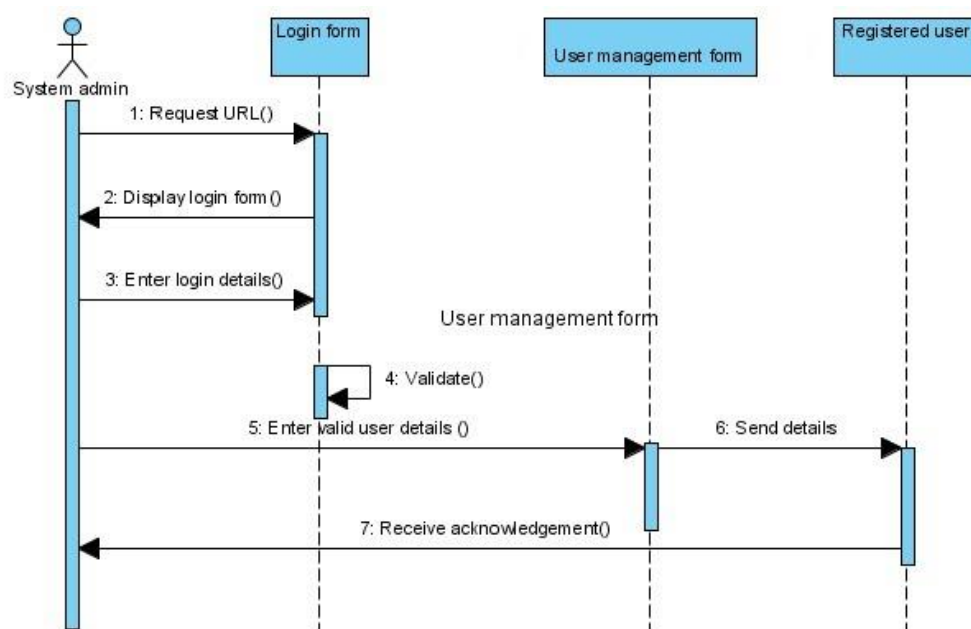


## Seminar Session 04. Sequence Diagram (using Enterprise Architect)

### Introduction

A **sequence diagram** is a visual interaction that details how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time-focused and show the order of the interaction visually by using the vertical axis of the diagram to represent time, what messages are sent and when. You can use it to:

- Depict workflow, Message passing and how elements in general cooperate over time to achieve a result.
- Capture the flow of information and responsibility throughout the system, early in analysis; Messages between elements eventually become method calls in the Class model.
- Make explanatory models for Use Case scenarios; by creating a Sequence diagram with an Actor and elements involved in the Use Case, you can model the sequence of steps the user and the system undertake to complete the required tasks.



Sequence elements are arranged in a **horizontal sequence**, with **Messages** passing back and forward between elements. **Messages on a Sequence diagram** can be of several types and can also be configured to reflect the operations and properties of the source and target elements.











An **Actor** element can be used to represent the user initiating the flow of events. Stereotyped elements, such as **Boundary**, **Control** and **Entity**, can be used to illustrate **screens**, **controllers** and **database items**, respectively. Each element has a dashed stem called a **Lifeline**, where that element exists and potentially takes part in the interactions

## Objective





In this session, you will learn how to design a **Sequence diagram** using **Enterprise Architect**. You'll learn how to add interactions between actors, objects in use cases, depending on problem description analysis.

## Sequence Diagrams in Enterprise Architect

You can include the following elements in this type of diagrams:

Element	Description
 Actor	An <b>Actor</b> is a user of the system: a human user, a machine, or even another system or subsystem in the model.
 Lifeline	A <b>Lifeline</b> represents a distinct connectable element and is an individual participant in an interaction.
 Boundary	<b>Boundary</b> elements are used in analysis to capture user interactions, screen flows and element interactions.
 Control	A <b>Control</b> organizes and schedules other activities and elements.
 Entity	An <b>Entity</b> is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.
 Fragment	A Fragment element can represents iterations or alternative processes in a Sequence diagram.
 Endpoint	An Endpoint is used in Interaction diagrams to reflect a lost or found Message in sequence.
 Diagram Gate	A Diagram Gate is a simple graphical way to indicate the point at which messages can be transmitted into and out of interaction fragments.
 State/Continuation	The State/Continuation element serves two different purposes for Sequence diagrams, as State Invariants and Continuations.
 Interaction	You can use an Interaction element to insert an Interaction diagram as a child of a Class element.

Moreover, you can use connectors such as the following:

Connector	Description
 Message	A Message indicates a flow of information or transition of control between elements.
 Self-Message	A Self-Message reflects a new process or method invoked within the calling lifeline's operation.
 Recursion	A Recursion is a type of Message used in Sequence diagrams to indicate a recursive function.
 Call	A Call is a type of Message connector that extends the level of activation from the previous Message.



## Problem Description

A library database needs to store information pertaining to:

- Its customers
- Its workers
- The physical locations of its branches,
- And the media stored in those locations (two media types are considered: books and videos).

The library must keep track of the status of each media item: its location, status, descriptive attributes, and cost for losses and late returns. Books will be identified by their ISBN, while movies by their title and year. In order to allow multiple copies of the same book or video, each media item will have a unique ID number.

Customers will provide their name, address, phone number, and date of birth when signing up for a library card. They will then be assigned a unique user name and ID number, plus a temporary password that will have to be changed.

Checkout operations will require a library card, as will requests to put media on hold. Each library card will have its own fines, but active fines on any of a customer's cards will prevent the customer from using the library's services.

The library will have branches in various physical locations. Branches will be identified by name, and each branch will have an address and a phone number associated with it. Additionally, a library branch will store media and have employees.

Employees will work at a specific branch of the library. They receive a paycheck, but they can also have library cards; therefore, the same information that is collected about customers should be collected about employees.

Functions for customers (users):

- Log in
- Search for media based on one or more of the following criteria:
  - type (book, video, or both)
  - title
  - author or director
  - year
- Access their own account information:

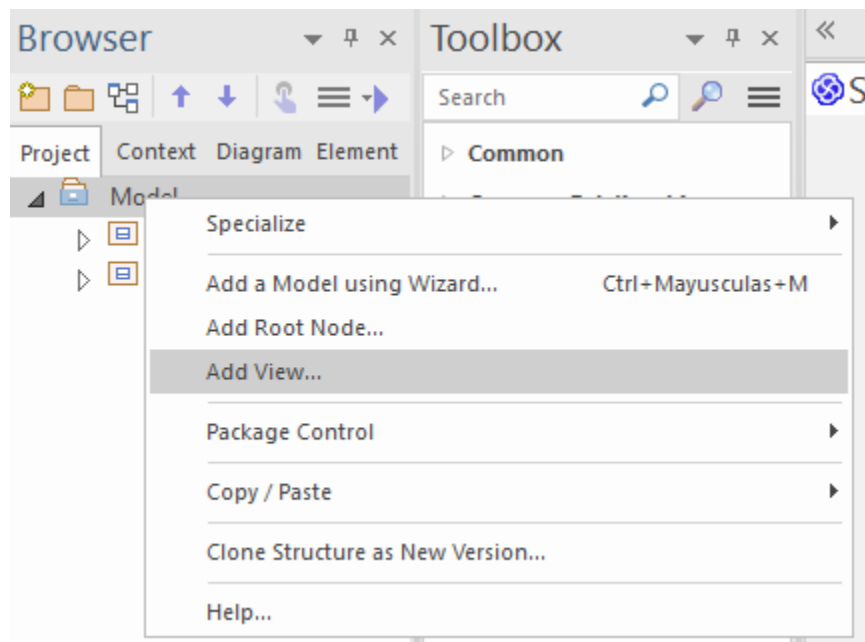
- Card number(s)
  - Fines
  - Media currently checked out
  - Media on hold
- Put media on hold
- Pay fines for lost or late items
- Update personal information:
  - Phone numbers
  - Addresses
  - Passwords

Functions for librarians (employees) are the same as the functions for customers plus the following:

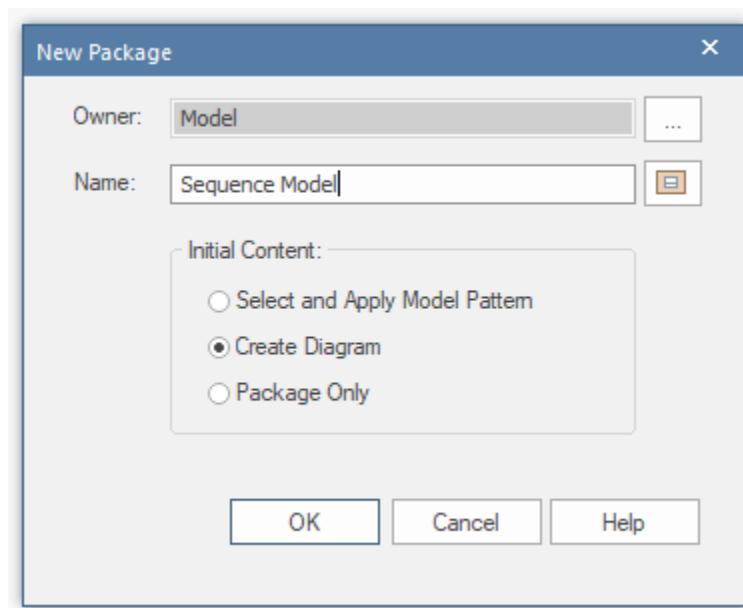
- Add customers
- Add library cards and assign them to customers
- Check out media
- Manage and transfer media that is currently on hold
- Handle returns
- Modify customers' fines
- Add media to the database
- Remove media from the database
- Receive payments from customers and update the customers' fines
- View all customer information except passwords

## Analysis

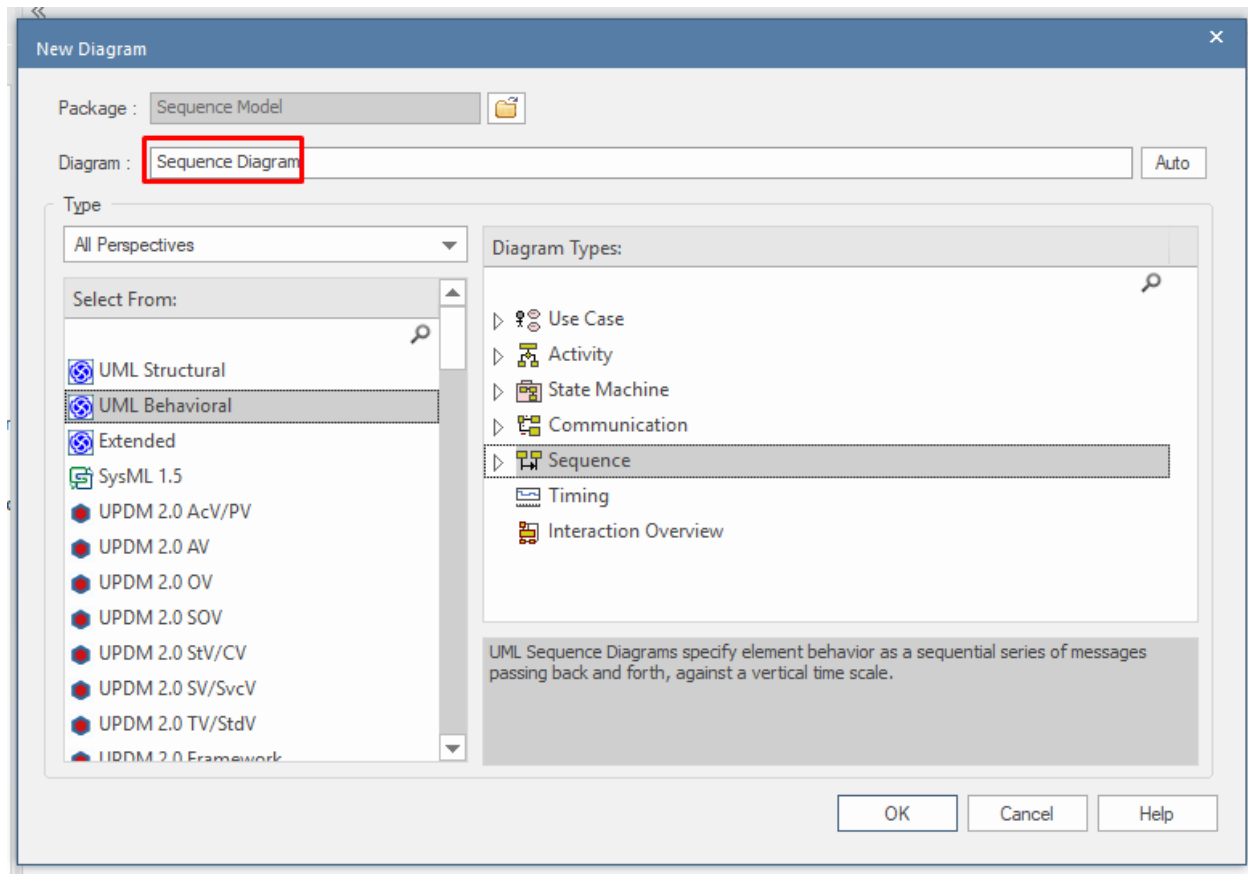
### Step 1. Add a **View** in your **Model**



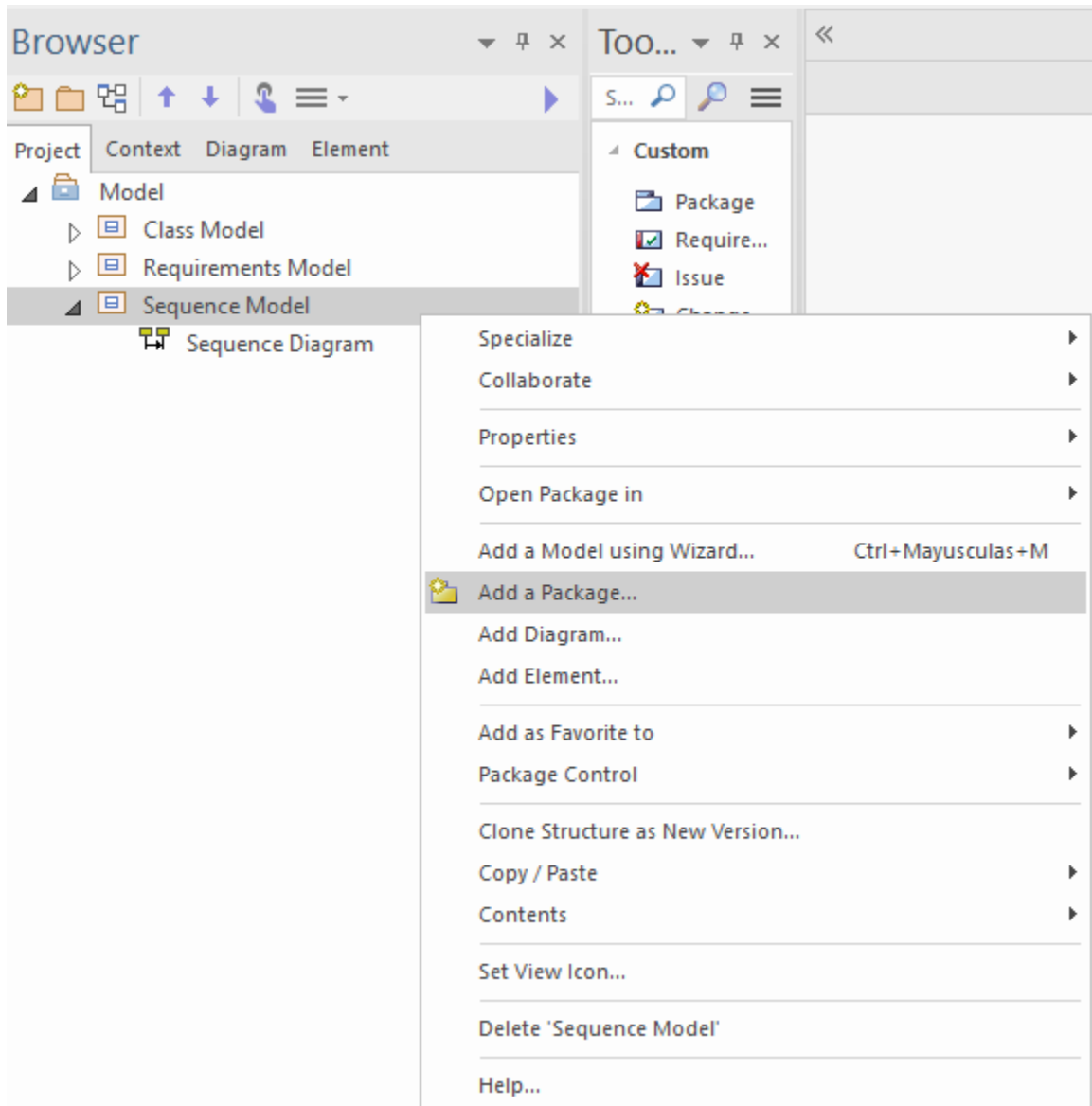
Name it **Sequence Model**, and select **Create Diagram** option



The name of the diagram is **Sequence Diagram**, and it's a **Sequence diagram** type (from **UML Behavioral** category)

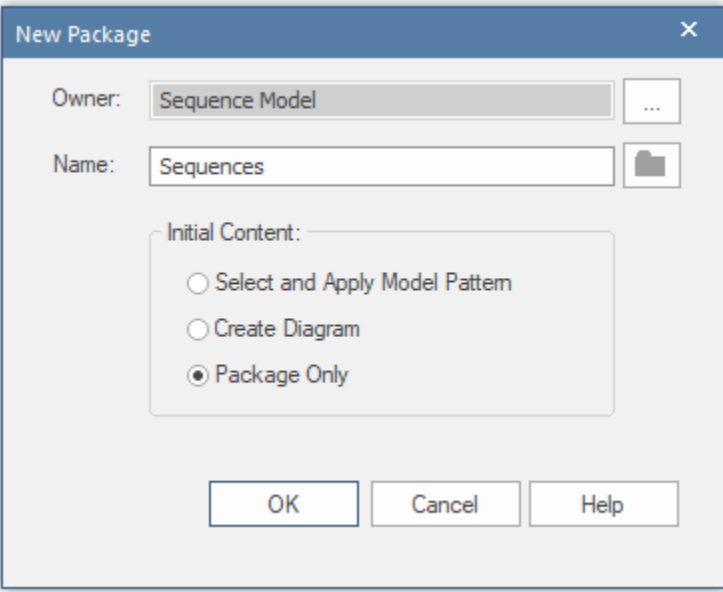


## Step 2. Add a **Package** inside the **Sequence Model**





The name is **Sequences** and selecting **Package Only**



A screenshot of a 'New Package' dialog box. The dialog has a title bar with 'New Package' and a close button. It contains two input fields: 'Owner' with the text 'Sequence Model' and a browse button (...), and 'Name' with the text 'Sequences' and a folder icon. Below these is a section titled 'Initial Content:' containing three radio button options: 'Select and Apply Model Pattern', 'Create Diagram', and 'Package Only' (which is selected). At the bottom are three buttons: 'OK', 'Cancel', and 'Help'.

New Package

Owner: Sequence Model ...

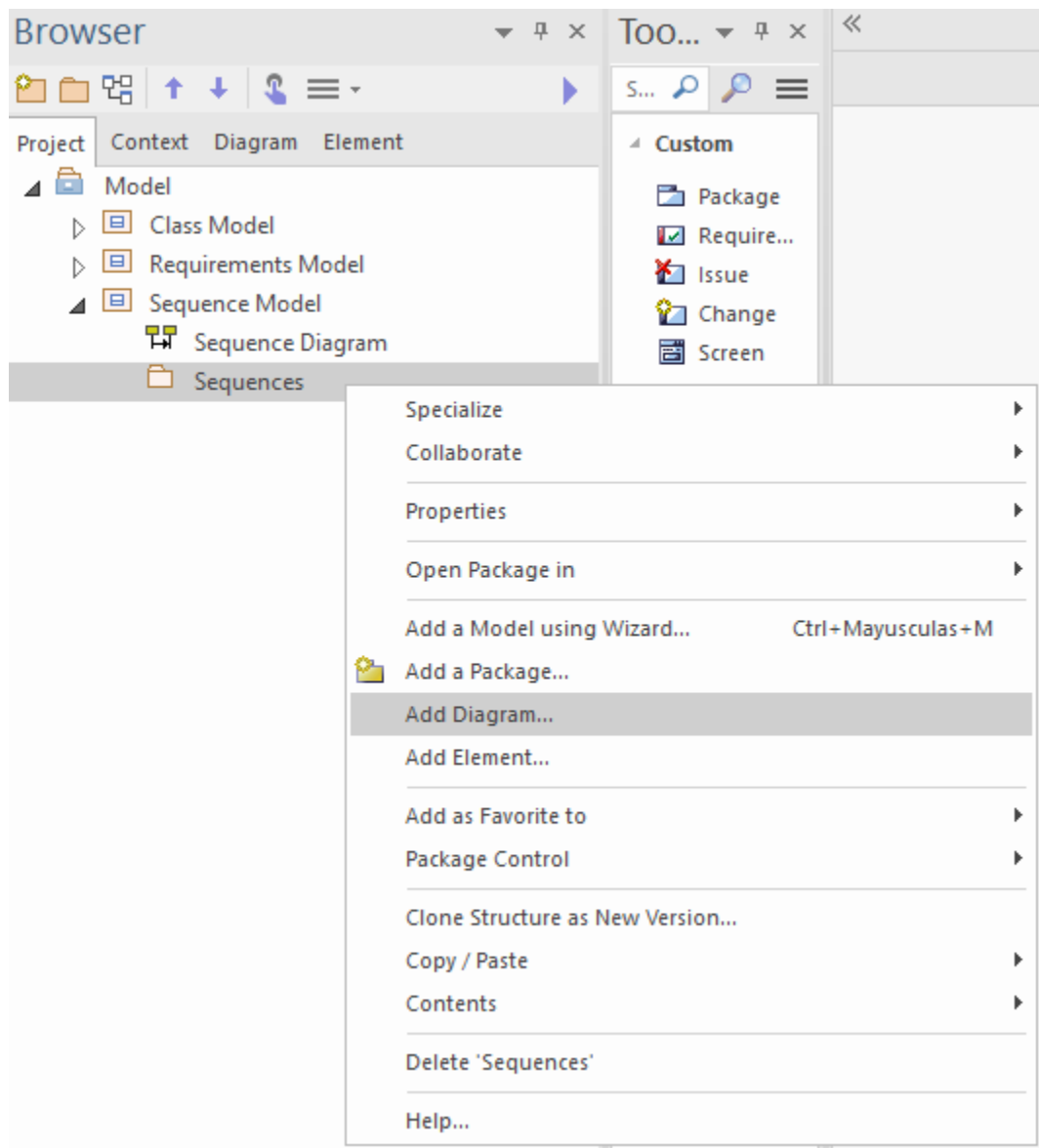
Name: Sequences

Initial Content:

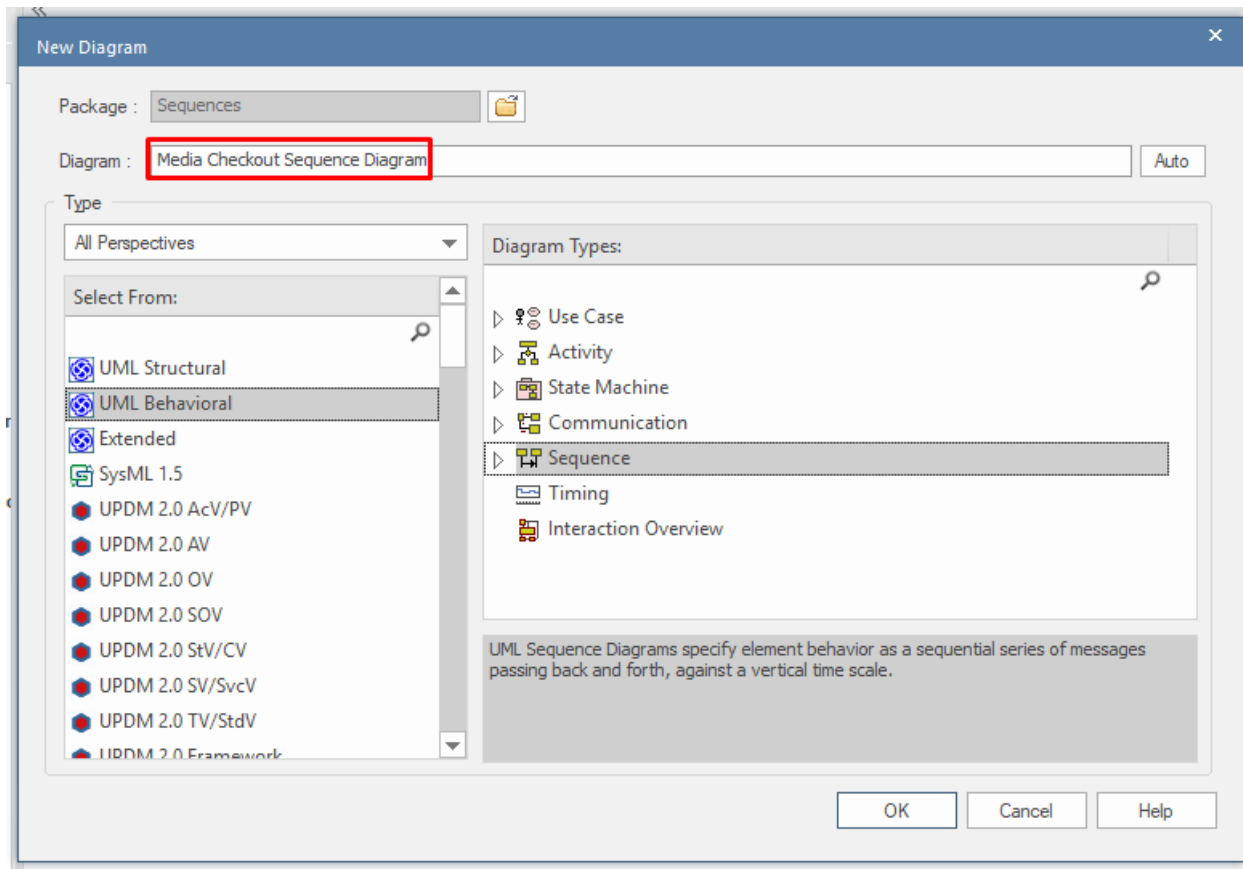
- ☐ Select and Apply Model Pattern
- ☐ Create Diagram
- ☒ Package Only

OK Cancel Help

**Step 3.** Let's add a **Sequence Diagram** for each Use Case that you identify:

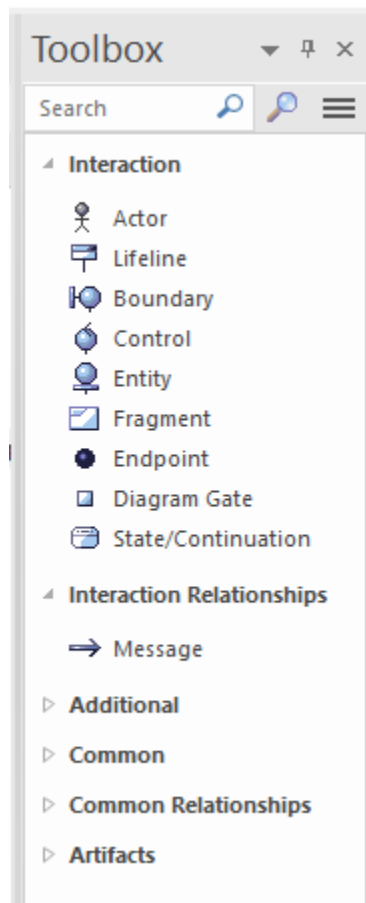


The first one is **Media Checkout Sequence Diagram**:

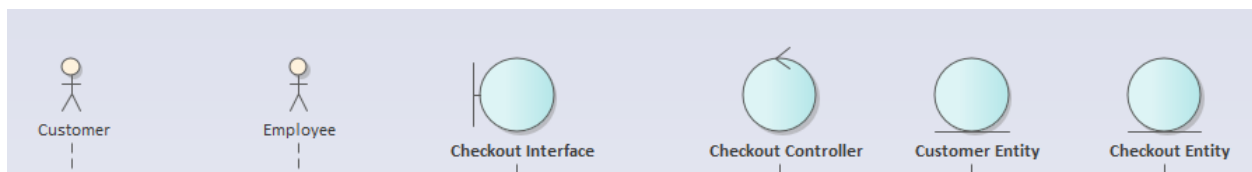


**Step 4.** Use the **toolbox** and add the following elements:

- **Customer: Actor**
- **Employee: Actor**
- **Checkout Interface: Boundary**
- **Checkout Controller: Control**
- **Customer Entity: Entity**
- **Checkout Entity: Entity**

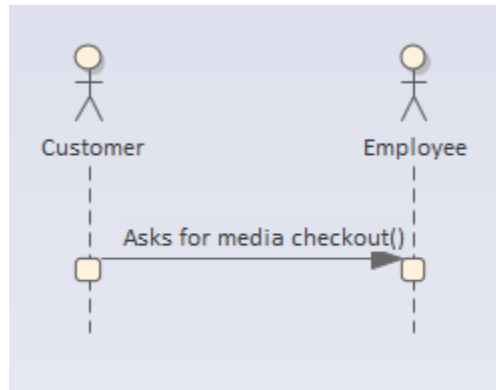


Expected output:

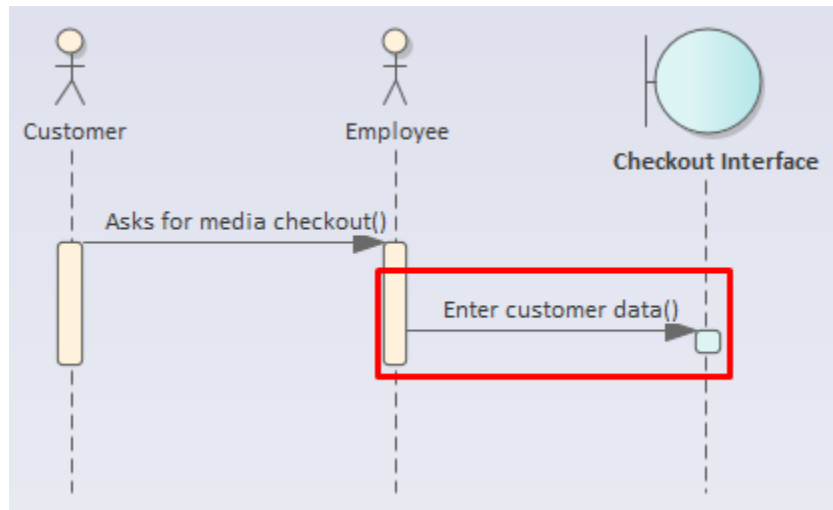


**Step 5.** Now let's start adding messages between elements in the Sequence Diagram:

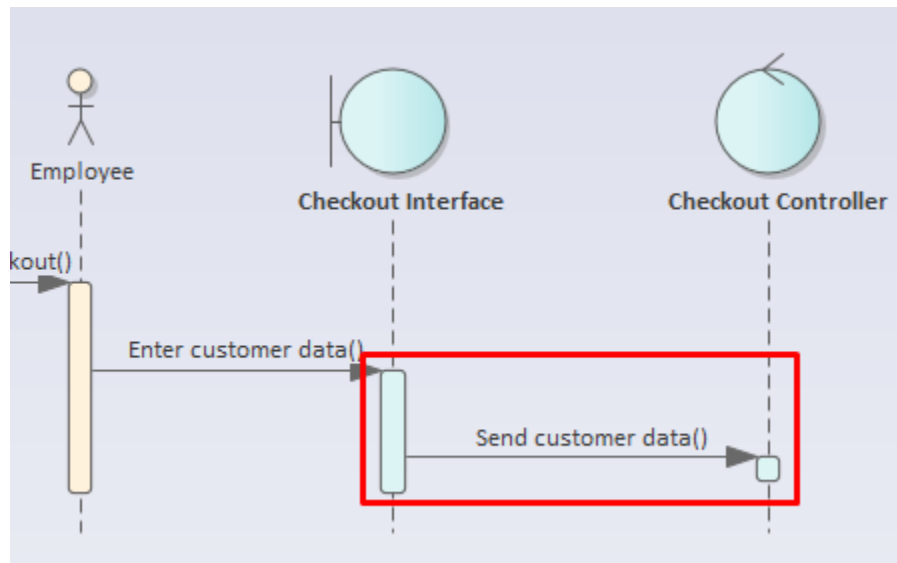
Using the toolbox, add a **message** from the **Customer** actor to the **Employee** actor. Double click it and under the message property, type **"Asks for media checkout"**



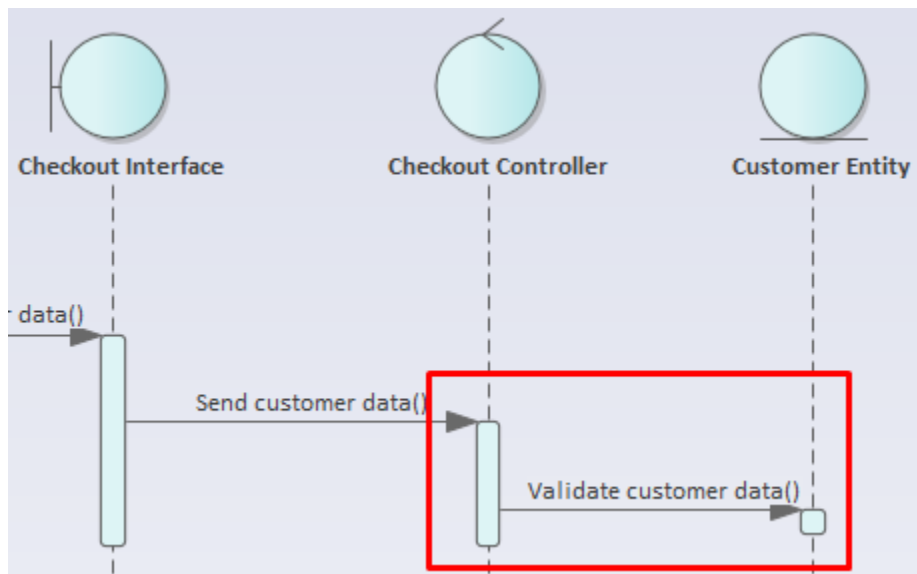
Next, add another message from the **Employee** actor to the **Checkout Interface** boundary with the **Enter customer data** value.



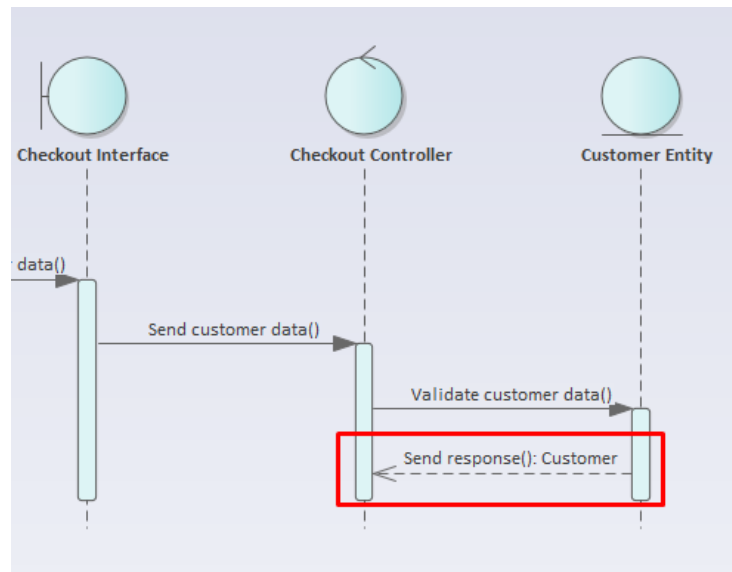
The third message is **Send Customer data** and goes from the **Checkout Interface** boundary to the **Checkout Controller** control.



Next up is the **Validate customer data** message, which goes from **Checkout Controller** control to **Customer Entity** entity.



**Step 6.** Now add another message, this time from the Customer Entity entity to the Checkout Controller control. The message value is Send response, with a Customer return value and the Is Return property activated (true):



Properties

Message

Signature

Message: **Send response** Operations

Parameters

Argument(s):

Return Value: **Customer** ☒ Show Inherited Methods

Assign To:

Stereotype:

Alias:

Sequence Expression

Condition:

Constraint:

☐ Is Iteration

Control Flow Type

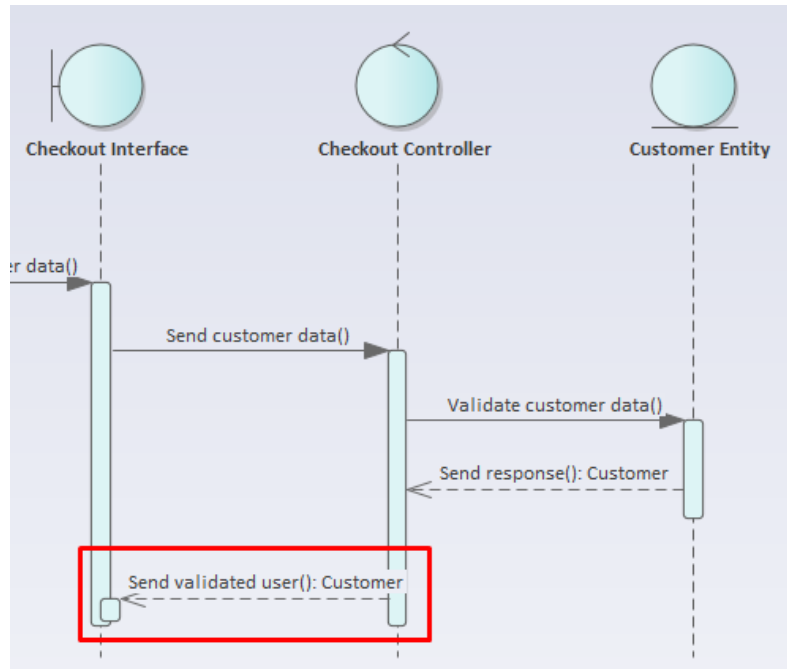
Synch: Synchronous

Kind: Call

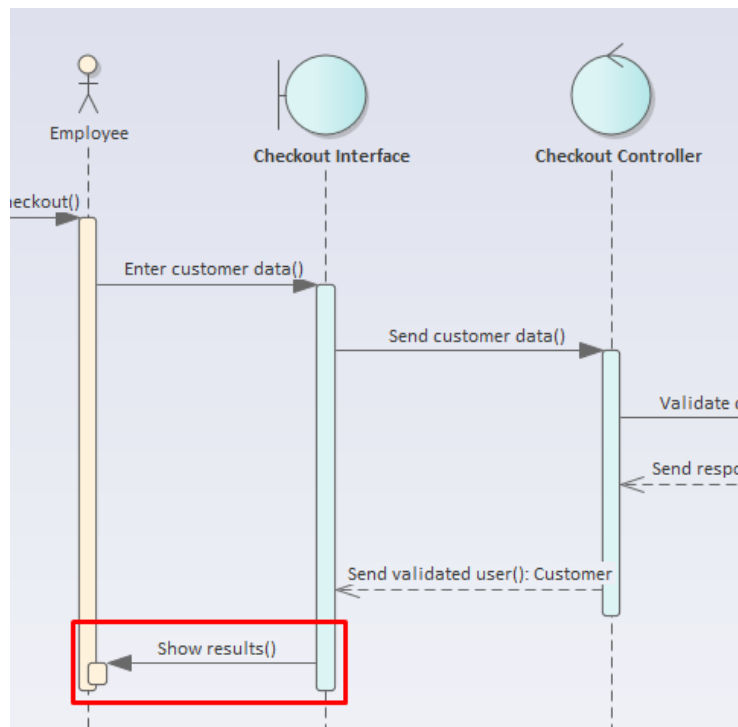
Lifecycle:

☒ Is Return

Another **return** message is **Send validated user**, again with **Customer** return value, and goes from **Checkout Controller** control to **Checkout Interface** boundary.



The **Checkout Interface** boundary sends a **Show results** message to the **Employee** actor:





**Step 7.** A message can be sent several times until certain condition is met. That's the concept of iteration. Add a new message from the **Employee** actor to **Checkout Interface** boundary with the following properties:

Properties

Message

Signature

Message: Enter media item

Parameters

Argument(s):

Return Value: void

Assign To:

Stereotype:

Alias:

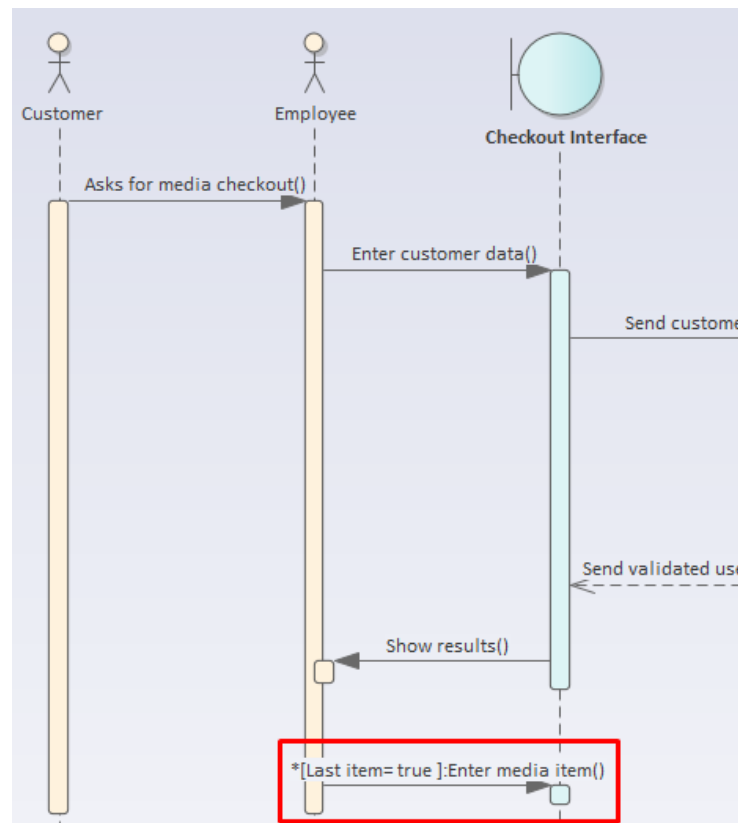
Sequence Expression

Condition: Last item = true

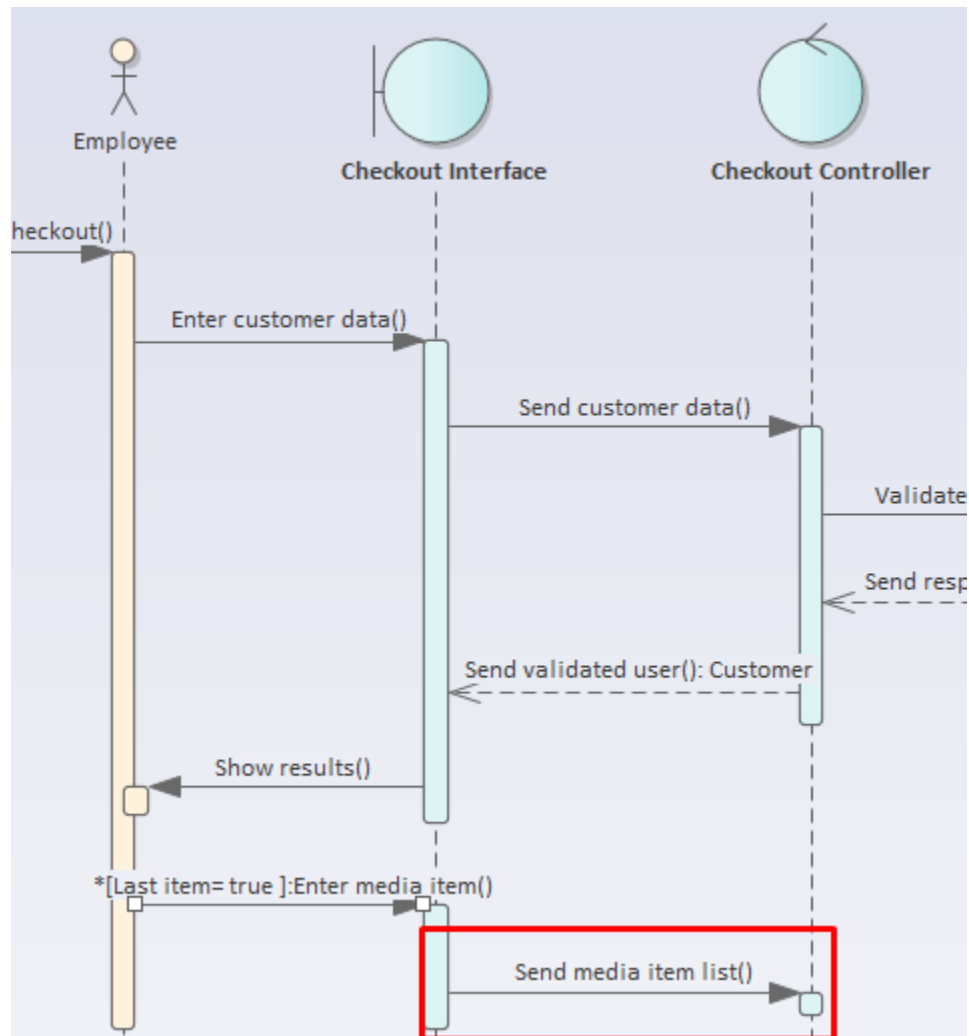
Constraint:

☒ Is Iteration

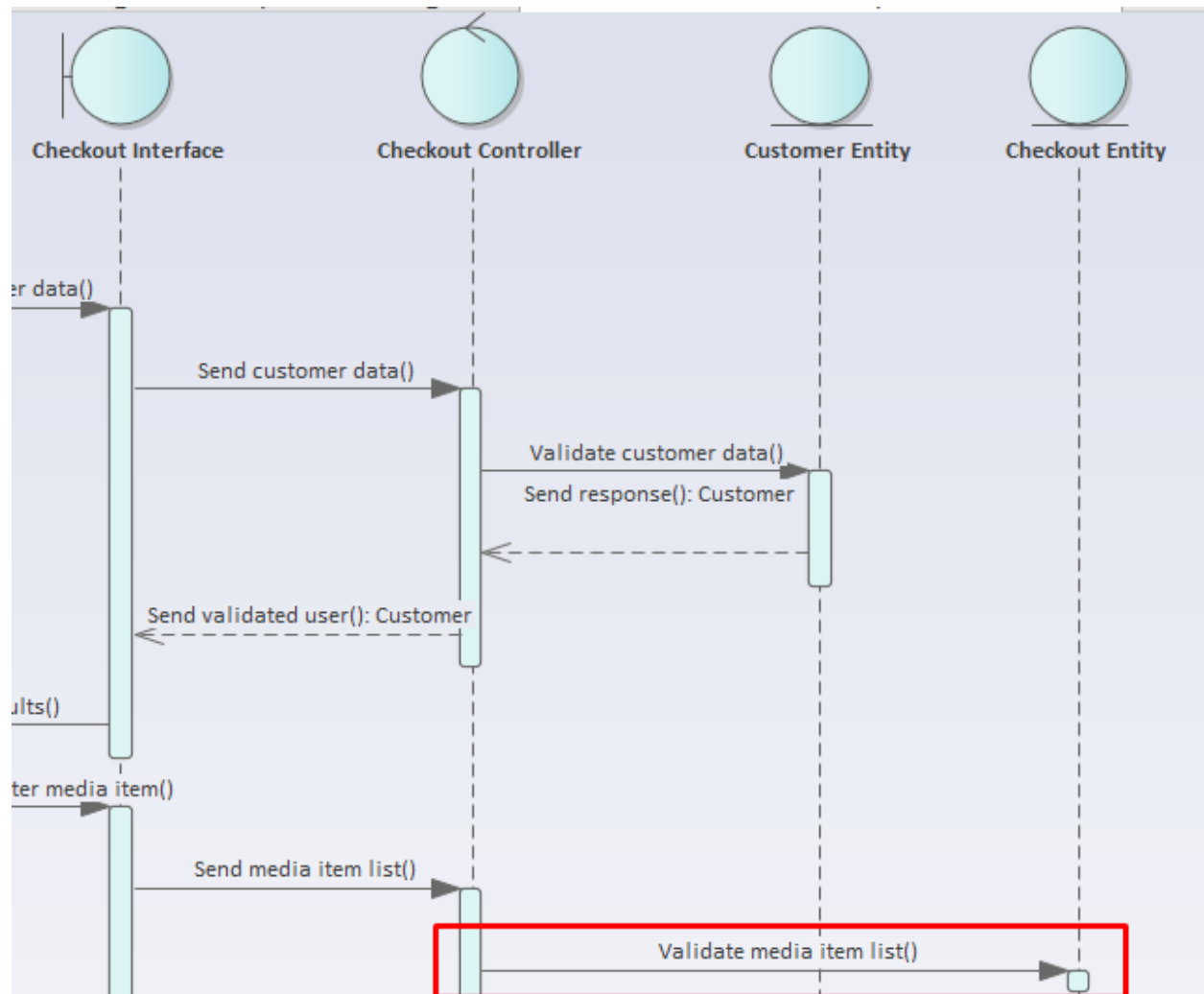
Expected result:



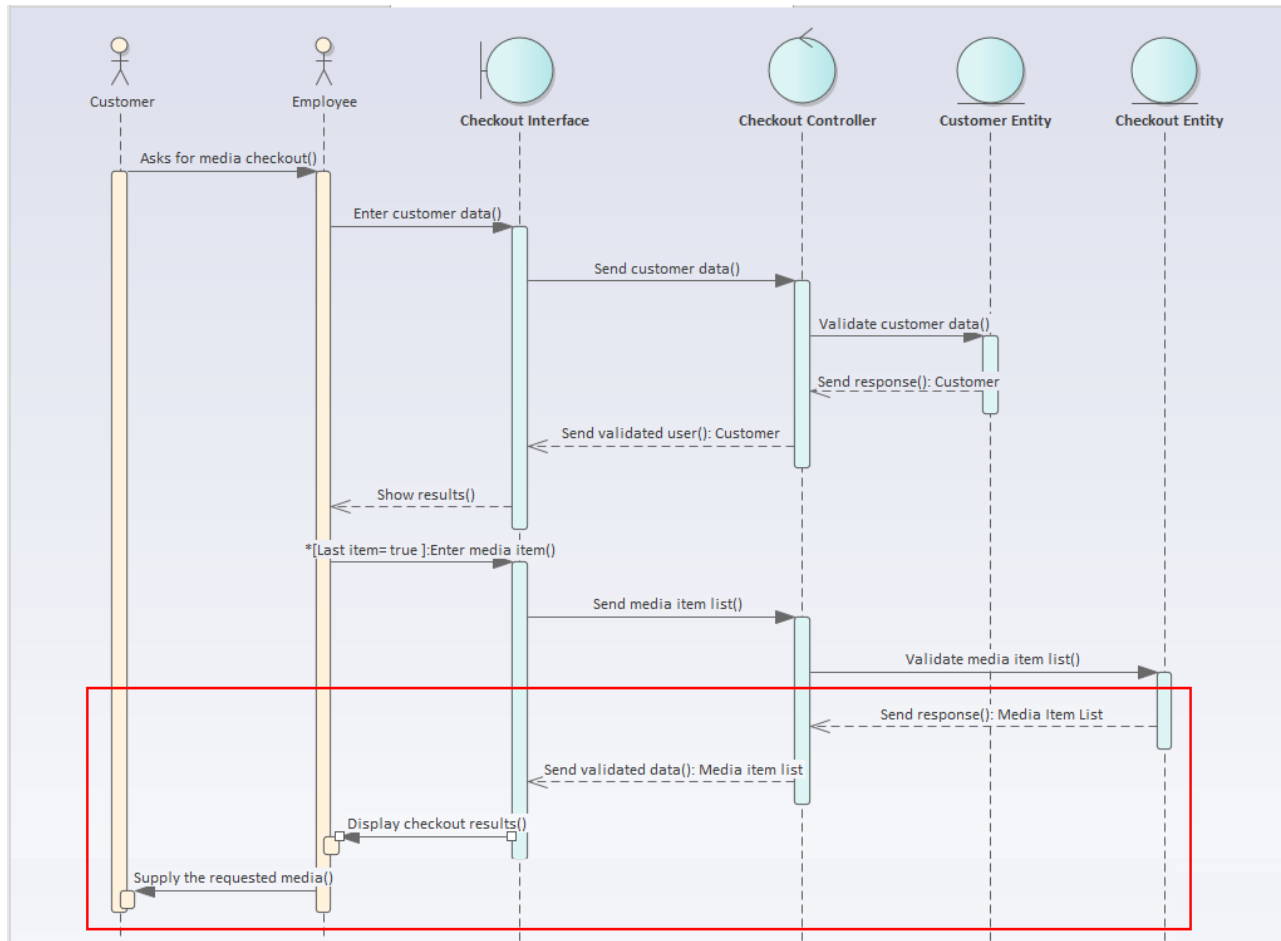
From **Checkout Interface** boundary, add a **Send media item list** message to **Checkout Controller** control:



And from the **Checkout Controller** control, add a **Validate media item list** message to the **Checkout Entity** entity.



**Step 8.** The rest of the messages deal with the validation of the media item list, in similar way that the Customer data was done. Check the diagram:



New elements added:

Message	From	To	Attributes
Send response	Checkout Entity	Checkout Controller	Return Value: Media Item List Is Return: ✓
Send validated data	Checkout Controller	Checkout Interface	Return Value: Media Item List Is Return: ✓
Display checkout results	Checkout Interface	Employee	
Supply the requested media	Employee	Customer	