In formulating our use cases from the requirements specification, we had to make several design decisions regarding what should constitute a single use case as opposed to providing more separation with an ‘include’ relationship. Most of the use cases describing actions taken (or events occurring) within the game were clear candidates for ‘extend’ relationships from a main ‘Move’ use case, as shown in our use case diagram.

We felt that steps taken to begin the game, including the selection of a ‘Start Game’ option from the main menu, and the selection of number of players and input of player names, should constitute a single ‘Initialise Game’ use case. While this necessitates several alternative flows to describe alternative Main Menu choices, as well as incorrect user input, none of these alternative events were of sufficient substance to necessitate an extra use case.

As ‘Move’ will be performed in addition to, but not necessarily part of, the ‘Initialise Game’ use case, we modelled it as an ‘extend’ use case of “Initialise Game’. In formulating the flow of events for the ‘Move’ use case, we felt that ‘Move’ should feature the full sequence of events that constitute non-optional parts of a player’s turn (mainly rolling dice, updating player location, checking for bankruptcy, and displaying menu after location update), with ‘extend’ use cases called within the main flow for the scenarios that may occur during this sequence of events. Alternative flows facilitate the options available to the player before they end their turn. Simple ‘display’ options can be easily described without an extra use case, while more complex actions such as ‘Trade Element’ and ‘Develop Element’ necessitate ‘extend’ use cases.