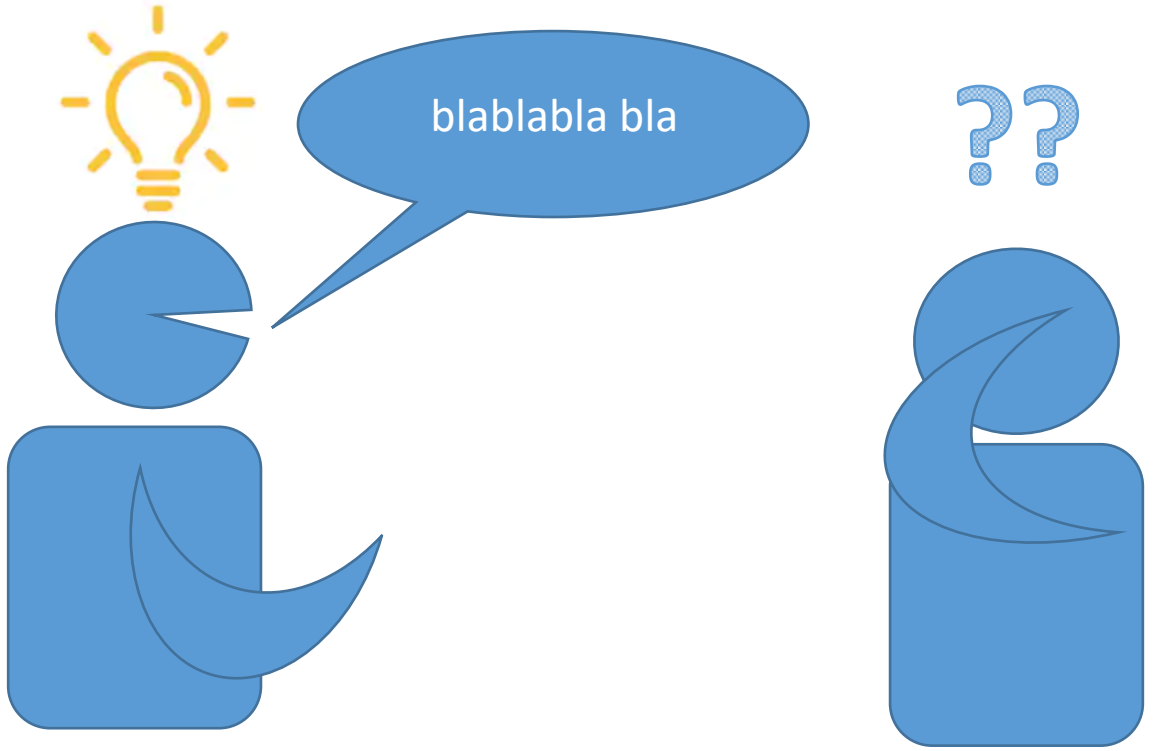
- In the previous set of slides
  - Formal specification of a software system
    - Unified Modelling Language, UML
    - tools
      - actors, systems, components
      - diagrams: component, use cases, sequence/collaboration, classes
- In this set of slides:
  - More details on use cases
    - Formally describing each UML use case
    - Online tools: Lucid Chart and Creately

What is a UML use case?

- It offers both graphical and descriptive representations of how actors (users of the system) will perform the tasks interacting with the system
- its description also indicates how the system will react to the actors' actions (the system behaviour)
  - for each task, (at least) one use case is specified as the sequence of actions and system's responses, starting with the goal that the actor targets when initiating the use case and finalising when that goal is accomplished
  - normally for each user requirement there will be a use case
- Benefits of elaborating and describing in a formal way the use cases
  - explain in an universal language how it is expected that the systems behaves to achieve the identified goals and support the requirements
  - and encourage discussion to refine or improve this behaviour as well as the objectives/requirements themselves

Use case diagrams – the motivation

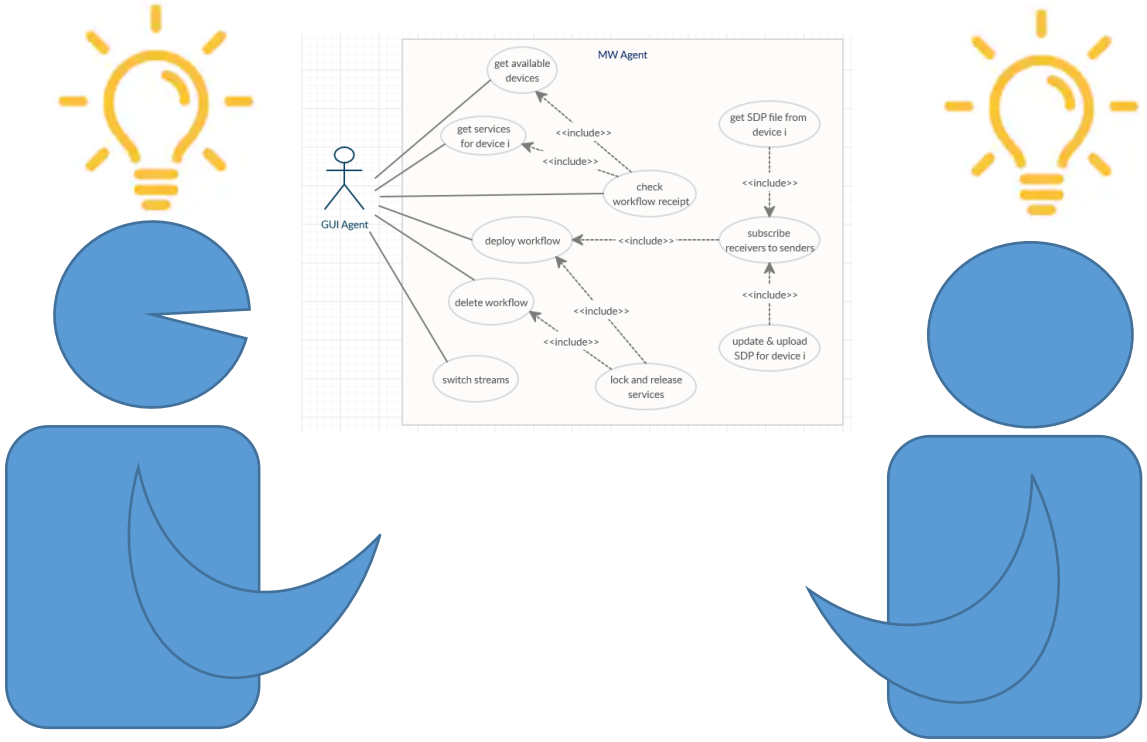
- To enable describing in a universal manner what the system allows to do



- and serve as a common and basis for starting the software development

Use case diagrams – the motivation (2)

- To enable describing in a universal manner what the system allows to do



- and serve as a common and basis for starting the software development

UML Use case diagrams

- Constitute a formal graphical method of describing funcional requirements
  - indicating the functionality that the system will offer to support the requirements and who or what will make use of that functionality or value
  - or, the tasks that will be executed to support the requirements and how it will be executed
    - for example, if one requirement is "as a user, I want to be able to register my expenses while traveling", then there will be a corresponding use case that will comprise a number of tasks necessary for the user to insert expenses for a certain travel
  - typically, the names of the use cases are derived from the user requirements

From requirements to use cases

What does the user want to do?

- As a user, I want to be able to register my expenses while traveling
- As a user, I want to be able to create folders to separate my travels and related data
- As a user, I want to be able to visualise information concerning one specific travel
- …

The user expectation

What will the system do/offer to fulfil the user's wishes/needs?

- Sequence of actions - use cases - that describe the interaction between the user and the system and between the components of the system, through which the user obtains what he/she wants
  - Register new spending
  - Create a folder "new trip"
  - Select trip
  - …

The use case ("what can the user do with the system?)

Relationship between user requirements and use cases

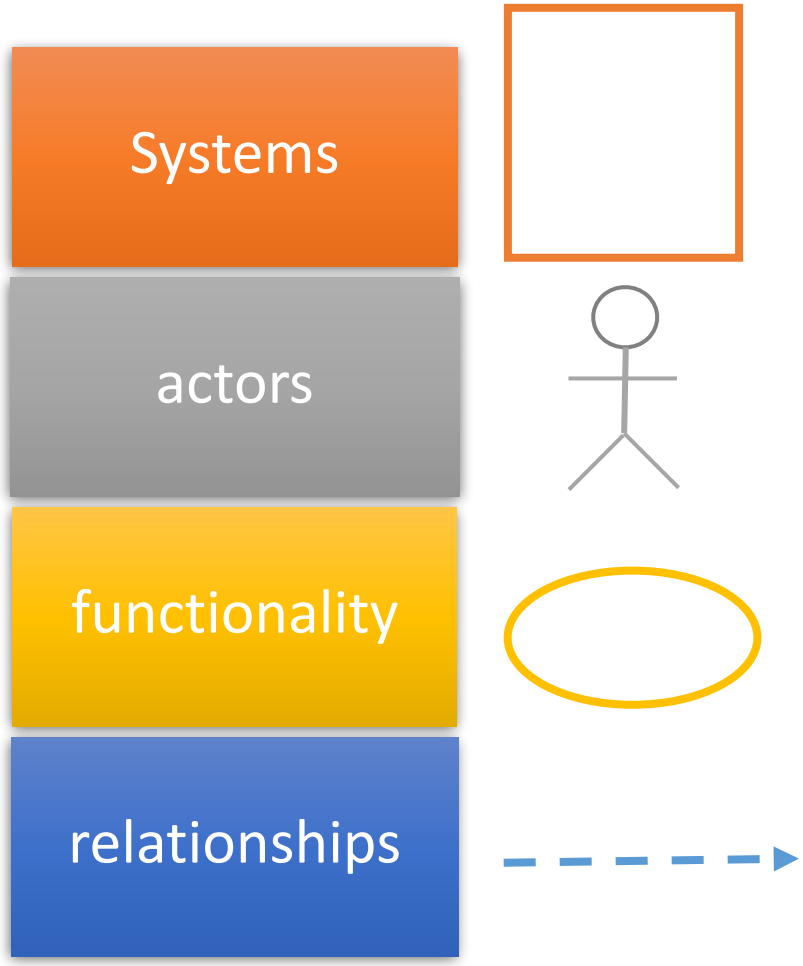
- Use cases are often derived from user requirements
  - each use case typically corresponds to a specific user requirement or a part of one or more user requirements
  - use cases help in breaking down the system's functionalities into manageable parts, making it easier to understand and implement
  - provide a structured and visual representation of how the system will fulfill those requirements in terms of user interactions and system functionality
- User requirements guide the identification and definition of use cases
  - serve as the initial input that drives the creation of UML use cases that collectively represent the complete functionality of the system
- This relationship helps ensure that the final software system aligns with the needs and expectations of its users

- Use cases help to

  - Capture the system's functional requirements from the users' perspective
  - Actively involve users in the requirements-gathering process
  - Provide the basis for identifying major classes and their relationships
  - Serve as the foundation for developing system test cases

Use case diagrams - concepts

- Use cases are built using the following concepts
  - Systems represent the application(s) or the software that offers the functionality to support the requirements
  - actors are people or other systems that will benefit from the functionality and that will initiate the tasks or the use cases
  - functionality of the system which is the collection of tasks that can be executed in the system and that are translated into use cases, decomposed in a sequence of actions
  - Relationships may exist between use cases and actors



Concepts of a User Case - system

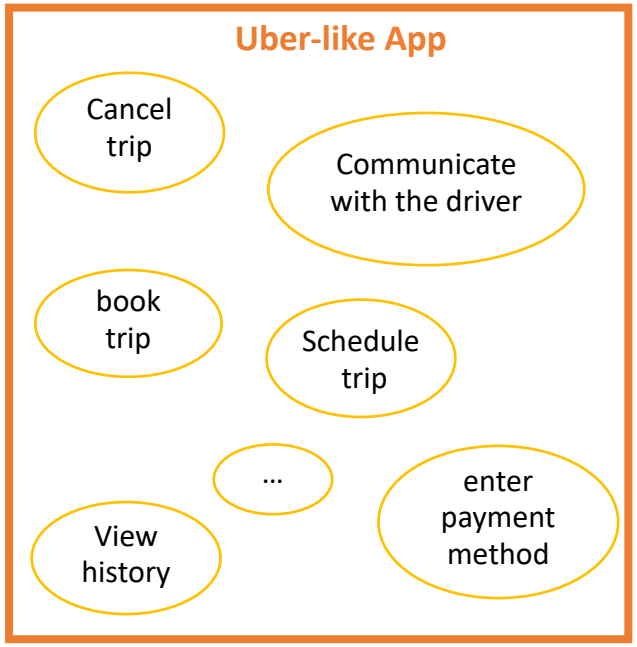
system

- is what you want to develop
  - a website
  - a smartphone app
  - a data management system
  - a video player
  - etc.
- it is represented by a rectangle and it is given a name

Concepts of a User Case - system (2)

system

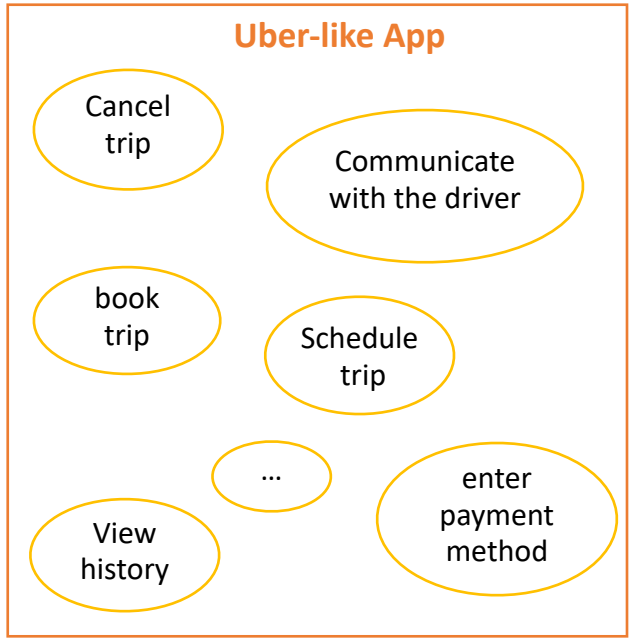
- The system implements the tasks or use cases
  - Everything that the system does must be placed inside the rectangle
- whatever is out will not occur in the system / app



Concepts of a User Case - actors

actors

- an actor is the element that uses the functionality offered by the system to achieve an objective
  - "Who" or "what" is going to use the system
    - Person
    - Organization
    - another system
    - an external device or application



Elements of a User Case (5)

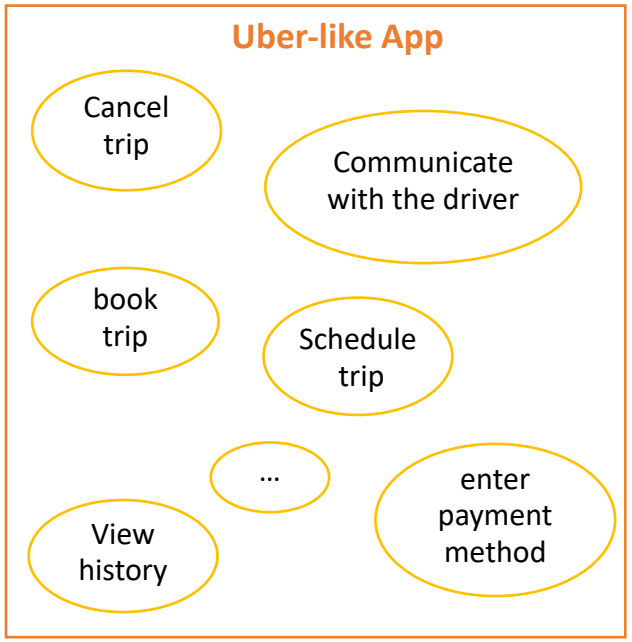
actors

- two types of actors
  - Primary: those who start using the features offered by the system
  - Secondary: those who react to initiated actions

Uber client

primary

initiate actions



Uber driver

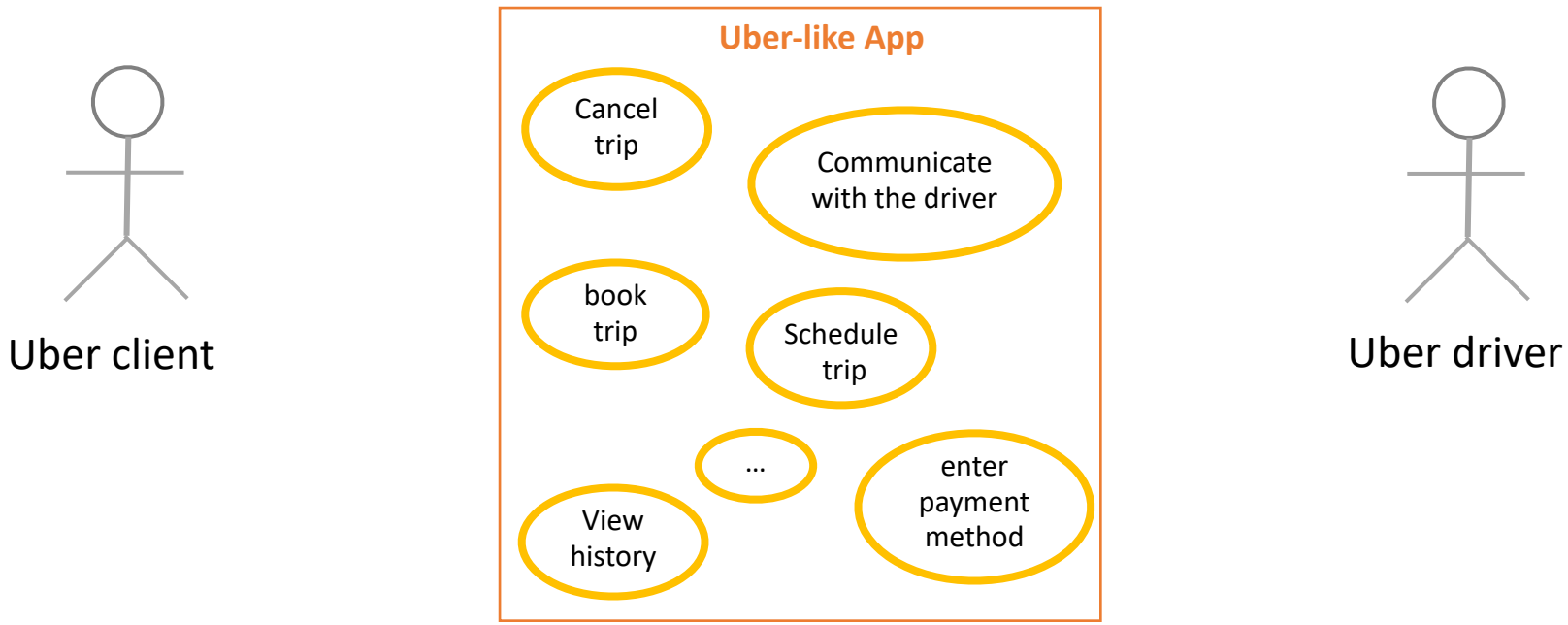
secondary

react to initiated actions

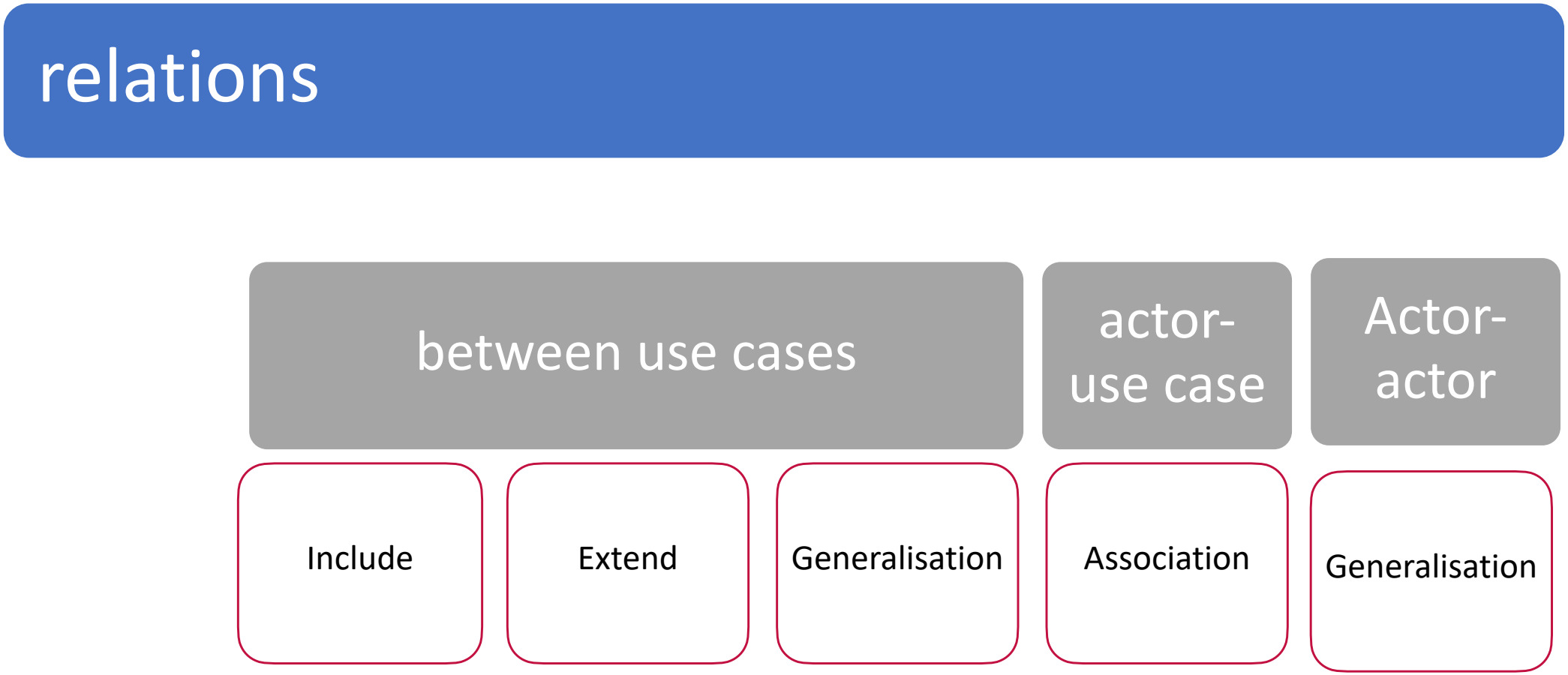
Elements of a User Case (6)

Functionality

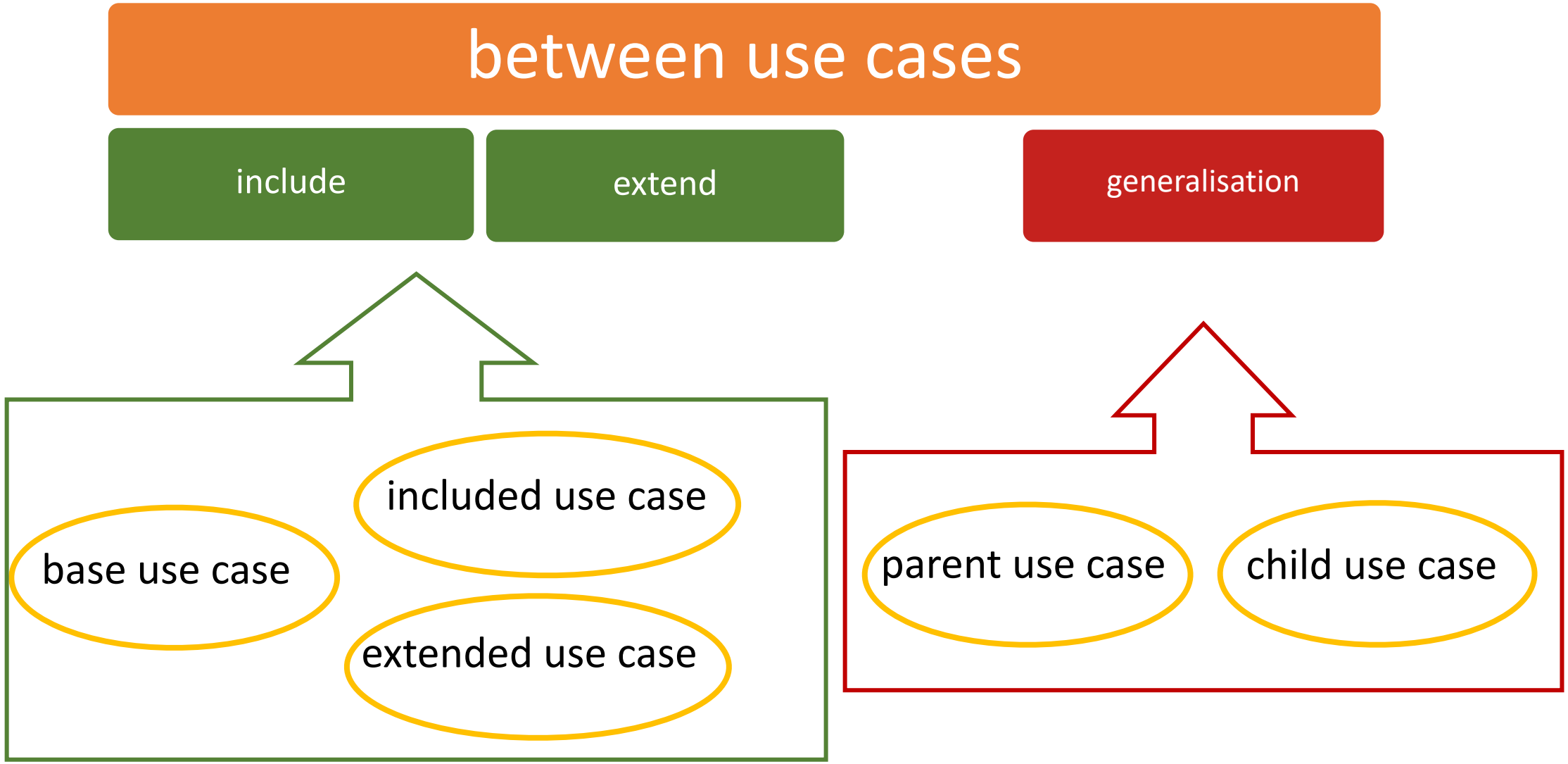
- it is actually the set of use cases
  - represents the value that the system offers to the actors
  - the actions or tasks that the actors can perform with the system



Use case diagrams relationships - include, extend, generalise, associate

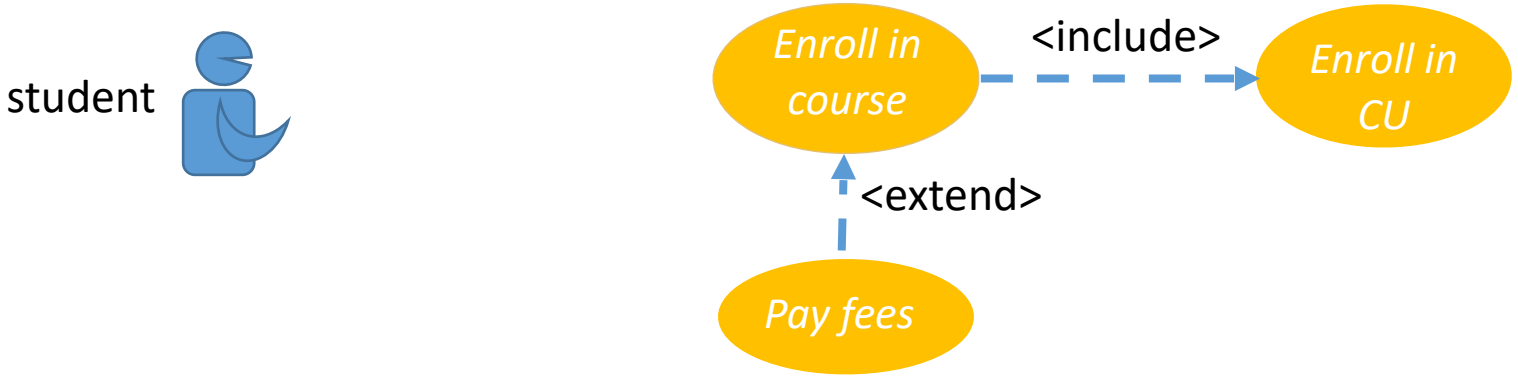


Use case relationships - include, extend, generalise (2)



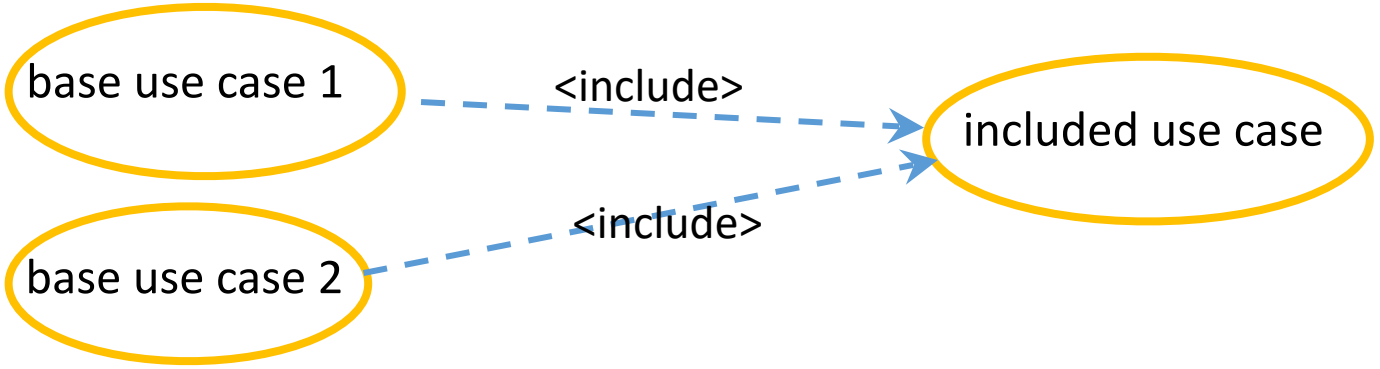
Use case relationships - include, extend, generalise (3)

- The use case *"enroll in course"* **includes** the use case *"enroll in CU"*
  - the student when registering to M.EEC, at some point will have to select and enrol in the CUs of M.EEC
    - to be completely executed, the base use case "enroll in course" will need all of the logic/sequence of steps of the use case "enroll in CU"
- The use case *"pay fees"* **extends** the use case *"enroll in course"*
  - *t*he base use case does not have the steps for the payment, because the student may have payed before enrolling
    - so the extension use case makes the base use case to be more generalised, applicable to any student who has payed or not (whereas the base use case without the extension applies only to students who have payed)



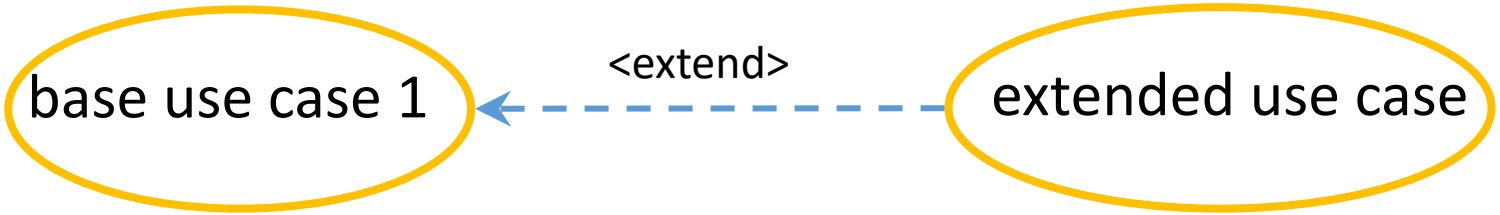
Use case relationships - include

- **<include>** one use case always involves the steps of another
  - The base use case needs always another use case to be complete
    - this other use case can be performed with other use cases (but never alone …)
  - whenever the first use case (base use case) is invoked, the second use case (inclusion use case) is automatically invoked by the system
  - *why not include the functionality offered by the inclusion use case within the base use case?*
    - *because other use cases may need it and it would be necessary to repeat it in those other use cases*



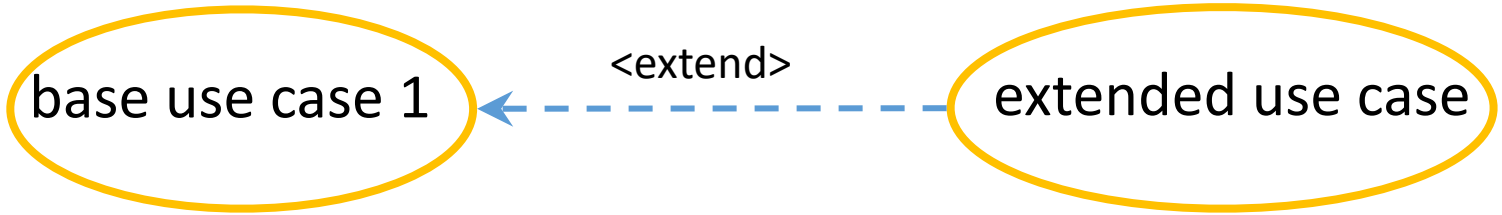
Use case relationships - extend

- **<extend>** under certain conditions, a use case may follow a variant
  - sometimes the execution of a use case may require the execution of another use case
    - this other use case when it occurs, extends the functionality of the first
    - it is only executed if certain conditions are met
  - other times, the base use case will occur without executing the extension use case
    - so, the base use case can occur without the extension use case



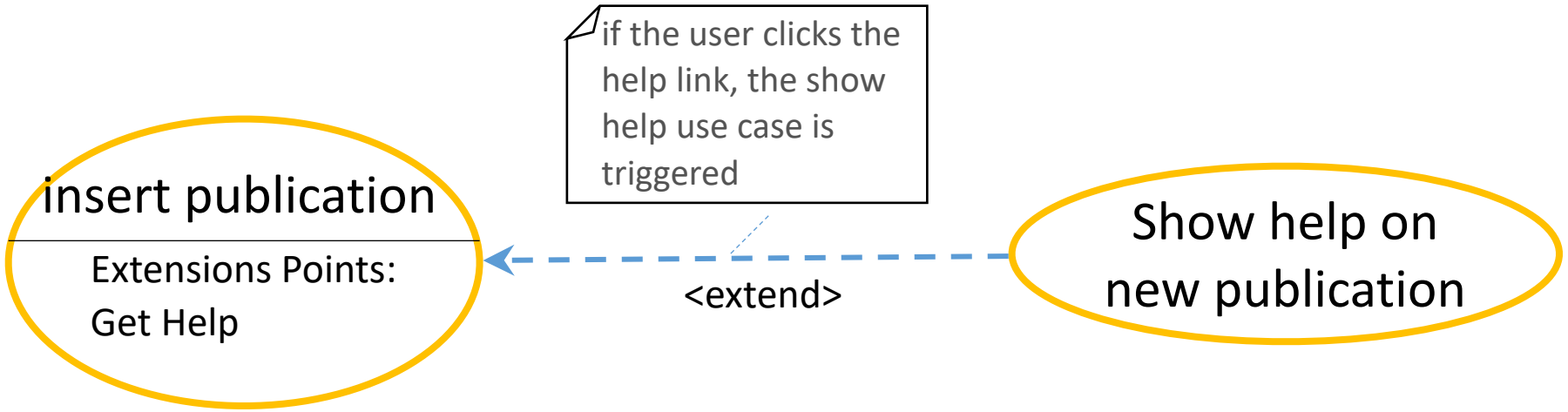
Use case relationships - extend (2)

- the base use case is defined independently and is meaningful by itself
- the extension use case is not meaningful on its own
  - it consist of one action or sequence of additional actions (segment) to augment the behaviour of the base use case under certain conditions
    - each segment can be inserted into the base use case at a different point, called an extension point
  - it can access and modify the attributes of the base use case whereas the opposite does not hold true because the base use case is not aware of the extension use case



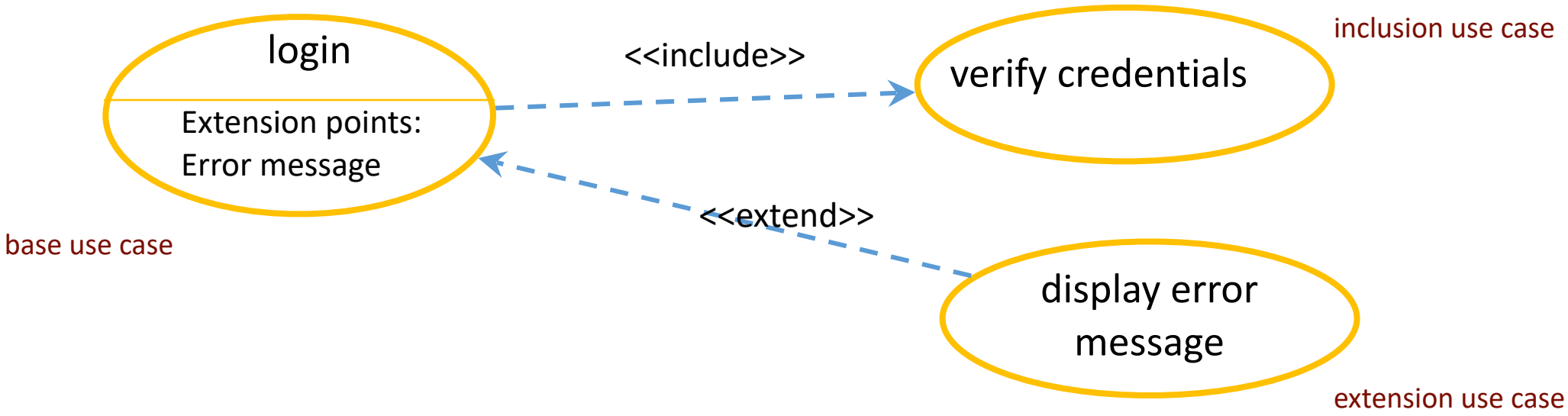
Use case relationships - extend (3)

- Base use case with extension points
  - when creating the base use case, it is possible to list extension points
    - related with the functionality offered by the extension use cases
  - on one side it offers more descriptive information on the system
  - on the other side it helps the developer in keeping track of the possible variants of the main course of action



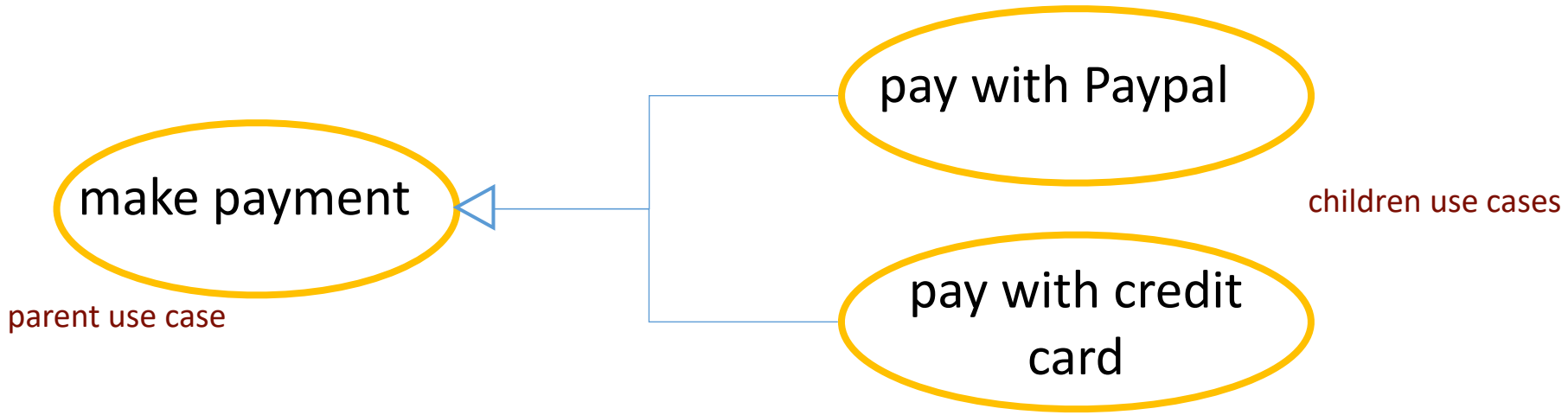
Use case relationships - include and extend

- **<include>** and **<extend>**
  - whenever the user invokes the use case *login*, the system automatically verifies the credentials, invoking the use case *verify credentials*
  - the system will only invoke the *display error message* use case, if the credentials were not correct



Use case relationships - generalisation

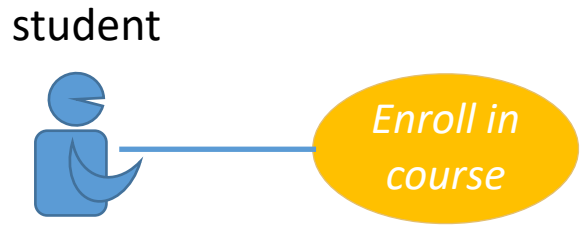
- generalisation
  - a task can have different execution variants
    - instead of defining a use case for each variant associated with the actor (it can complicate the GUI)
    - or define a single complex use case that considers all variants
    - a generic use case (parent) and specific use cases for each variant (children) are defined and are invoked depending on the interaction



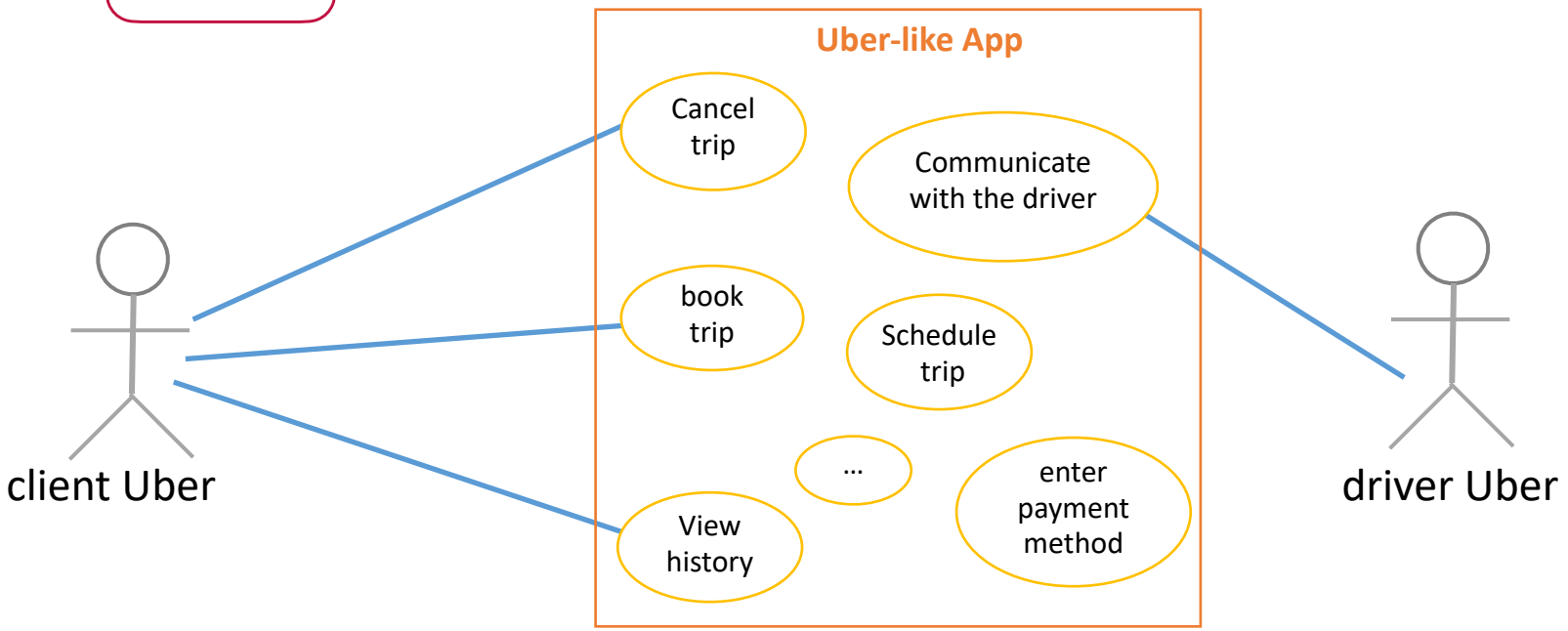
Use case relationships - association

relations

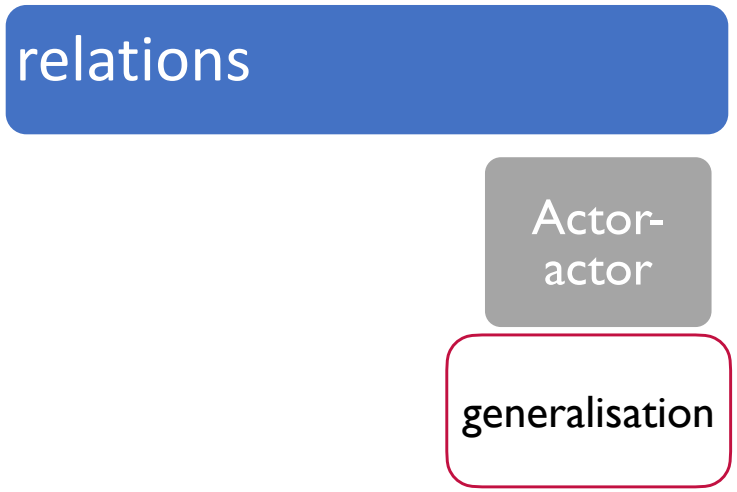
- actor-use case
- association

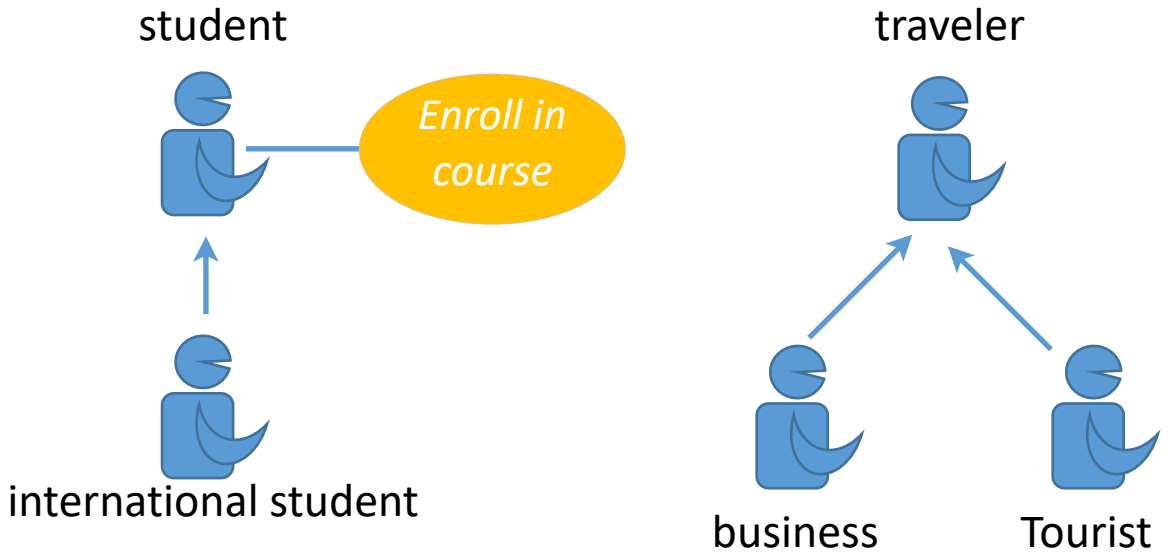


- to indicate that there is a communication or interaction between an actor and a use case



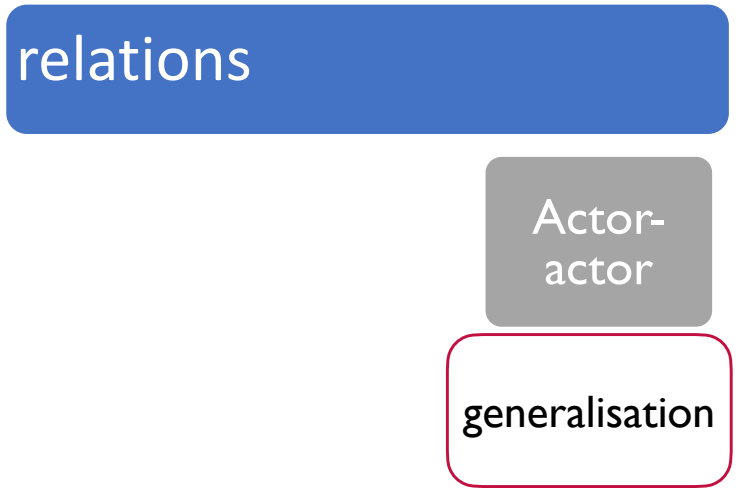
Use case relationships - generalisation

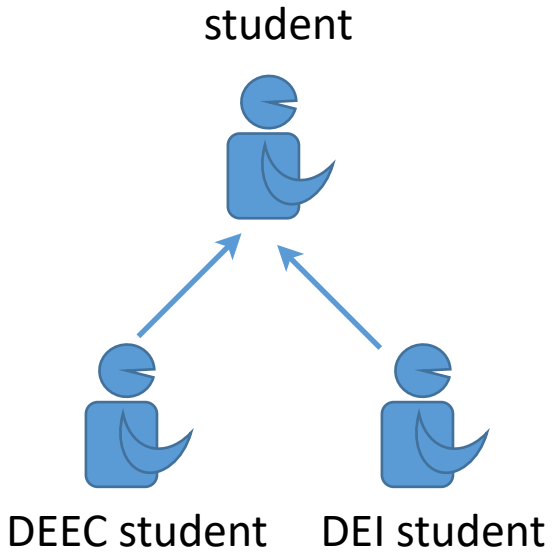




- It is a relationship that can exist between two actors
- when there are variants of an actor (ancestor) but those variants (descendants) share the role and properties of that actor
  - the descendant inherits all the use cases of the ancestor
    - the descendant can use all the use cases that have been defined for its ancestor
  - the descendant may be additionally associated to one or more use cases that are specific to it

Use case relationships - generalisation (2)





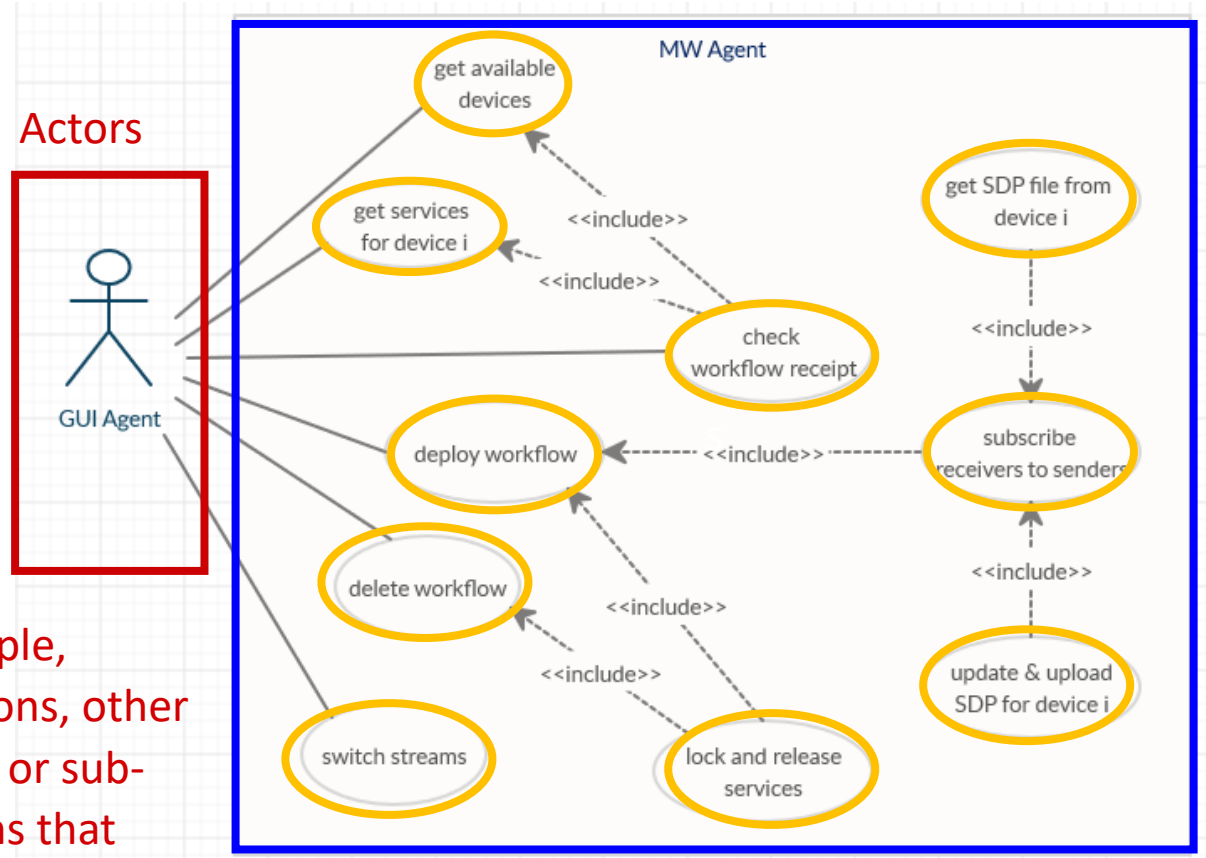
- actors in UML are classes in OO programming
  - the generalisation relationship between actors leads to the inheritance relationship between two classes by which one class inherits all the properties and methods of another class

What to include in a UML use case diagram?

- The system
- Who or what is using the system
  - actor
- What does the actor wants? What is his/her objective when using the system?
  - translates into a system requirement and in the functionality offered by the system to support that requirement or the objective the actor wants to achieve
- The steps the actor should execute to achieve the desired goal (the steps/actions of the task)
- How will the system react to the actors's actions

- But it does not include
  - Programming language details
  - Graphical interface details

Use case diagram



Developing and describing use cases

- The events/steps to be performed in a use case should be described in the form of a simple narrative
  - First thing is to list the use cases and actors
  - then, for each use case, identify the main actor (the one that initiates it) and all the others that intervene
  - mentally work out how the use case will play out
    - remember that a use case corresponds to one of the tasks that were outlined when building the low-fidelity prototype
  - then decide what sequence of steps to take to reach the goal under normal circumstances
    - “Normal/Typical flow of events" that match the base use case

Describing use cases (2)

- Describe in narrative form what the actor does and the respective reactions of the system
  - In particular those that are important for the actor to know (because they will in turn trigger an actor's new action)
- When the normal course of events/steps is completed, you should identify if there are alternative courses that may occur under certain conditions and describe them
  - "alternative flow of events"
    - that can correspond to the execution of extension use cases if the logic for an alternate course of action is at a complexity level similar to that of the base course of action

Describing use cases (3)

- When all use cases are described, try to identify sequences of events/steps that are common to several
  - bookmark these use cases
- A sequence of events/steps that is common to several use cases may be turned into a new use case
  - the bookmarked use cases, instead of having each one of them to detail that sequence of steps repeatedly, will include the new auxiliary use case (inclusion use case)

What is the best way of describing use cases?

- Adopting the referred methodology and using tables
  - an overall table to list all of the defined use cases

Use case ID	Name	Vale offered / objective to achieve	Primary actor
UC00	Book trip	Request and book a car for an immediate trip	client
UC01	View history	Visualise the list of trips booked previously	client
UC02	…	…	…
…	…	…	…
UCn	…	…	…

What is the best way of describing use cases? (2)

- And then two tables for each use case
  - one table with a brief description

Use case ID - name	UC00 - Book trip
Actors:	Client, driver
Objective:	Allow the actor-client to book a car at the moment, indicating the pickup and destination points and confirming the payment. If desired, the client can select the payment method and set a tip.
Summary:	The client opens his app and defines the pickup and destination locations. The system checks whether cars are available and presents a price estimate to the customer. The client confirms the trip or not.
Type:	Primary

What is the best way of describing use cases? (3)

- another table listing and briefly describing each step/action taken by the actors and the system responses that make up the use case, in the sequential order as they occur

Typical Course of Events					
Step #	**Actor-XXX action**		**actor-YYY action**		**System response**

- For some of the actors' actions, the system might provide different responses
  - the most likely sequence of actions and responses will be listed in the table under the designation of "Typical course of events" (the base use case)
  - Alternatives will be inserted below the table as "alternative courses" (the possible extension use cases)

What is the best way of describing use cases? (4)

Typical Course of Events					
Step	**Client action**		**Driver action**		**System response**
1	This use case starts when the actor client selects “new trip”			2	The system presents an interactive map to the client
3	The client selects pickup and destination addresses			4	The system checks whether there are cars available in the vicinity and contacts the driver
		5	The driver replies indicating availability	6	The system computes an estimative of the fare and presents it to the client
7	The client confirms the trip			8	…

Alternative courses:

Action 4: the system checks for available cars but does not find any. It sends a message to the client saying to try latter. The client finalises the request. - possible extension use case

Action5: the driver does not confirm availability and thus the system repeats action 4

Action 6: the client cancels the trip

What is the best way of describing use cases? (5)

Typical Course of Events					
Step	**Client action**		**Driver action**		**System response**
7	The client confirms the trip			8	The system presents payment options to the client
9	The client selects payment option			10	The system presents details of the payment method selected
11	The client insert requested data and pays			12	The system checks payment and finalises showing success message.

Other possible alternative courses:

Action 7: the client selects number of passengers option - possible extension use case

Action 7: the client selects type of car option - possible extension use case

Action 9: the client selects tip option - possible extension use case

Action 12: the system is not able to confirm data and display error message and goes back to action 8

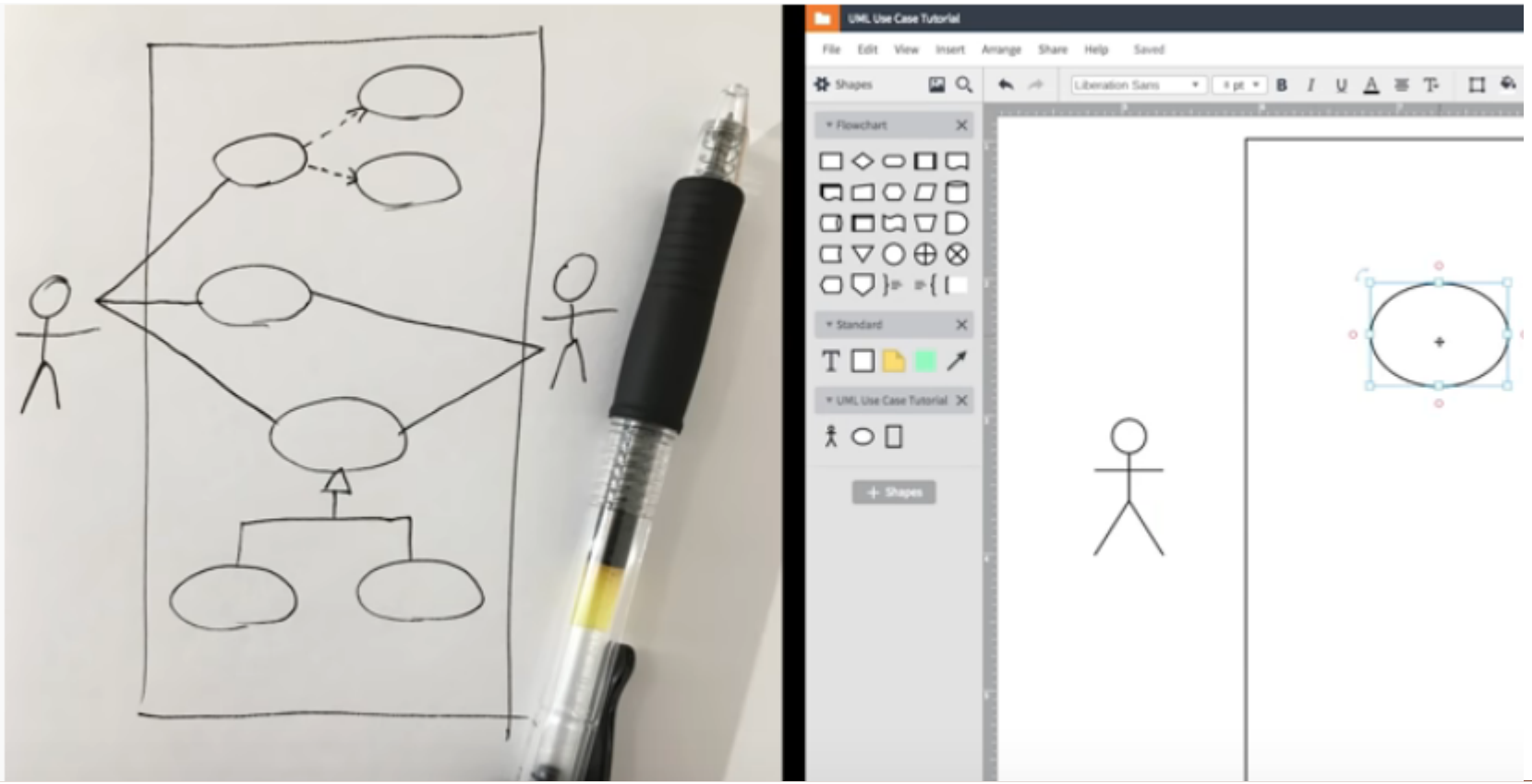
What is the best way of describing use cases? (6)

Alternatively, a more limited use case can be defined, leaving all of the payment process in a separate use case, which can be included in this one:

Typical Course of Events					
Step	**Client action**		**Driver action**		**System response**
1	This use case starts when the actor client selects “new trip”			2	The system presents an interactive map to the client
3	The client selects pickup and destination addresses			4	The system checks whether there are cars available in the vicinity and contacts the driver
		5	The driver replies indicating availability	6	The system computes an estimative of the fare and presents it to the client
7	The client confirms the trip			8	The system concludes with a message indicating that payment is needed and invokes inclusion use case “payment”

How to develop UML use cases?

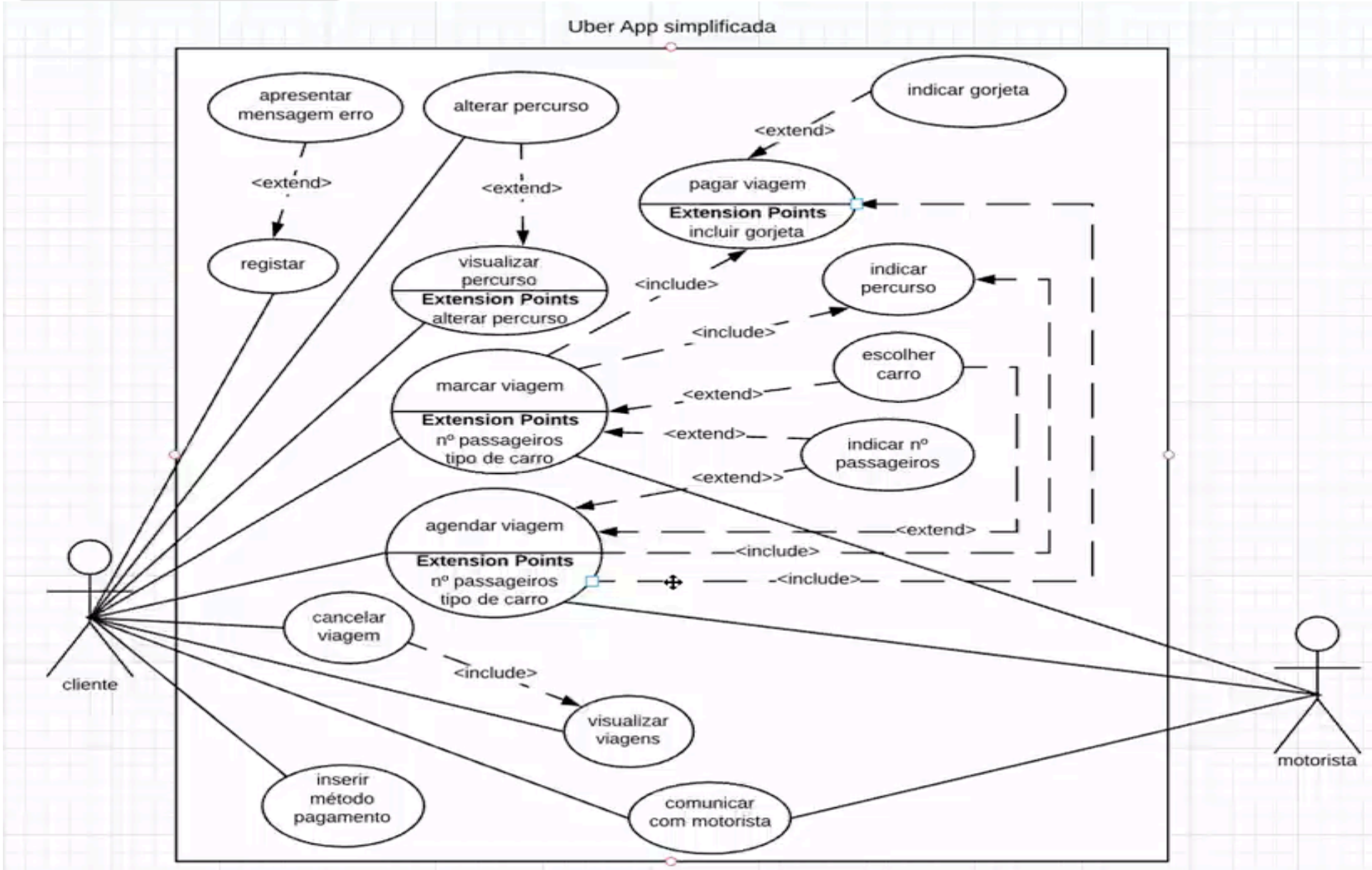
- Pencil and paper
- Using online tools



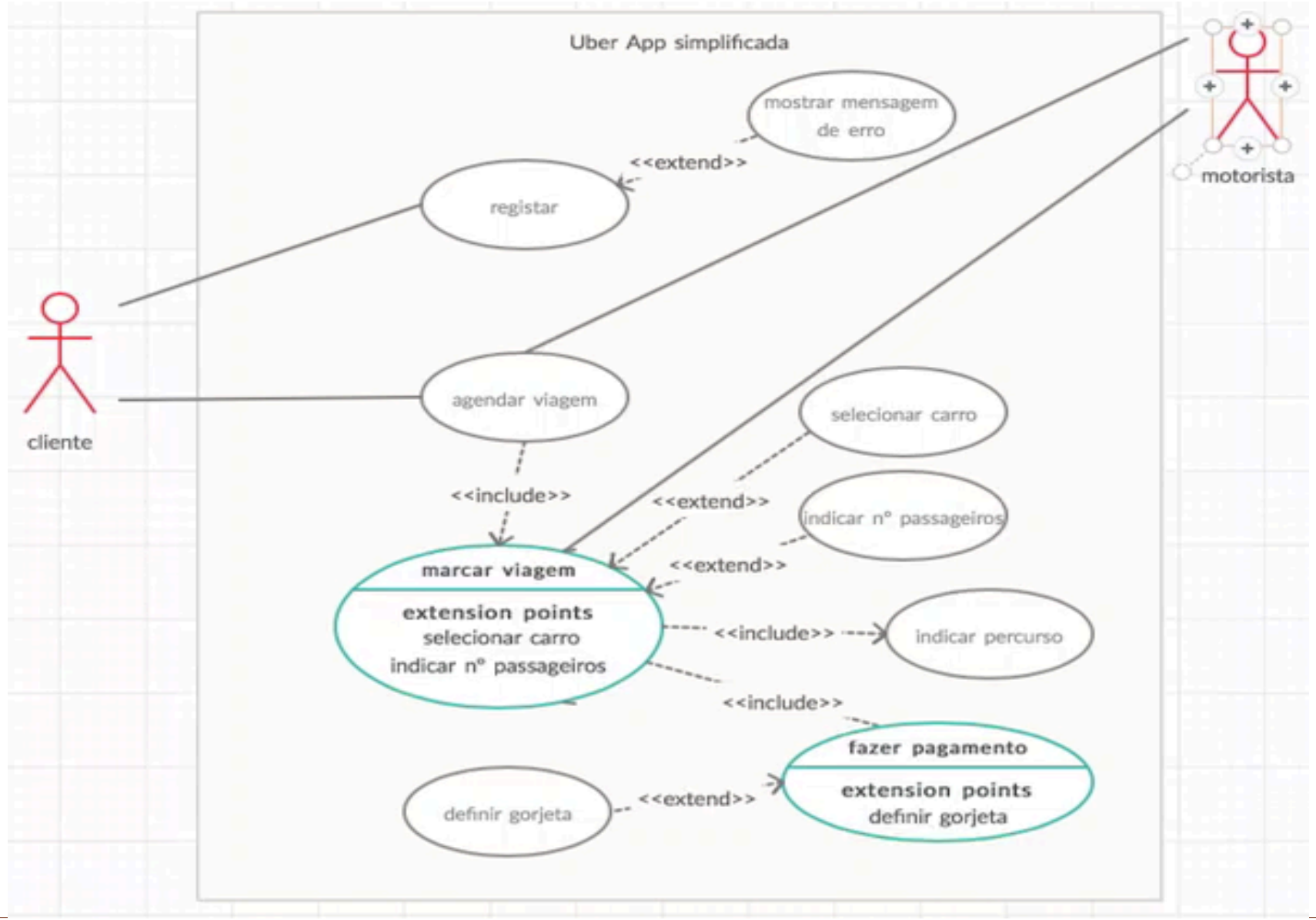
online UML tools

- Lucid Chart:
  - https://www.lucidchart.com/users/registerLevel?utm_source=youtube&utm_medium=video&utm_campaign=use_case_tutorial_card1&referer=https%3A%2F%2Fwww.youtube.com%2Fwatch
  - Get started: https://www.youtube.com/watch?v=hhowdkP1hlY
  - Use cases: https://www.youtube.com/watch?v=zid-MVo7M-E
  - Class Diagrams: https://www.youtube.com/watch?v=UI6lqHOVHic
  - Sequence Diagrams: https://www.youtube.com/watch?v=pCK6prSq8aw&list=RDCMUCnd94o169CbOIJyiEUkTL2A&index=2
  - https://www.lucidchart.com/pages/how-to-draw-a-use-case-diagram-in-UML#discovery__top
- Creately
  - https://app.creately.com/

Using Lucid Chart to build use cases of the Uber-like app



Using Creately to build use cases of the Uber-like app



Examples and references

- Look at documents D2.1-FotoInMotion e D5-Enthrone available on Moodle
- https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-creating-use-case
- https://www.uml-diagrams.org/use-case-diagrams.html
- https://creately.com/blog/diagrams/use-case-diagram-relationships/