Peer-Review 1: UML

Riccardo Milici, Riccardo Motta, Matteo Negro
Gruppo 2

A.A. 2021 - 2022

Valutazione del diagramma UML delle classi del gruppo 1.

1 Aspetti positivi

Struttura MVC Il gruppo ha sviluppato in modo estensivo il pattern Model-View-Controller, andando ad inserire View e Controller, oltre a Cliente e Server. Tutto ciò ha reso più agevole la comprensione delle dinamiche di gestione delle partite, fornendoci inoltre diversi spunti di riflessione sulle nostre scelte implementative dell'architettura di gioco e sulla suddivisione del pattern MVC.

Suddivisione delle classi Il diagramma presenta una buona suddivisione delle classi, una per ogni componete del gioco, come richiesto dal paradigma Object Oriented, e risultano tutte quante ben delimitate e chiare. Tutto ciò facilita sicuramente l'implementazione grafica del gioco, potendo associare, in una futura GUI/CLI, ogni oggetto ad un elemento grafico.

Gestione delle isole Dall'UML si evince che il gruppo ha deciso di sfruttare due classi distinte per la gestione delle isole e dei loro agglomerati. Tale decisione favorirà sicuramente l'aspetto grafico e alcune logiche di gioco. Le due strutture, malgrado la ridondanza, permettono di preservare l'integrià concettuale e strutturale della singola isola, cosa sicuramente utile in fase grafica.

2 Aspetti negativi

Schema UML Il diagramma del gruppo risulta talvolta approssimato e superficiale. Molti dei metodi, per quanto semplici, non vengono esplicitati, rendendo non sempre chiare le reali scelte implementative per la gestione di casi delicati, come nel caso della valutazione delle influenze. Lo schema UML fornisce soltanto un'implementazione molto ad alto livello, indubbiamente sufficiente per avere un quadro generale, ma carente di spunti implementativi. Si di riportano di seguito alcuni esempi:

- `abstract class CharacterCard`: manca un metodo per disattivare gli effetti;
- `class Cloud`: manca un metodo per riempire la nuvola con nuovi studenti;
- `class Wizard`: manca un metodo per rimuovere le monete;
- `class Group`: manca un metodo per aggiungere un'isola all'aggregazione;
- Mancano le accortezze per l'implemantazione delle carte speciali e per i conseguenti effetti sulle dinamiche di gioco.

Dinamiche di gioco Lo schema UML tralascia i metodi e gli attributi per gestire l'ordine di gioco dei vari utenti, condizioni però non banali. Per esempio, l'ordine di gioco ad ogni round varia, mentre le carte vengono sempre giocate in senso antiorario a partire dal giocatore che ha messo la carta più bassa al round precedente. Sono quindi necessari strutture dati e classi per salvare e gestire queste informazioni.

Gestione degli agglomerati di isole La scelta implementativa che ha portato alla divisione delle isole dagli agglomerati porta dei vantaggi gestionali al gioco. Come evidenziato in precedenza, il calcolo dell'influenza negli agglomentati risulta più complesso, in quanto, essendo gli studenti separati sulle diverse isole, occorre processarle tutte prima di ottenere il risultato.

3 Confronto tra le architetture

L'analisi approfondita del diagramma UML del gruppo 1 ha evidenziato delle differenze implementative e strutturali rispetto al nostro schema.

La scelta di frammentare maggiormente i vari oggetti di cui è costituita l'architettura comporta alcuni vantaggi. Nonostante ciò, ci siamo mossi nella direzione opposta ottenendo una struttura del gioco più semplice che ricorda di più quella dei suoi ideatori. Ad esempio, la scelta del gruppo 1 di creare degli oggetti separati per la plancia di gioco invece di accorparli in un unico oggetto può rendere più articolata e meno immediata la comprensione di come sono state strutturate le dinamiche di gioco.

Come evidenziato nella prima sezione, il gruppo 1 ha anticipato nel diagramma UML alcuni aspetti della gestione MVC e della connessione dei dispositivi dei giocatori. Ciò ha impattato anche la scelta organizzativa ed implementativa delle classi, che, a differenza del nostro schema più concentrato sul Model, risulta perdere di accuratezza.

A differenza dell'altro gruppo, abbiamo maggiormente approfondito gli aspetti relativi al Model, inserendo molti metodi all'interno delle classi principali del gioco. Questo però ha comportato anche una piccola definizione del Controller, seppur unita al Model.

Il gruppo 1 ha deciso di centralizzare completamente nel server la gestione della logica applicativa, rendendo quindi il client una mera interfaccia grafica del gioco. Questo comporta un maggior carico nel server che potrebbe venir tranquillamente distribuito tra i vari client, i quali, come nella nostra idea di implementazione, effettuerebbero localmente i controlli di validità delle operazioni che stanno svolgendo.

I due progetti UML si differenziano inoltre sulla gestione delle isole. Il gruppo ha trovato una soluzione elegante con la definzione di due classi, `Group` e `Island`, che implementate in maniera sensata permettono di sfruttare i lati positivi di un `Group`, per esempio per valutare una delle condizioni di vittoria, ma senza perdere l'identità `Island`, utile soprattutto in vista dell'implentazione della GUI.

Un'ultima differenza tra i due elaborati risiede nella gestione delle pedine dei professori. Il gruppo 1 ha scelto di associare i riferimenti a queste ultime al giocatore e non alla scuola, perdendo così la chiara suddivisione delle varie componenti del gioco, quali plancia, tabellone, etc.