Lab sessions group 1. Analysis tasks

JOY IKE
SUNDUS ISHAQUE

1. Introduction

In this session we are going to put in practice relevant object-oriented analysis tasks, namely: identification of user-level use cases, development of use cases, list of features, and domain model building.

Problem: a client wants you to build a software platform that will allow its subscribers to remotely play chess games.

2. Exercise 1

Prepare a list of questions that as an analyst you would make to the client. This list must be primarily focussed at discover functional requirements.

- Features that the software system has to have.
- Requirements that the software system has to meet.
- User-level use cases.

Finally, review the user-level use cases and identify the most relevant ones. Order them so that the first one shall be the first use case that you will completely develop, and so on.

List of use cases:

1. Play game
2. Register subscribers
3. Assign subscribers to games
4. Create games
5. Replay games
6. Close / End games
7. Check valid movements of each piece
8. Show notifications of each movement to user
9. Update the position of chess pieces

3. Exercise 2

Build the first user-level use case following the complete-formal format explained in class.

Analysis: Use case components

Use case Number:	1
Use case Name:	Play games
Goal in context (OPTIONAL):	Allows two players to participate in a game
Actors and their interests:	Players (interested in using the system for playing)
Preconditions:	○ Two players subscribed, correctly logged in, and having agreed to play a game. ○ System properly initialized ○ System has selected which player will start first.
Postconditions:	○ Game ended and stored
Main Success Scenario (Basic Flow)	1. System notifies which player to start (gives turn to one player) 2. System requests movement to the player 3. Player that has the turn proposes a movement of one of his pieces from one square of the board to another square of the board. 4. System executes movement in the trace. 5. System records movement in the trace. 6. System notifies movement to both players 7. System gives turn to the other player. 8. Repeat steps 2 to 7 while there is no winner. 9. Notify winner to players 10. System saves trace of game.

Extension (Alternative Flow)	3a. Player proposes suspending the game. 1. System notifies other player of the proposal to suspend game. 2. System saves trace of game
	3b. Player proposes to resign from the game 1. System notifies other player of proposal to end game. 2. System requests confirmation to end game. 3. Player provides confirmation to end game. 4. System notifies both players of the end of the game. 5. System closes initialized game.
	3c. Player proposes a draw (a tie) 1. System notifies other player of proposal to declare a draw. 2. System requests confirmation to declare a draw from other player 3. Player gives confirmation to agree to a draw. (other alternative flow exists here if player does not agree to a draw) 4. System notifies players of 'draw' outcome. 5. System records outcome in the trace. 6. System closes initialized game.
	4a. System notifies an error in proposed movement. 1. System requests new movement from the player. 2. Player proposes another movement. 3. System continues with step 4 of the basic flow.