Analysis model
Behavioural aspect

Behavioural model

- The behavioral models indicate how software will respond to external events
- The analysts need to develop:
  - Sequence diagram
  - Collaboration diagram
  - State diagram
  - Activity diagram
- Note: sequence diagram and collaboration diagram are also called interaction diagrams

Sequence Diagrams

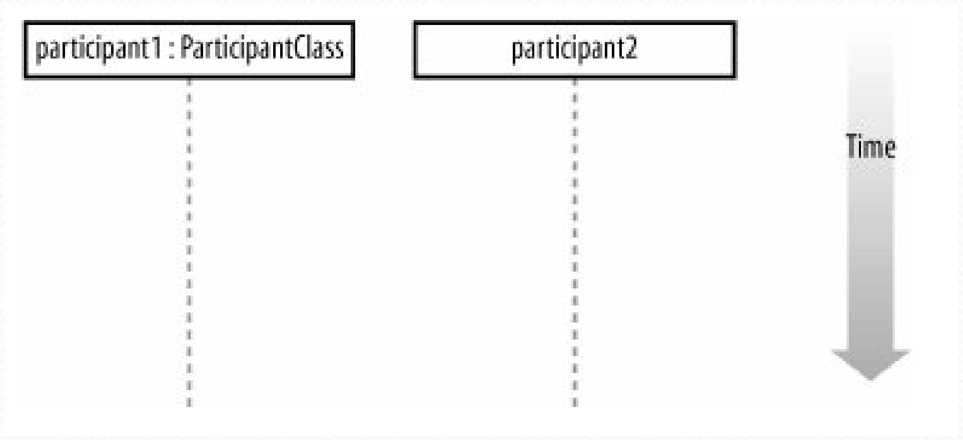
- Sequence diagrams represent the interactions among parts of the system or among a set of objects in a use case
- Can describe which interactions will be triggered when a particular use case is executed and in what order those interactions will occur

Sequence diagrams

- To draw a sequence diagram:
  - Evaluate all use-cases to fully understand the sequence of interaction within the system
  - Identify events that drive the interaction sequence and understand how these events relate to specific objects
  - Create a sequence for each (main) use-case.

Sequence Diagrams

- Time on a sequence diagram starts at the top of the page, just beneath the topmost participant heading
- Order that interactions are placed down the page on a sequence diagram indicates the order in which those interactions will take place in time

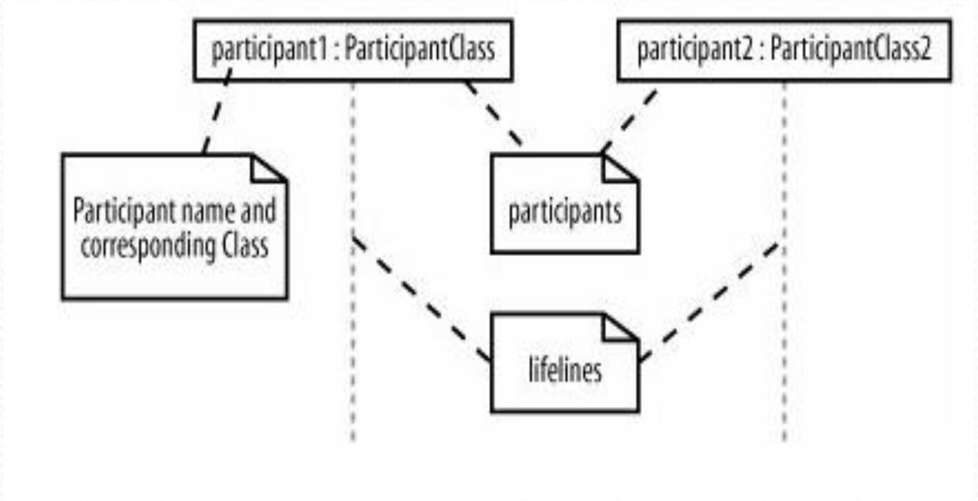


Participants in a Sequence Diagram

- A sequence diagram is made up of a collection of *participants* that interact with each other, during the sequence.
- Where a participant is placed on a sequence diagram is important.
- Time on a sequence diagram is all about ordering, not duration.
  - How much of the vertical space the interaction takes up has nothing to do with the duration of time that the interaction will take.

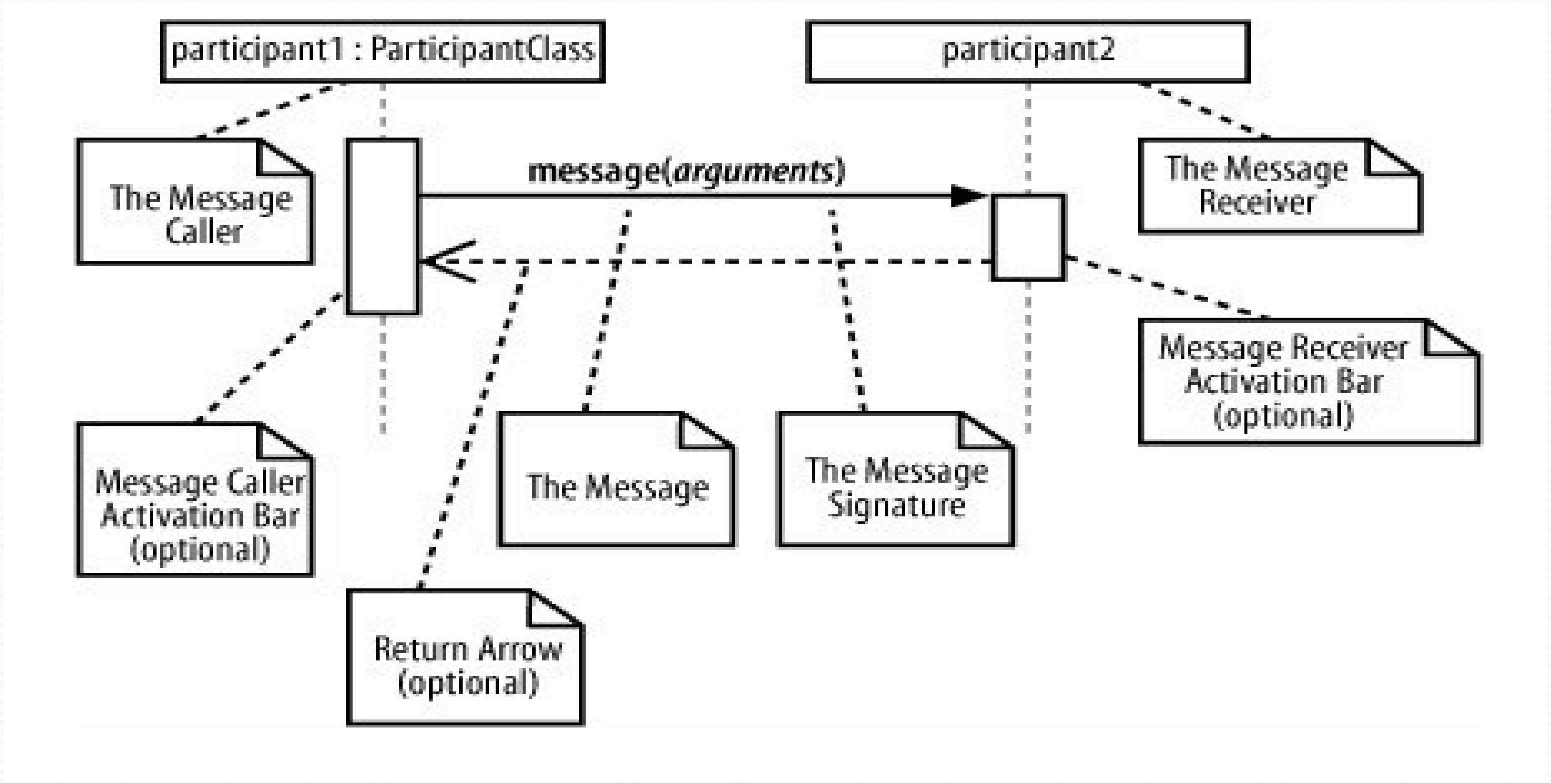
Participants in a Sequence Diagram

- Participants (objects) are arranged horizontally with no two participants overlapping each other.
- Vertical dash line: participant's life
- Participant object may be an entity object, boundary object or control object

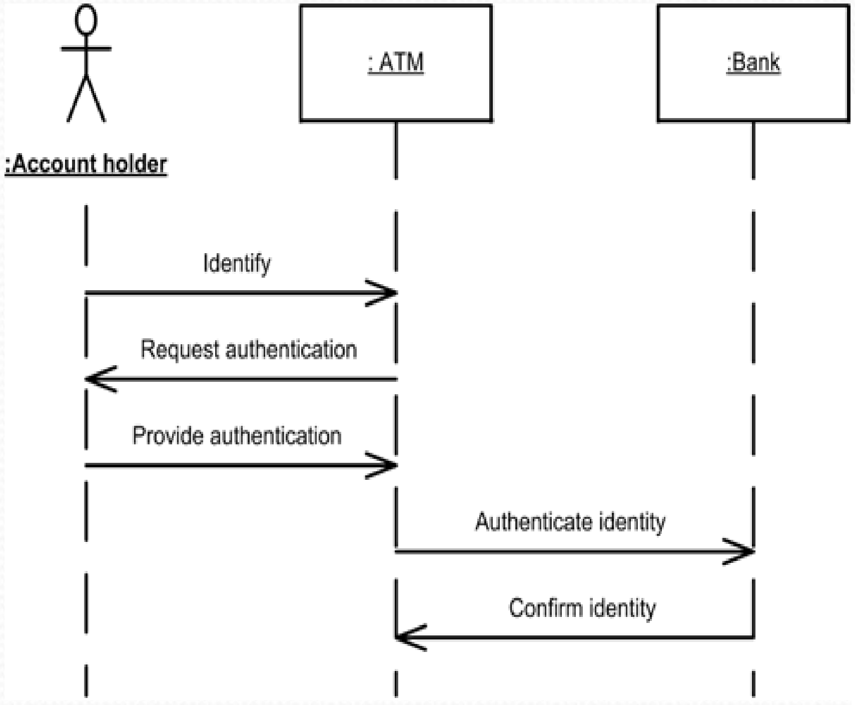

participant1 : ParticipantClass
participant2 : ParticipantClass2
Participant name and corresponding Class
participants
lifelines

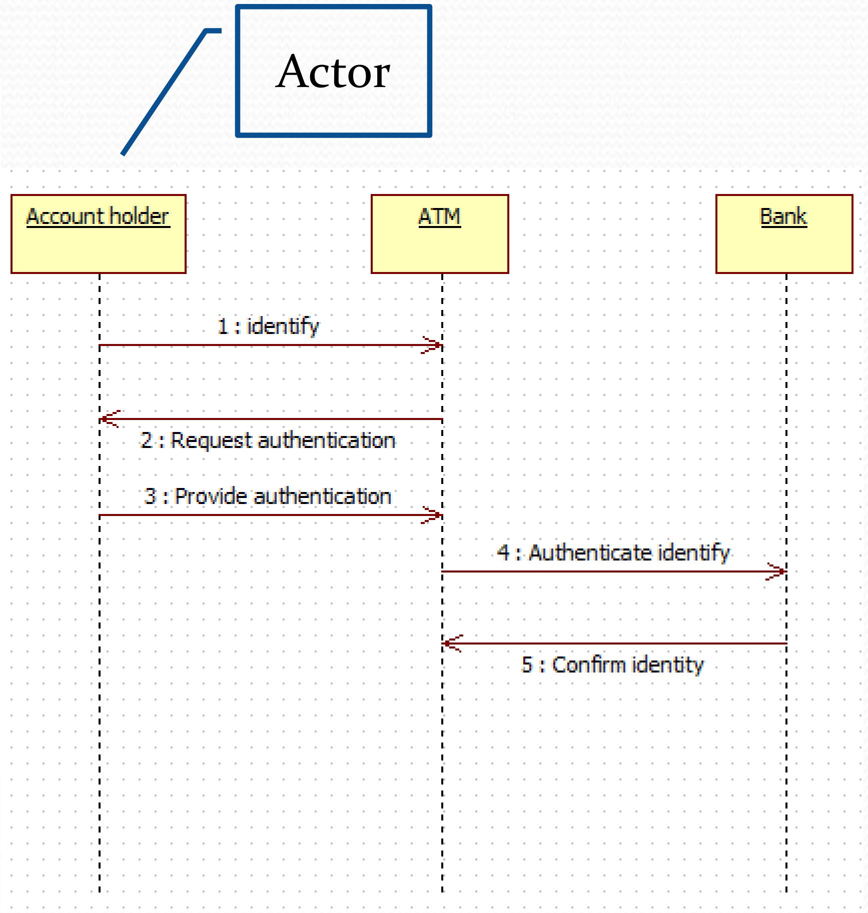
Interaction

- An interaction in a sequence diagram occurs when one participant decides to send a **message** to another participant:



Example: Sequence diagram for Authentication use case





Actor
Account holder
ATM
Bank
1 : identify
2 : Request authentication
3 : Provide authentication
4 : Authenticate identify
5 : Confirm identity

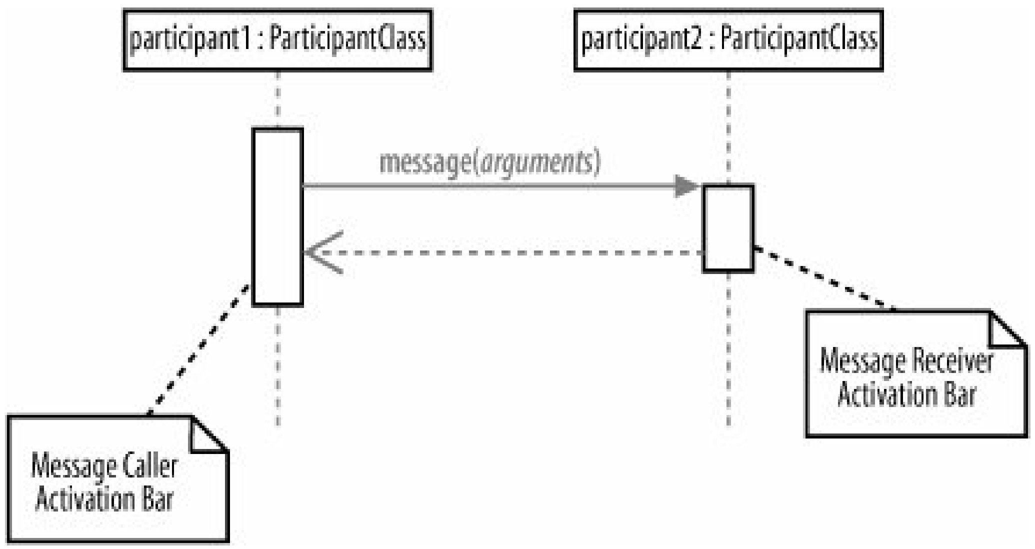
in StarUML

Message Signatures - Example

Example message signature	Description
doSomething( )	The message's name is doSomething, but no further information is known about it.
doSomething(number1 : Number, number2 : Number)	The message's name is doSomething, and it takes two arguments, number1 and number2, which are both of class Number.
doSomething( ) : ReturnClass	The message's name is doSomething; it takes no arguments and returns an object of class ReturnClass.
myVar = doSomething( ) : ReturnClass	The message's name is doSomething; it takes no arguments, and it returns an object of class ReturnClass that is assigned to the myVar attribute of the message caller.

Activation Bars

- When a message is passed to a participant it triggers the receiving participant into doing something → Active.
- Use an activation bar, as shown:

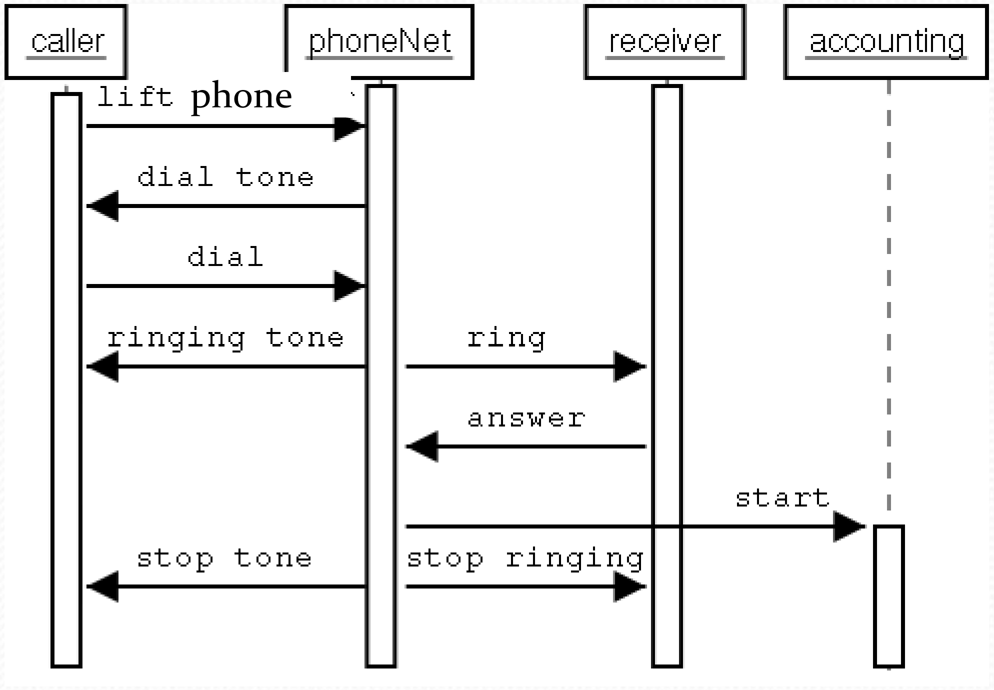

participant1 : ParticipantClass
participant2 : ParticipantClass
message(*arguments*)
Message Caller Activation Bar
Message Receiver Activation Bar

Activation Bars

- An activation bar can be shown on the sending and receiving ends of a message.
- It indicates that the sending participant is busy while it sends the message and the receiving participant is busy after the message has been received.
- Activation bars are **optional**: they can clutter up a diagram.

Example

- The process of making a phone call is described as follows:

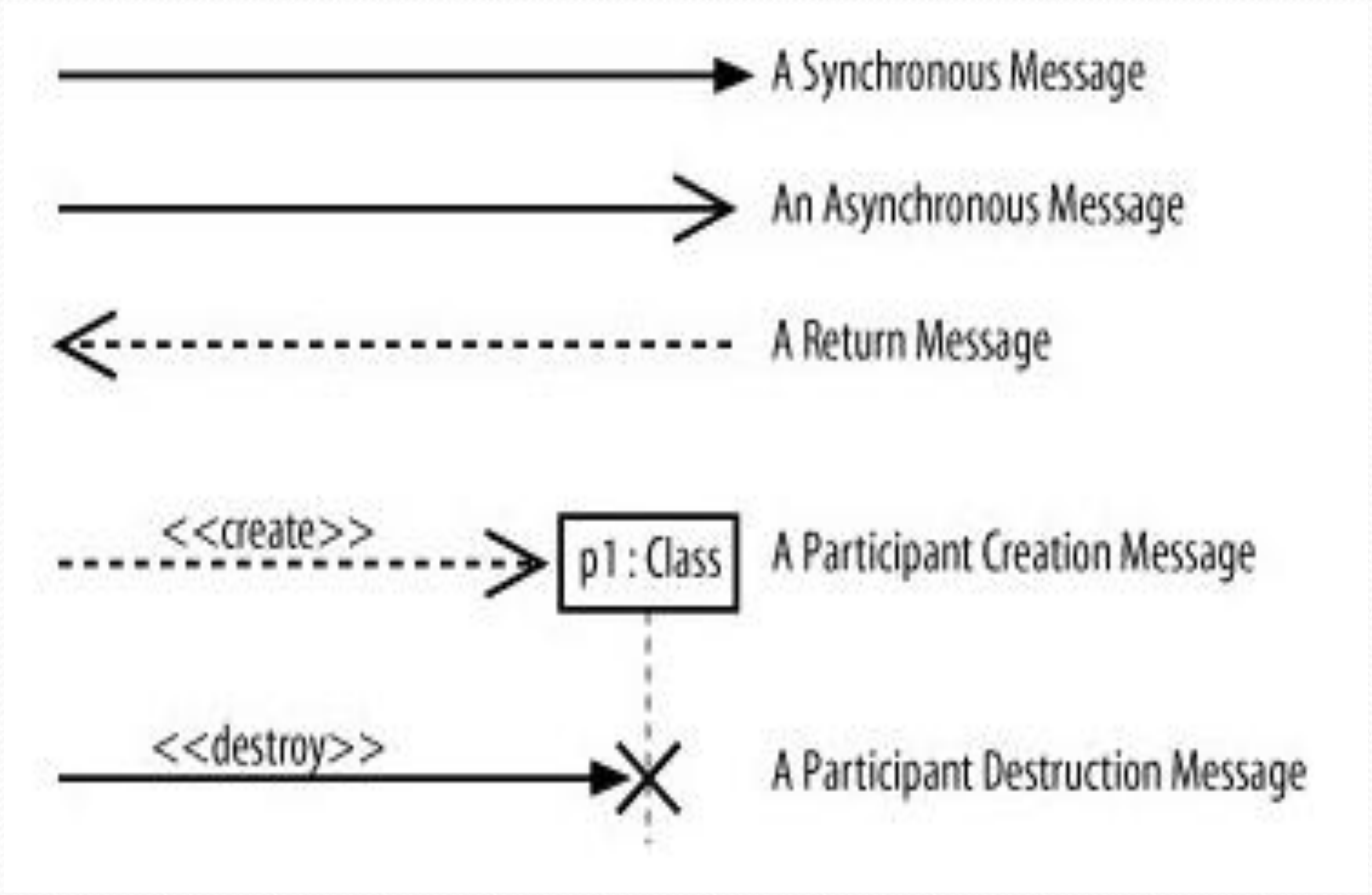


The objects involved are caller, phoneNetwork, the person who is called (receiver) and an accounting system. Various messages are passed between the objects.

Message Arrows

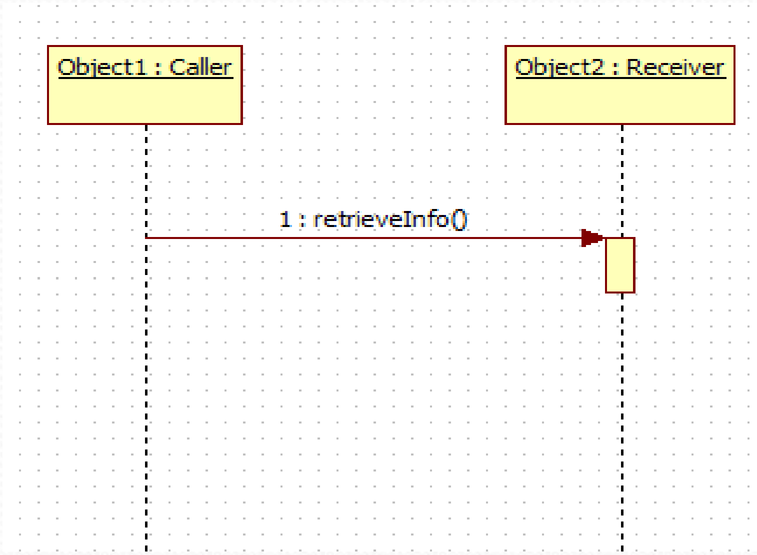
- The type of arrowhead - understand what type of message is being passed
- For example, the Message Caller may want to wait for a message to return before carrying on with its work - a synchronous message
- Or it may wish to just send the message to the Message Receiver without waiting for any return as a form of "fire and forget" message - an asynchronous message.

Message Arrows



Synchronous Messages

- A synchronous message is invoked when the Message Caller waits for the Message Receiver to return from the message invocation.



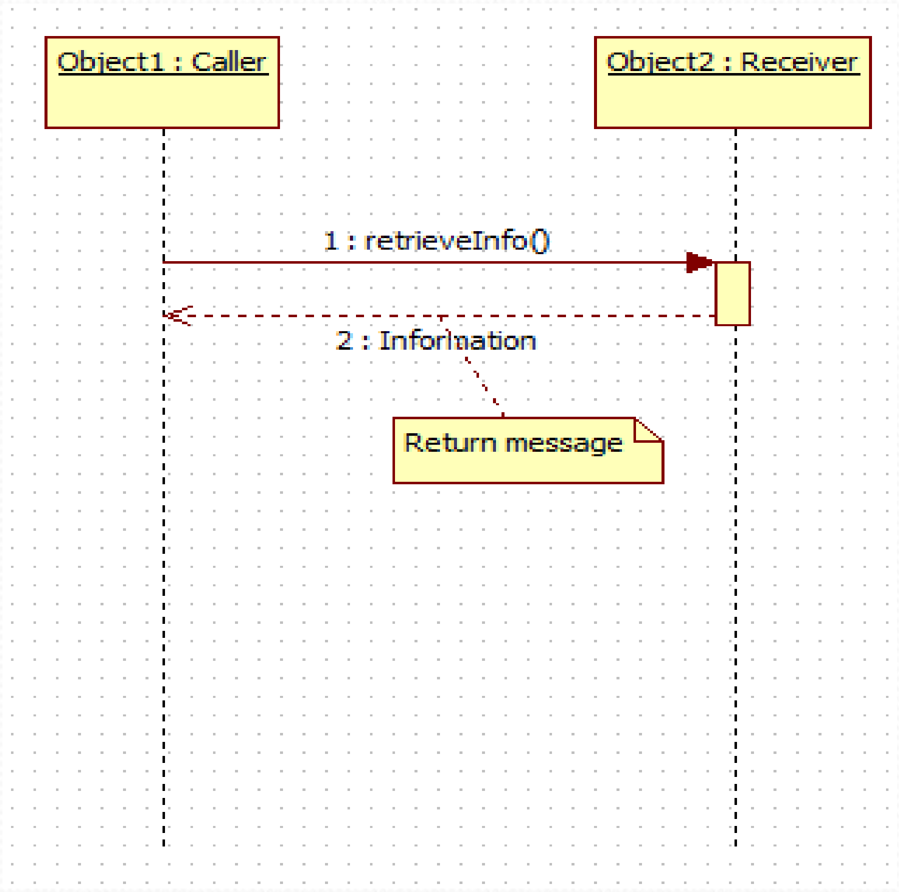
In StarUML, Message type is Call. The caller calls a method from Receiver and wait for the return message

The Return Message

- Optional piece of notation that you can use at the end of an activation bar to show that the control flow of the activation returns to the participant that passed the original message
- In code, a return arrow is similar to reaching the end of a method or explicitly calling a return statement

You don't have to clutter up your sequence diagrams with a return arrow for every activation bar since there is an implied return arrow on any activation bars that are invoked using a synchronous message

Return message - Example



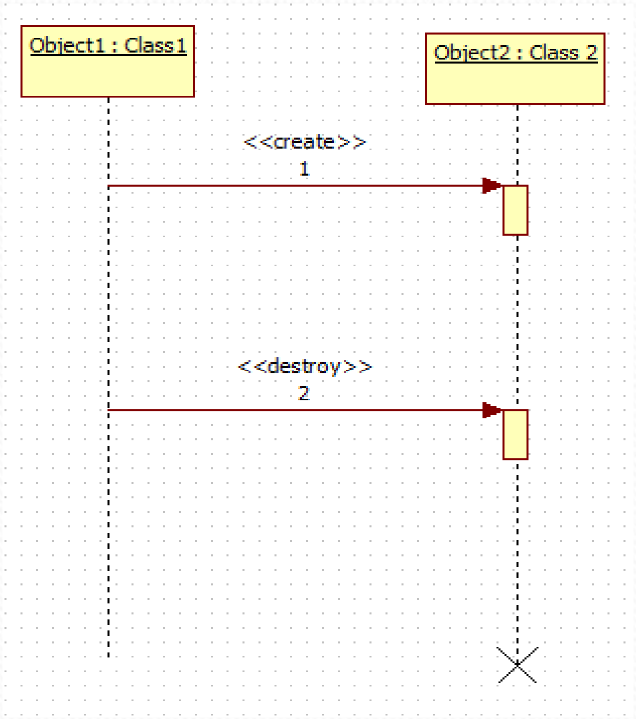
Asynchronous Messages

- The caller may send a message to receiver and continue their work without waiting the return message from the receiver.
- E.g.: Call to print a document and do other work
- In StarUML, message type is Send

Participant Creation/Destruction Messages

- Participants do not necessarily live for the entire duration of a sequence diagram's interaction

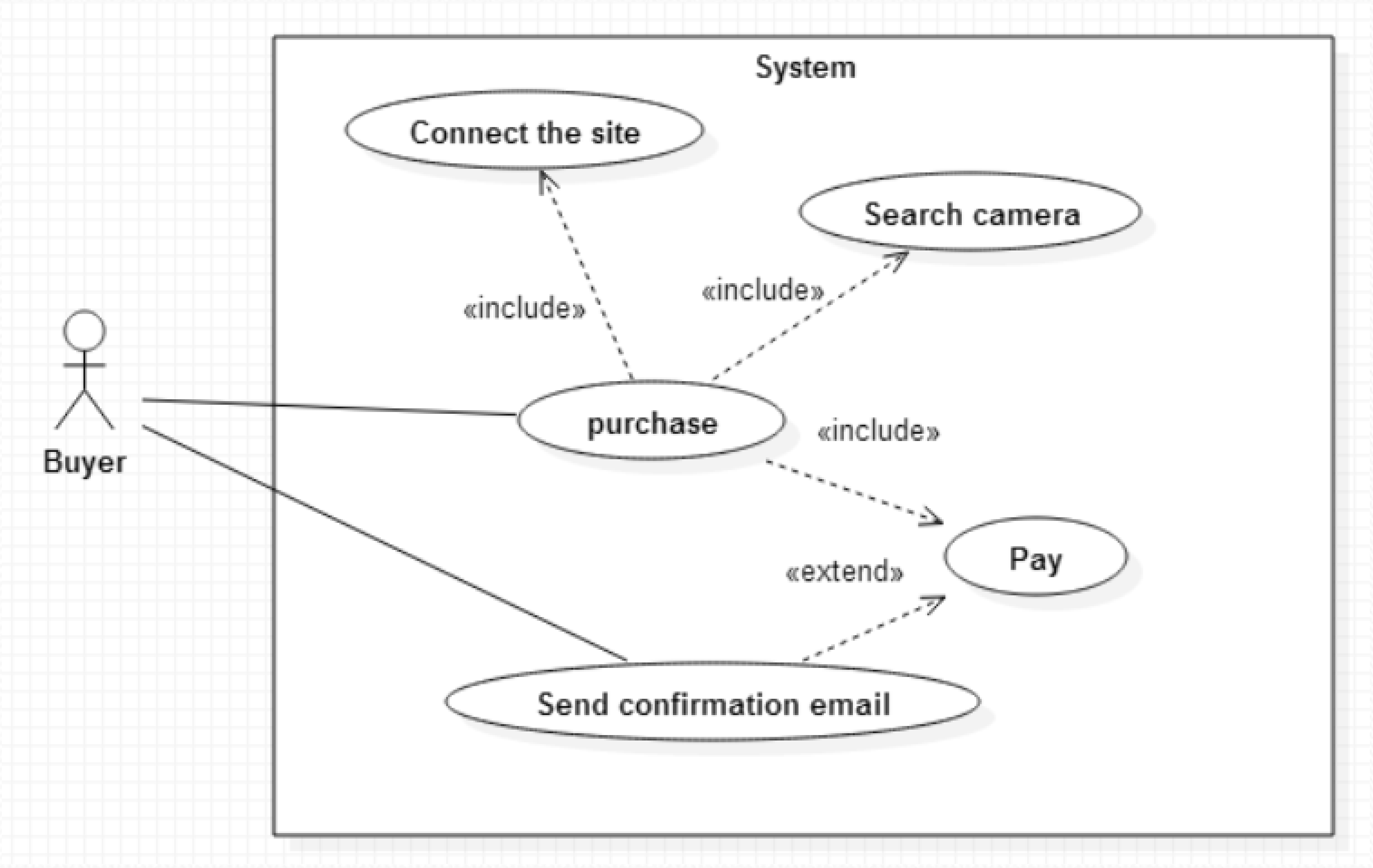
- Participants can be created and destroyed according to the messages that are being passed



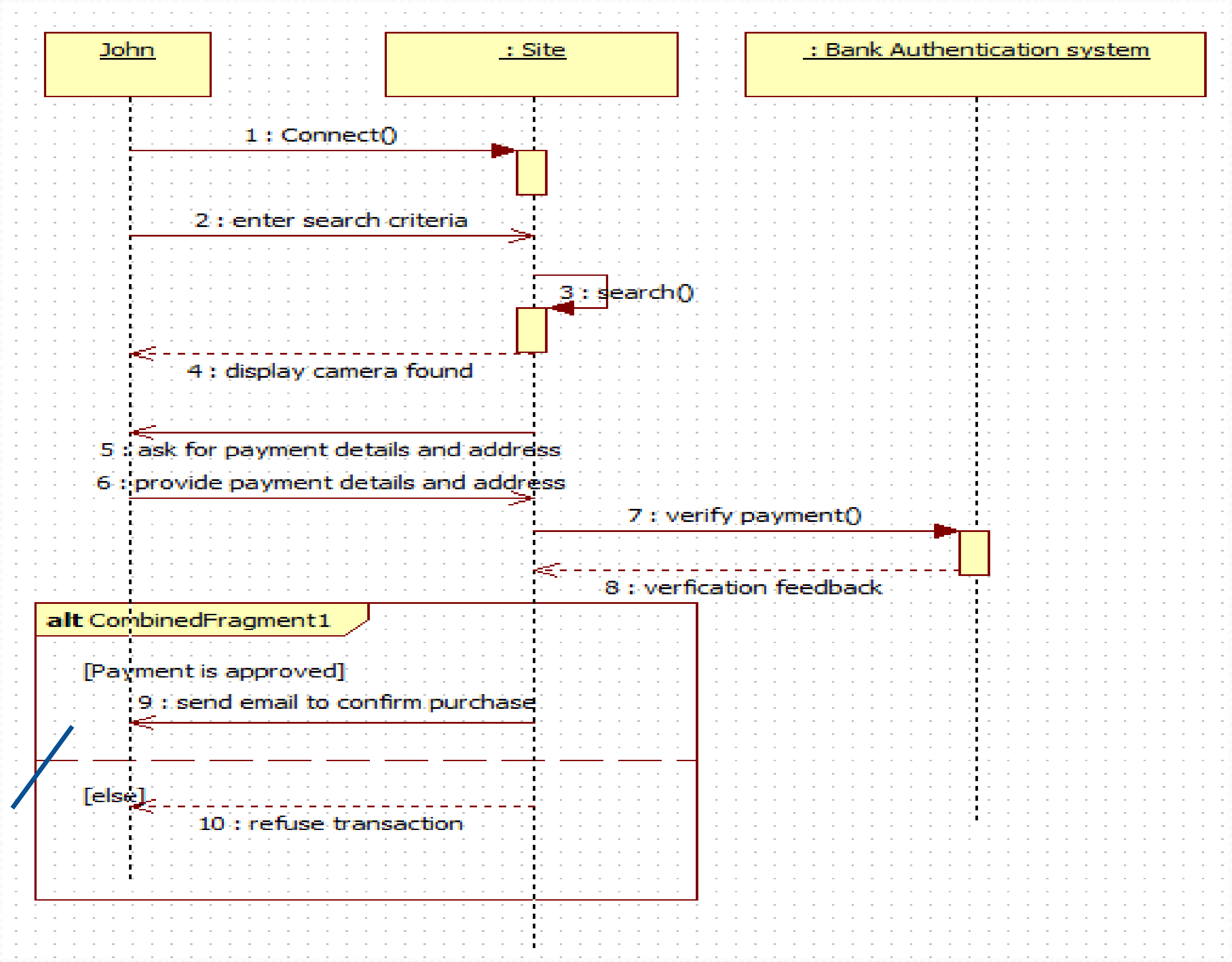
Online purchase example

- Create a sequence diagram to model John buying a camera on the Internet.
- John will connect to some site and search for a camera. He will choose a camera. The system will provide the appropriate searches to make sure that the supplier has the product.
- John will then be prompted to pay for the camera and provide payment details as well as a delivery address. Once the payment has been cleared by the bank authentication system an email is sent to John and the purchase is complete, otherwise the transaction is refused.

Use case diagram – Online purchase

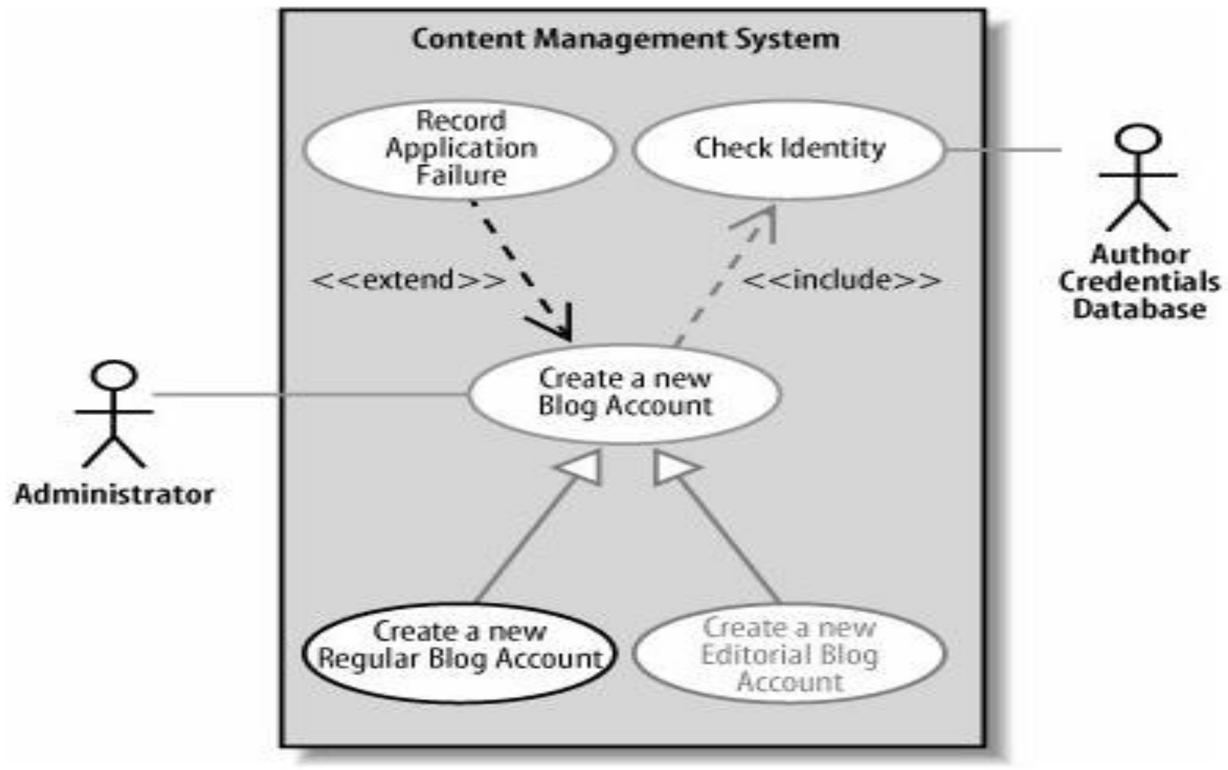


Sequence diagram, online purchase



Alternative interactions, if-else condition

From Use Cases diagram to Sequence diagram


Content Management System
Record Application Failure
Check Identity
<<extend>>
<<include>>
Author Credentials Database
Create a new Blog Account
Administrator
Create a new Regular Blog Account
Create a new Editorial Blog Account

From Use Cases diagram to Sequence diagram

- The Create_a_new_Regular_Blog_Account use case is a special case of the Create_a_new_Blog_Account use case.

- It also includes all of the steps provided by the Check_Identity use case and may optionally execute the steps provided by the Record_Application_Failure use case, if the application for a new account is denied.

Flow of events

- The steps that occur in the Create a new Regular Blog Account use case according to its detailed description are

Basic flows:

1. The Administrator asks the system to create a new blog account
2. The Administrator selects the regular blog account type
3. The Administrator enters the author's details
4. The author's details are checked using the Author Credentials Database
5. The new regular blog account is created
6. A summary of the new blog account's details are emailed to the author

Flow of events

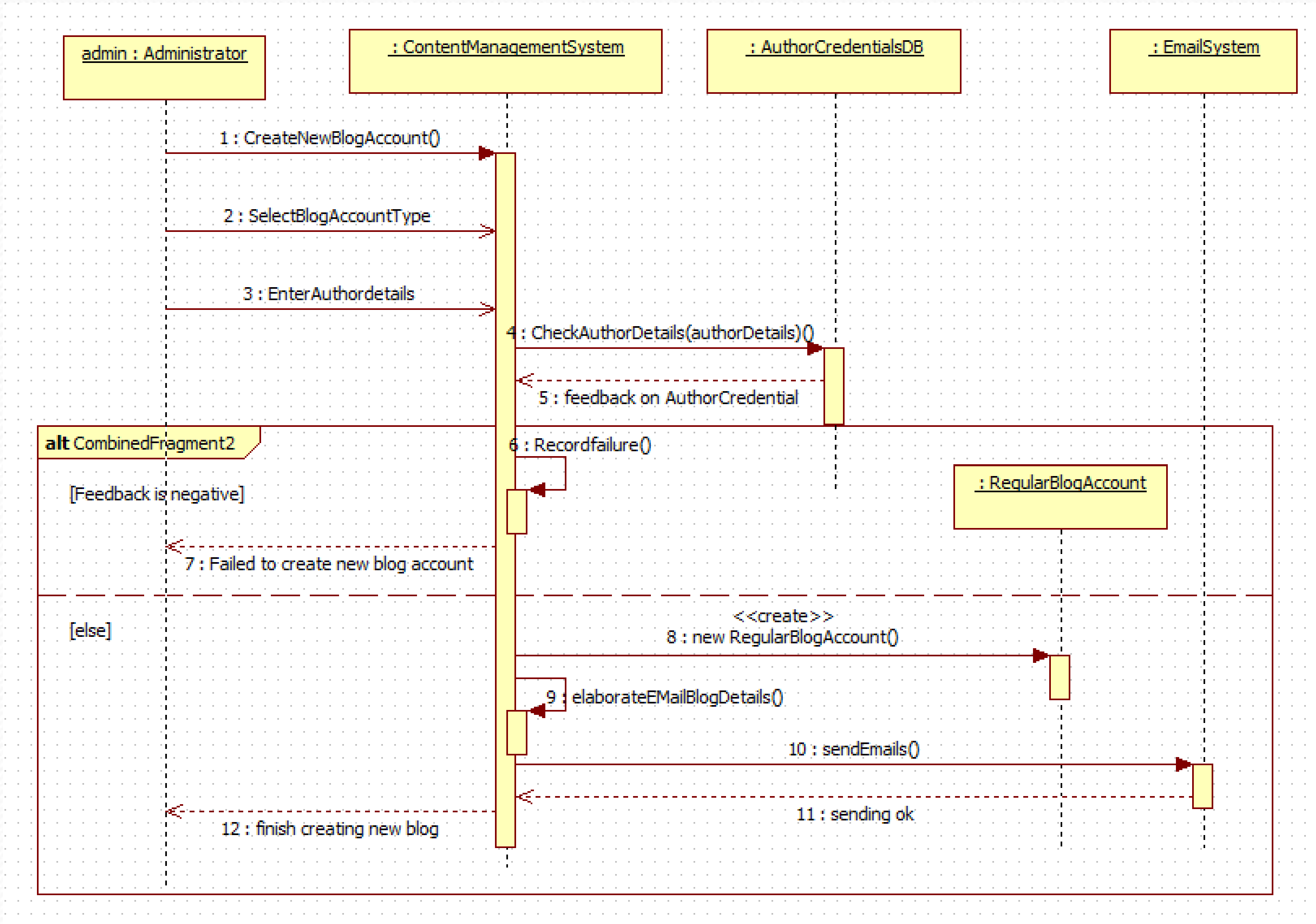
- Alternative flows:

4.1 Author details are not approved

4.2 The system record the application failure

4.3 Stop the use case

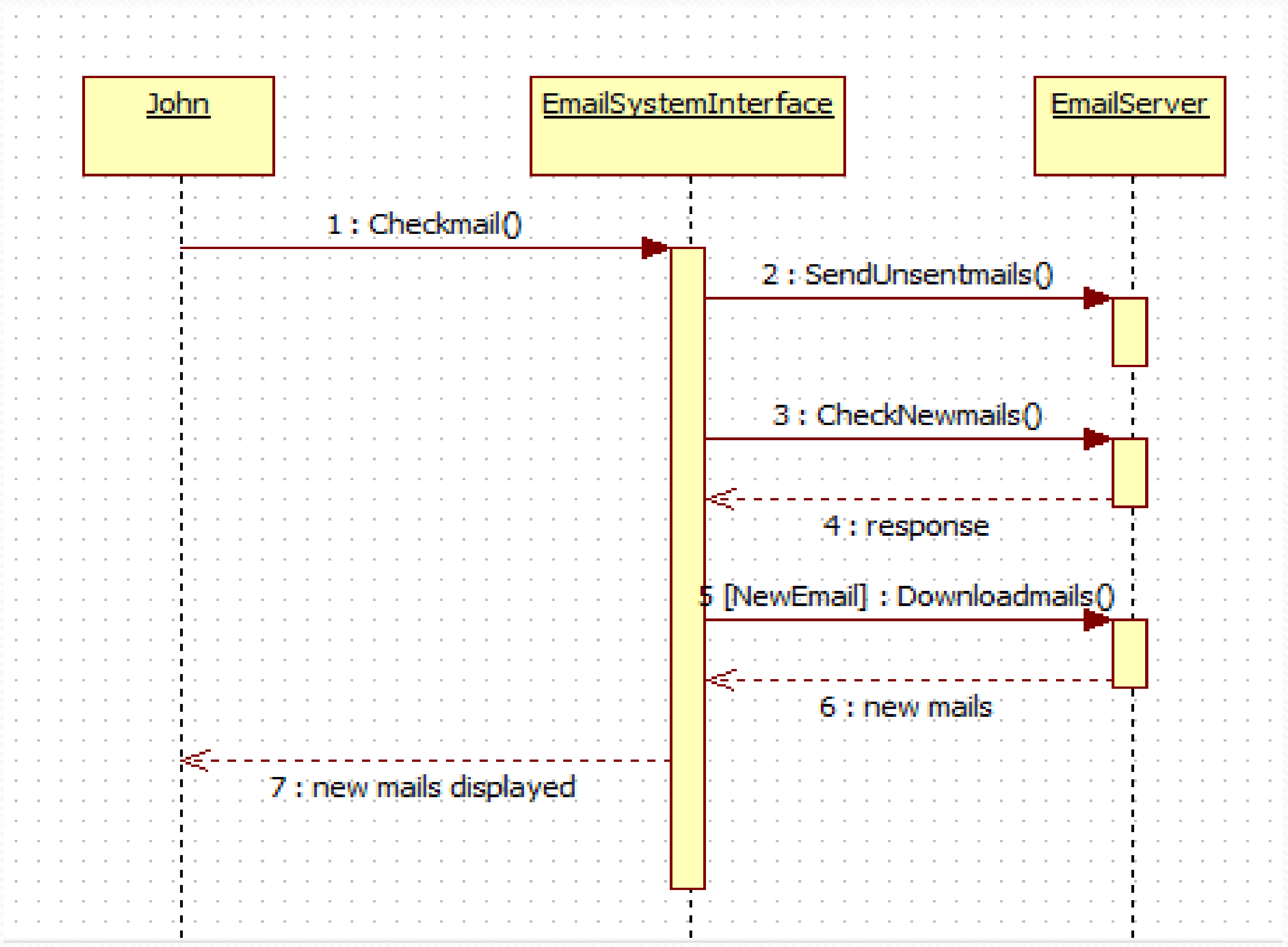
Sequence diagram


admin : Administrator
: ContentManagementSystem
: AuthorCredentialsDB
: EmailSystem
1 : CreateNewBlogAccount()
2 : SelectBlogAccountType
3 : EnterAuthordetails
4 : CheckAuthorDetails(authorDetails)()
5 : feedback on AuthorCredential
alt CombinedFragment2
6 : Recordfailure()
[Feedback is negative]
: RegularBlogAccount
7 : Failed to create new blog account
[else]
<<create>>
8 : new RegularBlogAccount()
9 : elaborateEMailBlogDetails()
10 : sendEmails()
11 : sending ok
12 : finish creating new blog

Sequence Diagram: Email Check

- Draw a sequence diagram with the following specification:
  - John is a computer user who wants to check his email
  - When he clicks on his computer to check for emails the server will send any unsent ones
  - It then checks for new emails and the server returns a corresponding response
  - If there are new emails these will be downloaded

Sequence Diagram: Email Check



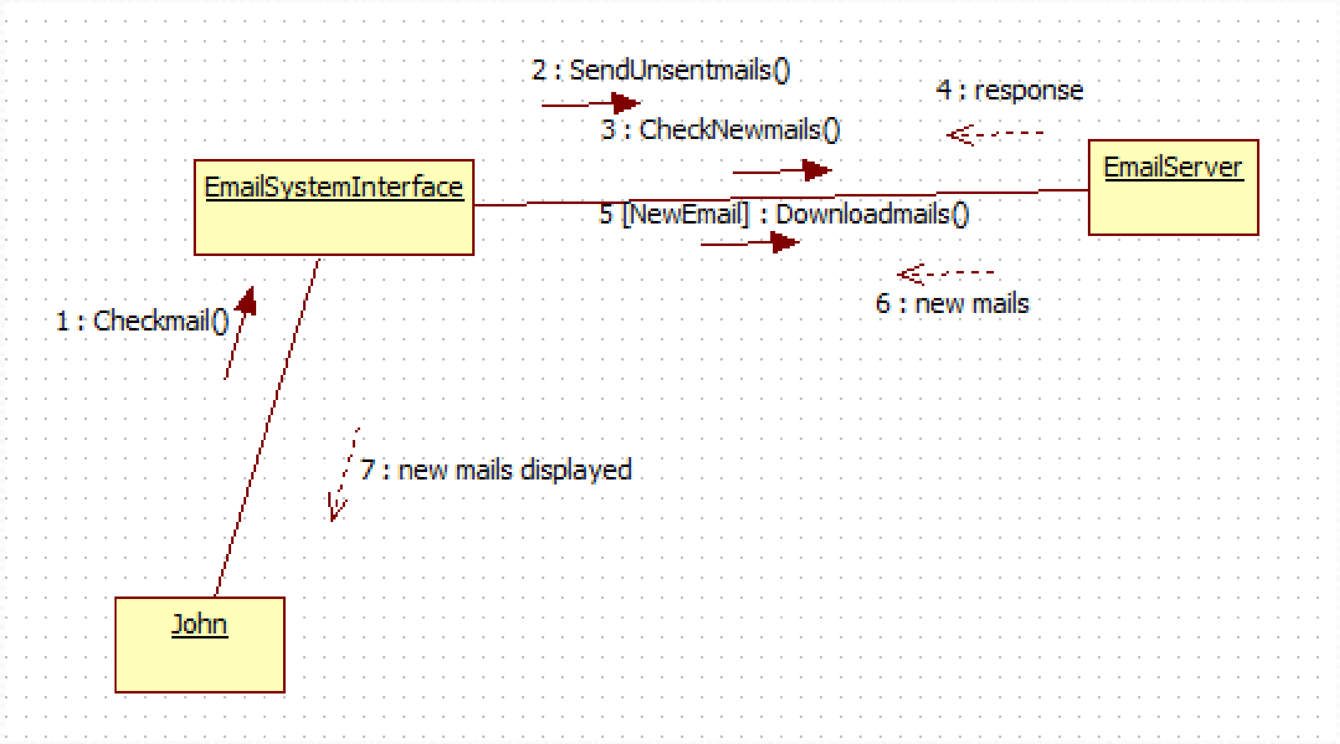
Collaboration diagrams

- Or communication diagrams
- An alternative representation of a sequence diagram
- A *collaboration diagram* captures the same information as sequence diagram, although less conveniently, but it additionally reveals the architectural relationships that exist between the objects.

Collaboration Diagrams

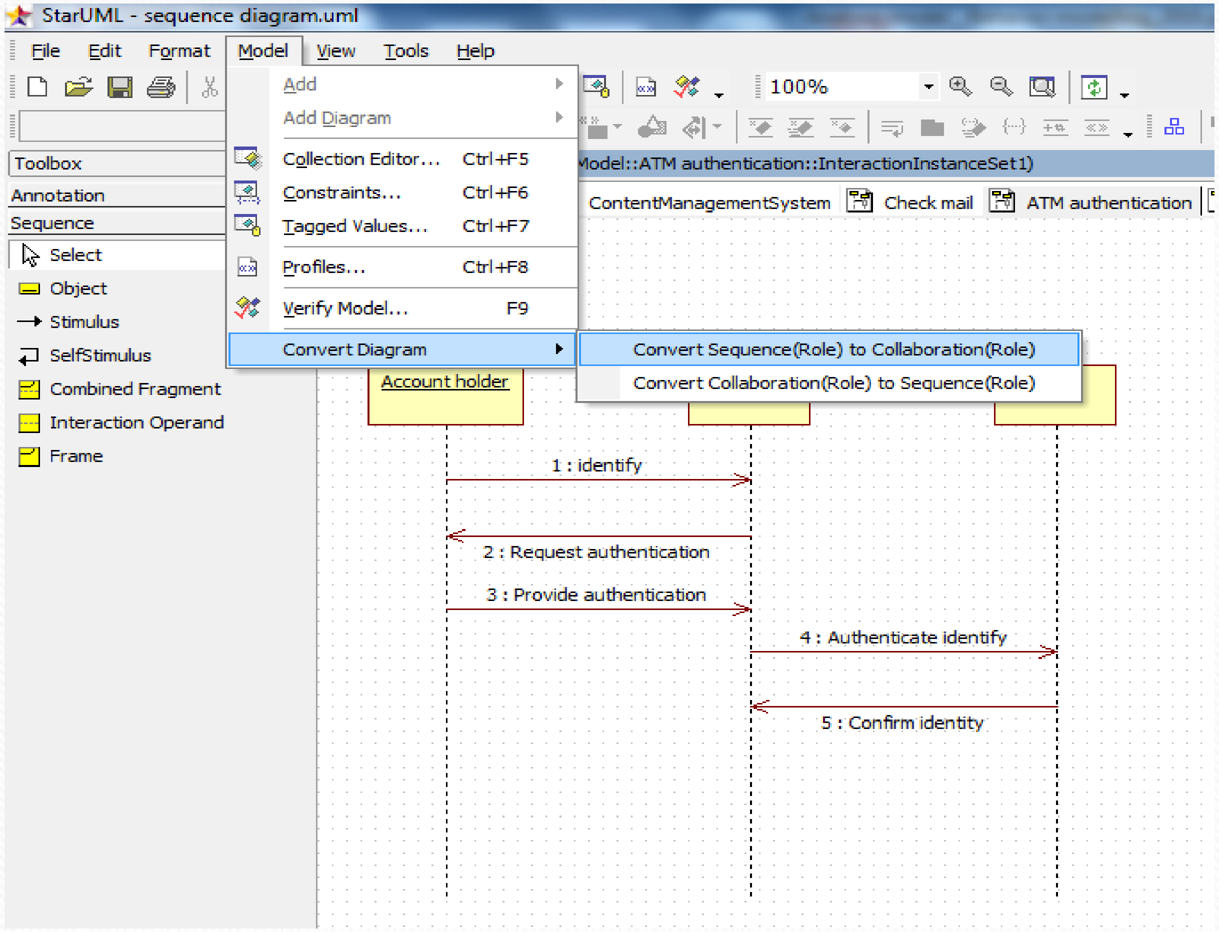
- Links between the objects are shown like associations in class models
- Since a link is an instance of an association, there should be an association in the class model between the classes of any two linked objects (of entity classes)
- If the collaboration is describing the realisation of a use case, the actors in the collaboration will correspond to the actors which are connected to the use case in the use case diagram

Email check Example

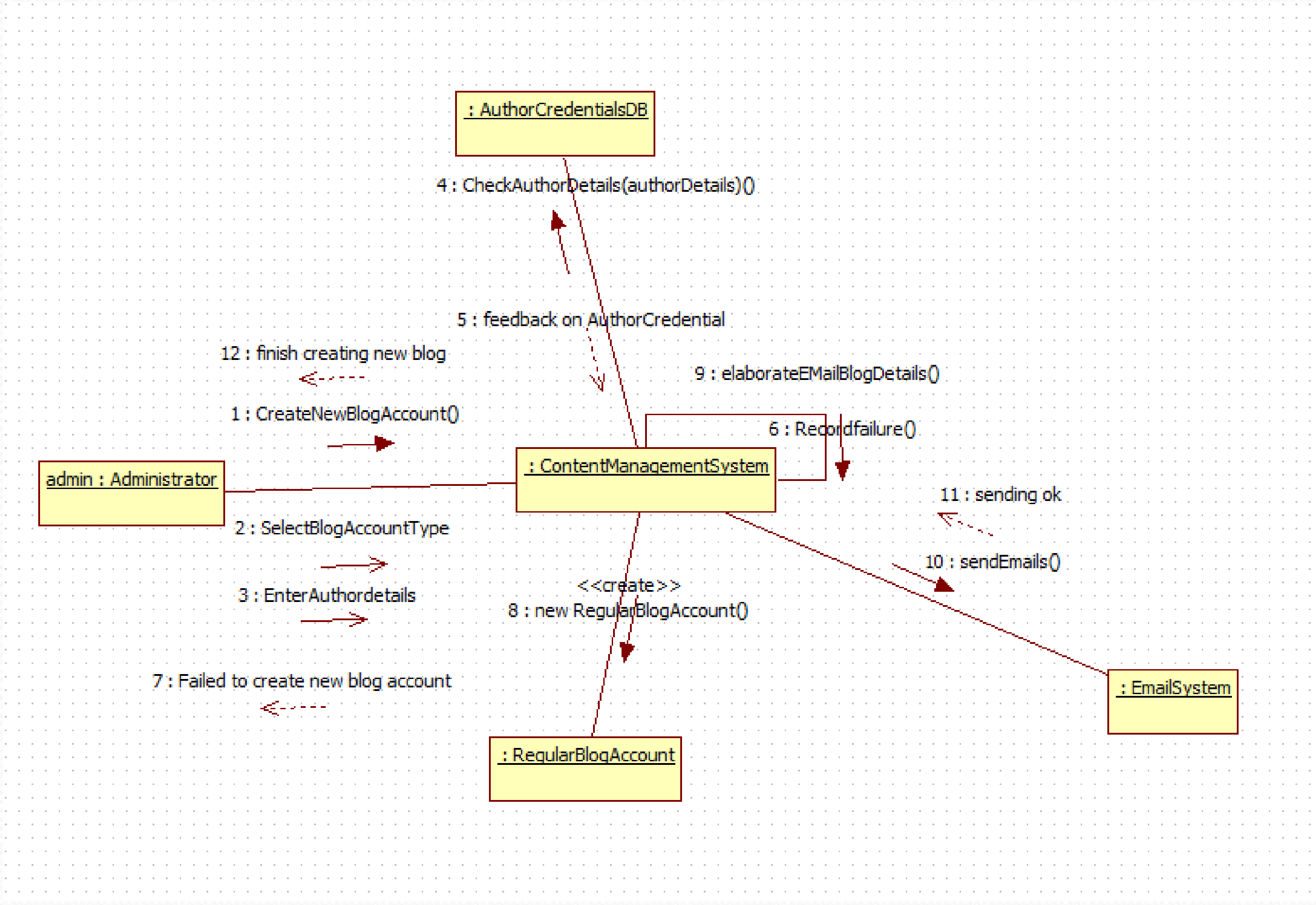


This collaboration model is automatically generated from the Email check sequence diagram with StarUML

Automatically convert Sequence diagram to Collaboration diagram



Create a new Blog example


: AuthorCredentialsDB
4 : CheckAuthorDetails(authorDetails)()
5 : feedback on AuthorCredential
12 : finish creating new blog
9 : elaborateEMailBlogDetails()
1 : CreateNewBlogAccount()
6 : Recordfailure()
admin : Administrator
: ContentManagementSystem
11 : sending ok
2 : SelectBlogAccountType
10 : sendEmails()
<<create>>
3 : EnterAuthordetails
8 : new RegularBlogAccount()
7 : Failed to create new blog account
: EmailSystem
: RegularBlogAccount

State transition model

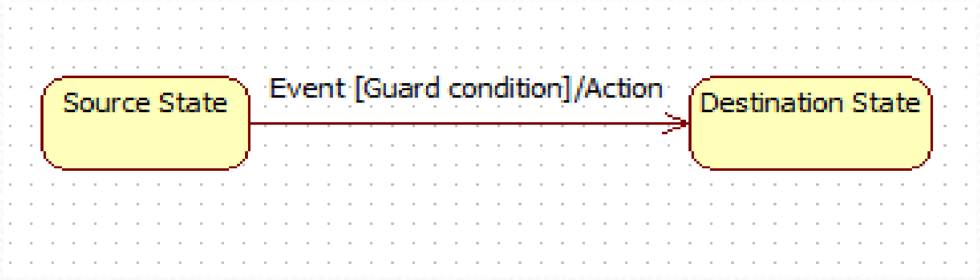
- Objects of the same class receive the same message but may respond differently
  - What dependencies are between the state of an object and its reaction to messages or events ?
- A state transition model specifies dynamic changes in a class. It describes various states in which objects of a class can be, behaviors in these states and events that cause state changes (reaction to that events)

State transition model

- A state transition diagram describes objects life cycle
- A state of an object is designated by the current values of the object's attributes
- E.g.: A copy of a book is *available* or *borrowed*, an order is *paid* or *unpaid*, an insurance claim is *processed* or *unprocessed*.

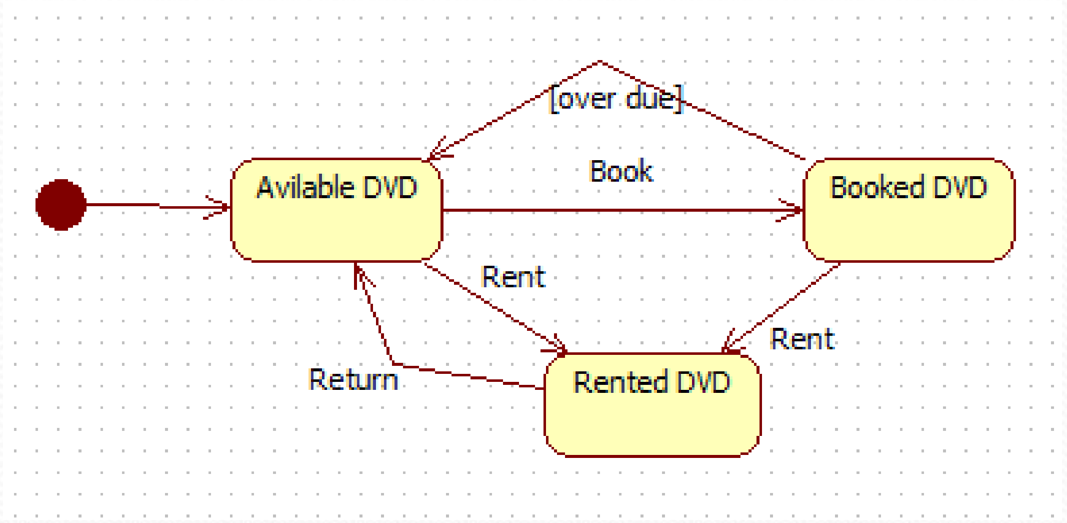
Concepts

- Object state—object situation in which the object satisfies certain conditions, it can perform relevant actions, or wait for certain events
- transition—the movement from one state to another
- event—an occurrence that causes the object to exhibit some predictable form of behavior
- condition—additional condition to events
- action—process that occurs as a consequence of making a transition – reaction of object to the event

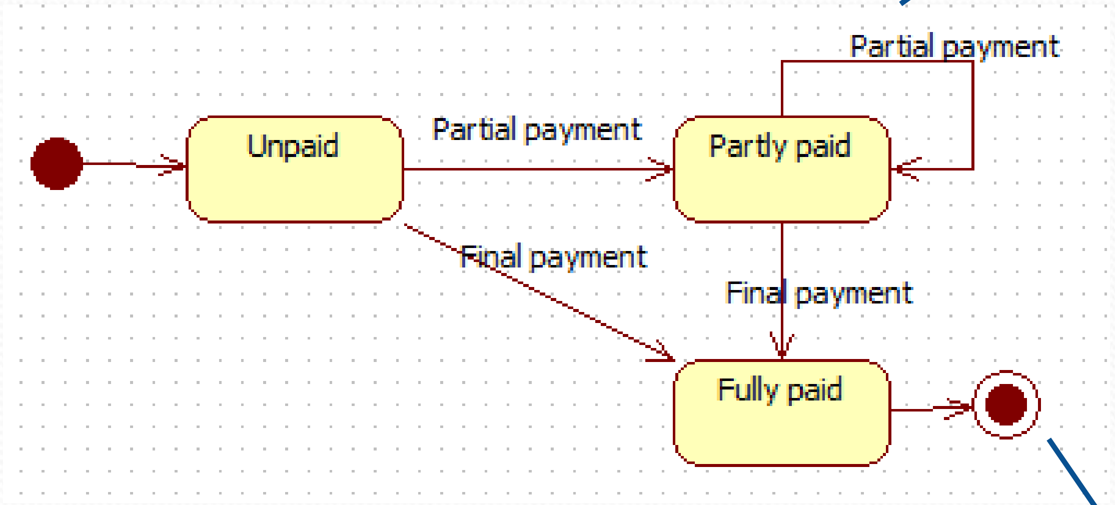


Object life cycle – DVD object

- A DVD can have states *available, booked, rented*
- Start marker/initial state (black blob with an unlabeled arrow): indicates the start of object life
- Stop marker/final state (black blob with a ring around it): object has reached the end of its life
- In a state transition diagram, there is one initial state and zero or several final states



State diagram –Rental Payment object



Self transition

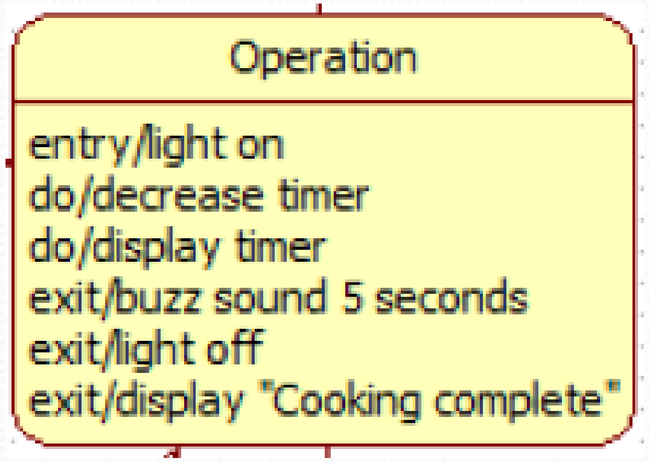
Final state

State-transition diagram

- State components :
  - Name describes object state
  - Action on internal event (*on event/action)*: the action is carried out at the time of occurrence of an event which does not lead to another state
  - Do action (*do / action*) : recurring or significant action, carried out in the state
  - Entry/Exit action (*entry/ action*), (*exit / action*) : action is executed instantly as of entry/exit state

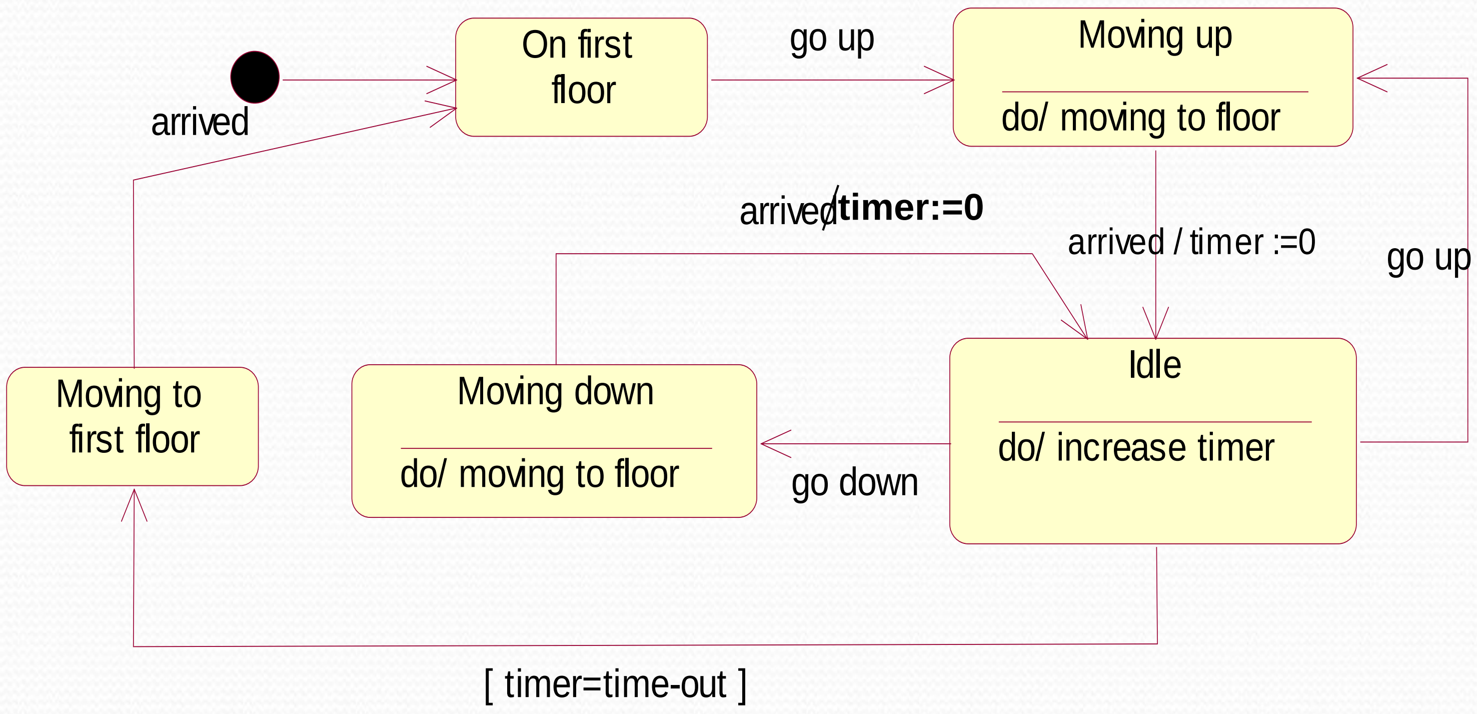
Example – Oven in operation state

- When an Oven is in operation state , the interior oven light is on. Display shows the timer countdown. On completion of cooking, the buzzer is sounded for five seconds. Oven light is off. Display shows 'Cooking complete' while buzzer is sounding



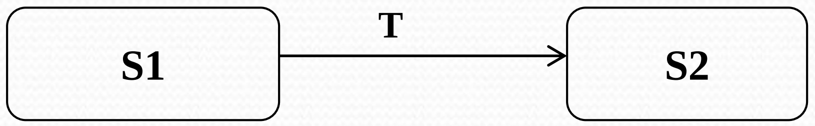
State-transition diagram (UML)

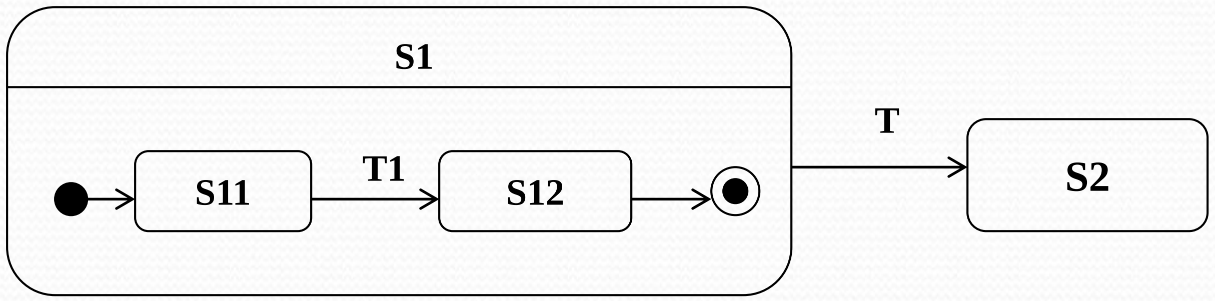
Exemple of a state-transition diagram of lift [Erikssons98]



Sub-states

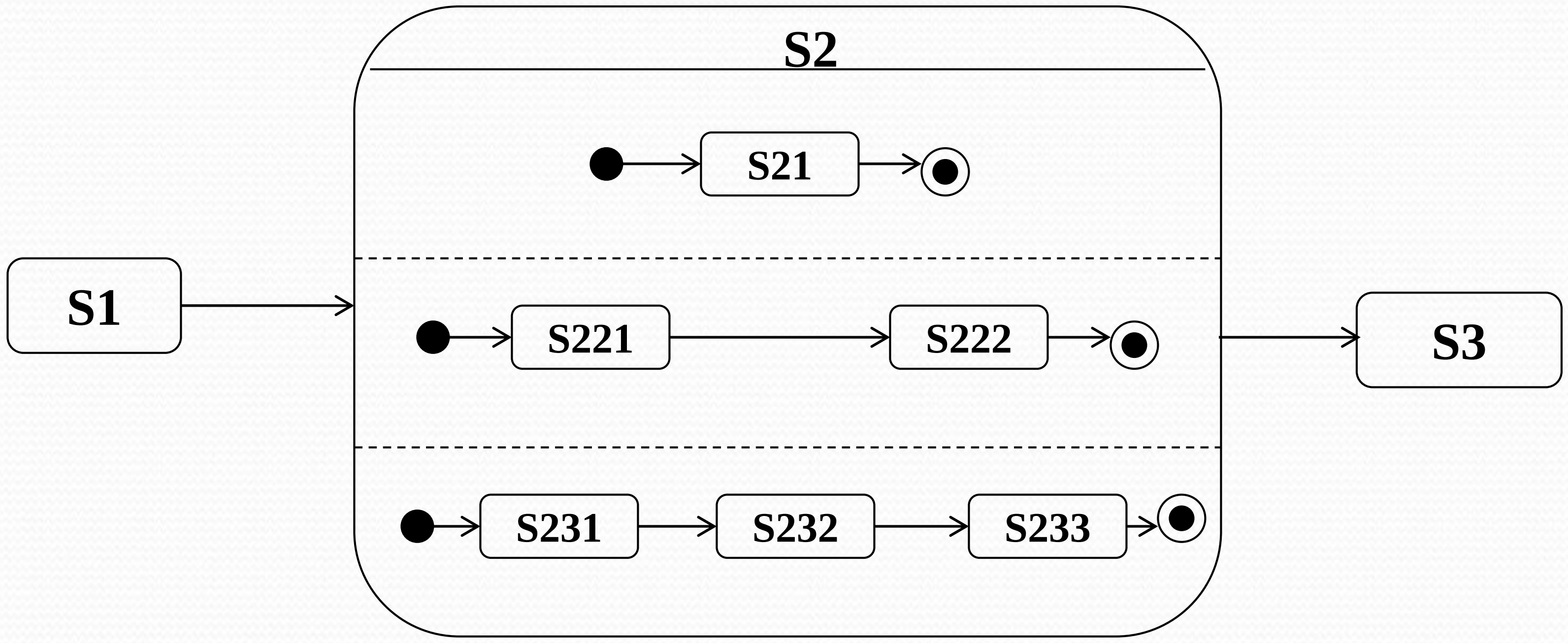
- Modelling complex behaviors. A state may be decomposed by several sub-states.
  Sequential sub-state :



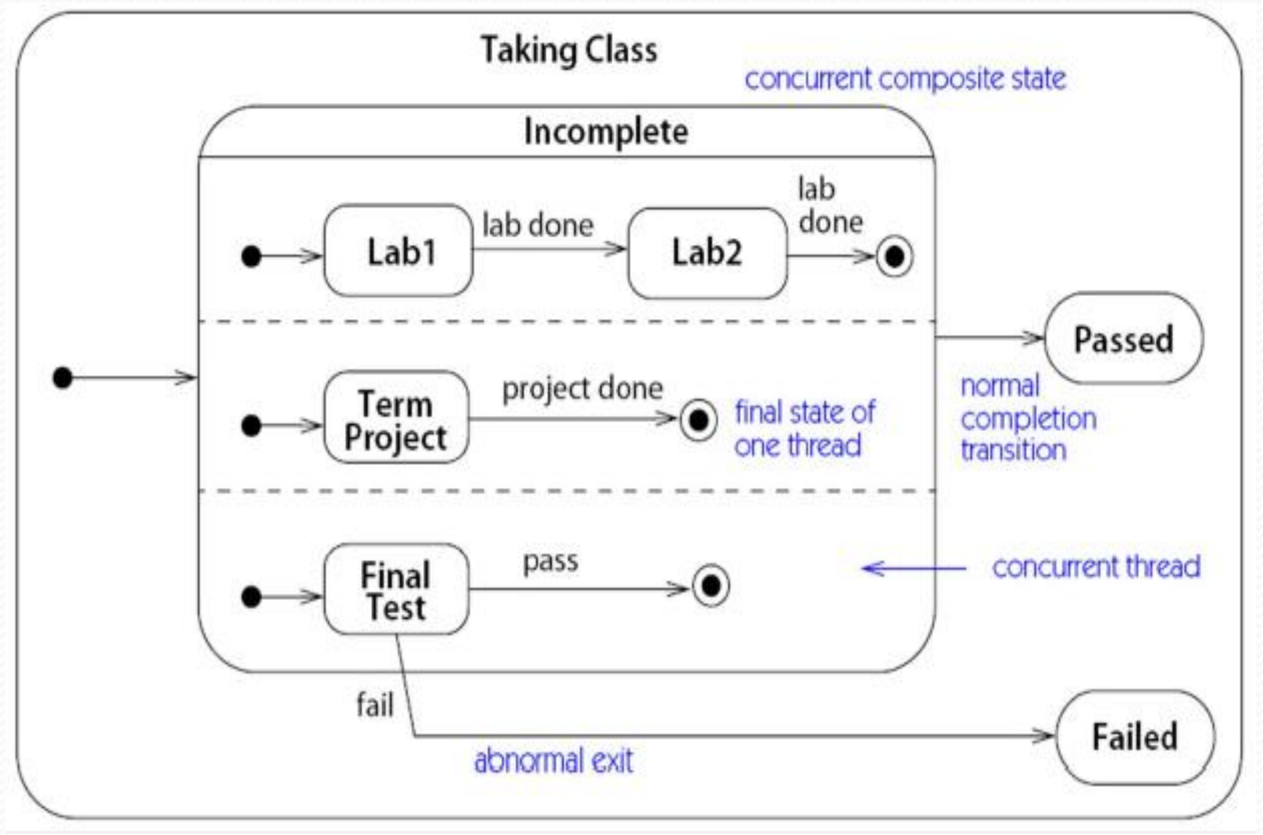


Sub-states

- Concurrent sub-states
  - Object in composed state in fact is in sequential state of each concurrent sub state.



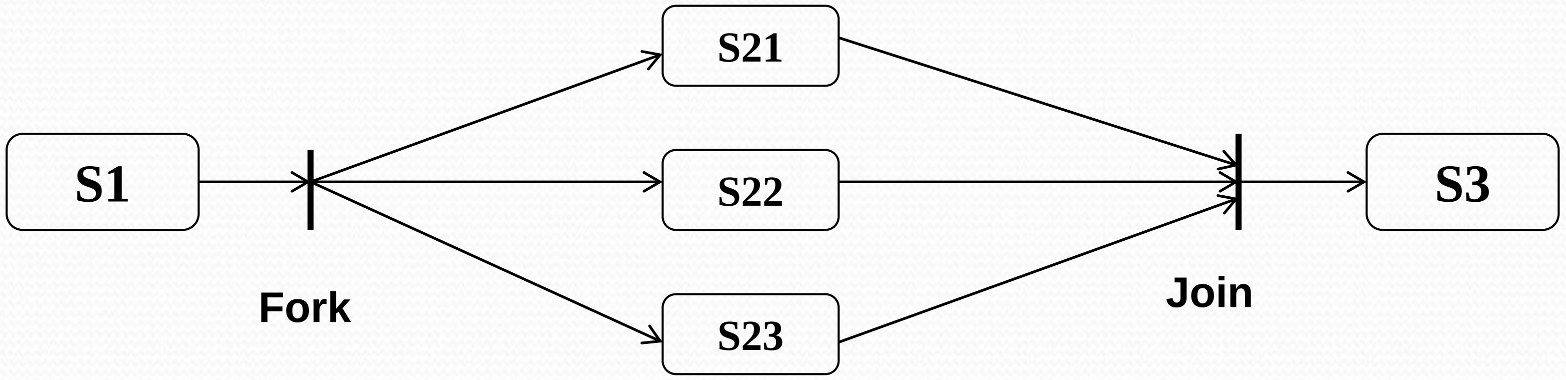
Example of sub-states



[Nguyen, 2004]

State-transition diagram (UML)

- Fork and Join/Transition Synchronization



Activity diagrams

- Represent the flow of logic in object oriented programs/computation (concurrent control and sequential control)
- Supplement the use-case by providing a diagrammatic representation of procedural flow of action in the execution of a use case
- Activity diagram contains activities, transitions between the activities (flow of actions), decision points, and synchronization bars

Components

- Actions
  - Each use case can be modelled by one or more activity graphs
  - Actions can be discovered base on the use case description
- Transition: flow of activity, when an activity is finished it is automatically moved to the next activity

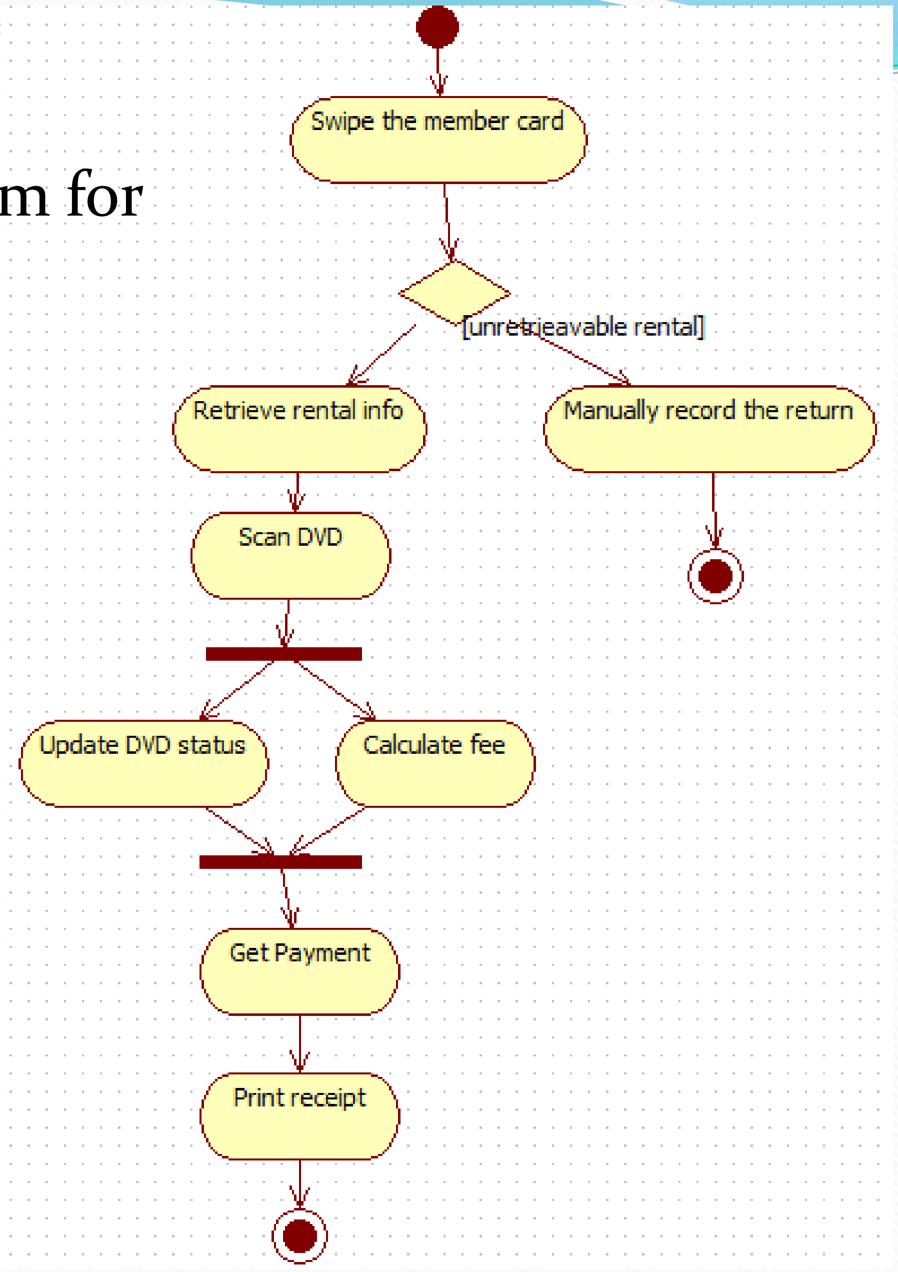
Example of Return DVD use case

Flow of events of Return DVDs use case
Basic flows:

1. Use case start by Staff scanning the member card
2. The system retrieve the renting order
3. Staff scan each DVDs
4. The system retrieve the DVD
5. The system updates DVD status to available
6. The system checks if there is late returns and the corresponding fee
7. User needs to pay for late returns
8. Staff prints out the receipt
9. The use case finishes

Flows of events of Return DVDs use case
Alternative flows:
2.1 Cannot retrieve any renting for that customer
2.2 Record the case and proceed manually and flag the case to check the cause
2.3 Stop the use case

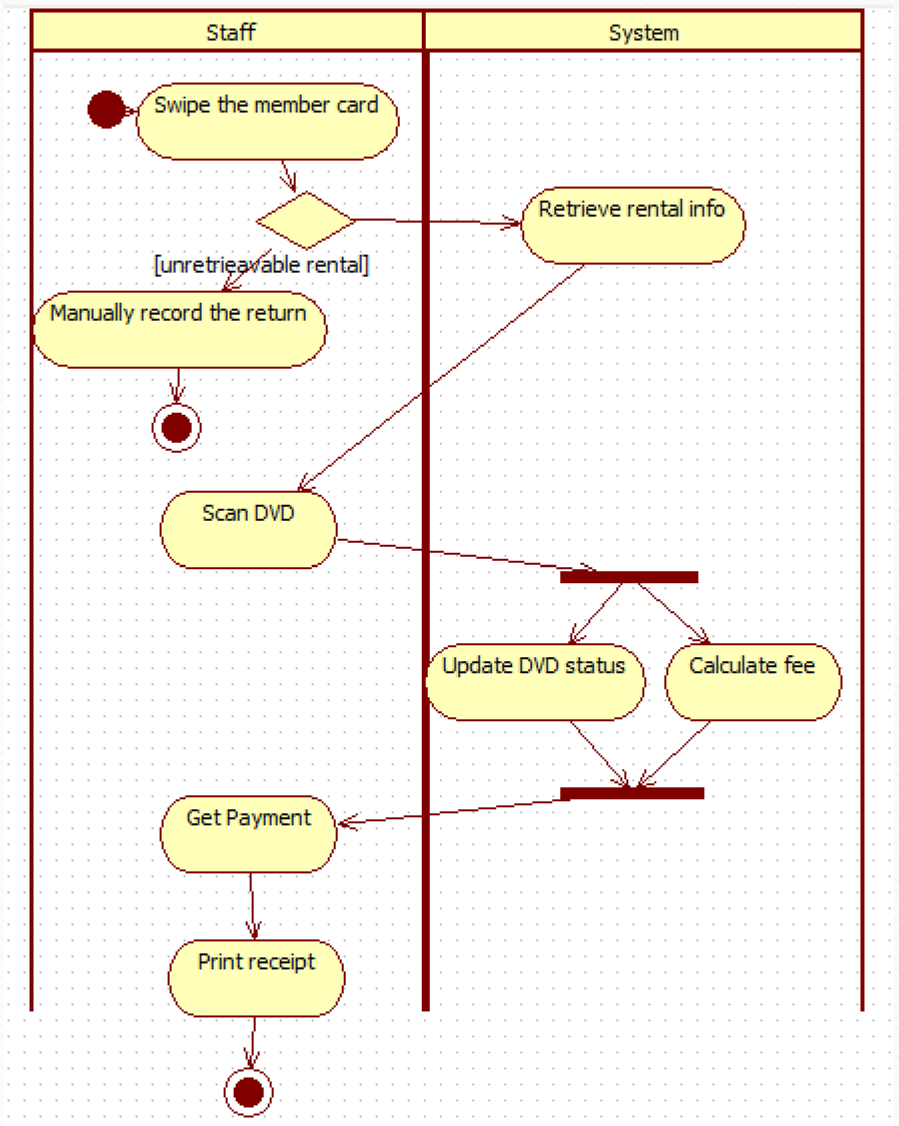
Activity diagram for Return DVD



Swimlane in Activity Diagram

- Represents the responsibility of actors/roles within an activity diagram
- Group activities of the same actor in a swimlane

Swimlane diagram example



Key points 1

- System analysis focuses on modelling 3 aspects of the system: Functional aspect, Informational aspect and Behavioural aspect
- Functional aspect is described with Use case diagrams which show the interactions between external actors (users and other systems) with the system in term of intended functions
- Class diagrams are used to define the static structure of classes in a system and their associations

Key points 2

- Behavioral models are used to describe the dynamic behavior of an executing system. This can be modeled from the perspective of the data processed by the system or by the events that stimulate responses from a system
- Sequence diagrams show the interaction between system objects or between objects in a use case
- Activity diagrams may be used to model the processing of data, where each activity represents one process step.
- State diagrams are used to model an object (or even system) 's behavior in response to internal or external events