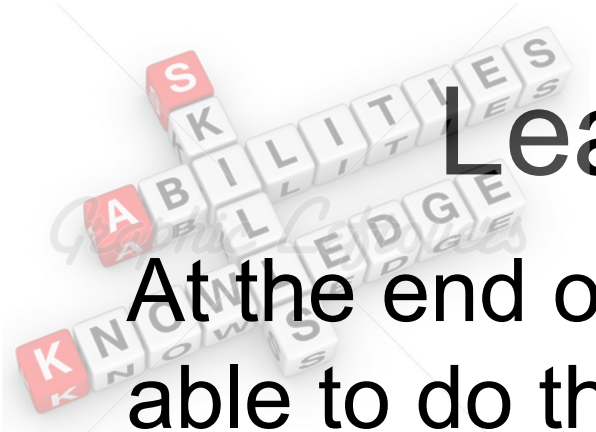
50.003
Elements of Software Construction
Lecture 5

UML Sequence Diagram 2
Software Architecture

Scope

- 3-layer architecture –
  - View layer, business logic layer, data layer
  - Advantages
  - Disadvantages
- Responsibilities of the layers
- Interaction of objects within a sequence diagram
- Iterative nature of the analysis and design process
- Case study



Learning Outcomes

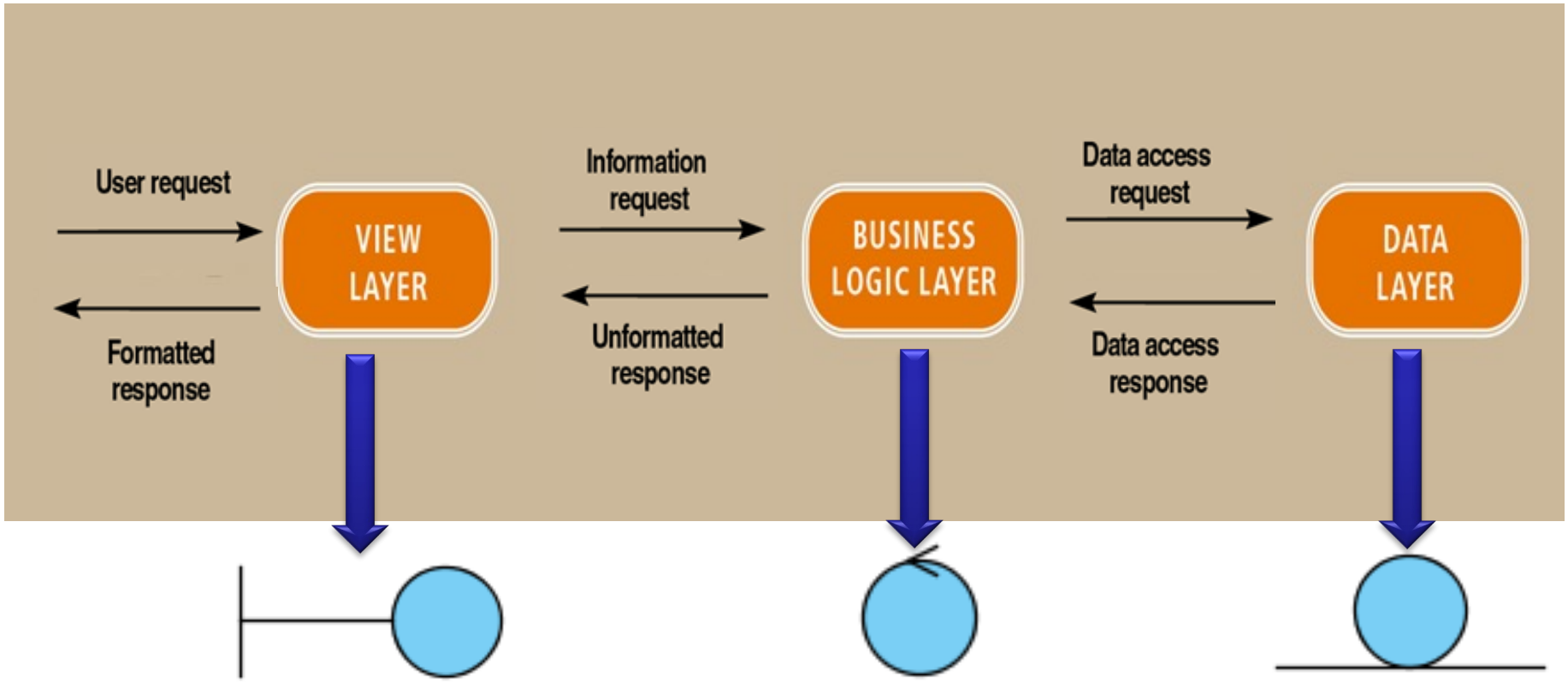
At the end of this session, you should be able to do the following:

- explain the 3-layer architecture
- explain the advantages and disadvantages of the 3-layer architecture
- describe the interaction of objects within a sequence diagram
- justify the Iterative nature of the analysis and design process

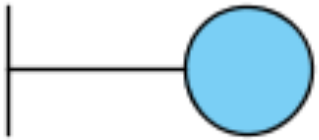
3 Layer Architecture

- Divides application software into independent processes
- Three-layers:
  - The data layer
  - The business logic layer
  - The view (presentation) layer

3 Layer Architecture



View layer (boundary object) responsibilities



Boundary object

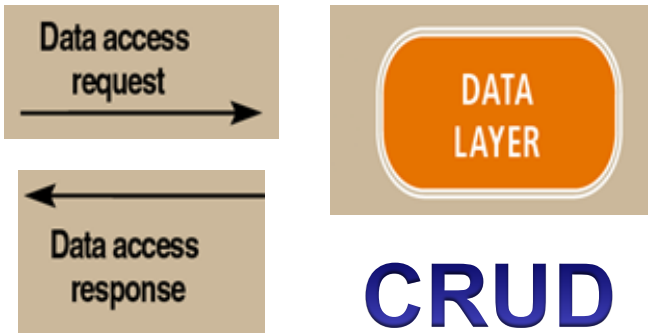
- Accepts and validates submission from user
- Relay service requests to business logic layer
- Receives unformatted response from business logic layer
- Format and displays responses to the user

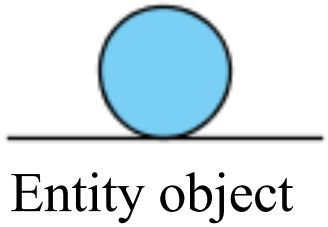
Business Logic layer (control object) responsibilities

Control object

- Receives service request from the view layer
- Relay data access request to the data layer
- Receives and process data access response from the data layer
- Return unformatted response to view layer

Data layer (entity objects) responsibilities





- Receives data access requests (insert, retrieve, update, delete) from the business logic layer
- Carry out the data access requests (insert, retrieve, update, delete) in the database
- Return an appropriate data access response to the business logic layer

Advantages of the 3 Layer Architecture

- Separation of concerns.
  - Each tier has designated and specific purpose
- Code reuse
  - All business logic can be defined once within the business layer and then shared by any number of components within the presentation layer
- Easy to implement changes
  - Changes in the contents of any one of tiers (layers) can be made without having to make corresponding changes in any of the others
- Enables parallel development of the different tiers of the application.

Advantages of the 3 Layer Architecture

- Improved data integrity
  - The business logic layer can ensures that only valid data is allowed to be updated in the database
- Improved security
  - The presentation layer/client does not have direct access to the database
  - The database structure is hidden from the caller.

Disadvantages of the 3 Layer Architecture

- More complex structure
- Communication between the tiers may moderately affect performance

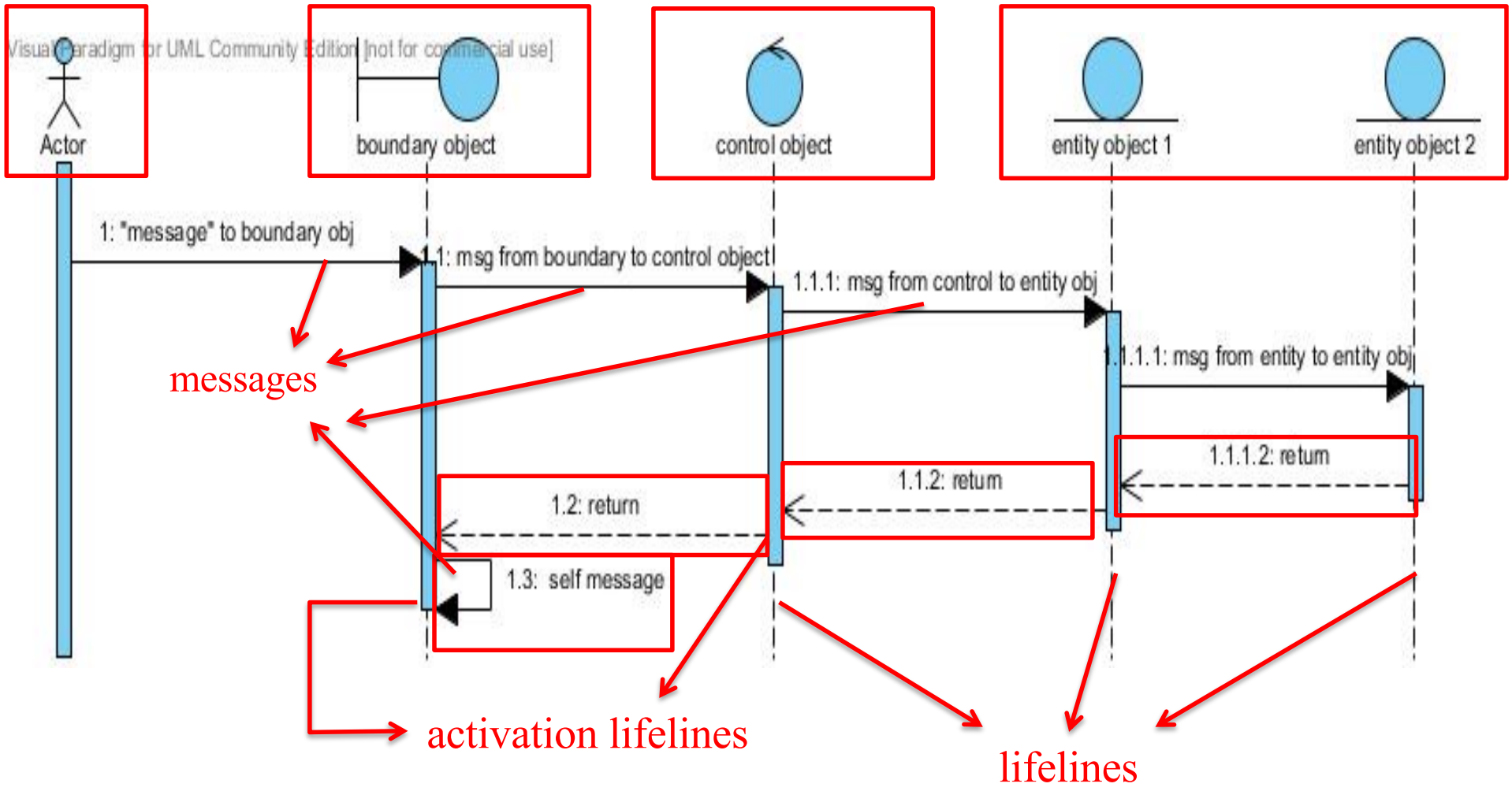
Elements of a Sequence diagram

- Only one boundary object
  - Normally just append the word "form" to the use case name
- Only one control object
  - Normally just append the word "controller" to the use case name
- One or more entities objects
  - Identified from the domain class diagram

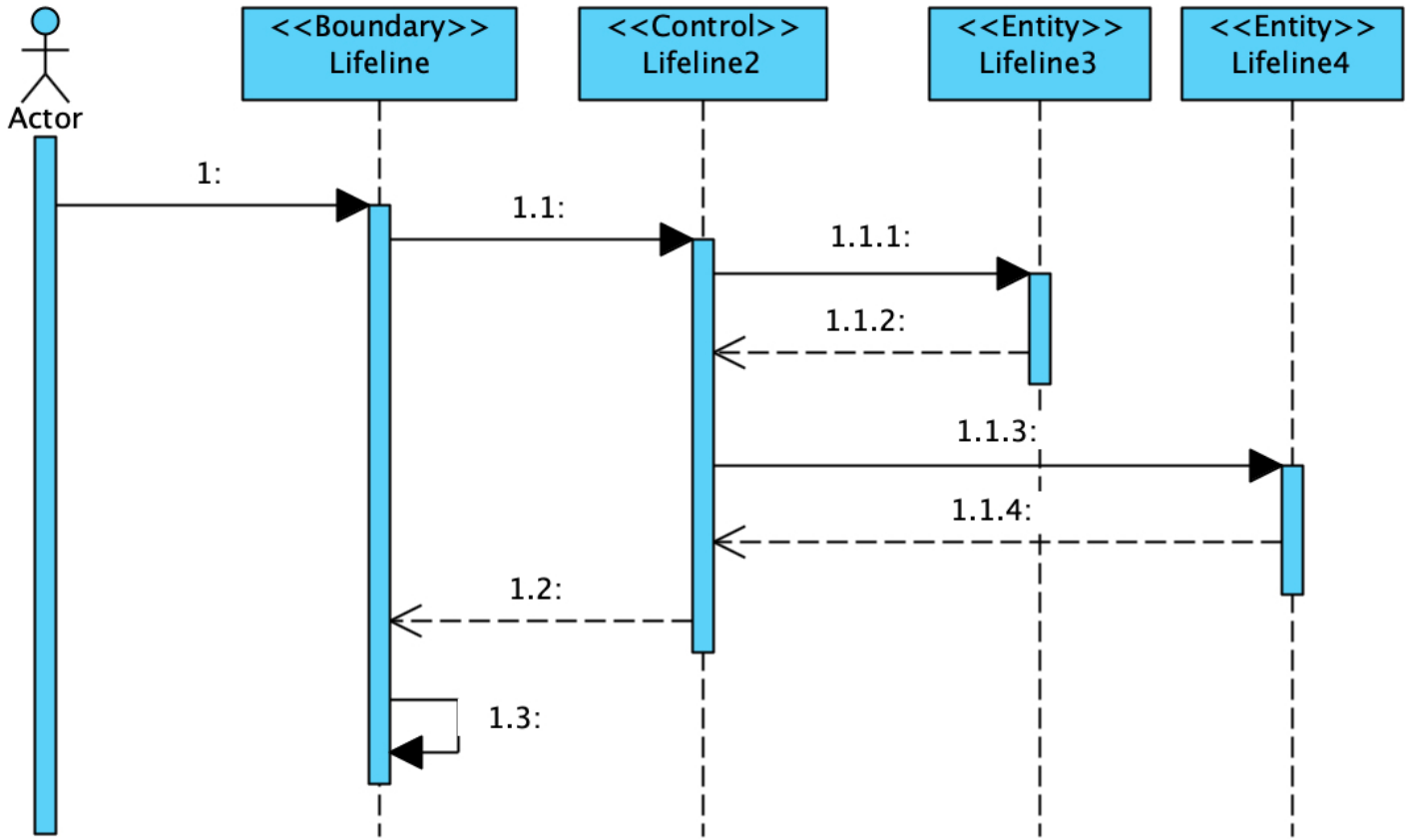
Interactions between analysis object in a Sequence diagram

- Actor can only interacts (i.e. send messages to) with the boundary object Boundary object can only interacts (i.e. send messages to) with controller object
- Controller object can only interacts (i.e. send messages to) with entity objects
- Entity objects can only interact (i.e. send messages to) with other entity objects

Elements of a Sequence diagram

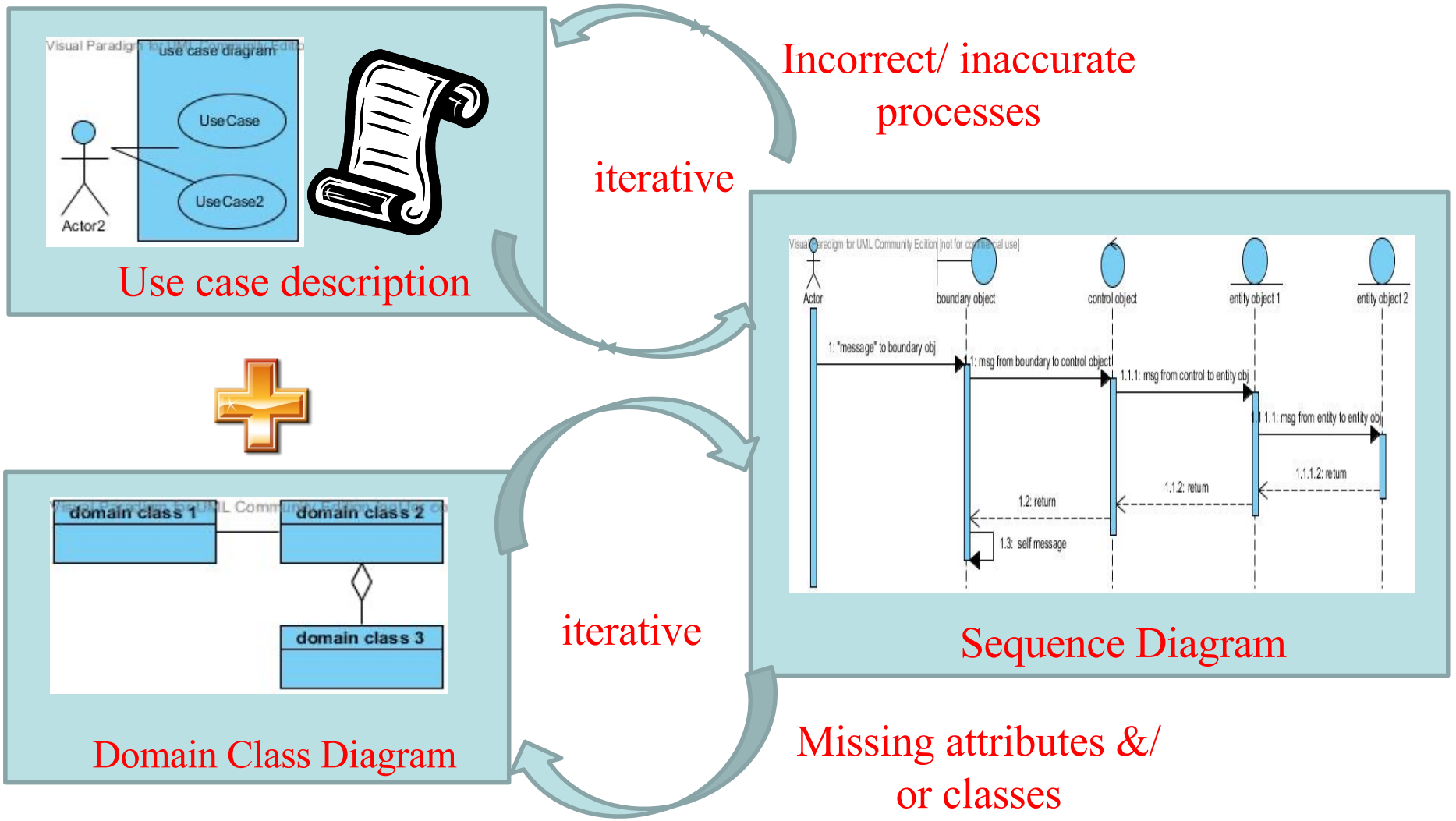


Alternate Representation



Using Stereotypes

Iterative nature of OOAD



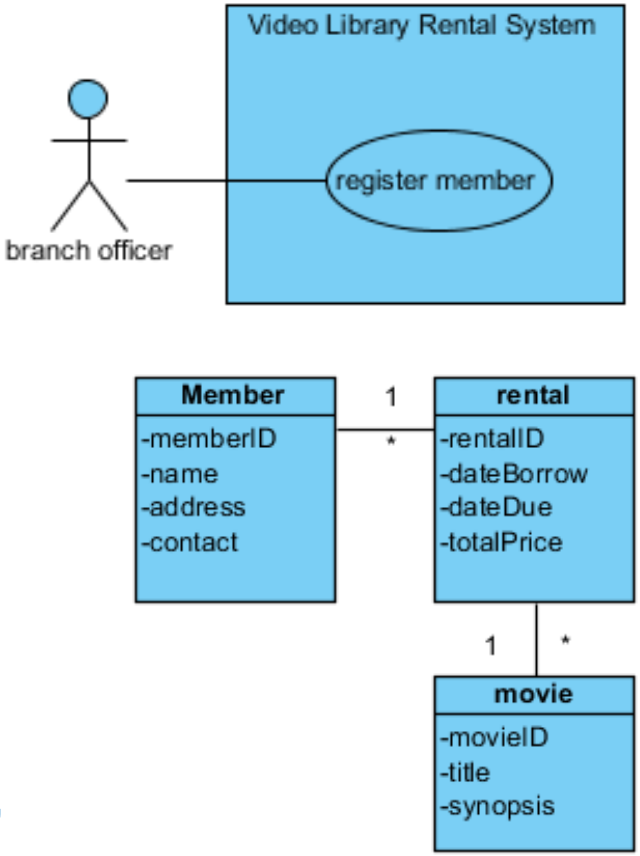
Case Study : Register Member Use Case

Consider the **Register Member** use case of the Video Rental System.

Basic Flow:

1. The branch officer enters and submits the customer details such as NRIC, name and contact information.
2. The system validates the submitted information.
3. The system creates a new member record with a unique id.
4. The system displays a “successful registration” message with the new member id.

Next -> Identify the classes needed (boundary, control and entity)



Case Study : Register Member Use Case - Identifying analysis objects

- Analysis classes identified are:
  - RegistrationForm **(boundary)**
  - RegistrationController **(control)**
  - Member **(entity) –** why only the Member class?
- They are represented as follows:

Next -> Identify the responsibilities

Case Study : Register Member Use Case - Assigning Responsibilities

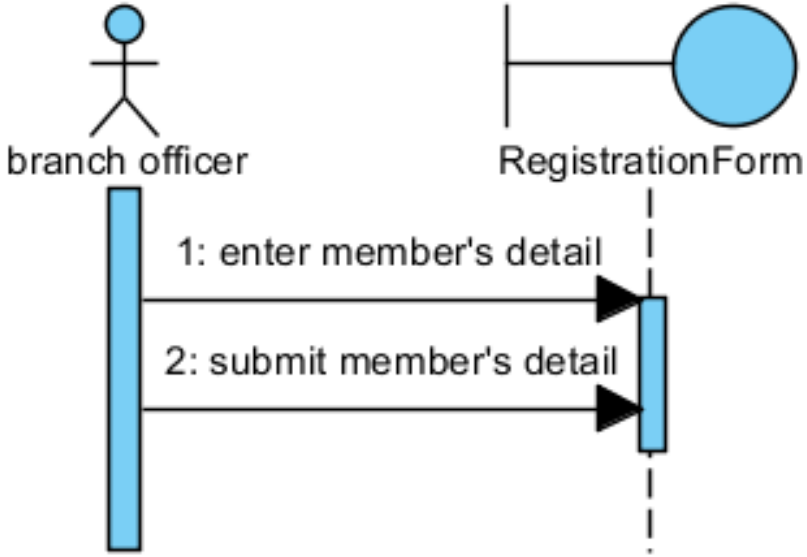
Based on the responsibilities of the 3 layers :

- **RegistrationForm boundary class**
  - Validates and accepts submission from actors
  - Displays responses from the controller
- **RegistrationController control class**
  - Receives service request from the RegistrationForm boundary object and delegates to Member entity object to provide necessary member details
- **Member entity class**
  - Receives request form the RegistrationController object to create a record to store the new member information.

Case Study : Register Member Use Case - Assigning Responsibilities

Step in the Register Member use case description

1. The branch officer enters and submits the customer details such as NRIC, name and contact information.

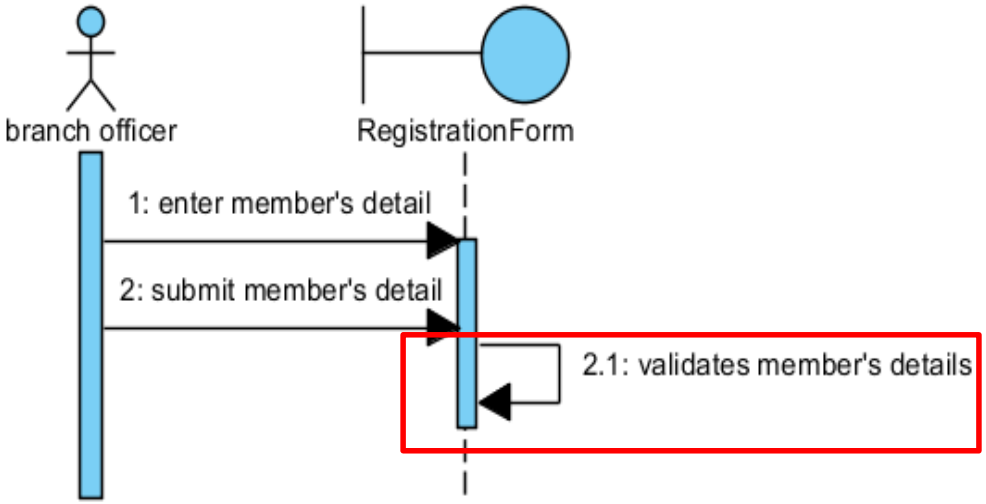


1. We assign a responsibility to the boundary object – **RegistrationForm** – to handle the submission of customer details.
2. This responsibility is indicated as **//submit customer's details,** and it is pointing to the RegistrationForm.

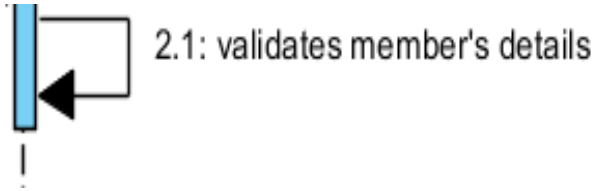
Case Study : Register Member Use Case - Assigning Responsibilities

Step in the Register Member use case description

2. The system validates the submitted information.



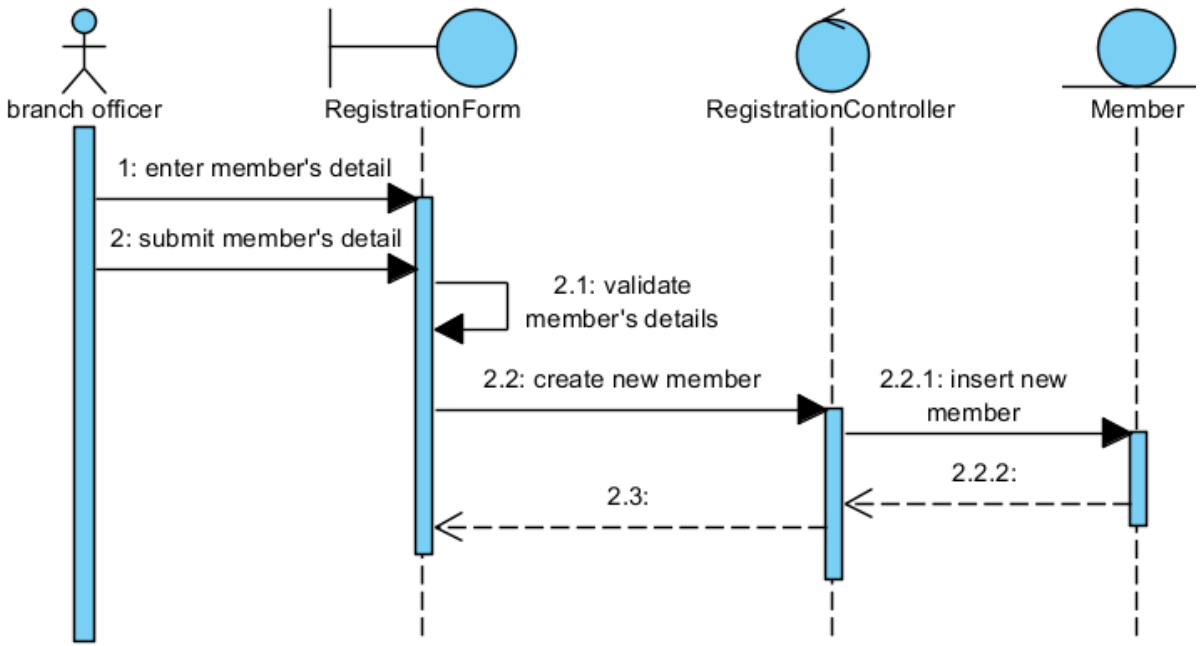
1. Since the boundary object can validates the information, we assign a **self- message** to the RegistrationForm as indicated above.
2. This responsibility is represented as



Case Study : Register Member Use Case - Assigning Responsibilities

Step in the Register Member use case description

3. The system creates a new member record with a unique id.

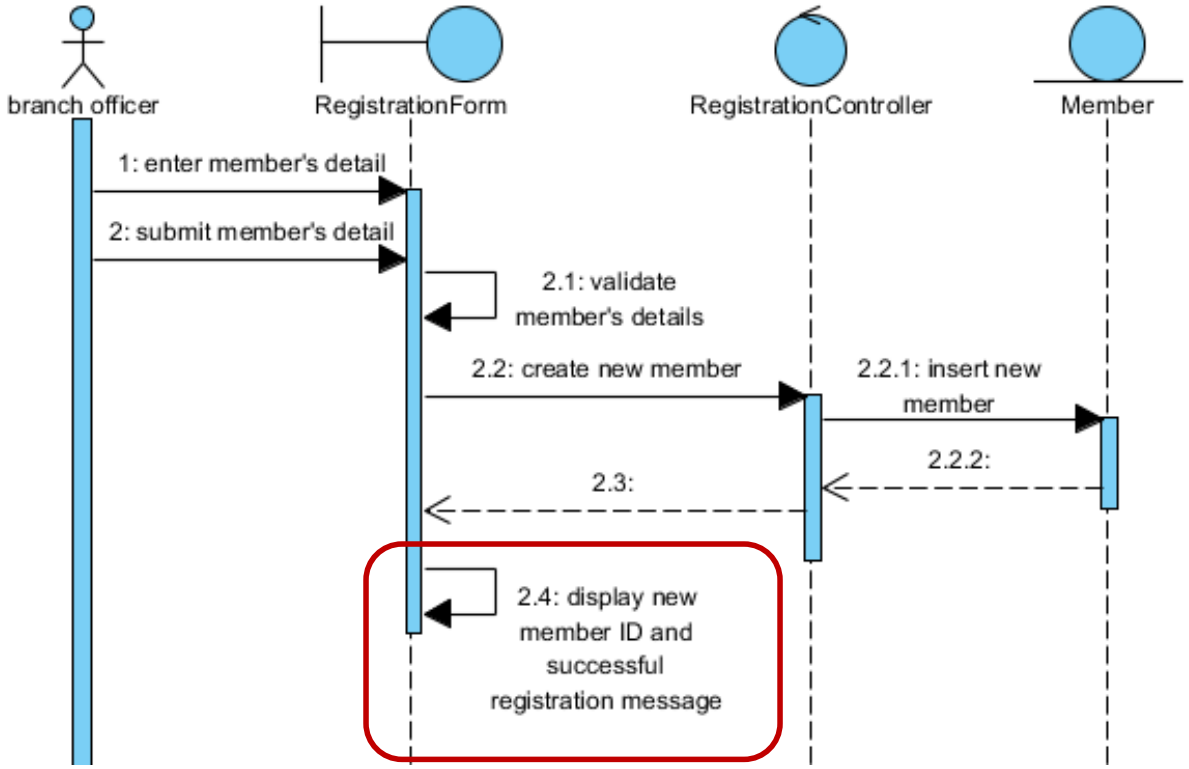


1. To create a new member, the boundary object has to pass the request to the **controller** and the controller then relay the request to the Member **entity.**
2. **2 responsibilities,** each assigned to different object are needed to complete the request
3. The entity class will return a unique member id which is created to the controller

Case Study : Register Member Use Case - Assigning Responsibilities

Step in the Register Member use case description

4. **The system displays a "successful registration" message with the new member id.**



Once the **RegistrationForm** gets valid member ID, it has the responsibility of displaying the "successful registration" message.

Case Study – rent video use case

Basic Flow :

- The staff enters and submit member ID
- The system validates member ID
- The system retrieves and displays member details
- The staff enters and submit movie ID
- The system verifies member's loan limit
- Repeat step 4 to 5 for all movie
- The system creates rental and rental item records
- The system displays a successful rental message.
- The staff acknowledges the message.

Alternate Flow

2a. Invalid member

i. The system prompt "Invalid Member" error message

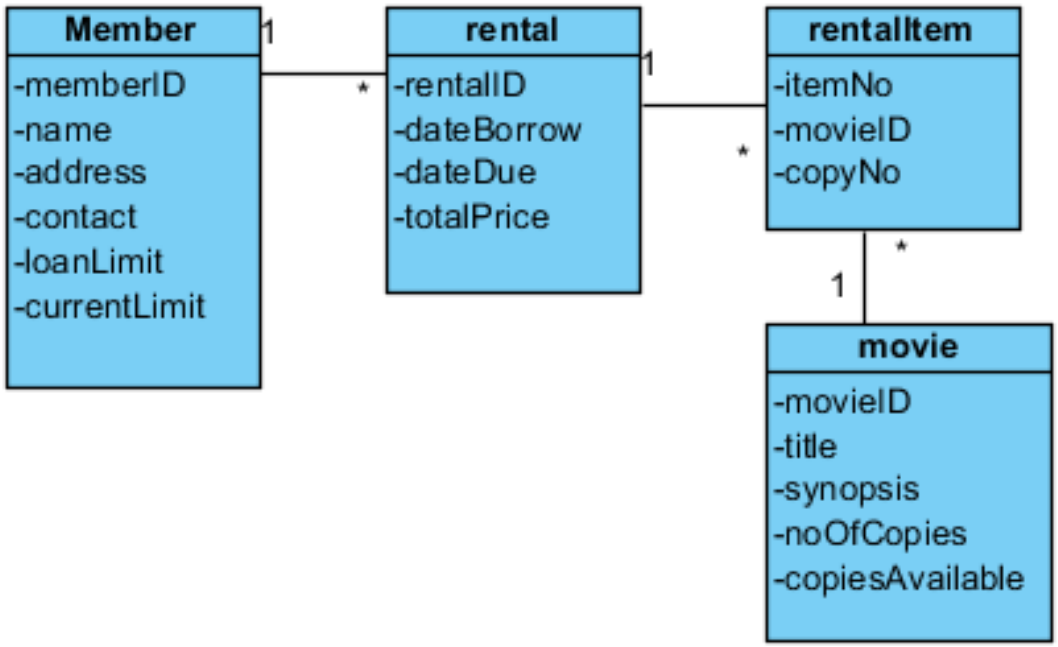
ii. Use case ends

5a. Loan Limit exceeded

i. The system will prompt "Loan Limit Exceeded" error message

ii. Use case ends.

Case Study – rent video use case



Domain Model

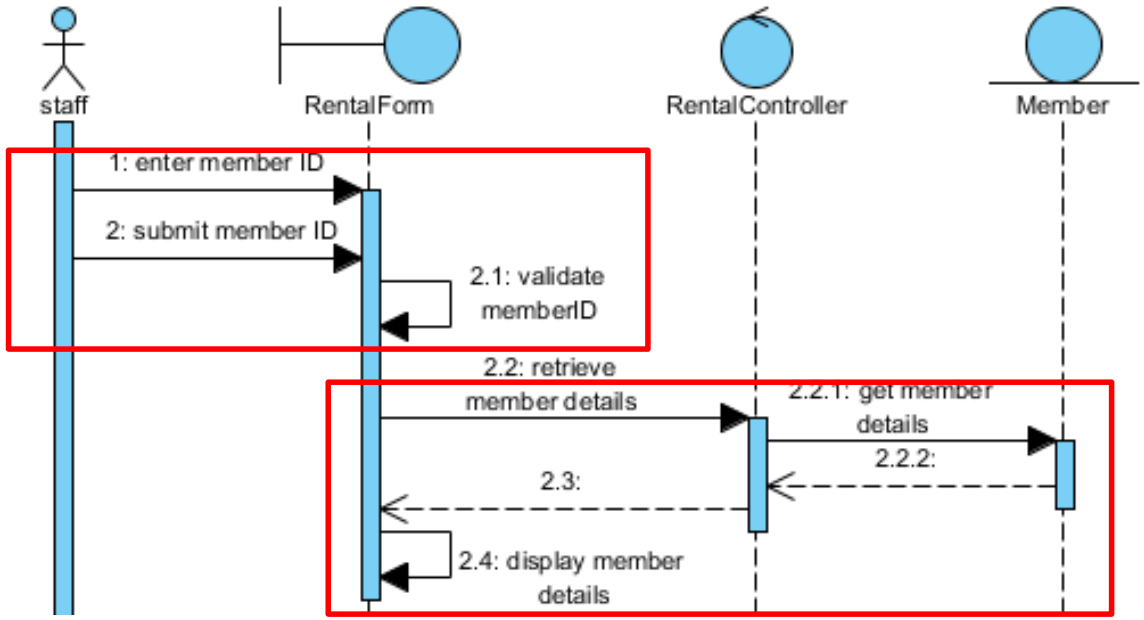
Case Study – rent video use case Analysis objects

- Rent video boundary object – RentalForm
- Rent video control object – RentalController
- Rental video entities:
  - Member – memberID, loanlimit and currentLimit
  - Movie - MovieID,
  - Rental – new rental details
  - RentalItem – new rental items details

Case Study – rent video use case

Basic Flow :

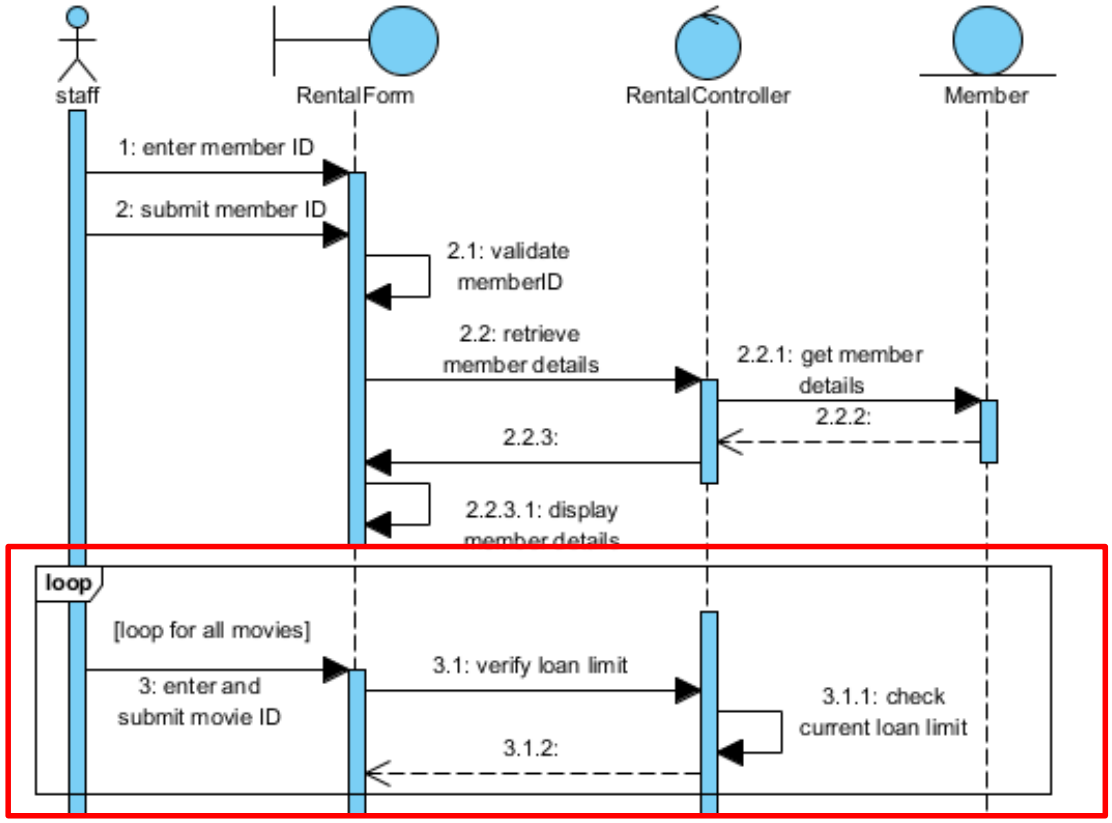
- The staff enters and submit member ID
- The system validates member ID
- The system retrieves and displays member details



Case Study – rent video use case

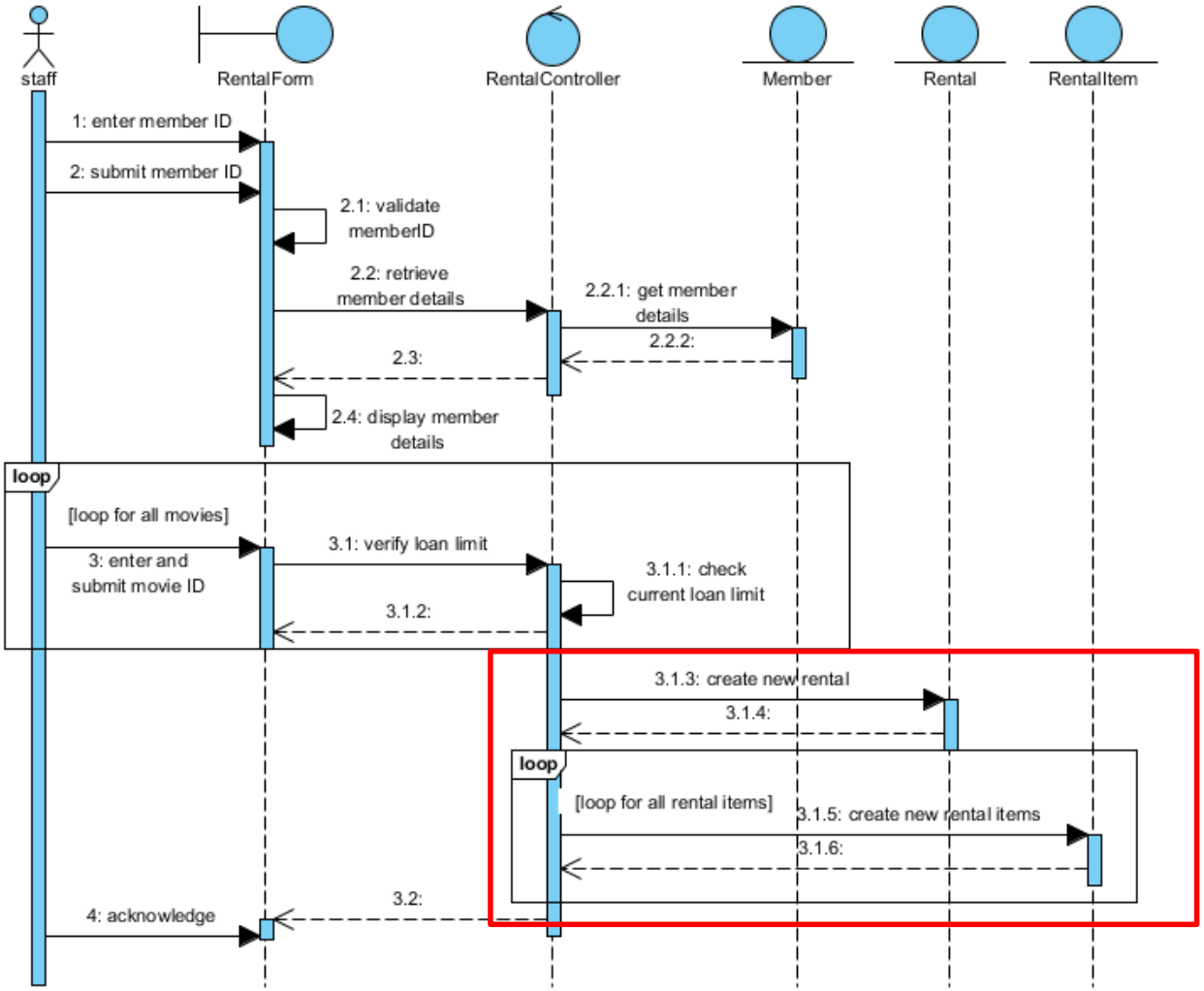
repetition

- The staff enters and submit movie ID
- The system verifies member's loan limit
- Repeat step 4 to 5 for all movie



Case Study – rent video use case

- The system creates rental and rental item records
- The system displays a successful rental message.
- The staff acknowledges the message.



Case Study – rent video use case

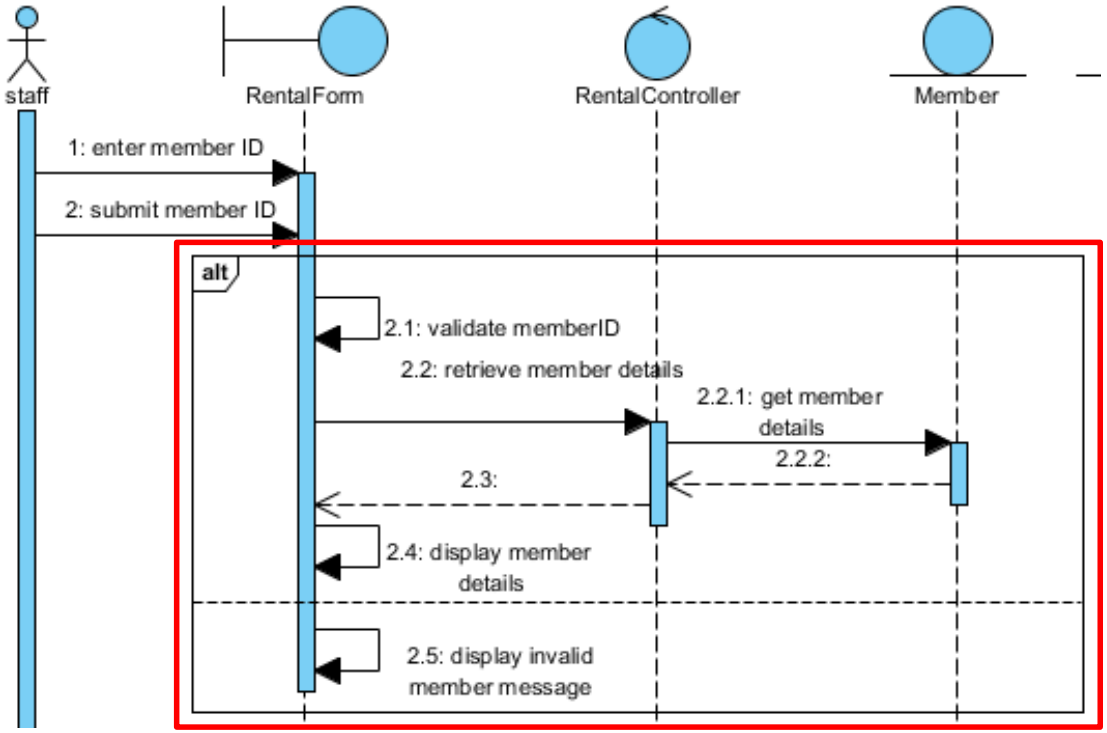
Alternate flow

Alternate Flow

2a. Invalid member

i. The system prompt "Invalid Member" error message

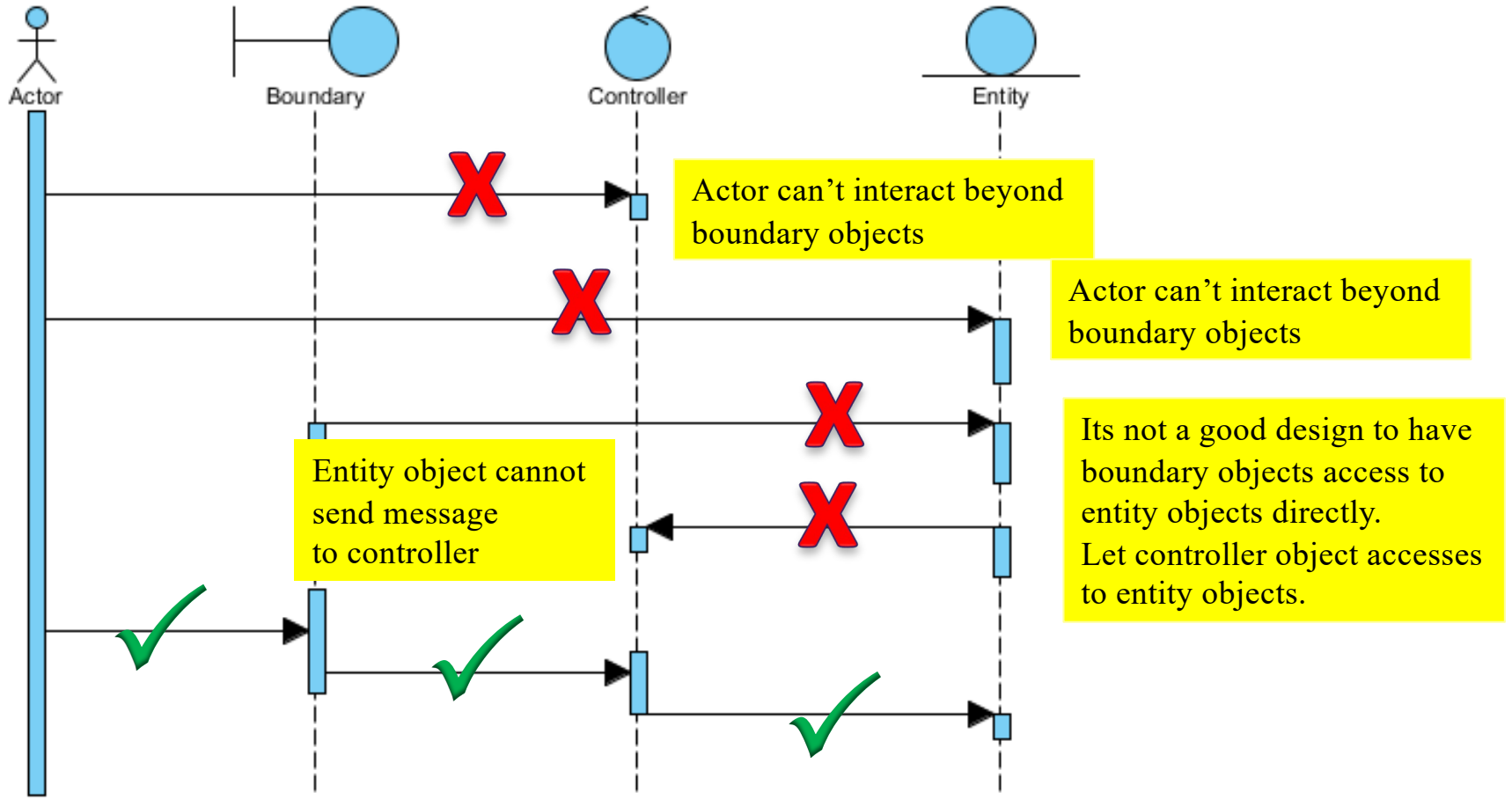
ii. Use case ends



Case Study – rent video use case

- what next?
- derive the design/solution model
- identify
  - attributes
  - operations

Common mistakes in constructing sequence diagrams



Summary

- 3-layer architecture
  - View layer
  - Business logic layer
  - Data layer
- Advantages and disadvantages of the 3-layer architecture
- Separation of concerns of the 3 layers
- Object interactions in a sequence diagram
- Constructing a sequence diagram
  - A step in the flow can be translated to a sequence of messages to different analysis objects
- Common mistakes in constructing a sequence diagram

Cohort Exercise

1. Explain the 3-tier software architecture and use a restaurant and an online eCommerce website to further illustrate your explanation.
2. In the context of the 3-tier software architecture, does it mean more tiers for more flexibility and benefits?
3. Describe the various elements that are in a sequence diagram.