OO Analysis and Design with UML 2 and UP

Dr. Jim Arlow,

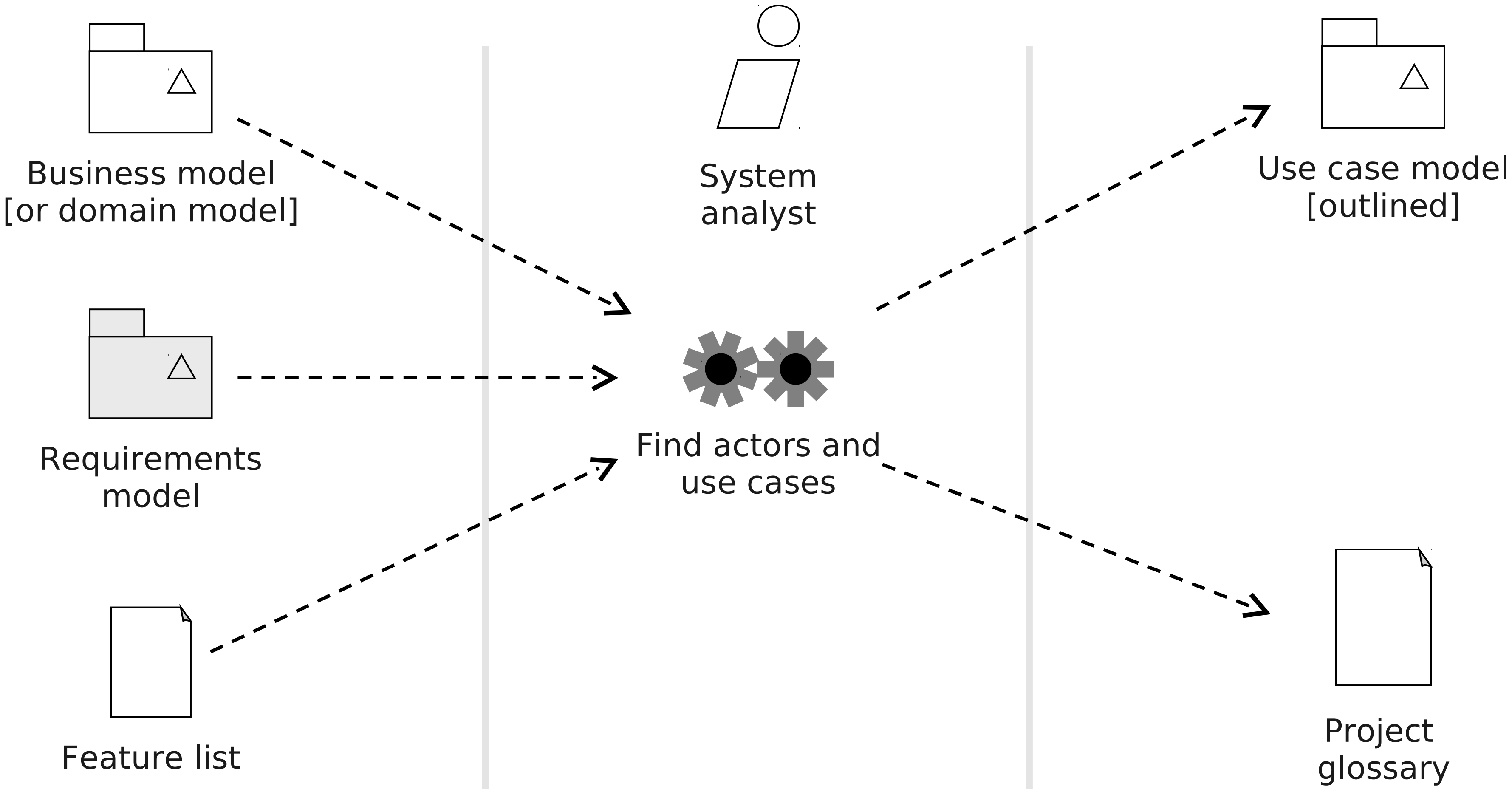
Zuhlke Engineering Limited

Requirements – use case modelling

Use case modelling

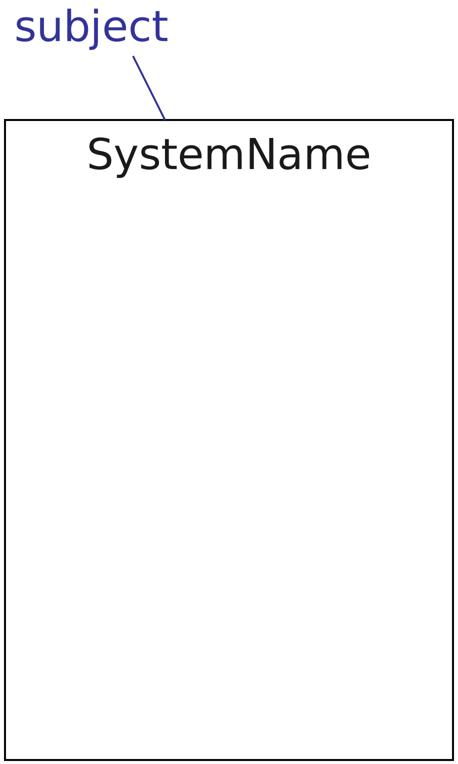
- Use case modelling is a form of requirements engineering
- Use case modelling proceeds as follows:
    - Find the system boundary
    - Find actors
    - Find use cases
        - Use case specification
        - Scenarios
- It lets us identify the system boundary, who or what uses the system, and what functions the system should offer

Find actors and use cases



The subject

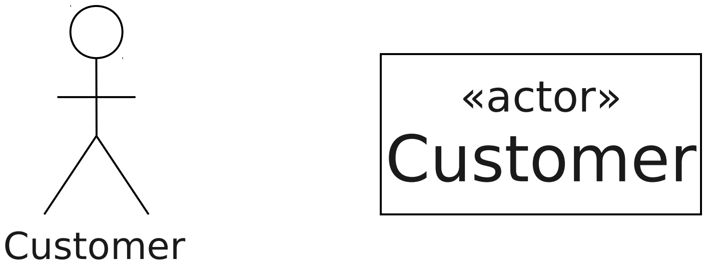
- Before we can build anything, we need to know:
  - Where the boundary of the system lies
  - Who or what uses the system
  - What functions the system should offer to its users
- We create a Use Case model containing:
  - Subject – the edge of the system
    - also known as the system boundary
  - Actors – who or what uses the system
  - Use Cases – things actors do with the system
  - Relationships - between actors and use cases



4.3.2

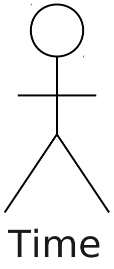
What are actors?

- An actor is anything that interacts *directly* with the system
  - Actors identify who or what uses the system and so indicate where the system boundary lies
- Actors are *external* to the system
- An Actor specifies a *role* that some external entity adopts when interacting with the system



Identifying Actors

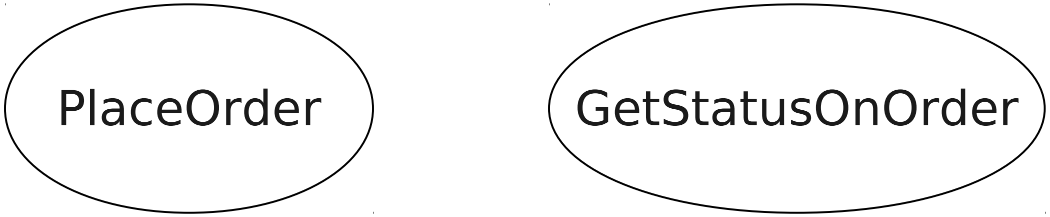
- When identifying actors ask:
  - Who or what uses the system?
  - What roles do they play in the interaction?
  - Who installs the system?
  - Who starts and shuts down the system?
  - Who maintains the system?
  - What other systems use this system?
  - Who gets and provides information to the system?
  - Does anything happen at a fixed time?



4.3.3

What are use cases?

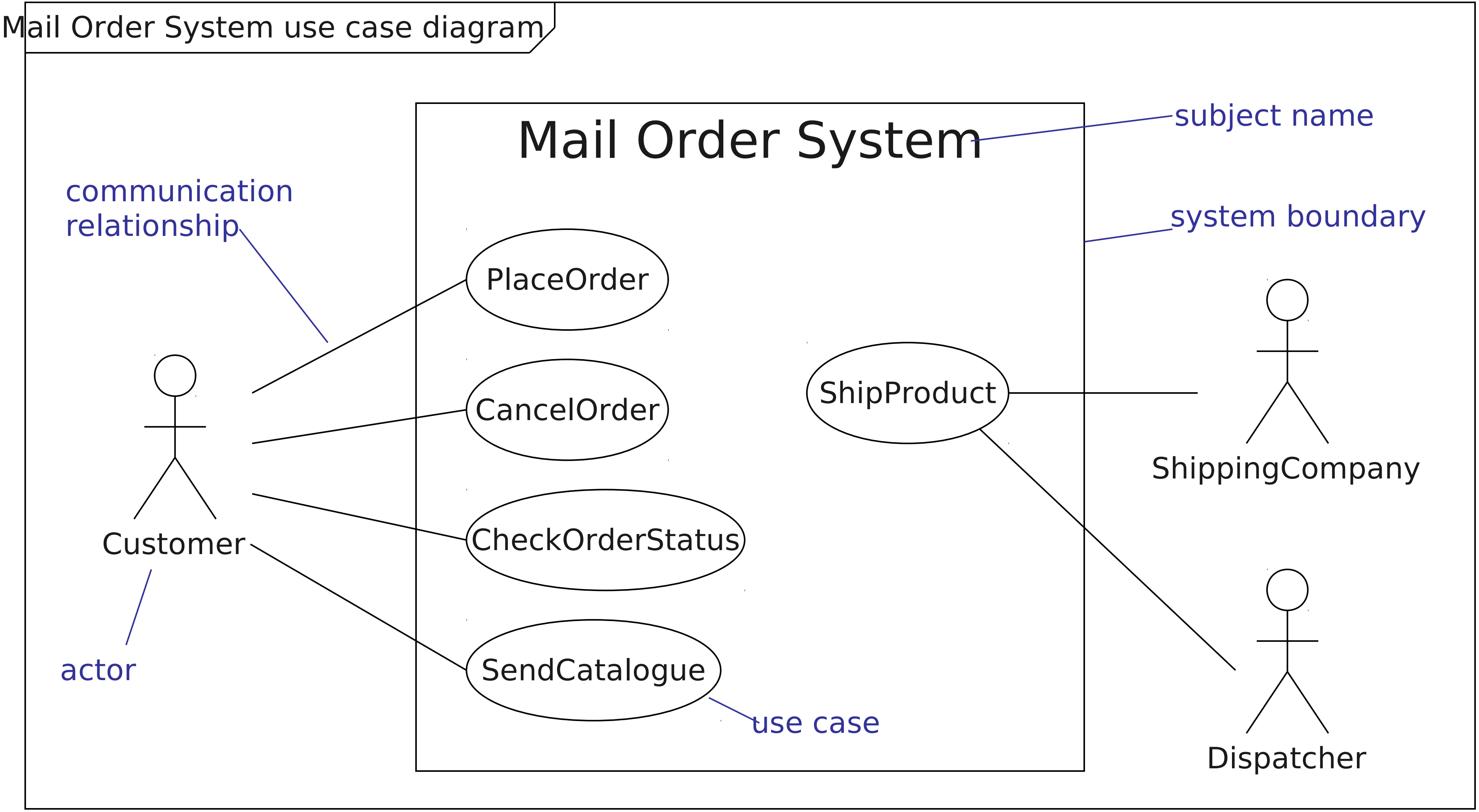
- A use case is something an actor needs the system to do. It is a “case of use” of the system by a specific actor
- Use cases are *always* started by an actor
  - The *primary actor* triggers the use case
  - Zero or more *secondary actors* interact with the use case in some way
- Use cases are *always* written from the point of view of the actors



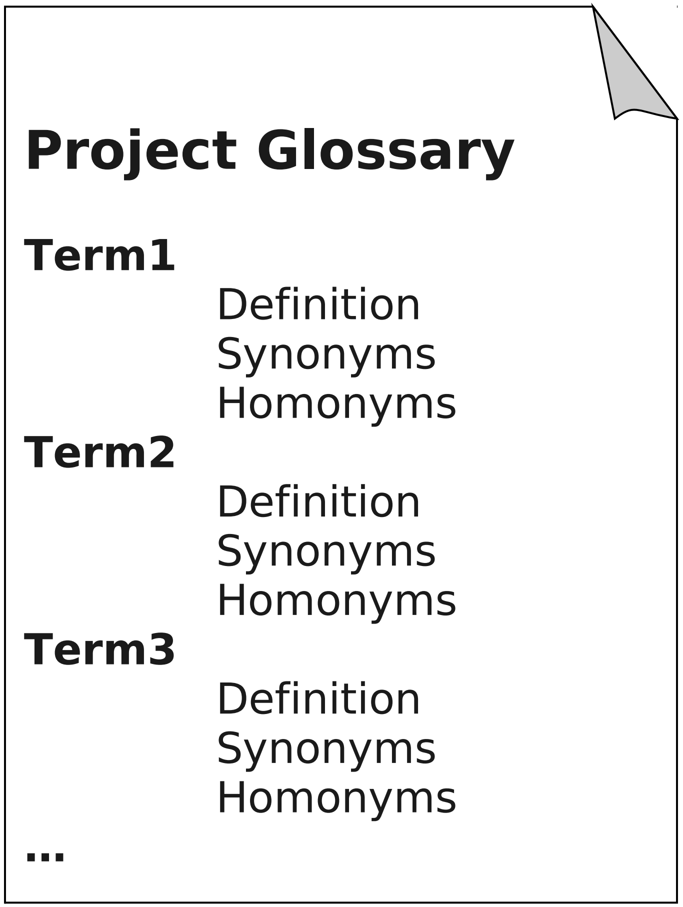
Identifying use cases

- Start with the list of actors that interact with the system
- When identifying use cases ask:
  - What functions will a specific actor want from the system?
  - Does the system store and retrieve information? If so, which actors trigger this behaviour?
  - What happens when the system changes state (e.g. system start and stop)? Are any actors notified?
  - Are there any external events that affect the system? What notifies the system about those events?
  - Does the system interact with any external system?
  - Does the system generate any reports?

The use case diagram



The Project Glossary



Project Glossary

Term1

Definition
Synonyms
Homonyms

Term2

Definition
Synonyms
Homonyms

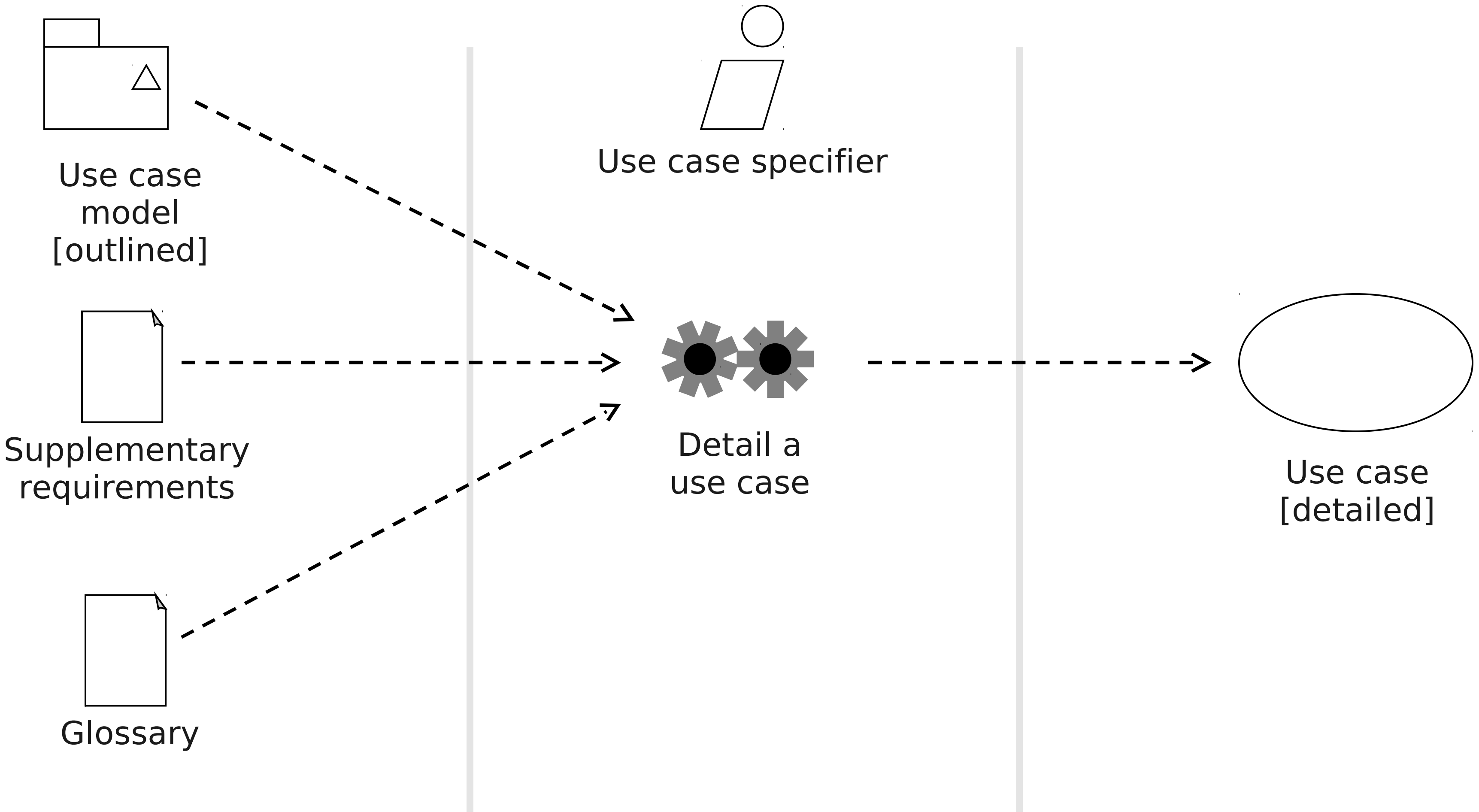
Term3

Definition
Synonyms
Homonyms

…

- In any business domain there is always a certain amount of jargon. It’s important to capture the language of the domain in a project glossary
- The aim of the glossary is to define key terms and to resolve synonyms and homonyms
- You are building a vocabulary that you can use to discuss the system with the stakeholders

Detail a use case



Use case specification

use case name	Use case: PaySalesTax
use case identifier	ID: 1
brief description	Brief description: Pay Sales Tax to the Tax Authority at the end of the business quarter.
the actors involved in the use case	Primary actors: Time
	Secondary actors: TaxAuthority
the system state before the use case can begin	Preconditions: 1. It is the end of the business quarter.
the actual steps of the use case	Main flow: implicit time actor 1. The use case starts when it is the end of the business quarter. 2. The system determines the amount of Sales Tax owed to the Tax Authority. 3. The system sends an electronic payment to the Tax Authority.
the system state when the use case has finished	Postconditions: 1. The Tax Authority receives the correct amount of Sales Tax.
alternative flows	Alternative flows: None.

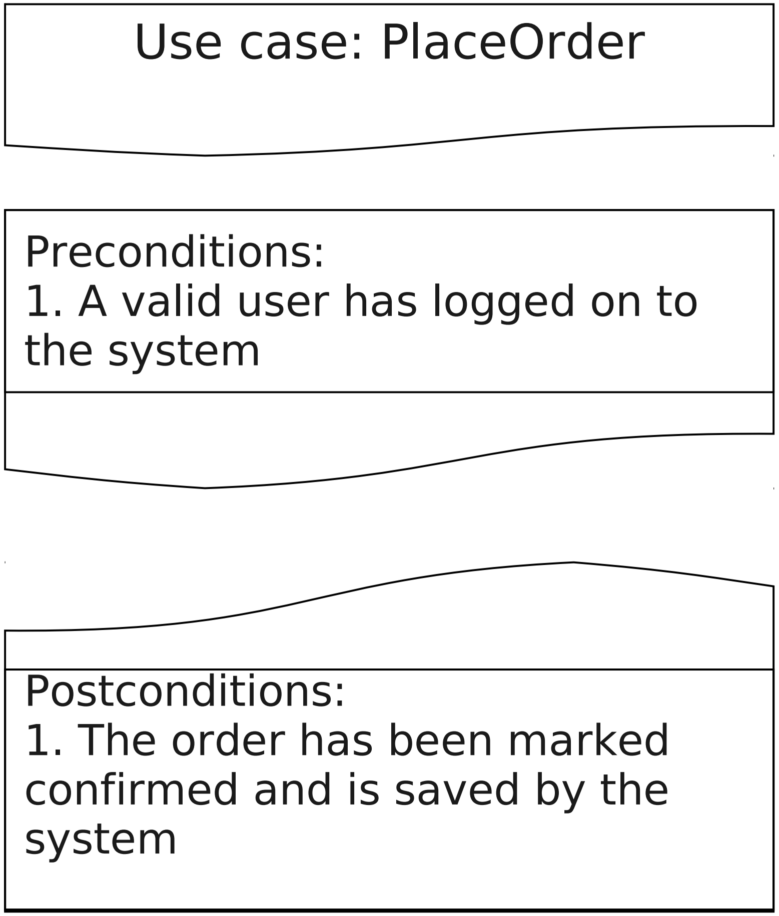
Naming use cases

- Use cases describe something that happens
- They are named using verbs or verb phrases
- Naming standard [1]: use cases are named using UpperCamelCase e.g. PaySalesTax

[1] UML 2 does not specify *any* naming standards.
All naming standards are our own, based on industry best practice.

Pre and postconditions

- Preconditions and postconditions are *constraints*
- Preconditions constrain the state of the system *before* the use case can start
- Postconditions constrain the state of the system *after* the use case has executed
- If there are no preconditions or postconditions write "None" under the heading



Use case: PlaceOrder

Preconditions:
1. A valid user has logged on to the system

Postconditions:
1. The order has been marked confirmed and is saved by the system

Main flow

<number> The <something> <some action>

- The flow of events lists the steps in a use case
- It *always* begins by an actor doing something
  - A good way to start a flow of events is:
    1) The use case starts when an <actor> <function>
- The flow of events should be a sequence of short steps that are:
  - Declarative
  - Numbered,
  - Time ordered
- The main flow is always the *happy day* or *perfect world* scenario
  - Everything goes as expected and desired, and there are no errors, deviations, interrupts, or branches
  - Alternatives can be shown by branching or by listing under Alternative flows (see later)

4.5.6.2

Branching within a flow: If

- Use the keyword if to indicate alternatives within the flow of events
  - There must be a Boolean expression immediately after if
- Use indentation and numbering to indicate the conditional part of the flow
- Use else to indicate what happens if the condition is false (see next slide)

Use case: ManageBasket

ID: 2

Brief description:
The Customer changes the quantity of an item in the basket.

Primary actors:
Customer

Secondary actors:
None.

Preconditions:
1. The shopping basket contents are visible.

Main flow:

1. The use case starts when the Customer selects an item in the basket.
2. If the Customer selects "delete item"
   - 2.1 The system removes the item from the basket.
3. If the Customer types in a new quantity
   - 3.1 The system updates the quantity of the item in the basket.

Postconditions:
None.

Alternative flows:
None.

4.5.6.4

Repetition within a flow: For

- We can use the keyword For to indicate the start of a repetition within the flow of events
- The iteration expression immediately after the For statement indicates the number of repetitions of the indented text beneath the For statement.

Use case: FindProduct
ID: 3
Brief description: The system finds some products based on Customer search criteria and displays them to the Customer.
Actors: Customer
Preconditions: None.
Main flow: 1. The use case starts when the Customer selects "find product". 2. The system asks the Customer for search criteria. 3. The Customer enters the requested criteria. 4. The system searches for products that match the Customer's criteria. 5. For each product found 5.1. The system displays a thumbnail sketch of the product. 5.2. The system displays a summary of the product details. 5.3. The system displays the product price.
Postconditions: None.
Alternative flows: NoProductsFound

4.5.6.5

Repetition within a flow: While

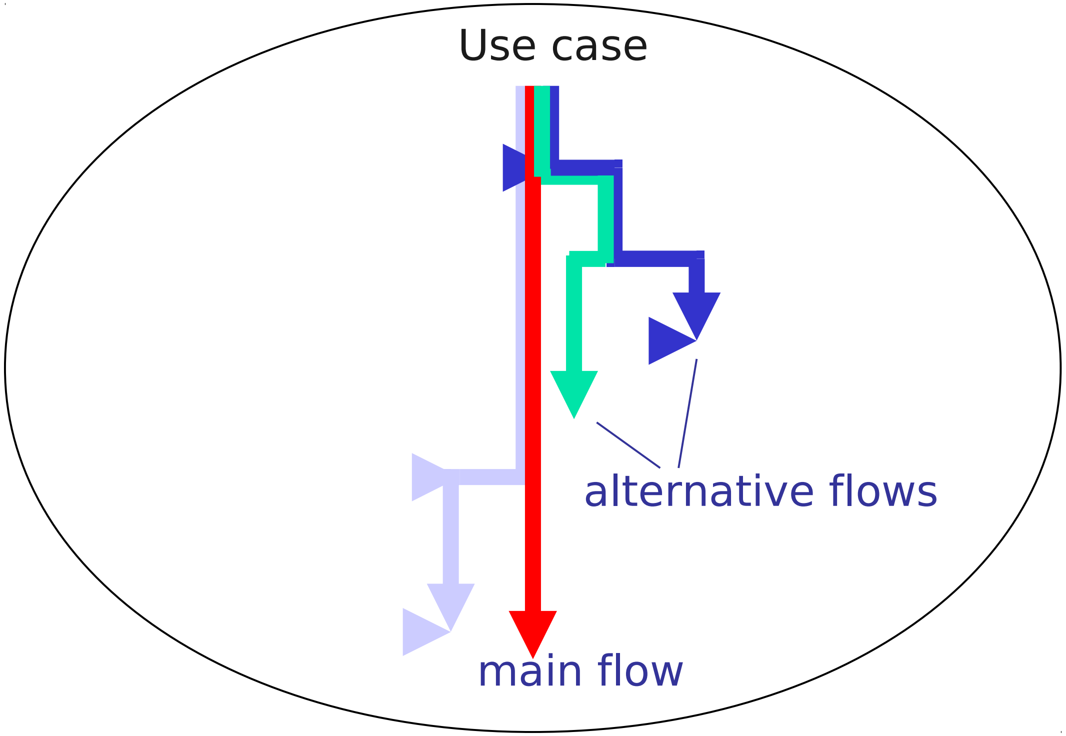
- We can use the keyword while to indicate that something repeats while some Boolean condition is true

Use case: ShowCompanyDetails
ID: 4
Brief description: The system displays the company details to the Customer.
Primary actors: Customer
Secondary actors: None
Preconditions: None.
Main flow: 1. The use case starts when the Customer selects "show company details". 2. The system displays a web page showing the company details. 3. While the Customer is browsing the company details 4. The system searches for products that match the Customer's criteria. 4.1. The system plays some background music. 4.2. The system displays special offers in a banner ad.
Postconditions: 1. The system has displayed the company details. 2. The system has played some background music. 3. The systems has displayed special offers.
Alternative flows: None.

4.5.7

Branching: Alternative flows

- We may specify one or more *alternative flows* through the flow of events:
  - Alternative flows capture errors, branches, and interrupts
  - Alternative flows *never* return to the main flow
- Potentially very many alternative flows! You need to manage this:
  - Pick the most important alternative flows and document those.
  - If there are groups of similar alternative flows - document one member of the group as an exemplar and (if necessary) add notes to this explaining how the others differ from it.



Only document enough alternative flows to clarify the requirements!

4.5.7

Referencing alternative flows

- List the names of the alternative flows at the end of the use case
- Find alternative flows by examining each step in the main flow and looking for:
  - Alternatives
  - Exceptions
  - Interrupts

Use case: CreateNewCustomerAccount

ID: 5

Brief description:
The system creates a new account for the Customer.

Primary actors:
Customer

Secondary actors:
None.

Preconditions:
None.

Main flow:

1. The use case begins when the Customer selects "create new customer account".
2. While the Customer details are invalid
    - 2.1. The system asks the Customer to enter his or her details comprising email address, password and password again for confirmation.
    - 2.2 The system validates the Customer details.
3. The system creates a new account for the Customer.

Postconditions:
1. A new account has been created for the Customer.

Alternative flows:
InvalidEmailAddress
InvalidPassword
Cancel

alternative flows

4.5.7

An alternative flow example

notice how we name and number alternative flows

Alternative flow: CreateNewCustomerAccount:InvalidEmailAddress
ID: 5.1
Brief description: The system informs the Customer that they have entered an invalid email address.
Primary actors: Customer
Secondary actors: None.
Preconditions: 1. The Customer has entered an invalid email address
Alternative flow: 1. The alternative flow begins after step 2.2. of the main flow. 2. The system informs the Customer that he or she entered an invalid email address.
Postconditions: None.

always indicate how the alternative flow begins.
In this case it starts after step 2.2 in the main flow

- The alternative flow may be triggered *instead* of the main flow - started by an actor
- The alternative flow may be triggered *after a particular step* in the main flow - after
- The alternative flow may be triggered *at any time* during the main flow - at any time

Requirements tracing

- Given that we can capture functional requirements in a requirements model *and* in a use case model we need some way of relating the two
- There is a many-to-many relationship between requirements and use cases:
  - One use case covers many individual functional requirements
  - One functional requirement may be realised by many use cases
- Hopefully we have CASE support for requirements tracing:
  - With UML tagged values, we can assign numbered requirements to use cases
  - We can capture use case names in our Requirements Database
- If there is no CASE support, we can create a Requirements Traceability matrix

		Use cases			
		U1	U2	U3	U4
Requirements	R1	🟥			
	R2		🟥	🟥	
	R3			🟥	
	R4				🟥
	R5	🟥			

Requirements Traceability Matrix

When to use use case analysis

- Use cases describe system behaviour from the point of view of one or more actors. They are the *best* choice when:
  - The system is dominated by functional requirements
  - The system has many types of user to which it delivers different functionality
  - The system has many interfaces
- Use cases are designed to capture *functional* requirements. They are a *poor* choice when:
  - The system is dominated by non-functional requirements
  - The system has few users
  - The system has few interfaces

Summary

- We have seen how to capture functional requirements with use cases
- We have looked at:
  - Use cases
  - Actors
  - Branching with if
  - Repetition with for and while
  - Alternative flows
  - Requirements tracing