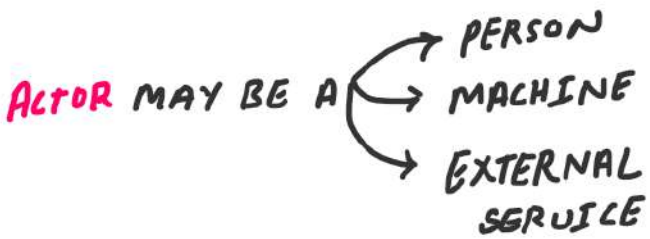
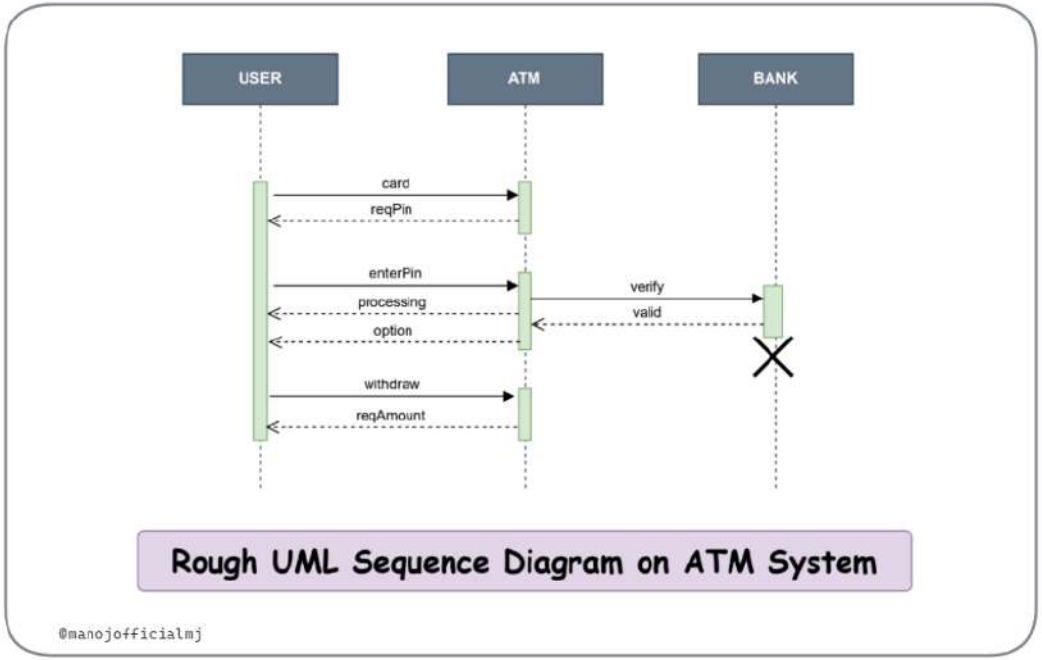
Sequence Diagram

- *A sequence diagram is a form of communication diagram that illustrates how different actors and objects interact with each other or between themselves.*
- *The diagram represents these interactions as an exchange of messages between various entities and the type of exchange.*
- *Sequence diagrams also demonstrate the sequence of events that occur in a specific use case and the logic behind different operations and functions.*

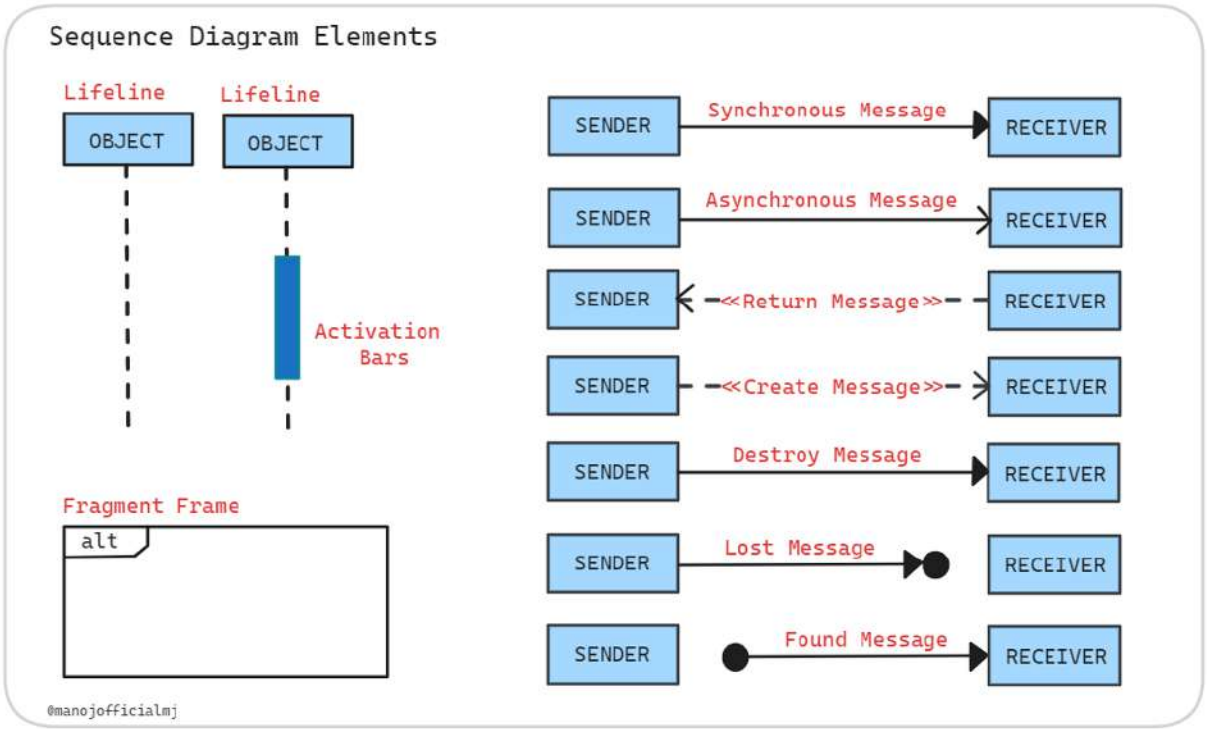
In short, *a sequence diagram is just an interaction diagram in which different* ***models/objects/actors*** *communicate with each other in the form of* ***chronological*** *messages and functions.*





Elements of Sequence Diagram

1. Lifeline Element
2. Activation Bars Element
3. Messages Elements
    A. Synchronous Message
    B. Asynchronous Message
    C. Synchronous Return Message
    D. Create Message
    E. Destroy Message
    F. Lost Message
    G. Found Message
4. Fragment Frame Element



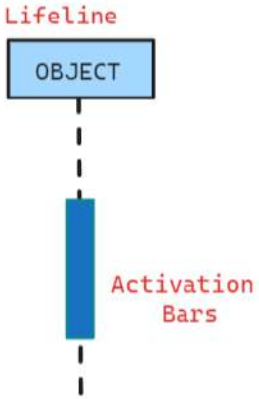
1. Lifeline Element

- *Lifeline represents the existence of an object over a period of time during the interaction.*
- *It is depicted as a vertical **dashed line** (a dotted line hanging down from top) extending downwards from the object name or class name.*
- *Each lifeline corresponds to a specific object or actor participating in the interaction.*
- *These lifelines stand for different parts of a system, like objects or actors, and they don't cross each other.*
- *Each thing has a lifeline showing when it's active or not, shown as a dotted line hanging down from it.*

2. Activation Bars Element

- *Activation bars, also known as activation rectangles, indicate the period during which an object is actively processing a message.*
- *They are drawn as horizontal lines extending from a lifeline, typically with a **LABELED activation time**.*
- *Activation bars visually illustrate the duration of an object's processing activity during the interaction.*

***In short,** Activation bars show when an object is busy sending or getting messages. We make these as straight up-and-down boxes on the object's lifeline.*

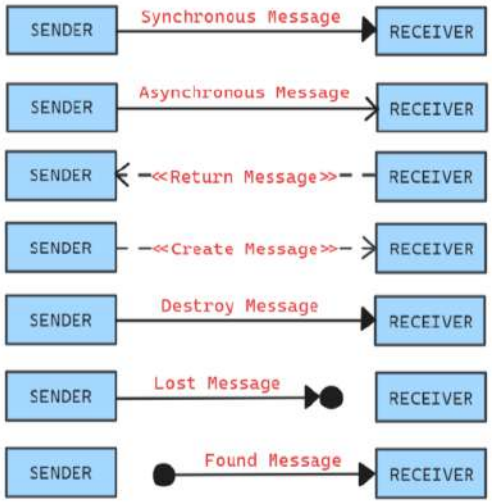


3. Message Elements

In sequence diagrams, a message is when two things talk to each other. This can happen when they send messages back and forth, start an action, or make a new thing or part of the system. Messages are shown as straight lines that can go from left to right, right to left, or even circle back to the same thing.

Messages are the means by which objects communicate with each other in a sequence diagram.
Different types of messages include:

A. *Synchronous Message*
B. *Asynchronous Message*
C. *Synchronous Return Message*
D. *Create Message*
E. *Destroy Message*
F. *Lost Message*
G. *Found Message*



A. Synchronous Message

- *A synchronous message is when the sender sends a message and has to wait for a response from the receiver before proceeding (it can do anything else).*
- *We draw these messages with **a solid line** and **a solid arrow head/tip** pointing from the sender to the receiver.*

B. Asynchronous Message

- *An asynchronous message is when the sender sends a message but doesn't have to wait for a response from the receiver and continues its processing immediately.*
- *We draw these messages with **a solid line** with **an open arrow head/tip** pointing from the sender to the receiver.*

C. Synchronous Return Message

- *Synchronous return messages are used to indicate the response from the receiver to the sender after processing a synchronous message.*
- *Whenever we send a synchronous message, we need to get one of these replies to show that the message was received and handled.*
- *We draw these replies with **a dotted line** that **ends in a open arrow head/tip** pointing from the receiver back to the sender.*

D. Create Message

- *Create messages are used to indicate the creation of a new object by another object during the interaction.*
- *They are shown as **a dashed line** that **ends in an open arrow head/tip** with the label "**create**" pointing from the creating object to the newly created object.*

E. Destroy Message

- *Destroy messages signify the destruction or termination of an object during the interaction.*
- *They are depicted as **a solid line** that **ends in solid arrow head/tip** with the label **"destroy"** pointing from the destroying object to the object being destroyed.*
- *A destroy message is used when something is removed or ended during a series of events. This can happen because of a message or action, and it means the end of its timeline.*

F. Lost Message

- *Lost messages represent messages that are sent but not received by the intended recipient.*
- *We show lost messages with **an arrow** (A solid line with solid arrow head/tip) that **ends in a filled circle**, typically with the label **"lost,"** pointing from the sender to the receiver.*

***In short,** A lost message is when a message starts from something but never makes it to where it was supposed to go. It looks like a message that just stops.*

G. Found Message

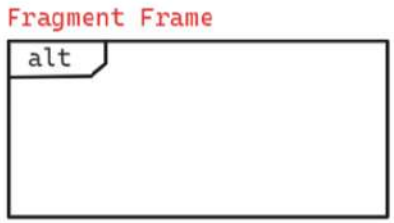
- *Found messages represent messages that are unexpectedly received by an object without being explicitly sent by any other object.*
- *We show found messages with **an arrow** (A solid line with solid arrow head/tip) that **begins with a filled circle**, typically with the label **"found"** and an arrow pointing from the receiver to the sender.*

***In short,** A found message is when a message comes in, but we don't know who sent it. It looks like a message that arrives but doesn't seem to start from anywhere.*

4. Fragment Frame Element

- *Fragment Frame Element is like a utility function.*
- *Fragment frames are used to represent conditional or repetitive behaviour within a sequence diagram.*
- *They enclose a portion of the diagram and indicate alternative paths, loops, or other logical structures.*
- *Fragment frames can be of different types such as **alt** (alternative), **opt** (optional), **loop**, etc., each with its specific notation and semantics.*

When we use the fragment frame:
*We want our sequence diagram to be easy to understand, without too much detail all at once. But sometimes, we need to show more complicated things like "what if" scenarios or repeating actions. Sequence diagrams have a special part called "**sequence fragment**" for this. **We can tell what a fragment does by a label in its top left corner**.*



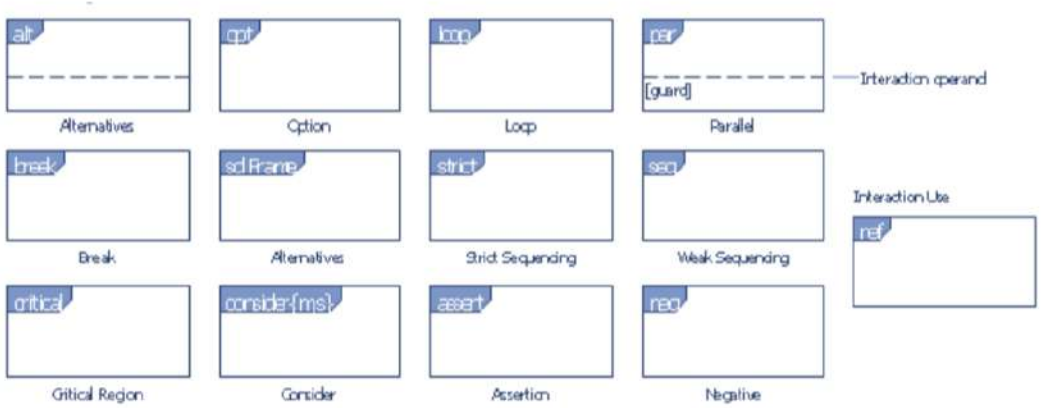
Here are some common types:

A. "Alternative" (alt): *This is like an "if-else" in coding. It splits the fragment into parts, and depending on a certain condition, one part happens.*

B. "Option" (opt): *This is like an "if" condition. The fragment only happens if a certain condition is true. If not, it's skipped, and the diagram goes on.*

C. "Loop" (loop): *This is for things that repeat. The fragment keeps going over and over until a condition is met.*

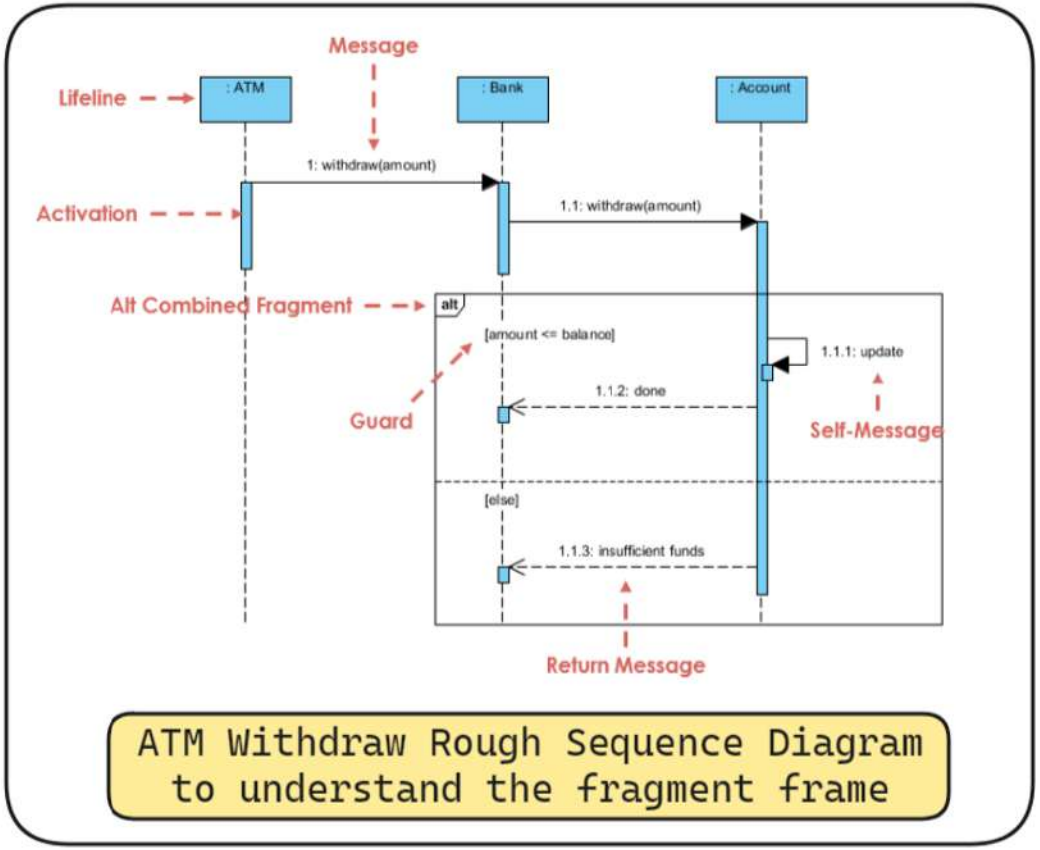
D. "Reference" (ref): *This is for when diagrams get too big. It lets us refer to another part of the diagram, so we don't have to draw everything again.*



Rough Sequence Diagram to Understand Fragment Frame:

In the diagram, *we use something called an "alternatives operator" that splits the sequence into two parts. The first part happens if a certain condition is true. If not, then the second part happens instead.*

In this case, *the condition is checking if the account has enough money compared to what the customer wants to take out. If the account has enough money, the first part happens; if not, the second part does.*



5. How to Draw a Sequence Diagram:

To make a clear sequence diagram, we have several steps to take. But first, remember that there's no single right way to make one. Different people might do it differently.

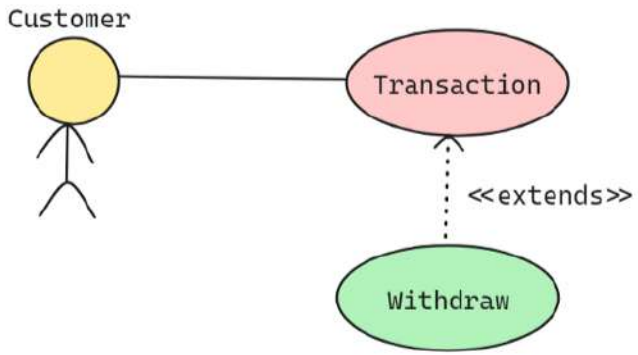
Step 1: Identify the use-case
Step 2: Identify the actors and objects
Step 3: Identify the order of actions
Step 4: Create the diagram

Step 1: Identify the use-case

To begin with the sequence diagram, we need to understand our situation. The sequence diagram helps us show the steps in order for that situation.

*Let's imagine a simple situation where a customer is taking out **money from an ATM**. Here's what that looks like:*

1. Transaction
2. Cash Withdraw
And more...

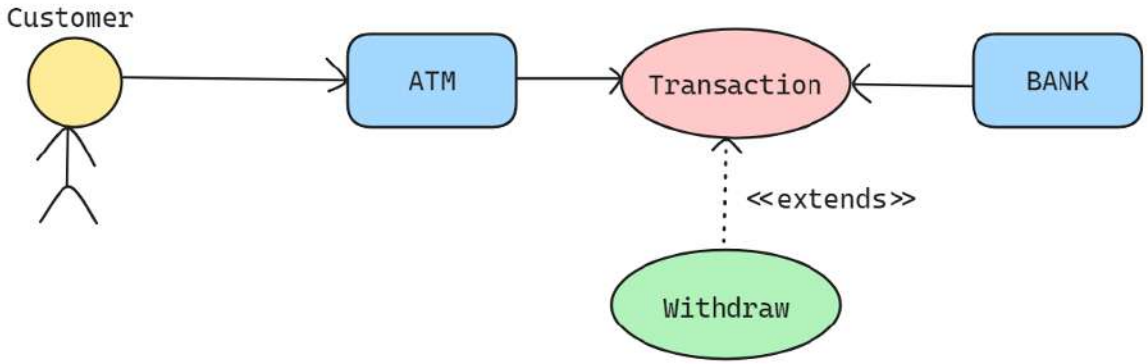


Step 2: Identify the actors and objects

Now, we need to write down all the people and things that will take part in the steps. They are:

1. *Customer*
2. *ATM*
3. *Transaction*
4. *Account*
5. *Cash Dispenser*
6. *Bank*

And more...



Step 3: Identify the order of actions (Flow)

We know who's involved in getting cash from the ATM, but we need to figure out how they all work together. We should write down what each one does, step by step.

Here's a simple way to understand their actions:

1. The customer asks the ATM for money, using their account info and how much they want.
2. The ATM starts the process to take out money from the account for the asked amount.
3. The transaction is checked to make sure the account can give that amount of money.
4. If the amount is okay, the account agrees to the transaction.
5. The ATM tells the cash dispenser to give out the needed money.
6. The cash dispenser confirms it's giving out the cash.
7. The ATM tells the customer to take the money.

Step 4: Create the diagram

We've named all the things and people taking part in this activity and the steps they'll take. To make the sequence diagram, we need to connect these steps to the kinds of messages they'll send to each other. Here's how we could draw a diagram for taking out cash:

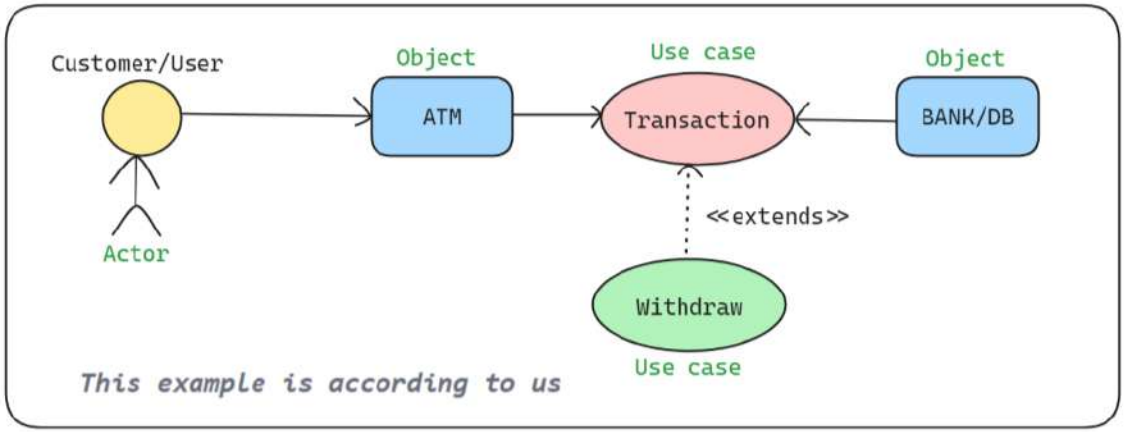
Example 1: ATM Withdraw Sequence Diagram Step by Step

Step 1: Identify the use-case

Use-case: Withdraw cash from an ATM

Step 2: Identify the actors and objects

Actors: User
Objects: ATM, Database (Bank)



Step 3: Identify the order of actions (Flow)

1. User inserts card into ATM.
2. ATM reads the card number and sends it to the database for verification.
3. Database checks if the card is valid. If it is, it informs the ATM.
4. ATM requests the user to enter their PIN (Personal Identification Number).
5. User enters their PIN, and it is sent to the ATM.
6. ATM checks the entered PIN by sending it to the database.
7. Database verifies if the PIN is correct. If it is, it informs the ATM.
8. ATM displays the options menu to the user.
9. User selects the option to withdraw cash and requests it from the ATM.
10. ATM asks the user to input the desired amount of cash.
11. User selects the amount, and it is sent to the ATM.
12. ATM checks with the database to see if the user has sufficient funds for the withdrawal.

Then, there are two alternate paths depending on whether the transaction is approved or rejected:

14. Transaction Approved:
    1. Database confirms that the user has sufficient funds, and it informs the ATM.
    2. ATM dispenses the requested cash to the user.
    3. User takes the cash.
    4. ATM returns the card to the user.
15. Transaction Rejected:
    1. Database confirms that the user has insufficient funds, and it informs the ATM.
    2. ATM shows details of the rejection to the user.
    3. ATM returns the card to the user.

Step 4: Create the diagram