SOFTWARE REQUIREMENTS

Huarong Path System

Group X
Author

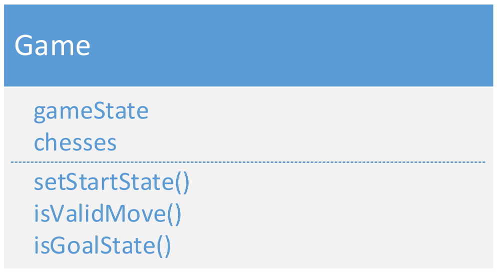
Table of Contents

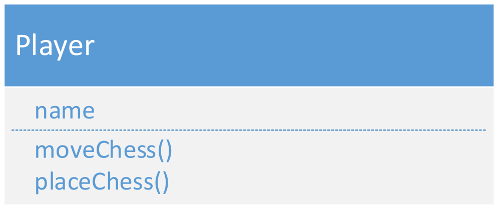
System Objective

In this project, we are developing a software that simulates the classical game Huarong Path. The software will take care of the graphic interface, ensure valid movements and determine whether the goal is achieved, which offers the users a chance to play Huarong Path on computer. Also a UPPAAL model will be provided to check whether a game state has a solution or not, and if it has, it will provide the solution.

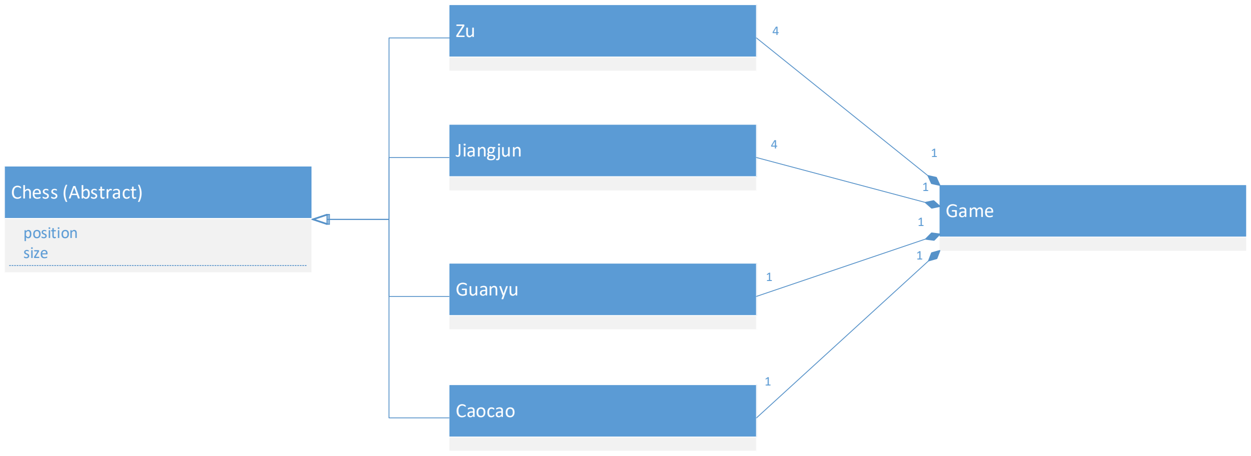
Domain Analysis

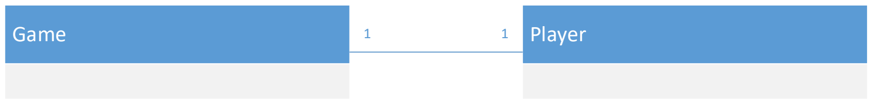
The participants of activities of the game can be categorized into the player and the game itself.



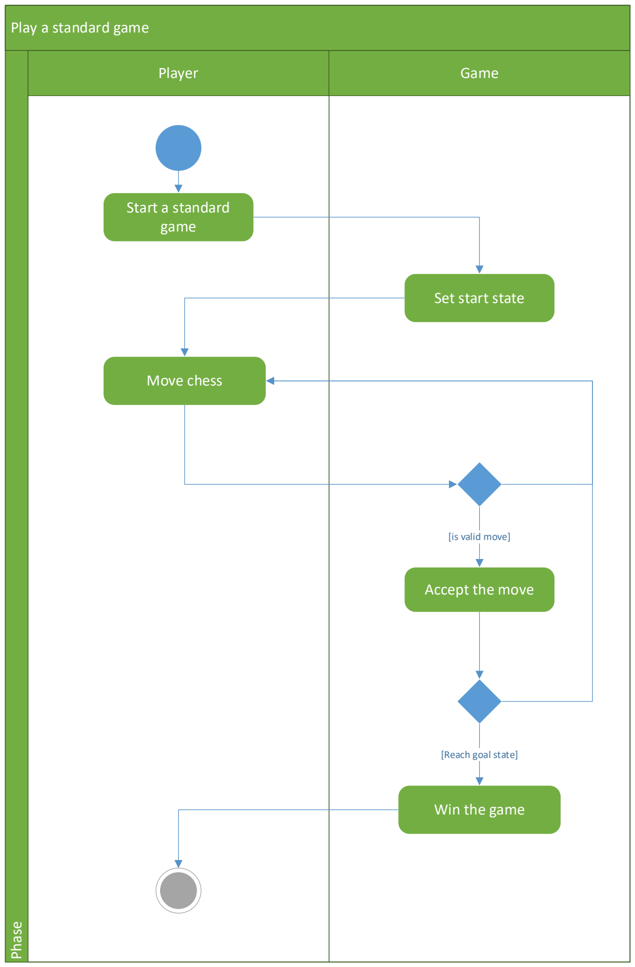


The relationships among different participants are shown as follows:

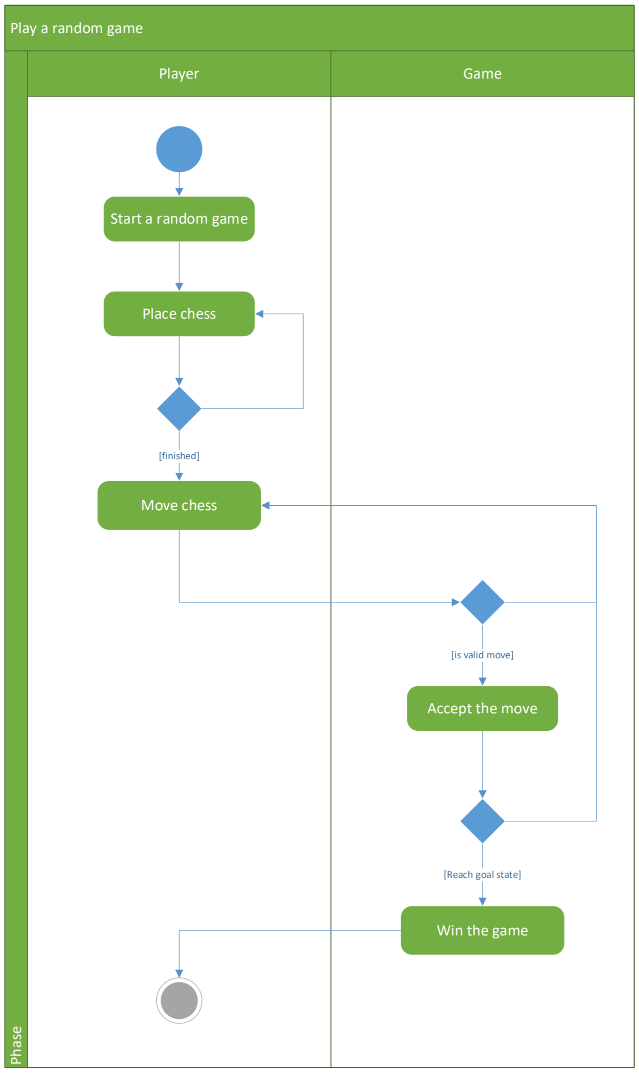




Here is the sequence of events for playing a standard game:



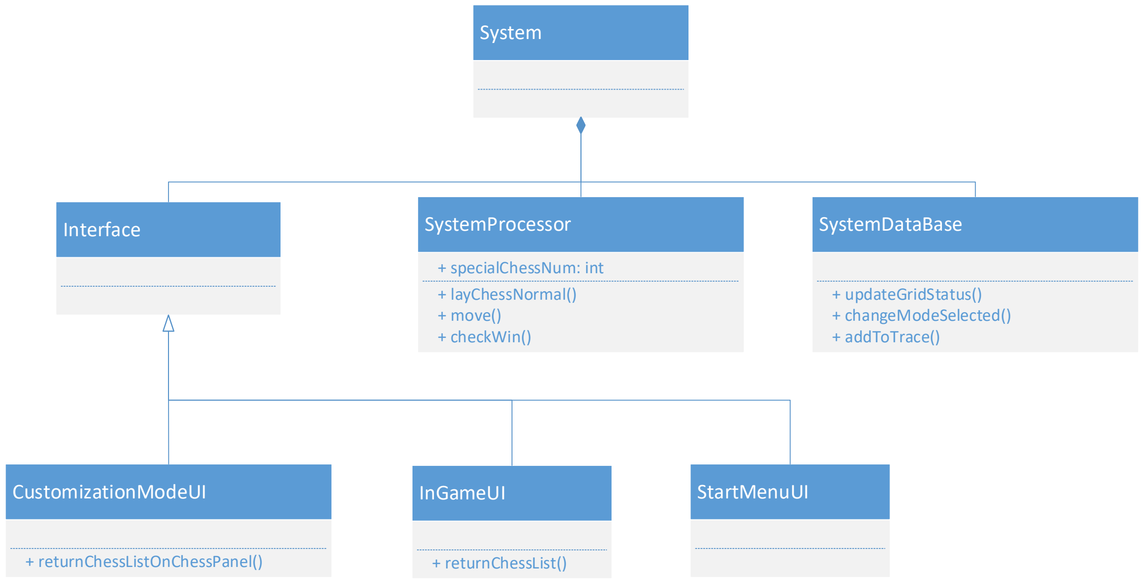
Here is the sequence of events for playing a random game:



System Architecture

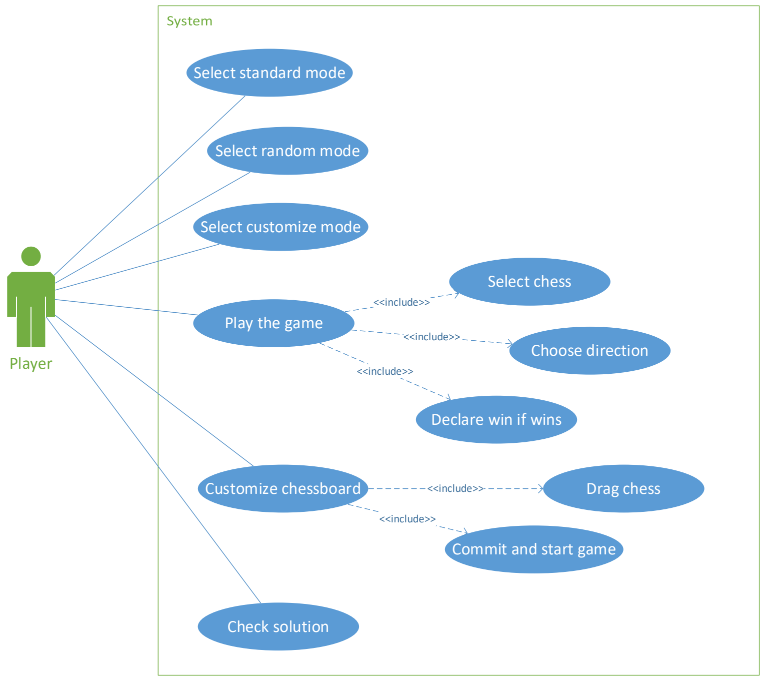
From the information above, we will design a software system that simulates the game process: prepare the chessboard, allow player to move the chess, decide whether it reaches the goal state. It also allows the player to place the chess in random mode.

The system architecture is shown below:



Use Cases

The system can achieve the following use cases from the player’s perspectives:



Software Requirements

R1: GameUI

- R1.1: The player should be able to select the game mode and start the game.
    - R1.1.1: The player should be able to select the standard modes.
    - R1.1.2: The player should be able to select the random mode.
    - R1.1.3: The player should be able to select the customize mode.
- R1.2: If the player selects a standard mode, he should get a fixed start state.
- R1.3: If the player selects the random mode, he should get different start state in different trials.
- R1.4: If the player selects the customize mode, he should be able to start the game.
    - R1.4.1: The player should be able to place the chess to the chessboard.
    - R1.4.2: The player should be able to commit the chessboard.
- R1.5: The player should be able to play the game.
    - R1.5.1: The player should be able to perform a valid move.
    - R1.5.2: The player should not be able to perform an invalid move.
- R1.6: When game ends, the winning message should be displayed.

R2: Controller

- R2.1: If the player selects the customize mode, the placement should be correct.
    - R2.1.1: The player should be able to place the chess to the valid place.
    - R2.1.2: The player should not be able to place the chess to invalid places.
    - R2.1.3: If a certain kind of chess number has reached its limit (i.e. only 5 1*2 or 2*1 chess are allowed, 1 Caocao is allowed, 4 soldiers are allowed), no more chess should be able to put on the chessboard.
    - R2.1.4: The player should be able to commit the chessboard if all chesses are placed correctly on the chessboard.
    - R2.1.5: The player should not be able to commit the chessboard if not all chesses are placed correctly on the chessboard.
- R2.2: The game should run correctly.
    - R2.2.1: The player should be able to activate any chess at any time.
    - R2.2.2: If a chess is activated, the possible directions to move this chess should be displayed.
    - R2.2.3: There will be at most one activated chess on the chessboard.
    - R2.2.4: The directions offered to the player should always be valid.
    - R2.2.5: The chess will move correctly after choosing the direction.
    - R2.2.6: The game should end if and only if the chess Caocao gets to the target place.
    - R2.2.7: The player should be able to undo his moves.

R3: Chess

- R3.1: The chess should be in valid places.
    - R3.1.1: The chess should always be in the chessboard.
    - R3.1.2: The chess should not be overlapped with other chess at any time.

R4: Validate

- R4.1: The system should be able to validate any game start state.
    - R4.1.1: For any given game start state, the system should be able to tell whether it has a solution or not.

- R4.1.2: For any given game start state with a possible solution, the system should be able to provide a valid solution for this state.