# COMP 201 – Software Engineering I
# Lecture 7 – System Models

*Lecturer: T. Carroll*

*Office: G.14*

*Email: Thomas.carroll2@Liverpool.ac.uk*

*See Vital for all notes*

# Coming Up…

# This Week…

- System Models
- Use-Case Diagrams
- Sequence Diagrams
- Data Flow Diagrams
- Statechart Models
- UML
- Finite State Automata
- Moore Machines
- Mealey Machines
- Petri Nets

# Today

# Overview

- Intro to System Models
- Use Case Diagrams
- Use Case Descriptions
- Architectural Model
- Process Models
- Sequence Diagrams

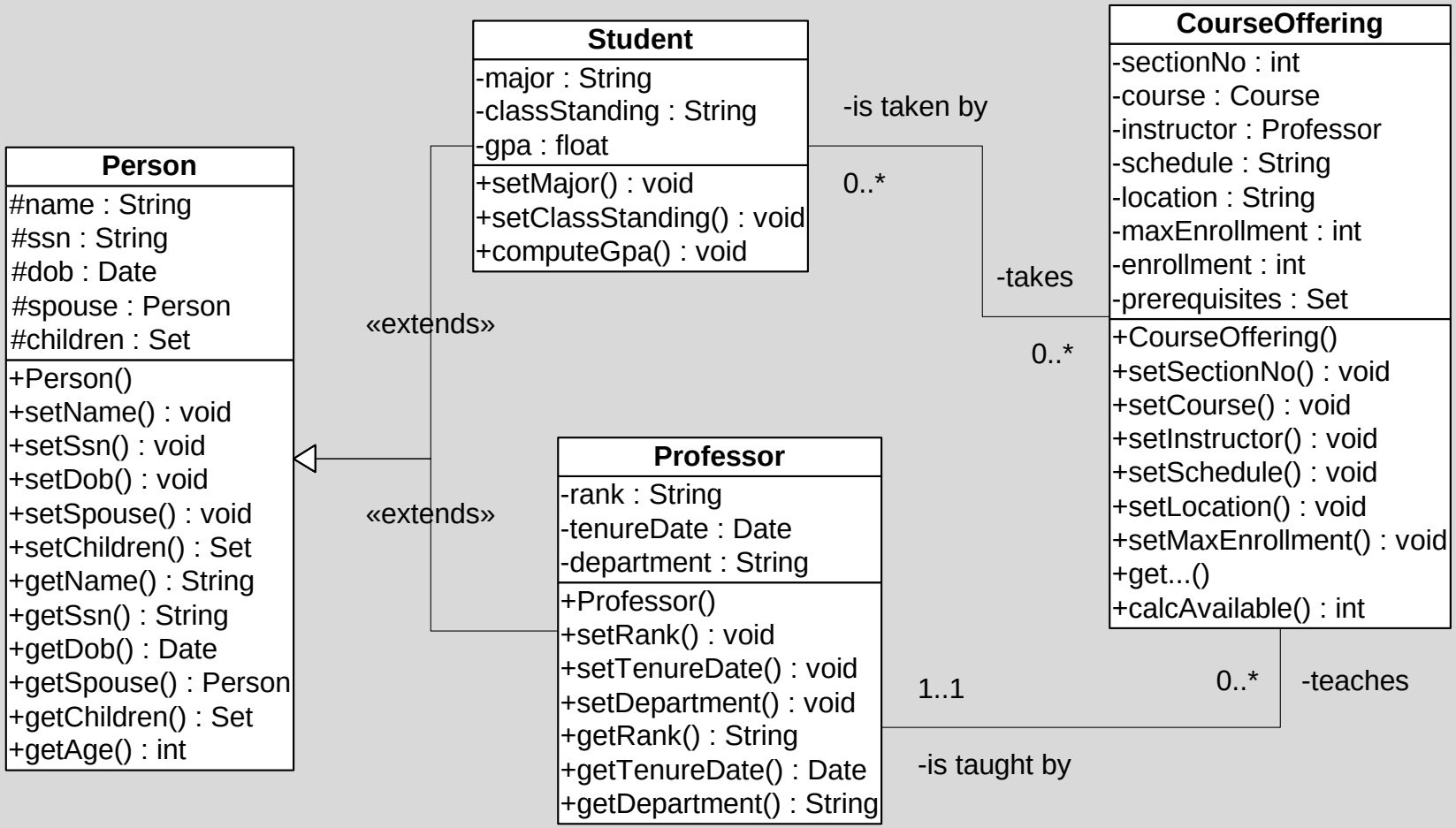
# System Models

- User requirements must be written in such a way that non-technical experts can understand them, e.g., by using natural language
- Detailed system requirements may be expressed in a more technical way
- Document the system specification as a set of system models
  - Graphical representations which describe business processes and the system to be developed
  - An important bridge between the analysis and design processes

# System Modelling

- **System modelling** helps the analyst to understand the functionality of the system
- Models are used to communicate with customers
- Different models present the system from different perspectives:
  - **External perspective**
    - shows the system's context or environment
  - **Behavioural perspective**
    - shows the behaviour of the system
  - **Structural perspective**
    - shows the system or data architecture

# A Picture Paints A Thousand Words



# System Model Advantages

- They can be easier to understand than a verbose natural language description
- System models can leave out unnecessary details of the system -focus on **what is important**
  - A system **representation** should maintain all the information of a system
  - An **abstraction** deliberately simplifies the system and picks out its most salient characteristics
- Different models can focus on different approaches to abstraction

# System Model Weaknesses

- They do not model **non-functional** system requirements
- They do not usually include information about whether a method is appropriate for a given problem
- They may produce too much documentation
- System models are sometimes too detailed and difficult for users to understand

# Model Types

- **Context Models**
  - shows the boundaries of a system
- **Data processing model**
  - Shows how the data is processed at different stages
- **Composition model**
  - shows how entities are composed of other entities
- **Architectural model**
  - shows principal sub-systems
- **Classification model**
  - shows how entities have common characteristics
- **Stimulus/response model**
  - shows the system’s reaction to events

# Use Case Diagrams

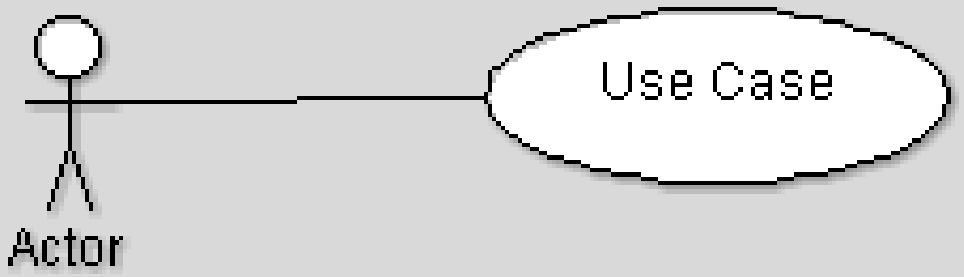
# Use Case Diagrams – Scenarios using UML

- Functional modelling of requirements
- Show the external view of the system (we don't look at internal behaviour)
- Show the system relative to different users of the system
- A set of use-cases should describe all possible interactions with the system.
  - Actors involved
  - Details of the interaction
- **Sequence diagrams** may be used to add detail to use-cases by showing the sequence of event processing in the system

# Actors and Use Cases

- An **actor** is a user of the system  acting in a particular role
  - Something external to the system
  - Can be human (Customer, Cashier, …)
  - Can be non-human (Robotic Arm, Sensor, …)
- A **use-case** is a task which the **actor** needs to perform with the help of the system
  - e.g., find details of a book, print receipt, …
- The details of each use case should also be documented by a use case description
  - **Print receipt** – A customer has paid for an item via a valid payment method. The till should print a receipt indicating the current date and time, the price, the payment type and the member of staff who dealt with the sale.
  - **[Alternate Case]** – No print paper available – Print out “Please enter new till paper” to the cashier’s terminal. Try to print again after 10 seconds.

# Example – Actor and Use Case

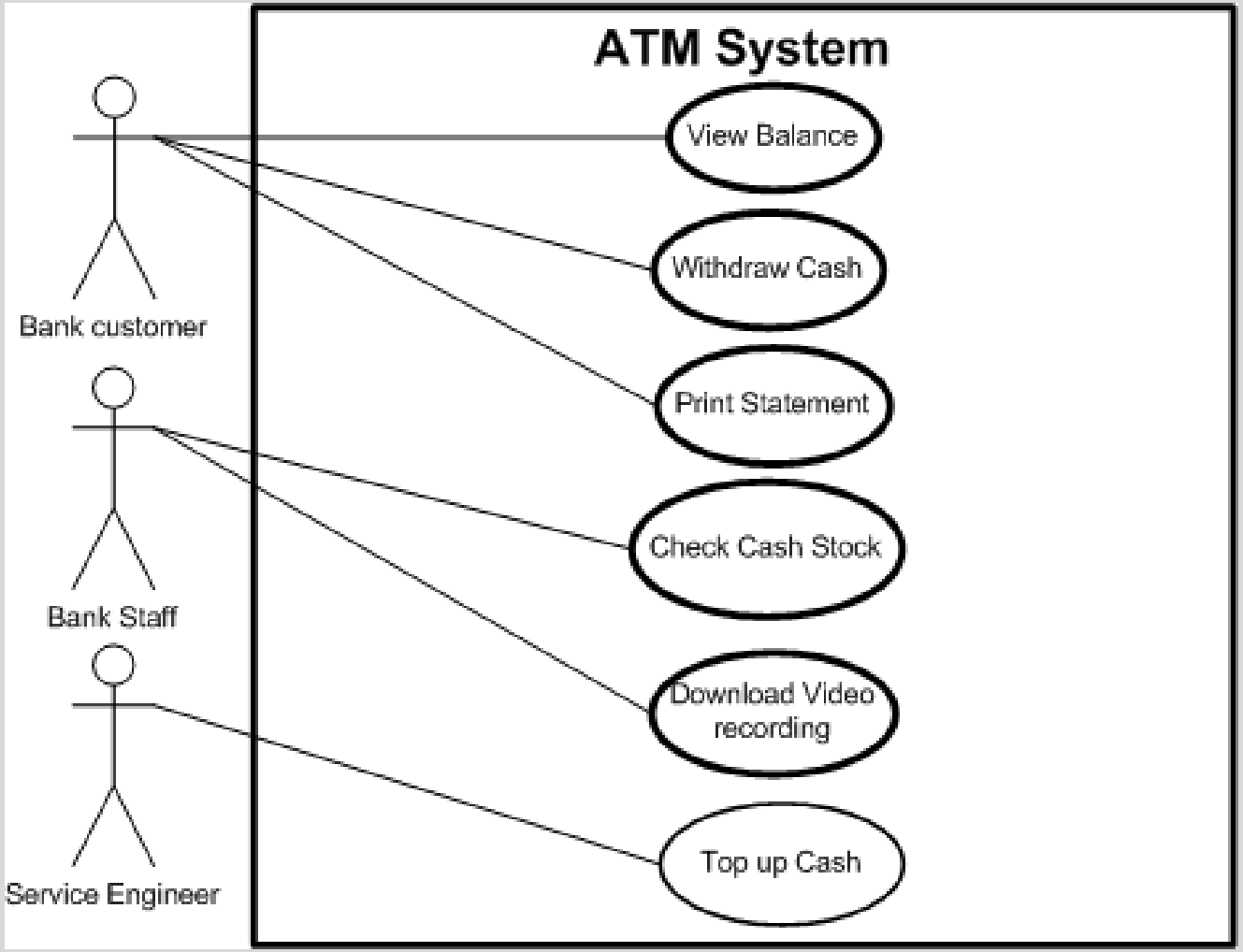


# ATM Actors and Use Cases

- Actors
  - Customers
  - Bank staff
  - ATM service engineer
- Use cases
  - Withdraw cash
  - Check balance
  - Add cash to machine
  - Check security video recording
  - Check Cash Stock

- Can your draw the associated Use Case Diagram?

# ATM Use Case Diagram

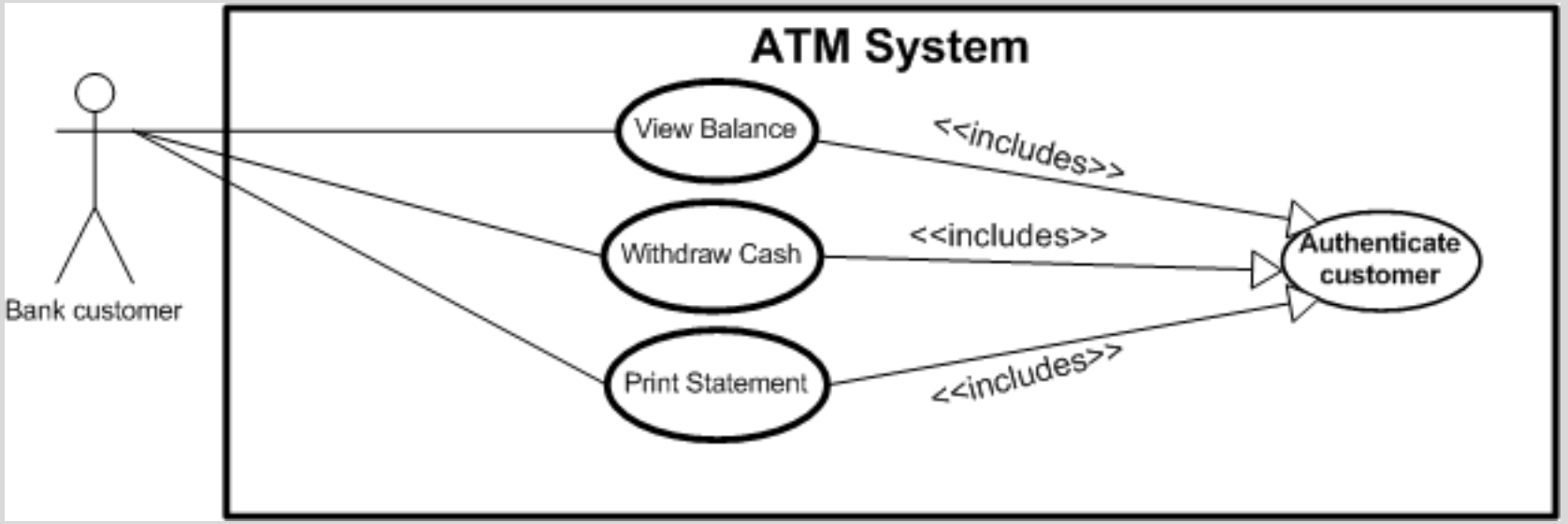


# System Boundary and Inheritance

- **Box** around a set of use cases to denote the system boundary
- **Inheritance** can be used between actors to show that all use cases of one actor are available to the other:

# <<includes>> relation

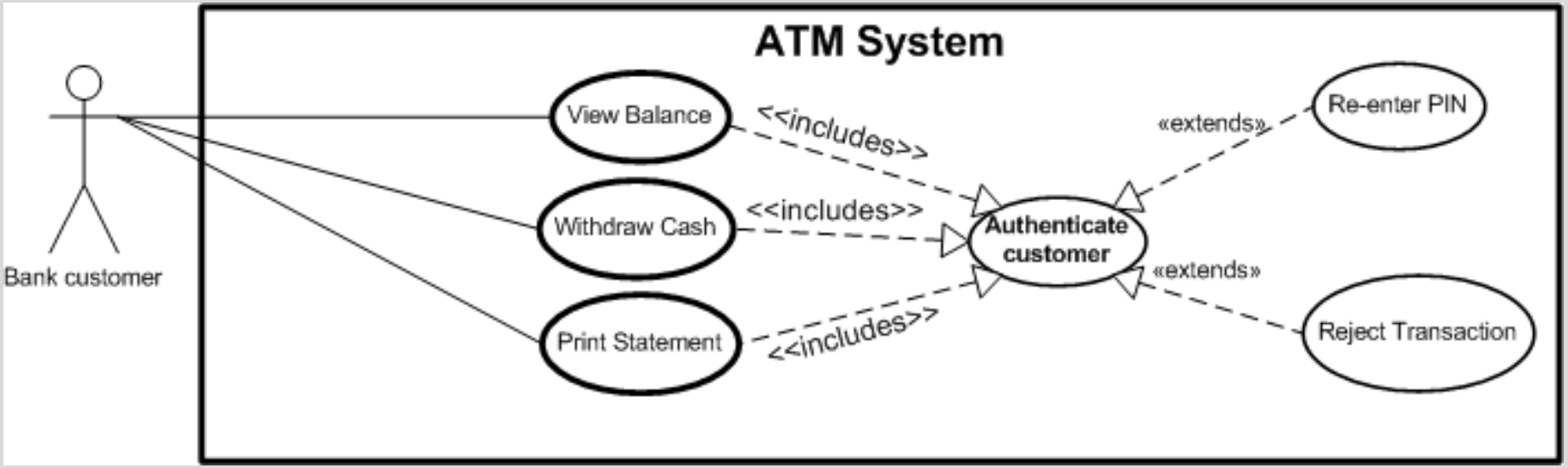
- A use case **always** makes use of another




**Note the direction of the arrows!**

# <<extends>> relation

- A use-case has two or more significantly different outcomes
- **Extends** the use case to a main use case and one or more subsidiary cases.
  - Use cases are **sometimes** needed




**Note the direction of the arrows!**

# Use Case Descriptions

# Use Case Description Template

- ID
  - Short ID  (useful for diagrams and reference)
- Name
  - Full name
- Description
  - Full description
- Pre-condition
  - What must be true before the use case can proceed
- Event flow
  - Flow of behaviour that makes up this use case
- Post-condition
  - What should be true if the use case successfully completes
- Includes
  - What other use cases are used
- Extensions
  - Optional behaviour
- Triggers
  - What makes this use case happen

# ATM use case descriptions

| | |
|---|---|
| **ID** | UC1 |
| **Name** | Withdraw cash |
| **Description** | Bank customer withdraws cash from machine |
| **Pre-condition** | ATM in service<br>ATM Has sufficient cash in stock |
| **Event flow** | 1. Include Use case 2 “Authenticate customer”<br>2. Choose quick cash or enter exact amount<br>3. Include usecase Check balance<br>3. Choose receipt option<br>4. Take cash |
| **Extension points** | Transaction Aborted<br>Insufficient funds |
| **Triggers** | Customer selected the “Withdraw Cash” option |
| **Post-condition** | Balance Updated, Cash dispensed |

# ATM use case descriptions

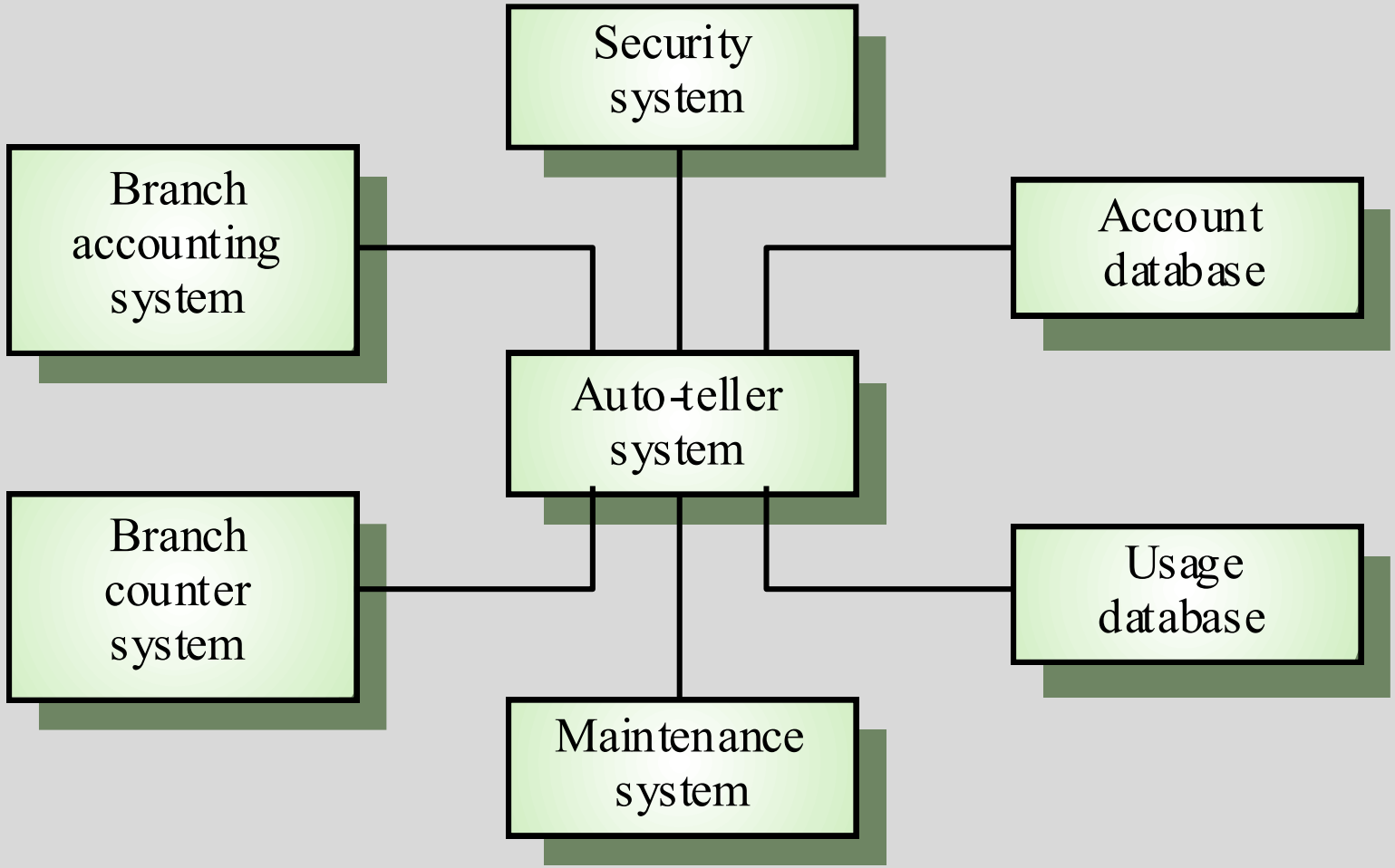
| | |
|---|---|
| **ID** | UC2 |
| **Name** | Authenticate customer |
| **Description** | Bank customer proves their identity to the ATM. |
| **Pre-condition** | ATM in service |
| **Event flow** | 1. If user already authenticated exit from user case.<br>2. User enters card and PIN number<br>3. User re-enters PIN if PIN incorrect |
| **Extension points** | Use case 5 “Card stolen”<br>Use case 6 “PIN entry failure” |
| **Triggers** | Authenticated service requested and user not authenticated |
| **Post-condition** | User is authenticated if credentials correct |

# ATM use case descriptions

| | |
|---|---|
| **ID** | UC3 |
| **Name** | Check balance |
| **Description** | Bank customer retrieves a balance for their account |
| **Pre-condition** | ATM in service |
| **Event flow** | 1. Include Use case 2 “Authenticate customer”<br>2. Choose onscreen or paper balance |
| **Extension points** | |
| **Triggers** | Check balanced requested |
| **Post-condition** | Balance displayed or printed |

# Architectural Model

# Example – Architectural Model of an ATM System

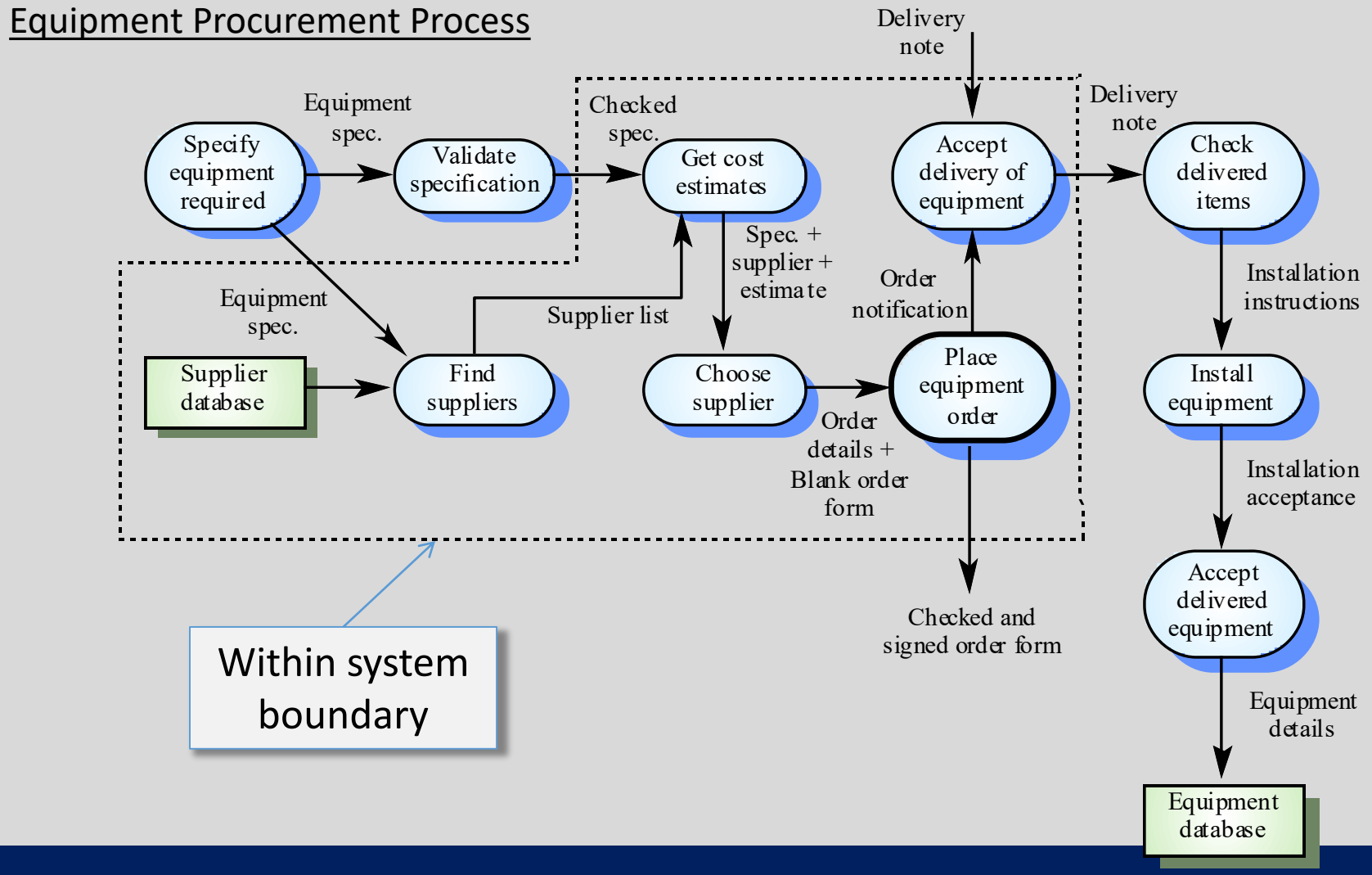


# Process Models

# Process Models

- **Process models** show the overall process and the processes that are supported by the system
  - In process models it is implicit 1 process is completed before another process begins
  - Process models are similar to flow charts
- **Data flow models** may be used to show the processes and the **flow of information** from one process to another
  - The data flow models it is implicit that processes will happen in parallel (concurrent processing)

# Example Process Model



# Sequence Diagram

# Sequence Diagrams

- Show actors and objects that partake in an interaction (like a use case)
- Shows messages passed between objects as time passes (top-to-bottom)
- Shows the lifetime of objects' computation

# Sequence Diagram: Receptionist Creates An Appointment