COMP1023 Software Engineering

Spring Semester 2019-2020

Dr. Radu Muschevici

School of Computer Science, University of Nottingham, Malaysia

Lecture 4
2020-02-26

Topics covered

System Modelling

A Interaction Models
- Use Cases
- Sequence Diagrams

System Modeling

Loosely based on
    Sommerville, Software Engineering, 10th Ed., 2015

Modeling

- A **model** is not a complete representation of a system. It purposely **leaves out detail** to make it easier to understand.
- **Abstraction** is key: deliberately simplify and pick out only the most significant characteristics of a system.

System modeling

- The process of developing **abstract models of a system**, with each model presenting a different view or perspective of that system.
- Often done by using some kind of graphical notation – nowadays almost always based on the Unified Modeling Language (UML).
- Helps the software engineer to **understand** the **functionality** of the system and to **communicate** with customers.

System perspectives

- ✧ **Interaction**
  - model the interactions between a **system and its environment**, and between the **components** of a system.
- ✧ **Structural**
  - model the **organization** of a system or the **structure of the data** that is processed by the system.
- ✧ **Behavioral**
  - model the **dynamic behavior** of the system and how it **responds** to events.

The Unified Modeling Language (UML)

- Current **standard approach** for developing models of software systems (focus on object-oriented systems)
- 13 diagram types used to model **different aspects** of software systems.
- Three main perspectives:
  - **Requirements:** use-case diagrams
  - **Static:** object and composite structure diagrams
  - **Dynamic:** sequence and state diagrams

Five UML diagram types (the most useful/used)

- ✧ **Use case diagrams**, which show the interactions between a system and its environment.
- ✧ **Sequence diagrams**, which show interactions between actors and the system and between system components.
- ✧ **Activity diagrams**, which show the activities involved in a process or in data processing.
- ✧ **Class diagrams**, which show the object classes in the system and the associations between these classes.
- ✧ **State diagrams**, which show how the system reacts to internal and external events.

Use of graphical models

- ✧ As a means of **facilitating discussion** about an existing or proposed system
  - ▪ Incomplete and incorrect models are OK as their role is to support discussion.
- ✧ As a way of **documenting** an existing system
  - ▪ Models should be an accurate representation of the system but need not be complete.
- ✧ As a detailed system description that can be used to generate a system implementation
  - ▪ Models have to be both correct and complete.

Interaction models

Interaction models

- ✧ Modeling user interaction is important as it helps to identify user **requirements**.
- ✧ Modeling **system-to-system interaction** highlights the communication problems that may arise.
- ✧ Modeling **component interaction** helps understand if a proposed system structure is likely to deliver the required system performance and dependability.
- ✧ **UML use case diagrams** and **sequence diagrams** may be used for interaction modeling.

Interaction modeling: Use Cases

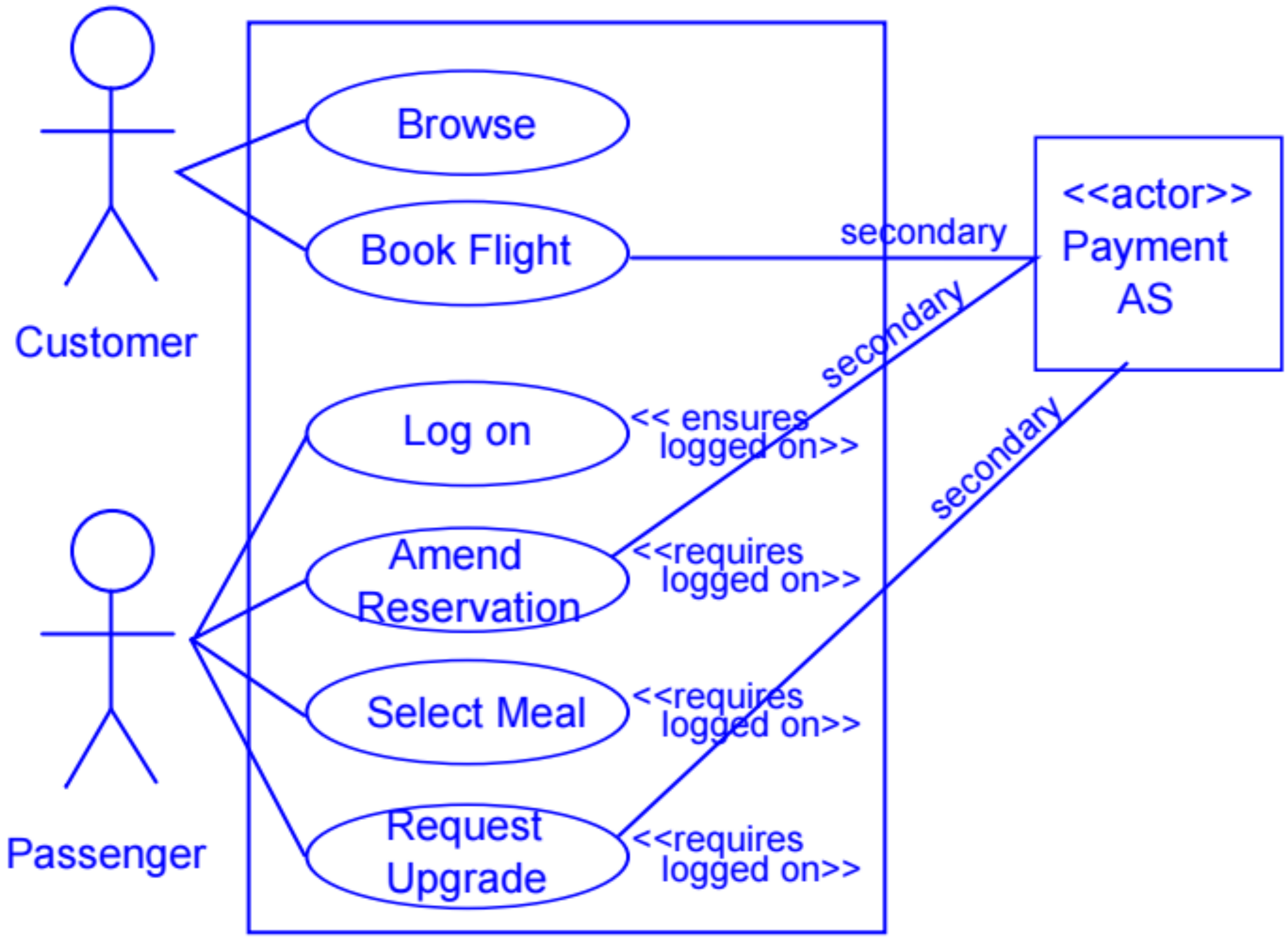
- Typically used to discover system requirements during **requirement analysis,** and for **high level design**.
- Each **use case** represents one **discrete task** and describes how a system is expected to interact with a user (or possibly another system).
- **Actors** in a use case may be people or other (external) systems.
- Representation:
  - Diagram provides **overview** of the use case.
  - Textual form provides more **detail.**

Use Case Diagrams

✧ Elements:

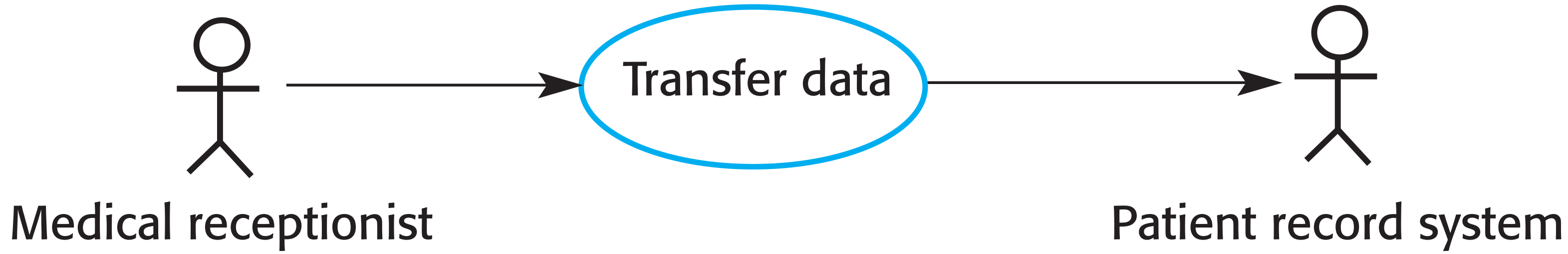
- **Actor**
    - Human actors are usually represented by a stick figure. Rectangular boxes are used for other connected systems.
- **Use Case**
    - Representation of a distinct business functionality. Shown as an ellipse.
- **System boundary**
    - Defines the scope of the system. Shown as a rectangle spanning all use cases in the system.
- **Relationship**
    - Dependency between use cases or actors (e.g. generalisation, inclusion, extension). Shown as (labelled) arrows.

Example Use Case Diagram



Another example: ‘Transfer data’

- A use case in the Mentcare system

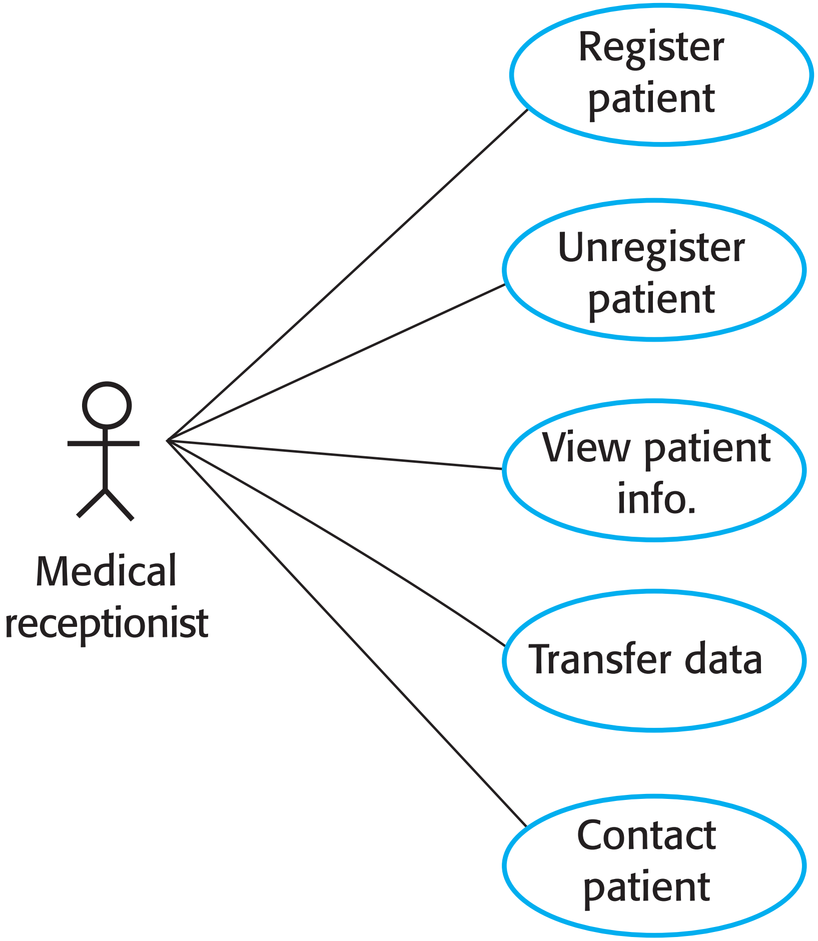


- Notes
  - Stick figure notation is sometimes used to represent other (non-human) external systems.
  - Formally, use case diagrams should use lines without arrows as arrows in the UML indicate the direction of flow of messages. Obviously, in a use case, messages pass in both directions. The arrows here are used **informally** to indicate that the receptionist initiates the transaction and data is transferred to the patient record system.

Tabular description (body) of the ‘Transfer data’ use case

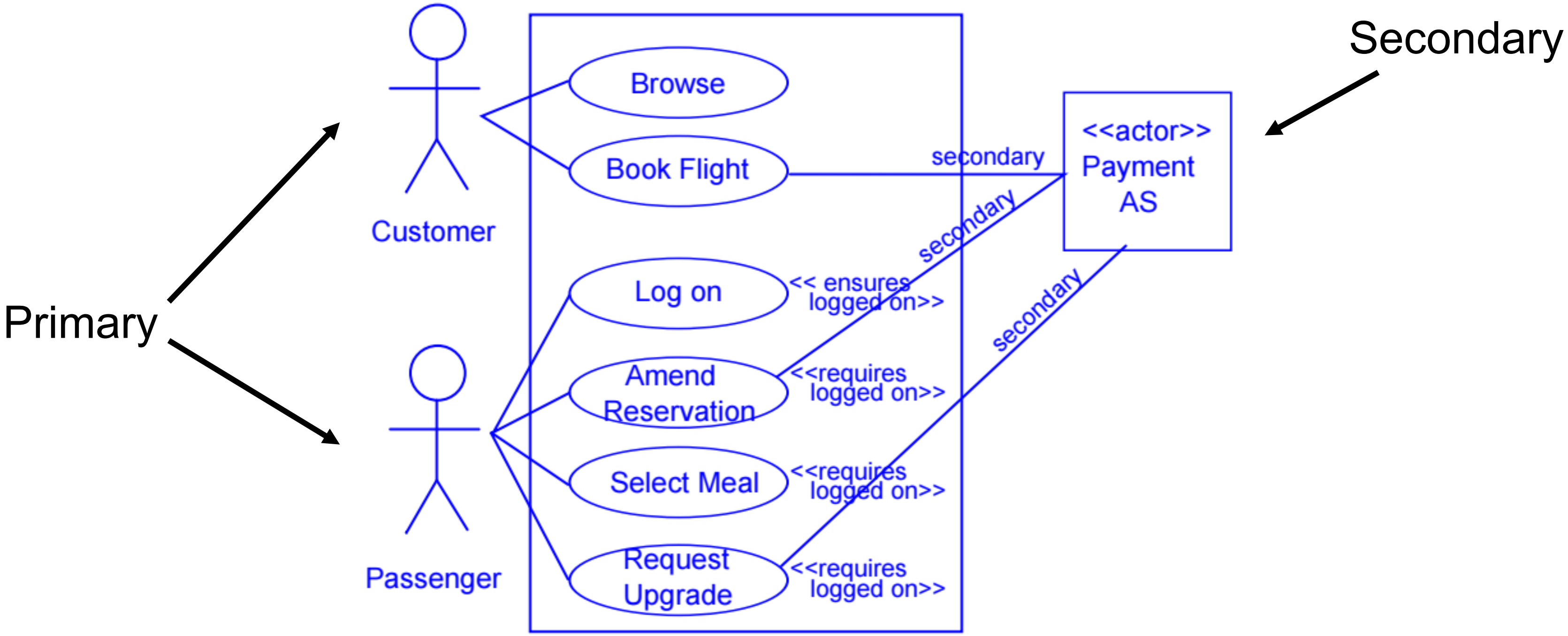
MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	user command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Use cases in the Mentcare system involving the role ‘Medical Receptionist’



Primary and secondary actors

- ✧ **Primary actors:** Use the system to achieve a goal.
- ✧ **Secondary actors:** Used by the system in order to achieve the primary actor's goal.



Modeling system interaction with Use Case diagrams

✧ Prerequisites:

- Clarify your **functional requirements**
- Identify the **actors** (work out who/what interacts with the system).

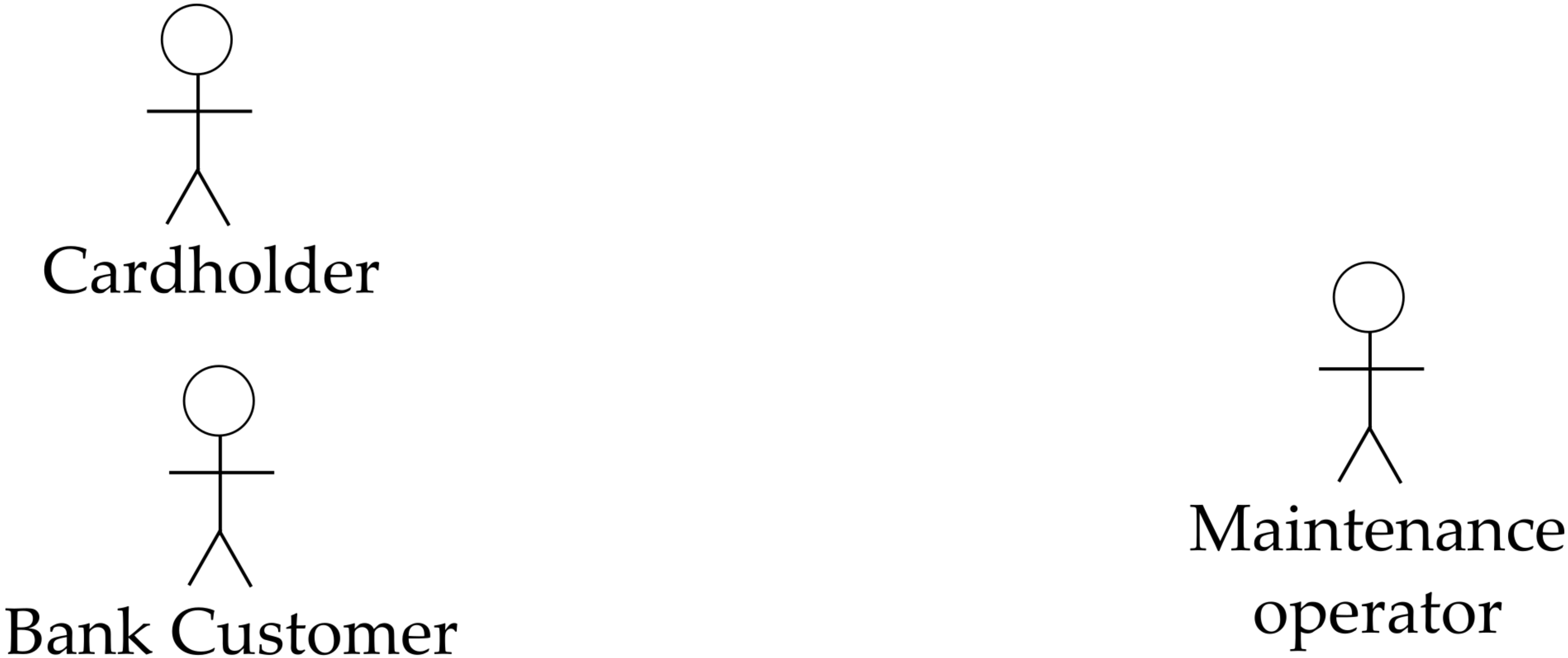
✧ Steps:

1. Add primary actors
2. Add functional requirements as **use cases**
3. Map actors to use cases
4. Add system boundary
5. Add secondary actors

An Example

You have been contracted to design a new ATM system for UNMBank. UNMBank is a new bank on campus and you have been asked to build an interactive ATM system that can handle multiple types of transactions (e.g. withdraw cash, check balance, make deposits) securely.

Example (1): Primary actors

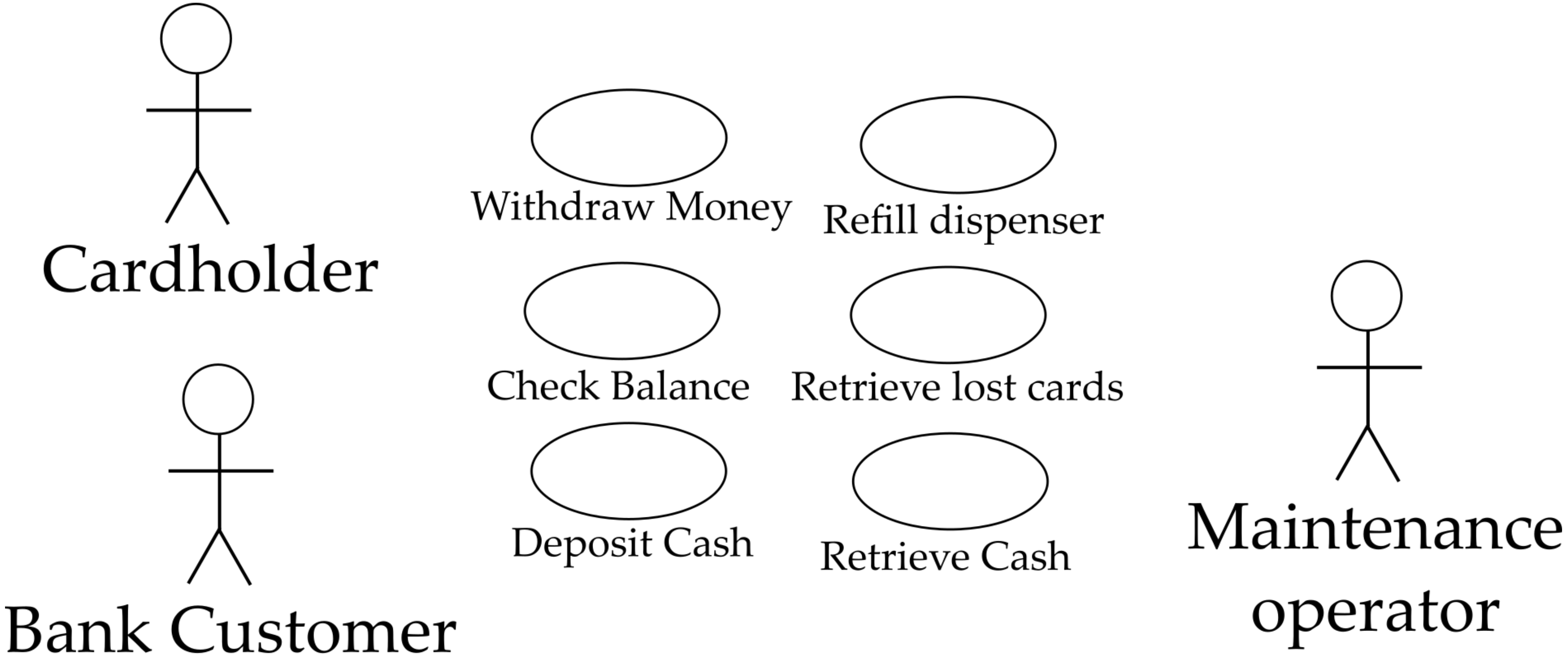


Cardholder

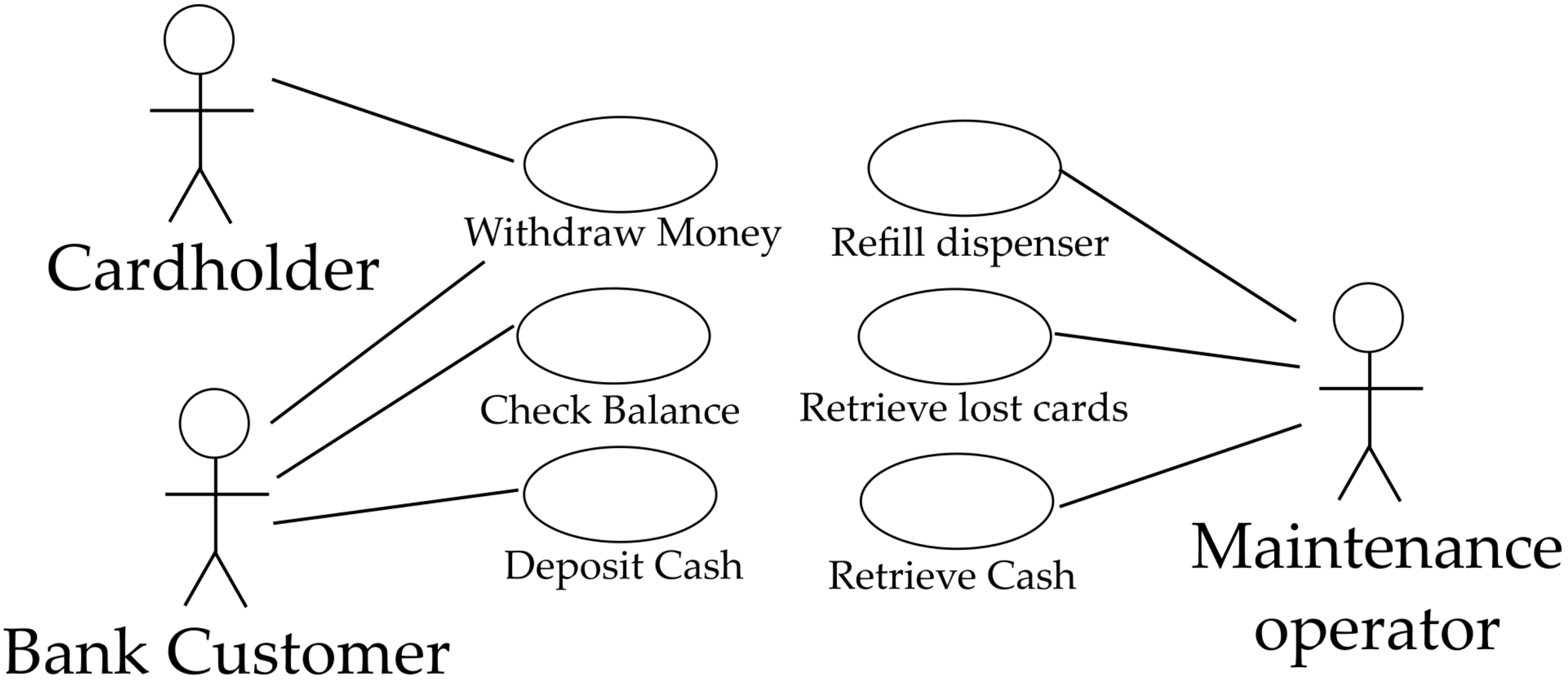
Bank Customer

Maintenance operator

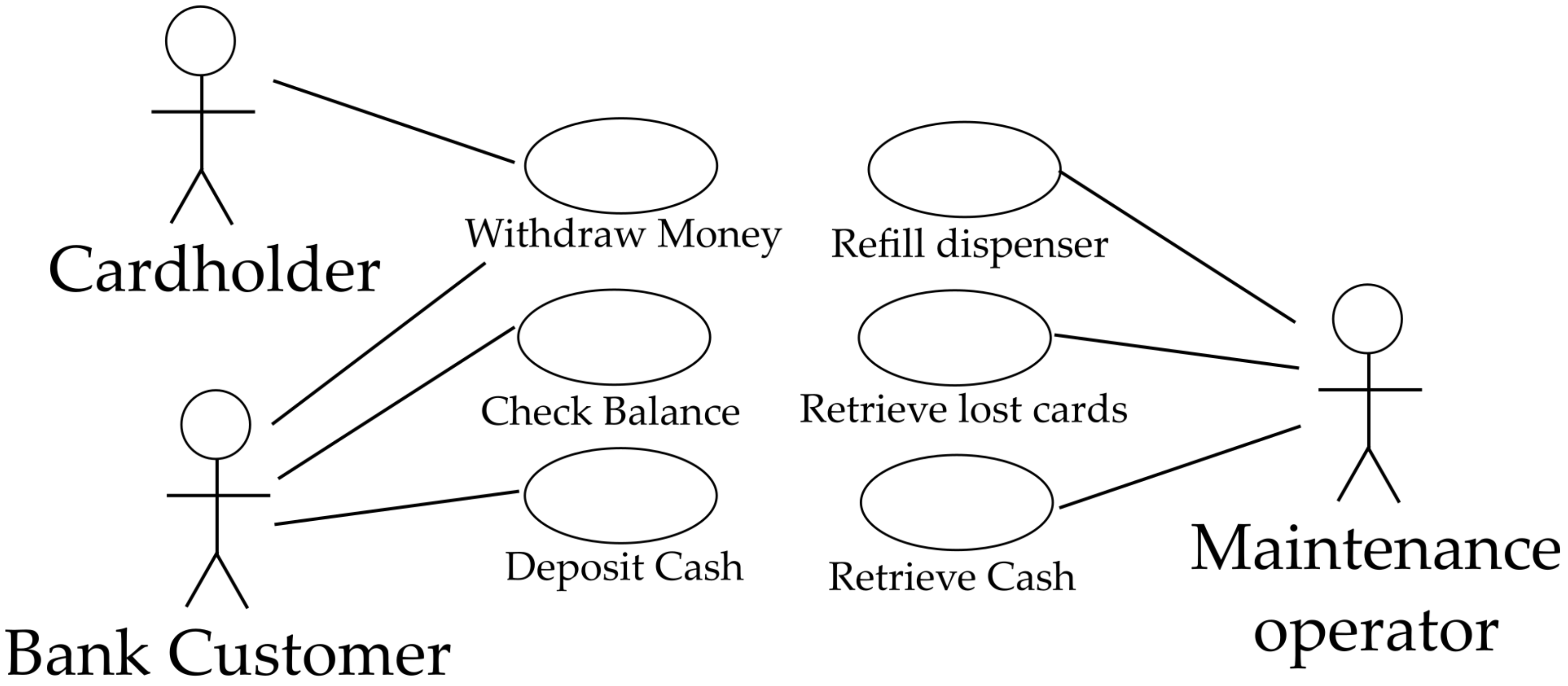
Example (2): Use Cases



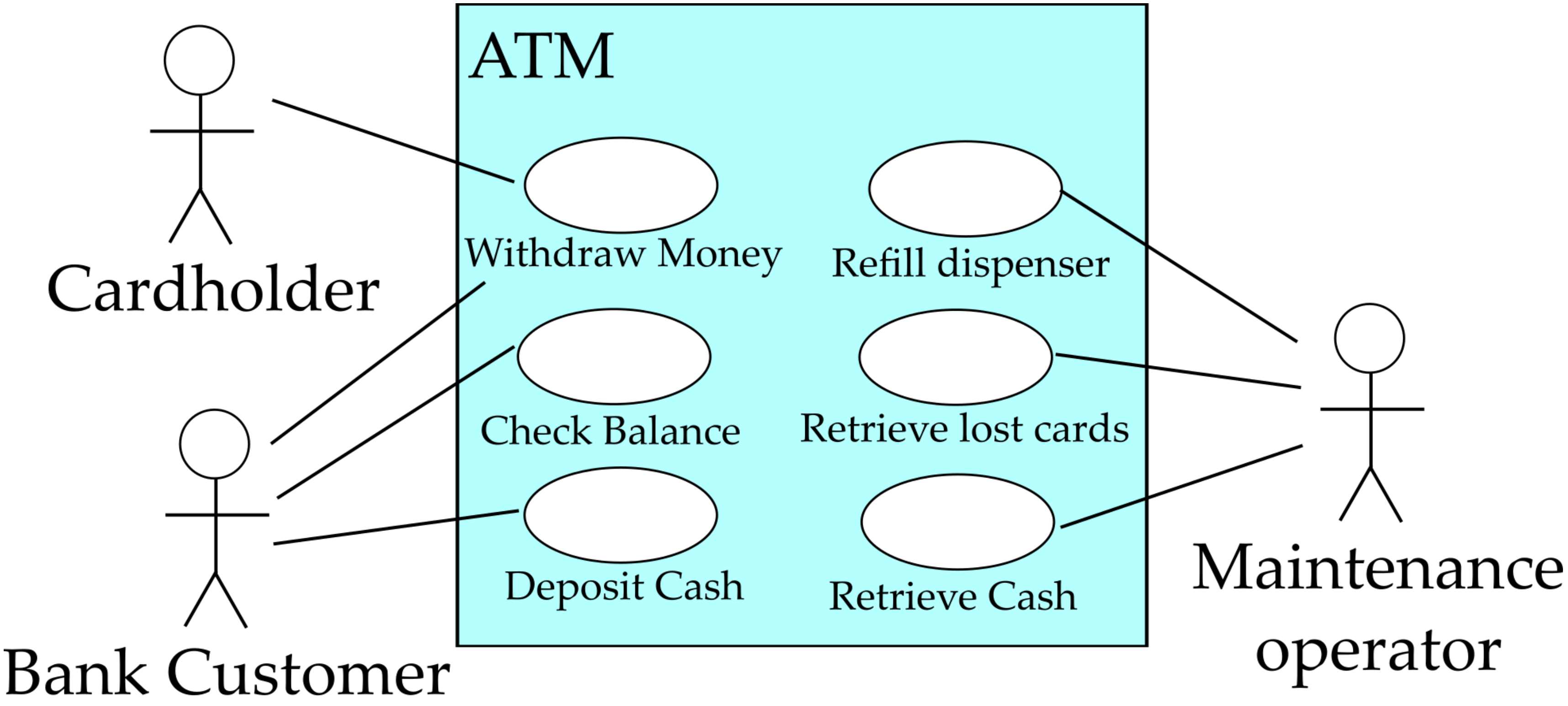
Example (3): Map actors to use cases



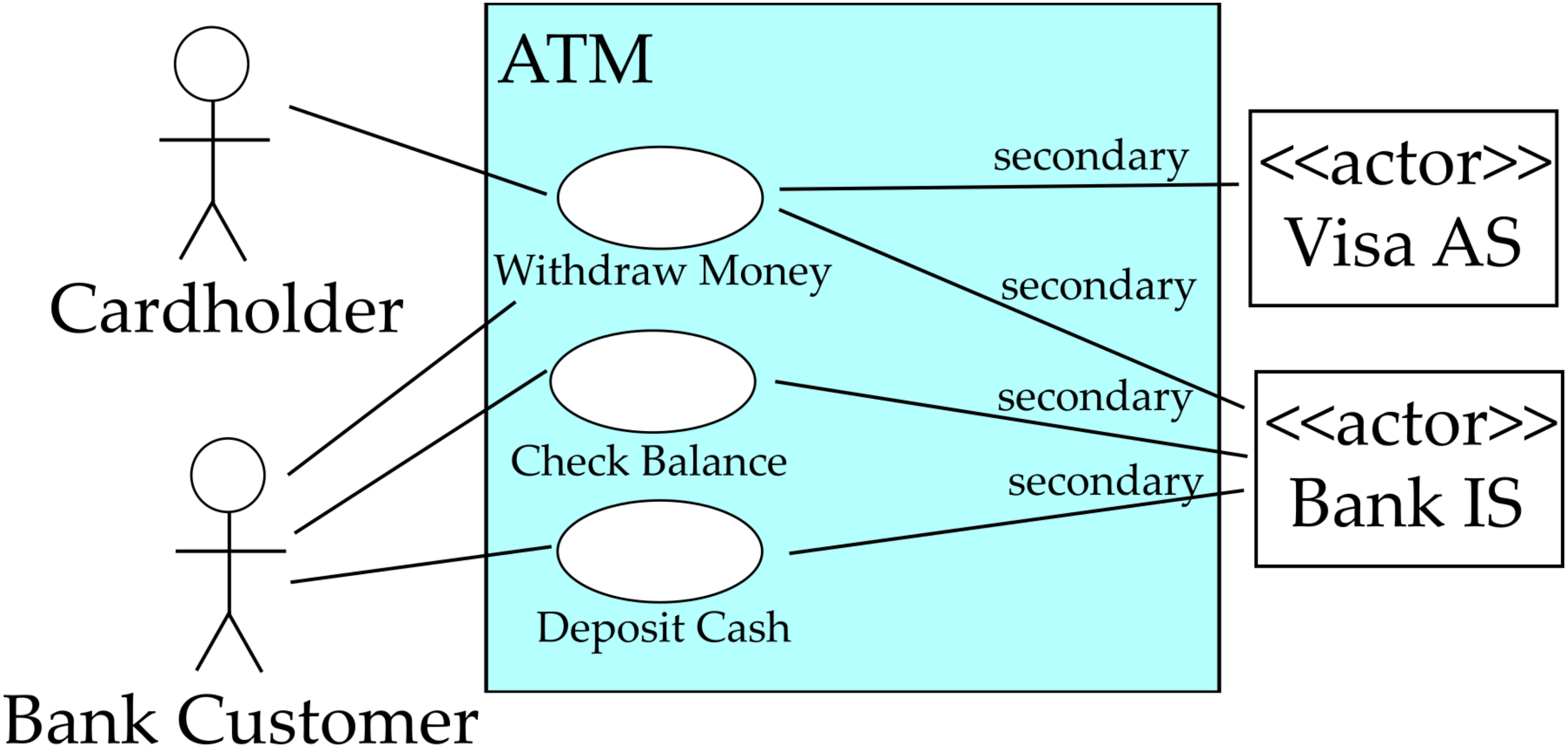
Example (4): Map actors to use cases



Example (5): Add system boundary

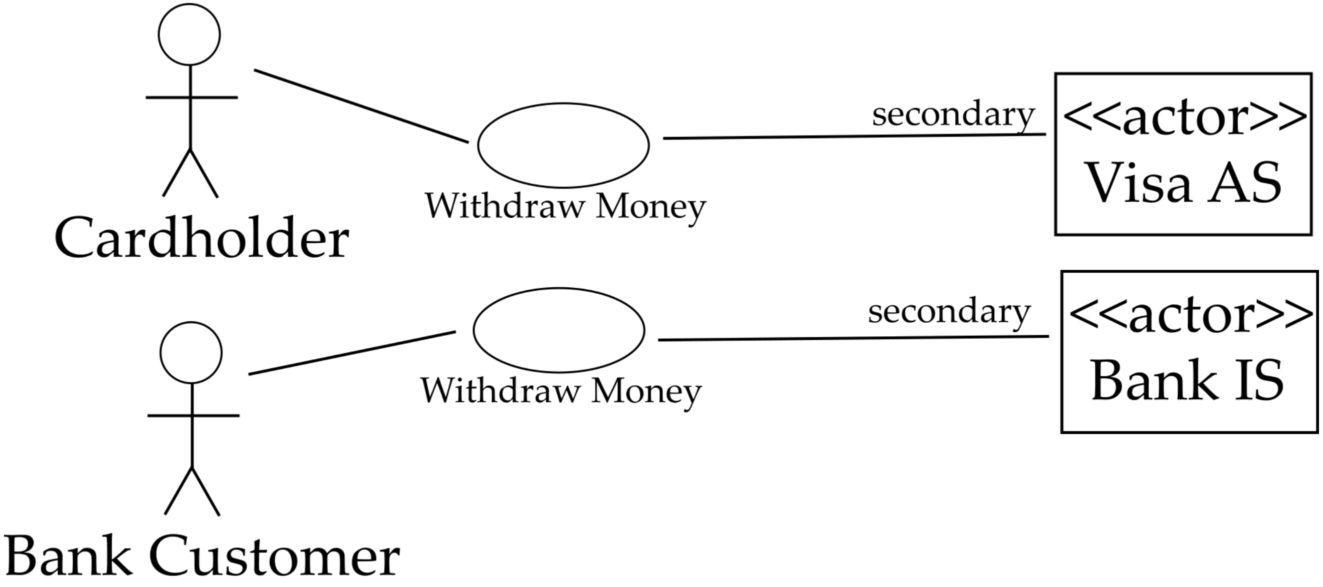


Example (6): Add secondary actors



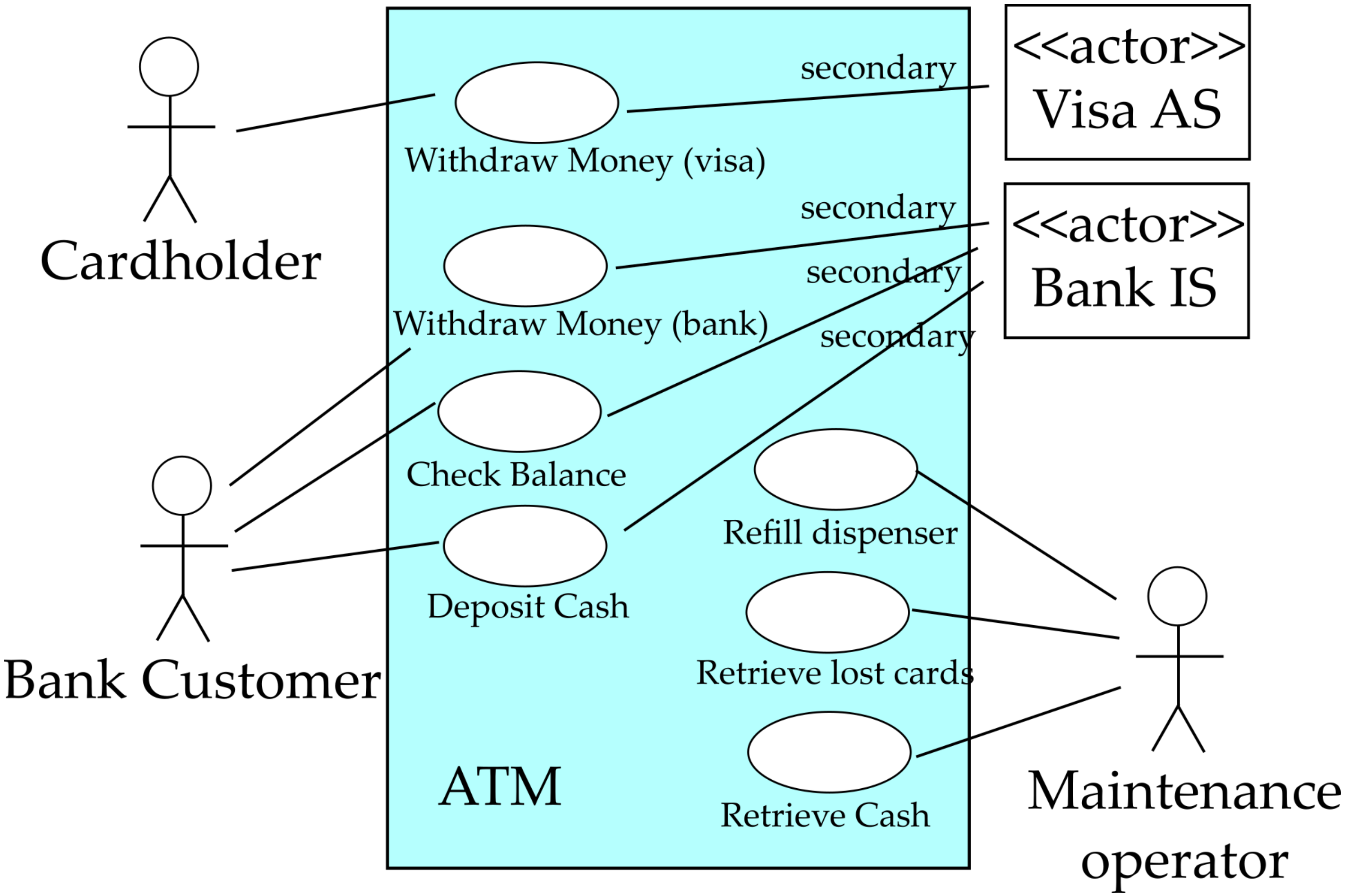
✧ Note: I‘ve removed the Maintenance operator for clarity in this picture.

Example (7): what about the shared use case?

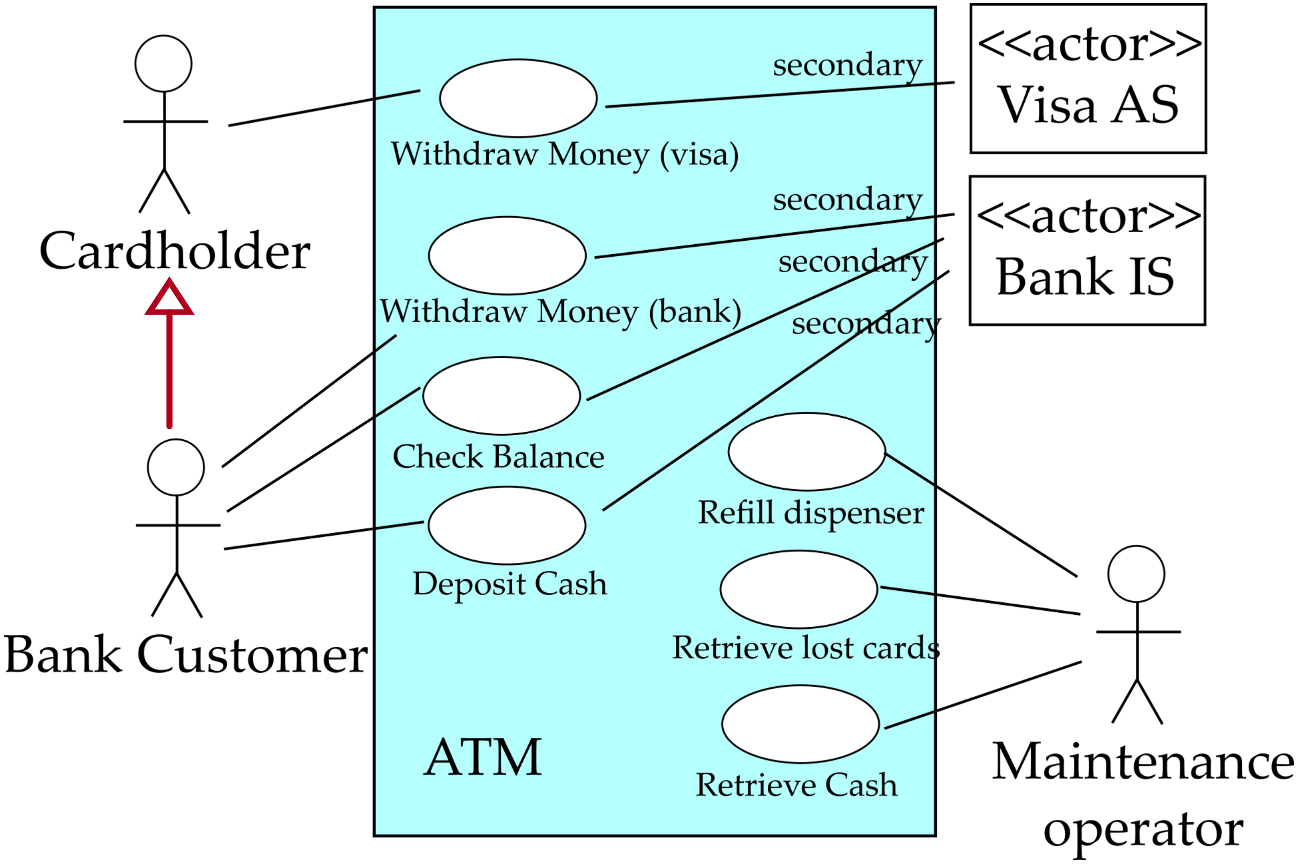


✧ **General guideline:** if a single use case could have two different sets of actors then break it down.

Example (8): Complete Use Case diagram



Inheritance (Generalisation/Specialisation)



✧ **Bank Customer** inherits all use cases from **Cardholder.**

Use case bodies (tabular descriptions)

- We are still missing important information
- We need to explain the **dynamics** of use cases.
- **For each** use case:
  - Construct a list of all interactions with the system
  - Describe interaction in a **table**
  - Each interaction needs a clearly defined beginning, end and sequence of events
- This process produces a **Use Case Body.**

Example (9): Use Case body

For the use case **Withdraw money (Visa)**: withdraw money using Visa card.
Actors: Cardholder, Visa AS.

1. Cardholder inserts card	2. Request PIN from Cardholder
3. Cardholder enters PIN	4. Request authorisation from Visa AS
5. Visa AS confirms	6. Show options
7. Cardholder selects “withdraw”	8. Ask Cardholder for desired amount
9. Cardholder enters amount	10. Requests limit from Visa AS
11. Visa AS reports limit	12. Checks if amount below limit
	13. Returns card
14. Cardholder takes card	15. Issues banknotes
16. Cardholder takes banknotes	
Actor actions	**System responsibilities**

Text vs. graphical descriptions (diagrams)

✧ Text descriptions (use case bodies) are essential

- Ease of communication with users
- Provides agreed upon domain **terminology**

✧ But...

- Difficult to show how sequences follow one another
- Hard to see where secondary actors are needed
- More tiresome to create and read

✧ Hence we use both diagrams and text.

Key points

- A model is an **abstract view** of a system that ignores details.

- Complementary system models can be developed to show the system’s interactions, structure and behaviour.

- Use cases are used for requirements documentation and in early design phases.

- Use sases describe the **interactions** between users external actors.

Short break