Monopoly

Davide Rossi, Alessio Minniti, Lucas Prati, Martina Ravaioli

18 giugno 2025

Indice

1 Analisi **2**
1.1 Descrizione e requisiti 2
1.2 Modello del dominio 3

2 Design **7**
2.1 Architettura 7

Capitolo 1

Analisi

1.1 Descrizione e requisiti

Il software mira alla costruzione di una versione virtuale del famoso gioco da tavolo "Monopoly". Quest'ultimo è un gioco di società nel quale più giocatori a turno lanciano i dadi e muovono le proprie pedine su un tabellone composto da caselle di cui la maggior parte sono proprietà ovvero dei terreni acquistabili dai giocatori. Lo scopo del gioco è di creare un monopolio aumentando i beni o il denaro posseduto. Durante la partita i giocatori possono acquistare le proprietà capitandovi sopra e pagando una somma alla banca, una volta acquistate il giocatore potrà anche decidere se eventualmente migliorarle. I giocatori si scambiano denaro a vicenda sotto forma di pagamenti, ad esempio, quando un giocatore capita su una proprietà altrui, questi è costretto a pagare a una somma di denaro al proprietario. L'obiettivo è mandare gli altri giocatori in bancarotta e vincere rimanendo l'ultimo giocatore sul tabellone. Il software permetterà di creare e giocare un partita con un determinato numero di giocatori dallo stesso terminale.

Requisiti funzionali

- Il gioco potrà essere giocato su uno stesso dispositivo da un minimo di 2 giocatori ed un massimo di 6. L'utente potrà impostare altre opzioni di configurazione relative al gioco prima di avviare la partita.
- Il giocatore potrà tirare i dadi per far muovere la propria pedina sul tabellone.
- Il giocatore, durante il proprio turno, avrà la possibilità di acquistare la proprietà sulla quale si trova la sua pedina se questa non appartiene

a nessuno. Se la proprietà è già posseduta da altri, il giocatore che vi capita sopra dovrà pagare l'affitto al proprietario per poter continuare a giocare.

- Sul tabellone saranno presenti delle caselle speciali con specifici effetti atti a modificare il corso del turno. Quando un giocatore capita su una di queste caselle speciali si attiverà il relativo effetto.
- Nel corso della partita un giocatore, quando capita sulle sue proprietà, le potrà migliorare spendendo del denaro per posizionare delle case che elevano il costo dell'affitto.
- Durante il proprio turno, se lo desidera, il giocatore potrà vendere alla banca proprietà ed eventuali case precedentemente costruite.

Requisiti non funzionali

- Monopoly dovrà permettere agli utenti di aggiungere e modificare gli effetti speciali del gioco in facilità e senza toccare il codice.
- NON LO INSERIREI salvare su file lo storico dei risultati delle partite precedenti
- Monopoly potrà essere giocato attraverso un'interfaccia grafica che dovrà essere in grado di adattarsi alle dimensioni di vari schermi.
- Monopoly dovrà permettere agli utenti di alterare in facilità la configurazione del gioco (impostazioni della partita, struttura del tabellone e caselle che lo compongono).
- NON LO INSERIREI rendere l'applicazione portabile su più SO

1.2 Modello del dominio

Nella versione da tavolo del gioco, solitamente sono i giocatori a coordinarsi tra di loro nella rotazione dei turni e nel controllare se un compagno di gioco ha perso e va dunque eliminato dalla partita. Questo aspetto del gioco può risultare anche confusionario alle volte, soprattutto in situzioni particolari come ad esempio nel caso in cui un giocatore finisce in prigione e deve rimanerci per un determinato numero di turni e le azioni che può svolgere nel suo turno cambiano. Per modellare questo aspetto è stata inserita un'entità

esterna che chiameremo arbitro (referee) a cui sono delegate tutte le operazioni di coordinazione del gioco e dei suoi partecipanti. Queste operazioni sono:

- Coordinare i turni dei giocatori concedendo la possibilità di giocare, il turno potrà esere passato solo una volta che tutte le azioni obbligatorie sono state eseguite.
- Decretare se un giocatore è eliminato oppure no sulla base della sua situazione finanziaria, se il saldo del portafoglio del giocatore è in negativo perde la partita e tutti i suoi contratti di proprietà ritornano alla banca, disponibili per l'acquisto.
- Decretare se il gioco è finito e determinare il vincitore.

Come è stato detto, a ogni giocatore (Player) viene concesso periodicamente dall'arbitro il proprio turno d'azione sulla base di una rotazione ciclica. Durante il suo turno il giocatore muove sul tabellone (Board) la propria pedina (Pawn) tirando dei dadi (Dices). Il tabellone è composto da una serie di caselle (Tile) disposte su un percorso chiuso. Il giocatore muove la propria pedina saltando un numero di caselle corrispondenti al suo tiro del dado, atterrando così su una nuova casella. Questa casella può essere una proprietà (Property) oppure una casella speciale (Special), e sulla base di ciò cambia radicalmente l'interazione con l'utente. Le proprietà sono caselle alle quali è associato un contratto di proprietà (Title deed). Quando un giocatore capita su una casella di tipo proprietà non ancora in possesso di nessun altro, può richiedere alla banca (Bank) di acquistare il contratto (Title deed) associato alla suddetta proprietà. Dopo aver pagato una somma alla banca e il contratto viene consegnato al giocaatore che da quel momento in poi possiederà la proprietà. Il contratto di proprietà è un documento che descrive tutte le informazioni monetarie relative alla proprietà come il prezzo d'acquisto, prezzo di vendita e le varie opzioni di affitto, il loro costo e le condizioni di applicabilità. La banca possiede tutti i contratti non acquistati.
Se la proprietà su cui è capitato il giocatore era già stata comprata in precedenza da un altro allora si deve pagare l'affitto al proprietario. I giocatori, controllando il contratto di proprietà, concorderrano a quanto ammonta la somma di denaro. Se in un turno successivo all'acquisto il giocatore capita novamente su una casella di sua proprietà può decidere se migliorarla aggiungendo case o alberghi scambiandoli per soldi con la banca. Infine ogni giocatore può rivendere alla banca un contratto di proprietà e ricevere del denaro in cambio. Ogni giocatore ha associato un portafoglio (wallet) contente del denaro. Attraverso il suo portafoglio il giocatore compie tutte le

operazioni di compravendita che comportano un addebito o prelievo di denaro (comprare/vendere contratti, pagare affitti...)
Se il giocatore invece capita su una casella speciale (special) si attiverà un effetto caratteristico della casella (Effect) che avrà ripercussioni sullo svolgimento del gioco (guadagno/perdita denaro, saltare un determinato numero di turni...). In particolare se si capita su una casella Imprevisti (Unexpected) o Probabilità (Probability) l'effetto non è specifico della casella stessa, ma viene letto da un mazzo di carte effetto (Unexpected deck e Probability deck). Il giocatore pesca una carta (Card) da uno dei due mazzi, legge l'effetto della carta e lo attiva.

Parte della difficoltà consisterà nel riadattare un dominio che non nasce intrinsecamente per un progetto software. Monopoly infatti nasce come gioco da tavolo e questo si riflette in molte interazioni e funzionalità, che non sono pensate nell'ottica di un'architettura software (un esempio è quello dato all'inizio della sezione sulla coordinazione fra i giocatori e l'entità arbitro). Molte azioni nel gioco spesso comportano la comunicazione di più entità (l'arbitro decreta il vincitore sia sulla base dei contratti che possiede sia sul suo portafoglio, migliorare una proprietà ha come conseguenza un cambiamento sulla casella ma per attuarlo il giocatore deve interagire con la banca facendo riferimento al contratto di proprietà, un giocatore può passare il turno solo se ha fatto determinate azioni che dipendono non solo dalla casella in cui si trova ma anche dal suo contratto...). Sarà quindi difficile progettare un'architettura software che permetta di rappresentare il dominio del gioco in maniera efficace e riadattabile.

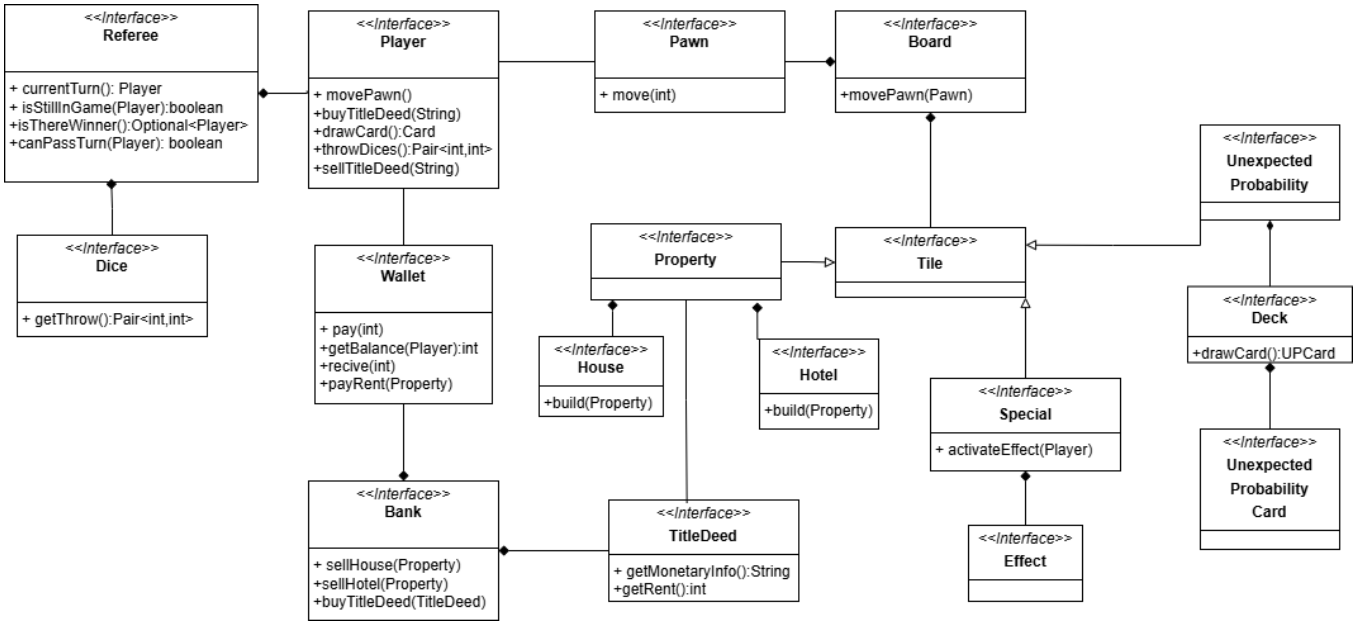

Figura 1.1: Schema UML dell'analisi del problema, con rappresentate le entità principali ed i rapporti fra loro

Capitolo 2

Design

2.1 Architettura

L'architettura di MPoly è basata sul pattern architetturale MVC.
Si è scelto di implementare MVC nella sua forma classica in quanto il gioco ha una modalità di interazione ad eventi: ovvero ogni modifica che avviene sul model è a seguito di un evento generato dall'utente.
L'utente interagisce con l' applicazione facendo scaturire un evento (pressione di un bottone nella view, tasto della tastiera...). Il controller cattura questo evento ed esegue una serie di operazioni specifiche per la sua gestione interagendo con la parte model dell'app, e susseguentemente aggiorna la view mostrando all'utente che cosa è successo.
In riferimento allo schema dell'architettura, è stato predisposto un controller principale denominato GameManager che ha un riferimento a un oggetto che implementa l'interfaccia MainView, sul quale chiamerà l'aggiornamento della view.
In questa versione dell'applicazione la modalità di interazione utente avviene tramite pressione di bottoni della view. Per questo si è deciso di implementare il pattern "Observer": MainViewImpl ha un riferimento al GameController ed è questa classe che notifica il Controller dell' inizio di un evento. Al momento della notifica il controller interroga il model.
A seguito dell'analisi sono stati individuati 3 punti d'entrata per la componente model dell'applicazione, ognuno dei quali si occupa di un macro aspetto del dominio. Nello specifico abbiamo:

- Board, che si occupa di gestire la struttura del tabellone, le caselle con case e alberghi e il movimento delle pedine.
- TurnationManager (che sarebbe il corrispettivo di referee nell'analisi)

che si occupa di gestire l'avvicendarsi dei turni dei giocatori e la fine del gioco.

- Bank, per orchestrare lo scambio di denaro fra i bank account dei giocatori e la compravendita di contratti di proprietà.

Questi punti d' entrata del model offrono delle primitive che incapsulano la logica di funzionamento delle principali azioni che si possono compiere durante il gioco, azioni che sono caratteristiche del gioco stesso Monopoly. Il controller orchestra varie chiamate ai metodi di queste 3 classi del model per far funzionare il gioco. Con questa architettura il model è perfettamente scorporabile e riutilizzabile per costruire un software diverso che risponda allo stesso dominio. L'assenza di un unico punto di entrata per il model previene l'esistenza di una macro classe con eccesiva responsabilità sulla quale dipenderebbe tutto il funzionamento del gioco, facilitando lo sviluppo e l'evoluzione del software.
Il controller coopera con model e view mediante delle interfacce che sono completamente indipendenti dall'implementazione di quest'ultimi. Questo, in particolare, fa sì che le scelte implementative della view non determinino cambiamenti sul controller o sul model in alcun caso rendendo dunque totalmente indipendente e modificabile l'implementazione. In aggiunta, si potrebbe prevedere l'esistenza di uno specifico componente che implementi il pattern observer con il GameController e che chiami i metodi di quest'ultimo permettendo altre modalità di interazione (cattura evento pressione dei tasti del mouse, della tastiera...) Il software prevede anche un menù iniziale di configurazione della partita. Questo menù è a sua volta costituito da una sua architettura MVC più ridotta, modellata tenendo in mente gli stessi principi descritti sovrastante. L'entità GameBuilder è colei che si occupa di instanziare poi le classi dell'MVC principale del software e avviare effettivamente il gioco, costruendo tutti gli oggetti.

riscrivere menu

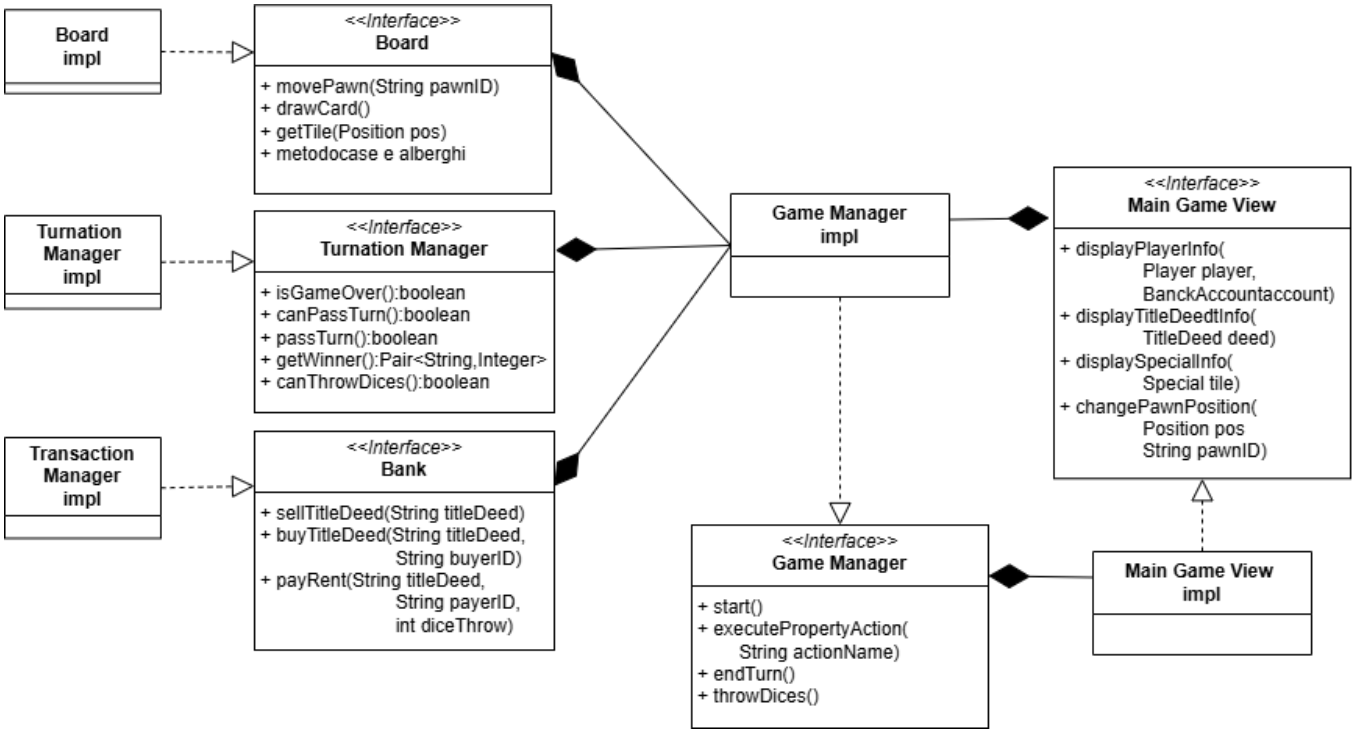

Figura 2.1: Schema UML dell'analisi del problema, con rappresentate le entità principali ed i rapporti fra loro

Cambiare caption