Sequence diagram

A **sequence diagram** simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event **diagrams** or event scenarios to refer to a **sequence diagram**. **Sequence diagrams** describe how and in what order the objects in a system function.

Sequence Diagram Tutorial

Start with one of SmartDraw’s included sequence diagram templates. You’ll notice that all the notations and symbols you need are docked to the left of your drawing area. Simply stamp them to your page and connect the symbols.

* Model and document how your system will behave in various scenarios
* Validate the logic of complex operations and functions

Basic Sequence Diagram Notations

* Class Roles or Participants

Class roles describe the way an object will behave in context. UML object symbol to illustrate class roles, but doesn’t list object attributes.

* Activation or Execution Occurrence

Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its life line.

* Messages

Messages are allows that represent communication between objects. Use half- arrowed lines to represent asynchronous message. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks. For message types See below

* Lifelines

Lifelines are vertical dashed lines that indicate the object’s presence over time and

Life span of the object that are participating over a period of time.

* Destroying Objects

Objects can be terminated early using an arrow labeled”<<destroy>>” that points to an X. This object is removed from memory. When that object lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

* Loops

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].

Objects/Participants

* Generally it is placed at the top on the x axis
* Object that initiate the interaction are placed at the left

Types of Messages In Sequence Diagrams

* Synchronous Message

A synchronous message requires a response before the interaction can continue. It’s usually drawn using a line with a solid arrowhead pointing from one object to another

* Asynchronous Message

Asynchronous message don’t need a reply fr interaction to continue. Like synchronous messages, they are drawn with an arrow connecting two lifelines; however, the arrowhead is usually open and there’s no return message depicted.

* Reply or Return Message

A reply message is drawn with a dotted line n an open arrowhead pointing back to the original lifeline.

* Self Message

A message an object sends to itself, usually shown as a U shaped arrow pointing back to itself.

* Create Message

This is a message that creates a new object. Similar to return message, it’s depicted with a dashed line and an open arrowhead that points to the rectangle representing the object created.

* Delete Message

This is a message that destroys an object. It can be shown by an arrow with an X at the end.

* Found Message

A message sent from an unknown recipient, shown by an arrow from an endpoint to a lifeline.

* Lost Message

A message sent to an unknown recipient. It’s shown by an arrow going from a lifeline to an endpoint, a filled circle or an X.

Use Case Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system’s users (also known as actors) and their interactions with the system. To build one, You’ll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

* Scenarios in which your system or application interacts with people, organizations or external systems
* Goals that your system or application helps those entities (known as actors) achieve
* The scope of your system

When to apply use case diagrams

A use case diagram doesn’t go into a lot of detail—for example, don’t expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high-level overview of the relationship between use cases, actors, and systems. Experts recommend that use case diagrams be used to supplement a more descriptive textual use case.

UMLi is the modeling toolkit that you can use to build your diagrams. Use cases are represented with a labeled oval shape. Stick figures represented with a labeled oval shape. Stick figures represent actors in the process, and the actor’s participation in the system is modeled with a line between the actor and use case itself.

UML use case diagrams are ideal for:

* Representing the goals of system-user interactions
* Defining and organizing functional requirements in a system
* Specifying the context and requirements of a system
* Modeling the basic flow of events in a use case

Use Case diagram components

* Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
* System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
* Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

Use case diagram symbols and notation

The notation for a use case diagram is pretty straightforward and doesn’t involve as many types of s symbols as other UML diagrams. Here are all the shapes you will be able to find in Lucidchart:

* Use cases: Horizontally shaped ovals that represent the different uses that a user might have.
* Actors: Stick figures that represent the people actually employing the use cases.
* Associations: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
* System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho killer is outside the scope of occupation in the chainsaw example found below.
* Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.