الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Topic 0

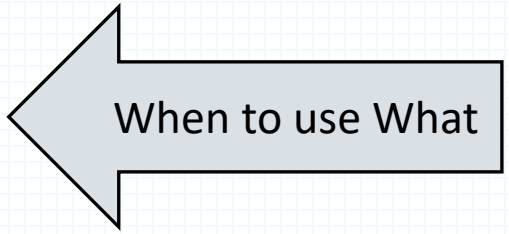
Software Architecture

Introduction - An Overview of Software Engineering Concepts and UML Modeling

Dr. Riad Sonbol

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?

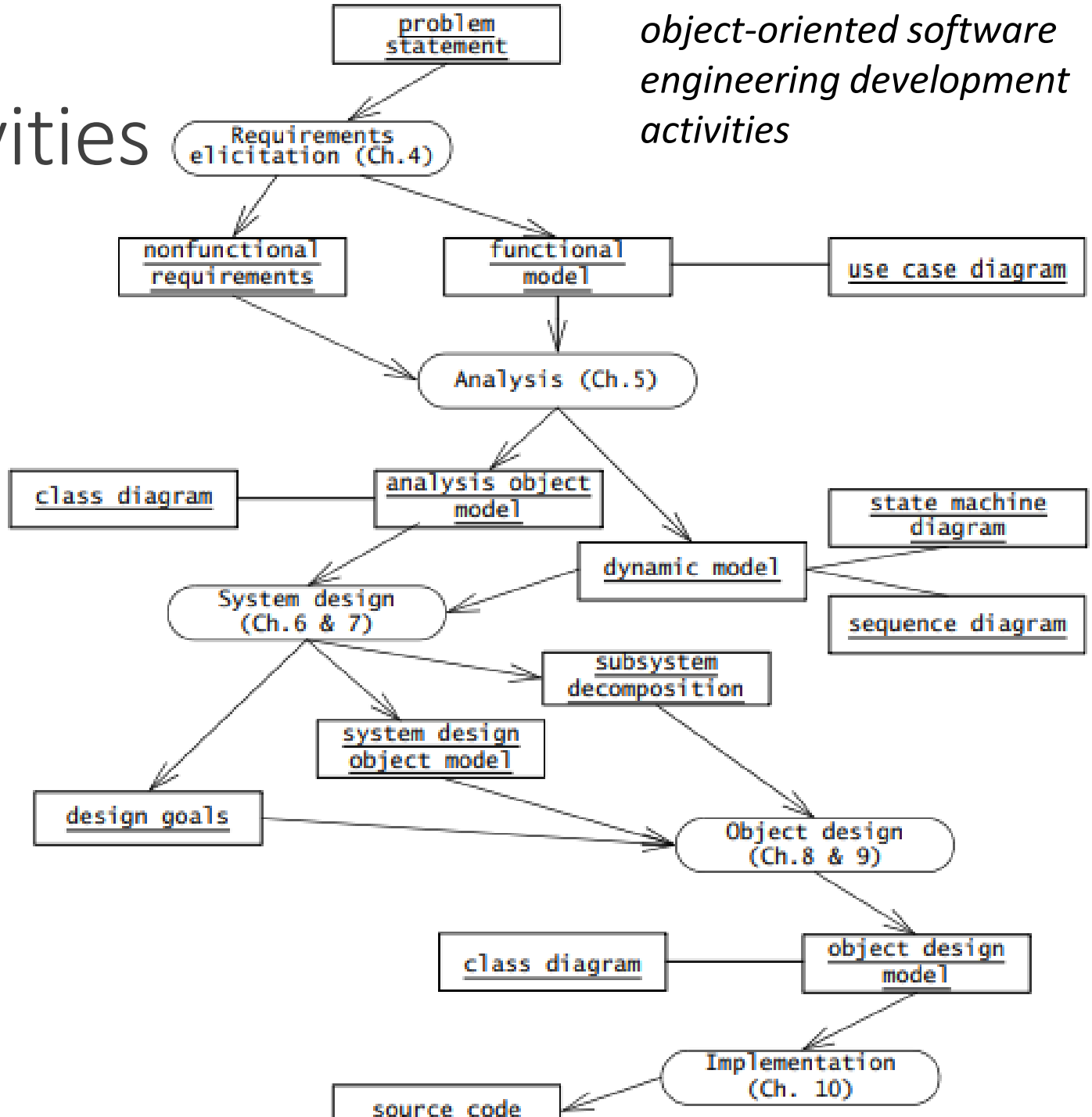
- **Software engineering** is the branch of computer science that deals with the **design, development, testing, and maintenance of software applications**.
- Software engineering is the application of engineering principles to the development of software systems.
- What are the main SE Development Activities?

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

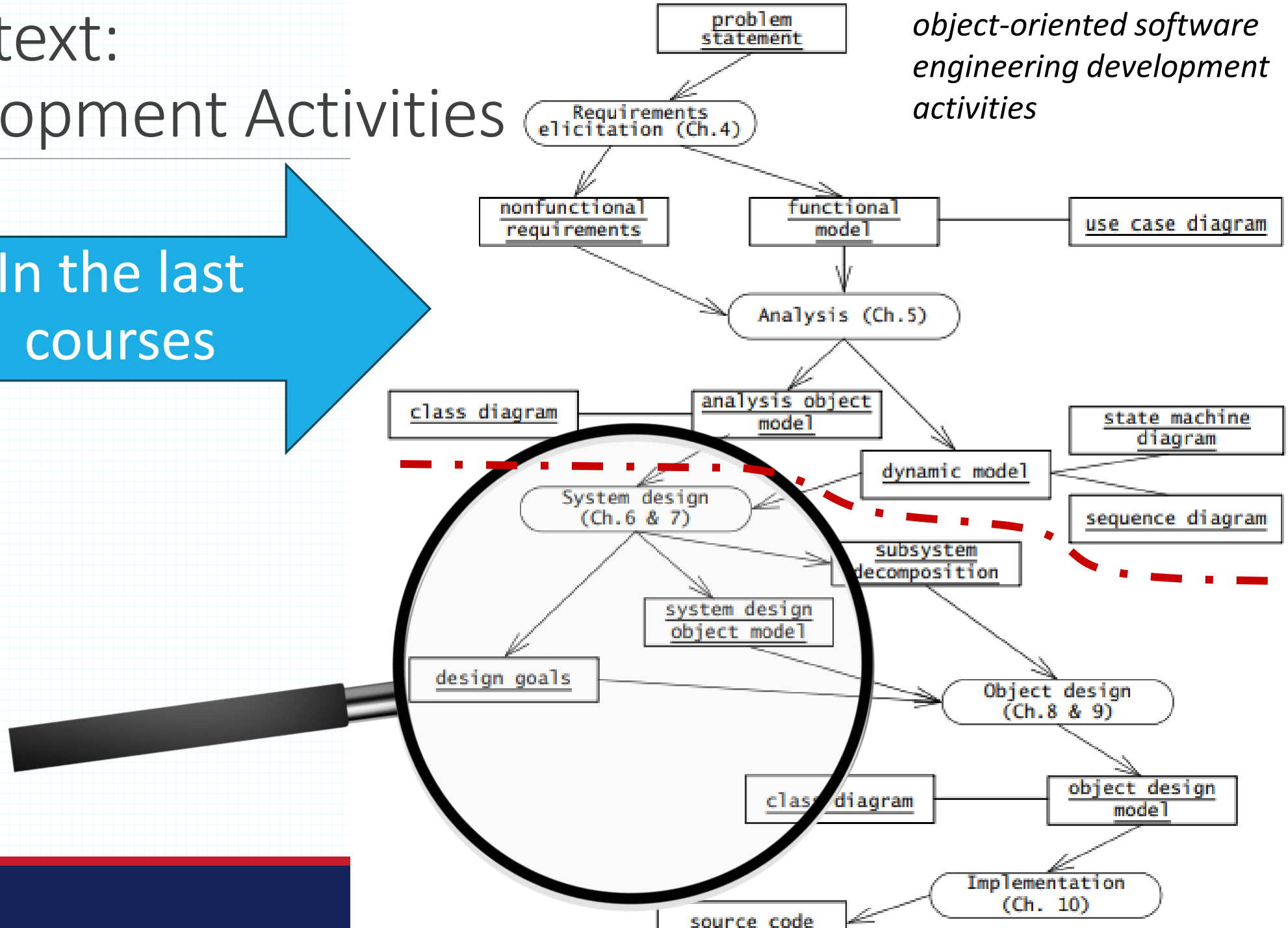
object-oriented software engineering development activities



The Context: SE Development Activities

object-oriented software engineering development activities

In the last courses



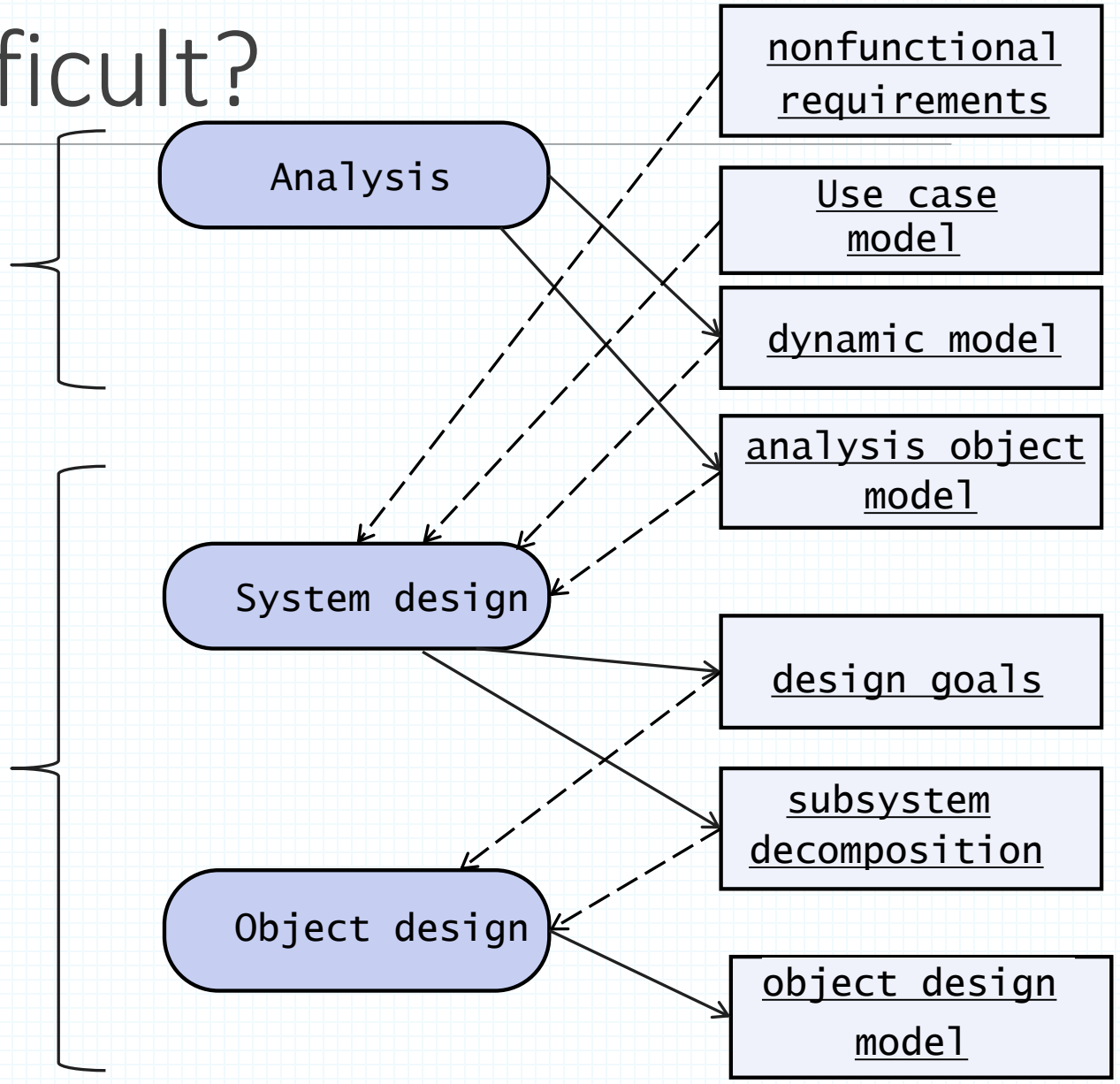
Why is Design so Difficult?

Focuses on the application domain

Focuses on the solution domain

The solution domain is changing very rapidly

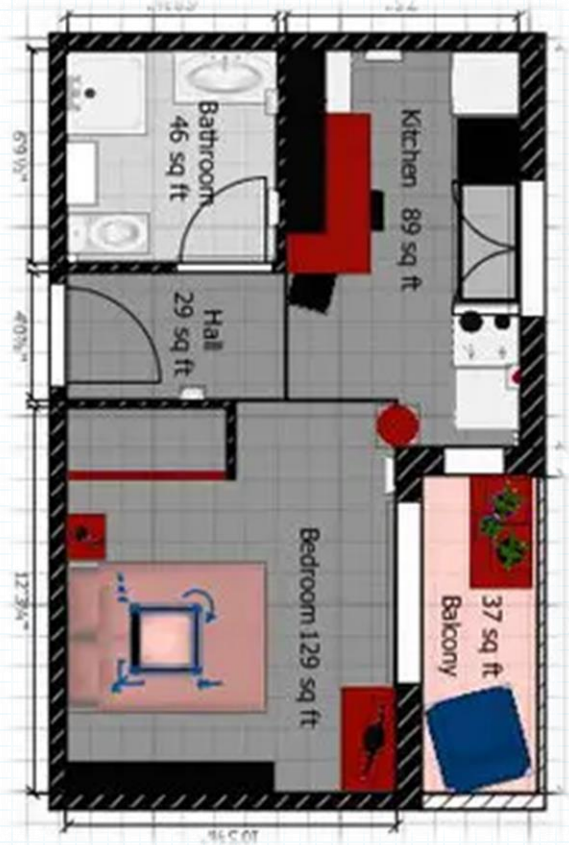
- *Halftime knowledge in SE ≈ 3-5 years*
- *Cost of hardware rapidly sinking*
- *Design knowledge is a moving target*



Modeling with UML- Quick Overview

What is modeling?

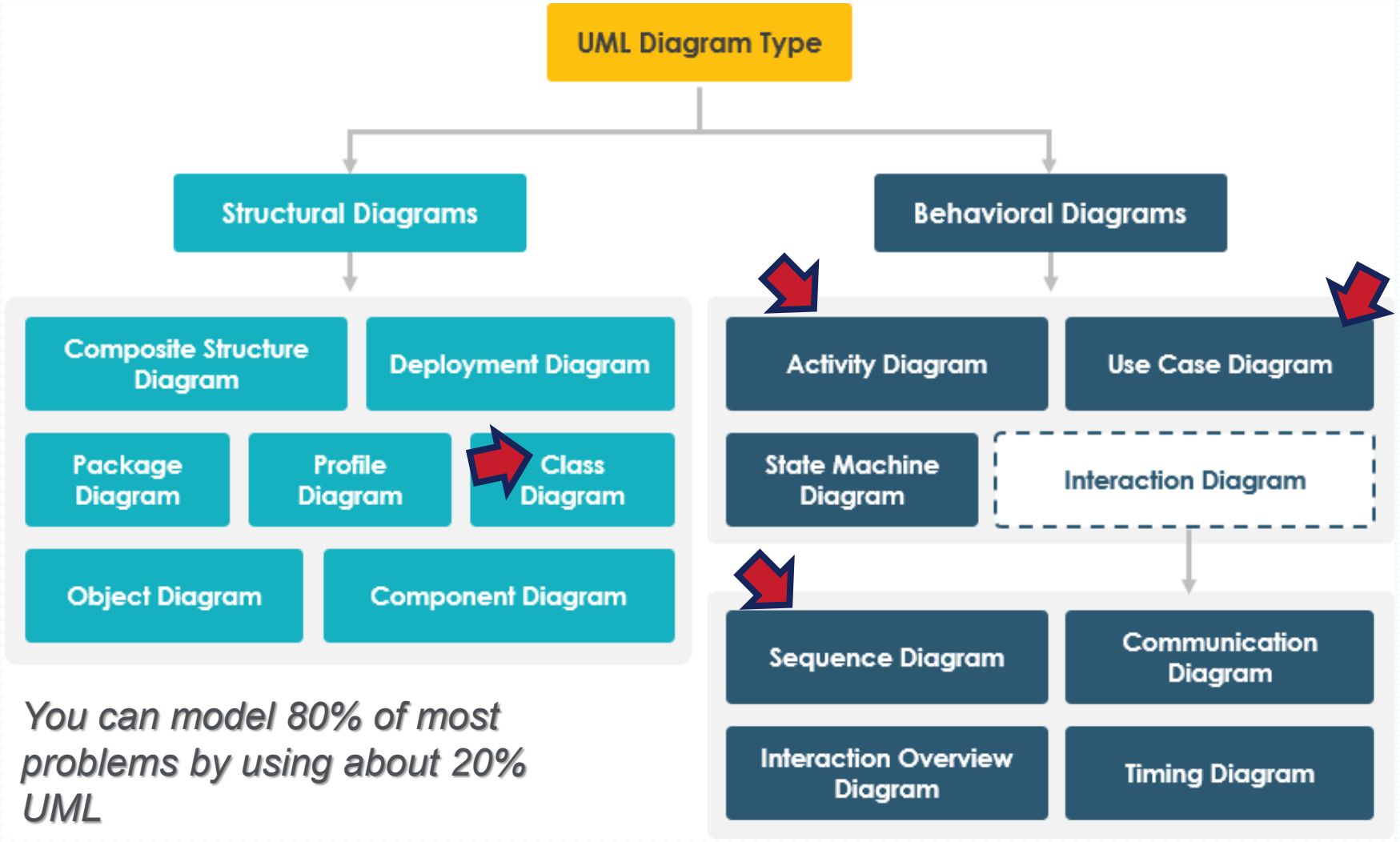
- the process of building an **abstraction** of reality.
- It involves creating a **simplified** representation that captures essential features of a system.
- The goal is to **make the complex reality more** understandable, manageable, and analyzable.



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

for different types of stakeholders

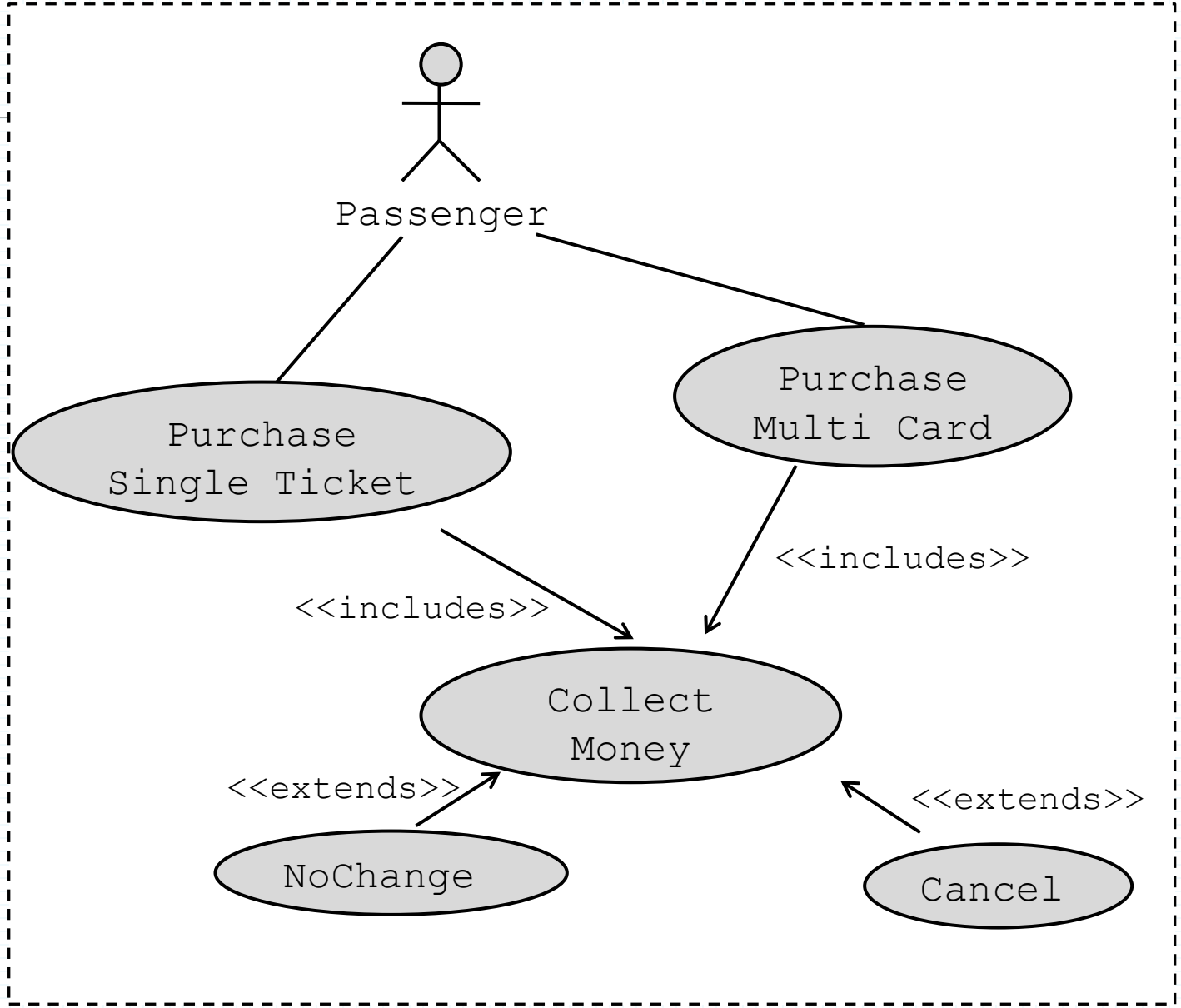
UML diagrams



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

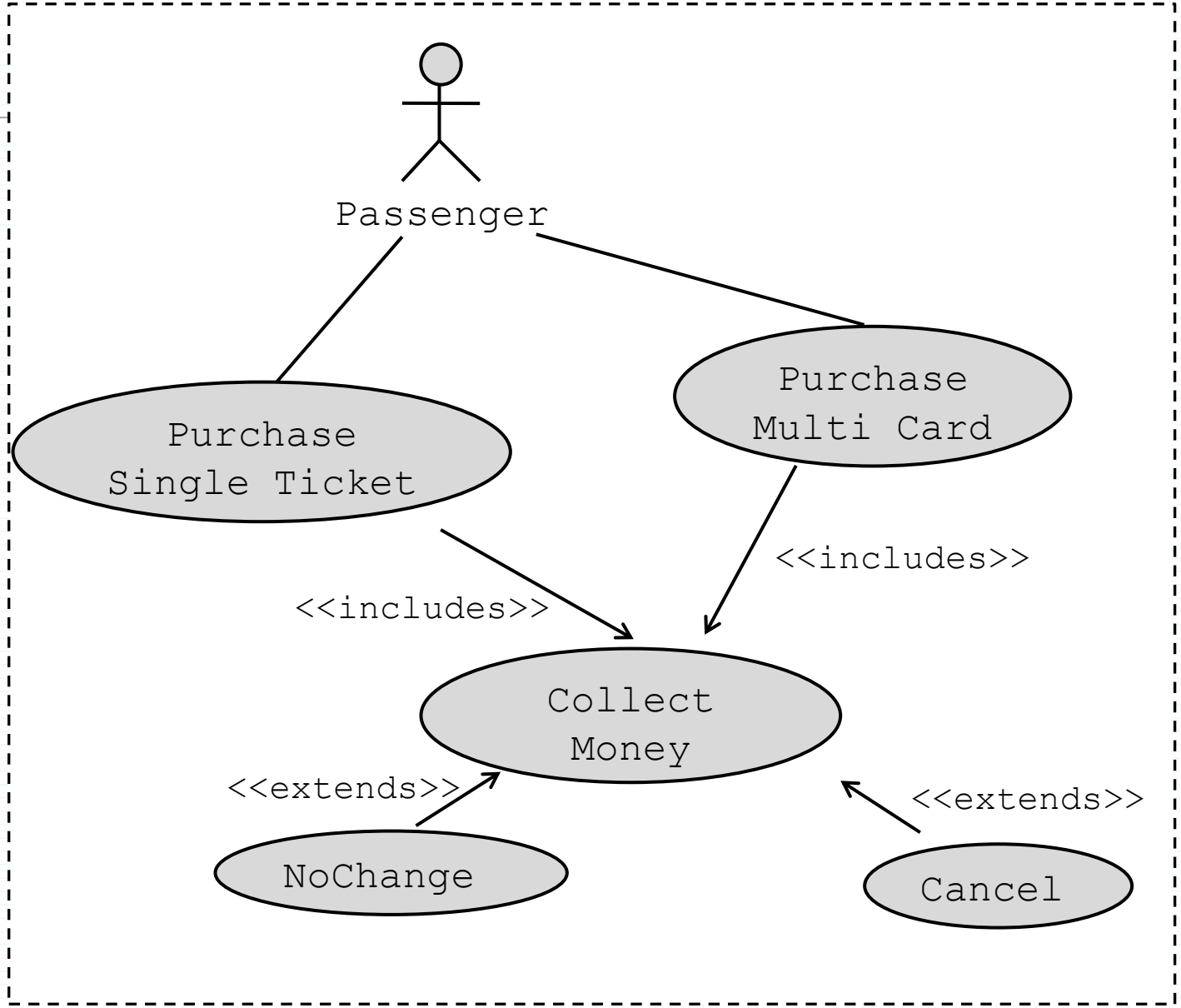
Use Case Diagram

- 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*



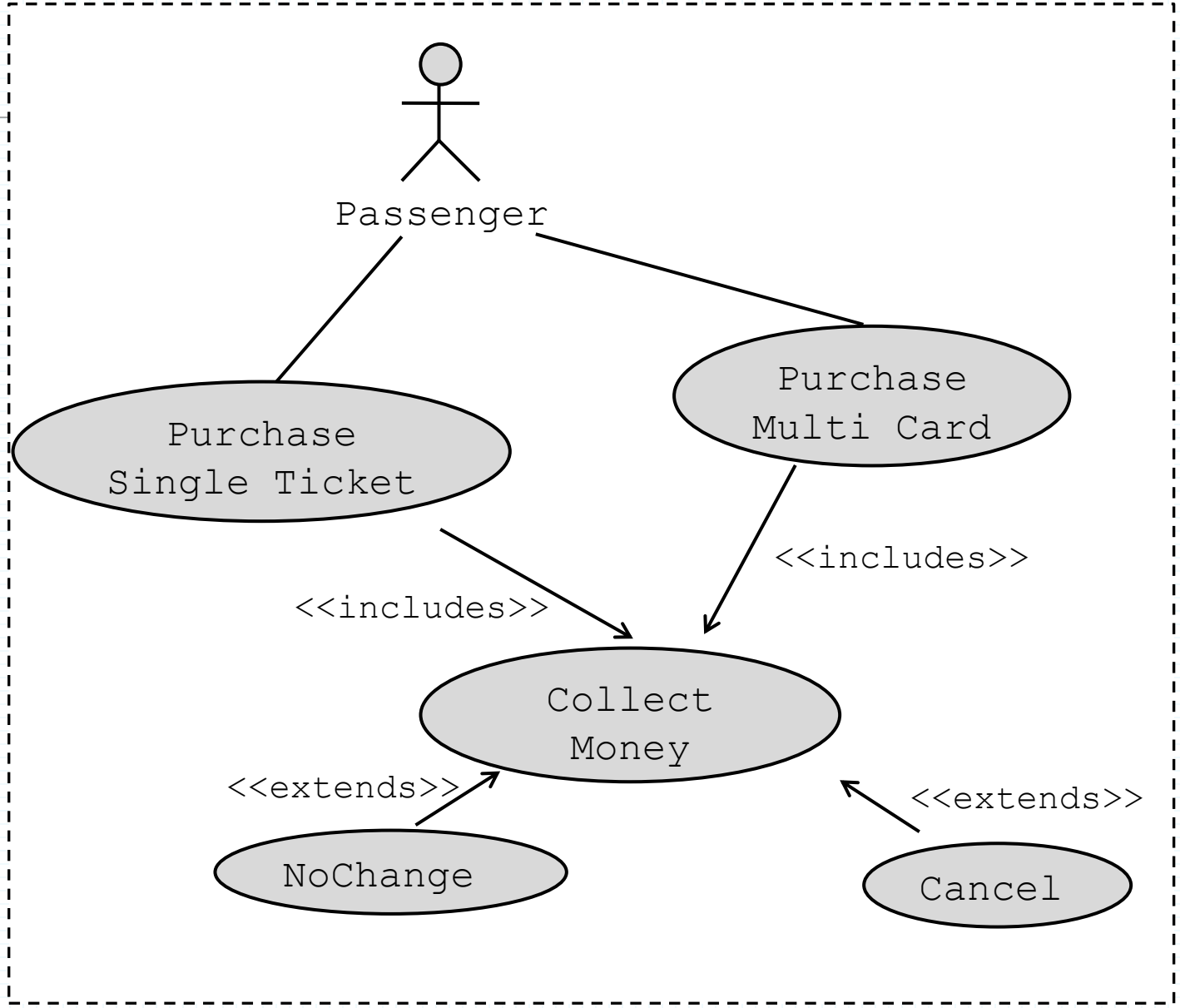
Use Case Diagram

- **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:

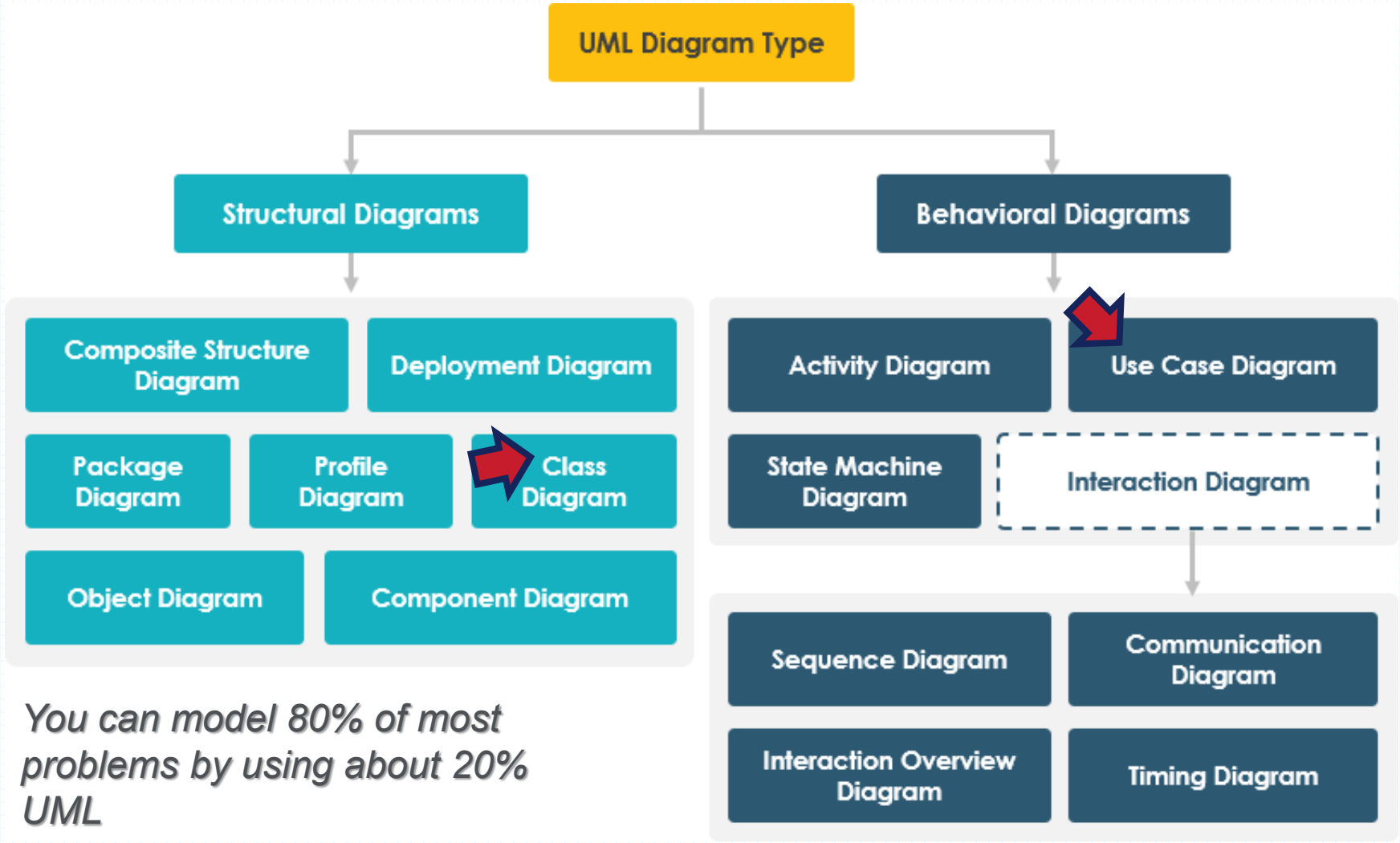
- `Passenger` has ticket.

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.

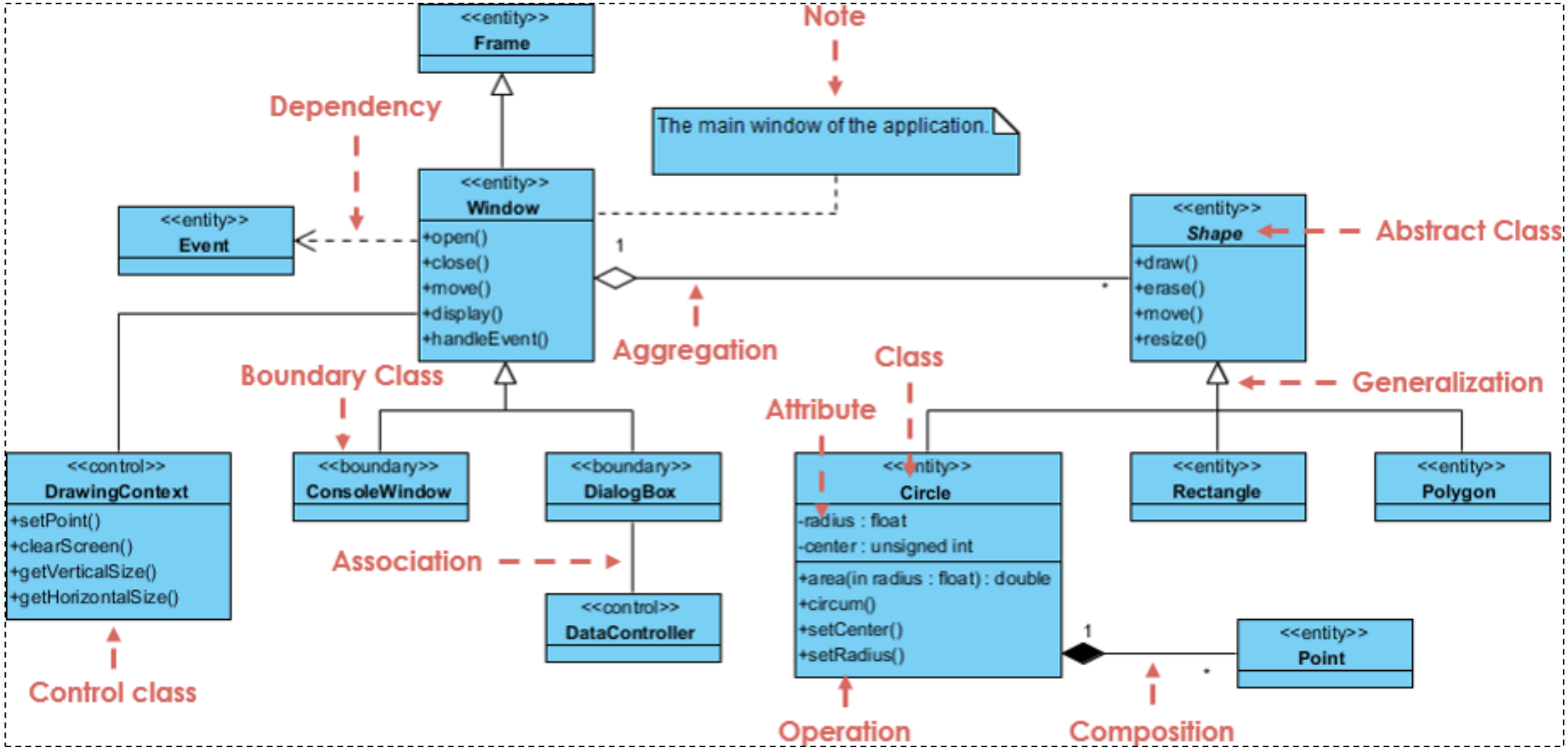
Exceptional cases!

UML diagrams



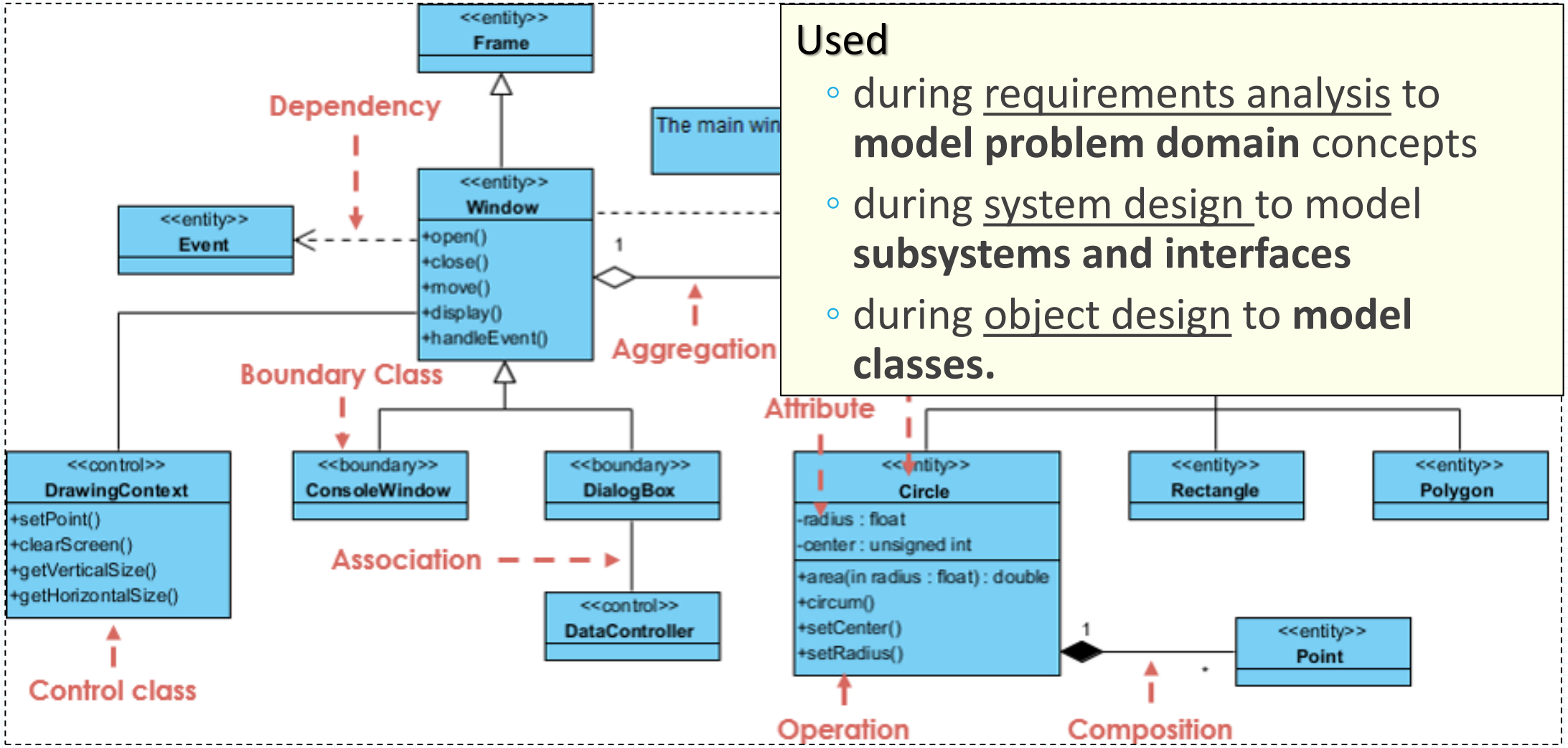
You can model 80% of most problems by using about 20% UML

Class Diagrams



Class Diagrams

Class diagrams represent the structure of the system.



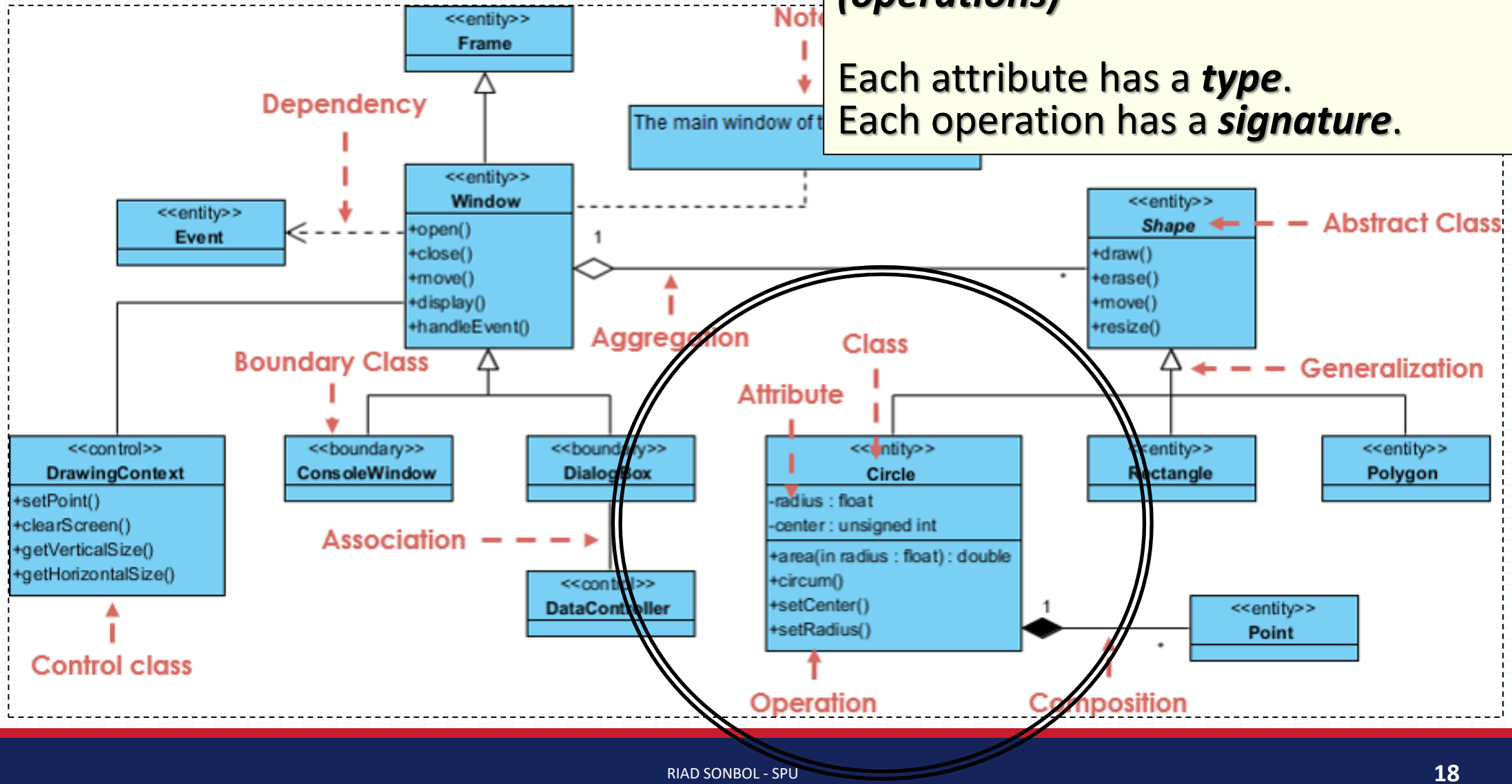
Used

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

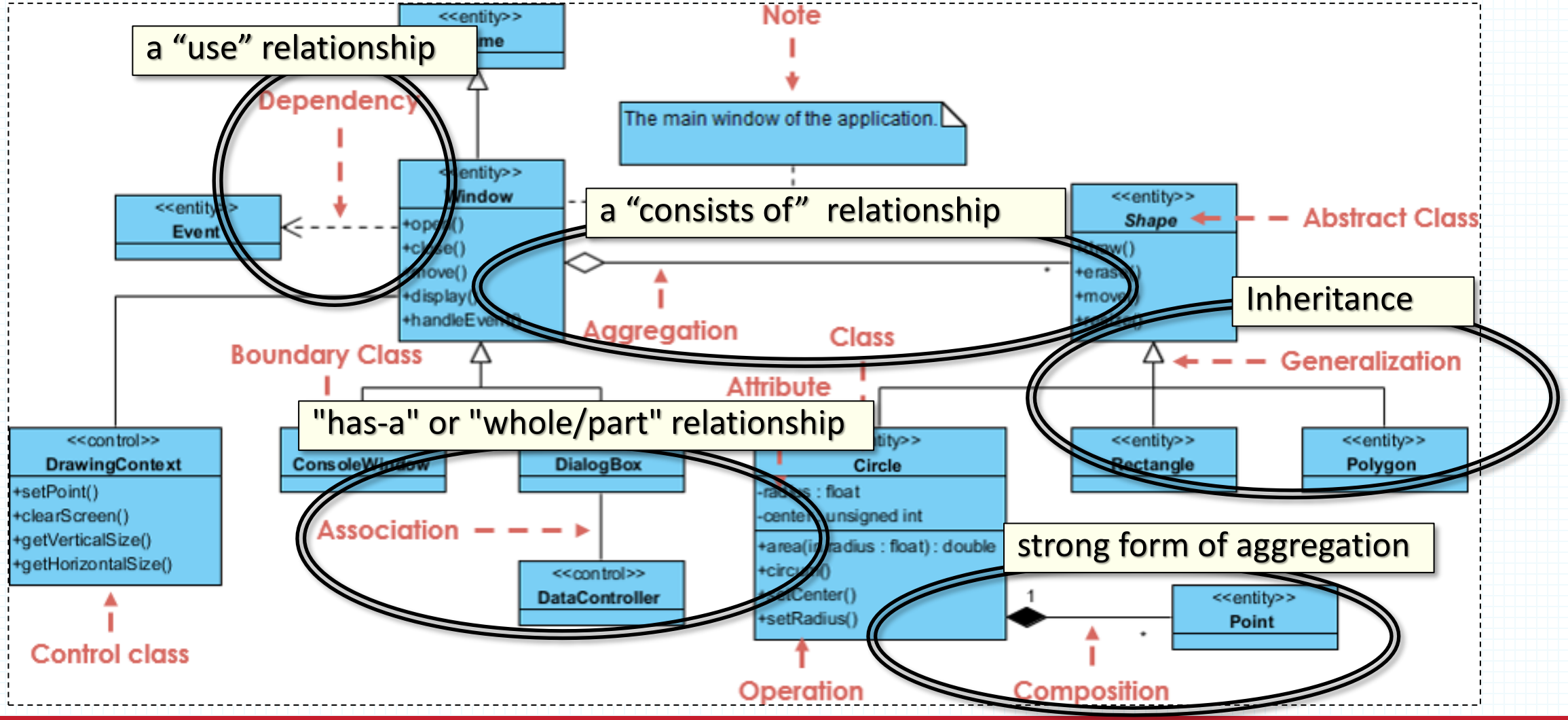
Class Diagrams

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

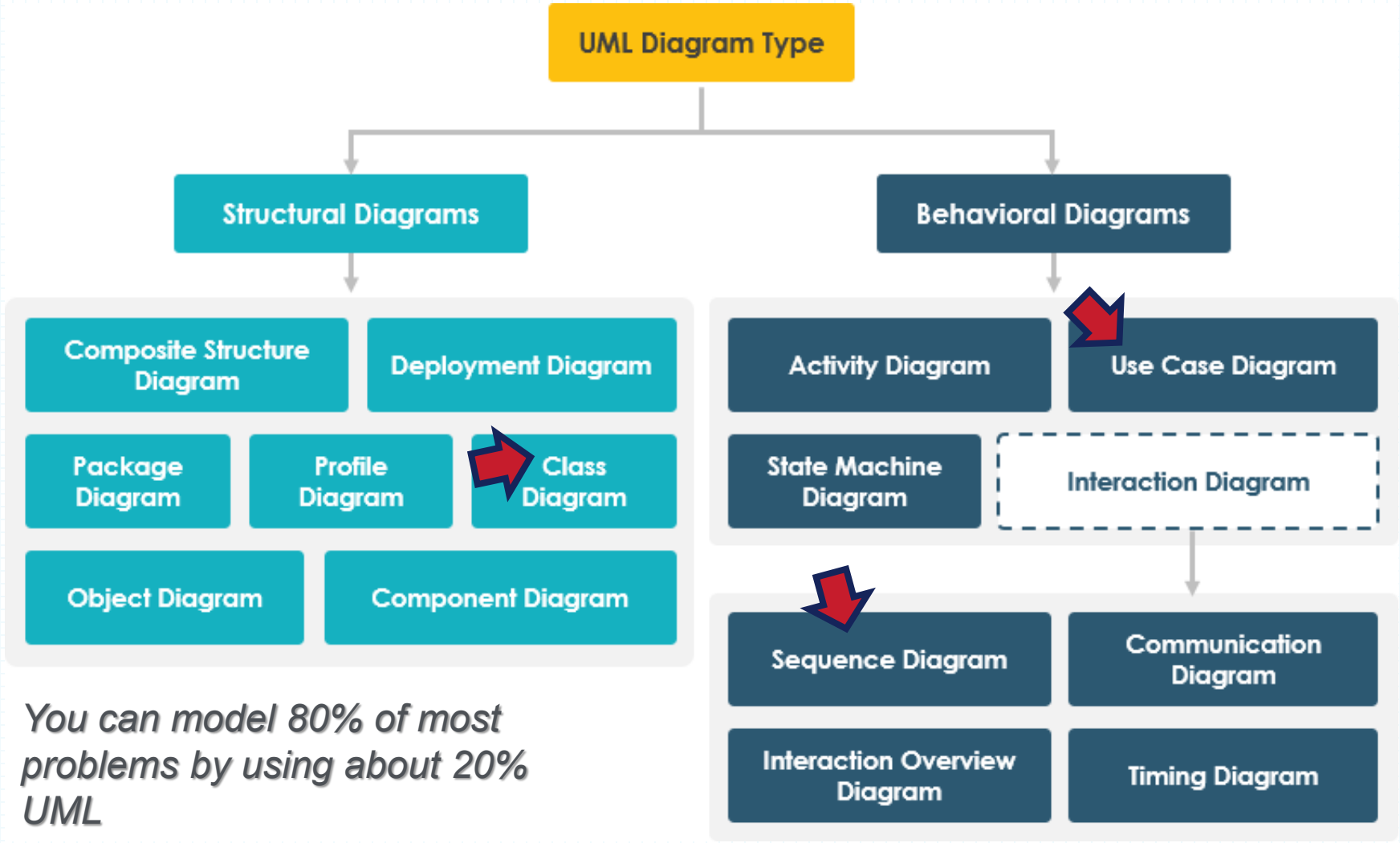
An abstraction modeling an entity in the problem domain **inside the system**

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

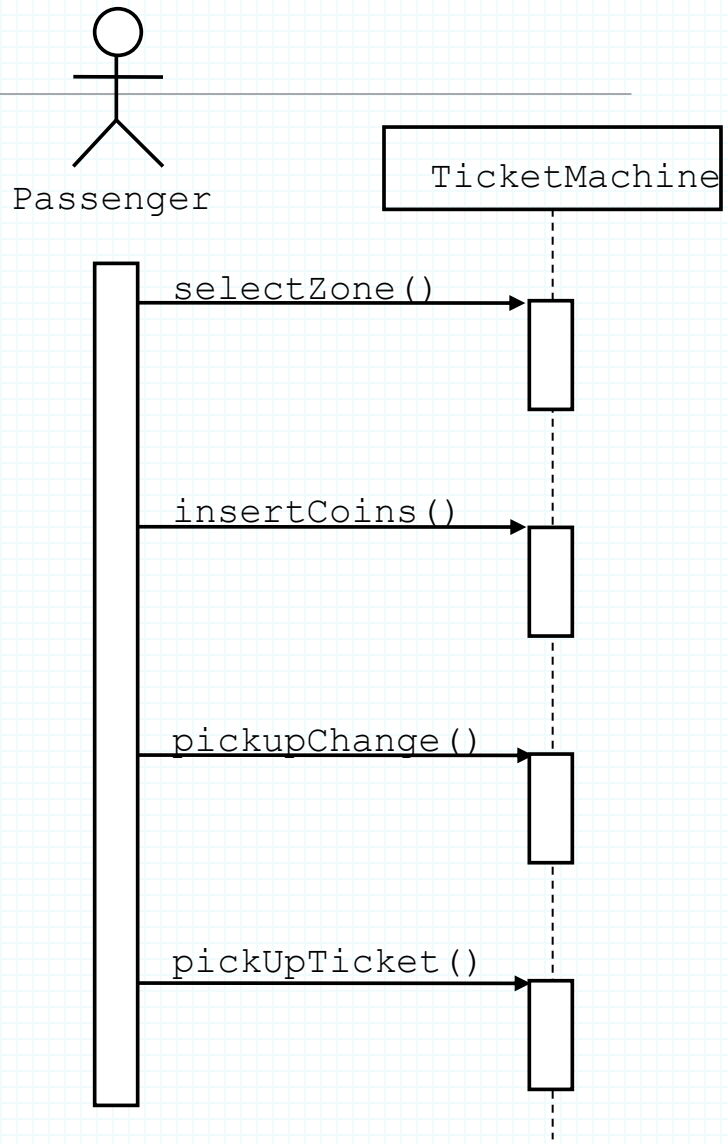
UML diagrams



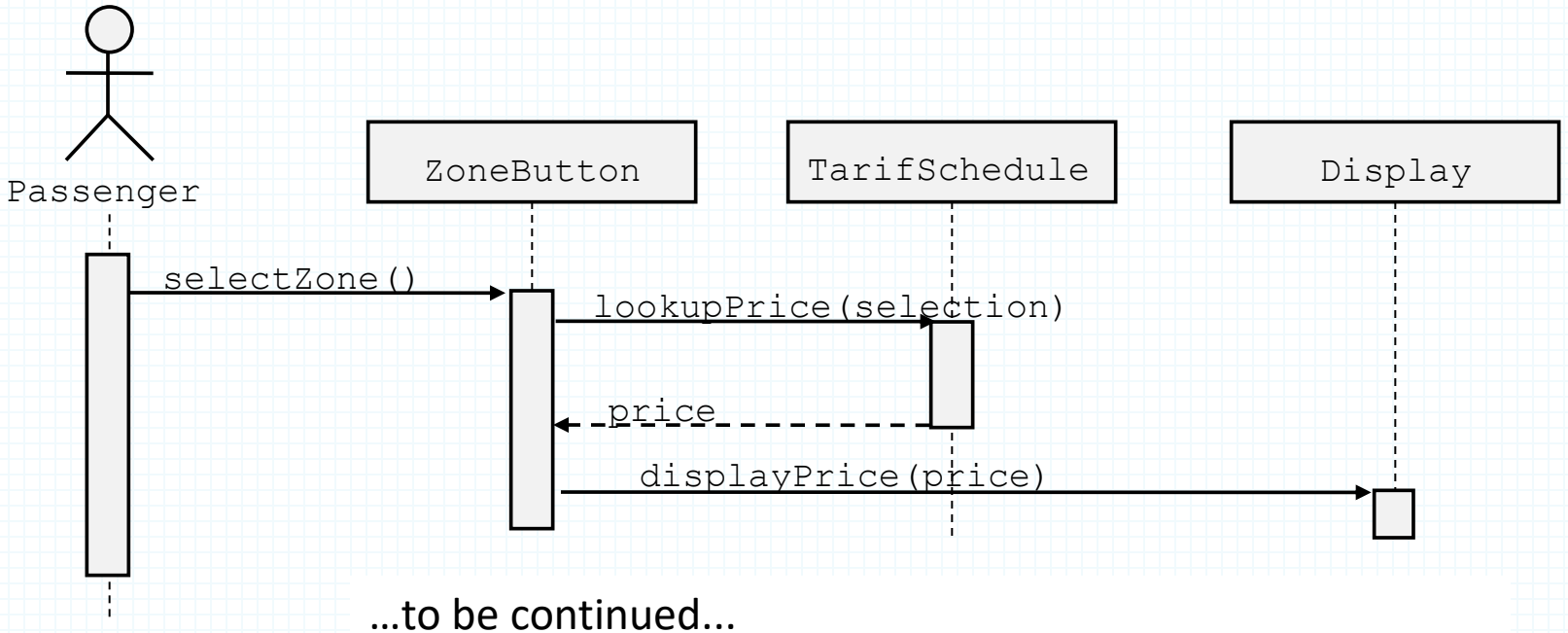
You can model 80% of most problems by using about 20% UML

UML sequence diagrams

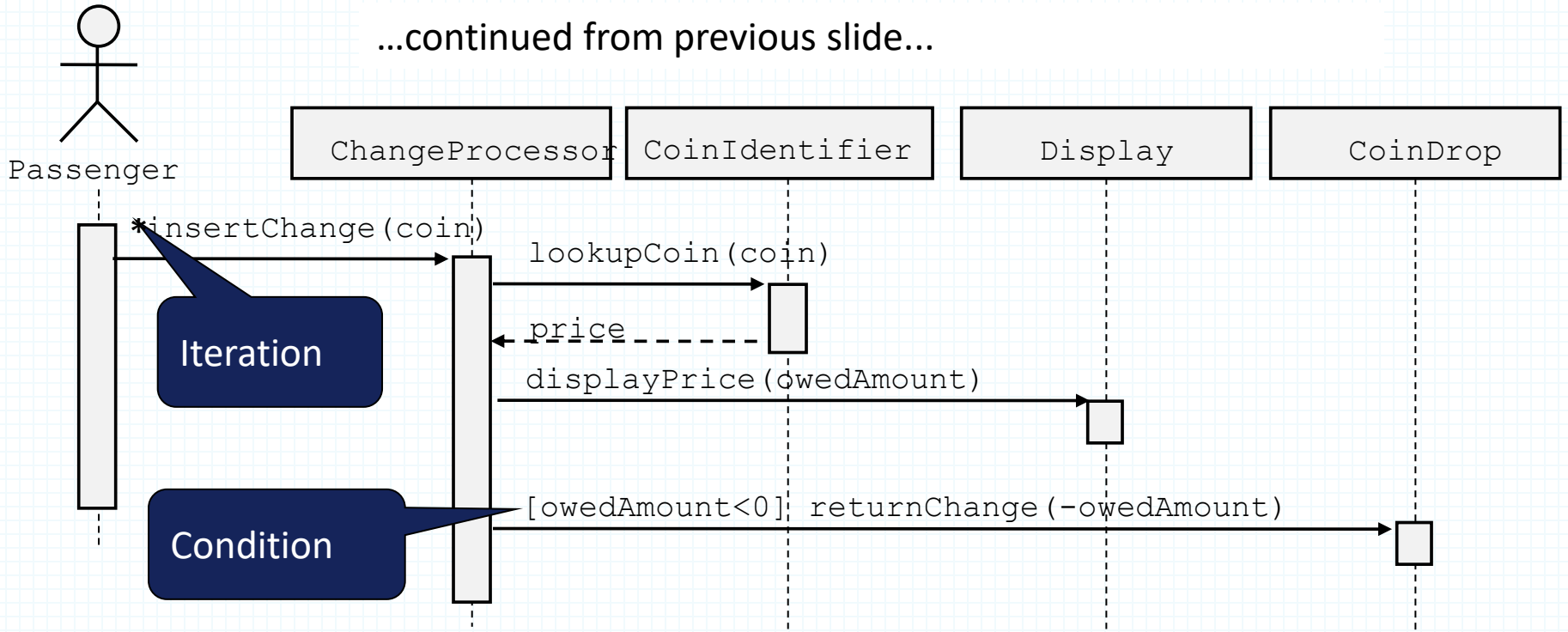
- **Why UML Seq Diagram?**
  - To understand **the flow of control and data** between various elements in a system.
  - **To identify the objects** involved in a particular scenario and the messages exchanged between them
  - To find **additional objects** (“participating objects”)
- System Sequence 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**



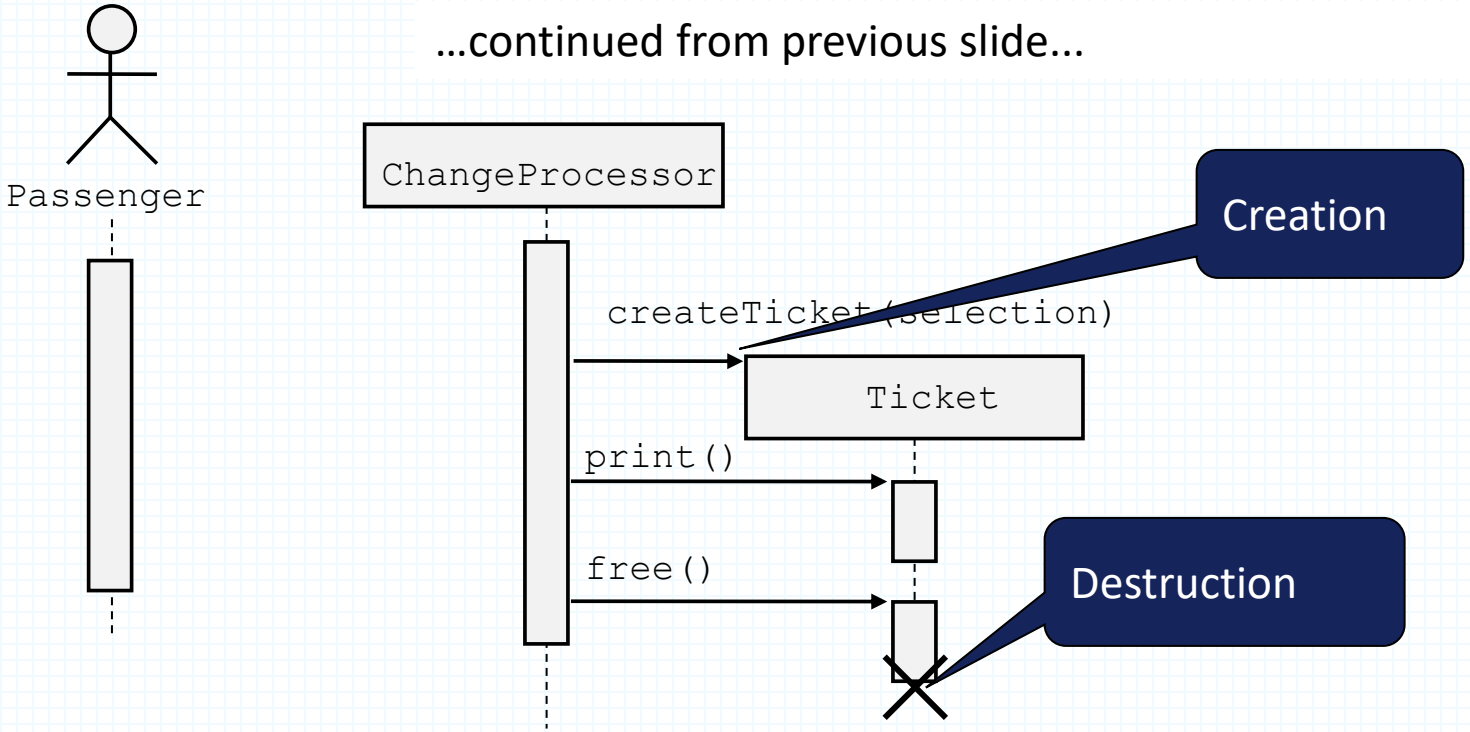
UML sequence diagrams



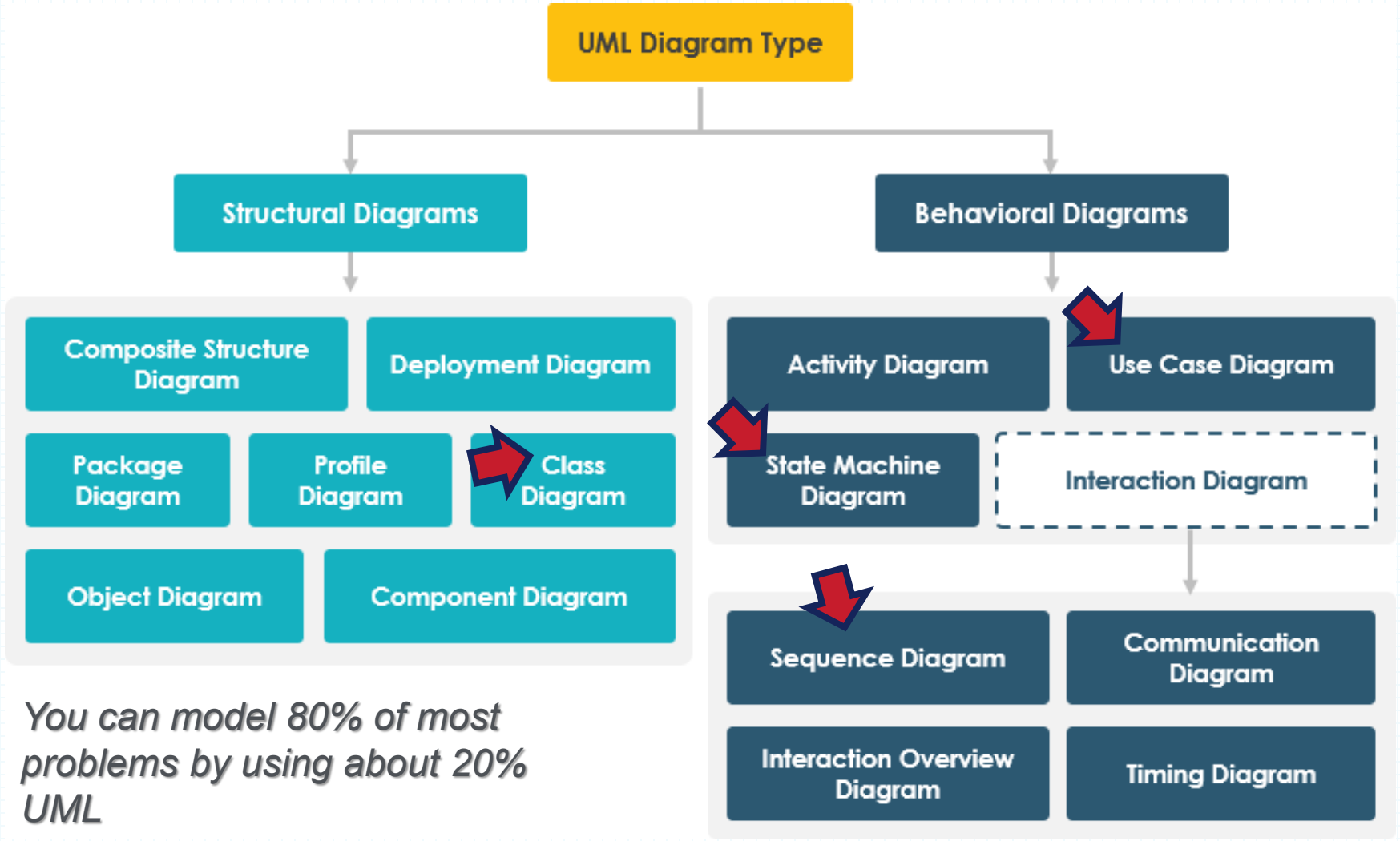
UML sequence diagrams



UML sequence diagrams



UML diagrams



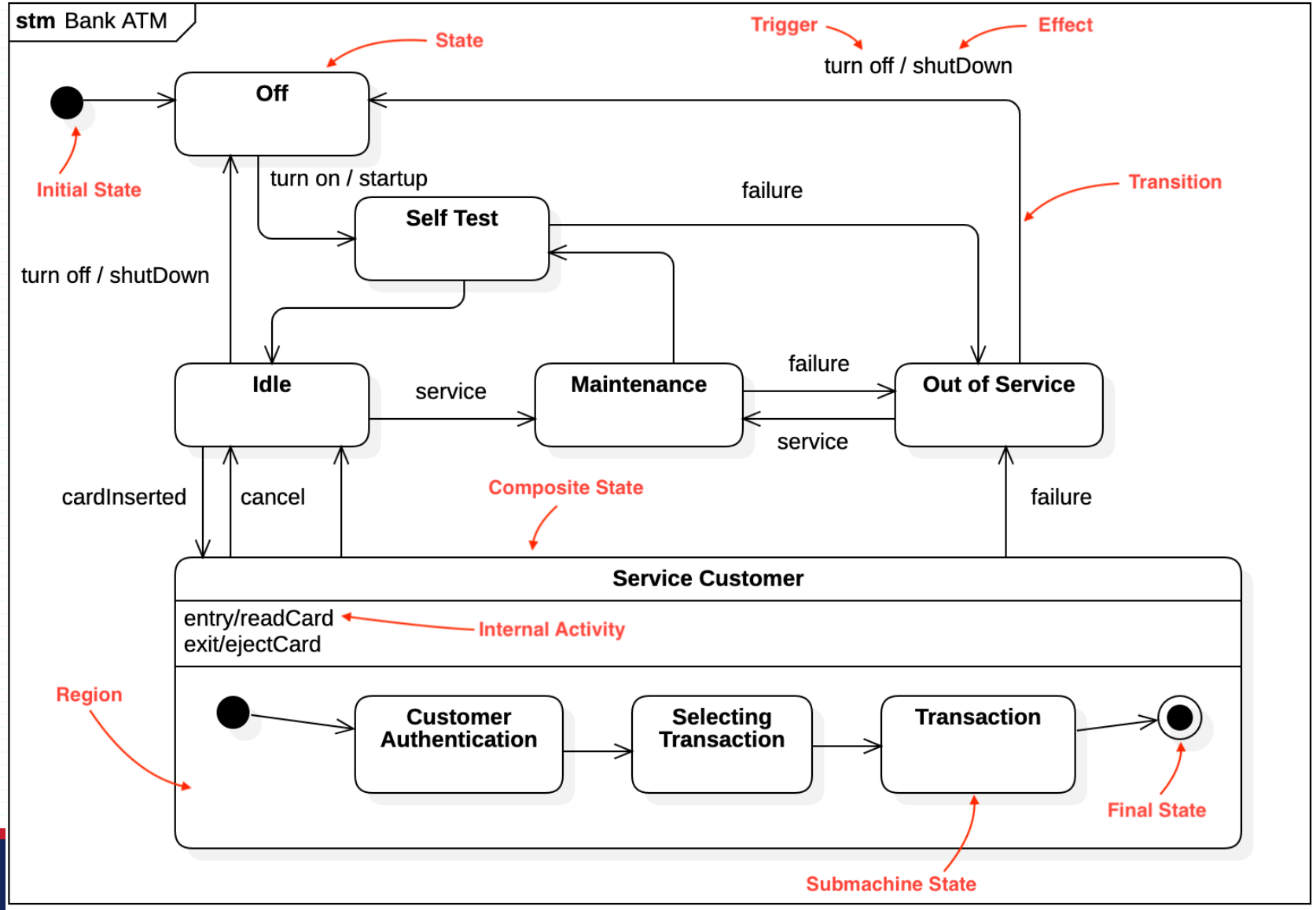
You can model 80% of most problems by using about 20% UML

State Chart Diagrams

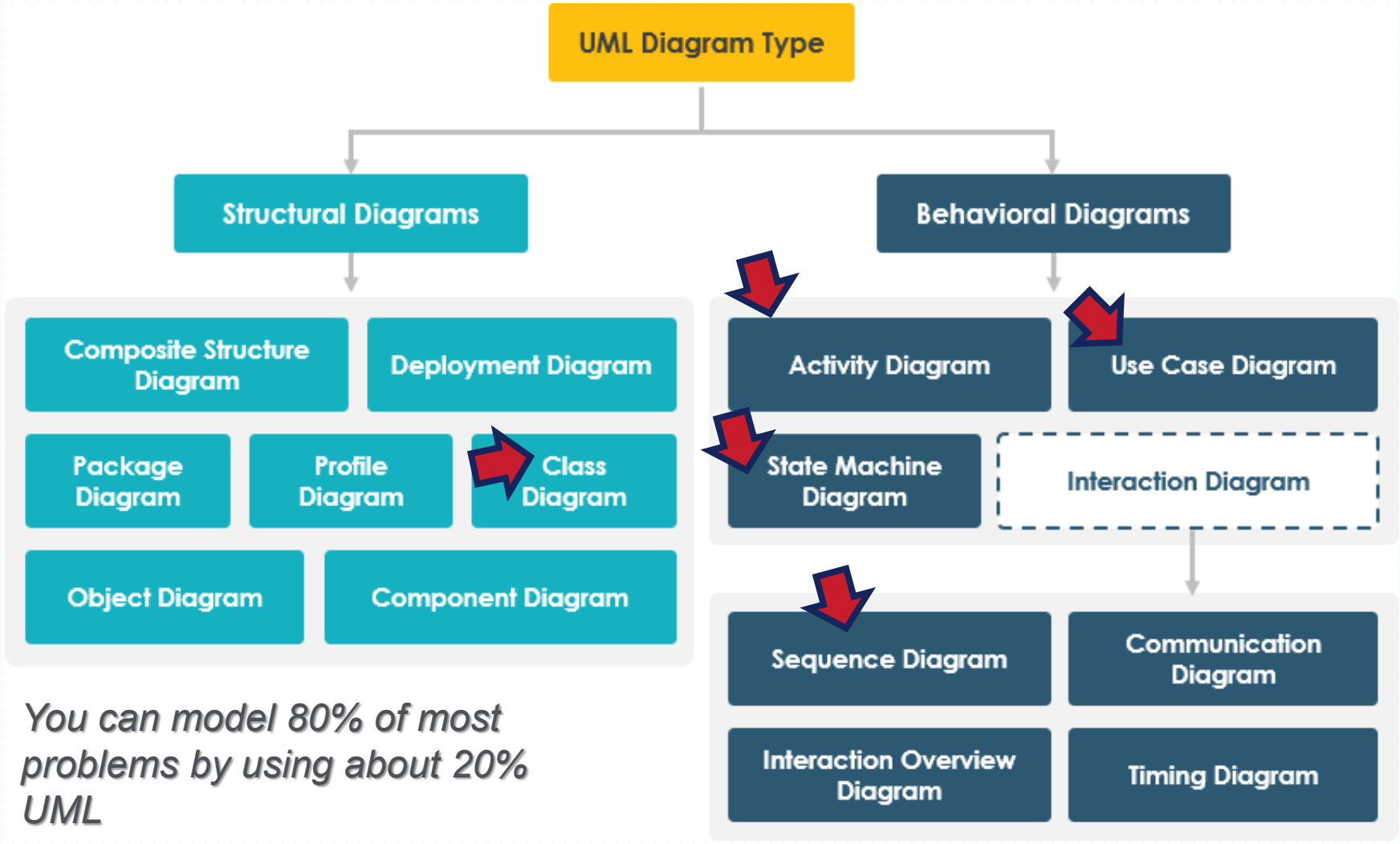
Why UML State Diagram?

- 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



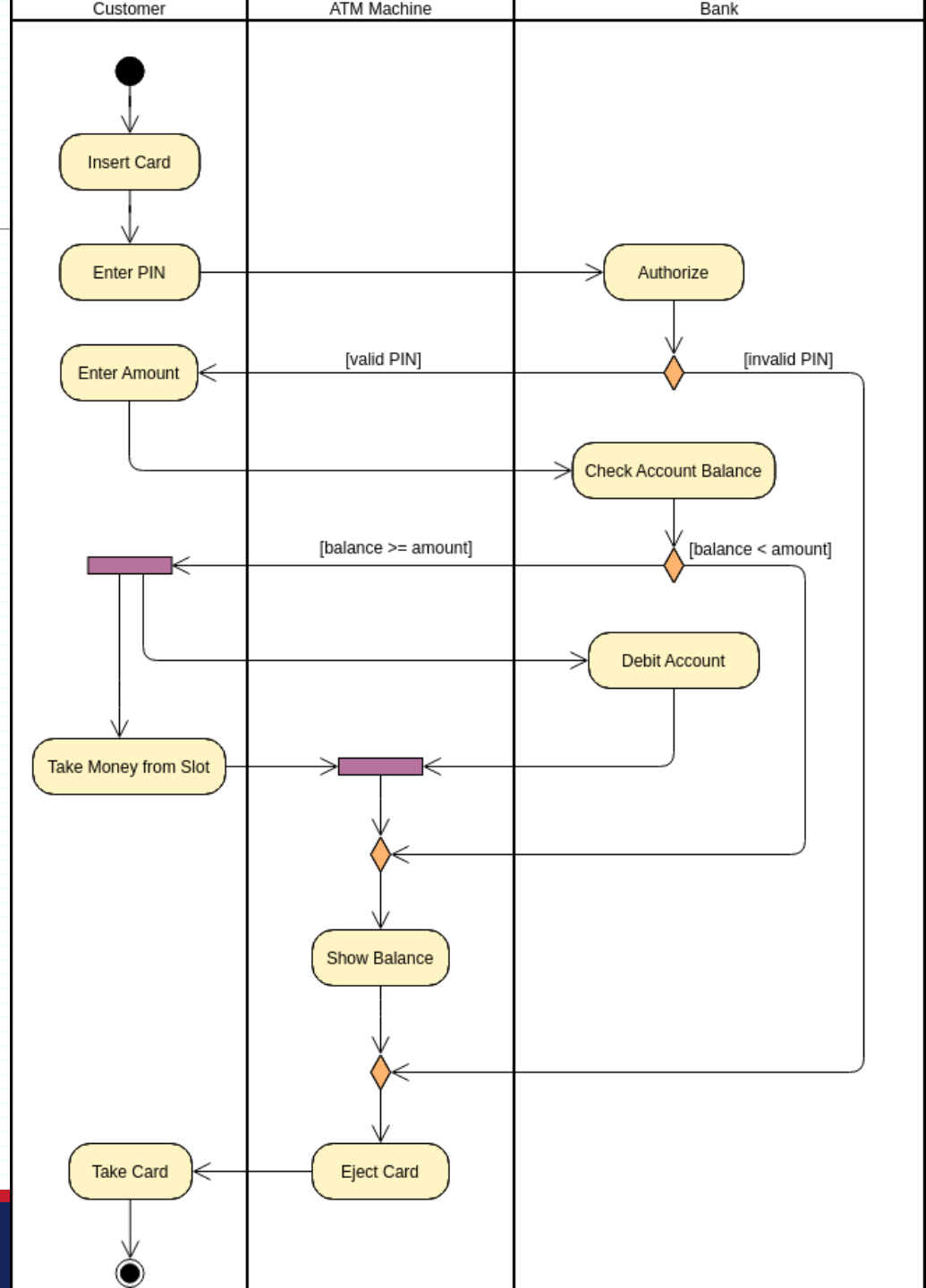
UML diagrams



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

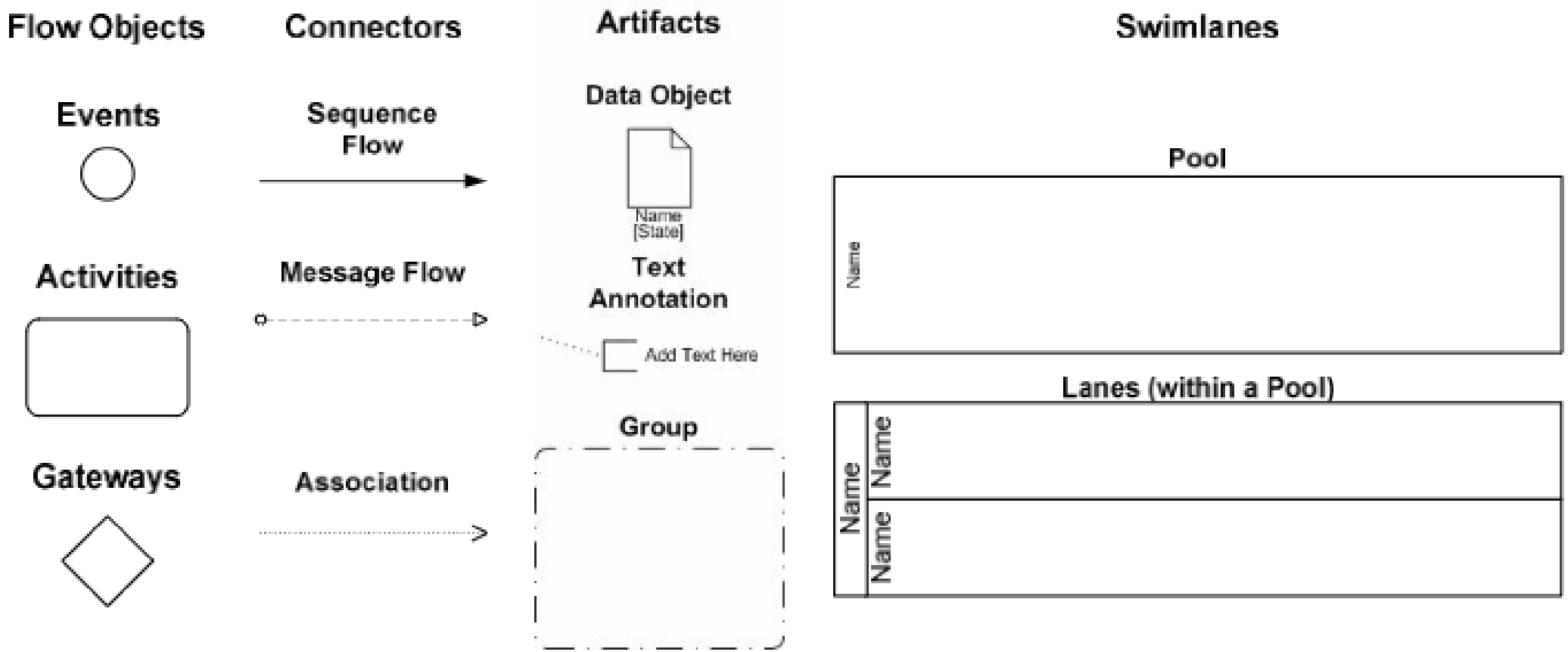
Activity Diagram

- **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

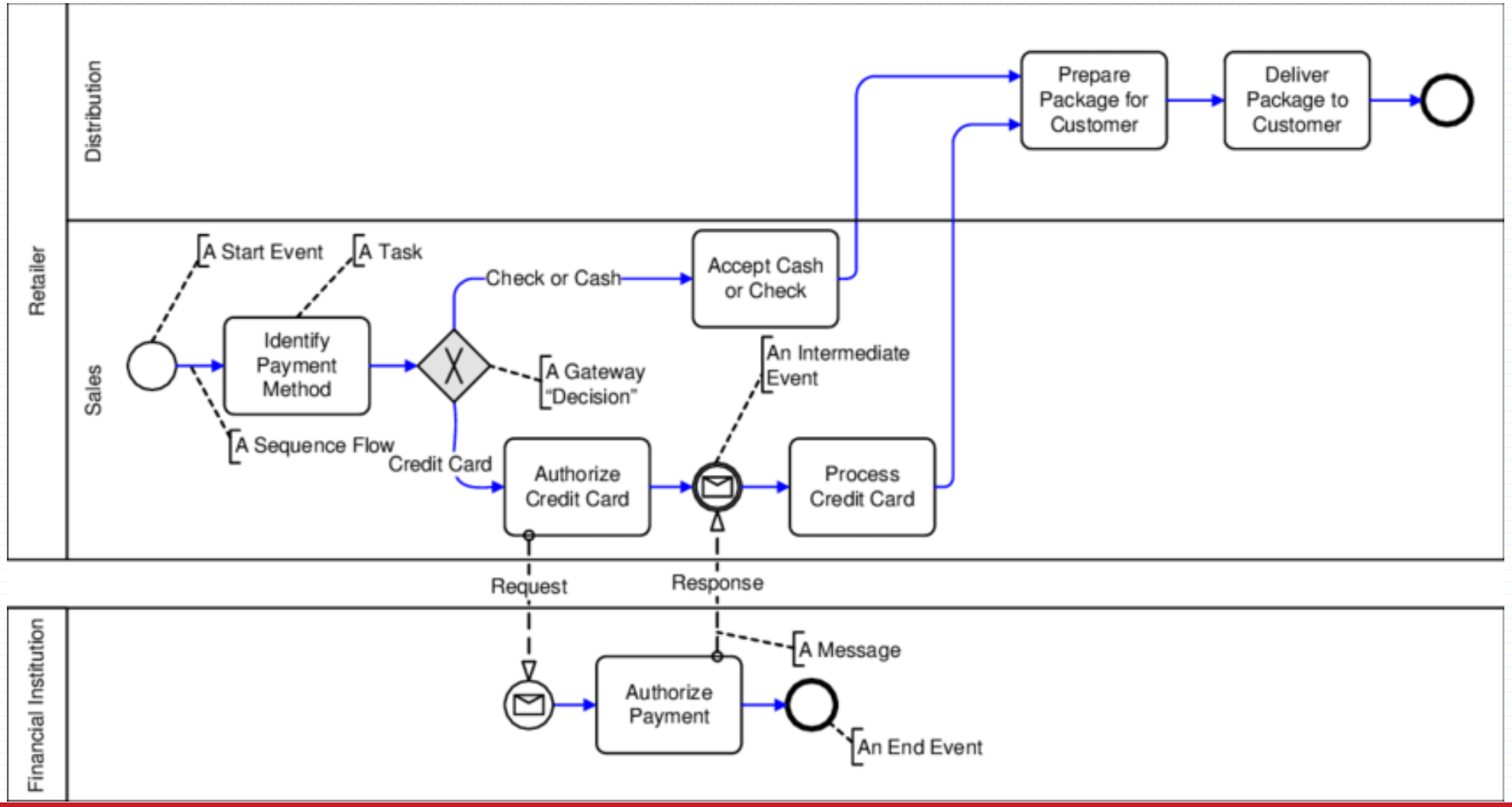


Other Modeling techniques!
UML is not everything in practical projects

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

Case Study: ARENA

Description

- ARENA is a multi-user, Web-based system for organizing and conducting tournaments.
- ARENA is game independent in the sense that organizers can adapt a new game to the ARENA game interface, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

Requirements Elicitation & Initial Analysis

Description

How to Start a New Software Project!!

b-based system for organizing and conducting

main terms, entities, and concepts (glossary)?

ent in the sense that org w
interface, upload it to t
immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.

- Organizers can also define new tournam ow players are mapped to a set of matches and how to nking of players by adding up their victories and losses (hence, figuring out who won the tourname

Main actions, services, (use cases)

Main processes, flows, etc

- To recoup s, organizers can also invite potential sponsors to display advertisement banners during games.

Good Starting Point

- ARENA is a multi-user, We… …ng and conducting tournaments.
- ARENA is game independe… …ganizers can adapt a new game to the ARENA game interfaces, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located …
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

Noun Phrases ≈ main concepts

Verbs ≈ Actions

Descript

You need more clarifications from your client to understand some terms

- ARENA is a multi-user, Wek
organizing and conducting
- ARENA is game independe
organizers can adapt a new
game interface, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

anners

Matches

multi-user web-based system

operational costs

Organizers

Players

potential sponsors

ranking of players

Server

Spectators

the tournament

tournament styles

New clarifications
➔Better Understanding
➔New possible Terms, actions, processes

- ARENA is a multi-u
organizing and co
- ARENA is game in
organizers can ada
game interface, upload it to the ARENA server,
and immediate
tournaments
anywhere on
- Organizers ca
describing ho
matches and
of players by a
(hence, figuring out who won the tourna

A Tournament is a series of Matches among a set of Players. Tournaments end with a single winner. The way Players accumulate points and Matches are scheduled is dictated by the League in which the Tournament is organized.

The TournamentStyle defines the number of Matches and their sequence for a given set of Players. For example, Players face all other Players in the Tournament exactly once in a round robin TournamentStyle.

Matches
multi-user web-based system
operational costs
Organizers
Players
potential sponsors
ranking of players
Server
Spectators
the tournament
tournament styles

Description

- ARENA is a multi-user, Web-based sys
organizing and conducting tournamer
- ARENA is game independent in the se
organizers can adapt a new game to t
game interface, upload it to the AREN
and immediate
tournaments
anywhere on
- Organizers ca
describing ho
matches and
of players by a
(hence, figuring out who won the tourna

A League represents a community for running Tournaments. A League is associated with a specific Game and TournamentStyle. Players registered with the League accumulate points according to the ExpertRating defined in the League. For example, a novice chess League has a different ExpertRating formula than an expert League.

A Tournament is a series of Matc among a set of Players. Tournam end with a single winner. The way ers accumulate points and Matche are scheduled is dictated by the League in which the Tournament is organized.

The TournamentStyle defines the number of Matches and their sequence for a given set of Players. For example, Players face all other Players in the Tournament exactly once in a round robin TournamentStyle.

operational costs

Organizers

Players

potential sponsors

ranking of players

Server

Spectators

the tournament

tournament styles

- ARE
orga
- ARE
orga
gam
and
tour
anyw
- Orga
desc
matc
of pl
(hen
- To re
also
adve

Table 4-4 Working glossary for ARENA. Keeping track of important terms and their definitions ensures consistency in the specification and ensures that developers use the language of the client.

Game	A Game is a competition among a number of Players that is conducted according to a set of rules. In ARENA, the term Game refers to a piece of software that enforces the set of rules, tracks the progress of each Player, and decides the winner. For example, tic tac toe and chess are Games.
Match	A Match is a contest between two or more Players following the rules of a Game. The outcome of a Match can be a single winner and a set of losers or a tie (in which their are no winners or losers). Some Games may disallow ties.
Tournament	A Tournament is a series of Matches among a set of Players. Tournaments end with a single winner. The way Players accumulate points and Matches are scheduled is dictated by the League in which the Tournament is organized.
League	A League represents a community for running Tournaments. A League is associated with a specific Game and TournamentStyle. Players registered with the League accumulate points according to the ExpertRating defined in the League. For example, a novice chess League has a different ExpertRating formula than an expert League.
TournamentStyle	The TournamentStyle defines the number of Matches and their sequence for a given set of Players. For example, Players face all other Players in the Tournament exactly once in a round robin TournamentStyle.

Finally!

Another Good Starting Point

develop an example scenario

Des

- ARENA is a multi-user
organizing and condu
- ARENA is game indep
organizers can adapt
game interface, upload it to the ARENA server,
and immediately announce and conduct
tournaments with players and spectators located
anywhere on the Internet.
- Organizers can also define new tournament styl
describing how players are mapped to a set of
matches and how to compute an overall ranking
of players by adding up their victories and losse
(hence, figuring out who won the tournament).
- To recoup their operational costs, organizers car
also invite potential sponsors to display
advertisement banners during games.

rs

Matches

multi-user web-based system

operational costs

Suggested Scenario
What if we want to lunch a Chess Tournament? we understand better the client's expectation of the system, including the boundary of the system and the kinds of interactions between the user and the system

tournament styles

Requirements and High-Level Use Case Diagram

Description

- ARENA is a multi-user, Web-based system for organizing and conducting tournaments.
- ARENA is game independent in the sense that organizers can adapt a new game to the ARENA game interface, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

advertisement banners

Game

Game interface

Matches

multi-user

operational costs

Organizers

Players

potential sponsors

ranking of players

Server

Spectators

the tournament

tournament styles

League

Description

- ARENA is a multi-user, Web-based system for organizing and conducting tournaments.
- ARENA is game independent in the sense that organizers can adapt a new game to the ARENA game interface, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

~~advertisement banners~~

~~Game~~

~~Game interface~~

~~Matches~~

multi-user

~~operational costs~~

Organizers

Players

potential sponsors

~~ranking of players~~

~~Server~~

Spectators

~~the tournament~~

~~tournament styles~~

~~League~~

Anonymous

Admin

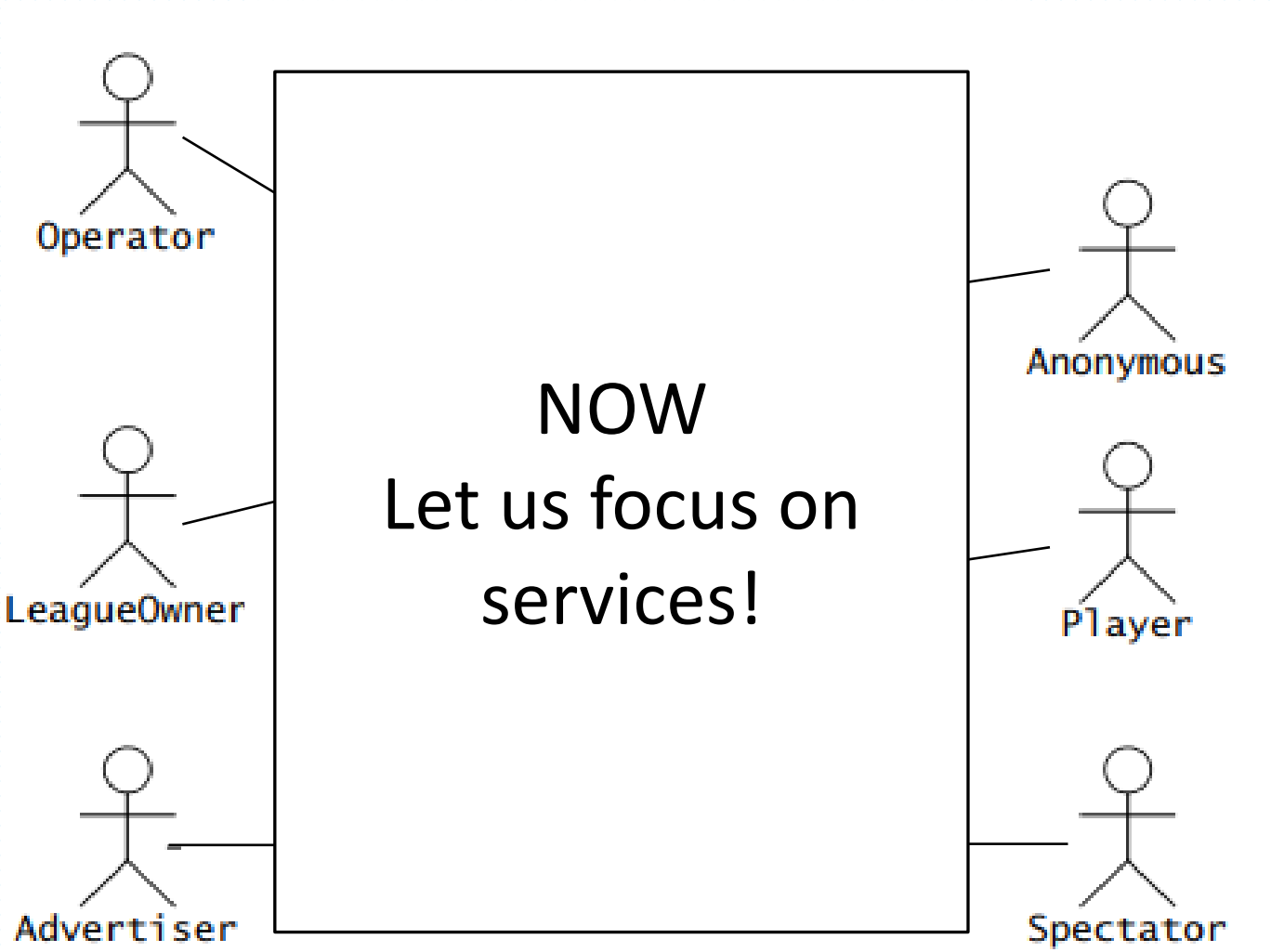
League owner

Player

advertiser

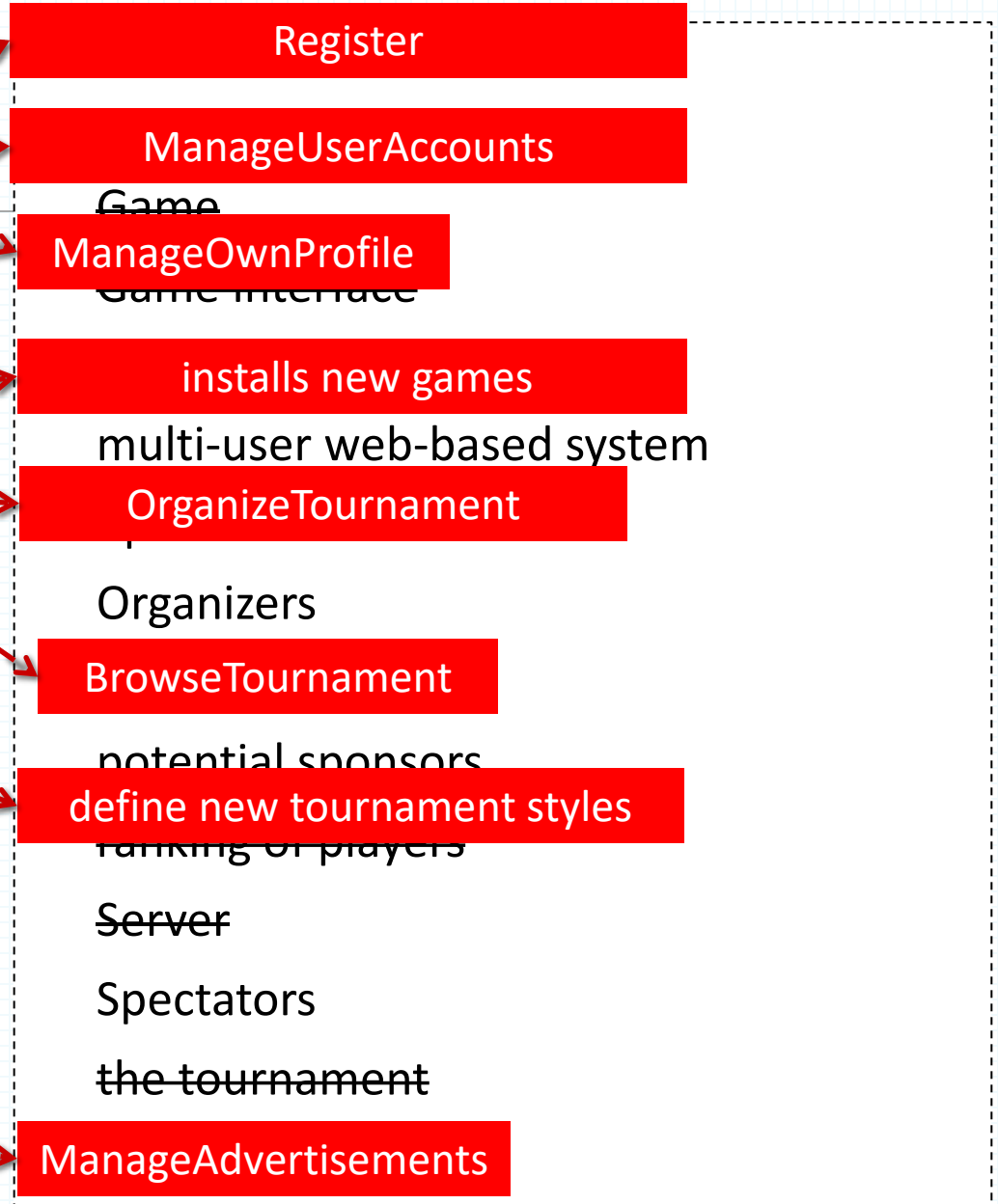
Spectators

Till Now ☺


Operator
LeagueOwner
Advertiser
NOW
Let us focus on
services!
Anonymous
Player
Spectator

Description

- ARENA is a multi-user, Web-based system for organizing and conducting tournaments.
- ARENA is game independent in the sense that organizers can adapt a new game to the ARENA game interface, upload it to the ARENA server, and immediately announce and conduct tournaments with players and spectators located anywhere on the Internet.
- Organizers can also define new tournament styles, describing how players are mapped to a set of matches and how to compute an overall ranking of players by adding up their victories and losses (hence, figuring out who won the tournament).
- To recoup their operational costs, organizers can also invite potential sponsors to display advertisement banners during games.

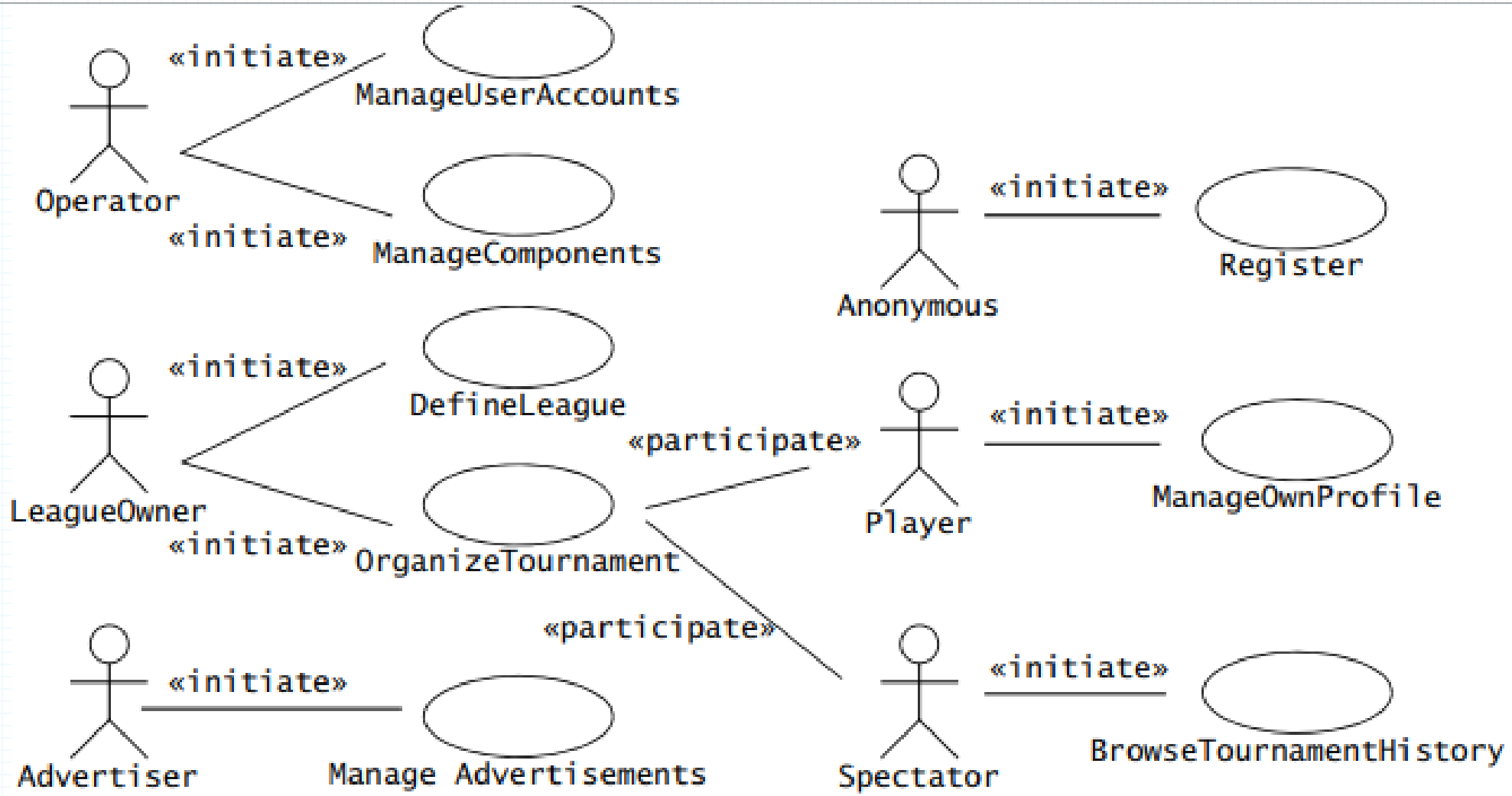


Main services for each Actor

High Level Functional Requirements

- The **operator** should be able to define new games, define new tournament styles (e.g., knock-out tournaments, championships, best of series), define new expert rating formulas, and manage users.
- **League owners** should be able to define a new league, organize and announce new tournaments within a league, conduct a tournament, and declare a winner.
- **Players** should be able to register in an arena, apply for a league, play the matches that are assigned to the player, or drop out of the tournament.
- **Spectators** should be able to monitor any match in progress and check scores and statistics of past matches and players. Spectators do not need to register in an arena.
- **The advertiser** should be able to upload new advertisements, select an advertisement scheme (e.g., tournament sponsor, league sponsor), check balance due, and cancel advertisements.

High-level use cases



High-level use cases

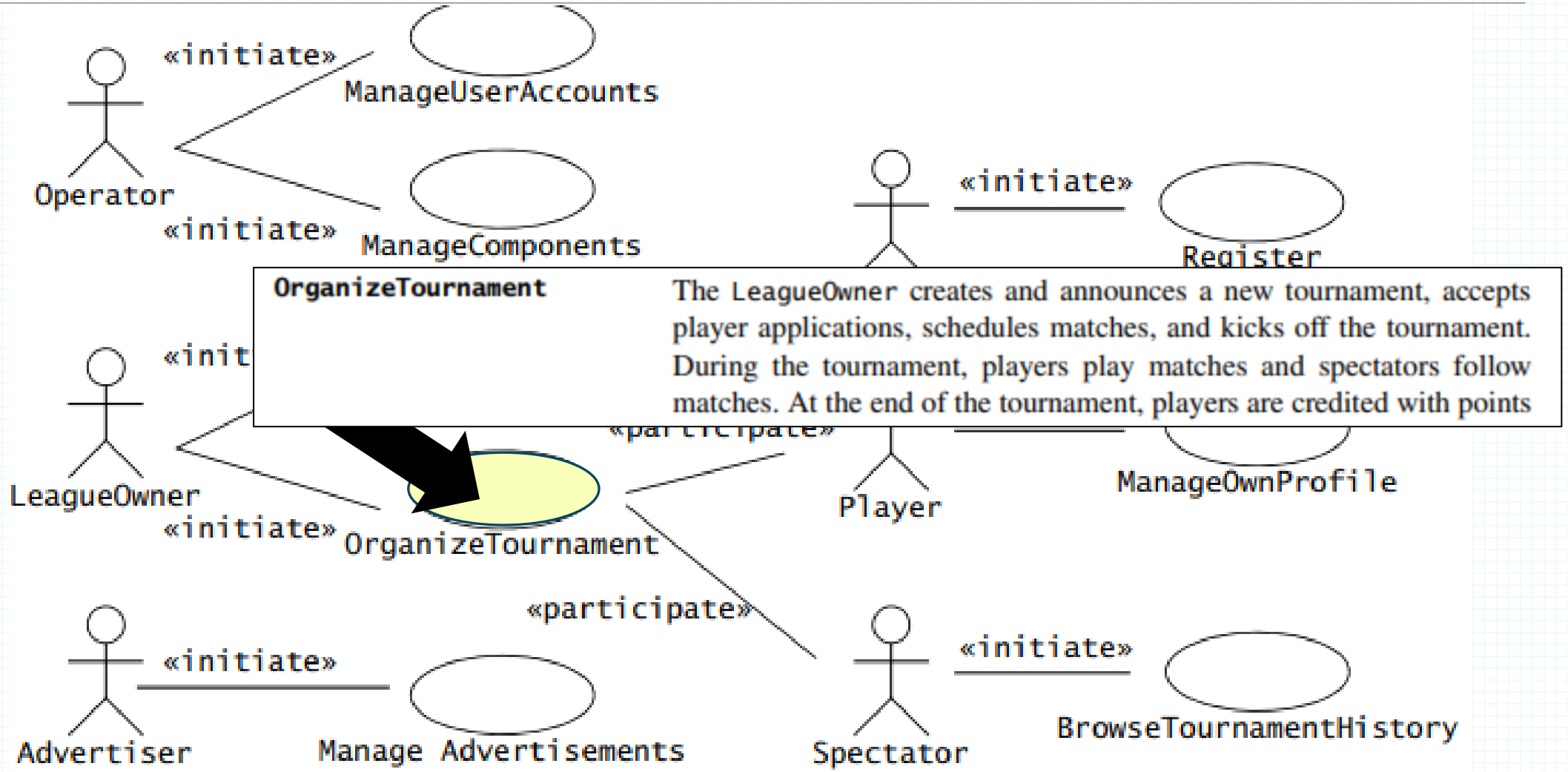
Register	Anonymous users register with an Arena for a Player or a League-Owner account. User accounts are required before applying for a tournament or organizing a league. Spectators do not need accounts.
ManageUserAccounts	The Operator accepts registrations from LeagueOwners and for Players, cancels existing accounts, and interacts with users about extending their accounts.
ManageComponents	The Operator installs new games and defines new tournament styles
DefineLeague	The LeagueOwner defines a new league (
OrganizeTournament	The LeagueOwner creates and announces a new tournament, accepts player applications, schedules matches, and kicks off the tournament. During the tournament, players play matches and spectators follow matches. At the end of the tournament, players are credited with points
ManageAdvertisements	The Advertiser uploads banners and sponsors league or tournaments
ManageOwnProfile	The Players manage their subscriptions to mailing lists and answer a marketing survey.
BrowseTournamentHistory	Spectators examine tournament statistics and player statistics, and replay matches that have already been concluded (

Non-Functional Requirements

- **Low operating cost:** The operator must be able to install and administer an arena **without purchasing additional software** components and without the help of a full-time system administrator.
- **Extensibility:** The operator must be able to **add new games**, new tournament styles, and new expert rating formulas. Such additions may require the system to be temporarily shut down and new modules (e.g., Java classes) to be added to the system. However, no modifications of the existing system should be required.
- **Scalability:** The system must support the kick-off of **many parallel tournaments** (e.g., 10), each involving up to 64 players and several hundreds of simultaneous spectators.
- **Low-bandwidth network:** Players should be able to play matches via a 56K analog modem or faster.

Detailed Use Cases

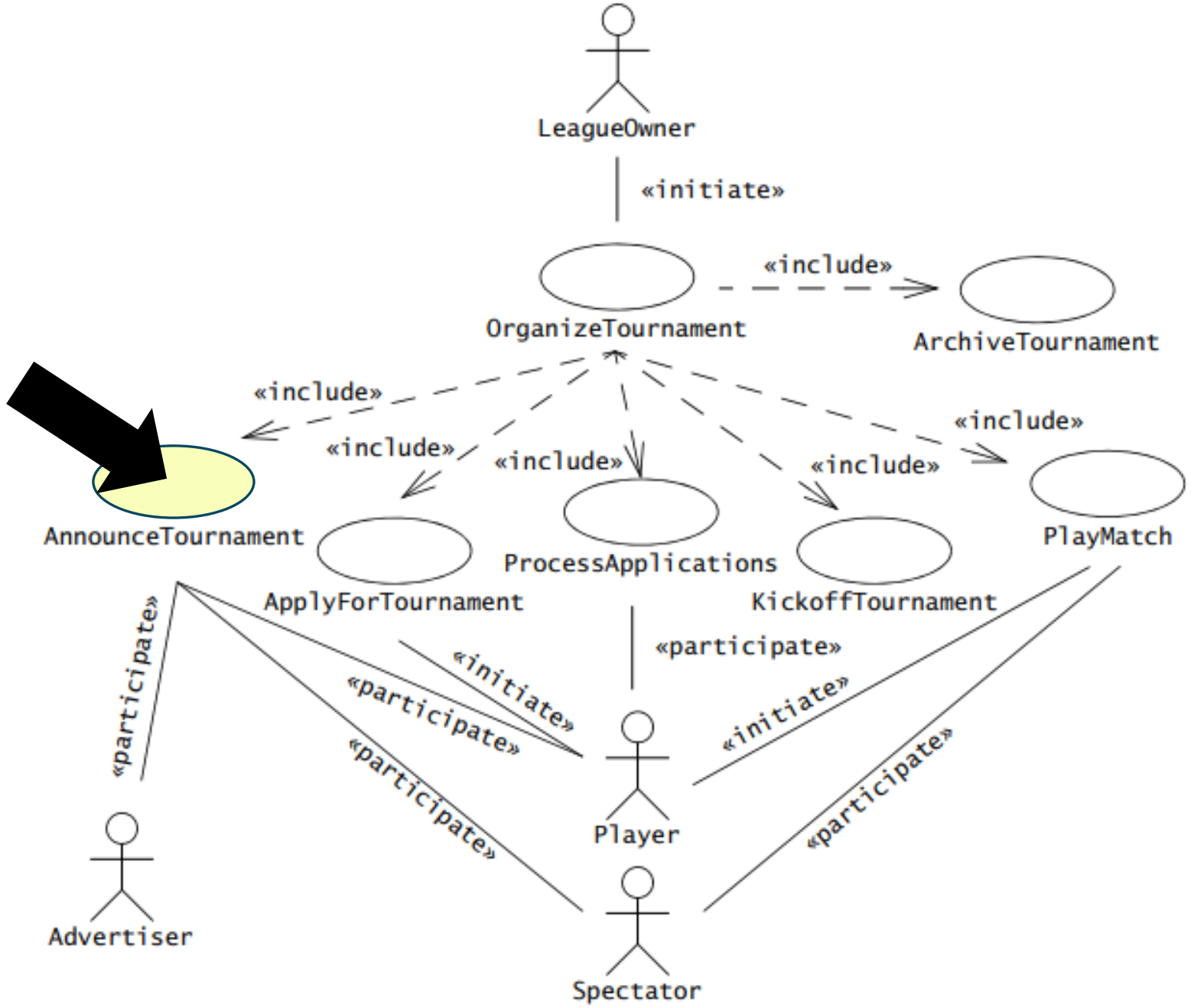
High-level use cases



Use Case Specification

Use case name	`OrganizeTournament`
Participating actors	Initiated by `LeagueOwner` Communicates with `Advertiser`, `Player`, and `Spectator`
Flow of events	1. The `LeagueOwner` creates a `Tournament`, solicits sponsorships from `Advertisers`, and announces the `Tournament` (include use case `AnnounceTournament`). 2. The `Players` apply for the `Tournament` (include use case `ApplyForTournament`). 3. The `LeagueOwner` processes the `Player` applications and assigns them to matches (include use case `ProcessApplications`). 4. The `LeagueOwner` kicks off the `Tournament` (include use case `KickoffTournament`). 5. The `Players` compete in the matches as scheduled and `Spectators` view the matches (include use case `PlayMatch`). 6. The `LeagueOwner` declares the winner and archives the `Tournament` (include use case `ArchiveTournament`).
Entry condition	• The `LeagueOwner` is logged into `ARENA`.
Exit conditions	• The `LeagueOwner` archived a new tournament in the `ARENA` archive and the winner has accumulated new points in the league, OR • The `LeagueOwner` cancelled the tournament and the players' standing in the league is unchanged.

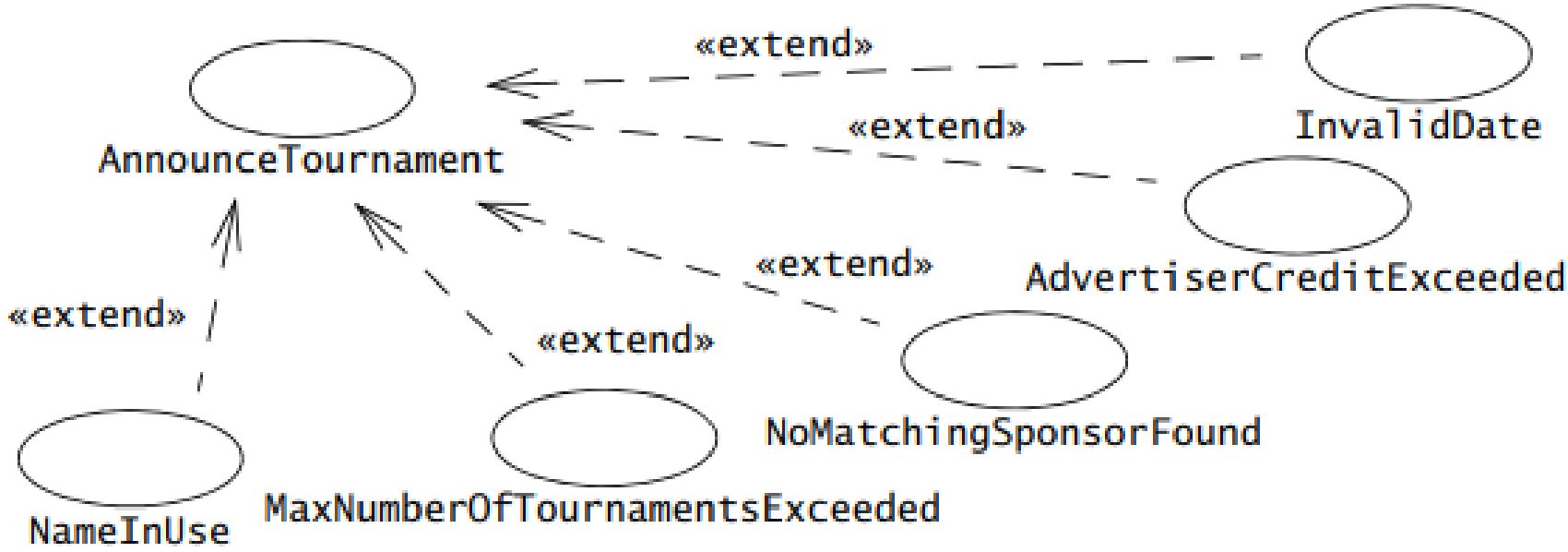
Detailed use cases refining the OrganizeTournament high-level use case.



Name	AnnounceTournament
Participating actors	Initiated by LeagueOwner Communicates with Player, Advertiser, Spectator
Flow of events	1. The LeagueOwner requests the creation of a tournament. 2. The system checks if the LeagueOwner has exceeded the number of tournaments in the league or in the arena. If not, the system presents the LeagueOwner with a form. 3. The LeagueOwner specifies a name, application start and end dates during which Players can apply to the tournament, start and end dates for conducting the tournament, and a maximum number of Players. 4. The system asks the LeagueOwner whether an exclusive sponsorship should be sought and, if yes, presents a list of Advertisers who expressed the desire to be exclusive sponsors. 5. If the LeagueOwner decides to seek an exclusive sponsor, he selects a subset of the names of the proposed sponsors. 6. The system notifies the selected sponsors about the upcoming tournament and the flat fee for exclusive sponsorships. 7. The system communicates their answers to the LeagueOwner. 8. If there are interested sponsors, the LeagueOwner selects one of them. 9. The system records the name of the exclusive sponsor and charges flat fee for sponsorships to the Advertiser's account. From tisement banners associated with the tournament ive sponsor only. either because no

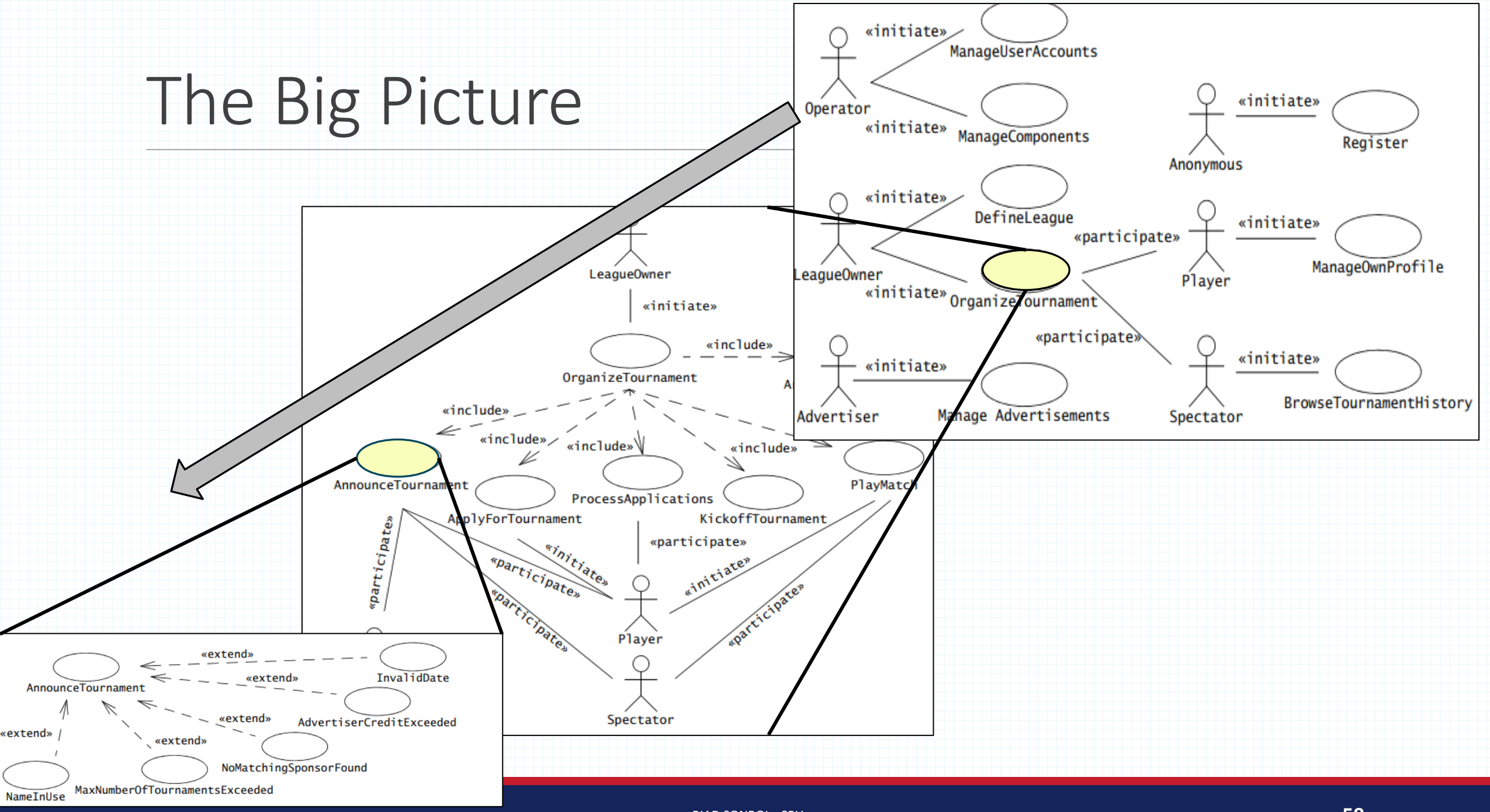
See the level of details!

Deeper



AdvertiserCreditExceeded	The system removes the Advertiser from the list of potential sponsors.
InvalidDate	The system informs the LeagueOwner and prompts for a new date.
MaxNumberOfTournaments Exceeded	The AnnounceTournament use case is terminated.
NameInUse	The system informs the LeagueOwner and prompts for a new name.
NoMatchingSponsorFound	The system skips the exclusive sponsor steps and chooses random advertisements from the advertisement pool.

The Big Picture



Identifying Entity Objects

a few simple heuristics to distinguish between noun phrases that correspond to objects, attributes, and associations

- Attributes represent a single property of an object.

Flow of events

1. The **LeagueOwner** requests the creation of a **tournament**.
2. The system checks if the LeagueOwner has exceeded the **number of tournaments** in the **league** or in the **arena**. If not, the system presents the LeagueOwner with a form.
3. The LeagueOwner specifies a **name**, **application start and end dates** during which Players can apply to the tournament, **start and end dates** for conducting the tournament, and a **maximum number of Players**.
4. The system asks the LeagueOwner whether an exclusive sponsorship should be sought and, if yes, presents a **list of Advertisers** who expressed the desire to be **exclusive sponsors**.
5. If the LeagueOwner decides to seek an exclusive sponsor, he selects a subset of the **names** of the **proposed sponsors**.
6. The system notifies the selected sponsors about the upcoming tournament and the **flat fee** for exclusive sponsorships.
7. The system communicates their **answers** to the LeagueOwner.
8. If there are interested sponsors, the LeagueOwner selects one of them.
9. The system records the **name** of the exclusive sponsor and charges the flat fee for sponsorships to the **Advertiser's account**. From now on, all **advertisement banners** associated with the tournament are provided by the exclusive sponsor only.
10. If no sponsors were selected (either because no Advertisers were interested or the LeagueOwner did not select any), the advertisement banners are selected at random and charged to each Advertiser's account on a per unit basis.
11. Once the sponsorship issues is closed, the system prompts the LeagueOwner with a **list of groups of Players, Spectators, and Advertisers** that could be interested in the new tournament.

a few simple heuristics to distinguish between noun phrases that correspond to objects, attributes, and associations

- Attributes have simple types.

- Complex concepts are represented as objects that are related to other objects with associations.

Flow of events

1. The **LeagueOwner** requests the creation of a **tournament**.
2. The system checks if the LeagueOwner has exceeded the **number of tournaments** in the **league** or in the **arena**. If not, the system presents the LeagueOwner with a form.
3. The LeagueOwner specifies a **name**, **application start and end dates** during which Players can apply to the tournament, **start and end dates** for conducting the tournament, and a **maximum number of Players**.
4. The system asks the LeagueOwner whether an exclusive sponsorship should be sought and, if yes, presents a **list of Advertisers** who expressed the desire to be **exclusive sponsors**.
5. If the LeagueOwner decides to seek an exclusive sponsor, he selects a subset of the **names** of the **proposed sponsors**.
6. The system notifies the selected sponsors about the upcoming tournament and the **flat fee** for exclusive sponsorships.
7. The system communicates their **answers** to the LeagueOwner.
8. If there are interested sponsors, the LeagueOwner selects one of them.
9. The system records the **name** of the exclusive sponsor and charges the flat fee for sponsorships to the **Advertiser's account**. From now on, all **advertisement banners** associated with the tournament are provided by the exclusive sponsor only.
10. If no sponsors were selected (either because no Advertisers were interested or the LeagueOwner did not select any), the advertisement banners are selected at random and charged to each Advertiser's account on a per unit basis.
11. Once the sponsorship issues is closed, the system prompts the LeagueOwner with a **list of groups of Players, Spectators, and Advertisers** that could be interested in the new tournament.

Entity Object	Attributes & Associations	Definition
Account	• balance • history of charges (?) • history of payments (?)	An Account represents the amount currently owed by an Advertiser, a history of charges, and payments.
Advertiser	• name • leagues of interest for exclusive sponsorships (?) • sponsored tournaments • account	Actor interested in displaying advertisement banners during the Matches.
Advertisement	• associated game (?)	Image provided by an Advertiser for display during matches.
Arena	• max number of tournaments • flat fee for sponsorships (?) • leagues *(implied)* • interest groups *(implied)*	An instantiation of the ARENA system.
Game		A Game is a competition among a number of Players that is conducted according to a set of rules. In ARENA, the term Game refers to a piece of software that enforces the set of rules, tracks the progress of each Player, and decides the winner.
InterestGroup	• list of players, spectators, or advertisers • games and leagues of interests *(implied)*	InterestGroups are lists of users in the ARENA which share an interest (e.g, for a game or a league). InterestGroups are used as mailing lists for notifying potential actors of new events.
League	• max number of tournament • game	A League represents a community for running Tournaments. A League is associated with a specific Game and TournamentStyle. Players registered with the League accumulate points according to the ExpertRating of the League.

Entity Object	Attributes & Associations	Definition
LeagueOwner	• name *(implied)*	The actor creating a League and responsible for organizing Tournaments within the League.
Match	• tournament • players	A Match is a contest between two or more Players within the scope of a Game. The outcome of a Match can be a single winner and a set of losers or a tie (in which their are no winners or losers). Some TournamentStyles may disallow ties.
Player	• name *(implied)*	
Tournament	• name • application start date • application end date • play start date • play end date • max number of players • exclusive sponsor	A Tournament is a series of Matches among a set of Players. Tournaments end with a single winner. The way Players accumulate points and Matches are scheduled is dictated by the League in which the Tournament is organized.

Identifying Boundary Objects

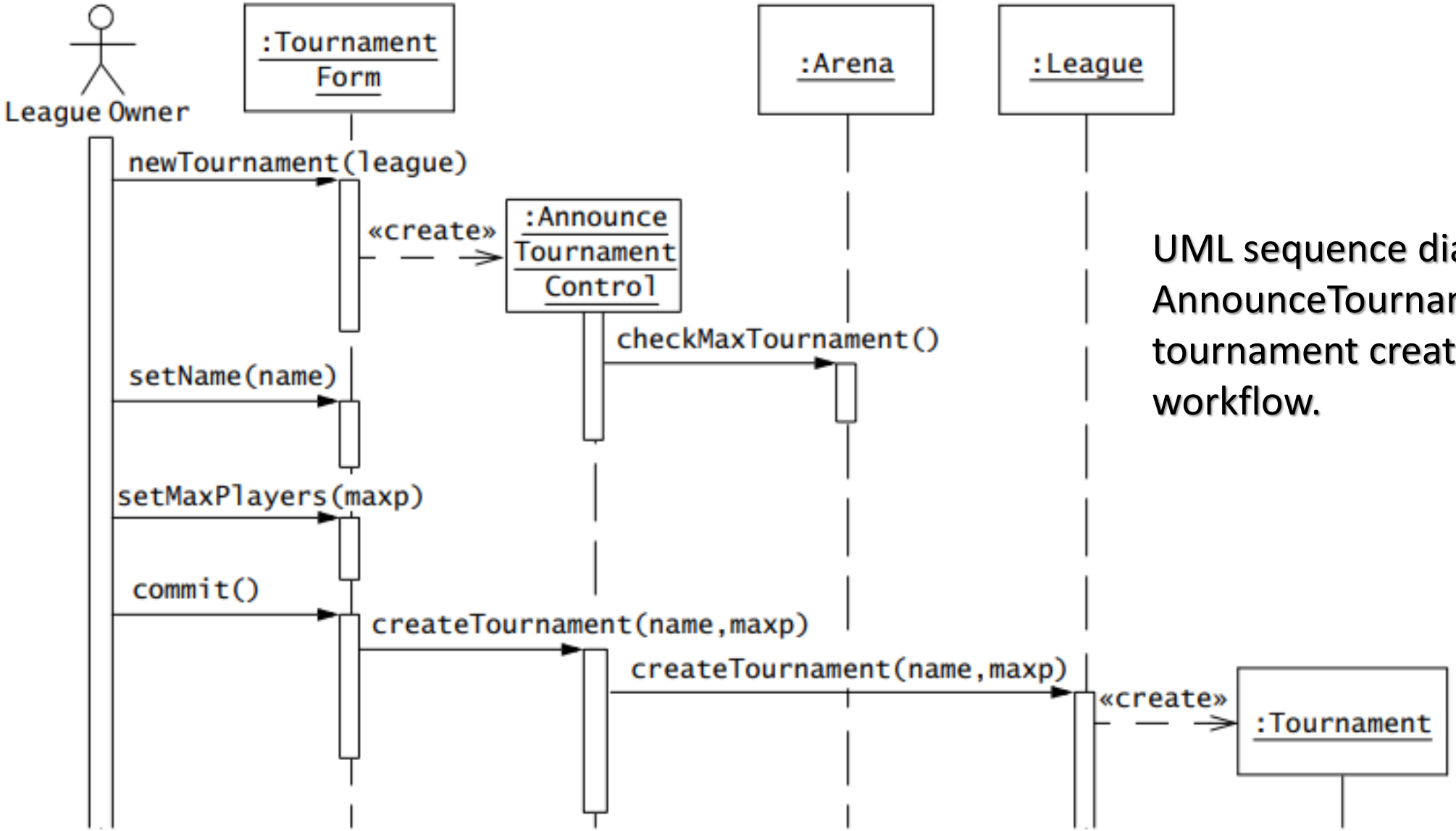
- Boundary objects represent the interface between the system and the actors. They are identified from the use cases and usually represent the user interface at a coarse level.

Boundary Object	Definition
`TournamentForm`	Form used by the `LeagueOwner` to specify the properties of a `Tournament` during creation or editing.
`RequestSponsorshipForm`	Form used by the `LeagueOwner` to request sponsorships from interested `Advertisers`.
`SponsorshipRequest`	Notice received by `Advertisers` requesting sponsorship.
`SponsorshipReply`	Notice received by `LeagueOwner` indicating whether an `Advertiser` wants the exclusive sponsorship of the tournament.
`SelectExclusiveSponsorForm`	Form used by the `LeagueOwner` to close the sponsorship issue.
`NotifyInterestGroupsForm`	Form used by the `LeagueOwner` to notify interested users.
`InterestGroupNotice`	Notice received by interested users about the creation of a new `Tournament`.

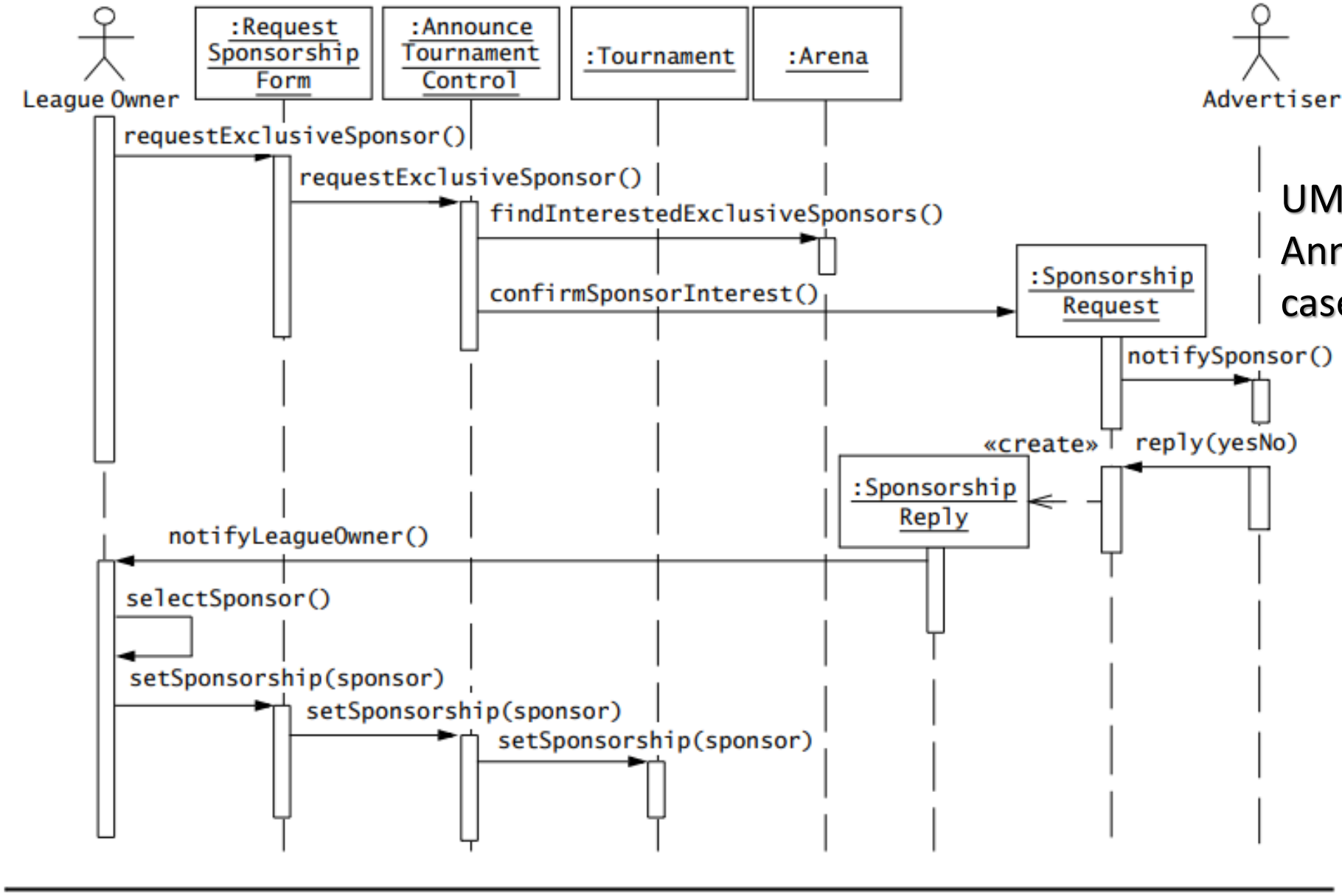
Identifying Control Objects

- Control objects represent the coordination among boundary and entity objects.
- In the common case, a single control object is created at the beginning of the use case and accumulates all the information needed to complete the use case, then destroyed with the completion of the use case
- Example:
  - **AnnounceTournamentControl**, which is responsible for sending and collecting notices to Advertisers, checking resource availability, and, finally, notifying interested users

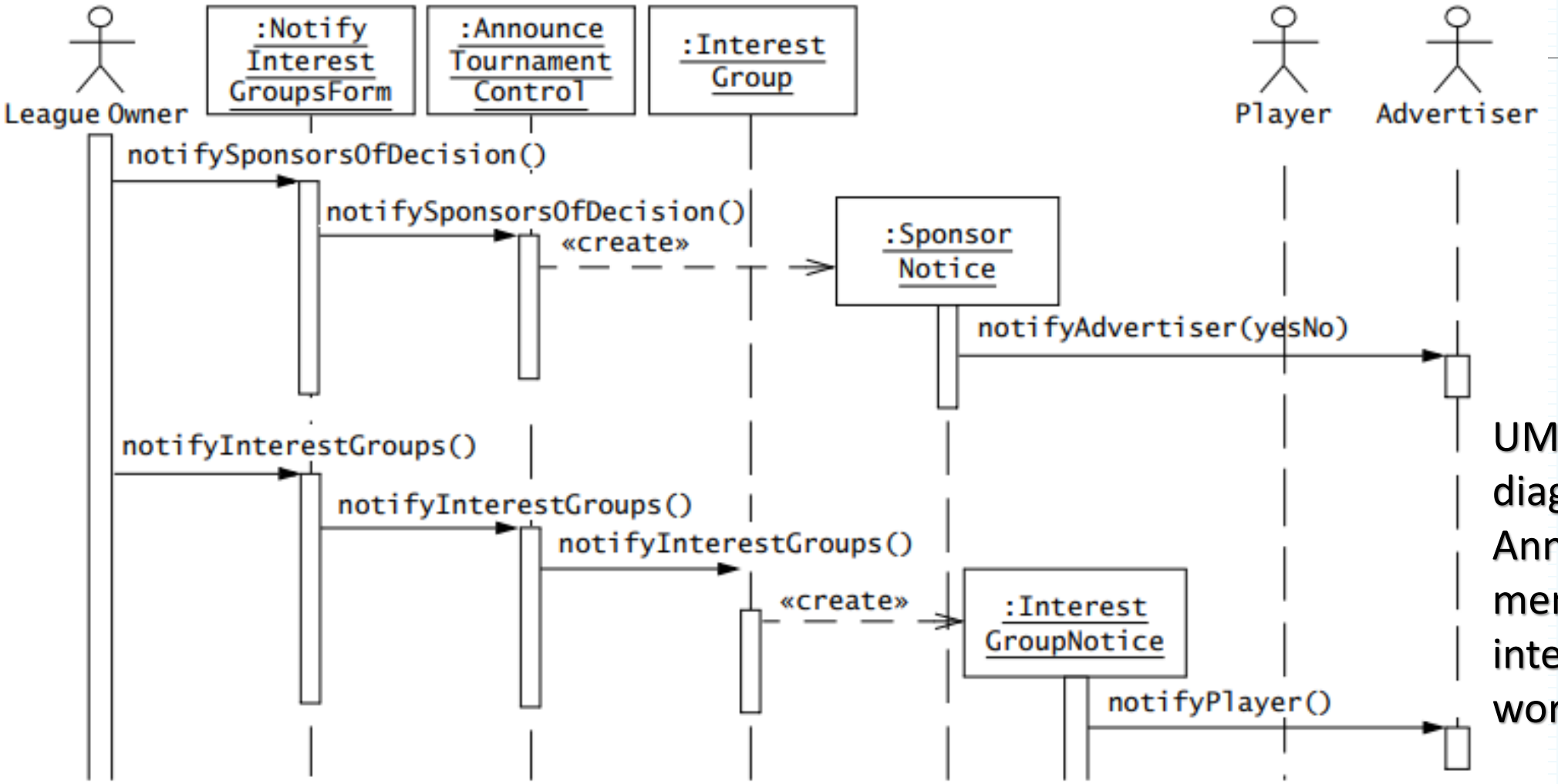
Modeling Interactions Among Objects



UML sequence diagram for AnnounceTournament, tournament creation workflow.

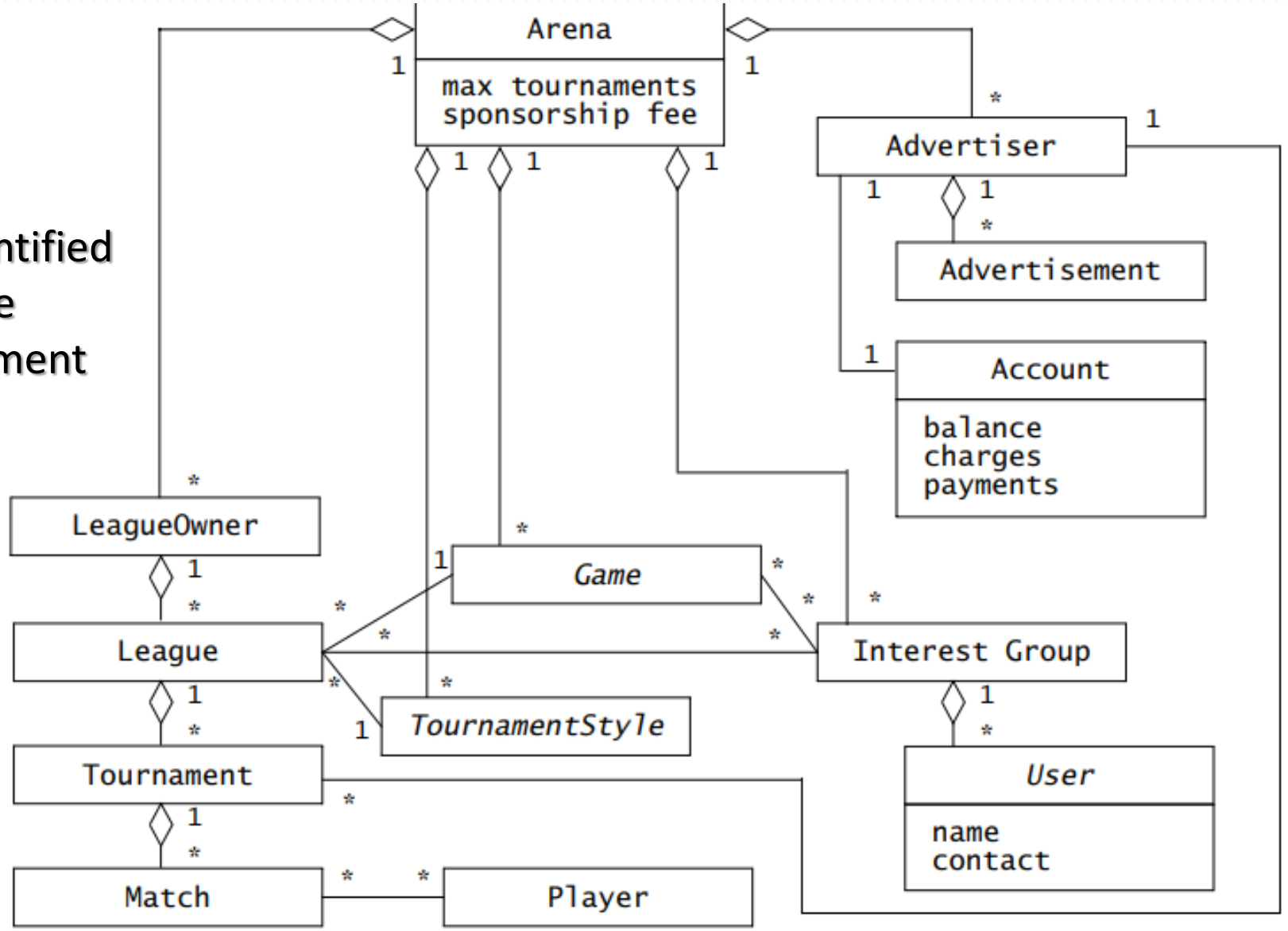
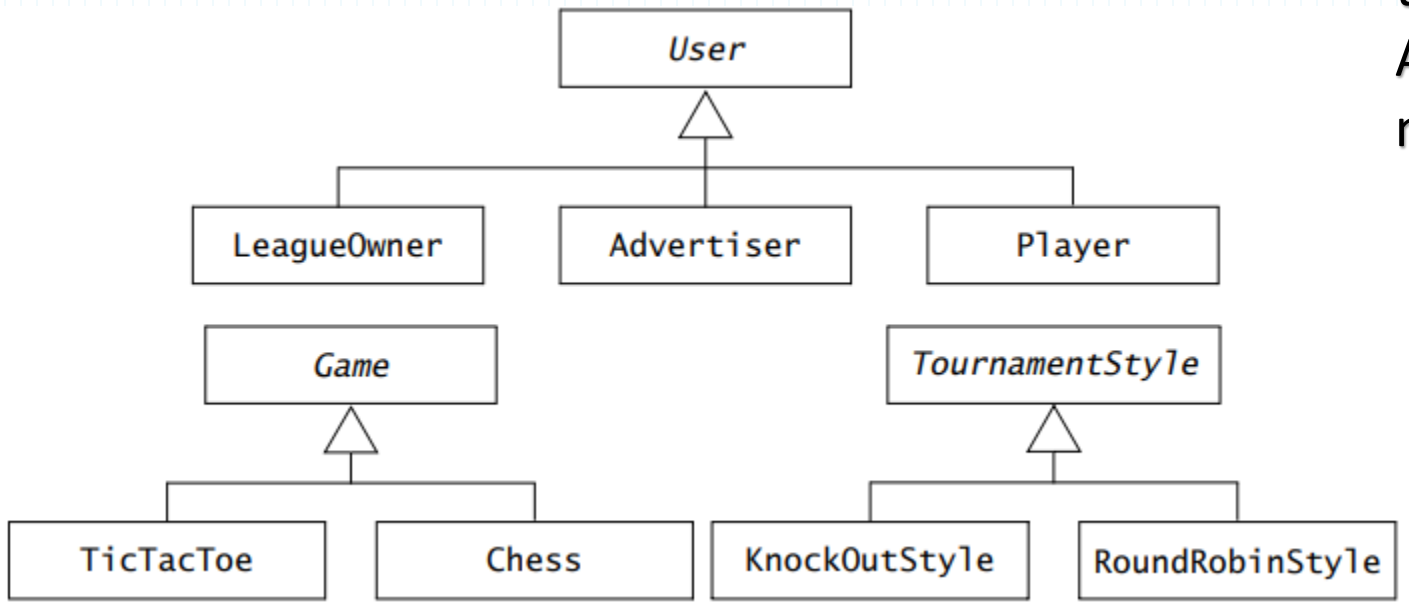


UML sequence diagram for AnnounceTournament use case, sponsorship workflow



UML sequence diagram for AnnounceTournament use case, interest group workflow

Entity objects identified after analyzing the AnnounceTournament use case



Inheritance hierarchy among entity objects of the AnnounceTournament use case