# ID2207 - Modern Methods in Software Engineering

# Tutorial 3

**September 2025**

**Exercise 1 [Entity - Boundary - Control]:** Use the domain knowledge and noun-phrase method to identify the entity, boundary and control objects

**Remember**:
***Entity objects****: Represent the persistent information tracked by the system (Application domain objects, "Business objects")*
***Boundary Objects****: Represent the interaction between the user and the system*
***Control Objects****: Represent the control tasks performed by the system*

| Scenario name | HappyHourMessage |
|---|---|
| Participating actor instances | bob, alice: BarFlies<br>john: Friend |
| Flow of events | 1. Bob and Alice are sitting in their favorite pub and it's happy hour.<br>They want to invite their common friend John who likes cocktails and didn't know about the newly scheduled happy hour. Alice takes out her "mobile phone" and activates the "SMS" function.<br>2. Alice enters John's cell phone number and writes the message about the happy hour into the "SMS text field". She sends the message and waits for John's answer.<br>3. John, who is still at work, is alerted by a sound of his cell phone that a new "SMS" has arrived. He reads the lines from Alice and answers that he will come immediately. He quits working and leaves the office.<br>4. Alice receives John's answer at her "mobile phone". |

Entity: SMS message, Contact information
Boundary: SMS form, Message notification
Control: use case ***SMS function***

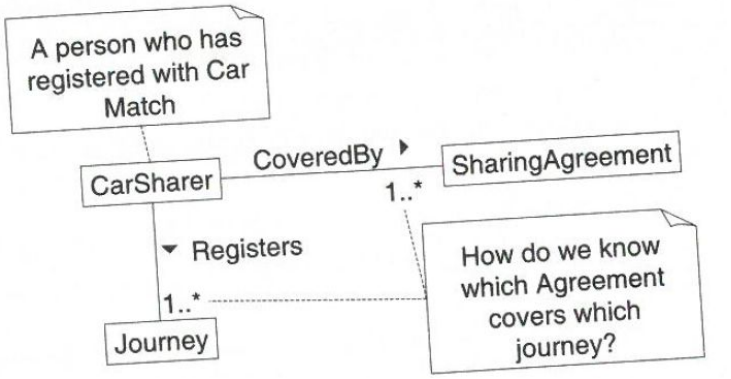
**********

**Exercise 2 [Class Diagrams - Relationships] :** Below is an excerpt from an interview transcript with one of the directors who are setting up CarMatch business. Remember, Mick Perez is the system analyst and Janet Hoffner is the director. Identify any classes and associations mentioned in the transcript:

---

**Mick**: Can we look at the way car sharing is actually organized now. I'd like to find out a bit more about the ideas you work with.
**Janet**: Sure. I guess at the heart of everything there is the car sharer. That's a person who has registered with us so that they can share journeys with other registered car sharers.
**Mick**: Tell me more about how you keep details of the journey that someone wants to share.
**Janet**: Well, a car sharer can actually register several journeys with us. They don't have to limit themselves to just one journey to share.
**Mick**: I guess they would have to want to share at least one journey to be classed as a car sharer though ?
**Janet**: That's right. Remember that they can register as many journeys as they want. When we find other car sharers that want to share a similar journey we match up the sharers and formalize things with a sharing agreement.

---




**A first draft conceptual class diagram**

Consider another extract from the transcript of the interview, and identify any attributes and operations that can be added to the class model.

---

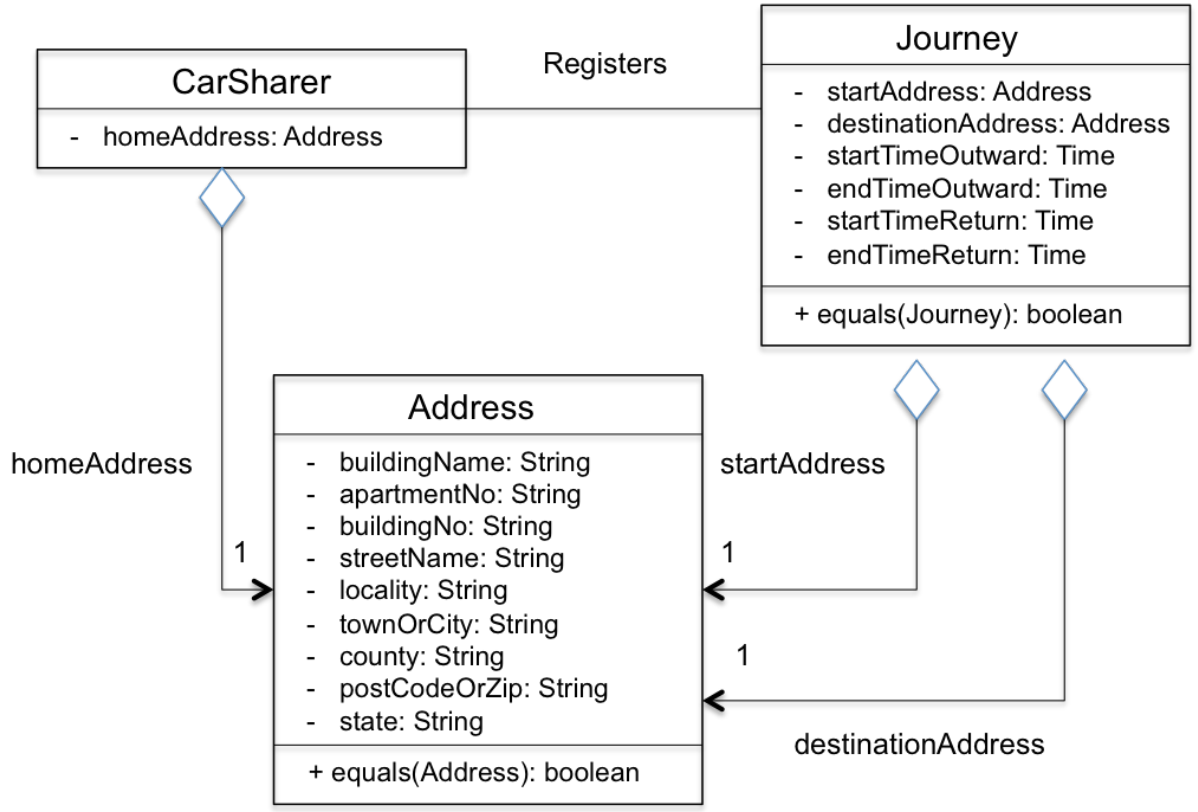
**Mick**: What kind of information do you hold about the journeys that car sharers will register with you ?

**Janet**: Well, I'm sure that you realize it will primarily be 'where from' and 'where to'. We need to know where each journey will start from and where it will end. We will also want to know travel times for each direction of the journey. That is to say, the desired departure and arrival times for both the outward and return journeys.

**Mick**: What kind of things do you need to know about the start and destination of each journey ? Will that information need to be used to match up possible shared journeys ?

**Janet**: Good point. Yes, I guess that however we hold the start and destination address, we will need to be able to use that information to automate the matching of car sharers. The home address of a person might also be used in some journey matching.

---




**Identification and allocation of attributes and operations**

**********

**Exercise 3 [CRC Cards]:**

*Class-Responsibility-Collaborators (CRC) card technique: A CRC card is a small piece of card or paper. The name of the class is written at the top of the card. The responsibilities are written of the left-hand side and on the right hand side the other classes with which the class collaborates with are listed. The goal from this brainstorming technique is to help in identifying the need for new classes or functional responsibilities.*

| Journey | |
|---|---|
| **Responsibilities** | **Collaborations** |
| Check if another Journey is the same as this one | Address supports checking for equality between one address and another |
| Maintain details of a journey | |

**Example of a CRC card**

**<u>Problem description</u>**: A museum requires a system that will enable museum staff to keep track of guided tours of the museum. When a party of visitors arrives, a staff member must be able to record the date, start time and number of visitors undertaking the tour. Also, a staff member must be able to assign a guide to the tour, from available museum tour guides (i.e., those not currently conducting a tour or performing some other duty); the assigned guide will be notified of the assignment. During a tour, its guide must be able to record special incidents that might occur, such as a visitor becoming ill.

Each such incident will be related to the tour on which it occurred and will include a description and a time. Incident reports must be communicated to the Museum's Safety Office.

• create a set of **CRC cards** for the key candidate classes needed to realize the use cases.

**********

| Tour | |
|---|---|
| **Responsibilities** | **Collaborations** |
| 1. Save date, start time and number of visitors in the tour<br>2. Keeps track of id of the assigned guide (those not currently conducting a tour or performing some other duty) ← condition<br>3. Notify guide of assignment | Guide, Incident |

| Guide | |
|---|---|
| **Responsibilities** | **Collaborations** |
| 1. Keeps track of guide information<br>2. Receive notifications of assignment | Tour, Incident |

| Guide | |
|---|---|
| **Responsibilities** | **Collaborations** |
| 1. Save visitor information | Incident |

| Incident | |
|---|---|
| **Responsibilities** | **Collaborations** |
| 1. Each such incident will be related to the tour on which it occurred and will include a description and a time. | Tour, Visitor |

**Exercise 4** [**State Chart Diagrams**]

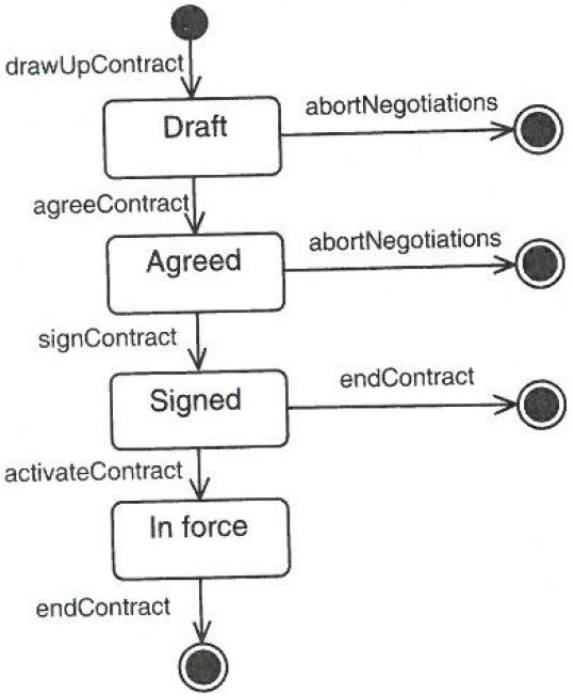
*State machines describe the behavior of dynamic model elements, and are closely related to activity diagrams. Wherease activity diagrams describe flow between areas of work, state machines describe flow between states. State machines are developed as follows:*

- *Identify entities that have complex behaviour*
- *Determine the initial and final states of the entities*
- *Identify the events that affect the entity*
- *Trace the impact of events and identify the intermediate states.*

---

**Problem description**: CarMatch intend that members could obtain a discount for road-pricing schemes. CarMatch negotiates with each authority independently. Each pricing scheme has its own computer system. CarMatch would prefer to take over the billing of their own members and attempt to negotiate this first, but sometimes that is not possible. They will then negotiate discounts for members. Finally, they will agree on the mechanism for exchanging information. If negotiations are successful, then contracts will be drawn up and agreed. Sometimes the contract stage will cause some negotiation. At any time, either party can suspend or cancel negotiations.

---



## *Additional Information:*

- [*Changes in Class Diagrams*] Please remember that the classes shown in the class diagram depend on the phase of the development process. In the analysis phase, the classes apparent in the problem domain are of primary interest (usually only class name is shown without attributes and operations). The end-user stakeholder would recognize many of these problem domain classes. However, as the development moves into the design phase, classes and relationship structures that reflect the solution model will be introduced. These classes (which relate to the interface of the application or to the data management) may not be part of the understanding of the end-user community but they are necessary to produce a well-structured model.

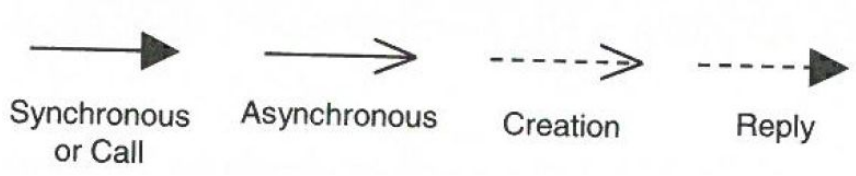
- [*Class diagrams versus Sequence Diagrams*] Class diagrams show the static structure of the classes that make up a system. They don't show how different components of a class model interact with each other. This is the purpose of interaction sequence diagrams. Sequence diagrams are used to show this interaction and emphasize the order of messages over time.

- [*Hint about Class Naming*] By convention a class's name should start with a capital letter and have no spaces between multiple words in the name, and should start each subsequent word with a capital letter. For example: BankAccount. Attributes and operation names should start with small letter.

- [*Association, Aggregation and Composition*] When drawing associations between classes on a class diagram, associations are sometimes created where the name of the association is something like consists of or is made up of. While there is clearly an association between the classes, it is desirable to be able to express the more subtle "objects of this class consist of objects of that class" semantic. This is the purpose of aggregation and composition.
  Aggregation which is a conceptual notion implies a whole-part structure between two classes. On the other hand, composition implies that the life-cycle of the "part" cannot extend beyond the life cycle of the "whole". So, the developer should notice that when the "whole" end is deleted, any "part" components should be deleted as well *programming level*.

- [Notations in Sequence Diagrams] When drawing your sequence diagrams, remember the following:
  a. The lifeline (the vertical dashed line) beings at the point when the object is created. The focus of control represented by the long thin rectangles on the lifeline, shows where an object is active, either because it is performing some action or because it has sent a message to another object, which is carrying out an object on its behalf.
  b.

| | |
|---|---|
| Synchronous | A message is sent by one object to another and the first object waits until the resulting action has completed. This may include waiting for the completion of actions invoked by the second object on other objects. |
| Asynchronous | A message is sent by one object to another but the first object doesn't wait until the resulting action has completed, it carries on with the next step in its own sequence of actions |
| Creation | Represents a message that causes the creation of an object instance to which the message is sent. |
| Reply | Represents the explicit return of control from the object to which the message was sent. |




c. Combined fragments ( alternative (alt),loop and parallel (par))

Where a sequence of messages takes place within an iteration, the messages can be shown grouped together in a combined fragment with the keyword loop and the expression which controls how many times the loop is executed.