

Peer-Review 1: UML

Matteo Rampone, Alessandro Rossi, Arianna Salerno

Gruppo 21

Valutazione del diagramma UML delle classi del gruppo 31.

In base agli elementi presenti nel modello consegnatoci e sugli elementi in esso contenuto abbiamo individuato i seguenti:

Lati positivi

- Apprezziamo la scelta di creare un pacchetto contenente tutte le enumerazioni
- Il modello è resiliente alla disconnessione di uno o più giocatori grazie all'attributo `activePlayers` presente dentro la classe `Game`, che invece abbiamo scoperto mancare al nostro modello.
- L'implementazione delle squadre con un parametro `teammate` è molto elegante, il concetto di team è di fatto molto semplice e risulta ben modellato in questa maniera
- L'entità `BoardCell` e' in se' una buona idea, sarebbe forse implementabile mediante l'uso di generics per ridurre il livello di ambiguità ma è interessante il concetto di poter modellare un contenitore di pedina come entità a sé stante ai fini di una maggiore chiarezza del codice. inoltre, pensiamo che in tal modo il concetto di cella possa essere esteso alle torri, rendendo il concetto di contenitore per pedine ancora più solido.

Lati negativi

- La classe `OtherCards` estende la classe `CharacterCard` senza aggiungere né attributi né metodi, converrebbe eliminarla e collegare le sottoclassi ("`Contadino`", "`Ladro`"...) direttamente a `CharacterCard`.
- `DiningRoom` contiene 50 `BoardCell` che idealmente possono ricevere le chiamate `SetProfessor/hasProfessor` (a run time sicuramente saranno controllate ma idealmente sembra che la `DiningRoom` possa contenere professori).
- La classe `ProfessorTable` ci è sembrata ridondante poiché già la classe `Professor` è in grado di mappare i docenti con i giocatori che li possiedono.
- La classe `Tower` è una semplice riscrittura dell'enumerazione `TowerColour` e lo stesso discorso si applica alla classe `Student` (probabilmente è stata aggiunta per poter inserire il metodo `getType` ma le enumerazioni possono contenere metodi quindi consigliamo di condensare il metodo `getType` proprio dentro `PawnType`).

A valle di queste caratteristiche individuate:

Confronto tra le architetture

Confrontando la nostra proposta di architettura con quella del Gruppo 31 ci siamo accorti di vari dettagli e accortezze di modellazione che troviamo desiderabili. In primis la creazione di più pacchetti oltre a `model` e `controller`.

Inoltre, abbiamo scoperto come alcune delle nostre scelte di progettazione fossero molto più complesse del previsto. Si prenda come esempio l'implementazione dei team all'interno dell'oggetto `Player`.

Alcune scelte del Gruppo 31, tuttavia, ci sono parse ridondanti o non completamente al passo col paradigma di programmazione a oggetti.

Abbiamo anche notato come il modello GC31 sia radicalmente diverso in termini di struttura dal nostro se si osserva il controller: nel nostro caso il controller si sviluppa attorno alle fasi e alle

azioni che possono essere svolte da un giocatore. Notiamo invece che il controller del Gruppo 31 sia frammentato in più classi senza un ovvio entry point per la view.