

# CS204362 – Object-Oriented Design

## L8: Developing Use Case Diagrams and Models

Kamonphop Srisopha



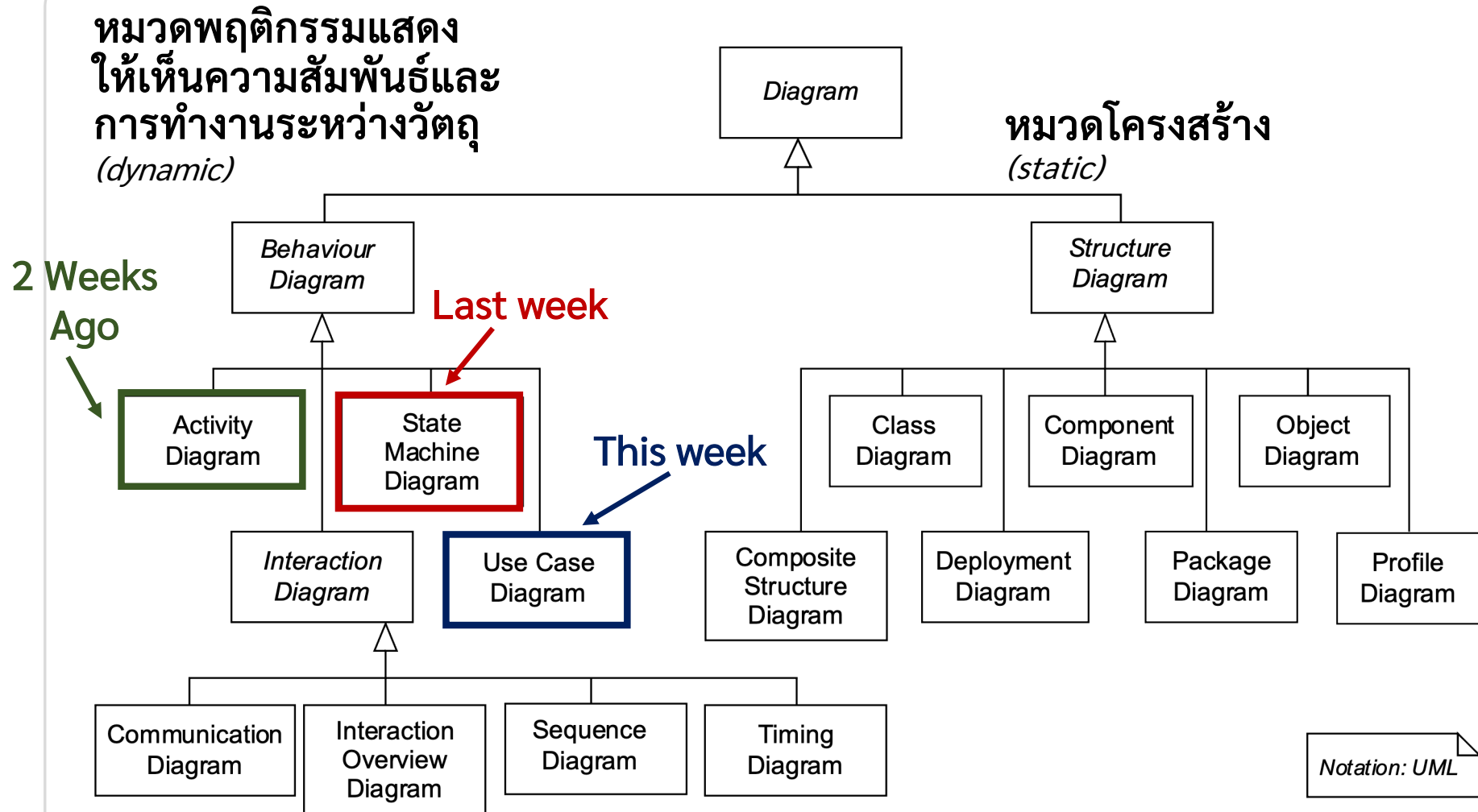
Faculty of Science, Chiang Mai University

คณะวิทยาศาสตร์ มหาวิทยาลัยเชียงใหม่

# Agenda

- Type of Requirements
- Use Cases
- Use Case Specification
- Prototyping

# UML Diagrams



# Types of Requirements

# Types of Requirements

## Functional Requirements

### “What”

- บ่งบอกความต้องการถึงสิ่งที่ระบบควรจะต้องทำ
- รายละเอียดเกี่ยวกับ input ที่ระบบสามารถรับได้ หรือ output ที่ระบบต้องสร้างออกมา
- รายละเอียดเกี่ยวกับกระบวนการต่างๆของระบบ

## Non Functional Requirements

### “How well”

- บ่งบอกความต้องการว่าระบบควรทำอะไรต่างๆได้ดีแค่ไหน
- อธิบายระดับคุณภาพและประสิทธิภาพของระบบ
- รวมถึงข้อจำกัดต่างๆที่ต้องมีในระหว่างการพัฒนาาระบบ

# Check Your Understanding

Q: นักศึกษาลองคิดว่าข้อไหนต่อไปนี้ควรจัดอยู่ในความต้องการที่เป็น Functional, Non-Functional, หรือ ไม่ควรเป็นความต้องการ

- A) If the alarm system is ringing, then the elevator doors will proceed to the ground floor, open doors, and suspend further operations.
- B) The student information system will provide output from all commands within one second.
- C) A merge-sort algorithm should be used to sort the flights by departure time.
- D) The system must run under Linux and Windows.
- E) The system should be able to expanded in the second release to handle new file formats for graphics.
- F) The system must send a confirmation email whenever an order is placed.

# Use Cases

# Use Cases

- บ่งบอกถึง ฟังก์ชันการทำงานของระบบในมุมมองของผู้ใช้ หรือ บอกว่าผู้จะใช้งานกับระบบอย่างไร
- มีลำดับการดำเนินการของกิจกรรม (sequence of actions) ที่ผู้ใช้อย่างหนึ่งต้องทำให้บรรลุเป้าหมายอย่างหนึ่ง
- ฉะนั้นเป้าหมายของการทำ use case analysis คือการโมเดลระบบ
  - ในมุมมองว่าผู้ใช้อยู่มีปฏิสัมพันธ์กับระบบอย่างไร เพื่อให้บรรลุเป้าหมาย
- Use case models ประกอบด้วย
  1. A set of use cases
  2. A use case specification (คำอธิบาย use case ในข้อ 1 โดยละเอียด)

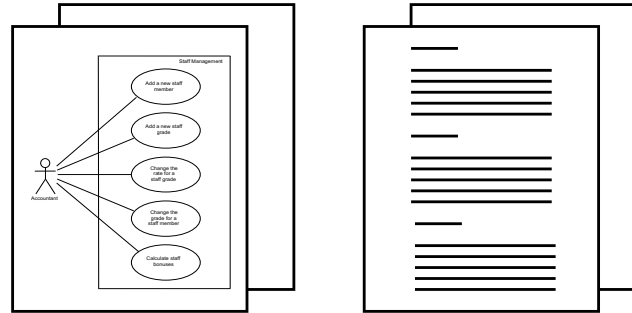


# Development of the Use Case Model Through Successive Iterations

## Iteration 1

### Obvious use cases.

Simple use case descriptions.

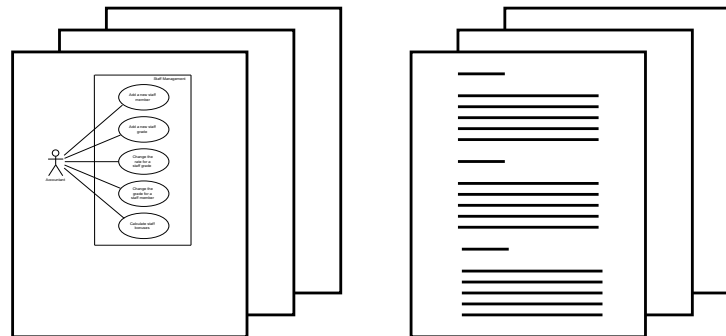


## Iteration 2

### Additional use cases.

Simple use case descriptions.

Prototypes.

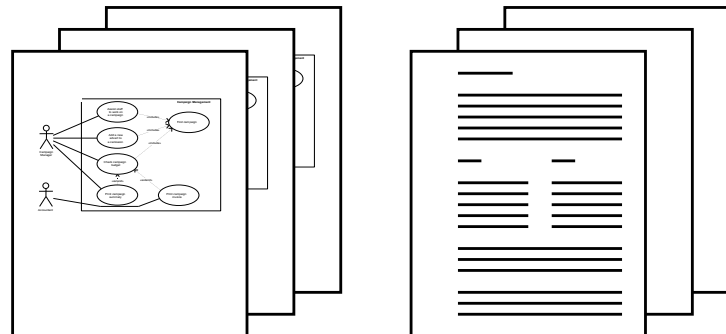


## Iteration 3

### Structured use cases.

Structured use case descriptions.

Prototypes.



## Iteration n

# UML Use Case Diagram

เพื่อที่จะ list use case ต่างๆของระบบเราต้องตอบคำถามที่ว่า

What are your users  
**goals/objectives?**

What are the **key activities** that  
make this business work?

# Check Your Understanding

ให้นักศึกษาลองนึกว่าระบบควรมี use case อะไรบ้างจากความต้องการของระบบต่อไปนี้

ระบบการจองที่ตั้งแคมป์สำหรับแคมป์ในเครือหลายแห่ง  
ที่ตั้งแคมป์แต่ละแห่งมี ผู้จัดการหนึ่งคนและเจ้าหน้าที่ดูแล  
หลายนาย ผู้จัดการและเจ้าหน้าที่สามารถลงข้อมูลเกี่ยวกับ  
แคมป์ได้ เช่น ชื่อของแคมป์ ตำแหน่ง แคมป์ไหนเปิดให้จองได้  
แคมป์ไหนปิดปรับปรุงเพื่อพัฒนาพื้นที่ ผู้พักแรมสามารถใช้  
ระบบเพื่อเลือกและจองแคมป์ได้

# Check Your Understanding

- **Actor: Campground Manager**
  - **Use Case:**
    - Register campground
    - Update information
- **Actor: Camper**
  - **Use Case:**
    - View campground information
    - Make reservation

# UML Notations

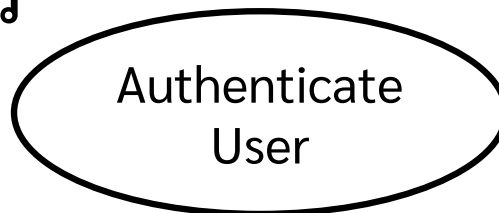
## Use case

เป้าหมายหรือจุดประสงค์ที่ต้องการให้บรรลุ หรือ ลำดับกระบวนการดำเนินการเพื่อให้บรรลุเป้าหมาย

โดยส่วนใหญ่ จะเริ่มต้นด้วย**คำกริยาแล้วตามด้วยคำนาม** เช่น

- Process Sale
- Add customer

ใช้สัญลักษณ์รูปวงรี

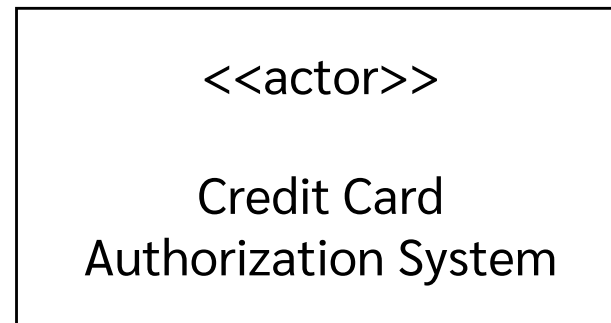


# Notations

## Actor

ผู้ที่มีปฏิสัมพันธ์กับระบบและกับ use case หนึ่งๆที่เป็นมนุษย์ จะใช้สัญลักษณ์รูปคน (stick figure) และที่ไม่ใช่มนุษย์ (เช่น ระบบที่อยู่ **นอกเหนือจากระบบ**ที่เราสนใจ) จะใช้สัญลักษณ์รูปสี่เหลี่ยมแล้วเขียนคำว่า “<<actor>>” มีชื่อของ actor กำกับด้วย

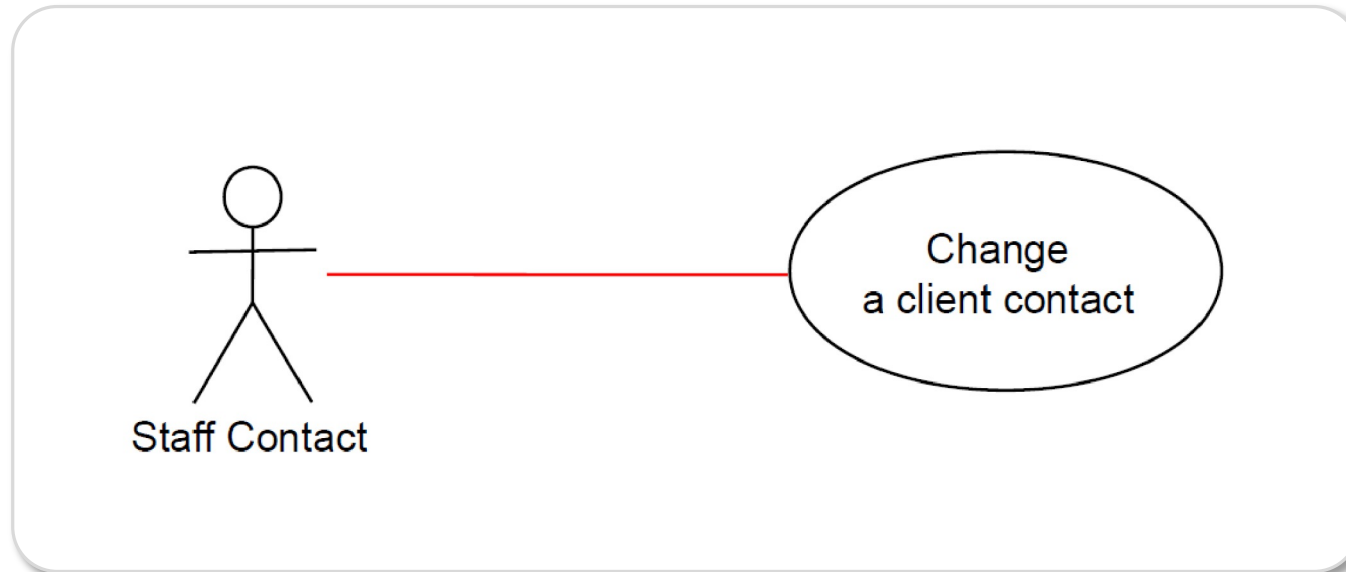
ไม่ควรเลือกชื่อของ actor ตามตำแหน่งงานเช่น ถ้า พนักงานตำแหน่ง 1, 2, และ 3 สามารถใช้งานกับระบบได้เหมือนกันก็ให้เลือกชื่อ actor ที่รวมทั้ง 3 เข้าด้วยกัน ไม่ใช่มี 3 actors.



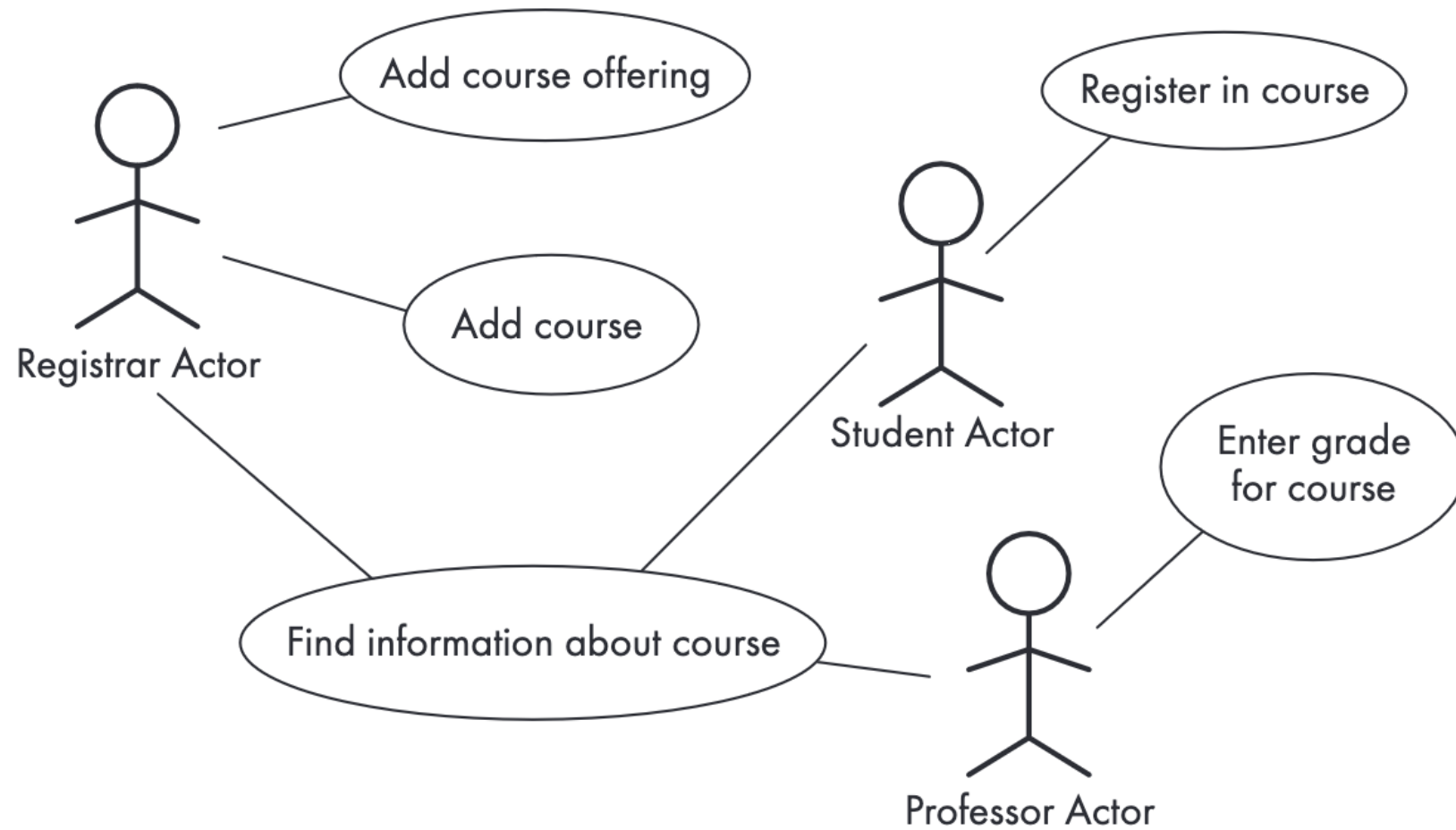
# Notations

## Association (Connector)

เส้นแสดงความสัมพันธ์ระหว่าง actor และ use case  
เป็นเส้นตรงไม่มีหัวลูกศร



# Example

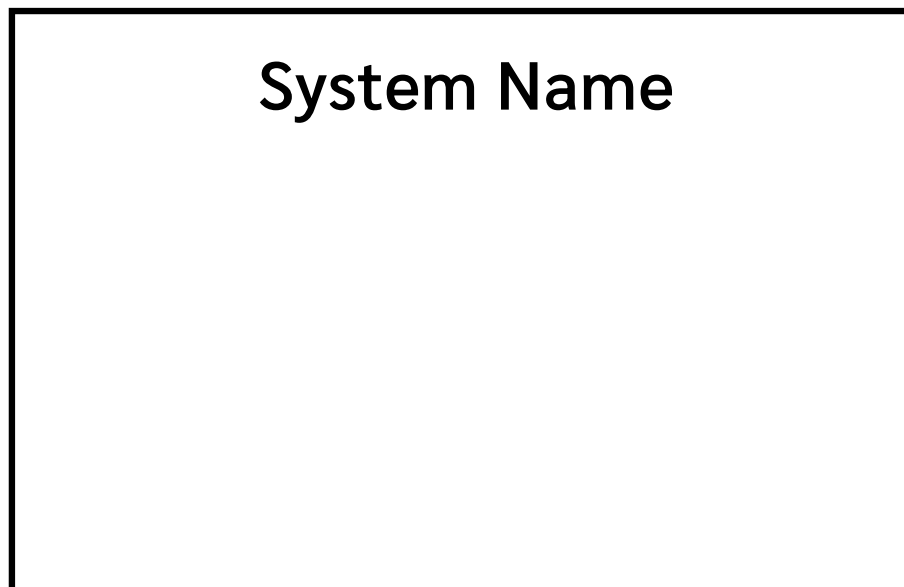




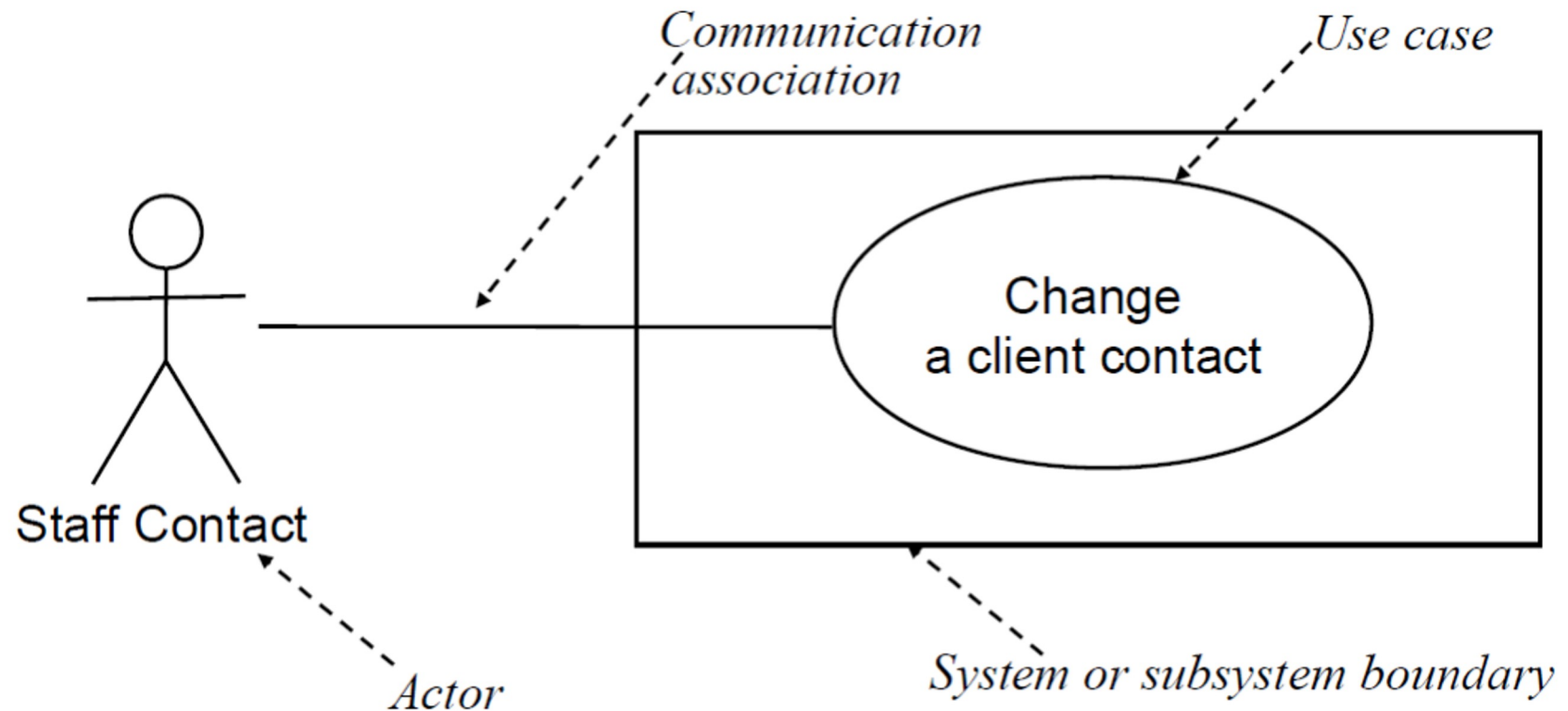
# Notations

## System Boundary

เส้นแบ่งขอบเขตของระบบ ใช้แสดงให้เห็นขอบเขต  
ระหว่างระบบกับผู้กระทำต่อระบบ (ระหว่าง actor กับ use case)  
ใช้สัญลักษณ์รูปสี่เหลี่ยม พร้อมเขียนชื่อระบบไว้ด้านใน



# Example: A Simple Use Case Diagram



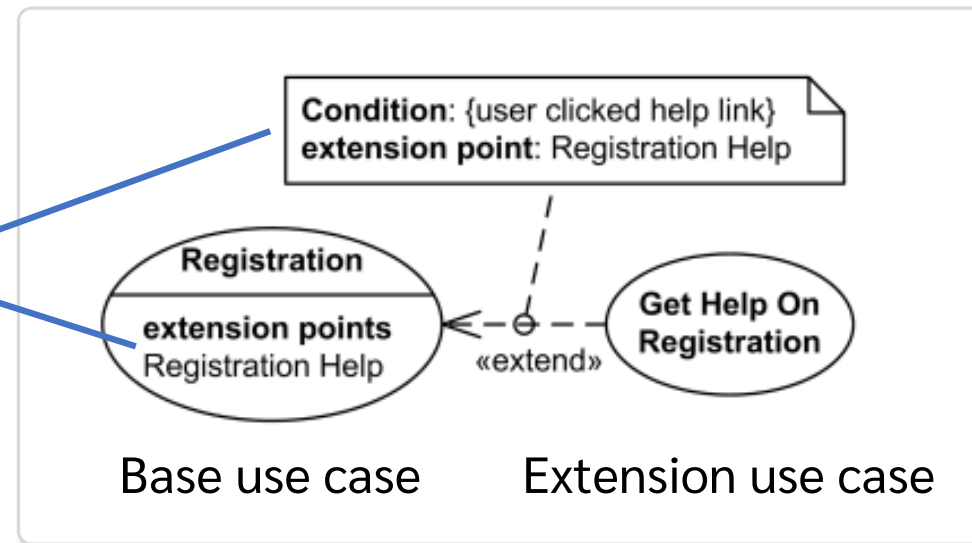
# Notations

## Extend Relationship (*different scenarios*)

ความสัมพันธ์แบบขยายหรือเพิ่มระหว่าง use cases เช่น กรณีที่ use case หนึ่งๆ ต้องทำบางอย่างที่ต่างออกไปจากสิ่งที่ผิปรกติหรือเพื่อตอบสนองต่อเงื่อนไขบางอย่าง เราจะสามารถเขียน เงื่อนไขหรือสิ่งที่ทำต่างเป็นอีก use case หนึ่งได้ โดยให้มันมีความสัมพันธ์ที่เรียกว่า extend relationship กับ use case หลัก

เรียก use case หลักว่า “base use case” และเรียก use case ที่เป็นส่วนเพิ่มว่า “extension use case”

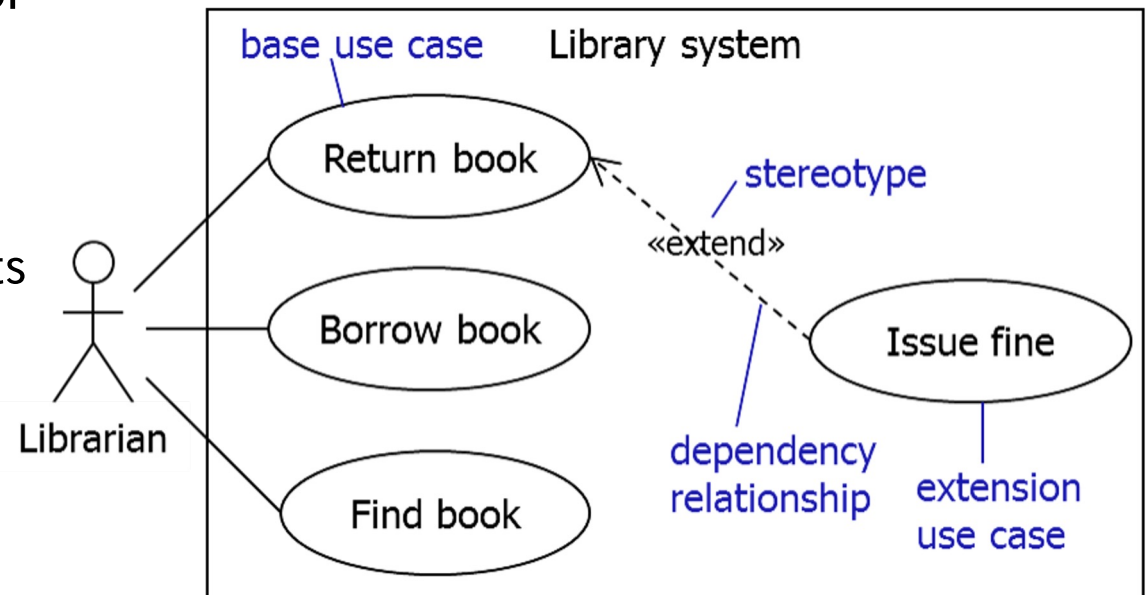
Extension points  
แสดงให้เห็นว่า  
extension use  
case จะถูกใช้  
งานในกรณีไหน



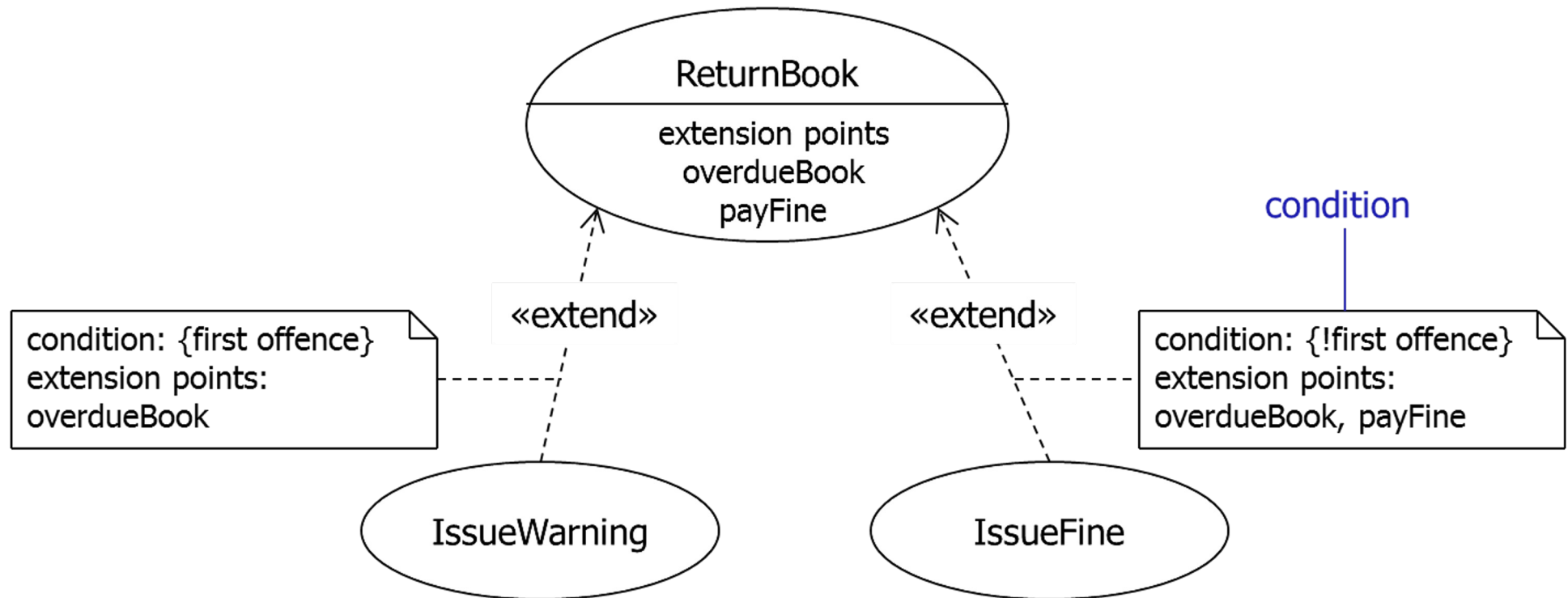
ลูกศรจะชี้ไป  
ที่ base case

# Notations: <<extend>>

- «extend» is a way of adding new behavior into the base use case by inserting behavior from one or more extension use cases
  - The base use case specifies one or more extension points in its flow of events
- The extension use case may contain several insertion segments
- The «extend» relationship may specify *which* of the base use case extension points it is extending



# Notations: Conditional Extensions



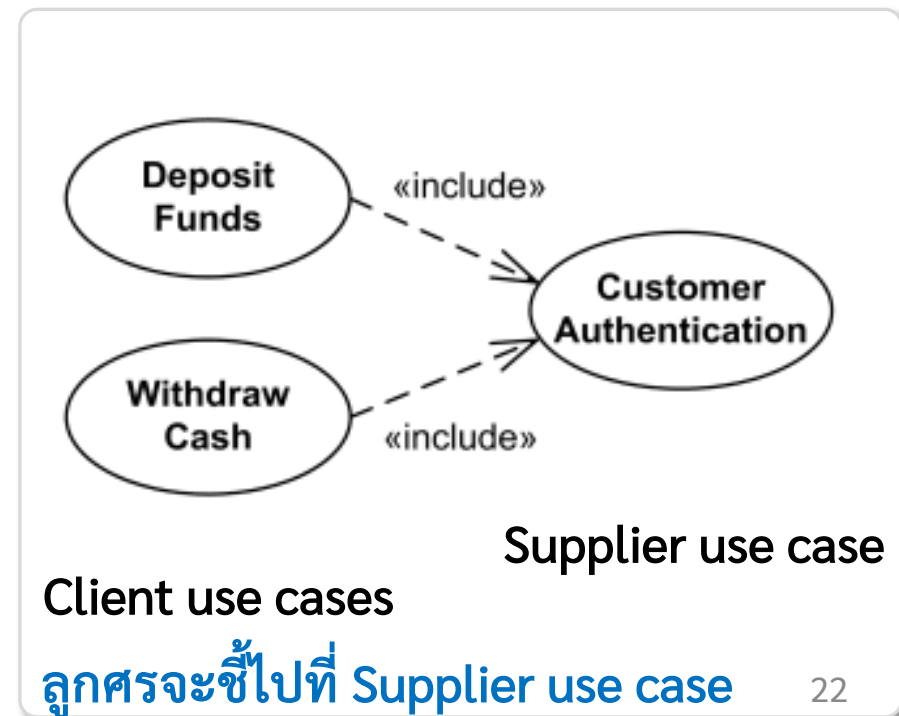
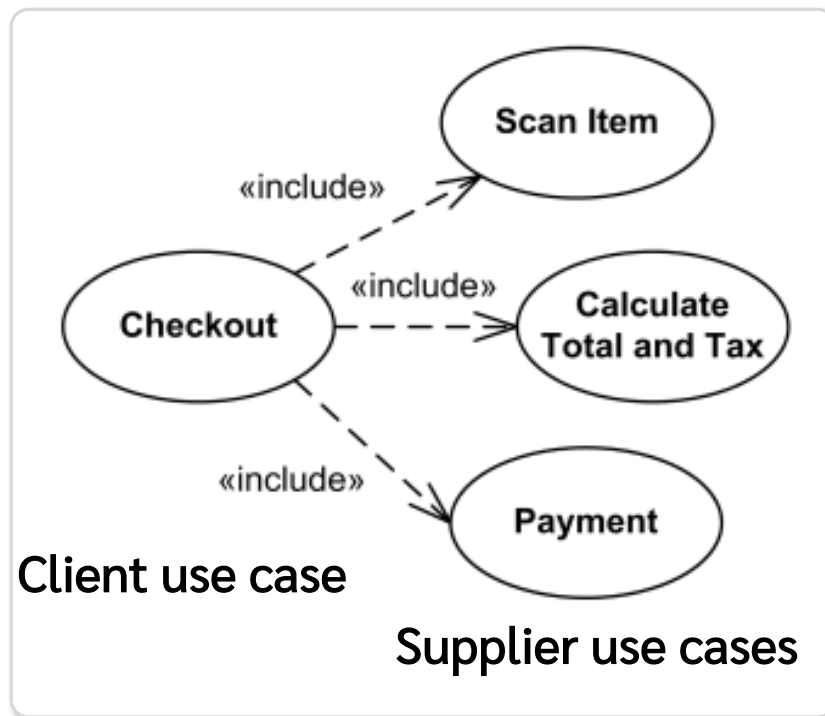
- We can specify conditions on «extend» relationships
  - Conditions are Boolean expressions
  - The insertion is made if and only if the condition evaluates to true

# Notations

## Include Relationship (*reuse*)

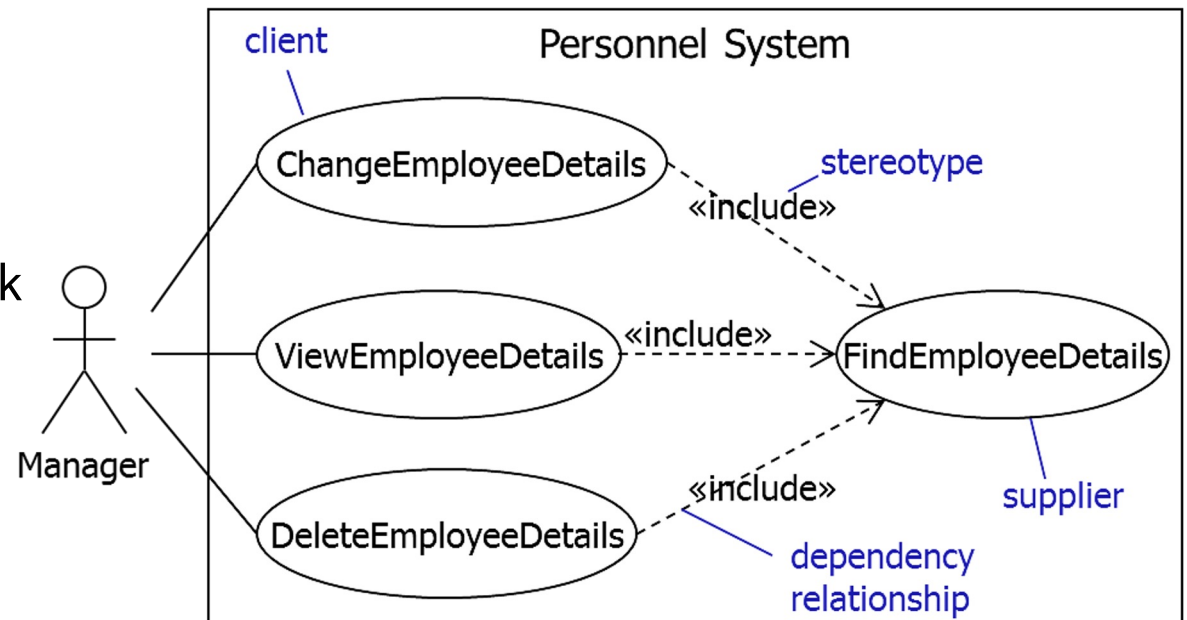
แสดงความสัมพันธ์แบบเรียกใช้ ในกรณีที่ use case หนึ่งต้องไปเรียกขั้นตอนการดำเนินงานของอีก use case หนึ่งมาใช้ **เสมอ** เพื่อให้ use case ของตัวเองบรรลุเป้าหมายได้

### สนับสนุนการนำกลับมาใช้ใหม่ของ use case (reuse)



# Notations: <<include>>

- The **client** use case executes until the point of inclusion: include (**Supplier** UseCase)
  - Control passes to the supplier use case which executes
  - When the supplier is finished, control passes back to the client use case which finishes execution
- Note:
  - Client use cases are **not complete** without the included supplier use cases
  - Supplier use cases may be complete use cases, or they may just specify a fragment of behaviour for inclusion elsewhere



When use cases share common behaviour we can factor this out into a separate supplier use case and <<include>> it in the clients

# Include vs Extend

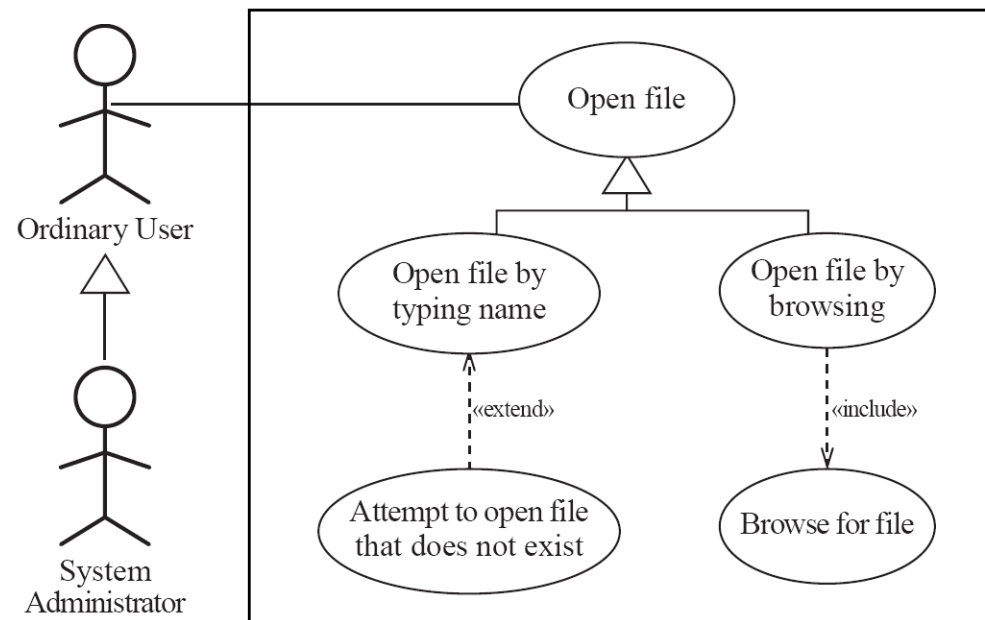
- **<<include>>** ใช้เมื่อ use case หนึ่งๆ ไม่สามารถจะบรรลุเป้าหมายของมันได้ถ้าไม่มีการดำเนินงานของ use case อื่น
  - เช่น ถ้าเป็น ธนาคาร จะถอนเงิน เราต้องมีการ ยืนยันตัวบุคคลด้วย  
เสมอ
- **<<extend>>** ใช้เมื่อ use case หนึ่งสามารถอยู่ได้หรือบรรลุวัตถุประสงค์ของมันได้เอง โดยไม่ต้องพึ่ง use case อื่น แต่ในบางกรณีหรือบางเงื่อนไข use case นั้นอาจจะต้องมีการดำเนินงานบางอย่างเพิ่มจากอีก use case แต่ไม่ทำทุกครั้ง
  - เช่น การคืนหนังสือ อาจจะมีกระบวนการคิดเงินเพิ่มในกรณีที่คืนหนังสือล่าช้ากว่าปกติ



# Notations

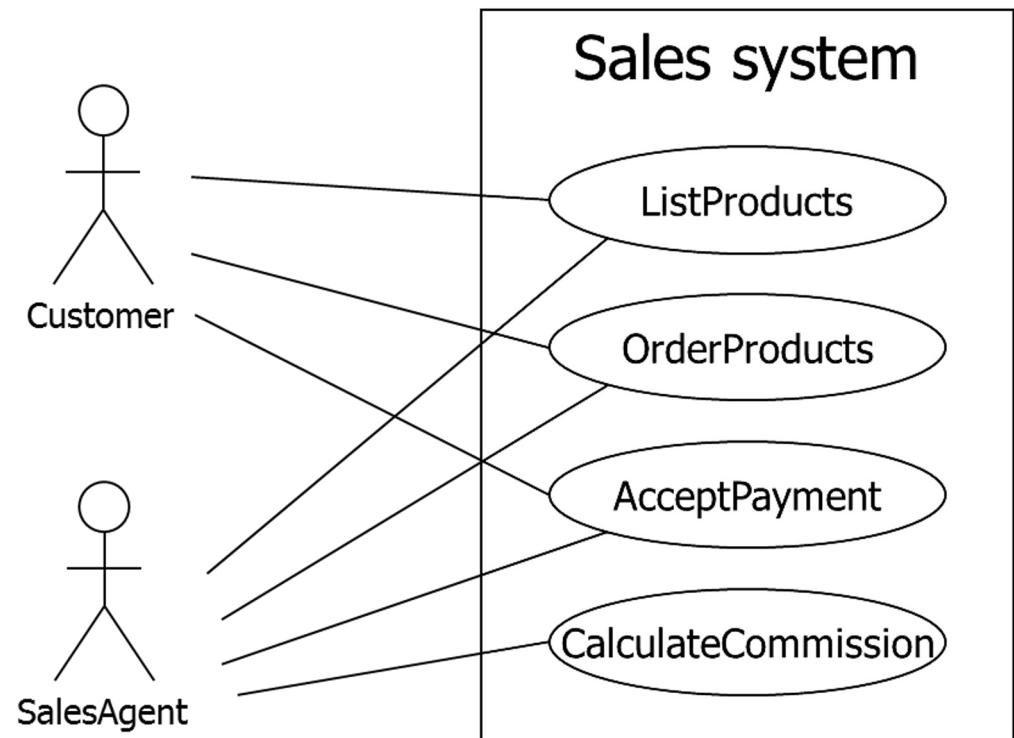
## Generalization/Specialization Relationship

- **ระหว่าง use case** ความสัมพันธ์นี้จะใช้เพื่อจำแนกประเภทของ use case คล้ายๆระหว่าง class (Child รับคุณสมบัติของ Parent และสามารถมีพฤติกรรมเพิ่มเติม)
- **ระหว่าง actor** ความสัมพันธ์ในรูปแบบการสืบทอดคุณสมบัติ บทบาทและหน้าที่เพื่อจำแนกประเภทของ actor คล้ายๆระหว่าง class



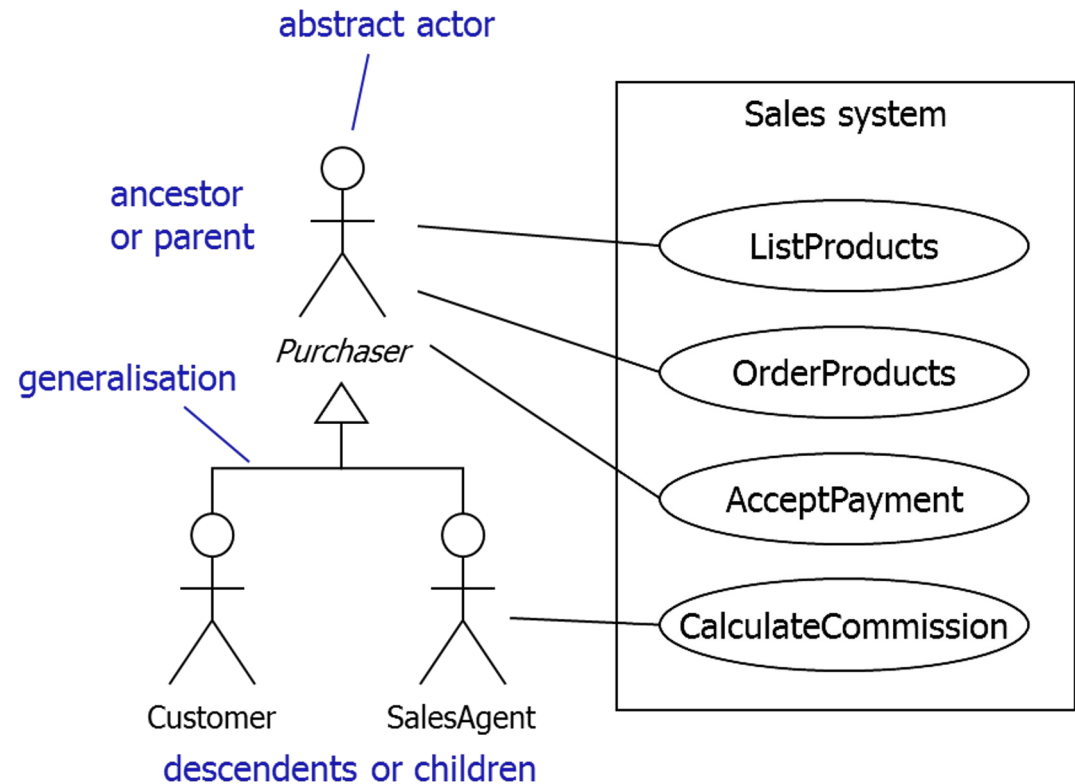
# Notations: Generalization between Actors

- The Customer and the Sales Agent actors are **very similar**
- They both interact with List products, Order products, Accept payment
- Additionally, the Sales Agent interacts with Calculate commission
- Our diagram is a *mess* – can we simplify it?





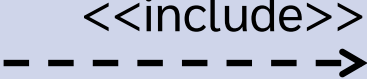

# Notations: Actor Generalization

- If two actors communicate with the same set of use cases in the same way, then we can express this as a generalisation to another (possibly abstract) actor
- The **child** actors inherit the roles and relationships to use cases held by the **parent** actor
- We can substitute a child actor anywhere the parent actor is expected. This is the *substitutability principle*

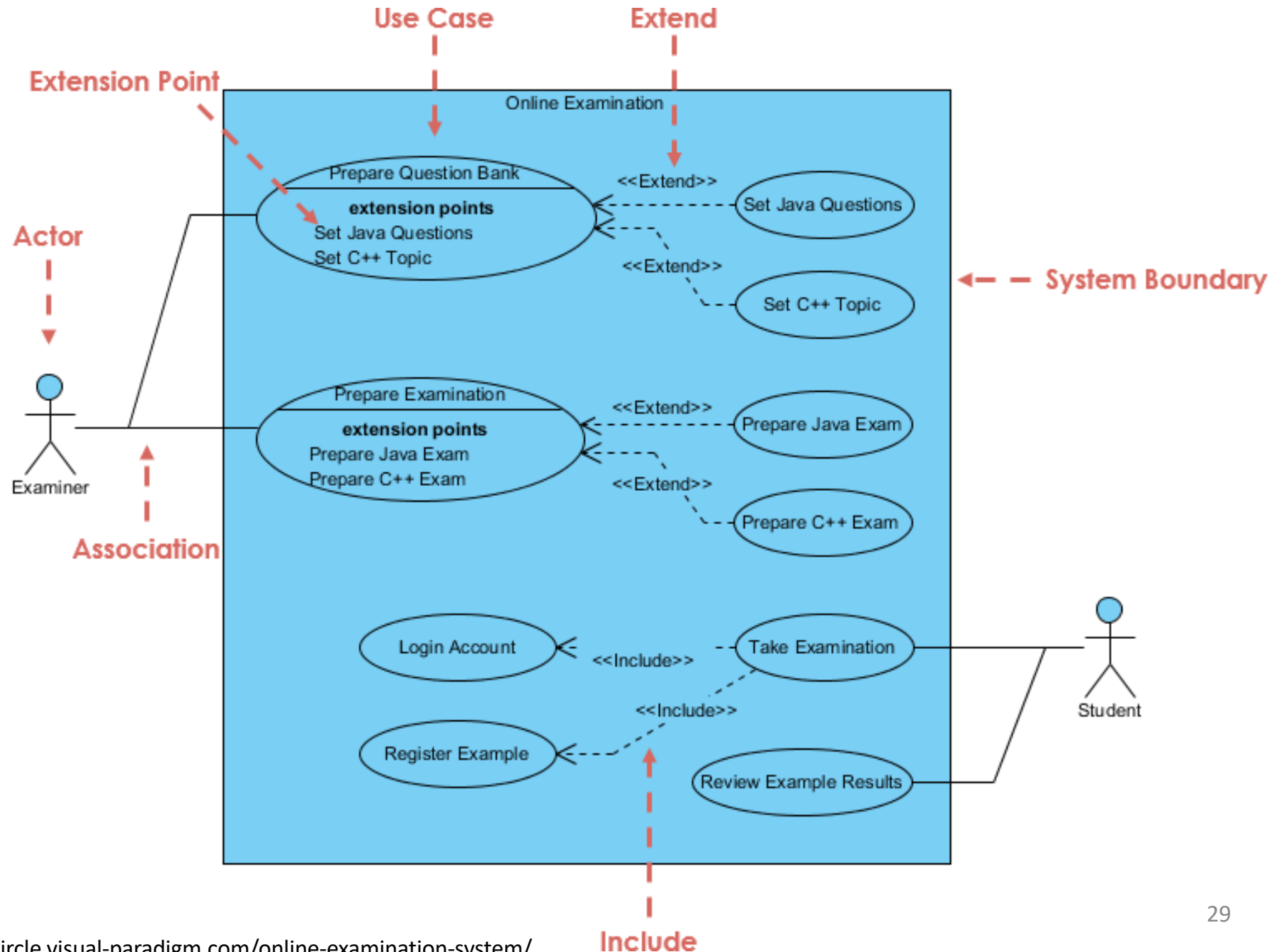


Use actor generalization when it simplifies the model

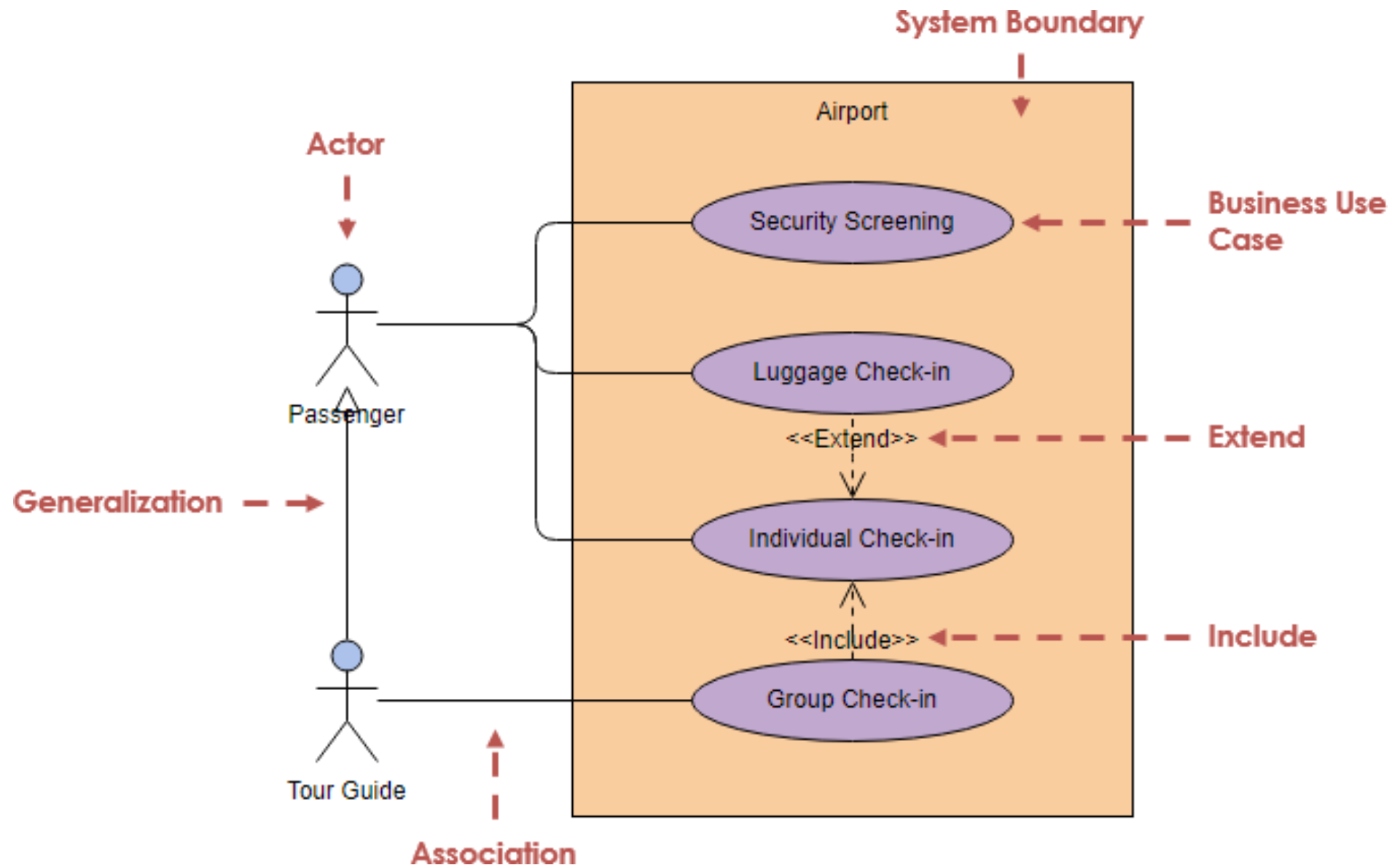
# Summary of Use Case Relationships

Relationship	Function	Notation
Association	The communication path between an actor and a use case it participates in	
Extend	The insertion of additional behavior into a base use case that does know about it	
Include	The insertion of additional behavior into a base case that explicitly describes the insertion	
Generalization	A relationship between a general use case and a more specific use case that inherits and adds features to it	

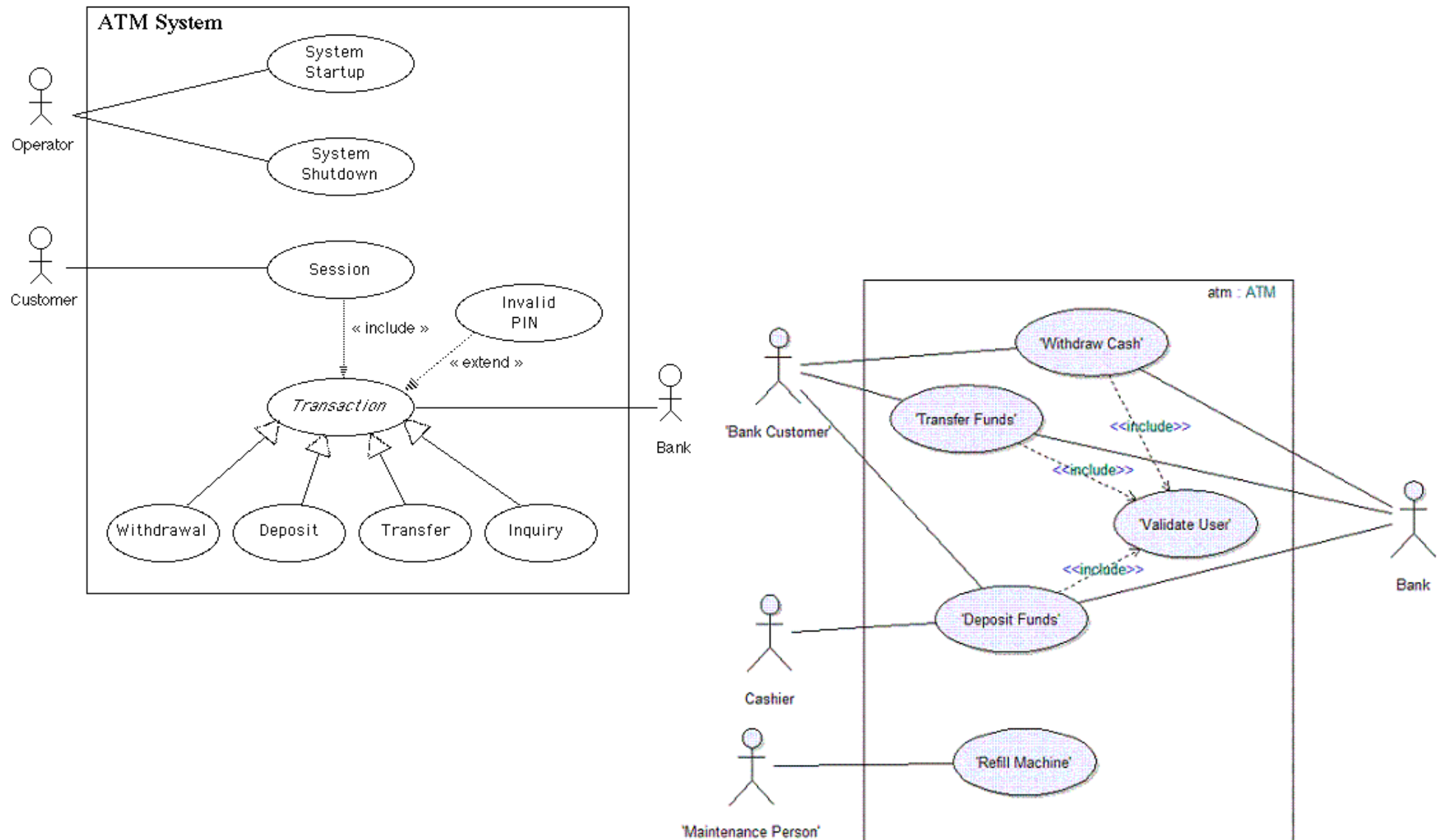
# Example: Online Examination System



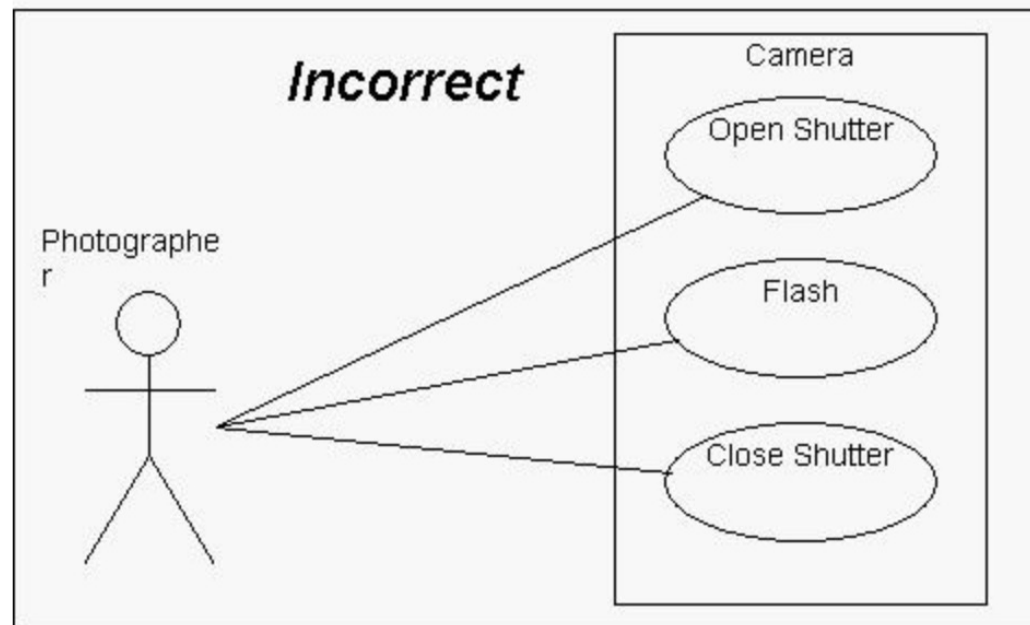
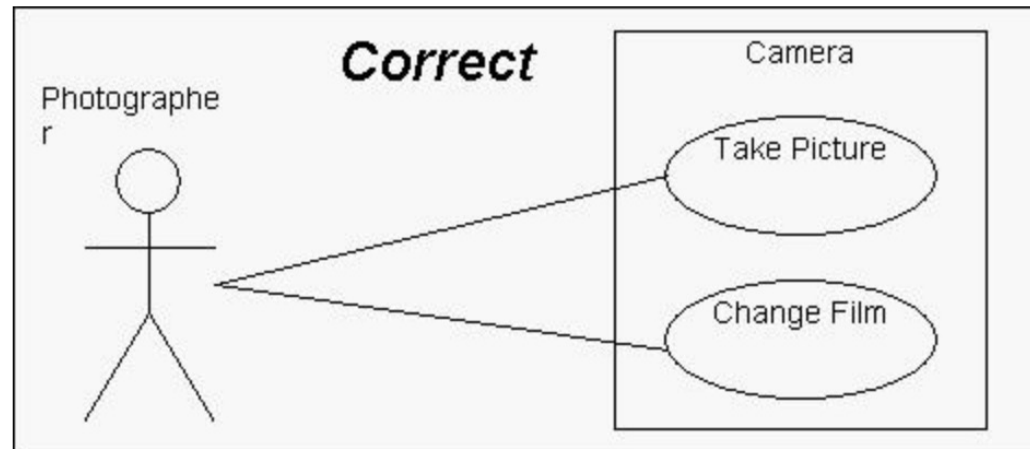
# Example: Airport Check-In System



# Example: ATM Machine



# Use Case or Not Use Case

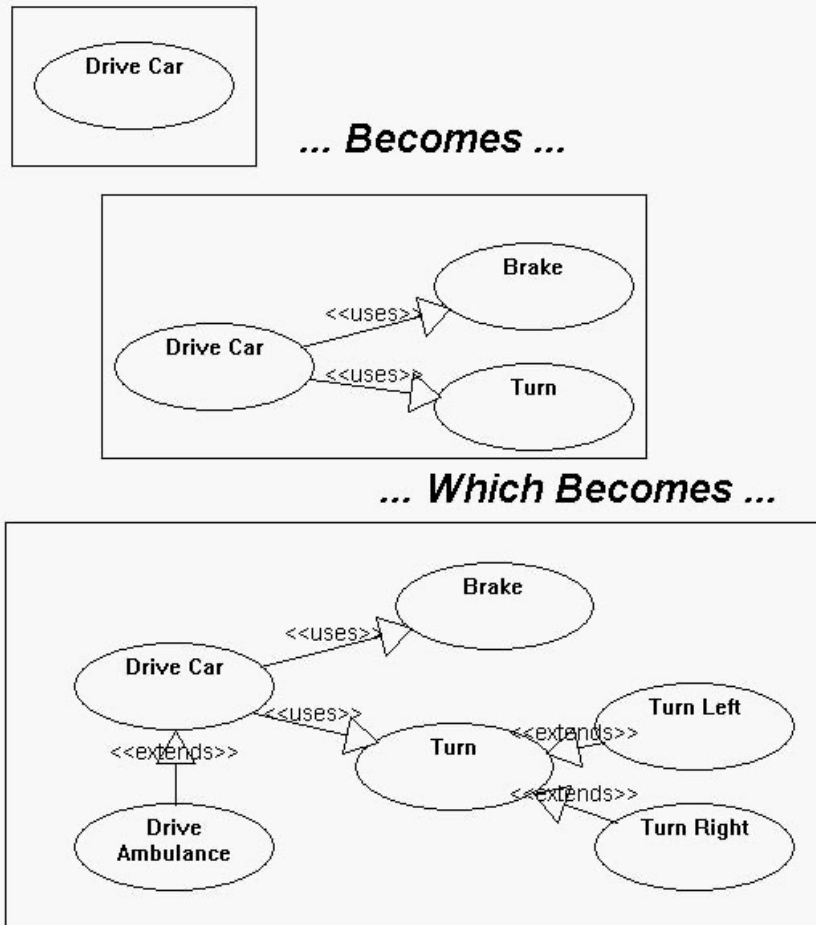


“The diagram should contain use case ovals, one for each **top-level service** that your system provides to its actors. Any kind of **internal behavior** that your system may have that is **only used by other parts of the system** should **not appear** in the diagram”

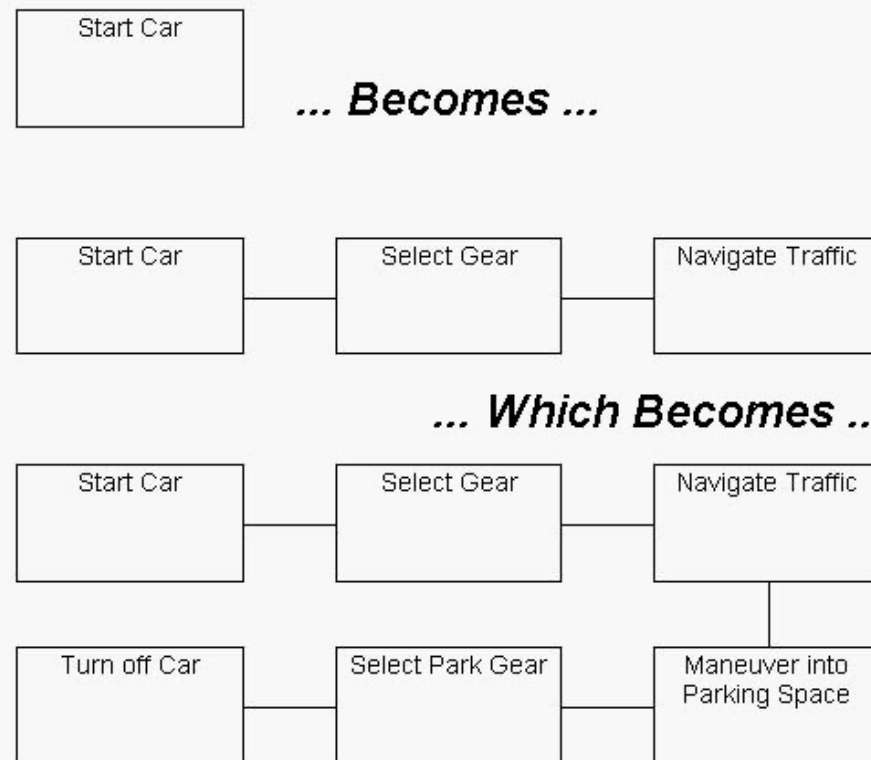


# Use Case Diagram vs Flow Chart

*Evolution of a UML Use Case Diagram*



*Evolution of a Traditional Flowchart Diagram*



“UCDs represent functionality in a top-down way, whereas flow charts represent behavior in a linear, time-based way. Also, the way you develop them is all-together different.”

# Common Mistakes in UML Diagrams

#	Mistakes
1	A use case describing activity outside the system
2	The direction of the <<include>> relationship
3	The direction of the <<extend>> relationship
4	<<include>> is used incorrectly among use cases
5	Using the System actor to represent the modeled system
6	Abuse the use of <<include>> where use case generalization should be used
7	Inappropriate use of inheritance between actors
8	Inheriting unwanted use case through generalization

# Steps To Create a Use Case Diagram

1. ค้นหา actor
2. ค้นหา base use cases ที่มีปฏิสัมพันธ์กับ actor โดยตรง
3. สร้างความสัมพันธ์ระหว่าง 1 และ 2
4. เพิ่ม use case ใหม่ อาจจะเป็น extension use case, supplier use case, หรือ base use case ที่ไม่ได้มีใน 2
5. Actor ต้องมีปฏิสัมพันธ์กับอย่างน้อย 1 use case
6. ทุก Use case ต้องมีปฏิสัมพันธ์บางอย่างกับ use case อื่นๆ หรือ actor ต่างๆ
7. ถ้ามันจะทำให้ use case diagram เข้าใจง่ายขึ้น อาจมีการจัดเรียงลำดับชั้น generalization ของ use case และ actor ต่างๆ
8. เขียน description/specification (คำอธิบาย) ของแต่ละ use case

# Check Your Understanding

ให้นักศึกษาลองวาด use case diagram จากข้อมูลดังต่อไปนี้

**ระบบจองตั๋วและ check-in ผู้โดยสารที่ใช้โดยพนักงานประจำ  
เคาท์เตอร์ของสายการบิน CS CMU.**

พนักงานสามารถจอง, ยกเลิกการจอง, และดำเนินการ check-in ผู้โดยสาร  
ได้ ในตอนทำการ check-in พนักงานจะเลือกที่นั่งให้ผู้โดยสาร โดยสายการ  
บินนี้ผู้โดยสารไม่สามารถเลือกที่นั่งได้เอง เครื่องบินทุกลำของสายการบินนี้  
จะมีที่นั่งสองแบบ แบบติดทางเดินและแบบติดเครื่องบิน ในการ check-in  
พนักงานก็จะชั่งกระเป๋เดินทางของผู้โดยสารด้วยว่าเกินหรือไม่หากเกินกว่า  
น้ำหนักกำหนดผู้โดยสารก็ต้องจ่ายค่าปรับ ส่วนในการจองและการยกเลิก  
การจองระบบนั้นทำงานไม่ต่างกับระบบทั่วไปที่ทำในส่วนนี้