TB141Ic – ICT System Engineering and Rapid Prototyping

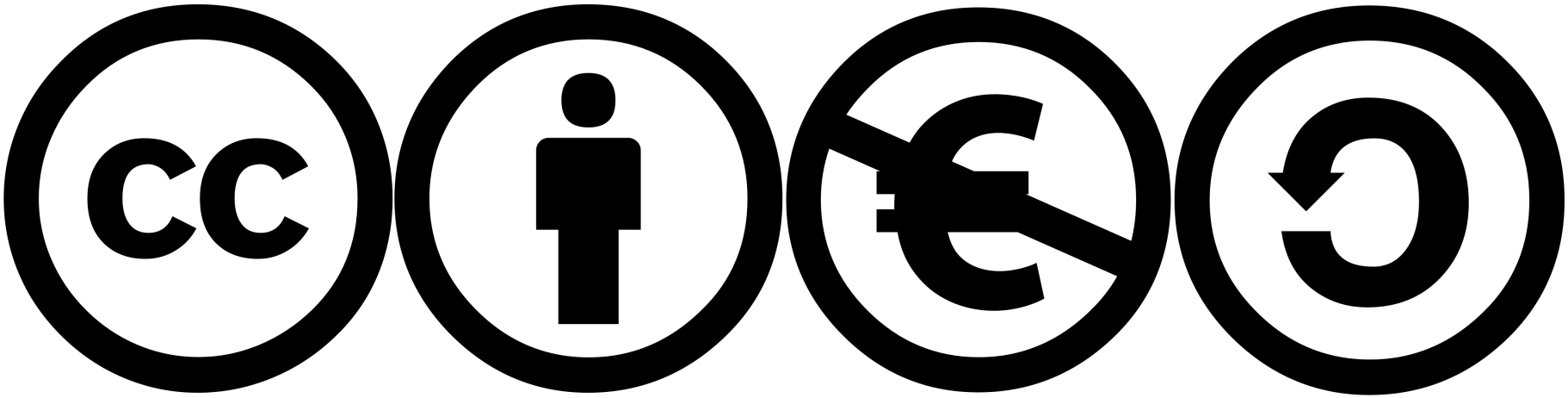
System modeling with UML - Class and Sequence diagrams

Jacopo De Stefani

Delft University of Technology, The Netherlands

27/02/2023

Licensing information



Learning objectives and related literature

Learning objectives

- Model using Class Diagrams and Sequence Diagrams
  - Recall the basic components
  - Read an existing diagram
  - Develop a novel diagram for a specific purpose

Related literature

- Chapter 5 [1]
- Online references - https://www.uml-diagrams.org/

[1] **sommerville2011software**

UML Views - Diagrams mapping

Use case view

Shows the functionality of the system as perceived by external actors.

Diagrams	Used by
Use case Activity	Customers Designers Developers Testers

Component View

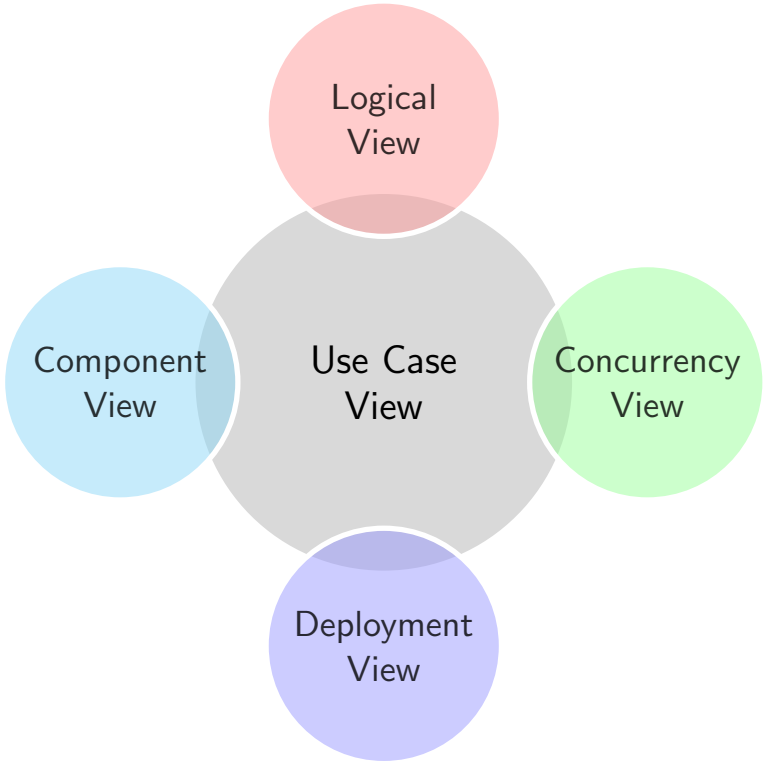
Shows the organization of the code components and their dependencies.

Diagrams	Used by
Component	Developers

Deployment View

Shows the deployment of the system into the physical architecture.

Diagrams	Used by
Deployment	Developers Testers System Integrators



Logical view

Shows how the functionality of the system is designed / provided.

Diagrams	Used by
Class State Sequence Collaboration Activity	Developers Designers

Concurrency view

Addresses the problems with communication and synchronization for a concurrent system.

Diagrams	Used by
State Sequence Collaboration Activity Deployment Components	Developers System Integrators Testers

UML Diagrams

- Use-Case diagram
- Class diagram
- Object diagram
- State diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Component diagram
- Deployment diagram

UML Diagrams

- **Use-Case diagram**
- **Class diagram**
- Object diagram
- State diagram
- **Sequence diagram**
- Collaboration diagram
- **Activity diagram**
- Component diagram
- Deployment diagram

UML Diagrams

- **Use-Case diagram**
- **Class diagram**
- Object diagram
- State diagram
- **Sequence diagram**
- Collaboration diagram
- **Activity diagram**
- Component diagram
- Deployment diagram

My approach to UML

Focus on the (Unified Modeling) **Language** aspect:

- Notation ≡ Grammar
- UML Understanding ≡ Reading
- UML Modeling ≡ Writing/Speaking

System perspectives

- **External/Context perspective:** where you model the context or environment of the system.
- **Interaction perspective:** where you model the interactions between a system and its environment, or between the components of a system.
- **Structural perspective:** where you model the organization of a system or the structure of the data that is processed by the system.
- **Behavioral perspective:** where you model the dynamic behavior of the system and how it responds to events.

Content Plan

- **Lecture 4:** Context and interaction perspective
- **Lecture 5:** Strucutral and behavioral perspective

Context models

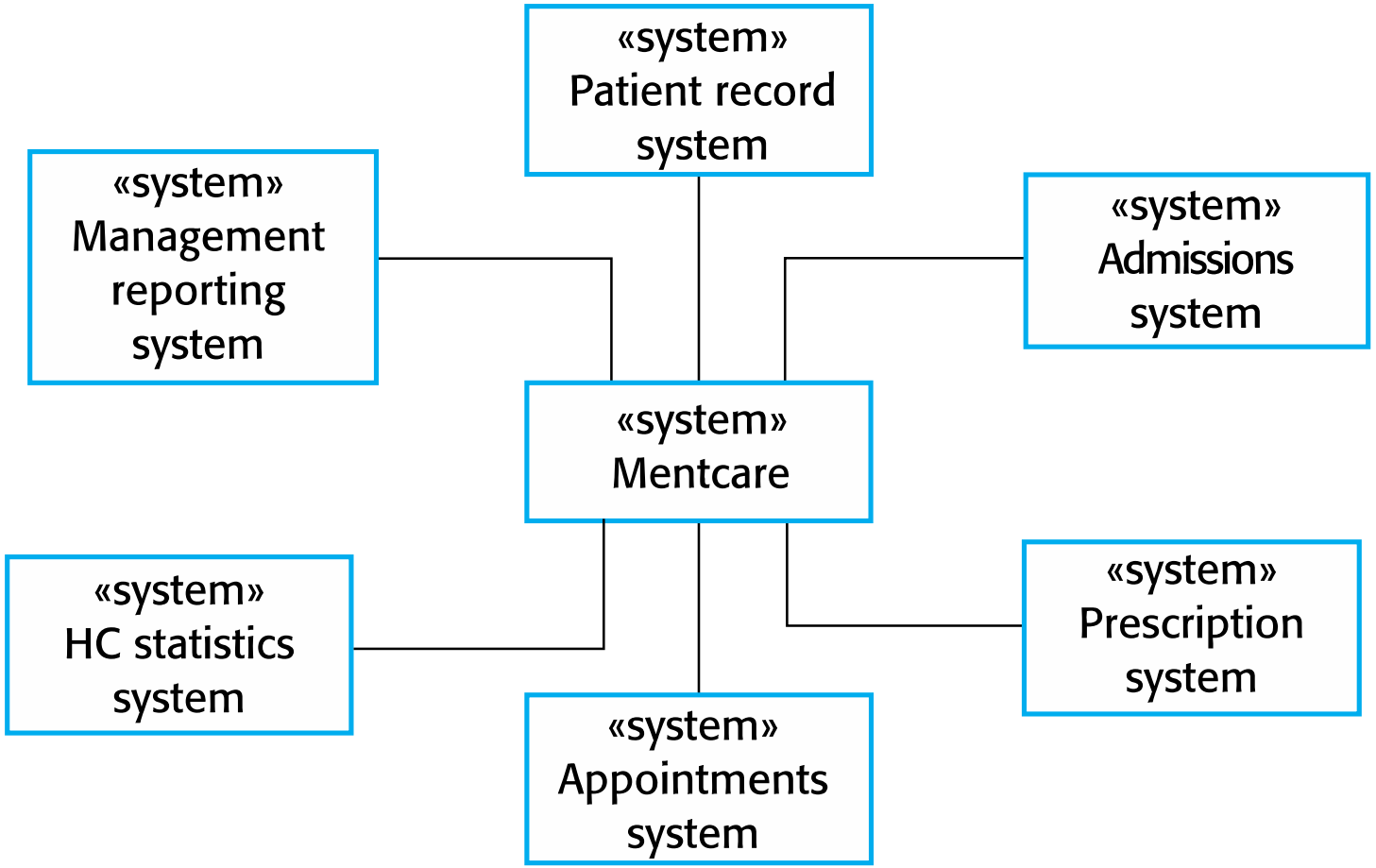
Context models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

System boundaries

- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

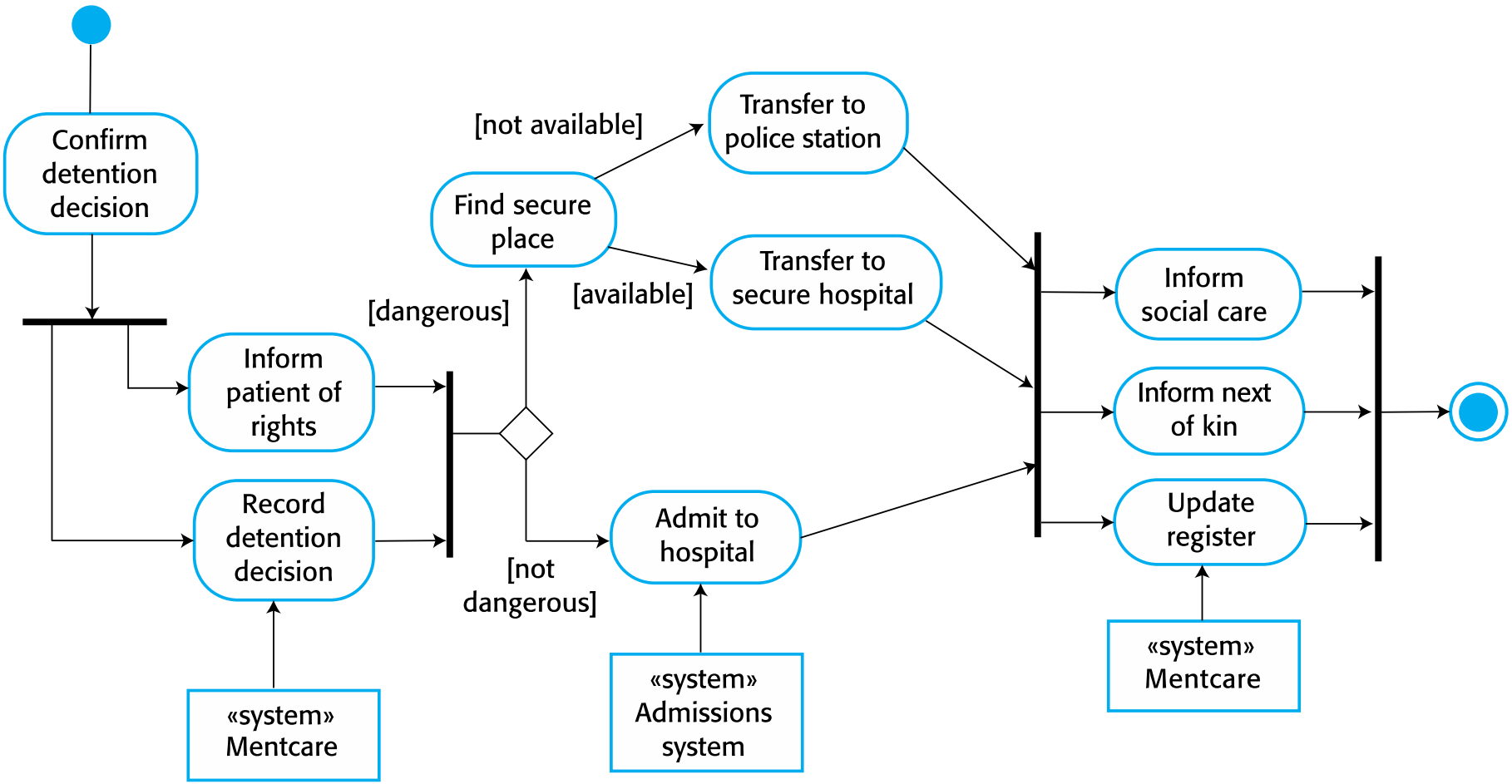
The context of the Mentcare system



Process perspective

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.

Process model of involuntary detention



Interaction models

Interaction models

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

Use case modeling

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagramatically to provide an overview of the use case and in a more detailed textual form.

Transfer-data use case

- A use case in the Mentcare system

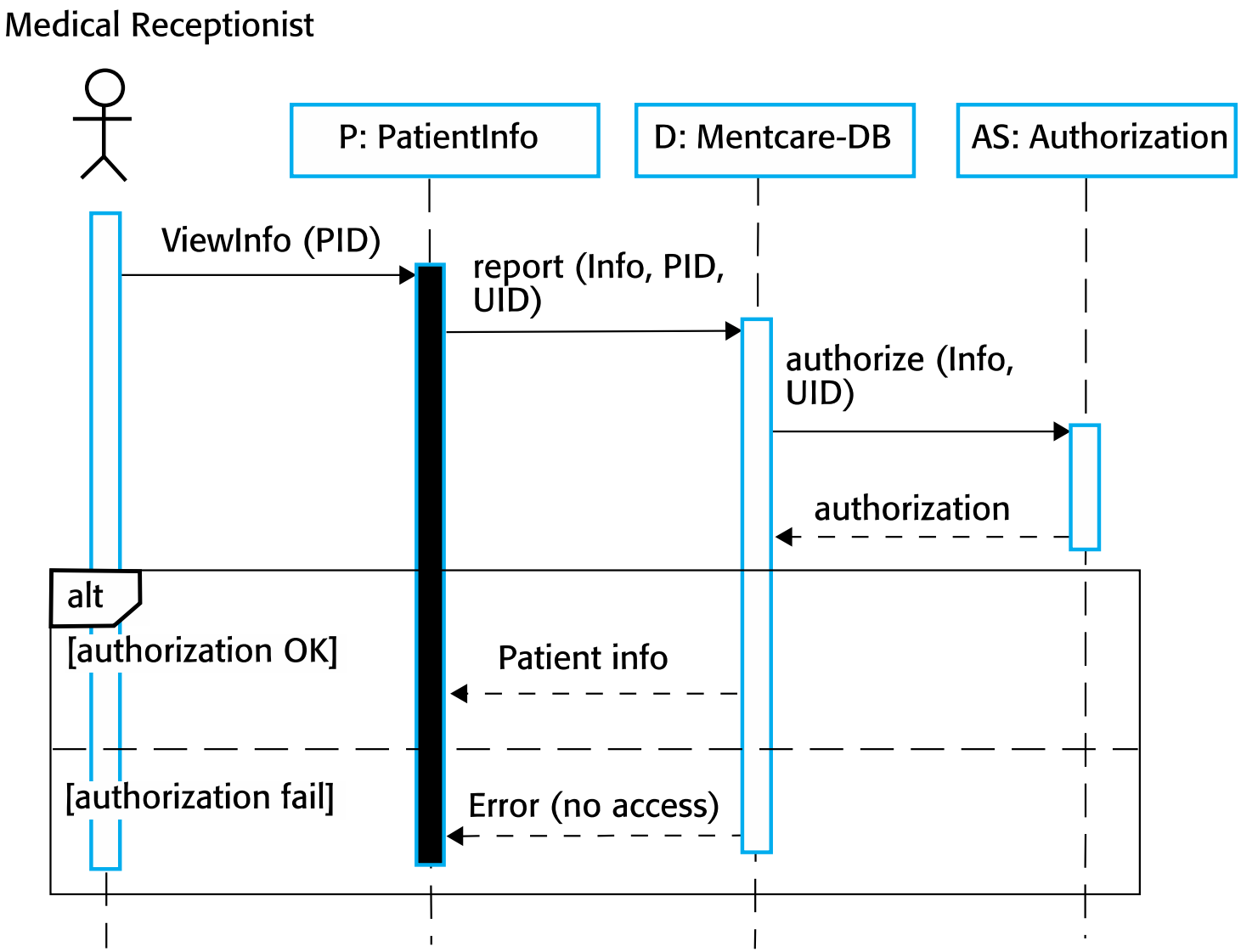
Tabular description of the 'Transfer data' use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcase system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Sequence diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence diagram for View patient information



Structural models

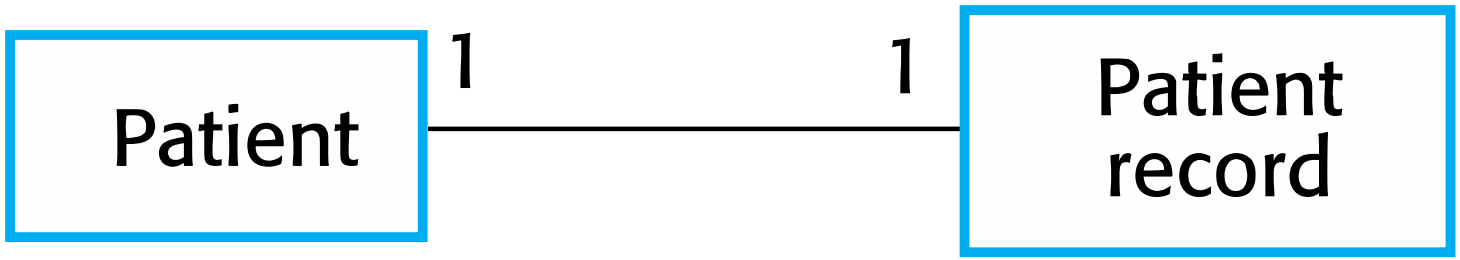
Structural models

- Structural models of software display the organization of a system in terms of the components that make up that system and their relationships.
- Structural models may be static models, which show the structure of the system design, or dynamic models, which show the organization of the system when it is executing.
- You create structural models of a system when you are discussing and designing the system architecture.

Class diagrams

- Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.
- An object class can be thought of as a general definition of one kind of system object.
- An association is a link between classes that indicates that there is some relationship between these classes.
- When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.

UML classes and association



Behavioral models

Behavioral models

- Behavioral models are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment.
- You can think of these stimuli as being of two types:
  - **Data:** Some data arrives that has to be processed by the system.
  - **Events:** Some event happens that triggers system processing. Events may have associated data, although this is not always the case.

Class Diagrams

UML Class Diagram

- Static model type
  - A view of the system in terms of classes (entities) and relationships
  - Focus on structural modeling, rather than interaction modeling
- Two main variants:
  - **Domain modeling:** High level, focused on entities and relationships
  - **Implementation modeling:** Low level, focused on data types, visibilities, implementation choices (abstract classes vs interfaces)
- **Classes**
- **Objects**

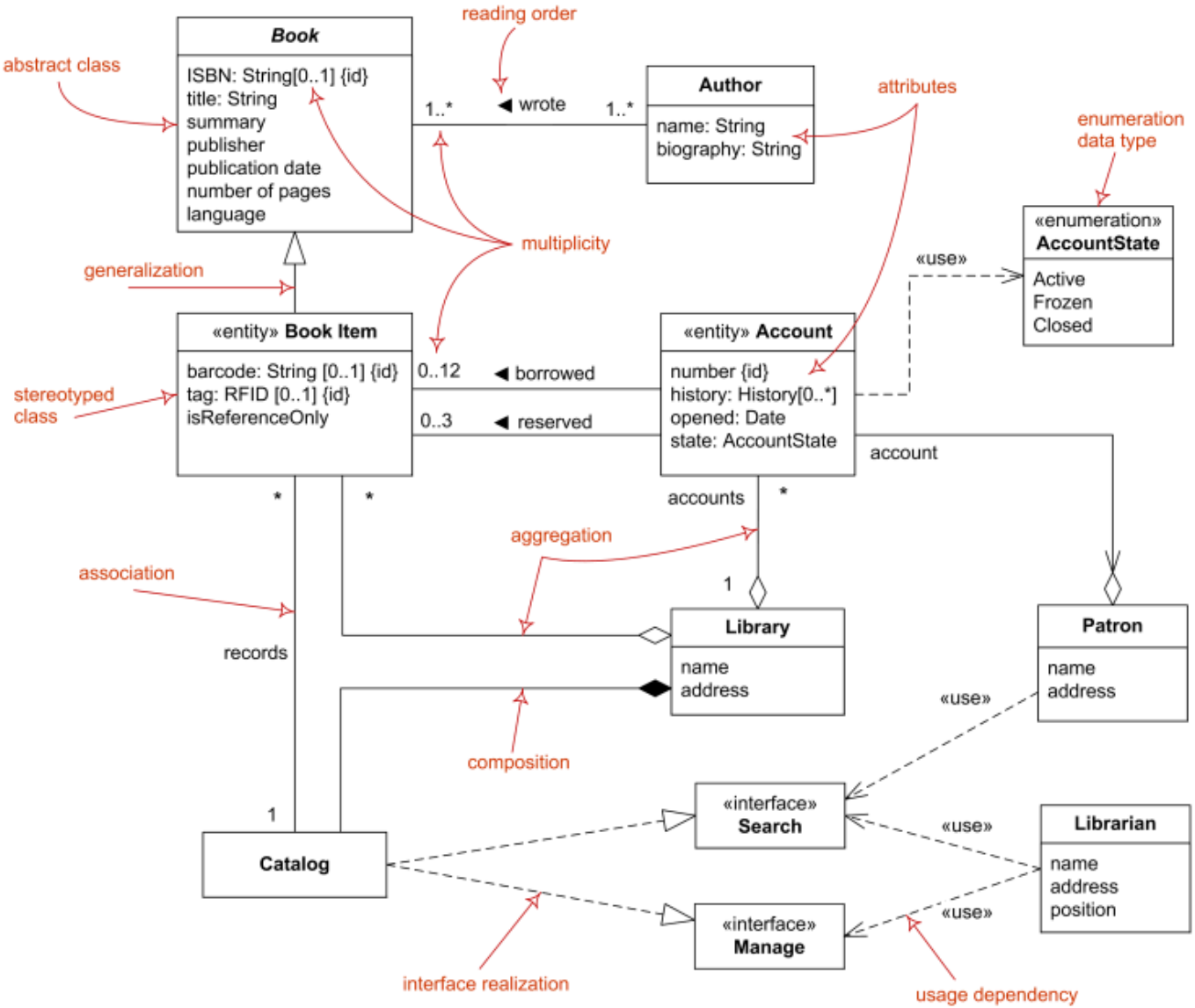
UML Class Diagram

- Static model type
  - A view of the system in terms of classes (entities) and relationships
  - Focus on structural modeling, rather than interaction modeling
- Two main variants:
  - **Domain modeling:** High level, focused on entities and relationships
  - **Implementation modeling:** Low level, focused on data types, visibilities, implementation choices (abstract classes vs interfaces)
- **Classes** → Blueprint/Recipe including common attributes **and** behavior
- **Objects**

UML Class Diagram

- Static model type
  - A view of the system in terms of classes (entities) and relationships
  - Focus on structural modeling, rather than interaction modeling
- Two main variants:
  - **Domain modeling:** High level, focused on entities and relationships
  - **Implementation modeling:** Low level, focused on data types, visibilities, implementation choices (abstract classes vs interfaces)
- **Classes** → Blueprint/Recipe including common attributes **and** behavior
- **Objects** → Concrete realization (instance) of a class including specific values for attributes.

Class diagram - Overview



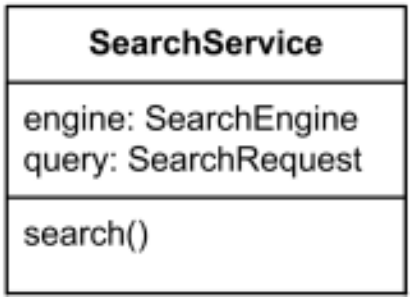
Class Diagram - Elements

Class - No compartments

A class is a classifier which describes a set of objects that share the same:

- features
- constraints
- semantics (meaning).

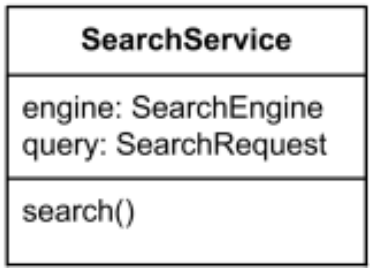
A class is shown as a solid-outline rectangle containing the class name, and optionally with compartments separated by horizontal lines containing features or other members of the classifier.



Class - With compartments

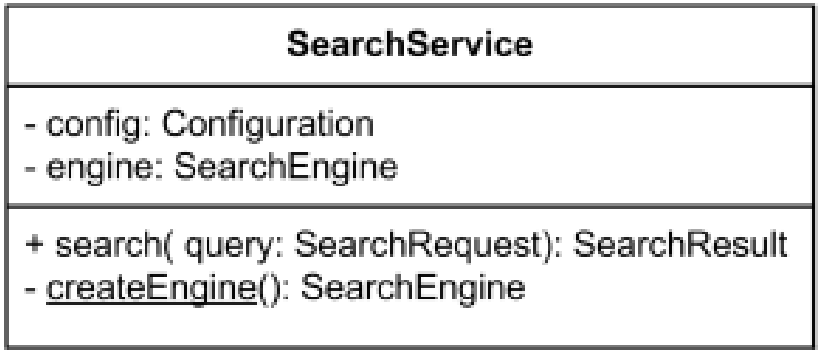
When class is shown with three compartments, the middle compartment holds a list of attributes (information that the class stores) and the bottom compartment holds a list of operations (or methods). Attributes and operations should be left justified in plain face, with the first letter of the names in lower case.

Class Diagram - Design vs Implementation



Class - Design level

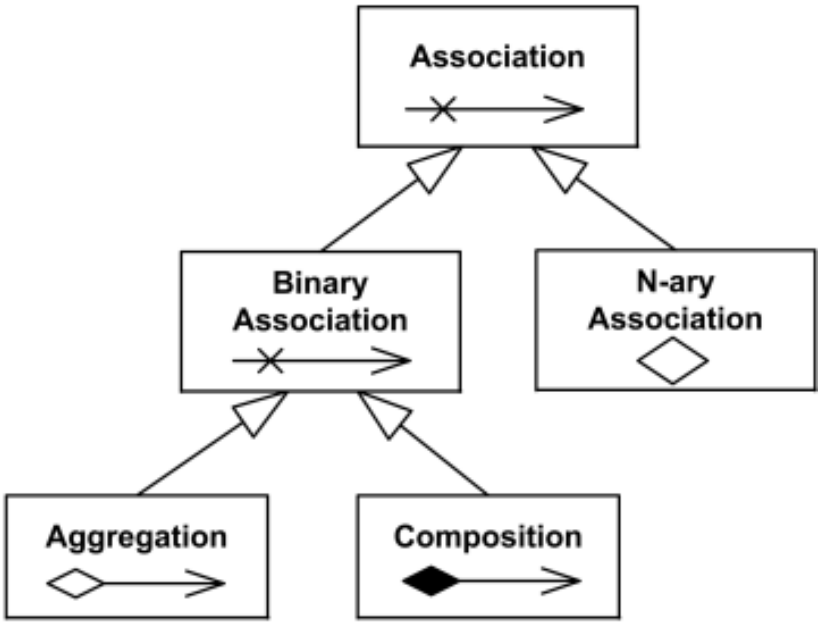
At the design level, the class contains general specifications of the attributes and operations of the class.

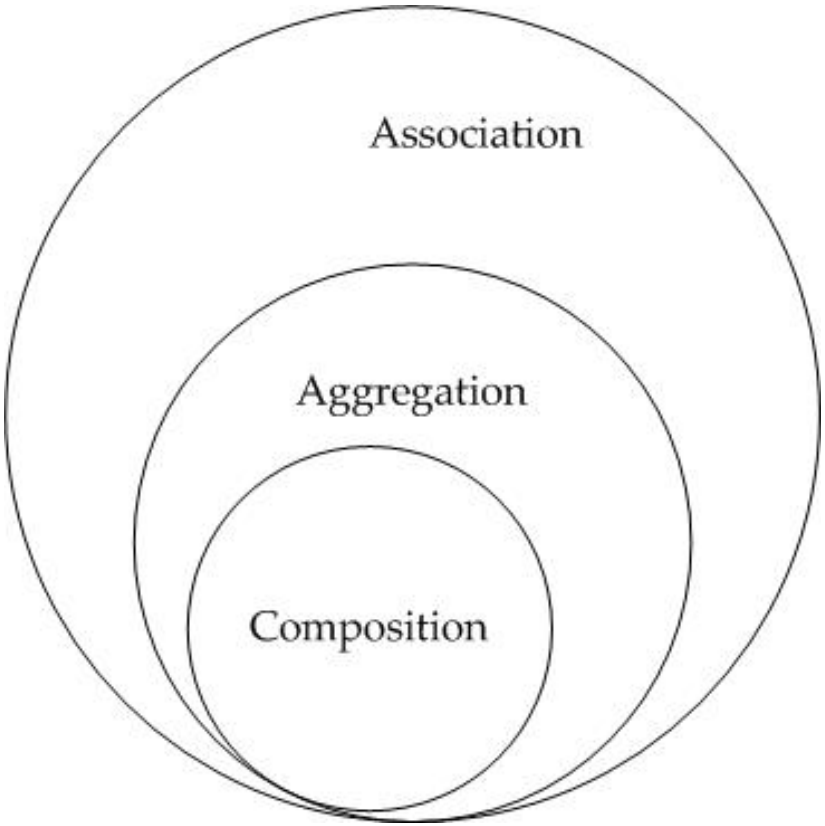


Class - Implementation level

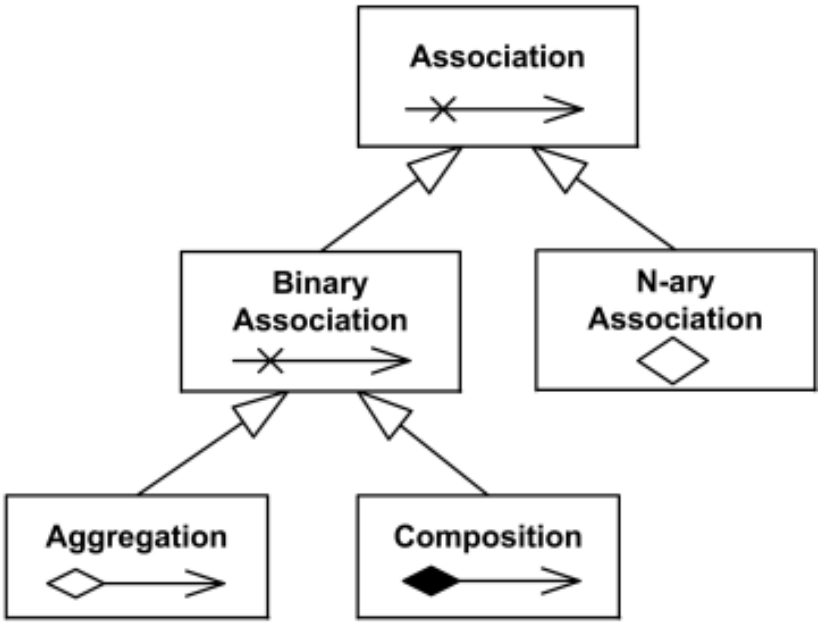
At the implementation level, the class contains precise specifications of the data type of the attributes, their visibility and the details of input operations of the class.

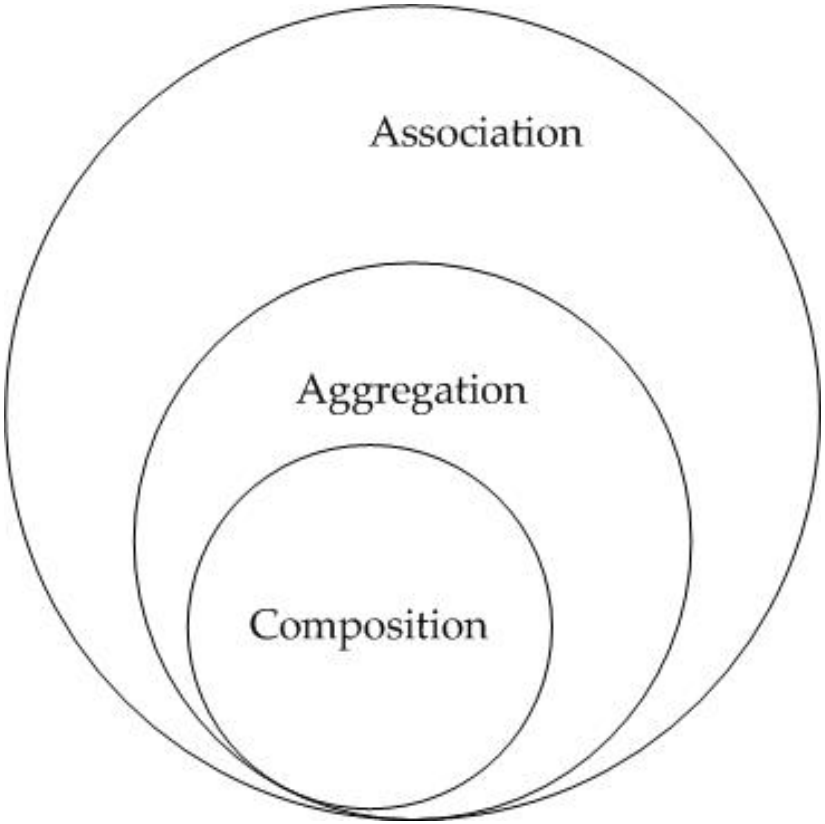
Class diagram - Relationship Overview





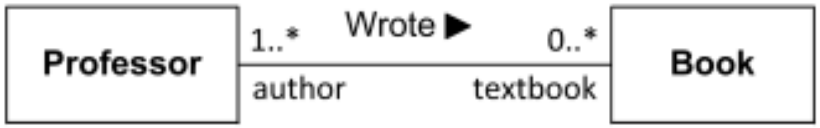
Class diagram - Relationship Overview





What is the purpose of the different associations?

Class diagram - Relationships - Association



Binary Association

Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation.
Binary association relates two typed instances. It is normally rendered as a solid line connecting two classifiers, or a solid line connecting a single classifier to itself (the two ends are distinct). The line may consist of one or more connected segments.

A small solid triangle could be placed next to or in place of the name of binary association (drawn as a solid line) to show the order of the ends of the association. The arrow points along the line in the direction of the last end in the order of the association ends. This notation also indicates that the association is to be read from the first end to the last end.

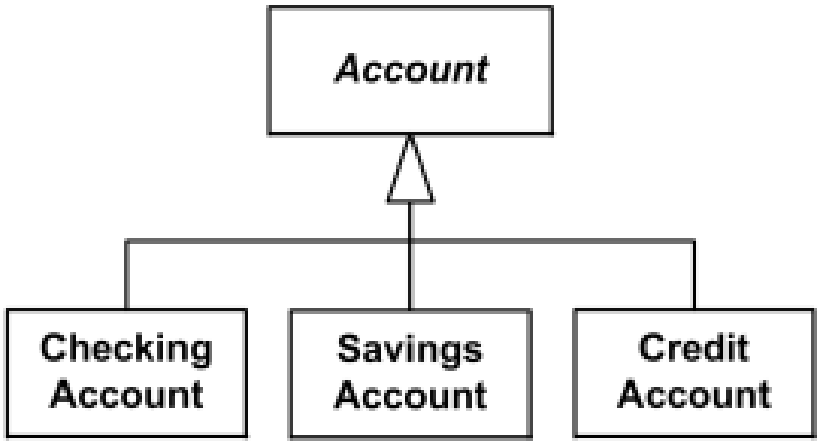
Multiplicity

Multiplicity is a definition of an inclusive interval of non-negative integers to specify the allowable number of instances of described element.

Some typical examples of multiplicity bounds:

0	Collection must be empty
1	Exactly one instance
5	Exactly 5 instances
$*$	Zero or more instances
0..1	No instances or one instance
1..1	Exactly one instance
0..$*$	Zero or more instances
1..$*$	At least one instance
$m..n$	At least m but no more than n instances

Class diagram - Relationships - Generalization



Generalization

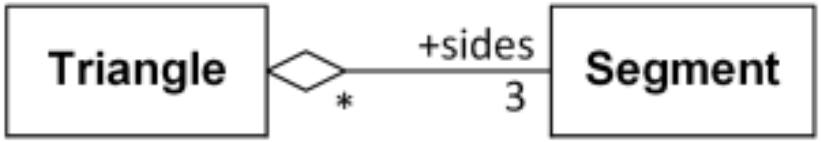
A generalization is a binary taxonomic (i.e. related to classification) directed relationship between a more general classifier (superclass) and a more specific classifier (subclass).

Each instance of the specific classifier is also an indirect instance of the general classifier, so that we can say "Patient is a Person", "Savings account is an Account", etc. Because of this, generalization relationship is also informally called "Is A" relationship.

Semantics

- Attributes and methods from the superclass are inherited in the subclass ⇒ They are included in the subclass implicitly.
- Generalization doesn't have multiplicity
- In implementation, notion of single versus multiple inheritance.

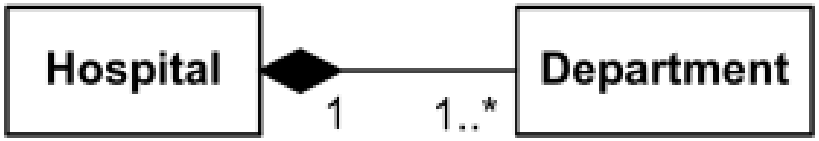
Class diagram - Relationships - Aggregation/Composition



Aggregation - Shared Aggregation

Shared aggregation (aggregation) is a binary association between a property and one or more composite objects which group together a set of instances. It is a "weak" form of aggregation when part instance is independent of the composite. Shared aggregation has the following characteristics:

- it is binary association,
- it is asymmetric - only one end of association can be an aggregation,
- it is transitive - aggregation links should form a directed, acyclic graph, so that no composite instance could be indirect part of itself,
- shared part could be included in several composites, and if some or all of the composites are deleted, shared part may still exist.



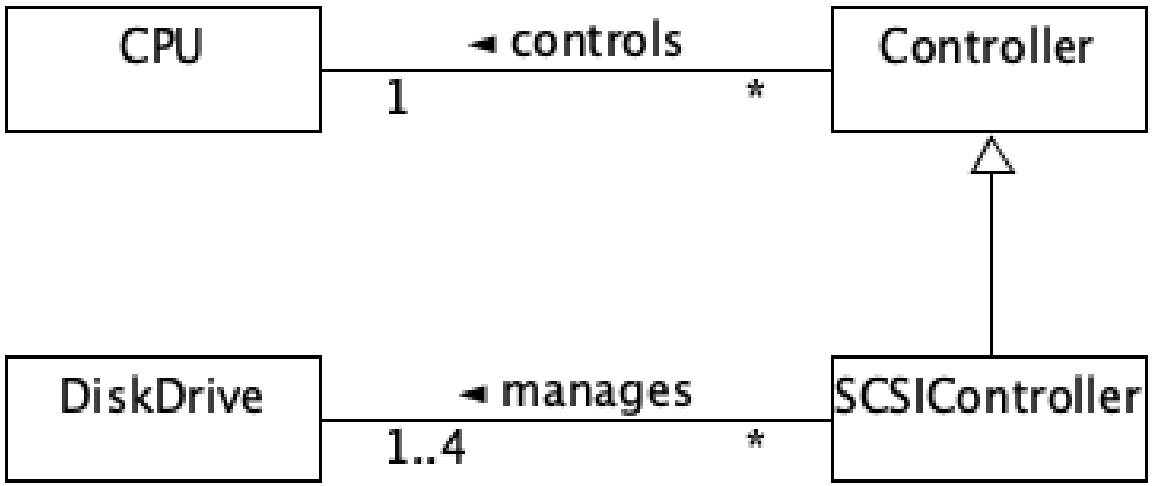
Composition - Composite aggregation

Composite aggregation (composition) is a "strong" form of aggregation with the following characteristics:

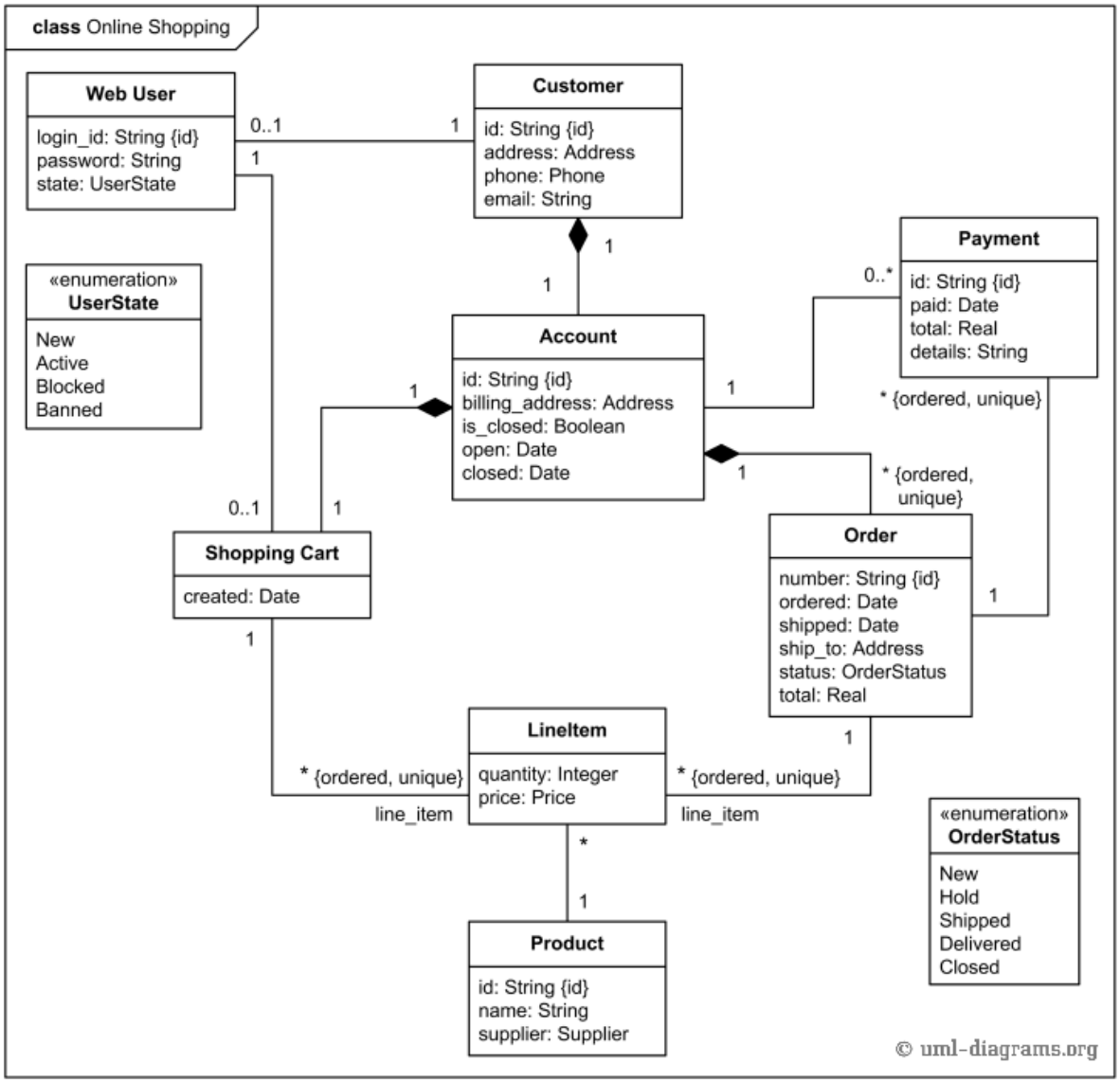
- it is binary association,
- it is a whole/part relationship,
- a part could be included in at most one composite (whole) at a time, and
- if a composite (whole) is deleted, all of its composite parts are "normally" deleted with it.

Note, that UML does not define how, when and specific order in which parts of the composite are created. Also, in some cases a part can be removed from a composite before the composite is deleted, and so is not necessarily deleted as part of the composite.

Class diagram - Example - Reading - 1



Class diagram - Example - Reading - 2



Class diagram - Example - Writing - 1

Exercise 1

You are required to model a system managing online bookings through airline companies. Each airline offers some flights and its characterized by an identifier. Airline pilots are hired by the airline, and each pilot has a different experience level: 1 being the lower while 3 being the higher. Each airline also own different types of airplanes. Each airplane could be in a working state or under repairs. Moreover, at any moment in time, an airplane is either flying or resting on the ground. Each specific type of airplane has a specific number of pilots required to fly it, with different roles (for instance capitain, co-pilot and navigator). On each plane though, there is at least one capitain, one co-pilot and a capitain must be a Level 3 Pilot. Each flight has an identifier, a departure airport and an arrival airport, as well as a departure time and arrival time. Each airport is characterized by a unique identifier. Each flight is performed using a specific type of airplane, driven by a pilot and a co-pilot.

Instructions

- Read the case (2-3 minutes)
- Participative design (7-10 minutes)
- Wrap-up (3-5 minutes)

Purpose: Experiment with modeling
Outcome: UML Class Diagram

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship
   - **Generalization**
   - **Aggregation**
   - **Composition**
   - **Binary association**
4. Identify specifiers to determine multiplicities and or direction of the associations.

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship → What are the relationships between entities? (Verbs)
   - **Generalization**
   - **Aggregation**
   - **Composition**
   - **Binary association**
4. Identify specifiers to determine multiplicities and or direction of the associations.

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship → What are the relationships between entities? (Verbs)
   - **Generalization** → Is there some sort of specialization between entities? Does two entities share some properties but one has some differences from the other?
   - **Aggregation**
   - **Composition**
   - **Binary association**
4. Identify specifiers to determine multiplicities and or direction of the associations.

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship → What are the relationships between entities? (Verbs)
   - **Generalization** → Is there some sort of specialization between entities? Does two entities share some properties but one has some differences from the other?
   - **Aggregation** → Is an entity a part of another entity?
   - **Composition**
   - **Binary association**
4. Identify specifiers to determine multiplicities and or direction of the associations.

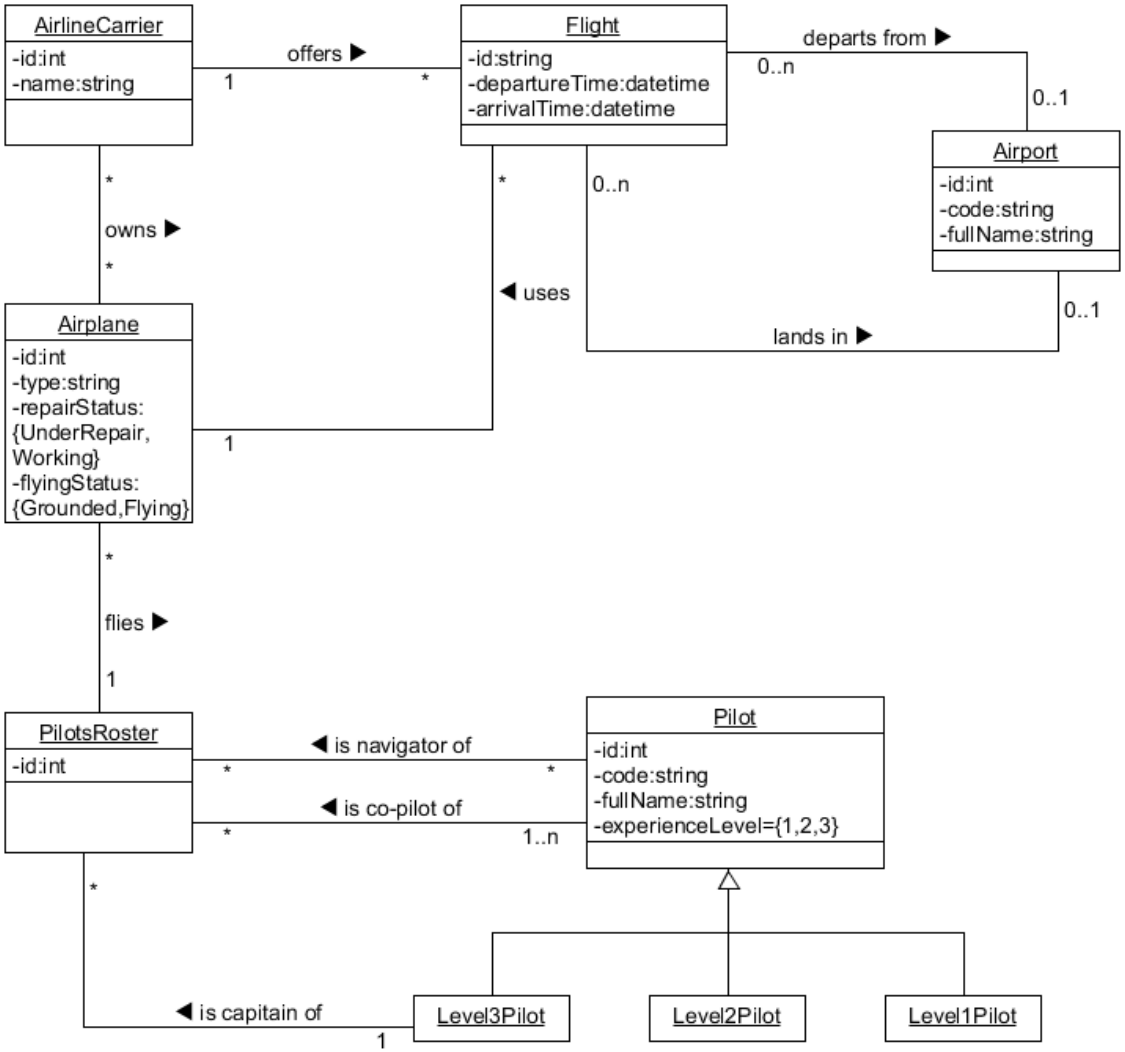
How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship → What are the relationships between entities? (Verbs)
   - **Generalization** → Is there some sort of specialization between entities? Does two entities share some properties but one has some differences from the other?
   - **Aggregation** → Is an entity a part of another entity?
   - **Composition** → Is an entity a part of another entity **and** they have the same life cycle?
   - **Binary association**
4. Identify specifiers to determine multiplicities and or direction of the associations.

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify relationship → What are the relationships between entities? (Verbs)
   - **Generalization** → Is there some sort of specialization between entities? Does two entities share some properties but one has some differences from the other?
   - **Aggregation** → Is an entity a part of another entity?
   - **Composition** → Is an entity a part of another entity **and** they have the same life cycle?
   - **Binary association** → Is there an association but with none of the above properties?
4. Identify specifiers to determine multiplicities and or direction of the associations.

Class diagram - Example - Writing 1 - Possible solution



Class diagram - Example - Writing - 2

Exercise 2

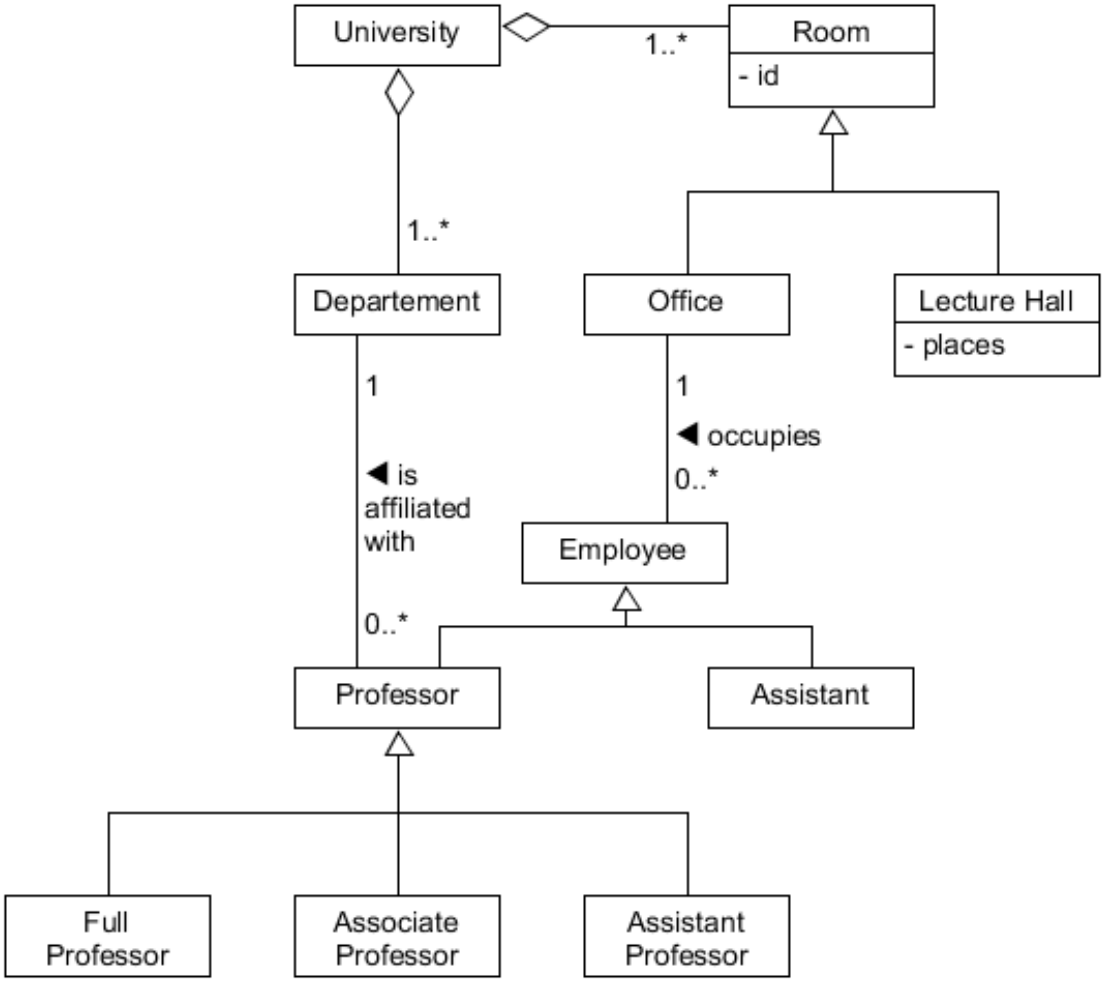
In a university, different types of rooms exists: offices and lecture halls. Both offices and lecture halls have a unique identification number and, in addition, a lecture hall also has a specific number of seats. The university is organized in different departments. Every employee of the university has a unique identifier and it is either a professor or a regular employee. Professor are organized into assistant professor, associate professors or full professor, and are part of exactly one depatement. Each employee works in one or more offices.

Instructions

- Please do try it at home!

Purpose: Experiment with modeling
Outcome: UML Class Diagram

Class diagram - Example - Writing 2 - Possible solution

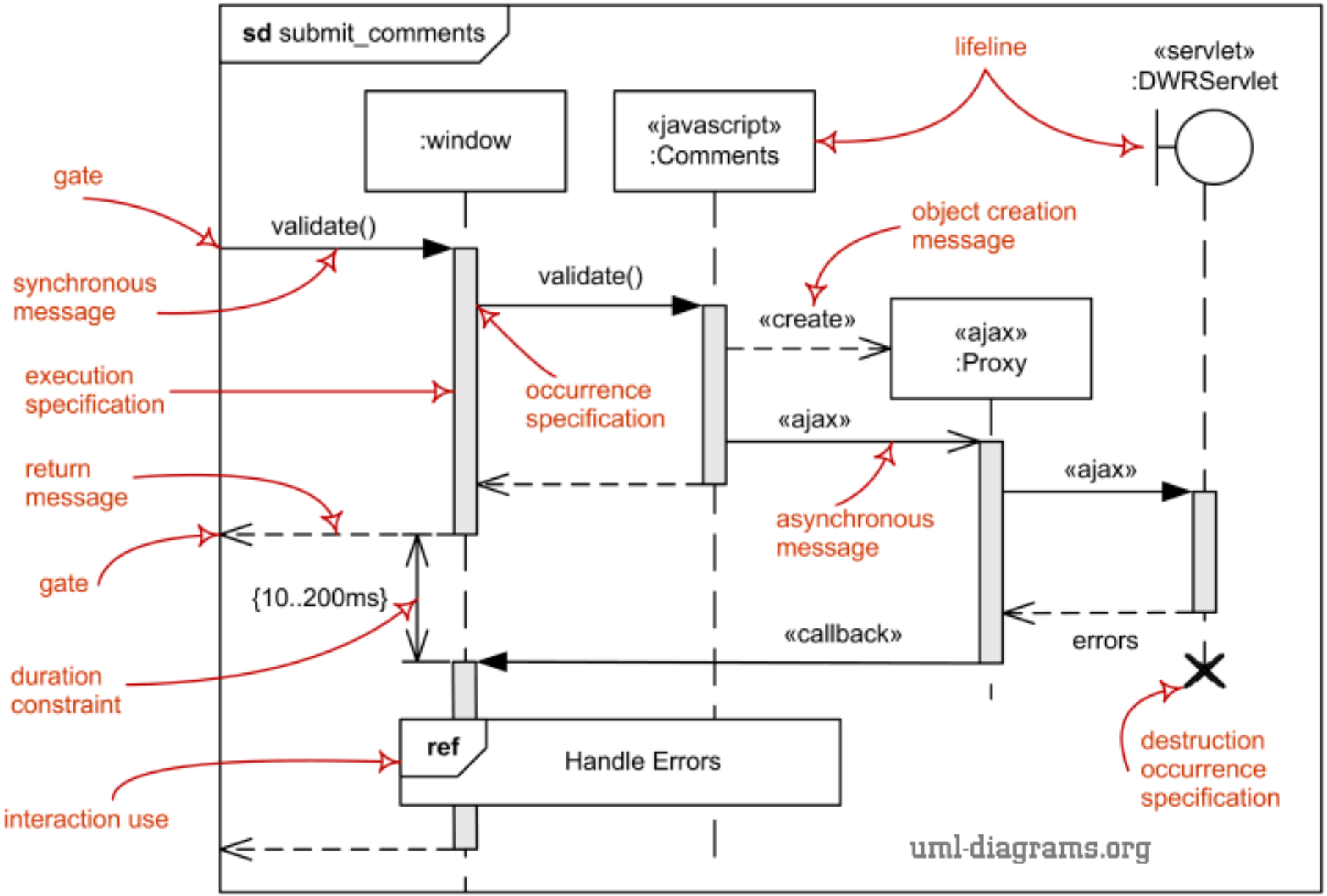


Sequence Diagrams

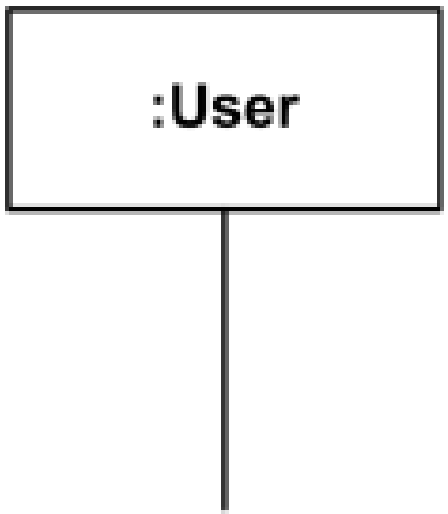
Sequence Diagrams

- Dynamic model type
  - A view of the system in terms of interactions, both within the system and between the system and external actors
  - Focus on flow of events/objects/activities
- Allows to specify sequence/parallelism
- Support the visualization of information sharing across components and synchronous/asynchronous interactions

Sequence diagram - Overview



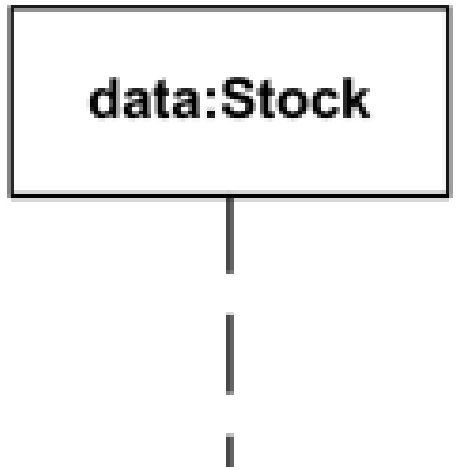
Sequence diagram - Elements



Lifeline - No name

Lifeline is a named element which represents an individual participant in the interaction. While parts and structural features may have multiplicity greater than 1, lifelines represent only one interacting entity.

A lifeline is shown using a symbol that consists of a rectangle forming its "head" followed by a vertical line (which may be dashed) that represents the lifetime of the participant.

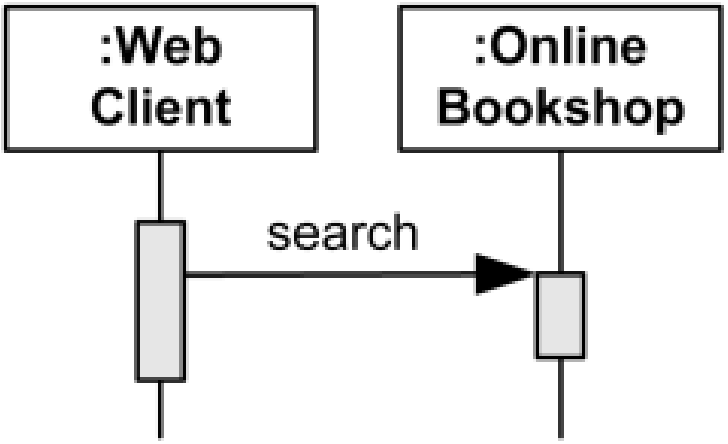


Lifeline - Named

While an unnamed lifeline refers to a generic class element, a named lifeline refers to a specific instance of the class indicated in the rectangle.

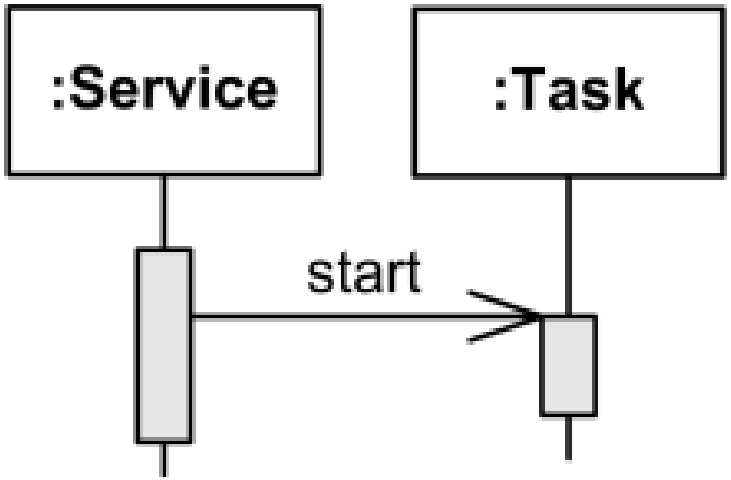
This notation is employed when the diagram needs to represent multiple instances of the same class interacting.

Sequence diagram - Messages



Synchronous message

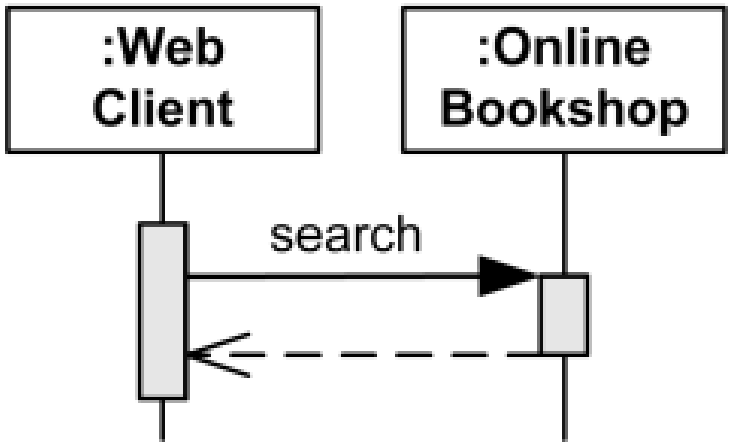
Synchronous message typically represents operation call - send message and suspend execution while waiting for response. Synchronous call messages are shown with filled arrow head.



Asynchronous message

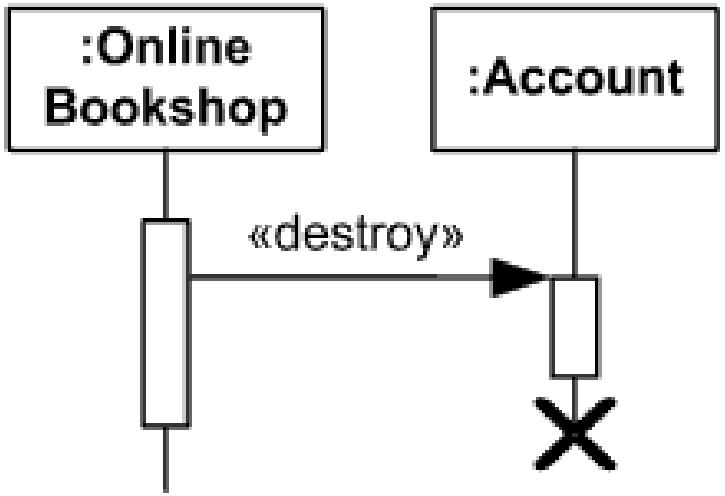
Asynchronous message - send message and proceed immediately without waiting for return value. Asynchronous messages have an open arrow head.

Sequence diagram - Message



Reply message

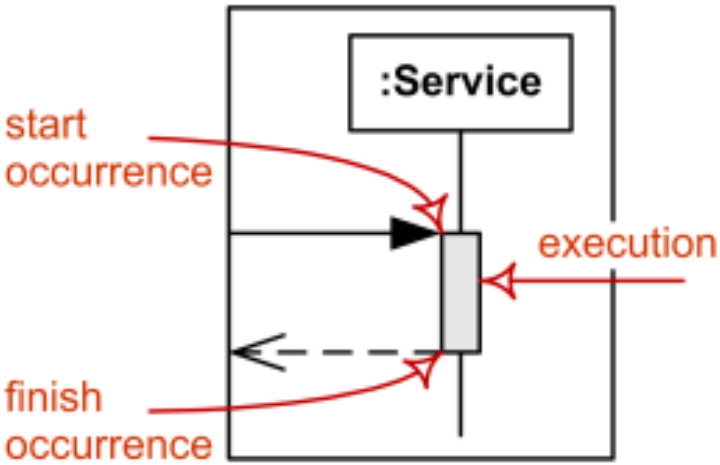
Reply message to an operation call is shown as a dashed line with open arrow head (looks similar to creation message).



Delete message

Delete message (called stop in previous versions of UML) is sent to terminate another lifeline. The lifeline usually ends with a cross in the form of an X at the bottom denoting destruction occurrence.

Sequence diagram - Execution



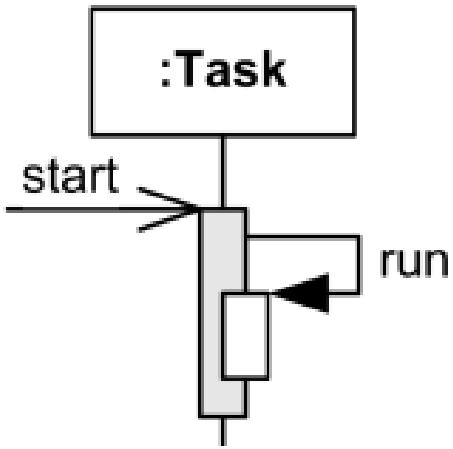
Execution

Execution (full name - execution specification, informally called activation) is interaction fragment which represents a period in the participant's lifetime when it is

- executing a unit of behavior or action within the lifeline,
- sending a signal to another participant,
- waiting for a reply message from another participant.

Note, that the execution specification includes the cases when behavior is not active, but just waiting for reply. The duration of an execution is represented by two execution occurrences - the start occurrence and the finish occurrence.

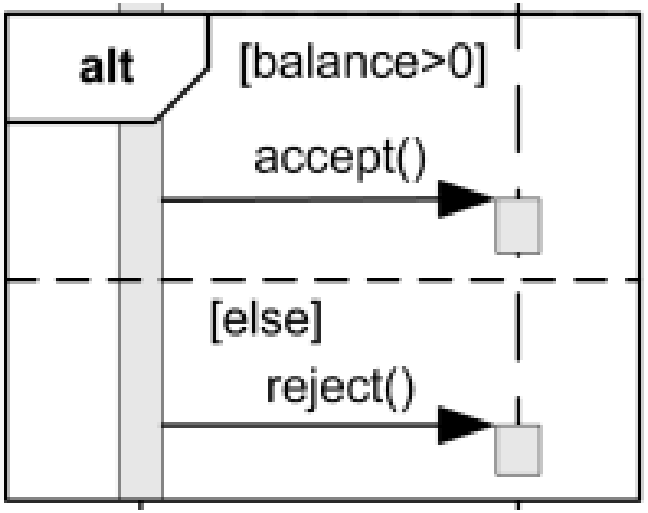
Execution is represented as a thin grey or white rectangle on the lifeline.



Overlapping execution

Overlapping execution specifications on the same lifeline are represented by overlapping rectangles.

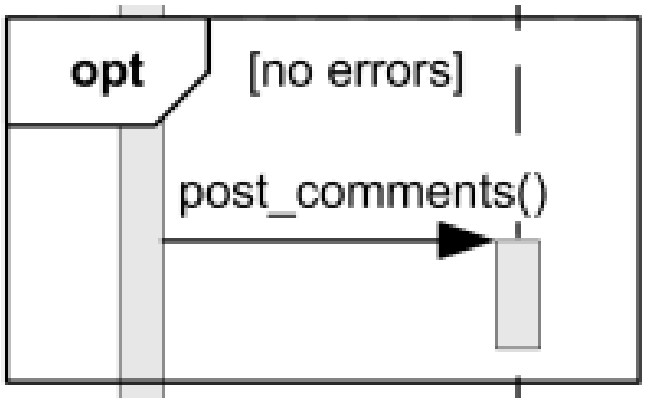
Sequence diagram - Operators



Alternative operator

The interaction operator alt means that the combined fragment represents a choice or alternatives of behavior. At most one of the operands will be chosen. The chosen operand must have an explicit or implicit guard expression that evaluates to true at this point in the interaction.
An implicit true guard is implied if the operand has no guard.

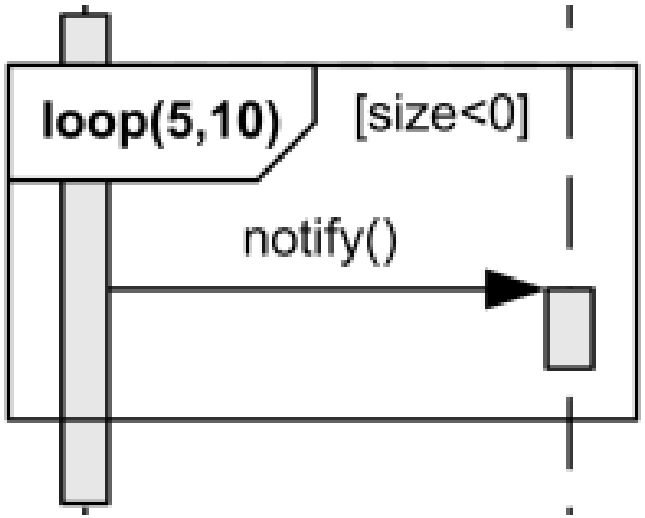
An operand guarded by else means a guard that is the negation of the disjunction of all other guards. If none of the operands has a guard that evaluates to true, none of the operands are executed and the remainder of



Option operator

The interaction operator opt means that the combined fragment represents a choice of behavior where either the (sole) operand happens or nothing happens. An option is semantically equivalent to an alternative combined fragment where there is one operand with non-empty content and the second operand is empty.

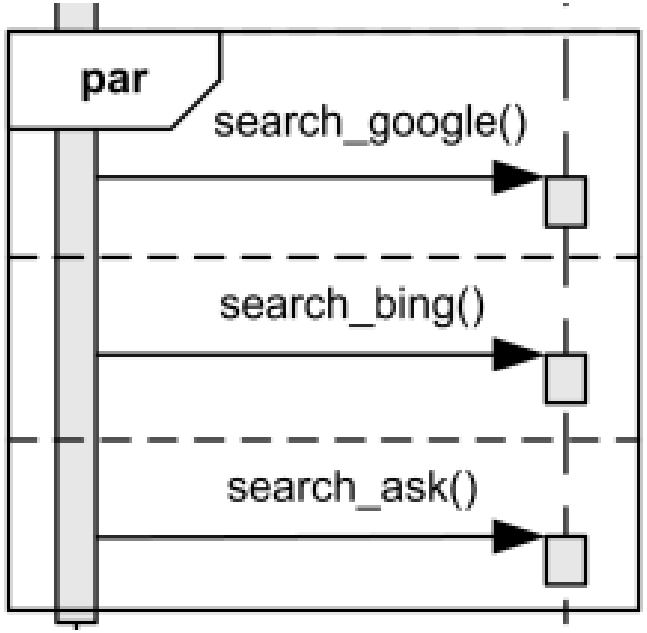
Sequence diagram - Operators



Loop operator

The interaction operator loop means that the combined fragment represents a loop. The loop operand will be repeated a number of times. The loop construct represents a recursive application of the seq operator where the loop operand is sequenced after the result of earlier iterations.
Loop could be controlled by either or both iteration bounds and a guard.

Loop operand could have iteration bounds which may include a lower and an upper number of iterations of the loop.

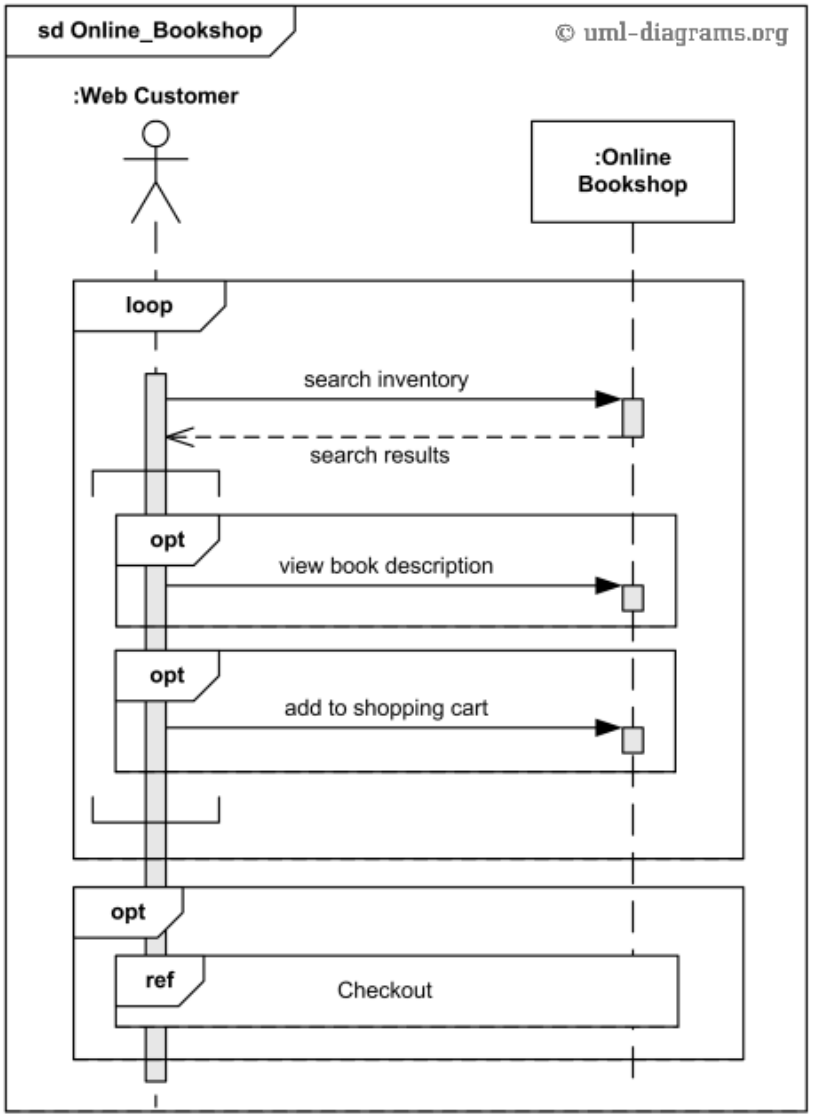


Parallel operator

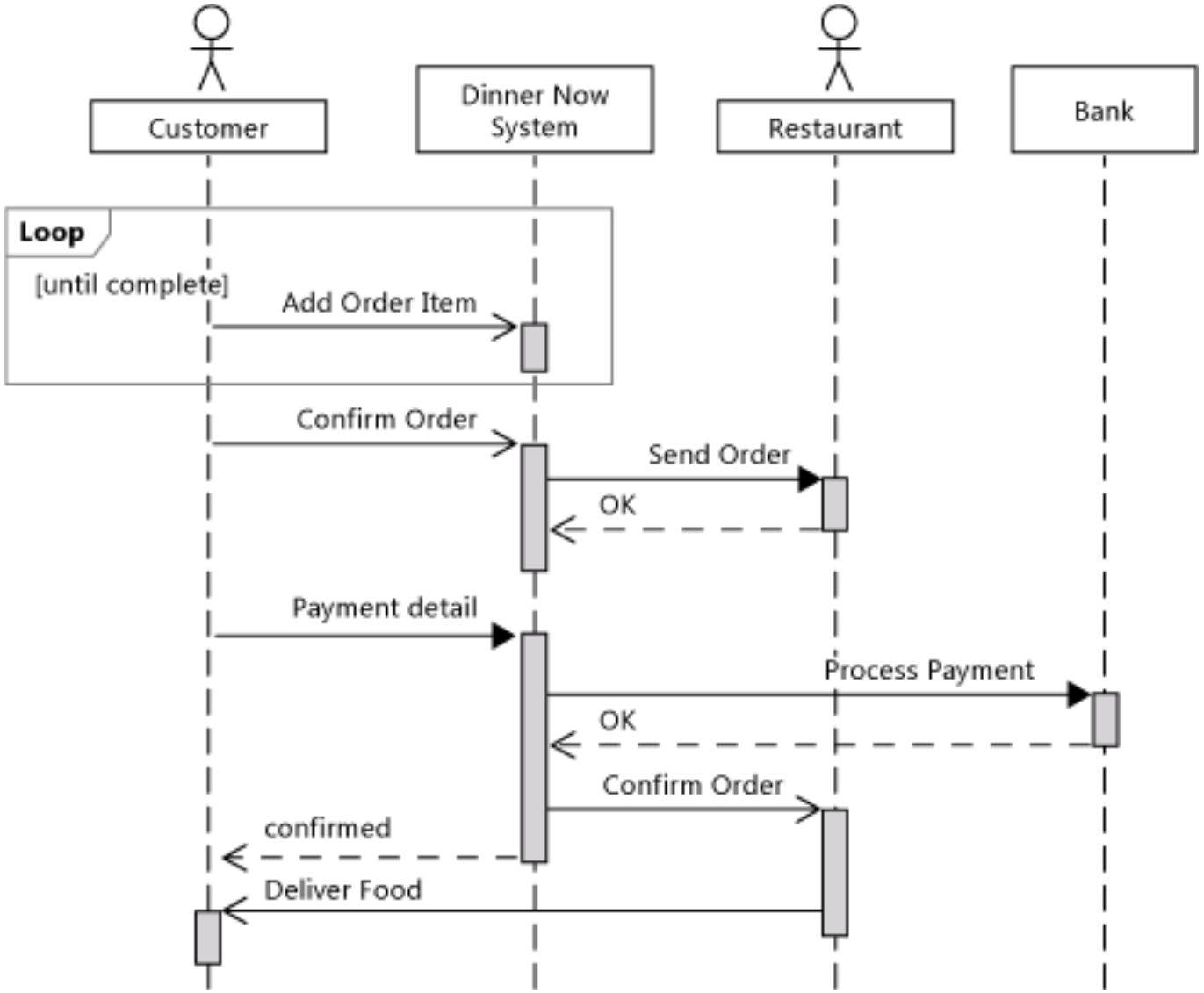
The interaction operator par defines potentially parallel execution of behaviors of the operands of the combined fragment. Different operands can be interleaved in any way as long as the ordering imposed by each operand is preserved.

Set of traces of the parallel operator describes all the possible ways or combinations that occurrence specifications of the operands may be interleaved without changing the order within each operand.

Sequence diagram - Example - Reading - 1



Sequence diagram - Example - Reading - 2



Sequence diagram - Example - Writing

Exercise 1

Model the process of ordering and delivering a takeaway meal through a Sequence Diagram.

Instructions

- Read the case and discuss with peers (2-3 minutes)
- Participative design (7-10 minutes)
- Wrap-up (3-5 minutes)

Purpose: Experiment with modeling
Outcome: UML Sequence Diagram

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages
   - **Synchronous**
   - **Asynchronous**
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after**
   - **simultaneously/at the same time/...**
   - **if/in case of/...**
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous**
   - **Asynchronous**
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after**
   - **simultaneously/at the same time/...**
   - **if/in case of/...**
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous**
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after**
   - **simultaneously/at the same time/...**
   - **if/in case of/...**
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous** → Can the first entity continue without waiting for the second one?
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after**
   - **simultaneously/at the same time/...**
   - **if/in case of/...**
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous** → Can the first entity continue without waiting for the second one?
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after** → Concept of sequence flow between the actions
   - **simultaneously/at the same time/...**
   - **if/in case of/...**
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous** → Can the first entity continue without waiting for the second one?
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after** → Concept of sequence flow between the actions
   - **simultaneously/at the same time/...** → Concept of parallel flow between the actions
   - **if/in case of/...**
   - **until/is repeated for/...**

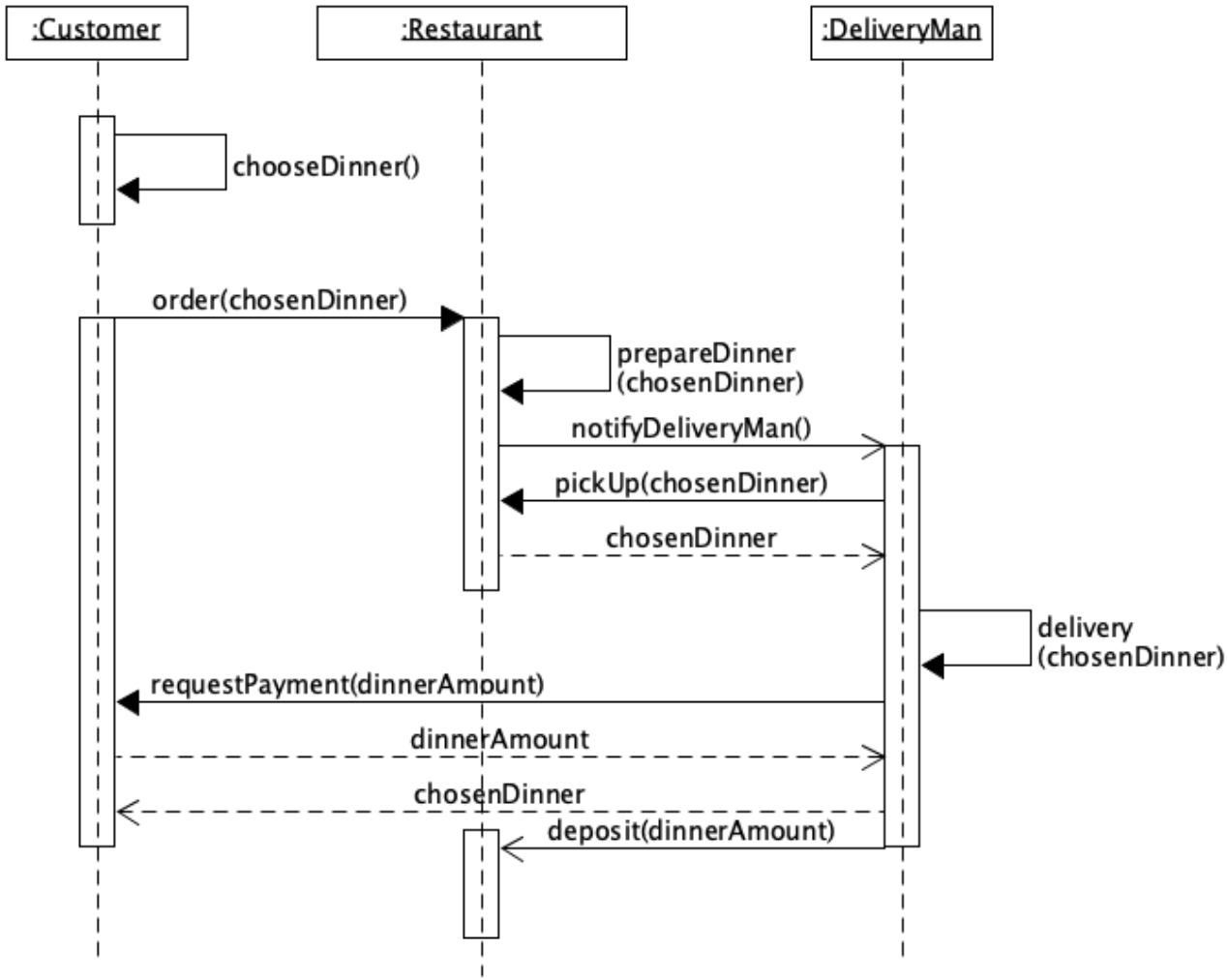
How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous** → Can the first entity continue without waiting for the second one?
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after** → Concept of sequence flow between the actions
   - **simultaneously/at the same time/...** → Concept of parallel flow between the actions
   - **if/in case of/...** → Concept of alternative/optional flow between the actions
   - **until/is repeated for/...**

How to solve the exercise?

1. Read text
2. Identify entities → What are the entities present in the system? (Nouns)
3. Identify messages → What are the messages/information shared between entities? (Verbs)
   - **Synchronous** → Does the first entity need to wait from a reply from the second entity?
   - **Asynchronous** → Can the first entity continue without waiting for the second one?
4. Identify specifiers to determine multiplicities and or direction of the associations.
   - **before/after** → Concept of sequence flow between the actions
   - **simultaneously/at the same time/...** → Concept of parallel flow between the actions
   - **if/in case of/...** → Concept of alternative/optional flow between the actions
   - **until/is repeated for/...** → Concept of repetitive/looping flow between the actions

Sequence diagram - Example - Writing 1 - Possible solution

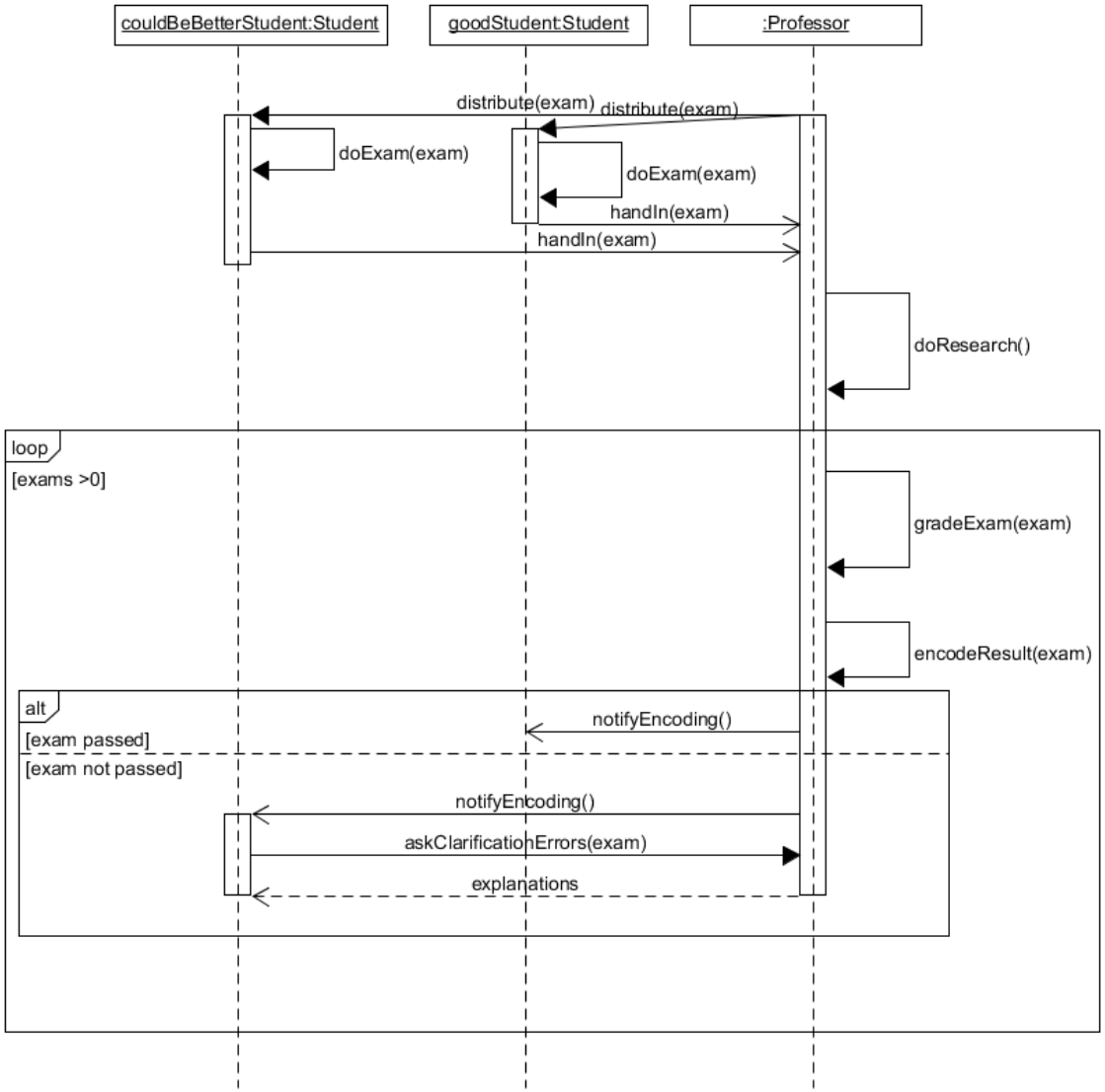


Sequence diagram - Example - Writing - 2

Exercise 2

Model the process of correction of an exam from the point of view of a professor using a Sequence Diagram. Model the different execution flows with a maximum of details.

Sequence diagram - Example - Writing 2 - Possible solution



Key points

- **A model is an abstract view of a system that ignores system details.** Complementary system models can be developed to show the system's context, interactions, structure and behavior.
- **Use case diagrams** and **sequence diagrams** are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; sequence diagrams add more information to these by showing interactions between system objects.
- Structural models show the organization and architecture of a system. **Class diagrams** are used to define the **static structure of classes** in a system and their associations.

Key points

- **Behavioral models are used to describe the dynamic behavior of an executing system.** This behavior can be modeled from the perspective of the data processed by the system, or by the events that stimulate responses from a system.
- **Activity diagrams** are used to model **the processing of data**, where each activity represents one process step.
- For Use Case, Activity, Class and Sequence Diagrams we have seen some techniques to support diagram preparations.
- Practice is key in mastering all the subtleties, strong points and limitations of the different models.

Formative Assignment - UML Modeling

- The formative assignment on UML modeling will be available on Brightspace during this afternoon
- The deadline for the assignment is the **20/03**

Formative Assignment - Programming Languages

- The formative assignment on Programming Languages is available on Brightspace
- **Flipped classroom approach**
- **Mandatory attendance and bonus points (0,25).**
- The presentations will take place on the **06/03** and **08/03**.