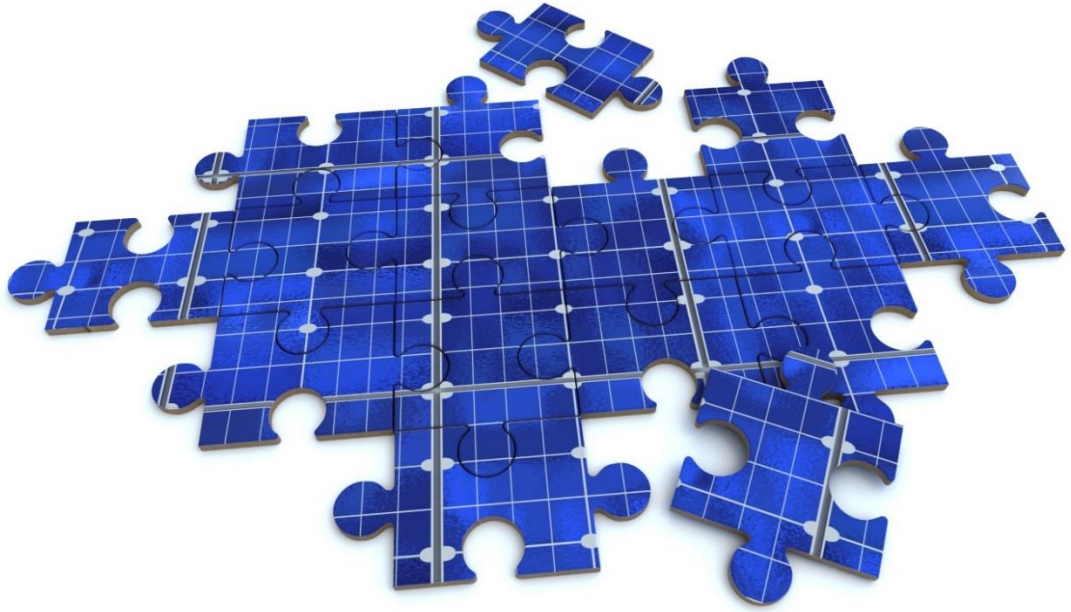
Week 9 Lecture:
Unified Modelling Language (UML): Use Cases

Dr Alireza Nili
IFB103



QUT Science and Engineering

Housekeeping

Last week we started working on the back-end of the system. We provided an overview of the two main related activities, including requirements determination and visualisation of the results via FMC.

We mentioned that drawing a requirements table is a method of determining and documenting functional requirements, and FMC and UML diagrams are two methods of visualising the analysis and design results.

Last week we learnt modelling FMC diagrams. This week we learn the first type of UML modelling technique. UML will be our focus for the next few weeks.

The class will be about visualisation of the results via UML use case diagrams.

CRICOS 00213J

Week 9 assessment structure

Before your week 9 online session:

1. Ensure you have watched Week 9 lecture recording.
2. Have another look at the assessment specification file related to Design Challenge 2.

During week 9:

1. Read the Week 9 tutorial slides. Think on items 3a and 3b in the assessment specification file. Draw the first draft of your use case diagram (3a) and answer item 3b **[Assessed items]**
2. Show the first draft of your diagram to your tutor during your tutorial session. If your tutorial is online, you need to use the screenshare feature. Your tutor will give you feedback on your work.

CRICOS 00213J

This is important, so I repeat:

Every week:

1. Read/review the assessment specification of your final assessment (Design Challenge 2).
2. Watch the lecture recording.
3. Read this set of slides and other teaching resources each week ‘before’ attending your session with your tutor.
4. You are expected to draw the first draft of your diagrams ‘before’ attending your session and show it to your tutor during the session. This is particularly important for online tutorials.
5. Your tutor will answer your questions and provides feedback for improving your diagram.

The above schedule gives you an extended time for reading the teaching material and an opportunity for a personalised consultation for receiving feedback on your work during the tutorial session.

Goals for today

UML: Introduction

Use Cases and UML Use Case Diagram

UML: Introduction

Use Cases and UML Use Case Diagram

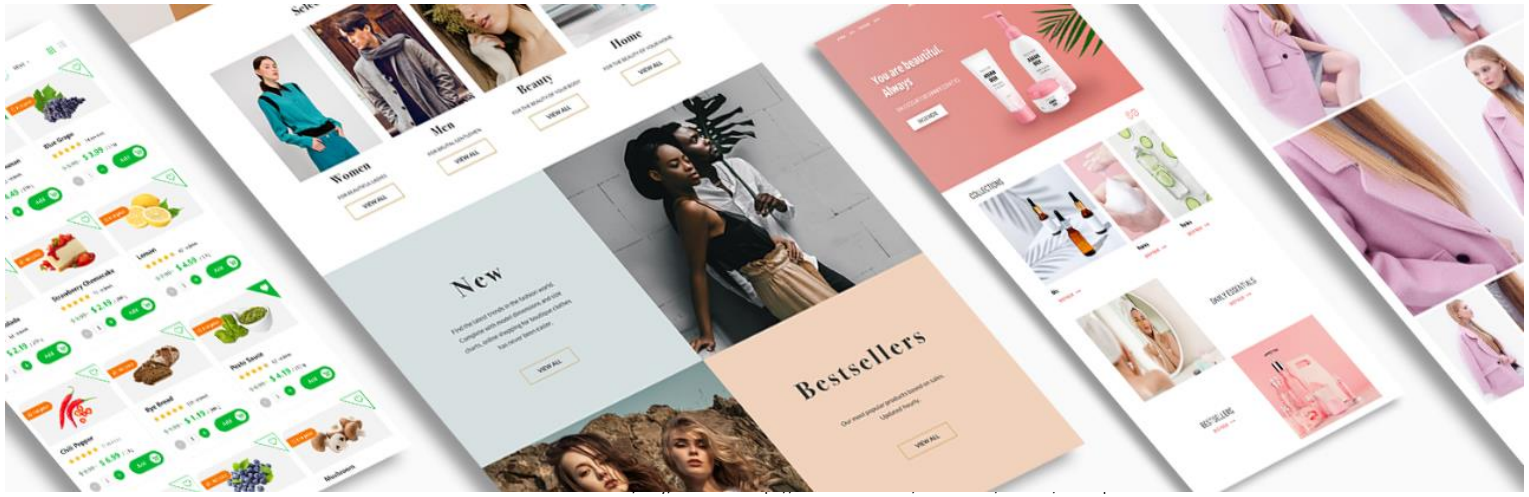
Wrap up

UML: A Short Introduction

Imagine you want to create a website (e.g. music web store) or a mobile app (e.g. an online game)

- How do you express what you want?
- How are you going to communicate with a software developer?

UML is an effective way.



Unified Modelling Language (UML)

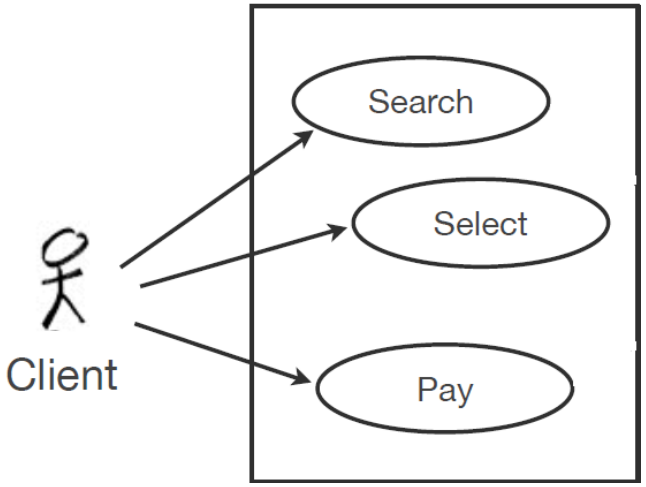
- A set of modelling conventions that is used to specify or describe a system.
- UML does not prescribe a method for building systems, only a notation that is widely accepted as a standard for modelling.
- UML supports different **views**, with various **concerns.**
- UML may be used to model almost anything.

Case — Music web store

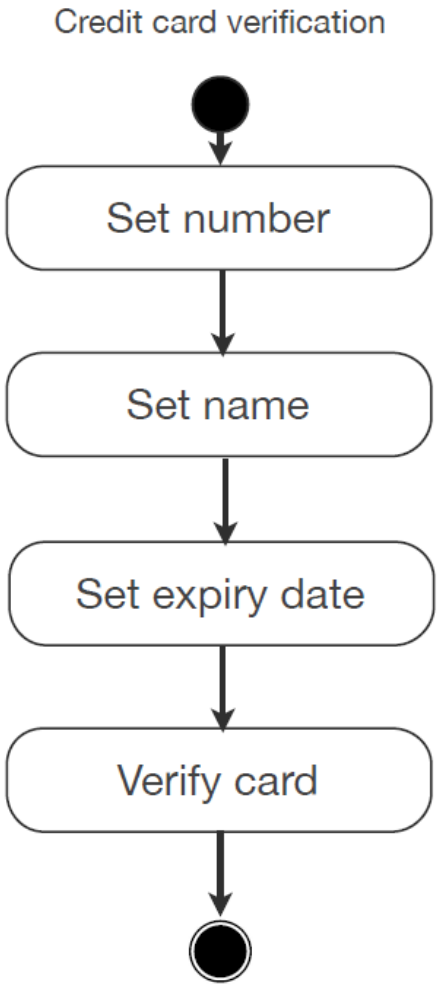
– Multiple views

- External - E.g. users
- Internal - E.g. how the system interacts with the users
- Functional - E.g. what the users do with the system
- Structural - E.g. what has to be implemented

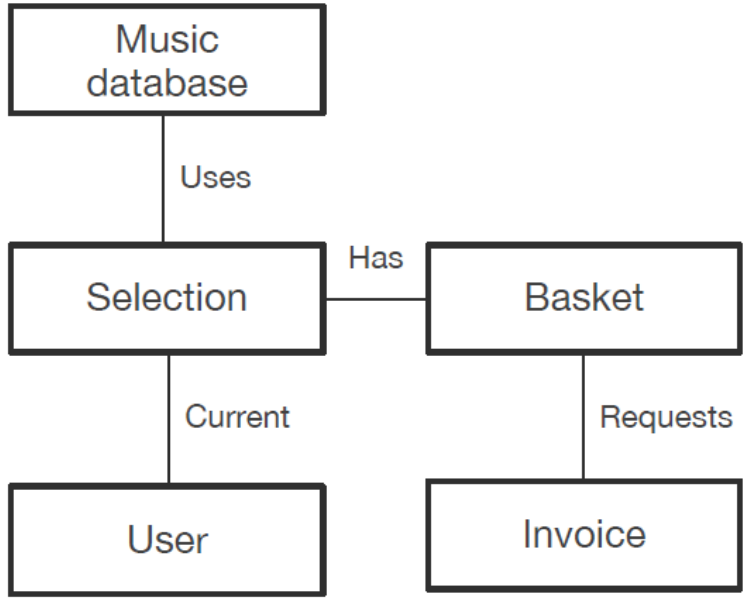
Example:
music web store: simple diagrams



A **use case** describes the users' goals



An **activity diagram** describes how the system implements the users' goals



A **class diagram** describes the software "things" that must be implemented by the system to support the users' goals

Where to start?

1) You will need to draw your requirements matrix (previous lecture)

2) UML which is used for drawing **Use Case** diagrams and several other types of diagrams including **Class** diagram, **Activity** diagrams, and **Sequence** diagrams.

UML: Introduction

Use Cases and UML Use Case Diagram

Wrap up

UML: Use Case

A UC illustrates the activities that are performed by certain types of users of a system.

- In this context, users may be human or not (i.e. other systems)

Each UC describes a functional requirement

- A goal that the user would like to achieve
- How the system will support users achieving that goal

UC are primary building blocks for continued design activities

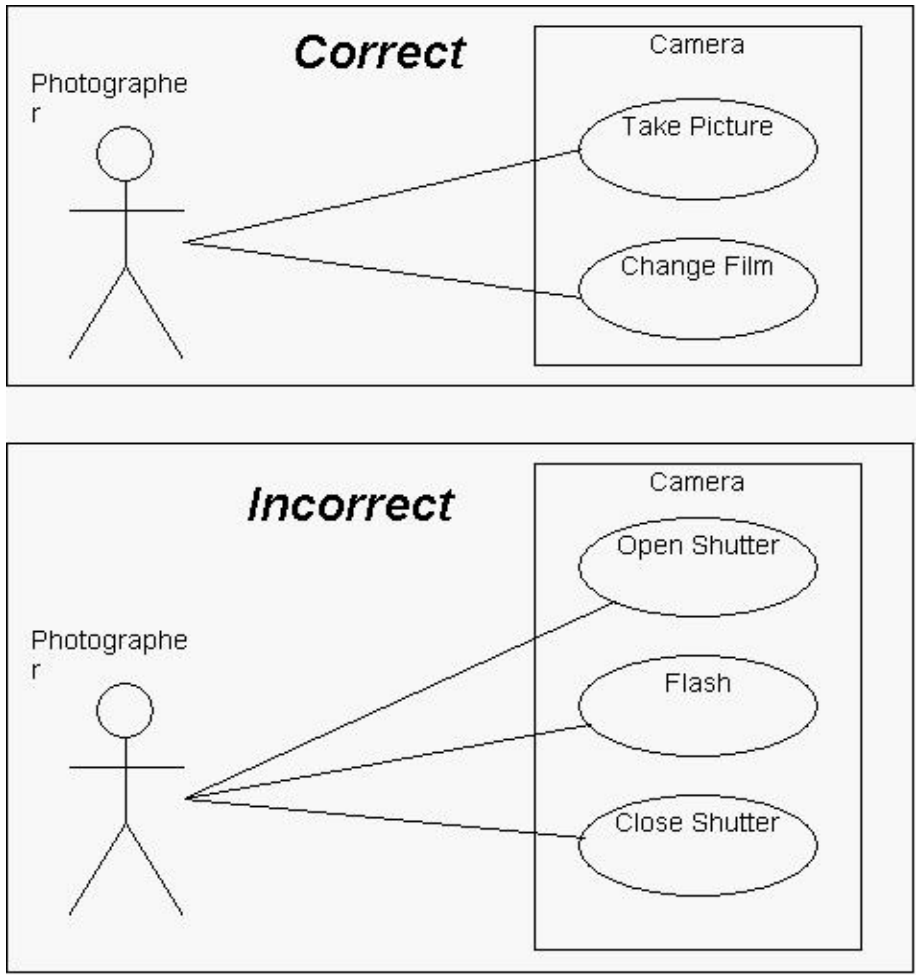
- Initial work centred on expressing the functional requirements
- Main reason why we say that system design is **user-centred**
- Entry points linking to other diagrams
- Also an entry points for software validation and verification

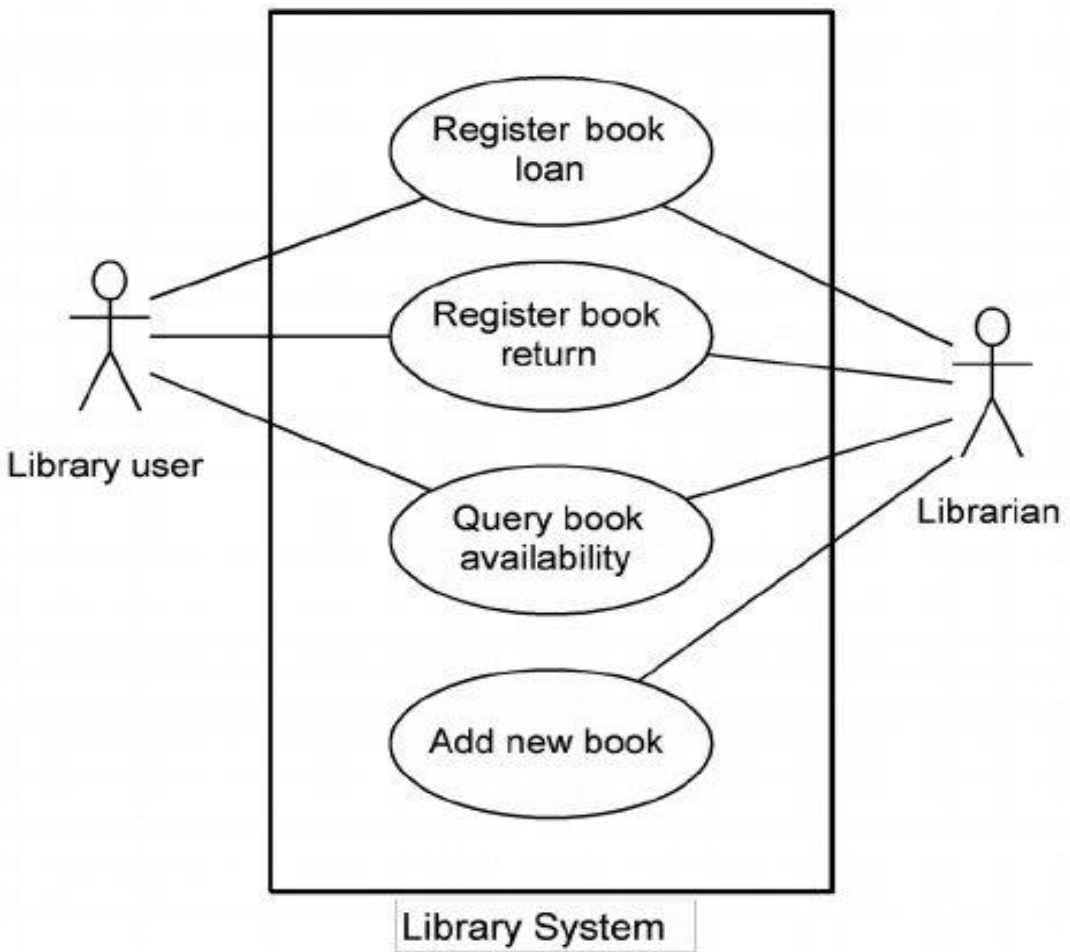
- Use case diagrams are usually produced at the early stages of a project and can be referred to throughout the development process (e.g. for planning and as criteria for testing).

- A complex system can be modeled using with a single use case diagram, but you also have the choice of creating several use-case diagrams to model the components of that system.

- A use case diagram indicates what the system should do, but it does not show how it operates internally.

A use case should say what the system does for the user (or what major things users can do with the system). See the correct example on the left and the correct example on the right.



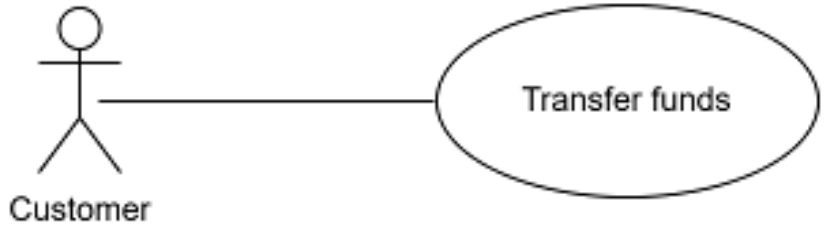


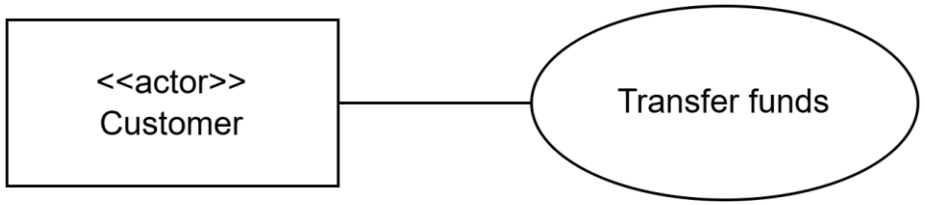
https://www.researchgate.net/publication/279404900_Metrics_for_use_cases_A_survey_of_current_proposals/figures?lo=1

Actor: is a person or system that interacts with the system and is external to the subject of analysis

Use case: represents a major piece of system functionality

Always try to identify the primary actors and their goals.





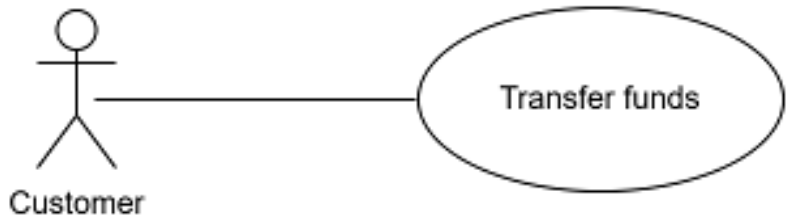
These two diagrams say the same thing, although the one on top is more readable and preferable when actors are **human**

Types of relationships between use cases:

- **association relationship**
- **<<includes>> relationships**
- **<<extends>> relationships**
- **generalisation relationship**

1) Association relationship:

A simple line that represents an interaction or communication between an actor of the system and a use case (please see the example below and the previous slides). It only indicates that the actor is involved in the functionality that the use case represents.



2) <<includes>> relationship:

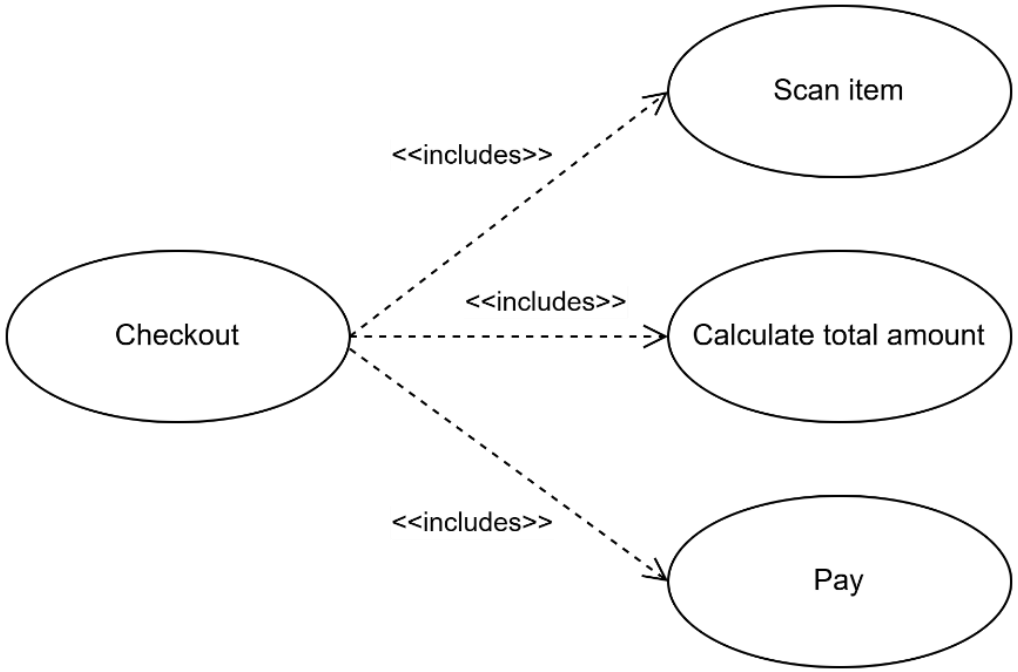
Shows that a use case has to include the behavior of another use case as part of its execution.

The original use case (the base use case) is not complete without the included use case (i.e. the base use case is ALWAYS dependent on the included use case or use cases).

Example: for the 'place an order' base use case, a use case for verifying the payment method needs to be included because verifying the payment method is mandatory. Another example is: to enable users to order a meal online (a base use case) you need to include the 'choose a menu' use case and 'select a menu item' use case.

Pay attention to the direction of <<includes>> relationship (from the base use case)

2nd example: Why the <<includes>> relationship?



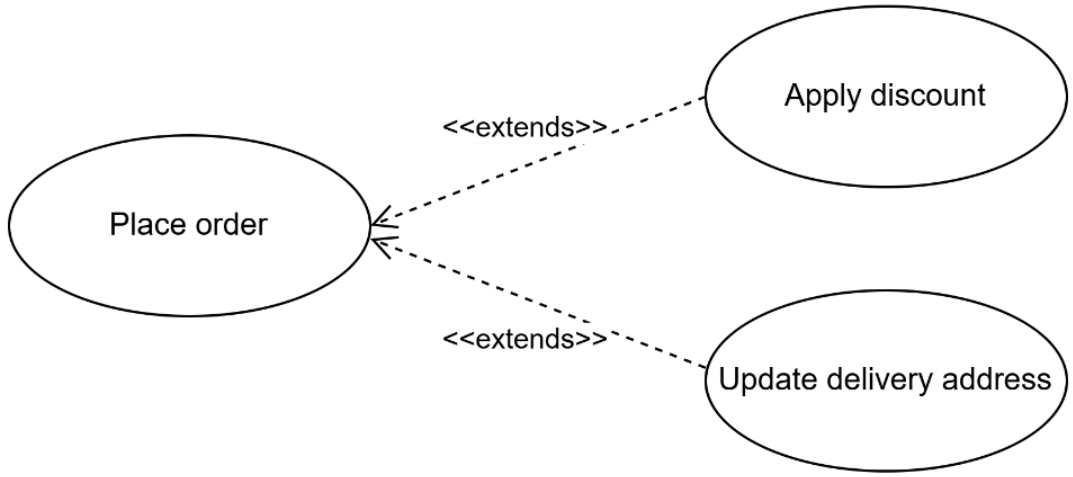
3) <<extends>> relationship:

Is used for optional and/or conditional behaviors (i.e. optional and/or when certain criteria are met) which are not necessary to the objective of a use case. In this case, the base use case is a completely functional use case without the need for any other use case. The extension use case is executed only SOMETIMES.

Example: the 'book a flight' base use case can be extended with the 'choose a seat' use case. This is because choosing a seat is optional and depends on the availability of at least one seat and the user's preference. As another example, handling a password error can be an extension for the login error. We know that an error happens sometimes and when a condition such as entering a wrong password or forgetting the password happens.

Pay attention to the direction of <<extends>> relationship (towards the base use case).

2nd example: Why the <<extends>> relationship?

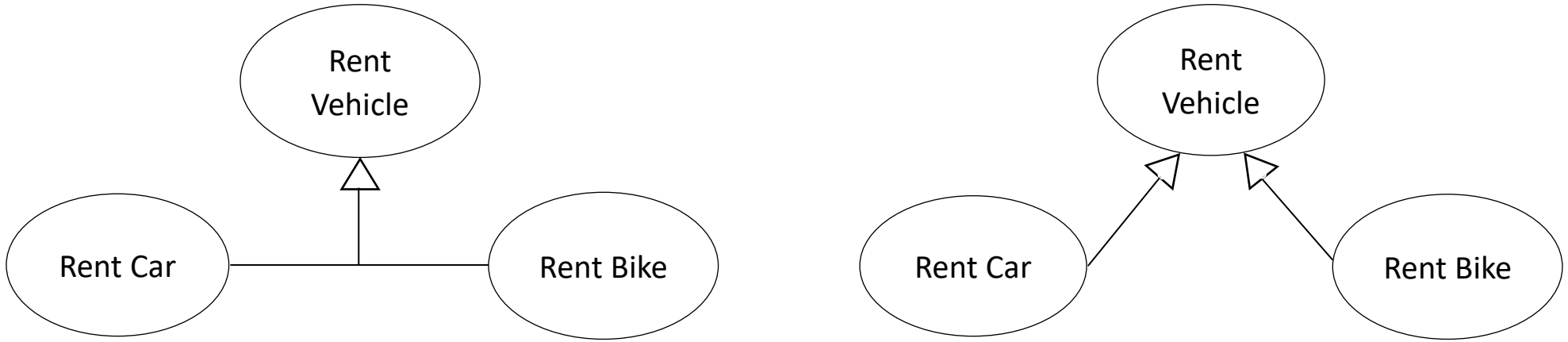


4) Generalisation relationship (also called a parent-child relationship):

Indicates that one use case (specialised use case, also called child use case) is a specialised version of a general use case (also called parent use case).

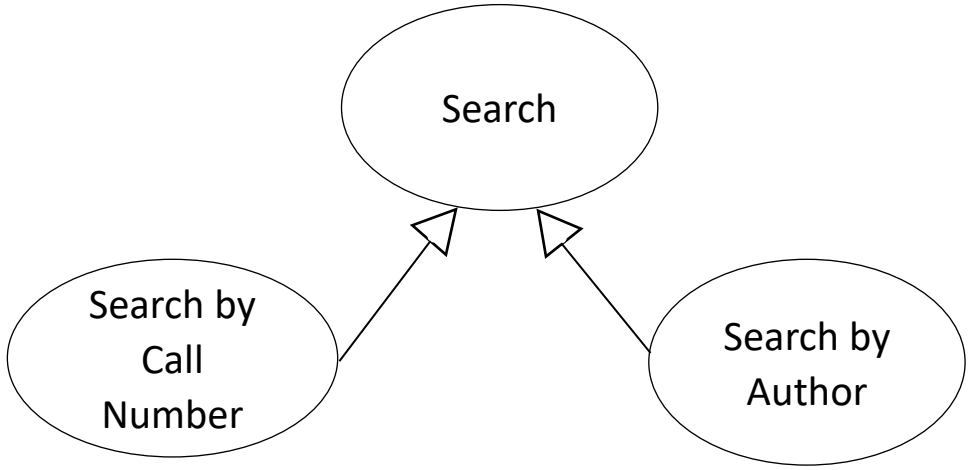
In a generalisation relationship, child use cases inherit the properties of the parent use cases.

Example: the figures below show that each of the Rent Car (a child use case) and Rent Bike (a child use case) is a specialise version of Rent Vehicle (the general use case).

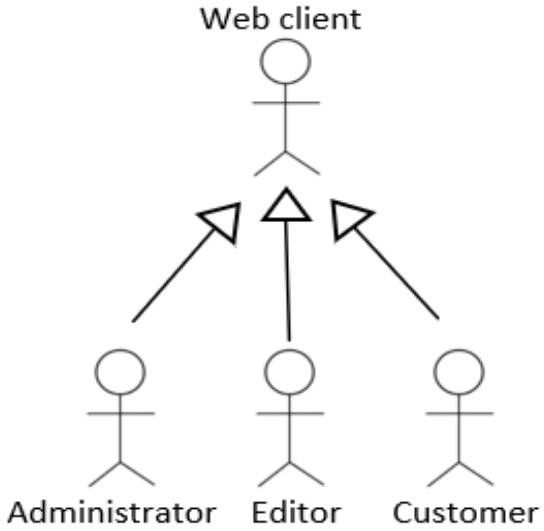


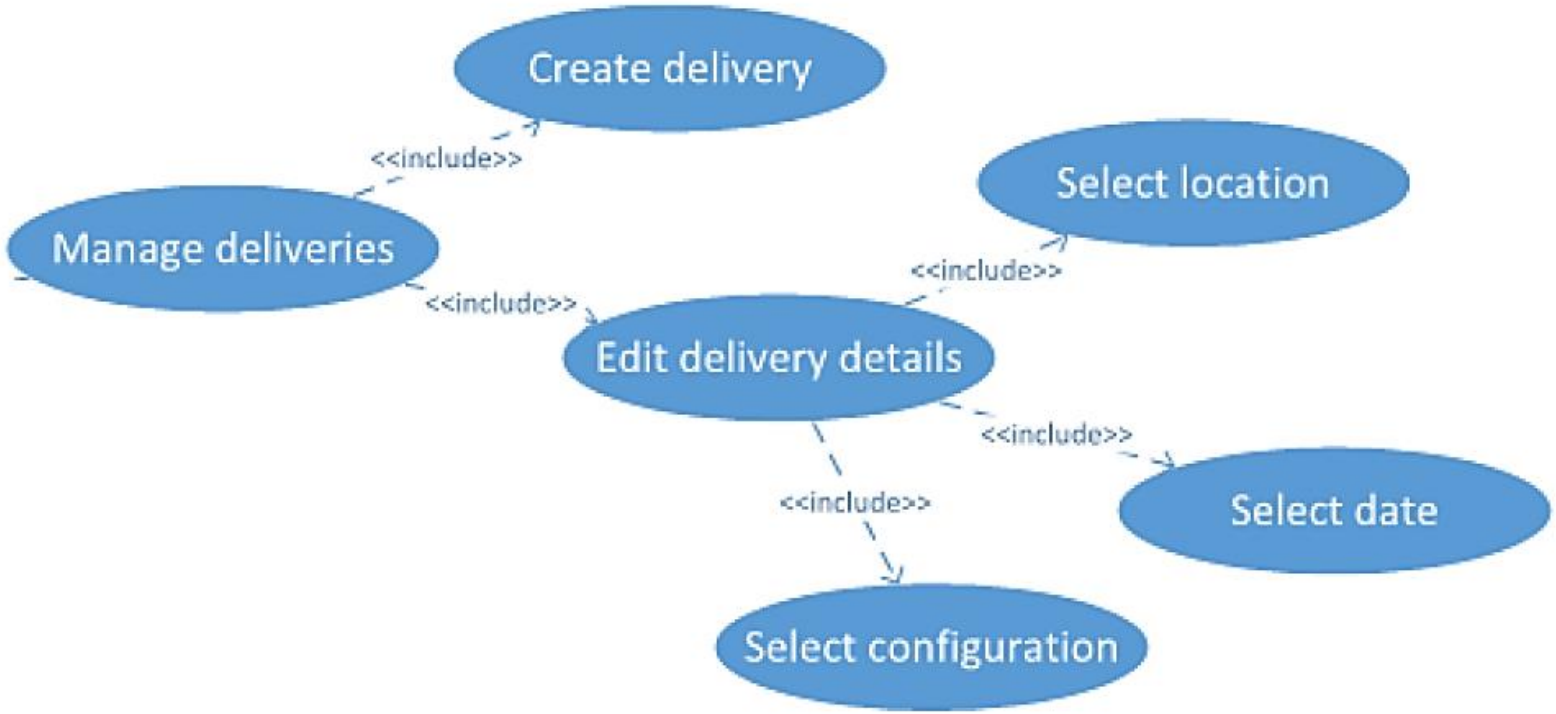
Two ways of showing the generalisation relationship

2nd example:



Moreover, generalisation can be between the actors:



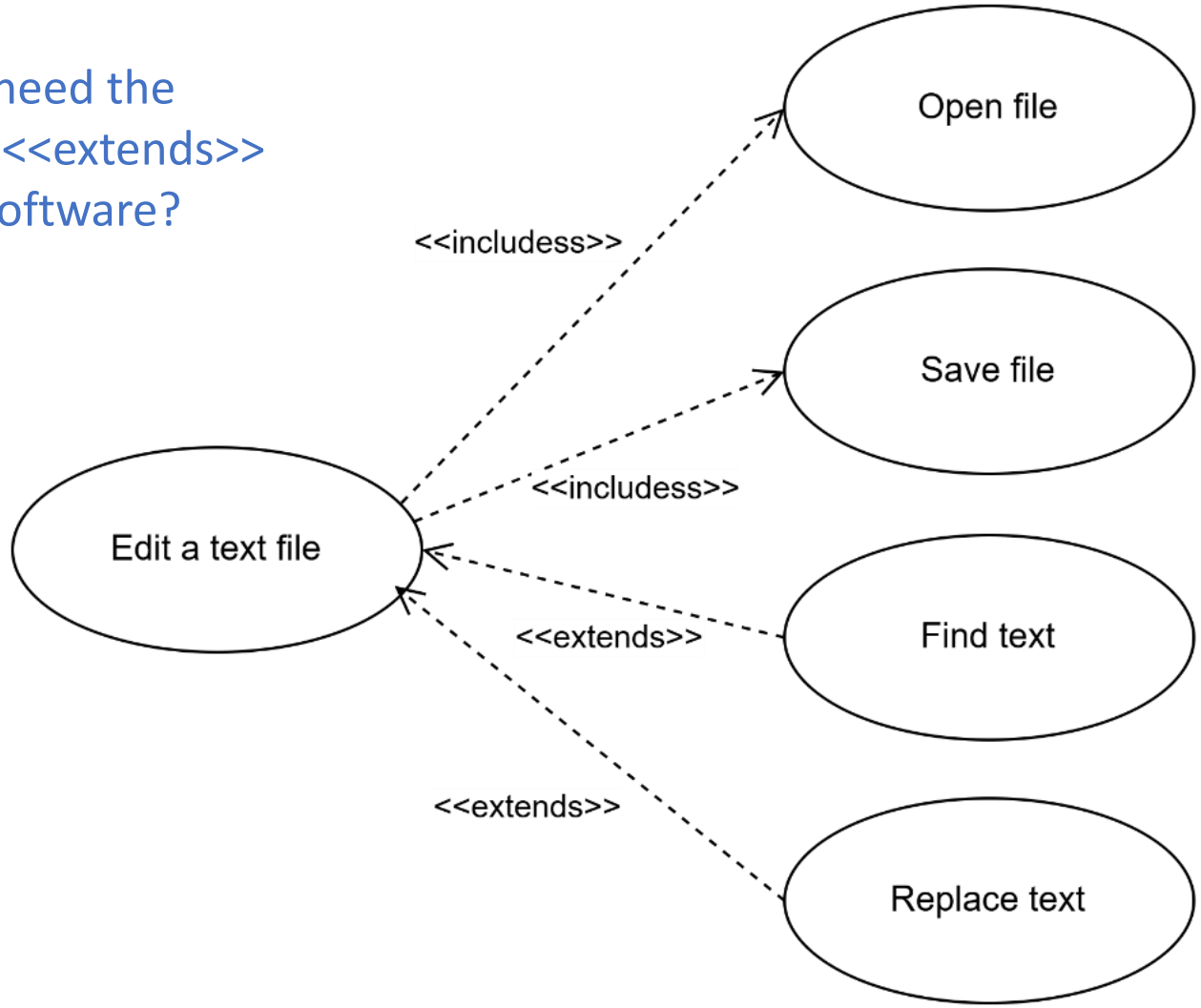


An example of detailed use case diagram for the “manage deliveries” requirement.

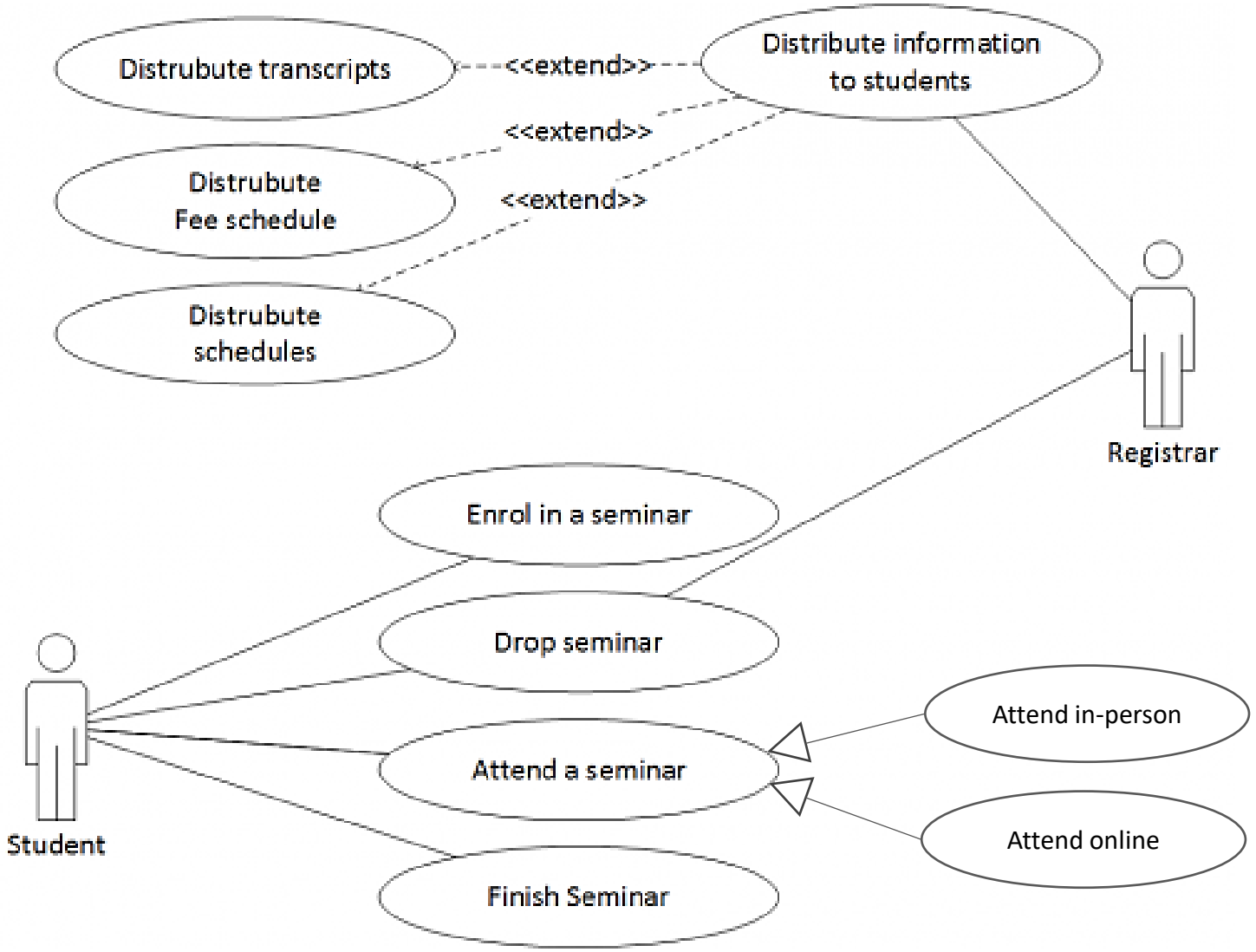
In this example, all relationships are <<include>>. Why?

This requirement is different from the requirements in your assessment. Also, your use case diagram may include both <<include>> and <<extend>>.

Example: Why do we need the <<includes>> and the <<extends>> relationships for the software?



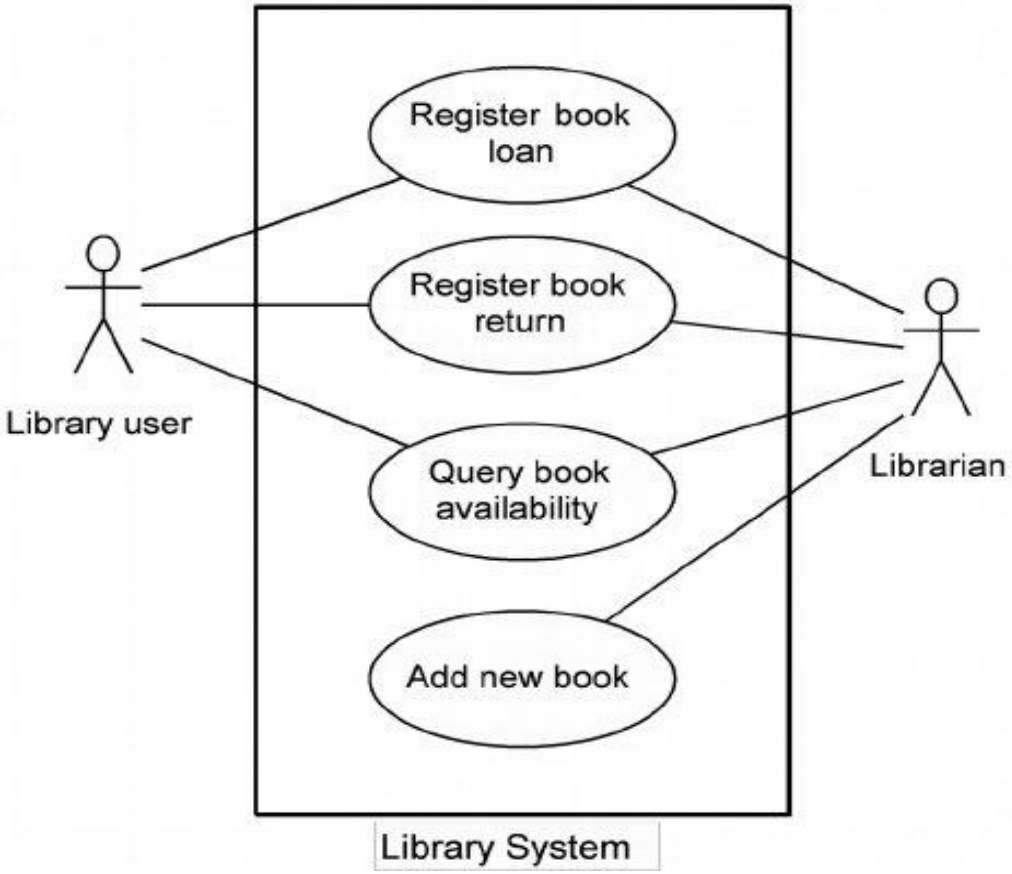
There is an error in this diagram. Can you find it?



Answer: this is just an example (it is not supposed to be a complete solution), but there is an error about the direction of the <<extends>> relationships.

Figure adapted from: https://uwaterloo.ca/ist-project-management-office/tools-and-templates/tools/use-cases

Subject boundary



The subject boundary highlights what is exactly modelled by the system.

It is particularly useful for complex systems.

Important: Your case study is an example of a relatively simple system (not a complex system that requires many use cases). You don't need to use/draw any subject boundary.

https://www.researchgate.net/publication/279404900_Metrics_for_use_cases_A_survey_of_current_proposals

Recommendations

- Identify the primary actors and their goals.
- Use Association relationships (not any other relationship) for general (high-level) diagrams.
- For detailed diagrams use <<include>>, <<extend>> and/or generalisation relationships.
- Avoid complex diagrams and too much detail.
- **Always start the name/label of each use case with a verb…**

Software (also, please see the assessment specification document: DC2)

Use Lucidchart, Draw.io, Signavio or another appropriate tool to draw your use case diagram. For use case diagrams we usually suggest Lucidchart or Draw.io

If you use Lucidchart: search “Lucidchart” via Google. Register if you don’t have an account. Select/find the UML section. You need to drag and drop relevant shapes and draw the relationships between them.

If you decide to use Draw.io, search it via Google or visit https://www.draw.io/ You may need to search for <<include>> and <<extend>> relationships and other shapes in the search box on the left side of the main page. You will need to drag and drop relevant shapes and draw the relationships between them.

What is a “Use Case Specification” document and why is it useful?

Do I need to create a Use Case Specification document for my assignment?

The guidelines for answering items 3a and 3b are available in the Assessment Specification file. Please read the guidelines carefully.

UML: Introduction

Use Cases and UML Use Case Diagram

Wrap up

Drawing it **all** together

After the “requirements determination step” (through drawing a requirements matrix) we need to visualise and communicate our design ideas.

FMC diagrams (week 8) focused on the structure of a system. FMC diagrams provide a high-level view of the structure. They do not aim to provide “detailed” information on what happens within a process.

There are several types of UML diagrams. Each of them has a specific purpose. Therefore, they all together provide “detailed” information on what happens within a process.

Acknowledgment

The examples on slides 10, 23 and 24 are based on the content taught at Victoria University of Wellington as a part of systems analysis and design course in 2013-14. Associate Professor Pedro Antunes (coordinator) and Dr Alireza Nili were the teachers of the course.

Questions?

a.nili@qut.edu.au