

# Analiza și Modelarea Sistemelor Software - Lab 2<sup>1</sup>

Traian Șerbănuță

2025

---

<sup>1</sup>Thanking Andrian Babii @ Endava for slide content

# Agenda

UML behavioral diagrams

Sequence diagrams

# Behavior diagrams

## Purpose

- ▶ individual aspects of a system and their changes are displayed at runtime.
- ▶ provide clarity about internal processes, business processes or the interaction of different systems.

## Elements and relationships

- ▶ elements resemble verbs in a natural language
- ▶ relationships typically convey passage of time

## Example

Elements of a behavioral diagram of a vehicle reservation system

- ▶ Make a Reservation
- ▶ Rent a Car
- ▶ Provide Credit Card Details.

## Types of behavioral diagrams

# Sequence Diagrams

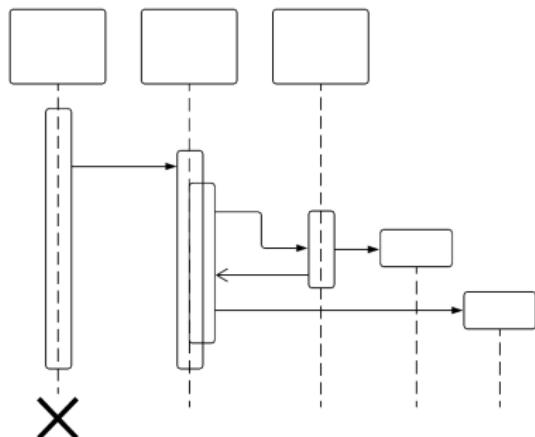
How –and in what order– a group of objects works together.

Also known as

- ▶ event diagrams
- ▶ event scenarios

Used by both software developers  
and business professionals, to

- ▶ understand new requirements
- ▶ document existing processes



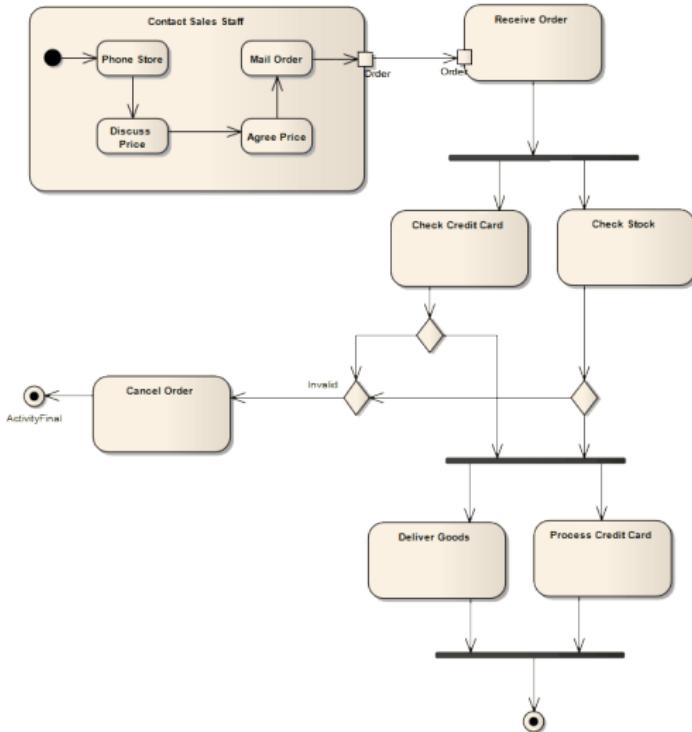
# Activity diagrams

## Model

- ▶ behaviors of a system
- ▶ how these are related (in the overall flow of the system).

Activities can be:

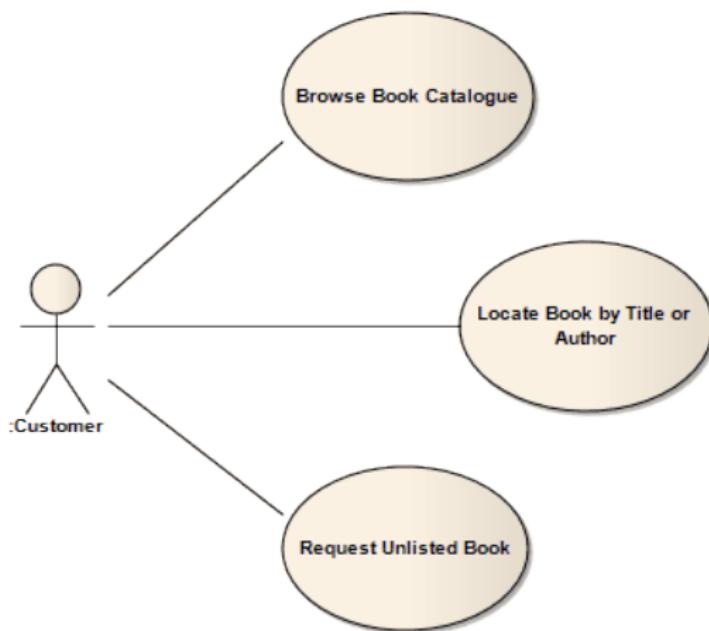
- ▶ sequential,
- ▶ branched
- ▶ concurrent.



# Use case diagrams

Model users interacting with the system

- ▶ Users: stick figures called “actors”
- ▶ high-level overview of relationships between actors and systems
- ▶ explain system to non-technical audience

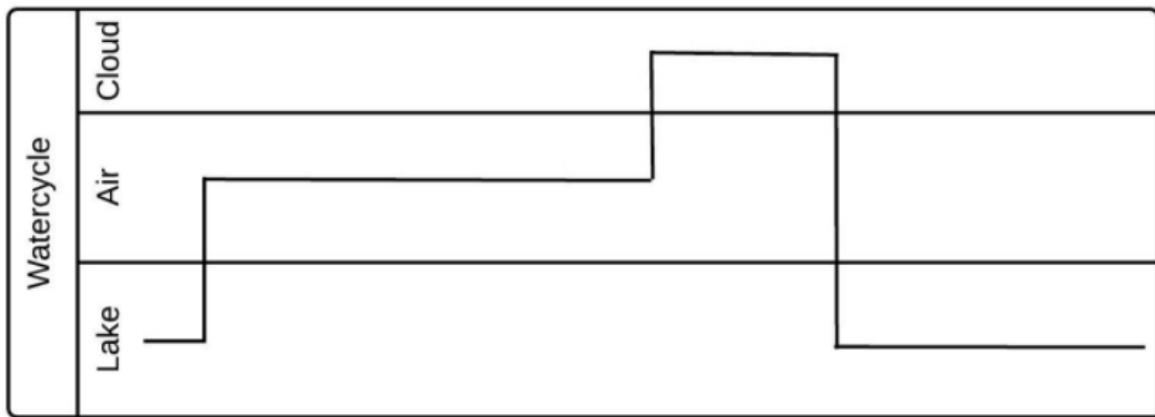


This diagram illustrates the use cases that support searching for a book and browsing the resultant record set. The customer can enter browse criteria and scroll through the results. The user can select an item for addition to their current shopping cart.

# Timing diagrams

Powerful tools for making a system as efficient as possible.

- ▶ define the behavior of different objects within a time-scale
- ▶ represent objects changing state and interacting over time.



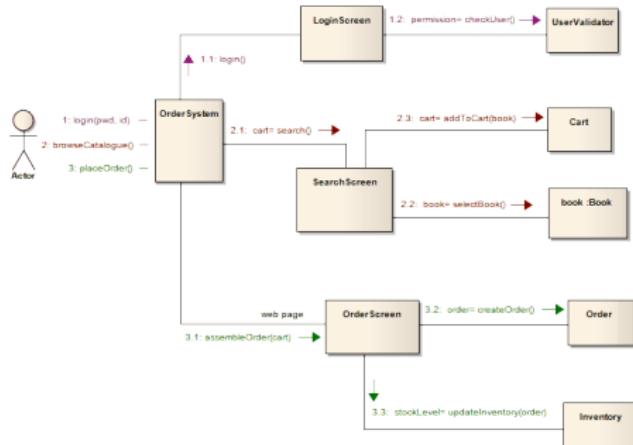
# Communication diagrams

Model how components communicate and interact

- ▶ like sequence diagrams
- ▶ but focus on interaction
- ▶ program communication

Useful for

- ▶ businesses
- ▶ organization
- ▶ engineers

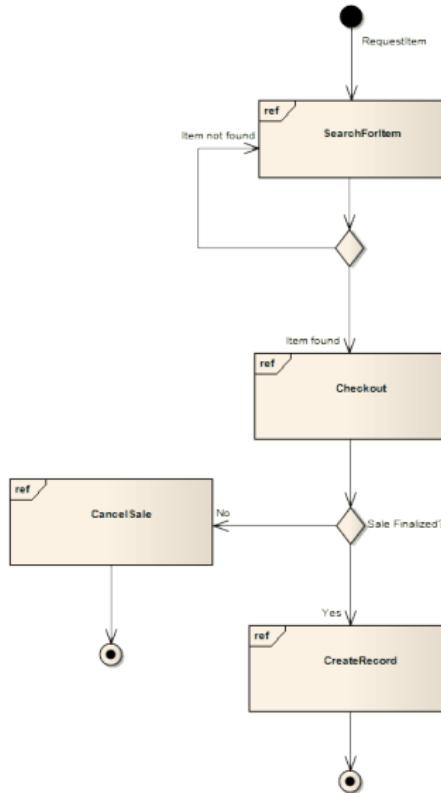


# Interaction Overview diagrams

Activity diagram where nodes are interaction diagrams

Diagrams used as nodes

- ▶ sequence
- ▶ communication
- ▶ interaction overview
- ▶ timing

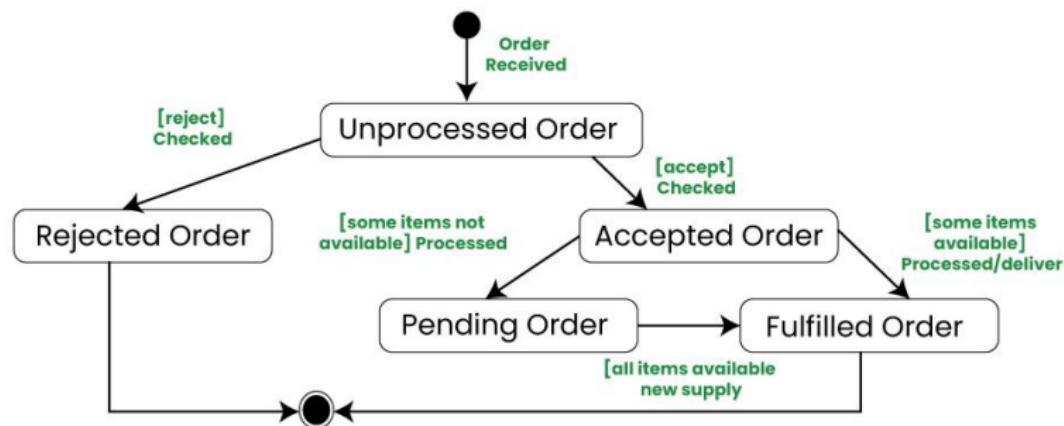


# State Diagram

Useful for representing state machines.

- ▶ describing all possible states of objects
  - ▶ combinations of information that an object can hold
  - ▶ **not** how the object behaves.
- ▶ how state changes through actions/inputs

State machine diagram for an online order



## Sequence Diagrams

# Sequence Diagrams

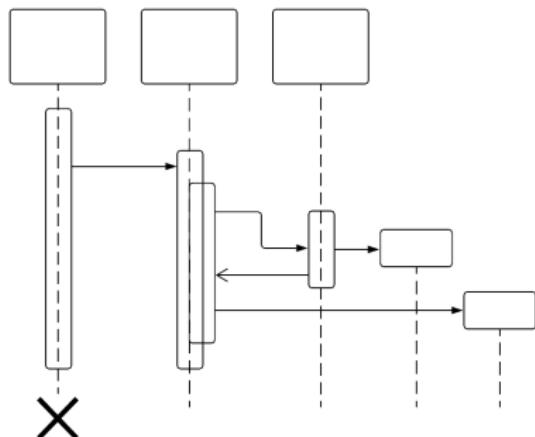
How –and in what order– a group of objects works together.

Also known as

- ▶ event diagrams
- ▶ event scenarios

Used by both software developers  
and business professionals, to

- ▶ understand new requirements
- ▶ document existing processes



# Use cases for sequence diagrams

## Usage scenario

Draw a diagram of how your system could be used.

- ▶ ensures thinking about the logic of every usage scenario.

## Method logic

Explore the logic of any function, procedure, or complex process.

## Service logic

Viewing a service as a high-level method used by different clients.

# Sequence diagram symbols and components



Figure 1: Object symbol



Figure 4: Actor

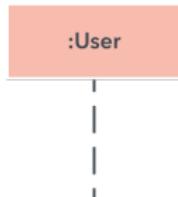


Figure 2: Lifeline symbol



Figure 3: Event



Figure 5: Alternative

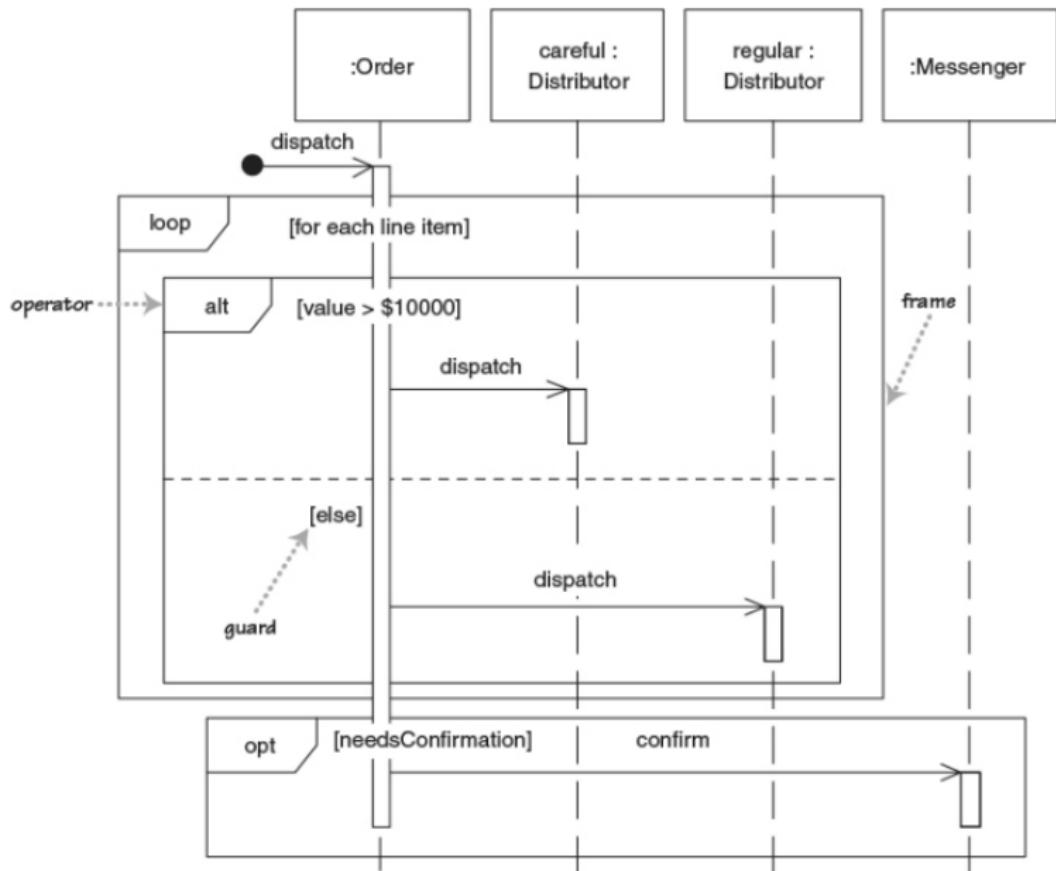


Figure 6: Loop

## ► Common message symbols

Symbol	Name	Description
→	Synchronous message	sender waits for response before it continues
→	Asynchronous message	don't require response before continuing. Only the call should be included in the diagram
-<<create>>-→	Create message (asynchronously)	creates a new object
<---	Reply message	replies to calls
X	Delete message	destroys an object

# Sequence diagram example



## Exercise

Model (using a sequence diagram) the possible scenarios of a **Customer** purchasing a **Product** using an online shopping system:

A *Customer* browses/searches the *Catalog*, adds and/or removes *Products* into a *Cart* (updating the stocks), then goes to a *Checkout* page which gets its credentials from a *Login* (which verifies/requires that the user is logged in), allows it to select payment/shipping info, and returns to the customer an order number to allow tracking the order.