Peer-Review 1: UML

Isaia Belardinelli, Lorenzo Biasiolo, Leonardo Brusini

AM28

Valutazione del diagramma UML delle classi del gruppo AM01.

Lati positivi

- Abbiamo apprezzato, oltre all'architettura nel suo complesso, in particolare l'immediatezza nell'invocazione dei metodi, nonostante crediamo possa talvolta violare il principio di modularità, mediante l'utilizzo esteso di riferimenti ad oggetti come parametri. Ciò rende l'architettura nel suo complesso molto snella e fruibile sin da una prima lettura, pur limitando in taluni casi il completo apprezzamento del ruolo svolto da tal componente, ad esempio nella gestione delle "CharacterCard".
- Classe apposita per "delegare" la gestione del turno e per salvare l'ordinamento del turno dei giocatori.

Lati negativi

- Valori di ritorno: Tra gli aspetti negativi, secondo il nostro punto di vista, annoveriamo l'utilizzo esteso del tipo booleano come parametro di ritorno dei metodi. Per come abbiamo letto ed interpretato il vostro UML, l'architettura potrebbe beneficiare del ritorno, da parte delle funzioni, del tipo void qualora non sia necessario restituire un riferimento all'oggetto su cui è terminata la computazione, evitando il rischio di complicare la successione delle invocazioni ai metodi e il progetto nel suo complesso. Ad es, il metodo joinIsland potrebbe tornare void.
  Nel caso il valore di ritorno bool rappresenti una sorta di "avviso" del metodo andato o meno a buon fine, ad esempio perché chiamato con dei parametri non accettabili o perché chiamato in un momento del turno diverso da quello in cui il metodo dovrebbe effettivamente essere chiamato, suggeriamo invece di rendere il model un po' meno "pensante", e di spostare questi controlli nelle classi del controller, che si occuperà di chiamare tali metodi unicamente con valori corretti e nei giusti momenti della partita.
- Win conditions: Per quanto riguarda la verifica delle condizioni di vittoria mediante i metodi isWinningPosition e isWonByResources, mantenendo una strategia iterativa potrebbe esser utile, per una questione concettuale, unificarli in modo da effettuare una sola chiamata per ciclo di gioco. Alternativamente, mantenendoli entrambi, potrebbero teoricamente giovare della loro integrazione in un pattern observer, in cui sarebbero gli stessi oggetti, su cui si vogliono effettuare le valutazioni, ad invocarli al verificarsi di un evento interessante.
- Gestione delle isole: Se è vero che l'implementazione potrebbe funzionare, è anche vero che l'attuale design di Archipelago, come sottoclasse di Isola, è "poco utile". Concettualmente poi un arcipelago non è un'isola "particolare", ma un insieme di esse, si presta poco quindi al concetto di sottoclasse

Confronto tra le architetture

I seguenti punti sono differenze progettuali tra i due gruppi, che reputiamo come scelte alternative. Non riteniamo necessariamente una di queste scelte come “migliore” dell’altra:

- Modalità esperti: il gruppo AM01 ha deciso di separare totalmente la modalità semplificata da quella per esperti. Ogni classe che, da regolamento, subisce delle modifiche dipendenti dalla modalità in cui si gioca viene estesa con una corrispondente classe “Expert”. Questo fa sì che durante una partita in modalità semplificata, le carte personaggio e le monete non siano presenti nel model, e vengono fatti meno controlli durante l’esecuzione di alcuni metodi.
  Il nostro gruppo, invece, prevede un’implementazione sempre completa del model, ossia compresa di carte personaggio, monete, madre natura che controlla una eventuale attivazione di carte che possano modificare il calcolo dell’influenza, ecc.
  Sarà poi il controller a distinguersi in partita semplice / per esperti e, nel caso, nascondere al client la possibilità di chiamare determinati metodi.
- Carte personaggio: il gruppo AM01 utilizza uno strategy pattern che però racchiude i 12 effetti in 3 sottocategorie, dando in questo modo la possibilità di implementare solo 3 classi derivate da “Character”. Immaginiamo che poi all’interno di queste classi vengano comunque distinti i vari effetti.
  Il nostro gruppo invece utilizza sì uno strategy pattern ma con una sottoclasse per ogni effetto, ad eccezione degli effetti che hanno un concreto utilizzo solo successivamente nel turno, in questo caso raccolte in un unico effetto “DefaultCardEffect” che si occupa di segnalare, con un booleano, l’attivazione della carta. Saranno poi le altre classi, se necessario, a chiedersi se quella determinata carta è stata attivata e prendere decisioni di conseguenza.