Peer-Review 2: UML

Lucrezia Tagliatti, Andrea Scaffidi, Federico Zanon

Gruppo GC47

Valutazione del diagramma UML delle classi del gruppo GC57.

Lati positivi

- Per giocare un Character è stato definito un unico tipo di messaggio, a cui va passato l'indice del Character che si vuole giocare. Se non è necessario passare ulteriori parametri, allora l'effetto viene subito applicato. Altrimenti il client, tramite un altro messaggio, fornisce i dati mancanti. Per giocare personaggi diversi ma con caratteristiche comuni (ad esempio il Character 3 e il Character 5, che entrambi richiedono all'utente di inserire l'indice di un'isola), si usa lo stesso tipo di messaggio. Questo permette di diminuire il numero di messaggi che svolgono la stessa funzione, evitando ridondanza.

- È stata utilizzata una Macchina a Stati Finiti per rappresentare tutte le fasi di gioco. A livello di codice, per gestire le fasi della partita indicate nel diagramma, è stata definita un'enumerazione. Il Controller, in base al tipo di messaggio ricevuto e alla GamePhase corrente, esegue tutti i controlli necessari per accertarsi che l'azione richiesta sia permessa e, in caso di esito positivo, esegue il comando ricevuto leggendo i parametri inseriti nel messaggio. Il diagramma che rappresenta questo processo permette di comprendere bene il processo decisionale dietro gli stadi del gioco.

- In generale i diagrammi forniti risultano molto chiari, sia grazie alla divisione in package per quanto riguarda l'UML, sia per la definizione di una FSM, sia per il Sequence Diagram che mostra tutte le interazioni del server con l'utente. Inoltre, l'UML è compatto e formato da poche classi, permettendo di distinguere le relazioni e le entità in modo chiaro.

Lati negativi

- Nel Sequence Diagram non è chiaro che tipo di messaggio il server invii al client per richiedere l'input e/o dare informazioni sullo stato della partita. L'interazione dovrebbe essere modellata in modo più realistico, soprattutto per gettare le basi della successiva fase del progetto, quella di interazione con l'utente (potrebbe essere utile esplicitare anche i messaggi di servizio). Inoltre non è chiaro cosa rappresenti il messaggio MatchStatus().

- Mancano alcuni tipi di errori nella parte di comunicazione, ad esempio una Character Card può essere giocata solo se si ha abbastanza coins per pagarla e se ne può giocare solo una a partita.

- Non sono state individuate particolari criticità.

Confronto tra le architetture

- Le differenze più grandi tra le architetture del gruppo revisionato e del nostro gruppo riguardano il modello ed il controller.

- Per quanto riguarda la rete, è stata sfruttata anche nel nostro caso un pattern state per gestire le fasi del gioco. Anche la gestione delle classi Client, Server e Connection e dei messaggi è molto simile alla nostra.

- Abbiamo trovato molto interessante il modo in cui vengono gestiti i messaggi dei Character. (Stesso messaggio per carte diverse che richiedono stessi parametri e un messaggio unico per giocare carte che non hanno bisogno di ulteriori input). Non avendo ancora implementato la possibilità di giocare un Character all'interno di un turno a livello di rete, probabilmente gestiremo l'aspetto allo stesso modo.