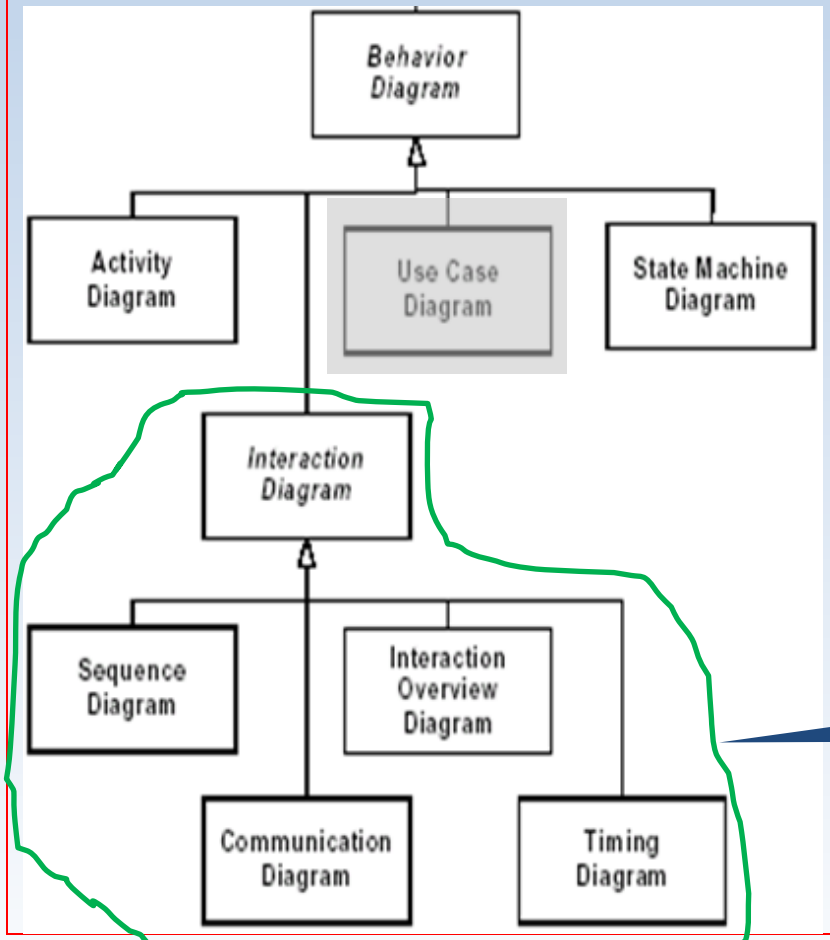
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

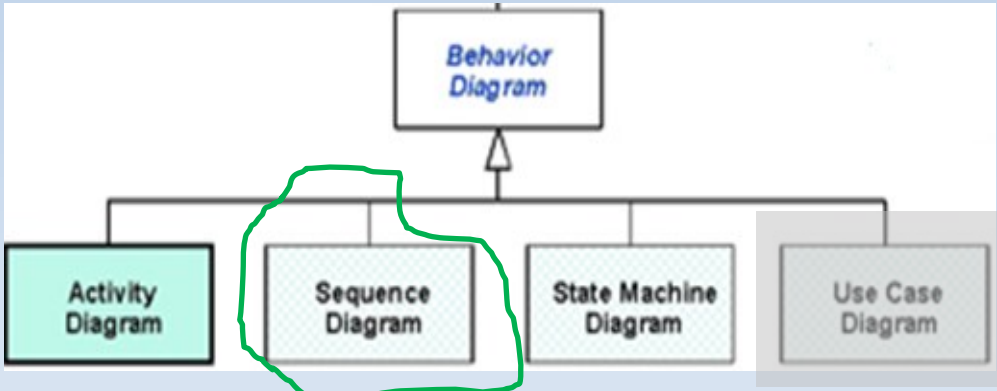
Behavioural Modelling with UML (and SysML)

Message-based Behaviour

UML



SysML



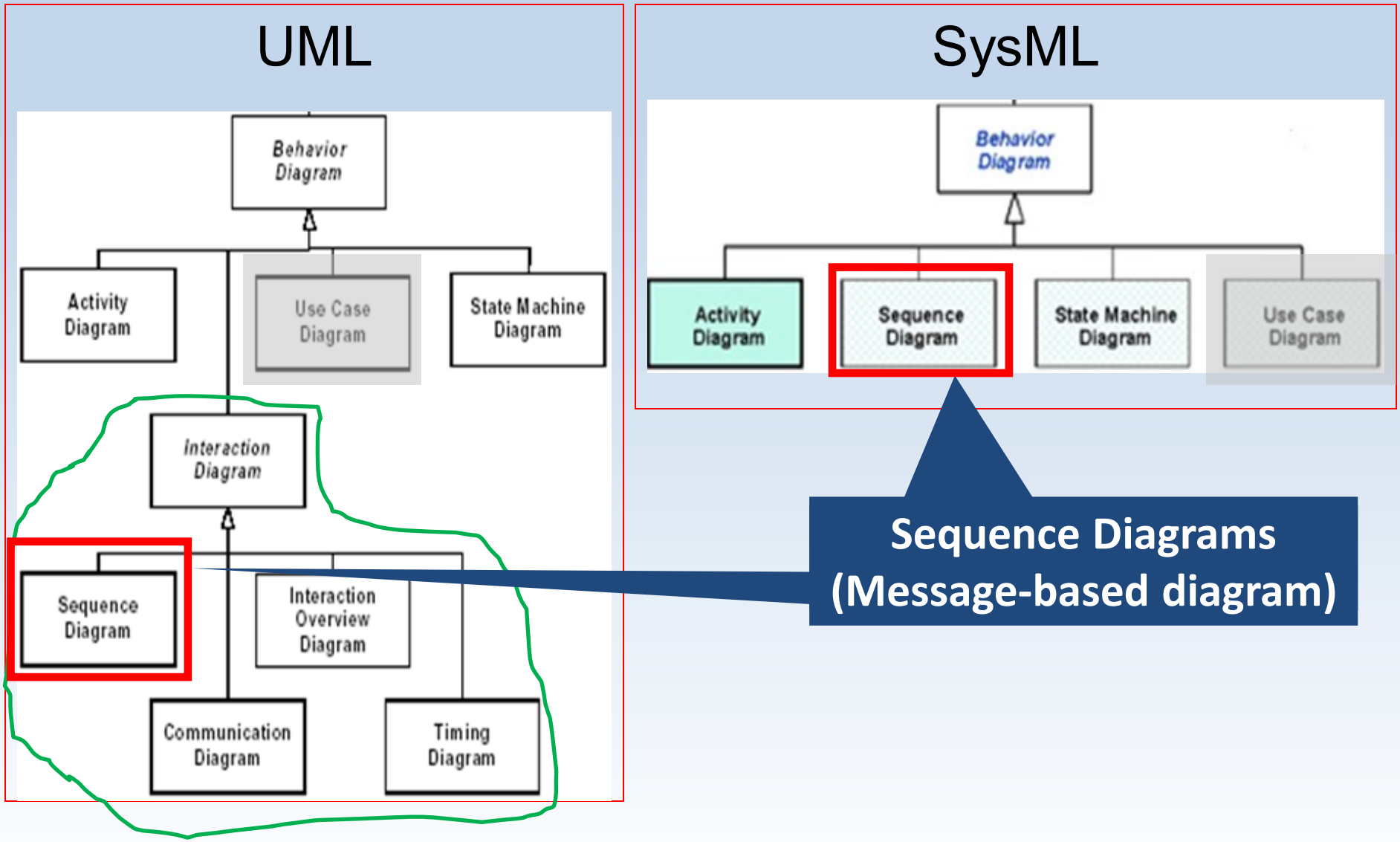
Message-based diagrams

Message-Based Behaviour

- Describe how instances (objects) communicate with each other.
- Describe how a function is performed as a result of a sequence of messages sent and received between a set of objects.

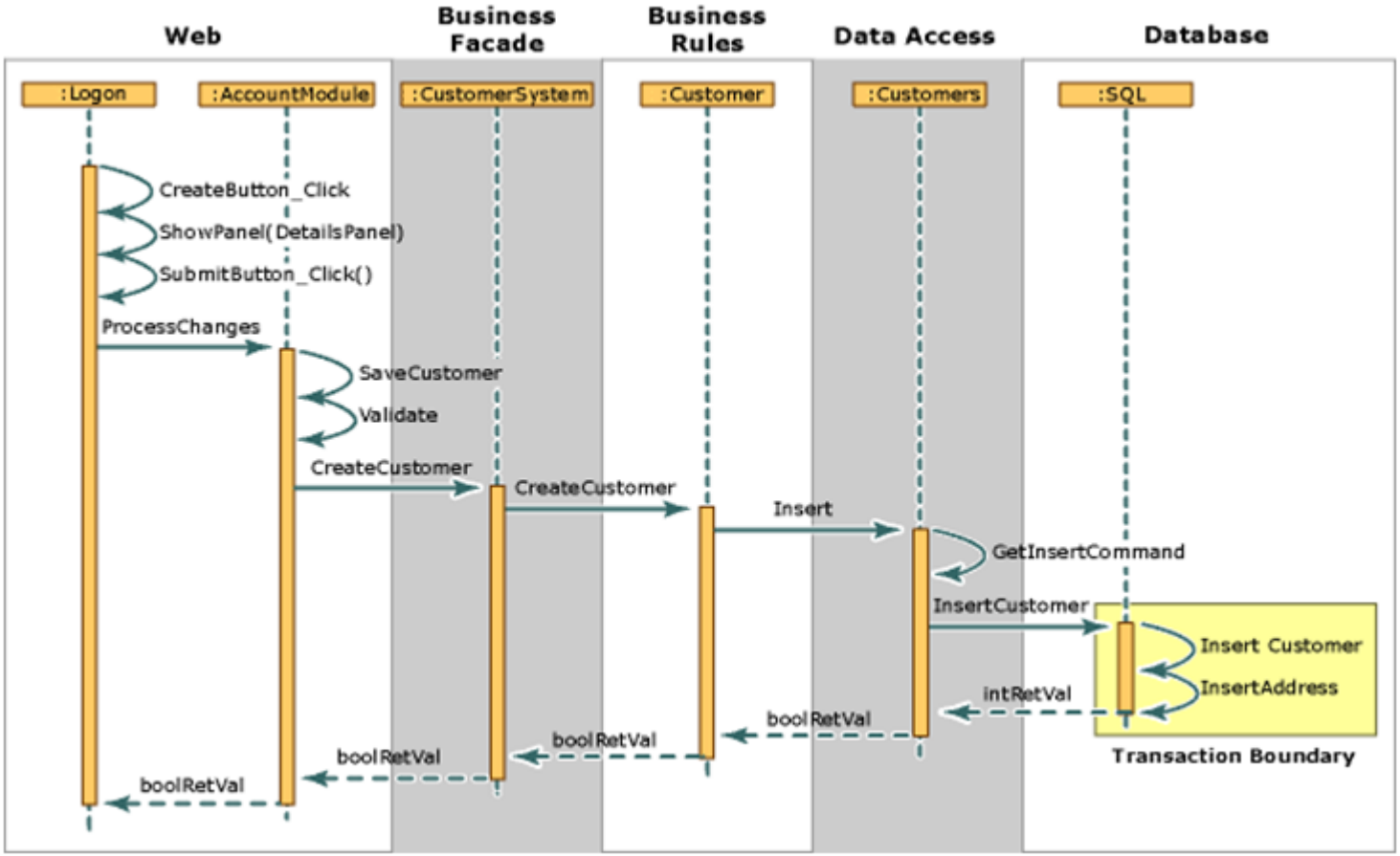
- In UML
  - Interaction diagrams
    - Sequence diagrams, Communication diagrams, Timing diagrams, Overview diagrams
- In SysML
  - Sequence diagrams

Message-based Behaviour



Sequence Diagrams (UML & SysML)

- A sequence diagram represent interactions between system parts, i.e. how objects communicate with each other.
- A sequence diagram describes a **specific execution scenario.**



Sequence Diagrams and Use Cases

- **A sequence diagram can *realize* a use case scenario(s).**
- The sequence diagram is used primarily to show the **interactions between objects** in the **order** that those interactions occur.

- Sequence diagrams may *refine* the requirements expressed as use cases (scenarios).

- Use cases are often refined into one or more sequence diagrams, depending on complexity.
  - NOTE: activity diagrams may be used as well.

Sequence Diagrams

The **sequence diagram** may describe the **interactions between actors and systems** or **between parts of a system**.

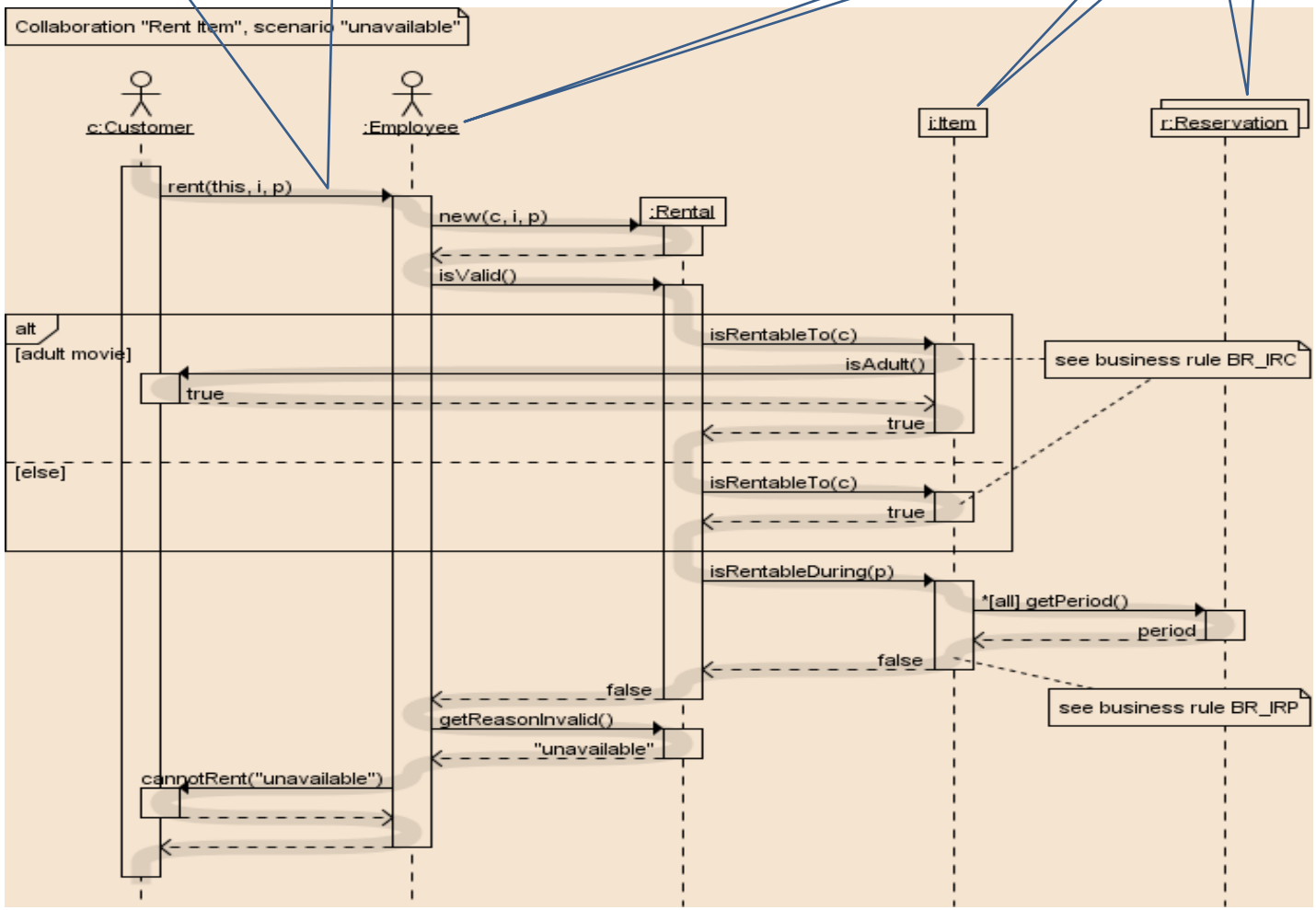
This diagram represents the **sending and receiving of messages** (**interactions**) between the interacting **entities** called **lifelines**, where time is represented along the vertical axis.

The sequence diagrams can represent highly complex interactions with **special constructs** to represent various types of control logic, reference interactions on other sequence diagrams, and decomposition of lifelines into their constituent parts.

Sequence Diagrams

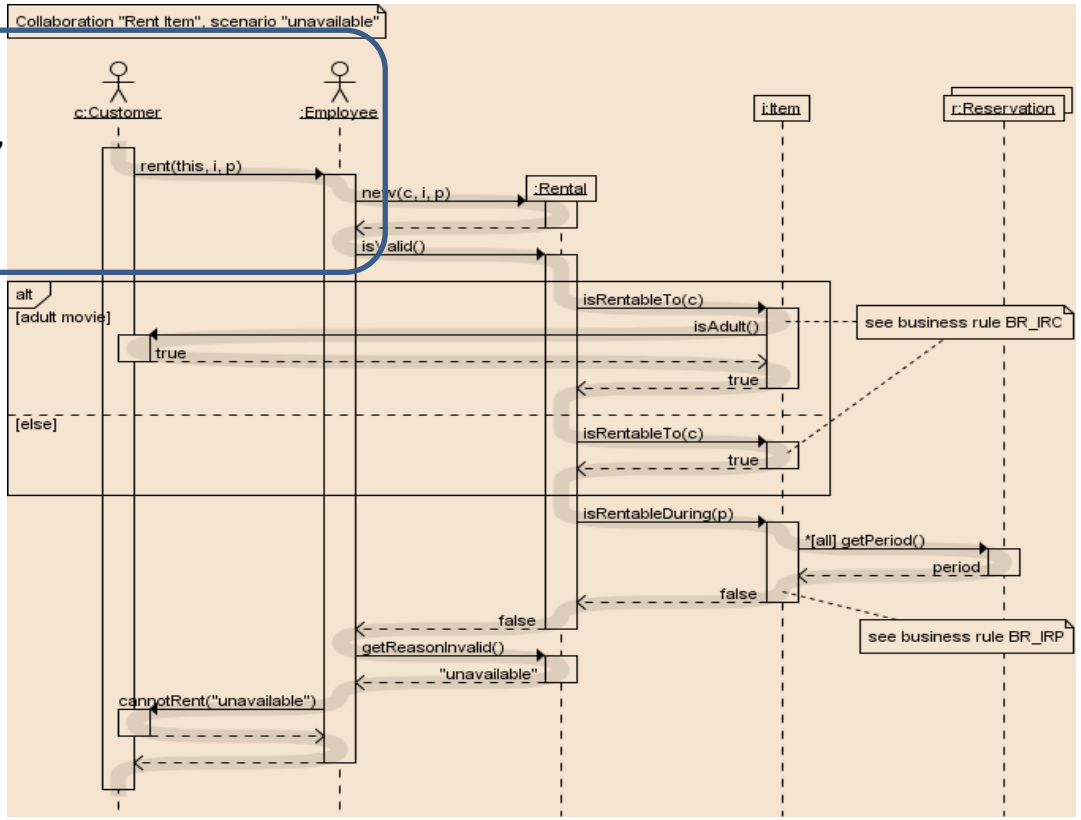
(warning: this diagram is not OK! See next slide ;-)

A sequence diagram describes a **specific execution context (i.e. a specific execution scenario).** Represent interactions between system parts, i.e. how **objects communicate with each other.**



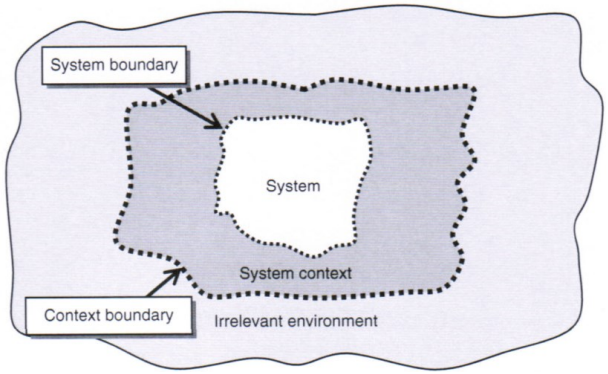
Sequence Diagrams

BTW, according to the UML language this is syntactical correct, BUT it is an incorrect modelling of requirements for a logical system!



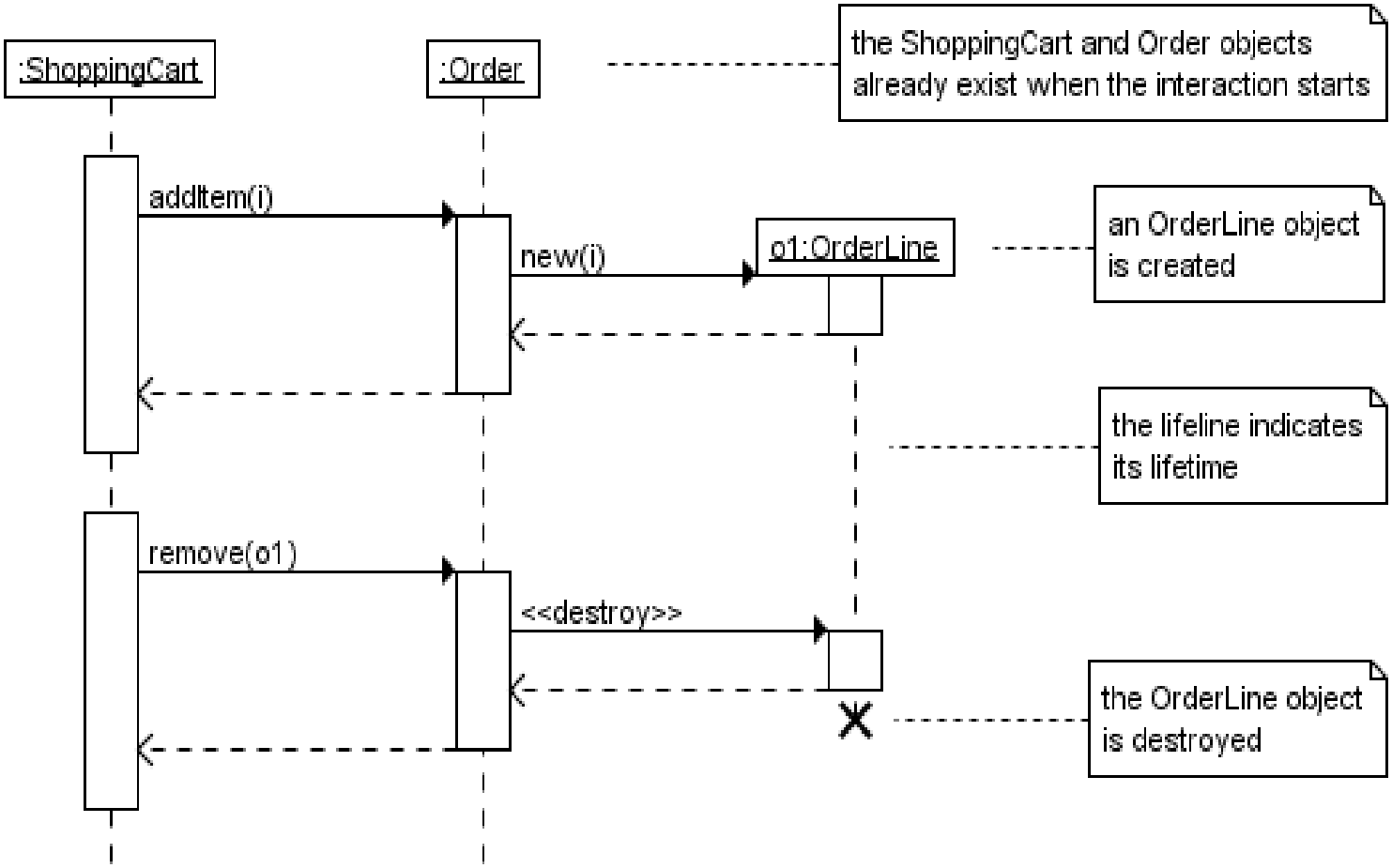
Why?

The reason is because it violates the principle of modeling a concerns that is outside to the system (the interaction between actors is a concern of the context of the logical system, BUT NOT OF THE SYSTEM!!!)



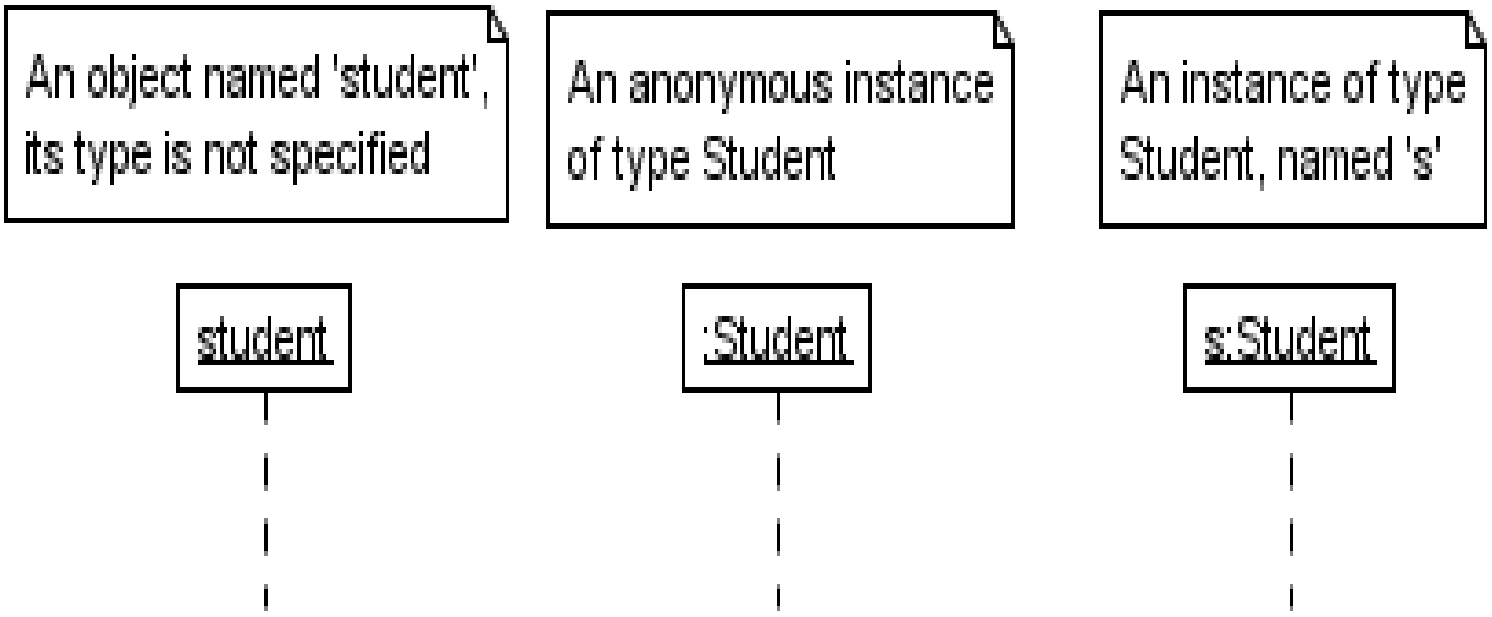
Sequence Diagram Notation

- Represent interactions between objects along time.
- Stress the lifetime of the objects in a specific execution context.



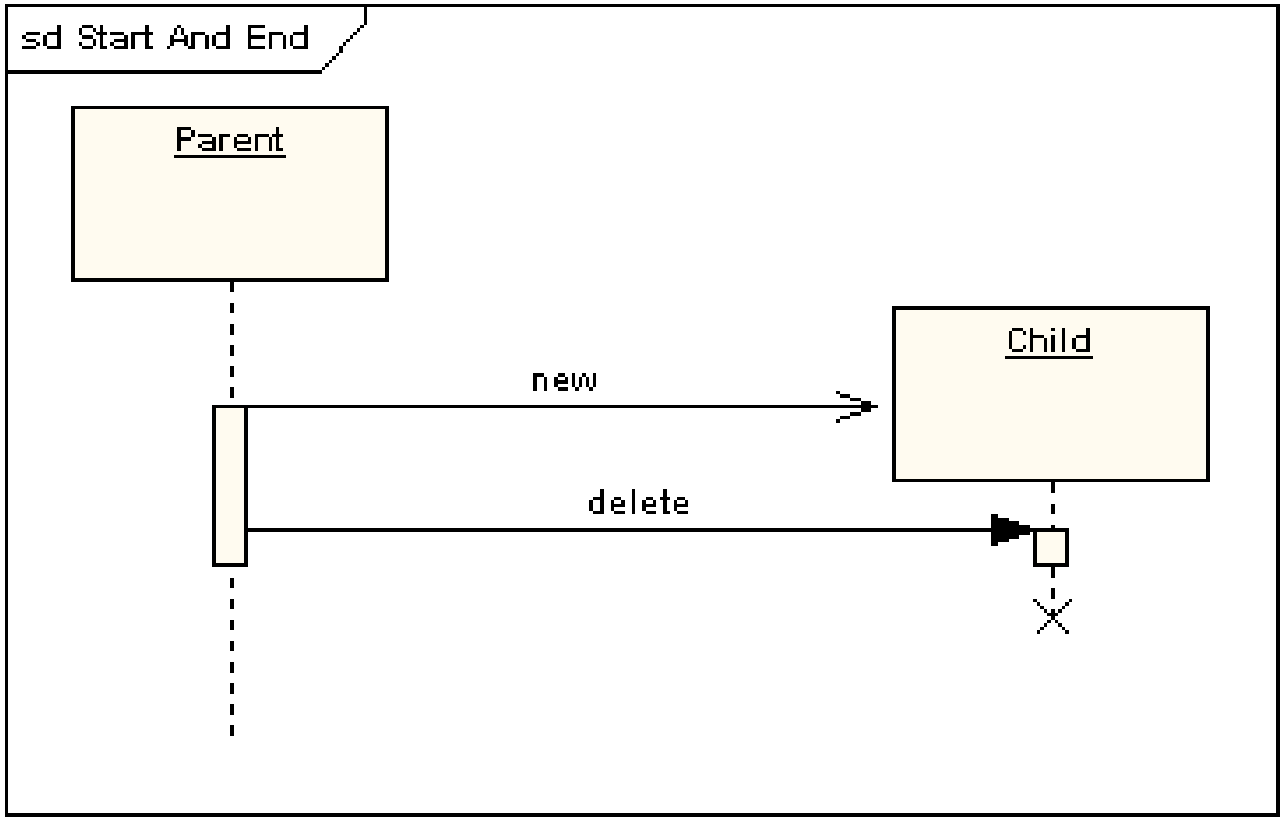
Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

Entities represented in sequence diagrams are instances

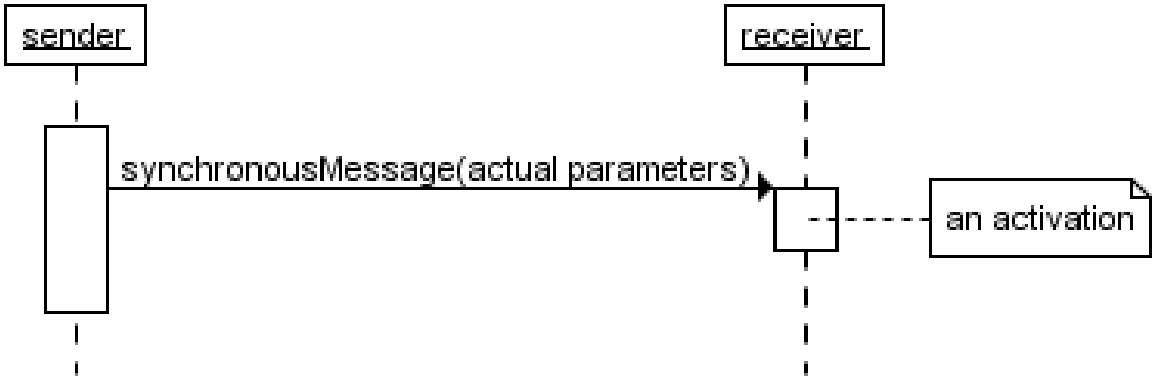


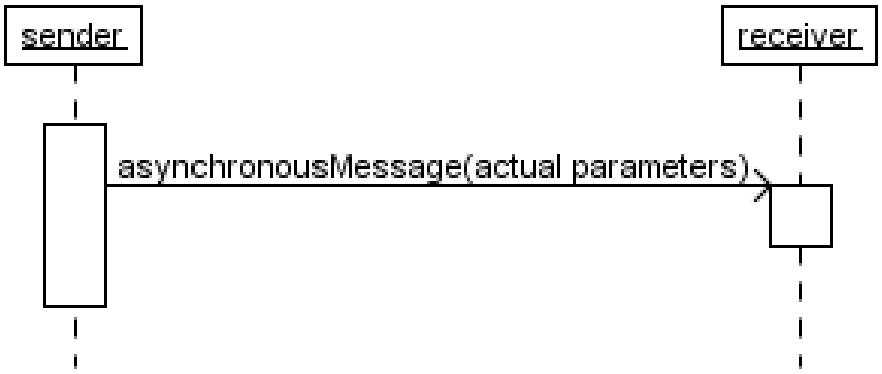
Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

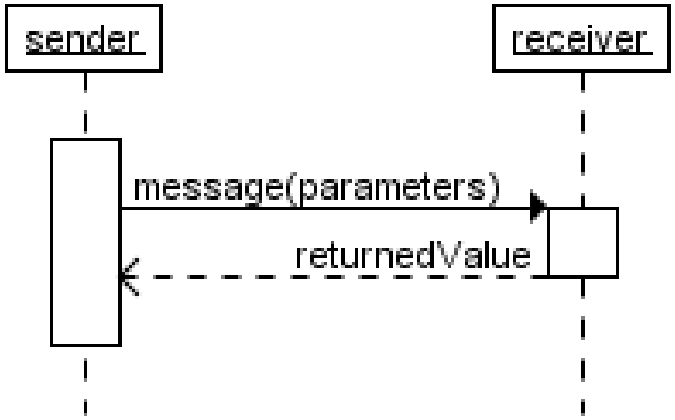
Creation and destruction



Synchronous, Asynchronous and Return Messages

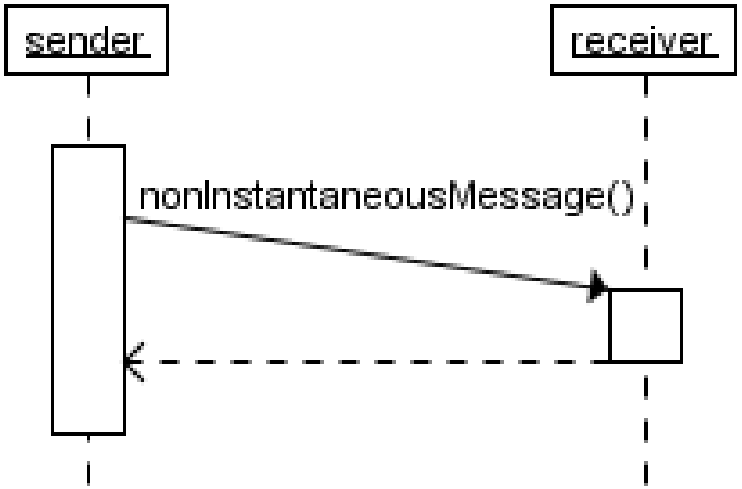


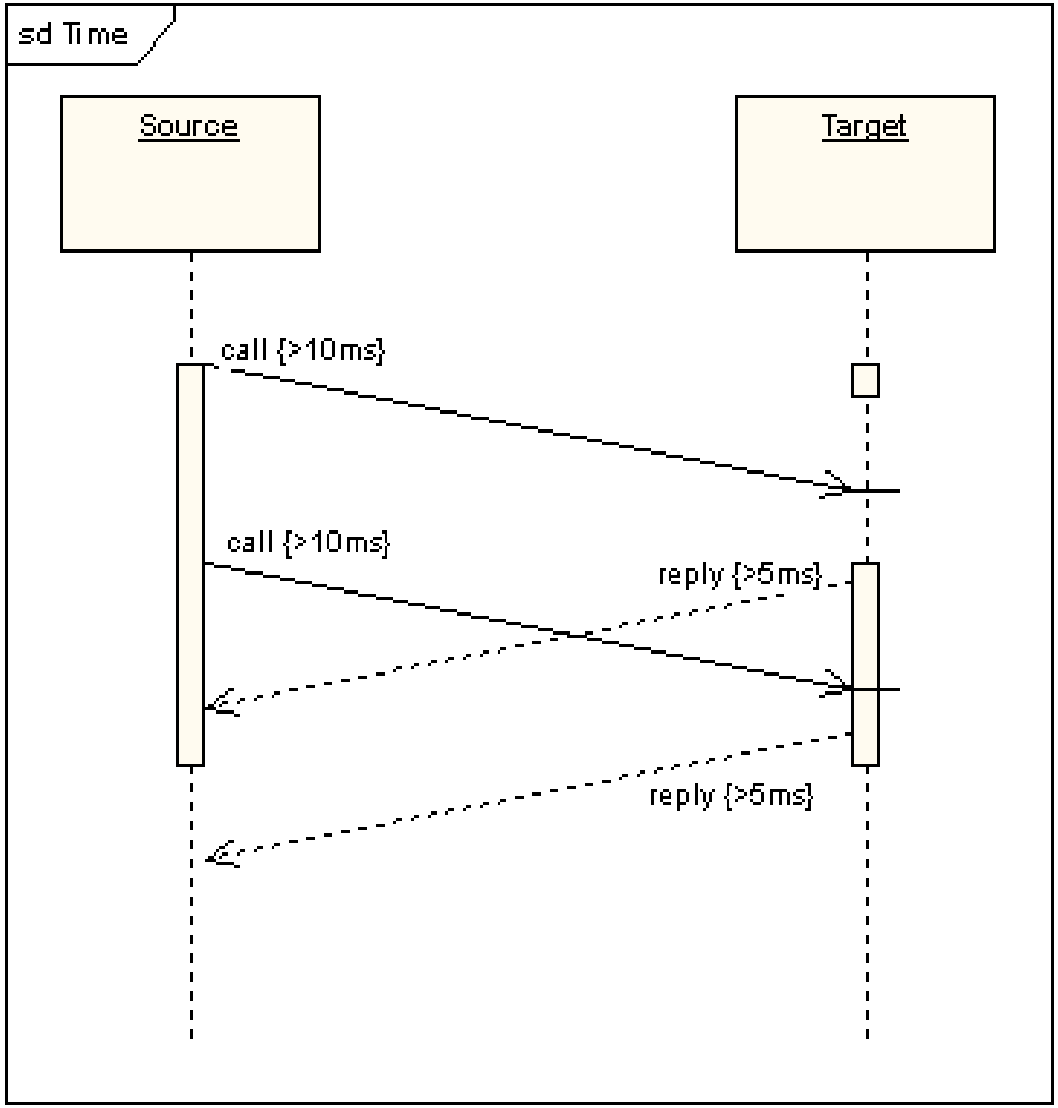




Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

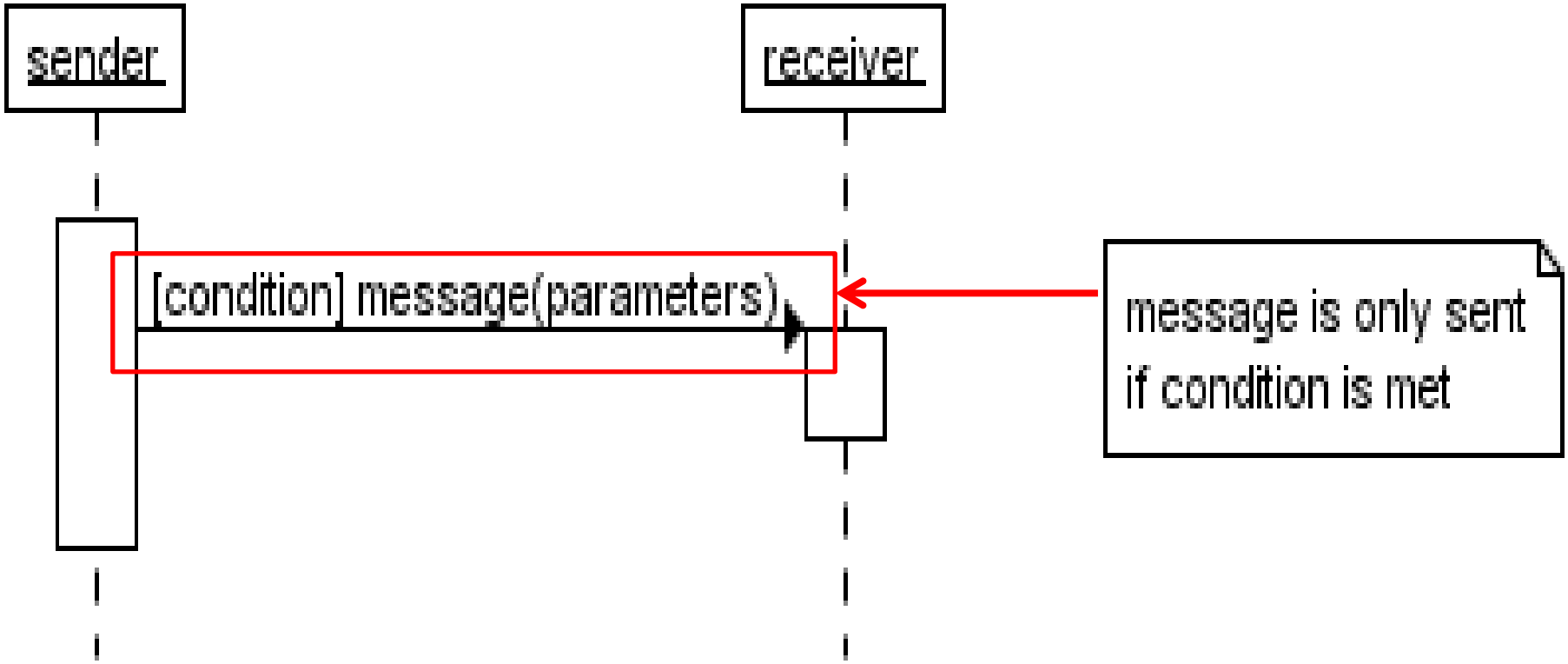
Messages and time constraints





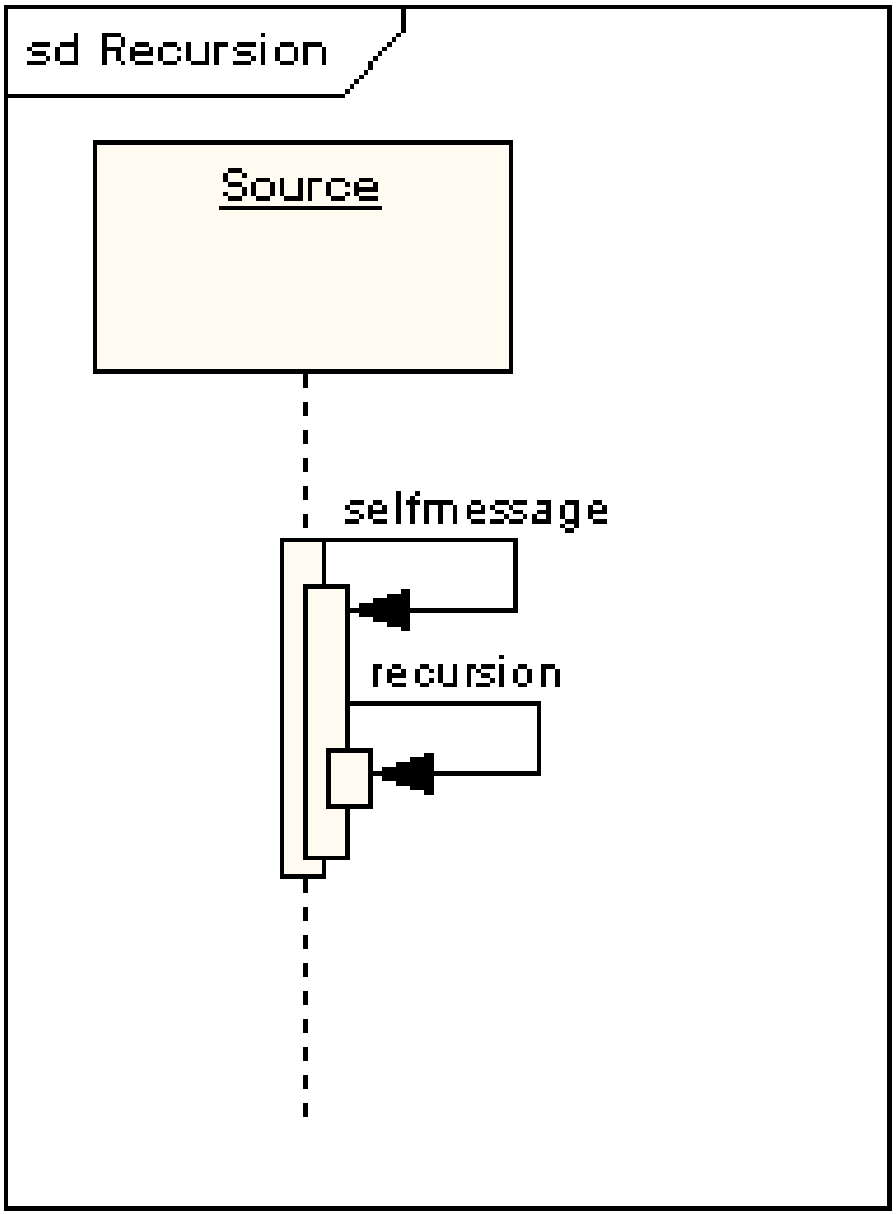
Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

Conditional interactions



Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

Recursion



Interaction Operators

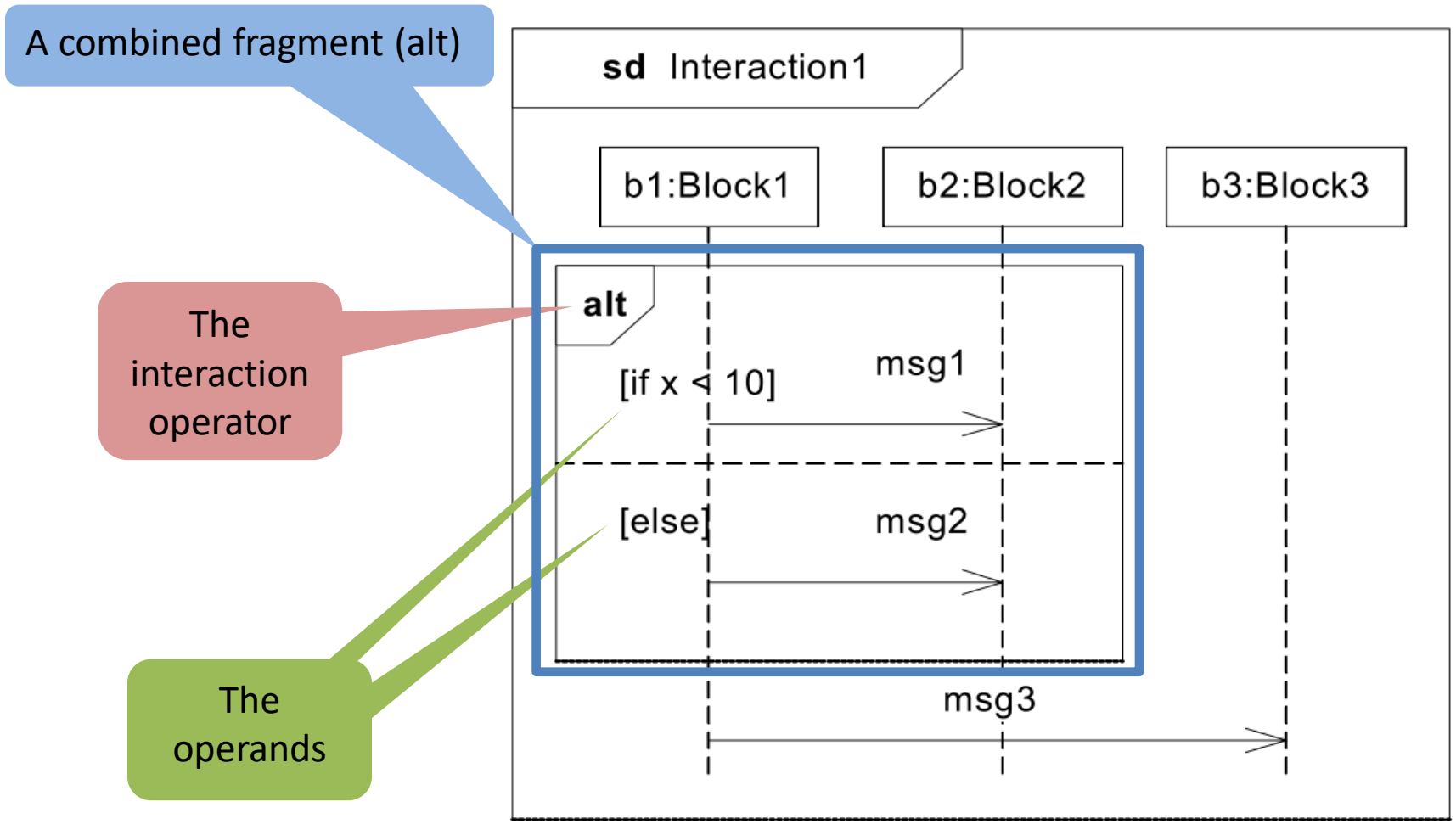
- **Branches and loops**
  - **alt, opt, break, loop**
- **Concurrency and order**
  - **seq, strict, par**
- **Filters and constraints:**
  - **critical, neg, assert, consider, ignore**

- The **interaction operators** apply to **combined fragments**

For further reading please refer to:

http://new.fengnet.com/book/UML/umlnut2-CHP-10-SECT-8.html

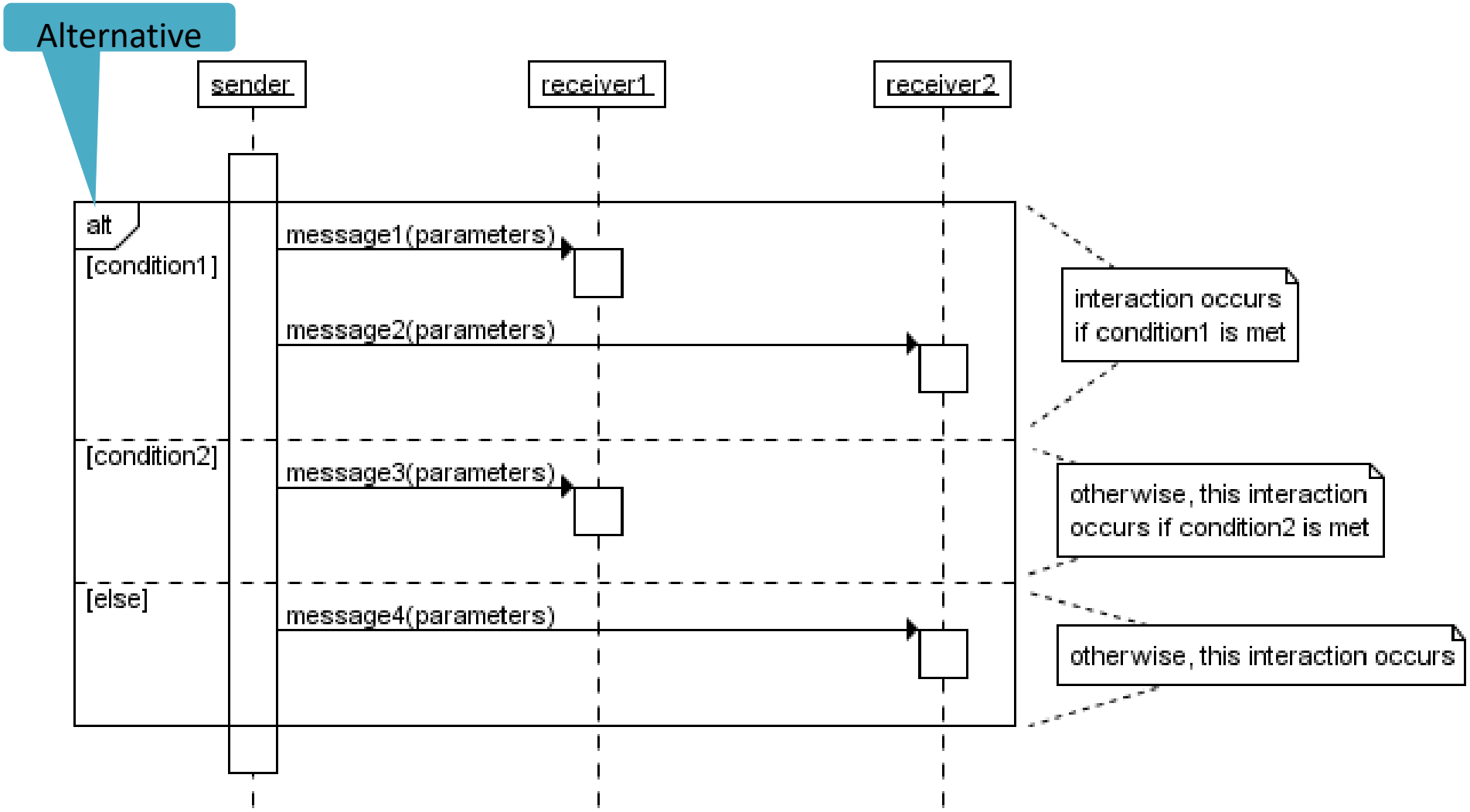
Combined Fragments



Basic Interaction Operators

- **Branches and loops:**
  - **alt, opt, break, loop**
- **Concurrency and order:**
  - seq, strict, par
- **Filters and constraints:**
  - critical, neg, assert, consider, ignore

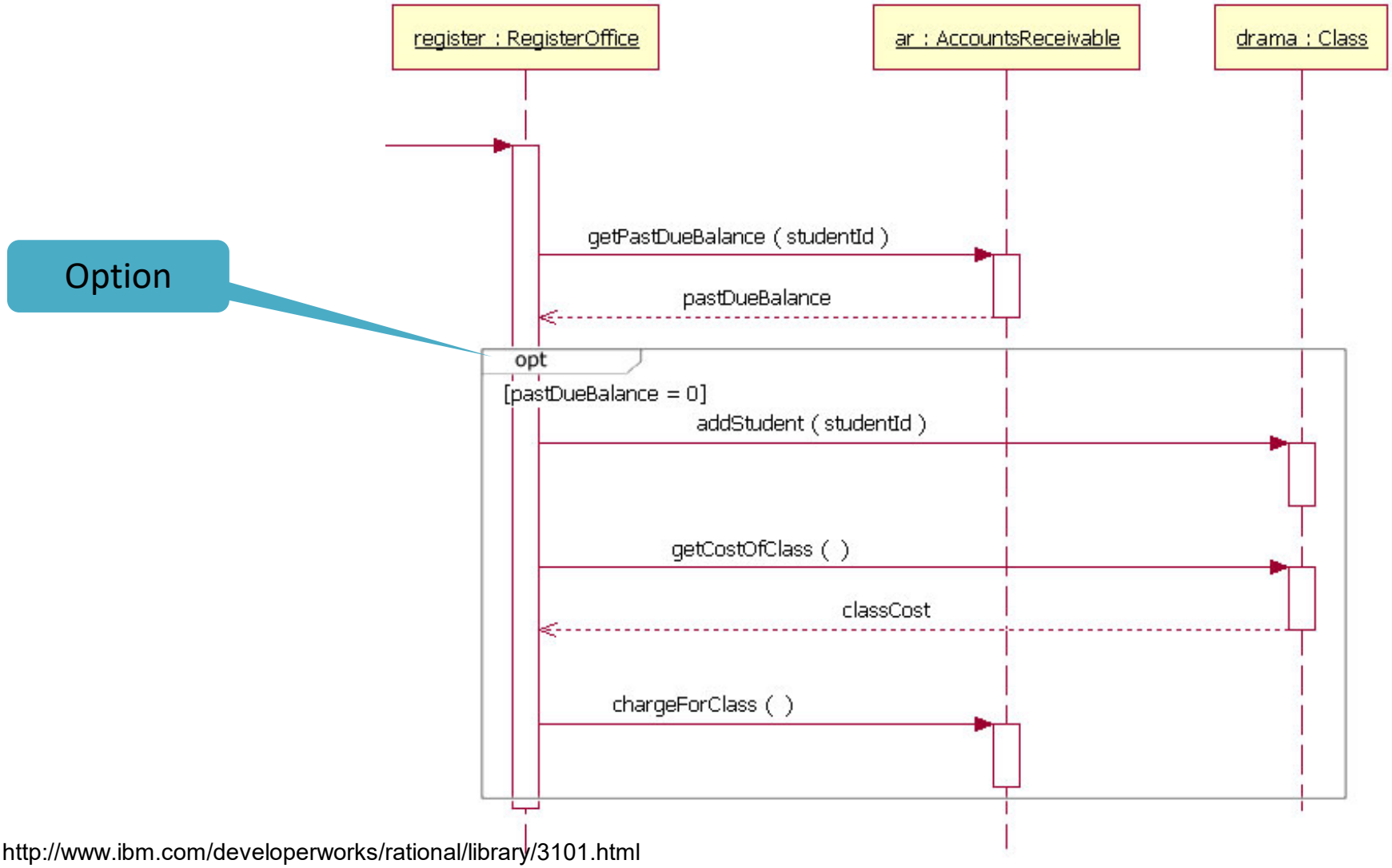
Combined fragments: alternative



Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

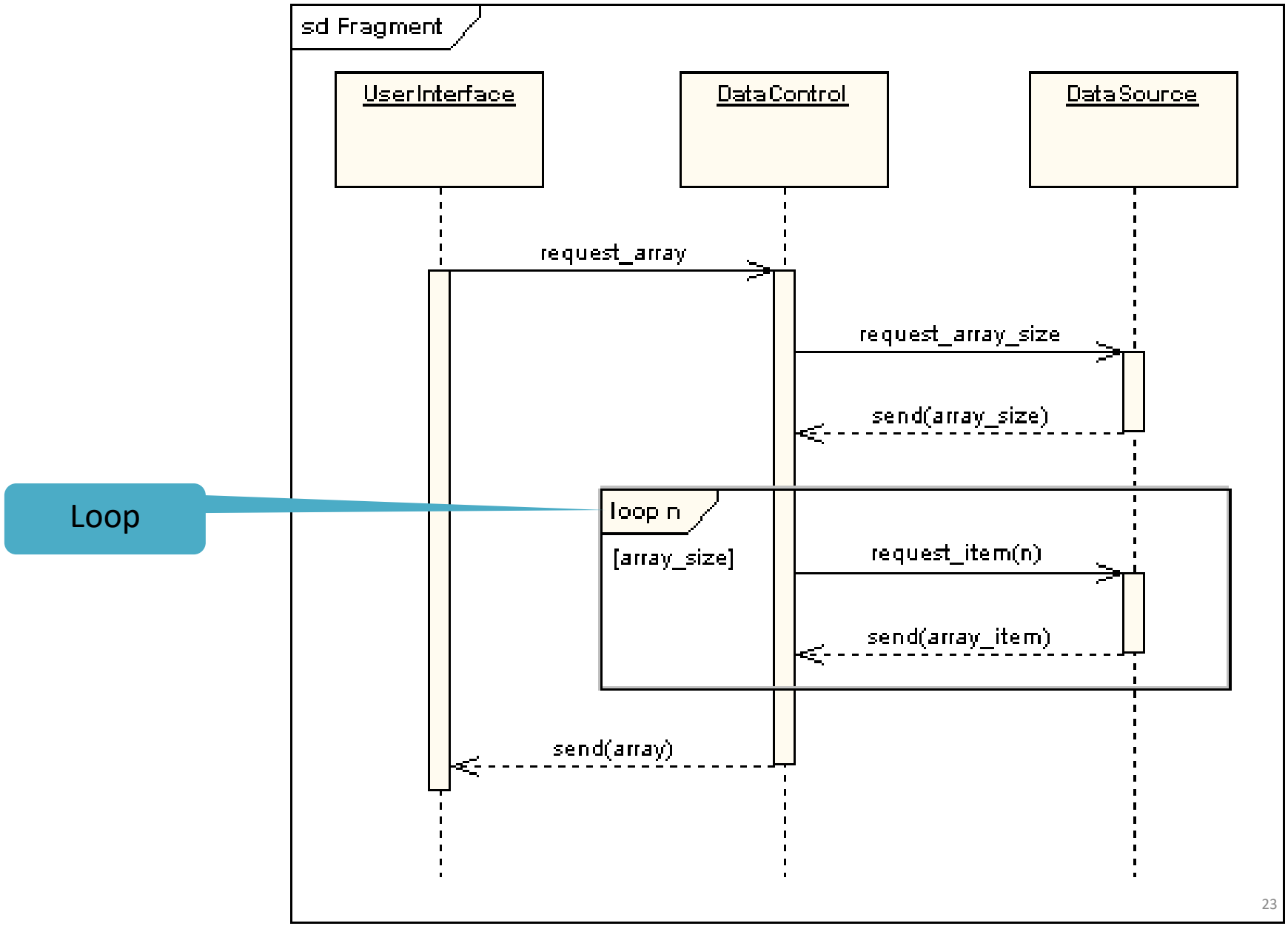
Combined fragments: option

opt is equivalent to alt with a single condition



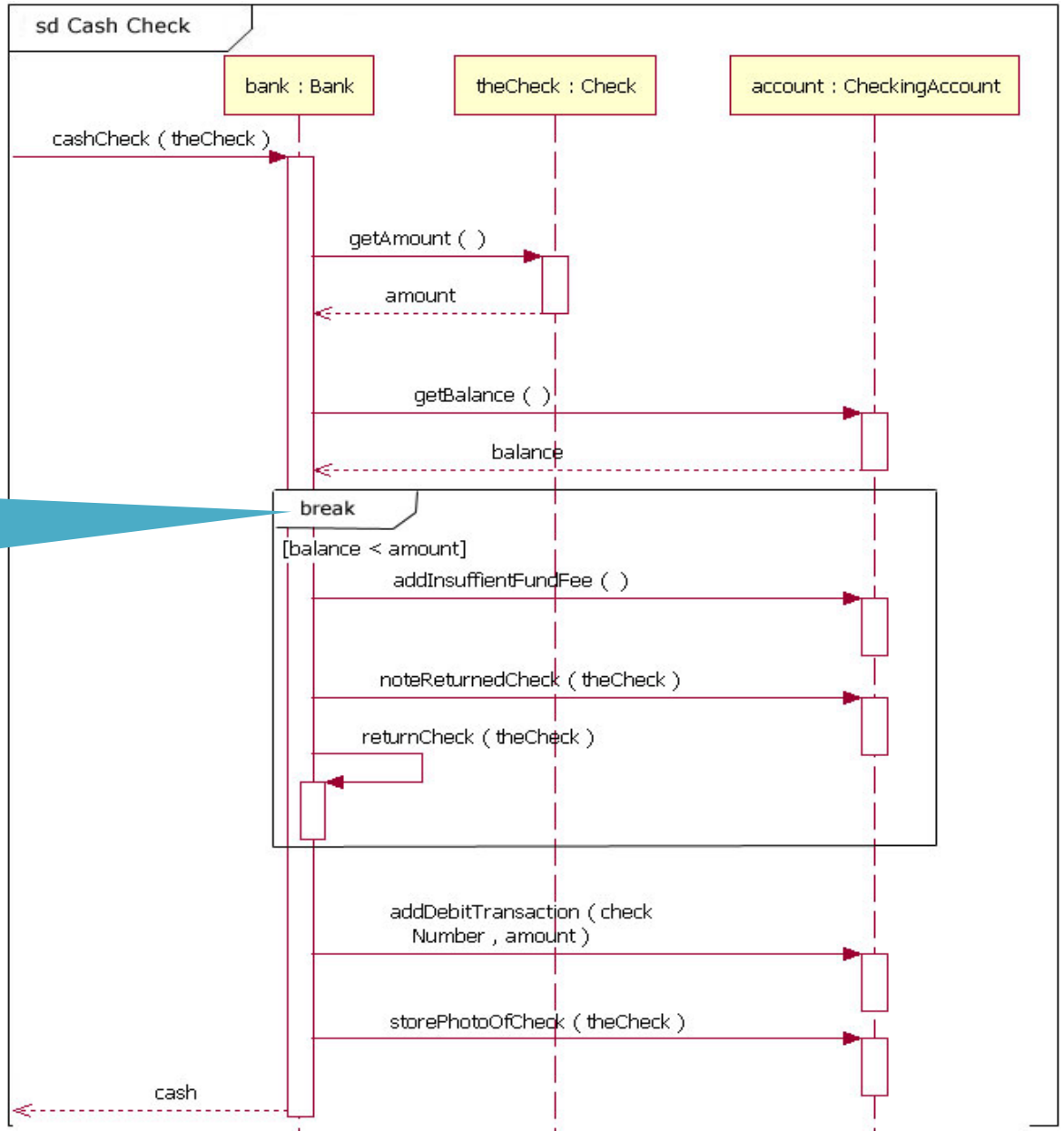
http://www.ibm.com/developerworks/rational/library/3101.html

Combined fragments: loop



Combined fragments: break

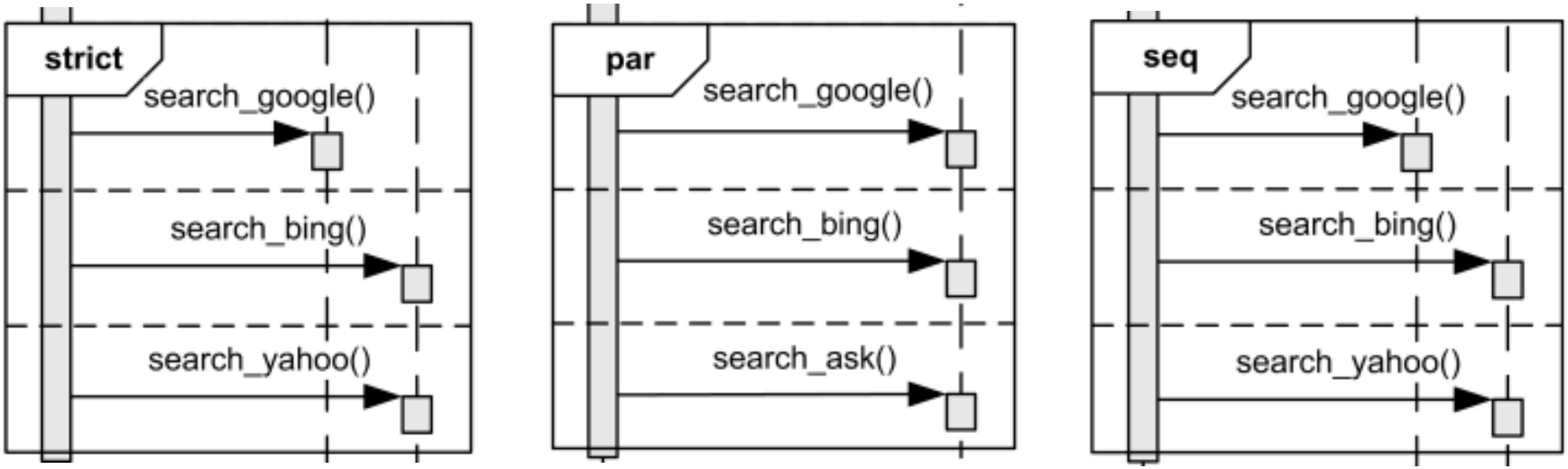
Break
(an alternative sequence processed instead of the rest of the diagram…)



Basic Interaction Operators

- **Branches and loops:**
  - alt, opt, break, loop
- **Concurrency and order:**
  - **seq, strict, par**
- **Filters and constraints:**
  - critical, neg, assert, consider, ignore

Strict, Parallel and Weak Sequencing

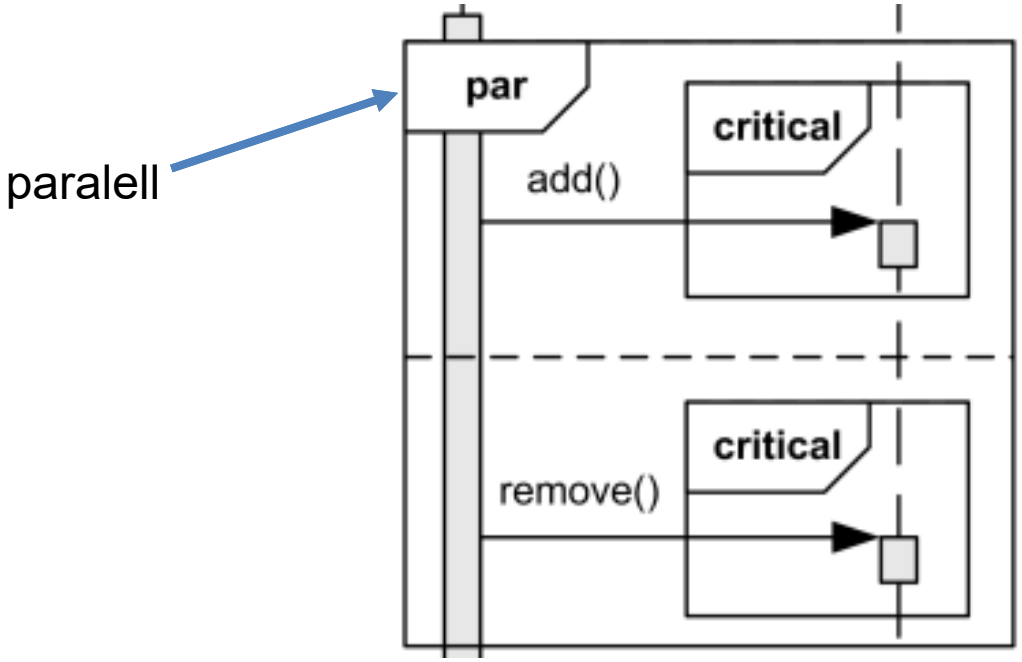


- **strict**: behaviour is sequentially executed exactly in the specified order.
- **par**: behaviour is executed in parallel.
- **seq**: behaviour is sequentially executed in any order.

Basic Interaction Operators

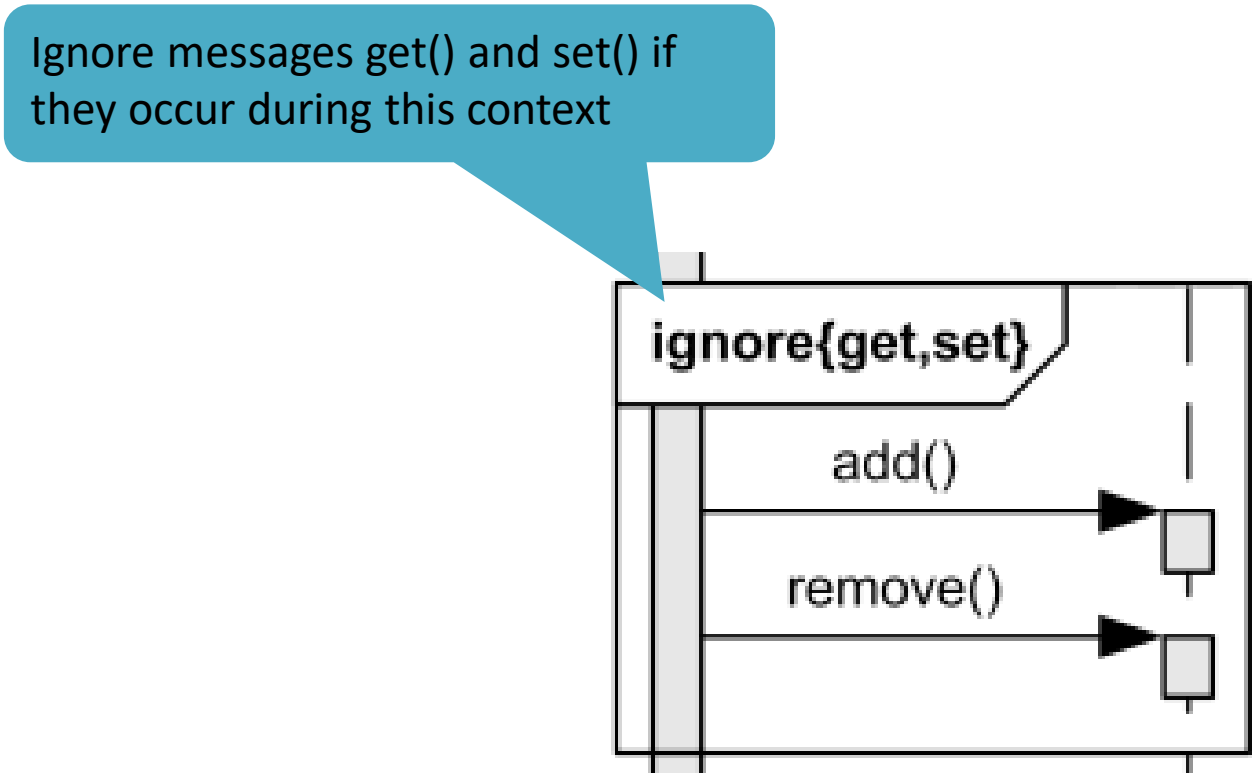
- **Branches and loops:**
  - alt, opt, break, loop
- **Concurrency and order:**
  - seq, strict, par
- **Filters and constraints:**
  - **critical, neg, assert, consider, ignore**

Combined fragments: critical



- A **critical** region is a region with traces that cannot be interleaved by other occurrence specifications.
- The region is treated **atomically** by the enclosing fragment and can not be interleaved (e.g. by a parallel operator).

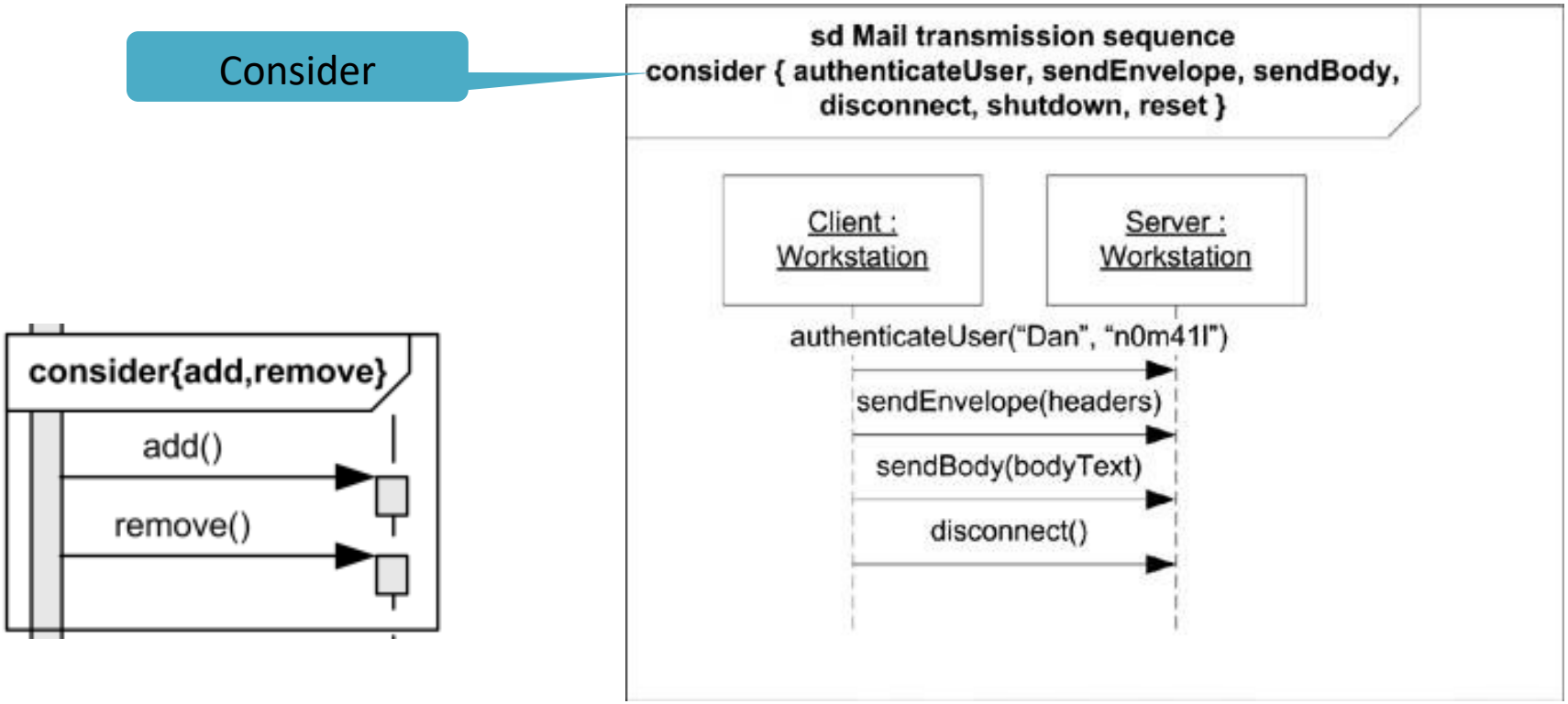
Combined fragments: ignore



- **Ignore** fragment declares a message or messages to be of no interest if it appears in the current context.

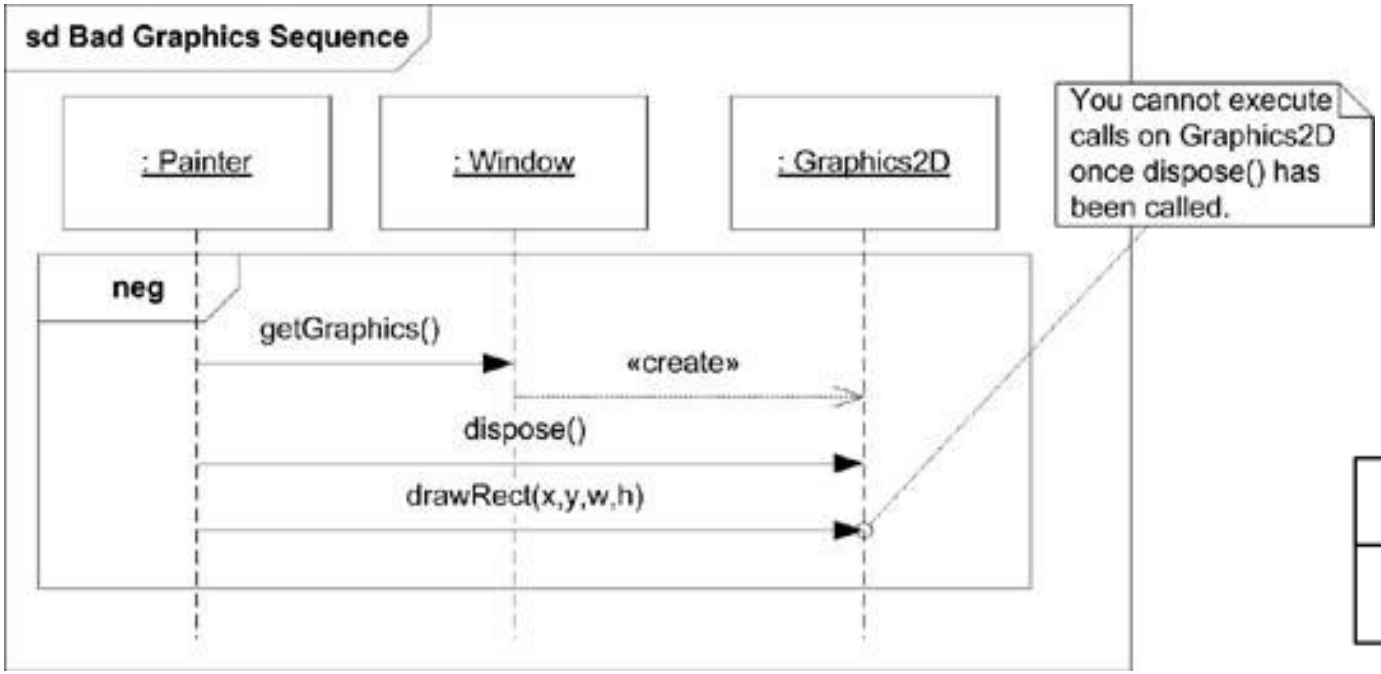
http://new.fengnet.com/book/UML/umlnut2-CHP-10-SECT-8.html

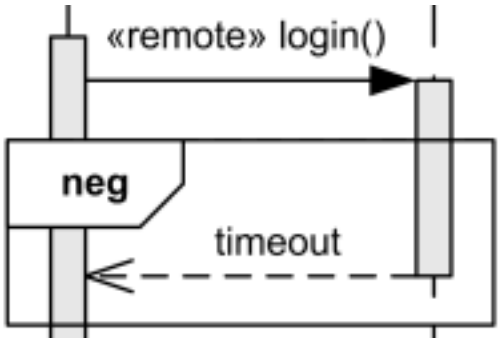
Combined fragments: consider



- **Consider** means any message not included in the fragment must be ignored
- **Consider** is the opposite of **Ignore**

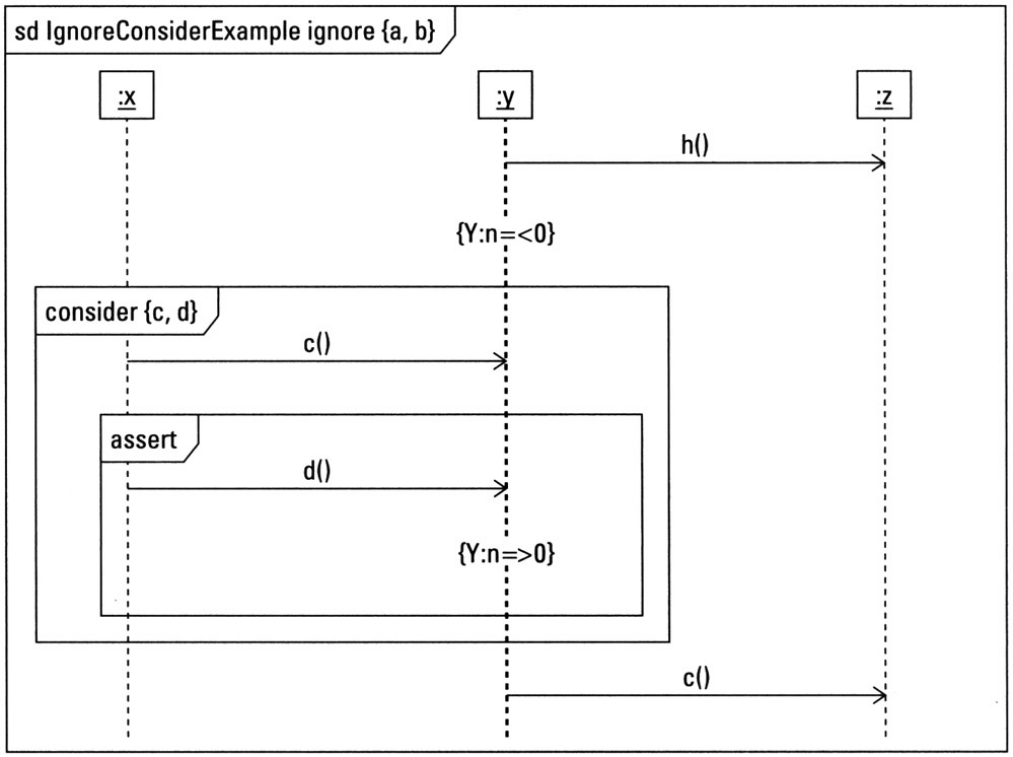
Combined fragments: negative

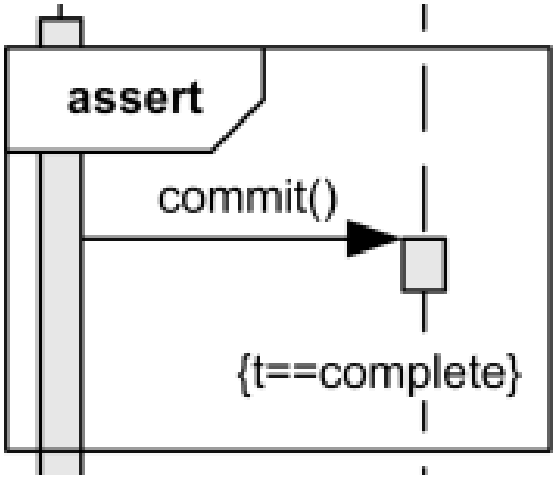




- **neg** encloses an invalid series of messages
- All interaction fragments that are different from the negative are considered **positive**, meaning that they describe traces that are valid and should be possible.
- Negative traces are the traces that occur only when the system has failed…

Combined fragments: assert



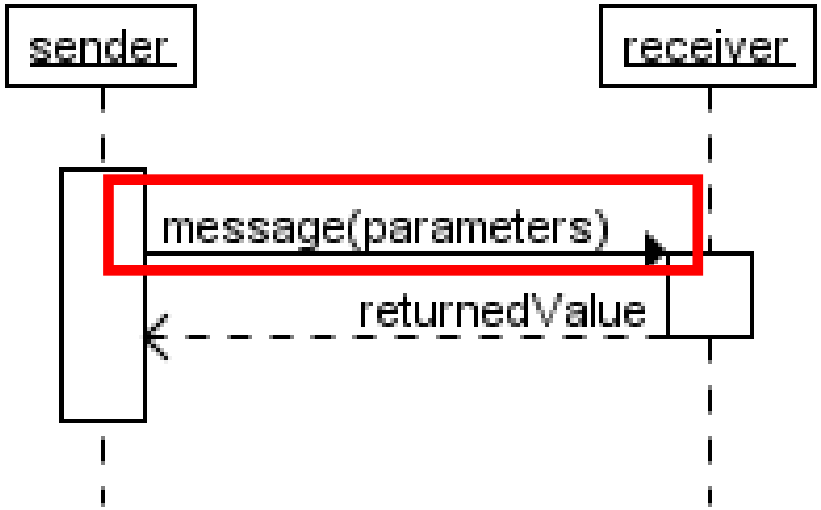


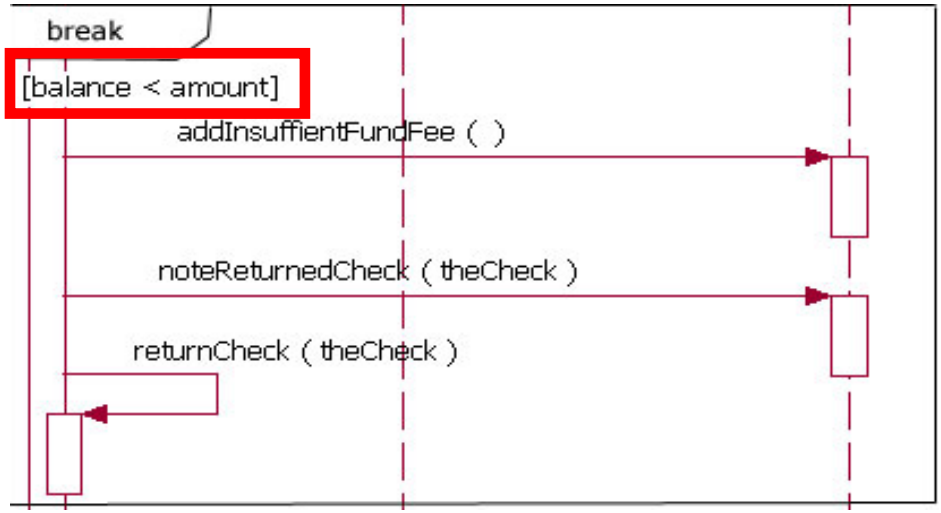
- **assert** identifies an operand as the only valid continuation of events in an interaction fragment.
- If any other messages occur they are considered to be invalid!

http://uml-dox.net/Wiley-UML.Bible/6081/DDU0074.html

VERY IMPORTANT

- **Sequence and communication diagrams** define interactions based on the **METHODS OF ENTITIES.** Thus, the messages must be methods that are specified in classes or blocks (traceability).
- All **PARAMETERS** (e.g. used in combined fragments) must also be valid in a given context.
- This Rigor is relevant particularly at Design phase models

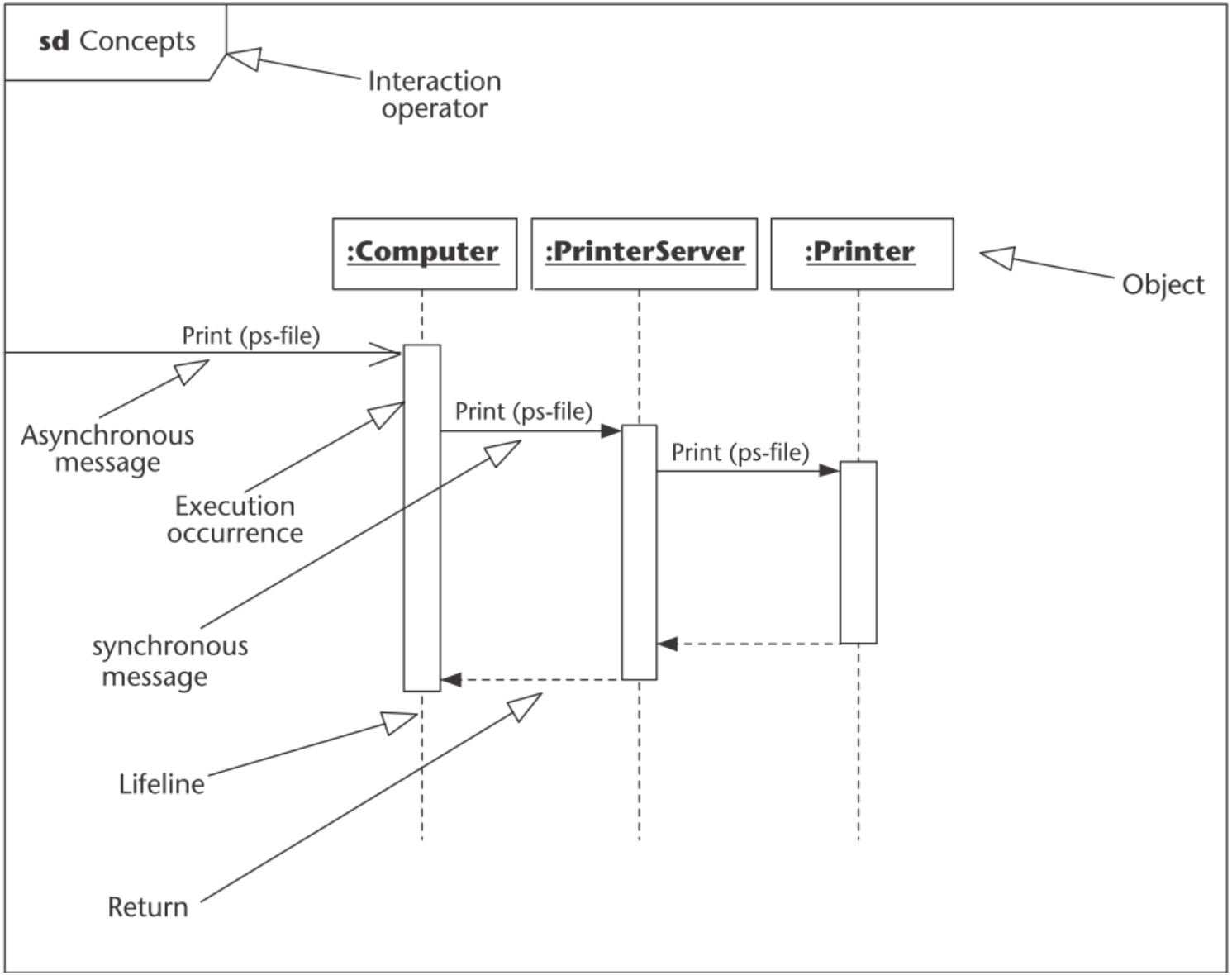




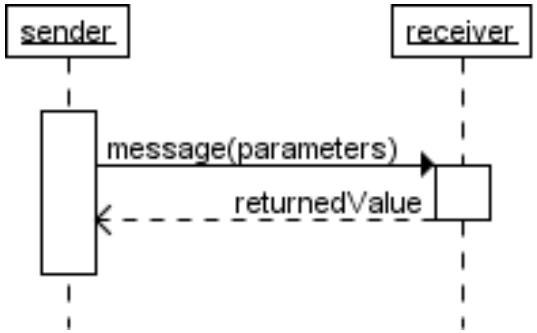
Note: Sequence and communication diagrams can also describe interactions among instances of components instead of classes/blocks.

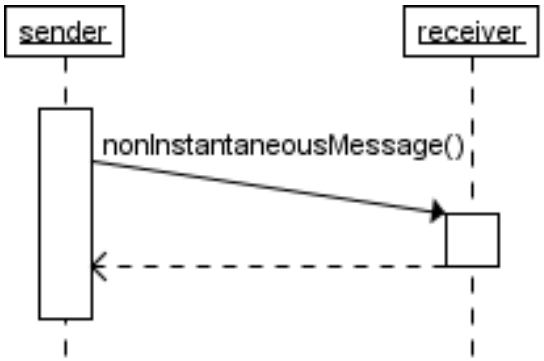
Examples from http://www.tracemodeler.com/articles/a_quick_introduction_to_uml_sequence_diagrams/

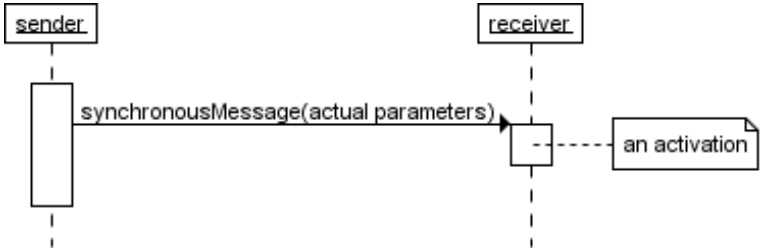
Sequence Diagram: basic concepts

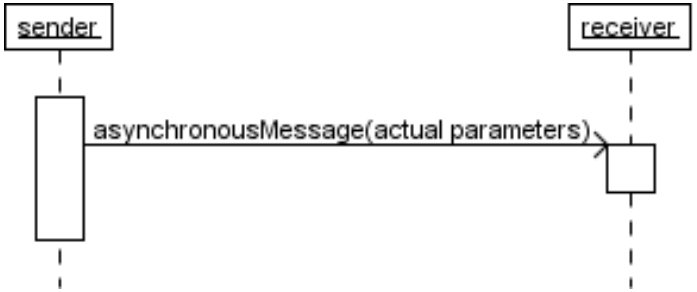


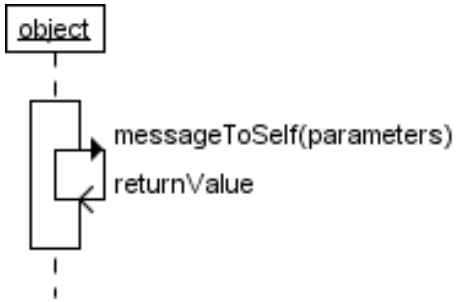
Examples of Sequence Diagram





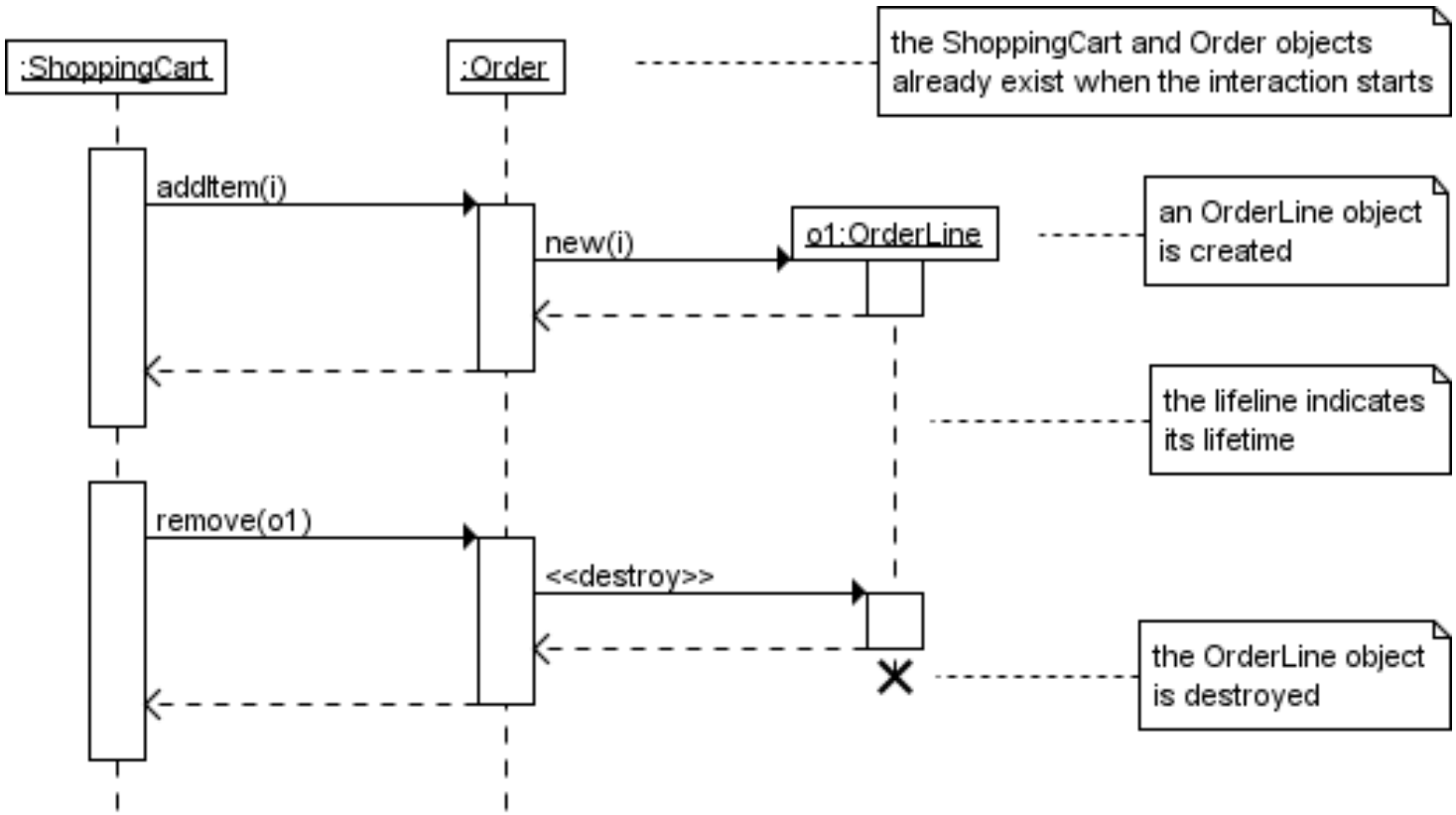




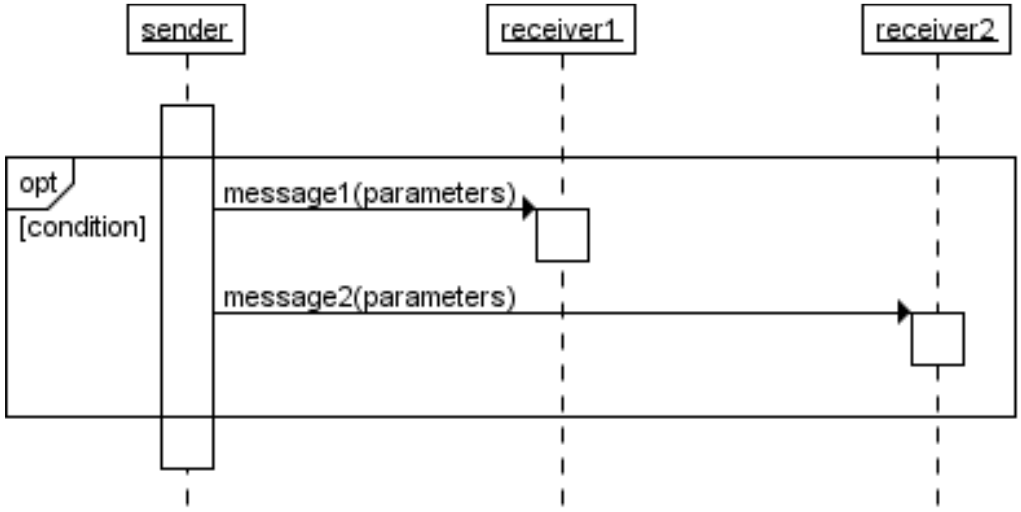


a message to self
and its return value

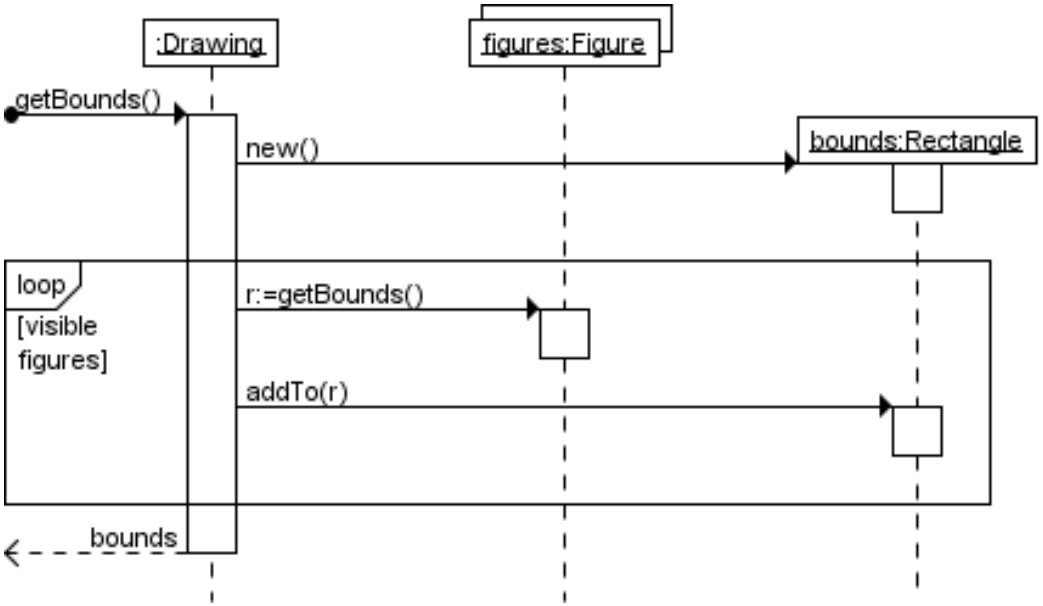
Sequence Diagram: Summary



Sequence Diagram: Summary

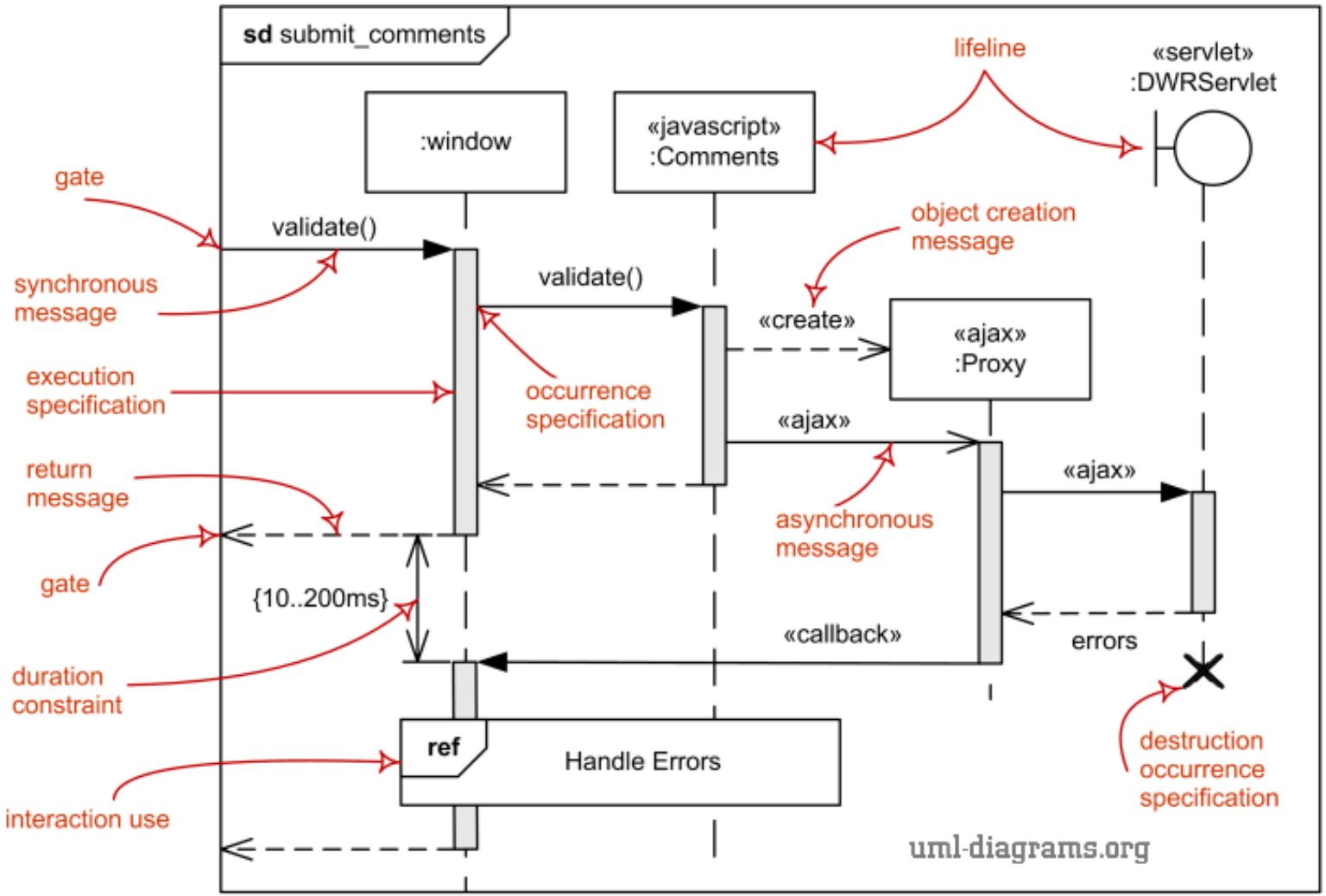


If condition is met,
both messages are sent



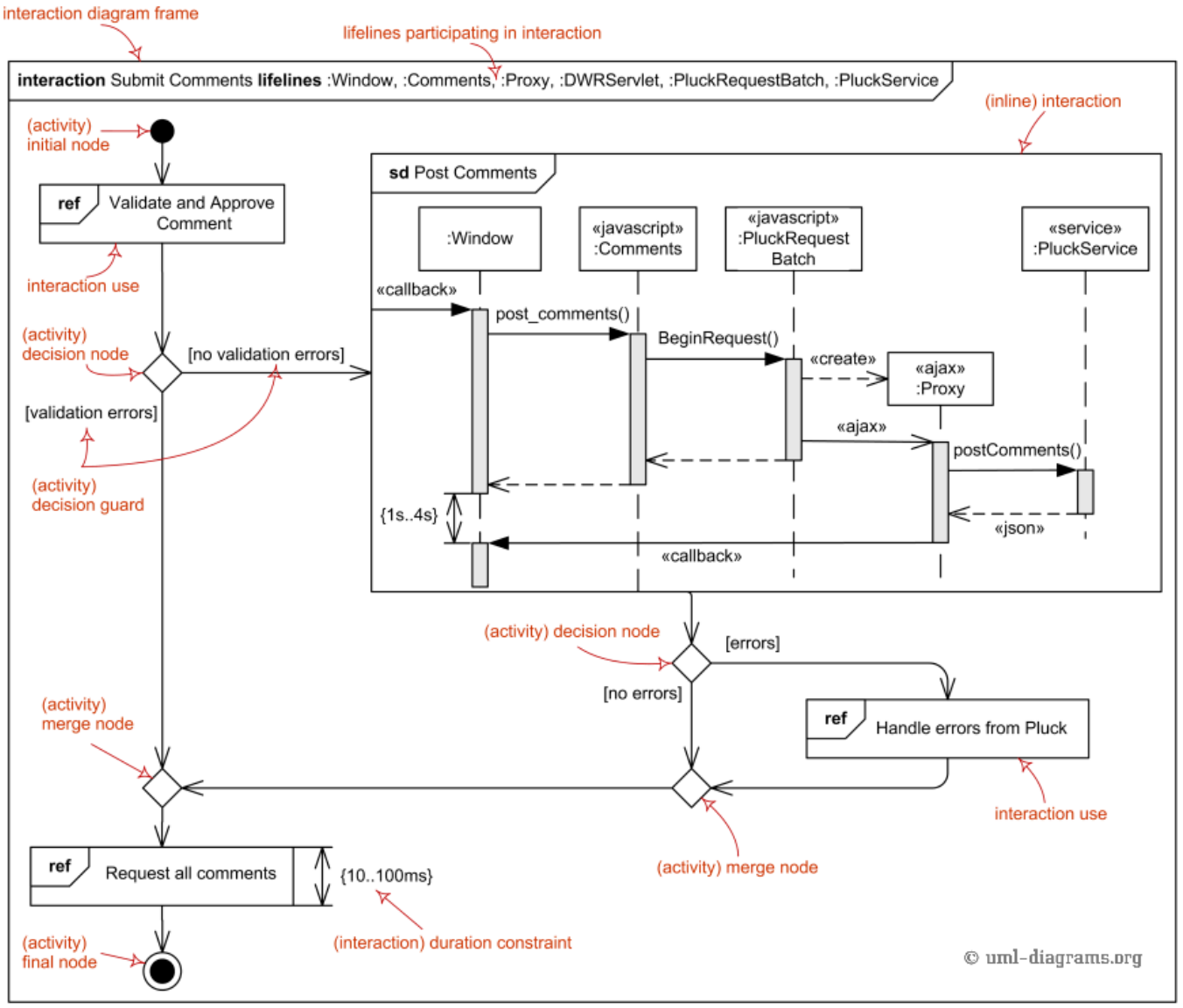
The iteration only applies to the
visible figures of the drawing

Sequence Diagram Overview



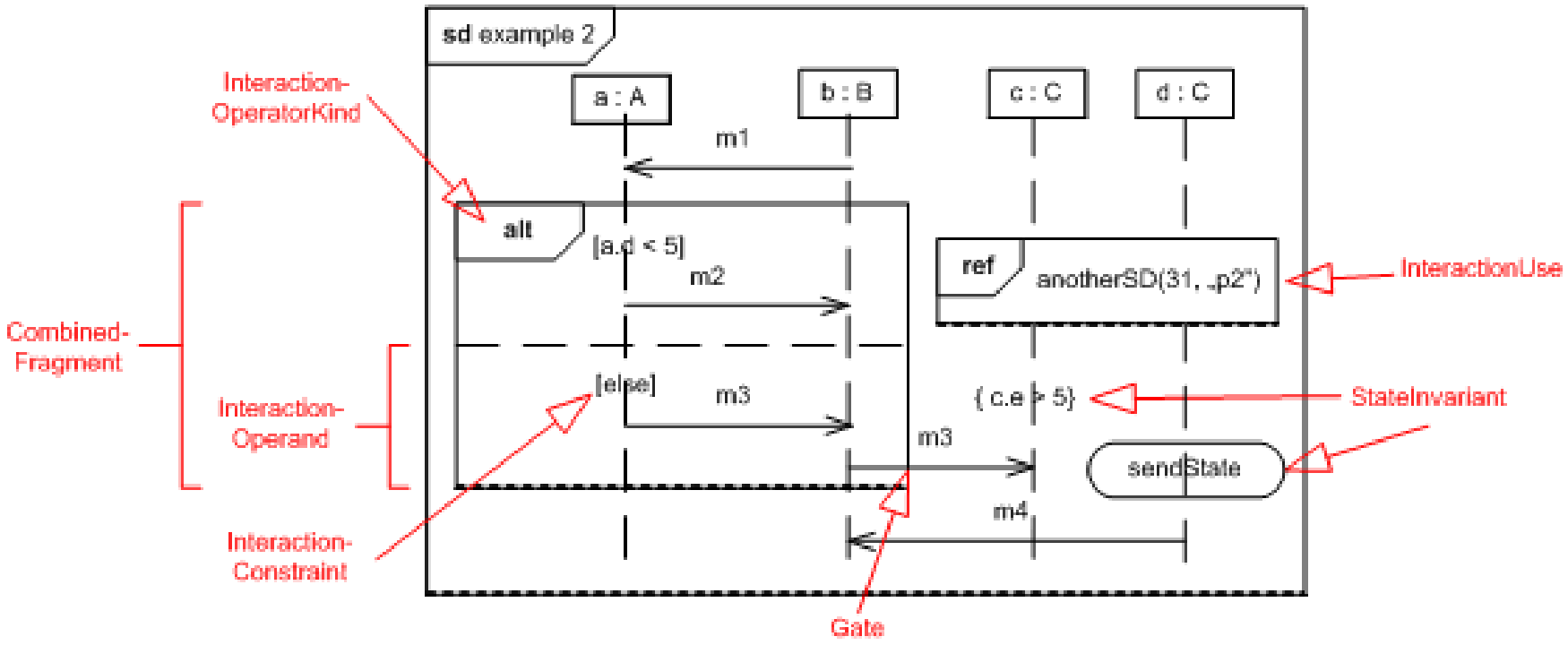
https://www.uml-diagrams.org/sequence-diagrams.html

Sequence Diagram Overview

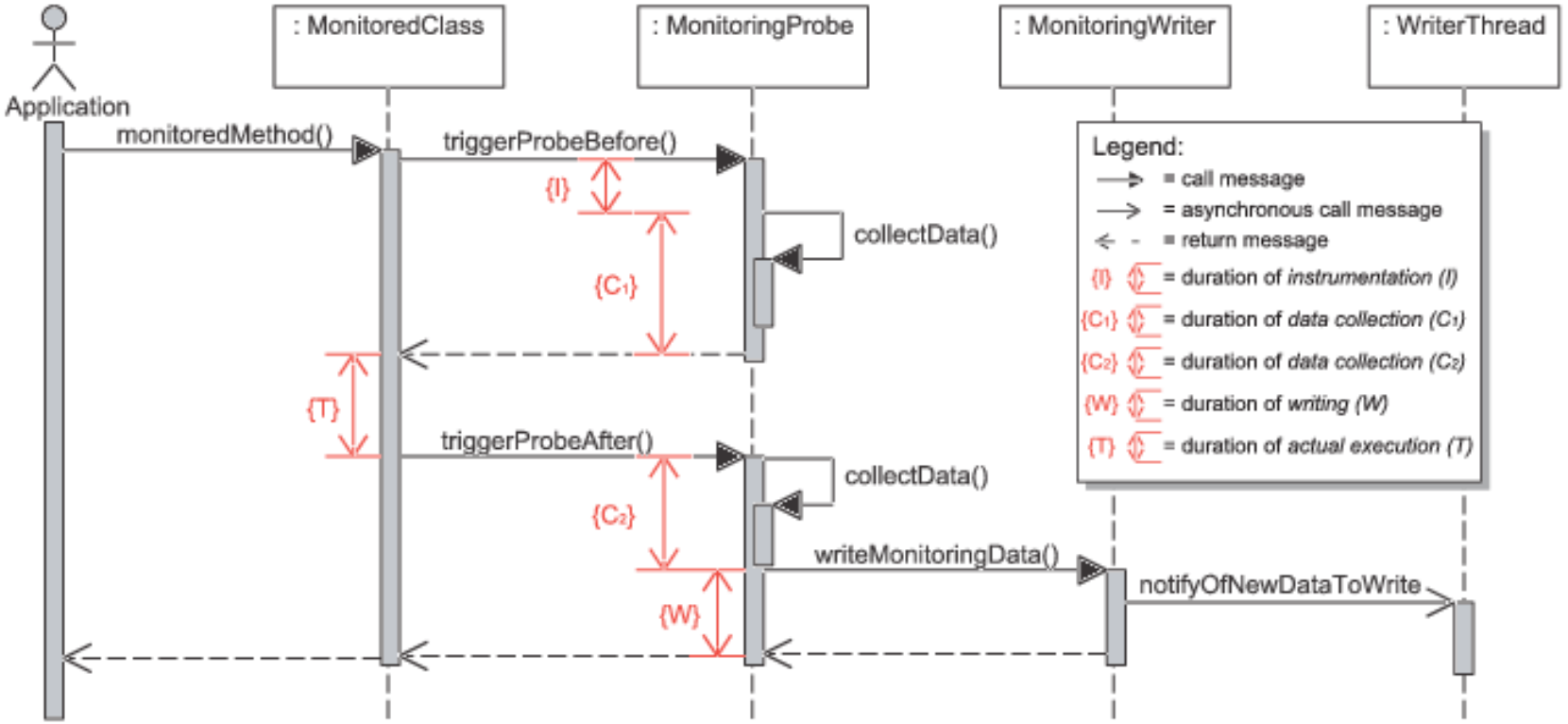


https://www.uml-diagrams.org/notation/uml-interaction-overview-diagram-elements.png

Sequence Diagram Overview

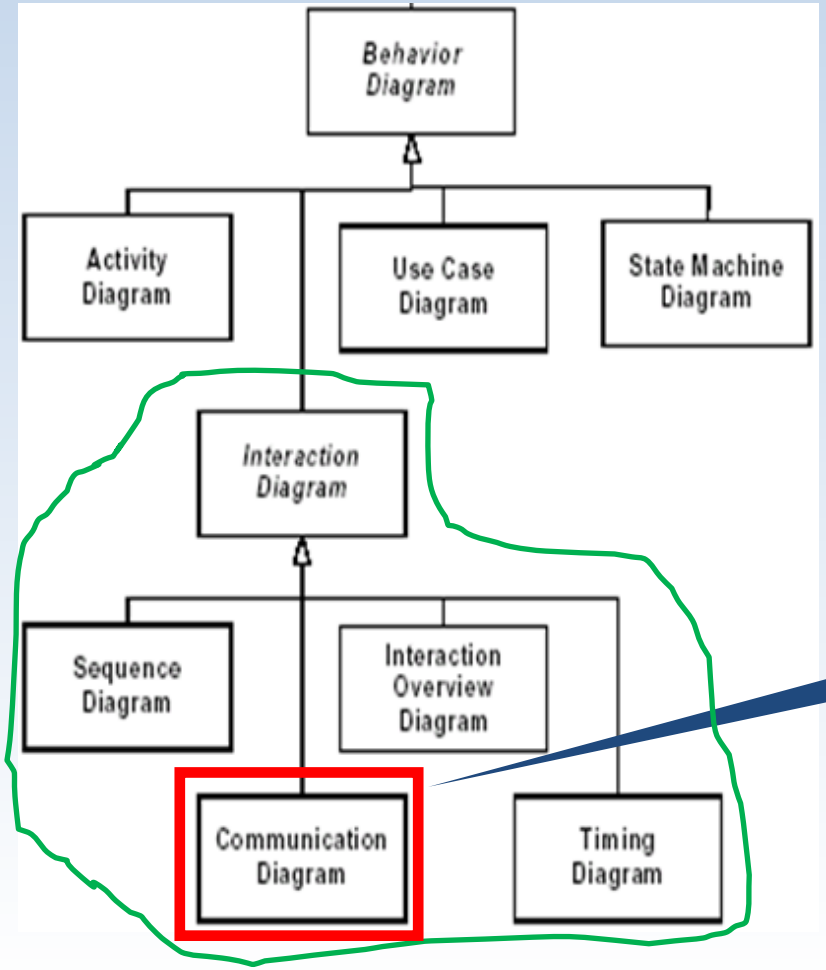


Sequence Diagram Overview



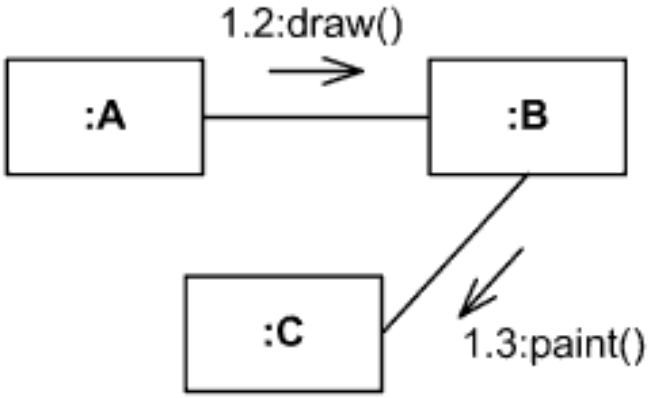
https://www.researchgate.net/publication/262275694_A_Comparison_of_the_Influence_of_Different_Multi-core_Processors_on_the_Runtime_Overhead_for_Application-Level_Monitoring/figures?lo=1

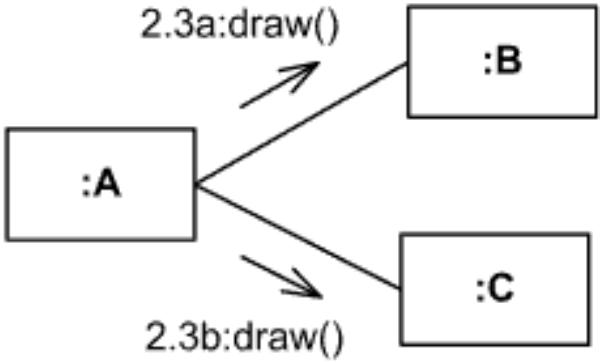
Message-based Behaviour



Communication Diagram (Message-based behavior)

Communication Diagrams

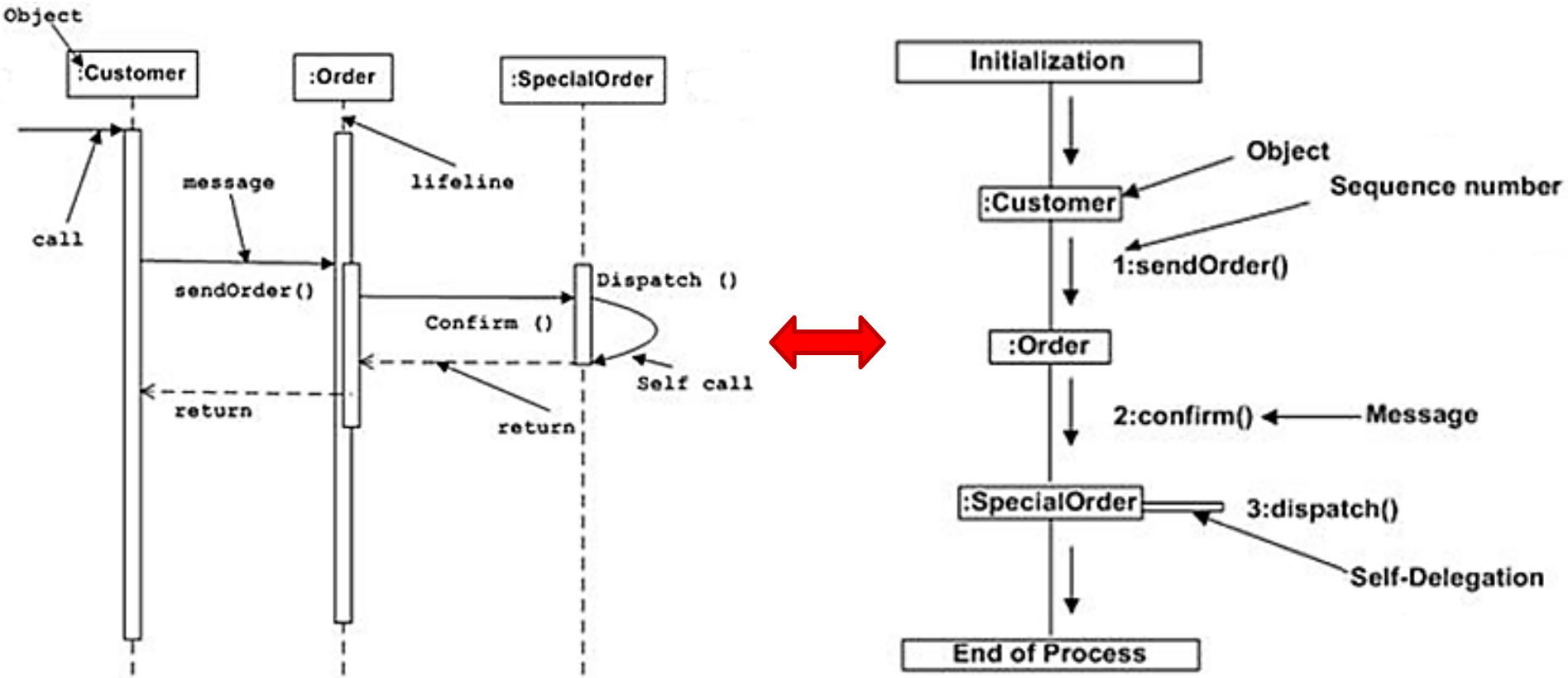




Communication diagrams *correspond* to simple Sequence diagrams without structuring mechanisms (such as Combined Fragments).

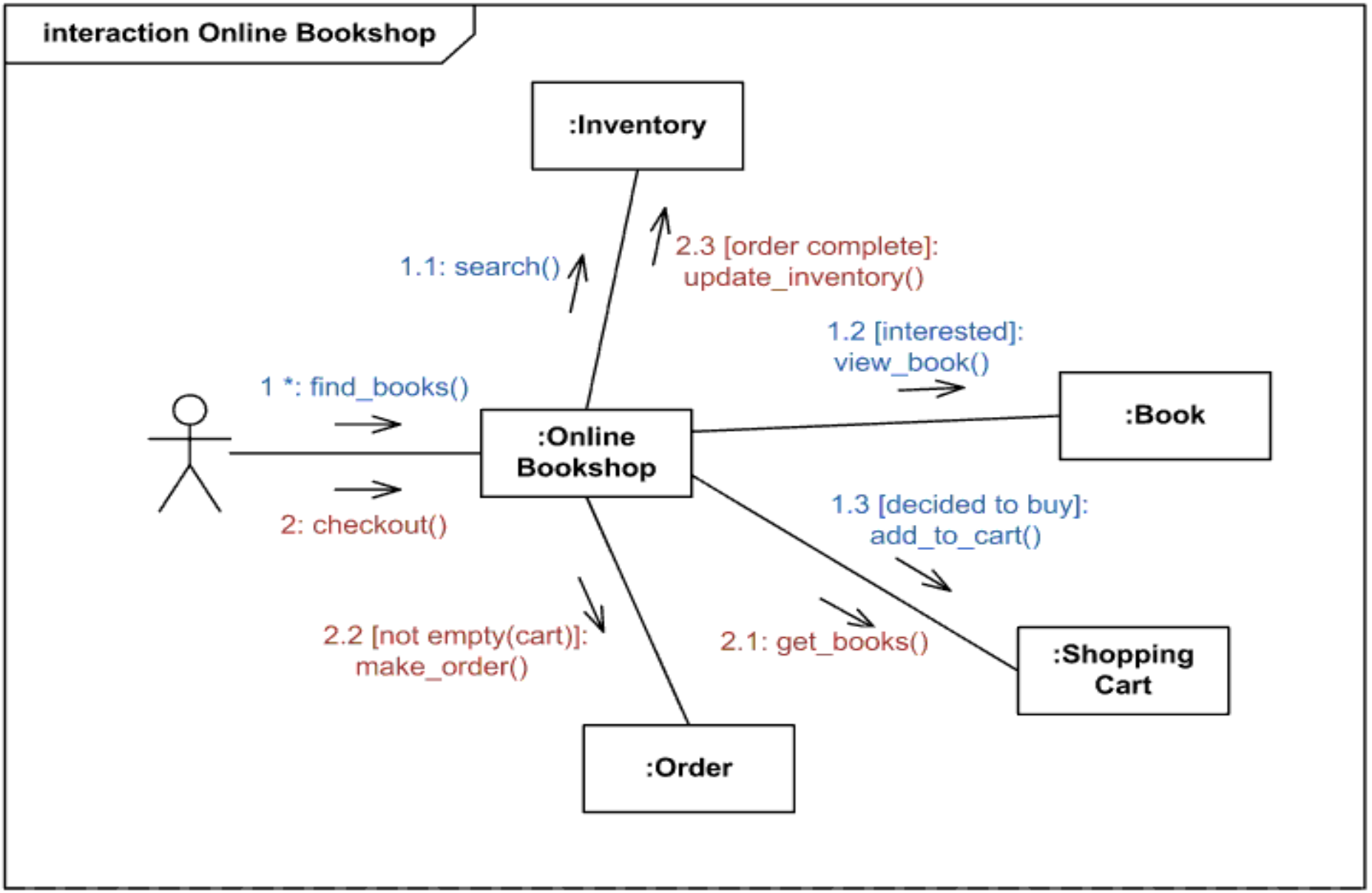
correspond = can be converted to/from or replaced by

Equivalence of Sequence and Communication

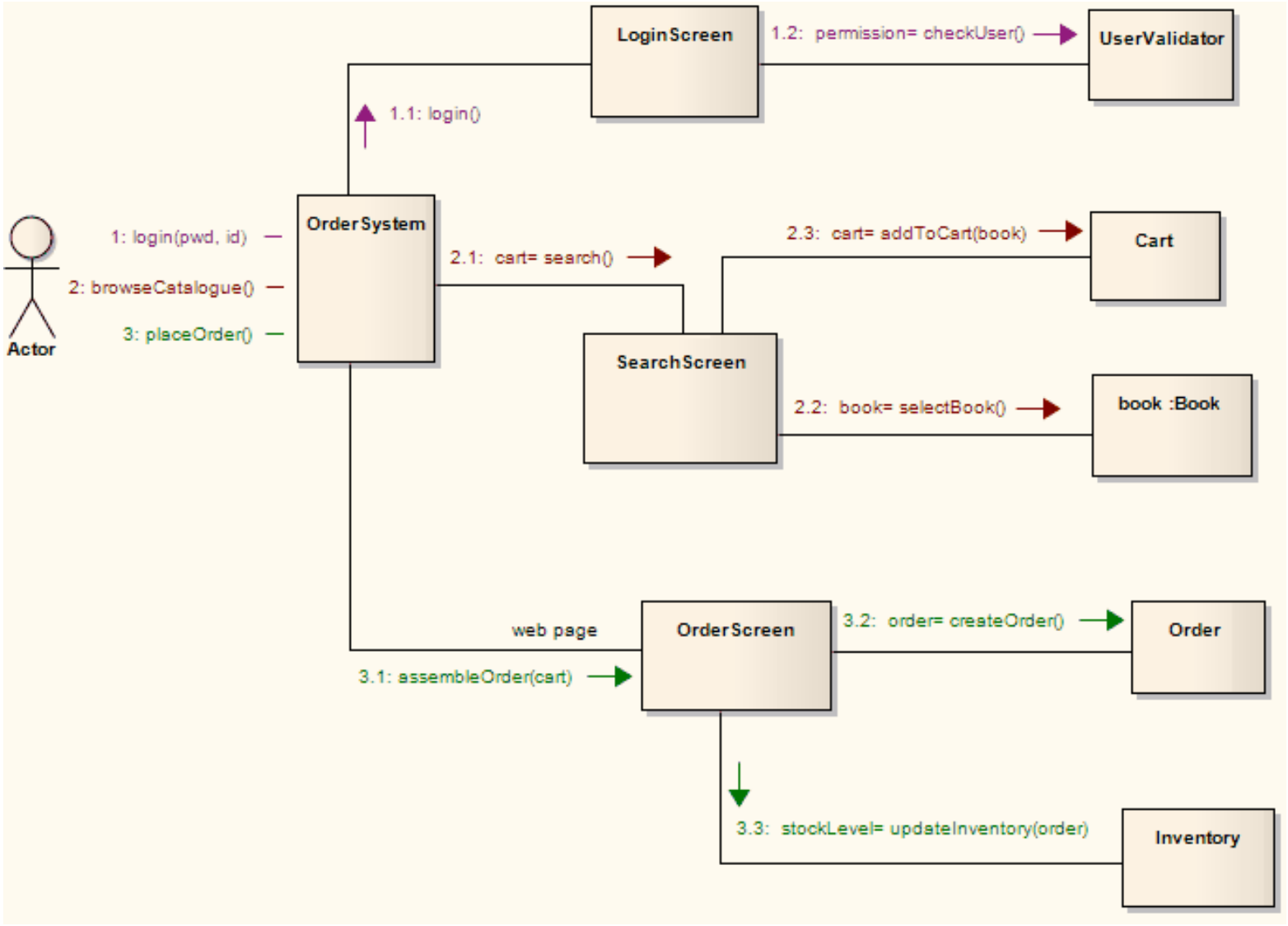


http://www.tutorialspoint.com/uml/uml_interaction_diagram.htm

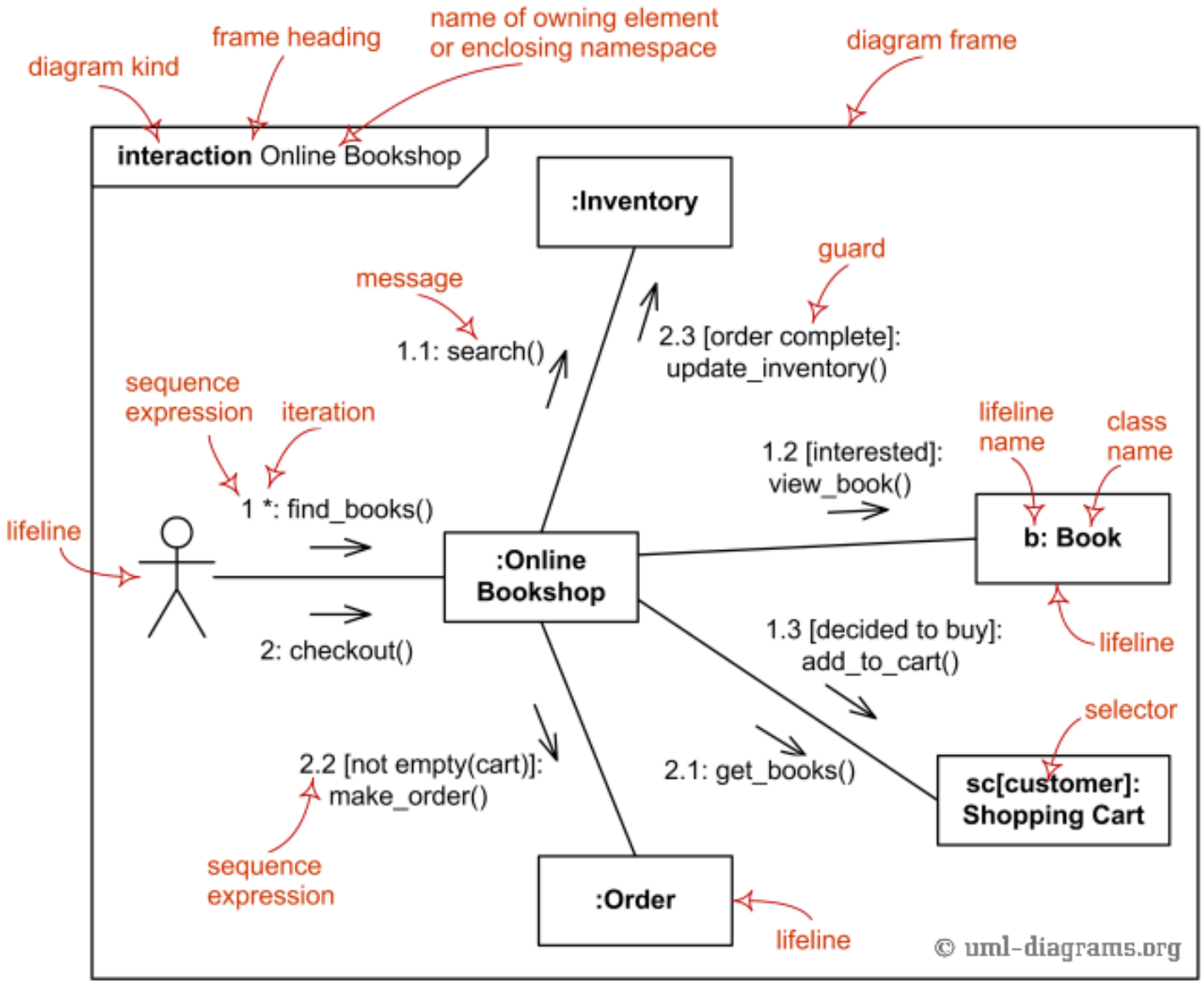
Communication Diagrams



Communication Diagrams

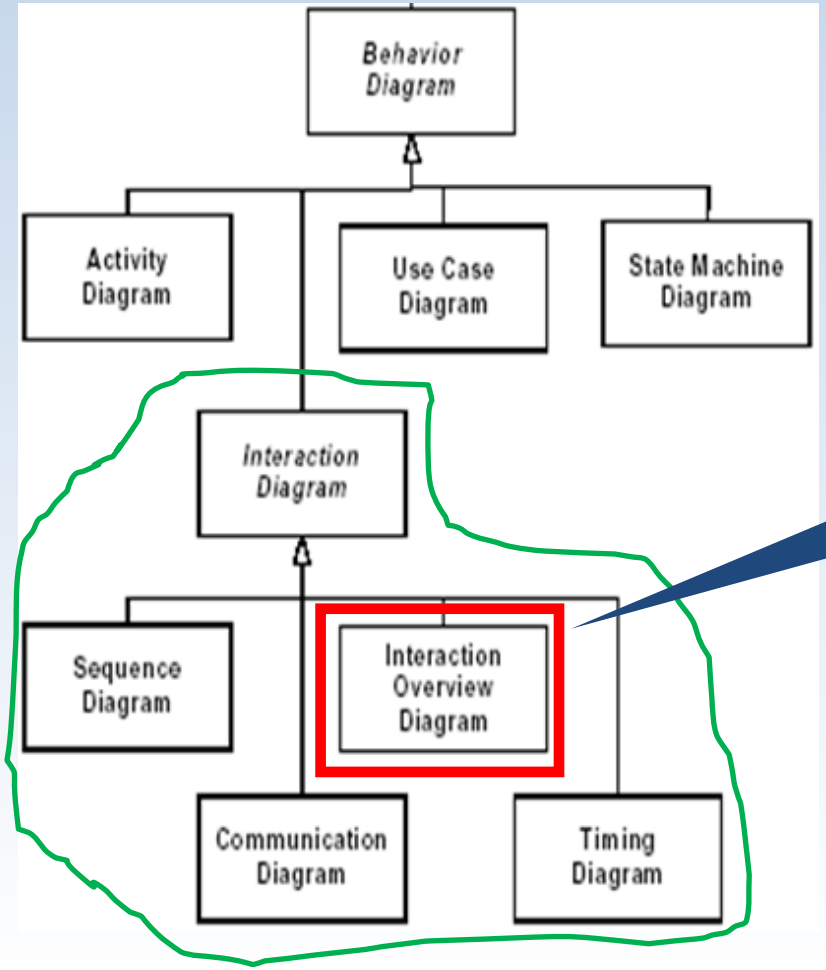


Communication Diagram Overview



https://www.uml-diagrams.org/communication-diagrams.html

Message-based Behaviour

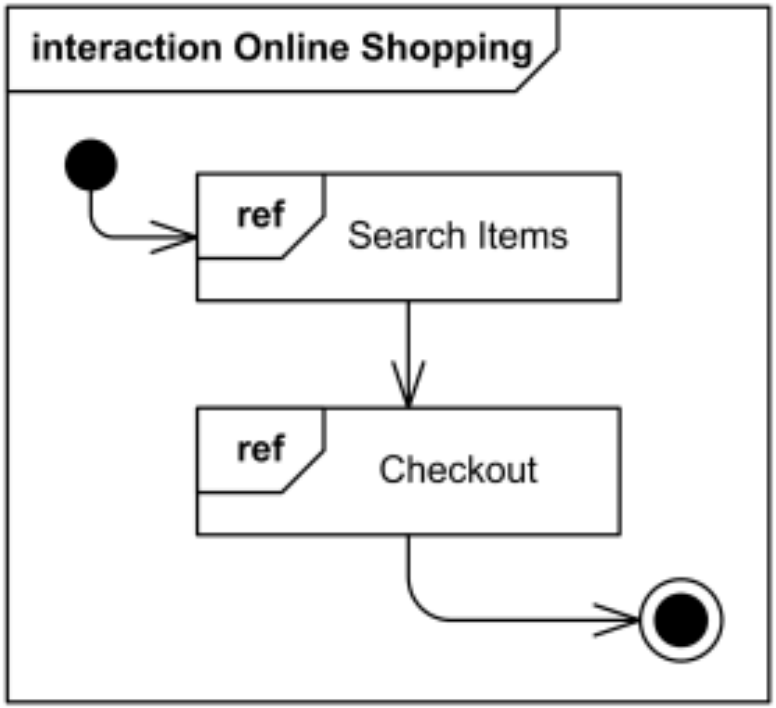


Back to Message-based behavior

Interaction Overview

Interaction overview diagrams provide an overview of the control flow where each node is an interaction

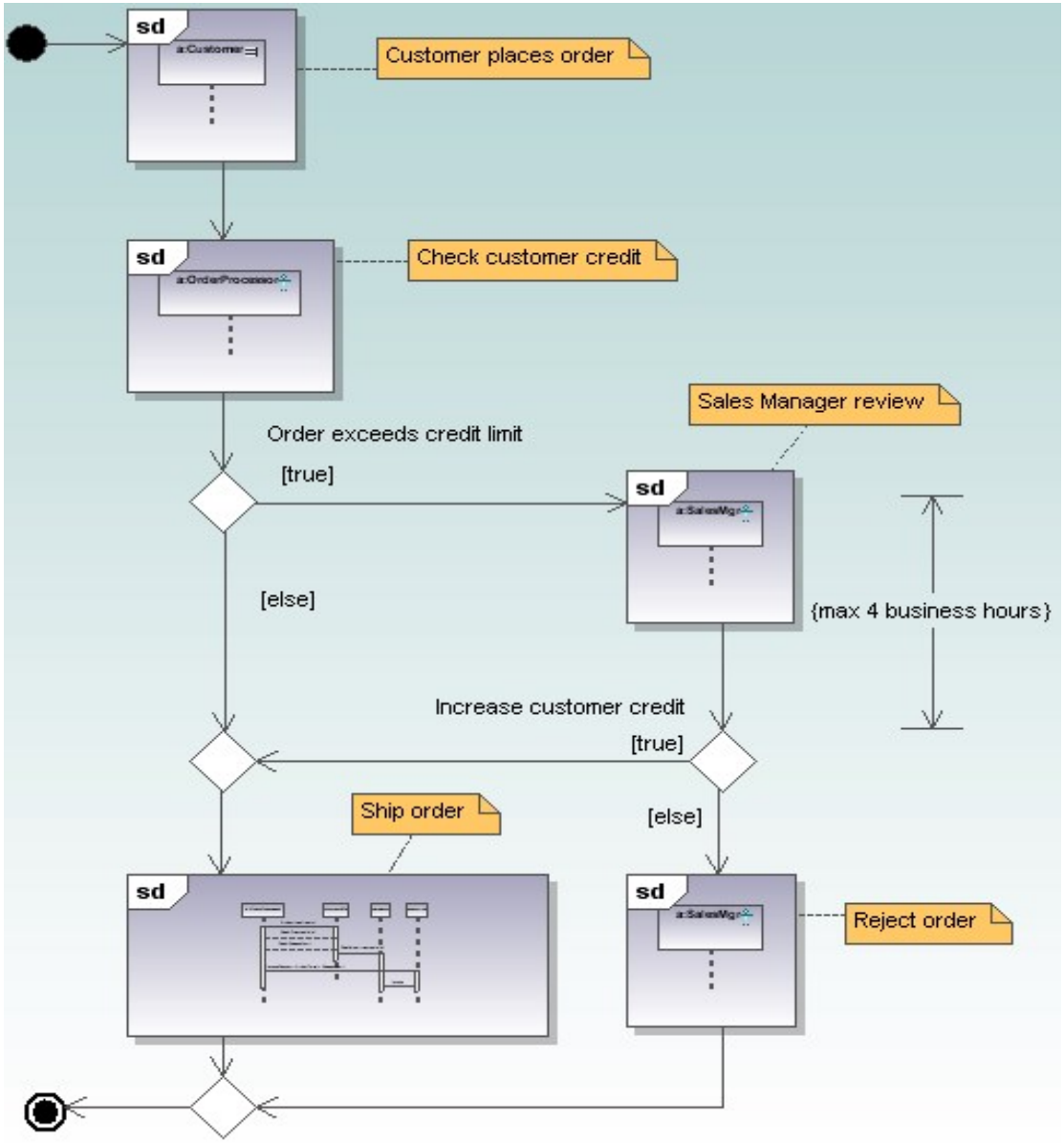
It has similarities with an activity diagram but they are not equivalent!



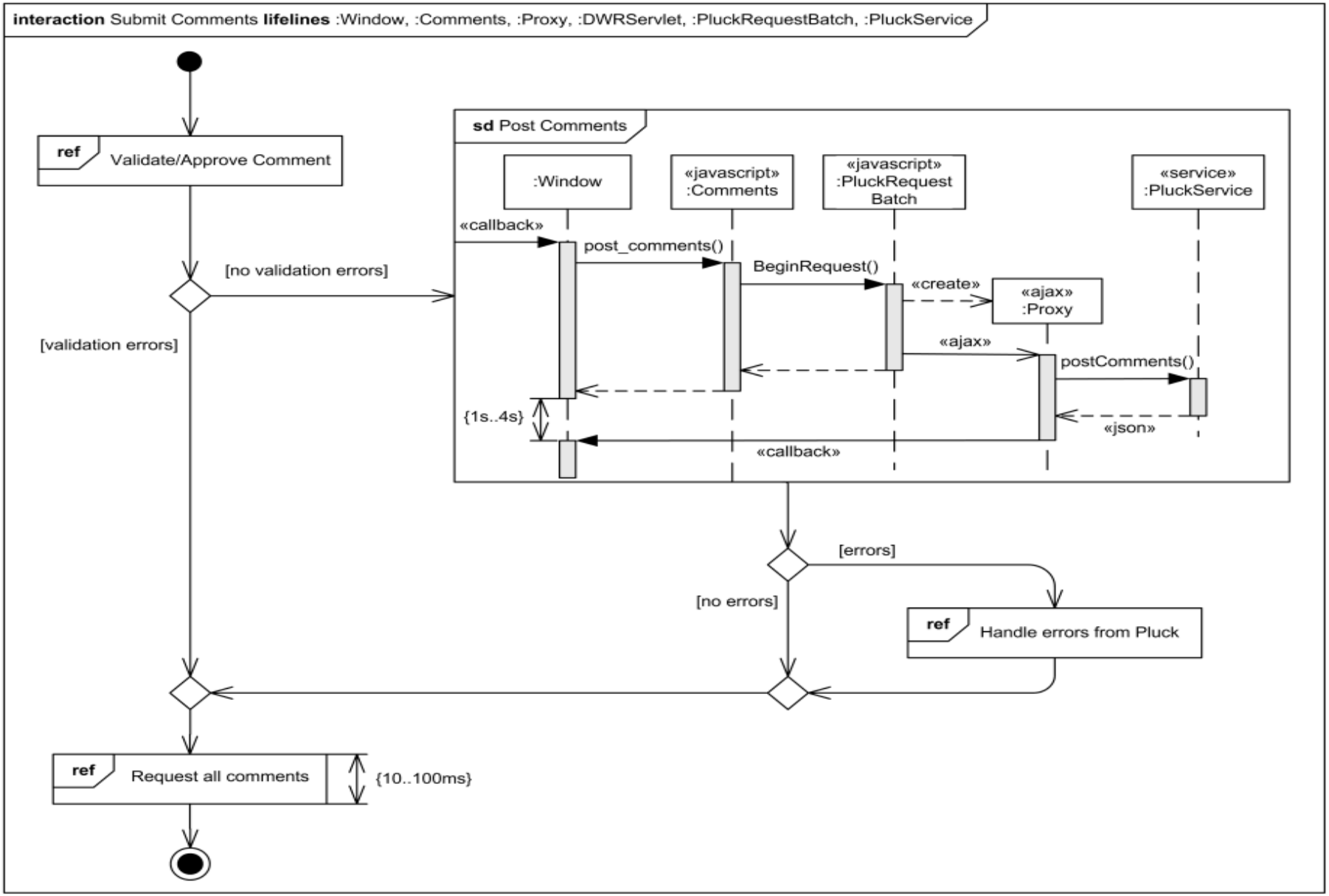
Interaction Overview

- Combine elements from **activity diagrams** and **sequence diagrams** to show long or complex sequences of action.

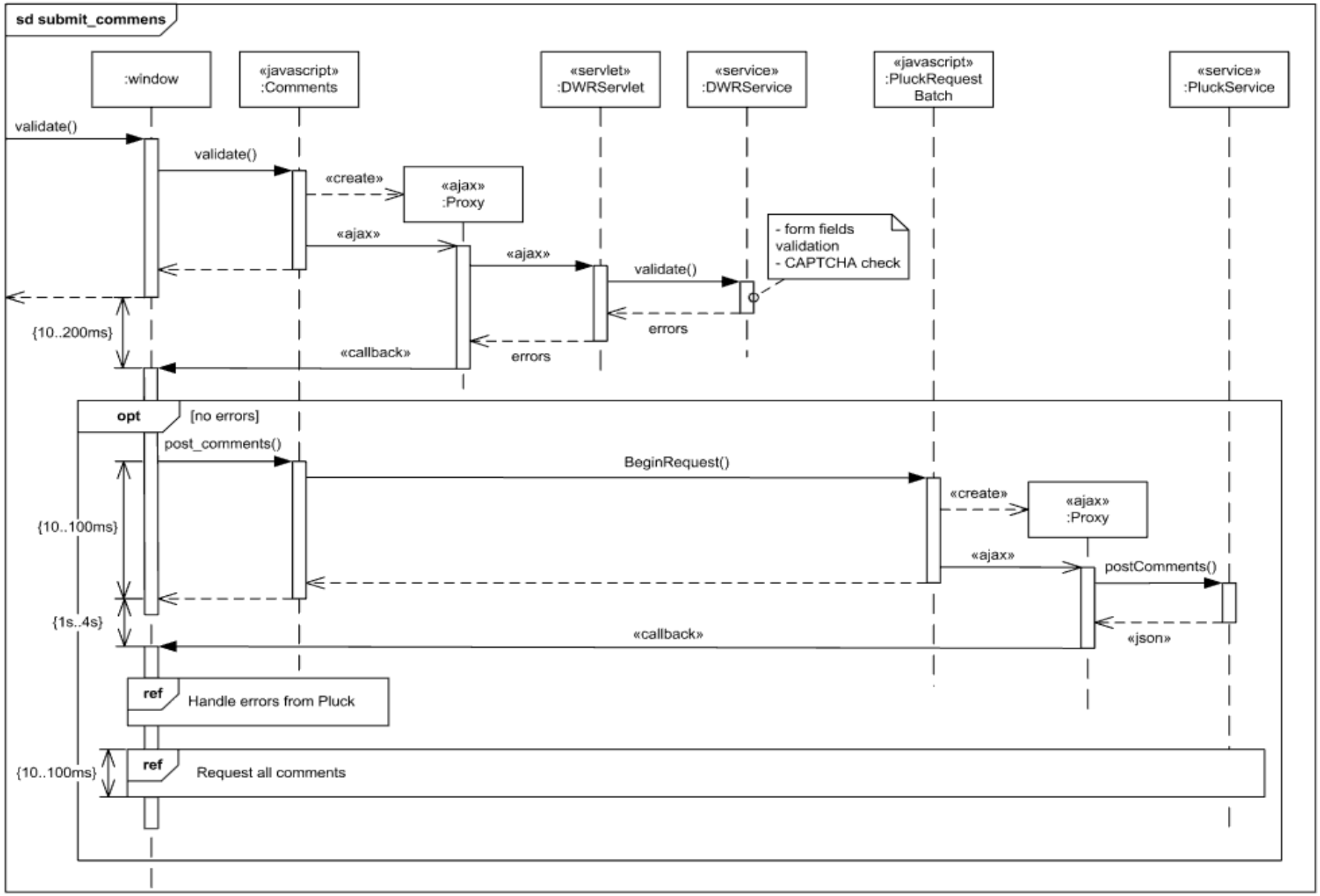
- Each node is a reference to an existing diagram.
  - *sd* - sequence diagrams
  - *cd* - communication diagrams
  - *td* - timing diagrams
  - *iod* - other interaction overview diagrams



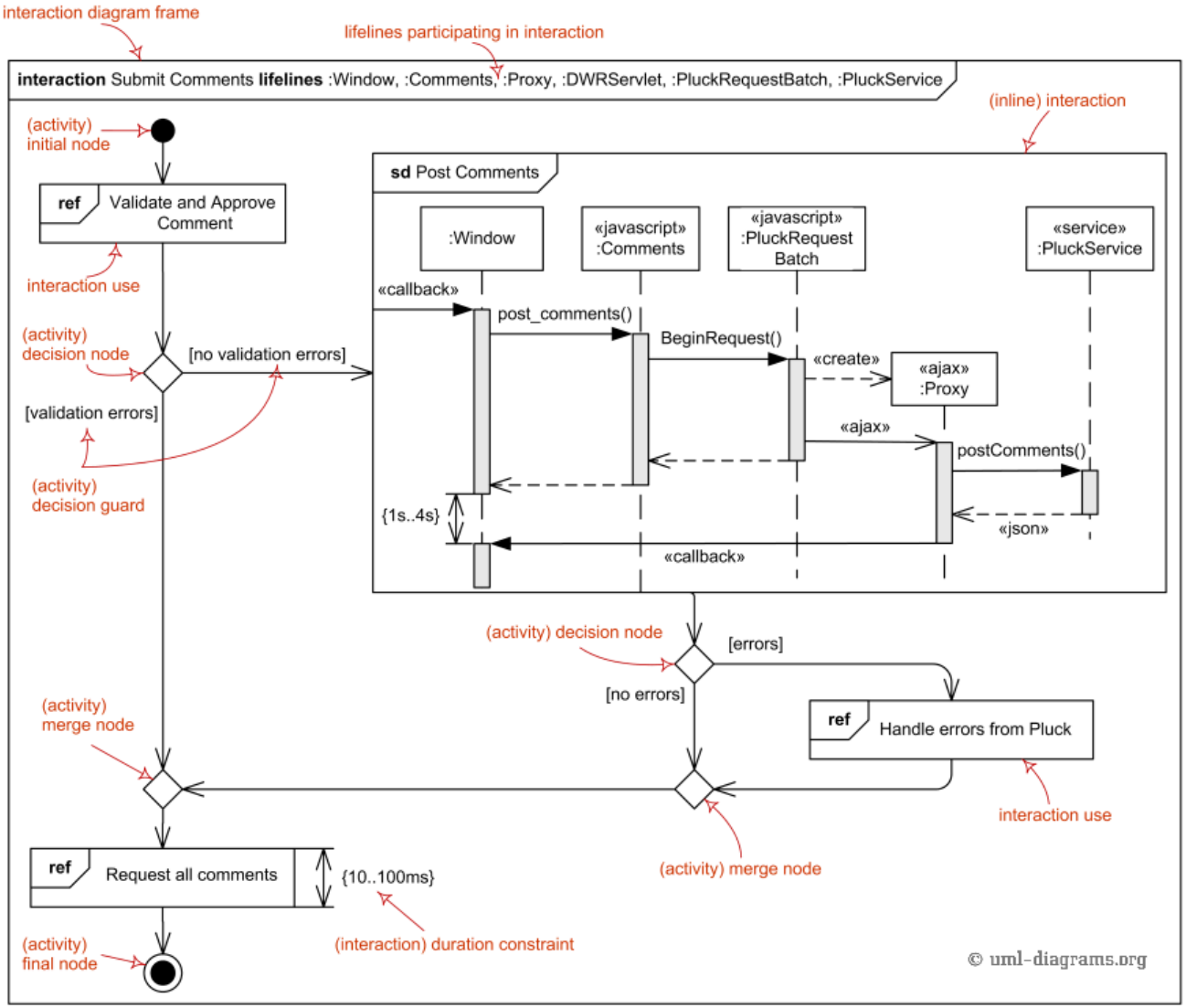
Interaction Overview



Example…

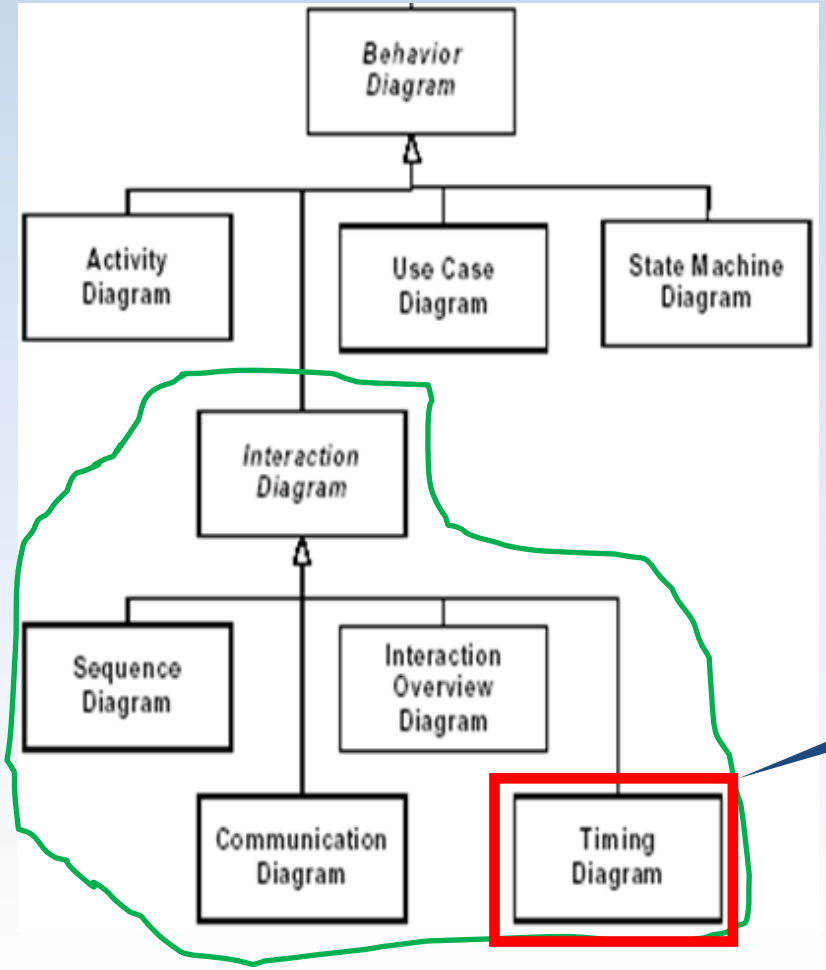

sd submit_commens
:window
«javascript»
:Comments
«servlet»
:DWRServlet
«service»
:DWRService
«javascript»
:PluckRequest
Batch
«service»
:PluckService
validate()
validate()
«create»
«ajax»
:Proxy
«ajax»
«ajax»
validate()
- form fields
validation
- CAPTCHA check
errors
{10..200ms}
«callback»
errors
opt
[no errors]
post_comments()
BeginRequest()
«create»
«ajax»
:Proxy
{10..100ms}
«ajax»
postComments()
{1s..4s}
«callback»
«json»
ref
Handle errors from Pluck
{10..100ms}
ref
Request all comments

Interaction Overview diagram… overview



https://www.uml-diagrams.org/interaction-overview-diagrams.html

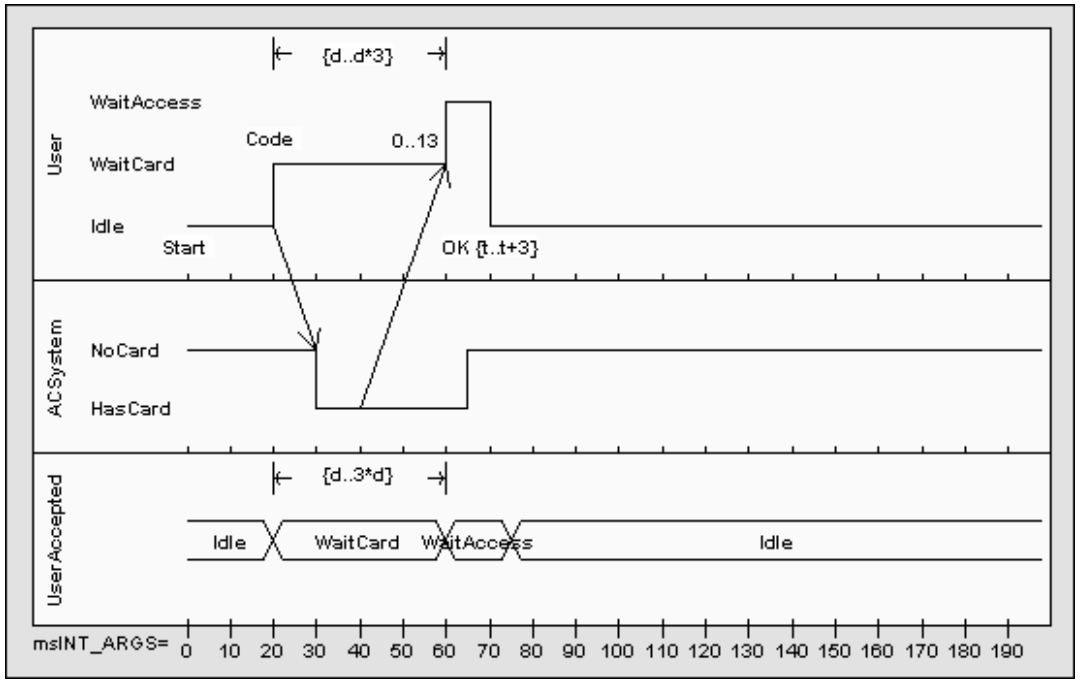
Message-based Behaviour

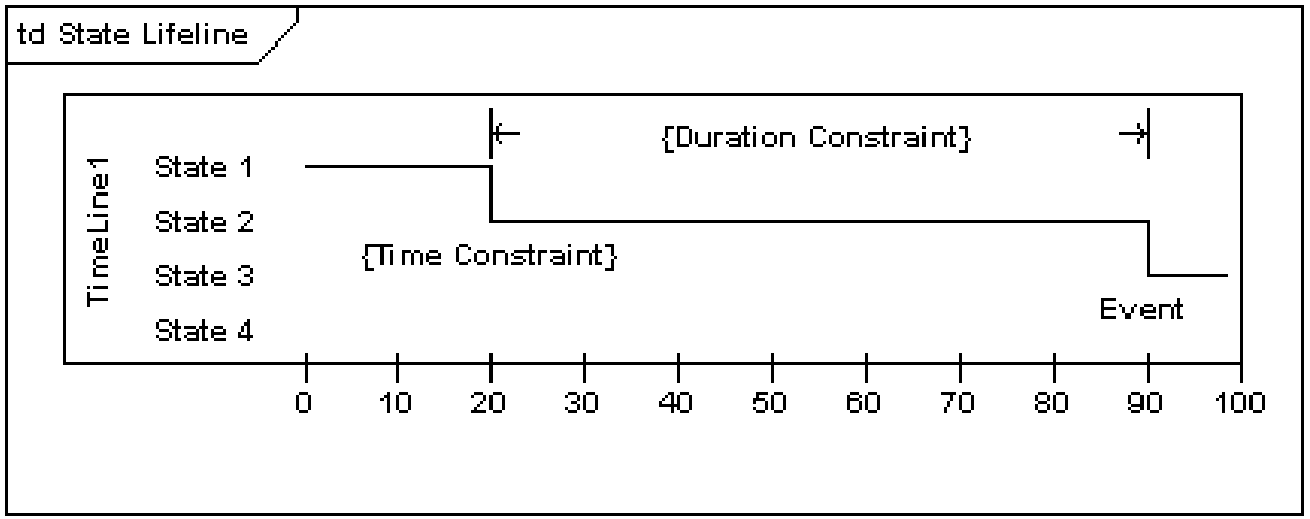


Timing diagram (Message-based behavior)

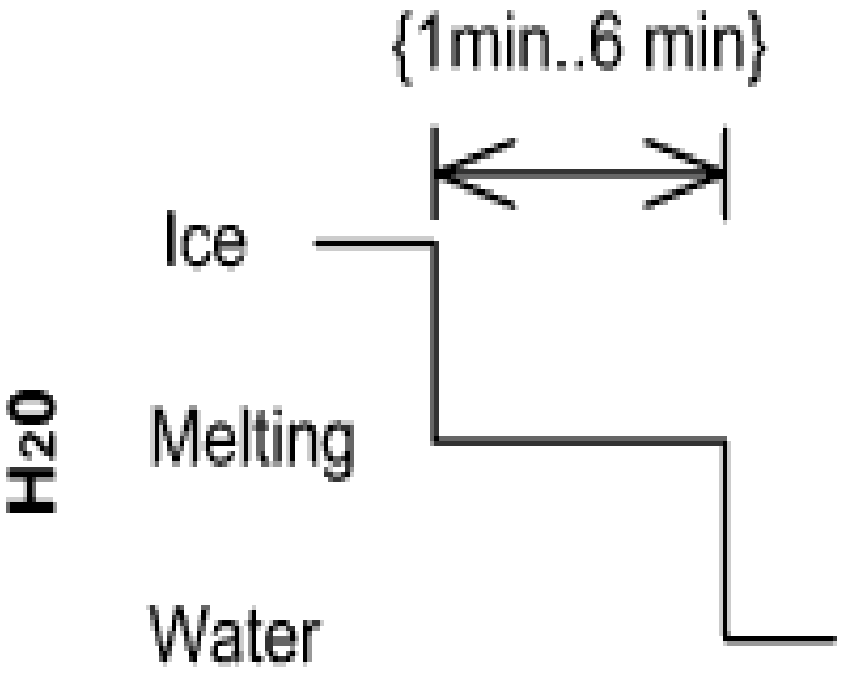
Timing Diagrams (UML)

- Describes how one or more objects change state over time.
- Can represent the same information as a cd, sd, smd, but detail temporal constraints (e.g. delay, duration, frequency).

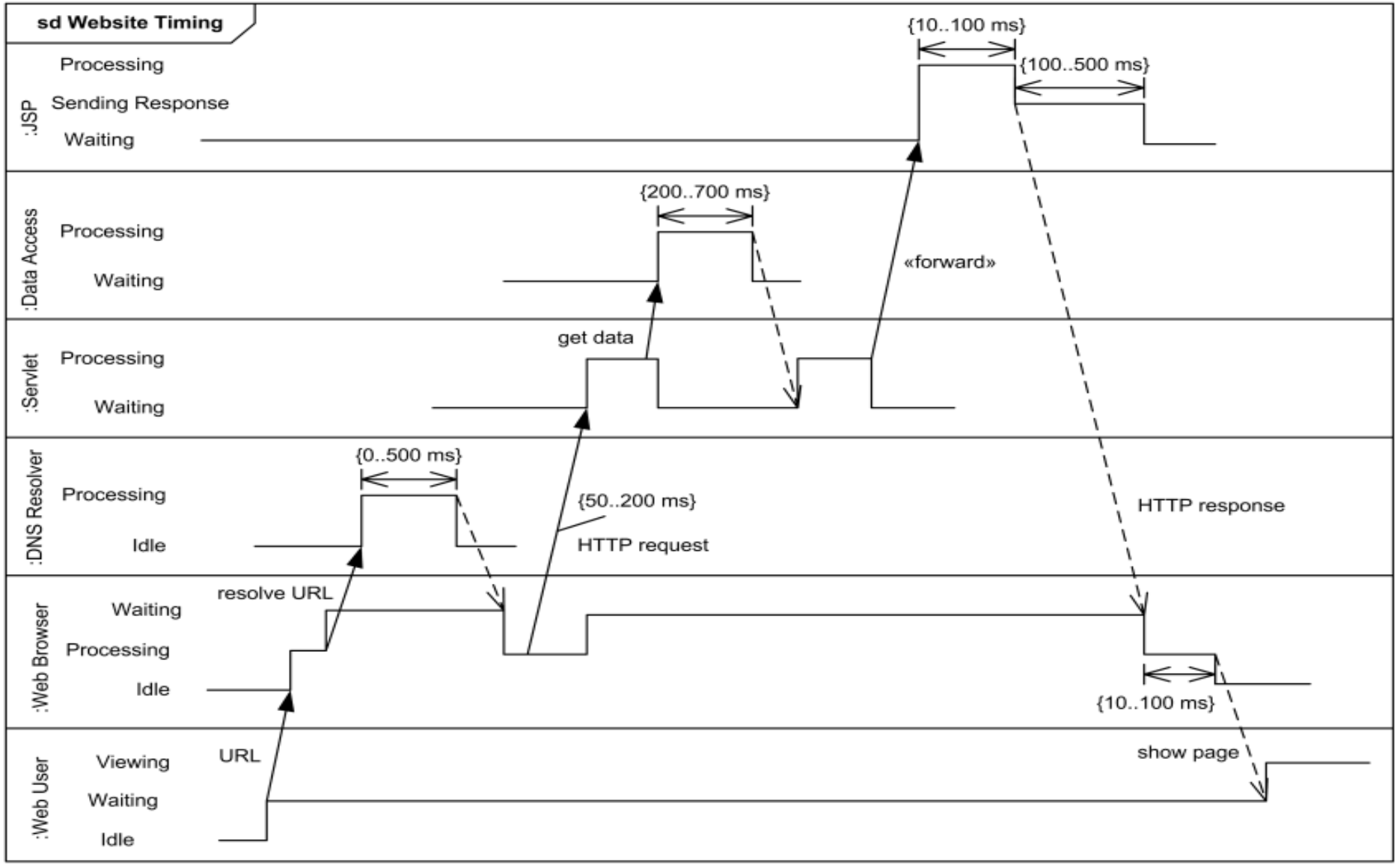




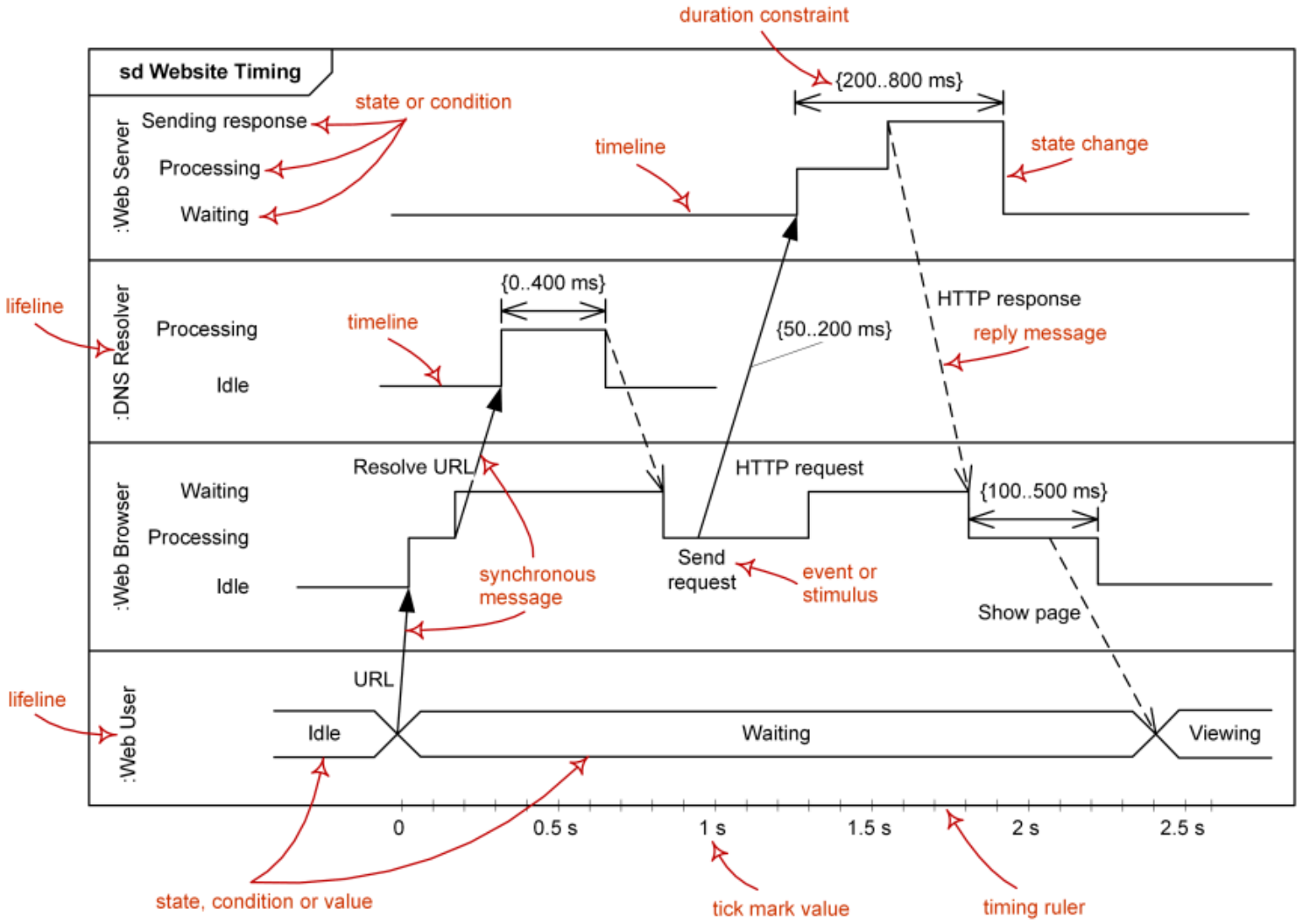
Timing Diagrams



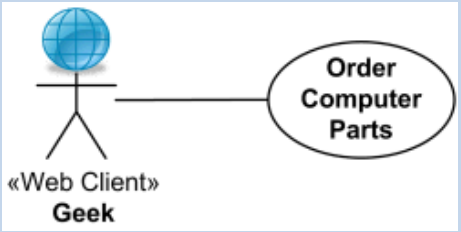
Timing Diagrams



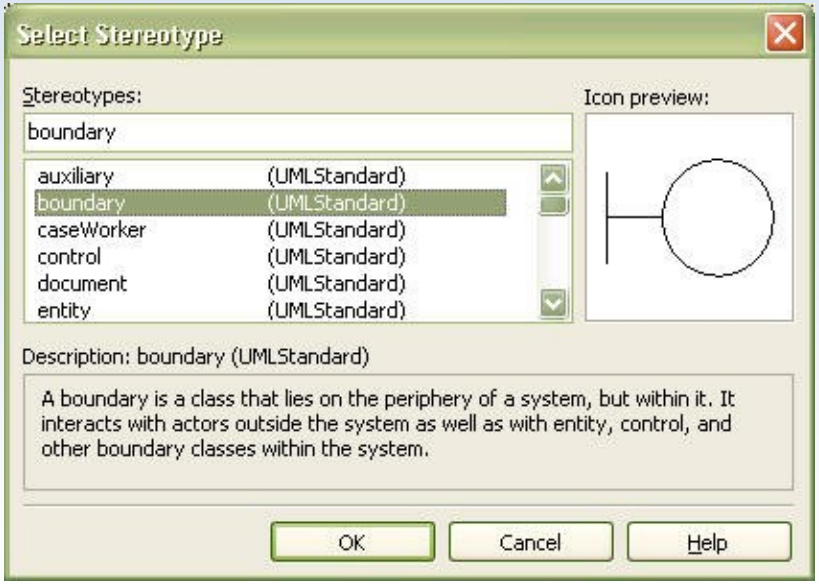
Timing Diagram Overview

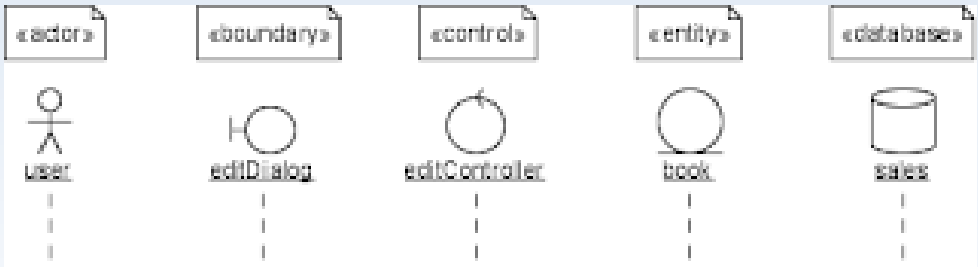


https://www.uml-diagrams.org/timing-diagrams.html



(extra…)
a short note on stereotypes…





http://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/stereotypedlg.html

UML Stereotypes

UML supports *stereotypes*, which are an inbuilt mechanism for logically extending or altering the meaning, display, characteristics or syntax of a basic UML model elements. You can apply stereotypes to a range of model element types, including:

- Elements (such as Classes and Objects)
- Relationships (such as Dependencies and Associations)
- Association Ends
- Attributes and Operations
- Operation Parameters

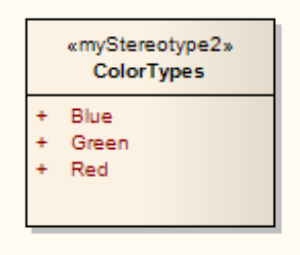
Different model elements have different stereotypes associated with them. You can create and use your own stereotypes in three different ways:

- To create a new object **type** based on a basic UML model element type, to be imported as part of a Profile into any model and made available for use through the Diagram Toolbox; examples of extended element types already provided in Enterprise Architect include a Table element (which is a stereotyped Class element) and Boundary, Control and Entity elements (which are are stereotyped Object elements)
- To customize the appearance or property of an **instance** of a model element of a specific type; these stereotypes are applied only through the Properties dialog of the object, within the model in which they are created, although you can transport custom stereotype definitions between models as Reference Data
- As a simple label on an element, to identify the role or nature of the object that the element represents

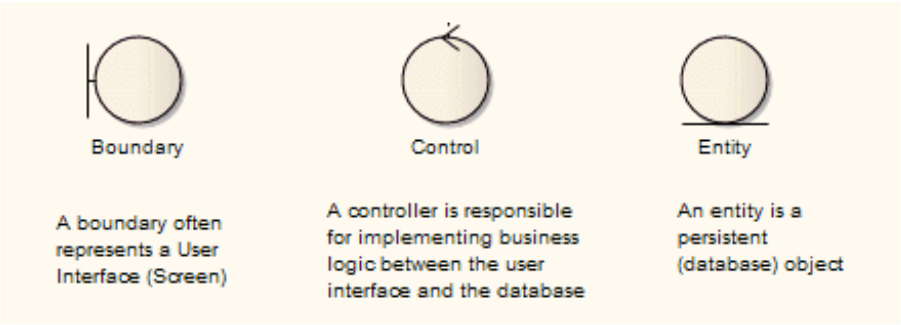
http://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/stereotypedlg.html

For further definitions of stereotypes, see the OMG UML specification (*UML Superstructure Specification, v2.1.1, section 18.3.8, pp. 667-672*).

Where a stereotype does not affect appearance, it is generally indicated by name on the base UML object shape. In the example below, *«myStereotype2»* is the stereotype name. Some of the built-in stereotypes are also represented by icons; see *Stereotype Visibility*.



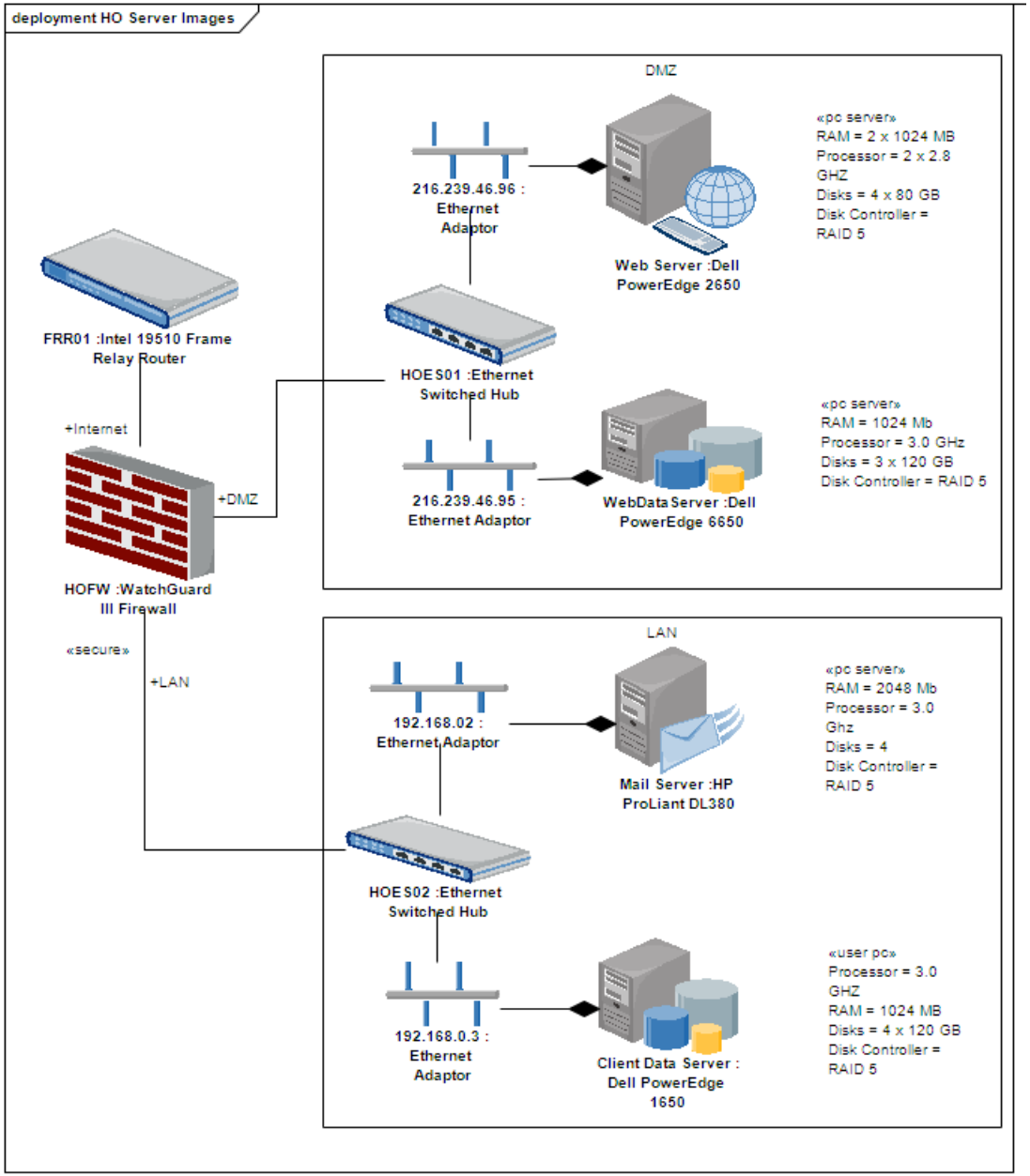
Where the stereotype causes the element to be drawn differently, or is used to define a new type of object, the element shape can be quite different, as illustrated by the three Robustness diagram stereotypes:



You apply a new appearance or shape by associating the stereotype with either a metafile (image file) and fill, border and text colors, or a Shape Script that defines the shape, dimensions and text of the object.

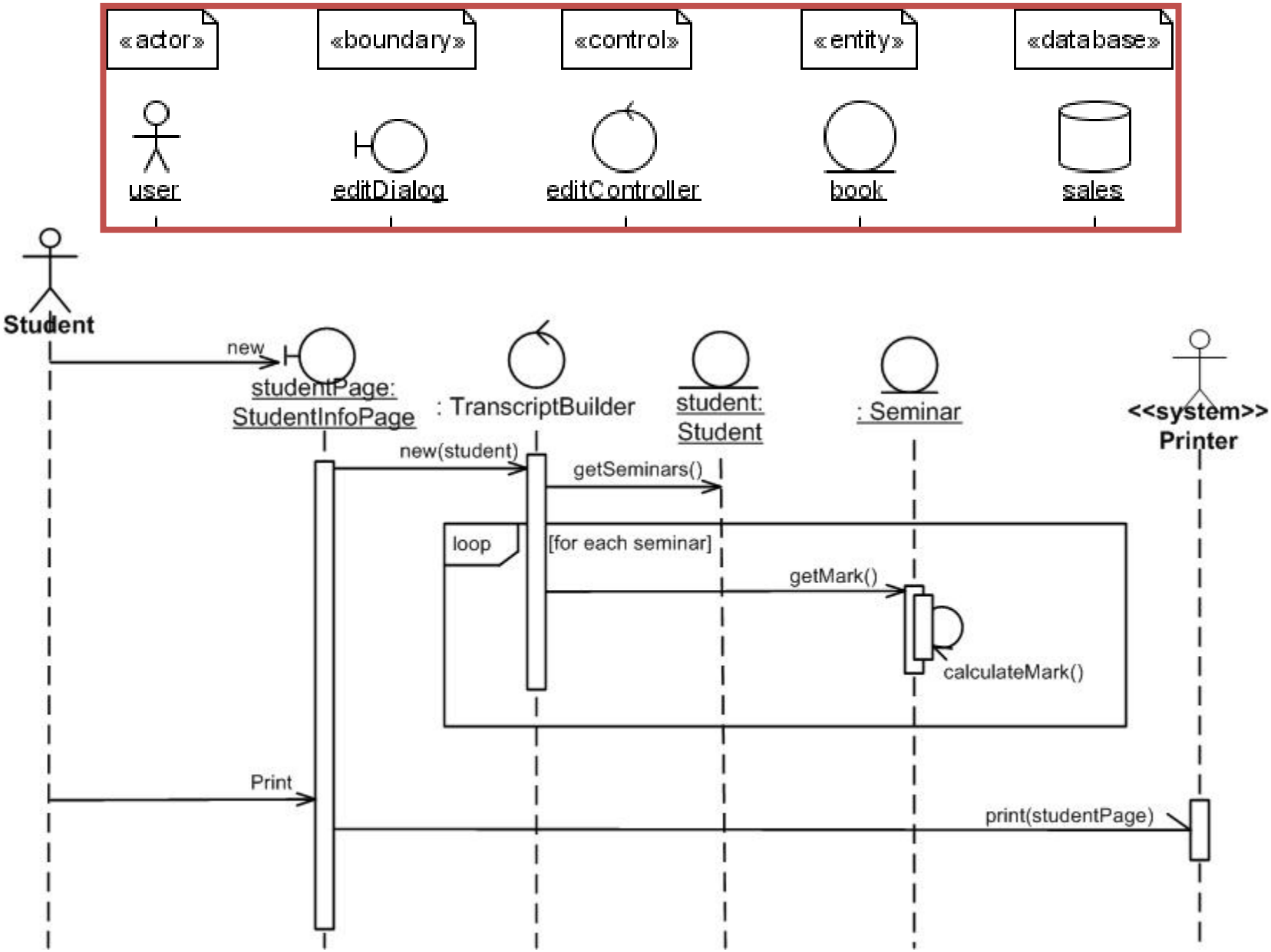
http://www.sparxsystems.com/enterprise_architect_user_guide/10/standard_uml_models/stereotypedlg.html

Stereotypes with Alternative Images



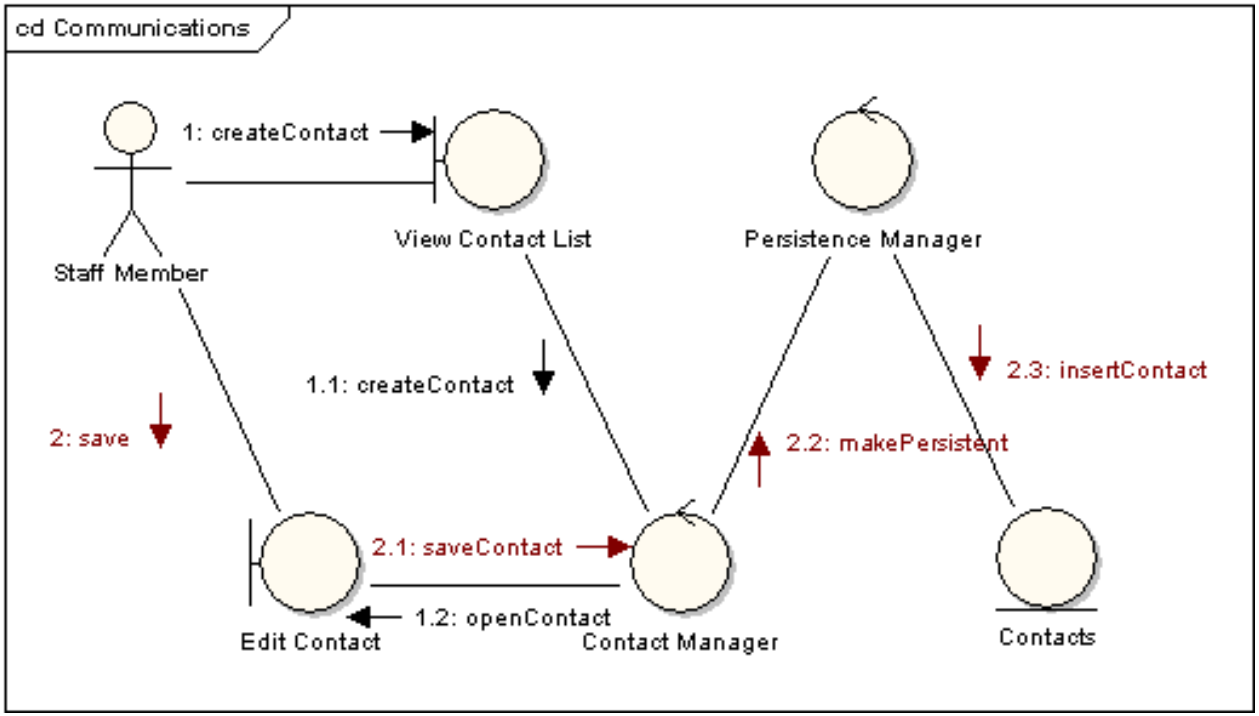
Notes
- You cannot change the representation of elements that include Lifelines, such as those in Sequence diagrams; the standard representation is important in the use and function of those elements

Some useful standard stereotypes: BCE – Boundary-Control-Entity



Communication Diagrams (UML)

- Communication diagrams show interactions between objects
- A **cd** always has an equivalent **sd**.
- Communication diagrams focus on the collaboration patterns between objects
- The sequence diagram focus on the timing (i.e. the sequence)

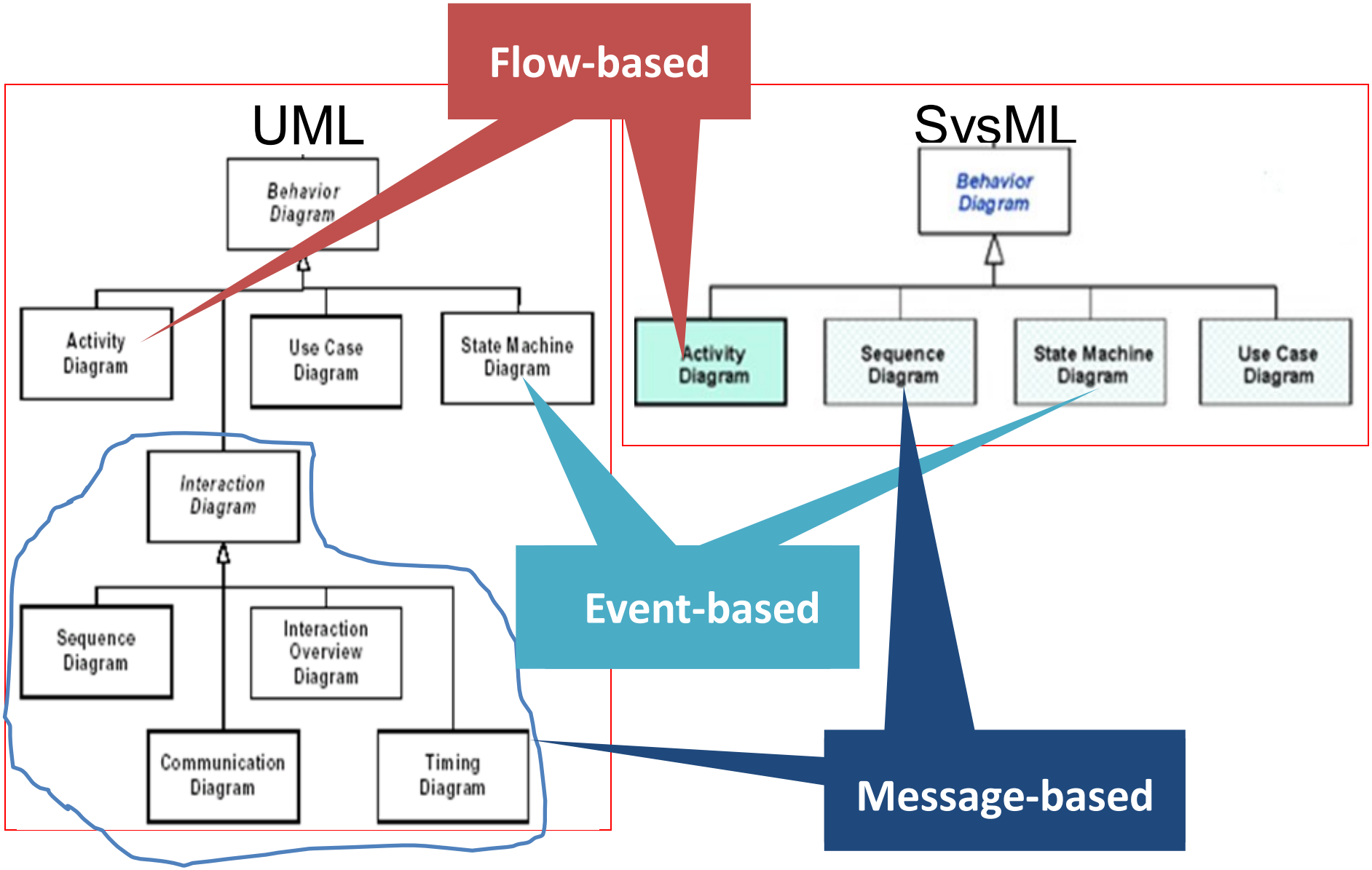


DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

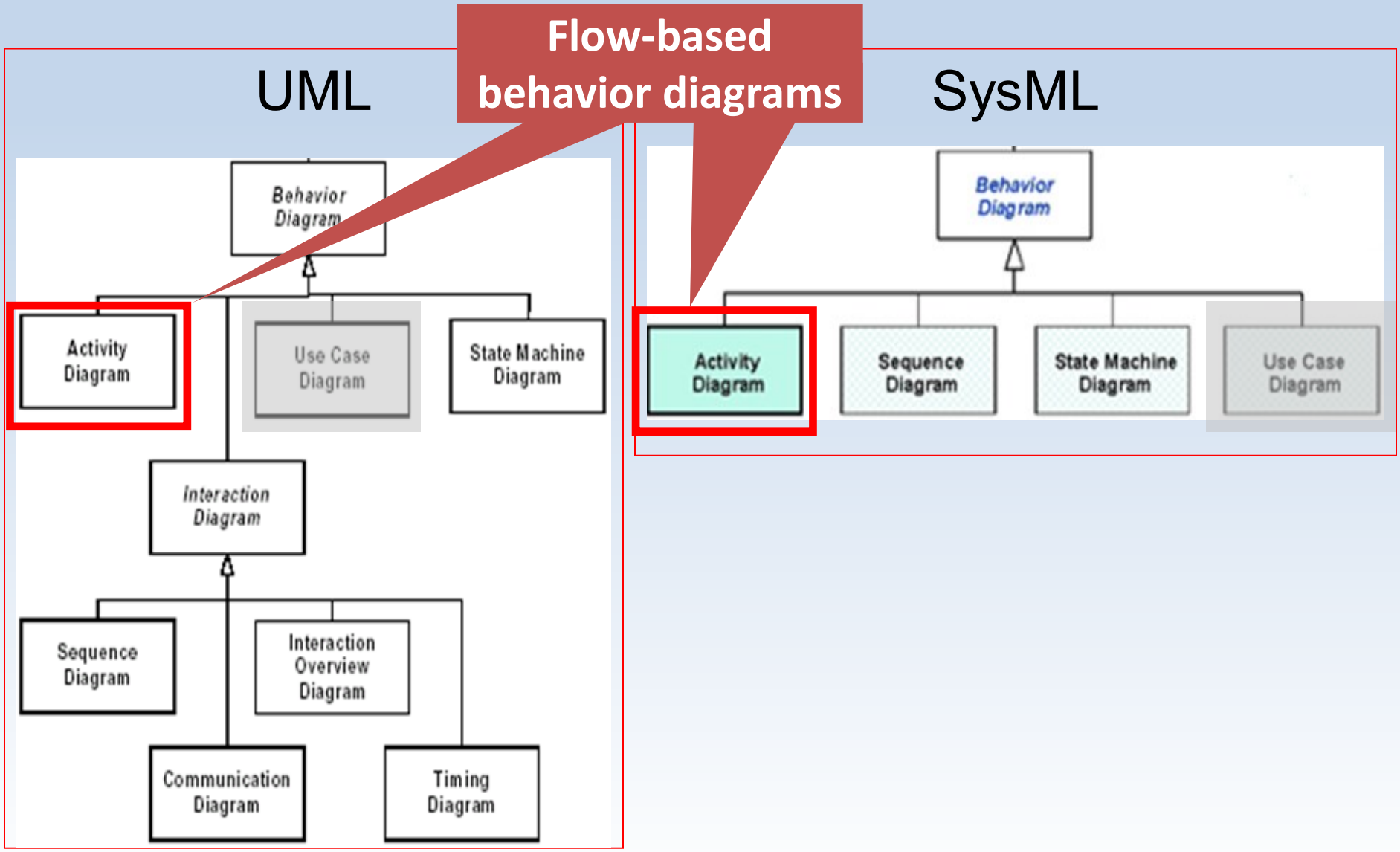
Behavioural Modelling with UML (and SysML)

- Activity diagrams

Behavior in UML and SyML



Flow-based Behaviour



Activity Diagrams (UML)

- An activity diagram describes valid sequence of activities performed by one or more entities of the system.
- An activity diagram describes a workflow detailing the control flow and data flow.
- Activity diagrams are useful for describing
  - Algorithms
  - Use case Scenarios
  - High level (business) modeling to describe how work is performed.

(BPMN provides a specialized language to describe business processes (a process is a specific type of workflow)).

Events as Activities

"Activity modeling emphasizes the inputs, outputs, sequences, and conditions for **coordinating** other behaviors. It provides a flexible link to blocks owning those behaviors."

OMG Systems Modeling Language (OMG SysML) (Page 85)

"Activity is some work that is carried out; **it might overlap several Use Cases or form only a part of one Use Case**."

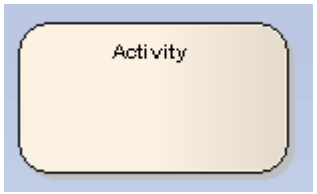
Enterprise Architecture User Guide

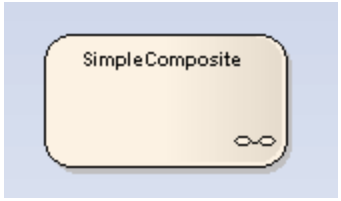
Activities and Actions

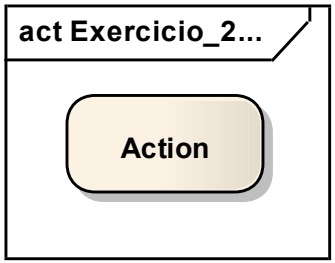
- An activity organizes and specifies the participation of subordinate behaviors, such as sub-activities or actions, to reflect the control and data flow of a process.
- Activities are used in activity diagrams for various modeling purposes, from procedural-type application development for system design, to business process modeling of organizational structures or work flow.

- In simple terms:
  - Activities are made of other Activities or Actions.
  - Actions are Activities that are not further decomposed.

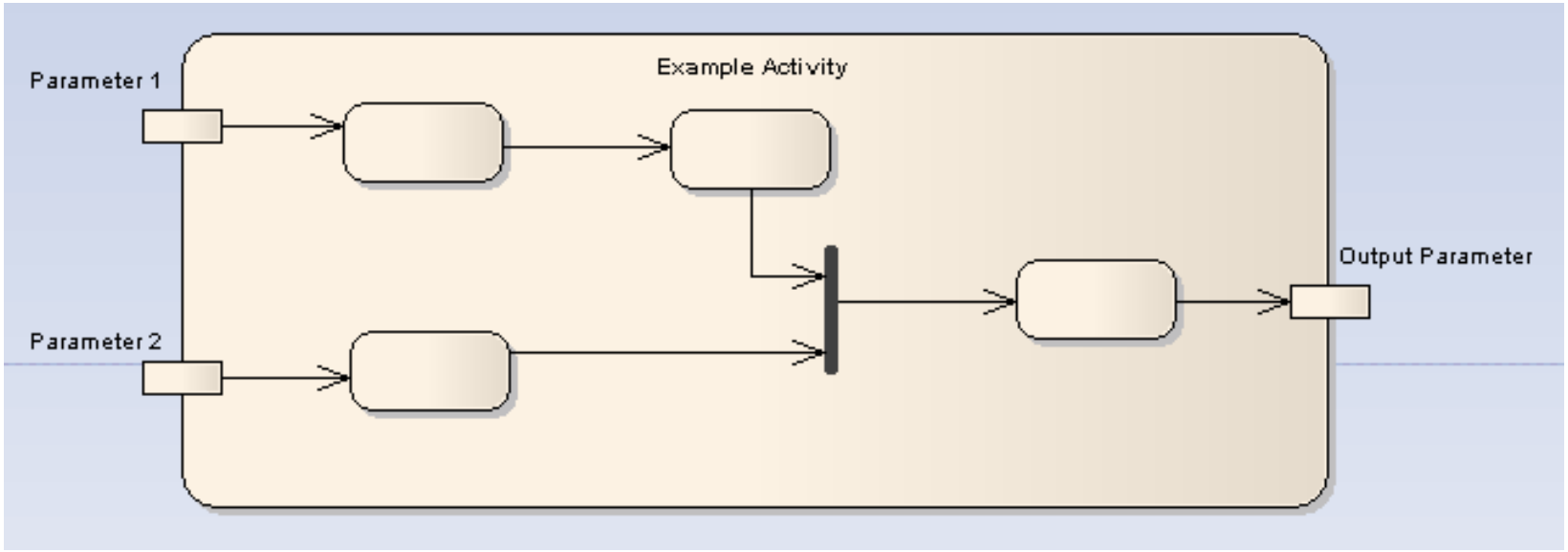
Activities and Actions





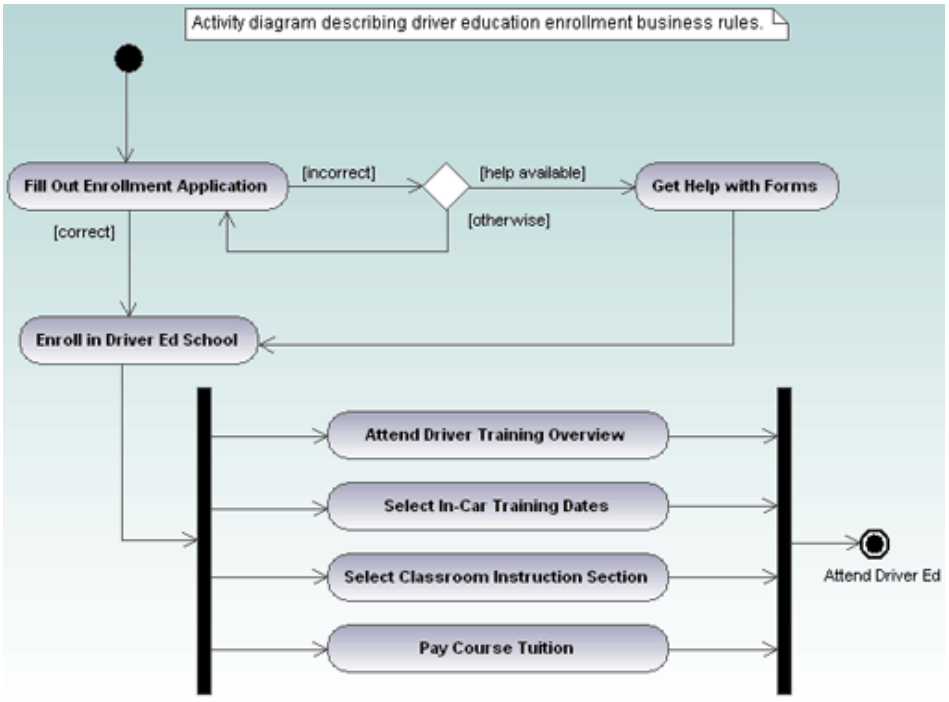


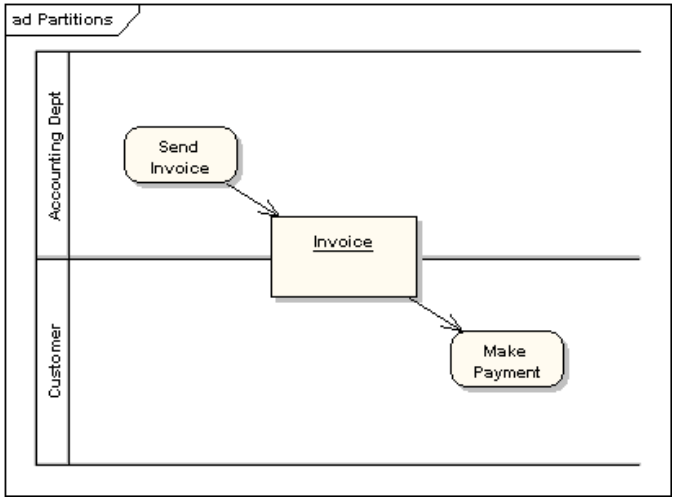
The following diagram of an Activity contains Action elements and includes input parameters and output parameters.



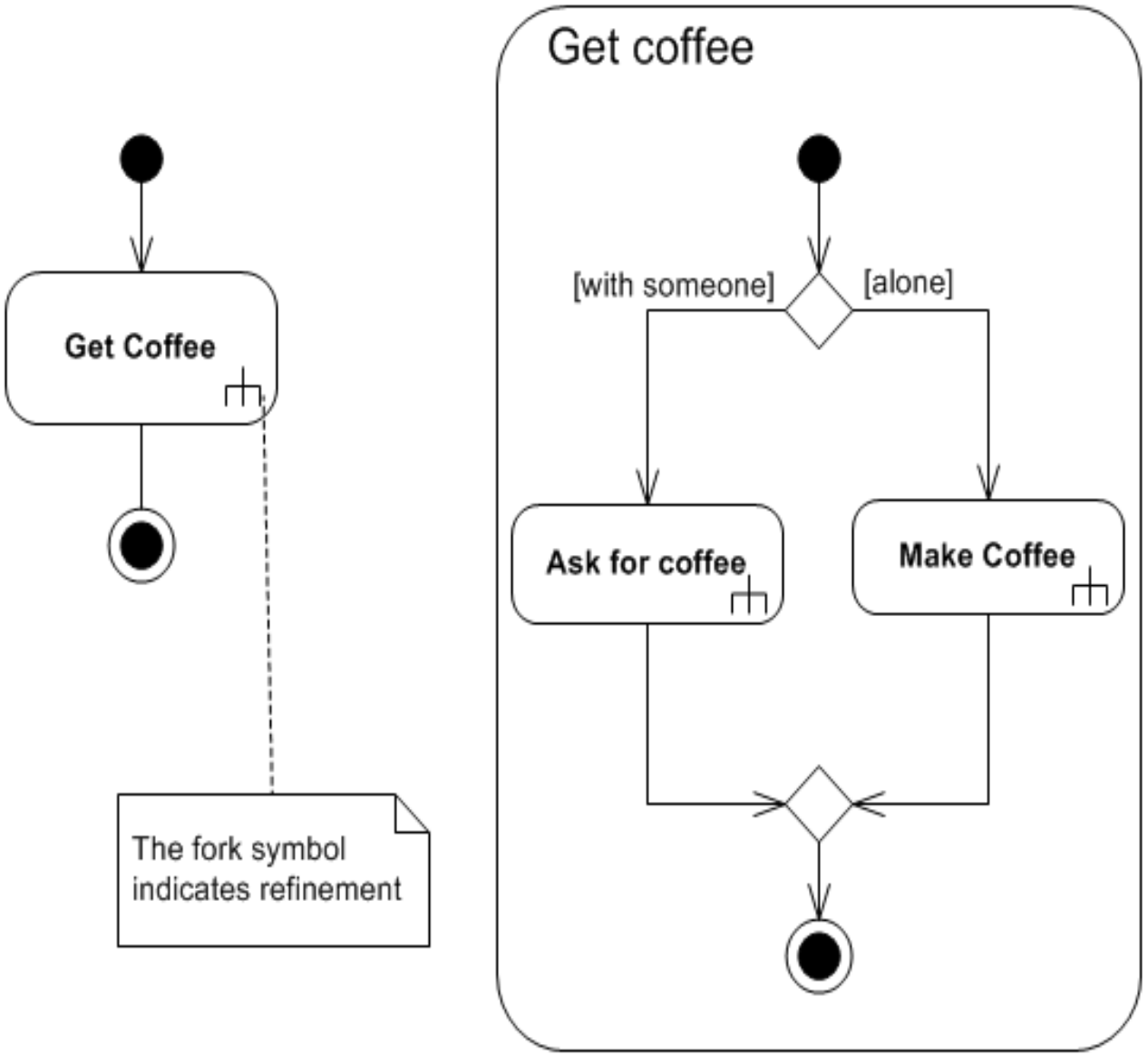
Activity Diagrams (UML)

- An activity diagram is used to display the sequence of activities involving one or more entities of the system. They show an **workflow** detailing the decision paths.
- Activity diagrams are useful for business modeling where they are used for detailing the processes involved in business activities.

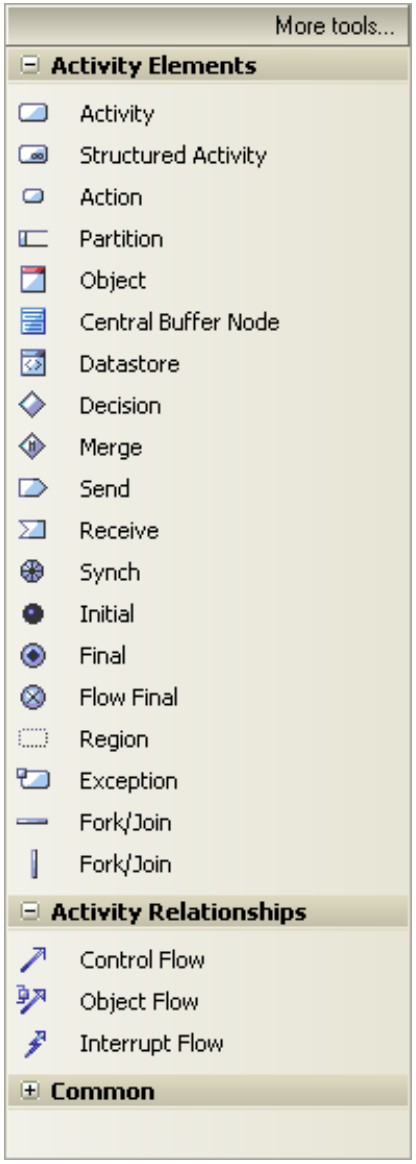


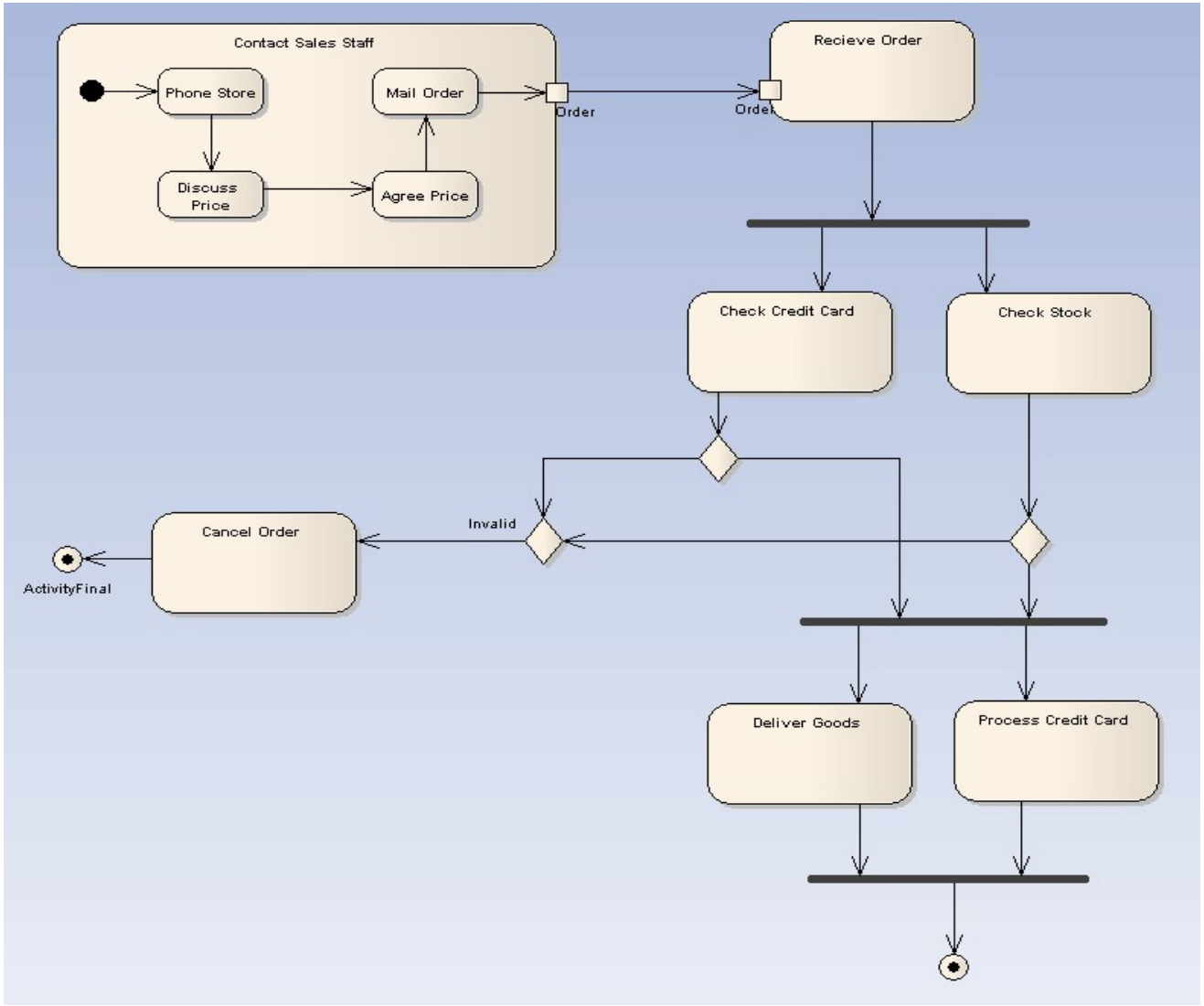


Activity Diagrams

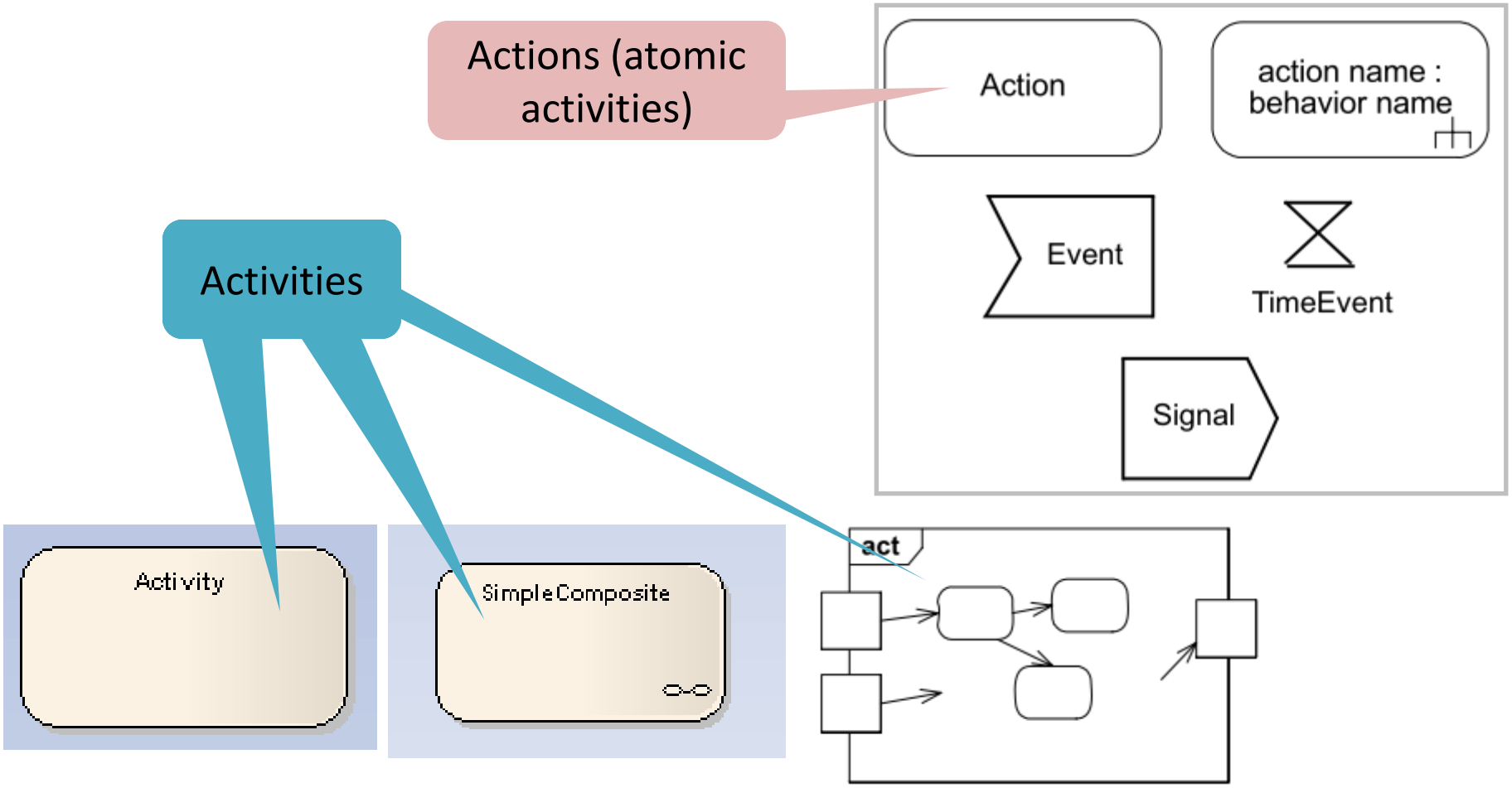


Activity Elements and Connectors

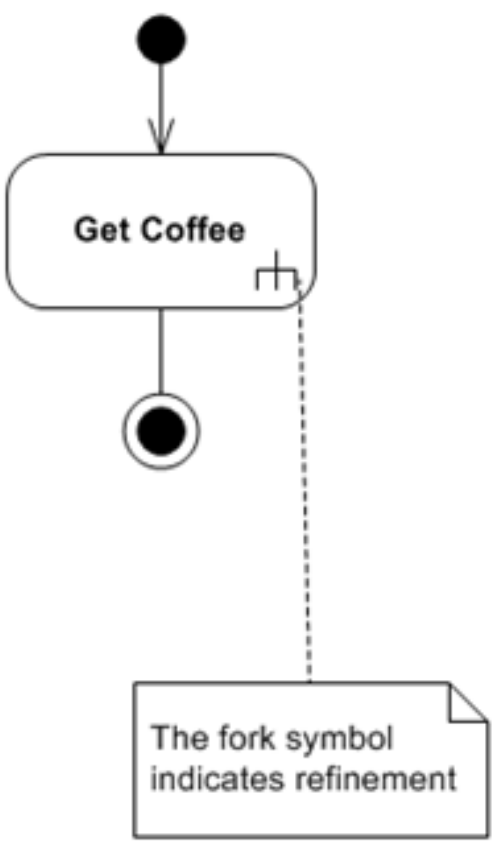


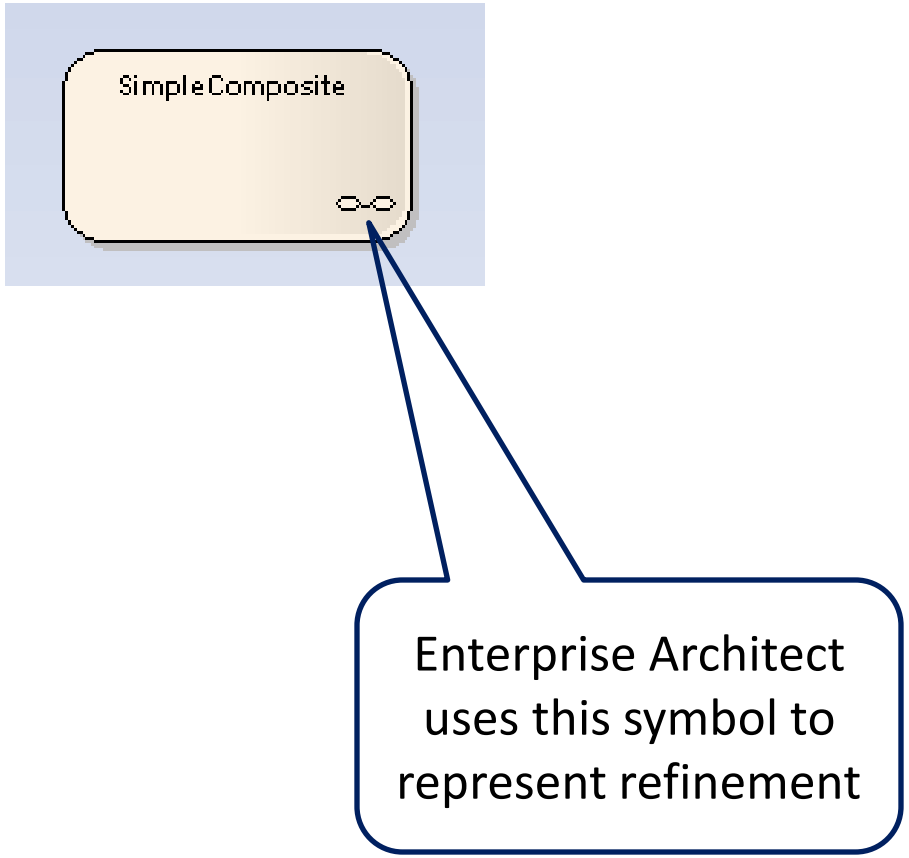


Graphical Nodes on Activity Diagrams

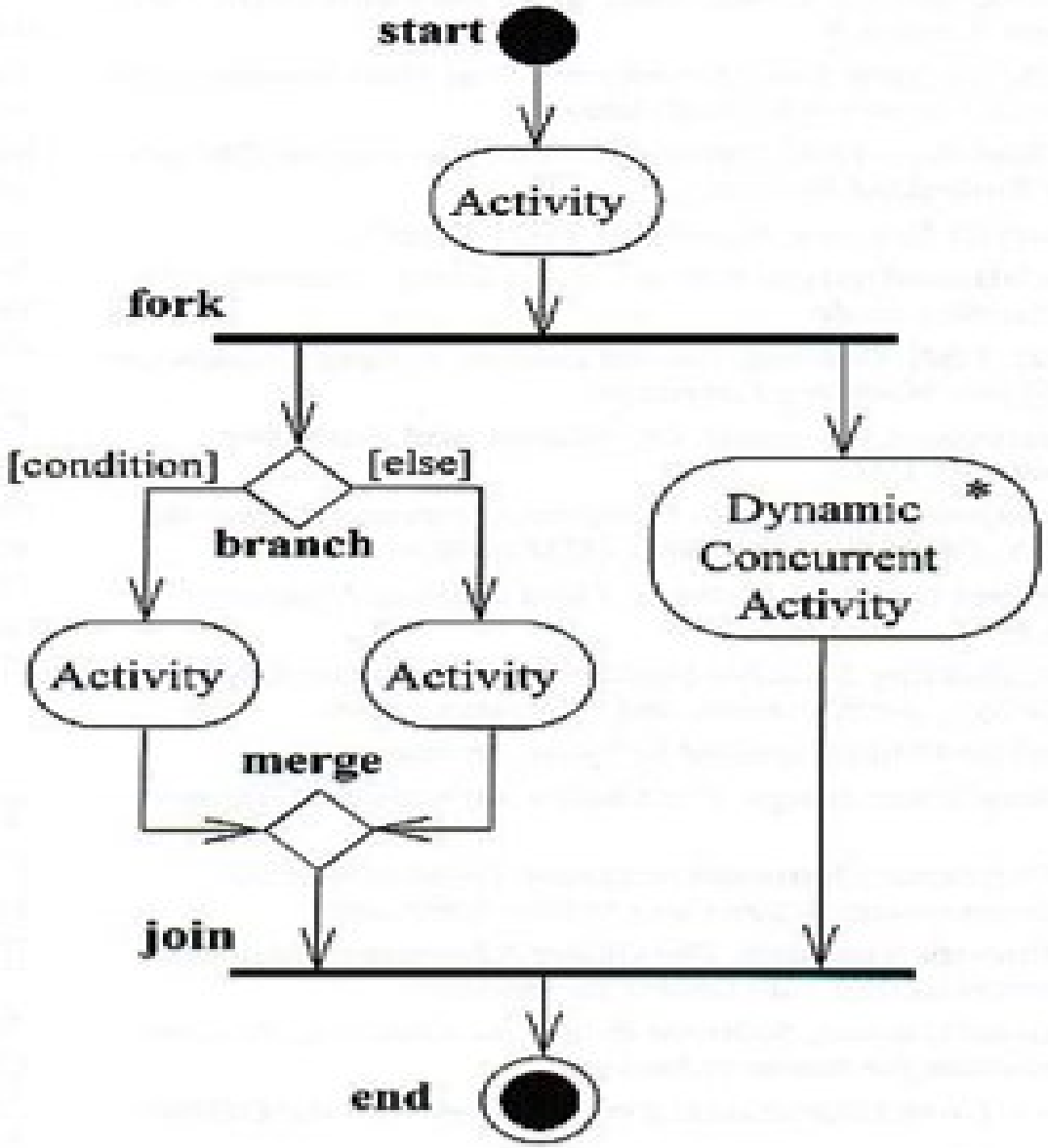


Refinement of Activity elements

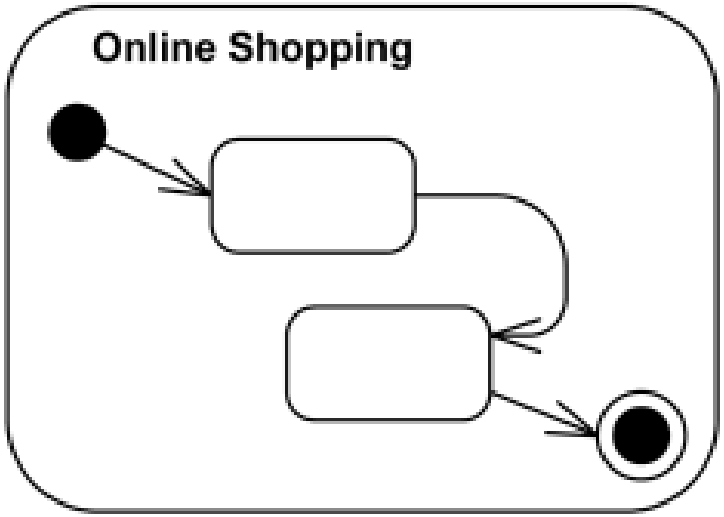




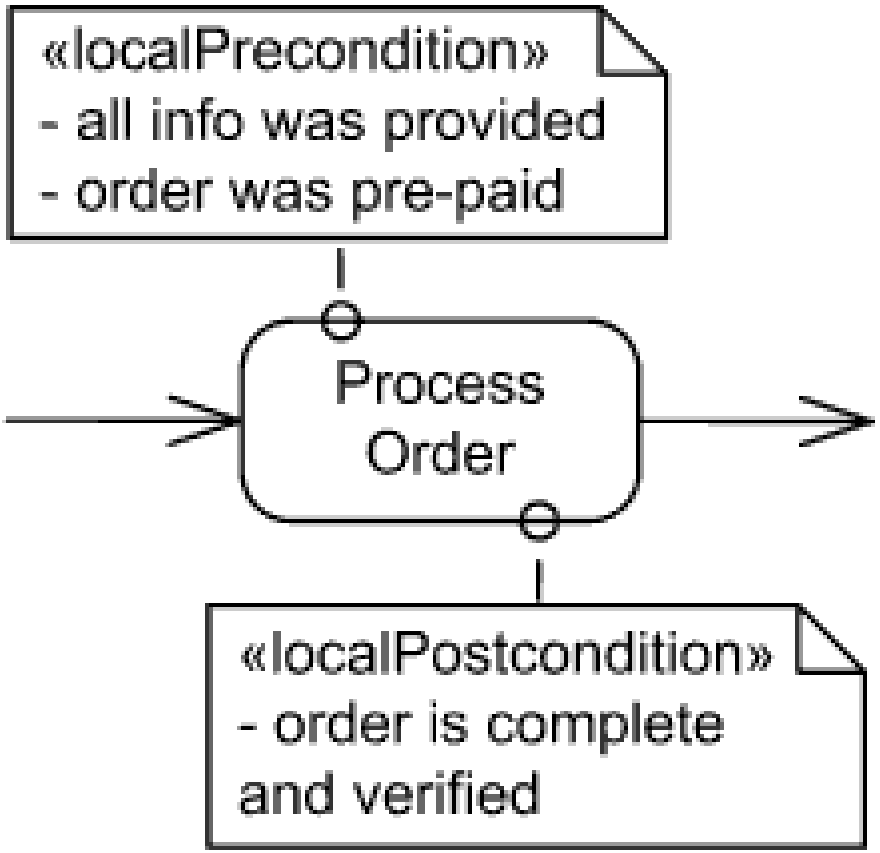
Control Flow



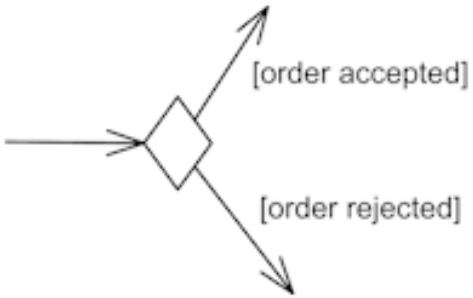
Initial and Final Nodes

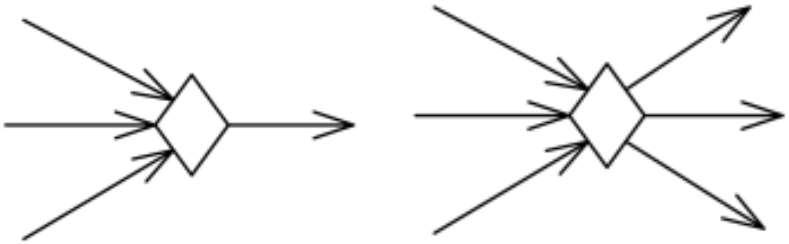


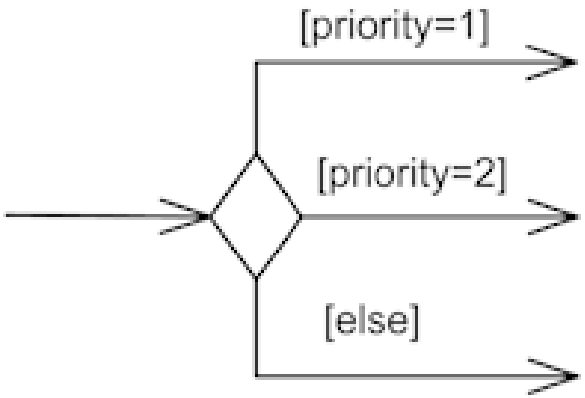
Pre- and Post-Conditions



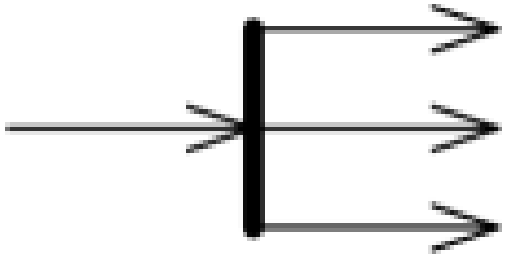
Decision and Merge

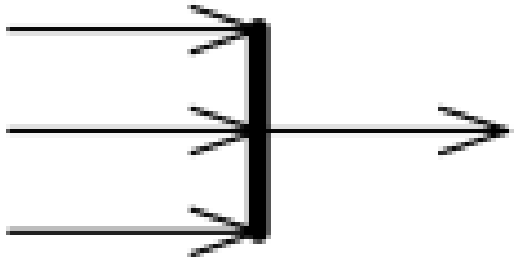






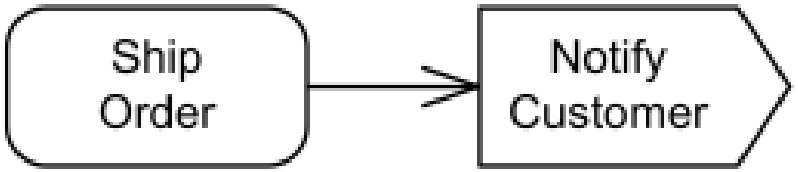
Fork and Join



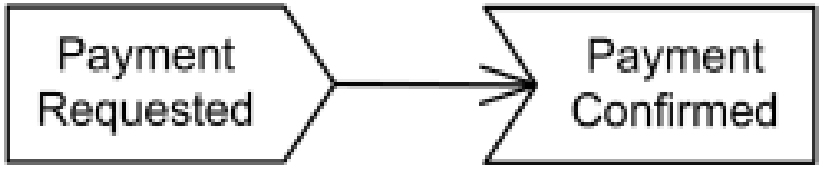


Send and Accept Signal

- After order is shipped, **Notify Customer** send signal action creates and sends **Notify Customer** signal.

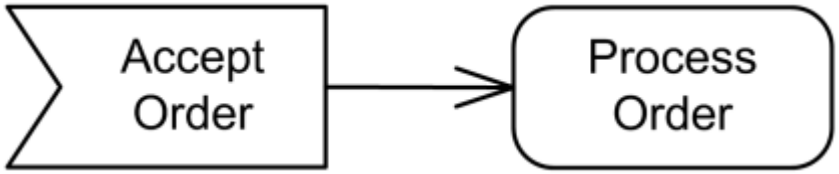


- **Payment Requested** signal is sent. The activity then waits to receive **Payment Confirmed** signal.
- Acceptance of the **Payment Confirmed** signal is enabled only after the request for payment is sent.



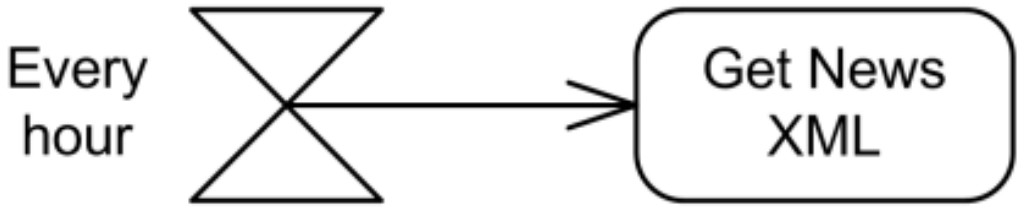
Accept Event

- Acceptance of the **Accept Order** event by **Accept Order** action causes an invocation of a **Process Order** action.
- The **accept event action** is enabled upon entry to the activity containing it.

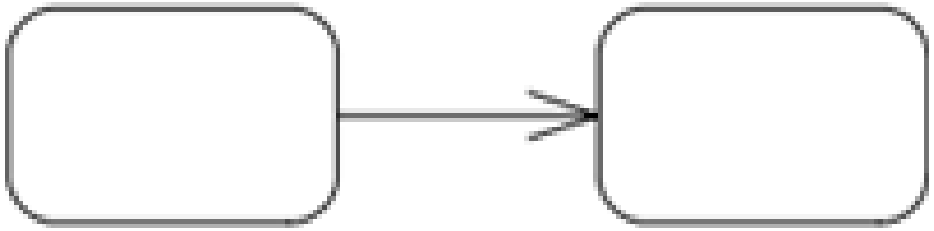


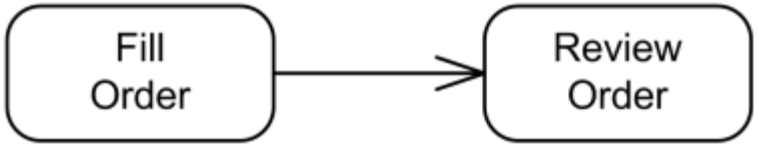
Wait Time Action

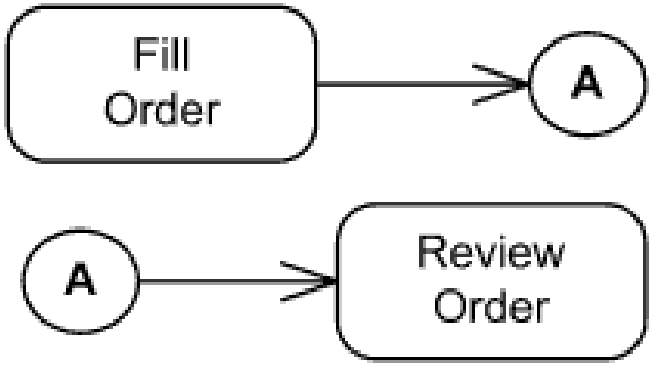
- The **Every Hour** accept time event action generates output every hour.
- There are no incoming edges to this time event action, so it is enabled as long as its containing activity or structured node is enabled.



Activity Edge (Simple Transition)

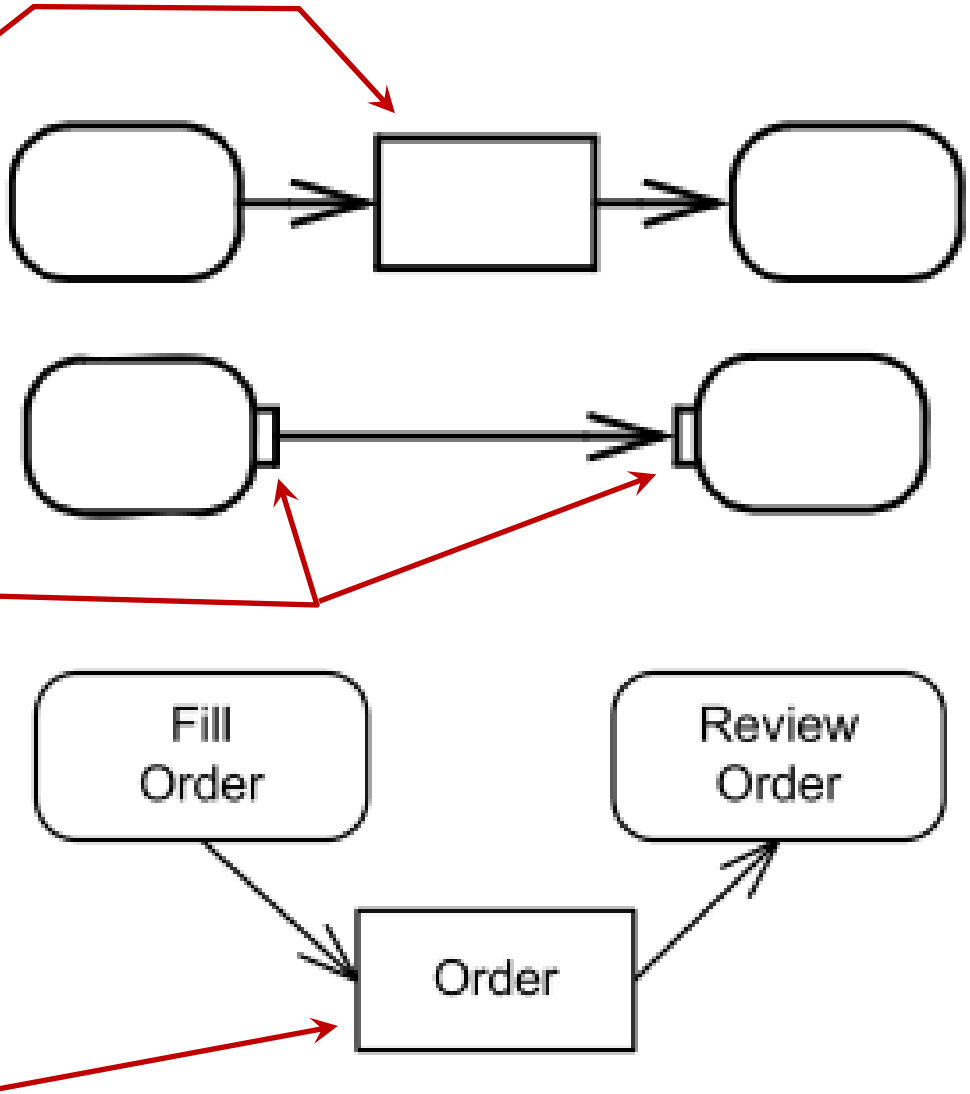






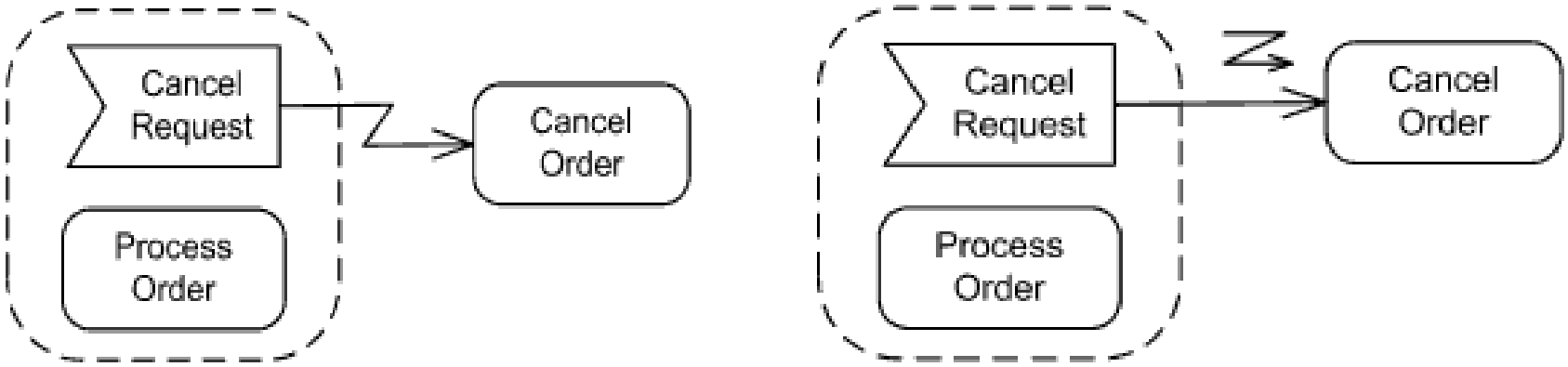
Object Flow Edge (data flow)

An information object is created in the frist activity (or, if it already existed, its state is changed) and is made available to the the second activity.



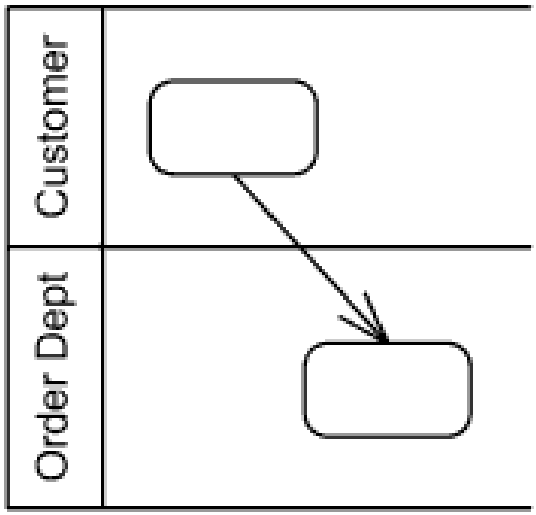
Interrupting Edge

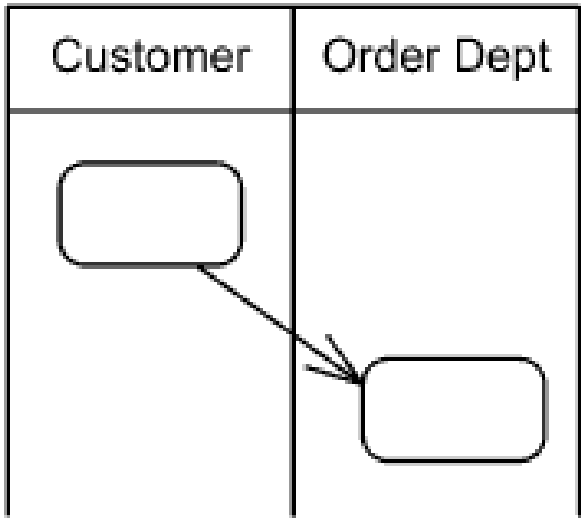
- **Cancel Request** signal causes interruption resulting in Cancel Order.



Partitions

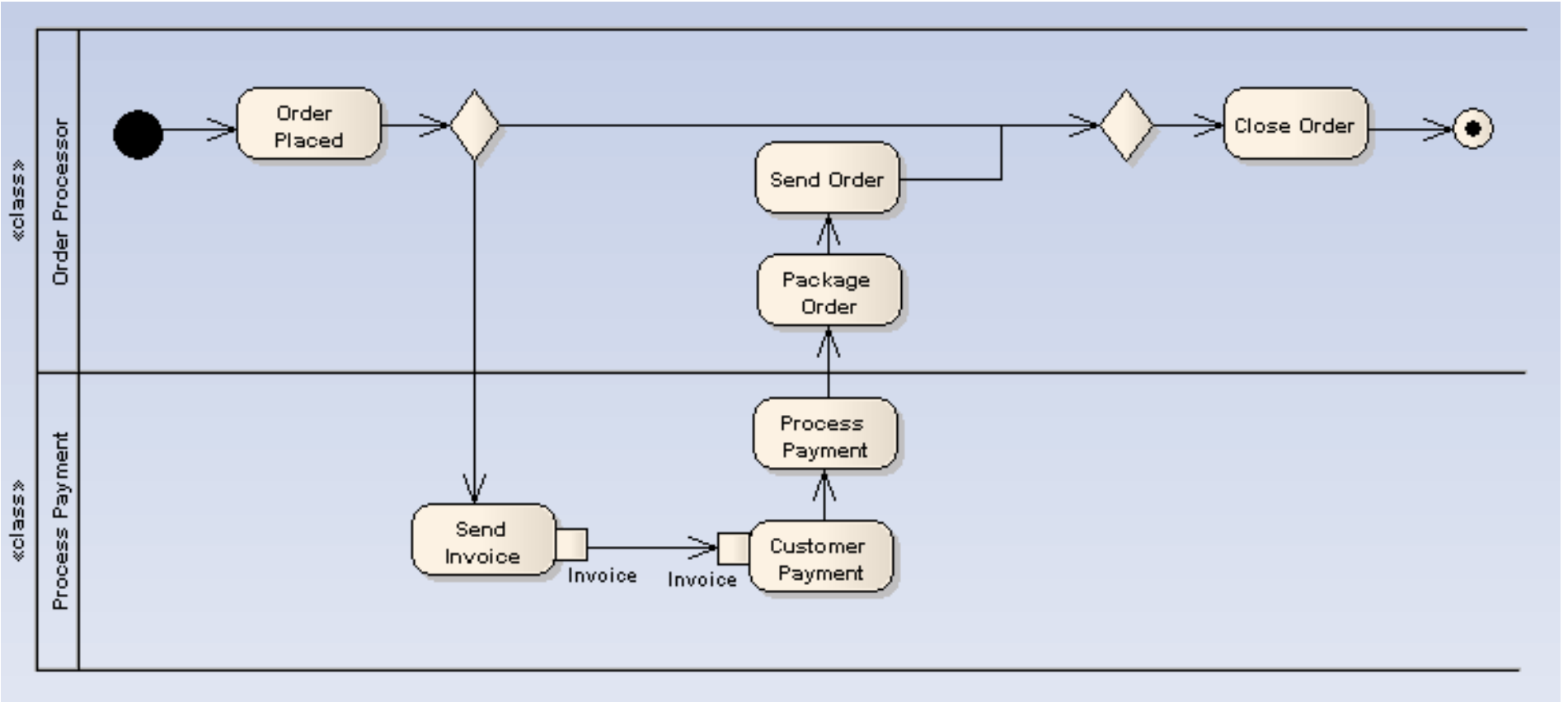
- **Partitions** make possible to contextualize activities/ and actions in specific scopes of execution (can be an abstract sub-domain of the domain model, to which we simply assign a name; a component; a class/block; …; ultimately, any entity it is found relevant to identify or define conceptually for this purpose…)
- Activity partitions **Customer** and **Order Dept** as horizontal or vertical swimlanes .



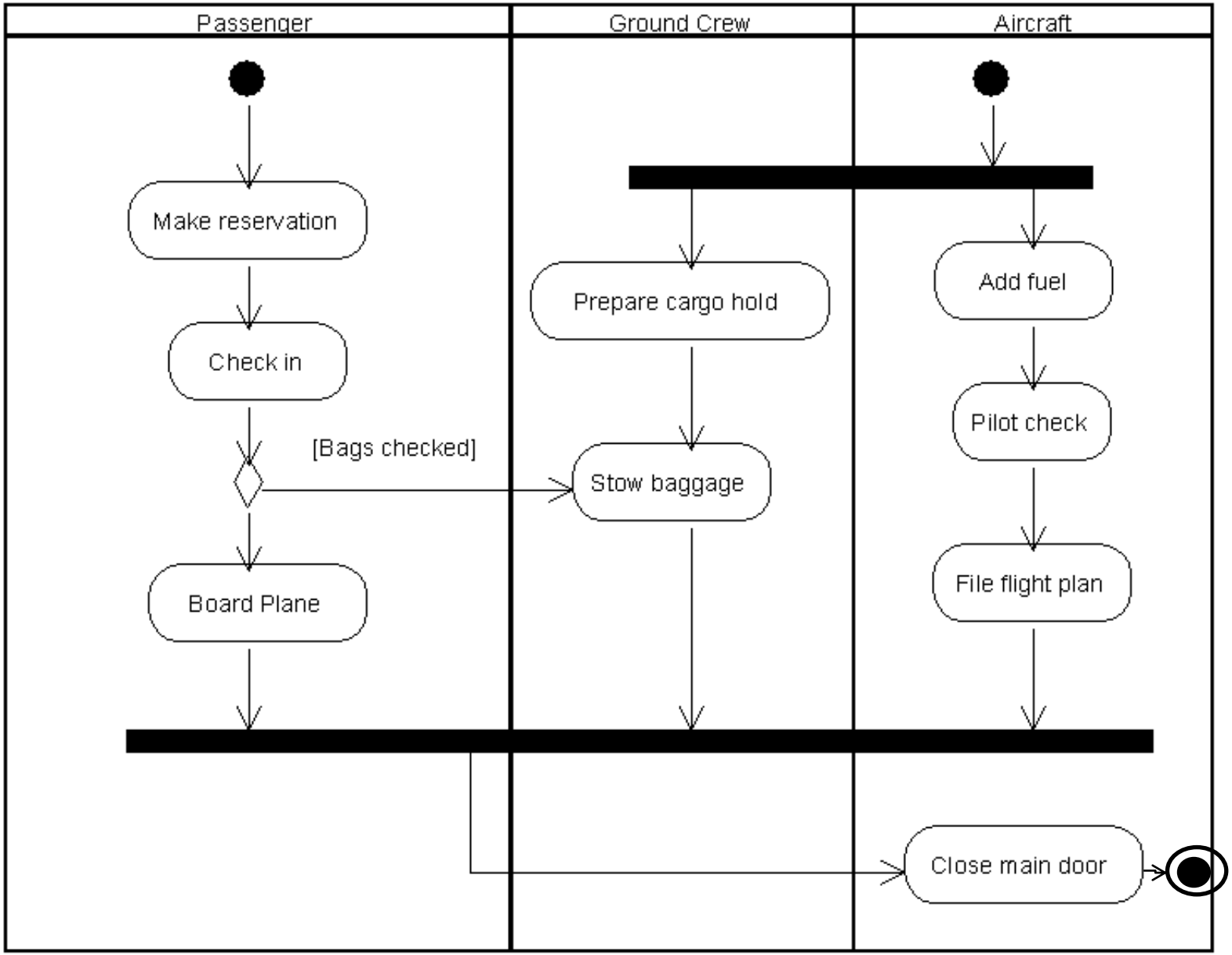


ad (activity diagram) Examples

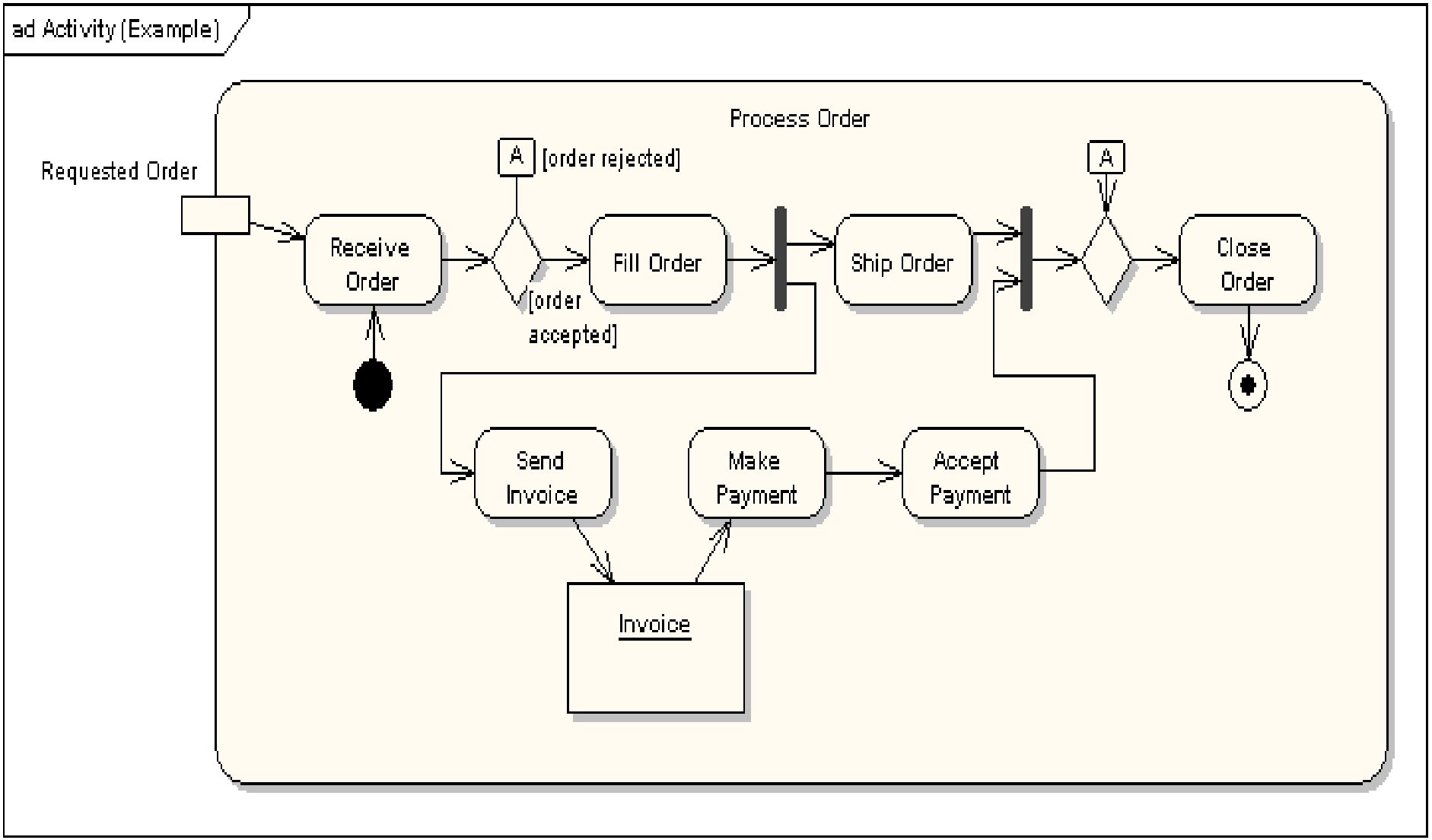
- The context of an activity is associated to the system's entity which is responsible for its execution.
- Activity diagrams can use partitions (lanes) to represent multiple contexts.

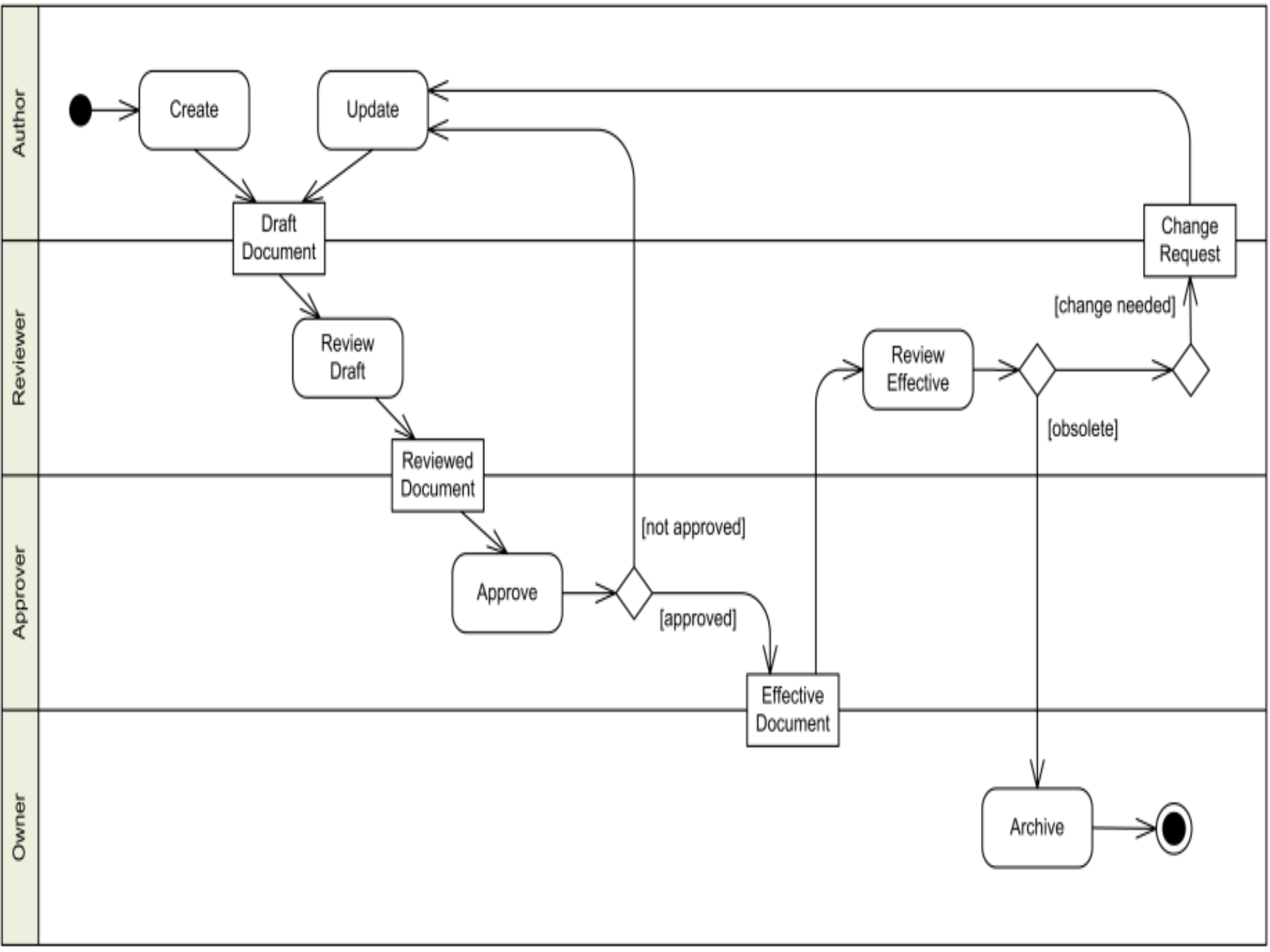


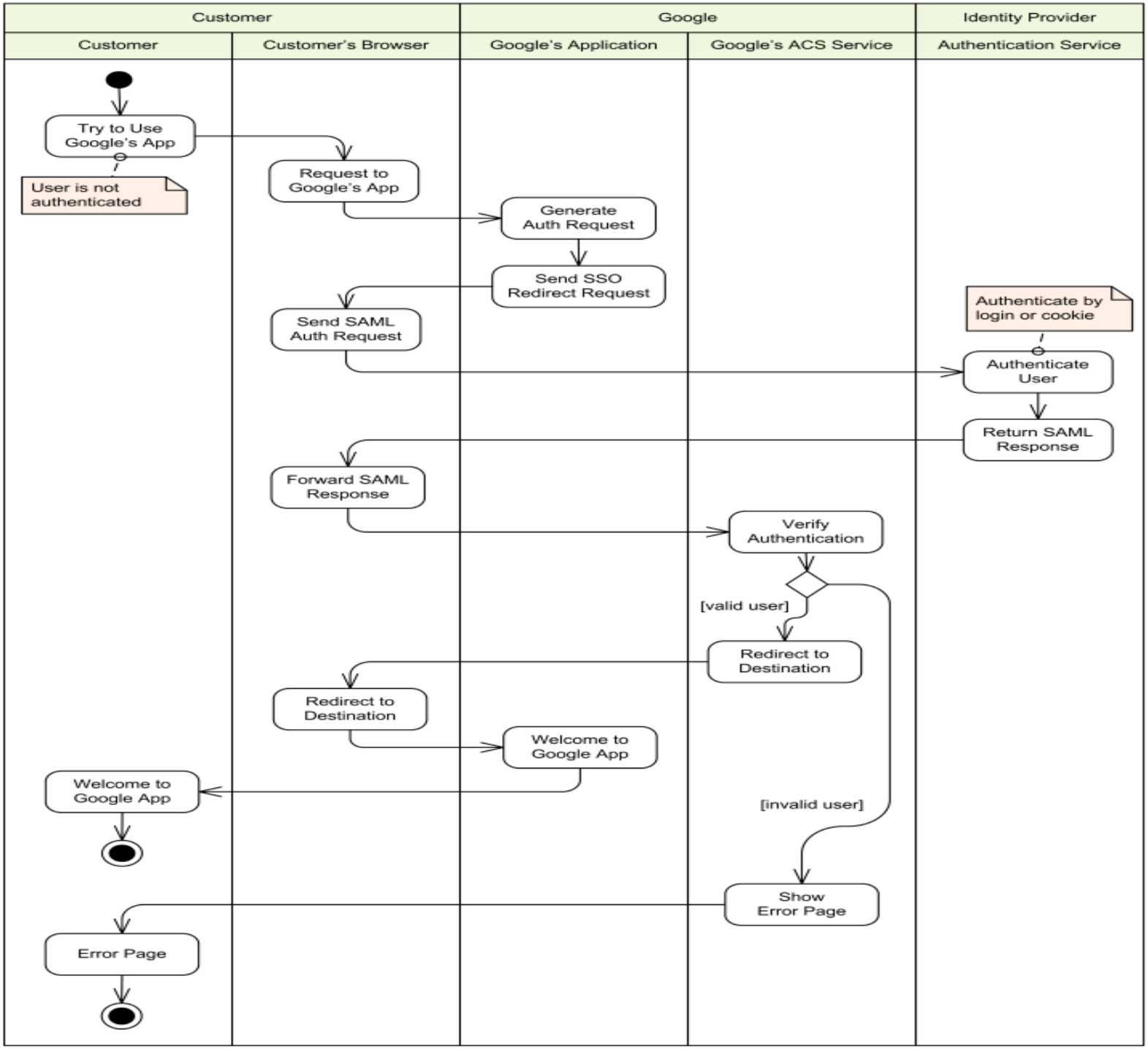
Activity Diagrams (UML)



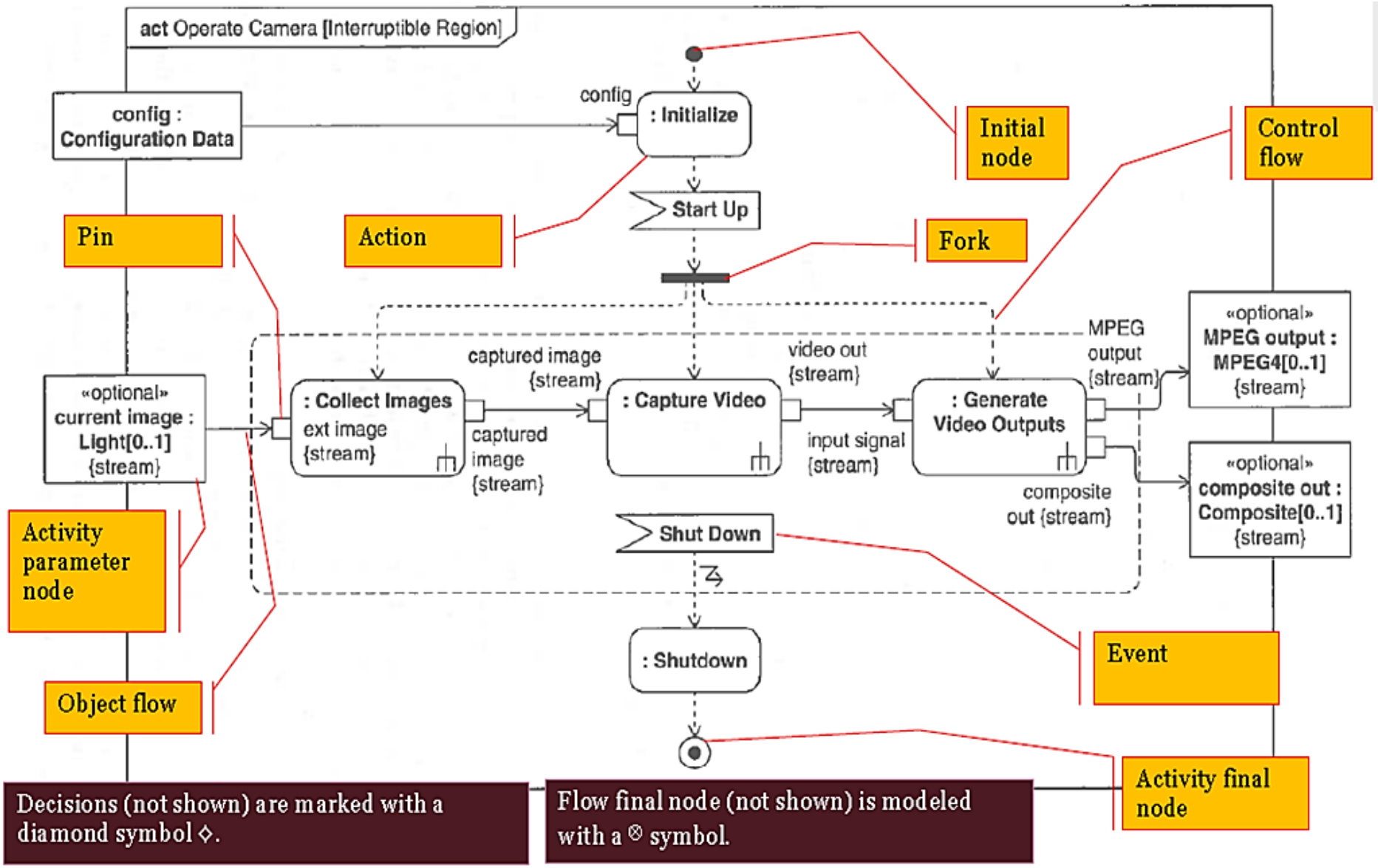
Activity Diagrams (UML)




Author
Reviewer
Approver
Owner
Create
Update
Draft Document
Review Draft
Reviewed Document
Approve
[not approved]
[approved]
Effective Document
Review Effective
[change needed]
Change Request
[obsolete]
Archive


Customer
Google
Identity Provider
Customer
Customer's Browser
Google's Application
Google's ACS Service
Authentication Service
Try to Use Google's App
User is not authenticated
Request to Google's App
Generate Auth Request
Send SSO Redirect Request
Send SAML Auth Request
Authenticate by login or cookie
Authenticate User
Return SAML Response
Forward SAML Response
Verify Authentication
[valid user]
Redirect to Destination
Redirect to Destination
Welcome to Google App
Welcome to Google App
[invalid user]
Show Error Page
Error Page

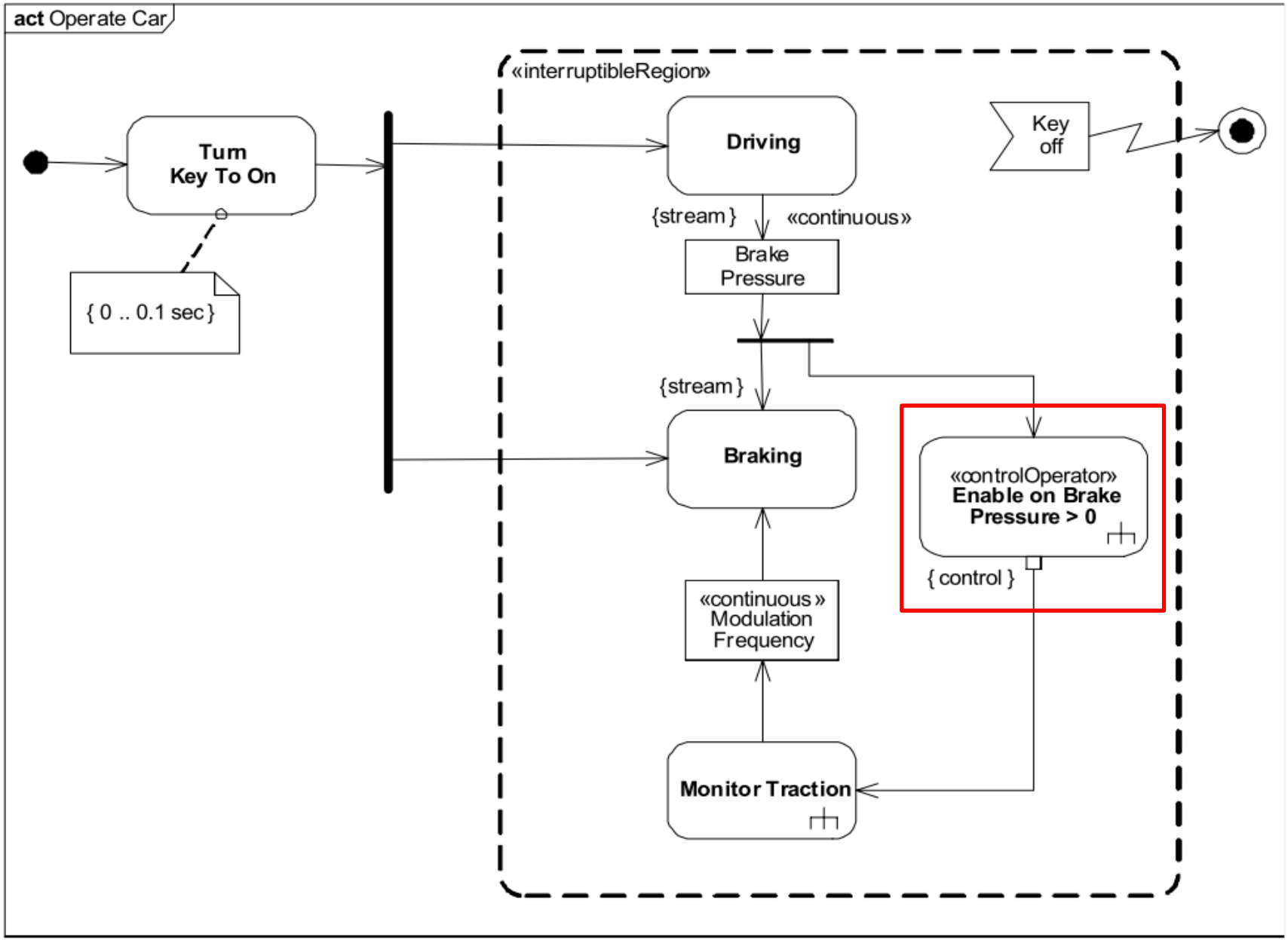
Activity Diagrams (SysML)



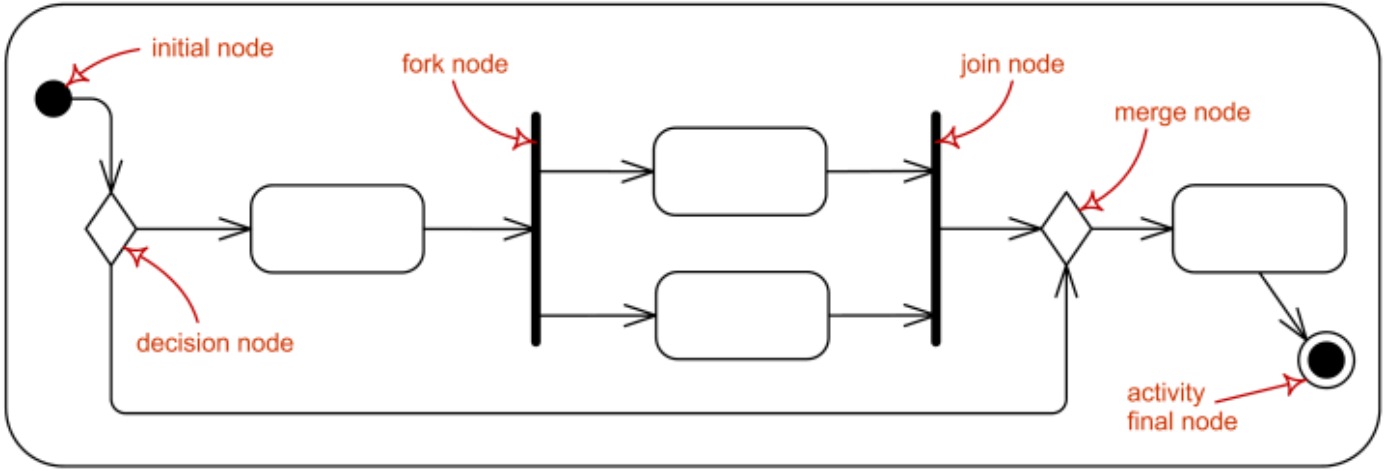
Activities in UML vs. SysML

- UML
  - Enables actions to start.
- SysML
  - Supports **disabling** running actions.
  - A control value is an input or output of a **control operator**, which is how control acts as data. A control operator can represent a complex logical operation that transforms its inputs to produce an output that controls other actions.
  - See section 4.8 of "System Engineering with SysML/UML" for more details.
- For all the rest, Activities are similar in UML and SysML.

Activity Diagram in SysML



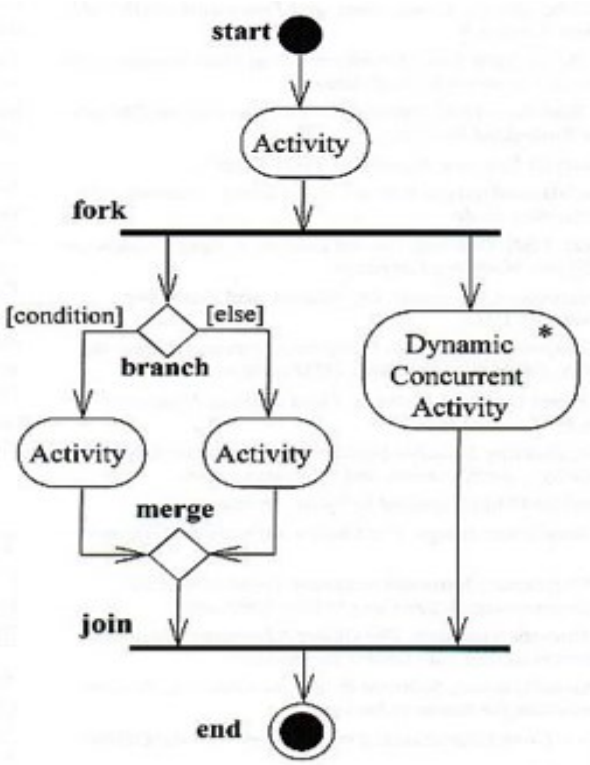
Activity Diagram Overview

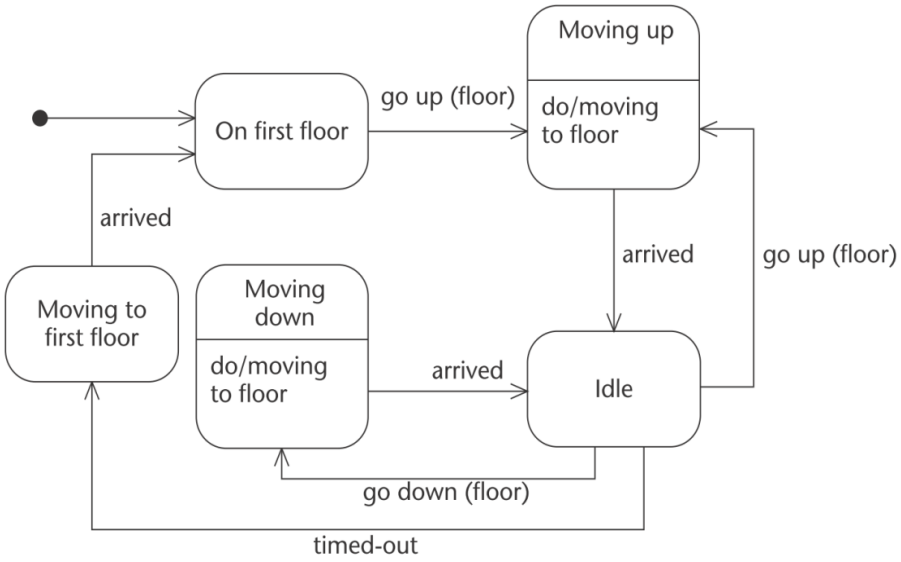


https://www.uml-diagrams.org/activity-diagrams-controls.html

VERY IMPORTANT: an ACTIVITY diagram IS NOT the same as a STATE MACHINE diagram!!!

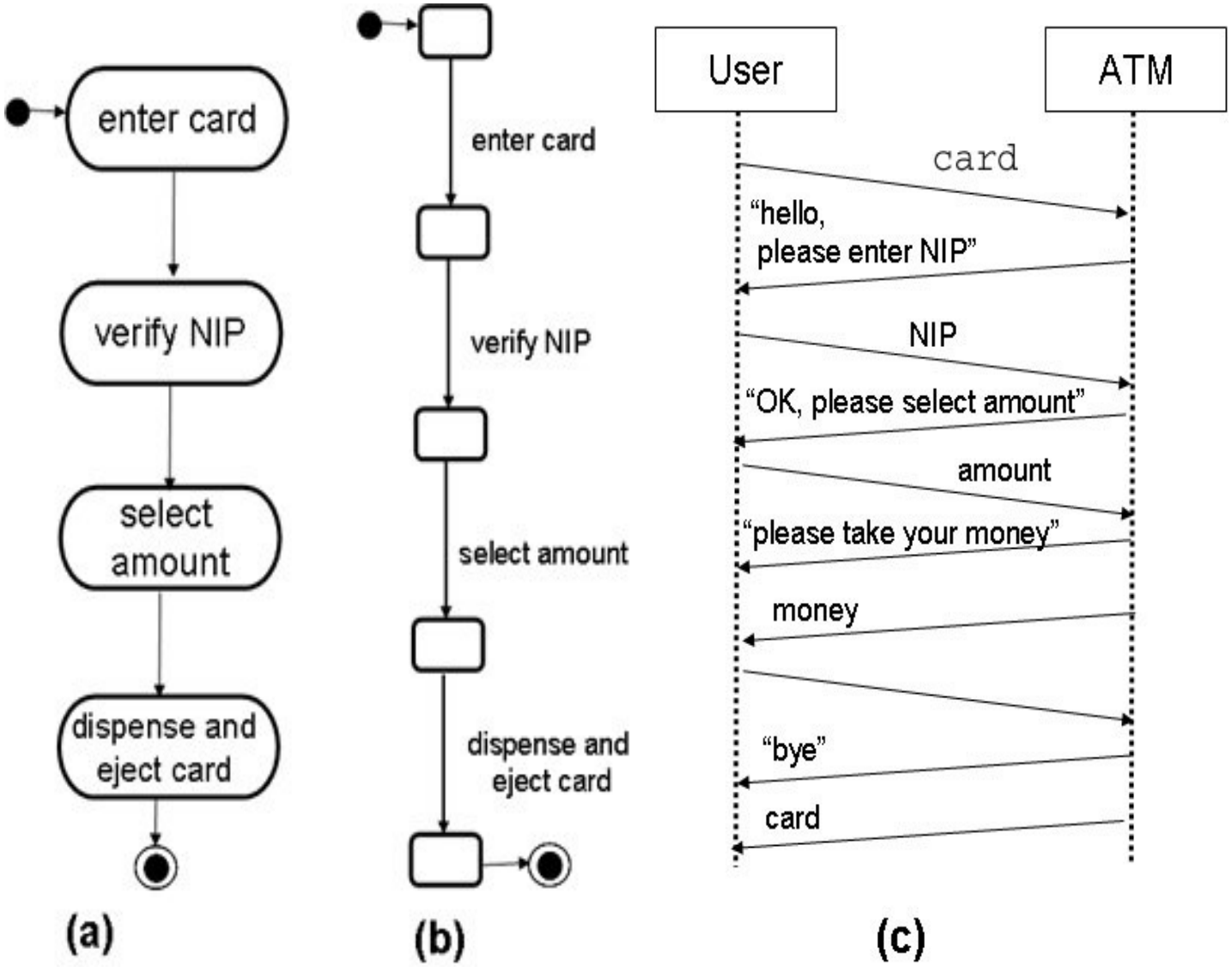
This is a common misunderstanding… Discuss in class…





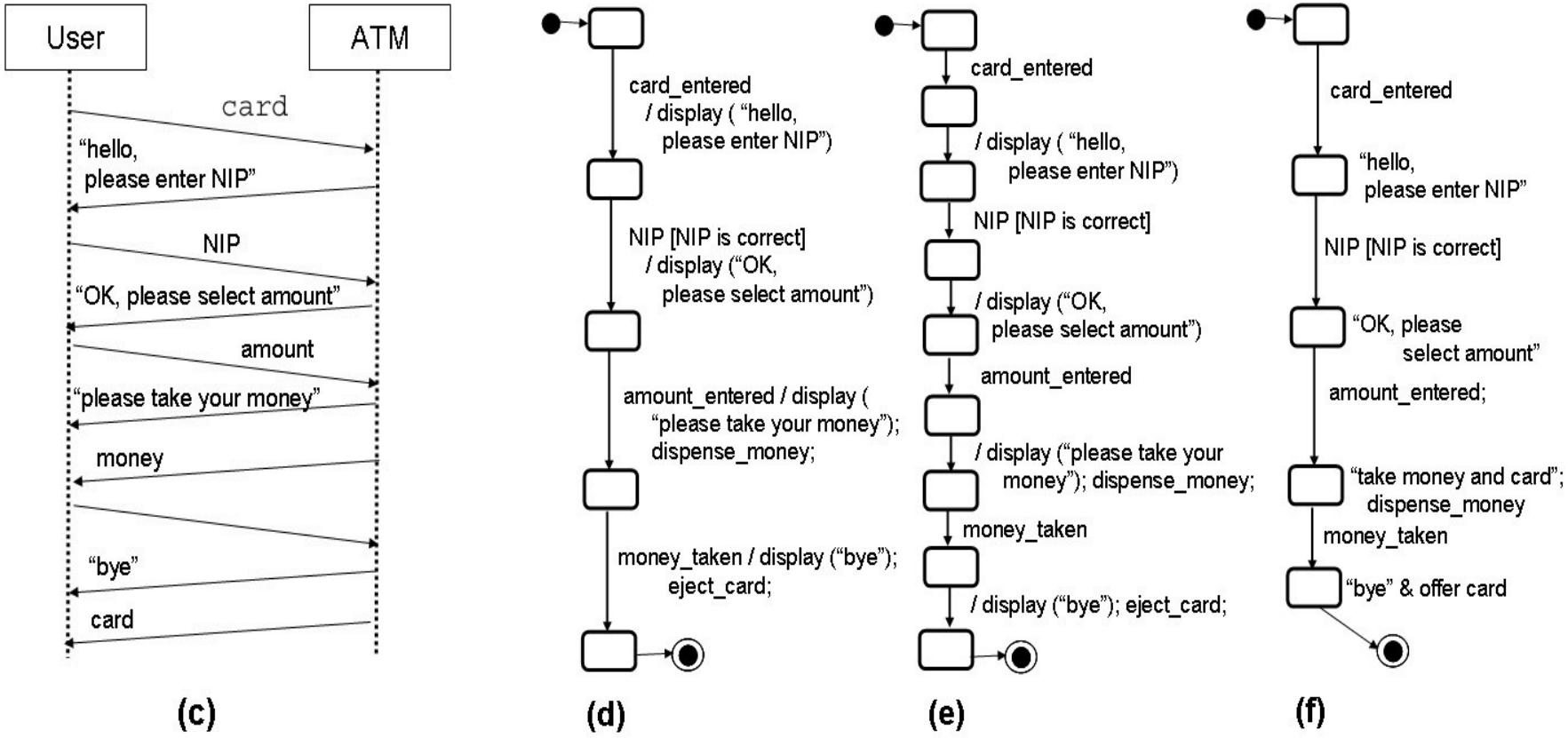
Different diagrams for the Same Problem

https://www.site.uottawa.ca/~bochmann/SEG-2106-2506/Notes/M1-2-StateMachines/index.html



Different diagrams with the same semantics

https://www.site.uottawa.ca/~bochmann/SEG-2106-2506/Notes/M1-2-StateMachines/index.html



Anyway, PLEASE be aware that this particular case is interesting to illustrate this discussion, BUT would be not a good example of a set of requirements to model as a State Machine, as we can understand it more objectivelly a sequence of events…

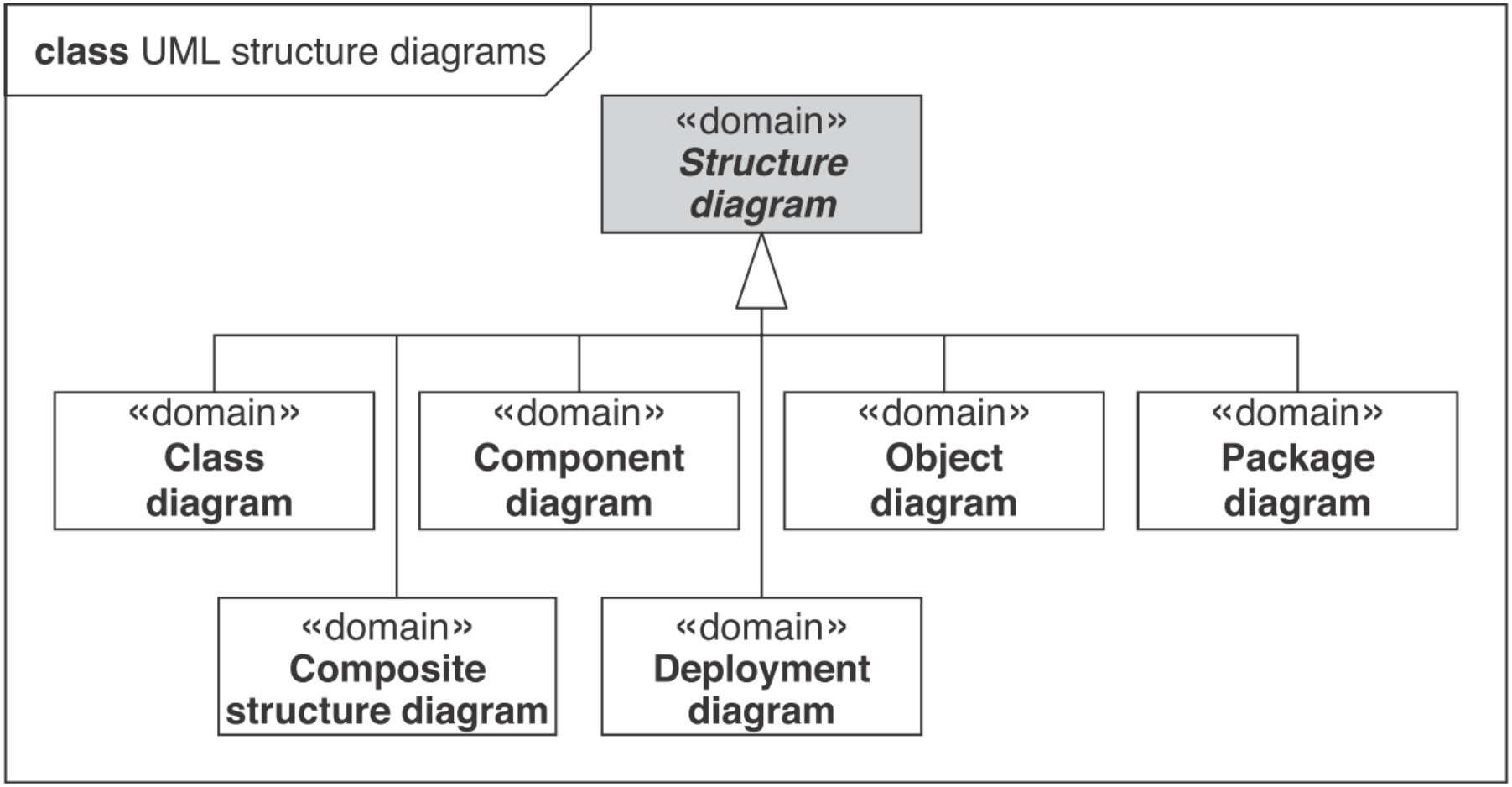
Different diagrams with the same semantics
https://www.site.uottawa.ca/~bochmann/SEG-2106-2506/Notes/M1-2-StateMachines/index.html

- *Figure (c) is not a state machine, but a sequence diagram showing the interactions between the ATM and its environment, the user, defining the behavior of the ATM in terms of message exchange between the ATM and its user. Here each activity of Figure (a) has been replace by an input message from the user and one or two output messages provided by the ATM in response. Many of the output messages are in fact displays of texts presented to the user. This model provides more details than Figures (a) and (b): It shows the initiatives for the different actions (mainly the sending of messages); it also shows more details about the user interface (text displayed during the different steps of the process).*
- *Based on Figure (c), the UML State machine of Figure (d) has been developed. It is of type "finite state machine" with inputs and outputs. It represents the behavior of the ATM and includes the following changes in respect to Figure (c):*
  - *The names of the messages from the user to the ATM have been replaced by events with names that better reflect the nature of the interactions with the user. E.g."card" message has been replaced by the event "card_entered".*
  - *The second transition includes a condition that is assumed to be valid for the subsequent behavior; written "[NIP is correct]".*
  - *Some of the message sending operations have been replaced by the name of local actions that better reflect the nature of these interactions with the user. E.g. the sending of the "money" message is replace by calling the local procedure "dispense_money()".*
- *Diagram (e) represents the same behavior as diagram (d) - it is given in the form of an "Input-Output Automata" (each transition has either a single input or a single output interaction). The output transitions are spontaneous.*
- *Diagram (f) shows the same behavior in the form of a Moore machine (output is associated with the state of the machine, not with a transition).*

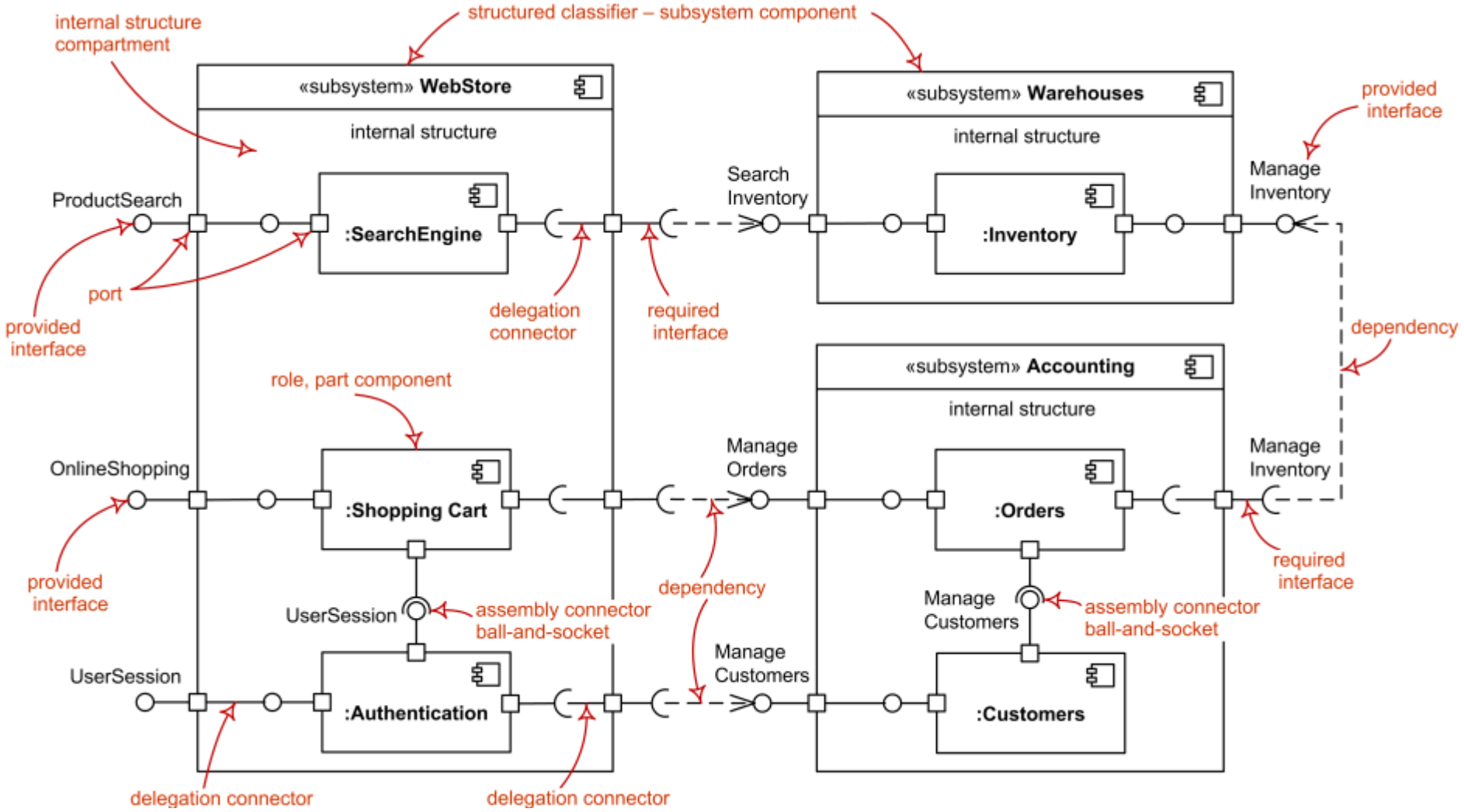
DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

More on UML structure...

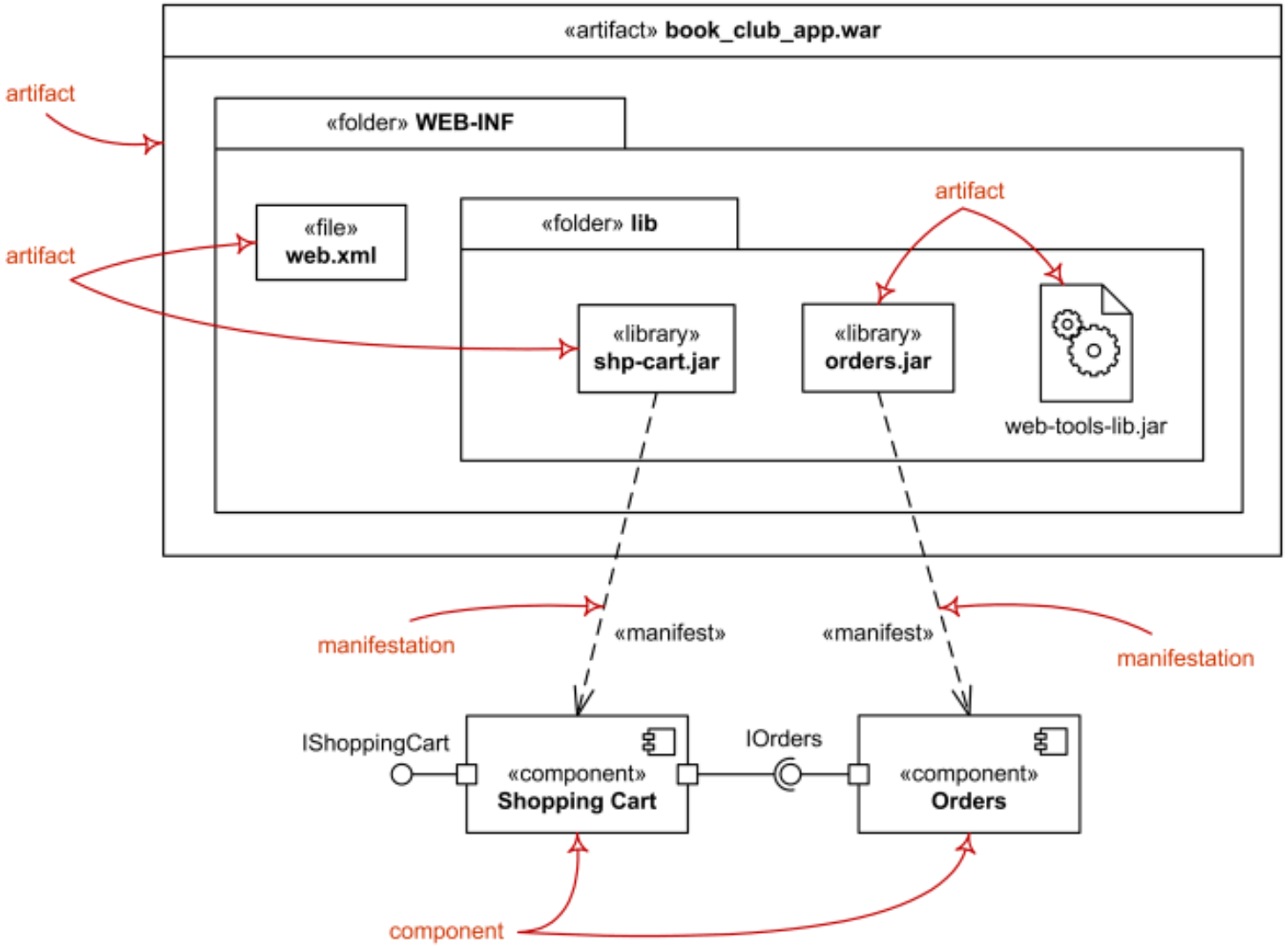
Remaining UML 2.x structure diagrams (informative)



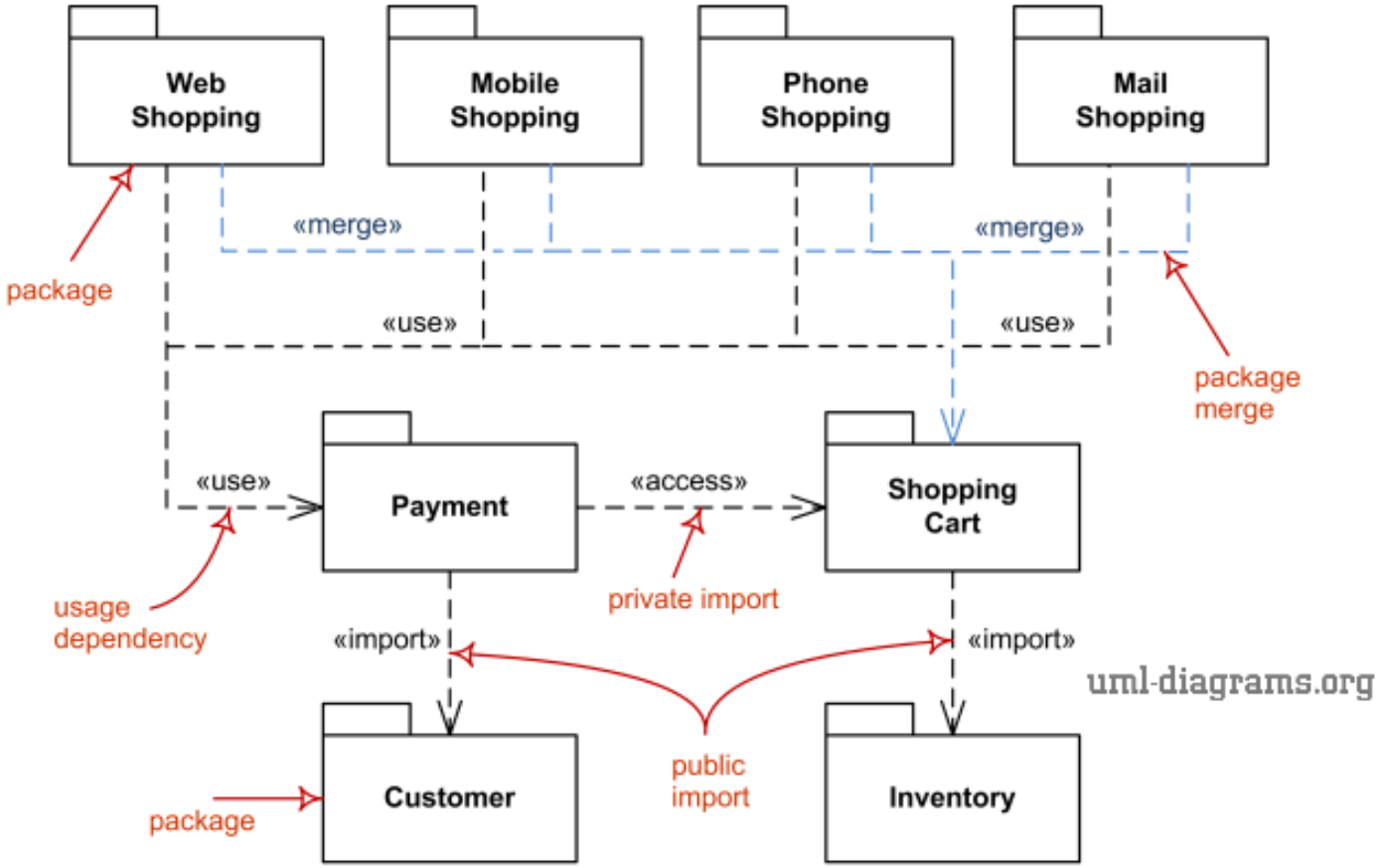
https://www.uml-diagrams.org/component-diagrams.html



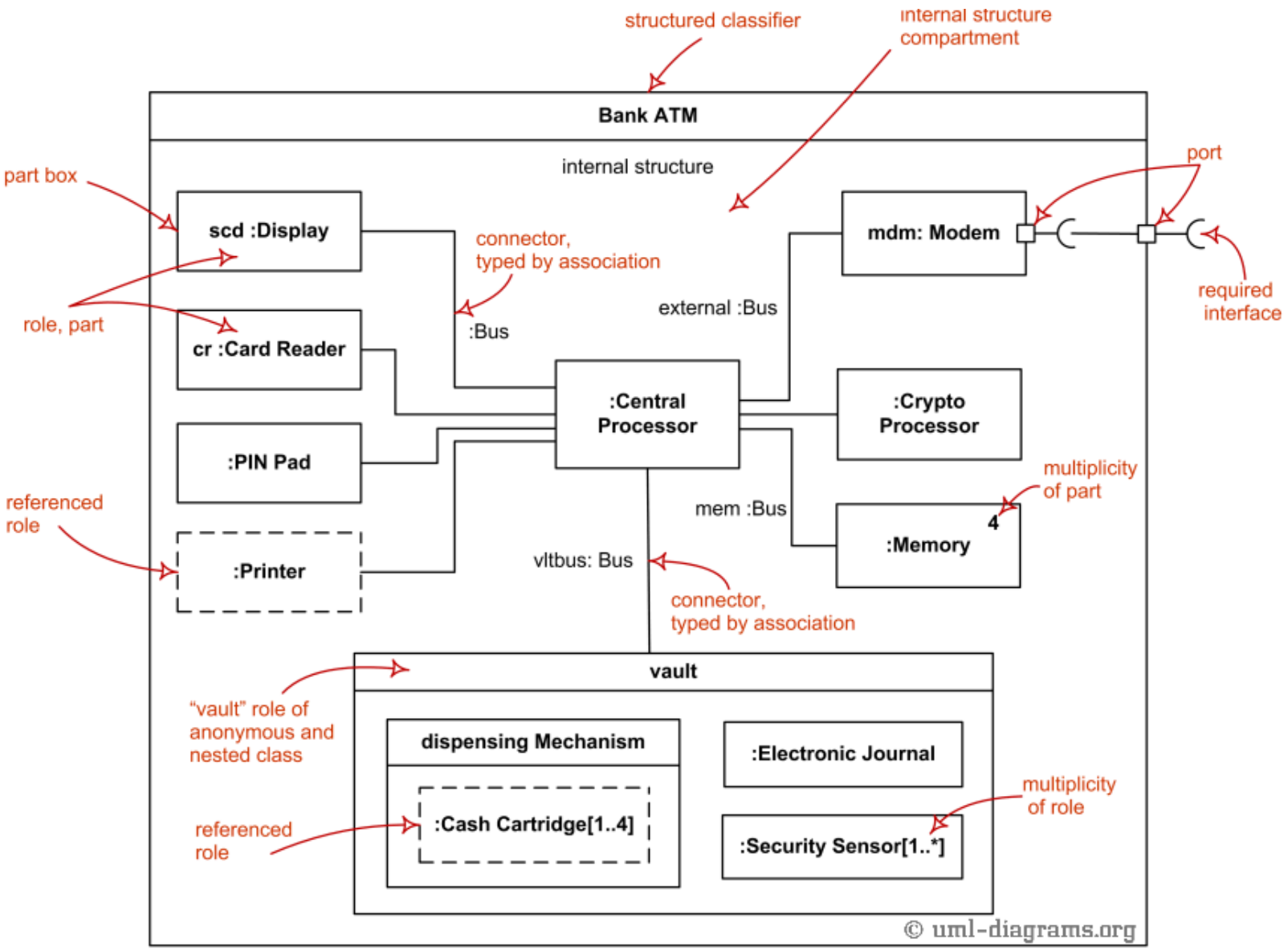
https://www.uml-diagrams.org/deployment-diagrams-overview.html



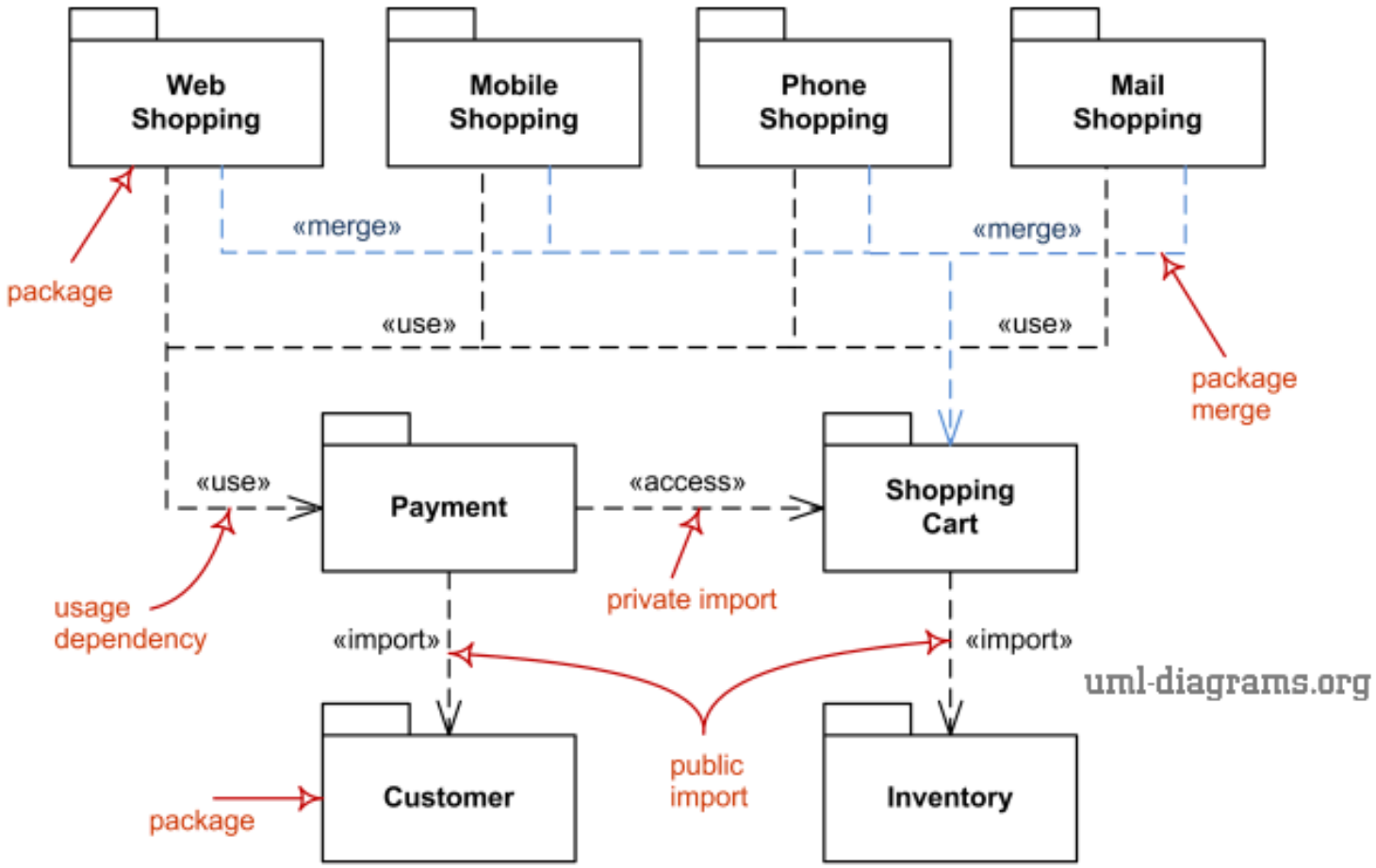
https://www.uml-diagrams.org/package-diagrams-overview.html



https://www.uml-diagrams.org/composite-structure-diagrams.html



https://www.uml-diagrams.org/package-diagrams-overview.html



And… Have fun!!!
… cartoons at:
https://www.modernanalyst.com/Resources/BusinessAnalystHumor.aspx