Peer-Review 2: Protocollo di Rete

Arturo, Luca, Lorenzo
Gruppo 42

9 maggio 2022

Valutazione del Protocollo di Rete e diagramma UML del gruppo 52.

Questa peer-review ha il compito di mostrare i punti di forza e debolezza del protocollo di rete ideato dal gruppo GC52. Sfortunatamente non è stato possibile effettuare un'analisi approfondita, poiché la specifica della gestione del gioco non è stata ultimata.

1 Lati positivi

- Utilizzo di **messaggi** di conferma / errore brevi e coincisi (`OK/KO`).
- Buon utilizzo dello **strategy pattern** per CalcInfluenceStrategy
- Possibilità di **filtrare le partite** in base al numero di giocatori.
- Buona gestione dei `Message Objects`. È possibile ne manchino alcuni (es. `ActionNextTurn`) o che ve ne siano di superflui; ciononostante, il modello generale sembra essere valido.

2 Lati negativi e consigli

- **I messaggi di "wait"** nella fase di "start?" sono superflui. Finché la partita non inizia (e quindi arriva un messaggio di inizio gioco) si potrebbe assumere di essere in attesa.
- **Mother_nature** è modellato come un `singleton`, eppure ci sembra di aver capito che prevediate l'implementazione della funzionalità aggiuntiva "Partite multiple". Le due cose non sembrano poter coesistere.
- Non è chiaro l'utilizzo di **Bag** all'interno dei `Character`: il bag è unico? Ne vengono creati altri appositi per i vari characters? In questo caso si controlla che il numero di studenti sia consistente rispetto ai vincoli imposti dalle regole (numero di studenti totale, numero di studenti di ogni colore, etc...)?
- **Game - ExpertGame** sarebbero meglio modellate come parte del controller: di fatto la partita non è qualcosa di effettivamente visibile. Inoltre, entrambe le classi contengono molta logica, quindi potrebbero benissimo essere considerate come parte del controller. Nonostante la

scelta risulti poco rilevante in termini pratici, è importante avere chiaro il motivo per cui le classi siano state classificate come parte dell'uno o dell'altro.

- **ActionSetPlayerNickname** sembra superfluo. Il nickname viene già inviato come plain text nella fase di SetUp.
- Non è chiara la differenza di utilizzo tra `send` ed `asyncSend` in `ClientConnection`.
- Non è chiaro dove avvenga la **scelta del wizard**.

2.1 Possibili refusi

- `Island.getIslandSize()` ritorna void.
- `Island` ha come attributo un singolo studente, invece che una lista di studenti.
- Sono presenti diversi typo nell'UML, fate attenzione!

3 Confronto tra le architetture

L'architettura analizzata è solo parziale, e quindi non è stato possibile apprezzare ed esaminare appieno alcune scelte progettuali. Ciononostante, la scelta dei messaggi di conferma/errore (`OK/KO`) risulta essere pratica e chiara. Di conseguenza, valuteremo l'aggiunta di questa notazione nel nostro progetto.