# Peer-Review 2: Protocollo di comunicazione

Alberto Nidasio, Matteo Pignataro, Alberto Sandri

Gruppo GC09

Valutazione del protocollo di comunicazione del gruppo GC19.

## Lati positivi

Per la distinzione e applicazione automatica dei messaggi è stata applicata la tecnica del polimorfismo che rispecchia a pieno le potenzialità della programmazione orientata agli oggetti.

Le classi ClientConnection e Server utilizzano una thread pool con dei metodi synchronized per rispettare la funzionalità aggiuntiva “Partite multiple”. Questo risulta essere molto vantaggioso lato server perché permette di gestire le partite senza avere l’oneroso processo di creazione di thread ogni volta.

Utilizzo del pattern Observer combinato con la VirtualView per notificare i vari client.

## Lati negativi

Le classi UserAction, ViewUpdateMessage e BaseUserMessage sono classi con tutti attributi privati e nessun metodo, quindi risulta impossibile accedervi dall’esterno (anche solo se fossero classi per la comunicazione di dati).

Usare le stringhe come messaggi (BaseServerMessage, CustomMessage) è funzionale ma non sfrutta a pieno la programmazione object-oriented, ad esempio usando la serializzazione si potrebbe far uso del command pattern per distinguere messaggi con payload diversi ed evitare una catena di if per la distinzione del comando.

Nella distinzione della UserAction dato che vi è un parametro ActionType si suppone che ad un certo punto nel codice debba essere fatta una catena di if-else if tale da distinguere ogni messaggio.

Il controllo della validità di una azione è oneroso da parte del controller, si suppone che ogni metodo del controller debba interrogare più classi del model per decidere se l’attuale azione sia valida o meno.

Vi consigliamo di provare a fare un sequence diagram che mostri l’invocazione dei vari metodi quando il server riceve un messaggio dal client.

Errori di battitura:

- La classe Controller non ha alcun getter per l’utilizzo dello stesso da parte della classe Match (se viene utilizzato il pattern Observer, ne manca la sua implementazione);
- In BaseActionController e TurnFlow manca l’azione per muovere madre natura;
- RoundManager probabilmente andrebbe messo nel controller.

## Confronto tra le architetture

Anche noi per quanto riguarda la gestione delle partite multiple abbiamo optato per una organizzazione delle partite come Match (collection di player) utilizzando la classe Server come unità centrale.

Abbiamo utilizzato il polimorfismo per l'esecuzione di comandi e azioni di gioco evitando quindi la distinzione forzata all'interno del codice.