OOAD, UP and UML

System Analysis and Design

School of Computer Science and Technology, Tongji University

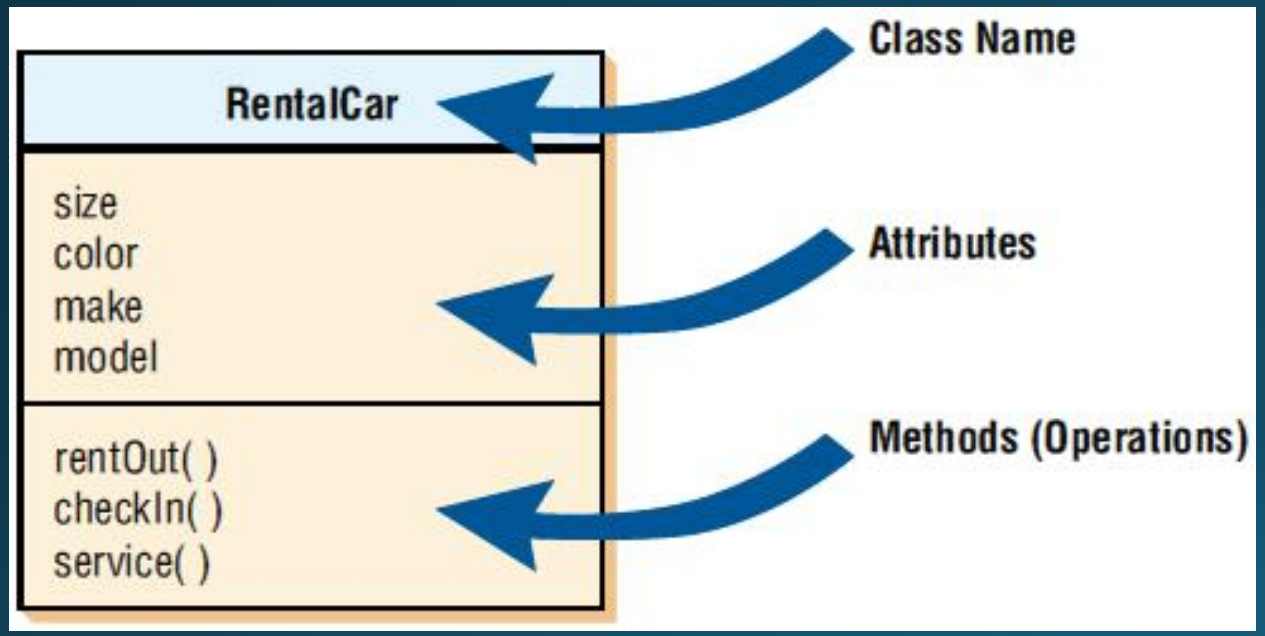
Index

- Object-Oriented Style of Design and Programming
- Object-Oriented Analysis and Design
- Unified Process
- UML Overview
  - UML Structure
  - Describe Software Architecture with the “4+1” views
  - Common Usage of UML (A typical workflow)
  - More Examples of UML Diagrams
- Discussion: Using Generative AI for Object-Oriented Software Engineering with UML

Review: OO Concepts

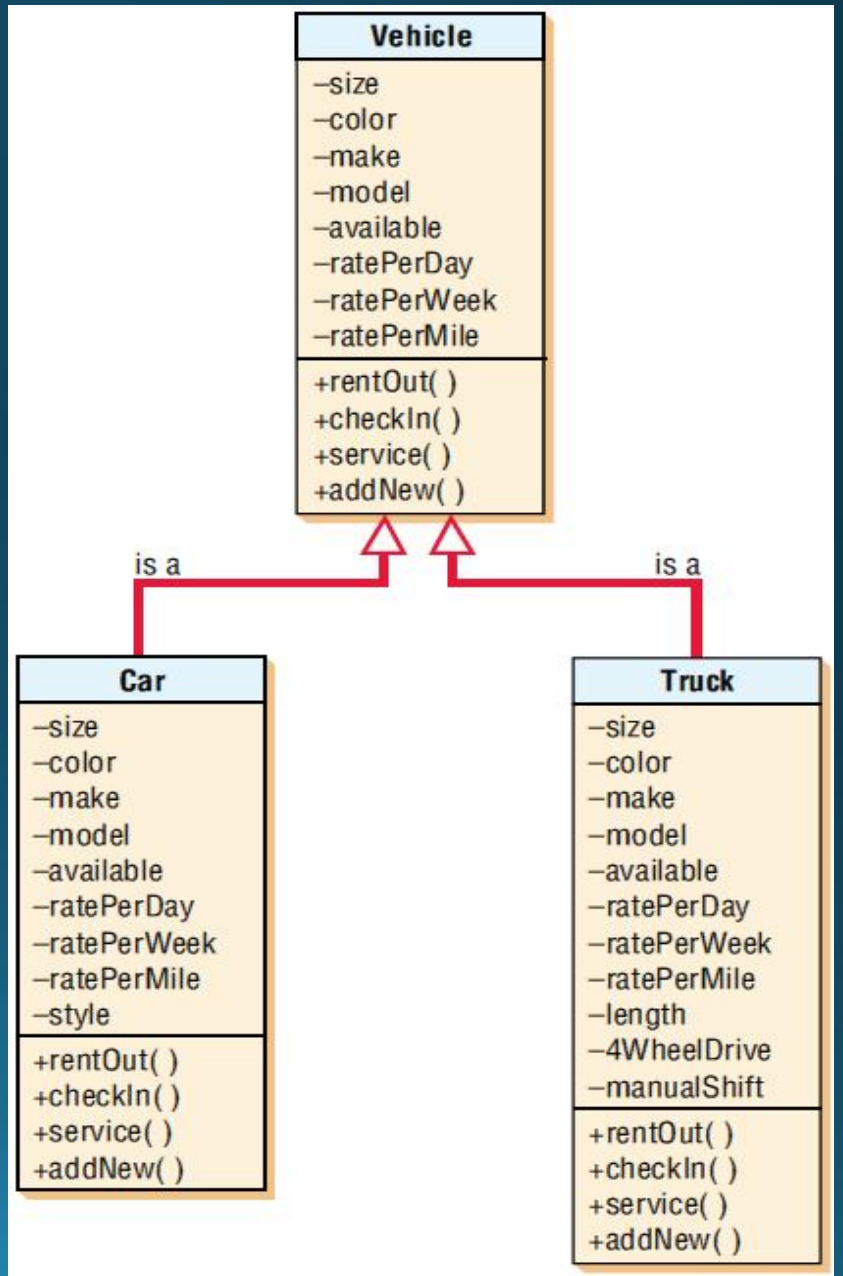
- Objects
  - A cohesive cluster of data and behavior
  - Contains data and perform functions
    - Persons, places, or things that are relevant to the system being analyzed
    - May be customers, items, orders, and so on
    - May be GUI displays or text areas on a display
- Classes
  - Defines the set of shared attributes and behaviors found in each object in the class
  - Should have a name that differentiates it from all other classes
  - Instantiate is when an object is created from a class
  - An attribute describes some property that is possessed by all objects of the class
  - A method is an action that can be requested from any object of the class
- Inheritance
  - When a derived class inherits all the attributes and behaviors of the base class
  - Reduces programming labor by using common objects easily
  - A feature only found in object-oriented systems

Examples



Representing a Class in UML

A Class Diagram Showing Inheritance



Object-Oriented Style of Design and Programming

- Object-Oriented Programming
  - Encapsulation
    - Hide details in a class, provide methods
  - Polymorphism
    - Same name, different behavior, based on type
  - Inheritance
    - Capture common attributes and behaviors in a base class and extend it for different types
- SPD/OOPD
  - Structured programming and design
    - Systems are modeled as a collection of functions and procedures that pass data all over the place
  - Object-oriented programming and design
    - Systems are modeled as a collection of interacting objects where the data is encapsulated with the methods that need and operate on those values
      - Repeating "The one thing we missed was putting the data and functions together"

The OO approach

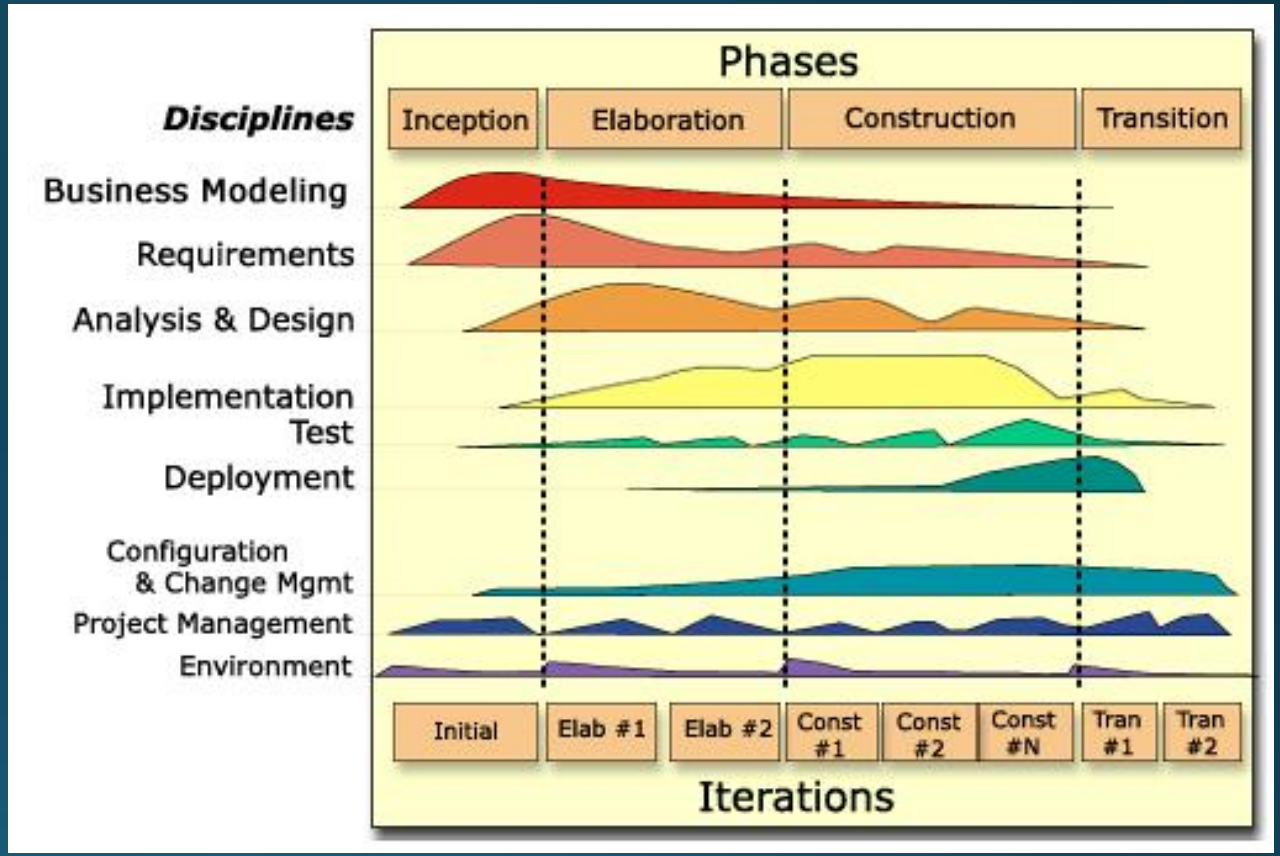
- More natural–models real world, not computers
- More flexible
- More maintainable
  - People can understand it better
  - touch as few classes as possible when enhancing or fixing
- More reliable – can perform unit testing
- Can build reusable modules (classes)
- Extensible: can add functionality and offer alternatives
- OOT offers
  - Techniques for creating flexible, natural software modules.
  - Systems that are much easier to adapt to new demands
  - Reuse shortens the development life cycle
  - Systems are more understandable
    - easier to remember 50 real objects rather than 500 functions

Object-Oriented Analysis and Design

- OOAD: Methods and technologies for analysis, design, and testing have been created to deal with larger systems
  - Use cases help analyze and capture requirements
  - Responsibility-Driven Design helps determine which objects are needed and what they must do
  - Test Driven Development aids in implementation, design, and maintenance of software
  - Unified Modeling Language (UML) used to design and document large systems
- OOAD Works well in situations where complicated systems are undergoing continuous maintenance, adaptation, and design
  - Reusability
    - Recycling of program parts should reduce the costs of development in computer-based systems
    - Objects, classes are reusable
  - Maintaining systems
    - Making a change in one object has a minimal impact on other objects
- UML is an industry standard for modeling OO systems.

Unified Process (UP)

- The **Unified Software Development Process** or **Unified Process** is an **iterative** and **incremental** software development process framework.
  - The best-known and extensively documented refinement of the Unified Process is the Rational Unified Process (RUP). Other examples are OpenUP and Agile Unified Process.
- The RUP Organization
  - By time : Phases and iterations
    - Four phases
      - Inception
      - Elaboration
      - Construction
      - Transition
    - One or more iterations in each phase
    - Each iteration last 2 to 3 months
  - By content: Disciplines
    - Five core workflows in each iteration
      - Requirements, Analysis, Design, Implementation, Test



Iteration: A distinct sequence of activities with a baselined plan and evaluation criteria resulting in a release (internal or external).

What Is the UML?

- The UML is a language for
  - Visualizing
    - An explicit model facilitates communication.
  - Specifying
    - precise, unambiguous, and complete.
  - Constructing
    - Forward engineering and Reverse engineering
  - Documenting
    - The UML addresses documentation of system architecture, requirements, tests, project planning, and release management.

  the artifacts of a software-intensive system.

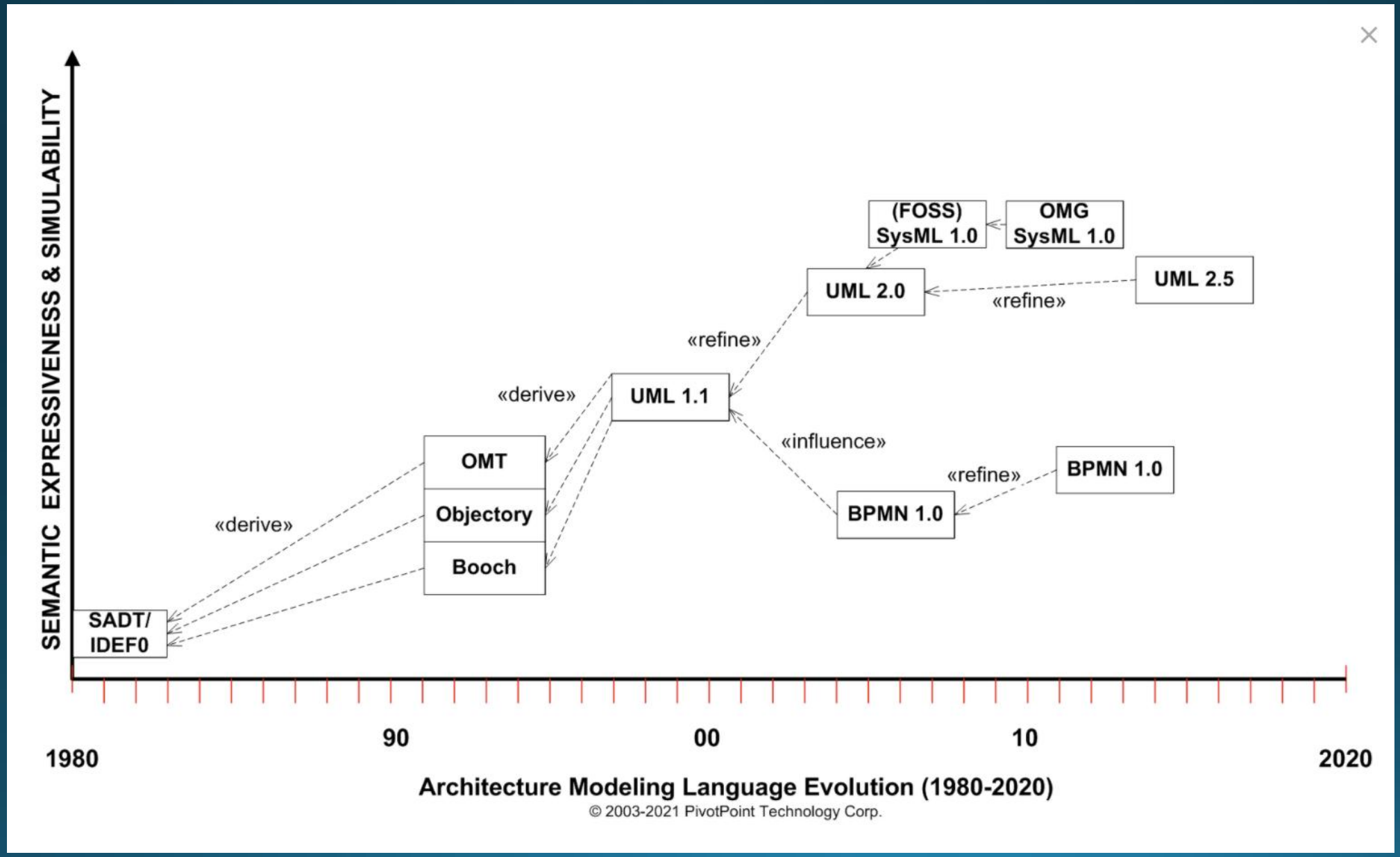

UML® Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

Title:	Unified Modeling Language
Acronym:	UML®
Version:	2.5.1
Document Status:	formal ⓘ
Publication Date:	December 2017
Categories:	Modeling Software Engineering

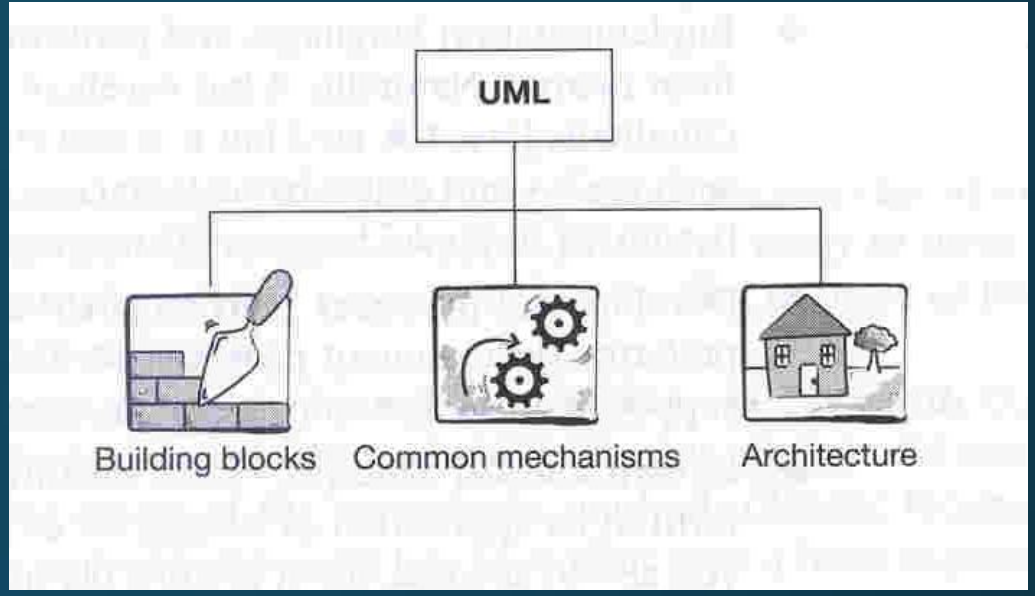
https://www.omg.org/spec/UML/About-UML/

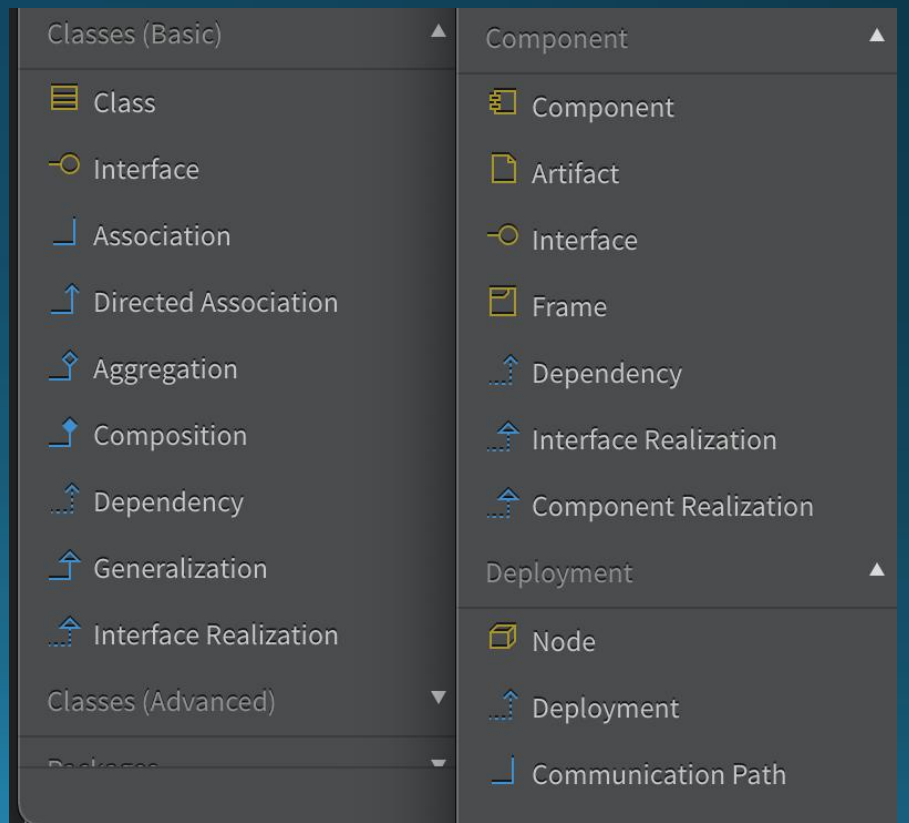
(Part of) Modeling Language Evolution



UML Structure

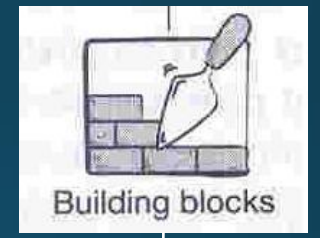
- The structure of UML:
  - Building blocks
    - Things
      - Structural
      - Behavioral
      - Grouping
      - Annotation
    - Relationships
    - Diagrams
  - Common mechanisms
    - UML ways to achieve specific goals
  - Architecture
    - The view of system architecture





UML Building Blocks

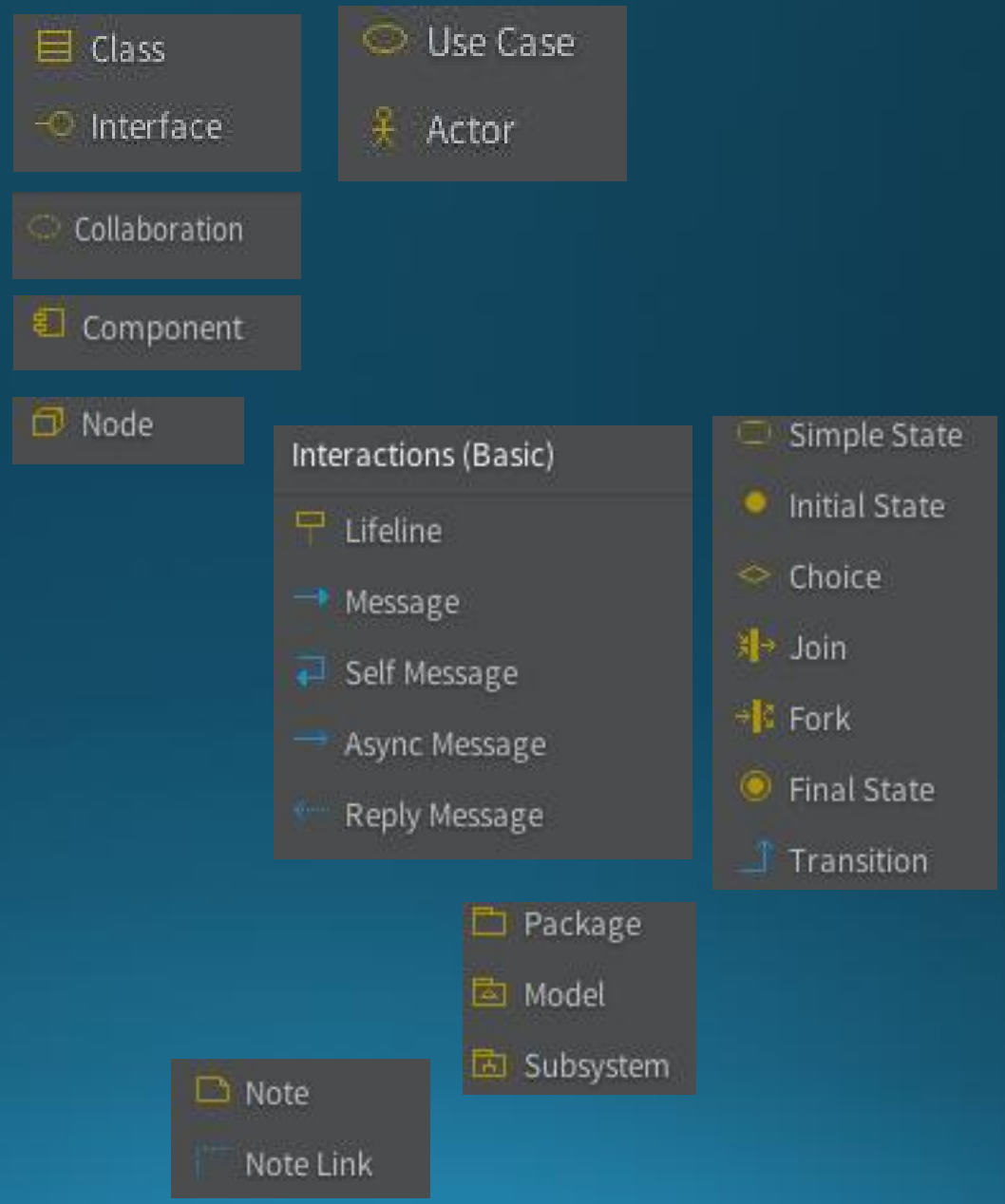
- Things
  - modeling elements
- Relationships
  - how things are semantically related
- Diagrams
  - Views of models
  - Show collection of things
  - What system do (analysis level diagrams)
  - How it will do (design level diagrams)



Things | Relationships | Diagrams

UML Things

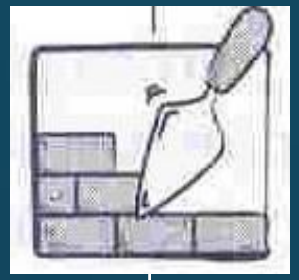
- Structural things
  - The nouns of a UML model:
    - classes
    - interfaces
    - collaboration
    - use case
    - component
    - node
- Behavioral Things
  - The verbs of a UML model :
    - interactions
    - state machine
- Grouping Things
  - Package:
    - to group semantically related elements
- Annotation Things
  - The note to explain the diagrams



UML Relationships

- To show on a model
  - How two or more things relate to each other
- To capture meaningful connections between things
- Type of relationships
  - Association
    - Links between objects
    - Aggregation – "whole-part" relationship
    - Composition – " comprise/contains" – relationship
  - Dependency
    - Change to one object affects behavior of another object
  - Generalization
  - Realization
    - One classifier specifies a contract
    - the other classifier to carry out

Fundamental UML Relationships



Relationships

Association

Dependency

Generalization

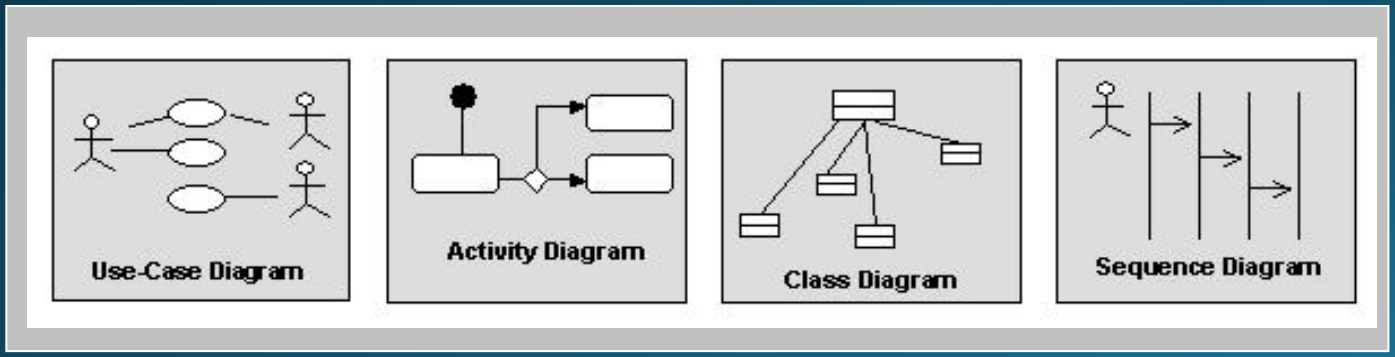
Realization

UML Relationships and their graphical representation

Demonstration

UML Diagrams

- Diagrams graphically depict a view of a part of your model.
- Different diagrams represent different views of the system that you are developing.
- A model element will appear on one or more diagrams.

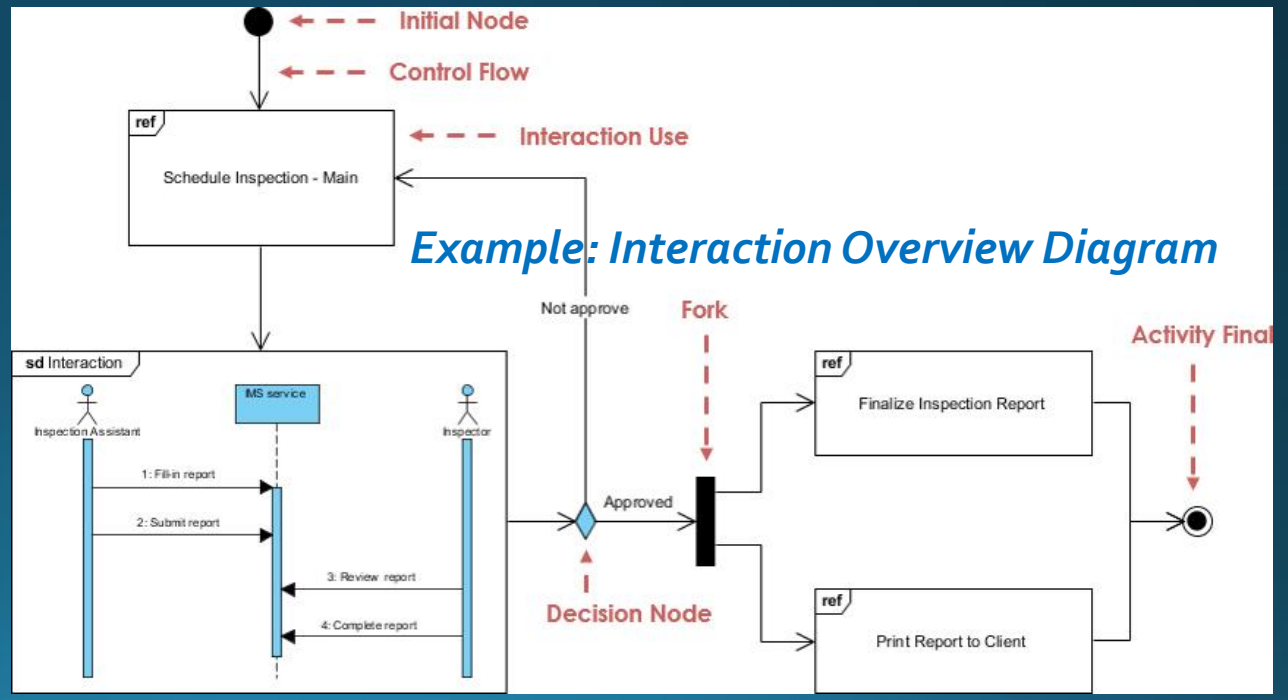


UML Category	UML Elements	Specific UML Details
Things	Structural Things	Classes Interfaces Collaborations Use Cases Active Classes Components Nodes
	Behavioral Things	Interactions State Machines
	Grouping Things	Packages
	Annotational Things	Notes
Relationships	Structural Relationships	Dependencies Aggregations Associations Generalizations
	Behavioral Relationships	Communicates Includes Extends Generalizes
Diagrams	Structural Diagrams	Class Diagrams Component Diagrams Deployment Diagrams
	Behavioral Diagrams	Use Case Diagrams Sequence Diagrams Communication Diagrams Statechart Diagrams Activity Diagrams

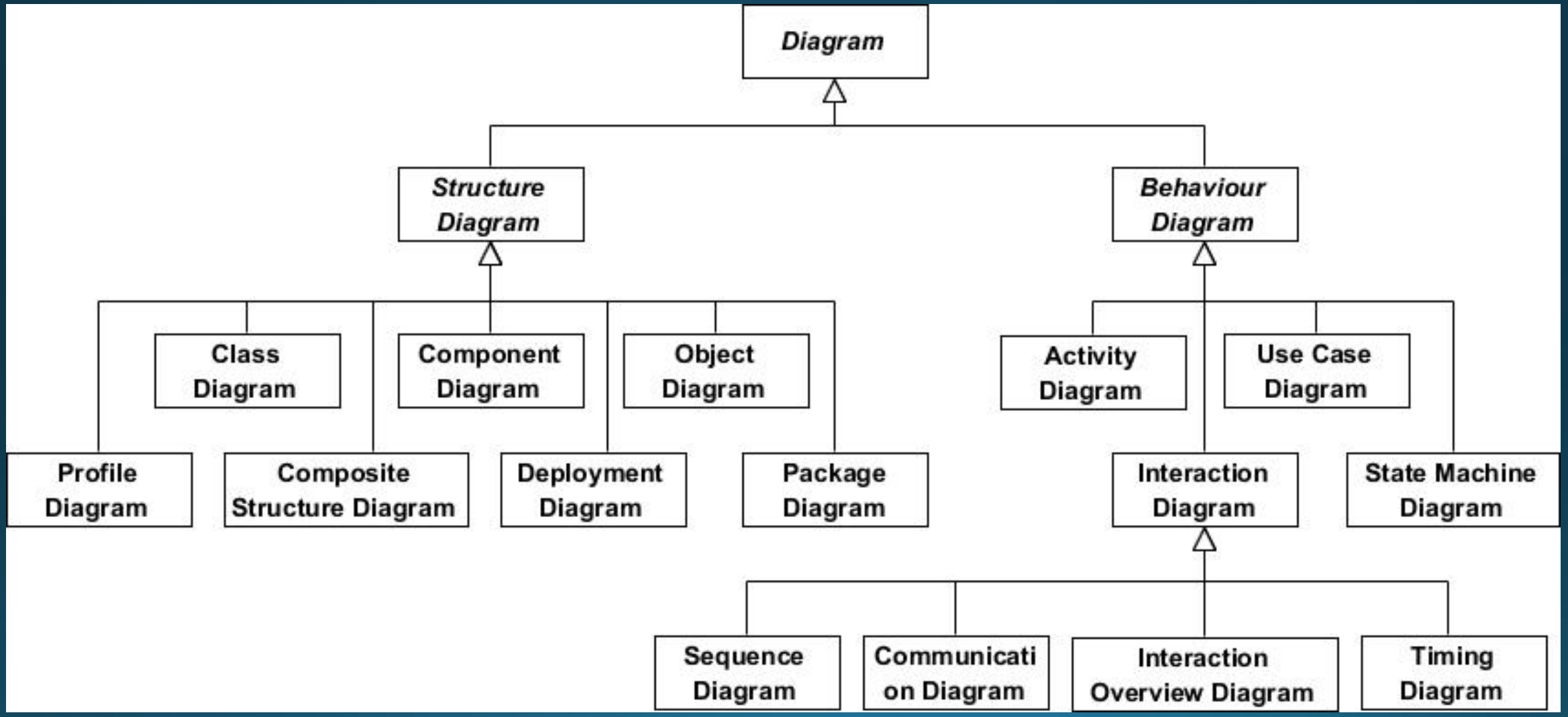
UML2 Diagrams (Static and Dynamic)

- **Structure diagrams** show the static structure of the system and its parts on different abstraction and implementation levels and how they are related to each other.
- The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

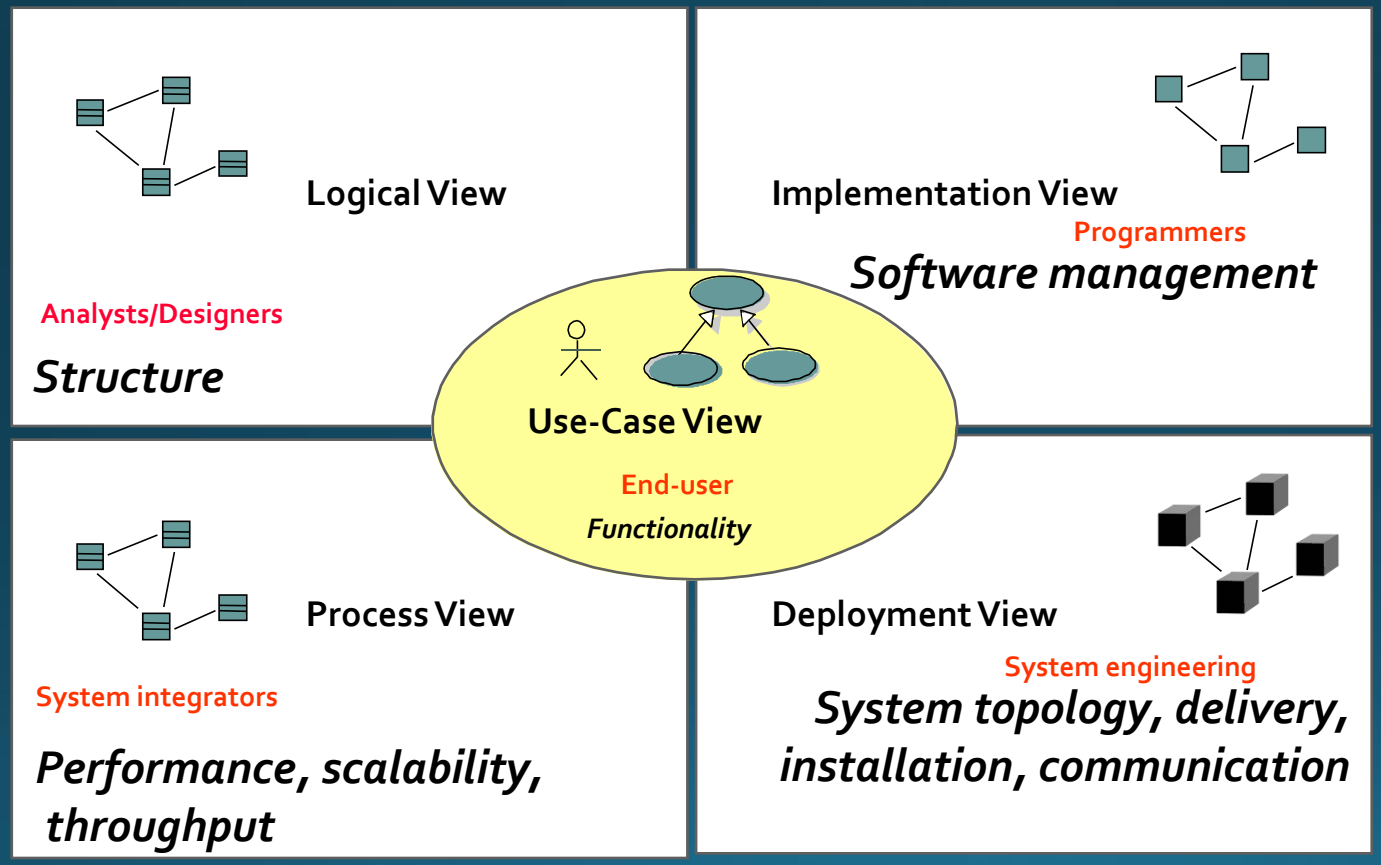
- **Behavior diagrams** show the **dynamic behavior** of the objects in a system, which can be described as a series of changes to the system over **time.**



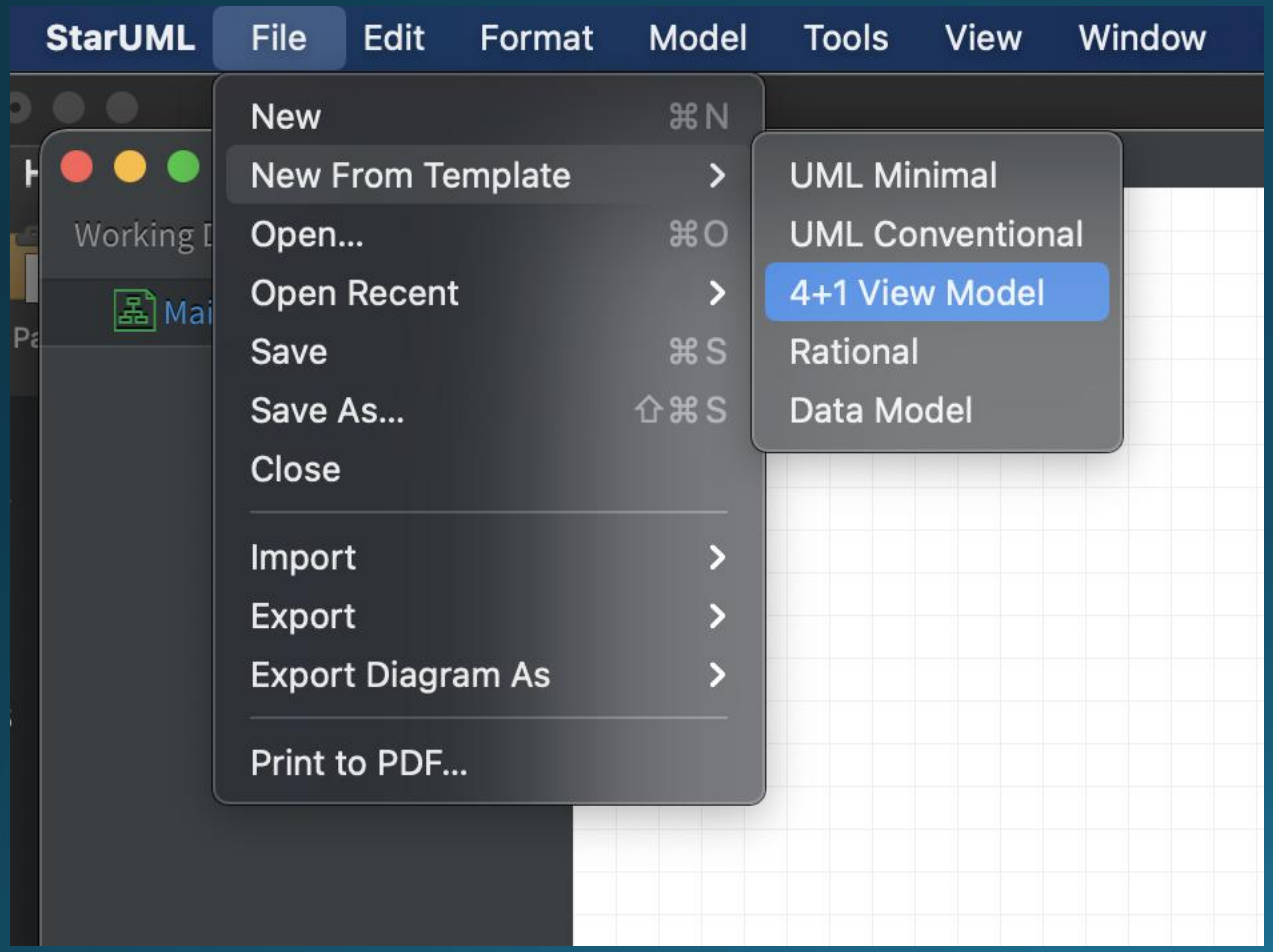
Available UML Diagrams

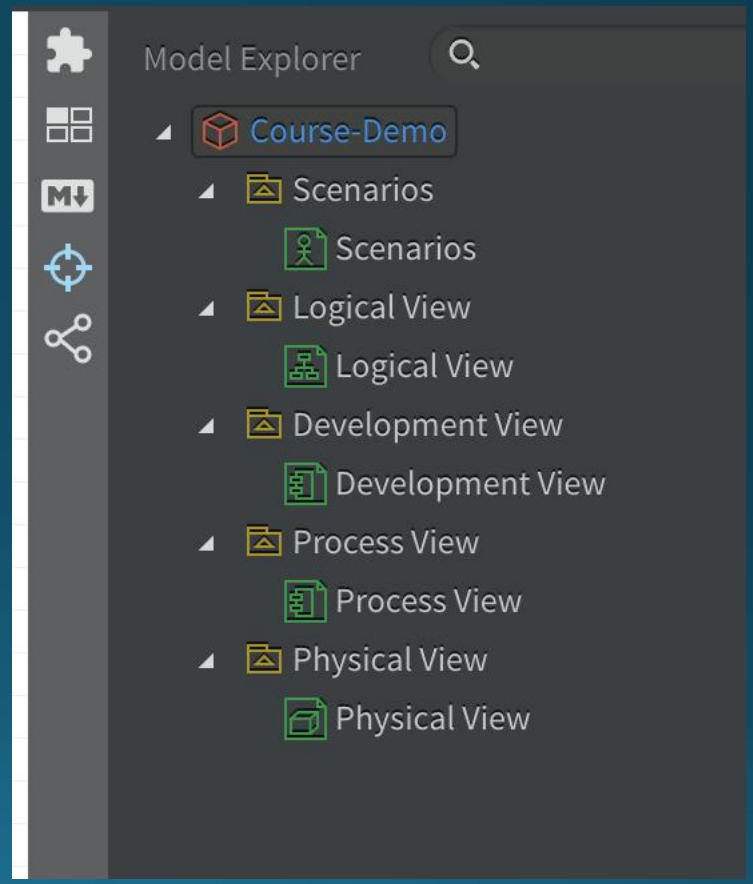


Software Architecture: The "4+1 View" Model

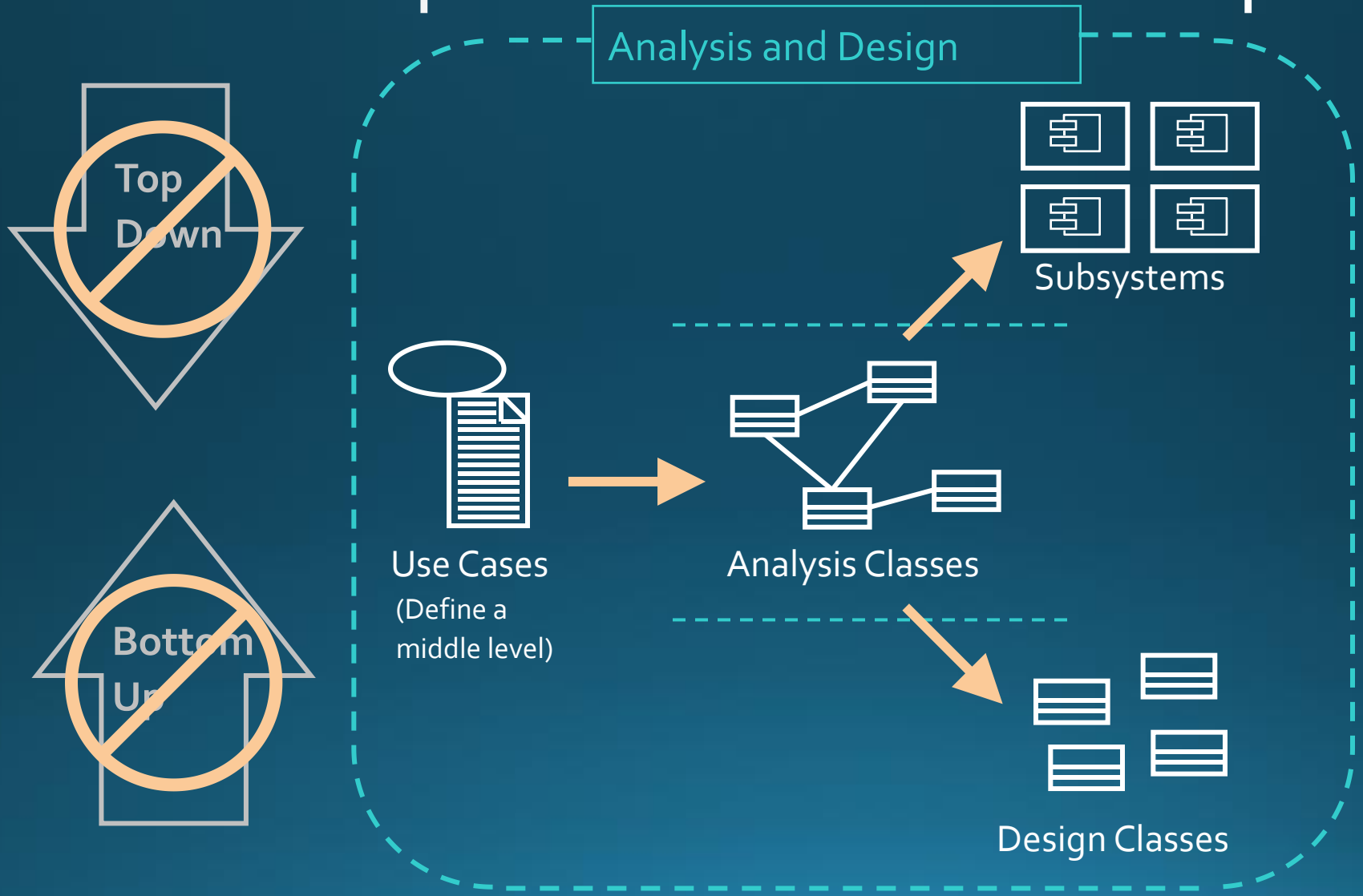


“4+1” Views Example

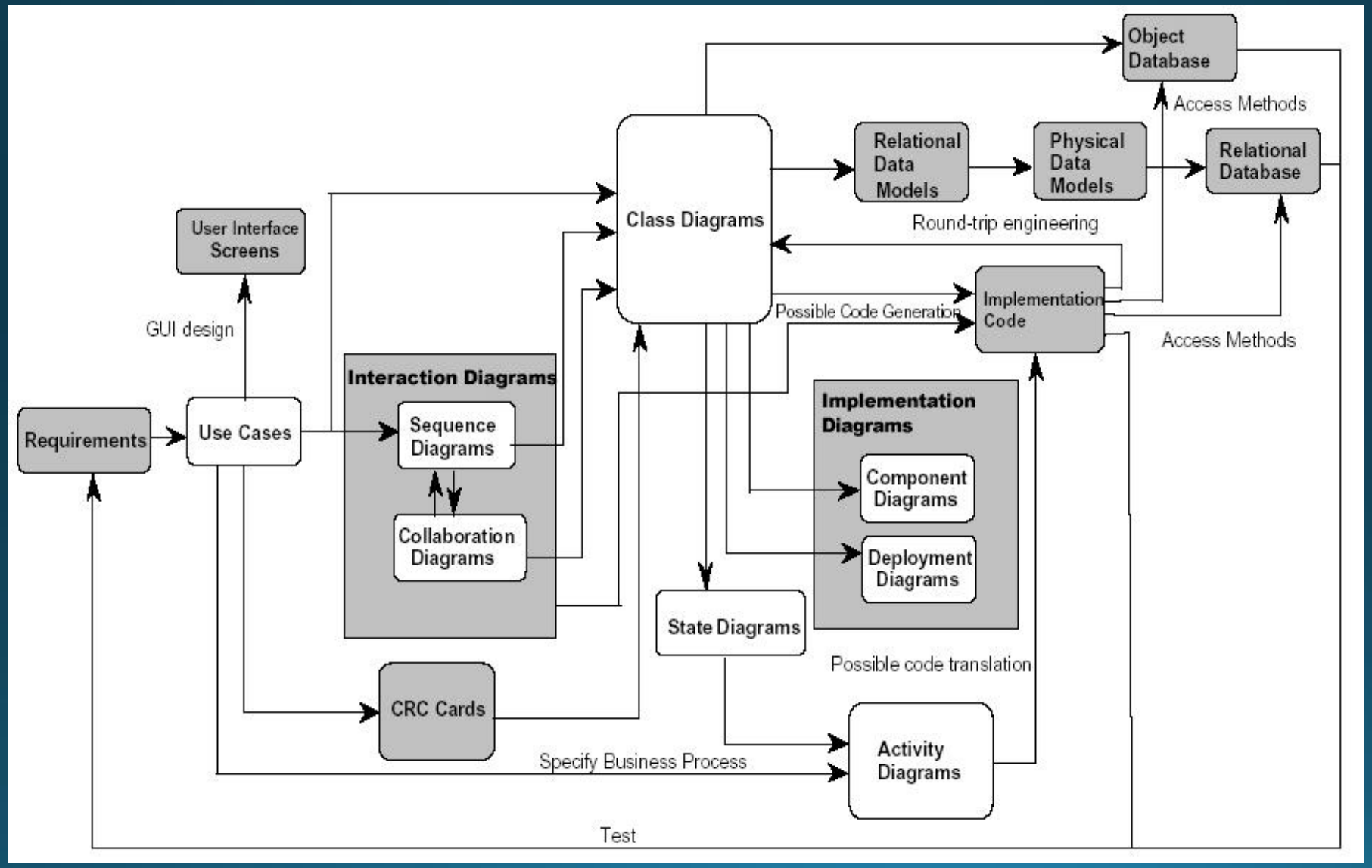




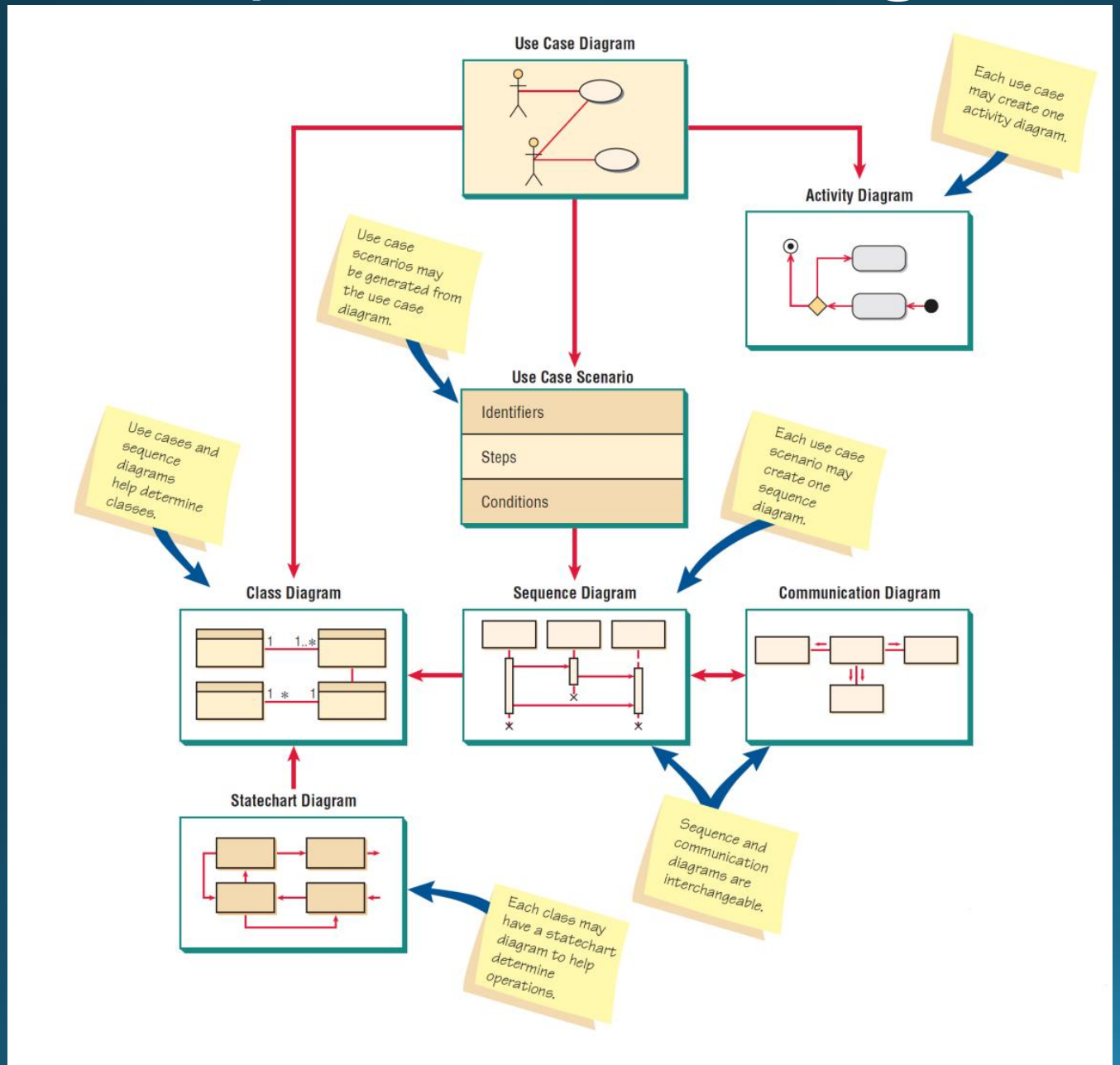
Analysis and Design Are Not Top-Down or Bottom-Up



Design with UML – A Typical Workflow

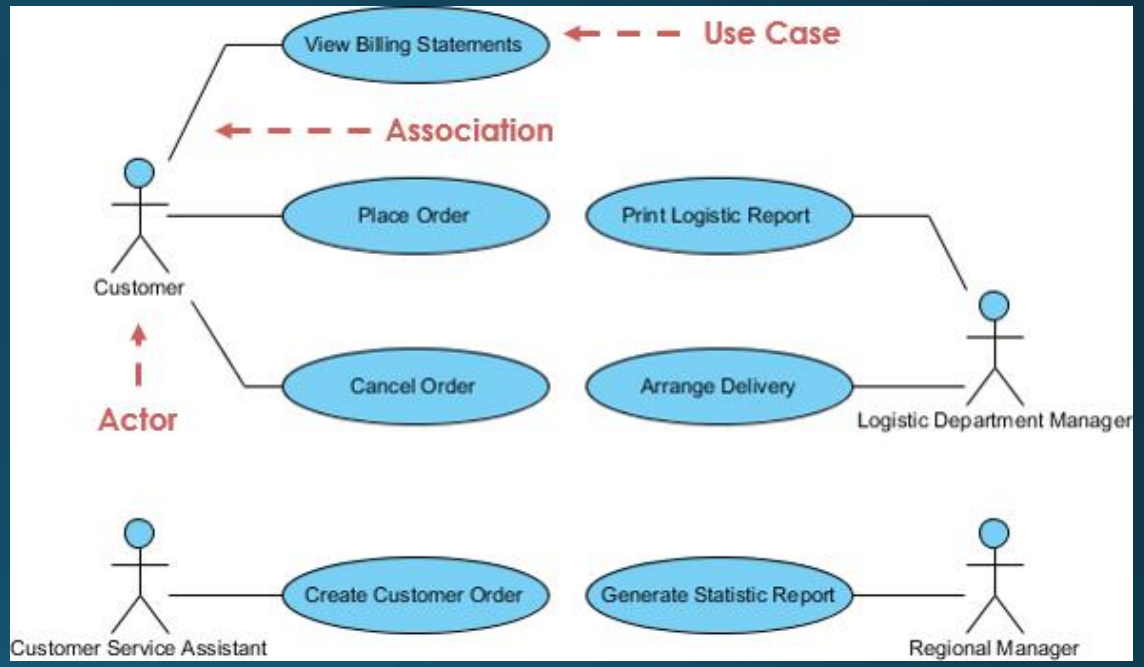


Commonly Used UML Diagrams



Use Case Diagram Example

- Describes what the system does, without describing how the system does it
- Based on the interactions and relationships of individual use cases
- Use case describes
  - Actor
  - Event
  - Use case

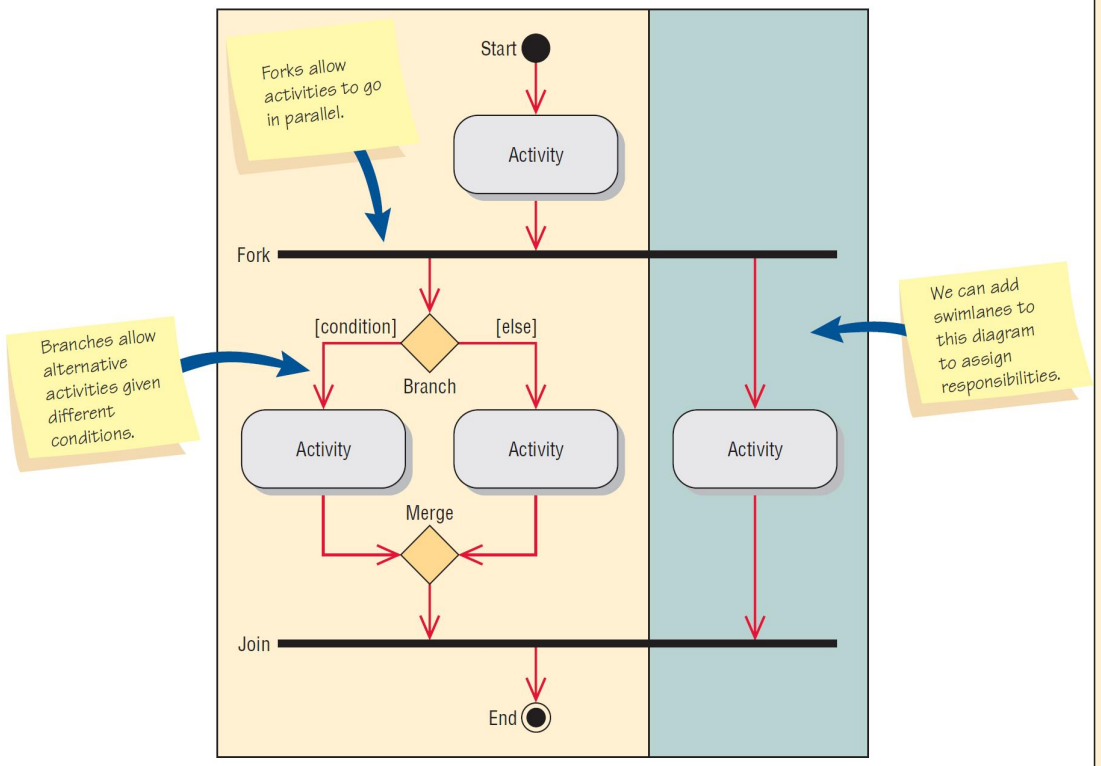


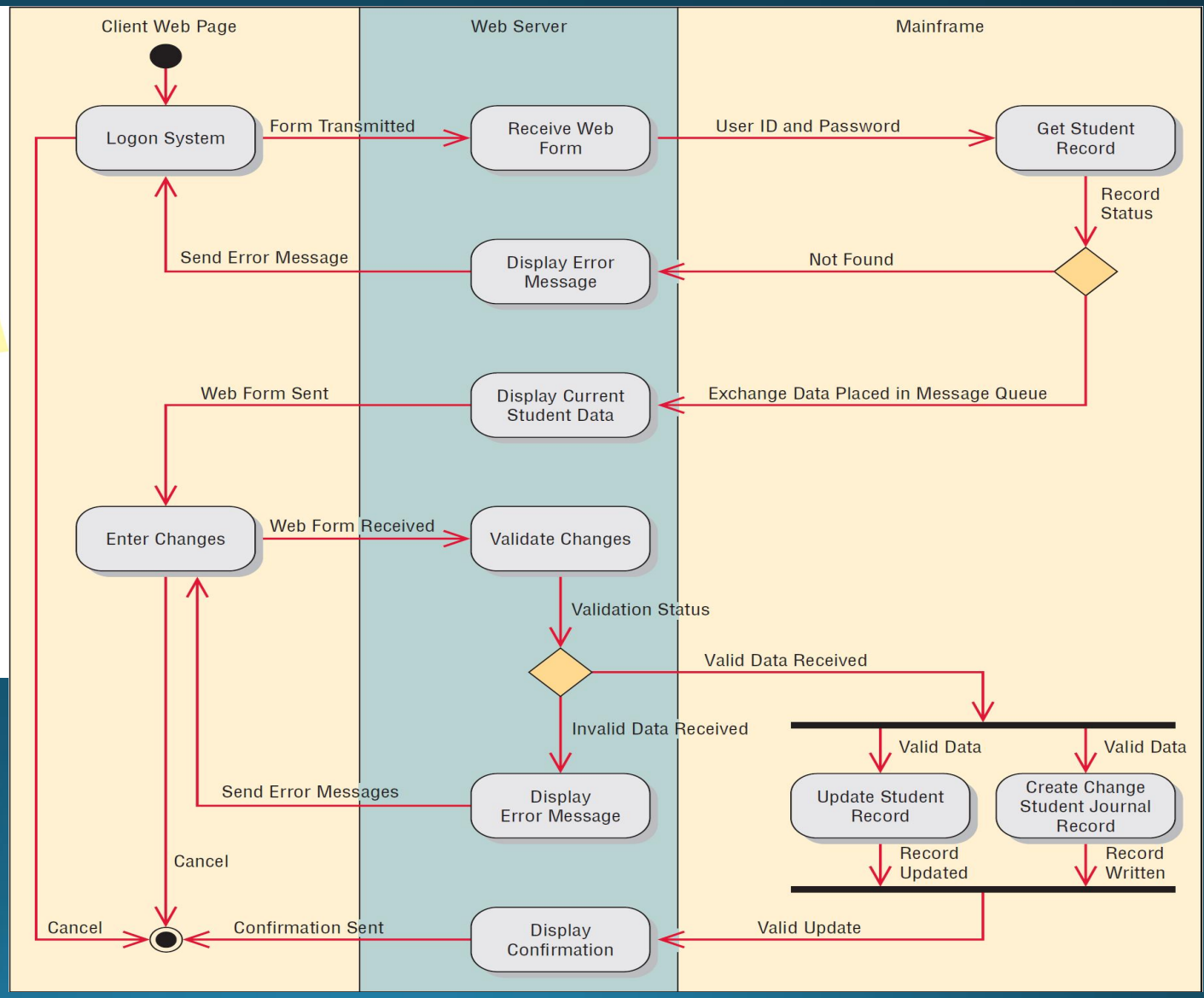
Use case modeling will be introduced as a part of reqirements modeling in details next lecture

Use Case Specification Example

Use case name:	Change Student Information	**UniqueID:** Student UC 005
Area:	Student System	
Actor(s):	Student	
Description:	Allow student to change his or her own information, such as name, home address, home telephone, campus address, campus telephone, cell phone, and other information using a secure Web site.	
Triggering Event:	Student uses Change Student Information Web site, enters student ID and password, and clicks the **Submit** button.	
Trigger type:	☒ External ☐ Temporal	
Steps Performed (Main Path)		**Information for Steps**
1. Student Logons on to the secure Web server.		Student ID, Password
2. Student record is read and password is verified.		Student Record, StudentID, Password
3. Current student personal information is displayed on the Change Student Web page.		Student Record
4. Student enters changes on the Change Student Web form and clicks **Submit** button.		Change Student Web Form
5. Changes are validated on the Web server.		Change Student Web Form
6. Change Student Journal record is written.		Change Student Web Form
7. Student record is updated on the Student Master.		Change Student Web Form, Student Record
8. Confirmation Web page is sent to the student.		Confirmation Page
Preconditions:	Student is on the Change Student Information Web page.	
Postconditions:	Student has successfully changed personal information.	
Assumptions:	Student has a browser and a valid user ID and password.	
Requirements Met:	Allow students to be able to change personal information using a secure Web site.	
Outstanding Issues:	Should the number of times a student is allowed to logon be controlled?	
Priority:	Medium	
Risk:	Medium	

Activity Diagram Examples





- Decision
- Fork and Join
- Swimlane

Class Diagram

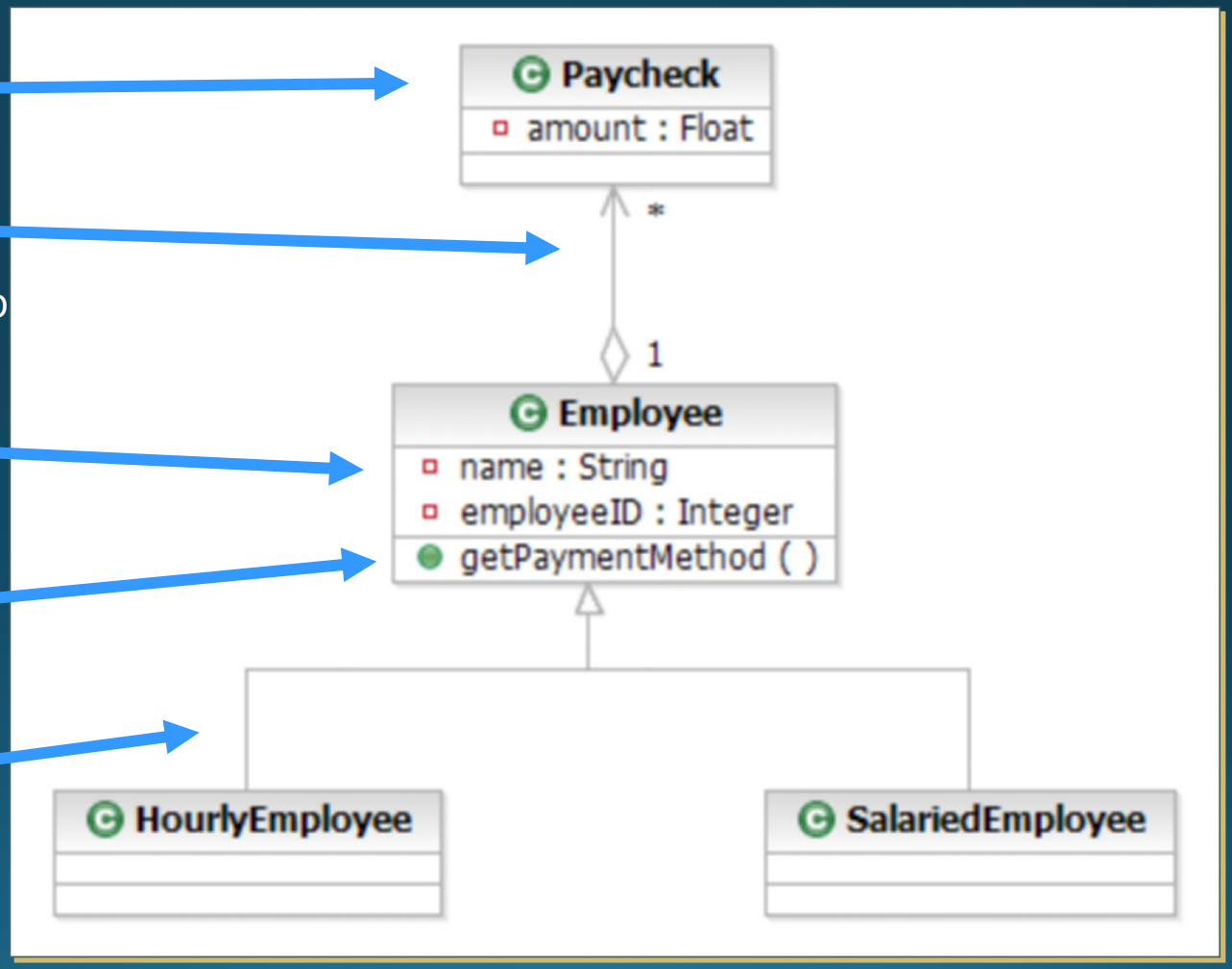
Class
A description of a set of objects

Aggregation
Represents a part-whole relationship

Attribute
Named property of a class

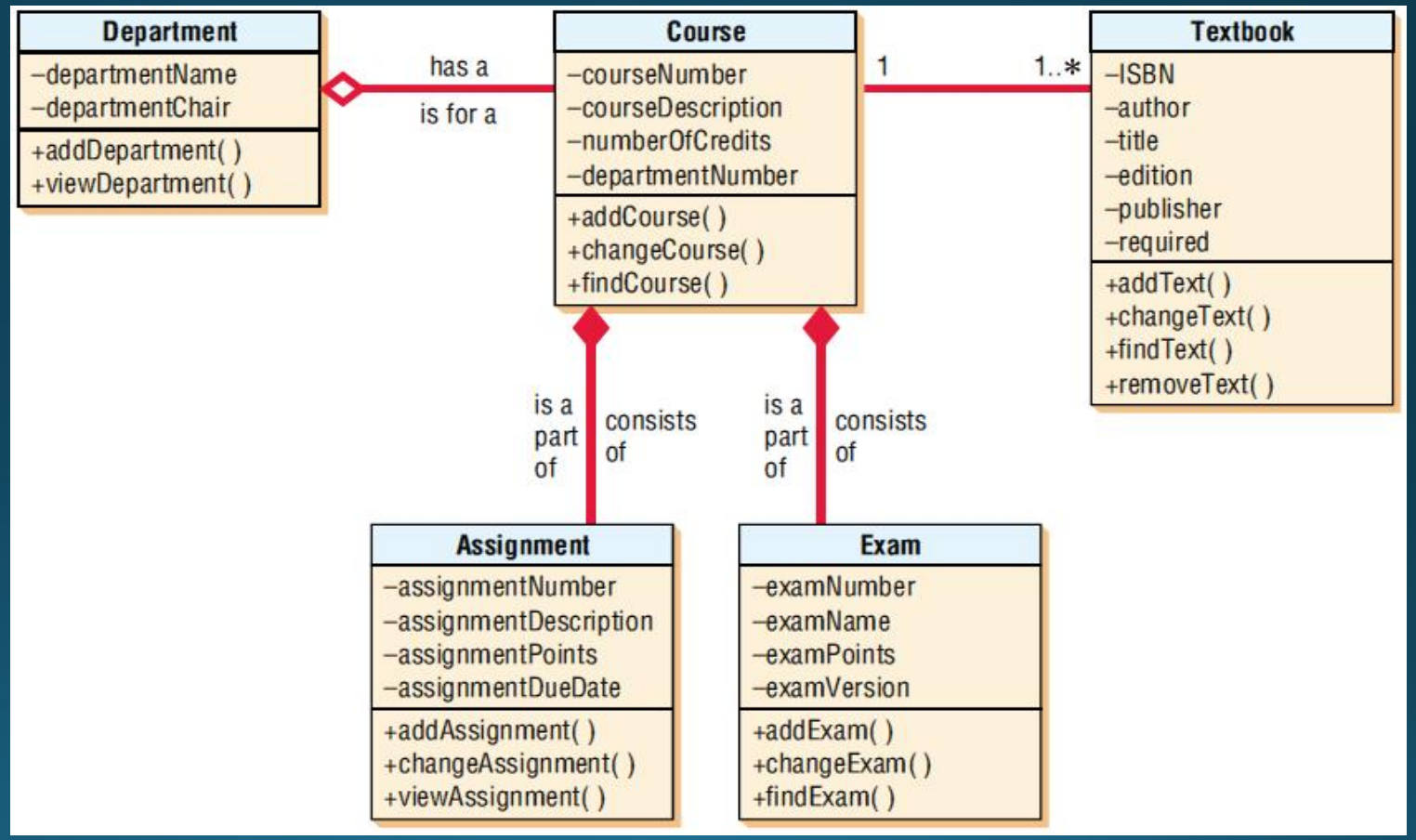
Operation
Class behavior

Generalization
Shows an inheritance relationship

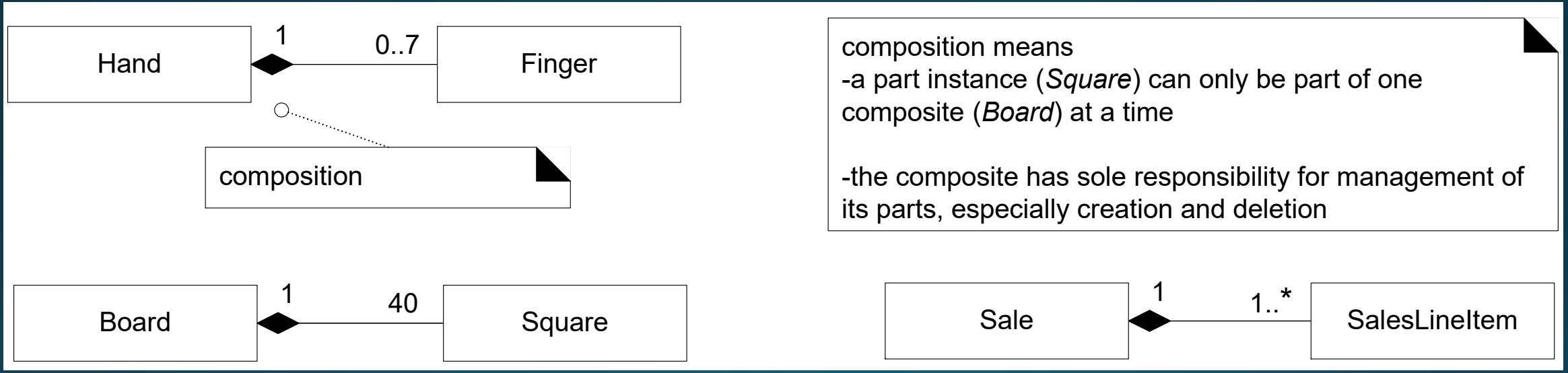


Class Diagram Example

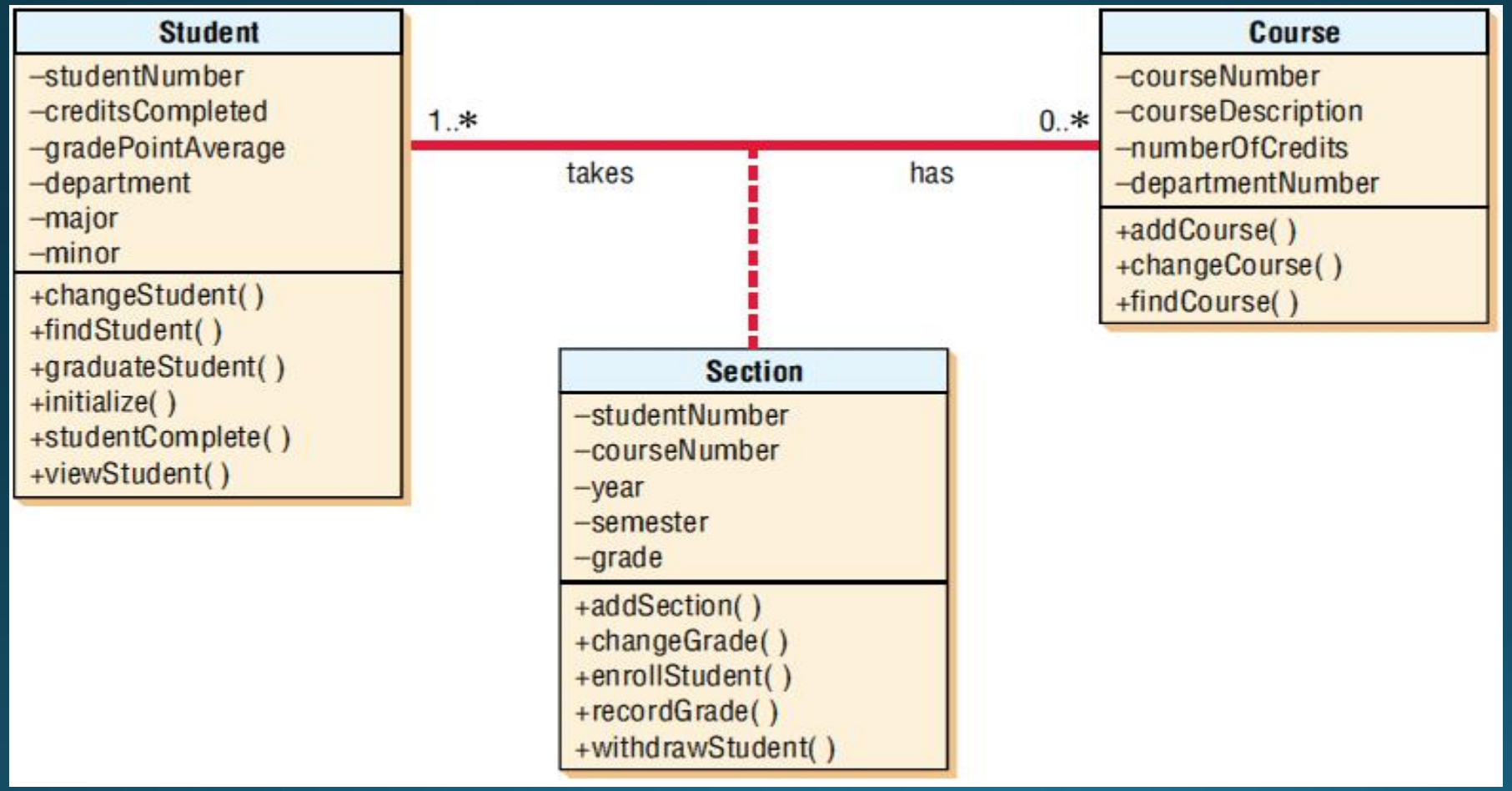
- Classes
- Attributes
  - Private
  - Public
  - Protected
- Methods
  - Standard
  - Custom



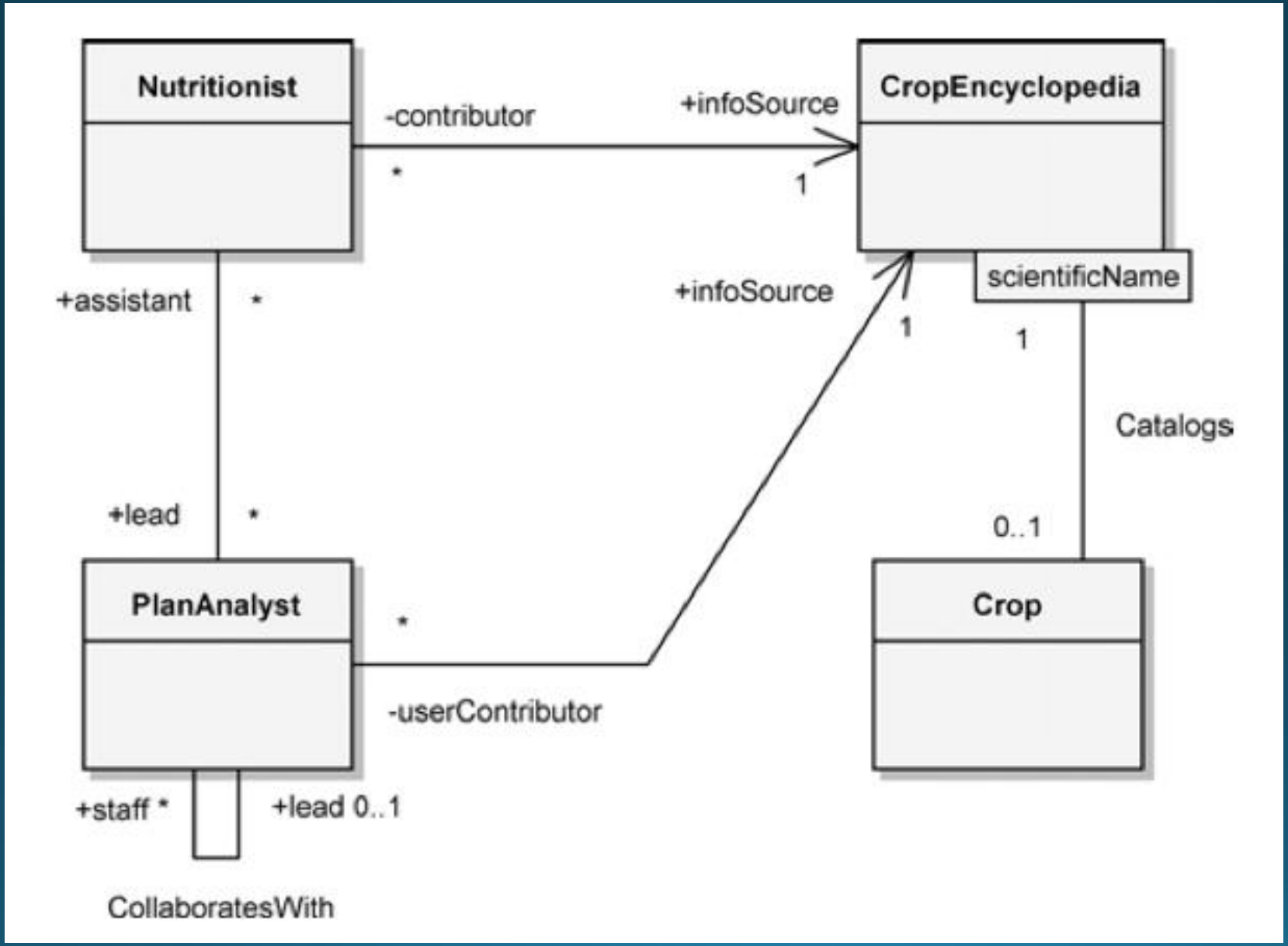
Stronger Aggregation: Composition (whole-part) relations



Example: Associative Class



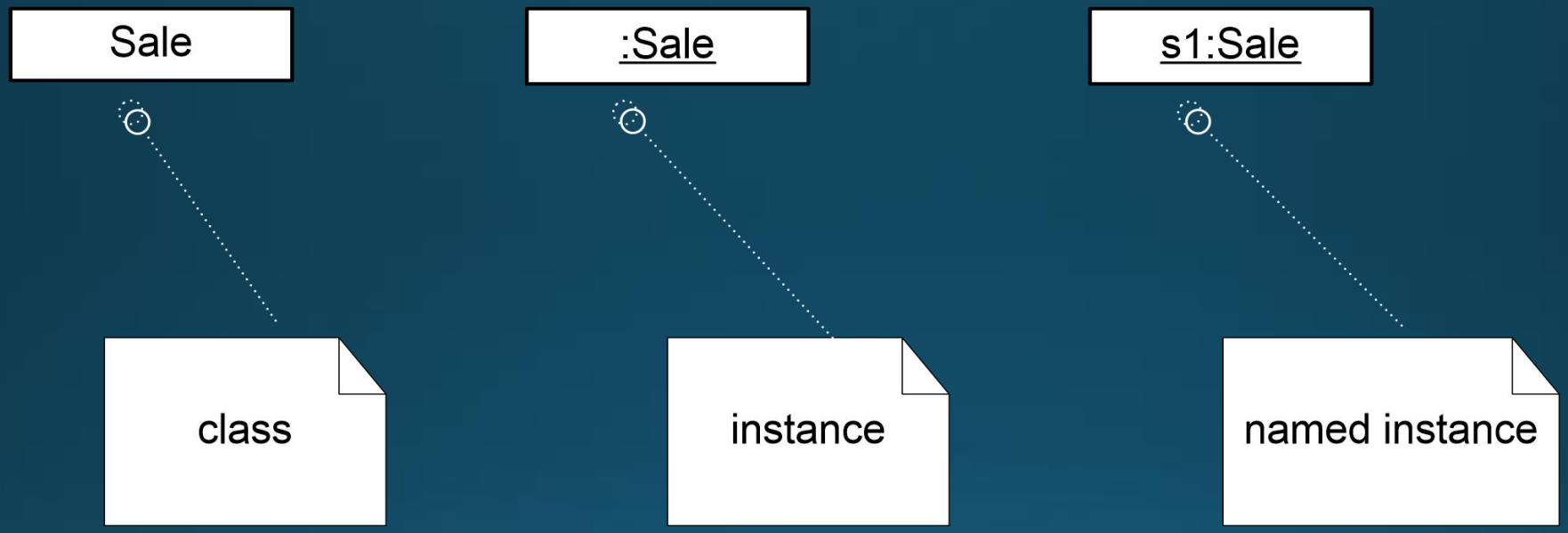
Association End Names and Qualifiers



Two kinds of UML Interaction Diagrams

- **Sequence** Diagrams
  - show object interactions arranged in time sequence, *vertically*
- **Communication** Diagrams
  - show object interactions arranged as a flow of objects and their links to each other, *numerically*
- Semantically equivalent, structurally different
  - Sequence diagram emphasize time ordering
  - Communication diagrams make object linkages explicit

Interaction Diagram Notation



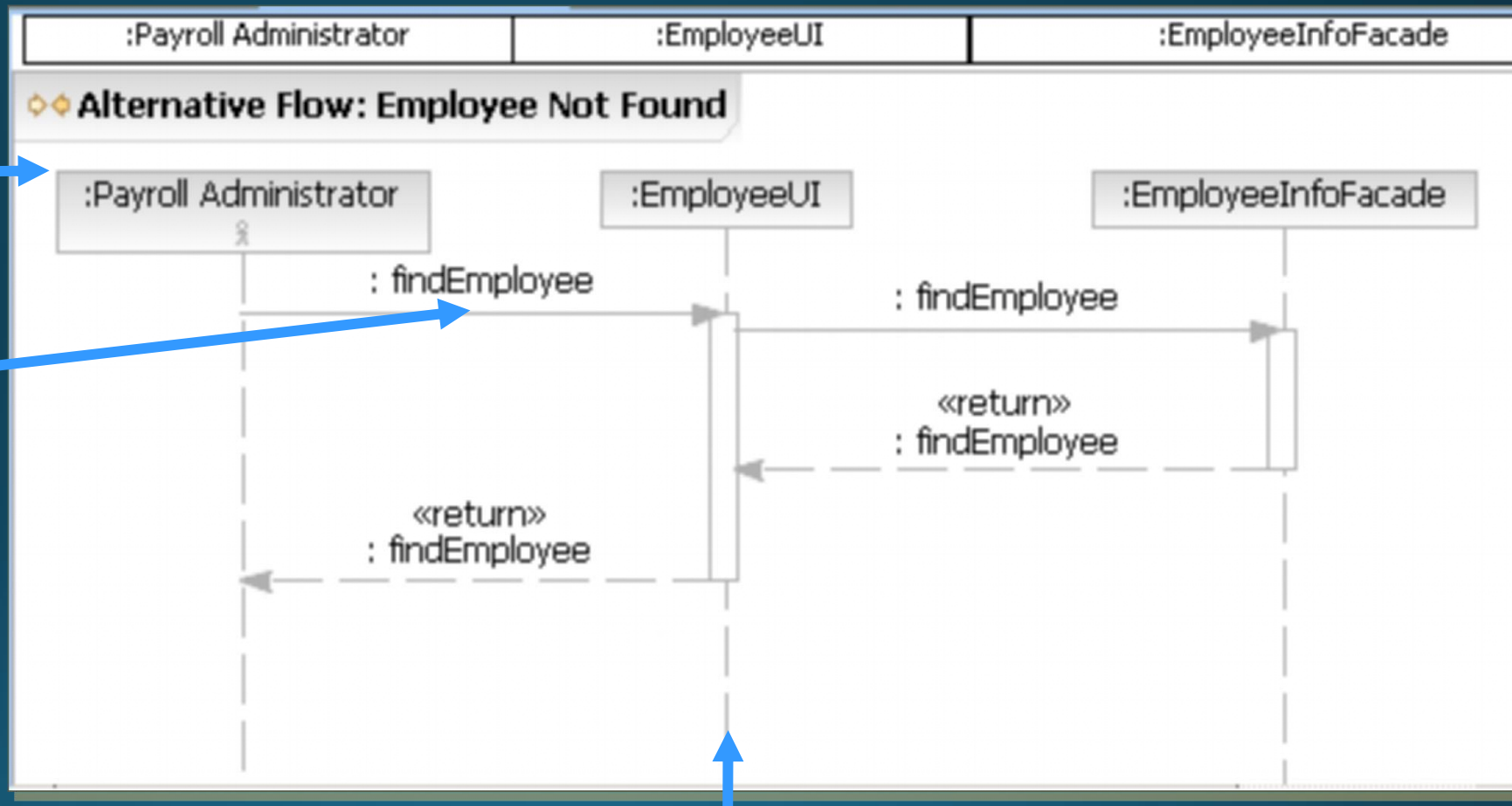
Which would you expect to find most often in Interaction diagrams?

What do you think of “:Sale” instead of “aSale”?

Sequence Diagram

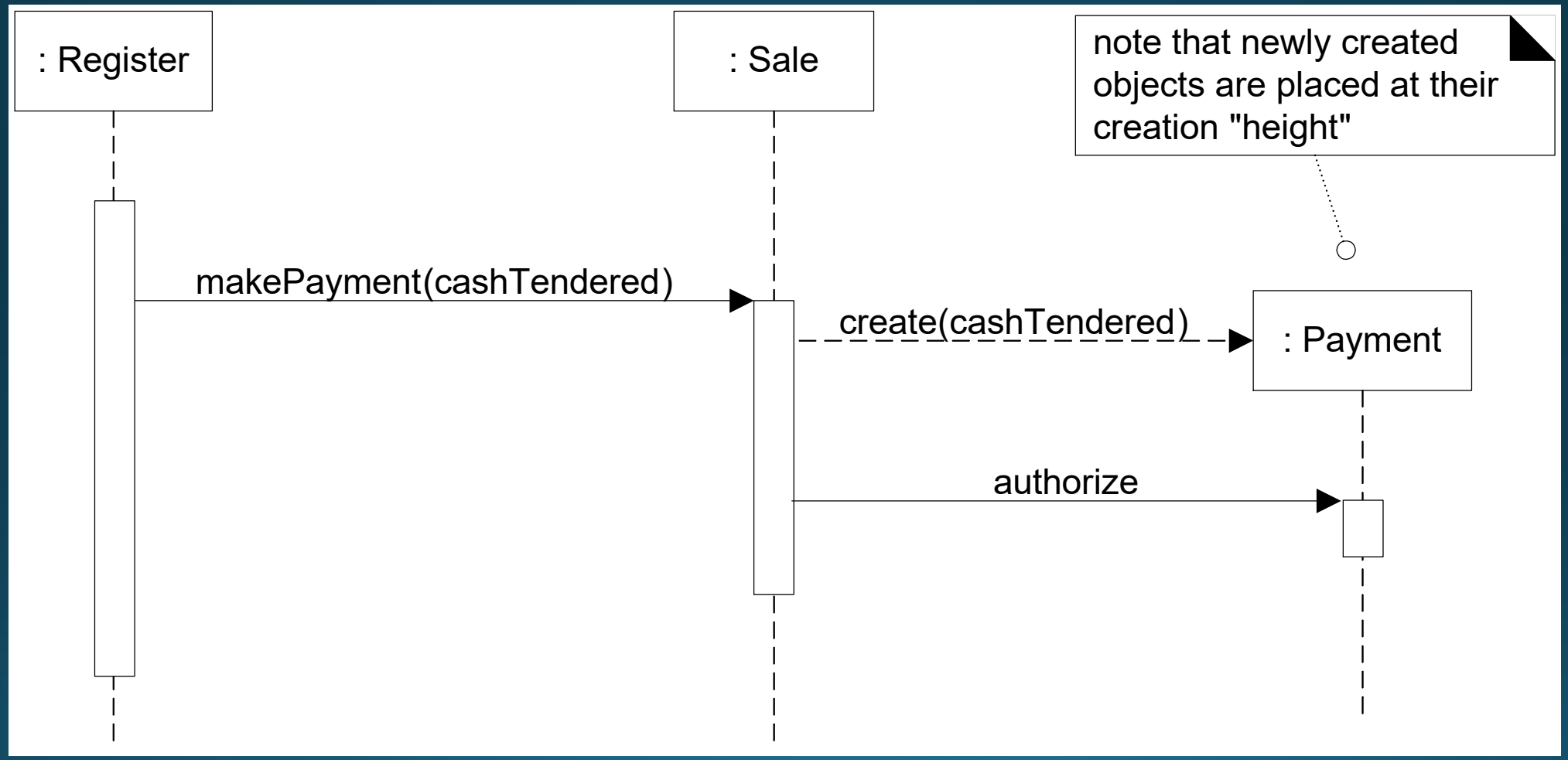
Object
Shows the object involved in the interaction

Messages
Show data exchanged between objects



Lifeline
Shows the life of the object

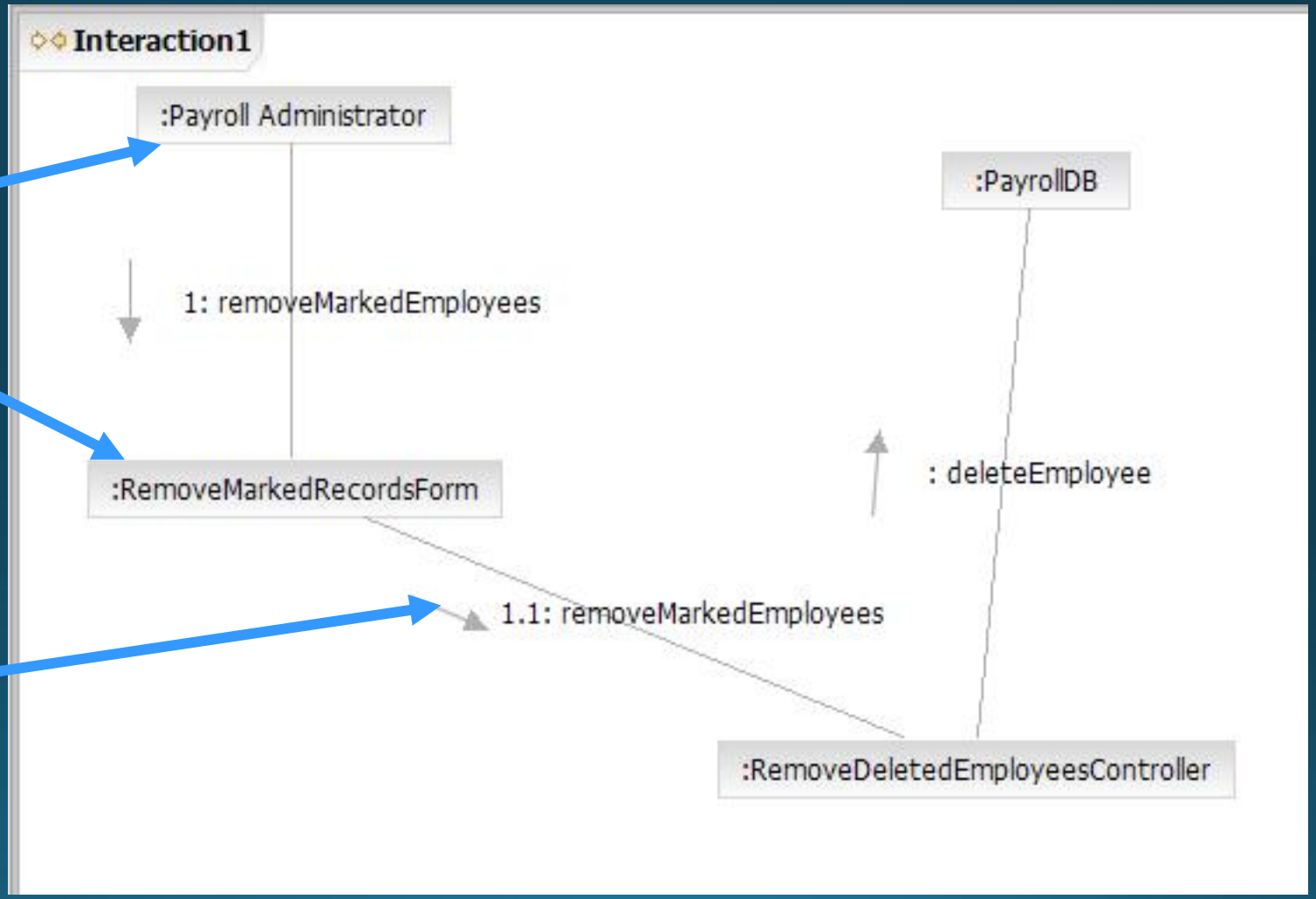
What does vertical placement communicate?



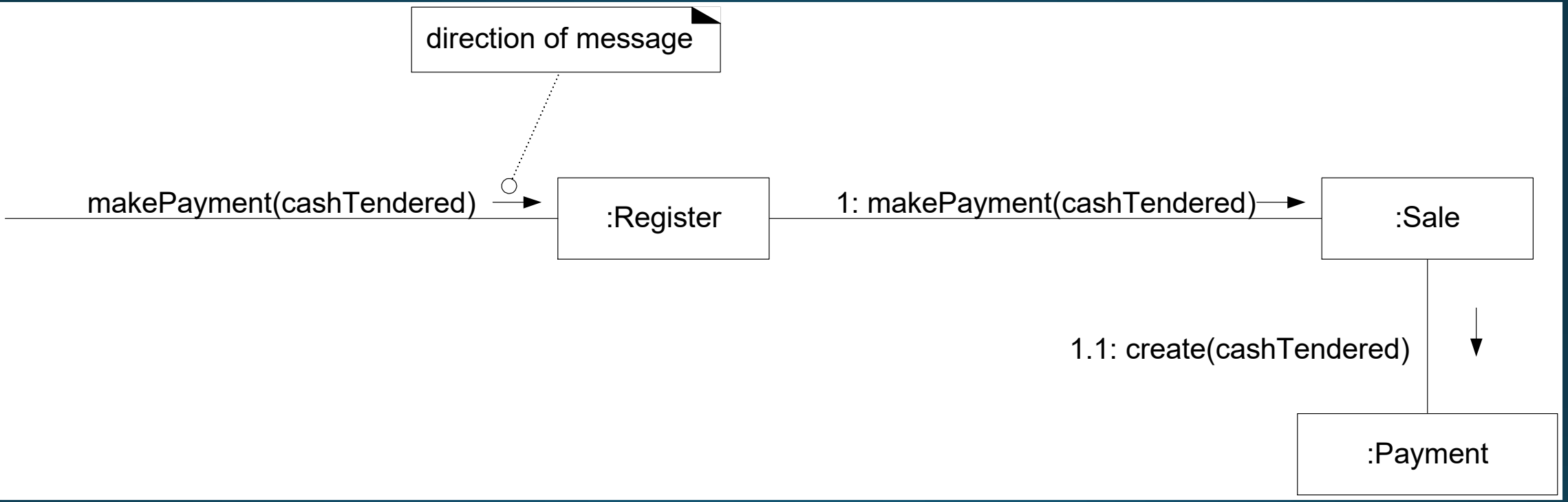
Communication Diagram

Objects
Shows the object involved in the interaction

Message
Shows data exchanged between objects



Communication Diagram Example: makePayment



What do the numbers communicate?

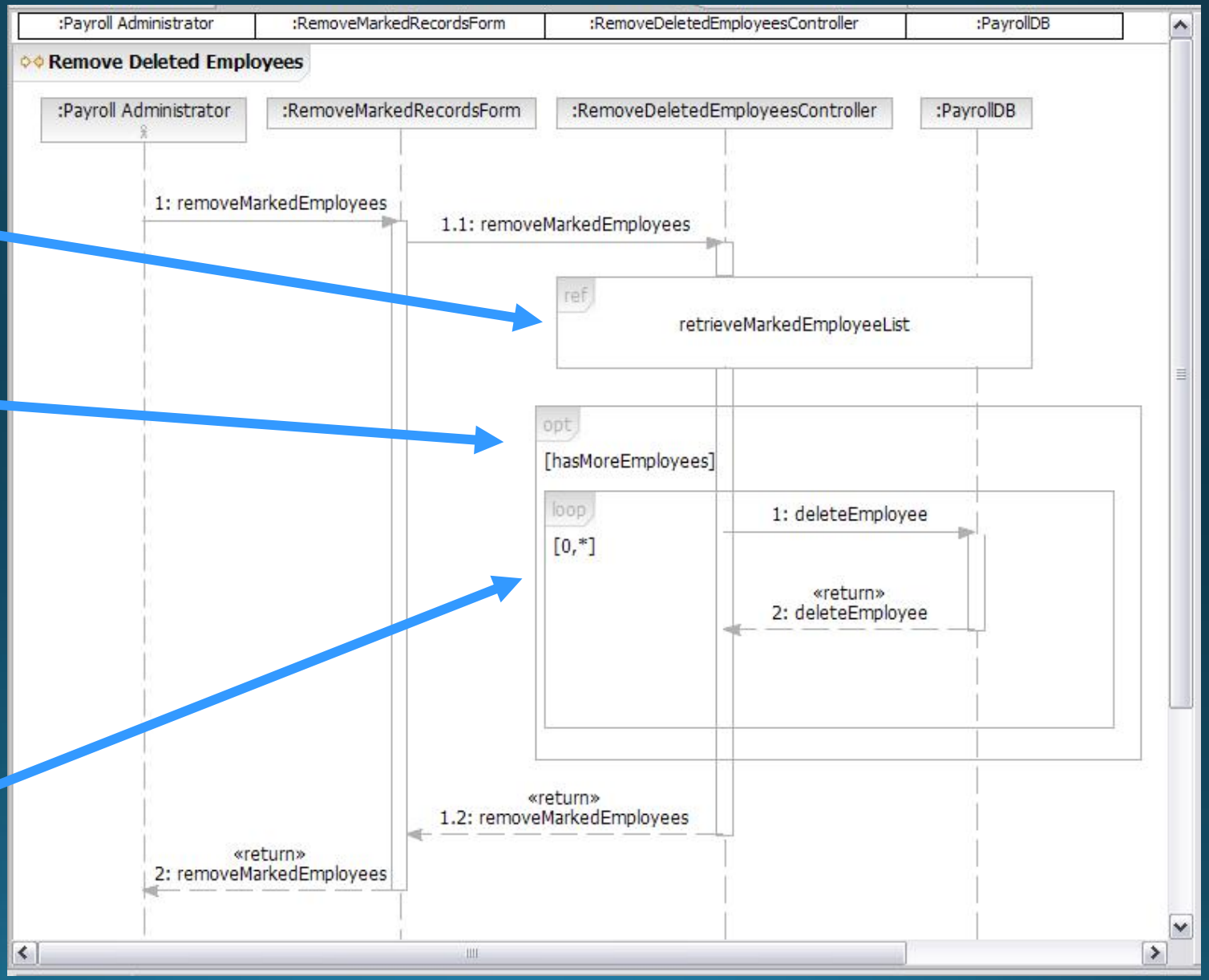
What does a create message communicate?

Sequence Diagram: Combined Fragments

Interaction Use (ref)
References another interaction

Optional Fragment (opt)
Executed if guard condition evaluates to true

Loop (loop)
Executed as long as the first guard condition evaluates to true



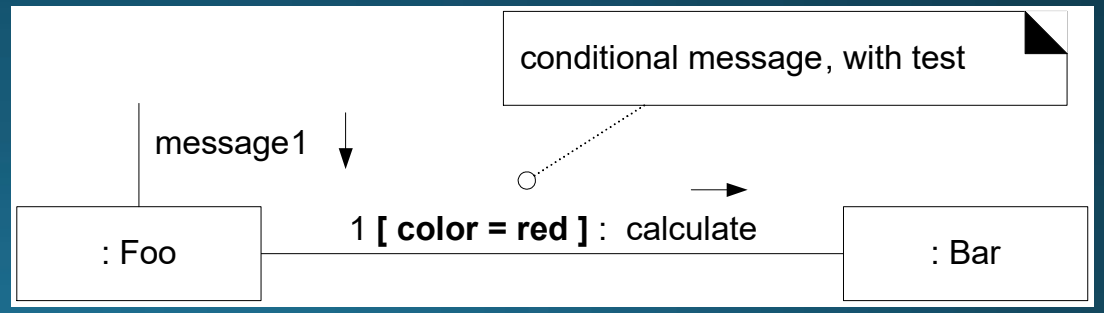
Communication (aka Collaboration) diagrams

- Objects are rectangular icons
  - e.g., Order Entry Window, Order, etc.
- Messages are arrows between icons
  - e.g., prepare()
- Numbers on messages indicate sequence
  - Also spatial layout helps show flow
- *Which do you prefer: sequence or communication?*

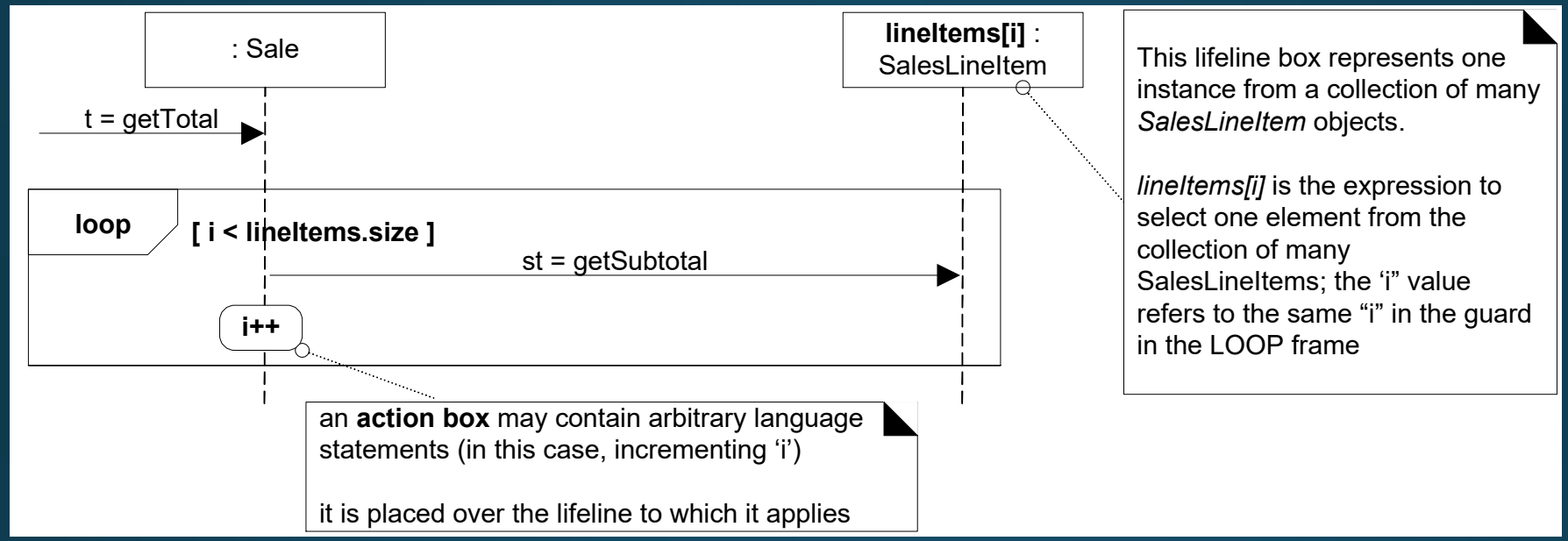
Control logic in Interaction Diagrams

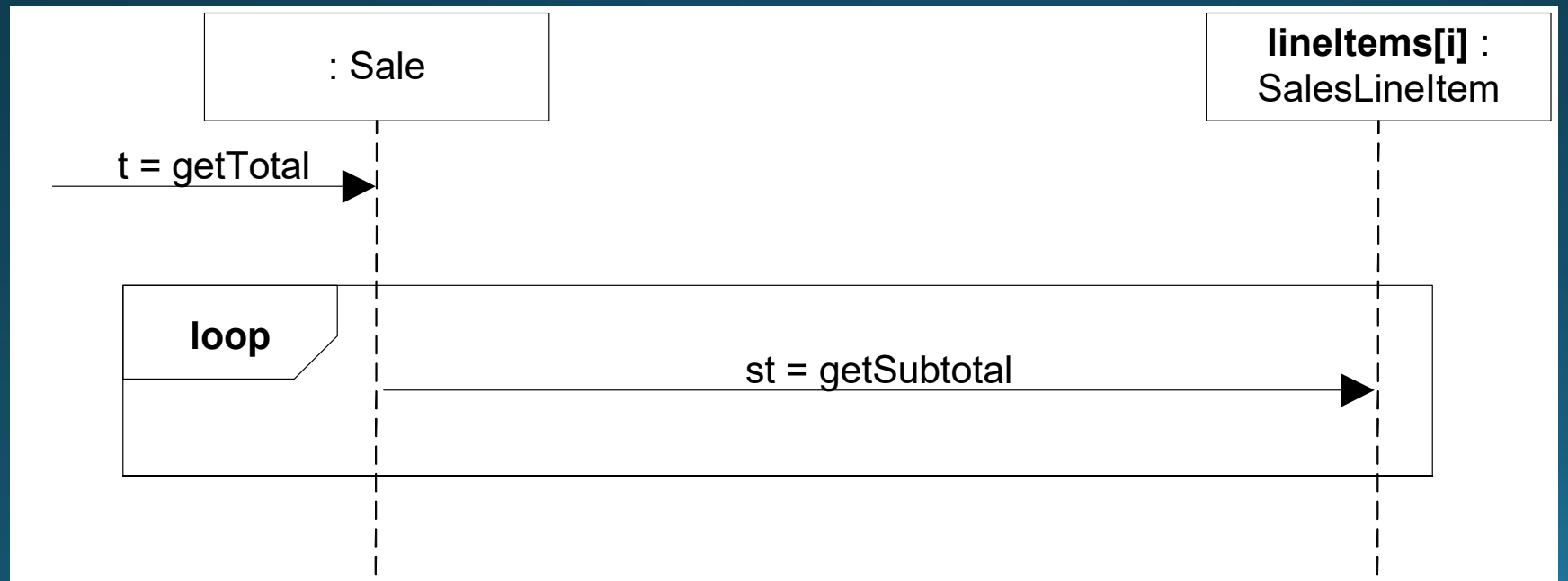
- Conditional Message
  - *[variable = value]* : message()
  - Message is sent only if clause evaluates to *true*
- Iteration (Looping)
  - ** [i := 1..N]*: message()
  - "*" is required; [...] clause is optional

Example: Logic in communication diagrams

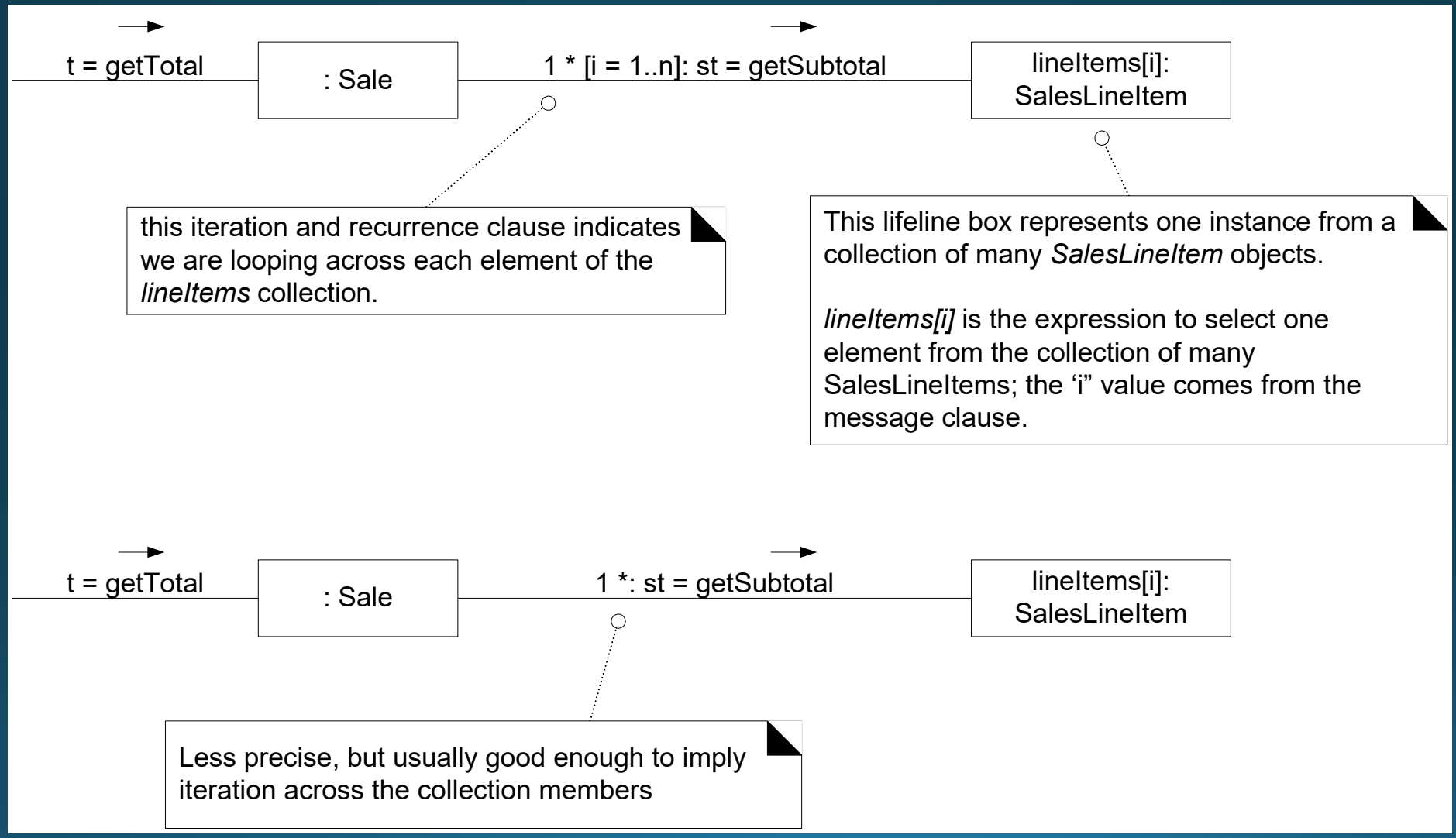


Loops in sequence diagrams

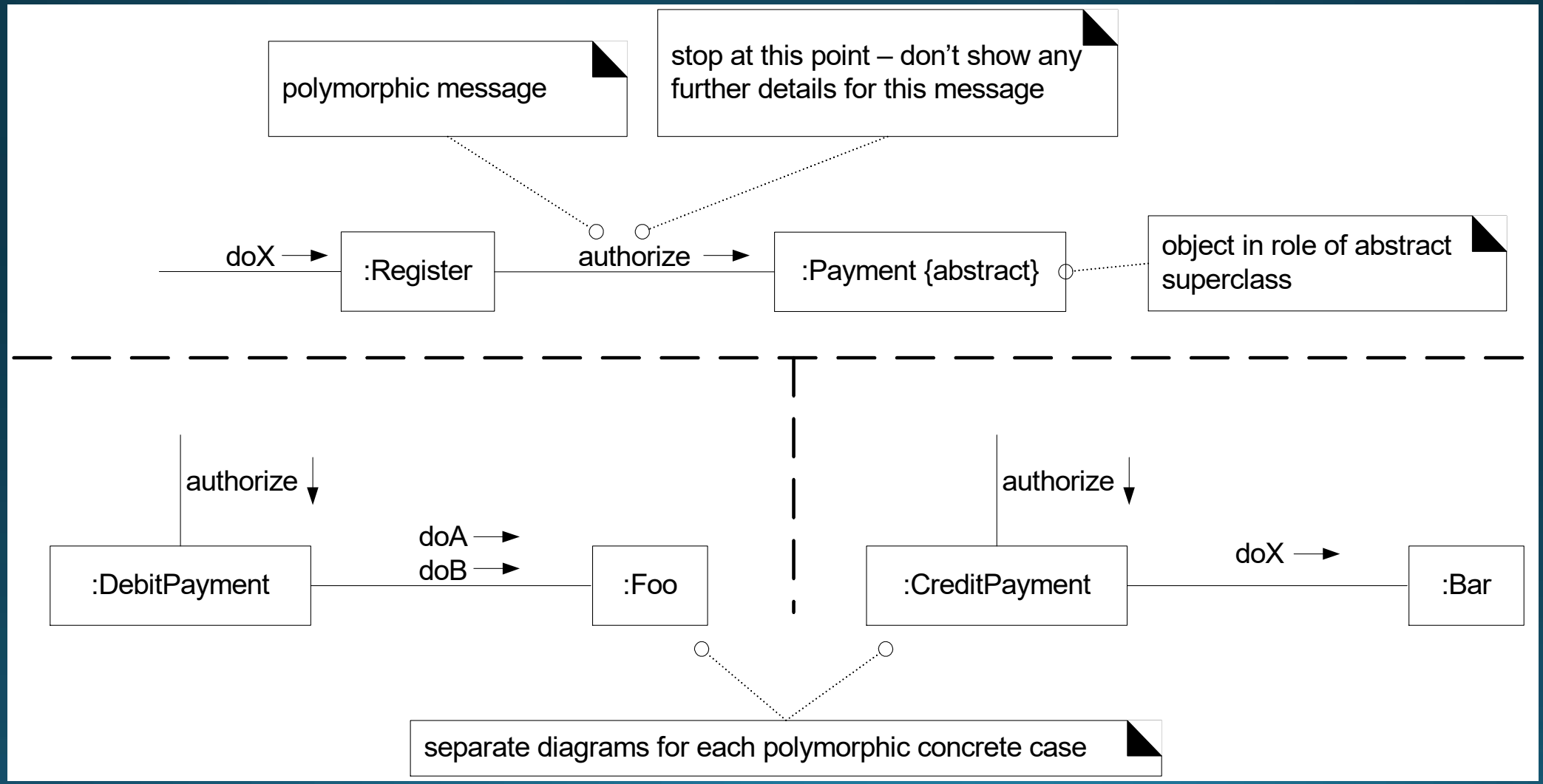




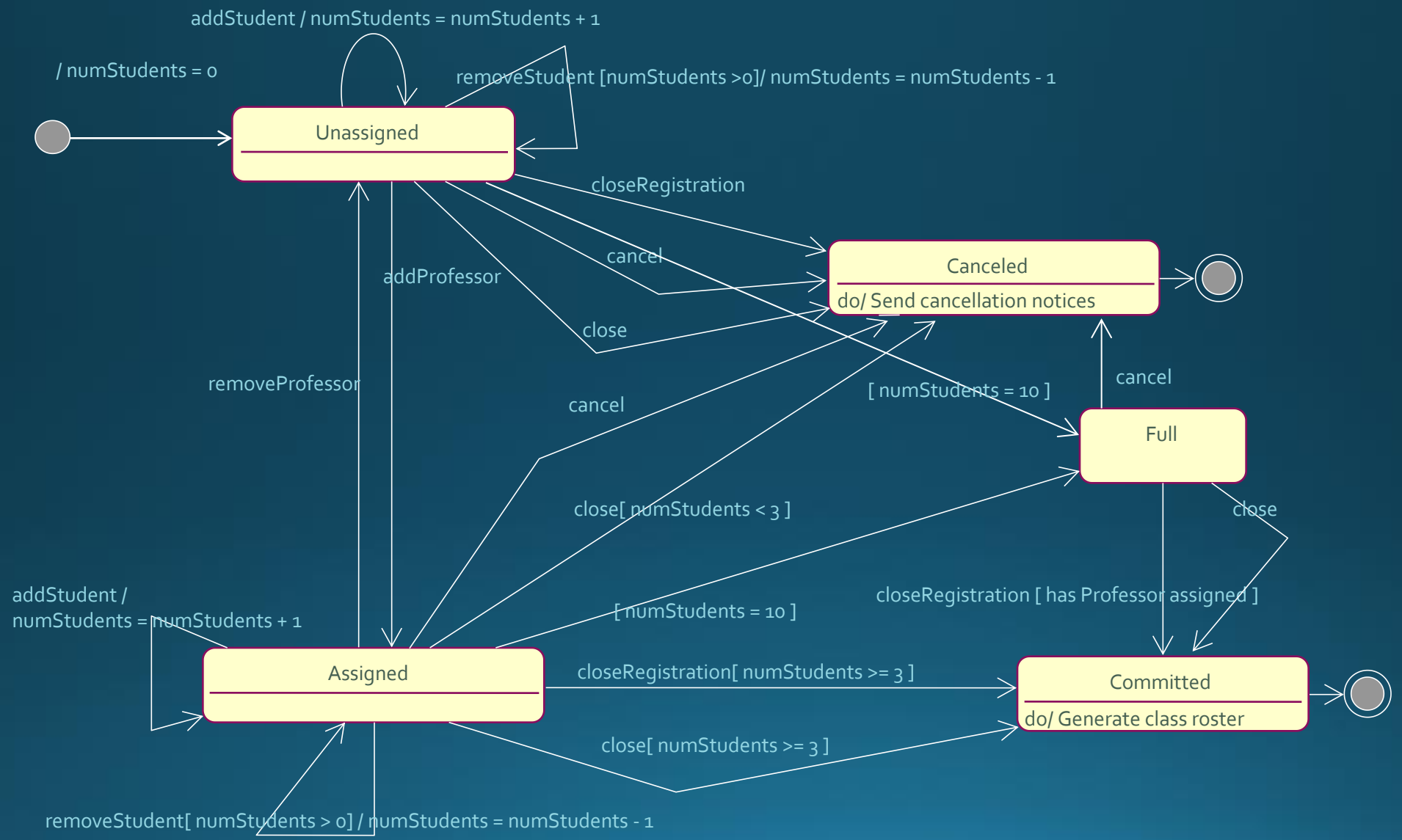
Iteration in communication diagrams



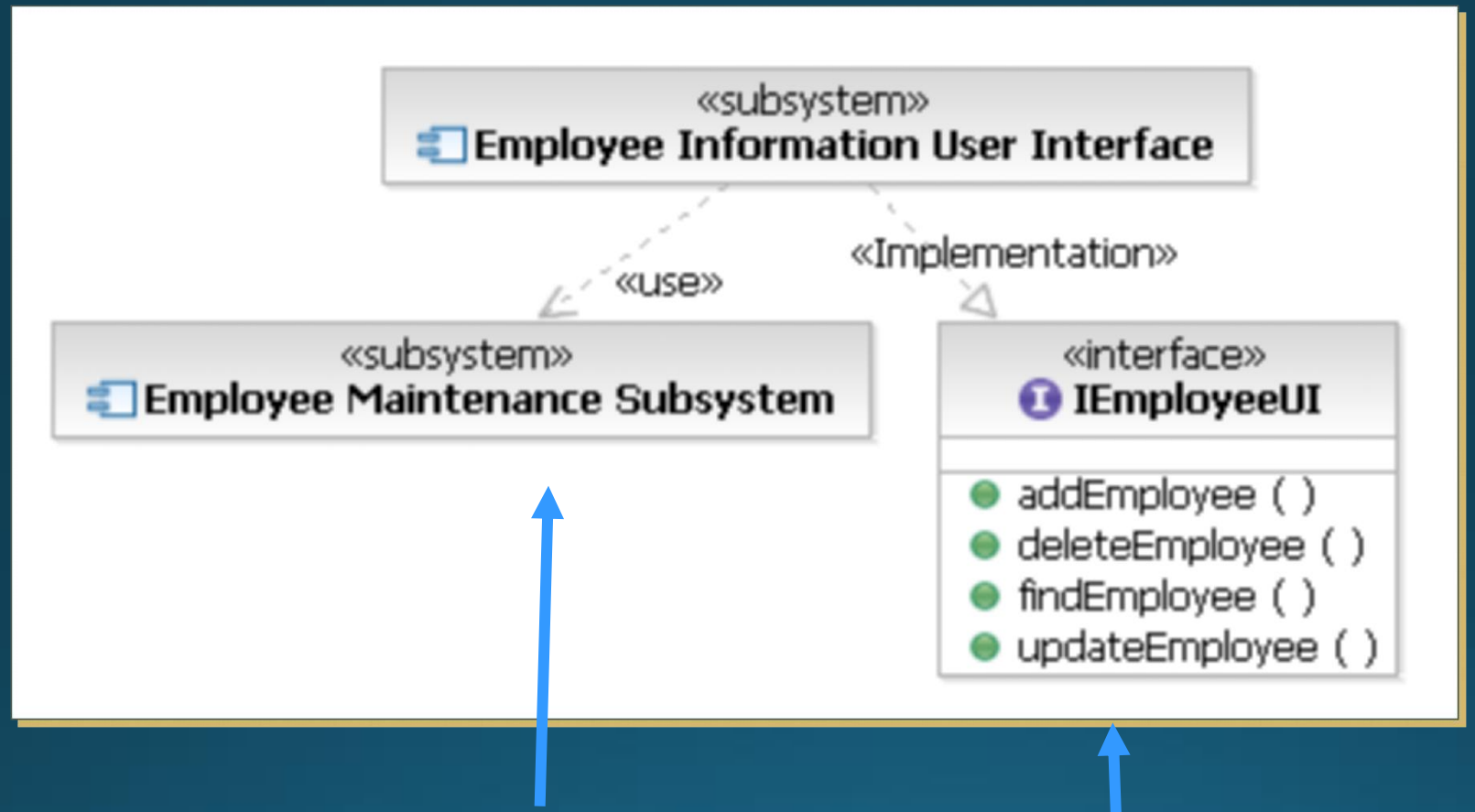
Polymorphism:
How is it shown in interaction diagrams?



State Machine Diagram Example



Component Diagram



Component
Modular parts of the system

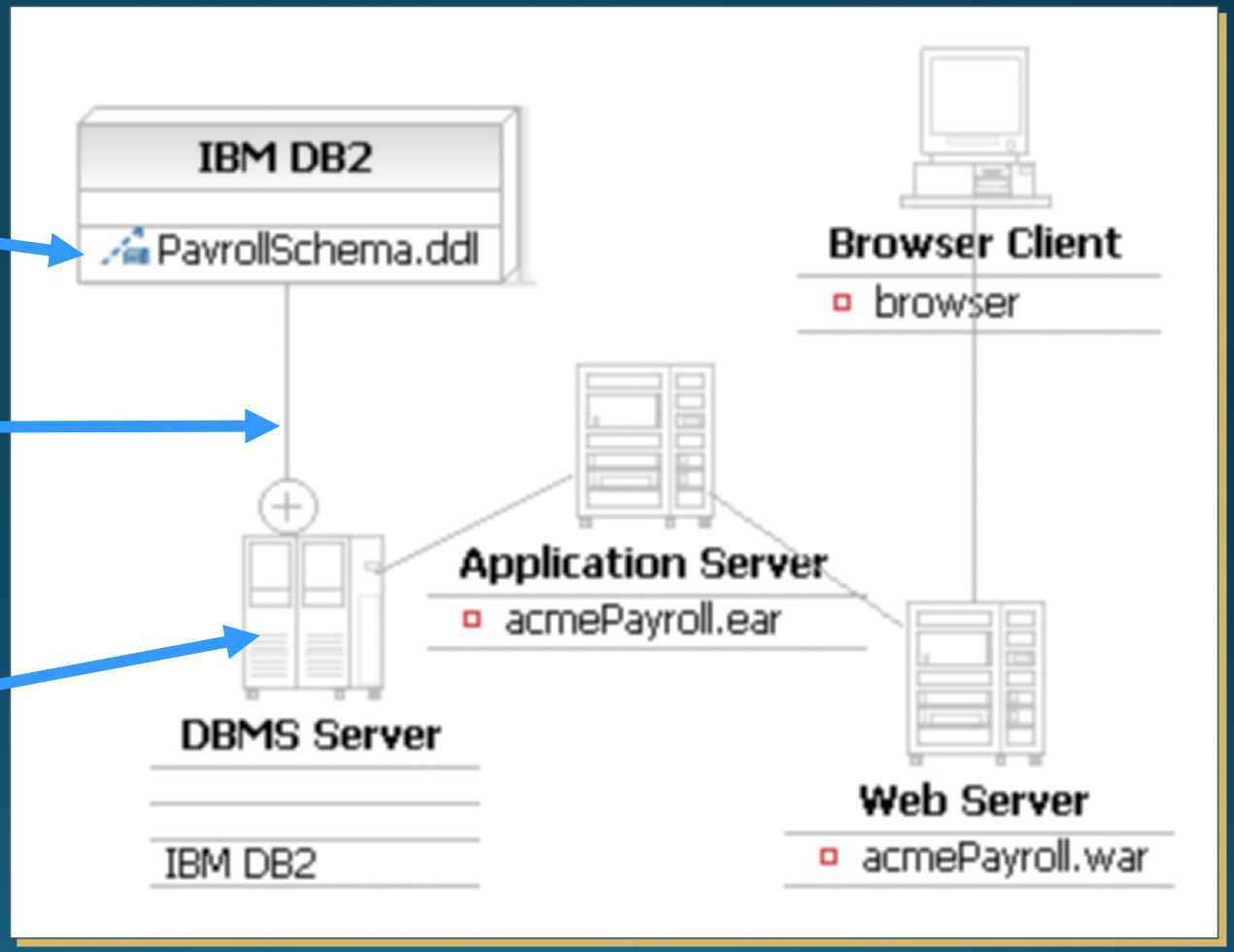
Class
Included to show implementation relationships.

Deployment Diagram

Artifact
Represents a physical file

Owned Element Relationship
Shows another way of
showing nested elements

Node
Represents a physical machine



Discussion: Leveraging Generative AI for OOAD with UML

agent.tongji.edu.cn/product/llm/mall/application/curapg92d2648liem8ig/chat

同济大学 TONGJI UNIVERSITY

Deepseek-R1-671B-0528

Conversational

Sessions Develop

刘岩

Generate PlantUML for the following: "1 company has 0..* departments. 1 department employs 1..* employees. A company has a name, address, email address and vat number. A department has a name and a unique identifier. An employee has a first name, last name, unique identifier, address, email address, salary, and a job title. A company shall be able to return a list of its employees. A company shall be able to find an employee by name. A company shall be able to find an employee by unique identifier."

Deepseek-R1-671B-0528

We are given a description of a system involving Company, Department, and Employee.
We need to model the classes and their relationships, and also include the sp ecified methods.
Steps:
1. Identify classes: Company, Department, Employee.
2. Identify attributes for each class:
- Company: name, address, email, vatNumber
- Department: name, uniqueIdentifier (we can use id for the unique identifier)
- Employee: firstName, lastName, uniqueIdentifier (id), address, email, salary, jobTitle
3. Relationships:
- "1 company has 0..* departments": so Company has a composition or aggre gation with Department (0 or more).
- "1 department employs 1..* employees": Department has a composition or a ggregation with Employee (1 or more).

Plantuml

```
@startuml
skinparam class {
    BackgroundColor PaleGreen
    ArrowColor #0078D7
    BorderColor #0078D7
}

class Company {
  - name: String
  - address: String
  - email: String
  - vatNumber: String
  + getEmployees(): List<Employee>
  + findEmployeeByName(name: String): Employee
  + findEmployeeById(id: String): Employee
}

class Department {
  - name: String
  - id: String {unique}
}
```