Peer-Review 2: UML parte di rete

<Federico Perico>, <Davide Edoardo Pellegrino>, <Pier Luigi Porri>

Gruppo <GC56>

Valutazione del diagramma UML delle classi del gruppo <GC01>.

# Lati positivi

Ottima gestione della comunicazione di rete, in quanto questa viene gestita tramite la serializzazione di oggetti(messaggi). Questo a nostro parere è il miglior modo di gestire il protocollo di rete.

Abbiamo valutato positivamente l’adozione di un thread per ogni client, il che permette di mandare messaggi a tutti i client contemporaneamente; ciò rende il protocollo molto più efficiente. Una nostra interpretazione dell’UML e del testo fornitoci, ci permette di concludere che la parte di rete sia nascosta a model e controller, perché viene adottato l’incapsulamento, evitando quindi problemi come, ad esempio, una modifica di Model e Controller da parte del Client.

L’adattamento della rete al modello MVC è stato gestito in maniera corretta grazie alla presenza della VirtualView, che supponiamo possa fungere da View in remoto, permettendo quindi di conservare il modello MVC, senza che questo venga stravolto dalla parte di rete.

# Lati negativi

La mancanza di Sequence Diagrams ci ha reso difficile la comprensione del protocollo di scambio messaggi, di conseguenza abbiamo supposto molti dettagli e non ci è stato possibile analizzare coerentemente il protocollo.

Dall’UML notiamo la presenza di Observer e Observable, i quali sappiamo essere deprecati: vi consigliamo l’utilizzo e l’implementazione di EventListeners.

Non ci è chiaro l’utilizzo della serializzazione: a quanto ci risulta dall’UML, gli oggetti vengono inviati all’interno di altri oggetti (oggetti all’interno di oggetti-Messaggio). Abbiamo dei dubbi riguardo la nostra interpretazione data la mancanza di Sequence Diagrams, ma qualora fosse corretta l’implementazione comporterebbe un errore nella serializzazione.

# Confronto tra le architetture

Interessante il numeroso utilizzo di Enumerazioni per gestire tutti i tipi di Messaggio, in quanto potrebbe essere più comprensibile e pulito il codice. Stiamo, inoltre, valutando l’implementazione delle Lobby della partita; potrebbe un buon metodo per gestire le code di creazione delle partite.

Dal testo inviatoci e dall’analisi dell’UML, ci sembra di capire che vi sarà una VirtualView per ciascun Player. Questo tipo di gestione è totalmente diverso dal nostro: nel nostro modello l’utilità della VirtualView è di essere univoca, mentre ciascun client avrà una propria View, che sarà aggiornata in locale a seguito di cambiamenti del modello opportunamente notificati.