Software engineering

Analysis

TASK

Choose one use-case from your project (medium complexity) and build (use Abbot heuristic)

- ➢ Object model Identified entity, boundary and control objects
- ➢ Identifying associations, aggregates, attributes and create a class diagram
- ➢ Modelling inheritance relationships
- ➢ Create the sequence diagram for the use-case
- ➢ Reviewing original use-case, adjust if neccesary
- ➢ Complete Requirements Analysis Document

Software engineering

Analysis

Object model

- focuses on the individual concepts that are manipulated by the system, their properties and their relationships
- represented by class (high-level abstractions of the class from the source code) and object diagrams
- consists of
  - Entity objects - **persistent information** tracked by the system
  - Boundary objects- **the interaction between the actors and the system**
  - Control objects - in charge of **realizing use cases** (responsible for coordinating boundary and entity objects)

- **Class diagram** - describes the interdependencies of objects

- An **association** shows a relationship between two or more classes. Every association should be named, and roles should be assigned to each end

- **Attributes** are properties of individual objects

- **Inheritance** enables us to organize concepts into hierarchies.

- **Generalization** is the modeling activity that identifies abstract concepts from lower-level ones.

- **Specialization** is the activity that identifies more specific concepts from a high-level one.

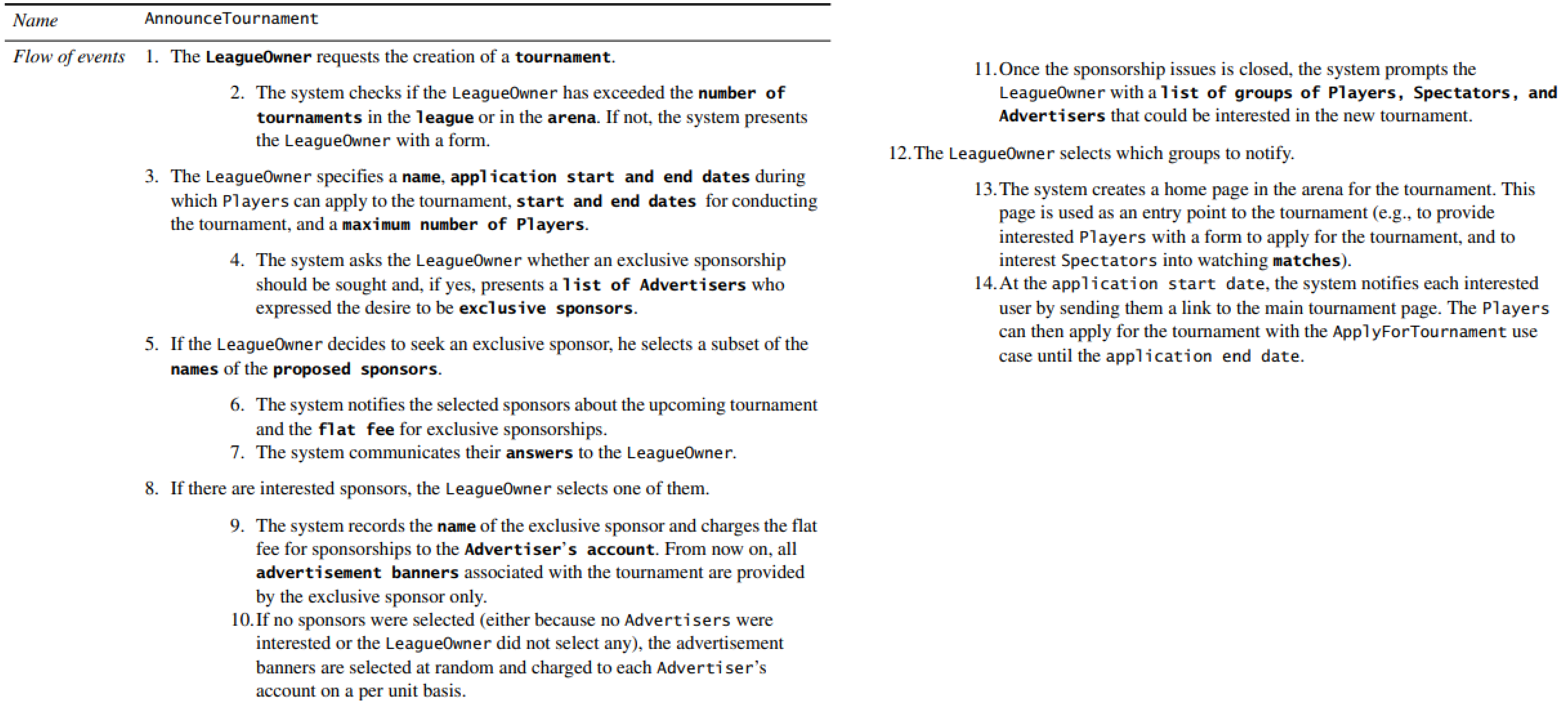
Dynamic model

- focuses on the behavior of the system (model the user interface at a coarse level)
- represented by:
  - state machine - **the behavior of a single object** (or a group of very tightly coupled objects).
  - sequence diagrams - **the interactions among a set of objects** during a single use case over time

- **Sequence diagram** – shows how the behavior of a use case is distributed among its participating objects

ARENA Case Study

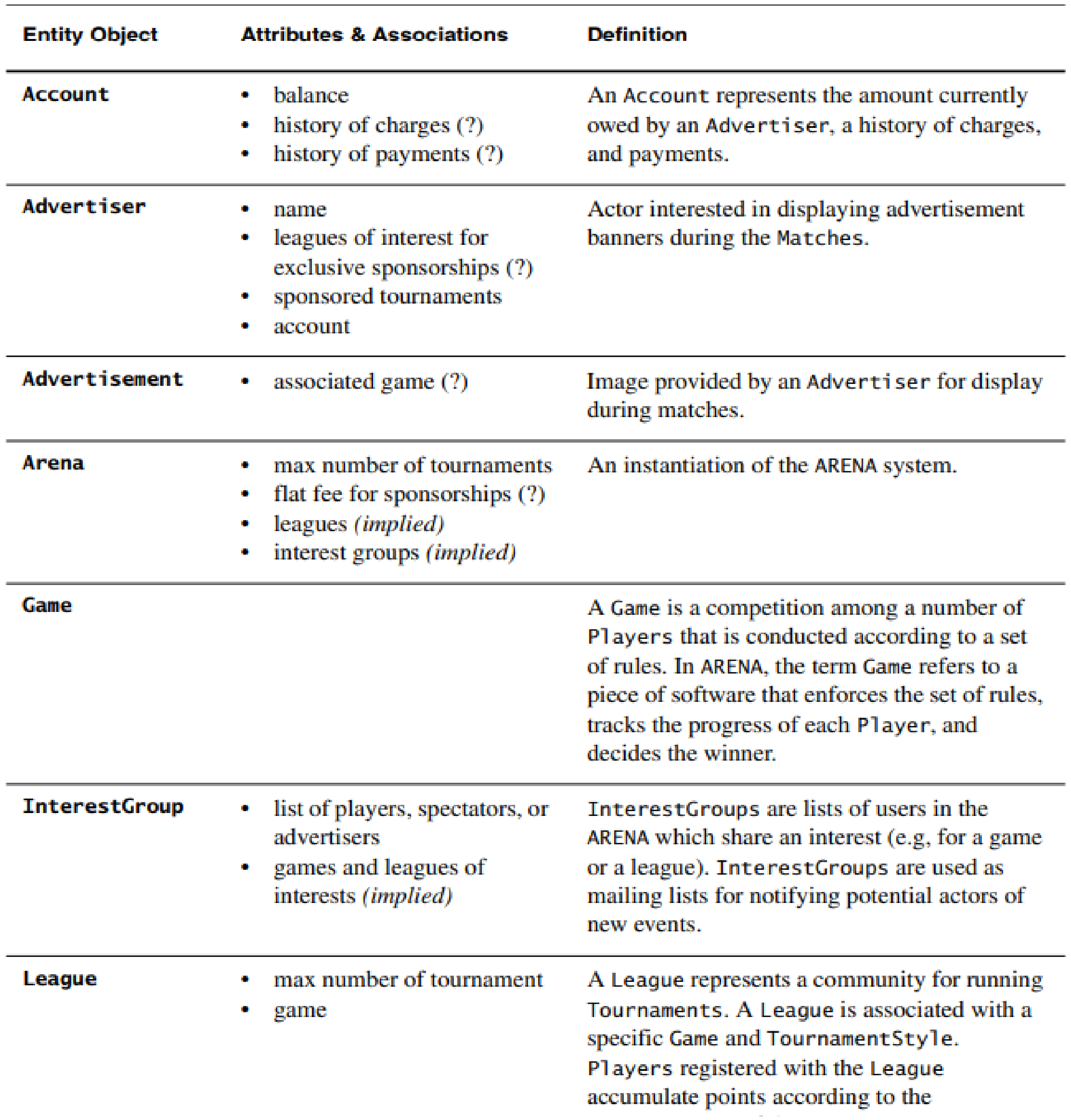
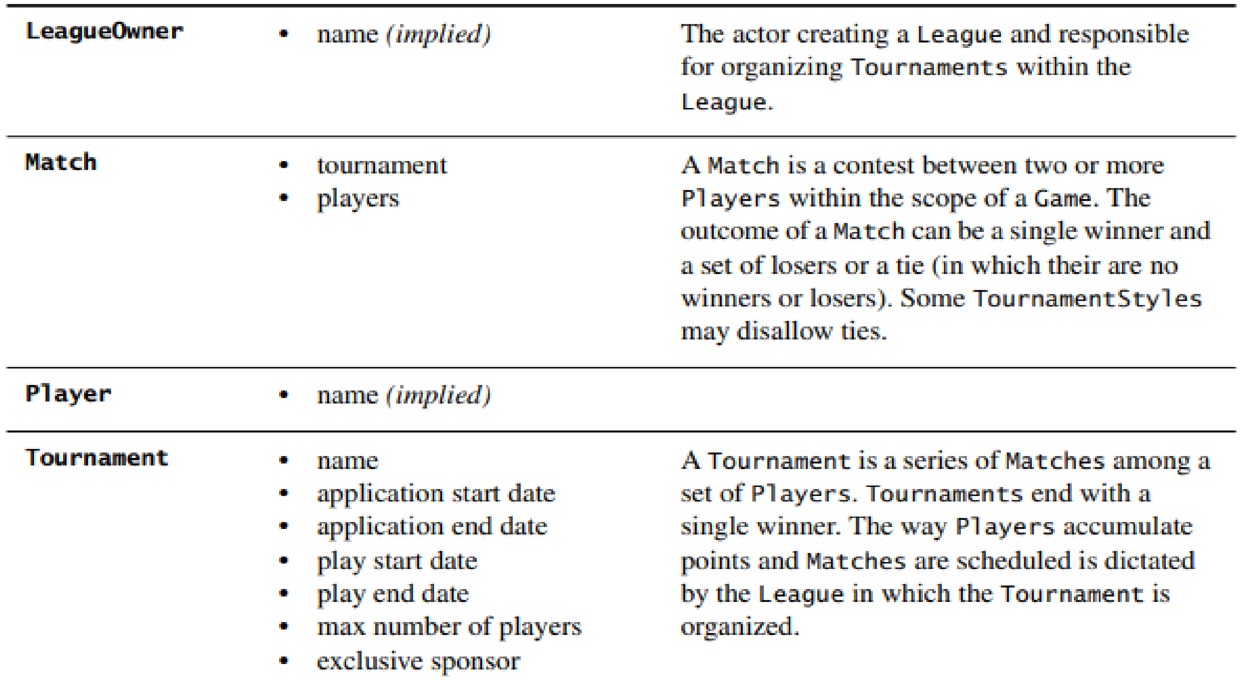
Use case



Name	AnnounceTournament
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. 12. The LeagueOwner selects which groups to notify. 13. The system creates a home page in the arena for the tournament. This page is used as an entry point to the tournament (e.g., to provide interested Players with a form to apply for the tournament, and to interest Spectators into watching **matches**). 14. At the application start date, the system notifies each interested user by sending them a link to the main tournament page. The Players can then apply for the tournament with the ApplyForTournament use case until the application end date.

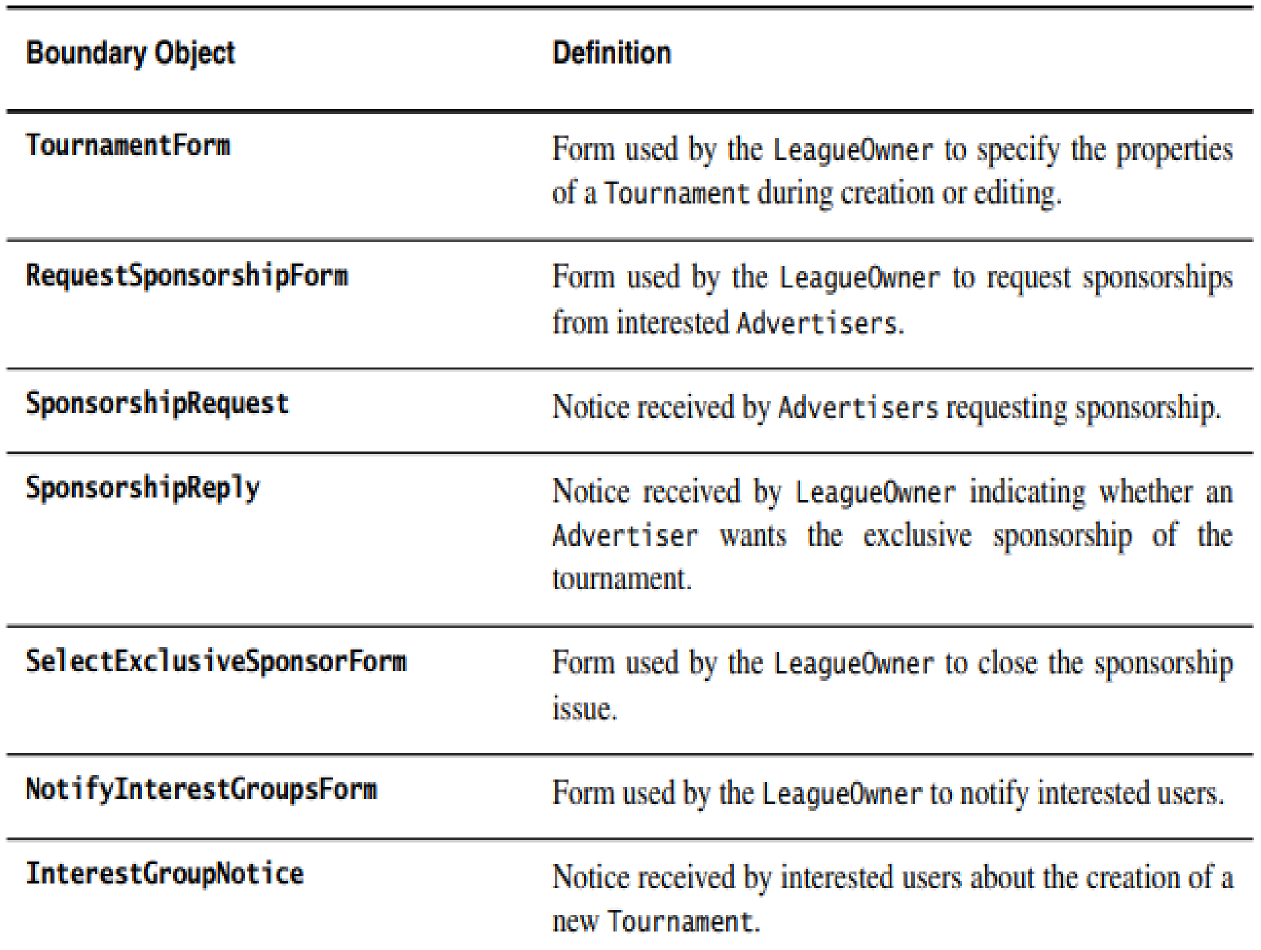
Software engineering

ARENA Case Study

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
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.

Analysis – ARENA Case Study

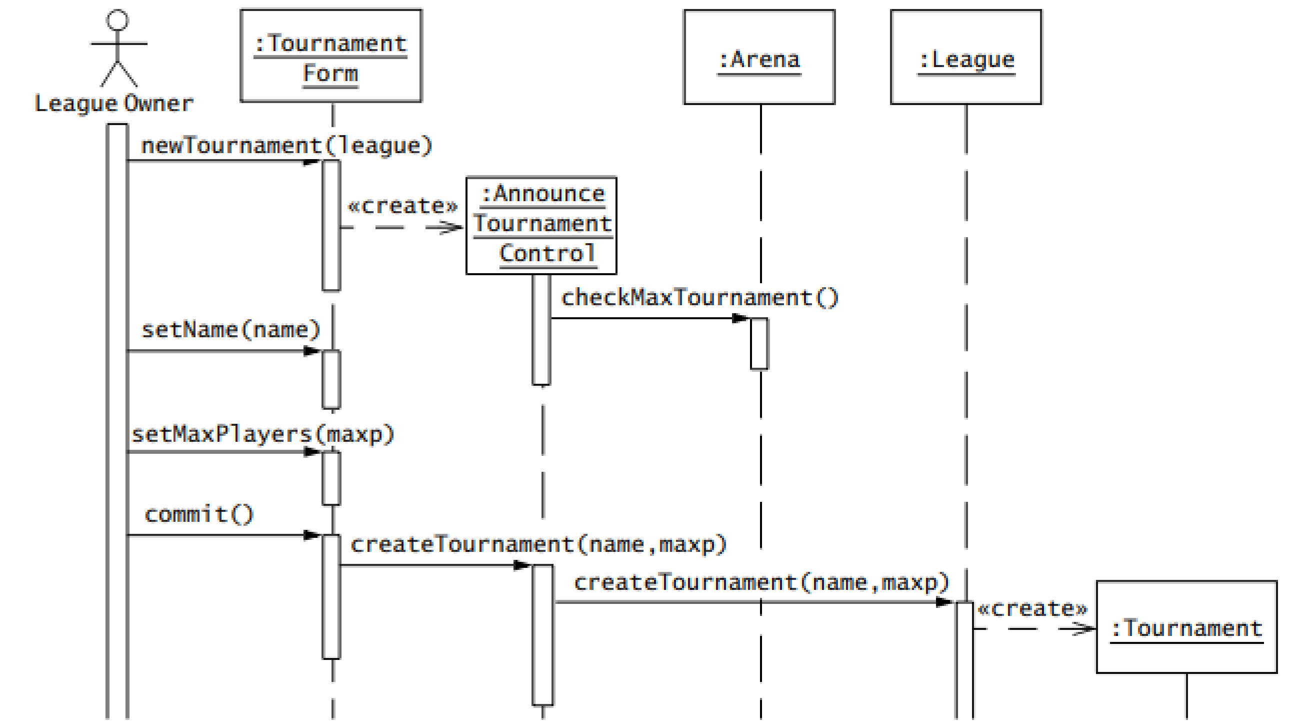


Boundary Object	Definition
TournamentForm	Form used by the LeagueOwner to specify the properties of a Tournament during creation or editing.
RequestSponsorshipForm	Form used by the LeagueOwner to request sponsorships from interested Advertisers.
SponsorshipRequest	Notice received by Advertisers requesting 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.

Control Objects	
AnnounceTournamentControl	Responsible for sending and collecting notices to Advertisers, checking resource availability, and, finally, notifying interested users

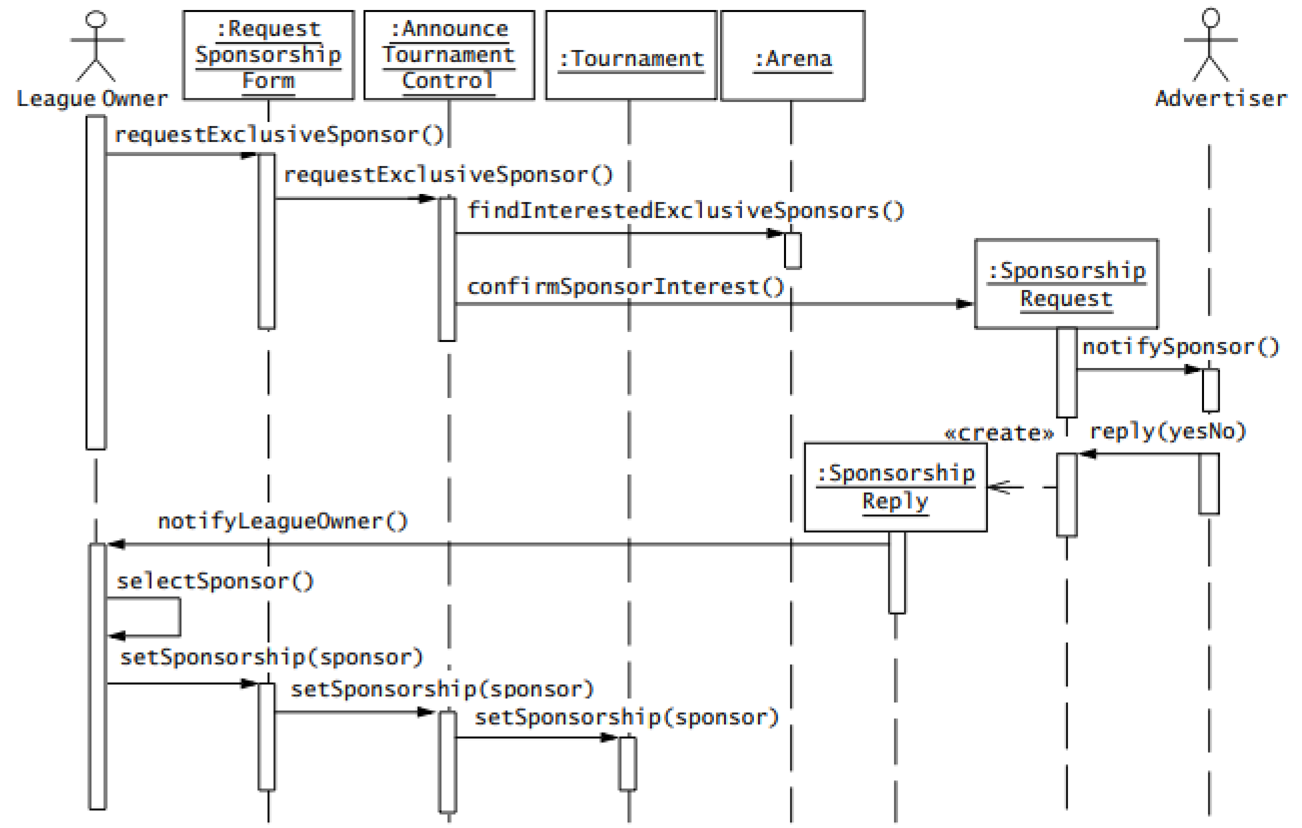
Analysis – ARENA Case Study

UML sequence diagram for AnnounceTournament, tournament creation workflow.



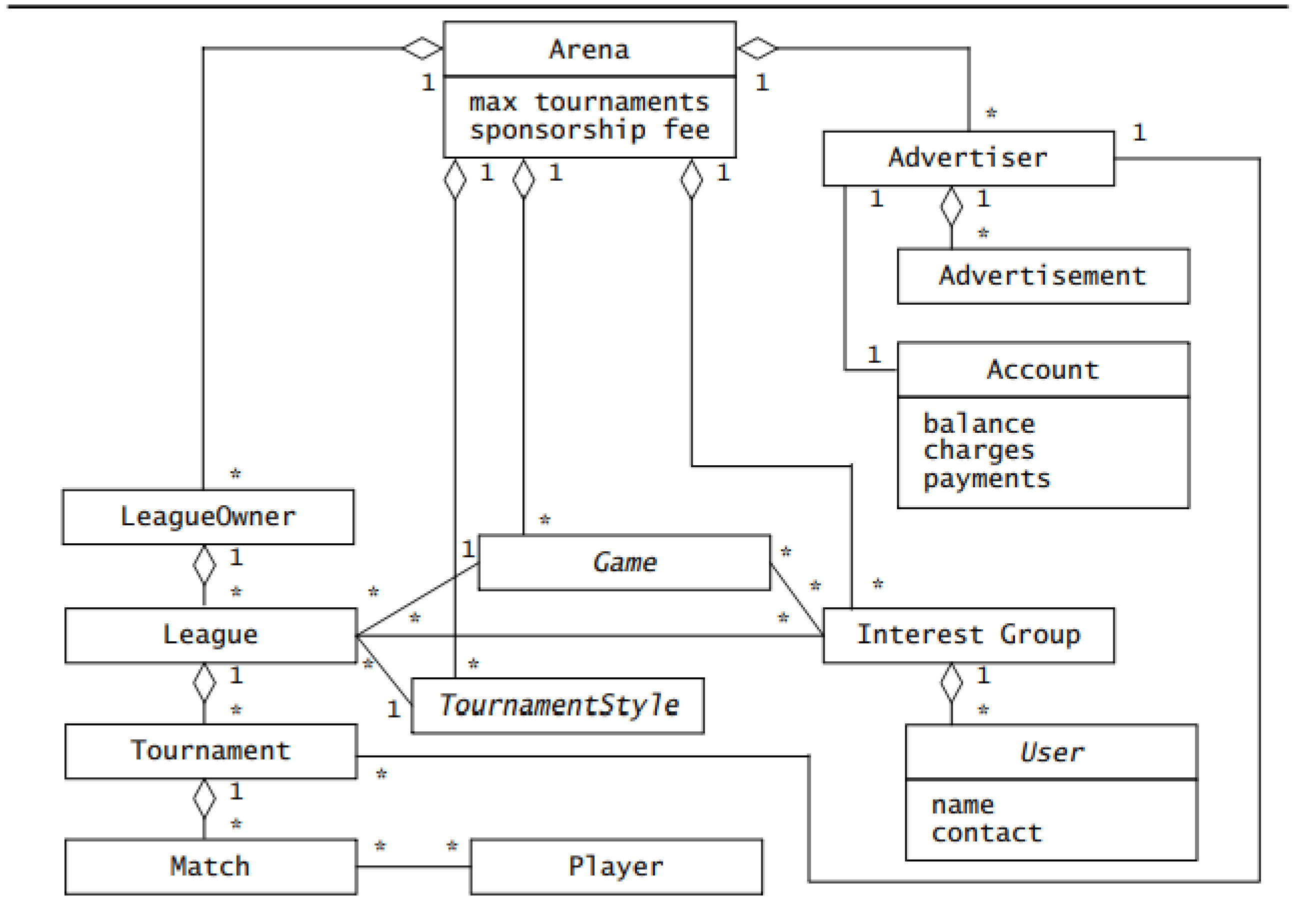
Analysis – ARENA Case Study

UML sequence diagram for AnnounceTournament use case, sponsorship workflow.



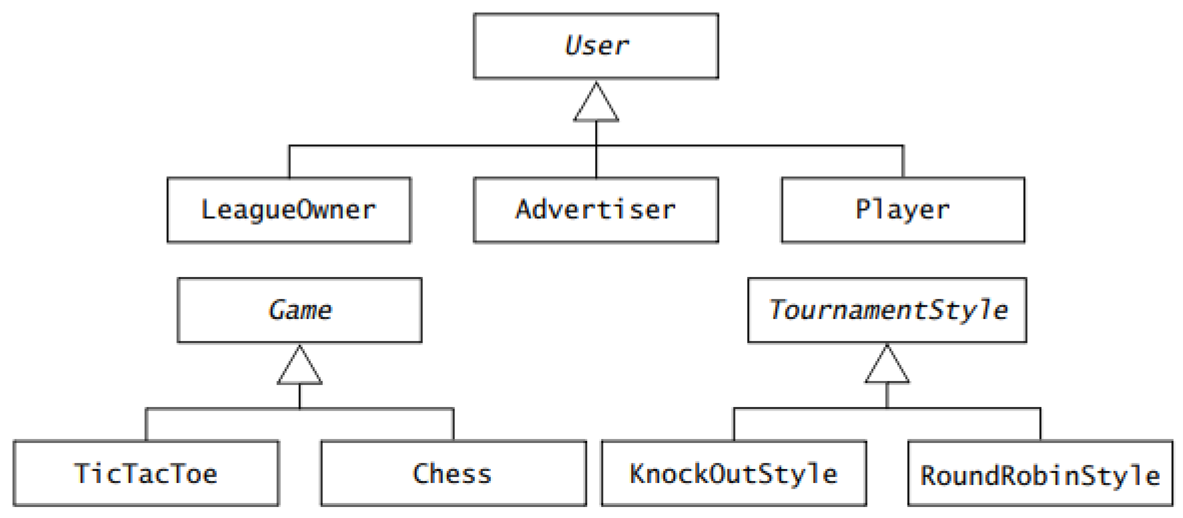
Analysis – ARENA Case Study

Entity objects identified after analyzing the AnnounceTournament use case.



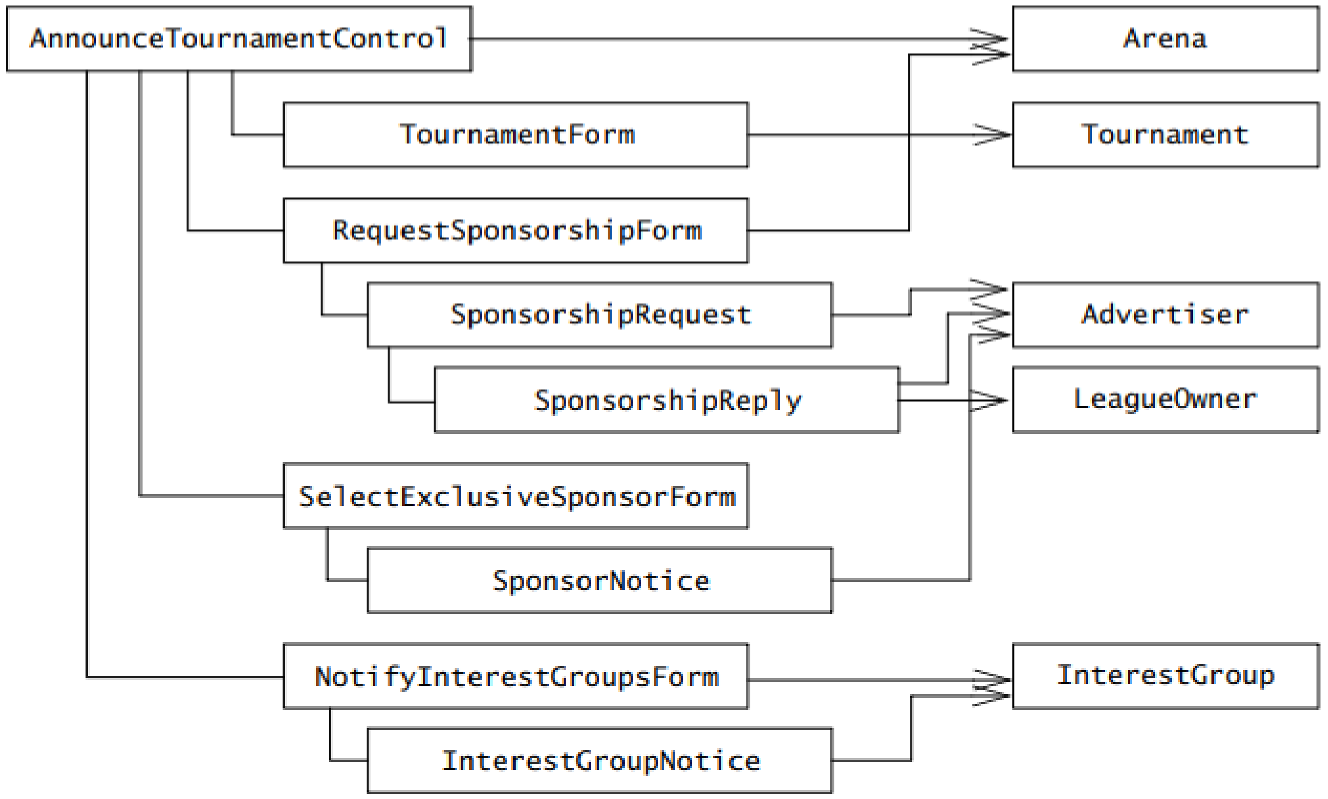
Analysis – ARENA Case Study

Inheritance hierarchy among entity objects of the AnnounceTournament use case.



Analysis – ARENA Case Study

Associations among boundary, control, and selected entity objects participating in the AnnounceTournament use case.



References:
Bernd Bruegge & Allen H. Dutoit – Object-Oriented Software Engineering Using UML, Patterns, and Java – Third Edition – Prentice Hall 2010