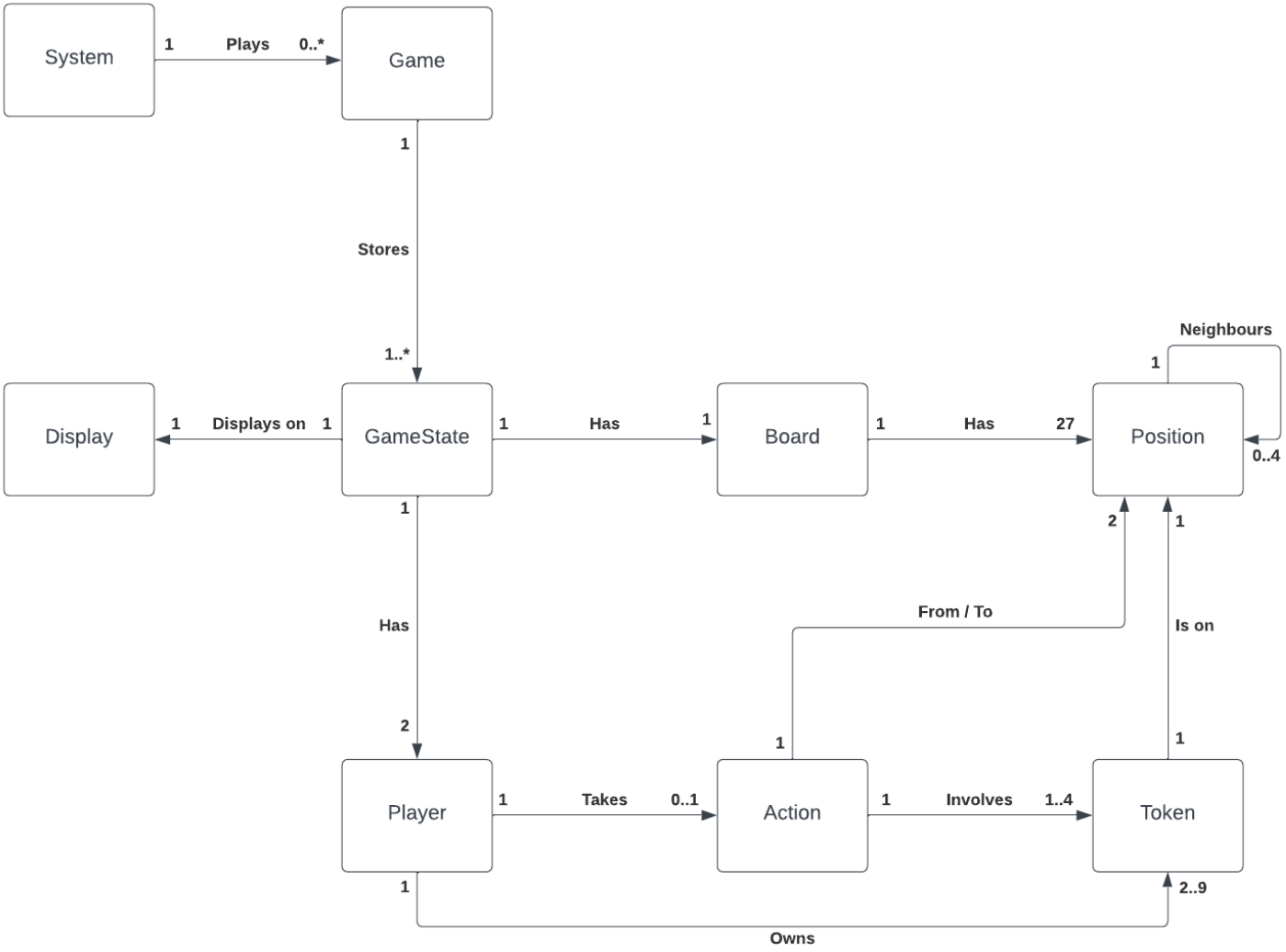
Architecture

Domain Model



Domain Model Rationale

System

This entity represents the entire system of the game, containing every game played.

Its relationship to Game represents how the system can contain 0 to many games (eg. Game 1, Game 2, Game 3 etc.).

We did not include this entity in our original domain model as we did not need to store multiple games. But after including Advanced Requirement B, “able to fully reload any previously saved game(s)”, into the domain model, we would need to be able to store multiple games, hence introducing System entity which stores all games.

Game

This entity represents a single game that can be played on the system.

Its relationship to GameState shows that each individual game can store 1 to many game states. For example, the initial state of the game right after the game has started (before any tokens have been placed or moved) will be stored as the first game state of that game.

We originally had another entity called GameStack, which stores each GameState. However, we realised that it would be redundant as we could just have each Game stores every GameState.

Game State

This entity represents the different states, or turns, that a game can have or go through.

Adding this entity was a design decision made halfway through the modelling process, as we realised that in order to cover Advanced Requirement B, “should support the undoing of moves until there are no previous moves available”, we would need to store each state/turn of the game.

GameState has a 1-to-1 relationship with Display, meaning that each turn of a game can have 1 display, which shows what the board looks like at this specific state/turn of the game.

GameState also has a 1-to-1 relationship with Board, as each turn of the game can only have one board that looks a specific way.

The relation to Player represents how each GameState will always have 2 players. For example, there are always 2 players at each state/turn of the game.

There was an idea to change the relationship between GameState and Player to 1-to-1 as only 1 player makes a move during a turn. However, the other player still exists with active tokens during the turn. As such, this relationship remained unchanged as 1-to-2.

Display

This entity represents how the game looks at a specific state/turn, which is essentially what the players see on their screens when playing the game.

Board

This entity represents the setup of the game, including the nodes/points on which an active token can be placed during the game and the spots an inactive token, either unplaced (in the “dog house” or “cat house”) or “dead” (in the “grave”), can be placed.

It has a 1-to-27 relationship to Position, as each board contains 24 nodes/points and 3 other special positions for the “cat house”, “dog house” and the “grave” (see more below in the *Position* section).

Initially we thought that there would only be 24 positions as each board only has 24 nodes/points that a token can be on. However, this would cause some discrepancies in its relationship with Action, as it would mean players would be moving tokens from nowhere/no position to a node/point on the board when placing their tokens, or from a node/point on the board to nowhere/no position when removing their opponents’ tokens. Therefore, the 3 special positions, “cat house”, “dog house” and “grave”, were added. Adding these 3 extra positions would also make it consistent with the physical game, where you would move tokens from the “house” when placing them and to the “grave” pile when removed.

Position

This entity represents each spot that a token can be placed on. There are 4 types of position that a token can be on throughout the whole game:

1. A node/point on which an active token is placed
2. The “dog house” where all unplaced dog tokens are stored
3. The “cat house” where all unplaced cat tokens are stored
4. The “grave” where all “dead” tokens removed from the game are stored

Its relationship with itself denotes that each position can neighbour 0 to 4 other positions. If the position is for the “cat house”, “dog house” or the “grave”, it means that the position will neighbour 0 position. If the position is one of nodes/points in the actual game, then it can neighbour 2 to 4 other positions.

Player

This entity represents the players that are involved in the game. There are 2 players in each game.

It has a 1-to-2 relationship to Action, as each player can take 0 to 1 action during each turn and each action can be performed by only 1 player. If a player takes 1 action at a turn, it means that this turn is theirs to move their tokens; if a player takes 0 action at a turn, it means that this turn is their opponent’s turn and hence they cannot move their tokens.

Player also has a relationship to Token that signifies each player can have 2 to 9 tokens at any state/turn of the game. This is because each player has 9 tokens to start with (maximum), and the game will end as soon as one player has only 2 tokens left (minimum).

Action

This entity represents the movement that a player can perform on a token.

While modelling the domain, it was determined that moving a mill to a new position counts as 1 action, and if this action results in forming a mill, the removal of one of the opponent’s token is counted as a separate action. The reason behind this decision was to accommodate for Advanced Requirement B, “should support the undoing of moves until there are no

previous moves available". Making the removal of one of the opponent's tokens as its individual action means that players can undo only this action after forming a mill and choose to remove another of the opponent's token, without undoing the mill formed.

Action has a relation with Token where each action can involve 1 to 4 tokens.

If the action performed was removing one of the opponent's token, this action only involves 1 token (ie. the token that was removed).

With each token being moved to an adjacent position, a check would be required to see whether a mill has been formed. This check can involve up to 3 other tokens, as there are 3 other adjacent positions (excluding the one that the token just moved from) and we would only need to check the positions "in-line" with the previous position. Therefore, up to 4 tokens are involved (ie. the moved token itself and up to 3 other tokens).

An action should know the position from and to which a token has been moved, therefore, Action entity has a 1-to-2 relationship with Position entity.

Token

This entity represents the tokens that the players have on the board in a game.

It has a 1-to-1 relationship to Position as each token knows about its position in the game. For example, token A should know that it is on position/point 2.

We initially thought that Position would store a token. However, when an action is performed, it would be easier to move the involved token itself and then deciding where it currently is and where it can move to, rather than figuring out if a position has a token and if it belongs to the current player before making a move.