Peer-Review 1: UML

Martino Piaggi, Amrit Singh, Lorenzo Perini

GroupGC23

Evaluation UML diagram of GC33.

Positives

The GC33 has understood that a MotherNature, ProfessorPawn and Students classes were not necessary at all and correctly did not implement them. They also reduced the number of classes and rendundant methods in different classes by using Tile as the father class of Island, Cloud.

Negatives

We have noticed that methods don’t have any parameters specified. There aren't any rule based methods that could potentially use the project structure: for example, there are the getter/setter methods necessary to move Mother Nature but there isn't a method that receives the request to move Mother Nature from the player, checks if the moving distance requested by the player is legal, etc.

Bag and Schoolboard as specification of Tile are maybe easy to implement but logically aren't a specification of Tile.

We also noticed that genIslands() is inside the island itself, and this seems to be a paradox (to generate an island is it needed an island istance?).

Better data structures could be implemented (using a linked list for the islands): by using a linked list, you could remove from the island class 2 attributes (next and prev) and 4 methods (the getter and setter of the 4 attributes).
Furthermore, links between classes could be reworked slightly to be more efficient; for example, methods such as PlayAssistantCard and deck generators such as genHand and genCharacterCards could be moved to the player itself and two designated deck classes, respectively.

Comparison between architectures

Our solution is more centralized in the Game class, and this gives us more control during the implementation, testing, and debugging of the game. In fact from Game we control every function call that contributes to the game progress and is the contact point with the controller. In the case of GC33 design the functions needed to the progression of the game

are decentralized in all classes; while not an error we think that this could create confusion and many problems during implementation because it will be hard to follow ‘the calls’ and to make sure to not be performing illegal moves.

Without centralizing the game flow into the Game class, it seems that GC33 doesn’t have complete control of the rounds, of the different phases, and it ‘s not clear how they check who is the current player in each turn so that they can easily find the illegal moves from other players.

Also, without centralizing, there are methods like isExpertMode() or getTeams() that we don’t need because the only class that uses that information is the game itself, while all other components of the game simply get the information during the moment of ‘spawning’ through their constructor methods.

We use a JSON file to load the powers and images of the character cards, which (in our design) has subclasses for the ‘macro types’ of the cards. We notice that in the design of GC33, the architecture is different and they are not using subclasses at the moment.