Use Case Diagrams

Dr. Jason Barron

South East Technological University

September 26, 2022

Overview

Use Case Introduction

- A formal way of representing how a business system interacts with its environment
- Illustrates the activities that are performed by the users of the system
- A scenario-based technique in the UML
- A sequence of actions a system performs that yields a valuable result for a particular actor
- A major process performed by the system that benefits an actor(s) in some way
- Models a dialogue between an actor and the system
- Represents the functionality provided by the system

Use Cases

- Each use case in a use case diagram describes one and only one function in which users interact with the system
  - May contain several paths that a user can take while interacting with the system
  - Each path is referred to as a scenario
- Use Cases describe scenarios that describe the interaction between users of the system (the actor) and the system itself
- Use case diagrams describe what a system does from the standpoint of an external observer
- The emphasis is on what a system does rather than how
- Use case diagrams are closely connected to scenarios
- A scenario is an example of what happens when someone interacts with the system.

Use Cases

- Depiction of a systems behavior or functionality under various conditions as the system responds to requests from users
- Full functioning for a specific business purpose
- Consists of three things:
  - An actor (user) that initiates an event
  - An event that triggers a use case
  - The use case that performs the actions triggered by the event

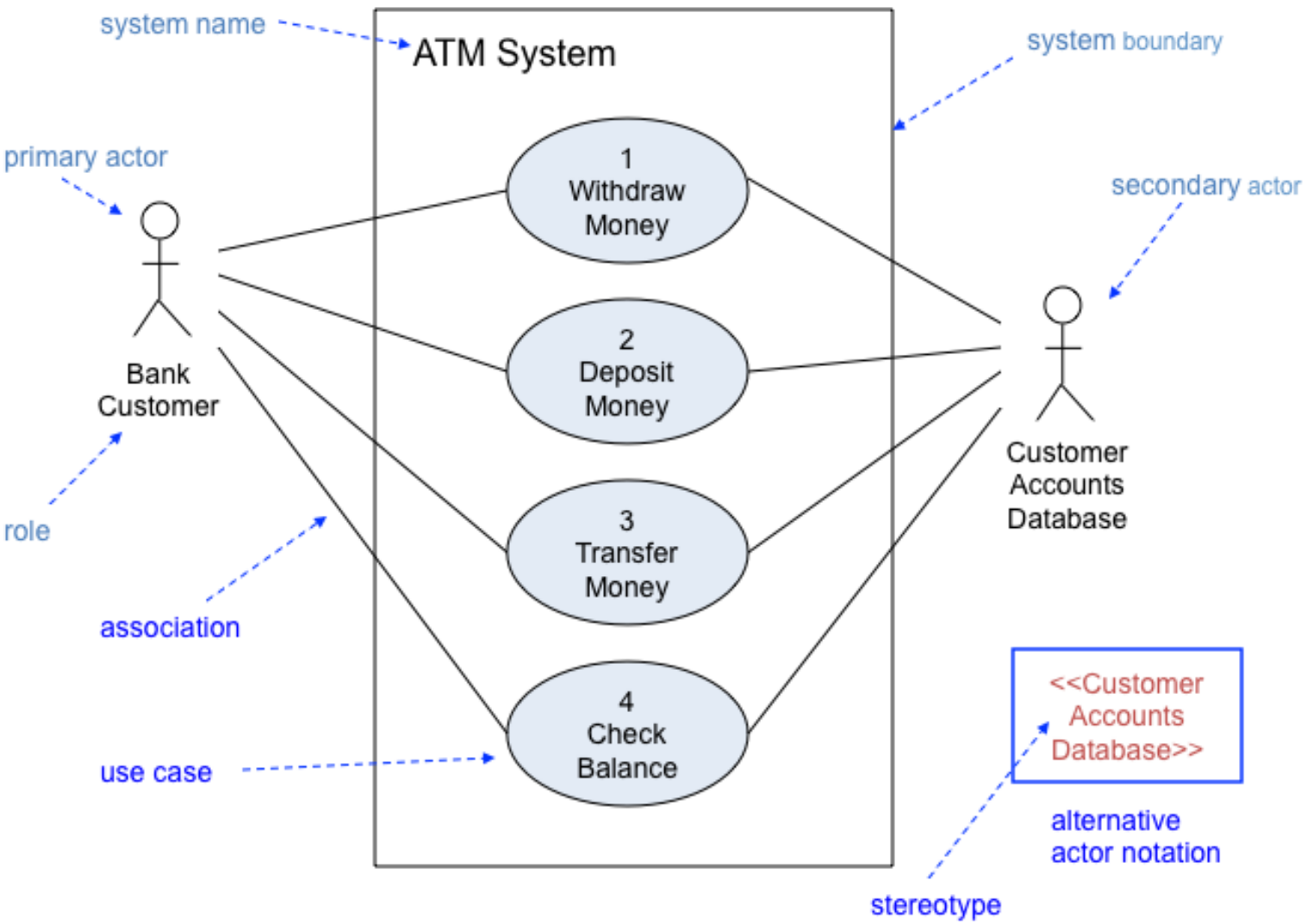
Use Case Example

An ATM system displays the account types available to be withdrawn from and the user indicates the desired type. The system asks for the amount to be withdrawn and the user specifies it. Next, the system debits the users account and dispenses the money. The user removes the money, the system prints a receipt, and the user removes the receipt. Then the system displays a closing message and dispenses the users ATM card. After the user removes his card, the system displays the welcome message.

Use Case Specification Template Example

Number	1
Name	Withdraw Money
Summary	User withdraws money from one of his/her accounts
Priority	5
Preconditions	User has logged into ATM
Postconditions	User has withdrawn money and received a receipt
Primary Actor(s)	Bank Customer
Secondary Actor(s)	Customer Accounts Database

Use Case Diagram



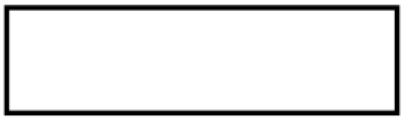
Use Case Diagrams

- A use case describes what the system does, not how it does the work
- Labelled using a descriptive verb-noun phrase and represented by an oval
- The use case model reflects the view of the system of the user outside of the system
- Symbols are:
  - Actor, a stick figure
  - Use case, an oval
  - Connecting lines

Use Case Symbols

Use Case

Actor



Boundary

Connection

<<include>>

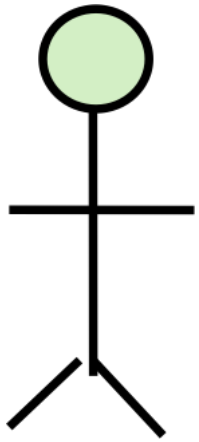
Include relationship

Extend relationship

<<extend>>

Actors

- An Actor is outside or external to the system
  - A user or outside system that interacts with the system being designed in order to obtain some value from that interaction
- It can be a:
  - Human
  - Peripheral device (hardware)
  - External system or subsystem
  - Time or time-based event
- Represented by a stick figure
- Labelled using a descriptive noun or phrase



Actors

- Represent role played by one or more users
- Exist outside of the system
- Can initiate an instance of a use case
- May interact with one or more use cases and a use case may involve one or more actors
- Actors may be divided into two groups:
- Primary actors supply data or receive information from the system
- Secondary actors help to keep the system running or provide help
    - Help desk, analysts, programmers, etc.

Questions for Identifying People Actors

- Who is interested in the scenario/system?
- Where in the organisation will the scenario/system be used?
- Who will benefit from the use of the scenario/system?
- Who will supply the scenario/system with this information, use this information, and remove this information?
- Does one person play several different roles?
- Do several people play the same role?

Questions for Identifying Other Actors

- What other entity is interested in the scenario/system?
- What other entity will supply the scenario/system with this information, use this information, and remove this information?
- Does the system use an external resource?
- Does the system interact with a legacy system?

Boundary

- A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system
- A boundary is the dividing line between the system and its environment
- Use cases are within the boundary
- Actors are outside of the boundary

Connection

- A connection is an association between an actor and a use case
- Depicts a usage relationship
- Connections do not indicate data flow
- Connects an actor to a use case

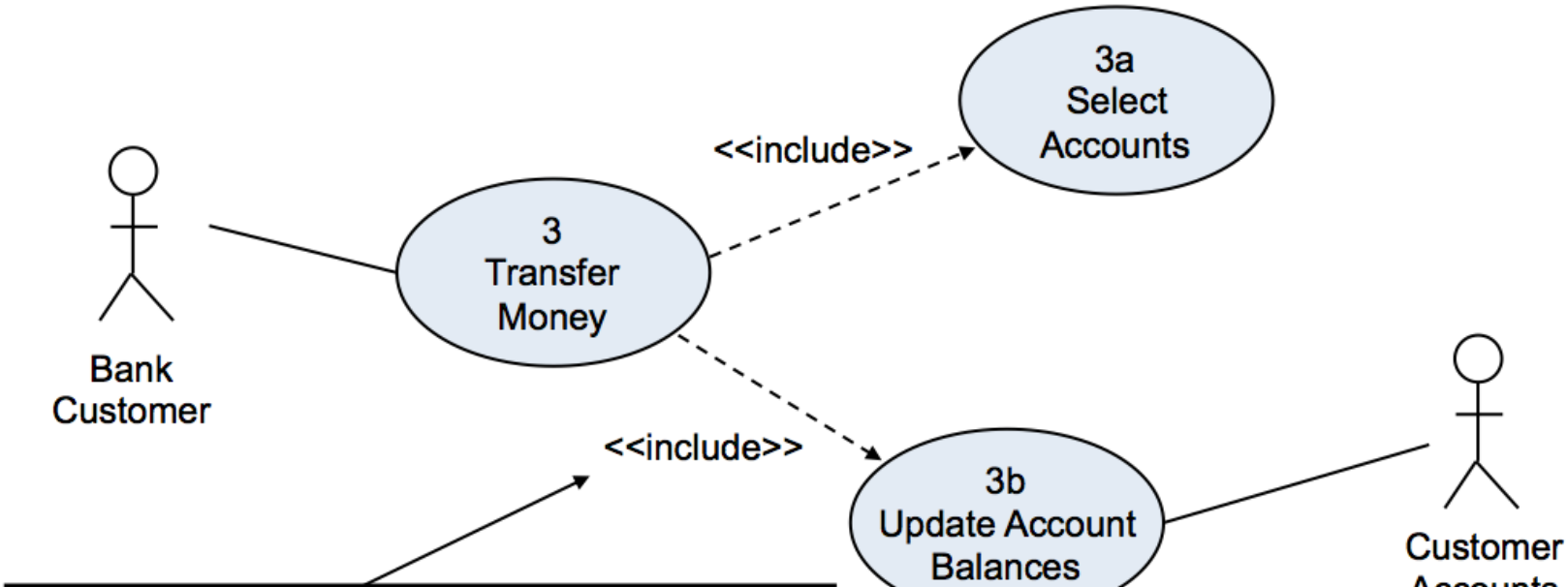
Use Case - Relationships

- Relationships
  - Represent communication between actor and use case
  - Depicted by a line or a double-headed arrow line
  - Also called an association relationship
- Types of Relationships for Use Cases
  - Include
  - Extend
  - Generalisation

≪include≫ Relationship

- A connection between two use cases
- Indicates a use case that is used (invoked) by another use case
- Links to general purpose functions, used by many other use cases
- Includes
  - Use case contains a behaviour that is common to more than one use case
  - The common use case is included in other use cases
  - Dotted arrow points toward common use case
- Represents the inclusion of the functionality of one use case within another
- Arrow is drawn from the base use case to the used use case
- Write ≪include≫ above arrowhead line

≪include≫ Relationship



This is an **include dependency**.

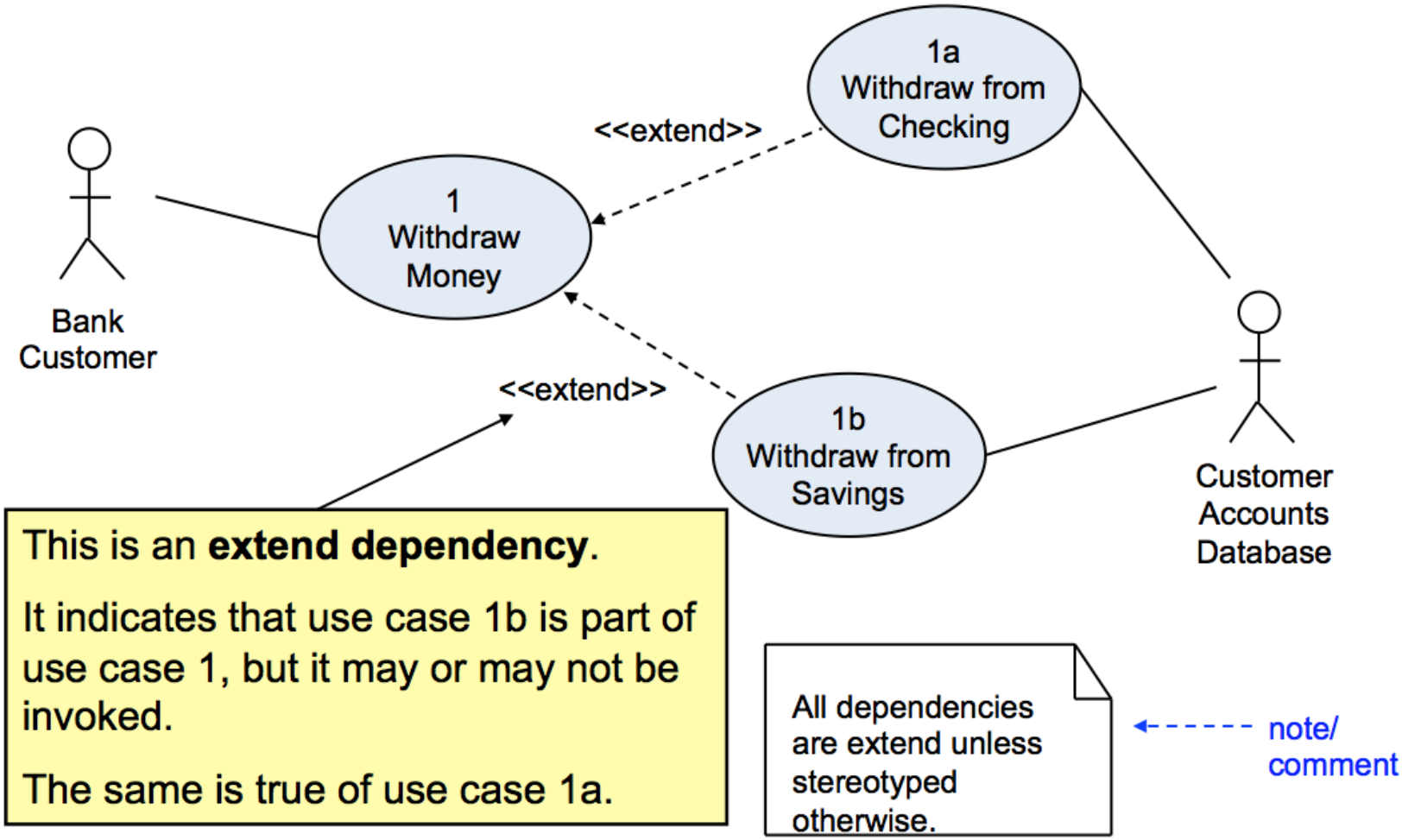
It indicates that use case 3b is "included" in use case 3 and will be invoked.

The same is true of use case 3a.

≪extend≫ Relationship

- A connection between two use cases
- Extends a use case by adding new behaviour or actions
- Specialised use case extends the general use case
- Extends
  - A different use case handles variations or exceptions from the basic use case
- Represents the extension of the use case to include optional functionality
- Arrow is drawn from the extension use case to the base use case
- Write ≪extend≫ above arrowhead line

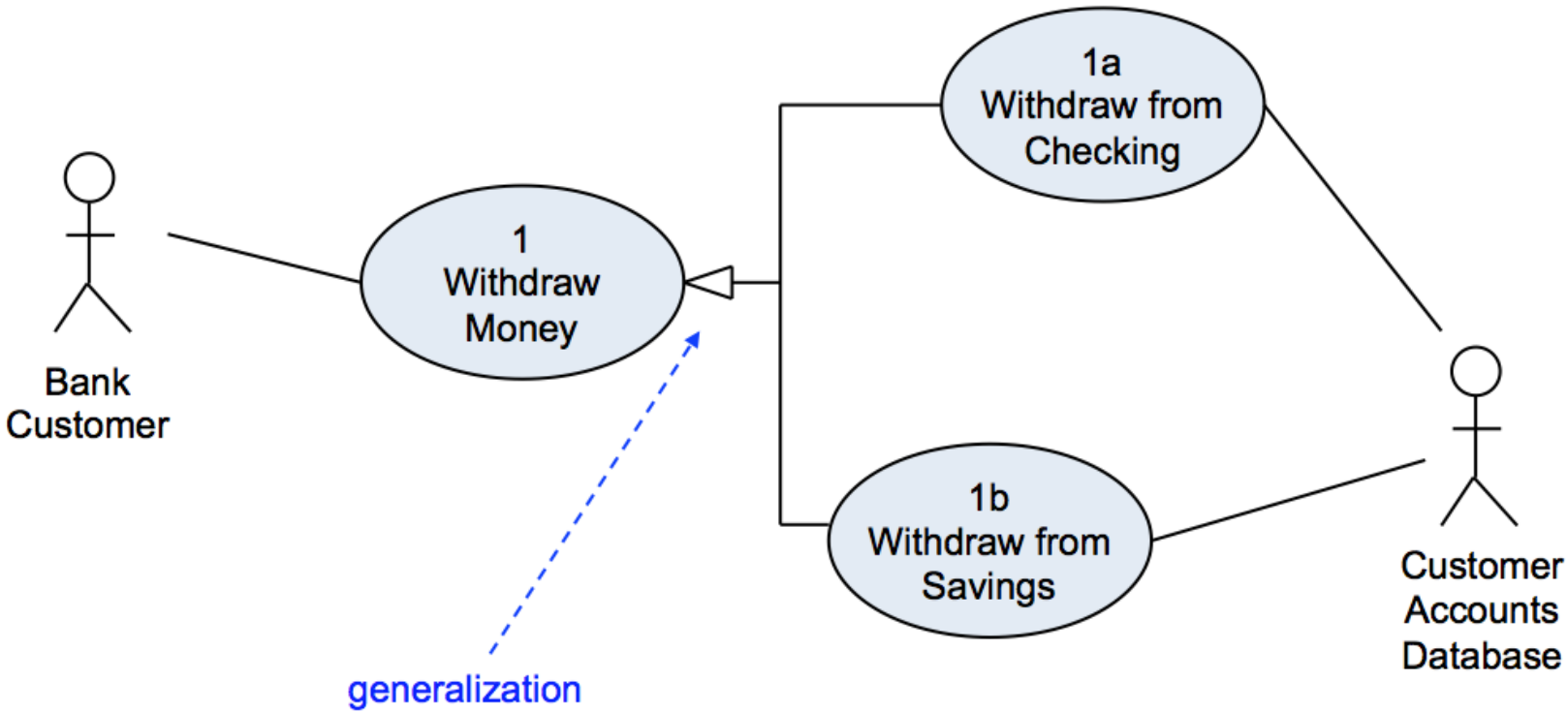
≪extend≫ Relationship



Generalisation Relationship

- Generalises
  - One thing is more general than another thing
  - Arrow points to the general thing
- Generalisation Relationship
  - Represented by a line and a hollow arrow
    - E.g. From child to parent

Generalisation Relationship



Use Case Relationships

- Pros
  - Reduces redundancy in use cases
  - Reduces complexity within a use case
- Cons
  - May introduce complexity to use case diagram

Creating a Use Case Model

- Review the business specifications and identify the actors within the problem domain
- Identify the high-level events and develop the primary use cases that describe the events and how actors initiate them
- Review each primary use case to determine possible variations of flow through the use case
- Develop the use case documents for all primary use cases and all important use case scenarios

How to create use case diagram

1. List main system functions (use cases) in a column
   - Think of business events demanding system's response
   - Users' goals/needs to be accomplished via the system
   - Create, Read, Update, Delete (CRUD) data tasks
   - Naming use cases - users' needs usually can be translated into data tasks

How to create use case diagram

2. Draw ovals around the function labels
3. Draw a system boundary
4. Draw actors and connect them with the use cases
5. Specify include and extend relationships between use cases

Advantages of Use Cases

- Use cases are used for capturing users' requirements
- Use cases are described using the language of the users
- Use cases provide an easily-understood communication mechanism
- When requirements are traced, they make it difficult for requirements to be overlooked
- Use cases provide a concise summary of what the system should do at an abstract level

Disadvantages of Use Cases

- With functional decomposition, it is often difficult to transition from functional description to object description to class design
- Reuse at the class level can be hindered by each developer "taking a Use Case and running with it". Since UCs do not talk about classes, developers often wind up in a vacuum during object analysis, and can often wind up doing things their own way, making reuse difficult
- Use Cases make stating non-functional requirements difficult (where do you say that X must execute at Y/sec?)
- Testing functionality is straightforward, but unit testing the particular implementations and non-functional requirements is not obvious

Use Case Tips

- Better to create fewer use cases
- 20 use cases for a large system
- 50 use cases would be the maximum for a large system
- Can nest use cases, if needed