

Analiza și Modelarea Sistemelor Software - Lab 5

Traian Șerbănuță

2025

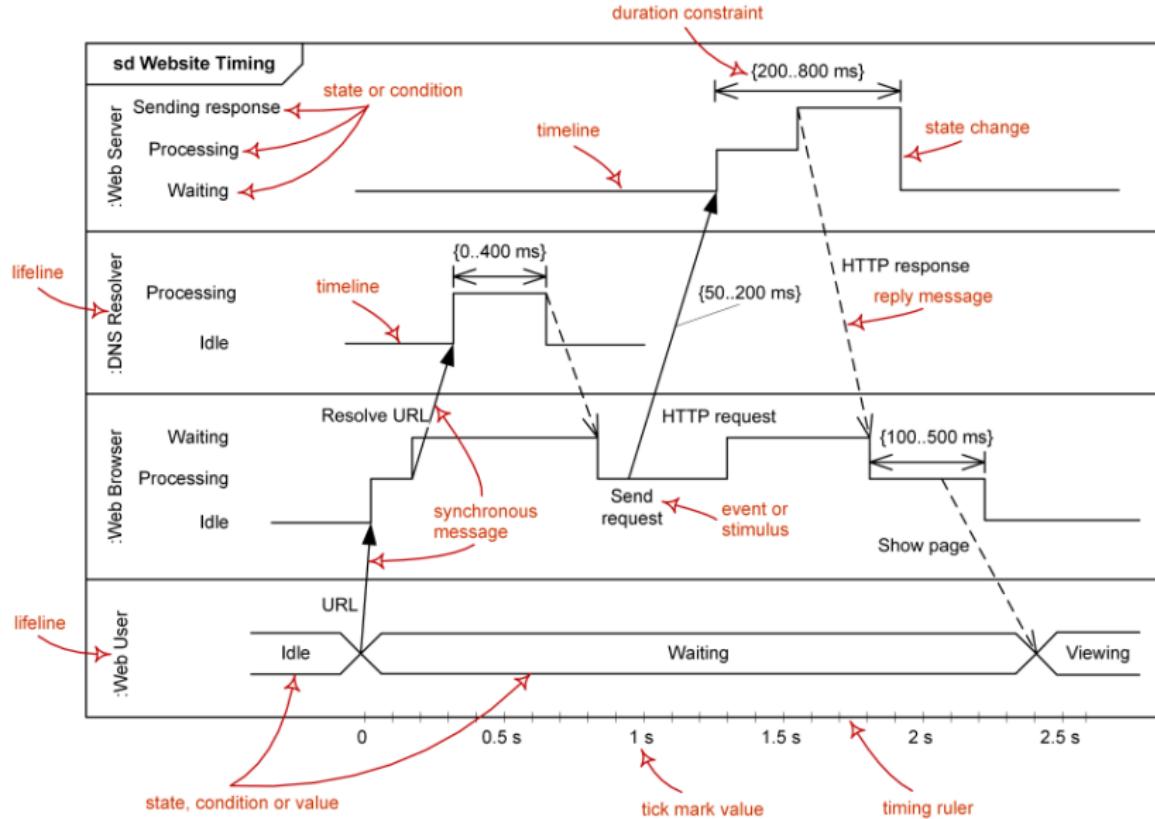
Timing Diagrams

Timing Diagrams — Quick Refresher

- ▶ Show how **state changes over time** for multiple lifelines
- ▶ Emphasize **temporal constraints, durations, and deadlines**
- ▶ Useful for:
 - ▶ embedded systems
 - ▶ robotics
 - ▶ real-time software
 - ▶ safety-critical controllers

Key Elements: - Lifelines - States / Value changes - Timing constraints - Duration and time markers - Messages and asynchronous events

Timing diagram example



Exercise Overview

You will create a **Timing Diagram** for a multi-sensor **Collision Avoidance System** (CAS) used by a drone. The system integrates multiple periodic and event-driven signals, strict deadlines, and interrupt conditions.

System Components (Lifelines)

- ▶ **FPS** — Front Proximity Sensor
- ▶ **ALT** — Downward Altimeter
- ▶ **VPU** — Vision Processing Unit
- ▶ **APC** — Autopilot Control Module
- ▶ **MC** — Motor Controller

Your diagram must include all five.

Normal Timing Behavior

- ▶ **FPS**: emits distance reading every **50 ms**
- ▶ **ALT**: emits altitude reading every **50 ms**
- ▶ **VPU**: emits obstacle classification every **150 ms**
- ▶ **APC**: computes flight-plan update every **100 ms**

These periodic events continue unless interrupted.

Event: Obstacle Detected (FPS)

When FPS detects an object within **2.0 m**:

1. Sends **CloseObstacle** to APC immediately
2. APC must respond within **25 ms** with **Brake** to MC
3. MC performs braking maneuver for **200 ms**
4. MC signals **BrakeComplete** when done

MC braking may be interrupted later by emergency events.

Event: Rapid Altitude Drop (ALT)

If ALT senses a drop > **0.5 m within 100 ms**:

1. ALT sends **RapidDrop** to APC
2. APC must send **Ascend** to MC within **40 ms**
3. MC applies upward thrust for **300 ms**

APC's internal update cycle continues during this.

Event: Vision Override (VPU)

If VPU classifies an obstacle as **Critical**:

1. APC overrides all previous commands
2. Sends **EmergencyStop** to MC within **15 ms**
3. APC broadcasts **CriticalObstacle** to FPS, ALT, and VPU
4. MC halts any action (braking or ascending) immediately

This takes precedence over all other behaviors.

Interaction Rules

- ▶ FPS, ALT, and VPU continue periodic outputs at their frequencies
- ▶ Motor actions (**Braking, Ascending**) last assigned durations
- ▶ **EmergencyStop** interrupts any motor maneuver
- ▶ APC deadlines must be shown (25 ms, 40 ms, 15 ms)
- ▶ Students must show overlapping timing (e.g., Critical detected during braking)

Student Task

Create a **UML Timing Diagram** showing:

- ▶ All five lifelines
- ▶ State/value timelines for each component
- ▶ Periodic readings (50 ms, 150 ms, 100 ms)
- ▶ Event-driven transitions:
 - ▶ CloseObstacle
 - ▶ RapidDrop
 - ▶ Brake, Ascend, EmergencyStop
 - ▶ CriticalObstacle
- ▶ Duration constraints:
 - ▶ Braking: **200 ms**
 - ▶ Ascending: **300 ms**
- ▶ APC deadlines: **25 ms, 40 ms, 15 ms**
- ▶ At least **one interrupt case**
- ▶ Time markers (t0, t1, t2...)

Tips

- ▶ Start by placing time axis and the five lifelines
- ▶ Add periodic outputs first
- ▶ Insert event-driven triggers
- ▶ Represent deadlines clearly
- ▶ Use timing marks ($t_0, t_1\dots$) for clarity
- ▶ Ensure emergency paths override other states cleanly

Communication Diagrams

Communication Diagrams — Quick Refresher

Purpose Show interactions between objects in a system and their message flows

Focus Relationships and links, not time sequence

Key Concepts

Objects/Actors represented as rectangles with names

Links lines connecting objects

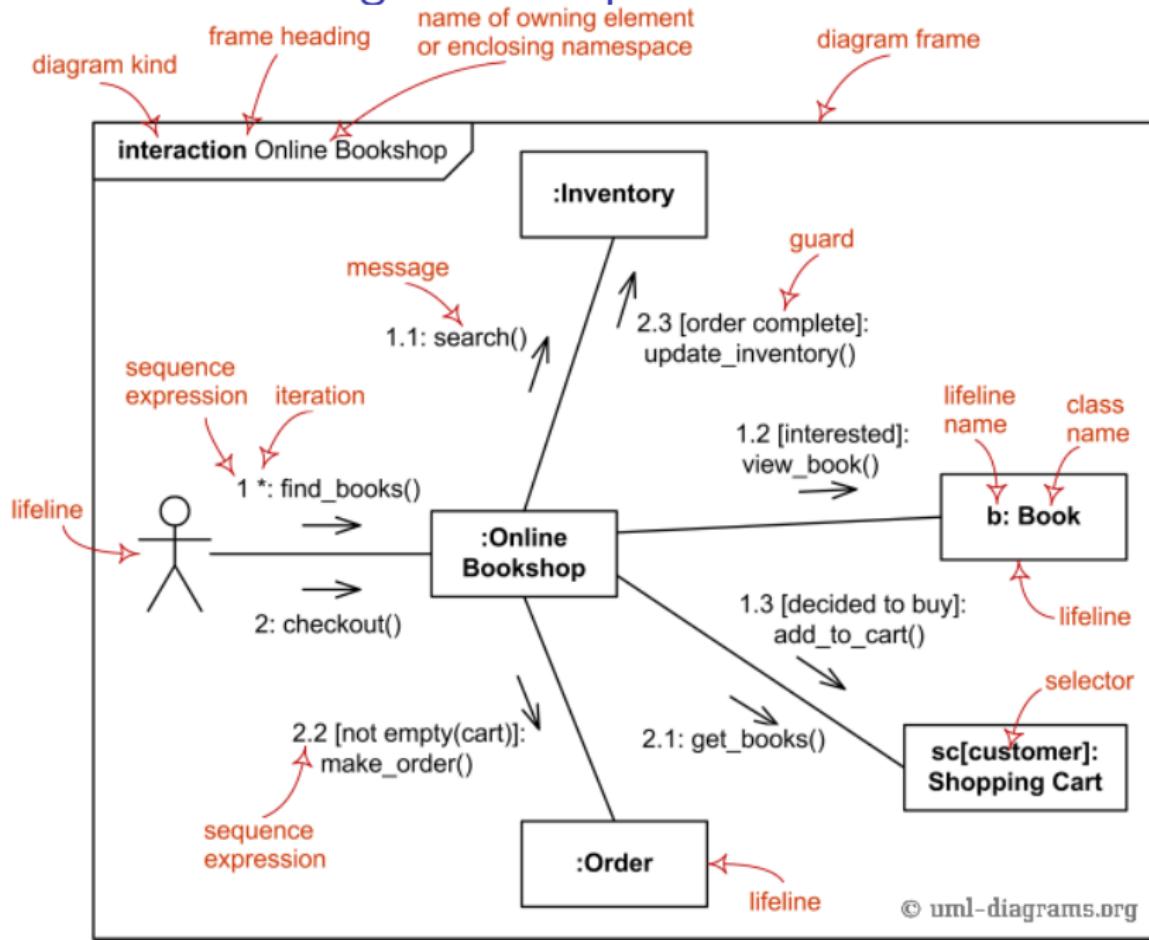
Messages arrows along links, optionally numbered to indicate sequence

Sequence numbers to show nested/parallel interactions
(e.g., 1, 1.1, 1.2, 2)

Use Cases

- ▶ Modeling collaborative behavior
- ▶ Complementary to Sequence Diagrams
- ▶ Useful for analyzing message paths and responsibilities

Communication diagram example



Communication Diagram modelling exercise (summary)

Model two **autonomous warehouse robots** that negotiate access to a **shared loading bay**, using a **UML Communication Diagram**.

Scenario Overview

A distributed system coordinates **autonomous warehouse robots** that negotiate access to a **shared loading bay**.

Key components:

- ▶ **Robot A**
- ▶ **Robot B**
- ▶ **Bay Controller (BC)**
- ▶ **Task Scheduler (TS)**
- ▶ **Collision Monitor (CM)**
- ▶ **Fleet Manager (FM)**

Robots must request access, negotiate conflicts, and handle overrides and safety events.

Normal Behavior

1. Robot A and Robot B each send **AccessRequest** to the Bay Controller (BC).
2. BC queries the Task Scheduler (TS) for priority values.
 - ▶ TS returns **priority scores** for each robot.
3. BC grants the bay to the robot with higher priority and sends **AccessGranted**.
4. BC sends **AccessDenied** to the losing robot.

Negotiation Phase

If both robots have **equal priority**:

1. BC initiates a **TieBreak** procedure:
 - ▶ BC sends *NegotiationStart* to both robots.
2. Robots exchange **Proposal** and **CounterProposal** messages directly.
3. After exchanging proposals, both robots send **FinalOffer** to BC.
4. BC selects the best offer and grants access accordingly.

Safety Override Behavior

At any time:

- ▶ The **Collision Monitor (CM)** may send **ProximityAlert** to:
 - ▶ Robot A
 - ▶ Robot B
 - ▶ Bay Controller (BC)

On receiving **ProximityAlert**:

- ▶ BC must immediately send **AbortNegotiation** to both robots.
- ▶ BC informs the **Fleet Manager (FM)** with **SafetyEventReport**.
- ▶ FM sends **StandDown** to both robots to halt movement.
- ▶ After CM clears the danger, FM sends **ResumeOps**.

Additional Constraints

Students must incorporate:

- ▶ **Numbered message sequences** typical of communication diagrams.
- ▶ **Message ordering** within:
 - ▶ Normal access negotiation
 - ▶ Tie-break protocol
 - ▶ Safety override interrupt sequence
- ▶ **Conditional messages** (equality of priority).
- ▶ **Loops** for proposal exchanges (Proposal - CounterProposal).
- ▶ **Asynchronous safety interrupt** that can occur at any point.

Task: Produce a UML Communication Diagram showing

1. Objects / Lifelines

- ▶ RobotA, RobotB, BC, TS, CM, FM

2. Links

- ▶ Show communication paths (e.g., BC - TS, Robot - Robot, etc.)

3. Message Flows

- ▶ Access request cycle
- ▶ Priority request to TS
- ▶ Tie-break negotiation if required
- ▶ Safety override sequence

4. Message Numbering

- ▶ Use hierarchical numbers (e.g., 1, 1.1, 1.2, 2, 3.1, 3.1.1...)

5. Conditional & Loop Indicators

- ▶ Show repeated proposal exchanges
- ▶ Show priority-equality decision branches

6. Interrupt Modeling

- ▶ Show how **ProximityAlert** interrupts negotiation
- ▶ Show BC → Robot abort sequence
- ▶ Show BC → FM reporting
- ▶ Show FM's commands