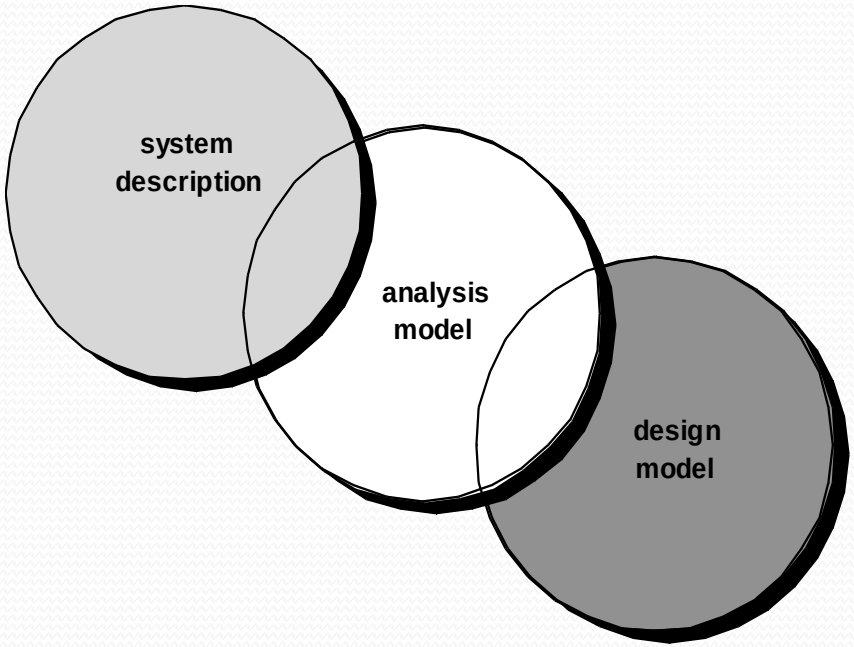
Requirement Analysis
System Analysis

Outlines

- Three aspects of analysis: functional, informational, and behavioural aspects
- Analysis models corresponding to these aspects
- Two analysis approaches
  - Structured analysis
  - Object Oriented analysis
- Use case diagram for functional aspect analysis
- Class diagram for informational aspect analysis

Analysis Model forms a Bridge between Requirements & Design

- Analysis model
  - First technical representation of a system.
- Two approaches
  - Structured analysis
  - Object oriented analysis



Analysis principles

- The functions that the software is to perform must be defined.
- The information domain of a problem must be represented and understood.
- The behavior of the software (as a consequence of external events) must be represented.

Analysis models and Design models

- *Analysis models* represent the **customer requirements** by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain

- *Design models* represent **characteristics of the software** that help to construct it effectively: the architecture, the user interface, and component-level detail.

Object Oriented Analysis

- A system is considered as a number of objects that interact
- Semantic gap between models and reality is minimized
- An object is characterized by a number of operations and a state which remembers the effect of these operations

Object Oriented Modelling

- Using Unified Modelling Language (UML)
- Functional aspect with scenario based model: Use case model
- Informational aspect with data modelling: Class model
- Behavioural aspect with behavioural modelling: Activity model, state diagram, sequence model

Use Cases Model

- Use case model illustrates the system's intended functions (use cases), its surroundings (actors), and the relationship between use cases and actors
- It also expresses the functional partitioning of the system, i.e. specifies well-defined subsets of functionality
- A use case model includes **Actor, Use cases** and **Association**

Use cases model - Actor

- Actors represent external entities that interact with the system. An actor can be human or an external system.
- Actors can be identified based on the requirement statements by answering some questions
  - Who is interested in a certain requirements ?
  - How the actor interacts with the system (e.g. two actors use the system in the same way, they must be the same actor)
  - Which user group are supported by the system to perform their work ?

Actor

- Which user groups execute the system's main functions ?
- Which user groups perform secondary functions, such as maintenance and administration ?
- With what external hardware or software system will the system interact ?

- Example, in Library Management System, actors are Borrower, Librarian

Use cases model – Use case

- A use case represents a unit of functionality of value to an actor.
- A use case is an abstraction that describes a class of scenarios, a scenario is a narrative description of what people do and experience as they try to make use of computer system
- Use case is often named with a verb to describe an action.
- Example in a Library Management System, use cases are Searches Books/Resources, Reserve Resources, Return Resources, etc.

Derive actors and use cases from Video Store example

- R1. The system shall allow users to place a renting order

  R1.1 New user needs to register first, after registration, the user receives a code and member card

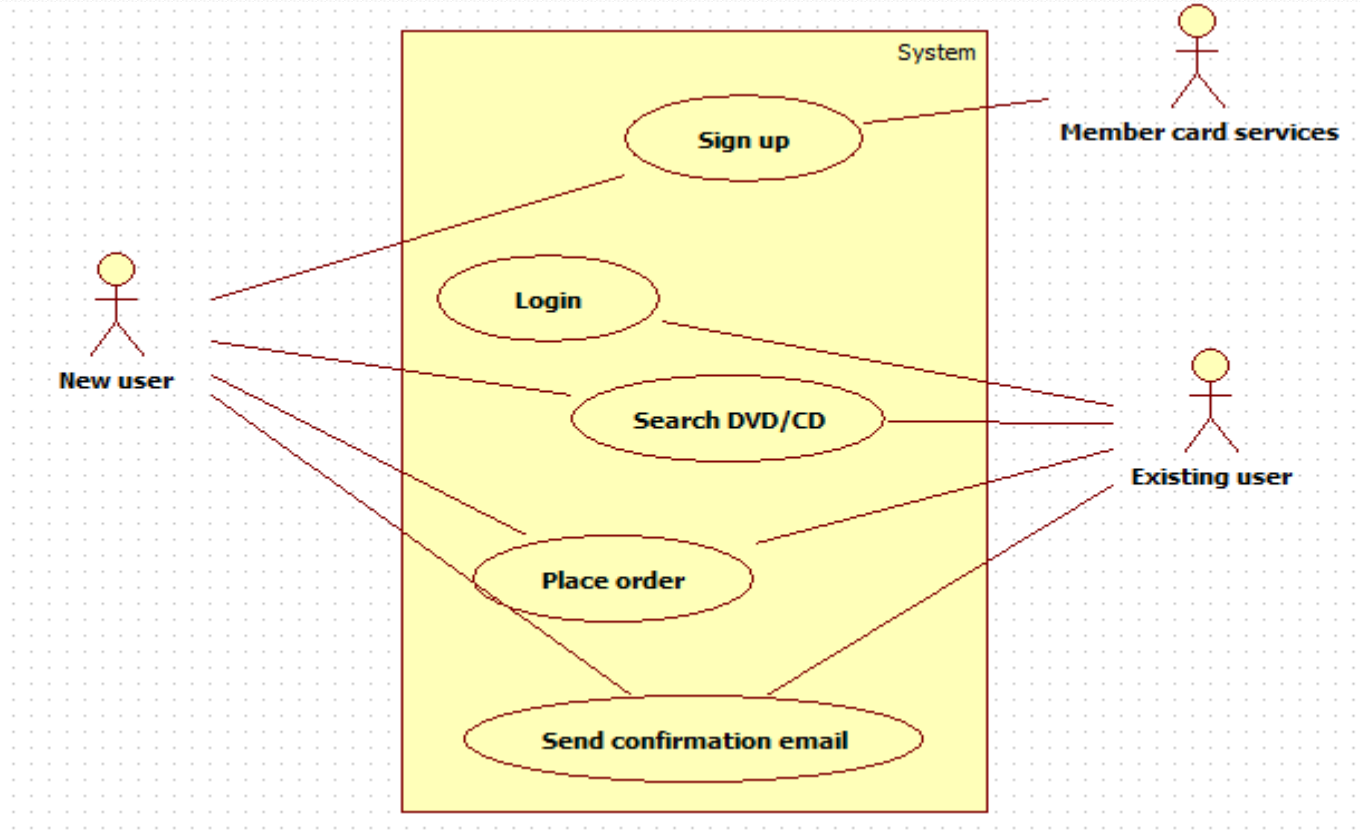
  R1.2 Existing user can login to the system

  R1.3 User shall search the DVD by name, by type, or by actors

  R1.4 User shall select the DVD they want to rent to make a reservation

  R1.5 The system shall send a confirmation email to user after placing a renting order.

Use cases diagram - Place a renting order



Use case model - Association

- Association
  - A communicate association between a use case and actor
- Three other types of association between use cases
  - <<Include>> association
  - <<Extend>> association
  - <<Generalisation>> association
- Generalisation association can be an association between actors

<<Include>> association

- A relationship from a base use case to an inclusion use case, specifying a dependency relationship, the execution of the base use case including the execution of the inclusion use case

- Redundancies among use cases can be factored out using include relationship

- Example: *Place order* <<include>> *Search video*

<<Extend>> association

- A relationship from an *extension* use case to a *base* use case, specifying an optional dependency relationship between these use cases.

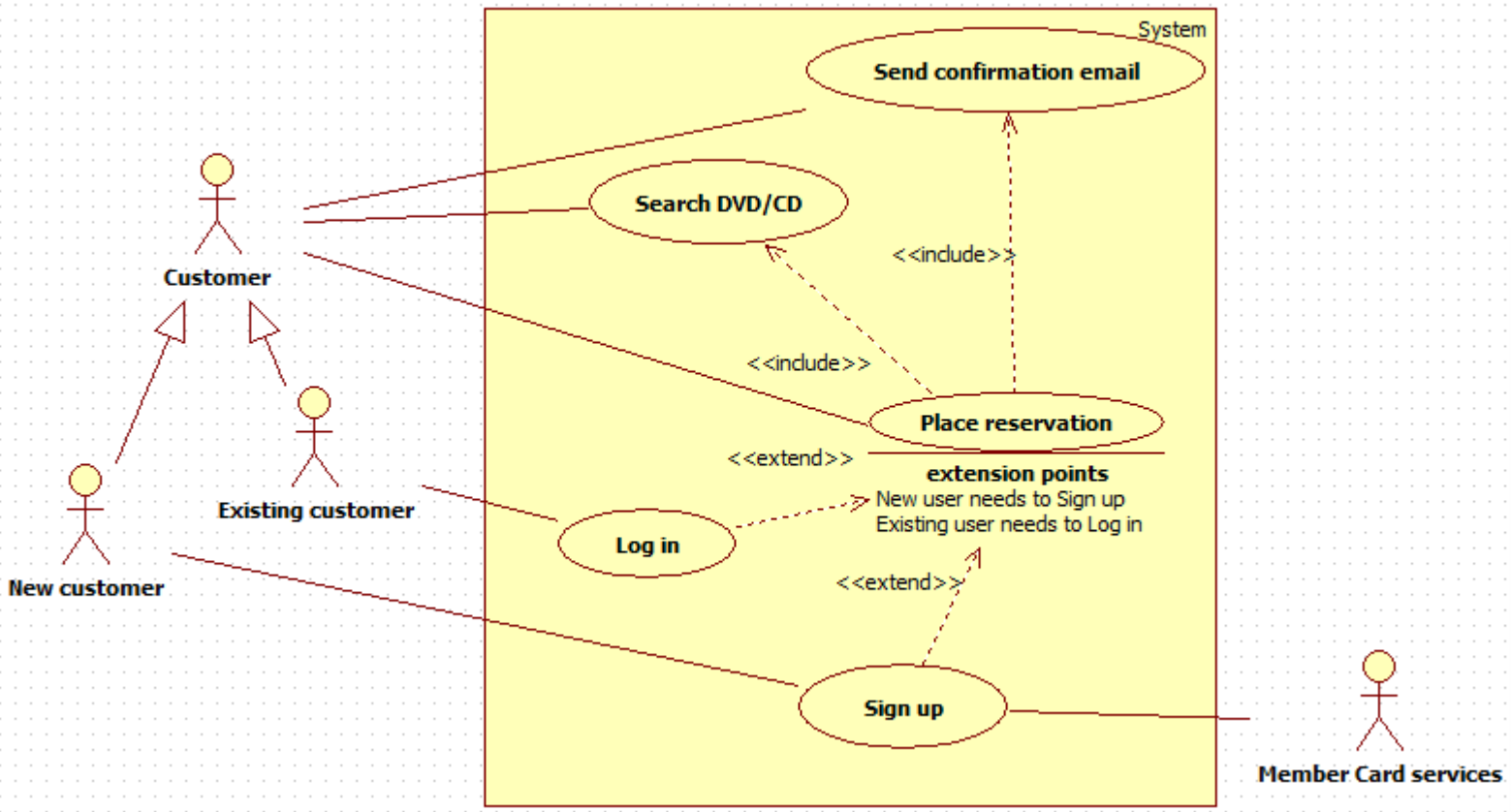
- Example
  - *Place order* <<extend >> *Log in*
  - *Place order* <<extend>> *Sign in*

This case log in and sign in is an option

<<Generalisation>> association

- Generalisation association between actors/use cases
  - A taxonomic relationship between a more general actor/use case and a more specialized actor/use case
  - The specific actor/use case inherits bahaviours of the more generic actor/use case
- Example:
  - *Existing customer* and *New customer* are specialized actors of Customer
  - *Place order online* and *Place order by phone* are specialized of *Place order* use cases (suppose the system allow place order by phone)

Example


System
Send confirmation email
Search DVD/CD
<<include>>
<<include>>
Customer
Place reservation
<<extend>>
extension points
New user needs to Sign up
Existing user needs to Log in
Existing customer
Log in
New customer
<<extend>>
Sign up
Member Card services

Use case narratives/description

- A use case can be described with narratives to

specify more details of scenarios, pathways or exceptions of interactions

- The narrative include
  - Pre-condition
  - Main flows/Basic flows
  - Alternative flows
  - Post-condition

It is not obliged that each use case has an associated narrative, only main use case does.

Conditions

- Pre-condition: A statement of the conditions that must exist before the use case can successfully take place.

- Post-condition: A statement describing the modified state of the system as a result of executing a use case.
- E.g.: Use case Place order
  - Precondition: User must be logged in to the system.
  - Post-condition: An order object is created and a confirmation email was sent out.

Flow of events – Use case narratives

- A flow of events is associated to a use case. It describes the events needed to accomplish the required behaviour of the use case. In other words, it describes the scenario of what the use case does.

- The flow of events describes ***what*** the system should do, not ***how*** to do it.

Flow of Events (contd.)

- A flow of events description covers:
  - How and when the use case starts
  - When the use case interacts with the Actors and what data they exchange
  - What data is needed by the use case
  - The normal sequence of events for the use case
  - The description of any alternate or exceptional flows
  - How and when the use case ends

Flow of events

- Basic flow of events (main scenarios)

  Describe the normal sequence of events for the use case

- Alternative flow

  Describe some variation or exceptions from the steps in the basic flow

Example flow of events-Main flow

- Use case Sign up
- **Main flow:**
  - 1. User provides name, a postal address, date of birth, an email address (login account) and a password
  - 2. The system validate the information such as if the email address is valid, the password is valid
  - 3. A barcode is generated and the customer information is transformed to the card service
  - 4. User registration is finished

Example flow of events-Alternative flow

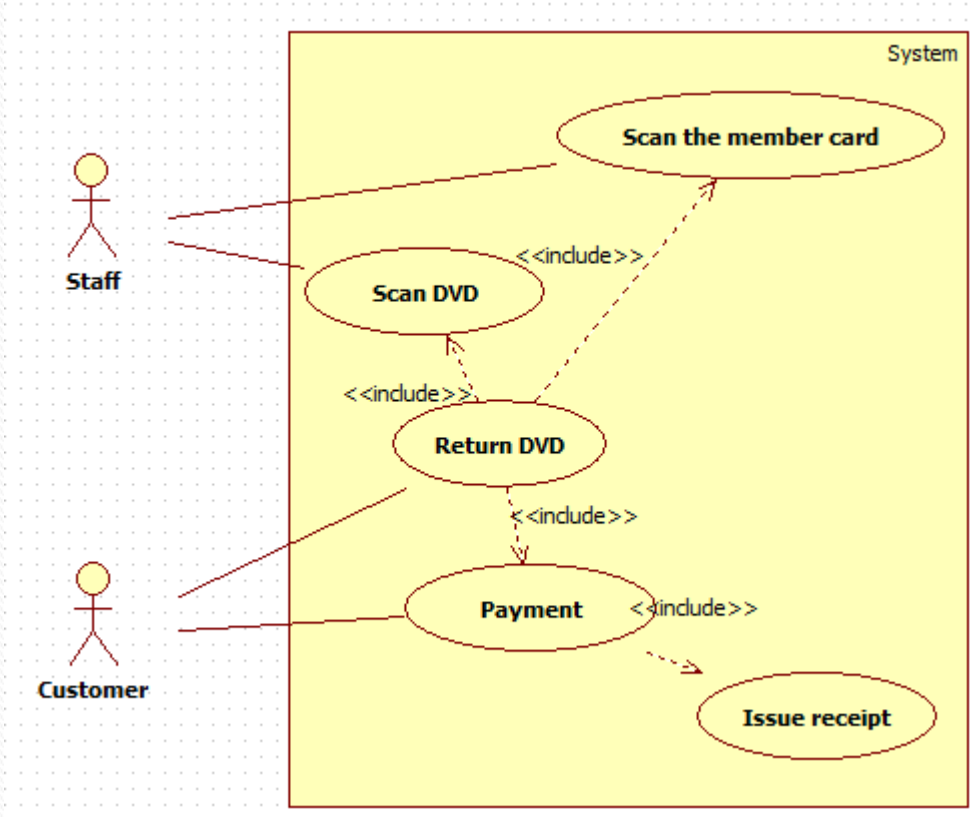
- **Alternative flow**:
  - 2.1 If invalid email address, announce invalid email address and repeat Flow 1
  - 2.2 If invalid password, announce invalid password and repeat Flow 1

Second example

- *R2. When the customer returns the DVDs, the correct payment shall be calculated based on renting time, and the status of DVDs must be updated.*

- Details, for example:
  - 2.1 A staff scans the user card
  - 2.2 The system retrieves the renting details
  - 2.3 The staff scans each DVD
  - 2.4 The system retrieves the DVD and updates its status from "rented" to "available"
  - 2.5 The system calculates the fee
  - 2.6_Customer pays fee
  - 2.7 The system prints out the receipt

Use cases diagram – refined version



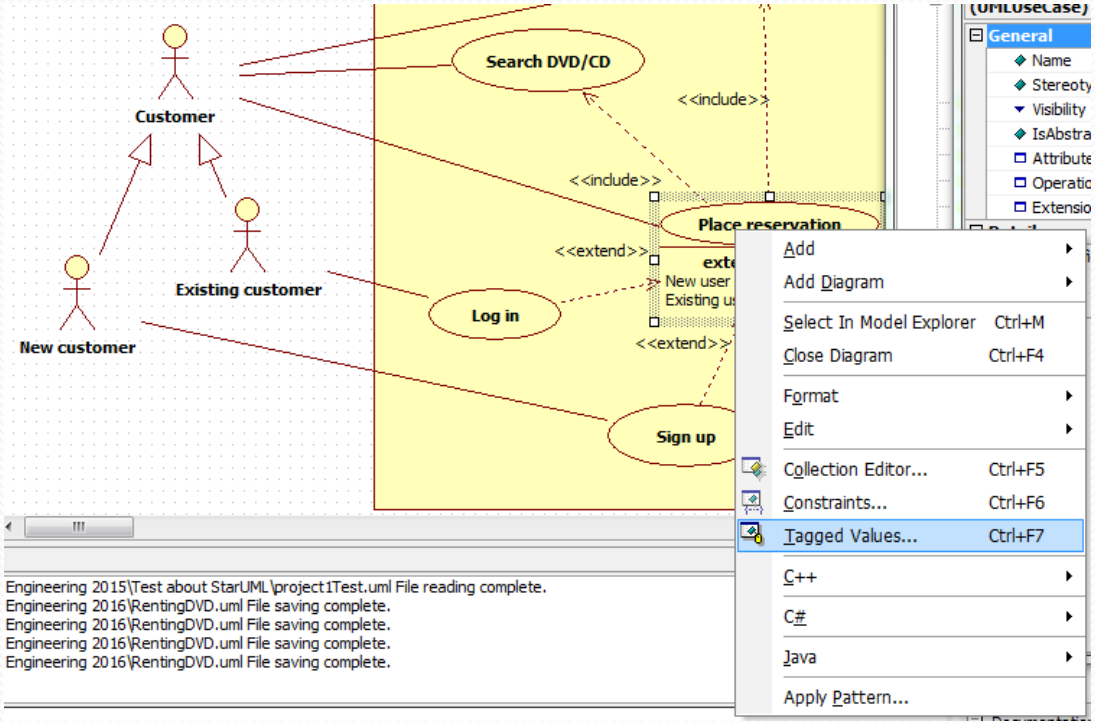
Flow of events of Return DVDs use case

Basic flows:

1. Use case starts by Staff scanning the member card
2. The system retrieves the renting order
3. Staff scans each DVDs
4. The system retrieves the DVD
5. The system updates DVD status to available
6. The system calculates the corresponding fee
7. User pays
8. System prints out the receipt
9. The use case finishes

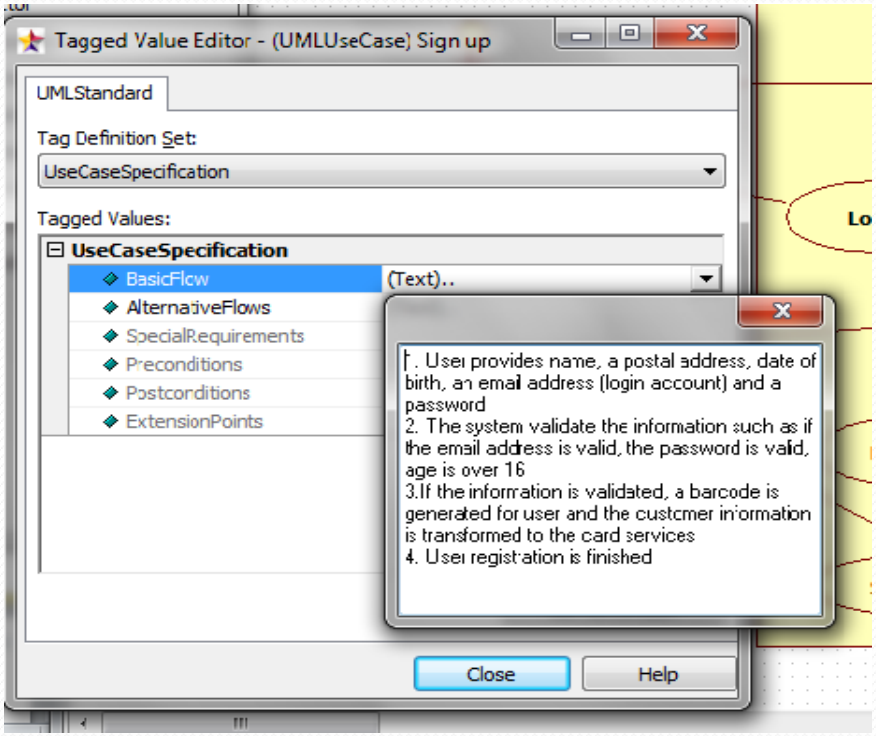
Adding Flow of events in StarUML

- Adding flow of events, right click on the use case, select Tagged Values



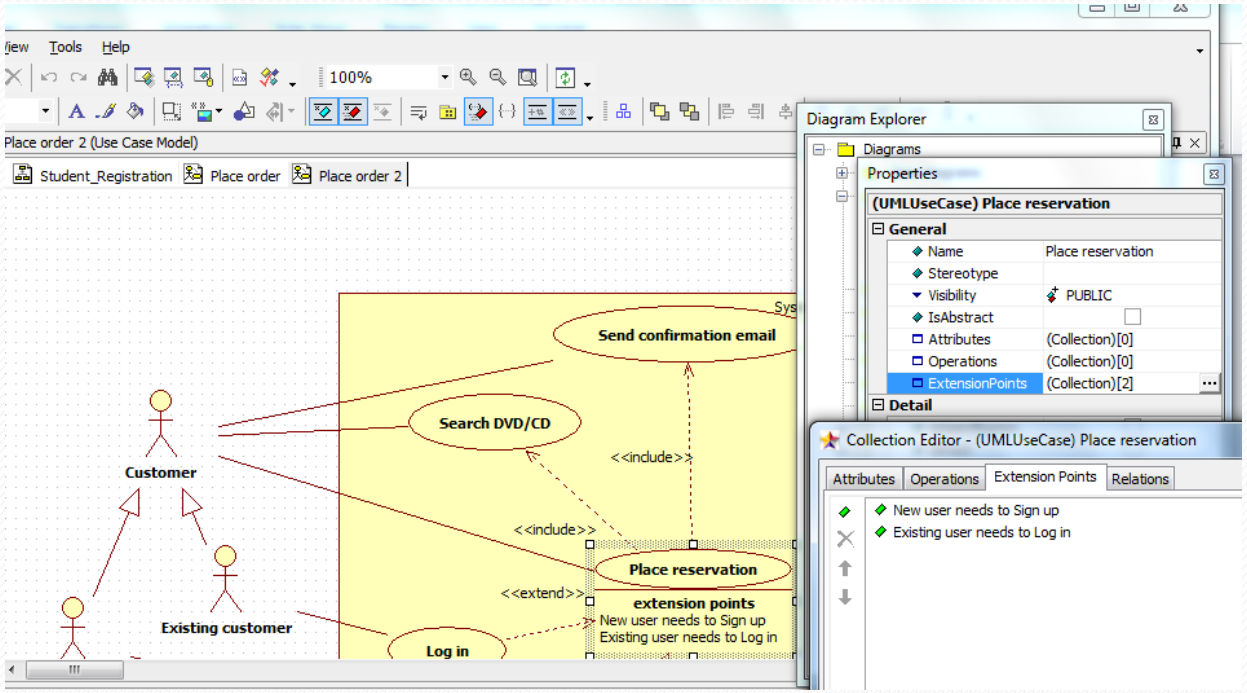
Adding Flow of events

- Select UseCaseSpecification in Tag Definition set
- Enter the texts for Basic flow/ Alternative flow /Precondition/Postcondition



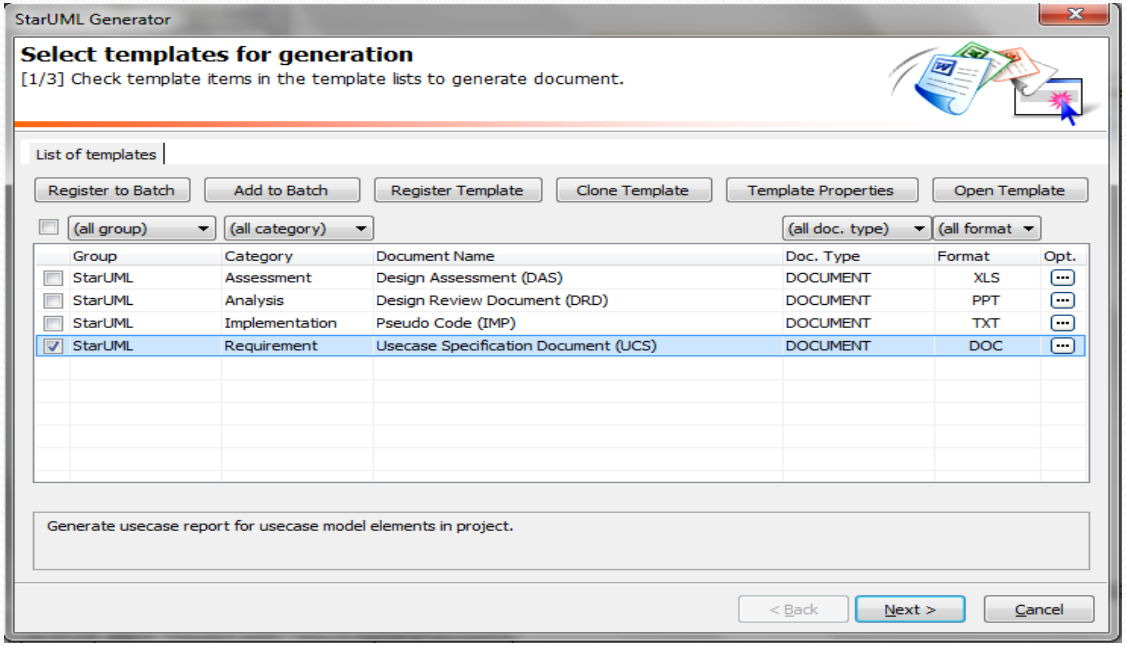
Adding Extension point to a use case

- Click ExtensionPoints in the Properties window of the use case, then Click insert (small green rectangle) to add extension point



Generate Use cases documents in StarUML

- Select Tools from the Menu and StarUML Generator



Remarks

- Use cases model aims to describe the functional aspect of the system in terms of interactions between the system and external actors
- Use cases model is a scenario-based model
- There is a variety of level of detail of Use case , a use case should be of value to an actor

Data Modelling

- Examines data objects independently of processing
- Focuses attention on the data domain
- Creates a model at the customer's level of abstraction
- Indicates how data objects relate to one another

Data Object Concepts

- Objects: represents an entity in the real world or conceptual object. Each object has state, behaviour and identity.
- An object state is a possible condition in which the object may exist
- Attributes: define characteristics of objects. Set of attributes value define the object state
- Behaviour: determines how an object responds to requests from the other objects and typifies everything an objet can do. Object behaviour is represented by Operations
- Object identifier: each object/instance is identified uniquely

Classes

- Class: a class is a description of a group of objects which have the same properties (attributes), the same behaviour and the same relationship with the other objects
- Class is named with vocabulary in the domain, usually it is a noun.
- At this stage we just take into account *Entity classes* which represents persistent database objects

- Example:
  - Before a video can be rented out, the system confirms the customer identity by swiping his/her video store membership card over a scanner
  - The customer must pay the nominal fee before the video can be rented out. The payment may be in cash or by credit/debit card

Class: Video, Customer/ Customer card

Class: Customer, Video, Payment, Rental

Class-Attributes

- Attributes of a class define the class structure. Attributes are discovered from user requirements and domain knowledge.
- Key of a class : one or more attributes that have unique values across all instances of the class.
  - Video:

  Video Code (key), Video Name, Type, Year of production.
  - Customer:

  Customer Code (key), Name, address, date of birth, email address, etc.
  - Rental

  Rental Number (key), Date, etc.

Association/Relationship

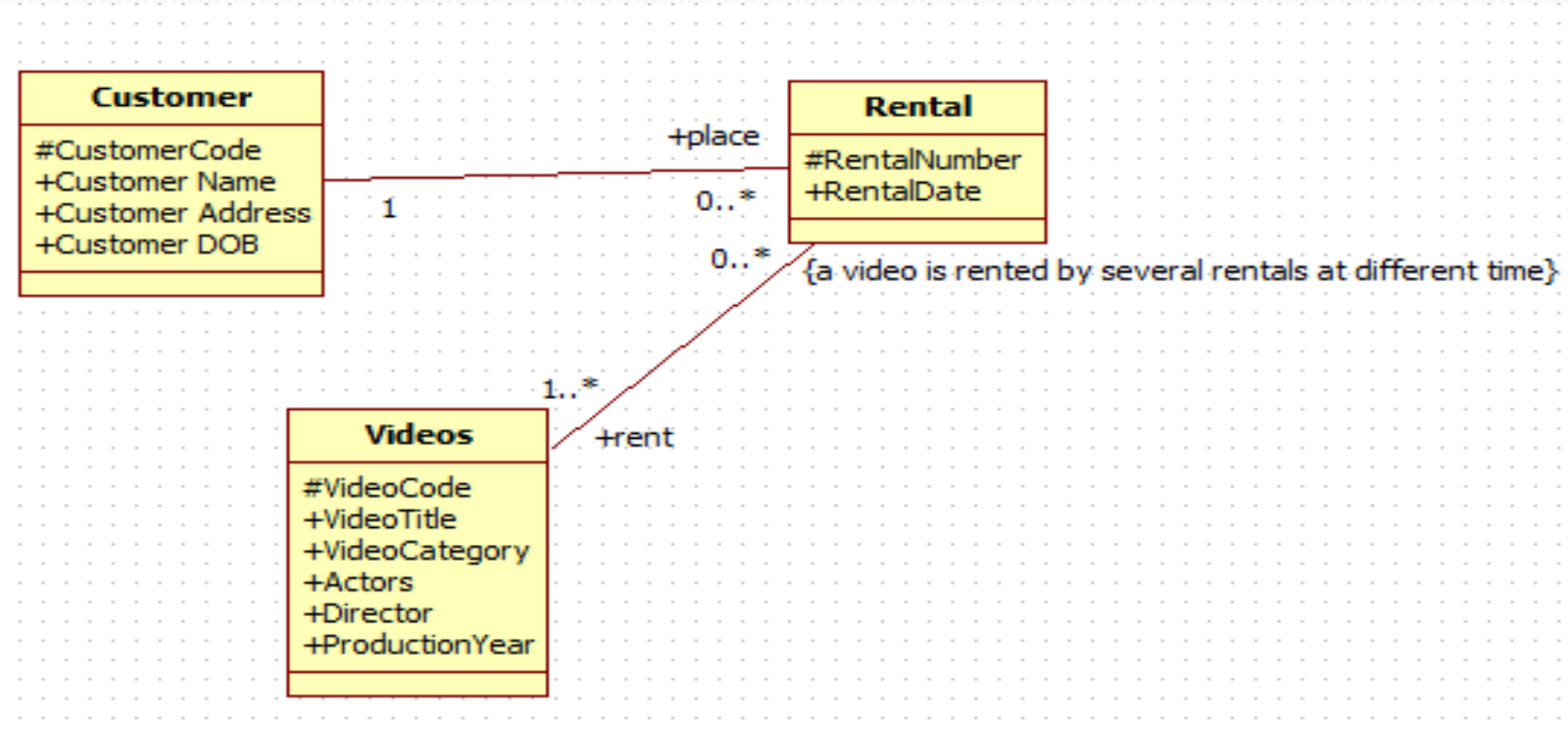
- Association: structural relationship that specifies that object of one thing are connected of objects of another, an object plays a role in an association
- Multiplicity: shows how many objects of a target class (other side of the association) can be associated with a single object of the source class.

Common multiplicities are: 0..1 ; 0..n; 1..1 ; 1..n

Identifying classes and association example

- A customer can place one or several rental orders
- An order is placed by only one customer
- A rental is made on 1 or several videos, a video can be rented with several rentals at different times

Example of Association

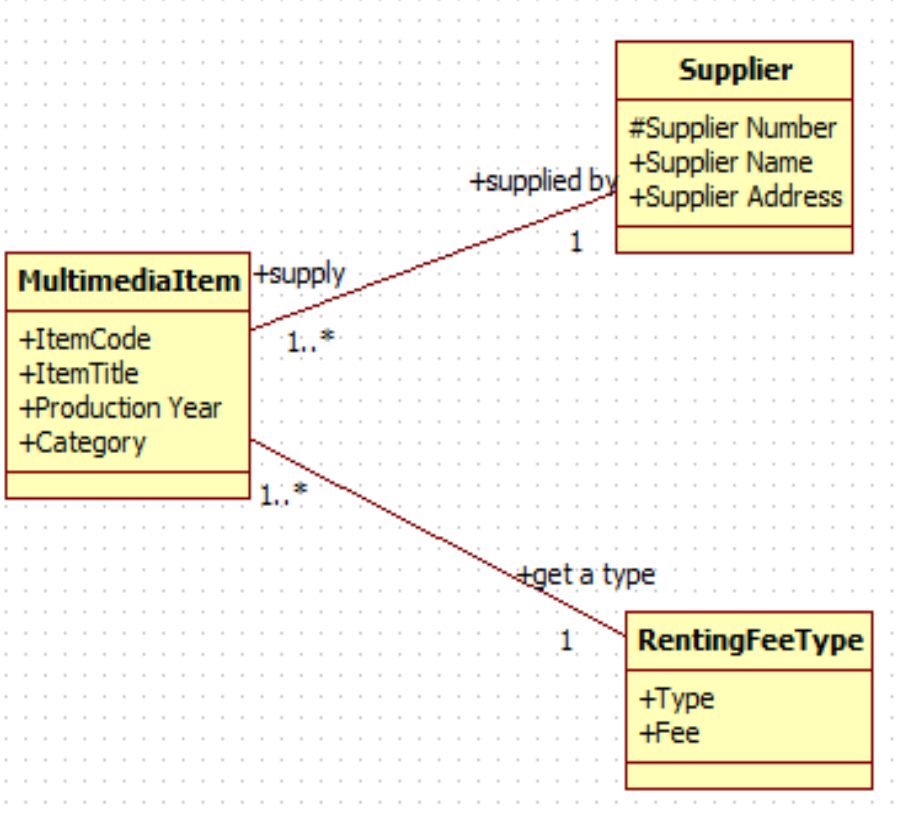


Example

Develop a classes diagram from this functional description:

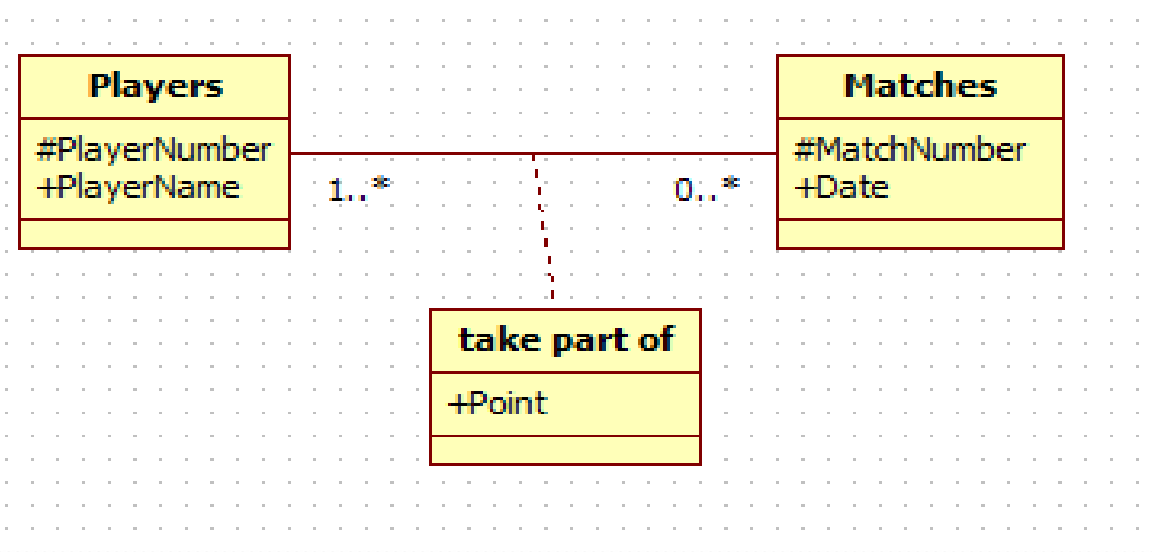
The video store will keep a stock of videotapes, CDs (games and music) and DVD (films, music, and documentary). The inventory is provided by one supplier at the moment, but in the future there will be more suppliers. The store needs to keep track of the supplier of each multimedia item (videotape, CD, DVD) for cases of return/changes. All videotapes and disks are barcoded so that a barcode reader integrated with the system can support the rentals and returns. Each item type has a different renting fee, for instance it is 3€/week for DVD, 1€/week for videotape and 2€/week for a CD.

- Class: MultimediaItems, FeeType, Supplier



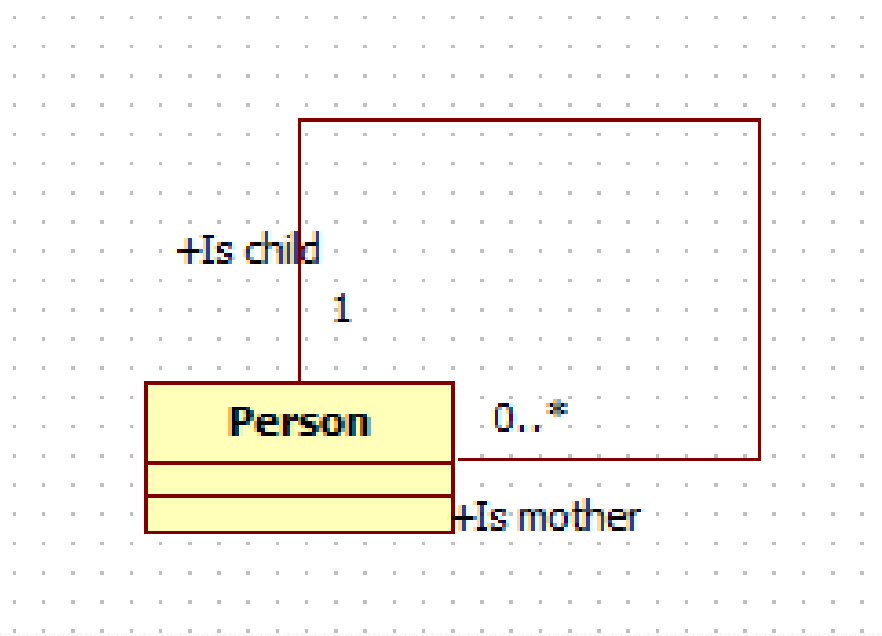
Association Class

- An association class is an association with their own property/attributes



Unary association

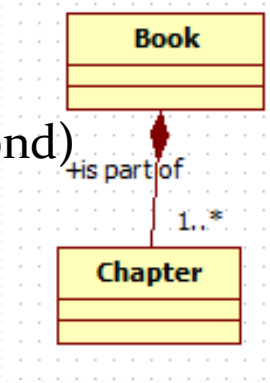
- In a general class Person, a person can be a mother of none or several persons. A person is a child of a person (as mother)



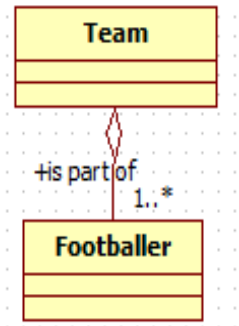
Special types of association

- Aggregation association: A Whole-part relationship between an assembly class and a component class. There is no existence dependency between the two classes.
- Composition association: a form of aggregation with strong ownership and coincident life time as part of the whole, a part cannot exist without the whole.

Composition (solid diamond)



Country, County

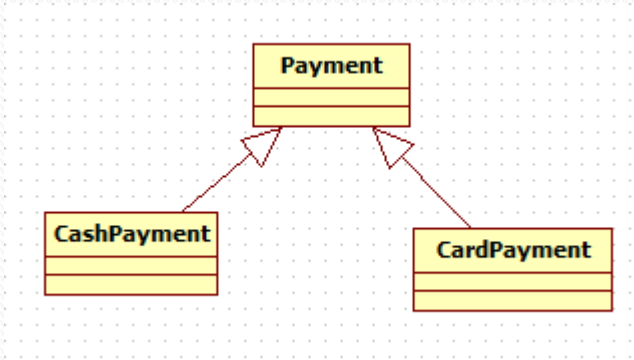


Aggregation

Car, Wheels, Engine

Generalisation

- A taxonomic relationship between a generic class (super class) and specialized class (subclass)
- Objects in subclasses inherit all characteristic from its super class



Another example

We need to develop a software system for the School of Monitoring to support its business activities. The school has many well-qualified instructors. A new client must first attend an interview with an instructor to discuss an appropriate plan of learning. During this interview the client provides personal information including his/her provisional driving license number. Each client is uniquely identified using a client number. A client may request individual lessons or book a block of five, ten or twenty lessons. The more lessons a client books in advance, the less they are charged per hour for a lesson.

An individual lesson is an hour. A lesson is with a particular instructor in a particular car at a given time. After each lesson, the instructor records the progress made by the client and notes the mileage used during the lesson. The school has a pool of cars which are adapted for the purpose of teaching. Each instructor has a sole use of a particular car.

Another example

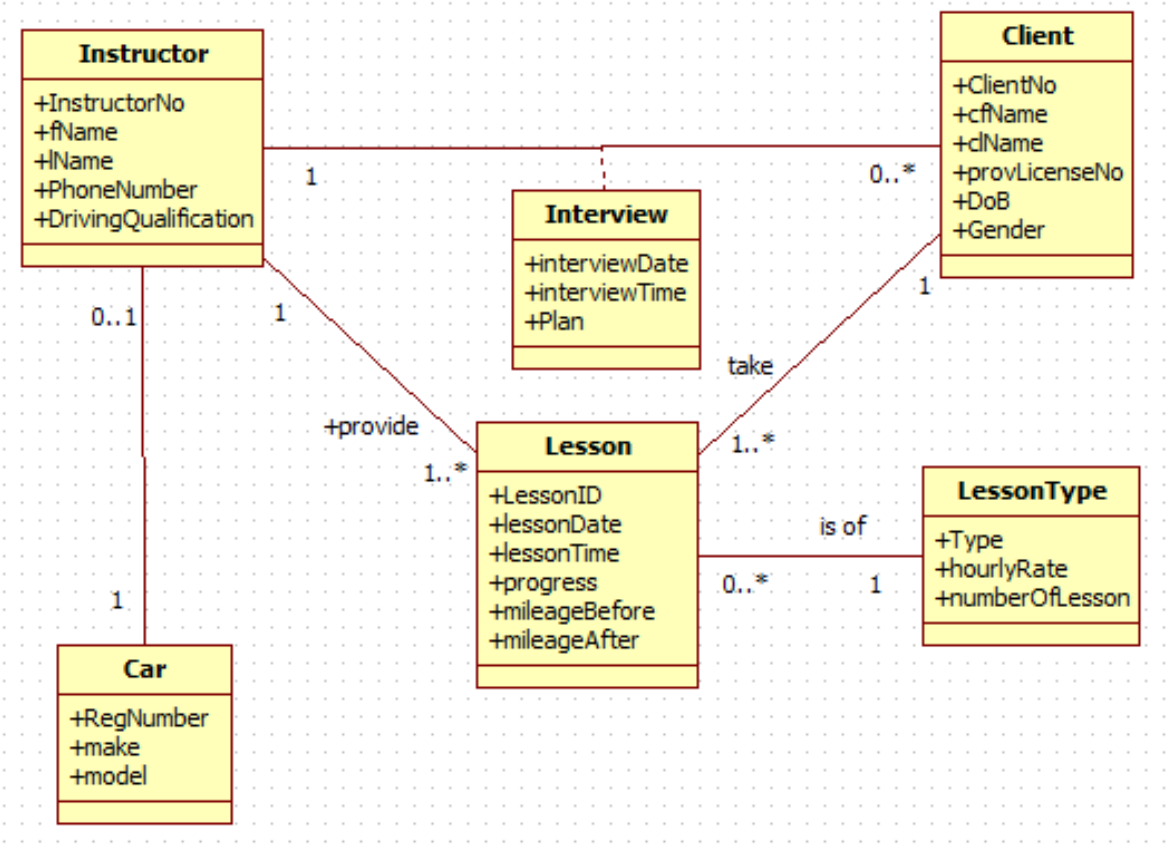
We need to develop a software system for the School of Monitoring to support its business activities. The school has many well-qualified instructors. A new client must first attend an interview with an instructor to discuss an appropriate plan of learning. During this interview the client provides personal information including his/her provisional driving license number. Each client is uniquely identified using a client number. A client may request individual lessons or book a block of five, ten or twenty lessons. The more lessons a client books in advance, the less they are charged per hour for a lesson.

An individual lesson is an hour. A lesson is with a particular instructor in a particular car at a given time. After each lesson, the instructor records the progress made by the client and notes the mileage used during the lesson. The school has a pool of cars which are adapted for the purpose of teaching. Each instructor has a sole use of a particular car.

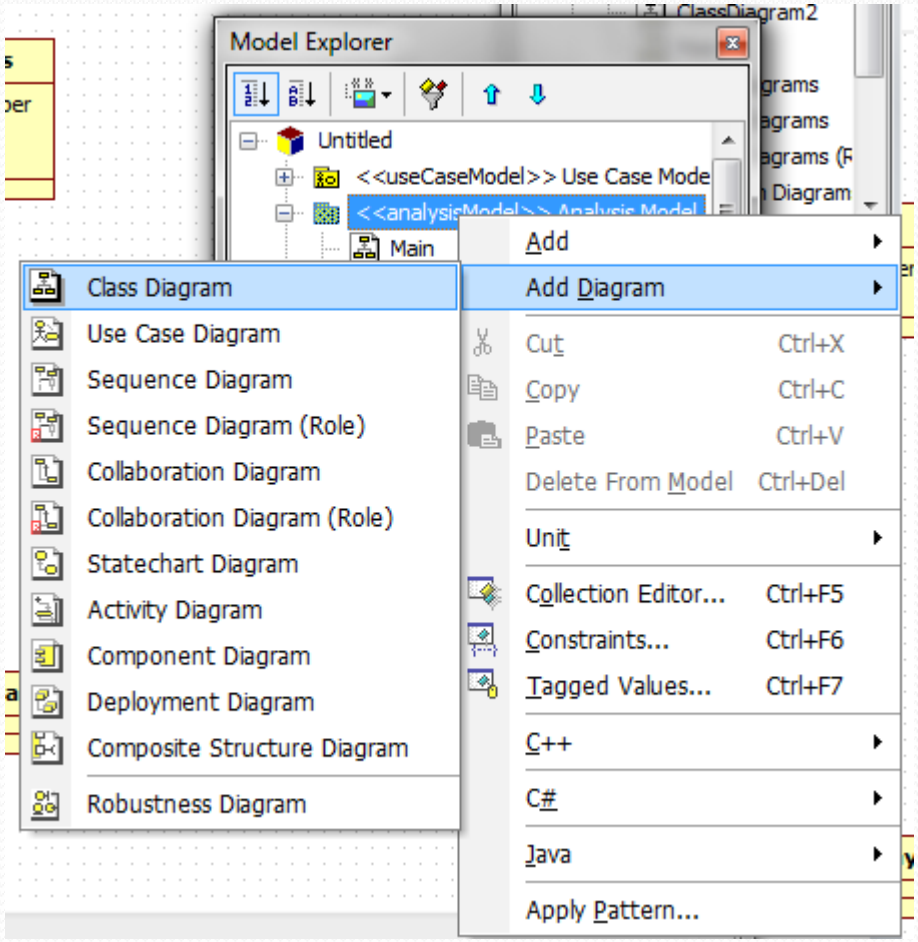
Questions

- Provide the class diagram to represent the data aspect of the system to be developed.
  - Class:
    - Instructor, Client, Car, Lesson, LessonType, Interview

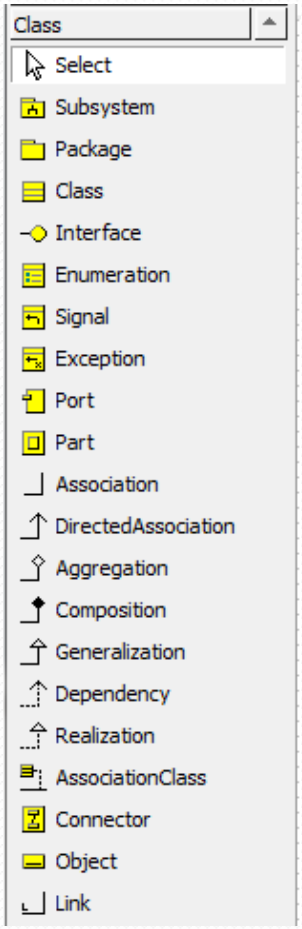
Class diagram



Class modelling with StarUML



Create a Classes diagram



Tool box for Classes diagram

Summary

- System analysis allows to better understand the system and represent the system in technical notations which are close to the design
- There are 3 aspects we need to take into account in analysis: functional aspect, informational and behavioural aspects
- There are 2 approaches for system analysis: structured analysis and OO analysis
- Use cases diagram for functional aspect representation
- Class diagram for information aspect representation