Peer-Review 1: UML

Simone Bevilacqua 10720775 Raffaele Chiaverini 10711746
Luca Barda 10697790 Michael Alibeaj 10750827
Gruppo AM-11

Valutazione del diagramma UML del progetto del gruppo AM-20

Lati positivi

Il diagramma UML del model implementa gran parte delle funzionalità di gioco riuscendo a gestire le regole presenti nella specifica di gioco.

- La carte relative ai personal goals vengono caricate dinamicamente da un file JSON di configurazione;
- Il riempimento della living room viene gestito tramite un file JSON di configurazione che specifica le diverse caselle utilizzabili in base al numero di giocatori della partita;
- L'utilizzo di diversi file di configurazione permette di modificare facilmente diversi aspetti del gioco senza mettere mano al codice che resta pulito e non dipendente da una specifica implementazione.

Lati negativi

In generale il diagramma presenta delle lacune sotto diversi aspetti.

- Non è presente alcuna interfaccia che specifichi i metodi esposti dal model verso il controller, ed eventualmente verso la view. Questi metodi dovrebbero corrispondere, in linea generale, alle mosse che può compiere il giocatore per far evolvere il modello, come ad esempio: pesca una tessera, finito di scegliere le tessere, scegli la colonna, ordina le tessere, inserisci le tessere;
- Per quanto riguarda la gestione delle tessere queste vengono istanziate tutte e 132 all'inizio della partita. Questa soluzione è valida e facilmente gestibile anche se non molto efficiente dal punto di vista della memoria occupata. Una soluzione alternativa potrebbe essere quella di tenere traccia del numero di tessere ancora generabili per ogni tipo (nel nostro caso 22) e ogni volta scegliere in maniera casuale il nuovo tipo di tessera da pescare, generandola solo nel momento in cui viene effettivamente utilizzata;
- Non è presente alcun riferimento allo stato di gioco. Non tenendo conto di quale fase della partita sia in esecuzione potrebbe risultare difficile gestire la

comunicazione tra il server e i giocatori come, ad esempio, capire quali mosse siano valide in un determinato momento;

- La classe Tile potrebbe essere sostituita da una classe di tipo enumeration in modo da semplificare la gestione delle tessere. Nella corrente implementazione del gioco ogni volta che si fa riferimento alle tessere bisogna effettuare un controllo dei valori ammessi, il che risulta una soluzione non molto elegante dal punto di vista stilistico, oltre che poco scalabile;
- Dal diagramma risulta poco chiara la modalità di generazione e gestione dei common goals. Il design corrente della classe CommonGoalCard porta a una serie di problematiche come la scrittura di codice non conforme ai principi della programmazione ad oggetti per la creazione degli obiettivi. Inoltre non viene sfruttata la somiglianza tra le strategie di controllo dei vari obiettivi portando alla duplicazione di parti di codice;
- Tutti i metodi delle classi del model non restituiscono né Optional<> né eccezioni, il che rende difficile (se non impossibile) la gestione di errori o chiamate dei metodi con parametri non ammessi o durante fasi di gioco non opportune;
- Il diagramma sembra non considerare la possibilità che un giocatore si possa disconnettere durante la creazione della partita. Si dovrebbe quindi predisporre un metodo per la rimozione di un giocatore dalla lobby di gioco;
- Non è chiaro dove verranno gestiti i turni di gioco, in particolare i controlli della bookshelf ad ogni fine turno. Nel caso in cui vengano effettuati dalla classe Game potrebbe risultare utile avere un riferimento al primo giocatore (possessore della chair) visto che, con la corrente implementazione, per ottenere tale informazione si dovrebbe “chiedere” singolarmente ad ogni giocatore;
- La classe PersonalGoalCardData potrebbe sfruttare la classe Coordinates sostituendo i due attributi di tipo int[] con un unico array di tipo Coordinates[] in modo tale da facilitare la comprensione del codice e del design della classe. Inoltre si potrebbe valutare l’accorpamento delle classi PersonalGoalCardData e PersonalGoalCard.

E’ importante notare anche alcune imprecisioni rispetto allo standard UML.

- Mancano i metodi costruttori di tutte le classi;
- Mancano i metodi getter e setter per gli attributi delle classi;
- Il tipo di alcuni attributi non è specificato propriamente (vedi Tile Matrice);

- Alcune relazioni tra le classi sono generiche, ad esempio la relazione tra Bookshelf e Player potrebbe essere una relazione di composizione visto che la bookshelf è strettamente legata al concetto di player;
- Viene utilizzato un tipo di dato (ScoringToken) relativo ad una classe non esistente.

Confronto tra le architetture

Il diagramma revisionato presenta un buon design generale per la gestione del gioco anche se non sono chiare le funzionalità aggiuntive che verranno implementate visto che non si evidenzia nessuna particolare scelta progettuale per il funzionamento né in multipartita, né in persistenza, né in caso di disconnessioni. L'unica funzionalità che potrebbe essere implementata è la chat che in effetti non dovrebbe avere a che fare con la parte model del pattern MVC.

I punti di forza da sottolineare sono il caricamento di alcuni file JSON di configurazione che rende l'intero progetto scalabile e facile da gestire.