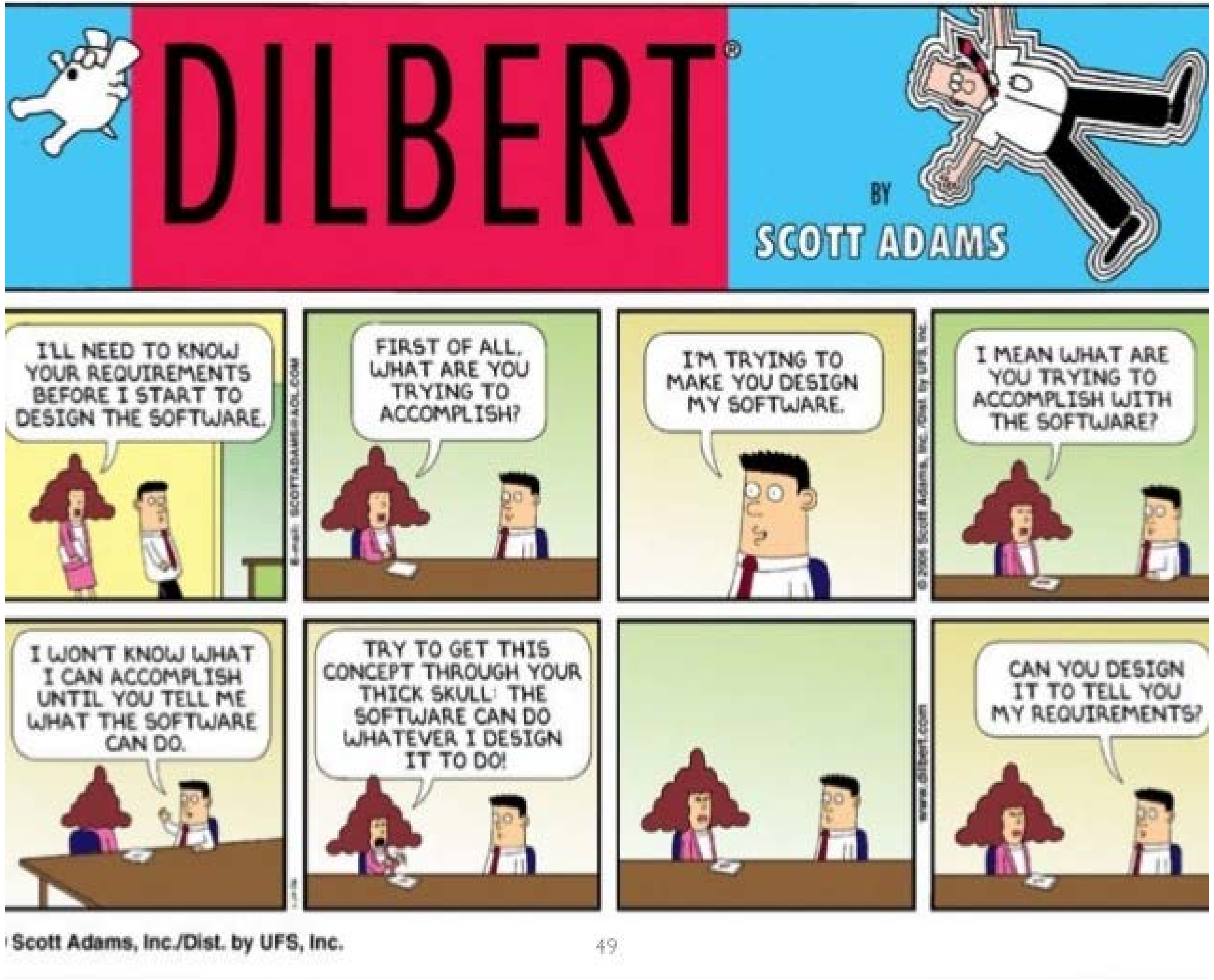
# CS350 Lecture 3-1 Requirements Engineering

Doo-Hwan Bae

bae@se.kaist.ac.kr

# A Big Problem!


DILBERT®
BY SCOTT ADAMS
I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
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!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
Scott Adams, Inc./Dist. by UFS, Inc.
49

## Requirements Engineering Overview (1/7)

- Must state what to do, not how to do.
- Identify the qualities required for the application.
- Functional and nonfunctional
- Used by both customer and designers
- Separate functional requirements into three views:
  - A model of data: Entity relationship diagrams, **domain model (class diagram)**
  - A model of function: Data flow diagrams, **use case diagram**
  - A model of control: Control flow diagrams, State Machines, Petri nets, **sequence diagram**

## Requirements Engineering Overview(2/8)

- Three steps in RE
  - Requirements Elicitation (gathering and analysis)

    -- understand/analyze the problem
  - Requirements Specification

    -- Once the problem is understood, it has to be specified

    -- Or, describe the product to be delivered

    -- Informal and formal methods, used.
  - Requirements validation and verification

    -- Once described, the different parties involved have to agree upon its nature.

    -- Validation: the correct R's are stated.

    -- Verification: The R's are stated correctly.

## Requirements Engineering Overview(3/8)

- Group User R’s into several categories (MoSCoW)
  - Must haves: Essential features
  - Should haves: Nice features
  - Could haves: if time allows
  - Won’t haves: not today

# Requirements Engineering Overview(4/8)

- Separate functional requirements into three views:
  - A model of data/static structure:
    - Entity relationship diagrams, **domain model (class diagram)**
  - A model of function:
    - Data flow diagrams, **use case diagram**
  - A model of control:
    - Control flow diagrams, State Machines, Petri nets, **sequence diagram**

## Requirements Engineering Overview(5/8)

- Other Requirements
  - Which machine, which OS, which DBMS, terminal?
  - Which classes of users can be distinguished?
  - What is the size of the DB, how is it expected to grow?
  - What response time should the system offer?
  - How much will a system of this kind cost?

# Requirements Engineering Overview(6/8)

## Seven Deadly Sins

- Noise: Irrelevant information, Confusing presentation
- Silence: Omissions
- Over-specification: Premature implementation decisions
- Contradiction: Inconsistency
- Ambiguity
- Forward reference
- Wishful thinking

## Requirements Engineering Overview(7/8)

Specification Qualities

- Clear, unambiguous, understandable
- Consistent
- Complete
- Functional
- Verifiable
- Traceable
- Easily changed

## Classification of Specification Methods

- Formal and informal
- Operational or descriptive
- Representative methods:
  - OO Analysis: object-oriented view
  - Structured Analysis: function-oriented view
  - Finite State Machines: State transition view
  - Petri-Nets: operation or control  view

## Examples: Wrong Specifications(1/2)

- Example:  "select" command in word processing.
  - ``Selecting is the process for designating areas of your document that you want to work on. Most editing and formatting actions require two steps:first you select what you want to work on, such as text or graphics, then you initiate the appropriate."
  - Q:  What are ambiguous?
- Example: Space shuttle monitoring system
  - ``The message must be triplicated. The three copies must be forwarded through three different physical channels.  The receiver accepts the message on the basis of a two-out-of-three voting policy."
  - Q: What is not clear?

# Examples: Wrong Specification(2/2)

- Example: A text editor
  - ``The whole text should be kept in lines of equal length, with the length specified by the user. Unless the user is given an explicit hyper-nation command, a carriage return should occur only at the end of a word."
  - Q: What is inconsistent?

## Object-Oriented Requirements Eng.

UML(Unified Modeling Language)

- Use case diagram
- Sequence diagram
- Class diagram
- ….

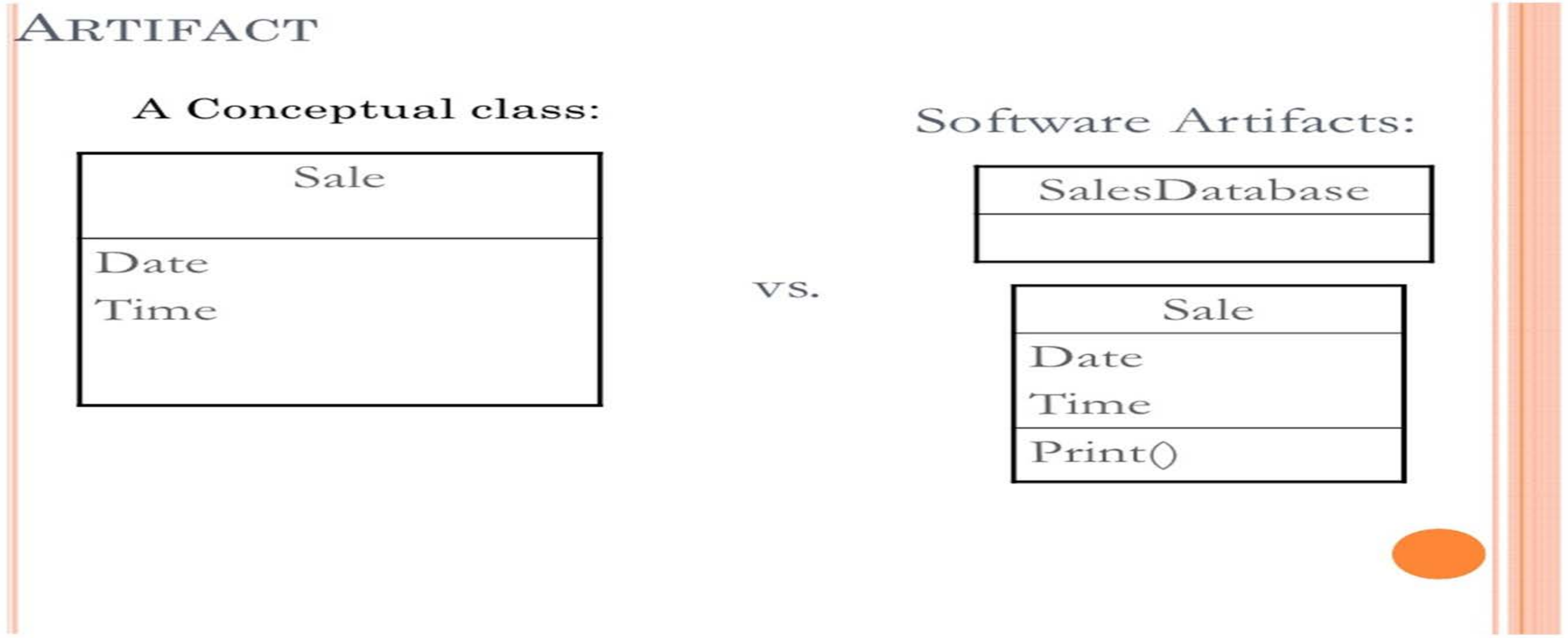
## Domain Modeling (Part of RE)

- Why? —The goal of domain modeling is to understand how system-to-be will work
  - Requirements analysis determined how users will interact with system-to-be (external behavior)
  - Domain modeling determines how elements of system-to-be interact (internal behavior) to produce the external behavior
- Where? —We do domain modeling based on  the following sources:
  - Knowledge of how system-to-be is supposed to behave (from requirements analysis, e.g., use cases)
  - Studying the work domain (or, problem domain)
  - Knowledge base of software designs
  - Developer’s past experience with software design

# Domain Modeling(Part of RE)

- What?
  - Illustrates meaningful *conceptual classes* in a problem domain
  - Is a representation of real-world concepts, not software artifacts
  - Is the most important objects/classes in OO development
    - Decompose the system into a set of  objects, rather than functions!
- How?
  - Identify nouns and noun phrases in textual descriptions of the domain
    - Use use cases in use-case diagrams
  - Add associations for relationships that must be retained
  - Add attributes necessary for information to be preserved

# Conceptual Class vs. Software Artifacts

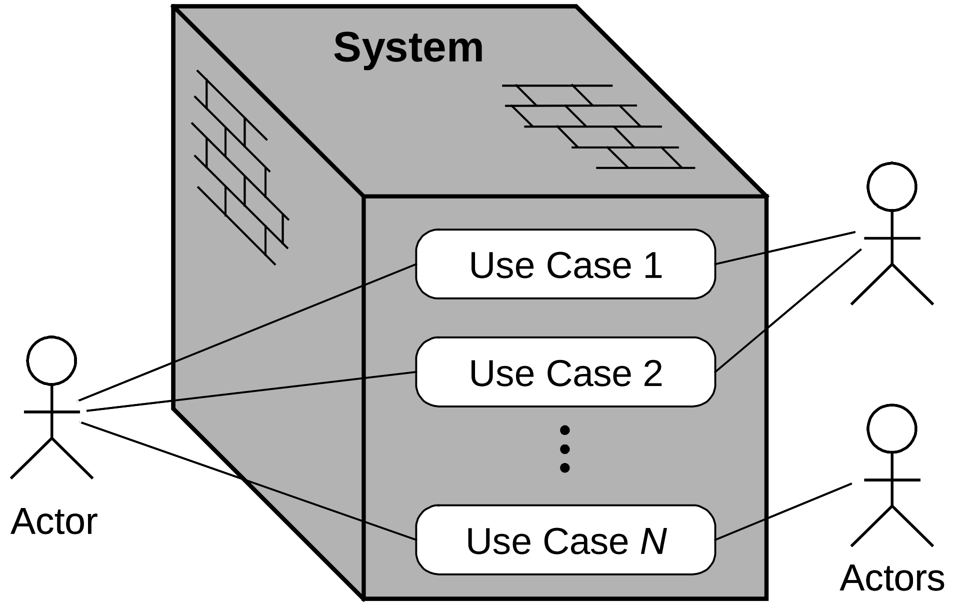


# Use Case Analysis vs. Domain Analysis

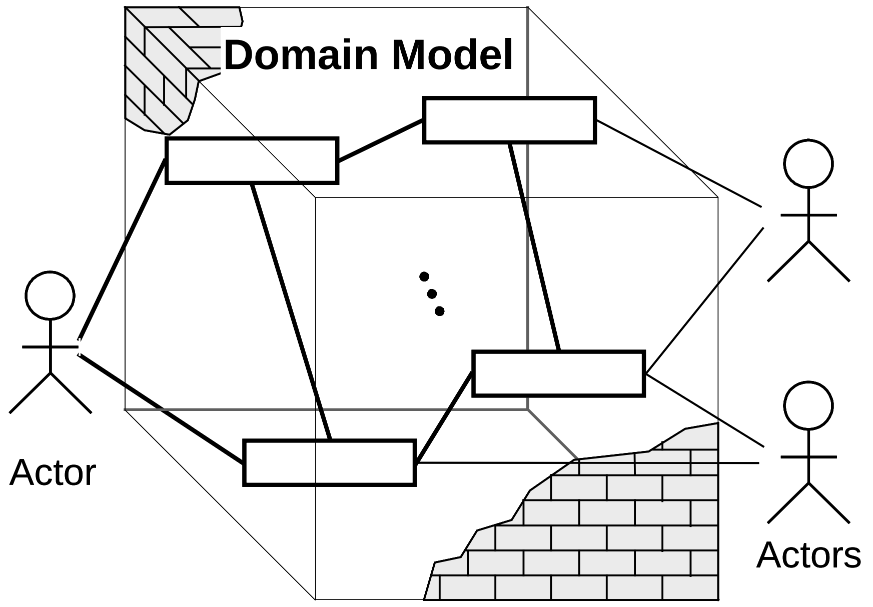
In **use case analysis**, we consider the system as a "**black box**"

In **domain analysis**, we consider the system as a "**transparent box**"

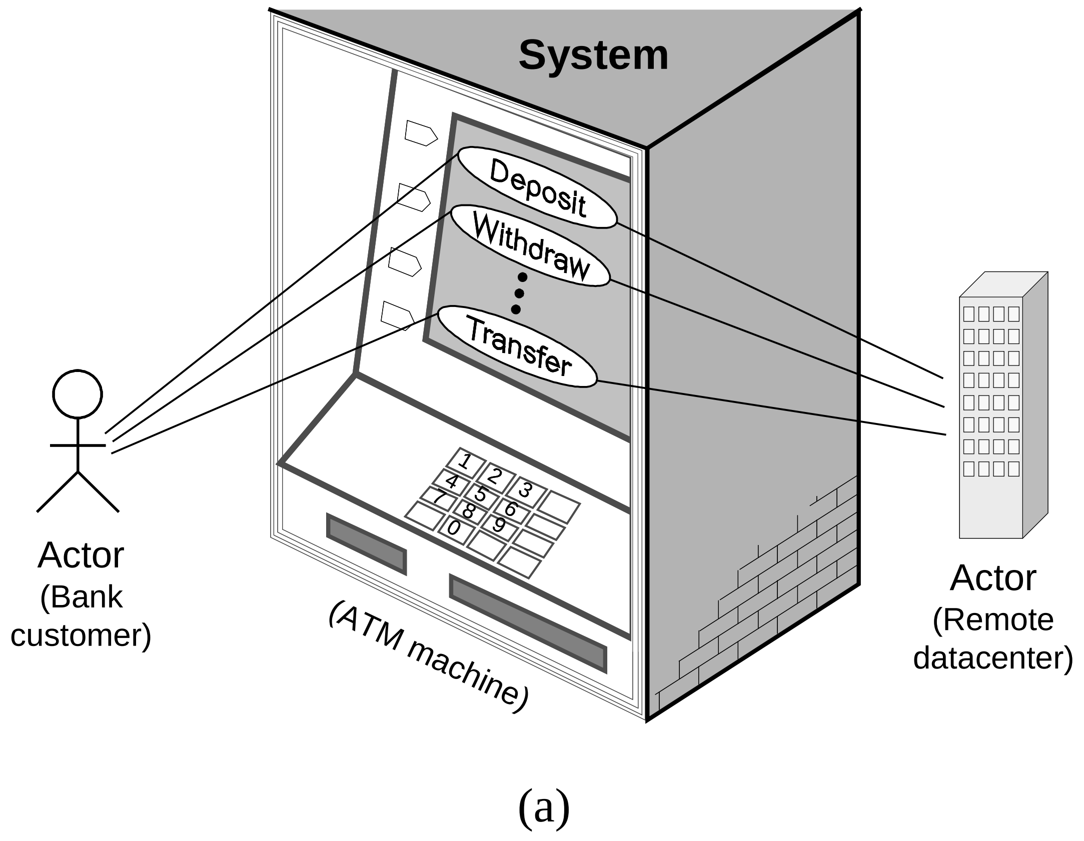
(a)

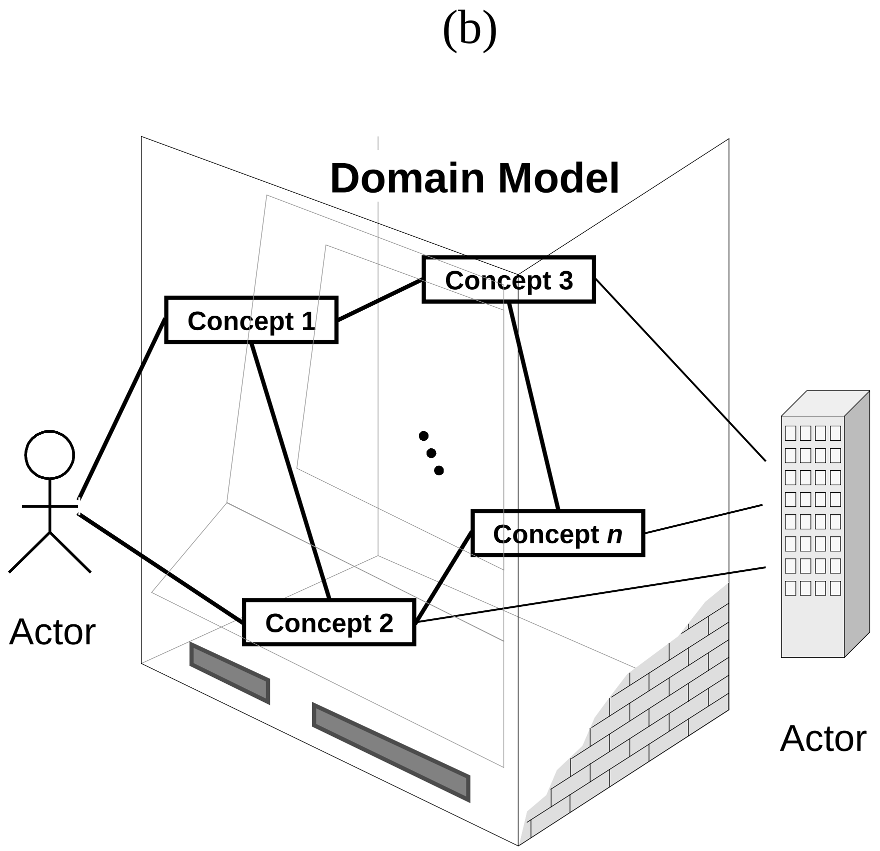


(b)



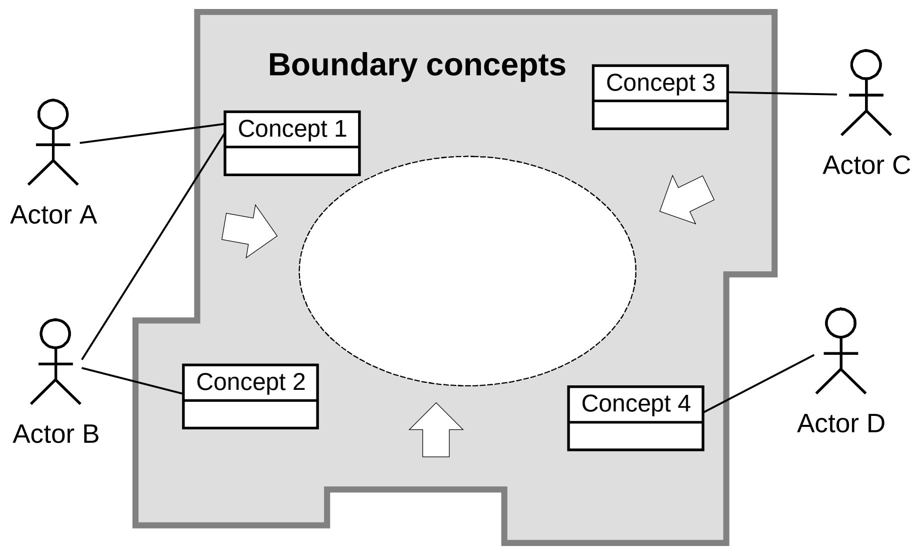
# Example: ATM Machine



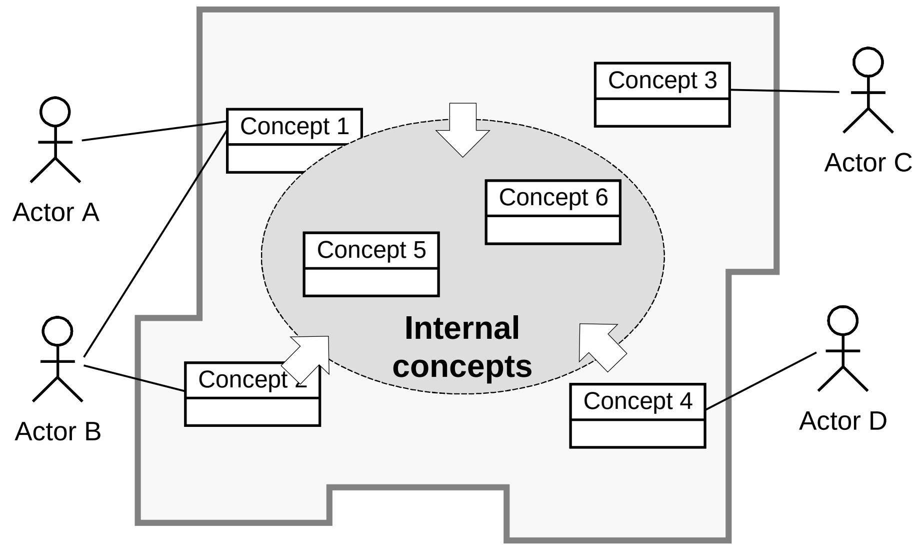


# Building Domain Model from Use Cases

Step 1: Identifying the boundary concepts



Step 2: Identifying the internal concepts



## Initial ATM Requirements ( Problem )

- A customer uses a bank ATM for checking balances of bank accounts, depositing funds, withdrawing cashes, and transferring funds.
- A bank ATM consists of a keypad, screen, cash dispenser, ATM card reader, cash deposit slot.
- A user can have a checking account and saving account.
- A customer enters his/her card and password for authentication and then choose a transaction to be executed.  After the transaction is over, the ATM returns the card and receipt, and stores the transaction for bookkeeping.
- Once the request from the ATM is passed, the bank checks the account and process a requested transaction and update the account.

## Identify Use Cases

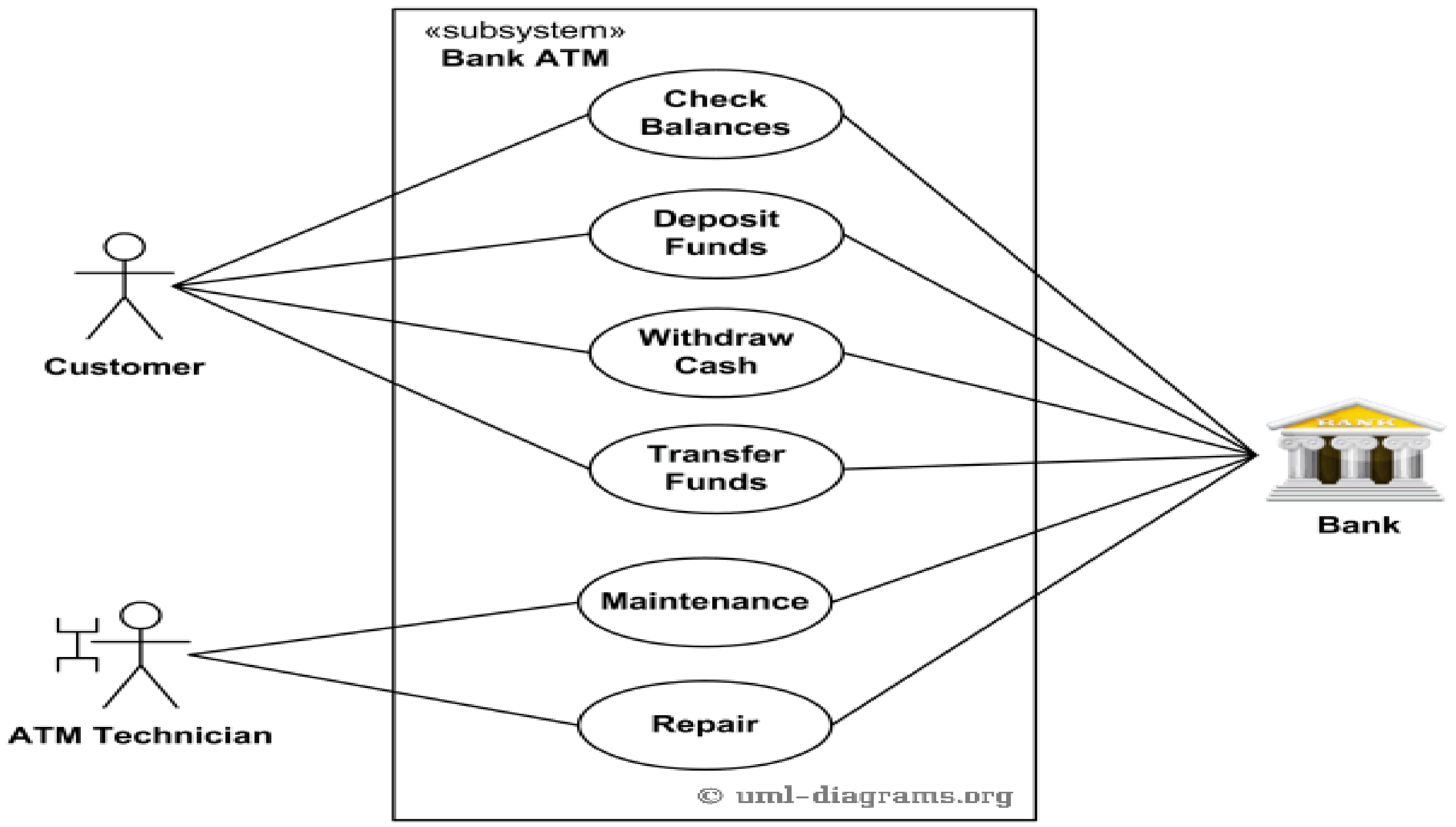
- Identify actors first
    - Actor: a person/thing that initiate actions to a system or receives results of actions from the system
- Use cases are "Why the customer goes to the ATM?"
- Find action terms from the initial requirements
    - Check balances
    - Deposit funds
    - Withdraw cashes
    - Transfer funds
    - Enters card/password
    - Authenticate
    - Choose a transaction
    - Return a card..
    - Check, process, update

## Use Case Diagram for ATM

- Let us  try
    - Identify actors, use cases, system boundary

# Top-level Use Cases

- Technician?



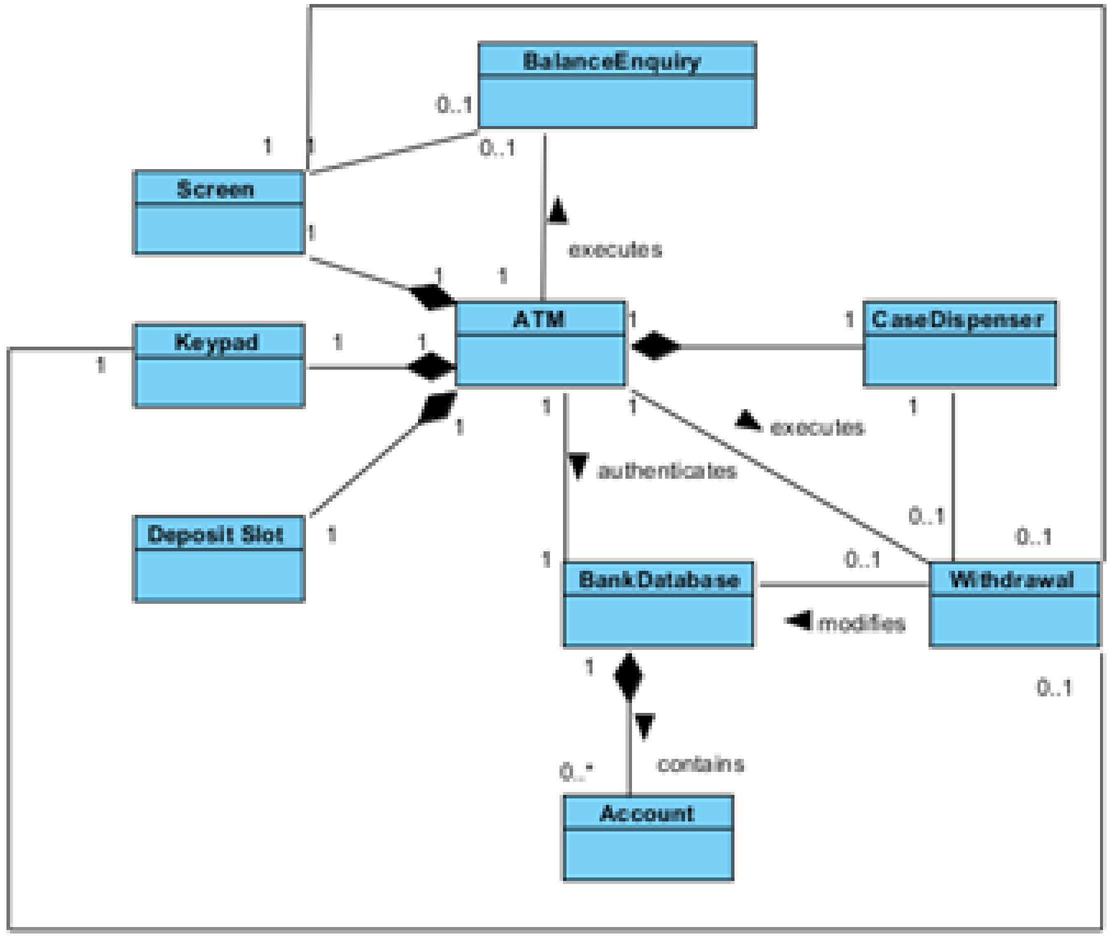
## Identify the classes

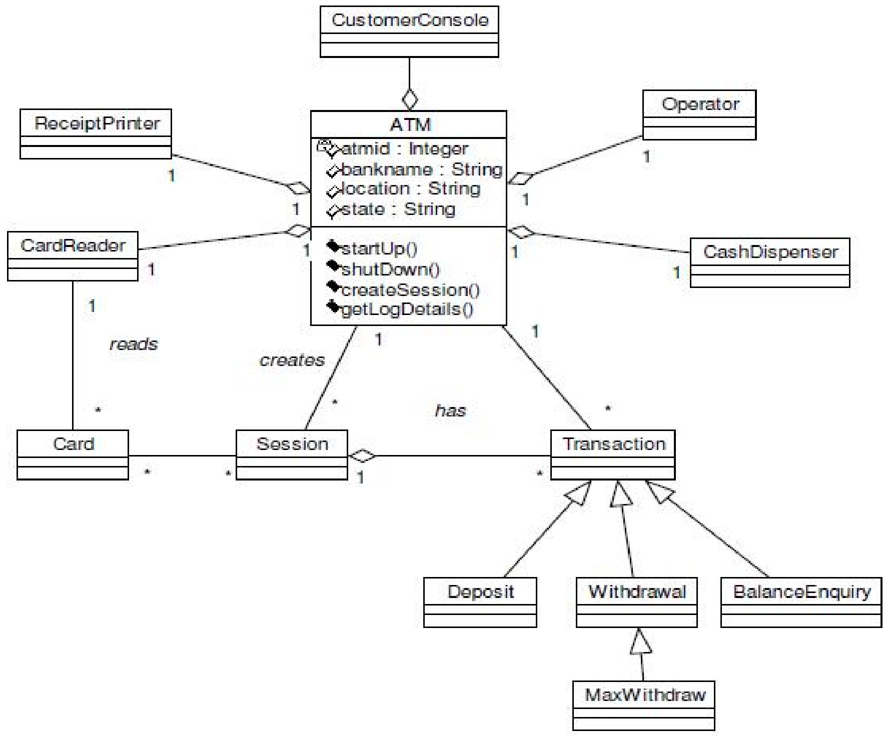
- Identify all the nouns or noun phrases.
  - Customer, bank, ATM, (bank) account, funds, cash, keypad, screen, cash dispenser, cash deposit slot,  user, checking account, saving account, card, password, transaction, receipt, bookkeeping, request.
- Eliminate redundant, Irrelevant, implementation constructs, attributes, etc.

# Initial Class Diagram (Domain Model)

- Objects: ATM, Keypad, Deposit slot, Cash dispenser, Screen, Transaction, Bank, Account, Customer
- Draw a class diagram with only classes and relationships

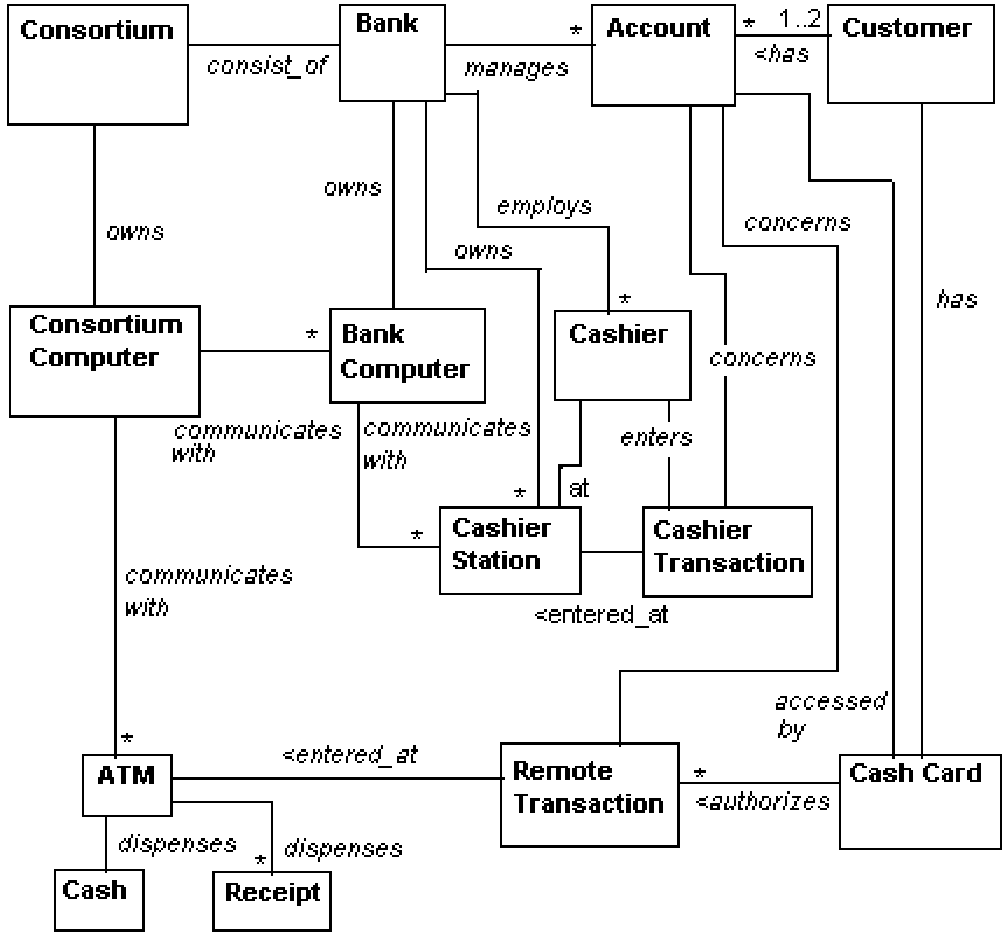
# Some Options?





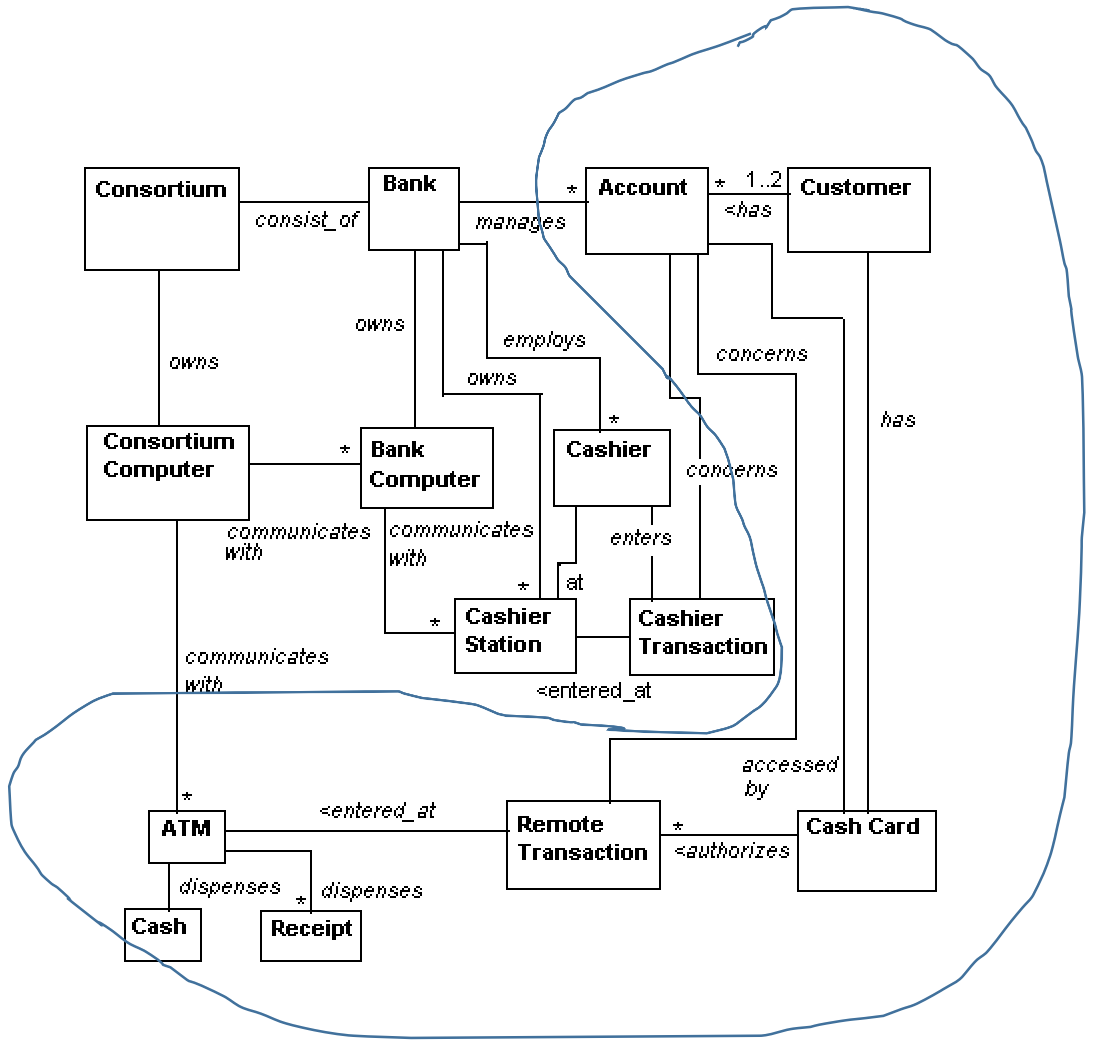
# Domain Model for an ATM & Bank

- What's the difference?

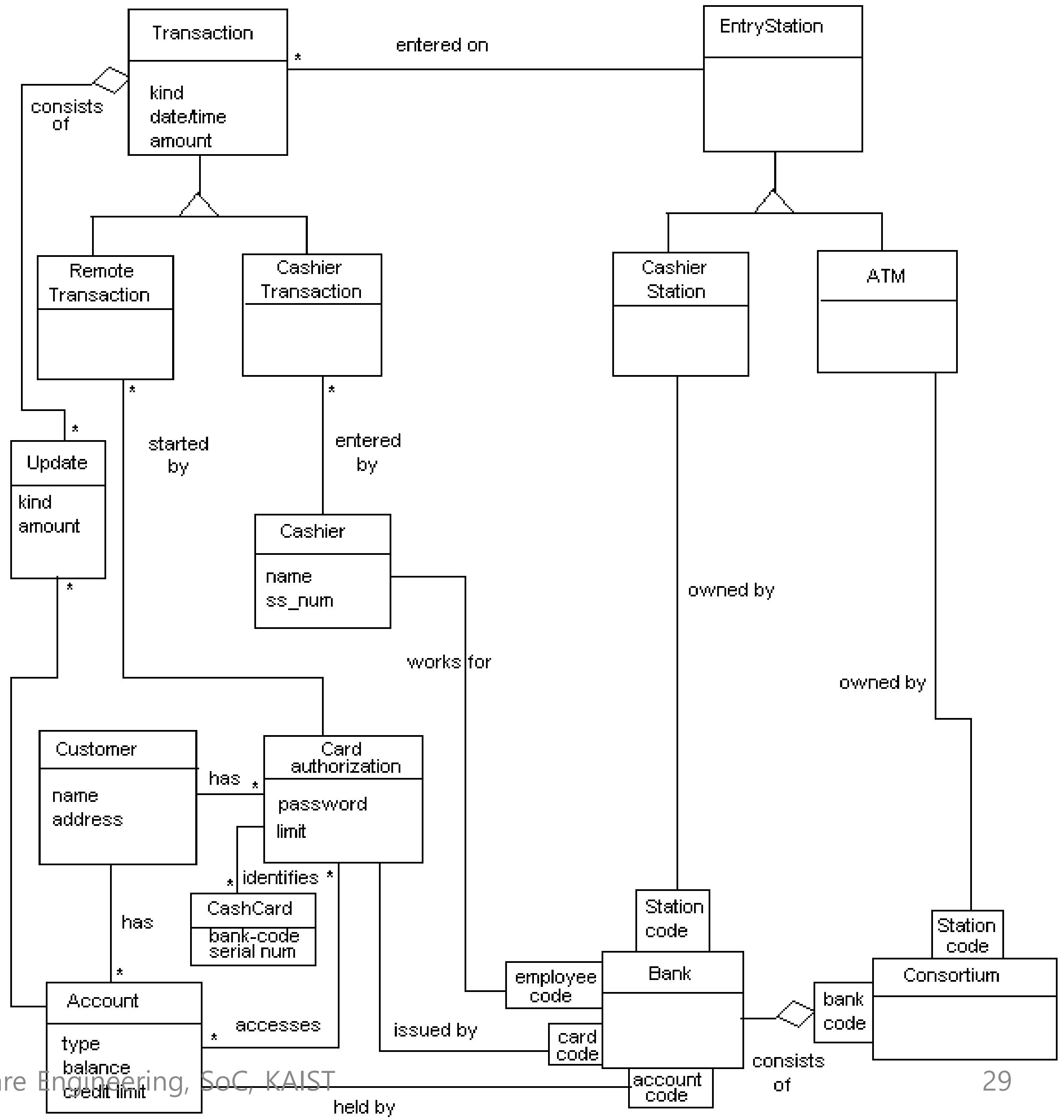


# Domain Model for an ATM & Bank

- What's the difference?

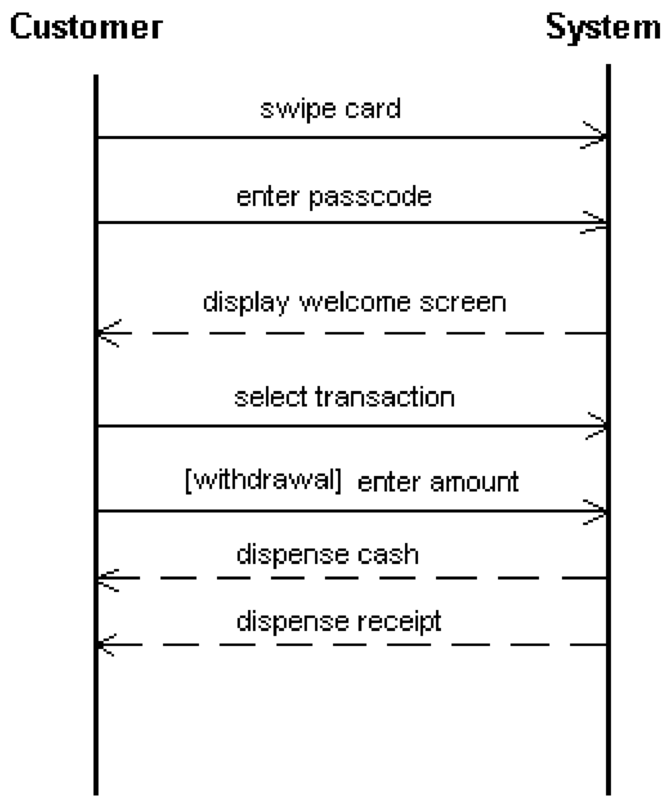


# Refined Class D.



## High Level Sequence Diagram

- You can have more SDs…



# Refining Diagrams

- Now we have
  - Use case diagram
  - Initial class diagram (domain diagram)
  - Sequence diagram
- Once, the initial requirements analysis/specification is over, refining those diagrams until ..
- Then, verify the results
- Also, identify Non-functional requirements
- ….