UML Practice Session Report
Activity & Sequence Diagrams

Mohamed Amine El Kalai

October 23, 2025

1 Introduction

This report presents the solutions to four exercises on UML modeling. The exercises focus on two main types of diagrams: Activity Diagrams and Sequence Diagrams. These diagrams are important tools for showing how systems work and how different parts interact with each other.

2 Exercise 1: Car Repair Management System

2.1 Problem Description

This exercise asks us to model the process of creating a repair record in a car workshop. The workshop manager uses software to search for cars, check warranty status, enter repair dates, and handle insurance information.

2.2 Solution Approach

The solution uses an Activity Diagram with partitions (also called swimlanes). The diagram has two partitions:

- **Manager**: Shows the actions done by the workshop manager
- **Management Software**: Shows the actions done by the system

2.3 Key Elements in the Diagram

- **Start point**: The manager begins by searching for a car
- **Decision points**: The diagram includes several decisions:
  - Does the car exist in the system?
  - Is the car under warranty?
  - Is the repair paid by insurance?
- **Actions**: Different steps like entering dates, selecting insurance, and providing car information

- **End point**: The process ends when the repair form is recorded

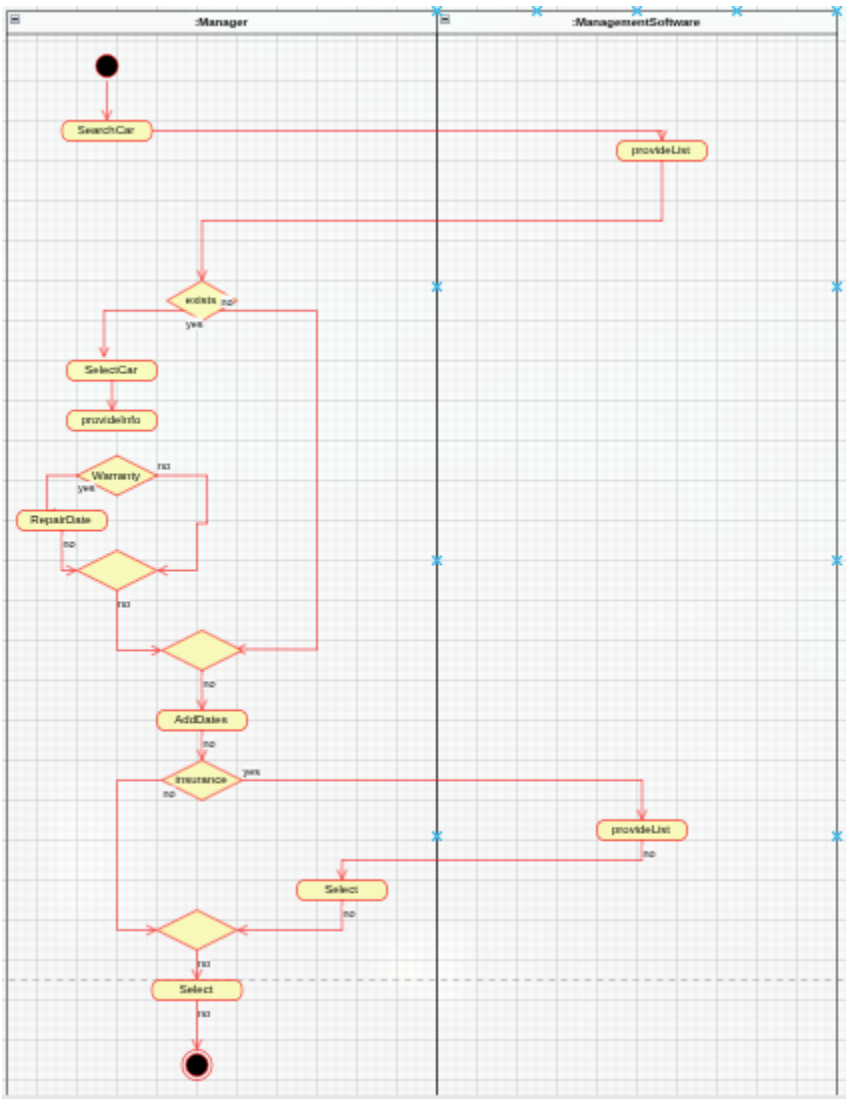

Figure 1: Activity Diagram for Car Repair Management System

3 Exercise 2: Minesweeper Game

3.1 Problem Description

This exercise models what happens when a player reveals a cell in the Minesweeper game. There are three possible outcomes:

- The cell contains a mine (game over)
- The cell shows a number (check if player won)
- The cell is empty (reveal all nearby cells)

3.2 Solution Approach

The solution uses a Sequence Diagram to show the interaction between the Interface and the Cell object. The diagram shows the messages sent between these objects based on what the cell contains.

3.3 Key Elements in the Diagram

- **Actors**: Interface and Cell
- **Alt fragment**: Shows the three different scenarios (mine, numbered, empty)
- **Opt fragment**: Shows optional behavior when player wins
- **Loop fragment**: Shows recursive revealing of neighbor cells when cell is empty

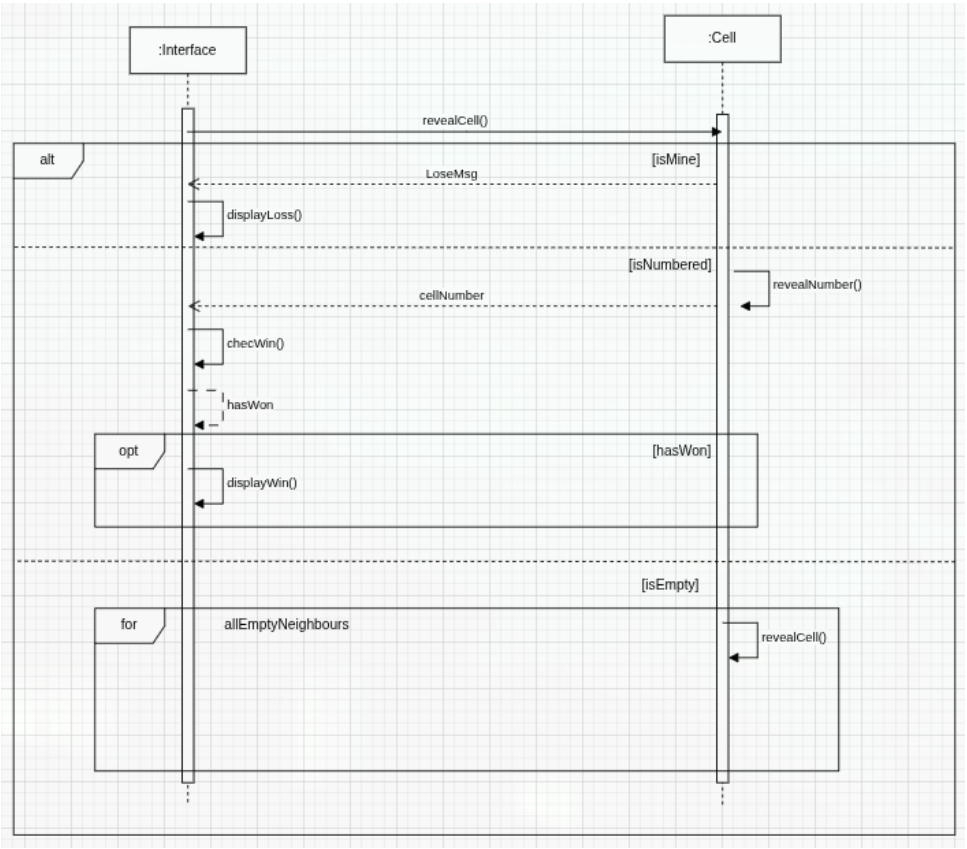

Figure 2: Sequence Diagram for Minesweeper Game

4 Exercise 3: Chocolate Mousse Recipe

4.1 Problem Description

This exercise asks us to model a chocolate mousse recipe as an activity diagram. Some steps happen at the same time (concurrent activities) while others must happen in order.

4.2 Solution Approach

The solution uses an Activity Diagram with fork and join nodes to show which steps can happen at the same time and which must wait for others to finish.

4.3 Key Elements in the Diagram

- **Fork node**: Splits the flow into two parallel activities:
  - Breaking chocolate into pieces, then melting it
  - Breaking eggs and separating whites from yolks
  - whisking whites
- **Join node**: Waits for mutiple parallel paths to finish before continuing

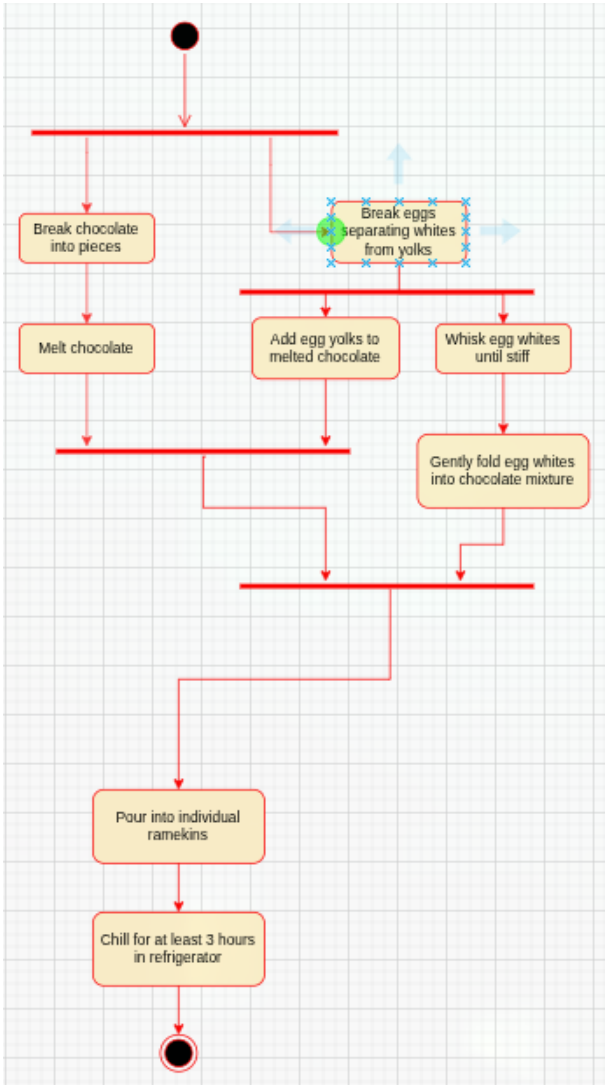

Figure 3: Activity Diagram for Chocolate Mousse Recipe

5 Exercise 4: Library Management System

5.1 Problem Description

This exercise has two scenarios:

1. **Check Overdue**: The system checks if a member has any late book returns
2. **Borrow a Book**: A member tries to borrow a book from the library

5.2 Solution Approach

Two separate Sequence Diagrams are created, one for each scenario. The diagrams show the interaction between objects like Application, Member, Book, Loan, and Librarian.

5.3 Scenario 1: Check Overdue

Participants: Application, Member, Loan
Flow:

1. Application calls getLoans() on Member
2. Member returns the list of loans
3. Loop through each loan:
    - Call isReturnDateOverdue() on Loan
    - Loan returns true or false (isOverdue)
    - If overdue (opt fragment): Call setStatus(SUSPENDED) on Member

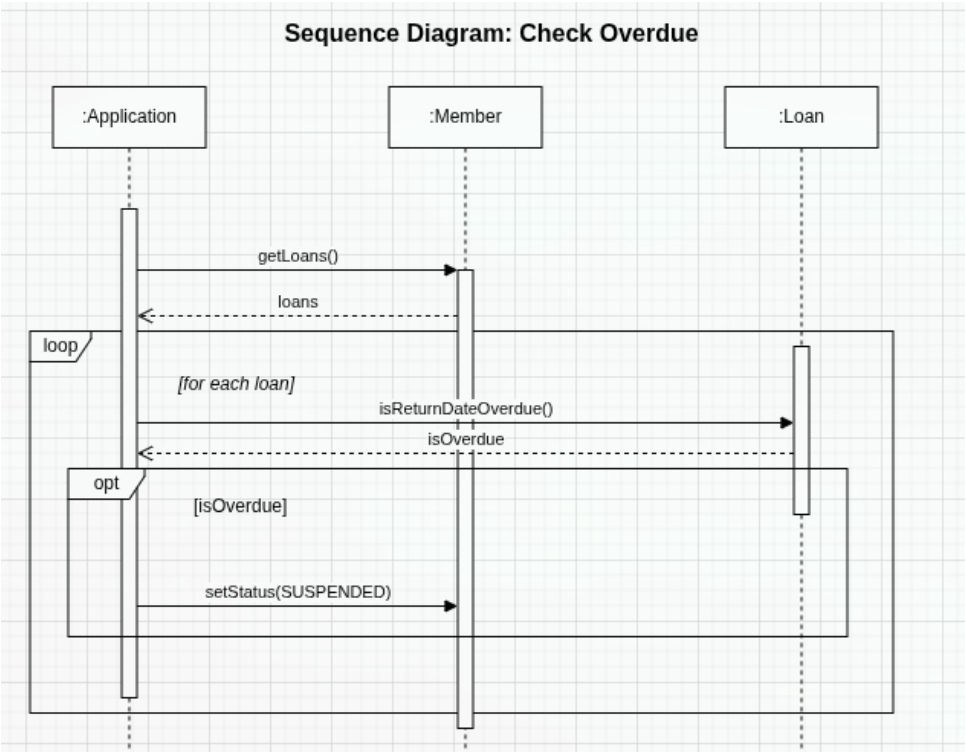

Figure 4: Sequence Diagram for Check Overdue Scenario

5.4 Scenario 2: Borrow a Book

Participants: Librarian, Application, Member, Book, Loan
Flow:

1. Librarian calls selectBorrow() on Application
2. Application calls Check Overdue (reference to other diagram) and updates status if so
3. Application calls getStatus() on Member to check if member can borrow
4. Member returns status
5. Application checks if book is available with isAvailable()
6. Alternative fragment (alt):
    - If status is maxBorrow or suspended or book not available: Application sends refusalMessage to Librarian
7. If everything is fine:
    - Application creates a new Loan with createDate, returnDate, memberId, and bookId
    - Application calls setUnavailable() on Book
    - Application calls incrementNbBooksBorrowed() on Member
    - Application calls updateStatus() on Member

- Application sends successMessage to Librarian

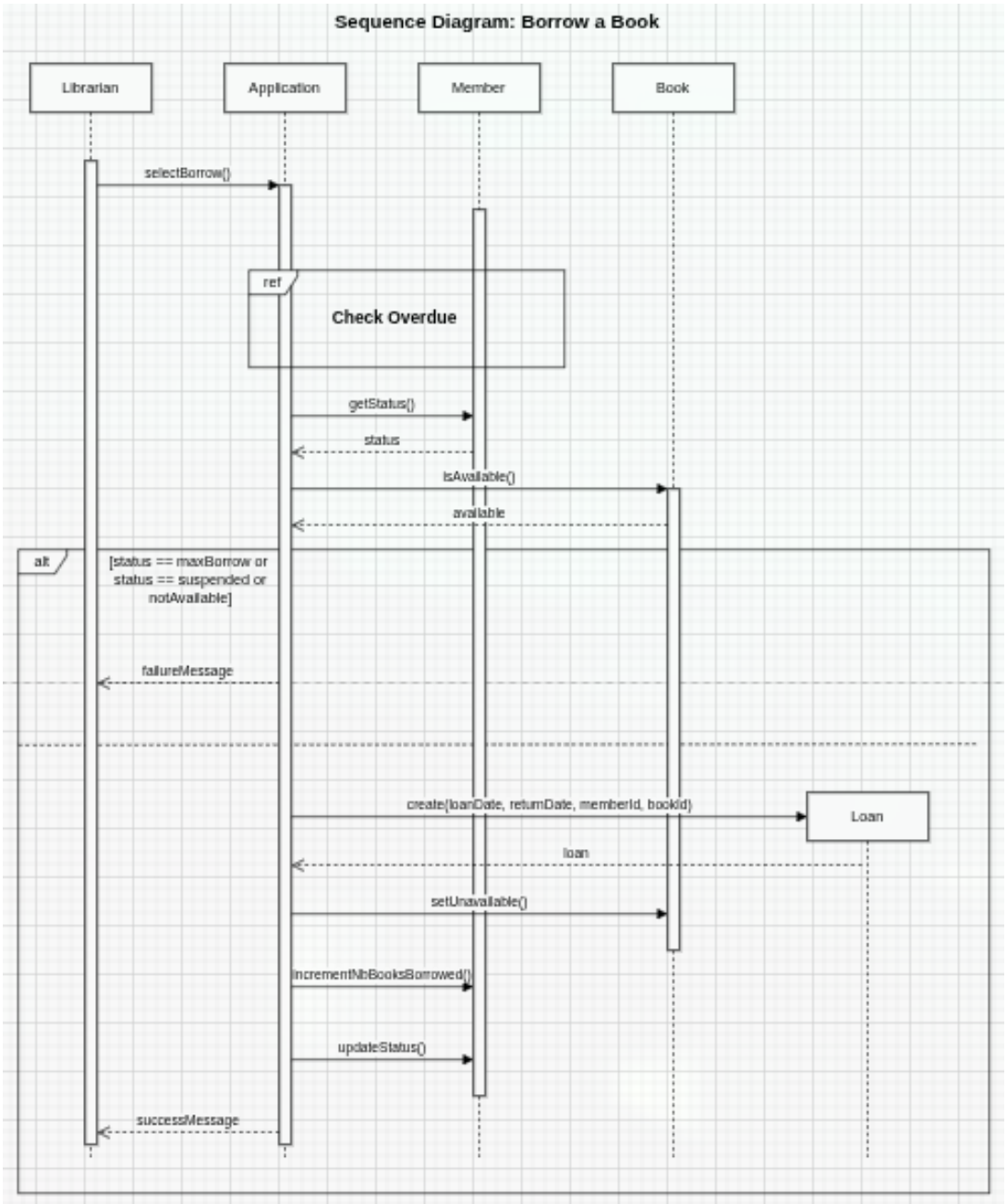

Figure 5: Sequence Diagram for Borrow a Book Scenario

6 Conclusion

This practice session helped me understand important UML diagrams used in software design. Through solving these exercises, I learned from several common mistakes:

Common Mistakes and Lessons:

- **Forgetting partitions in activity diagrams**: At first, I didn't clearly separate the responsibilities between actors. Using swimlanes made it much clearer who does what in the process.

- **Missing the fork and join nodes**: In Exercise 3, I initially drew everything sequentially. I learned that fork and join nodes are necessary to show tasks that can happen at the same time.
- **Not including all necessary checks**: In Exercise 4, I forgot to check member status multiple times. Real systems need to verify conditions at different points to work correctly.