Peer-Review 1: UML

Pietro Legramandi, Simone Leone, Filippo Locatelli
Gruppo 60

3 aprile 2022

Valutazione del diagramma UML delle classi del gruppo 59.

1 Lati positivi

Il model si presenta abbastanza completo per quanto riguarda le entità in gioco. Ci è piaciuta la classe Game come database generale di una particolare partita e soprattutto il fatto che fosse l'unica classe ad interfacciarsi esternamente al Model. Abbiamo notato l'abbozzo di comunicazione client-server che, seppur non richiesto, aiuta molto a capire come verranno gestite le richieste di partite da più giocatori. La parte su carte personaggio, carte assistente e relativi Deck (con ereditarietà) ci è parsa molto utile per scopi futuri di implementazione delle regole esperti. Complessivamente buona la scelta delle classi da utilizzare e delle loro interazioni. Corretta la scelta delle associazioni e delle cardinalità. Analizzando anche metodi e attributi ci risultano corrette le scelte per la loro visibilità.

2 Lati negativi

A nostro parere ci sembra inutile avere un'intera classe dedicata a studenti, professori e torri quando (in modo molto più semplice) questi possono essere rispettivamente due array di interi (o list) ed un contatore (simile al contatore dei coins). A corroborare la nostra tesi vi è il fatto che queste classi risultano assai prive di attributi e metodi e risulta perciò una forzatura averne una classe dedicata. Mancano alcuni metodi base ma immaginiamo (come è normale pensare) che questi vengano prima implementati su codice. Non è

stato mappato il modo in cui si unificano isole con stessa influenza. Risulta poco chiara dall'UML la scelta dell'architettura di rete, dove implementate MVC e se preferite che la gestione delle funzioni venga gestita maggiormente a lato controller oppure a lato model. Alcuni metodi di Game ci sembrano appartenere al Controller; c'è però da dire che se l'idea di Model che avete in mente consiste nell'avere delle classi ibride che non fungono solo da database ma che implementano anche logiche di gioco allora è accettabile. Riteniamo ridondante l'utilizzo dell'ereditarietà con BoardComponent così come nel caso della pedina Pawn: da entrambe non vi è nessuna associazione in partenza e i metodi a loro appartenenti sono ridottissimi.

3 Confronto tra le architetture

Noi non abbiamo ancora iniziato la parte di comunicazione internet e nemmeno la gestione delle richieste a modello client-server in multithreading. Ci è risultato molto utile visionare il vostro abbozzo di comunicazione. A differenza della nostra architettura, abbiamo notato la vostra scelta di implementare classi di Model View Controller sul Client definendo una sorta ti ripartizione Fat Client. A nostro parere ciò rischia di aumentare la complessità nella scrittura del codice.