**Peer Review**

**Revisione svolta dal gruppo 68 sul lavoro riguardante le interazioni client – server del gruppo 67**

Quella che segue è la peer review che noi, gruppo 68 dello scaglione del Professor Sanpietro, ha svolto in merito alla bozza di progetto riguardante le interazioni client – server ideata dal gruppo 67 per la prova finale del corso di Ingegneria del Software.

Data revisione si basa sui sequence diagram che i nostri colleghi ci hanno fornito precisando che il lavoro da loro svolto per il momento non è da considerarsi come il protocollo di comunicazione di rete definitivo, ma sono solamente i primi accenni a quella che potrebbe essere poi la struttura definitiva.

In questa relazione, dopo un primo commento generale al diagramma, esprimeremo le nostre opinioni a riguardo valutandone quelli che riteniamo possano essere i suoi punti di forza e gli aspetti che invece ci fanno esprimere delle perplessità. Infine faremo una comparazione tra quanto pensato dal gruppo 67 e cosa invece avevamo in mente noi.

**Sequence Diagram**

I nostri colleghi ci hanno mostrato sei sequence diagram.

Il **primo sequence diagram** descrive la dinamica per la quale avviene la connessione iniziale al server di gioco da parte del client.

Si può notare come agli estremi di comunicazione si trovano la view (dalla parte del client) e il model (sull’altro estremo). Questi due elementi comunicano tra loro attraverso diversi elementi: dapprima un client controller, che rappresenta la rete client vera e propria, poi un network handler, che funge da intermediario tra client e server (rappresentato dal server controller).

Dopo che il network handler ha inoltrato la richiesta del client di potersi connettere per iniziare a giocare si attende l’esito, che poi viene inoltrato al client. Il giocatore riceve quindi attraverso la view una conferma e successivamente gli vengono chieste le modalità di gioco che preferisce.

Infine, dopo che il controller del server ha ricevuto le preferenze del giocatore, viene settato il model nella maniera opportuna e viene notificato il client.

Il **secondo sequence diagram** si occupa di chiarire come avviene il login del giocatore.

Ancora una volta agli estremi si trovano la view (che in questo caso rappresenta la *lobby* d’attesa) e il model; questa volta però il network handler mette in comunicazione tra loro cliente virual view, che serve per interfacciarsi con il controller che gestisce propriamente il model.

Il giocatore effettua il login alla lobby inserendo il nickname prescelto, che poi viene mandato come pacchetto in rete dopo una serializzazione (che avviene nel network handler). Al che la virtual view deserializza il nickname e lo passa al controller, che dopo aver controllato la sua validità lo passa al model per poter effettivamente registrare il player, che viene infine notificato dell’esito dell’operazione.

Il **terzo sequence diagram** mostra come avviene il passaggio di uno studente all’interno della dining room della board personale del giocatore. La comunicazione in questo caso è piuttosto semplice e consiste meramente in un passaggio di pacchetti di rete con le istruzioni per poter spostare correttamente lo studente nella fila opportuna della dining room.

Il **quarto sequence diagram** è analogo al terzo, ma mostra invece il passaggio di uno studente dalla hall del giocatore ad un isola. In questo caso, oltre che dover notificare il server del colore di studente che si vuole spostare, il client deve pure indicare l’isola su cui si vuole effettivamente posizionare la pedina.

Il **quinto sequence diagram** si occupa di mostrare come avviene la preparazione della fase effettiva di gioco, ossia il momento in cui il giocatore sceglie una carta. La view passa al network handler la scelta, che la serializza e la passa alla virtual view; questa, dopo averla deserializzata, la inoltra al server controller, che preso atto della scelta notifica il client che è stata registrata la sua azione.

Il **sesto e ultimo sequence diagram** illustra invece ciò che avviene nel momento in cui finisce il gioco. La view notifica il client controller dell’azione di fine partita e questo manda un messaggio apposito al network handler, che lo serializza e lo inoltra alla virtual view; quest’ultima, dopo aver deserializzato il messaggio, avverte il game cotroller, che si occupa di effettuare l’azione pratica che chiude la partita sul model. A questo punto viene lanciata un’eccezione, di cui viene avvertito il client. Dopo ciò il gioco viene effettivamente concluso, notificando anche il giocatore attraverso la view.

**Alcune considerazioni – pro e contro**

Questi diagrammi, per quanto schematici, riescono a rendere un po’ le idee che i nostri colleghi hanno riguardo all’implementazione del protocollo di rete per il loro progetto.

Per quanto all’interno dei vari grafici non sono raffigurati gli scenari nei quali ci sono dei problemi di connessione oppure errori dovuti ad input scorretti (o comunque situazioni simili), sono ben presenti per ogni situazione gli attori che devono intervenire nel corso dell’azione.

Per quanto riguarda i lati postivi del progetto del gruppo 67, c’è sicuramente da menzionare il fatto che l’accesso al model è sempre salvaguardato: non è mai possibile per il giocatore modificare direttamente attraverso il client controller le varie azioni di gioco (come per esempio lo spostamento delle pedine, la scelta di una carta e così via).

Un altro elemento che sicuramente è da tenere in considerazione è il fatto che lo scambio di messaggi tra client e server è gestito attraverso pacchetti di rete, che vengono serializzati all’invio e deserializzati alla ricezione. Inoltre, a gestire tale scambio c’è sempre un network handler che serve come intermediario tra i due soggetti.

Inoltre c’è anche notare come il giocatore si interfaccia con tutta la logica attraverso una view, che permette l’interazione della persona fisica con il gioco. Tale view ha un suo speculare su lato server, ossia la virtual view, che permette di agire sul model in alcune circostanze.

A quest’ultimo punto però c’è da aggiungere una perplessità in merito al fatto che non è particolarmente chiaro il modo in cui effettivamente la virtual view si frappone tra i vari componenti del lato server: in alcuni punti del diagramma non si capisce in maniera particolarmente immediata la dinamica con cui infatti essa si pone come intermediario tra gli attori in gioco, anche se tale confusione è probabilmente data dal fatto che tutte le varie azioni prese in considerazione sono di natura piuttosto diversa tra loro.

Un altro aspetto che ci sarebbe piaciuto che fosse stato approfondito di più è il caso in cui l’input dato dal giocatore sia scorretto o non accettabile (come per esempio il caso in cui il nickname non sia valido) oppure ancora non sia ammissibile l’azione di gioco scelta. In ogni caso ci rendiamo conto del fatto che in realtà tali situazioni non fossero strettamente richieste, e quindi di conseguenza non ci sentiamo di mettere tale carenza nei diagrammi come un aspetto a sfavore dei nostri colleghi.

**Una comparazione tra modelli**

Il modello proposto dal gruppo 67 non è poi così dissimile da quello che avevamo in mente noi.

Anche nel nostro caso infatti client e server si scambiano tra loro i messaggi attraverso dei pacchetti di rete e un processo di serializzazione e deserializzazione. Inoltre anche noi abbiamo pensato di usare una view, munita di un opportuno *observer* design pattern, per gestire le comunicazioni tra il giocatore e il client della rete.

Altra caratteristica che accomuna i due progetti è sicuramente il fatto di avere un model attentamente sorvegliato da un controller, che ne gestisce le varie attività e si assicura del fatto che non vengano svolte azioni illecite sul tavolo di gioco o comunque all’interno delle sue dinamiche.

Una delle differenze che ci sono saltate all’occhio sono state quelle legate al setting della partita: nel modello dei nostri colleghi le scelte sono fatte insieme in maniera piuttosto diretta, nel nostro caso invece al giocatore viene dapprima chiesto il numero di giocatori che vogliono partecipare alla partita e successivamente viene chiesta la modalità di gioco desiderata, quindi le due scelte vengono svolte in momenti ben distinti (cosa che nel caso della nostra implementazione del gioco fa la differenza, ma poiché non conosciamo a fondo l’idea di gioco che hanno i componenti del gruppo 67 non sappiamo se effettivamente tale distinzione sia significativa o meno).

Oltre a ciò crediamo che fare un confronto più approfondito non sia possibile: in primo luogo perché, come già detto in precedenza, i diagrammi mostratici dai nostri colleghi non sono che una prima stesura di quella che potrebbe essere la loro implementazione di rete; secondariamente perché appunto non avendo sottomano anche i casi in cui la comunicazione tra client e server non va a buon fine non è per noi possibile comparare i casi critici per lo scambio di messaggi.

**Conclusioni**

In conclusione, anche se non sono i diagrammi definitivi della comunicazione di rete, questi sequence diagram lasciano intendere l’idea che hanno i nostri colleghi a riguardo.

Riteniamo che il lavoro a noi presentato sia ben pensato e sicuramente ponderato attentamente, anche se ovviamente ha bisogno ancora di essere sviluppato e perfezionato.

Ci sentiamo di consigliare al gruppo 67 di continuare su questa linea ma trattare più approfonditamente tutti i casi in cui la comunicazione possa non andare a buon fine, poiché infatti tali casi critici sono quelli sui cui bisogna probabilmente focalizzare di più la propria attenzione dal momento che sono i più delicati da dover gestire.