RisiKOOP

Il gioco strategico per la conquista del mondo

Matteo Caruso, Matteo Ceccarelli, Franceso Sacripante

23 luglio 2025

Indice

Capitolo 1

Analisi

1.1 Descrizione e requisiti

Il software mira a replicare il gioco Risiko, un gioco da tavolo di strategia a turni dove ogni giocatore controlla una squadra di unità allo scopo di completare un obiettivo determinato da una Carta Obiettivo pescata a inizio partita. Questa richiederà di conquistare dei continenti, annientare un'altra armata oppure conquistare un certo numero di territori. Il gioco inizia spartendo tutti i territori tra i giocatori e dà dei territori iniziali con cui rinforzarli. Ogni turno turno, il giocatore otterrà vari carri armati da posizionare sui suoi territori. Potrà poi attaccare territori adiacenti ai propri. Se riesce a conquistare almeno uno stato otterrà una Carta Territorio, utilizzabile per giocare combo al fine di ottenere ulteriori unità nei successivi turni. Infine avrà l'opportunità di spostare delle unità fra i suoi territori.

Tipi di Combo

Le combo sono sempre tris di carte territorio, ognuna ricompensa un certo numero di unità:

- 3 cannoni: 4 unità.
- 3 fanti: 6 unità.
- 3 cavalieri: 8 unità.
- Un fante, un cannone e un cavaliere: 10 unità. [1]
- Un Jolly e due carte uguali: 12 unità.

[1] Non è possibile sostituire una delle carte con un Jolly in questa combo.

Requisiti funzionali

- Il software dovrà permettere di giocare a una semplice versione di Risiko.
- Ogni giocatore ha una sua Carta Obiettivo e varie Carte Territorio.
- L'attacco avviene tramite il tiro di dadi, il cui confronto ne determinerà l'esito.

Requisiti non funzionali

- La mappa è selezionabile, scelta dai giocatori a inizio partita.
- I giocatori dovranno poter nascondere le proprie Carte Obiettivo e Territorio agli altri giocatori.

1.2 Modello del Dominio

Il gioco inizia con la selezione dei giocatori, del loro colore e della mappa. Successivamente vengono assegnati i territori, ed è chiesto ai giocatori di posizionare le loro unità rimanenti in quei territori. Ora inizia il game-loop del gioco, che si ripete fino a quando un giocatore non vince:

- Fase di rinforzo.
- Fase di attacco.
- Fase di spostamento finale.

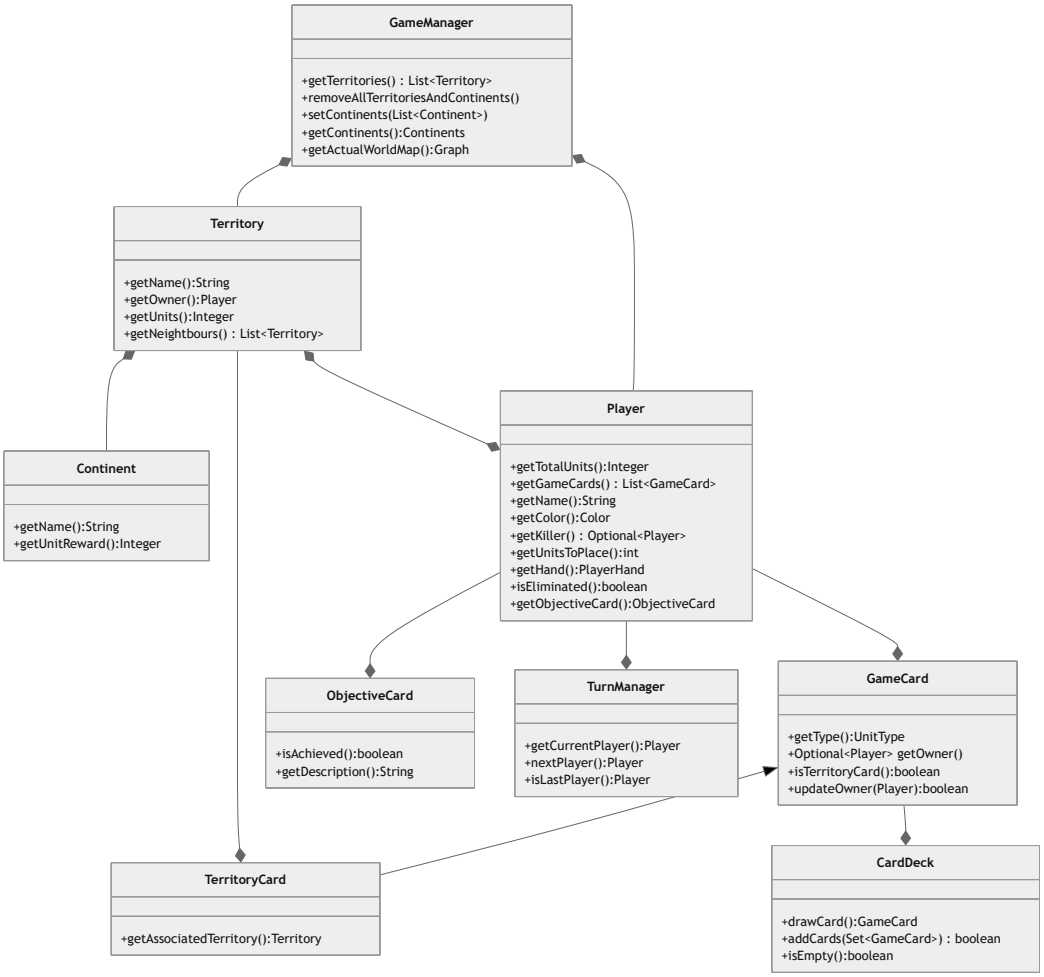

Figura 1.1: UML del modello svolto dopo l'analisi dei requisiti.

Capitolo 2

Design

2.1 Architetura

L'architettura del software è basata su un pattern Model-View-Controller (MVC). L'entry point dell'applicazione è il `Controller`, che si occupa di avviare il model, che eredita da `GameManager`, e le view registrate, che eredtano da `RisikoView`. Sicccome questo gioco è un gioco abbastanza modulare, si è ritenuta utile la realizzazione di una macchina a stati finiti. Una macchina a stati è una macchina che suddivide la logica del gioco in tante piccole logiche dette stati. In un determinato istante del gioco, solo uno stato può essere eseguito alla volta, se si vuole eseguire un'altro stato si deve per forza cambiare e rimpiazzare lo stato vecchio con uno nuovo. Il vantaggio di questo approccio è che ogni stato è appunto una logica diversa scollegata dalle altre e quindi molto più maneggiabie, inoltre da uno stato si può passare solo ad una cerchia ristretta di altri stati.

2.2 Design dettagliato

2.2.1 Matteo Caruso

Validare le combo di carte

Problema Bisogna validare vari tipi di combo di carte, ognuna con requisiti diversi. Inoltre ogni combo ricompensa il giocatore con un numero di unità diverso.

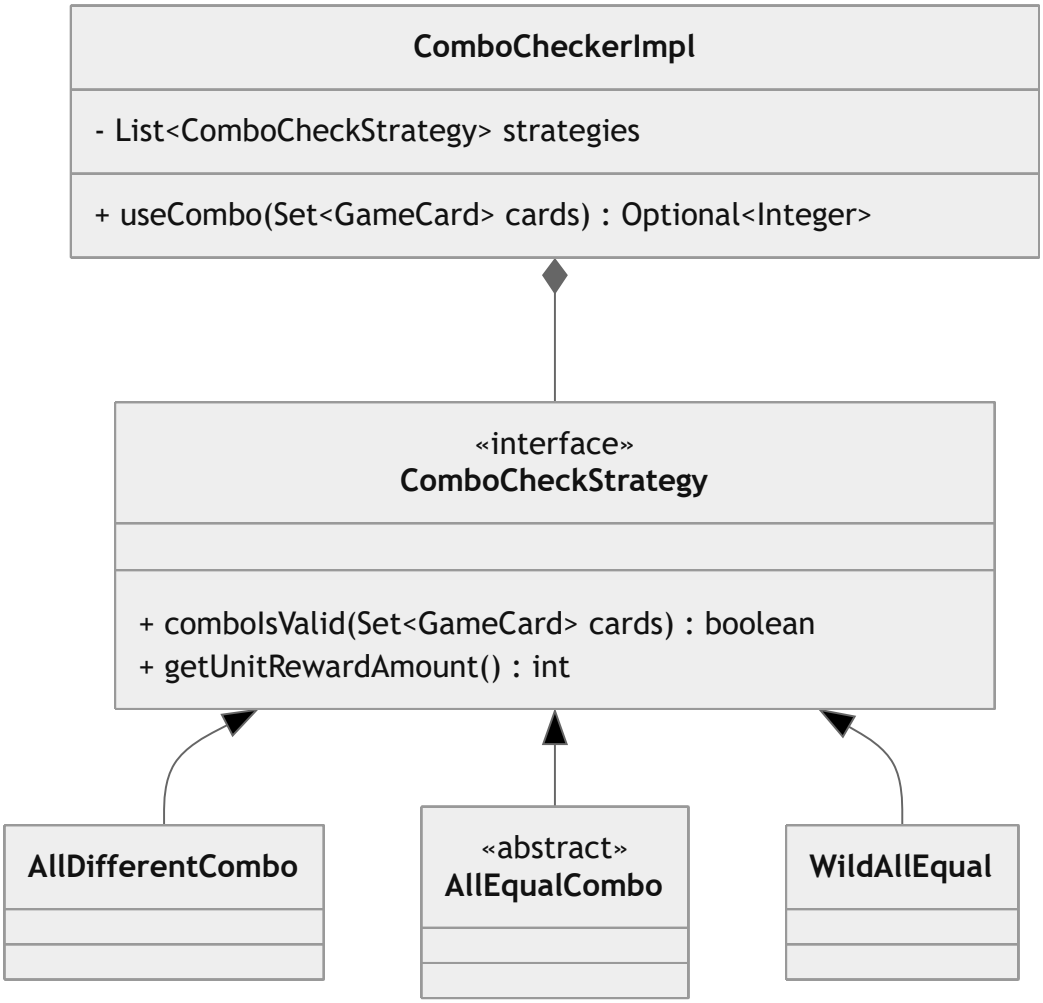

Figura 2.1: UML del pattern Strategy per la validazione delle combo.

Soluzione La validazione delle combo usa il pattern *Strategy*, in cui ogni validatore di combo è una strategia diversa.
Il pattern *Strategy* è più adatto rispeto al pattern *Template Method*, siccome ogni combo ogni validatore di combo differisce molto dagli altri, fatta eccezione dei validatori `AllEqualCombo`.

Validare le `AllEqualCombo`

Problema Bisogna validare le combo di carte dove hanno tutte lo stesso tipo.

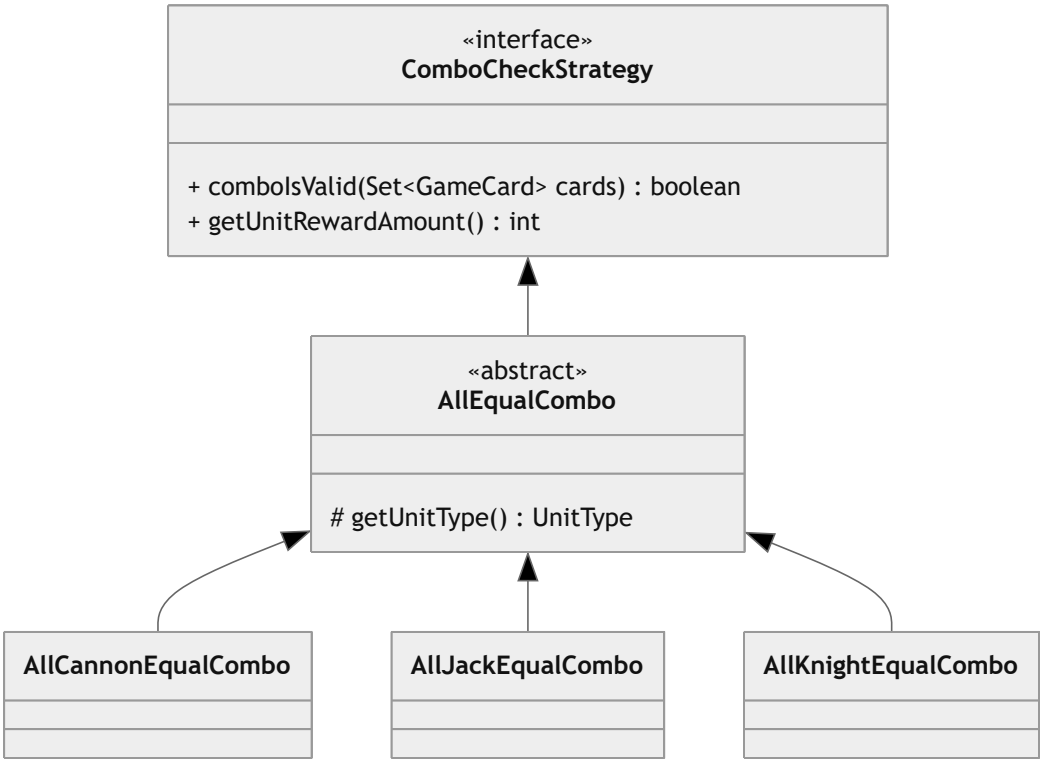

Figura 2.2: UML del pattern Template Method per la validazione delle combo di carte con tipo uguale.

Soluzione Qui è possibile usare il pattern *Template Method*, dove la classe astratta `AllEqualCombo` definisce il template method `comboIsValid` e l'operazione primitiva `getUnitType` [1].
Gli lascia anche la responsabilità di implementare `getUnitRewardAmount`. Le classi che estendono questa classe astratta sono `AllCannonEqualCombo`, `AllJackEqualCombo` e `AllKnightEqualCombo`, che implementano le operazioni primitive sopracitate.

2.2.2 Matteo Ceccarelli

2.2.3 Francesco Sacripante

Creazione logica di due livelli

problema Le prime due fasi hanno due logiche diverse tra loro e tra il resto del gioco

[1] Restituisce `UnitType`, un enumeratore che rappresenta i semi delle carte.

soluzione Scomporre il controller in diversi dipi di controller Risiko è un software molto legato alla visualizzazione del gioco, quindi per favorire il *Separation of Concerns*, il controller è diviso in sotto-controller: `DataAddingContrller` permette di impostare giocatori e la mappa;

`DataRetrieveController` favorisce l'ottenimento di informazioni quali il giocatore corrente; `TurnManager` permette di gestire le fasi del gioco.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Il testing automatizzato è stato realizzato tramite JUnit, focalizzato principalmente sul model, come l'insermineto della mappa, la gestione dei giocatori, la validazione delle combo di carte, e sulla gestione delle fasi di gioco. L'interfaccia grafica è stata testata manualmente durante lo sviluppo del software.

3.2 Note di sviluppo

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

4.2 Difficoltà incontrate e commenti per i docenti

Appendice A

Guida utente

A.1 Avviare la partita

A.2 Rinfori iniziali

A.2.1 Giocare le combo

A.3 Attaccare

A.4 Spostamento finale

Appendice A

Esercitazioni di laboratorio

A.1 matteo.caruso7@studio.unibo.it

- Laboratorio 08: `https://virtuale.unibo.it/mod/forum/discuss.php?d=178723#p247198`
- Laboratorio 09: `https://virtuale.unibo.it/mod/forum/discuss.php?d=179154#p247764`
- Laboratorio 10: `https://virtuale.unibo.it/mod/forum/discuss.php?d=180101#p248784`
- Laboratorio 11: `https://virtuale.unibo.it/mod/forum/discuss.php?d=181206#p250854`

A.2 matteo.ceccarelli@studio.unibo.it

- Laboratorio XX: `https://virtuale.unibo.it`

A.3 franceso.sacripante@studio.unibo.it

- Laboratorio XX: `https://virtuale.unibo.it`