University of Kent

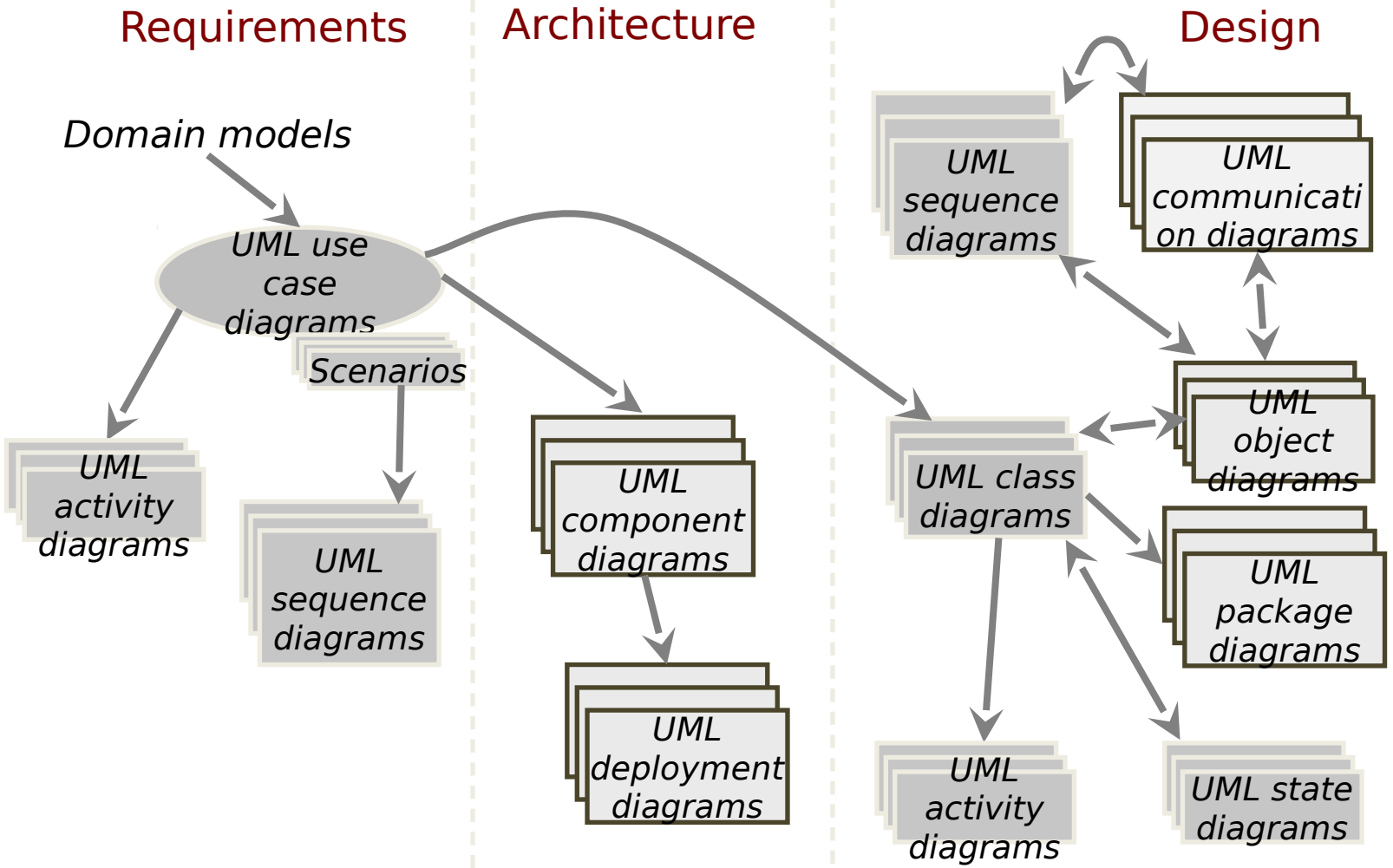
Software Engineering

Rogério de Lemos

- Use cases

OO Designs in the UML

How UML is used in the development process



University of Kent

Lecture Outline

Some of the topics

- Use cases
- Actors
- Associations
- Use case diagrams
- Extending and including use cases
- Scenarios
- Examples

University of Kent

Requirements

Requirements

- describe of the needs or desires for a product
- primary goal of the requirements phase
  - identify and document what is really needed
    - in a form that clearly communicates to the client and to development team members

In UML requirements are captured via **use cases**

Use cases are detailed with a scenario, or number of scenarios

University of Kent

Why Use Cases?

Describe requirements of the system from the point of view of a user

- a use case describes a sequence of actions a system performs that yields a value to a particular user
  - a typical interaction between a user and a computer system
  - the sequence of actions is know as a scenario

University of Kent

Use Cases

- Primary artefact in project development and planning
- Describes the requirements of a system
- Helps in requirements analysis
- Aids providing a test plan
- Creating user guide and documentation
- Validating a design
- Creating project schedule
- Risk analysis

Identifying System Boundaries

System boundaries help to find out

- what things are inside the system (to be created)
- what is outside the system (to interface with)
  - the system is whatever you are planning to create

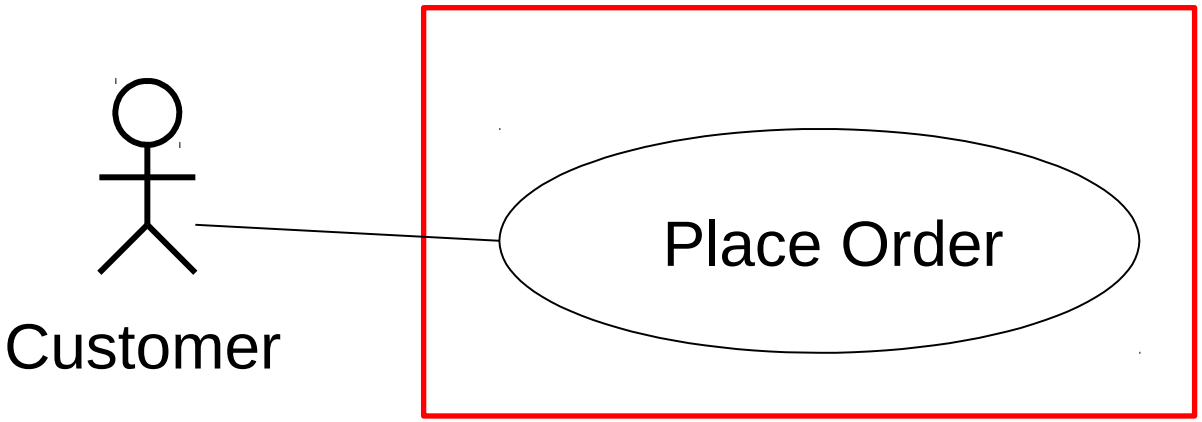
The project encompasses all things to create a system

- this includes planning, scheduling and documenting

Identifying System Boundaries

To find the boundaries you have to identify

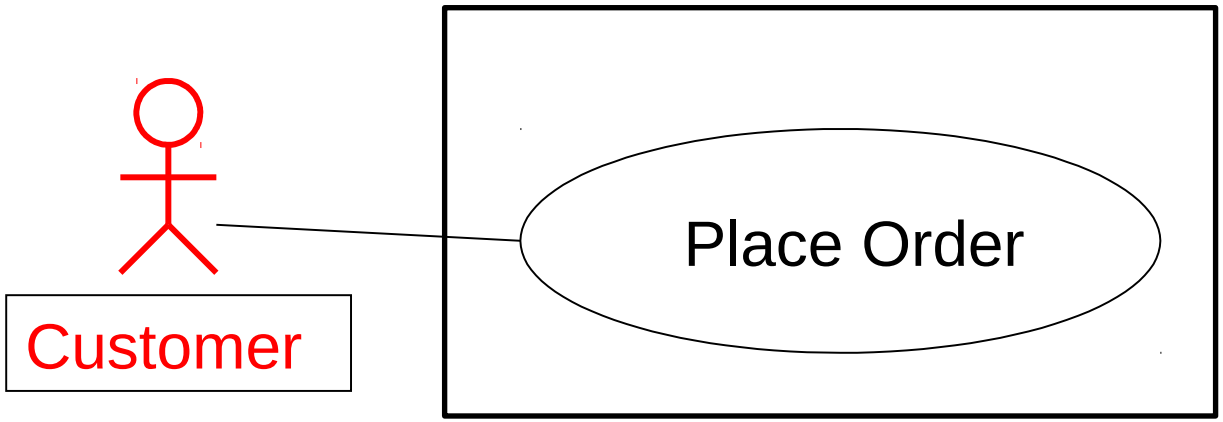
- actors
- use cases related to each actor



Actors

Actors

- anything that interfaces with the system
  - people, other software, hardware device, data stores or networks
- an actor is a role that user plays with respect to the system



University of Kent

Actors

Characteristics of actors

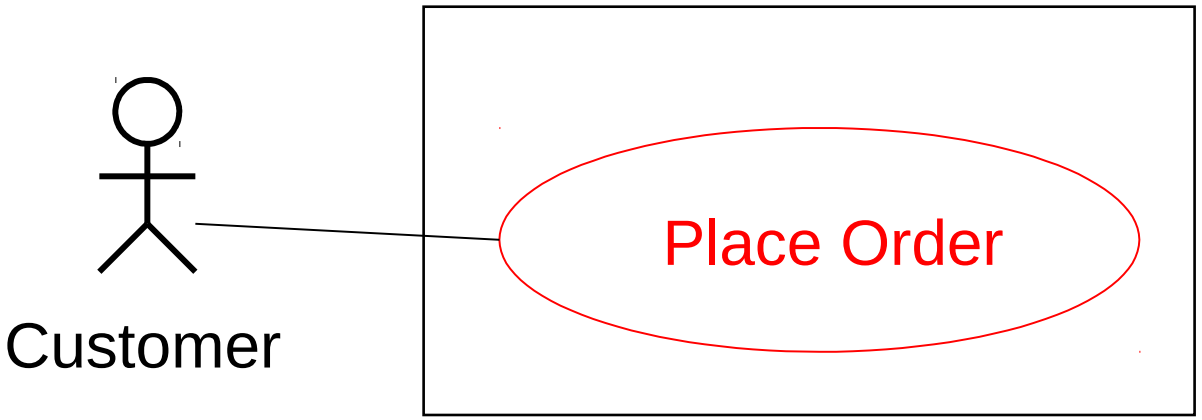
- actors are always external to the system
- we don’t have any control over actors

Use Cases

Use cases related to each actor

A use case is a behaviour of the system that produces a measurable result of value to an actor

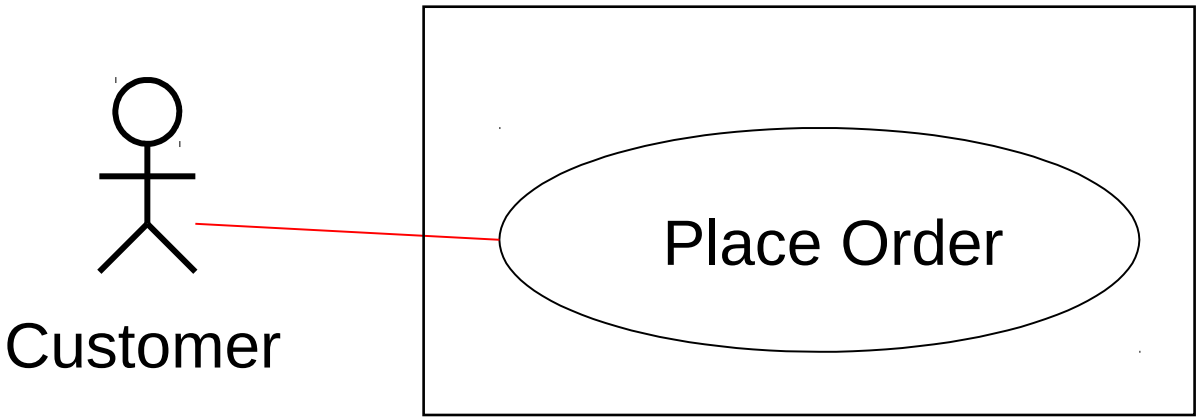
A use case should be a complete task from the perspective of the actor



Associations

Capture the relation between

- a use case and an actor
- actors
- use cases



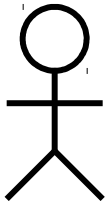
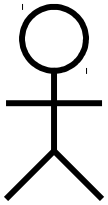
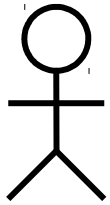
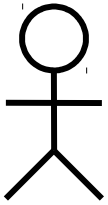
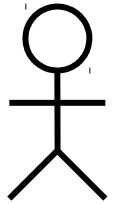
University of Kent

Case Study

Project – **mail order company**

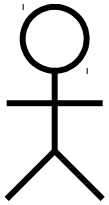
- to develop an order-processing software for a mail order company, which is a reseller of products purchased from various suppliers
- customer purchases products by submitting a list of products with the payment
- the company fills the order and ship the products to customer

Actors:
Mail Order Company

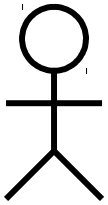






What are the actors in the example system ?

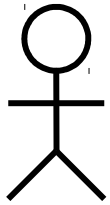
Actors:
Mail Order Company



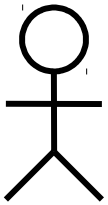
Customer



Shipping Company



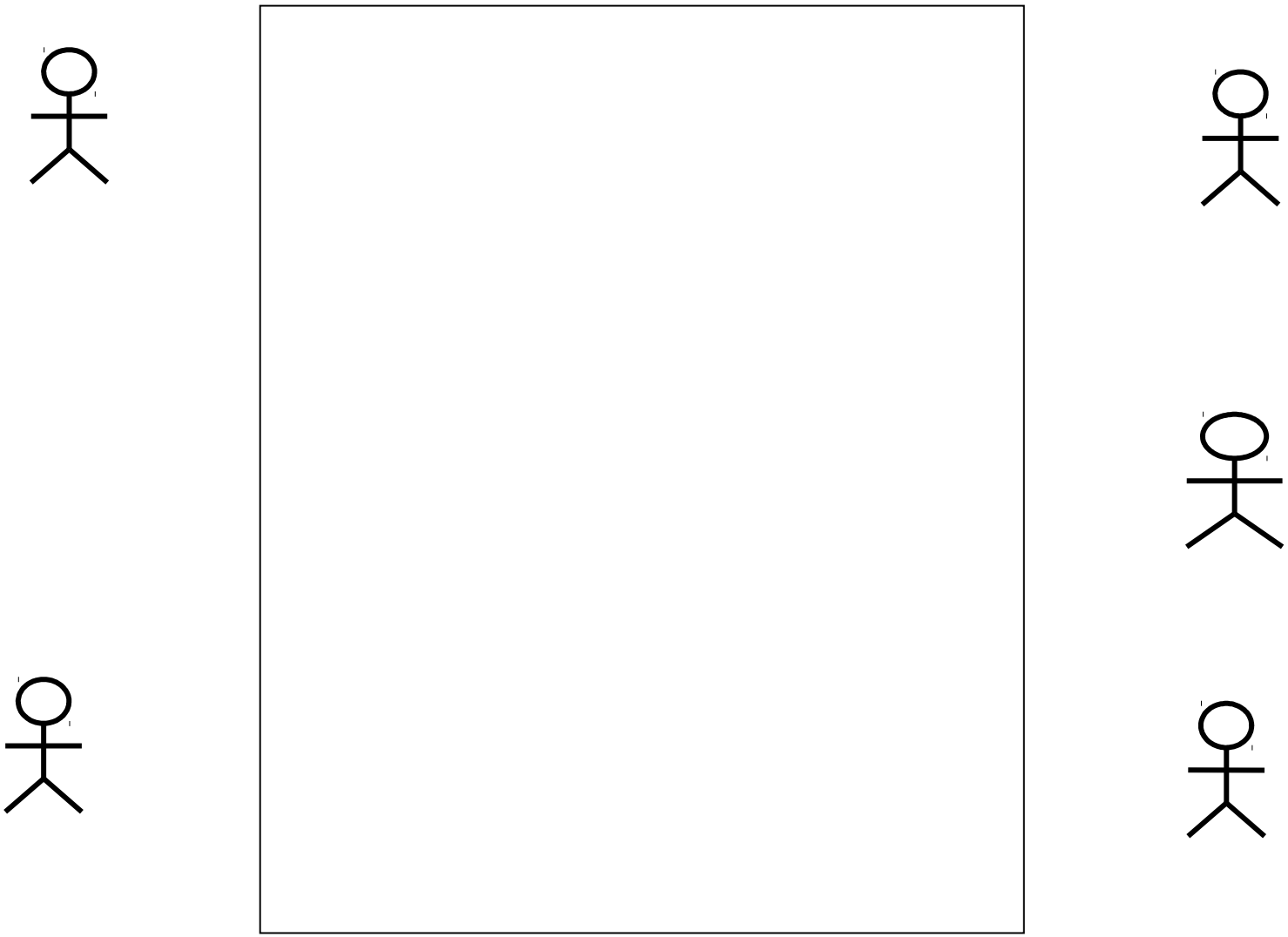
Clerk



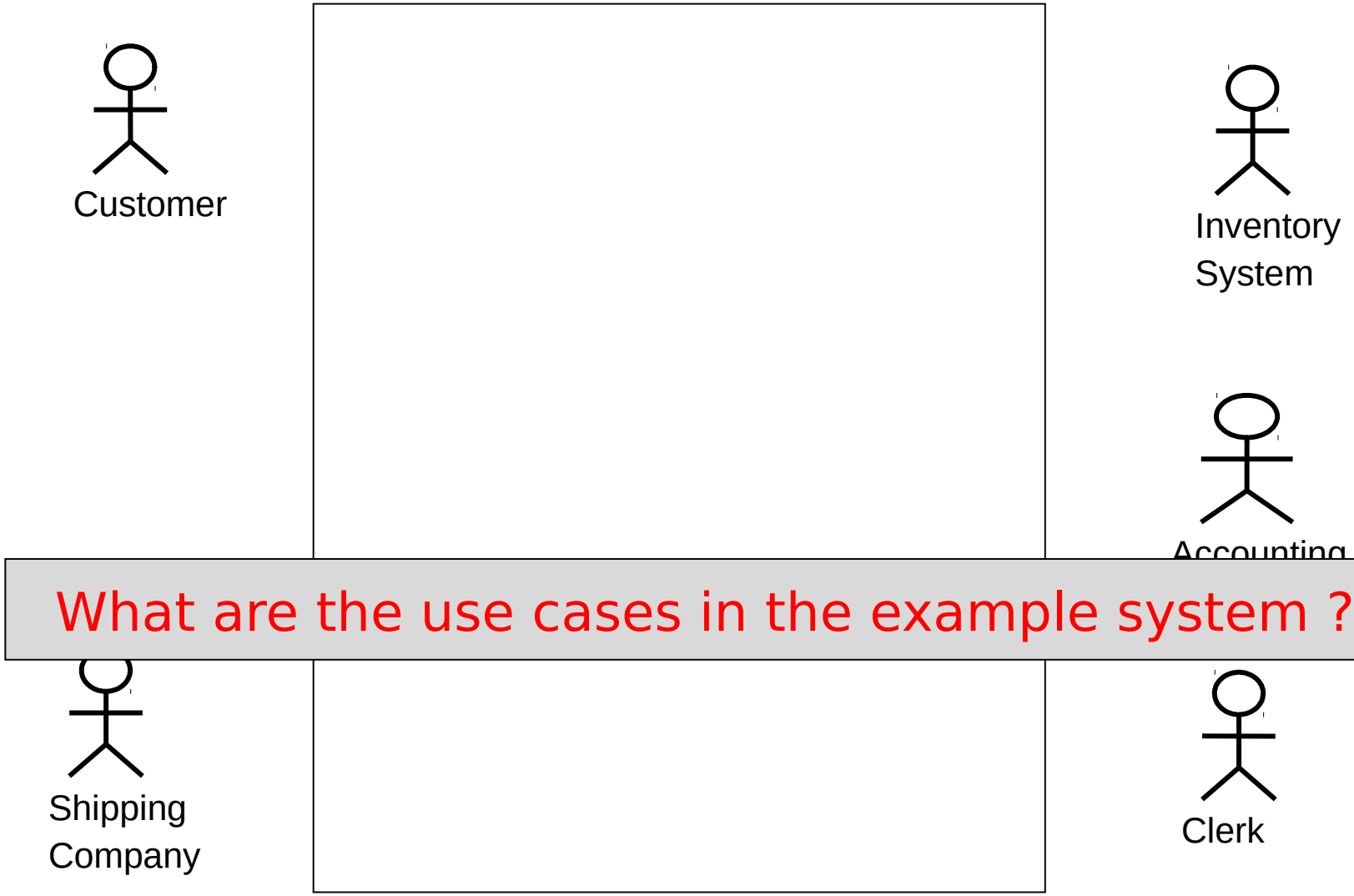
Inventory System

Accounting System

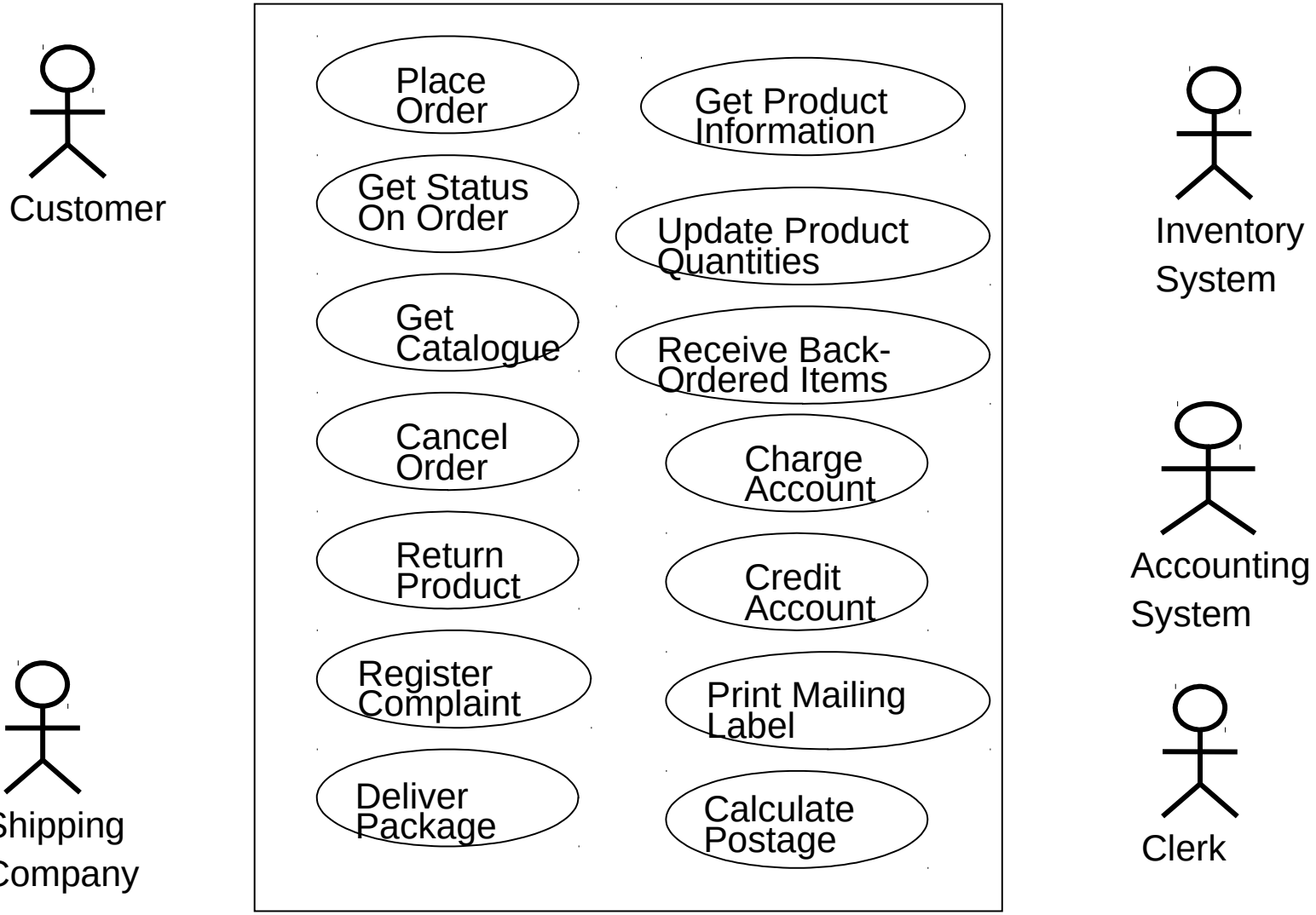
Mail Order Company



Mail Order Company

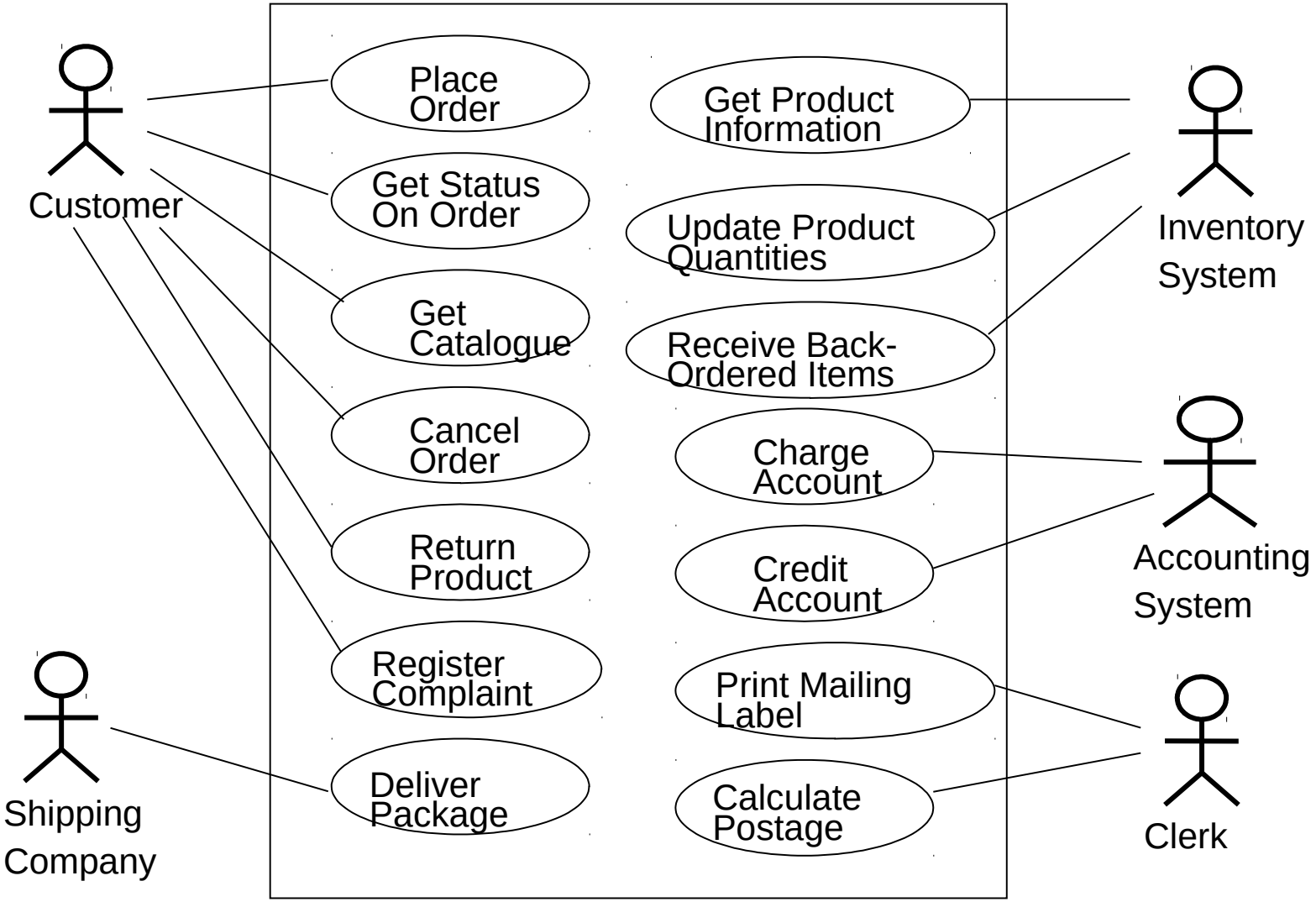


Mail Order Company



University of Kent

Mail Order Company



Detailing an Use Case

The description of an use case is a sequence of steps describing an interaction between an actor and a system

Example: **Place order** (not related to previous example)

- A customer browses the catalogue and adds desired items to the shopping basket
- When she wishes to pay, the customer inputs the shipping and credit card information and confirms the sale
- The system checks the authorisation on the credit card and confirms the sale both immediately and with a follow up email

University of Kent

Sequence of Steps

Place order

1. Customer browses the catalogue
2. Customer selects items
3. Customer goes to check out
4. Customer fills in the shipping info
5. System presents full pricing + shipping info
6. System authorises purchase
7. System confirms sale immediately
8. System sends confirmation to customer via email

Normal Path

Idealised scenario

- only the normal path
  - no branching or alternatives in your basic path

Use case: **Place order**

1. Customer browses the catalogue
2. Customer selects items
3. Customer goes to check out
4. Customer fills in the shipping info
5. System presents full pricing + shipping info
6. System authorises purchase
7. System confirms sale immediately
8. System sends confirmation to customer via email

Alternative Path

Alternative path

- a different sequence of events, normally if one or more preconditions is violated

University of Kent

Alternative Path

Example: **Place order**

1. Customer enters his/her name and address

...

6. When payment is confirmed...

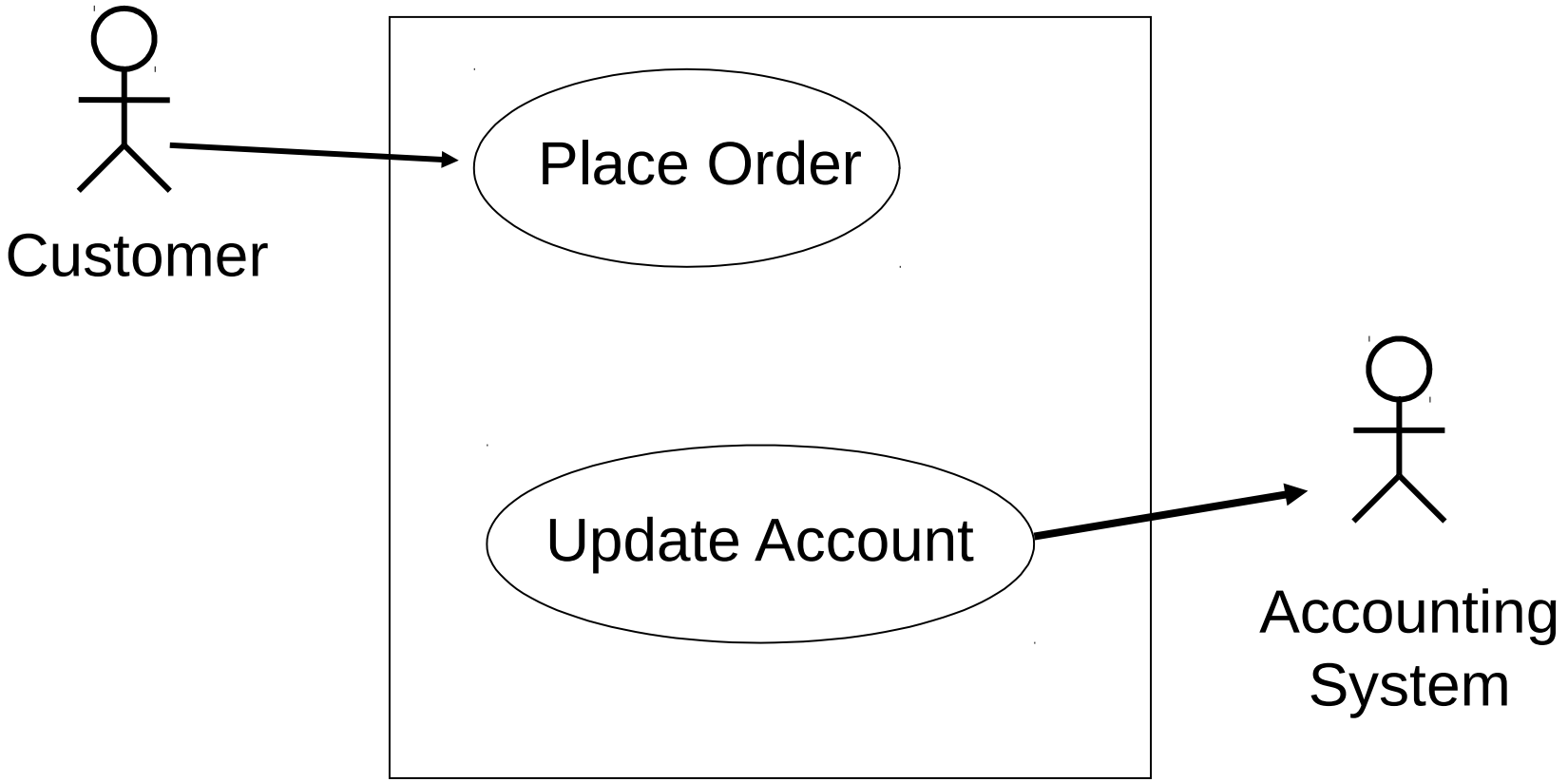
Alternative Path

- At any state before selecting submit, the customer can select cancel. The order is not saved and the use case ends
- In final step, if payment is not confirmed, the system prompts the customer to correct payment information or cancel. If the customer chooses to correct, go back to step 4 of the Normal Path, otherwise end the use case

Adding Direction

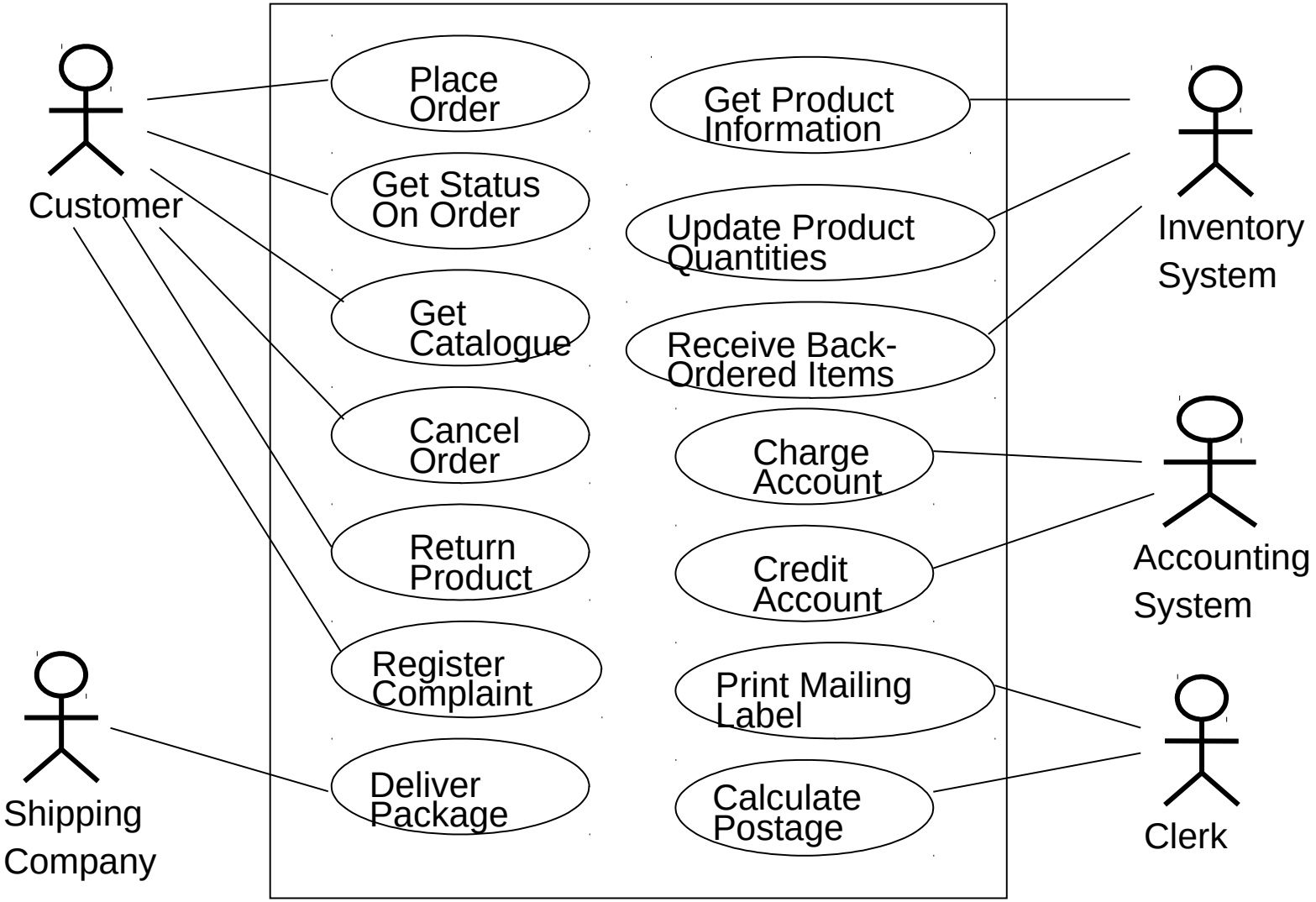
Who initiates the use case

- system, actor or both



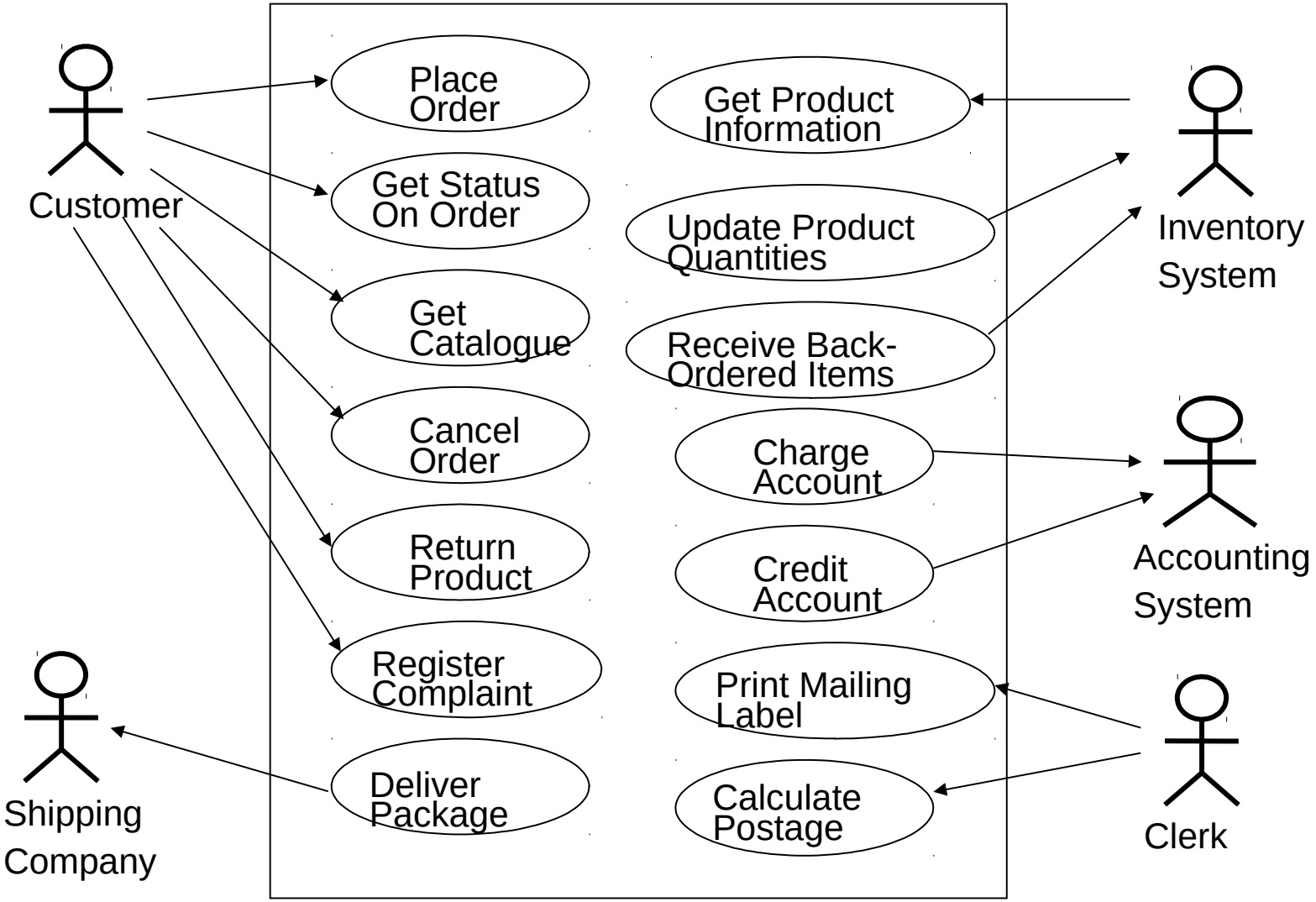
University of Kent

Mail Order Company



University of Kent

Mail Order Company



Relationships between Use Cases

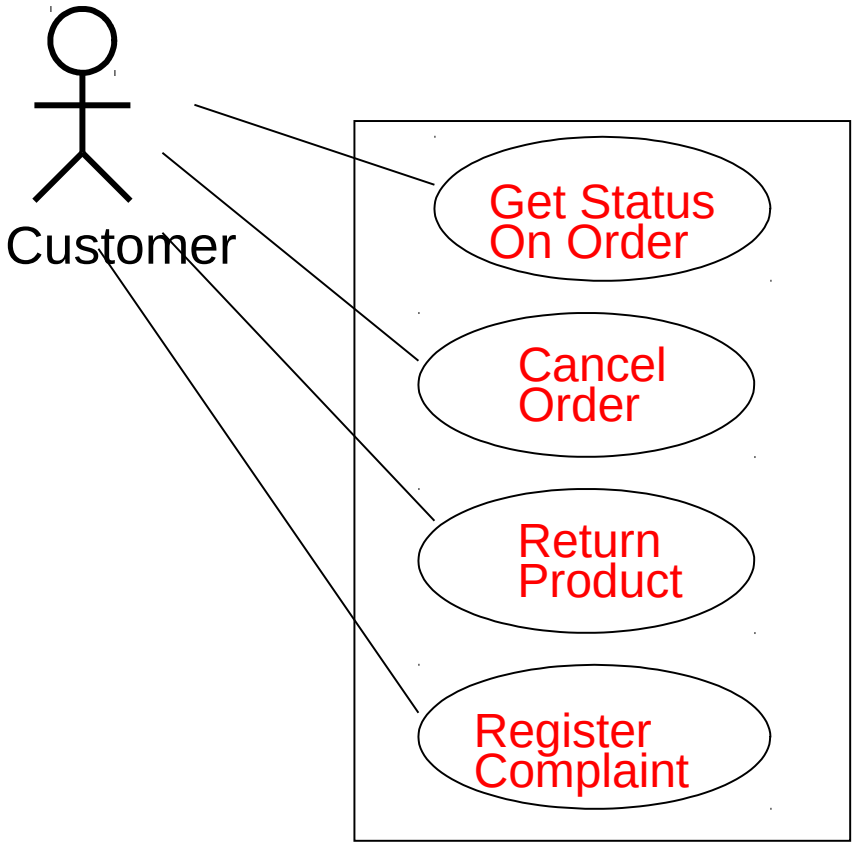
- Use cases for reuse: <<include>>
  - to factor out common behaviour from two or more use cases
- Separating variant behaviour: <<extend>>
  - two or more scenarios
    - one is the main case and the other subsidiary cases
- Generalisations
  - the idea is to show a use case and a generalised version of it
  - also applied to actors

Include

If there is something generic that you can use

- abstract the common behaviour

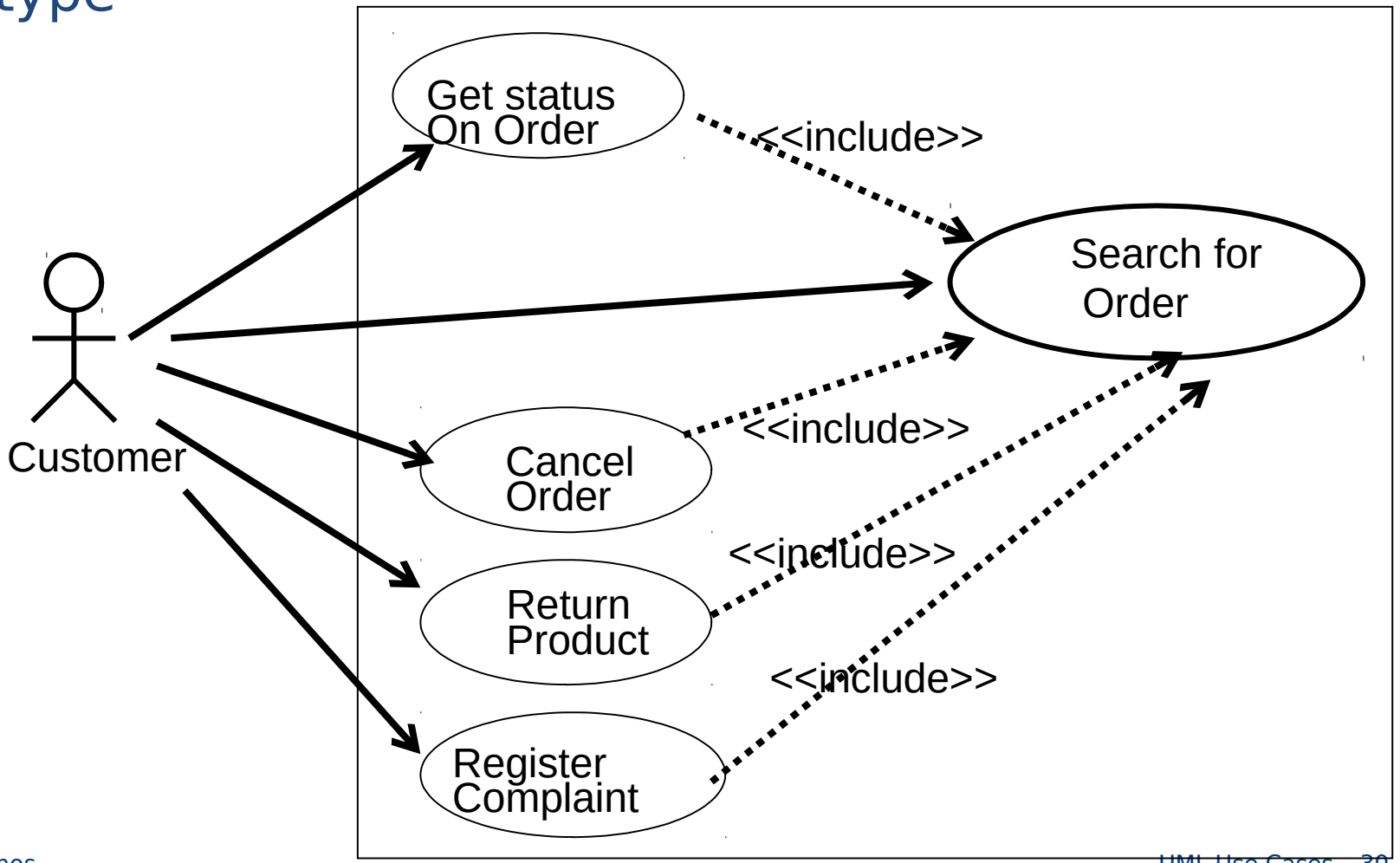
Example



All of them need to search for an order by customer ID, Order ID or name

Include

In use case diagrams, <<include>> is represented as a stereotype



University of Kent

Use Case

Use case: **Search for Order**

1. The use case begins when the Customer enters an order ID, Customer ID or Customer name
2. The Customer clicks on Find
3. If the Customer enters an order ID
   a) The system displays order and use case ends
4. If the Customer entered a Customer name or ID
   a) The system returns a list of orders for that Customer
   b) The Customer selects one order from that list
   c) The System displays that order and the use case ends

University of Kent

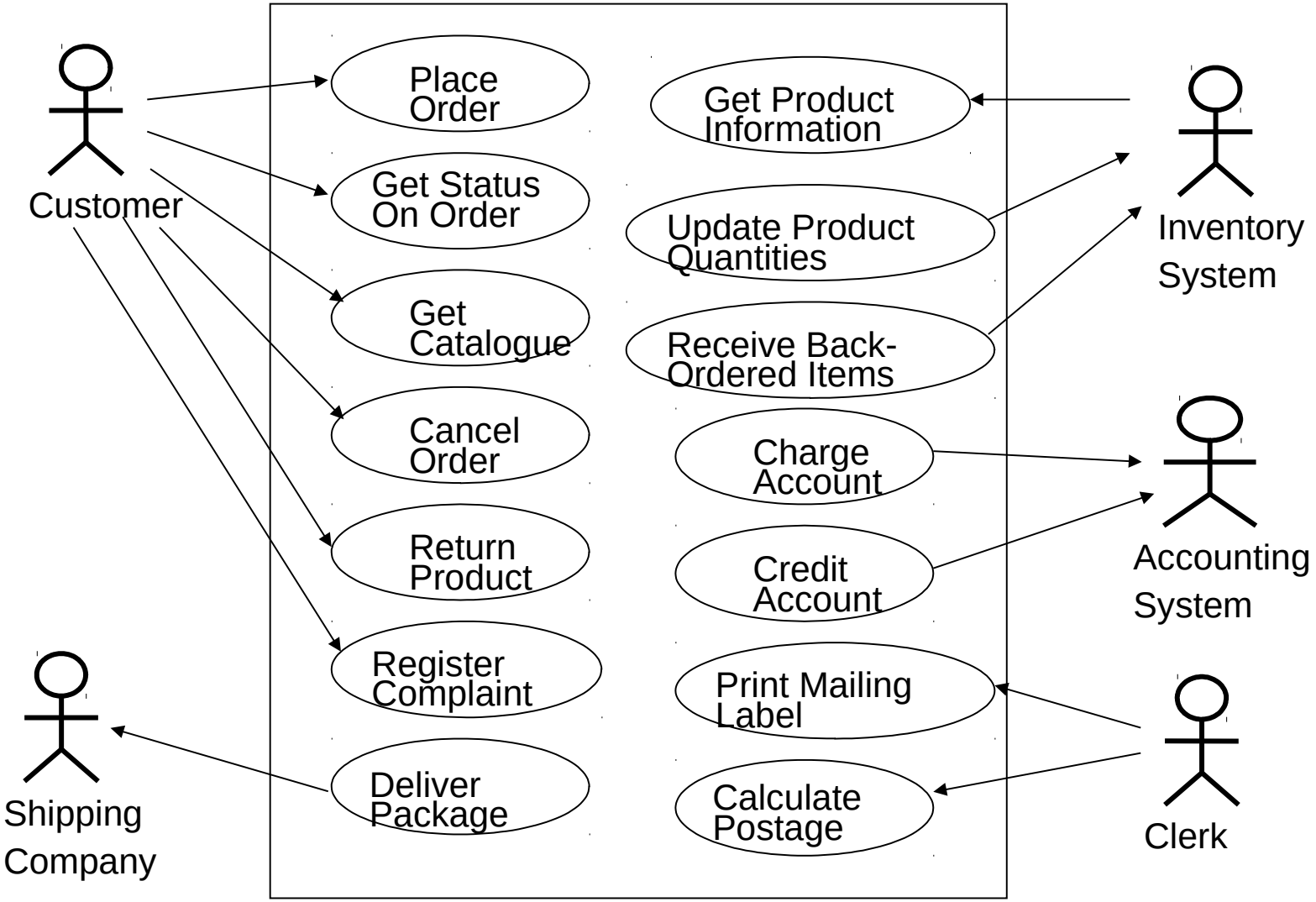
Use Case

Use case: **Cancel Order**

1. The use case begins when the Customer requests to cancel an order
2. Include **Search for Order**
3. If the order status is confirmed
   a) The system marks the order cancelled
   b) The system notifies the accounting system to credit the customer's accounts and the use case ends
4. If the order status is shipped
   a) The system notifies the Customer to return the good and the use case ends

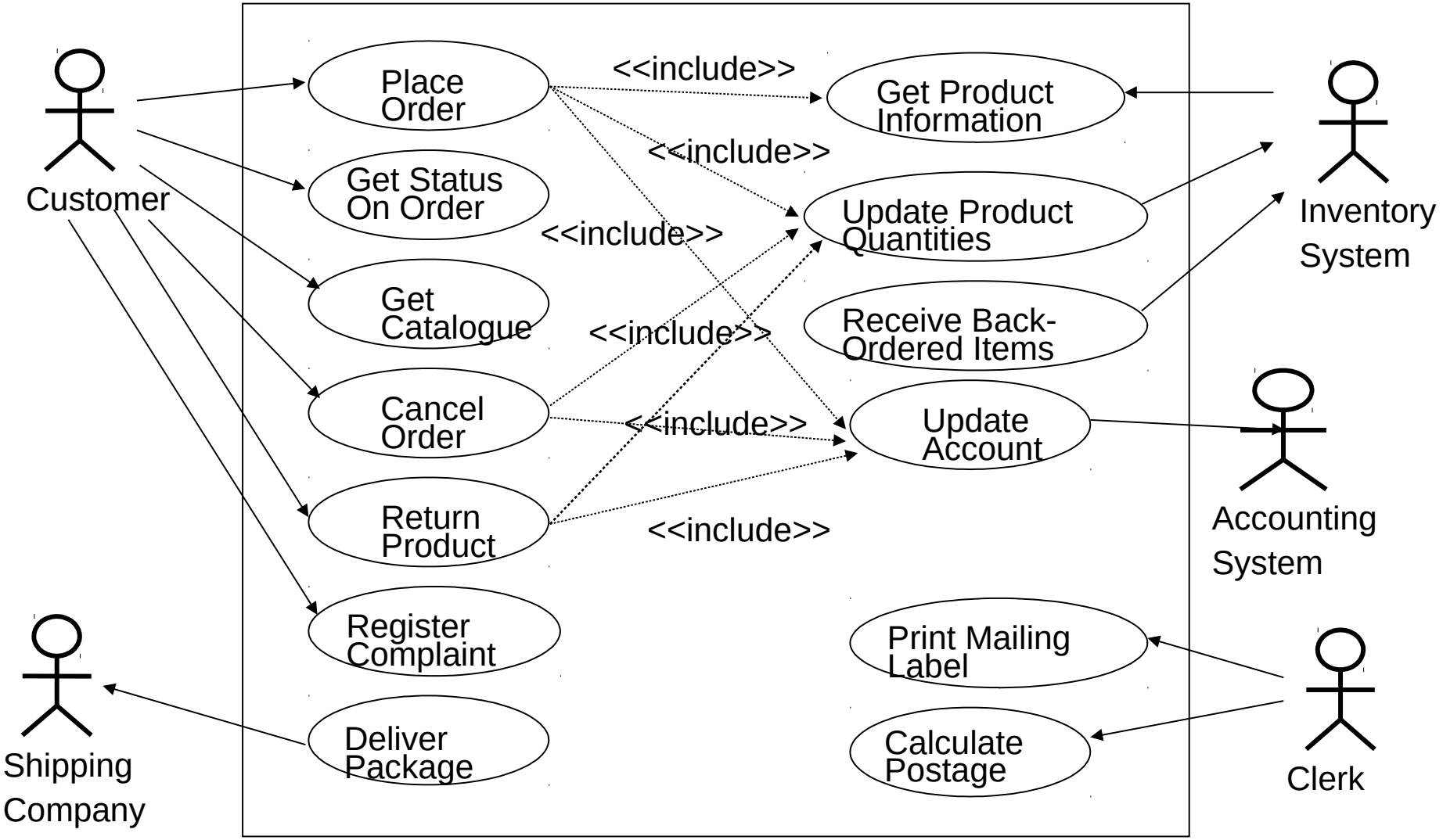
University of Kent

Mail Order Company



University of Kent

Mail Order Company



University of Kent

Extend

Conditionally extend the behaviour of an existing use case

- adding behaviour without changing the original use case

Use case: **Provide Seasonal Sale Price**

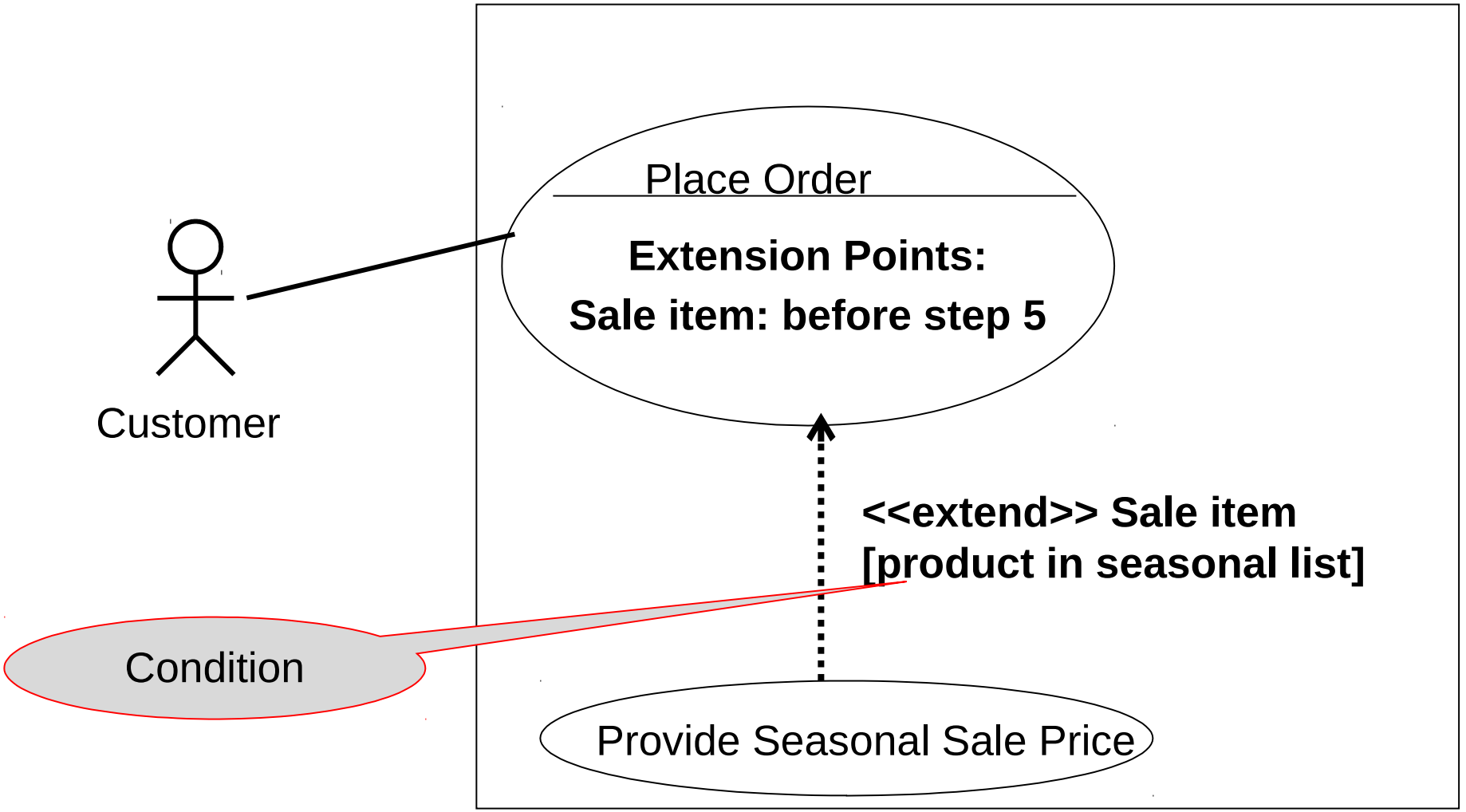
1. The use case starts when the system gets the sale for the product
2. The system displays the discounts on the order
3. The system calculates the discount amount by multiplying the original price by the sale discount
4. The system subtracts the discount amount from the total and the use case ends

Extend

Use case: **Place Order including Seasonal Sale**

1. The Customer enters his/her name and address
2. While customer enters product code
   a) The system provides product spec and price
      [Extension point: Use Case Provide Seasonal Price Sale]
   b) The system adds up the price of items
3. The customer enters credit card info
4. The customer selects submit
5. The system verifies the info, saves the order, and forwards the info to accounting system
6. When payment is confirmed, the order is marked confirmed, an order ID is issued and returned to customer

Use Case Diagram



University of Kent

Generalisation

Inheritance can be between actors and between use cases

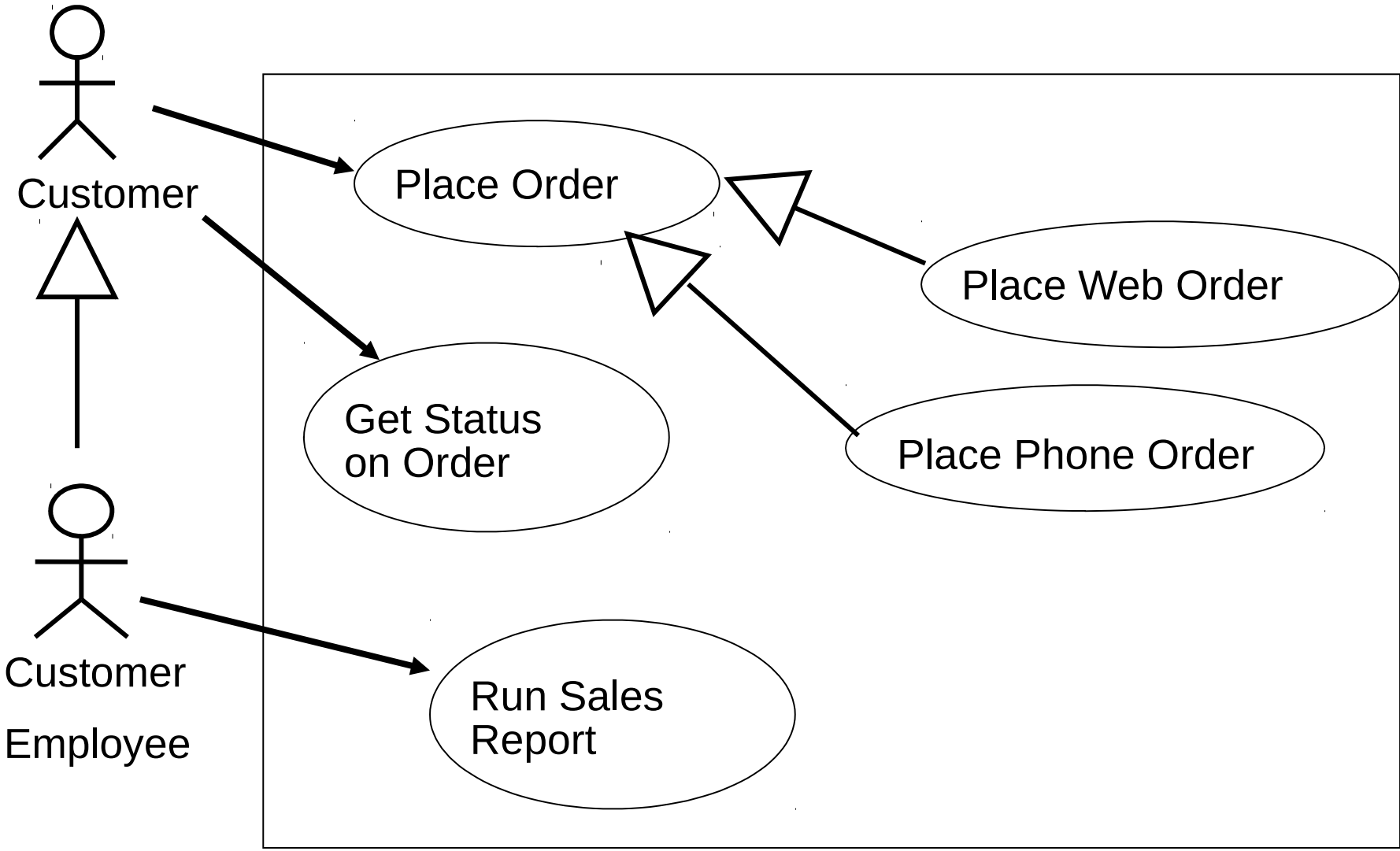
Inheritance between actors

- one actor fills the same role as other actor
- it also may fill additional role
- the subtype actor can interact with the same use case in the same way

Inheritance between use cases

- one use case is generalised version of the other
- the generalised version inherits behaviour from the case and may add into it

Example: Place Phone Order



University of Kent

Use Case

Use case: **Place Phone Order**

1. The use case starts when the Customer calls a Customer rep.
2. The Customer rep. gets the Customer ID form the Customer and enters it into the system
3. The system gets the Customer name and address from the database
4. The Customer rep. verifies the info with the Customer
5. The Customer rep. gets the product code and enters them into the system
6. For each product code entered

   a) The system supplies a product description and price

   b) The system adds the price of the items to the total end loop

University of Kent

Use Case (cont.)

7. The Customer rep. get the payment information from the Customer and enters it into system.
8. The Customer rep. submit the order to the system.
9. The system saves the order as pending and forward payment information to the accounting system
10. When the payment confirmed, the order is marked confirmed, an order ID is retuned to the Customer, and the use case ends.

University of Kent

Use Case Formats

[More details see A. Cockburn book on use cases]

There are different formats for writing use cases

- fully dressed
- one column of text (not a table)
- numbered steps
- no if statements
- numbering convention for extensions (e.g., 2a, 2a1, 2a2,…)
- casual
- one-column table
- two-column table
- RUP style

Description of Use Cases

Simple Table (to be used in the module as a template)

Use case name	`Place order`
Participating actors	`Customer`
Flow of events: Normal flow	Normal path 1. `Customer` browses the catalogue 2. `Customer` selects items 3. `Customer` goes to check out 4. `Customer` fills in the shipping info 5. `Mail Order company` presents full pricing + shipping info 6. `Customer` pays using a credit card 7. `Mail Order company` authorises purchase 8. `Mail Order company` confirms sale immediately 9. `Mail Order company` sends confirmation to `Customer` via email
Flow of events: Alternative flow	`Customer` credit card details don't match 1. `Mail Order company` cancels purchase 2. ...
Pre-condition	• `Customer` starts transaction
Post-condition	• `Customer` has received a confirmation • `Customer` has received an explanation why purchase couldn't go ahead

University of Kent

Use Case Formats

Fully Dressed

Use Case 24 – Fully Dressed Use Case Template <name>

<the name should be the goal as a short active verb phrase>

Context of Use: <a longer statement of the goal, if needed, its normal occurrence conditions>

Scope: <design scope, what the system is being considered black-box under design>

Level: <one of: summary, user-goal, subfunction>

Primary Actor: <a role name for the primary actor or description>

Stakeholders and Interests: <list of stakeholders and key interests in the use case>

Precondition:<what we expect is already the state of the world>

Minimal Guarantees: <how the interests are protected under all exits>

Success Guarantees: <the state of the world if goal succeeds>

Trigger: <what starts the use case, may be time event>

Use Case Formats

Fully Dressed (cont.)

Main Success Scenario:

<put here the steps of the scenario from trigger to goal delivery and any cleanup after>

<step#><action description>

Extensions:

<put here the extensions, one at the time, each referring to the step of the main scenario>

<step altered><condition>:<action or sub use case>

<step altered><condition>:<action or sub use case>

Technology & Data Variations List:

<put here the variations that will cause eventual bifurcation in the scenario>

<step or variation #><list of variations>

<step or variation #><list of variations>

Related Information:

<whatever your project needs for additional information>

University of Kent

Use Case Formats

One-Column Table

Use Case #	<the name should be the goal as a short active verb phrase>	
Context of Use	<a longer statement of the goal, if needed, its normal occurrence conditions>	
Scope	<design scope, what the system is being considered black-box under design>	
Level	<one of: summary, user-goal, subfunction>	
Primary Actor	<a role name for the primary actor or description>	
Stakeholders and Interests	**Stakeholder**	**Interest**
	<stakeholder name>	<put here the interest of the stakeholder>
	<stakeholder name>	<put here the interest of the stakeholder>

Use Case Formats

Precondition	<what we expect is already the state of the world>	
Minimal Guarantees	<how the interests are protected under all exits>	
Success Guarantees	<the state of the world if goal succeeds>	
Trigger	<what starts the use case, may be time event>	
Description	**Step**	**Action**
	1	<put here the steps of the scenario from trigger to goal delivery and any cleanup after>
	2	<…>
Extensions	**Step**	**Branching Action**
		<condition causing branching> : <action or sub use case>
Technology & Data Variations List	1	<list of variations>

University of Kent

Use Case Formats

Two-Column Table

- also known as *conversations*
  - prepared for designing the user interface in terms of two columns
    - the primary actor's actions in the left and the system's actions on the right
- the difficulty is to capture behavioural requirements involving supporting actors
  - no third column has yet been considered
- aimed at user interface requirements rather that overall system behavioural requirements

Use Case Formats

Two-Column Table

Customer	System
Enters order number	
	Detects that the order number matches the winning number of the month
	Registers the user and order number as this month's winner
	Sends an e-mail to the sales manager
	Congratulates the customer and gives her instructions on how to collect the prize
Exists the system	

Use Case Formats

RUP Style

- it should include a use case diagram

1. Use Case Name

1.1. Brief Description

…text..

1.2. Actors

…text..

1.3. Triggers

…text..

2. Flow of Events

2.1. Basic Flow

…text..

2.2. Alternative Flows

2.2.1. Condition 1

…text..

2.2.2. Condition 2

…text..

2.2.3. Condition 3

…text..

3. Special Requirements

3.1. Platform

…text..

3.2…

…text..

4. Preconditions

…text..

5. Postconditions

…text..

6.Extensions Points

…text..

University of Kent

Scenarios

A use case is an abstraction that describes all possible scenarios involving the described functionality

A **scenario** is an instance of a use case describing a concrete set of actions

- use cases are used to describe all possible cases
  - focus on completeness
- scenarios are used to illustrate common cases
  - scenarios focus on understandability

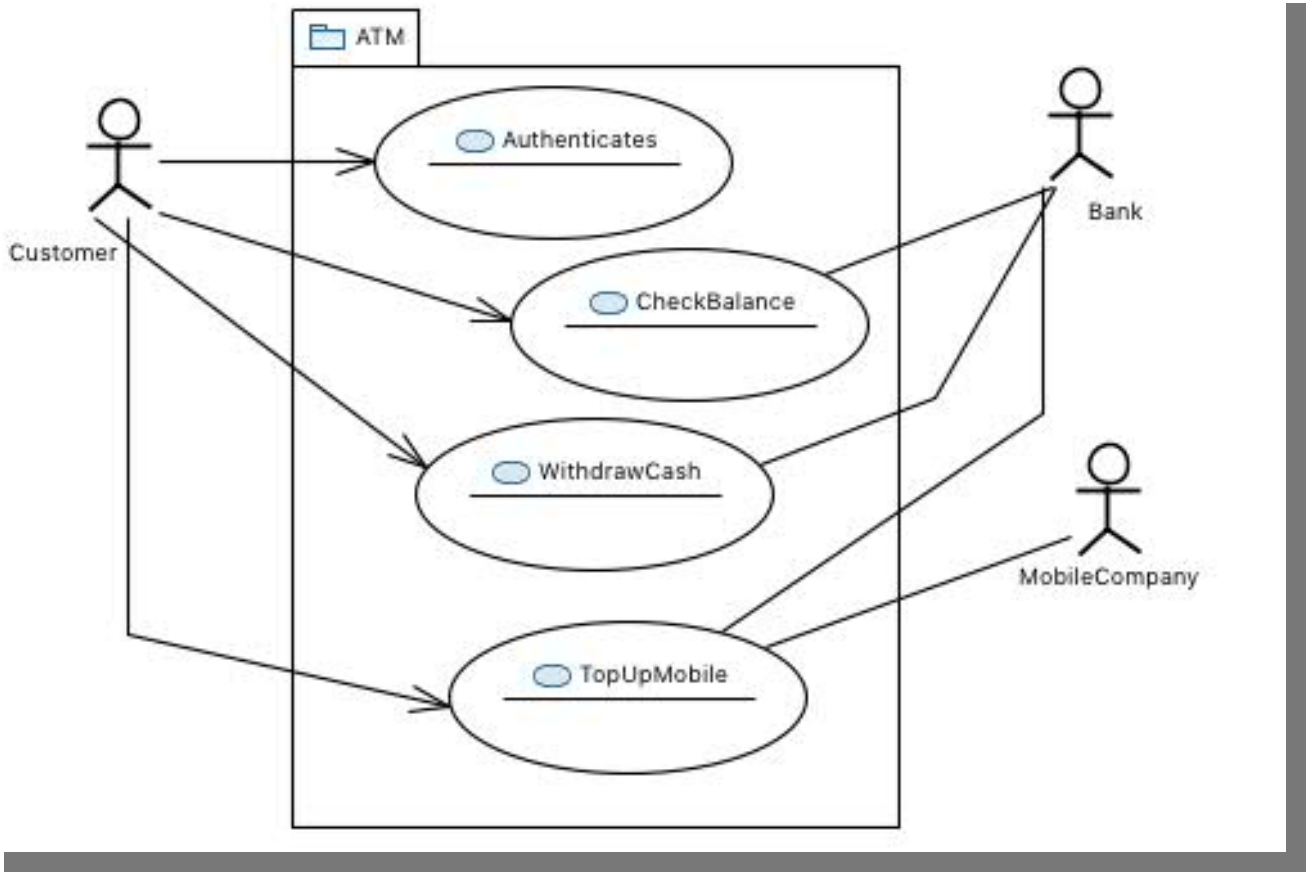
Scenario Example

Instance of the Place order use case

Use case name	`Buy Computer`
Participating actors	`Bob`
Flow of events: Normal flow	Normal path 1. `Bob` browses the catalogue 2. `Bob` selects items 3. `Bob` goes to check out 4. `Bob` fills in the shipping info 5. `UltimateGoods` presents full pricing + shipping info 6. `Bob` pays using a credit card 7. `UltimateGoods` authorises purchase 8. `UltimateGoods` confirms sale immediately 9. `UltimateGoods` sends confirmation to `Bob` via email
Flow of events: Alternative flow	`Bob` credit card details don't match 1. `UltimateGoods` cancels purchase 2. ...
Pre-condition	• `Bob` starts transaction
Post-condition	• `Bob` has received a confirmation • `Bob` has received an explanation why purchase couldn't go ahead

Example: ATM

- Consider a cash machine (ATM) that provides the following services to a customer: authentication, check balance, withdraws cash, and top-up her/his mobile phone.



Example: ATM

Use case name	`Authenticates`
Participating actors	`Customer`
Flow of events: Normal flow	Normal path 1. `Customer` inserts card 2. `ATM` requests PIN 3. `Customer` enters PIN 4. `ATM` presents the services window
Flow of events: Alternative flow	`Customer` enters wrong PIN 1. `ATM` gives another two chances 2. `ATM` confiscates card
Pre-condition	• `ATM` idle state
Post-condition	• `Customer` authenticated • `Customer` not authenticated • `Customer` card confiscated by `ATM`

Example: ATM

Use case name	`TopUpMobile`
Participating actors	`Customer, Bank, MobileCompany`
Flow of events: Normal flow	Normal path 1. `ATM` prompts the top up screen 2. `Customer` choses service provider 3. `Customer` enters the amount to top up 4. `ATM` requests confirmation from `customer` 5. `ATM` contacts `Bank` 6. `Bank` removes top up amount from `customer` account 7. `Bank` confirms transaction to `ATM` 8. `ATM` inform `MobileCompany` to top up `customer` mobile phone 9. `ATM` confirms to `customer` success of transaction
Flow of events: Alternative flow	`Customer` doesn't have enough funds 1. … `Customer` provides wrong number 2. …
Pre-condition	• `Customer is` authenticated
Post-condition	• `Customer` topped his mobile and less money on the bank account • `Customer` not successful in topping up mobile

University of Kent

Example: Accident Management System

- field officers, like police officer or fire fighter, have access to an application that enable them to contact a dispatcher for dealing with emergencies
- the dispatcher can visualise the status of all the resources (ambulances, fire trucks and police cars), and dispatch appropriate resources to the emergency

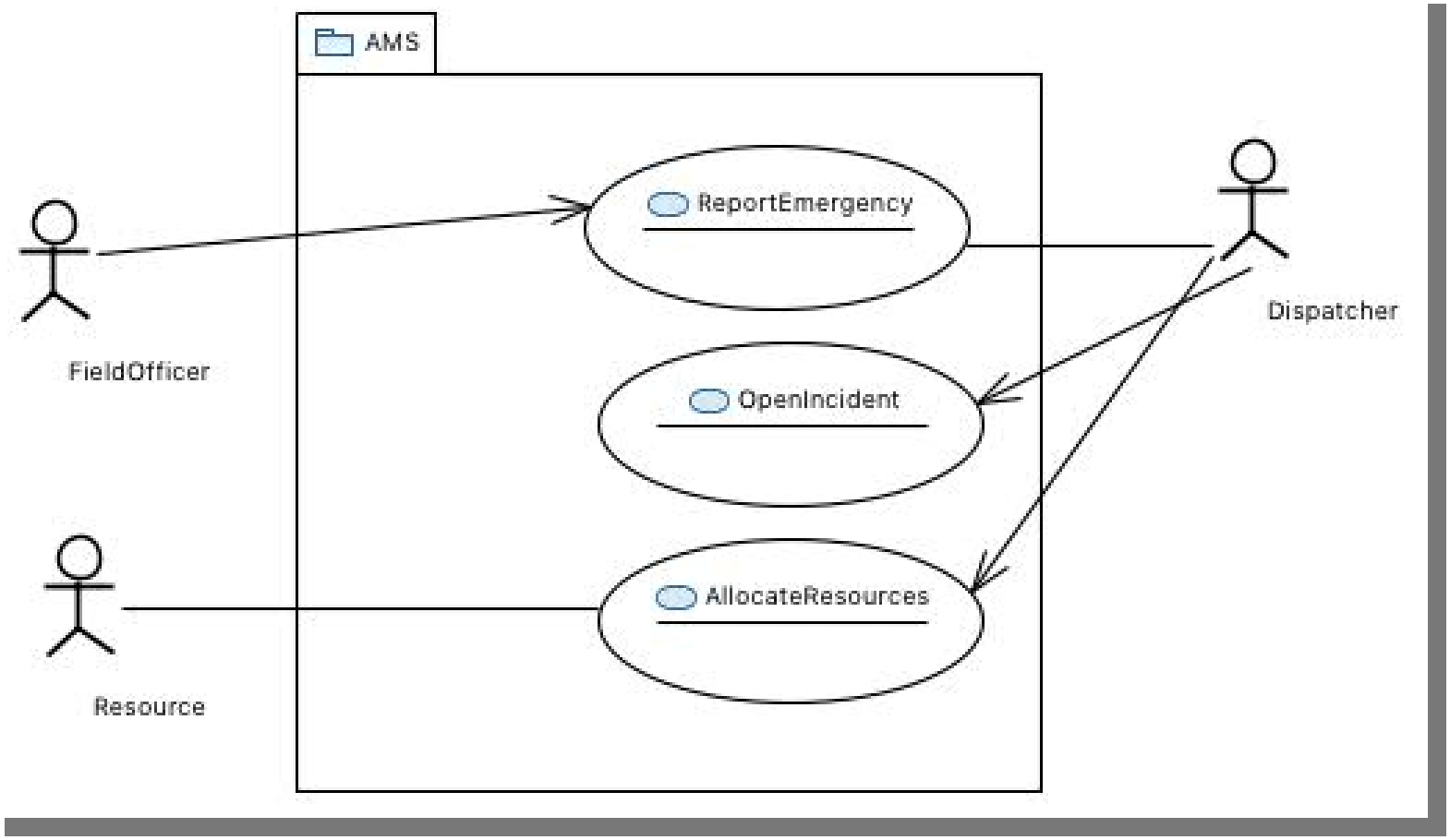
University of Kent

Example: Accident Management System

In term of a use case diagram

- actors
  - `FieldOfficer`
  - `Dispatcher`
- use cases
  - `ReportEmergency`
    - `FieldOfficer` invokes use case `ReportEmergency` to notify `Dispatcher`
  - `OpenIncident`
    - `Dispatcher` invokes `OpenIncident`
  - `AllocateResources`
    - `Dispatcher` invokes `AllocateResources`

Example: Accident Management System



Example: Accident Management System

Use case name	`ReportEmergency`
Participating actors	`FieldOfficer, Dispatcher`
Flow of events: Normal flow	Normal path 1. `FieldOfficer` starts app 2. `FieldOfficer` enters location and description of incident 3. `FieldOfficer` enters services that are needed 4. `AMS` sends request to `Dispatcher` 5. `Dispatcher` sends acknowledgement
Flow of events: Alternative flow	`FieldOfficer` has no connection with `Dispatcher` 1. `FieldOfficer` closes app and finds alternative means
Pre-condition	• `FieldOfficer` notices an emergency
Post-condition	• emergency is logged with the `Dispatcher` • emergency is not logged with the `Dispatcher`

Exercise: Accident Management System

- consider a variant of the previous example in which a database of emergencies and incidents needs to be considered
  - additional actor
    - `Database`

University of Kent

Exercise: Simple Library

Developing a system for a small library to enable a librarian working at the counter to perform the following:

1. Getting the status of a member by scanning the barcode on her library card. The status of a member includes her card number **(cno)**, her personal info such as name, address, phone, fax and email. Her list of borrowed books, dates of borrowing and returned dates. The status also include possible fines at a fixed rate.
2. Getting details of a book by scanning its barcode. Such details includes a unique book number **(bid)**, title, author, ISBN, year of publication. A borrowed book is marked as unavailable and the **cno** of the borrower is included.

University of Kent

Exercise: Simple Library

3. Receiving fine paid by a member and updating her status. The system must interact with an accounting software to record the detail of money paid to the library.
4. Registering book borrowed/returned by members.
5. Filling application form for people who wish to join.

For a new person to join the library her application needs to be marked as accepted by a senior librarian. Under some circumstances the library might decide to wave the fine assigned to a member. A senior librarian is ONLY allowed to void a fine.

University of Kent

Exercise: Simple Library

Only a system administrator, who is also a senior librarian, is allowed to add, cancel or make changes in the list of books, list of member or a librarian details.

Exercise: draw a use case diagram and describe use case steps for each use case.