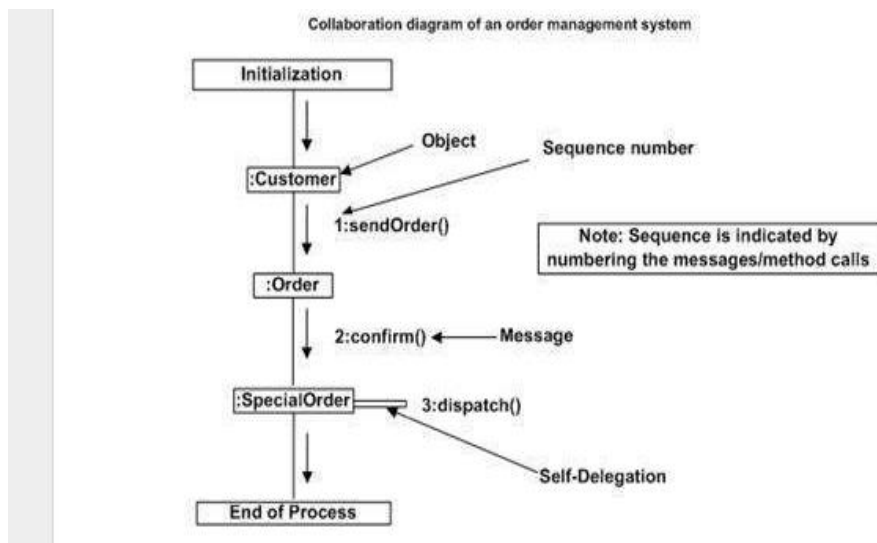


Unit-V

Communication Diagram:

UML communication diagrams, like the sequence diagrams - a kind of interaction diagram, shows how objects interact. A communication diagram is an extension of object diagram that shows the objects along with the messages that travel from one to another. In addition to the associations among objects, communication diagram shows the messages the objects send each other.



Purpose of Communication Diagram

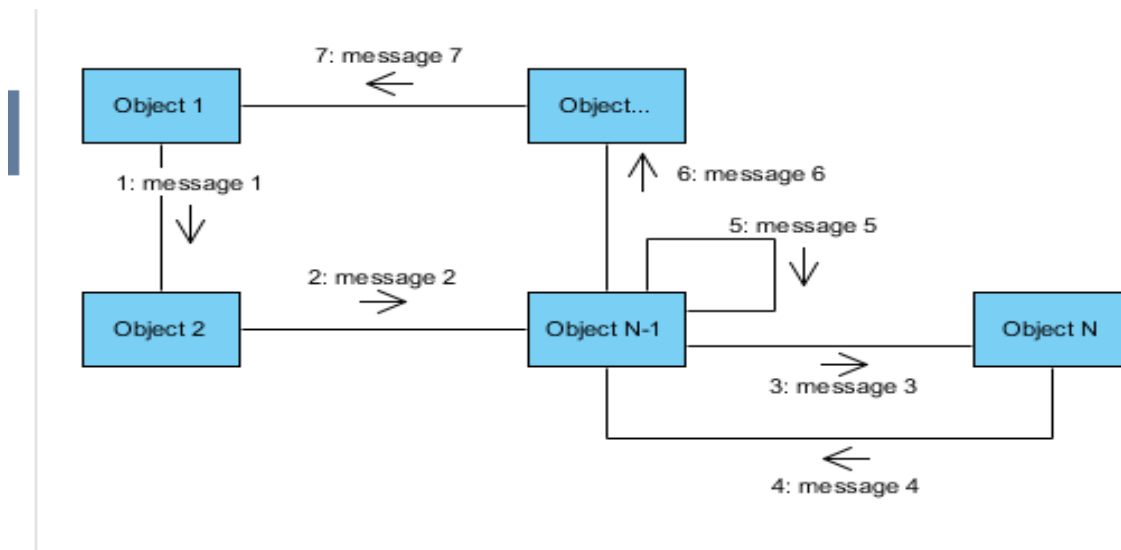
- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the passed messages between objects and roles within the collaboration scenario
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use cases

Communication Diagram at a Glance

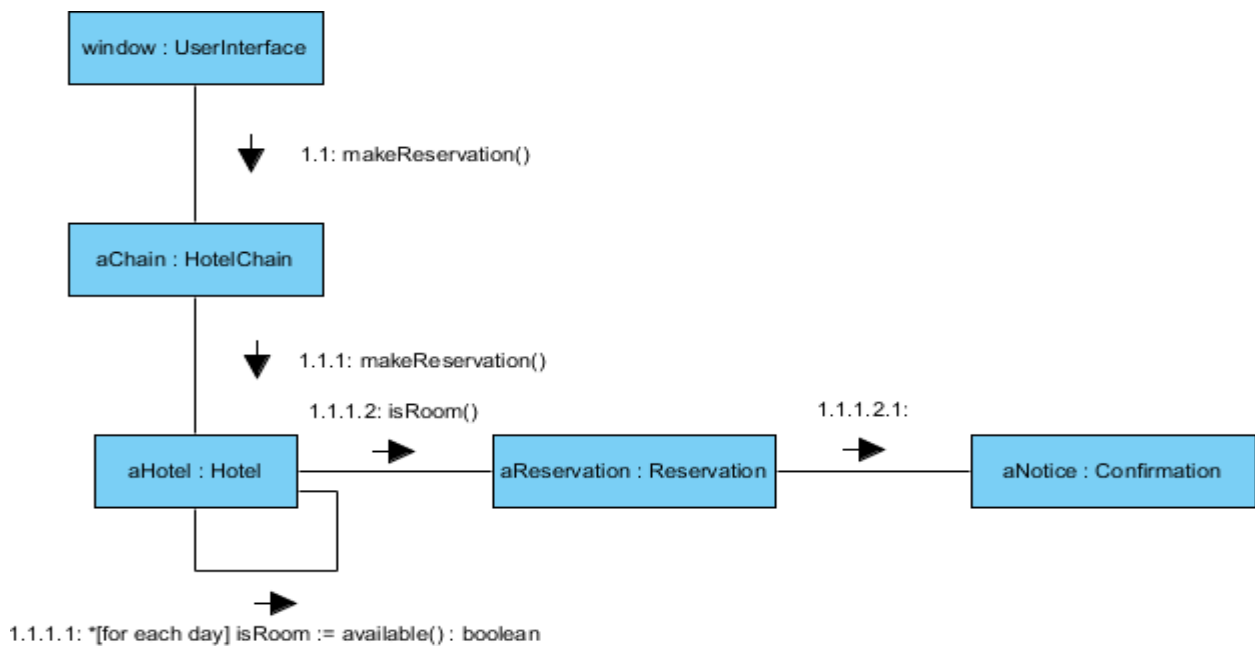
In the example of the notation for a communication diagram, objects (actors in use cases) are represented by rectangles. In the example (generic communication diagram):

- The objects are Object1, Object2, Object..., ObjectN-1 ..., and ObjectN.
- Messages passed between objects are represented by labeled arrows that start with the sending object (actor) and end with the receiving object.
- The sample messages passed between objects are labeled 1: message1, 2: message2, 3: message3, etc., where the numerical prefix to the message name indicates its order in the sequence.
- Object1 first sends Object2 the message message1, Object2 in turn sends ObjectN-1 the message message2, and so on.

- Messages that objects send to themselves are indicated as loops (e.g., message message5).



Communication Example - Hotel Reservation



- Each message in a communication diagram has a sequence number.
- The top-level message is numbered 1.
- Messages sent during the same call have the same decimal prefix, but suffixes of 1, 2, etc. according to when they occur.

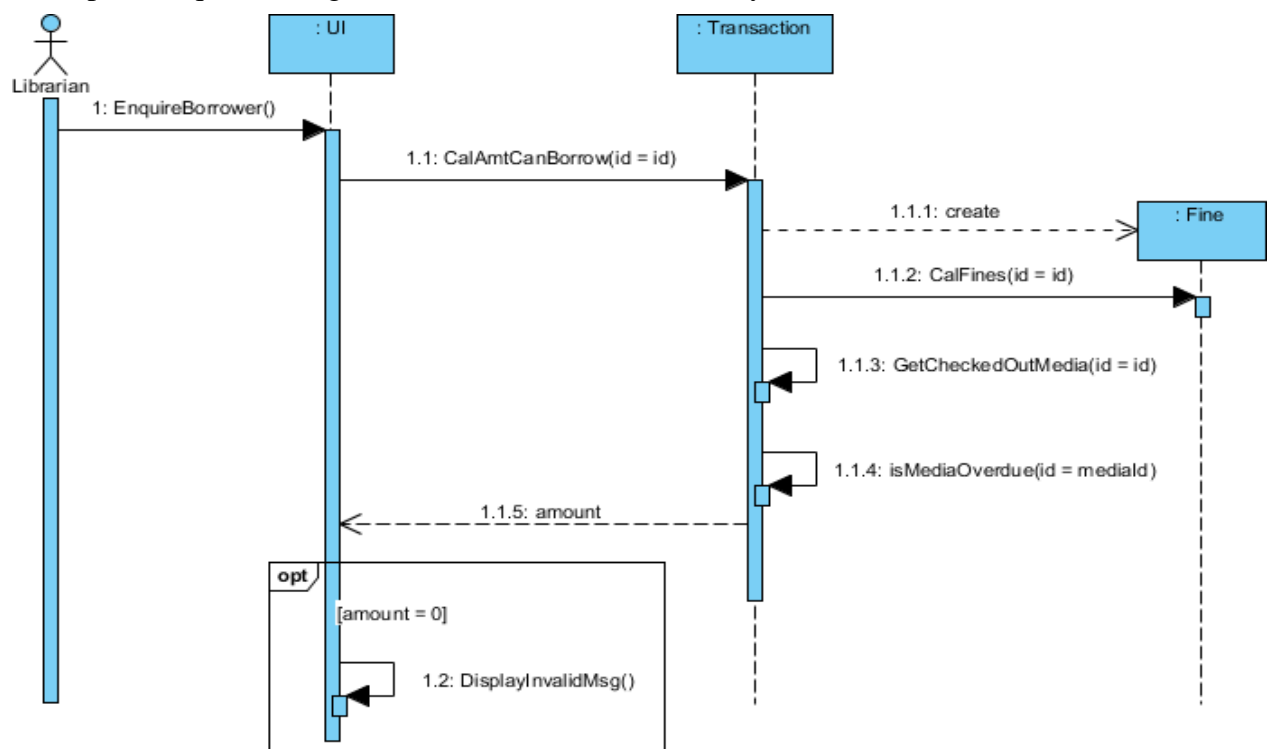
Communication Diagram vs Sequence Diagram

The communication diagram and the sequence diagram are similar. They're semantically equivalent, that is, they present the same information, and you can turn a communication to a

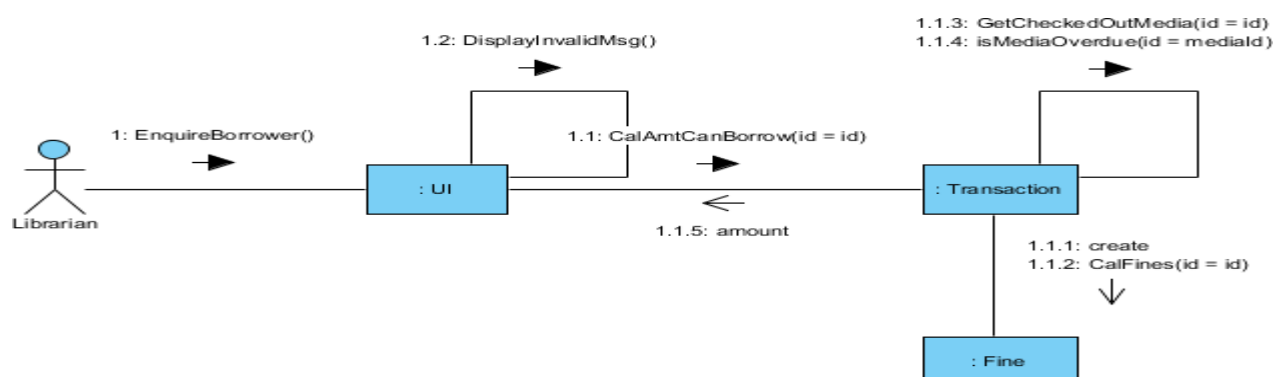
sequence diagram and vice versa. The main distinction between them is that the communication diagram arranged elements according to space, the sequence diagram is according to time.

Of the two types of interaction diagrams, sequence diagrams seem to be used far more than communication diagrams. So, why would you use communication diagrams? First of all, they are very useful for visualizing the relationship between objects collaborating to perform a particular task. This is difficult to determine from a sequence diagram. In addition, communication diagrams can also help you determine the accuracy of your static model (i.e., class diagrams).

Example - Sequence diagram vs Communication (Library Item Overdue)



If you open this sequence diagram in Visual Paradigm you can automatically generate the communication diagram shown in figure below:



Note: If you compare the two diagrams, you'll see they both contain objects and messages. It also becomes clear that it's much easier to determine the time ordering of messages by looking at the sequence diagram and it's easier to see the relationships between objects by looking at the communication diagram.

Communication Diagram Elements:

Objects: participating in a collaboration come in two flavors: supplier and client.

- Supplier objects are the objects that supply the method that is being called, and therefore receive the message.
- Client objects call methods on supplier objects, and therefore send messages.

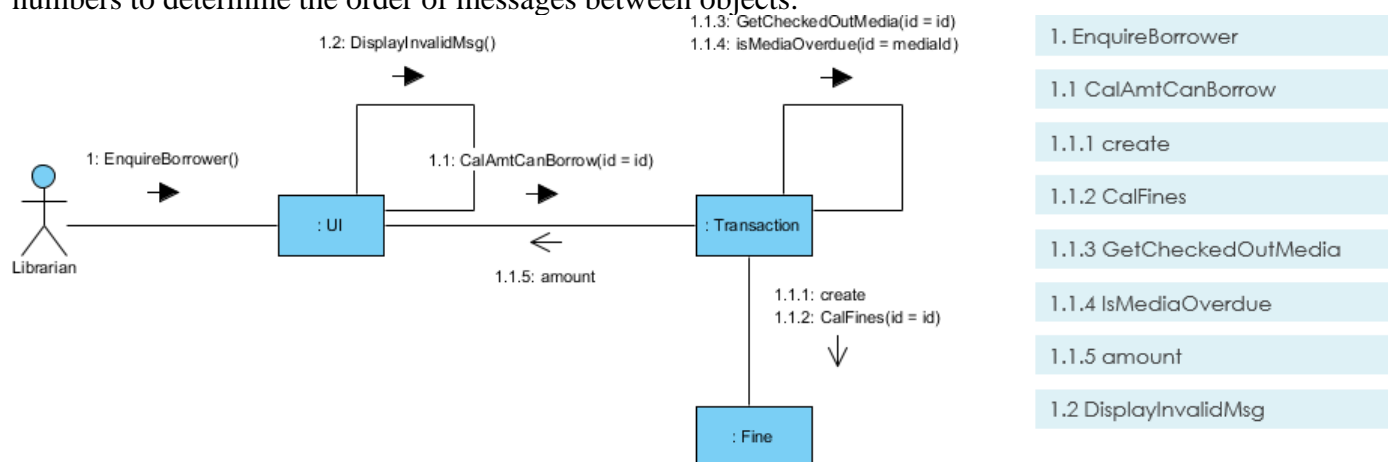
Links:

- The connecting lines drawn between objects in a communication diagram are links.
- These links are what set communication diagrams apart from sequence diagrams. They enable you to see the relationships between objects.
- Each link represents a relationship between objects and symbolizes the ability of objects to send messages to each other.
- If an object sends messages to itself, the link carrying these messages is represented as a loop icon. This loop can be seen on both the UI object and the Transaction object.

Messages: in communication diagrams are shown as arrows pointing from the Client object to the Supplier object. Typically, messages represent a client invoking an operation on a supplier object. They can be modeled along with the objects in the following manner:

- Message icons have one or more messages associated with them.
- Messages are composed of message text prefixed by a sequence number.
- This sequence number indicates the time-ordering of the message.

For example, in the communication diagram in the figure below, you can follow the sequence numbers to determine the order of messages between objects:



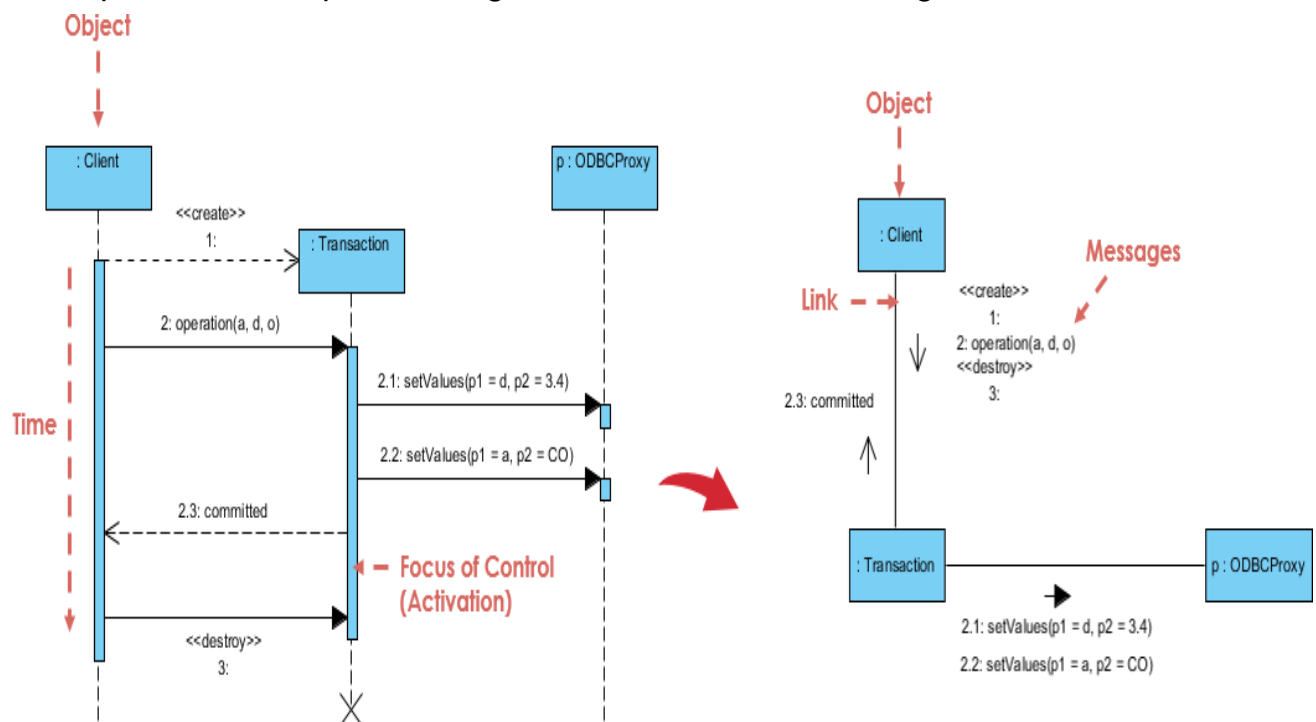
Understanding the Numbering of Messages in Communication Diagram

- The first message in a communication diagram is always numbered 1, the second is 2, and so on.
- You can indicate that a message is nested under a parent message by adding a decimal point and incremental digits to the parent's sequence number.

For example:

Based on the example above, the "CalAmtCanBorrow" message is the first nested message under "EnquireBorrower" and is given the sequence number 1.1. The second nested message under "EnquireBorrower" is "DisplayInvalidMsg", so it's given a sequence number of 1.2.

Example - From Sequence Diagram to Communication Diagram



Note that:

- Focus of control: also called execution occurrence/activation. It shows as tall, thin rectangle on a lifeline that represents the period during which an element is performing an operation.
- The top and the bottom of the rectangle are aligned with the initiation and the completion time respectively.
- In communication diagram focus of control is explicit and thus, can be represented by the message nest numbering.

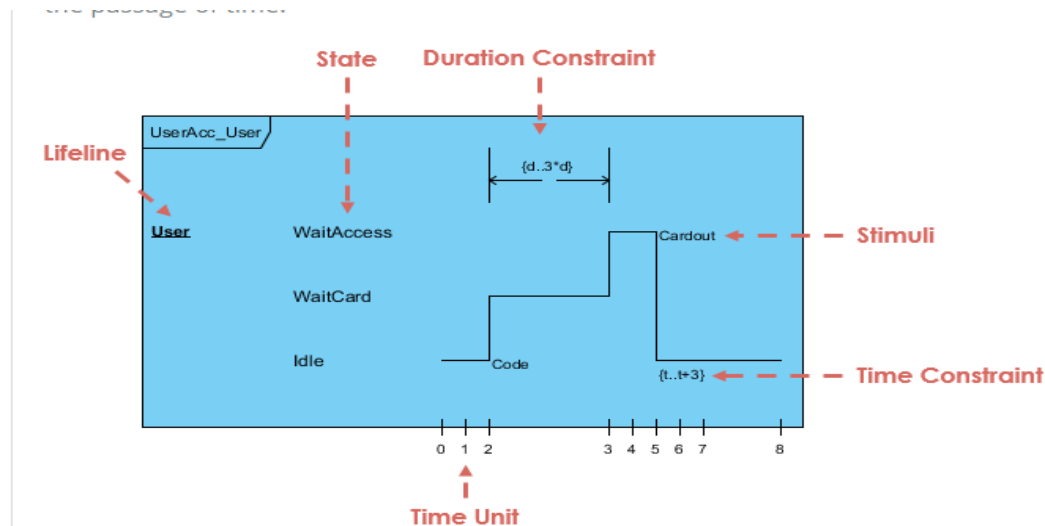
Timing Diagram:

Timing diagrams are UML interaction diagrams used to show interactions when a primary purpose of the diagram is to reason about time.

Timing diagrams focus on conditions changing within and among lifelines along a linear time axis. Timing Diagrams describe behavior of both individual classifiers and interactions of classifiers, focusing attention on time of occurrence of events causing changes in the modeled conditions of the Lifelines.

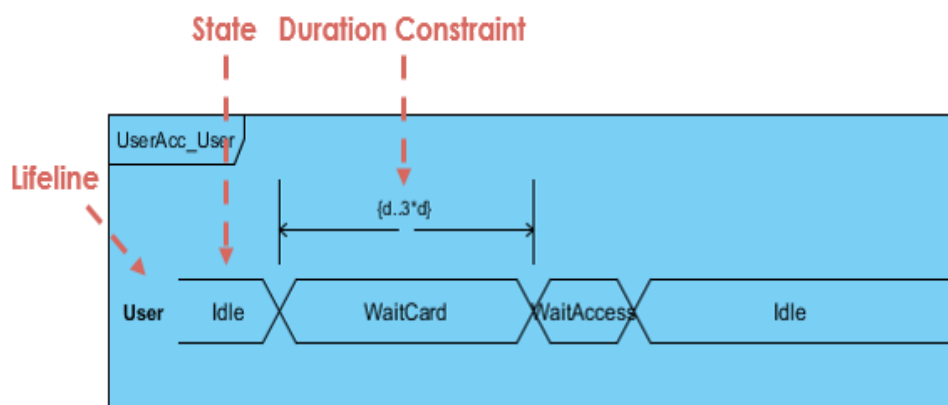
State Timeline Representation

Changes from one **state** to another are represented by a **change in the level of the lifeline**. For the period of time when the object is a given state, the timeline runs parallel to that state. A change in state appears as a vertical change from one level to another. The cause of the change, as is the case in a state or sequence diagram, is the receipt of a message, an event that causes a change, a condition within the system, or even just the passage of time.



Value lifeline Representation

The figure below shows an alternative notation of UML Timing diagram. It shows the state of the object between two horizontal lines that cross with each other each time the state changes.

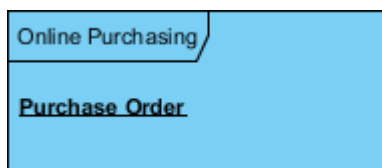


Basic Concepts of Timing Diagrams

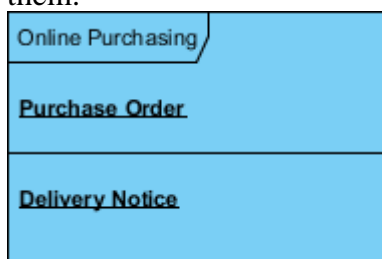
Major elements of timing UML diagram - lifeline, timeline, state or condition, message, duration constraint, timing ruler.

Lifeline:

A lifeline in a Timing diagram forms a rectangular space within the content area of a frame. Lifeline is a named element which represents an individual participant in the interaction. It is typically aligned horizontally to read from left to right.

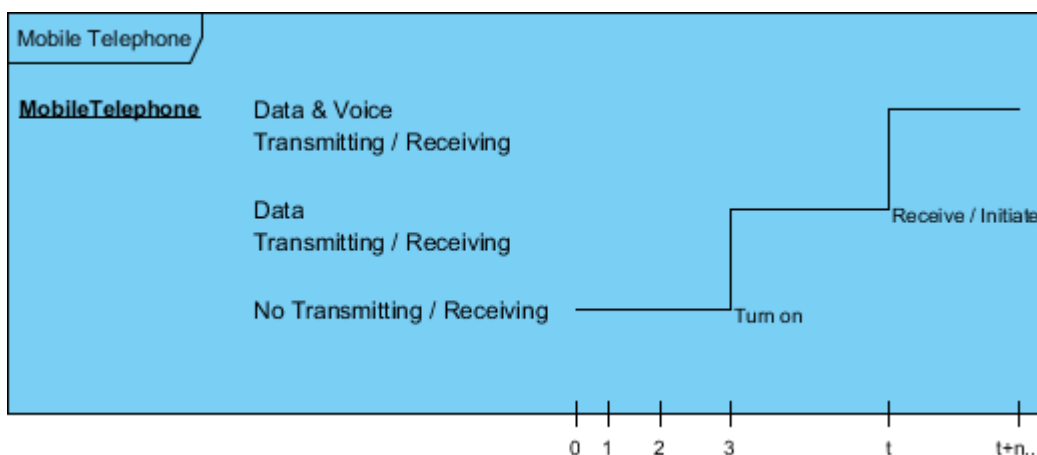


Multiple lifelines may be stacked within the same frame to model the interaction between them.



State Timeline in Timing Diagram

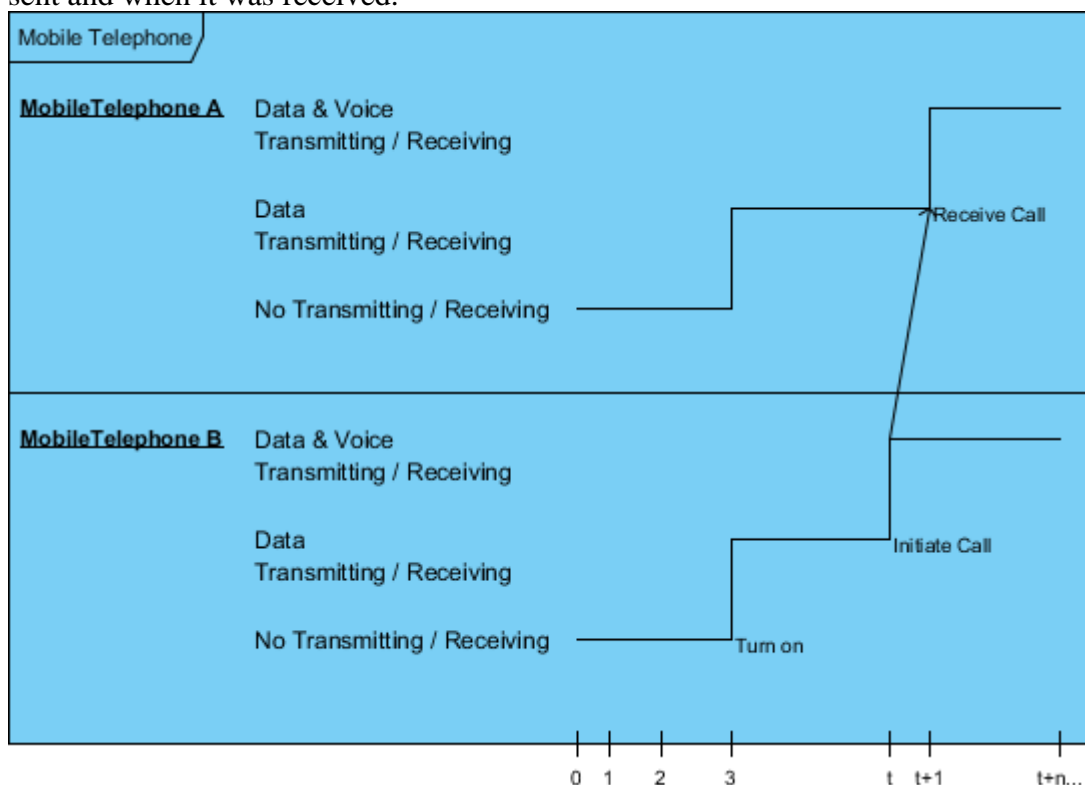
A state or condition timeline represents the set of valid states and time. The states are stacked on the left margin of the lifeline from top to bottom.



The cause of the change, as is the case in a state or sequence diagram, is the receipt of a message, an event that causes a change, a condition within the system, or even just the passage of time.

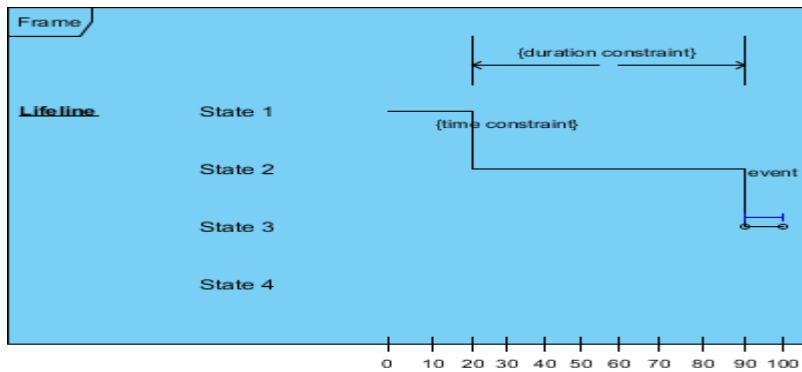
Multiple Compartments

It is possible to stack several life lines of different objects in the same timing diagram. One life line above the other. Messages sent from one object to another can be depicted using simple arrows. The start and the end points of each arrow indicate when each message was sent and when it was received.



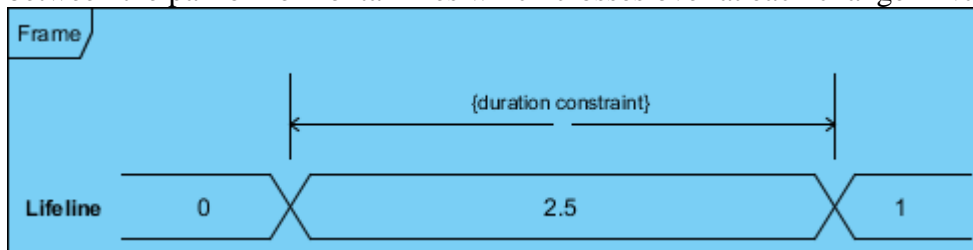
State Lifeline

A state lifeline shows the change of state of an item over time. The X-axis displays elapsed time in whatever units are chosen while the Y-axis is labelled with a given list of states. A state lifeline is shown below:



Value Lifeline

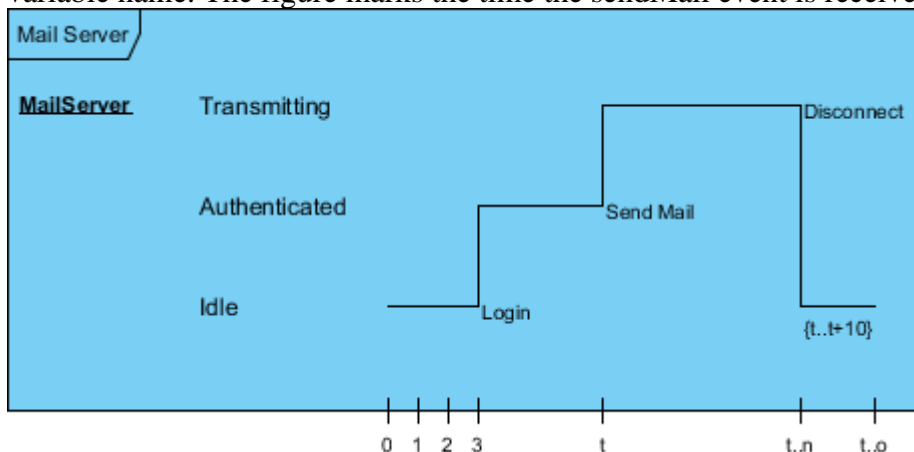
A value lifeline shows the change of value of an item over time. The X-axis displays elapsed time in whatever units are chosen, the same as for the state lifeline. The value is shown between the pair of horizontal lines which crosses over at each change in value.



Timeline and Constraints

We can use the length of a timeline to indicate how long the object remains in a particular state by reading it from left to right. To associate time measurements, you show tick marks online the bottom part of the frame.

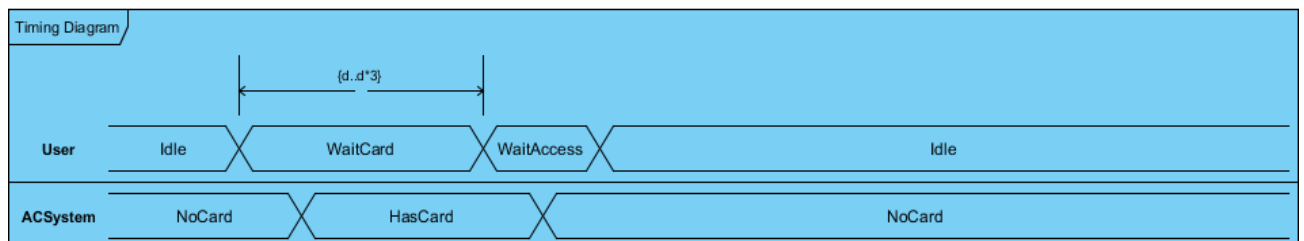
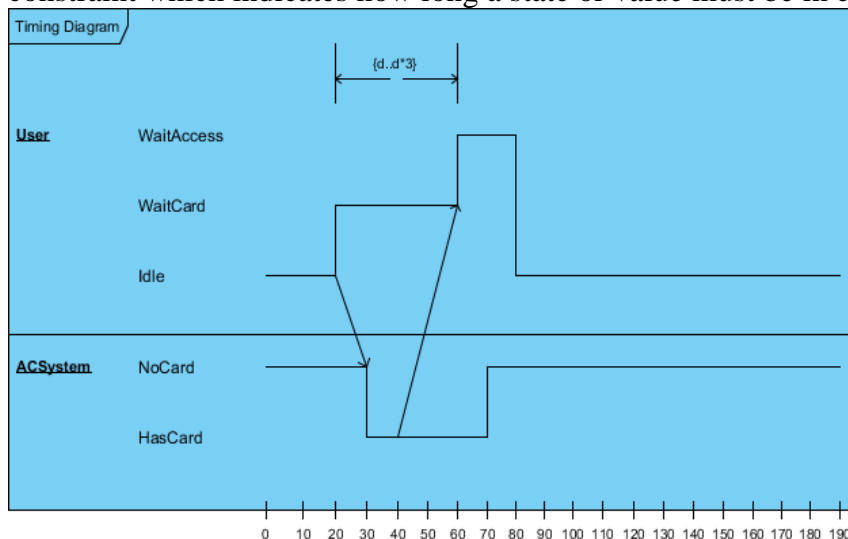
The example below shows that the Login event is received three time units after the start of the sequence. To show relative times, you can mark a specific instance in time using a variable name. The figure marks the time the sendMail event is received as time



You can use relative time marks in constraints to indicate that a message must be received within a specified amount of time.

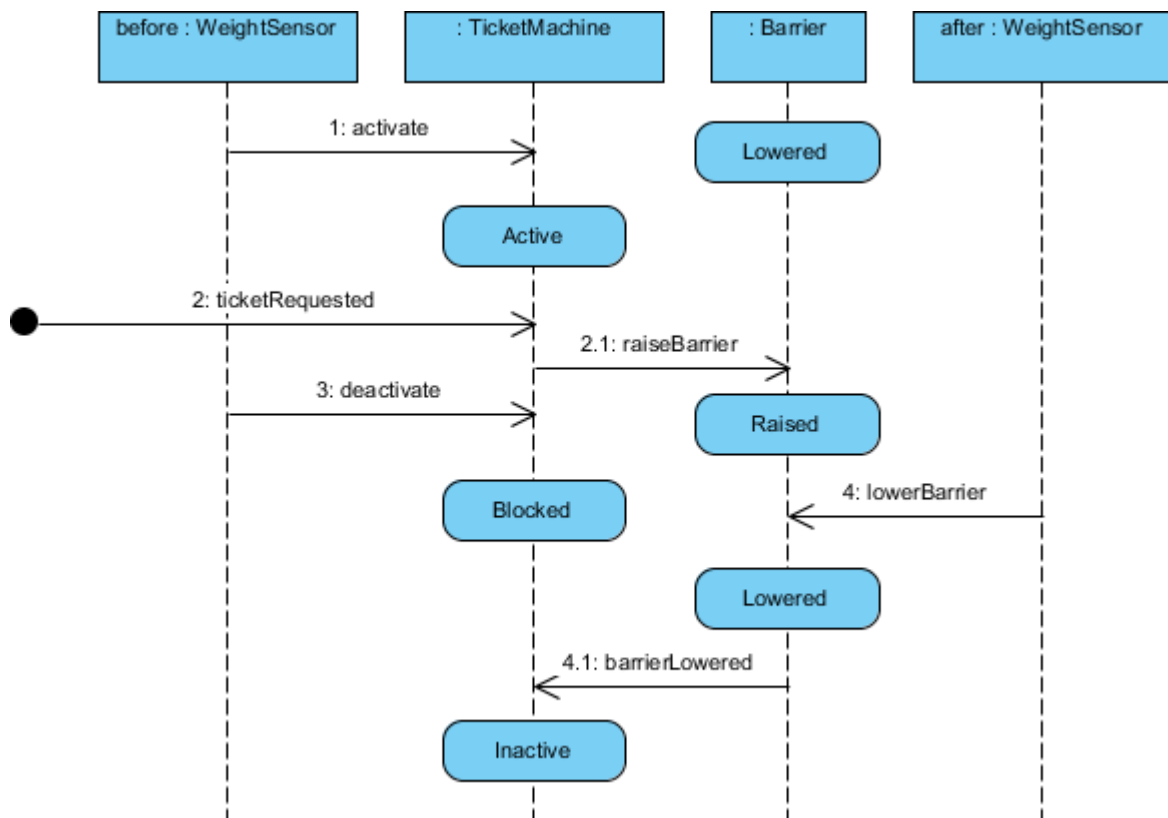
State and Value Lifeline Side-by-Side

State and Value Lifelines can be put one after the other in any combination. Messages can be passed from one lifeline to another. Each state or value transition can have a defined event, a time constraint which indicates when an event must occur, and a duration constraint which indicates how long a state or value must be in effect for.

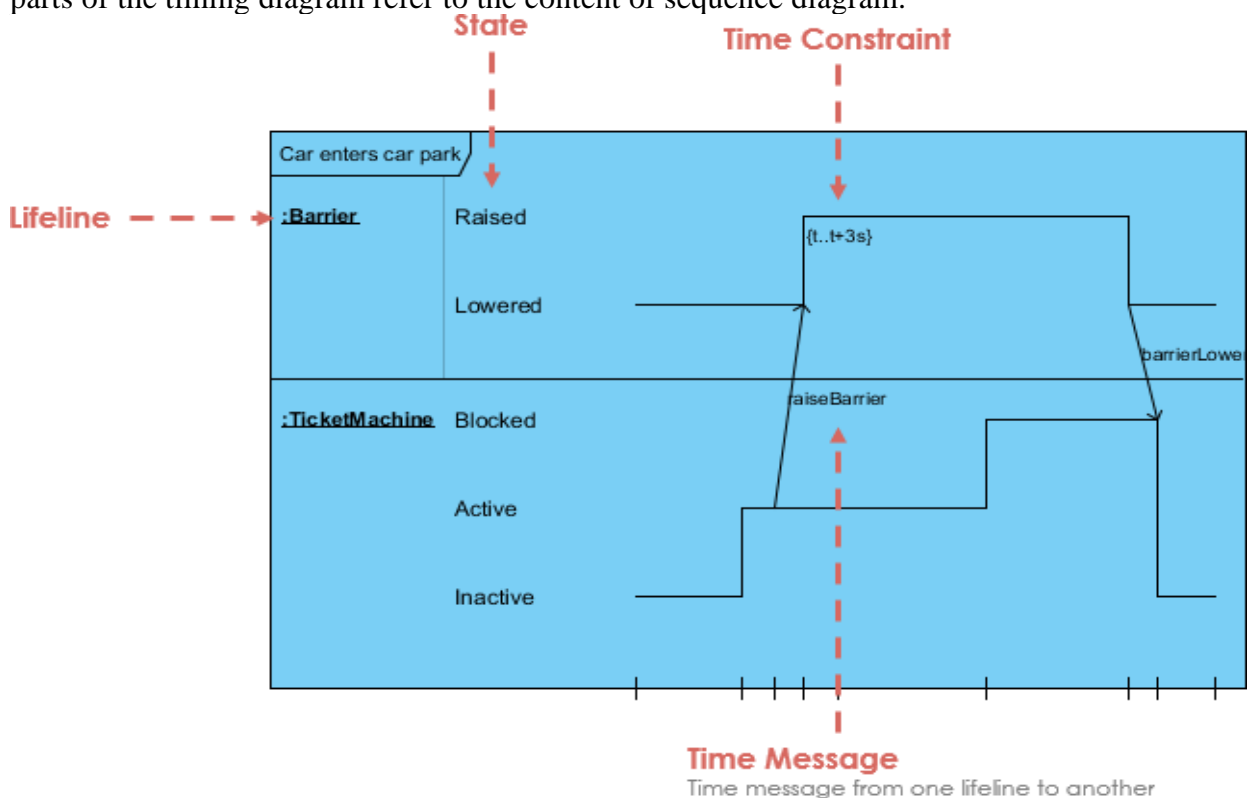


Model Consistency among Interaction Diagrams

Timing diagram should always be consistent with the relevant sequence diagram and state machine. To do this we can attach states in the lifeline for each of the objects in the sequence diagram. We can then derive the corresponding timing diagram much easier by inspecting the message passing between the objects against the states attached in the lifeline. The Carpark example below shows the model consistency between two interaction diagrams.



The figure above shows a sequence diagram of the car park example, while the figure below presents the corresponding timing diagram of the car park example. The various parts of the timing diagram refer to the content of sequence diagram.



Interaction Overview Diagram:

UML Interaction Overview Diagrams provide a high level of abstraction an interaction model. It is a variant of the Activity Diagram where the nodes are the interactions or interaction occurrences.

The Interaction Overview Diagram focuses on the overview of the flow of control of the interactions which can also show the flow of activity between diagrams. In other words, you can link up the "real" diagrams and achieve high degree navigability between diagrams inside an Interaction Overview Diagram.

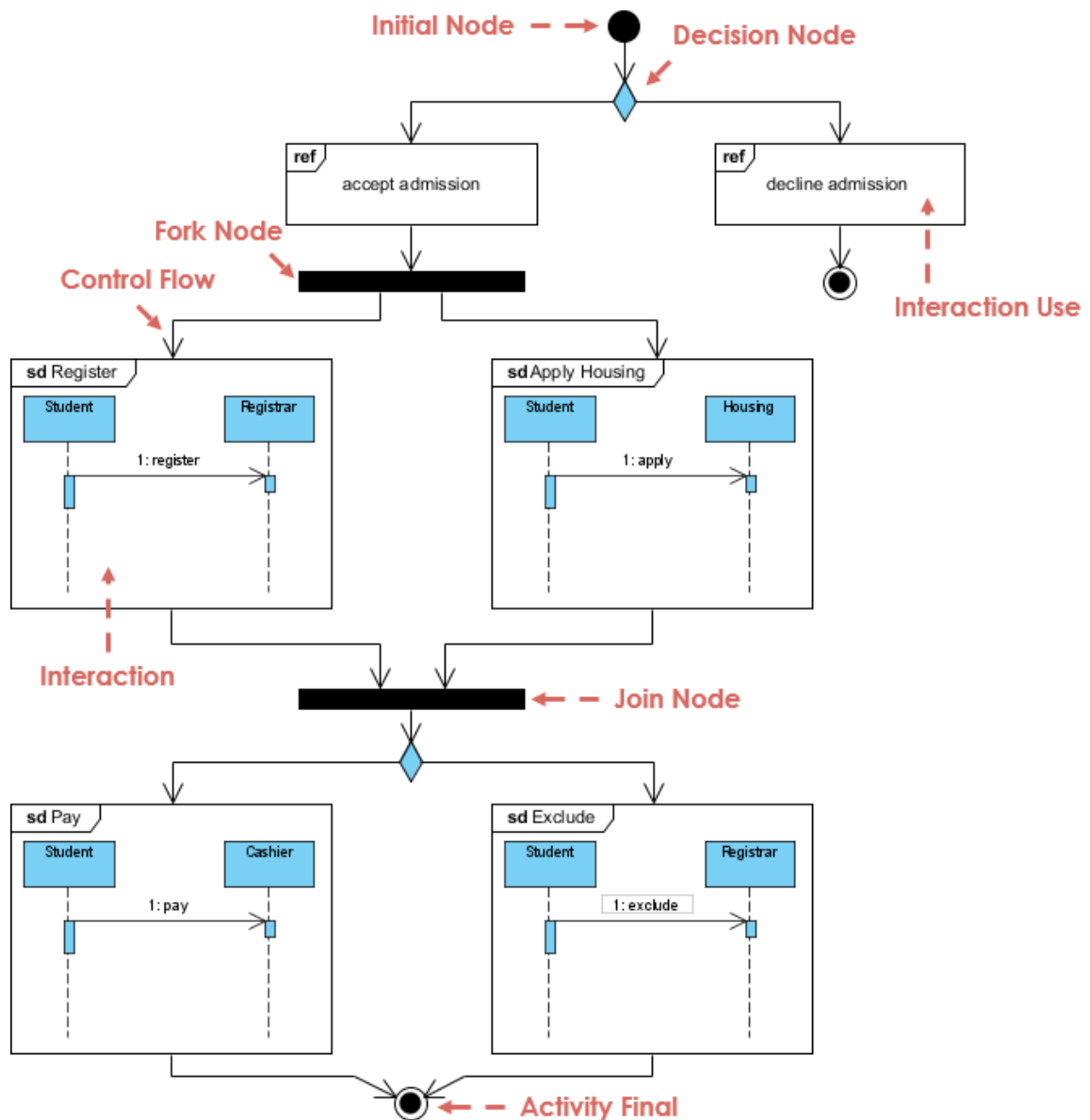
Interaction Overview Diagram is one of the fourteen types of diagrams of the Unified Modeling Language (UML), which can picture a control flow with nodes that can contain interaction diagrams which show how a set of fragments might be initiated in various scenarios. Interaction overview diagrams focus on the overview of the flow of control where the nodes are **interactions** (sd) or **interaction use** (ref).

The other notation elements for interaction overview diagrams are the same as for activity and sequence diagrams. These include initial, final, decision, merge, fork and join nodes.

The example above shows a student who has been accepted into a university. First the student must be accept or decline admission.

After accepting, the student must both register for classes and apply for housing. After both of those are complete, the student must pay the registrar.

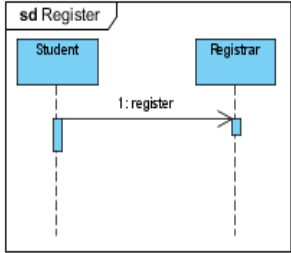
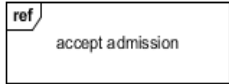
If payment is not received in time the student is excluded by the registrar.



**extracted from UML 2.0 Reference Manua*

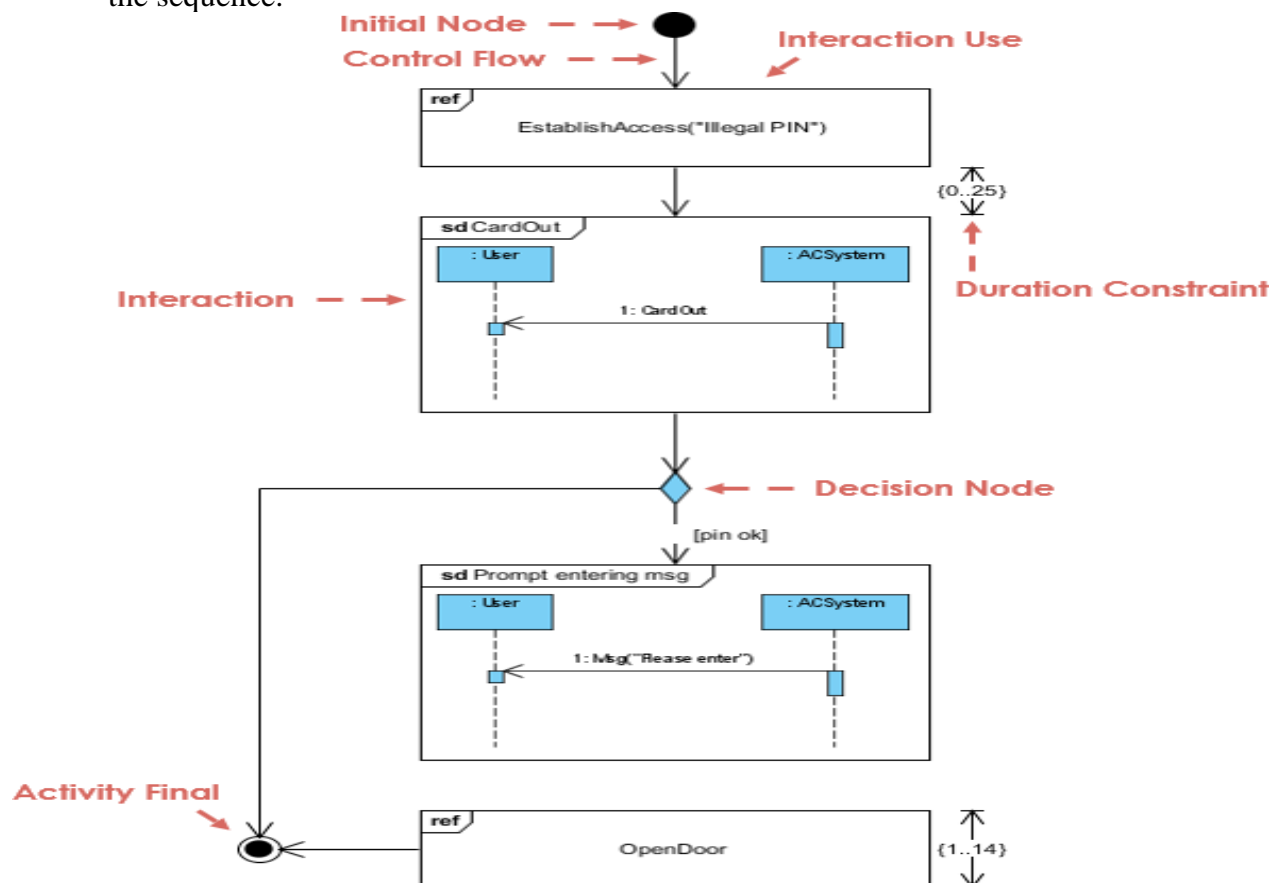
The example above shows a student who has been accepted into a university. First the student must be accept or decline admission. After accepting, the student must both register for classes and apply for housing.

After both of those are complete, the student must pay the registrar. If payment is not received in time the student is excluded by the registrar.

Node Type	Notation
<p>Interaction</p> <p>An Interaction diagram of any kind may appear inline as an Activity Invocation.</p>	
<p>Interaction Use</p> <p>Large and complex sequence diagrams could be simplified with interaction uses. It is also common to reuse some interaction between several other interactions.</p>	

Interaction Diagram Example - Access Control System

- The Interaction EstablishAccess occurs first with argument "Illegal PIN" followed by an interaction with the message CardOut which is shown in an inline Interaction.
- Then there is an alternative as we find a decision node with an InteractionConstraint on one of the branches.
- Along that control flow we find another inline Interaction and an InteractionUse in the sequence.



Interaction Diagram Example - Scheduling System

