Peer-Review 2: Protocollo di Comunicazione

Giovanni Manfredi, Mattia Martelli, Sebastiano Meneghin
Gruppo 27

Valutazione della documentazione
del protocollo di comunicazione del gruppo 26.

Lati positivi

Abbiamo trovato nella rappresentazione dei messaggi tramite UML una buona soluzione per mostrare attributi e metodi dei messaggi del protocollo.

Inoltre il protocollo utilizzato dal gruppo 26 prevede un client che manda attivamente le richieste al server per eseguire le sue azioni, scelta che riteniamo interessante in quanto opposta a quella da noi scelta.

Lati negativi

In generale, la descrizione del protocollo mediante i soli sequence diagram e l'UML dei messaggi ci è sembrata di difficile lettura in quanto ha lasciato diversi punti non specificati ad un livello di dettaglio sufficiente da poter comprendere se il protocollo funziona o no in modo corretto.

Dal protocollo, visto che non è esplicitato, sembra che il gruppo 26 abbia implementato le funzionalità avanzate (FA) seguenti:

- Partita a 4 giocatori
- 12 Character Cards
- Partite multiple

Ad un primo sguardo ai diagrammi si nota che non è stato esplicitato (mediante ad esempio dei rettangoli come sfondo) la suddivisione dei diversi process tra client e server, anche se riteniamo che solo client sia sul client e gli altri process appartengano al server.

Il protocollo inizia dalla fase di *Join* non esplicitando la fase di connessione che crediamo sia stata volontariamente omessa, considerando il client già connesso. Questo salto logico, non fa ben comprendere come i process siano stati iniziati e se il model sia già istanziato nel client. Vista la scelta di implementare la FA con le partite multiple, ci siamo aspettati di trovare una gestione transazionale delle operazioni in questa prima fase, visto che sul server saranno connessi più giocatori

contemporaneamente, che potrebbero scegliere lo stesso wizard e la stessa partita e superare il numero massimo di giocatori massimi in una partita. Ad esempio se un giocatore sceglie di un particolare match, non ci sono controlli rispetto al numero di giocatori in attesa e lo stesso vale per la scelta del wizard, che se fatta allo stesso momento può creare due giocatori con lo stesso wizard. Avremmo inoltre considerato opportuno l'uso di una *Map<IdMatch, List<players> >* al posto di una *List<List< > >* per *SendMatchesMessage()*.

Arrivati alla fase di *Pianificazione* abbiamo notato che tra la fase di *Join* e questa non c'è stata alcuna gestione delle attese dovute all'ingresso di giocatori in partita e che mai viene fatto partire il gioco esplicitamente (quando il gruppo è completo).

Quindi in *Pianificazione* vediamo la gestione del passaggio di turno, che però senza l'utilizzo di un ciclo rappresentandola in questo modo , permette di giocare solo il primo giocatore.

Passando alla *ActionPhase 1* notiamo l'assenza di un controllo su un posizionamento errato di uno studente (ad esempio avrò un errore se voglio posizionare uno studente in un tavolo già pieno). Leggendo la nota lasciata sulla gestione di *CharacterCard* ci è sembrata una scelta più che corretta limitare alcune operazioni creando sequenze atomiche tuttavia crediamo che il giocatore dovrebbe avere la possibilità di giocare la carta appena prima della fine del turno (dopo aver preso gli studenti da una nuvola). Lo scambio di messaggi per la *CharacterCard* ci è sembrato coerente con il protocollo, ma non appare chiaro il contenuto dei messaggi e manca il loop per richiedere al client le informazioni nel caso di un una richiesta errata del client.

Nella *ActionPhase 2* invece ci è sembrata un po' eccessiva la riduzione delle operazioni sul movimento di madre natura ad un'unica funzione in quanto le azioni da gestire con quel movimento sono tre: il movimento di madre natura, alternativamente il controllo o conquista dell'isola e l'unificazione delle isole. E' specificato tramite una nota che la funzione di occupa di tutte queste componenti, ma non è chiaro se al termine di controllo o conquista e di unificazione delle isole viene verificata la condizione di vittoria, in quando riterremmo fuori regolamento unificare le isole dopo che l'ultima torre di un giocatore è stata piazzata. Sempre in questa fase viene fatto un controllo sulla correttezza del movimento richiesto di madre natura, ma in caso di errore non viene data la possibilità di inserire un altro valore di movimento creando così un'azione illegale per il gioco ovvero il movimento (da regole madre natura deve muovere di almeno un'isola ogni turno (è risolvibile utilizzando un ciclo).

Arrivando alla fase di *EndGame* abbiamo notato la condizione dell'opzionale presente che dovrebbe definire quando il gioco potrebbe finire tuttavia non è chiaro quando dovrebbe essere controllata la condizione. Crediamo che per gestire questo problema ci siano due alternative possibili:

- Inserire questa parte di sequence diagram nei punti in cui il gioco può terminare e controllare la condizione di vittoria per entrare nell'opzionale

- Creare un processo parallelo che si occupi di gestire la fine del gioco che viene richiamato una volta che è vera la condizione di vittoria

Abbiamo inoltre notato l'assenza della gestione delle disconnessioni che si comporrà in una chiusura della partita per tutti gli altri giocatori.

Confronto

La scelta adottata dal gruppo 26 di utilizzare un client attivo, porta ad avere meno messaggi scambiati in generale e quindi ad un protocollo più lineare. Tuttavia lasciare la possibilità al client di fare richieste direttamente implica un maggiore controllo su ciò che l'utente manda al client che non sempre è stato fatto in modo completo o con i cicli richiesti. Il nostro protocollo invece ha previsto un client più passivo, che risponde alle richieste del server e sceglie tra quelle disponibili, questo porta appunto ad avere una richiesta esplicita del server al client ogni volta che c'è la possibilità di compiere un'azione.

Dopo aver visionato questo protocollo abbiamo notato alcuni controlli sull'input dell'utente o su errori del programma che non gestivamo e quindi ora andremo a sistemare. Riteniamo che il gruppo 26 dovrebbe sistemare le parti che abbiamo citato, in particolare la disconnessione degli utenti e la gestione della fine del gioco.