الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

مقرر بنيان البرمجيات	كلية الهندسة المعلوماتية	المحاضرة الأولى

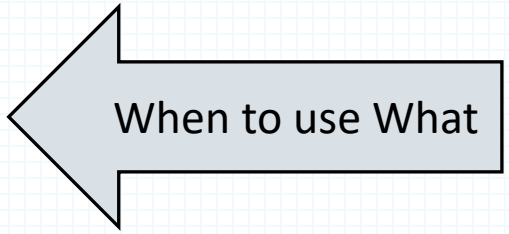
تذكرة بالمفاهيم الأساسية في هندسة البرمجيات

المراحل الأساسية – لغة النمذجة UML

د. رياض سنبل

Outline of the course

- **The course covers the following core topics:**
  1. Introduction to Software Architecture and Its Importance.
  2. General Principles in Software Design and Architecture.
  3. Main Architectural Styles:
     - Layered Architecture (and N-Tier),
     - Client-Server Architecture,
     - MVC and its variations.
     - Ports and Adapter
     - Microservices
     - etc
  4. The Concept of Clean Architecture.
  5. Describing and Documenting Software Architecture Using UML.



Outline of the course

- **Textbooks:**
- *Ingeno, Joseph. Software Architect's Handbook. Packt Publishing Ltd, 2018.*
- *Martin, Robert C., Clean Architecture – A Craftsman's Guide to Software Structre and Design, Pearson , 2018.*

What do you know about Software Engineering?

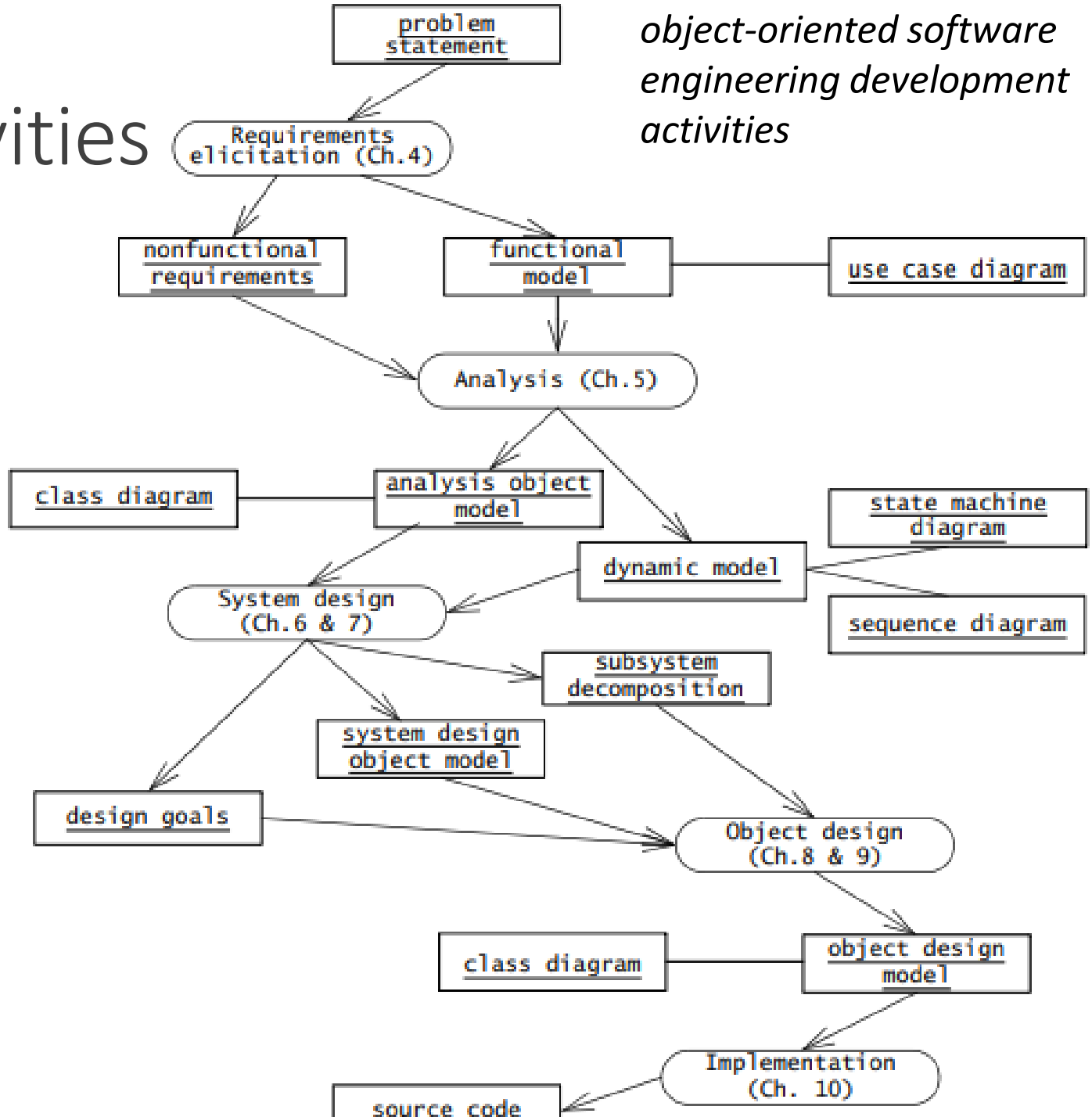
- **Software engineering** is the branch of computer science that deals with the **design, development, testing, and maintenance of software applications**.
- Software engineering is the application of engineering principles to the development of software systems.
- What are the main SE Development Activities?

The Context: SE Development Activities

1) Requirements Elicitation.
2) Analysis.
3) System Design.
4) Object Design.
5) Implementation.
6) Testing.

- Object-oriented software engineering is **iterative**; that is, activities can occur in parallel and more than once

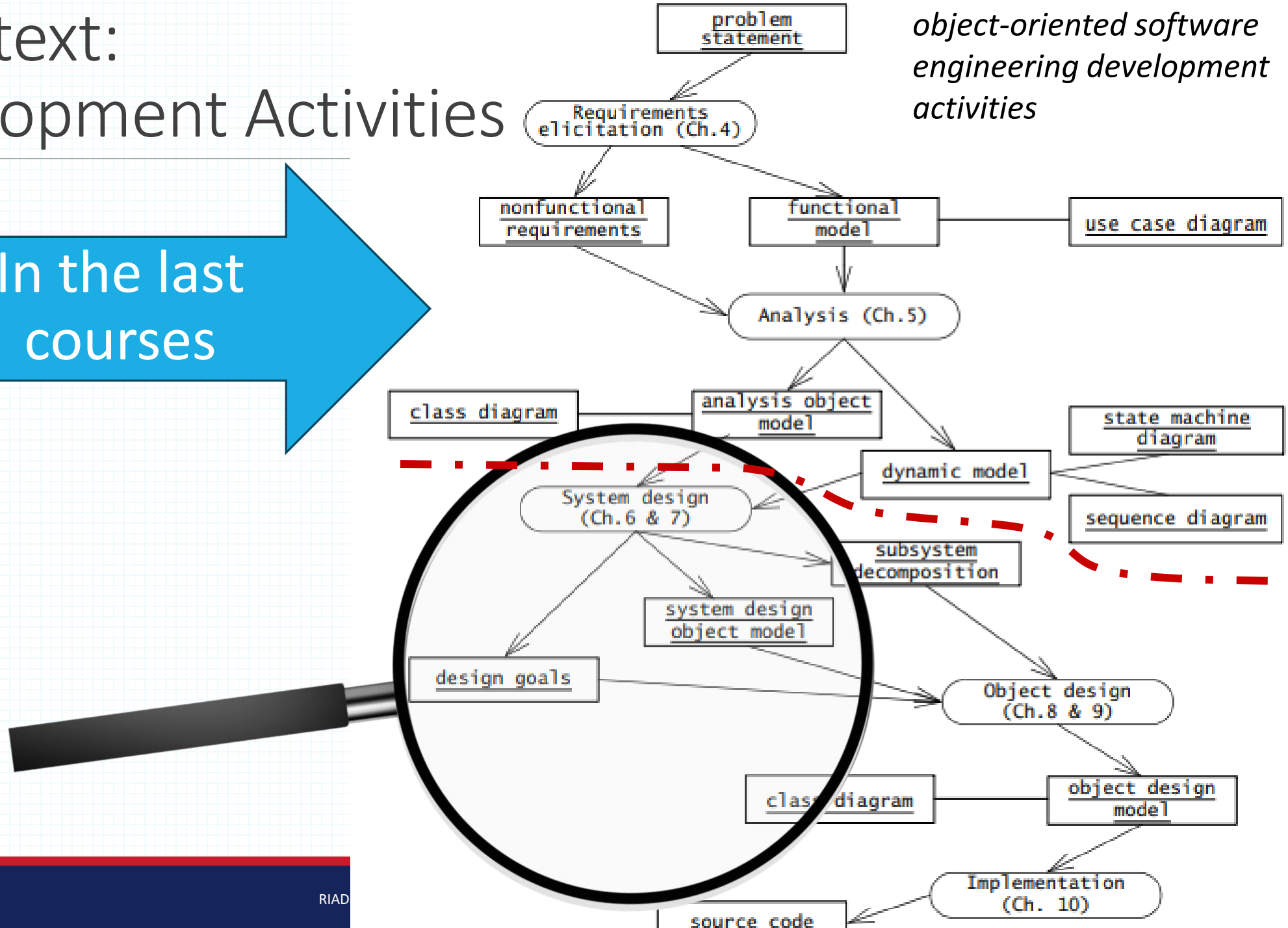
object-oriented software engineering development activities



The Context: SE Development Activities

object-oriented software engineering development activities

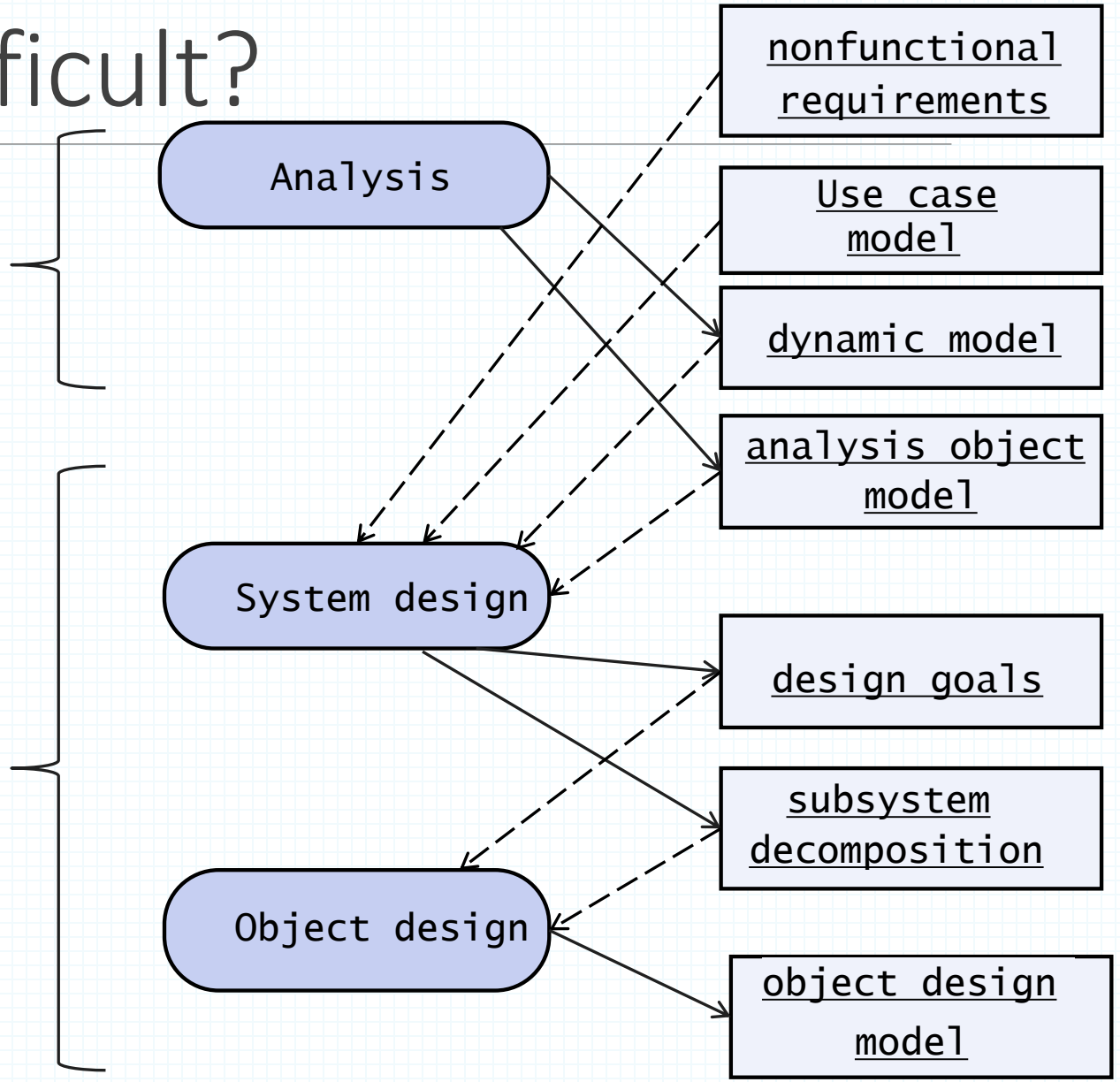
In the last courses



Why is Design so Difficult?

Focuses on the application domain

Focuses on the solution domain



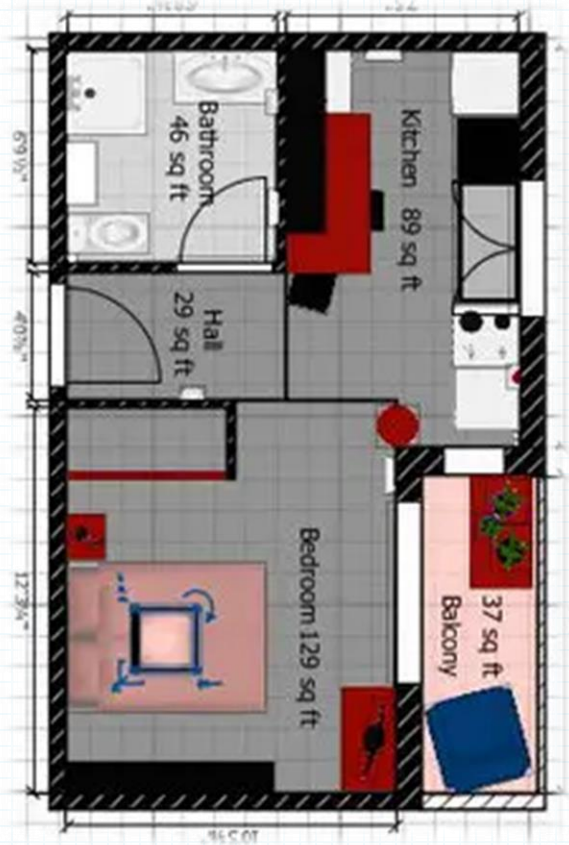
The solution domain is changing very rapidly

- *Halftime knowledge in SE ≈ 3-5 years*
- *Cost of hardware rapidly sinking*
- *Design knowledge is a moving target*

Modeling with UML- Quick Overview

What is modeling?

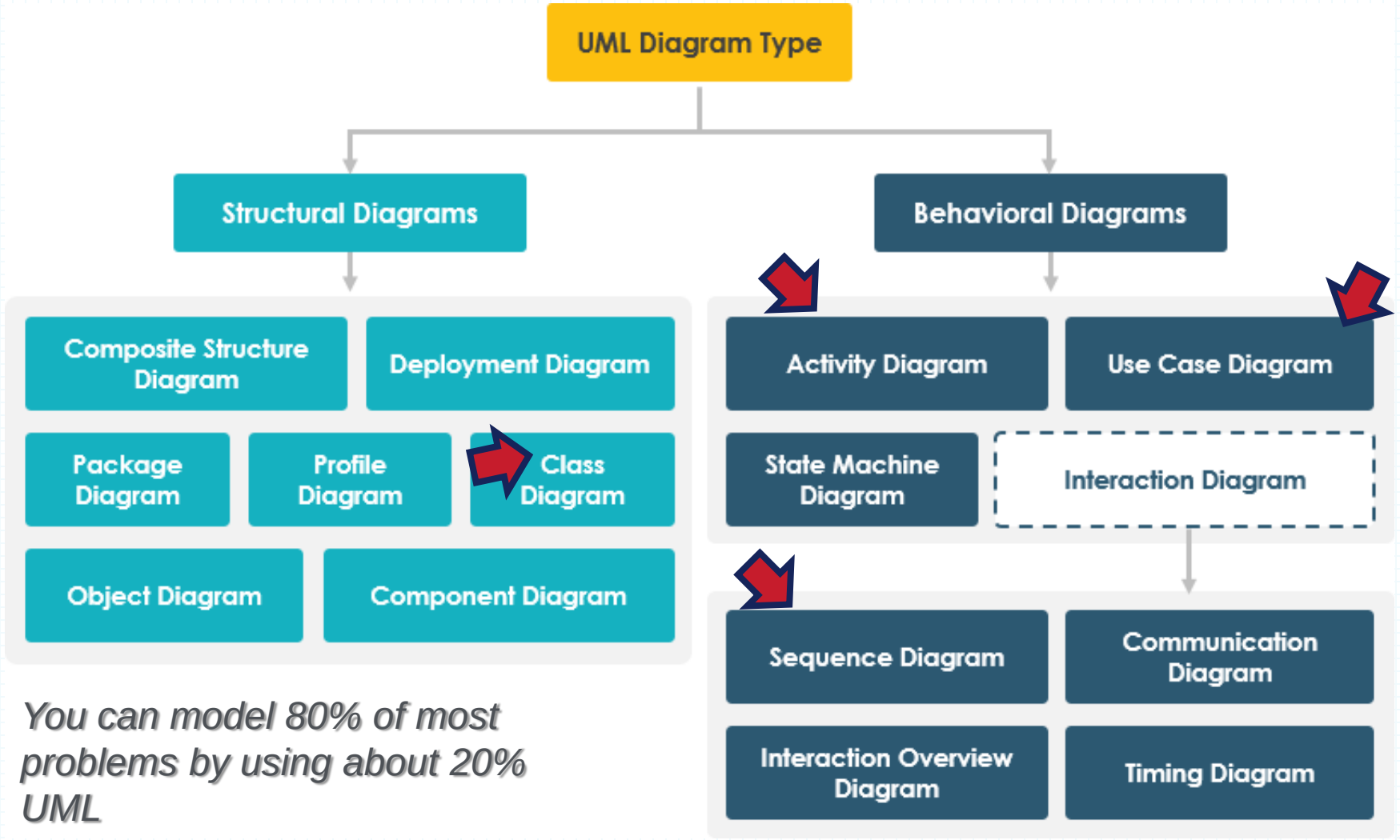
- the process of building an **abstraction** of reality.
- It involves creating a **simplified** representation that captures essential features of a system.
- The goal is to **make the complex reality more** understandable, manageable, and analyzable.



Significant to the problem at hand i.e. depends on the purpose of the model

for different types of stakeholders

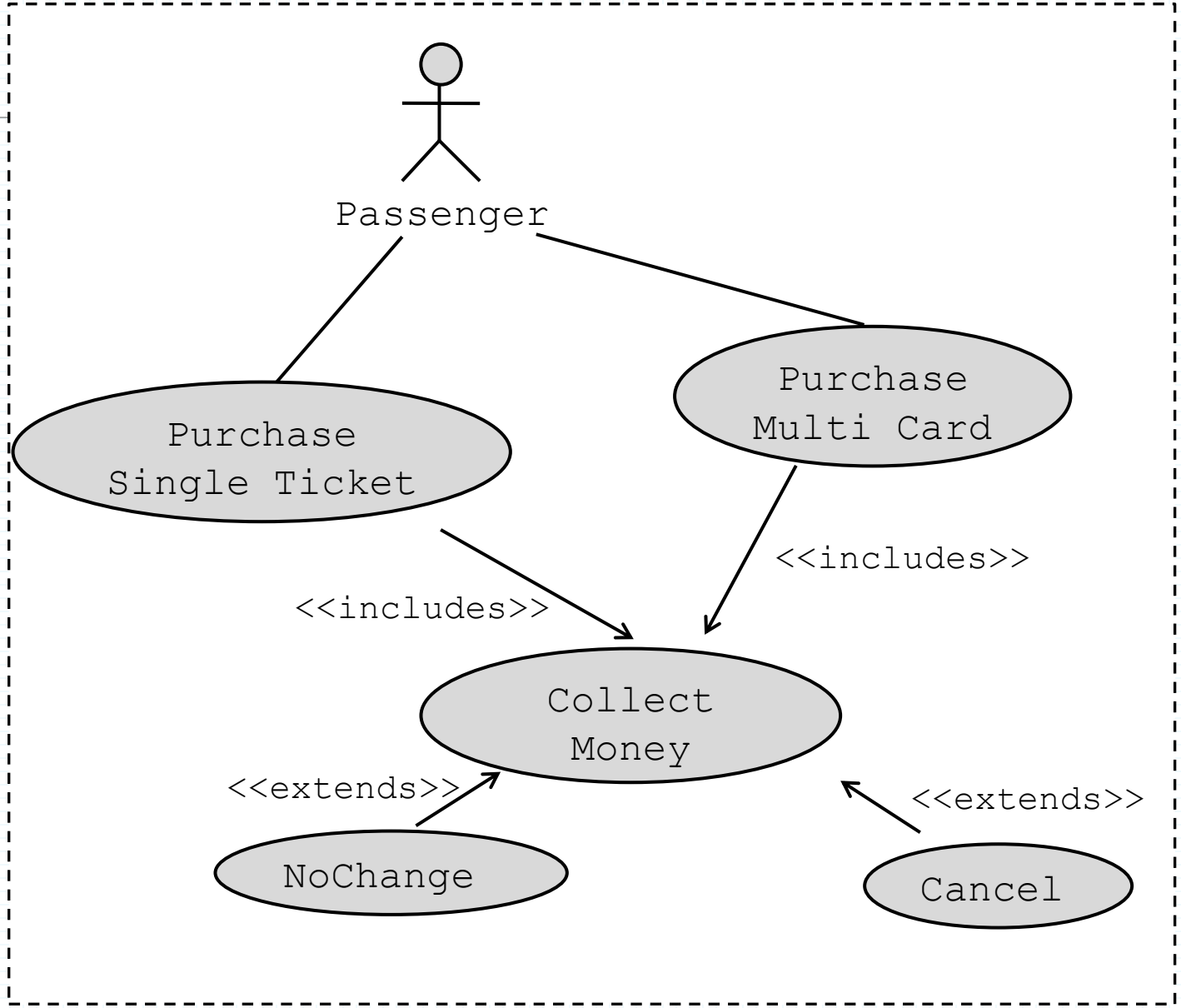
UML diagrams



You can model 80% of most problems by using about 20% UML

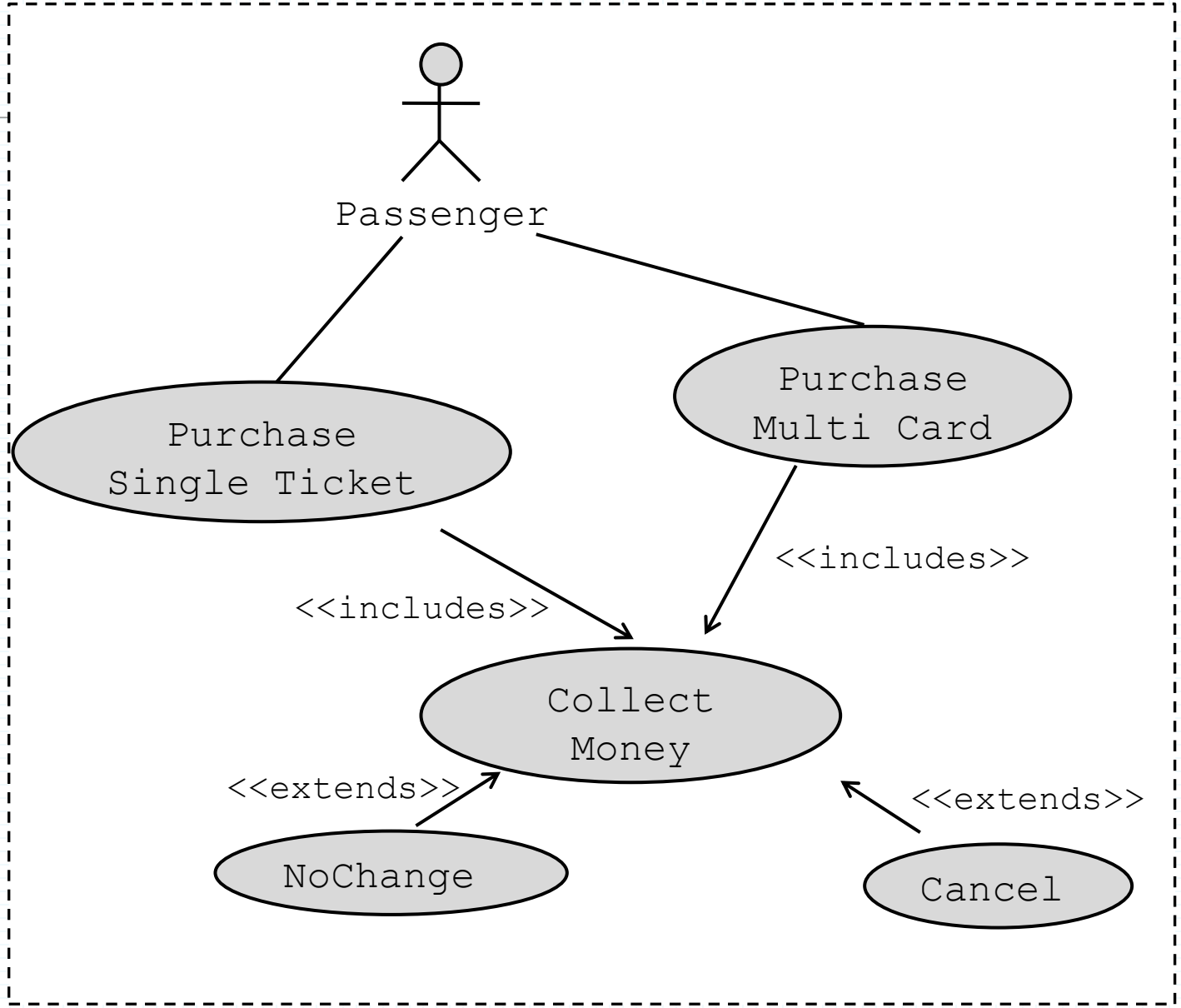
Use Case Diagram

- Why use a UC diagram?
  - *Represent interacts with people, organizations, or external systems*
  - *Represent the scope of your system*
- Actors?
  - *External objects that produce or consume data*
  - *a person, an organization, or an outside system*



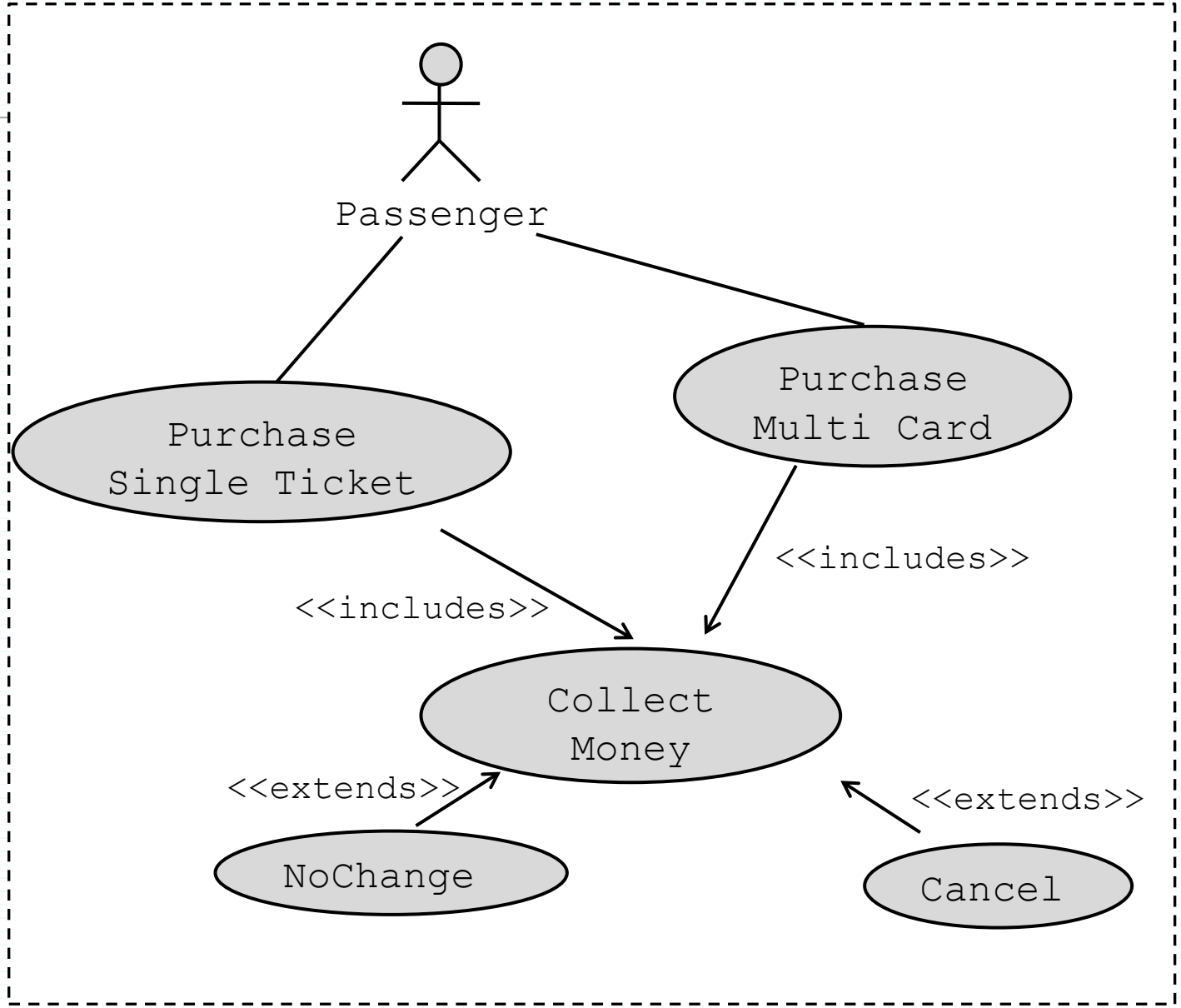
Use Case Diagram

- **There can be 5 relationship types in a use case diagram**.
  - Association between actor and use case
  - Generalization of an actor
  - Extend between two use cases
  - Include between two use cases
  - Generalization of a use case



Use Case Diagram

- **Some Common Mistakes**
  - Extend vs Include?
  - Generalize vs Extend?
  - primary actor vs. secondary actor?



Use Case Description

Name: `Purchase ticket`

Participating actor: `Passenger`

Entry condition:

- `Passenger` standing in front of ticket distributor.
- `Passenger` has sufficient money to purchase ticket.

Exit condition:

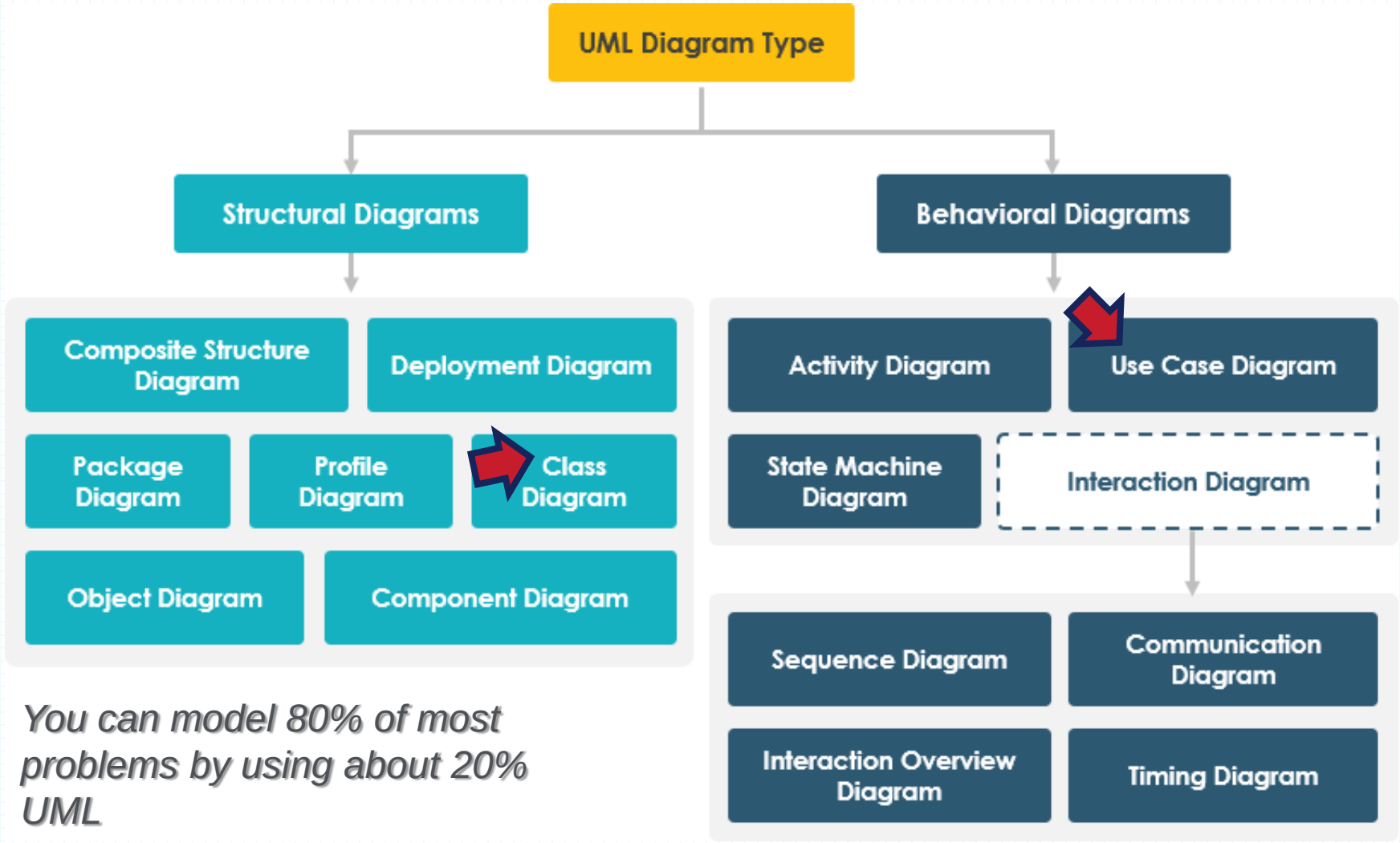
- `Passenger` has ticket.

Event flow:

1. `Passenger` selects the number of zones to be traveled.
2. Distributor displays the amount due.
3. `Passenger` inserts money, of at least the amount due.
4. Distributor returns change.
5. Distributor issues ticket.

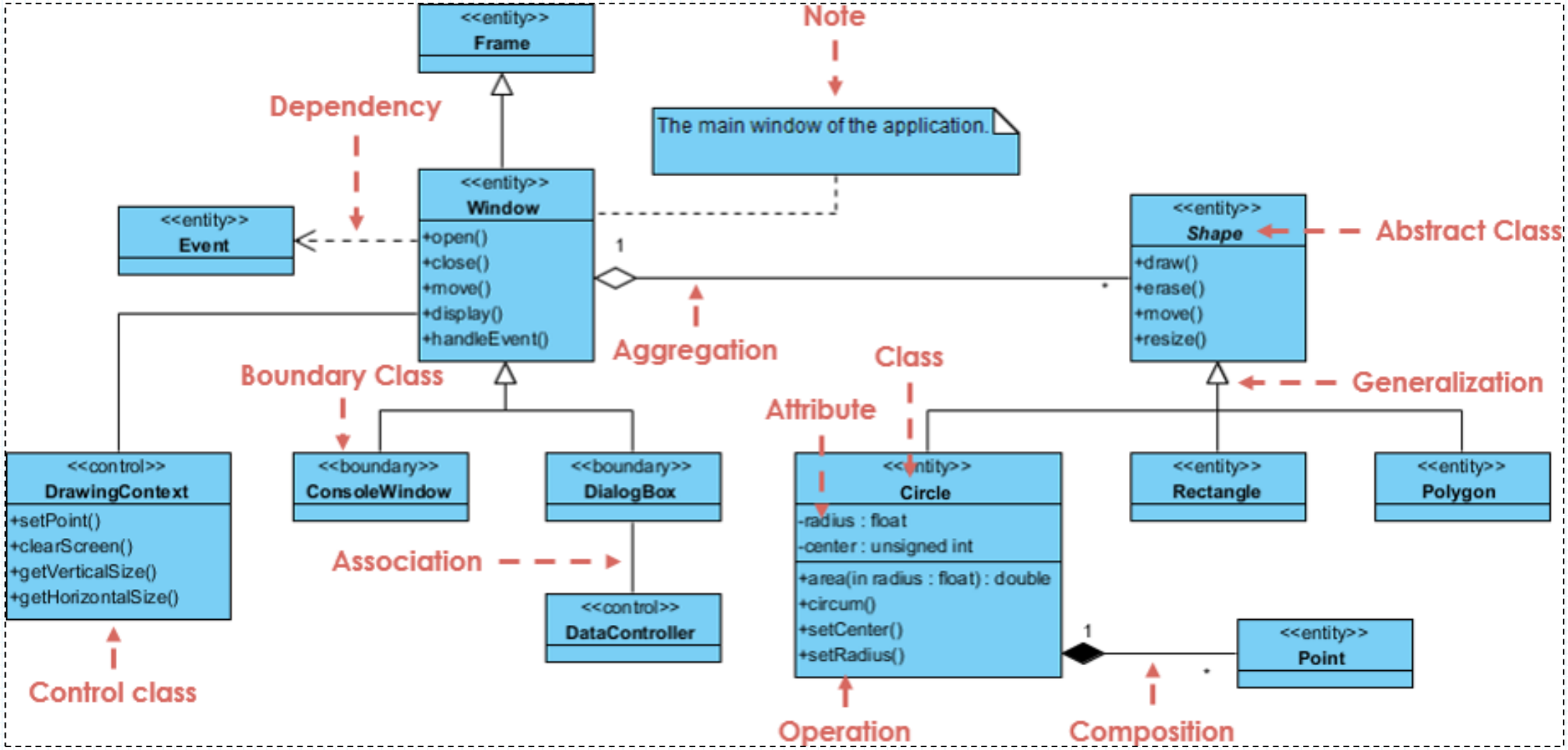
Exceptional cases!

UML diagrams



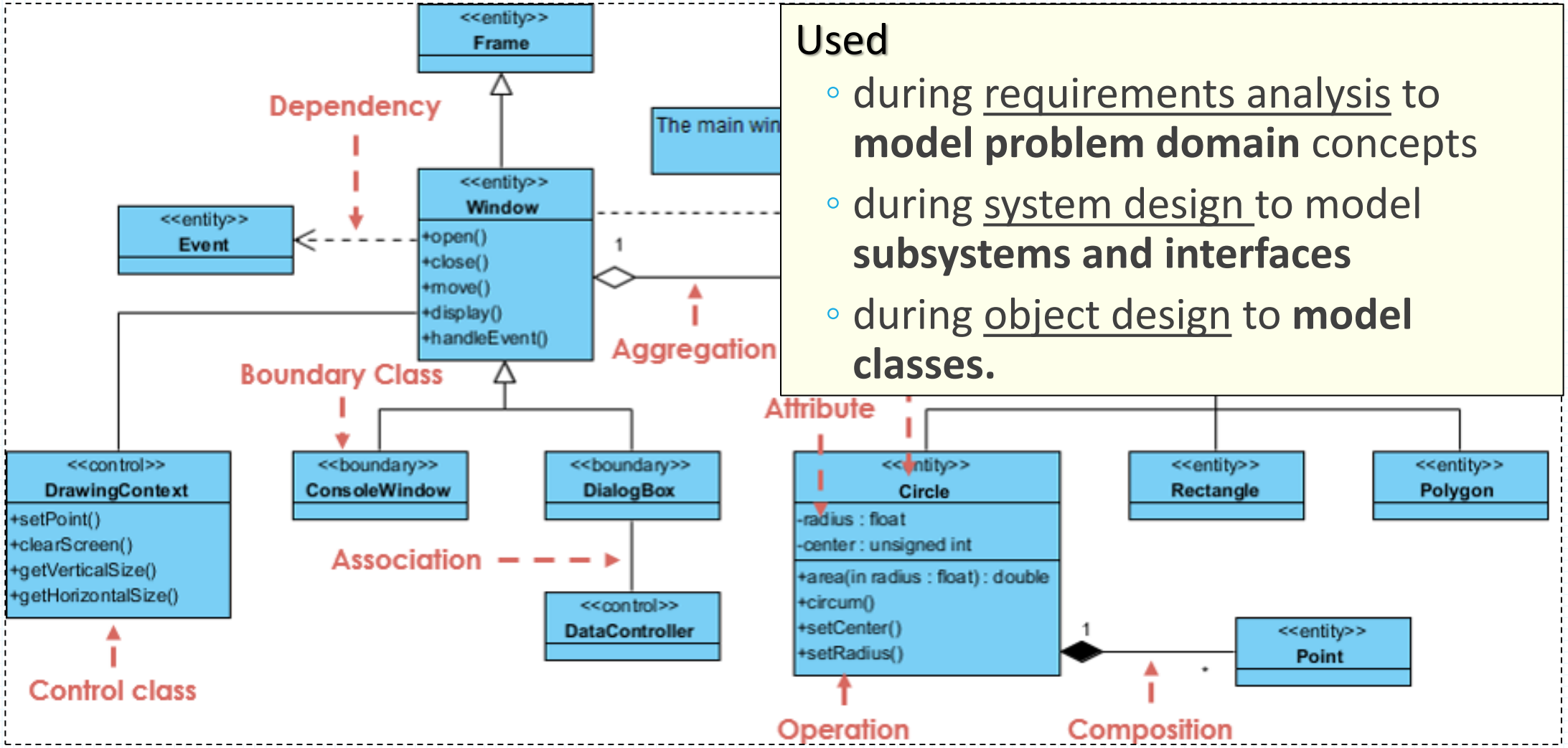
You can model 80% of most problems by using about 20% UML

Class Diagrams



Class Diagrams

Class diagrams represent the structure of the system.



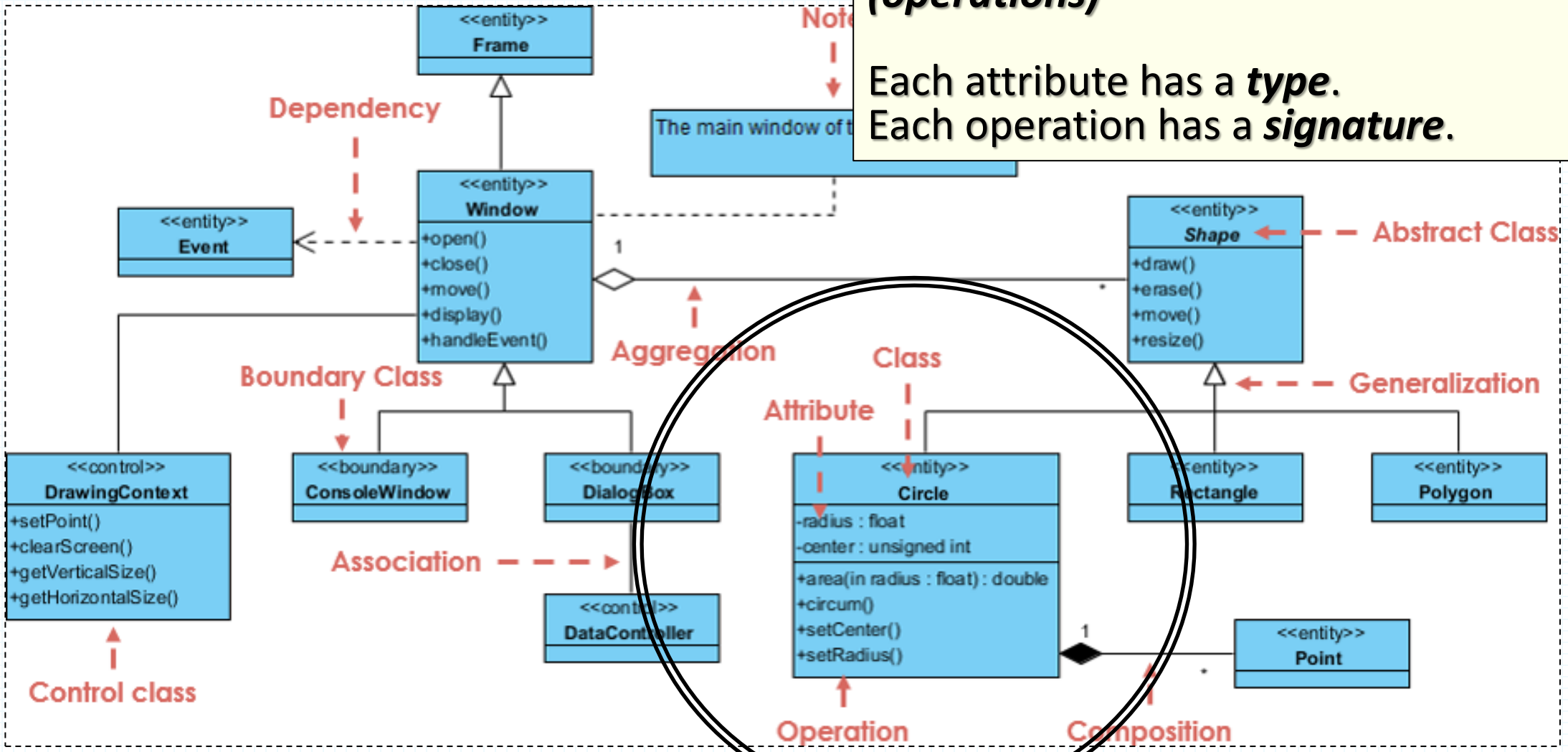
Used

- during requirements analysis to **model problem domain** concepts
- during system design to model **subsystems and interfaces**
- during object design to **model classes.**

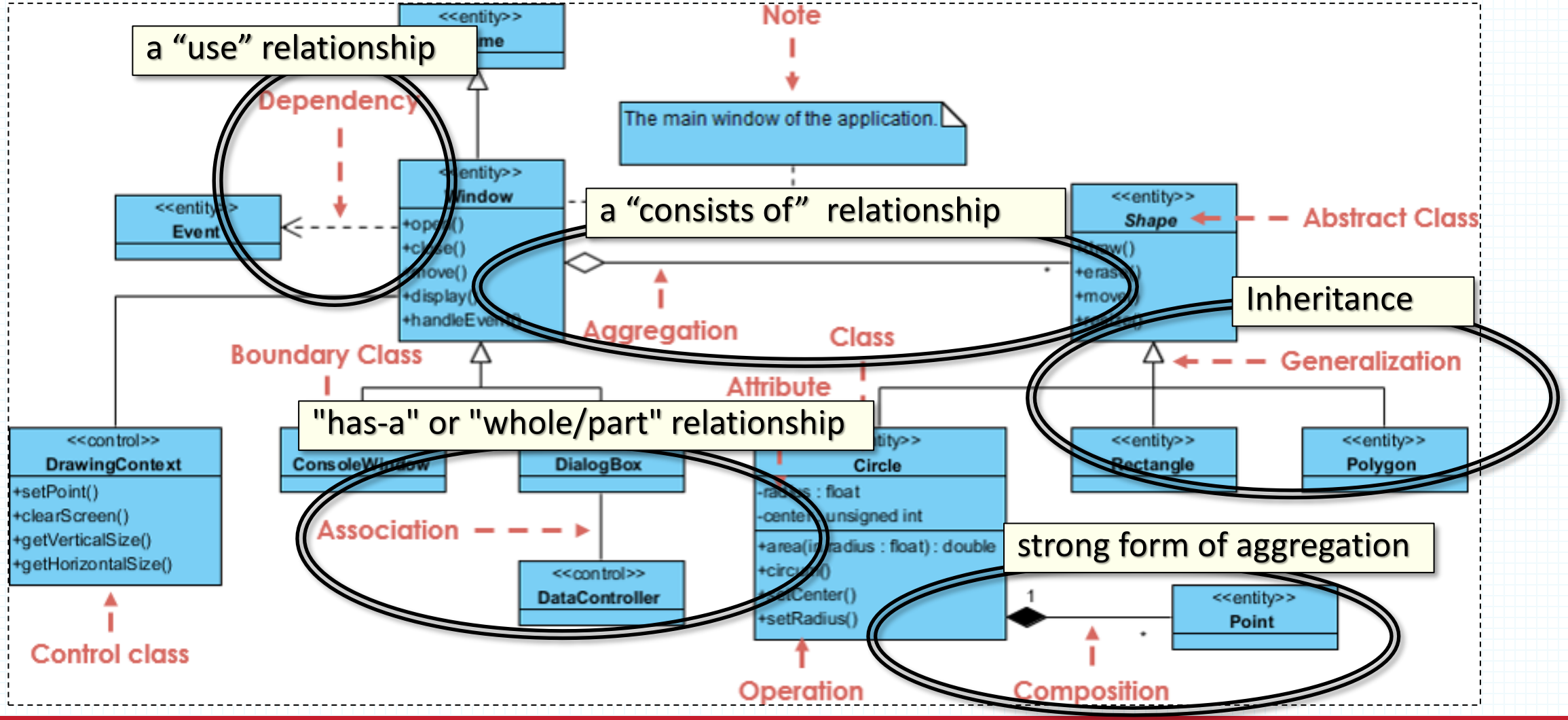
Class Diagrams

A ***class*** represent a concept
Class = state ***(attributes)*** + behavior ***(operations)***

Each attribute has a ***type***.
Each operation has a ***signature***.



Class Diagrams



Actor vs Class vs Instances?

An entity **outside** the system

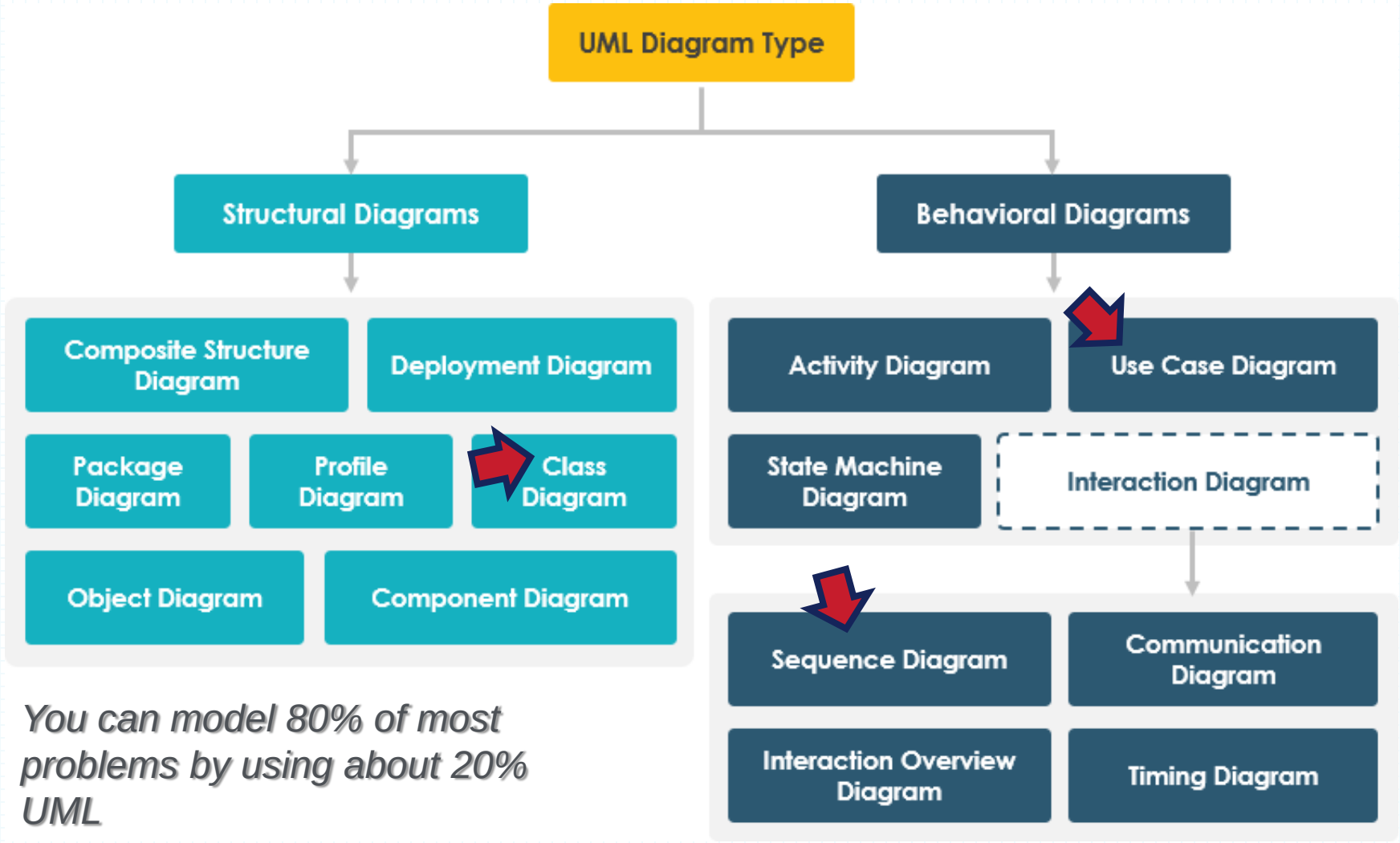
An abstraction modeling an entity in the problem domain **inside the system**

A specific instance of a class

Association, Aggregation, and Composition

- Association(using relationship): The objects created and destroyed independently and represented as one-to-one, one-to-many, or many-to-many (also known as cardinality).
- Aggregation(HAS-A relationship): parent can exist without child and a child ***can*** exist independently of the parent
- Composition("death" relationship, PART-OF): parent destroyed, child dies

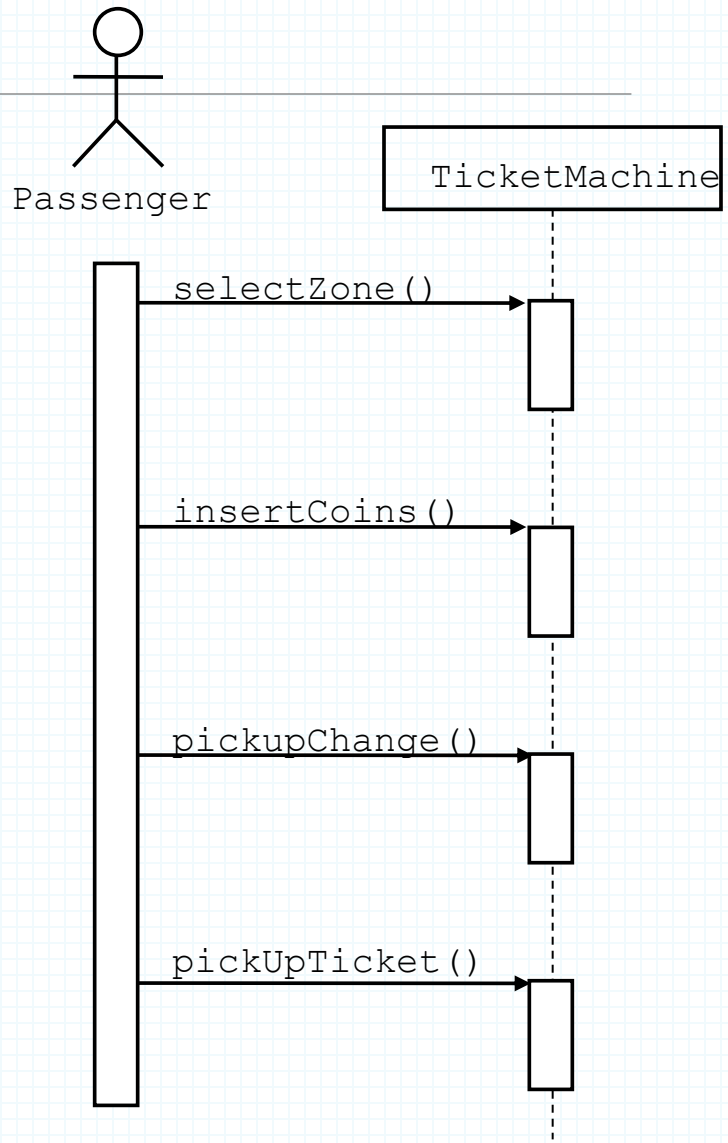
UML diagrams



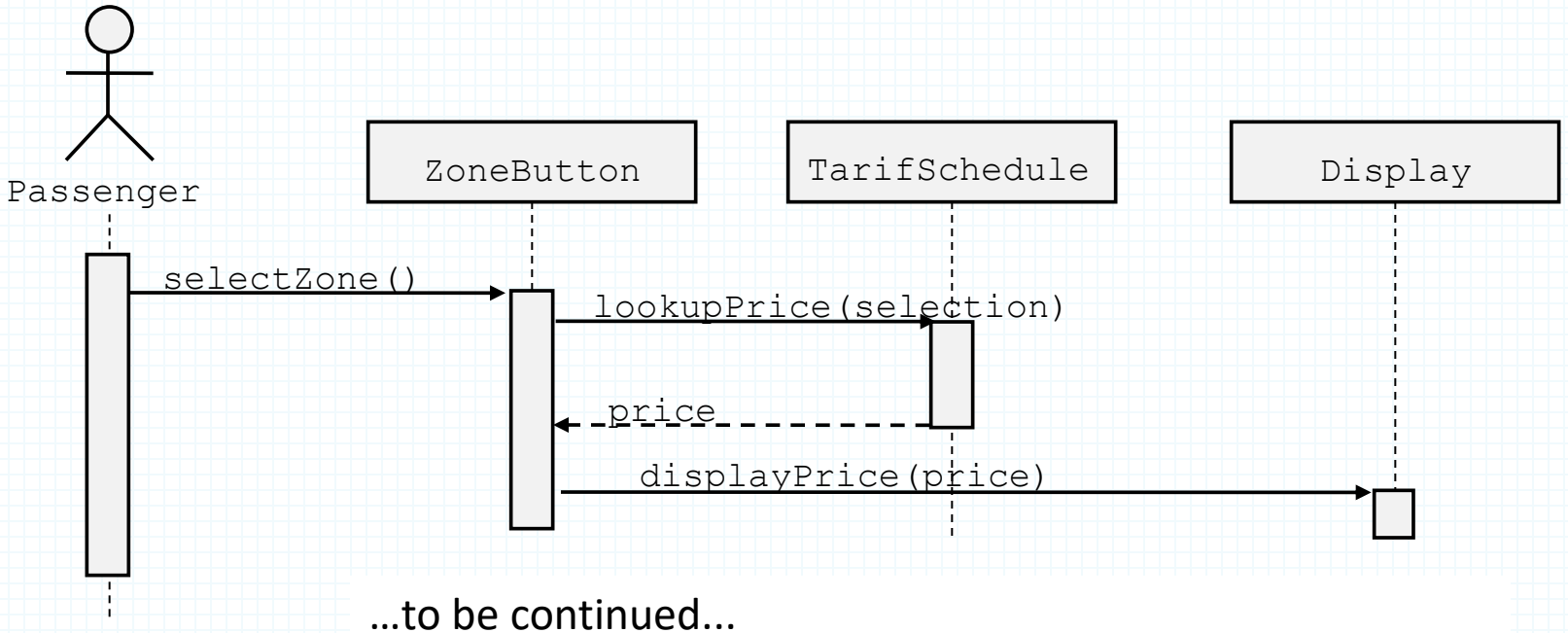
You can model 80% of most problems by using about 20% UML

UML sequence diagrams

- **Why UML Seq Diagram?**
  - To understand **the flow of control and data** between various elements in a system.
  - **To identify the objects** involved in a particular scenario and the messages exchanged between them
  - To find **additional objects** (“participating objects”)
- System Sequence Diagram (SSD) vs Sequence Diagram
  - A Sequence Diagram emphasizes the interactions between **objects within a system**, while a System Sequence Diagram (SSD) focuses on the interactions between **a system and its environment**

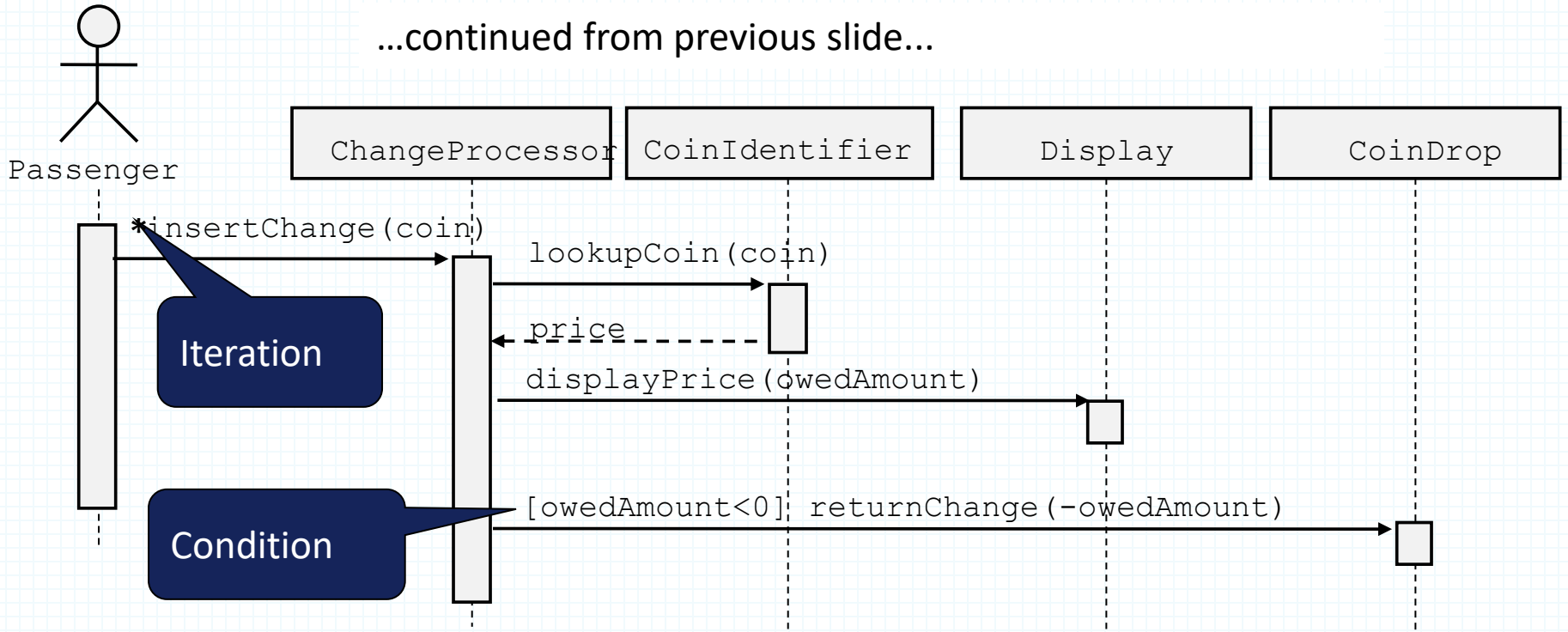


UML sequence diagrams

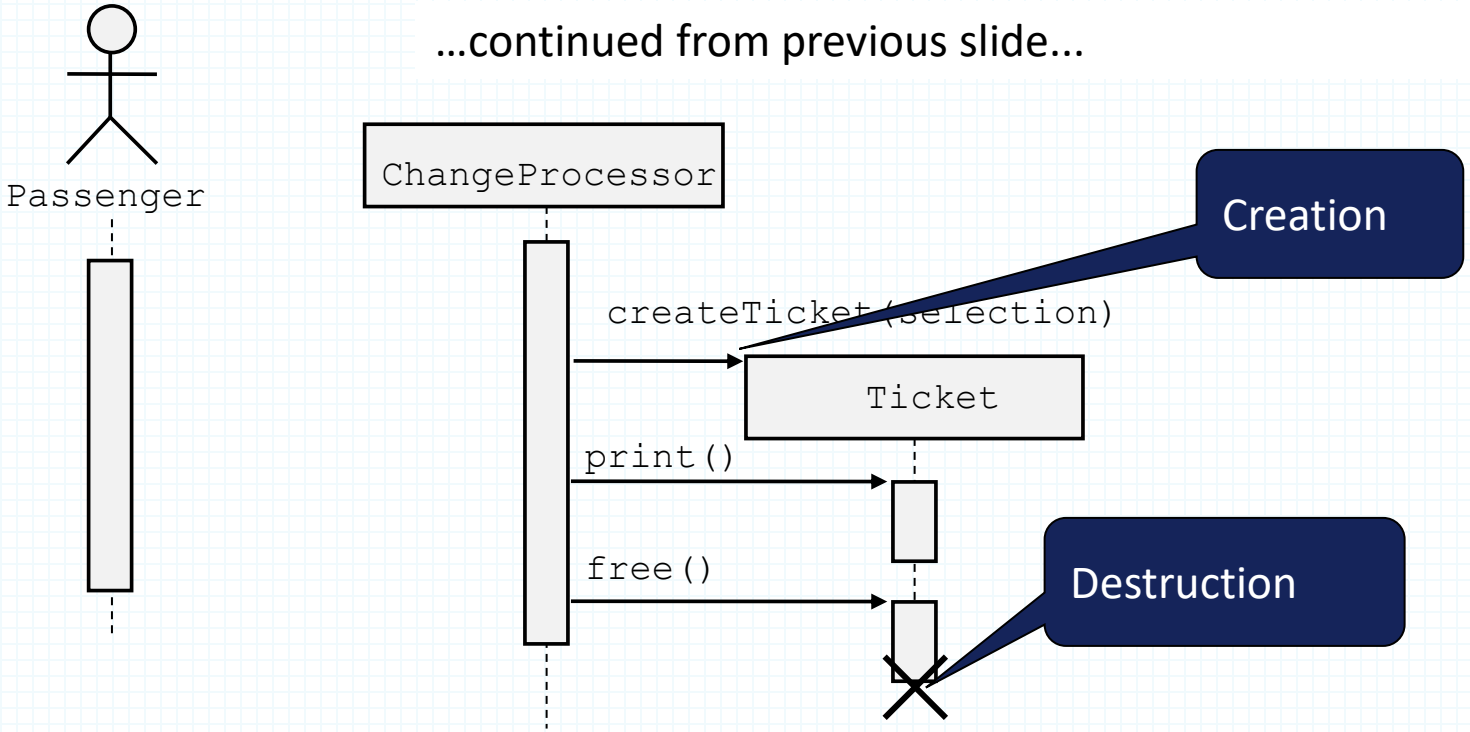


…to be continued...

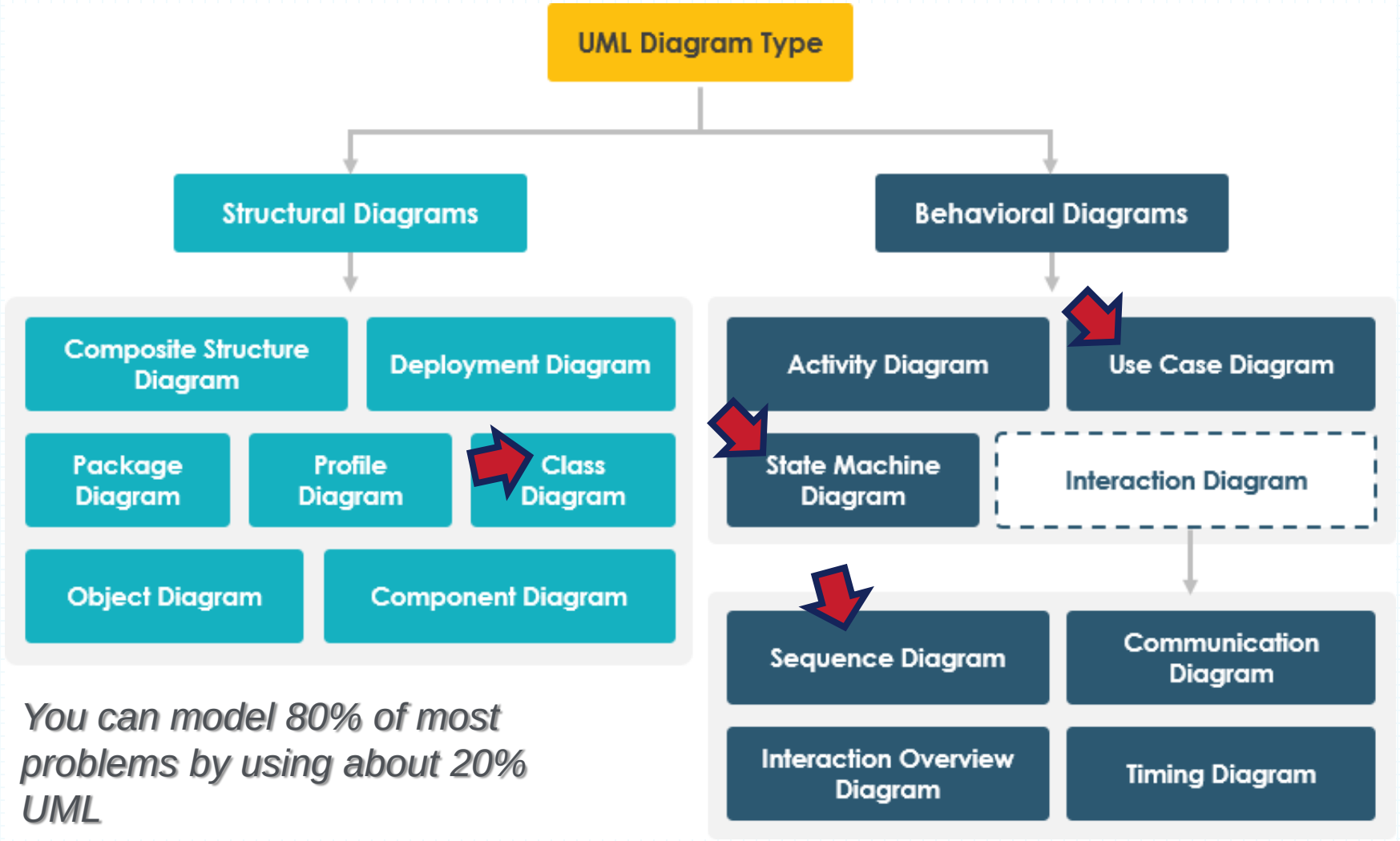
UML sequence diagrams



UML sequence diagrams



UML diagrams



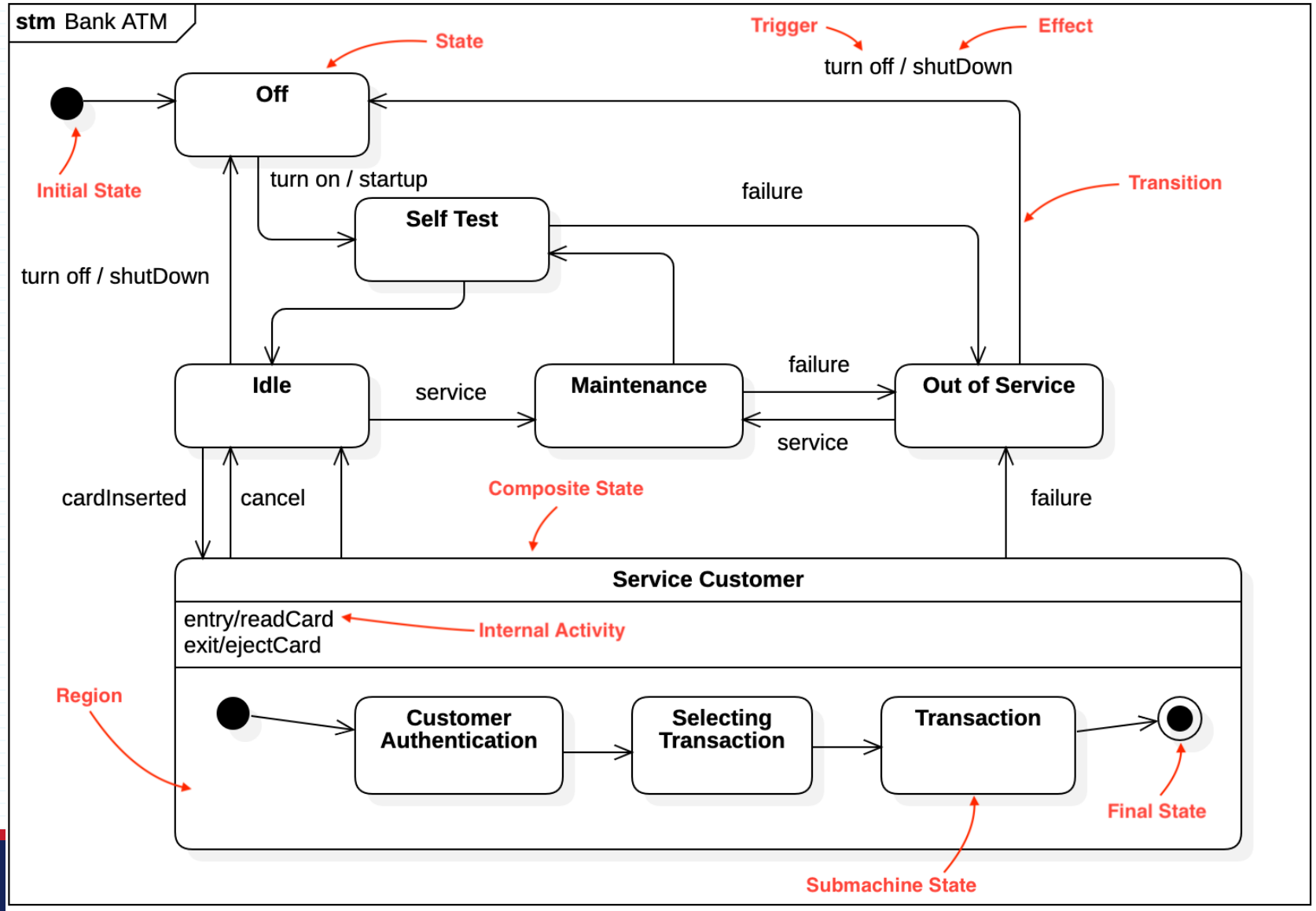
You can model 80% of most problems by using about 20% UML

State Chart Diagrams

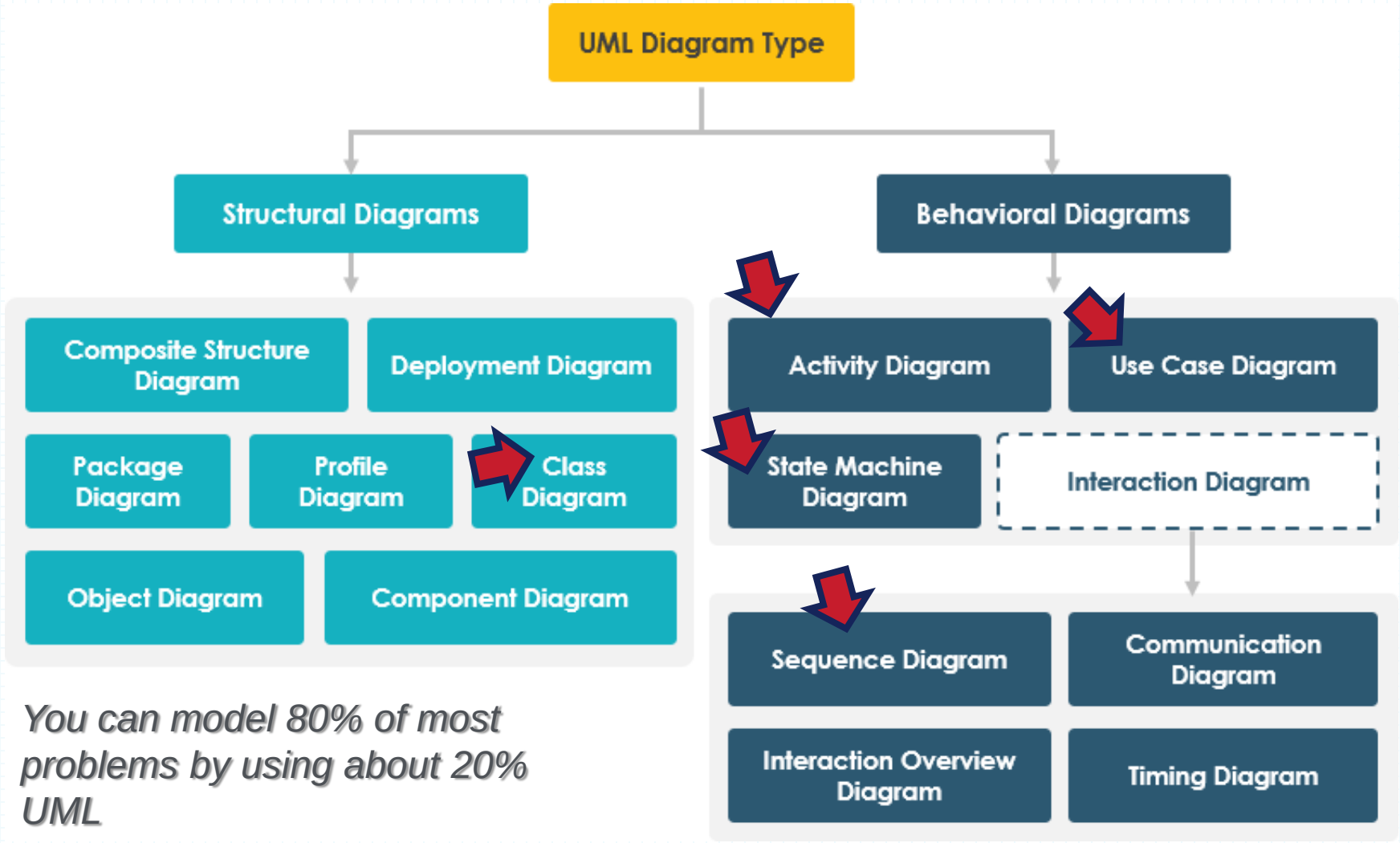
Why UML State Diagram?

- Modeling the behavior of an interface, class, or collaboration.
- State diagrams emphasize the event-ordered behavior of an object, which is especially useful in modeling reactive systems.

Note that State is an Attribute or Collection of Attributes of object of type Incident



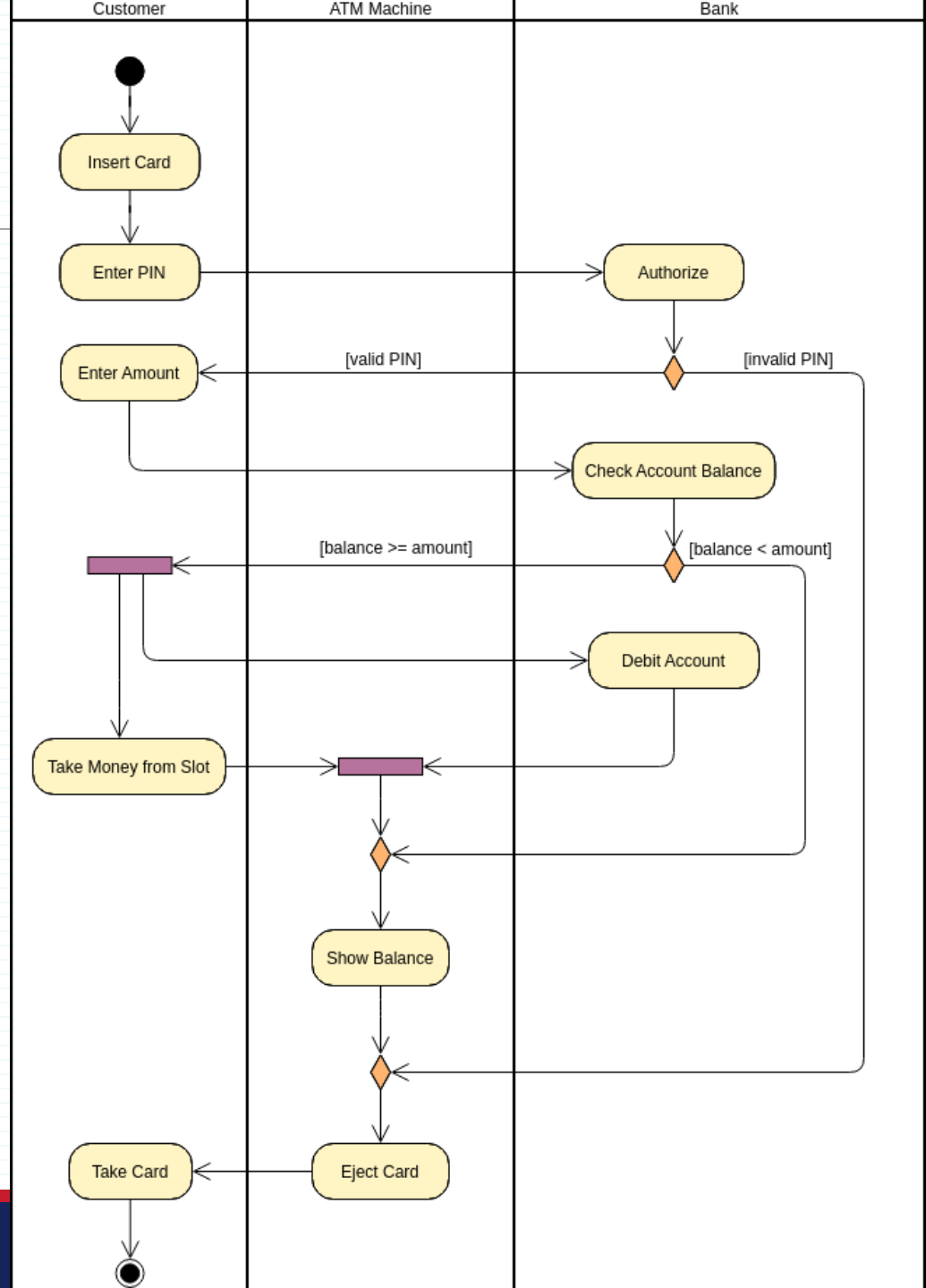
UML diagrams



You can model 80% of most problems by using about 20% UML

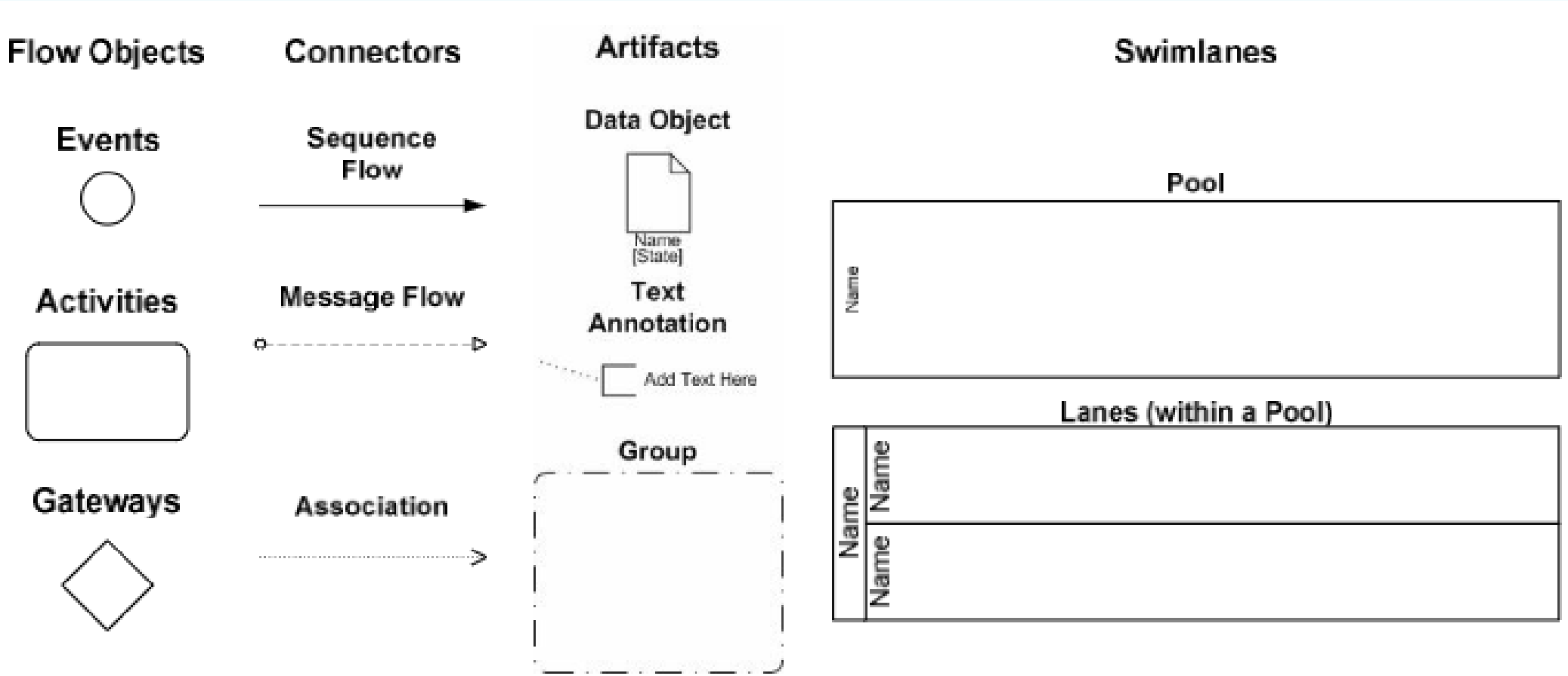
Activity Diagram

- **Why Activity Diagram?**
  - Illustrate a business process or workflow between users and the system.
  - Simplify and improve any process by clarifying complicated use cases.
- Note that State is an Operation!
- **State chart diagrams** focus on the internal state of an object or system and how it responds to external events, while **activity diagrams** focus on the sequence of activities in a process or workflow

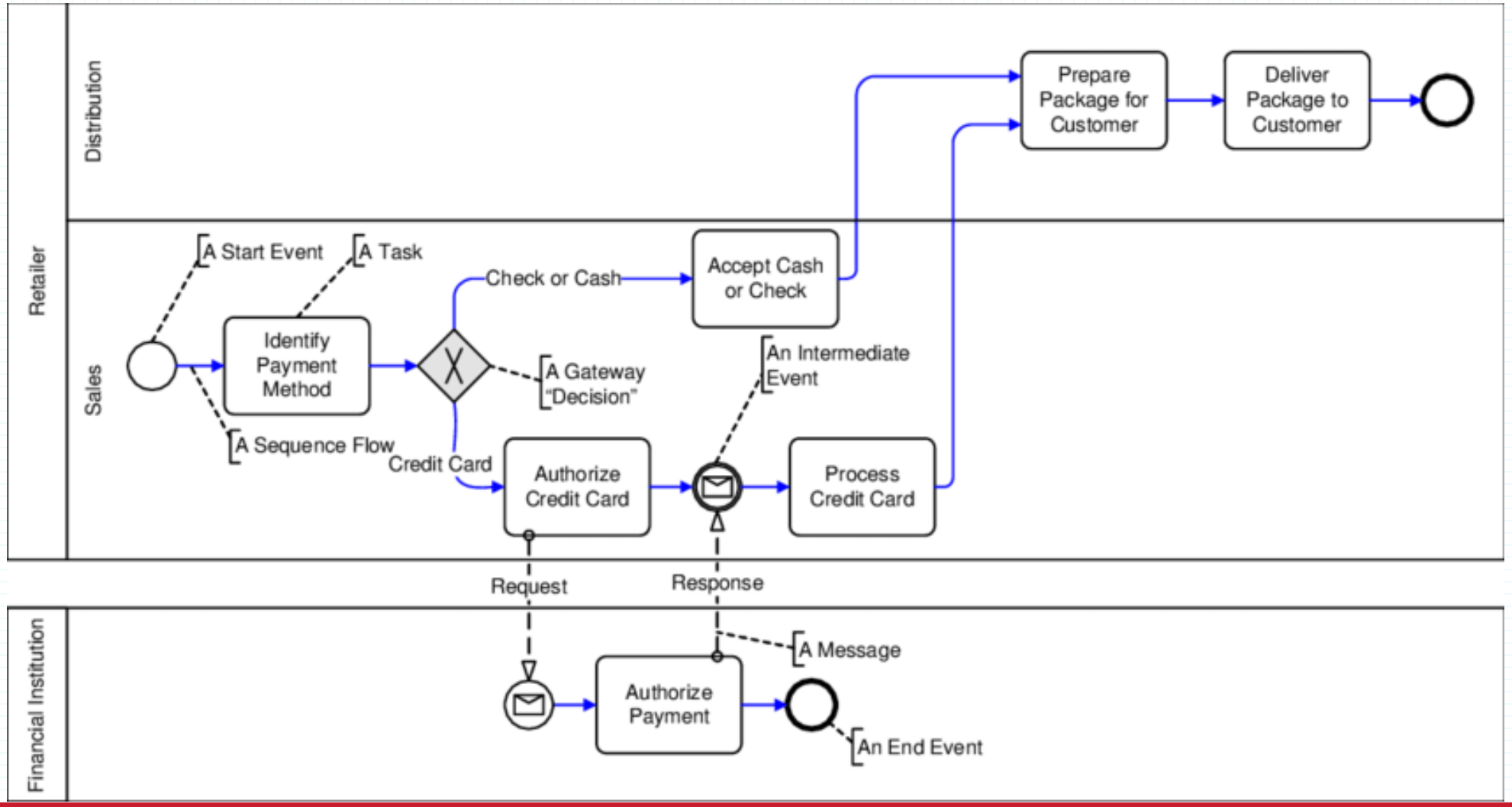


Other Modeling techniques!
UML is not everything in practical projects

Diagram Elements



Example


Retailer
Distribution
Sales
Financial Institution
Prepare Package for Customer
Deliver Package to Customer
A Start Event
A Task
Identify Payment Method
Check or Cash
Accept Cash or Check
A Gateway "Decision"
An Intermediate Event
A Sequence Flow
Credit Card
Authorize Credit Card
Process Credit Card
Request
Response
A Message
Authorize Payment
An End Event