLE– DOMAIN MODEL - UML CLASS DIAGRAM



EXAMPLE– DOMAIN MODEL - UML CLASS DIAGRAM



EXAMPLE – DOMAIN MODEL - ENTITY RELATIONSHIP MODEL



REQUIREMENTS

We will describe three types of requirements:

- Customer requirements (a.k.a informal or business requirements)
- Functional requirements
- Non-functional requirements

CUSTOMER REQUIREMENTS

We need to figure out exactly what the customer wants:

Requirements elicitation

- This is where the expectations of the customer are captured
- Done through interviews with stakeholders and analysing an existing system (whether it's manual or automated)
- Composed typically of high level, non-technical statements

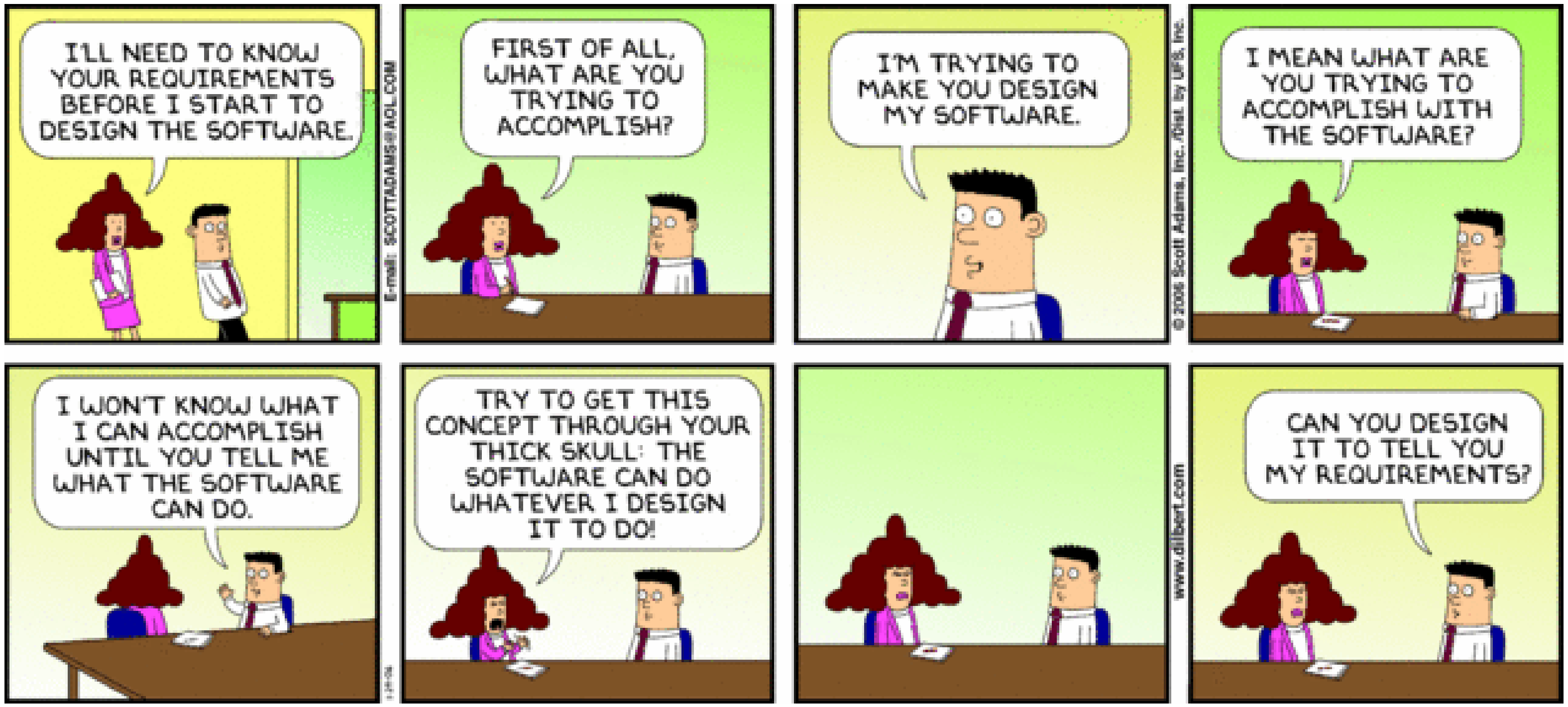
Example

- Requirement 1: "We need to develop an online customer portal"
- Requirement 2: "The portal must list all our products"
- …

Avoid adding design and implementation details

Consider only the functions the application must support

CUSTOMER REQUIREMENTS


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

FUNCTIONAL REQUIREMENTS

Capture the intended behavior of the system

- May be expressed as services, tasks or functions the system performs

Use cases have quickly become a widespread practice for capturing functional requirements

- This is especially true in the object-oriented community where they originated
- Their applicability is not limited to object-oriented systems

USE CASE VIEW

The use case view captures the behavior of a system or subsystem as it appears to an outside actor

It partitions the system functionality into transactions meaningful to actors

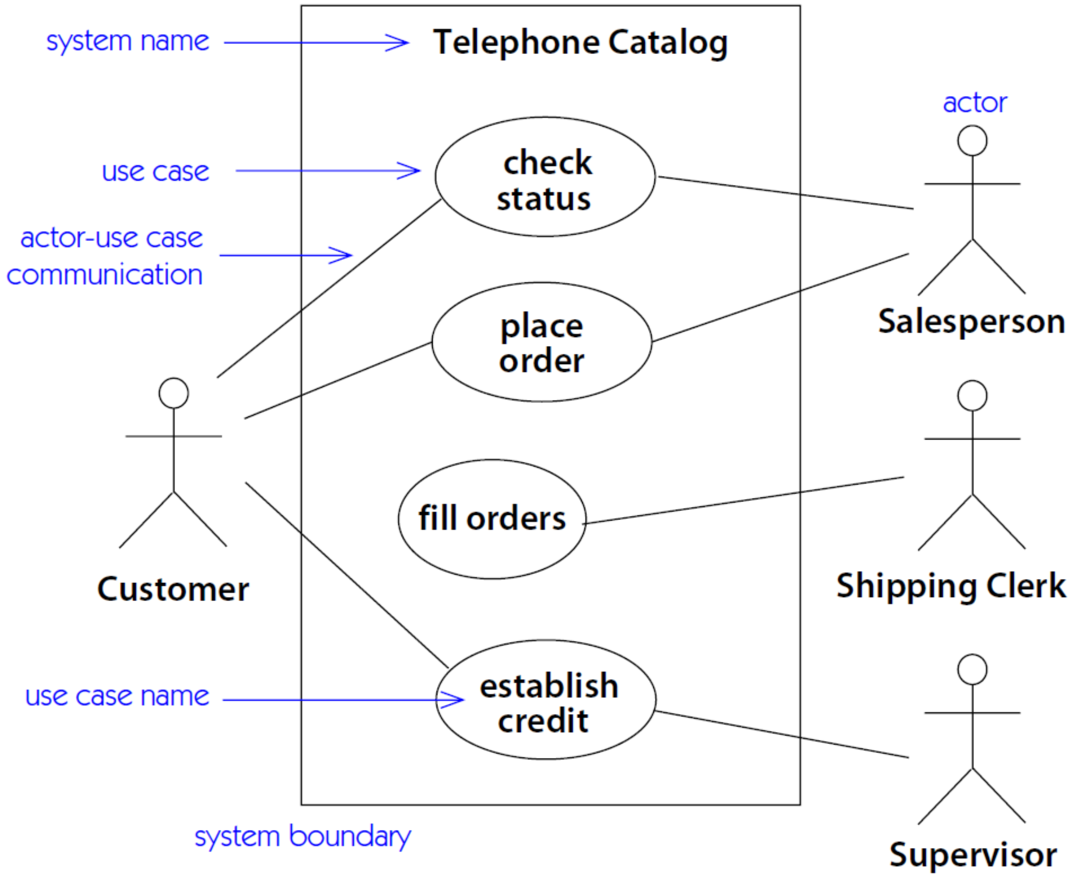
***Actors* are parties outside the system that interact with the system**

- An actor may be a class of users or other systems

A use case is initiated by an actor with a particular goal in mind, and completes successfully when that goal is satisfied

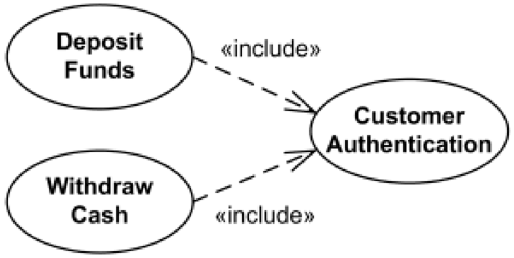
It describes the sequence of interactions between actors and the system necessary to deliver the service that satisfies the goal

USE CASE DIAGRAMS



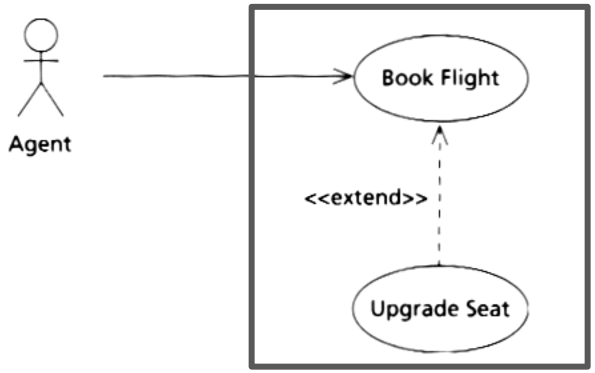
USE CASE RELATIONSHIPS

Include relationship: use case fragment that is *duplicated* in multiple use cases



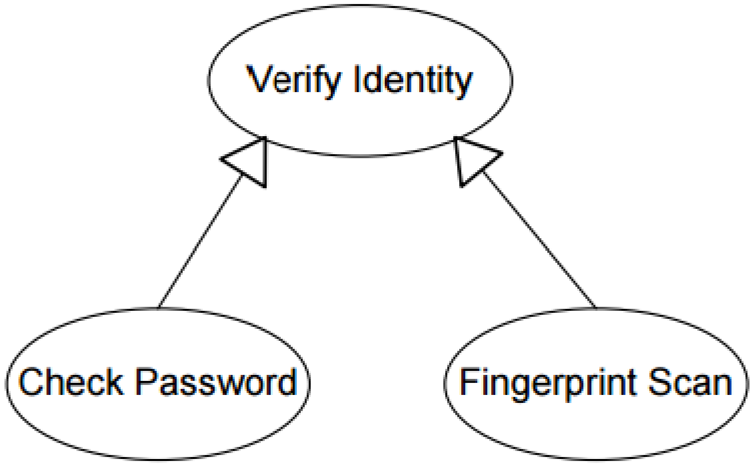
Extend relationship: use case conditionally adds steps to another first class use case

- Example:

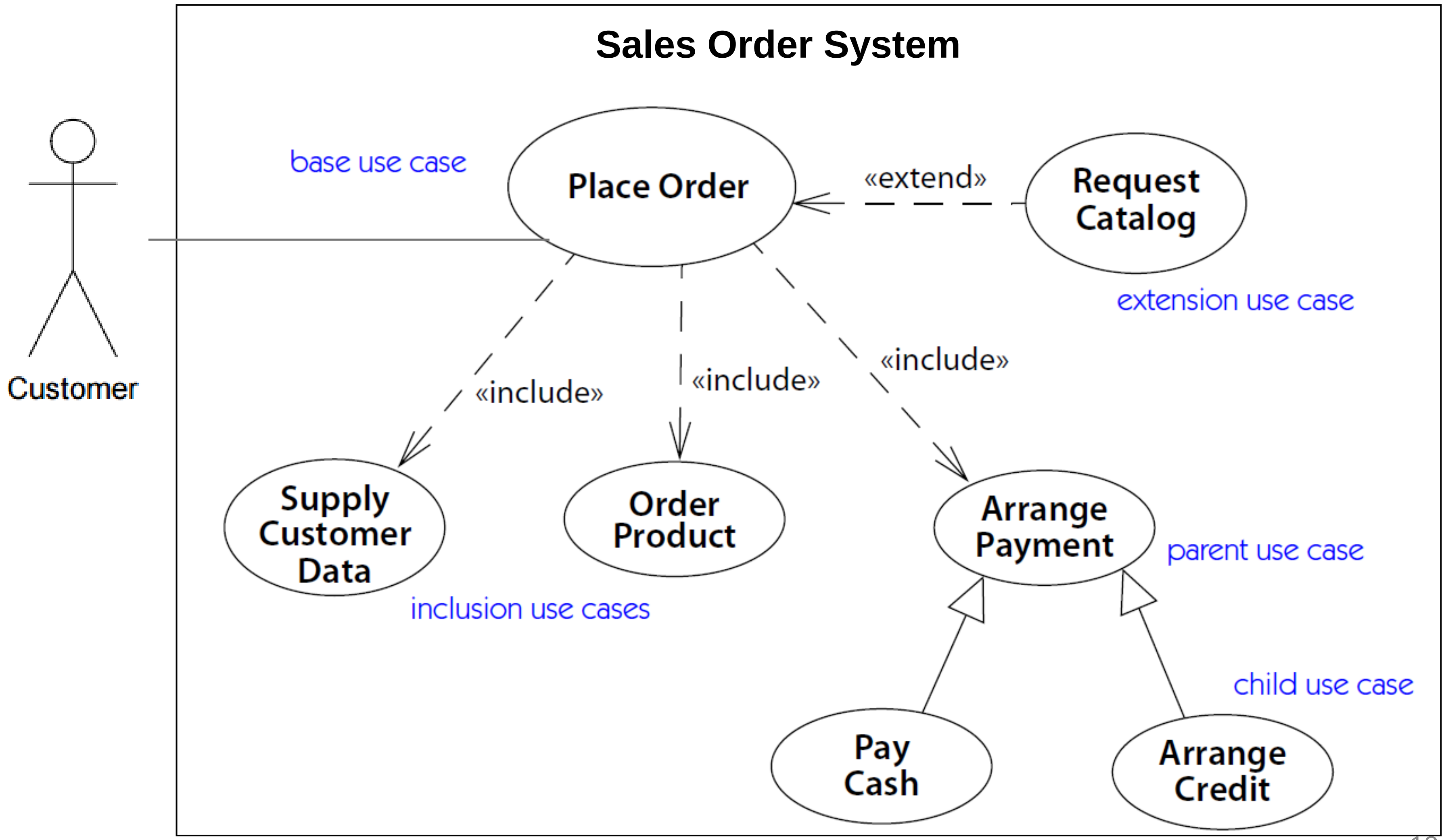


USE CASE RELATIONSHIPS

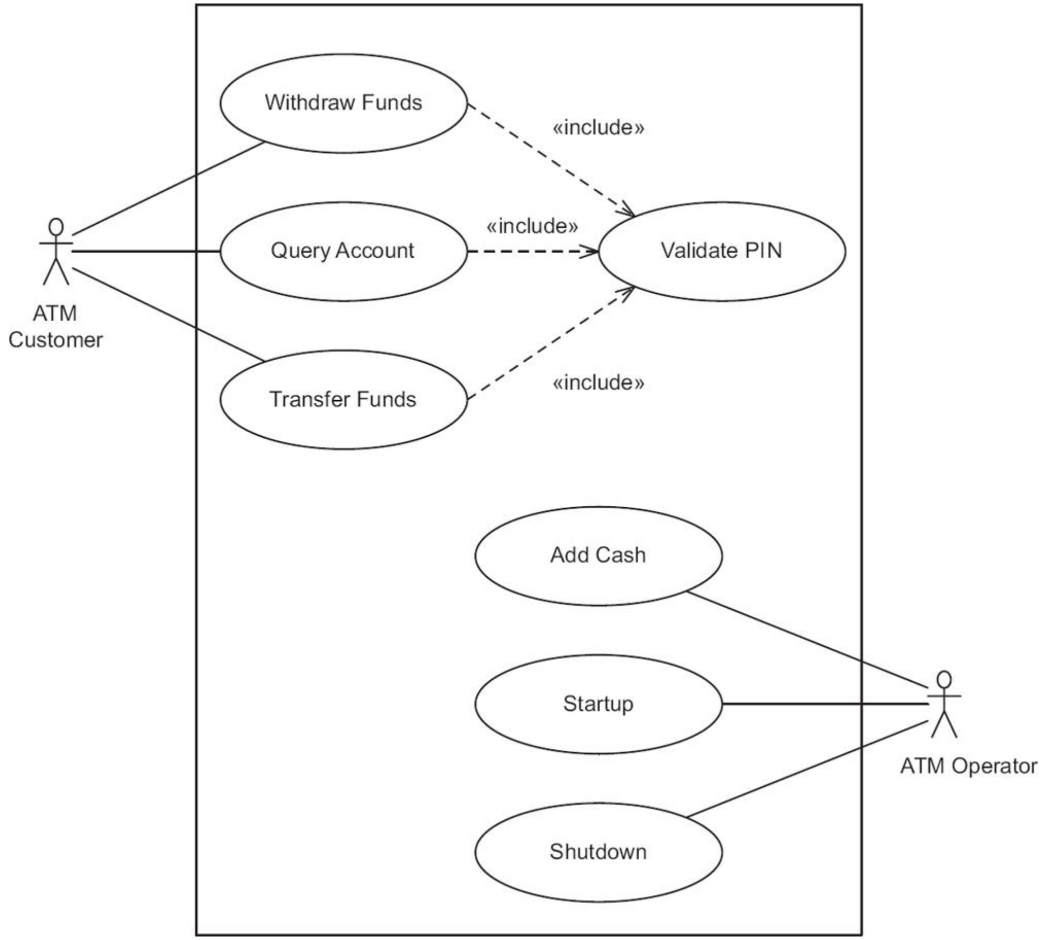
Generalization relationship: shows an either/or condition (one of the child use cases will be executed)



USE CASE RELATIONSHIPS



USE CASE – ATM EXAMPLE



USE CASE – ATM EXAMPLE

Actors:

- ATM Customer
- ATM Operator

Use Cases:

- The customer can
    - withdraw funds from a checking or savings account
    - query the balance of the account
    - transfer funds from one account to another
- The ATM operator can
    - Shut down the ATM
    - Replenish the ATM cash dispenser
    - Start the ATM

USE CASE – ATM EXAMPLE

Validate PIN is an *Inclusion Use Case*

- It cannot be executed on its own
- Must be executed as part of a *Concrete Use Case*

On the other hand, a *Concrete Use Case* can be executed

USE CASE – VALIDATE PIN (1)

Use case name: Validate PIN

Summary: System validates customer PIN

Actor: ATM Customer

Precondition: ATM is idle, displaying a Welcome message.

USE CASE – VALIDATE PIN (2)

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.

USE CASE – VALIDATE PIN (3)

Alternative sequences:

- **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 re-prompts 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.

Postcondition: Customer PIN has been validated.

USE CASE – WITHDRAW FUNDS (1)

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.

USE CASE – WITHDRAW FUNDS (2)

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.

USE CASE – WITHDRAW FUNDS (3)

Alternative sequences:

- **Step 3:** If the system determines that the account number is invalid, then it displays an error message and ejects the card.
- **Step 3:** If the system determines that there are insufficient funds in the customer's account, then it displays an apology and ejects the card.
- **Step 3:** If the system determines that the maximum allowable daily withdrawal amount has been exceeded, it displays an apology and ejects the card.
- **Step 5:** 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.

NON-FUNCTIONAL REQUIREMENTS

Functional requirements define what a system is supposed *to do*

Non-functional requirements define how a system is supposed *to be*

- Usually describe system attributes such as security, reliability, maintainability, scalability, usability…

NON-FUNCTIONAL REQUIREMENTS

Non-Functional requirements can be specified in a separate section of the use case description

- In the previous example, for the Validate PIN use case, there could be a security requirement that the card number and PIN must be encrypted

Non-Functional requirements can be specified for a group of use cases or the whole system

- **Security requirement:** System shall encrypt ATM card number and PIN.
- **Performance requirement:** System shall respond to actor inputs within 5 seconds.

SOFTWARE REQUIREMENTS SPECIFICATION

Combine the functional and non-function requirements into a specifications document

- You can also integrate the output of the domain analysis into the document

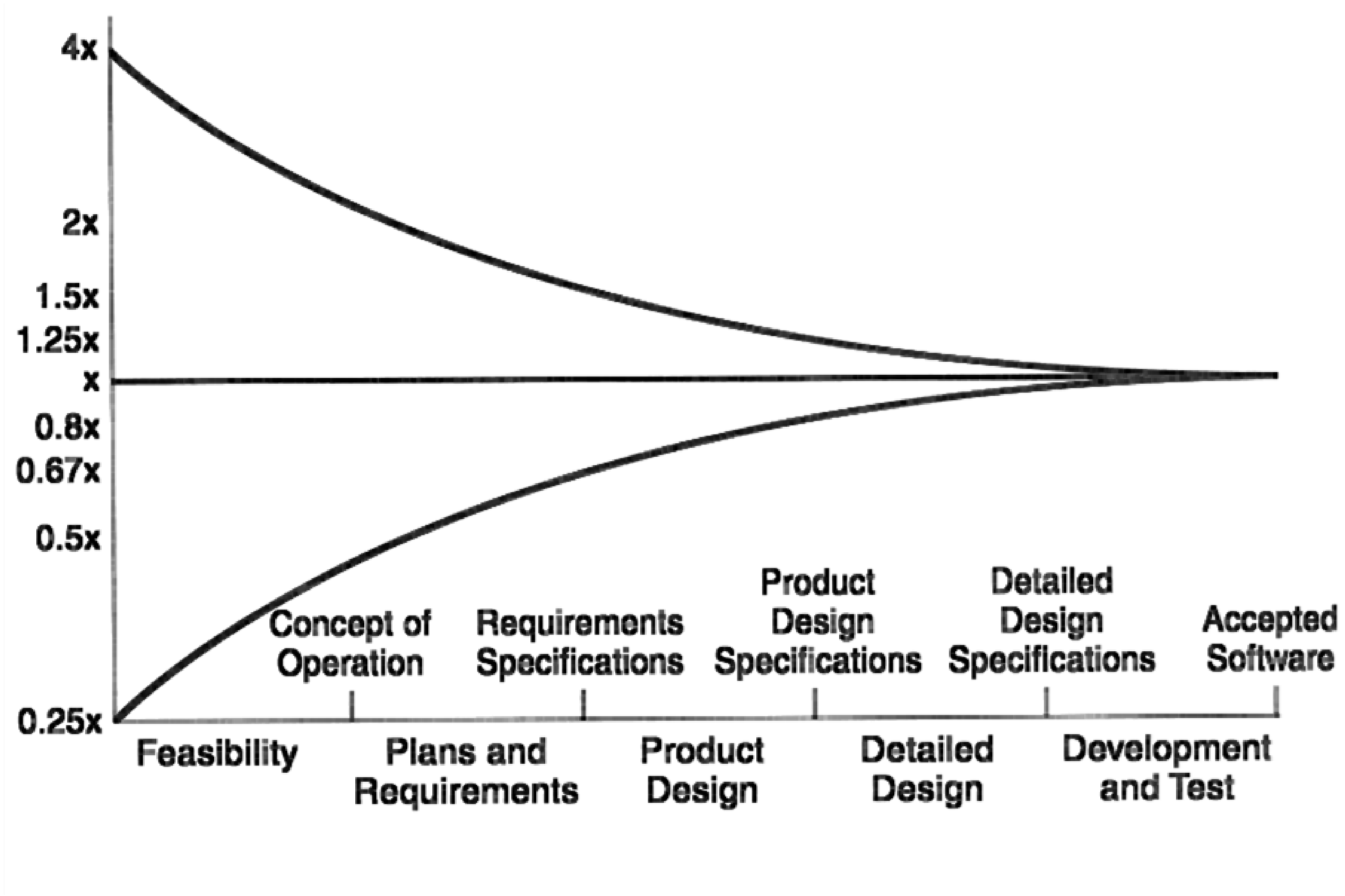
This document establishes the basis for the agreement between developer and customer

- **Do not proceed** until all stakeholders sign off on this document

This document provides a basis for estimating product costs, risks, and schedule

- *But shouldn't have you already estimated the cost (see next slide)?*

SOFTWARE SIZE ESTIMATION PRECISION



REQUIREMENTS IN AGILE PROCESSES

Requirements in SCRUM and XP Programing is captured using user stories

User Story is a high-level definition of a requirement

- Simple descriptions of a feature told from the perspective of the person who desires the new capability
- During product backlog population, they contain just enough information so that the developers can produce a reasonable estimate of the effort
- Later on, more details can be added in consultation with the customer

They typically follow a simple template:

As a <type of user>, I want <some goal> so that <some reason>

REQUIREMENTS IN AGILE PROCESSES

User stories can be written at varying levels of detail

A user story that covers a large amounts of functionality is typically known as an epic story

- An epic story cannot be completed in a single iteration
- It is decomposed into user stories that can be implemented in a single iteration

Example

- *Epic story:* As a user, I can backup my entire hard drive.
- User stories:
  - As a user, I can specify files or folders to backup based on file size, date created and date modified.
  - As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved.

THANK YOU!

QUESTIONS?