Peer-Review 1: UML

Filippo Gandini, Federico Mazzucato, Nazzareno Messinò
Gruppo 44

3 aprile 2022

Valutazione del diagramma UML delle classi del gruppo 43.

1 Lati positivi

- Gestire l'informazione del mago nella classe Player anzi che nella classe Assistant consente un accesso più diretto alla risorsa e implica un risparmio di memoria.
- Il progetto descrive le componenti di gioco tramite una codifica di interi e gestisce i diversi insiemi con degli array statici. Questo approccio rende il progetto più leggero in memoria e consente dei calcoli più efficienti.

2 Lati negativi

- La classe Tower è superflua, le torri sono rilevanti solo in funzione della loro presenza ma non svolgono nessun ruolo attivo nel gioco. Si consiglia una progettazione simile a quella di professori e studenti. Inoltre, si consiglia di aggiungere dei riferimenti tra School e Island per sostituire il ruolo di 'ponte' svolto da Tower per il calcolo dell'influenza.
- La classe Game risulta troppo pesante. Una sua implementazione completa includerebbe troppi attributi e dovrebbe gestire direttamente diverse dinamiche di gioco. Si consiglia di adottare una maggiore granulazione delle classi aggiungendo una classe Board (con riferimenti a

Cloud e Island) in modo da separare logicamente giocatori e campo di gioco.

- Ad alcune classi mancano dei riferimenti necessari per l'implementazione dei loro metodi, ad es:
  - Il metodo free_cloud() di Cloud necessita di un riferimento a Player/School per assegnare gli studenti
  - il metodo add_professor() di School necessita di un riferimento all'altra School per stabilire se ha i requisiti per guadagnare il professore

  Si consiglia di aggiungere le rispettive associazioni o dei parametri alla segnatura dei metodi.

- Alla classe Cloud manca un attributo che definisca il numero di studenti che contiene in base alla modalità. L'attributo è necessario per salvare il valore passato come parametro al costruttore.

- L'attributo pubblico has_mama di IslandComponent è poco safe, potrebbe portare a errori inaspettati. Si consiglia di rendere l'attributo privato oppure di gestire l'informazione di madre natura nella classe Game/Board.

- Il progetto descrive coerentemente il gioco, ma ha bisogno di meccanismi per la gestione delle classi: classi aggiuntive che gestiscono le interazioni tra le diverse classi (observers, modifiers ...)

- Per una modellazione esaustiva, la classe Player prevede altri attributi. Consigliamo l'aggiunta di IP, porta e nome.

- Il diagramma UML non esplicita tutte le cardinalità delle associazioni (es Game -> Island)

- La segnatura di alcuni metodi (es: get_student() di Island, get_supremacy() di IslandDecorator) è errata e non consente di comprendere il relativo comportamento della classe.

3 Confronto tra le architetture

- Il gruppo 43 ha favorito una progettazione basata su vettori e convenzioni a differenza della nostra incentrata su collezioni ed enumerazioni. Grazie a queste scelte, l'UML del gruppo 43 risulta più 'leggero' in memoria e, di conseguenza, predisposto a calcoli più efficienti. Tuttavia, questo approccio pronostica poca scalabilità e una scarsa leggibilità del codice.

- L'UML del gruppo 43 prevede pochi meccanismi per la gestione delle classi. Il nostro diagramma prevede delle classi Modifier e Observer per la modifica diretta del Model. Questa differenza di progettazione preannuncia un Controller più pesante per il gruppo 43.