ENTERPRISE
ARCHITECT

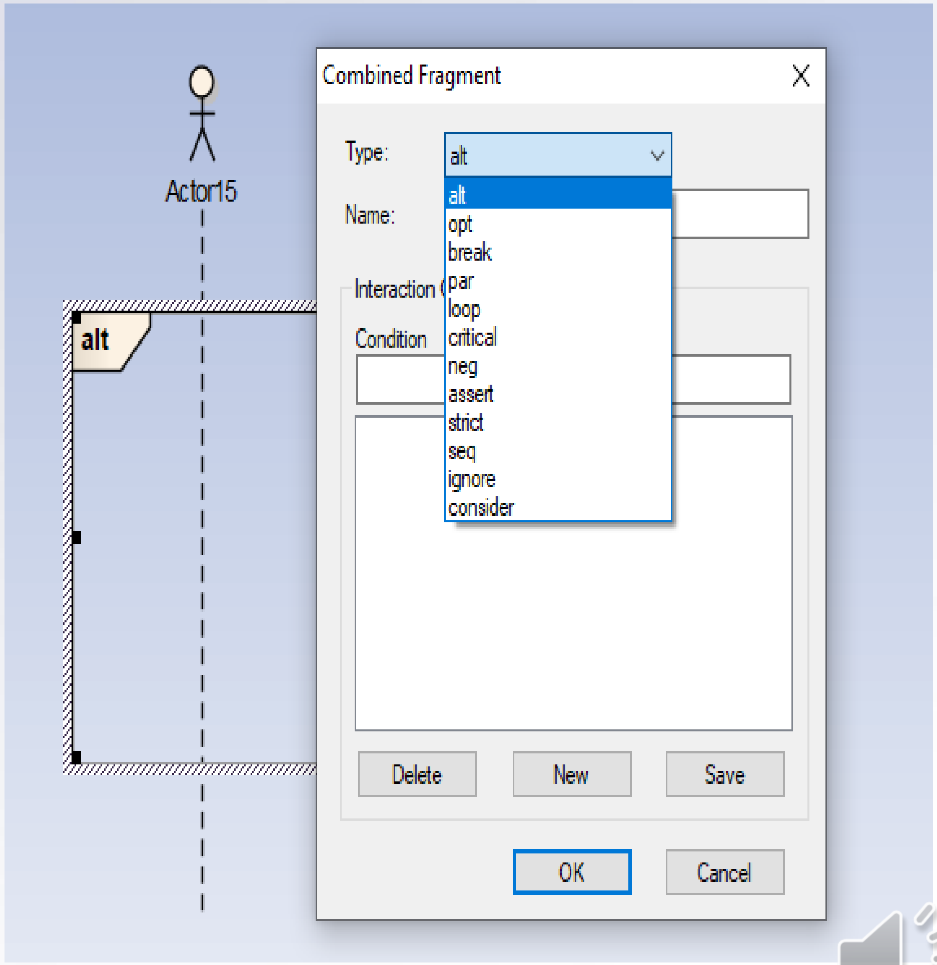
sequence diagram

Department of Software

Sequence Diagram Elements

Fragment types:

1. alt / alternative *
2. opt / optional
3. par / parallel
4. loop *
5. critical
6. neg / negative
7. assert
8. strict
9. seq
10. ignore
11. consider
12. ref



Sequence Diagram Elements

Operator	Use to
alt	Divide up interaction fragments based on Boolean conditions.
opt	Enclose an optional fragment of interaction
par	Indicate that operands operate in parallel.
loop	Indicate that the operand repeats a number of times, as specified by interaction constraints.
critical	Indicate a sequence that cannot be interrupted by other processing.
neg	Assert that a fragment is invalid, and implies that all other interaction is valid.
assert	Specify the only valid fragment to occur. Often enclosed within a *consider* or *ignore* operand.
strict	Indicate that the behaviors of the operands must be processed in strict sequence.
seq	Indicate that the Combined Fragment is weakly sequenced. This means that the ordering within operands is maintained, but the ordering between operands is undefined, so long as an event occurrence of the first operand precedes that of the second operand, if the event occurrences are on the same lifeline.
ignore	Indicate which messages should be ignored during execution, or can appear anywhere in the execution trace.
consider	Specify which messages should be considered in the trace. This is often used to specify the resulting event occurrences with the use of an **assert** operator.
ref	Provide a reference to another diagram.

Sequence Diagram Connectors

Messages

Used to illustrate communication between different active objects of a sequence diagram, Used when an object needs to activate a process of a different object to give information to another object

The interaction in a sequence diagram between the objects can be shown by using messages. The messages on sequence diagram are specifies using an arrow from participant that wants to pass the messages to the participant that receive the messages . Messages can be flow in whatever direction required for interaction from left to right and right to left.

Sequence Diagram Connectors

Message format:

- Message _ name.
- Message _ name (arguments) : return type.
- Message _ name (parameters).
- Message _ name (return).

Sequence Diagram Connectors

1. Message:

Message indicate a flow of information or transition of control between elements. A Message defines a particular communication between Lifelines of an Interaction.

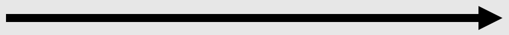
A Message is a Named Element that defines one specific kind of communication in an Interaction. A communication can be, for example, raising a signal, invoking an Operation, creating or destroying an Instance.

A Message associates normally two Occurrence Specifications - one sending Occurrence Specification and one receiving Occurrence Specification.

Sequence Diagram Connectors

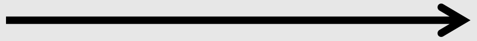
Types of message:

1. **Synchronous messages:**



It is a message where the sender is blocked and waits until the receiver has finished processing of message. It is invoked the caller waits for the receiver to return from the message invocation. It is represented by solid line with full arrow.

2. **Asynchronous messages:**

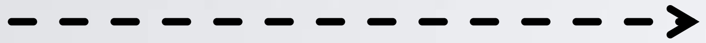


It is a messages where the sender is not blocked and can continue executing. It is represent by solid line with half arrow.

Sequence Diagram Connectors

Types of message:

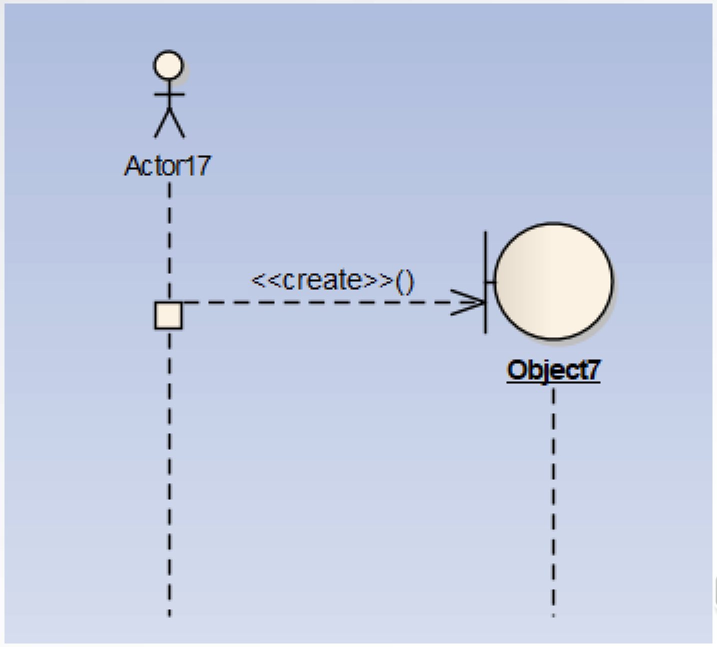
3. Return messages:



It can be used at the end of activation bar to show that control flow of activation returns to the participant that pass the original message. It is represent by dashed line from sender to receiver.

4. Create messages:

It is used to create object during interaction. The object can be created by using <<create>> to indicate the timing of creation.

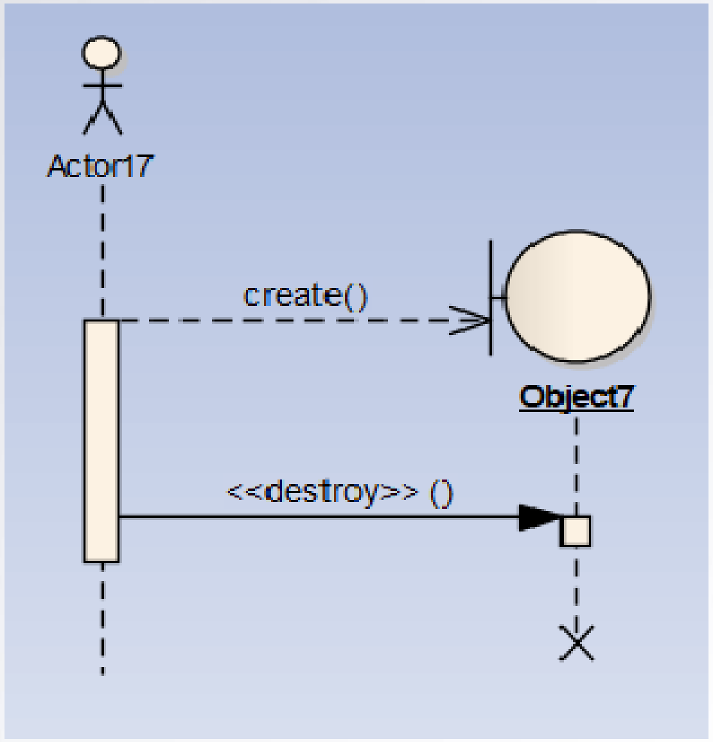


Sequence Diagram Connectors

Types of message:

5. Destroy messages:

It is used to destroy the objects during interaction. The objects can be terminated using <<destroy>> which points to an "x". It indicates that object named message is terminated.



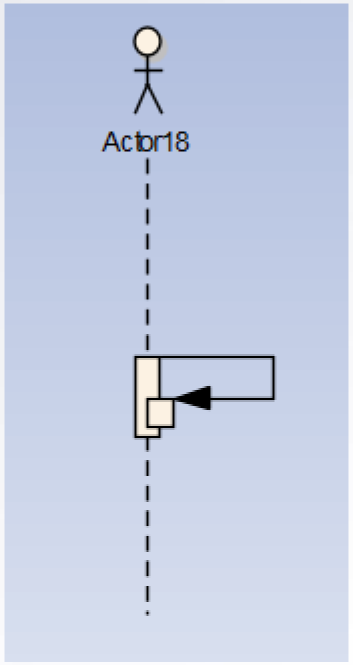
Sequence Diagram Connectors

2. Self message (Reflexive message):

Self-Message is a kind of message that represents the invocation of message of the same lifeline.

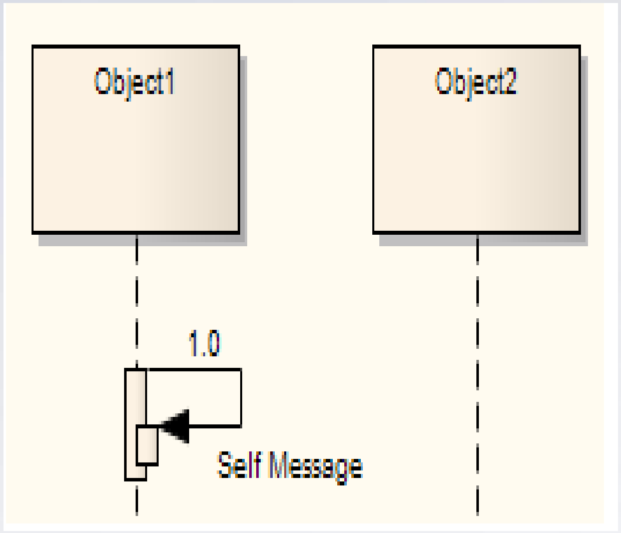
If the object sends the message to itself then it is called as Reflexive message (Self message).

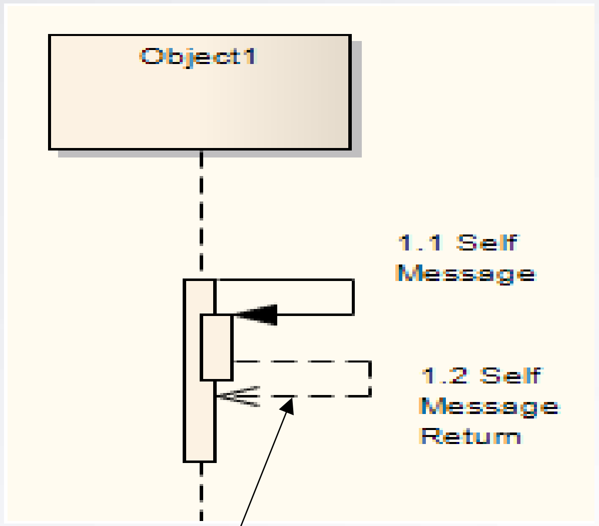
It is represented by solid line with loops the lifeline of object. It is indicated with a message arrow that starts and ends at the same lifeline



Sequence Diagram Connectors

- Self-Message as Return :





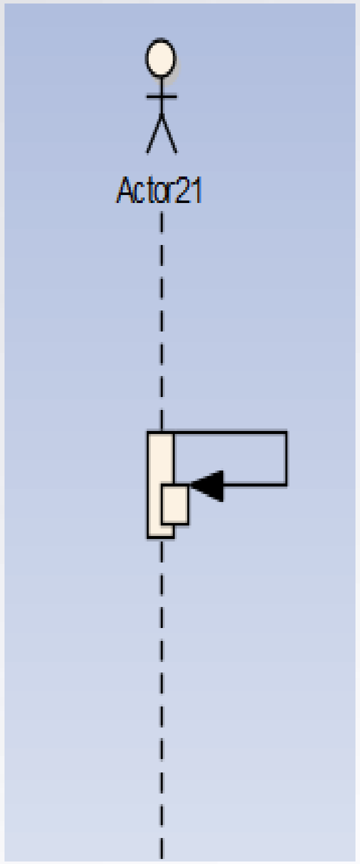
Self-Message as Return

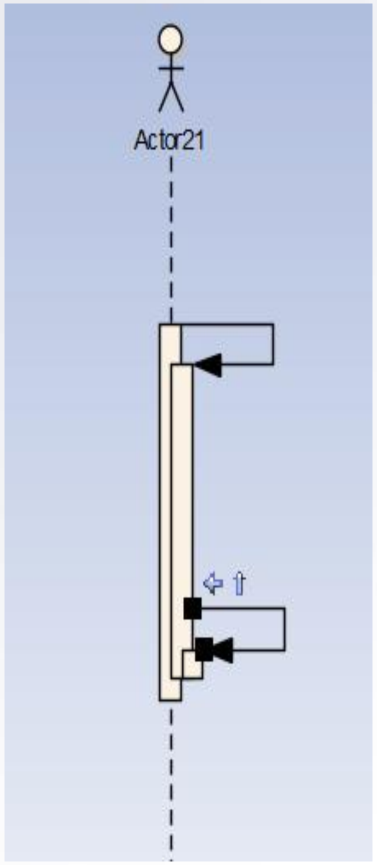
Sequence Diagram Connectors

3. Call Message:

A Call is a type of Message connector that extends the level of activation from the previous Message.

All Self-Messages create a new activation level, but this focus of control usually ends with the next Message

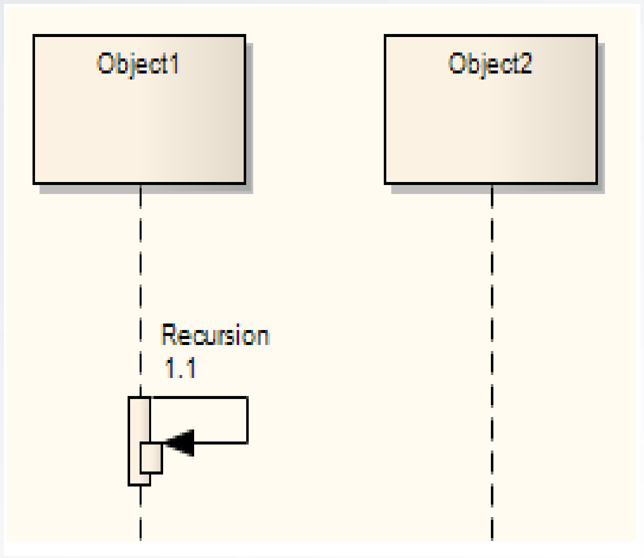




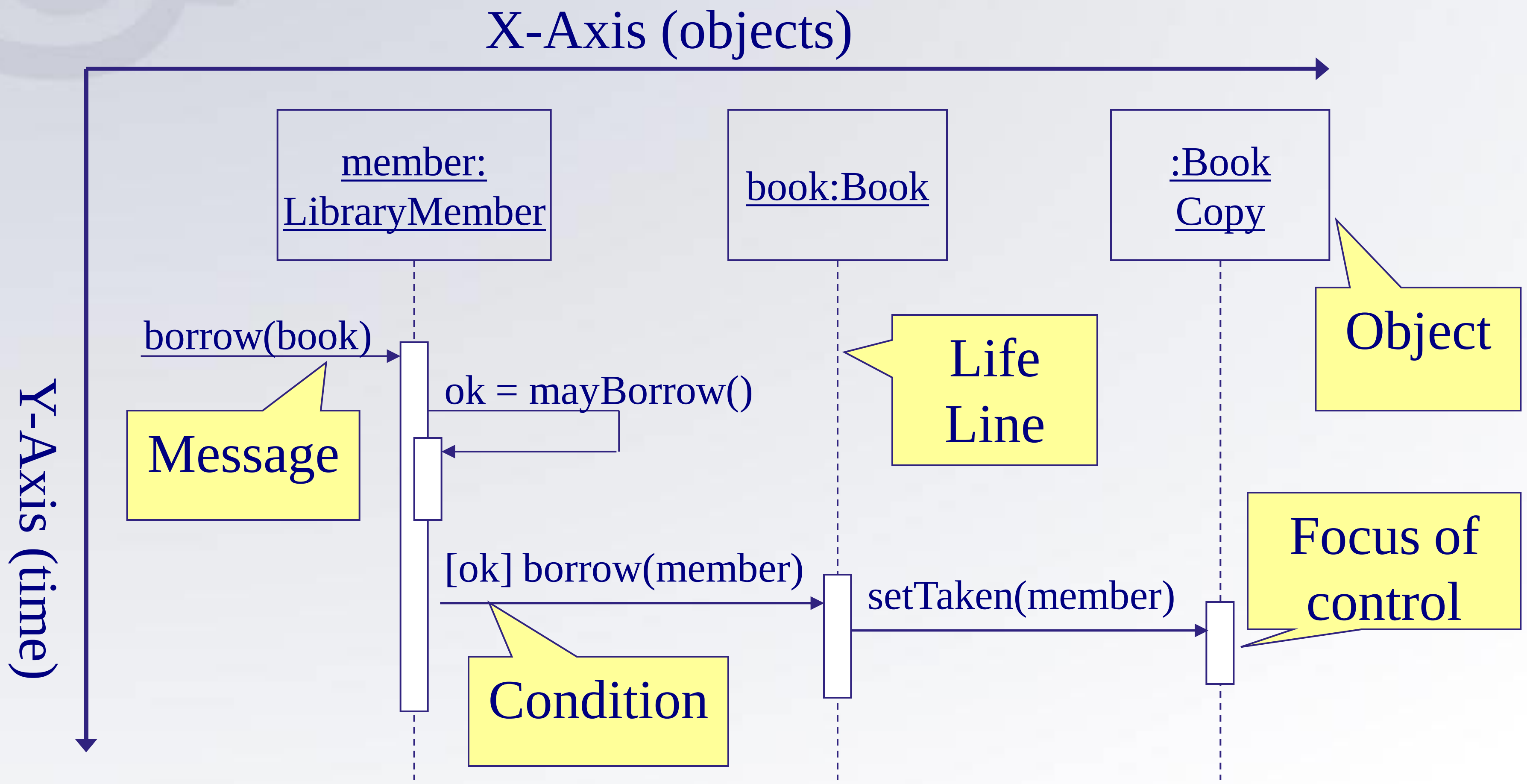
Sequence Diagram Connectors

4. Recursion message:

A Recursion is a type of Message used in Sequence diagrams to indicate a recursive function.



Sequence Diagram



the Difference Between Activity Diagram and Sequence Diagram

Sequence diagram	Activity diagram
1. a sequence diagram is a UML diagram that represents the object interactions arranged in time sequence.	1. An activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration, and concurrency.
2. the main focus in a sequence diagram is the interaction between objects over a specific period of time.	2. The main focus in an activity diagram is the flow of activities.
3. a sequence diagram helps to visualize the sequence of calls in a system to perform a specific functionality.	3. an activity diagram helps to model the workflow a system

Conclusion:

Activity and sequence diagrams are two behavior diagrams. The main difference between activity diagram and sequence diagram is that the activity diagram represents the flow of activities in a system while the sequence diagram represents the sequence of messages flowing from one object to another.

How to Draw a sequence Diagram?

When we are planning to draw a sequence diagram, we should identified the following items.

- Identify the relevant external actor to system using message notation objects involved in the system.
- Describe message from
- Establish the role of each object.
- Identify the controller.
- Identify and add any special conditions on input message, including iteration and true/false conditions
- Identify the collaborators.
- Decide on the messages between objects.

SPARX SYSTEMS

Thank You