Rationales for Domain Entities and Relationships

Defining the Domain Entities

Playing a game of Nine Men's Morris requires a board, two players, and some tokens. Therefore, board, player, and token are fundamental domain entities. The game itself could also be a domain entity. As part of one of the chosen advanced requirements, a player should be able to save and load a game. This reinforces that the game should have its own entity, and it has been included. Another part of the same advanced requirement is that the player should be able to undo moves, so a move could be another domain entity.

In the basic gameplay, the players alternate making moves. At the start of the game, the players can place tokens on empty board locations. So location is another entity. The game progresses by players forming mills with their tokens, so a mill is another entity.

If a player gets a mill then they can remove an opponent's token which isn't part of a mill. This means that on a player's single move they can move (or place) one of their own tokens and then potentially remove an opponent's token as well. These smaller 'moves' can be represented by an 'action' entity, which represents something that a player does. Since a move now contains many actions, the undo operation can either undo single moves or single actions. The undoing of single actions may be more useful to the players, as this will allow choosing a different token to remove from the opponent without having to remake the mill which allowed this. However, both features could be included.

Another option is that players take actions instead of moves, and the move entity is removed. This offers less flexibility since only individual actions can be undone instead of moves. This also doesn't show that a turn could contain many actions. On the plus side it is simpler for the player as they only take actions and don't worry about moves and actions. If only actions are done and undone then a move only serves to group actions. However, an important part of the problem description is that the players alternate moves, and modelling only actions doesn't describe this. So, this alternative won't be taken and a move will still be modelled in order to provide a clearer description of the gameplay.

Since an action models something that a player does, the loading and saving of games and undoing of actions on the board could also be modelled as actions themselves. A distinction needs to be made between actions which affect the board and can be undone, such as moving a token, and actions outside the board which cannot typically be undone, such as saving the game. The idea of undoing moves in the real world usually only applies to moves on the board as the token arrangement can be reversed. If a player takes an action outside the board such as saving a game, they can't undo it and instead have to take an additional action to discard the save data. A 'board action' will be introduced to represent actions which modify the token arrangement on the board and can be undone.

Another important specialisation of the action entity is the action of setting up a new game. To play a new game in the real world the board must be laid out and the players must be assigned tokens. This should also be modelled as a ‘setup action’ since this is something that a player must do to play the game.

Within the board, a player may place a token, slide a token, jump a token, or remove an opponent’s token. All of these are specialisations of the board action since they affect the token arrangement on the board and can be undone.

As part of another advanced requirement, the player should be able to play the game against the computer. Although the ‘bot’ will imitate the player entity, it will be modelled as another entity to clearly show that it is a computer and not a standard human player. To further clarify that the bot and human player are distinct, a ‘person’ entity is added which is another specialisation of the player. Now the player is just some ‘entity’ that plays the game, and it could be a computer or a person.

There is also the option to include entities describing the game state and history of states. Although this seems like it would be useful in describing how the actions affect the game, it is not part of the problem description and isn’t required to model the gameplay. It is more an implementation description and shouldn’t be part of the domain model. To play the game, the players use the board and tokens and take actions which affect them directly. An intermediate state or history only convolutes the diagram and doesn’t really help to understand the game. Therefore, a state or history entity won’t be included.

Another alternative with the tokens is to include a token bank entity to model the tokens which a player has but aren’t yet on the board. Since the player interacts directly with the tokens in the problem description, this intermediate entity isn’t needed and makes the model more complex. A simpler way to model these tokens is to say that a token need not be on a location. This is why the token bank entity won’t be included in the domain model.

Defining the Domain Relationships

To play a game requires one board, two players, and eighteen tokens, so appropriate relationships are formed with these entities. These relationships are aggregations since the game can’t exist without these entities, but the entities can exist without being in a game.

A board contains multiple locations through composition, since a location can’t exist outside the board. Adjacent locations are also related to each other. The corner locations have two neighbours, the edge locations have three neighbours, and centre locations have four neighbours. The locations may have a token on them, and also the tokens may not be on a location as mentioned before. An association will be used instead of aggregation since the tokens aren’t part of the location.

The players have a variable number of tokens depending on how many have been removed by the opponent. The tokens also aren’t part of the player, so an association is used instead of aggregation.

A mill could contain tokens through aggregation or a token could be part of a mill through association. The aggregation will be used to clearly show that a mill can’t exist without the correct token arrangement.

A move contains one or many actions, and the actions make up a move, so an aggregation will be used instead of an associated. Aggregation is used instead of composition as this allows the possibility of players taking actions outside their turn. For example, if a player is able to save the game when it’s the opponent’s turn, then they can take an action outside of their own move. This is an assumption which increases the flexibility of the gameplay. The associations between player and action and player and move are also shown. Also note that a move may have one action in the usual case but a maximum of three actions if two mills are formed in one go and two tokens can be removed from the opponent.

The specific actions a player may take are specialisations of the action entity, but the undoable actions are further specialisations of the board action entity. The person and bot are also specialisations of the player.

The board actions operate on and modify the board, which will be shown by association. All board actions move a token in some way, so they also need to move a token. This will be shown with an association from the generalised board action instead of multiple associations from each specific action. Any board actions added in the future should also move at least one token, since by definition a board action modifies the token arrangement. Currently only one token is moved by a single board action, but this could be changed later if new actions are introduced which move multiple tokens at a time. Zero-to-many is the multiplicity on the board action side in both relationships since one board or token may be adjusted by many successive board actions.

The save and load actions will capture and restore the game and all its encapsulated information for future use. The undo action will need to know about the game and its last move or action in order to undo them. The setup action will also need to set up a particular game and will need to have a relationship with it. Since all these actions require a relationship with the game, the base action itself can be associated with a game. This makes sense since all actions occur in the context of a single game. This doesn’t affect the board actions since they also occur in the context of a game. Therefore, the base action will be associated with a particular game.

In order to play the game, the players need to view the state of the game in order to strategize and decide what action to take next. They need to look at the board, see the token arrangement, and know how many tokens the opponent has left. All of this information is encapsulated in the game entity. Therefore, the player needs an association with the game in order to view it and take informed actions.

Assumptions

The following assumptions have been made about the gameplay and game rules.

- Saving the game, loading the game, and undoing a move in the game all cannot be undone explicitly.
- Either player may undo the last move regardless of who’s turn it is or who made the last move.
- Either player may save the game, load a saved game, or start a new game regardless of whose turn it is.
- If a player makes two mills in one move, they may remove two of the opponent’s tokens which aren’t in a mill.