Peer-Review 1: UML

<Finazzi Alessandro>, <Gallazzini Andrea>, <Lancini Andrea>

Gruppo <48>

Valutazione del diagramma UML delle classi del gruppo <47>.

Lati positivi

- Buona impostazione delle classi per la gestione del gioco nelle sue diverse modalità, ovvero l'estensione per la variante Esperto, e per la modalità a 4 giocatori con l'utilizzo della classe Team.
- L'UML risulta ordinato, sono state implementate tutte le classi possibili.
- Interfaccia actionStrategy per gestire gli effetti delle carte Personaggio è una buona soluzione.

Lati negativi

- Manca la cardinalità

- Classi enum non dichiarate

- Gestione della posizione di Professore, Studenti e Torri non efficiente, soprattutto le classi astratte non ritornano una posizione, non contenendo un metodo che ritorni l'id dell'isola di appartenenza oppure della board del player in cui sono situati. Di conseguenza la classe School potrebbe avere attributi come il numero di torri, il numero di professori posseduti, per una gestione più efficiente della dinamica del gioco.

- La classe MotherNature ci risulta ridondante in quanto esiste l'attributo/flag booleano hasMotherNature nella classe IslandGroup, che determina già la presenza in tale isola o gruppo di isole.

- Non è chiaro come si effettua il calcolo dell'influenza non essendo presente un attributo che rimandi al numero di studenti per colore presenti sull'isola e quindi si necessita di iterare per ogni giocatore per ricercare le loro posizione e la presenza del professore, ciò rende il processo macchinoso e di difficile gestione.

- School, IslandGroup estendono più di una classe

- Gestione non chiara della classe playerTourn, non capiamo la funzione degli attributi booleani che fanno riferimento alle carte

Confronto tra le architetture

- La decisione di gestire la modalità Esperta come estensione del gioco base può risultare vantaggiosa snellendo la parte del controller, ovvero facilitando l'interazione tra i due pattern.
- A parte la gestione della posizione delle pedine e torri, da noi implementata diversamente, ovvero tramite liste utilizzate all'interno delle classi di appartenenza, l'architettura risulta non troppo differente così come i come le interconnessione delle classi.