Interaction Diagrams:

Interaction diagrams are those which describes the dynamic behavior/view of object oriented system. This is used to describe the interaction between the objects of the system. An interaction is the specification of the way in which messages are sent between objects or class roles in order to perform a task. The basic purpose of the interaction diagram is to capture the dynamic behavior of the system and to show the flow of messages between the system's objects. Moreover, interaction diagrams outline the structural organization of the object and interaction among them.

There are two types of interaction diagrams which are collaboration diagram and sequence diagram. Collaboration diagram shows this interaction in the context of the class roles that participate in the participate in the interaction and show the structural relationship of the classes to one another using association roels. Sequence diagrams are used to show the same interaction as in the collaboration diagram. The focus of this diagram is on the structural association of the objects that are used to send and receive messages ***while the sequence diagram emphasizes on the time sequence of messages*****.** General way to draw the interaction diagram, we have to follow these steps:

- Objects taking part in the interaction.
- Message flows among the objects.
- The sequence in which the messages are flowing.
- Object organization.

Sequence Diagram:

Sequence diagram is one the interaction diagram, which describes the interaction among the objects or classes of the system in term of exchanges of messages over time. Sequence diagram is also called event diagram, and it is a good way of visualizing, and validating various runtime scenarios. In other words, sequence diagram is used to model and document how the system will behave in various scenarios and validate the logic of complex operations and functions. So the sequence diagram comes after the collaboration diagram which shows object instances hat play the role defined in the collaboration diagram. They don't show the structural relationship between objects, but it shows the order of interaction visually by using the vertical axis of the diagram to represent time. *The sequence diagram does highlight some aspects that are not so obvious from the collaboration diagram.*

From the business perspective, company could use such diagrams to communicate how exactly the business presently currently works by illustrating how the different business objects interact. A company's technical staff could utilize sequence diagrams in order to document the behavior of a future system. Besides being used to design new systems, sequence diagrams could also be utilized to document how objects in an existing system currently interact. This type of documentation is very useful when moving a system to another organization or person.

Purpose of Sequence Diagram:

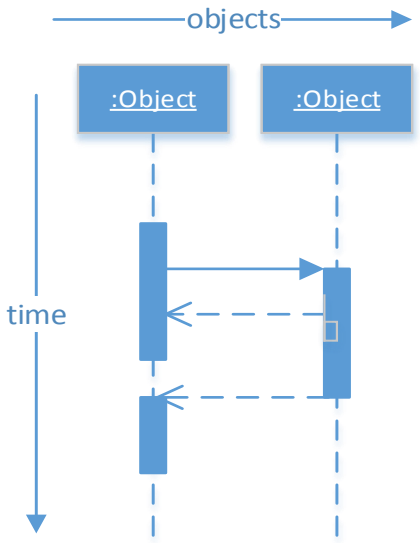
Sequence diagrams are used to model the interaction between object instances in the context of a collaboration. The collaboration is implicit in a sequence diagram, rather than explicitly represented as in a collaboration diagram. Instances are used rather than roles, *but it must be remembered that each*

instance is playing a role that has been defined in a collaboration. Sequence diagram can be used for the following purpose:

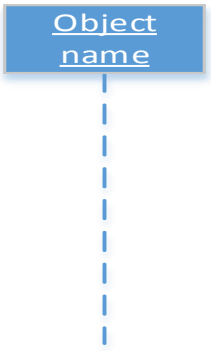
- They are used to model the high-level interaction between active objects in a system.
- They are used to model the interaction between object instances within a collaboration that realizes a use case.
- They are used to model the interaction between objects within a collaboration that realizes an operation.
- They can be used either to model generic interactions, *i.e. showing all possible paths through the interaction* or specific instances of an interaction, *i.e. showing just one path through the interaction.*
- A company's technical staff could utilize sequence diagrams in order to document the behavior of a future system.
- Sequence diagrams can be used to document how objects in an existing (i.e. "legacy") system currently interact. This documentation is very useful when transitioning a system to another person or organization.
- Sequence diagram can be used when you want to look up at the behavior of several objects within a single use case.

Notations:

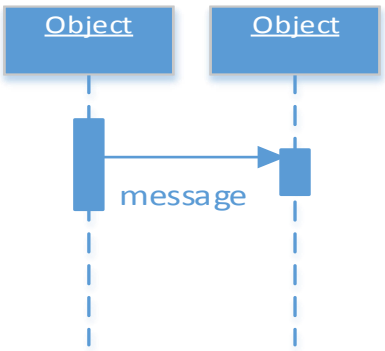
- **Objects:** In sequence diagram, each class or object in the interaction is represented by its named icon along the top of the diagram. Object instances are arranged horizontally across the page, and time runs vertically down the page as shown in the figure.



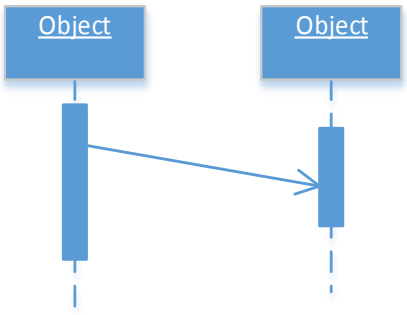
- **Lifeline:** The lifeline is used to represent the time during which the object exists. If a particular object exist before an interaction and continues to exist after the interaction ends, then its lifeline runs from top to bottom of the diagram. As object instances are on the top of the diagram with a dashed vertical line. This dashed line is the lifeline of the object as shown in figure.



- **Messages:** In a sequence diagram, a message or stimulus is shown using an arrow going from the sender to the receiver. In asynchronous interaction, one object can send a message to another object and the first object, then carries on with its next task without waiting for a reply; in a synchronous interaction each object waits for a response.



Message sending is usually assumed to be virtually instantaneous. However, in a situation where the message is sent over a communication link and a significant amount of time elapses between the sending and receiving of the message, the arrow can be shown slanting down the page like in the figure.



There are different types of messages between objects for interaction in the sequence diagram. These are synchronous, asynchronous, self message and return messages.

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 action invoked by the second object on other objects. Synchronous message is represented by arrow given in the figure below.

Asynchronous: A message is sent by one object to another but the first object does not wait until the resulting action has completed, it carries on with the next step in its own sequence of actions. Aysnchornous message can be represented as follows:

Return: A message which represents the explicit return of control from the object to which the message was sent. Return message can be drawn as follows:

Self Message: A message which represents a recursive call of an operation, or one method calling another method belonging to the same object. Self message is represented as:

How to make Sequence Diagram:

Sequence diagram can be drawn to model high-level interaction between users of the system and the system, between system and other system, or between subsystems , so it is also known as a *system sequence diagram.*

- Decide on the context of the interaction, i.e. system, subsystem, use case or operation.
- Identify the structural elements, i.e. class or objects that are necessary to carry out the functionality of the use case or operation.
- Consider the alternative scenarios that may be required.

1. **Decide on Context:** As we know that sequence diagram can model interaction at the system, subsystem, use case or operational level. The Stage in the development of the project and the task being undertaken will determine which is to be modelled.

2. **Identify Structural Elements:** The first step is to identify elements that will participate in the interaction sequence diagram. The elements will include classes or objects.
3. **Consider Alternative Scenarios:** If sequence diagram is created by using "Register Carsharer" use case where the journey is created, then we have to look up if there is any other use case in which journey is created. We also have to consider the different possible paths through the operation in response to different inputs.

Importance of Sequence Diagram:

- Depict object interactions in a given scenario identified for a given Use Case
- Specify the messages passed between objects using horizontal arrows including messages to/from external actors
- Time increases from Top to bottom

Points to Remember:

- Sequence diagrams model object interactions with an emphasis on time ordering
- Method call lines
    - Must be horizontal!
    - Vertical height matters!

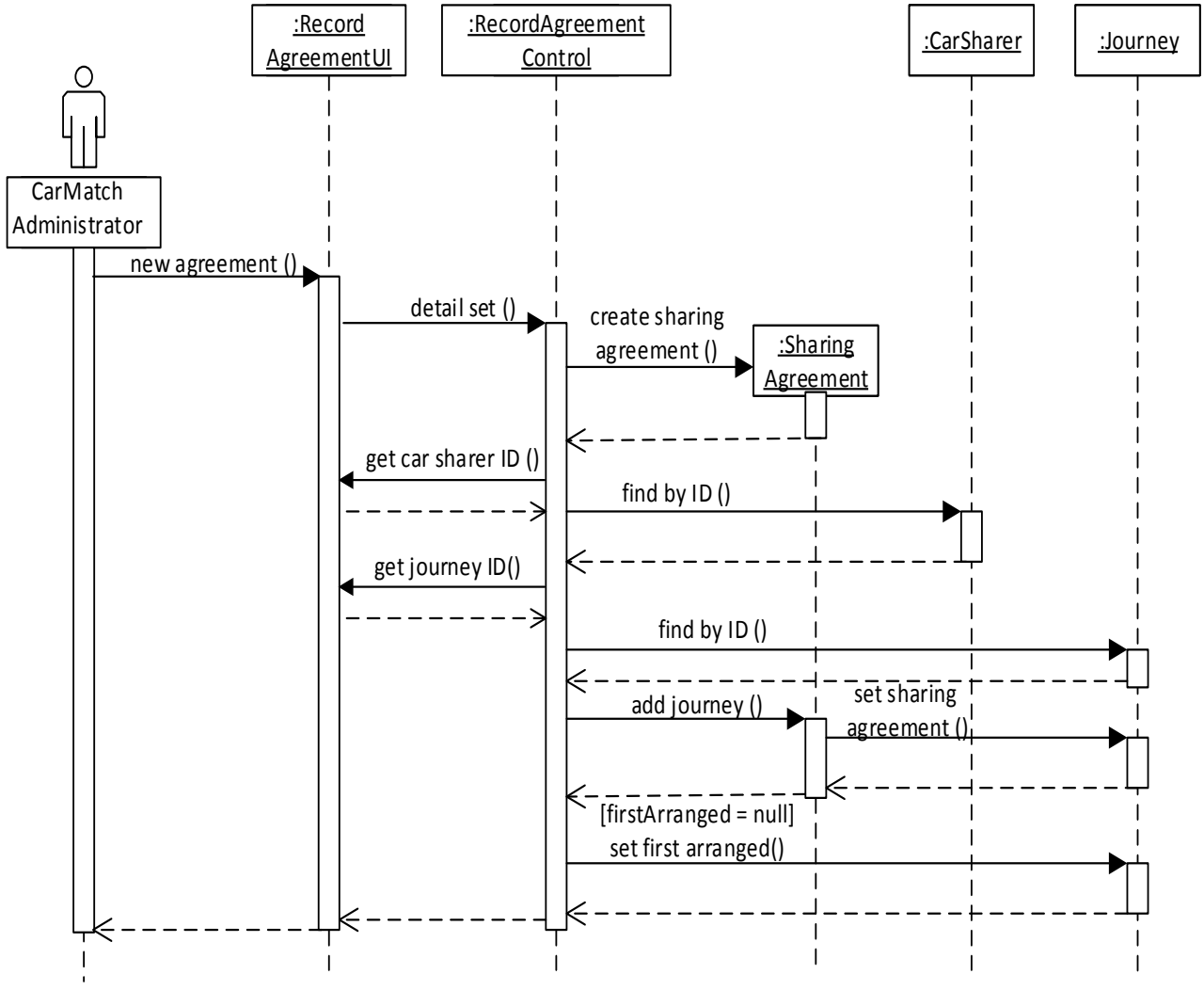
      "Lower equals Later"
    - Label the lines
- Lifeline – dotted vertical line
- Execution bar – bar around lifeline when code is running
- Arrows
    - Synchronous call (you're waiting for a return value) – triangle arrowhead
    - Asynchronous call (not waiting for a return) – open arrow-head
    - Return call – dashed line

Sequence diagrams can be somewhat close to the code level. So why not just code up that algorithm rather than drawing it as a sequence diagram?

- a good sequence diagram is still a bit above the level of the real code (not all code is drawn on diagram)

- sequence diagrams are language-agnostic (can be implemented in many different languages
- non-coders can do sequence diagrams
- easier to do sequence diagrams as a team
- can see many objects/classes at a time on same page (visual bandwidth)

CarMatch Case Study Sequence Diagram:



Recommended Links:

http://www.tutorialspoint.com/uml/uml_interaction_diagram.htm

http://www.sparxsystems.com/resources/uml2_tutorial/uml2_sequencediagram.html

http://www.smartdraw.com/sequence-diagram/

http://creately.com/diagram-type/article/understanding-basics-sequence-diagrams