1. Software Architecture:

1.1. *Abstract Representations of Software Structure*

Once the requirements were gathered, conceptual models of the software architecture were developed. These models outline the software's structure at a high level, as well as outline the general flow of gameplay and key gameplay features, which will then be used when implementing the system in Unity.

The following models were designed using UML 2, with the tool ArgoUML.

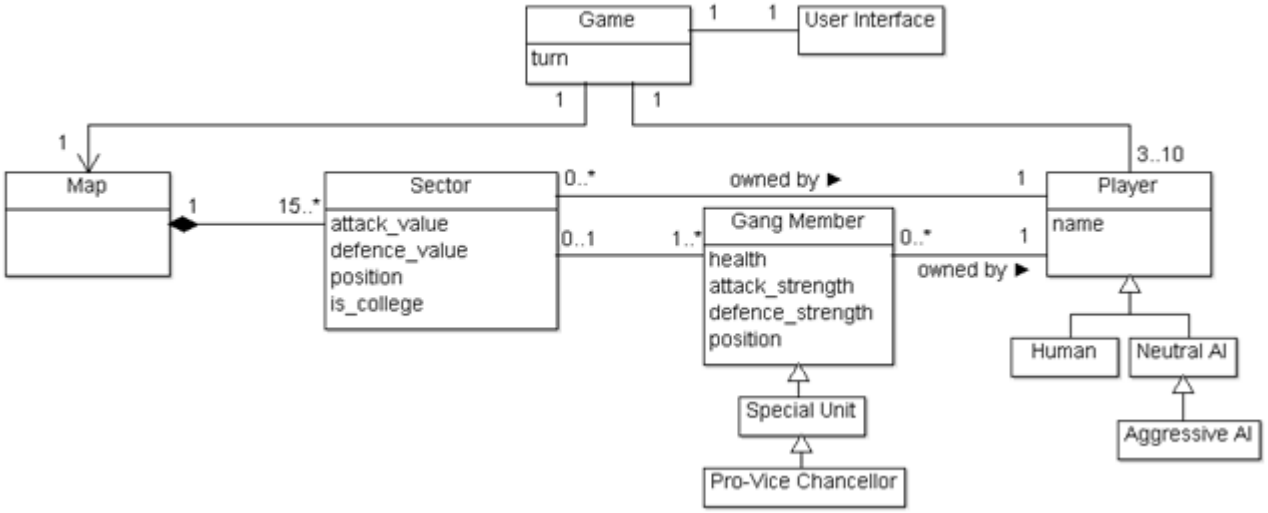

Figure 1. Abstract UML class diagram describing the game's main structure.

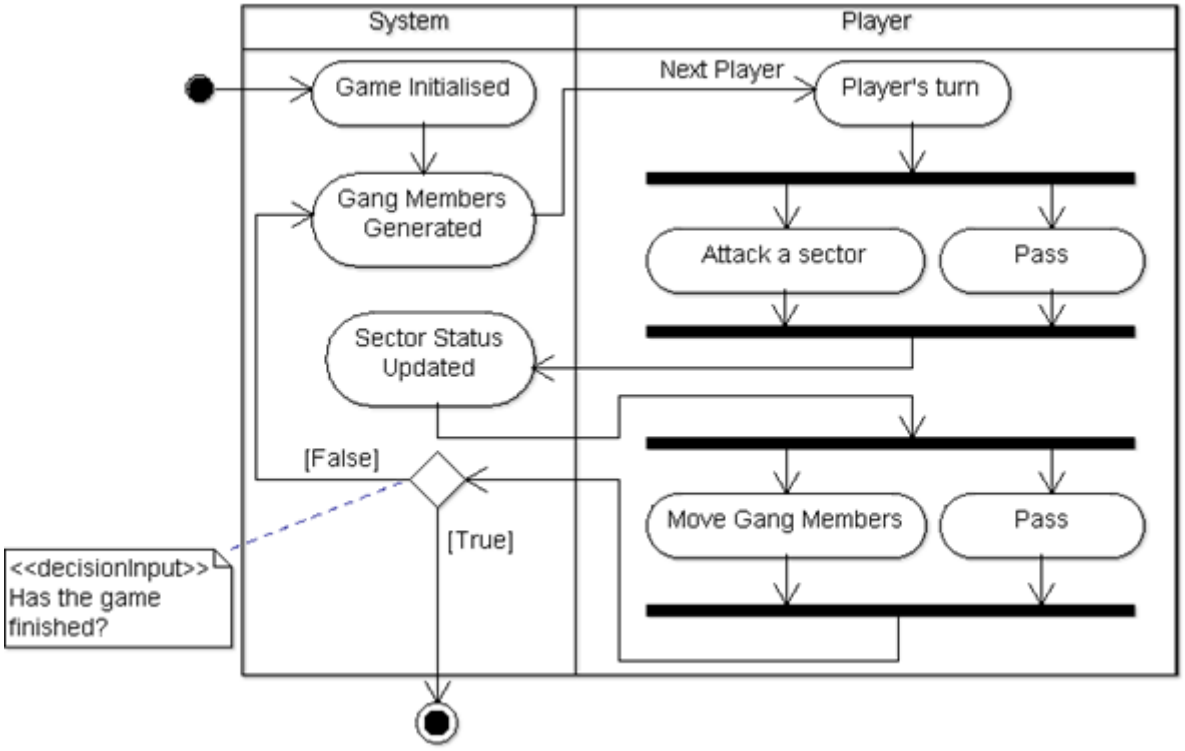

Figure 2. A UML activity diagram representing a generalised outline of the gameplay. The diagram is separated into the tasks carried out by the system and the players.

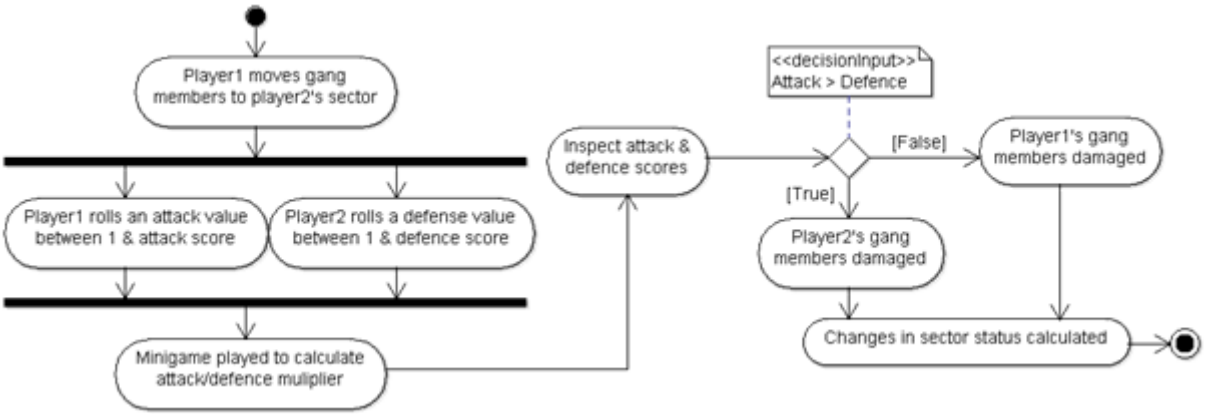

Figure 3. An activity diagram showing the general flow of an attack sequence between two players, “Player1” and “Player2”.

1.2. *Architecture Justification*

1.2.1. System Architecture Class Model

The abstract representation for the system architecture is displayed in figure 1. This, along with the other models, will be used to implement to system with Unity, which has been chosen because it is a widely used and supported game engine. Within the model, there is a Game class, which exists as an overhead, accessing the Map and Player classes, as well as handling the overall flow of the game. The Game class has a 1-to-1 association with the User Interface, which will allow the visual aspect of the system to be displayed (REQ-6.1).

The model also includes a Map class. This is present because it can be used to combine all of the sectors into one place (REQ-5.1) and has a composition relation between Map and Sector because of this. On this composition, Map has a multiplicity of 1 because the sectors will be a part of exactly one map. Also, the multiplicity for the sector class is 15..* because the players must start with 5 sectors, and the minimum number of players is three, as stated in REQ-7.1 and REQ-7.6.

The sector class was chosen because REQ-5.1 specifies that the map will be separated into a number of sectors. The class has a position attribute, because it represents a section of the map, as well as an 'is_college' attribute that will outline if the sector is in a college. This attribute is used because capturing a college gives the player a bonus as described in REQ-1.1. The sector also has two other attributes, attack_value and defence_value. These have been added to the system architecture because the conflict system takes into account the attack value of the attacking sector and the defence value of the defending sector (REQ-2.2).

Also, the sector class has an association between the Sector and Player classes, and the ► symbol indicates the direction of the comment; reading, "Zero or more sectors are owned by one player." A multiplicity of 0..* has been used for the Sector class because a Player can own zero sectors, when they have lost, but can also own all sectors in the map. A multiplicity of 1 has been used for the Player because at any one time a sector only has a single owner.

Additionally, there is an association between Sector and Gang Member, which exists because gang members belong to the sector they are standing in and players will be able to move gang members between sectors. The multiplicity for gang member is 1..*, meaning a sector must have at least one gang member in it, and the multiplicity for sector is 0..1. This is because gang members are initially unassigned, i.e. they do not have a sector, and then the player assigns gang members to sectors (REQ-1.2).

Furthermore, the system architecture model also includes a Player class, which will be used to represent the players within the game, and contains three subclasses, Human, Neutral AI and Aggressive AI. The human class will contain the methods used by the users, whereas the Neutral AI class will be used to simulate the passive computer opponent as described in requirements REQ-7.1 and REQ-7.5. Also, the aggressive AI will extend the functionality of the Neutral AI, adding an attacking algorithm alongside the defending function of the neutral AI (REQ-7.5). The Player class also has a name attribute, which has been chosen because it can be used to identify a player within the game.

Also, the model shows an association between Gang Member and Player. This relation is present because players own gang members and can control the gang members they own (REQ-7.3, REQ-7.4). The multiplicity of gang member on this relation is 0..* because within the game players can own any number of gang members, including 0 when they have lost. In addition, the multiplicity on the Player side is 1, since gang members only have a single owner.

Players are also associated with Game and this association has been added to the model because the player's turn and timing of other actions, such as moving gang members or attacking sectors, are dictated by the game. For instance, a player can only move their gang members when it is their turn. Additionally, the Player class has a multiplicity of 3..10 because the minimum number of players is three, as stated in REQ-7.1, and a maximum of 10 was chosen as this will be one player per college, plus the neutral player.

Finally, the model contains a 'Gang Member' class, since there needs to be a way of storing data about gang member's position, health, attack strength and defence strength. The model shows four attributes within the Gang Member class; health represents how alive an individual unit is, which will be reduced during combat and attack_strength/defence_strength represent a unit's attacking/defending power, which will be used to calculate the attack/defence values in the sector class.

Gang Member also has two subclasses, meaning gang members can also be special units, as described in requirement REQ-1.4, and the Pro-Vice Chancellor (PVC) is a type of special unit, which will be used to initiate the PVC minigame and bonus (REQ-8.1, REQ-8.2 and REQ-8.3). The PVC class has been added as a subclass of special unit since it will allow the PVC to easily use the methods for allocating bonuses that will lie within the special unit class.

1.2.2. Gameplay Activity Diagram

After the system architecture was decided, an activity diagram, figure 2, was created. The diagram shows the overall gameplay flow and summaries the general interactions between the system (game) and the players (users). This activity diagram was created because the activities displayed are a key part of the system, and the diagram can be used to easily relay this information to the customer.

To begin with, the game is initialised. This should involve instantiating the system's classes, allocating sectors to players and displaying the graphical user interface. Then the gang members are generated, where players are given a selection of gang members which they can then place at a later stage. These two steps have been chosen to fit in line with requirements REQ-1.1, REQ-6.2 and REQ-7.6, which describe the spawning of gang members, the contents of the graphical user interface and the assignment of sectors.

After players have been assigned gang members, the system prompts the first player to begin their turn. Initially, the player must place their assigned gang members into their sectors (REQ-1.2). This has been placed before the player can attack a sector or move gang members because requirement REQ-1.3 states that "The user should not be able to move on until all units are placed in the user's sectors." The player is then given the chance to either 'Attack a sector' or 'Pass' (REQ-7.4, REQ-7.7), and this has been described as two concurrent activities because both options will be offered to the player at the same time.

Next, the system performs the activity "Sector Status Updated" and causes the system to calculate any changes in sector ownership after an attack. The activity has been placed between the options of attacking and moving gang members so that the player who attacked a sector can then move more gang members into their new sector.

Once a player has completed their turn, the system checks if the game has finished, and if so the activity diagram ends. However, if the game has not finished, new gang members are produced, then the next player begins their turn (REQ-7.2) and the process repeats.

1.2.3. Combat Mechanism Diagram

The combat mechanism the system will provide is outlined in figure 3. Firstly, combat initiates when a player moves a number of gang members into an opponent's sector, as described in requirement REQ-7.4.

Next, both the attacking player and defending player roll a number between zero and their attack/defence scores. This was done to make sure that the combat system is not entirely random (REQ-2.2, REQ-2.3) because a stronger player will have a higher chance of scoring a larger attack/defence score.

However, another layer of complexity was chosen to be added to the combat system to make the mechanism more involved. As a result, a short slider mini game will be added to the system in which the two conflicting players will try to click when a marker passes a target, and the closer the click, the larger the score multiplier (REQ-2.3, REQ-3.2). This mini game is displayed immediately after the initial attack/defence scores are decided and before the two scores are compared. The order of this sequence was decided so that the players can see their initial scores and have the opportunity to decide a tactic for the mini game.

After that, the attack and defence scores are inspected and the player with the lower score loses the combat. The gang members in that sector are then damaged and any changes in sector ownership are calculated.