Peer-Review 2: sequence UML

Edoardo Bergamo, Osama Atiqi, Ferdinando Cioffi, Zining Chen

Gruppo 11

Valutazione del diagramma UML delle classi del gruppo 20.

Lati positivi

- Il sequence diagram presentato offre una chiara e ben strutturata rappresentazione delle interazioni tra i componenti del sistema. Il protocollo definito è efficace e completo, mostrando un'adeguata gestione delle azioni del client e delle risposte del server. Pertanto, le possibilità di apportare modifiche significative al fine di migliorare o ottimizzare il protocollo sono minime.
- Il protocollo copre tutte le funzionalità richieste dalla specifica del gioco da tavolo, assicurando che il sistema sia in grado di gestire in modo efficace e affidabile tutte le operazioni richieste durante una partita.
- Il protocollo descritto prevede che ogni azione eseguita dal client sulla partita generi una risposta immediata da parte del server, comunicando sia l'esito dell'azione (successo o insuccesso) sia informazioni aggiornate sullo stato attuale della partita. Questo approccio risulta intuitivo e immediato da implementare, ciò assicura una semplice interazione con il sistema. Inoltre, la struttura chiara del protocollo implica che eventuali future espansioni o modifiche al sistema saranno agevolmente implementabili, garantendo una maggiore flessibilità e adattabilità nel tempo.

Lati negativi

- I metodi del client che eseguono azioni includono tutti un parametro denominato "nickname". Si presume che questo parametro sia utilizzato durante la connessione RMI per identificare il giocatore che ha invocato un dato metodo. L'uso di un identificatore pubblico, noto a tutti i giocatori della partita, rappresenta una potenziale vulnerabilità di sicurezza che potrebbe essere sfruttata da un client malevolo. Un client modificato, dopo essere entrato nella partita con il proprio nickname e aver scoperto i nickname degli altri giocatori, potrebbe iniziare a inviare messaggi fingendosi un altro utente. Ad esempio, considerando una partita con due giocatori, Alice e Bob, durante il turno di Alice, Bob potrebbe inviare un messaggio del tipo `PlayCardMessage(Alice, index, angle, target, side)` per sabotare la partita di Alice. Dal diagramma di sequenza, non è chiaro se questo scenario è stato preso in considerazione. Una soluzione potenziale potrebbe essere l'assegnazione, all'inizio della partita, di un codice univoco e segreto a ciascun giocatore. Questo codice sarebbe utilizzato per identificare in modo sicuro ogni giocatore durante la sessione di gioco.

- Dal sequence diagram i messaggi che il server manda al client spesso contengano stringhe che descrivono lo stato del model, un esempio è `PickAckMessage("The game is almost done, the final phase starts now!", ResourceTop, GoldTop, ResourceVisible, GoldVisible, Cards, NextPlayer, mustPick: "False")`. Questo tipo di messaggi possono andare bene se utilizzati solo a scopo di debug, ma per segnalare ai giocatori lo stato della partita sarebbe preferibile lasciare che il client decida come mostrarlo. Una soluzione, simile a quella adottata dal nostro gruppo potrebbe essere l'utilizzo di `Enum` per segnalare in quale fase si trova la partita o quali azioni devono compiere i giocatori.
- Dal sequence diagram non è chiaro cosa succeda in caso un'azione di gioco risulti non valida, ad esempio cosa succeda nel caso si tenti di posizionare una carta senza che i requisiti siano soddisfatti. Sebbene ipotizziamo che tali dettagli siano semplicemente stati omessi durante la stesura del diagramma, è importante assicurarsi che nel programma siano considerati tutti gli scenari, anche quelli in cui il client tenta azioni invalide o in cui ci sia qualche errore nel server.
- Alla conclusione di ciascuna azione, il server comunica frequentemente al client informazioni sullo stato attuale della partita, informazioni che il client già dovrebbe possedere. Queste informazioni includono, ad esempio, le carte visibili o l'intero campo di gioco di un giocatore. Sebbene la quantità di dati trasmessi sia minima, sarebbe possibile ridurla all'essenziale comunicando al client soltanto le modifiche del gioco e non tutto lo status della partita.

Confronto tra le architetture

Nel diagramma UML di sequenza del Gruppo 20, si osserva un affidamento su un sistema di riconoscimento (acknowledge) per ogni azione intrapresa dal client. Questo approccio garantisce un elevato livello di sicurezza nella trasmissione dei messaggi.

Al contrario, il nostro Gruppo ha optato per un modello in cui il client invia esclusivamente messaggi di comando al server, mentre è compito del server inviare ai client messaggi che riflettono i cambiamenti di stato. In questo modo, il client riceve aggiornamenti basati sui messaggi del server, senza la necessità di attendere un riconoscimento per ogni azione eseguita. Questa strategia favorisce una maggiore flessibilità e scalabilità del sistema. In caso di errori, il client viene prontamente informato dal server e può quindi reagire di conseguenza, l'approccio basato su acknowledge assicura però una maggiora robustezza e garantisce una reazione immediata in caso di errori.

In conclusione, riteniamo che entrambe le architetture siano adeguate e valide nello svolgere il compito di protocollo di comunicazione tra client e server durante le fasi di gioco, sia di preparazione della partita, sia di gioco effettivo.