UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CORSO DI INGEGNERIA DEL SOFTWARE
PROF. A.R. FASOLINO - A.A. 2021 - 22

Progetto

Software di un semplice videogioco

di tipo Battaglia Navale

Studenti:

Antonio Paolino M63001394 e-mail
Alfonso Esposito M63001406 e-mail
Fabio De Nigris M60001397 e-mail

INDICE

Sistema software che implementi un semplice videogioco (di tipo Battaglia Navale).

1. Specifiche informali

Si vuole realizzare un sistema software che implementi un semplice videogioco (di tipo Battaglia Navale).
Il videogioco permette ad un insieme di giocatori di sfidarsi in partite in cui ogni giocatore dispone di un segnaposto posizionato in una griglia a due dimensioni e i giocatori devono spostare il segnaposto nelle caselle della griglia, cercando di raggiungere un bersaglio posto sulla stessa griglia e la cui posizione non è nota ai giocatori.
Ogni giocatore è identificato da un nome univoco, ogni partita ha un identificativo univoco (intero positivo) e ad essa possono partecipare da 1 a 3 giocatori. Ogni segnaposto ha un id intero univoco (a partire da 1) e presenta un numero di riga ed un numero di colonna (interi positivi) corrispondenti alla sua posizione corrente.
L'amministratore del gioco può inizializzare una partita specificando la dimensione della griglia, ossia il numero di righe e di colonne della griglia, il numero ed il nome di giocatori, la posizione del bersaglio (ossia il numero di riga e quello di colonna in cui si esso trova), ed il numero di tentativi massimo che complessivamente i giocatori potranno fare per vincere la partita. Tale numero è un multiplo intero del numero di giocatori.
Il sistema deve permettere ai giocatori, a turno, di effettuare una mossa, spostando il proprio segnaposto in un punto della griglia con l'obiettivo di indovinare dove si trova il bersaglio. A tal fine, il giocatore dovrà fornire in input il numero di riga e quello di colonna in cui vuole spostare il segnaposto, dopo di che il sistema verificherà che il numero di tentativi svolti sia inferiore o uguale al numero massimo di tentativi previsti per la partita. Se tale condizione non è soddisfatta, il sistema restituirà un messaggio di errore, altrimenti esso verificherà che la riga e la colonna fornite dal giocatore siano comprese all'interno della griglia (restituendo un messaggio di errore in caso di esito negativo del controllo), e quindi valuterà se il segnaposto ha raggiunto il bersaglio oppure no, confrontando le rispettive coordinate (numero di riga e di colonna). Se il segnaposto ha raggiunto il bersaglio (ossia i numeri di riga e di colonna coincidono), allora il sistema registrerà che la partita si è conclusa e quale è il nome del vincitore, che verrà visualizzato in output. Diversamente, la partita resterà in corso, il numero di tentativi svolti dal giocatore sarà incrementato di un'unità ed il turno passerà al prossimo giocatore.
Il sistema deve permettere a ciascun giocatore di visualizzare, in qualsiasi momento, l'elenco ordinato delle posizioni consecutive in cui ha posto il suo segnaposto durante il gioco.
Il sistema dovrà permettere all'amministratore del gioco di interrogare lo stato di una partita, per conoscere il numero di tentativi già svolti ed i tentativi residui rimasti. Il sistema deve inoltre permettere ad un giocatore di ottenere l'elenco delle partite a cui esso ha partecipato ed il nome del vincitore di ciascuna partita.
Per consentire un utilizzo agevole del sistema da parte dei suoi utenti, si richiede che il sistema sia dotato di interfacce grafiche user-friendly e che per ciascuna operazione di interrogazione dello stato di una partita il tempo di risposta sia non superiore a 5 secondi.

2. Analisi e specifica dei requisiti

2.1 Analisi nomi-verbi

Si vuole realizzare un sistema software che implementi un semplice videogioco (di tipo Battaglia Navale).
Il videogioco permette ad un insieme di giocatori di sfidarsi in partite in cui ogni giocatore dispone di un segnaposto posizionato in una griglia a due dimensioni e i giocatori devono spostare il segnaposto nelle caselle della griglia, cercando di raggiungere un bersaglio posto sulla stessa griglia e la cui posizione non è nota ai giocatori.
Ogni giocatore è identificato da un nome univoco, ogni partita ha un identificativo univoco (intero positivo) e ad essa possono partecipare da 1 a 3 giocatori. Ogni segnaposto ha un id intero univoco (a partire da 1) e presenta un numero di riga ed un numero di colonna (interi positivi) corrispondenti alla sua posizione corrente.
L'amministratore del gioco può inizializzare una partita specificando la dimensione della griglia, ossia il numero di righe e di colonne della griglia, il numero ed il nome di giocatori, la posizione del bersaglio (ossia il numero di riga e quello di colonna in cui si esso trova), ed il numero di tentativi massimo che complessivamente i giocatori potranno fare per vincere la partita. Tale numero è un multiplo intero del numero di giocatori.
Il sistema deve permettere ai giocatori, a turno, di effettuare una mossa, spostando il proprio segnaposto in un punto della griglia con l'obiettivo di indovinare dove si trova il bersaglio. A tal fine, il giocatore dovrà fornire in input il numero di riga e quello di colonna in cui vuole spostare il segnaposto, dopo di che il sistema verificherà che il numero di tentativi svolti sia inferiore o uguale al numero massimo di tentativi previsti per la partita. Se tale condizione non è soddisfatta, il sistema restituirà un messaggio di errore, altrimenti esso verificherà che la riga e la colonna fornite dal giocatore siano comprese all'interno della griglia (restituendo un messaggio di errore in caso di esito negativo del controllo), e quindi valuterà se il segnaposto ha raggiunto il bersaglio oppure no, confrontando le rispettive coordinate (numero di riga e di colonna). Se il segnaposto ha raggiunto il bersaglio (ossia i numeri di riga e di colonna coincidono), allora il sistema registrerà che la partita si è conclusa e quale è il nome del vincitore, che verrà visualizzato in output. Diversamente, la partita resterà in corso, il numero di tentativi svolti dal giocatore sarà incrementato di un'unità ed il turno passerà al prossimo giocatore.
Il sistema deve permettere a ciascun giocatore di visualizzare, in qualsiasi momento, l'elenco ordinato delle posizioni consecutive in cui ha posto il suo segnaposto durante il gioco.
Il sistema dovrà permettere all'amministratore del gioco di interrogare lo stato di una partita, per conoscere il numero di tentativi già svolti ed i tentativi residui rimasti. Il sistema deve inoltre permettere ad un giocatore di ottenere l'elenco delle partite a cui esso ha partecipato ed il nome del vincitore di ciascuna partita.
Per consentire un utilizzo agevole del sistema da parte dei suoi utenti, si richiede che il sistema sia dotato di interfacce grafiche user-friendly e che per ciascuna operazione di interrogazione dello stato di una partita il tempo di risposta sia non superiore a 5 secondi.
LEGENDA:
Classe
Attributo
Funzionalità
Attore
Classe-Attore

2.2 Revisione dei requisiti

1. Il videogioco deve permettere ad un insieme di giocatori di sfidarsi.
2. Ogni giocatore dispone di un segnaposto.
3. Ogni segnaposto è posizionato in una griglia a due dimensioni.
4. Ogni giocatore può spostare il segnaposto nelle caselle della griglia.
5. Ogni giocatore dev'essere identificato da un nome.
6. Ogni giocatore deve avere un nome univoco.
7. Ogni partita ha un identificativo univoco positivo.
8. Ogni partita deve ammettere un numero di giocatori da 1 a 3.
9. L' Amministratore deve specificare il numero dei giocatori durante la fase di inizializzazione di una partita.
10. Ogni segnaposto deve avere un id univoco maggiore di 0.
11. Ogni segnaposto deve presentare un numero di riga e di colonna.
12. Il Sistema deve permettere all' Amministratore del gioco di inizializzare una partita.
13. L' Amministratore deve specificare la dimensione della griglia durante la fase di inizializzazione di una partita.
14. L' Amministratore deve specificare il nome dei giocatori durante la fase di inizializzazione di una partita.
15. L' Amministratore deve specificare la posizione del bersaglio durante la fase di inizializzazione di una partita.
16. L' Amministratore deve specificare il numero di tentativi massimi che i giocatori possono effettuare durante la fase di inizializzazione di una partita.
17. Il numero di tentativi massimi che i giocatori possono effettuare dev'essere un multiplo intero del numero di giocatori.
18. Il Sistema deve permettere ai giocatori di effettuare una mossa.
19. Il Sistema non deve permettere ad un giocatore di effettuare più mosse consecutive.
20. Il Sistema deve permettere di spostare il segnaposto all'interno della griglia.
21. Ogni giocatore deve fornire in input il numero di riga e di colonna in cui vuole spostare il segnaposto.
22. Il Sistema dovrà tener conto del numero di mosse che effettua ogni giocatore.
23. Il sistema dovrà verificare che il numero di tiri di un singolo giocatore non superi il massimo di numero di tiri consentiti.
24. Il Sistema dovrà verificare che la riga e la colonna fornite dal giocatore siano comprese all'interno della griglia.
25. Il Sistema mostrerà un messaggio di errore se il numero di tentativi è superiore al massimo consentito.
26. Il Sistema mostrerà un messaggio di errore se la riga o la colonna fornite dal giocatore non sono comprese all'interno della griglia.
27. Il Sistema dovrà considerare terminata una partita quando il segnaposto avrà raggiunto il bersaglio.
28. Il Sistema deve permettere a ciascun giocatore di visualizzare l'elenco ordinato delle posizioni consecutive in cui ha posto il segnaposto.
29. Il Sistema deve permettere all' Amministratore del gioco di interrogare lo stato della partita mostrando il numero di tentativi effettuati e rimanenti di ogni giocatore.
30. Il Sistema deve incrementare il numero di tentativi di un'unità se il segnaposto non ha raggiunto il bersaglio nel turno corrente.

31. Il Sistema deve far passare il turno al prossimo giocatore se il segnaposto non ha raggiunto l'obiettivo nel turno corrente.
32. Il Sistema deve permettere ad un giocatore di ottenere l'elenco delle partite a cui esso ha partecipato e il nome del vincitore di ciascuna di esse.
33. Al termine di una partita il sistema mostra a video il nome del vincitore.
34. Il Sistema deve essere dotato di interfacce grafiche user-friendly.
35. Ciascuna operazione di interrogazione dello stato di una partita il tempo di risposta sia non superiore a 5 secondi.

2.3 Glossario dei termini

Termine	Descrizione	Sinonimi
Giocatore	Tipologia di utente che partecipa ad una partita.	
Amministratore	Tipologia di utente che inizializza e amministra una o più partite.	
Sistema	Entità che garantisce il corretto funzionamento del gioco.	
Griglia	Campo da gioco dove viene svolta la partita.	
Segnaposto	Puntatore assegnato ai giocatori per colpire il bersaglio.	
Bersaglio	Obiettivo del gioco, se colpito dal segnaposto, indica la fine di una partita.	

2.4 Classificazione dei requisiti

2.4.1 Requisiti funzionali

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RF01	Il Sistema deve permettere all'Amministratore di inizializzare una partita.	12
RF02	Per inizializzare una partita, l'Amministratore deve specificare la dimensione della griglia, il nome e il numero dei giocatori, la posizione del bersaglio e il numero di tentativi massimi consentiti.	9, 13, 14, 15, 16
RF03	Il Sistema deve permettere ai giocatori di effettuare una mossa.	18

RF04	Il Sistema non deve permettere ad un giocatore di effettuare più mosse consecutive.	19
RF05	Il Sistema deve permettere al giocatore di spostare il segnaposto all'interno della griglia.	4,20
RF06	Ogni giocatore deve fornire in input il numero di riga e di colonna in cui vuole spostare il segnaposto.	21
RF07	Il Sistema dovrà tener conto del numero di mosse che effettua ogni giocatore.	22
RF08	Il Sistema dovrà verificare che il numero di tiri di un singolo giocatore non superi il massimo di numero di tiri consentiti.	23
RF09	Il Sistema dovrà verificare che la riga e la colonna fornite dal giocatore siano comprese all'interno della griglia.	24
RF10	Il Sistema mostrerà un messaggio di errore se il numero di tentativi è superiore al massimo consentito.	25
RF11	Il Sistema mostrerà un messaggio di errore se la riga o la colonna fornite dal giocatore non sono comprese all'interno della griglia.	26
RF12	Una partita può considerarsi conclusa se il segnaposto ha raggiunto il bersaglio.	27
RF13	Il Sistema deve permettere a ciascun giocatore di visualizzare l'elenco ordinato delle posizioni consecutive in cui ha posto il suo segnaposto durante il gioco.	28
RF14	Il Sistema deve permettere all' Amministratore del gioco di interrogare lo stato della partita mostrando il numero di tentativi effettuati e rimanenti di ogni giocatore.	29
RF15	Il Sistema deve incrementare il numero di tentativi di un'unità se il segnaposto non ha raggiunto il bersaglio nel turno corrente.	30
RF16	Il Sistema deve far passare il turno al prossimo giocatore se il segnaposto non ha raggiunto l'obiettivo nel turno corrente.	31
RF17	Il Sistema deve permettere ad un giocatore di ottenere l'elenco delle partite a cui esso ha partecipato e il nome del vincitore di ciascuna di esse.	32
RF18	Al termine di una partita il sistema mostra a video il nome del vincitore.	33

2.4.2 Requisiti sui dati

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RD01	Ogni giocatore deve disporre di un segnaposto.	2
RD02	Ogni segnaposto dev'essere posizionato in una griglia a due dimensioni.	3
RD03	Ogni giocatore dev'essere identificato da un nome.	5
RD04	Ogni giocatore deve avere un nome univoco.	6
RD05	Ogni partita deve avere un identificativo univoco positivo.	7
RD06	Ogni partita deve ammettere un numero di giocatori da 1 a 3.	8
RD07	Ogni segnaposto deve avere un id univoco maggiore di 0.	10
RD08	Ogni segnaposto deve presentare un numero di riga e di colonna.	11
RD09	Per inizializzare una partita, l'amministratore deve specificare la dimensione della griglia, il nome e il numero dei giocatori, la posizione del bersaglio e il numero di tentativi massimi consentiti.	9, 13, 14, 15, 16
RD10	Il numero di tentativi massimi che i giocatori possono effettuare è un multiplo intero del numero di giocatori.	17

2.4.3 Vincoli / Altri requisiti

ID	Requisito	Origine (n. frase dei requisiti revisionati)
V01	Ogni giocatore può partecipare ad una sola partita per volta.	
RNF01	Il Sistema deve essere dotato di interfacce grafiche user-friendly.	34
RNF02	Ciascuna operazione di interrogazione dello stato di una partita il tempo di risposta sia non superiore a 5 secondi.	35

2.5 Modellazione dei casi d'uso

2.5.1 Attori e casi d'uso

Attori Primari:

- Amministratore
- Giocatore

Casi d'uso:

- UC1: Inizializza partita.
- UC2: Effettua mossa.
- UC3: Visualizza elenco posizioni.
- UC4: Interroga stato partita.
- UC5: Mostra elenco partite.

Casi d'uso di inclusione:

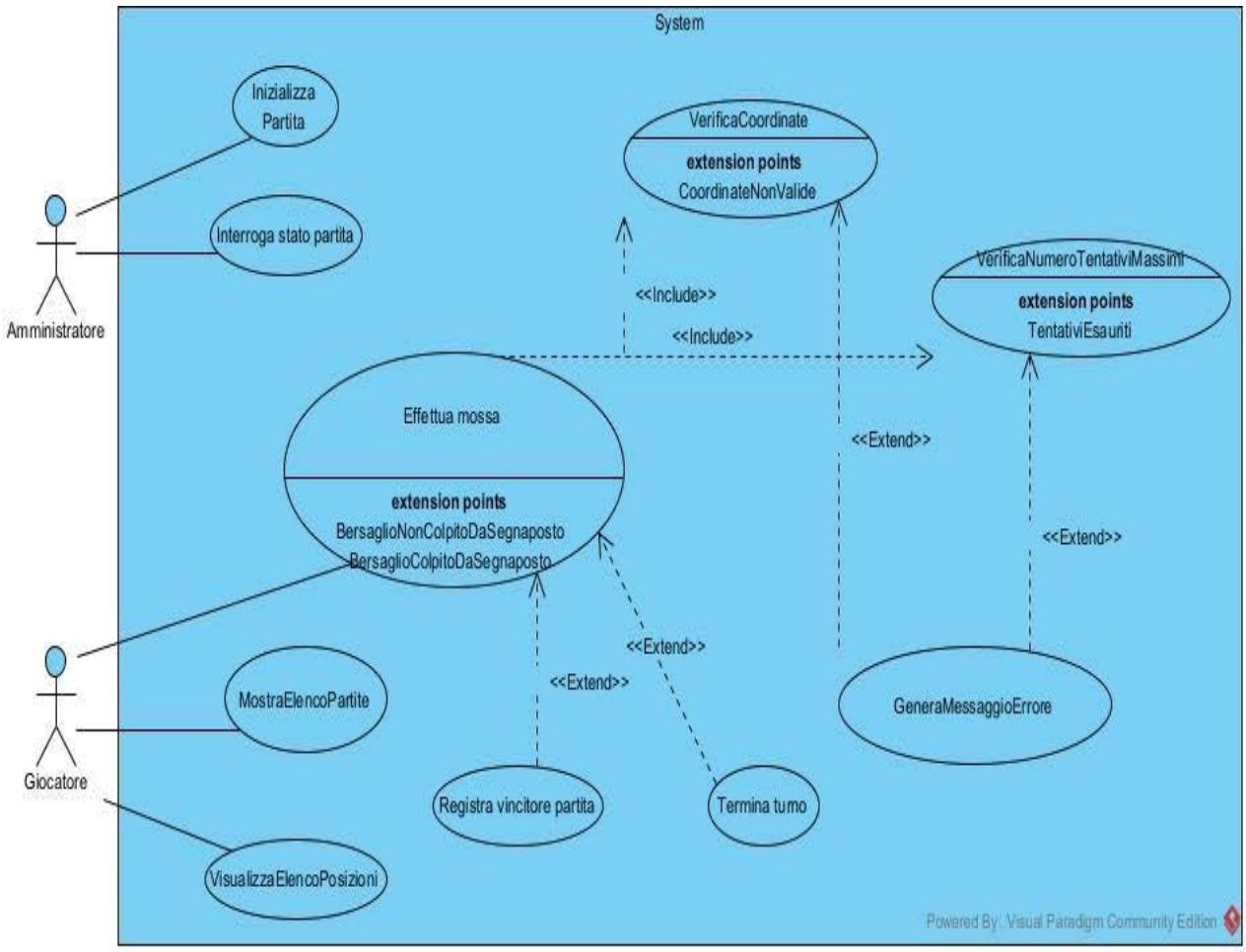
- UC6: VerificaCoordinate.
- UC7: VerificaNumeroTentativiMassimo.

Casi d'uso di estensione:

- UC8: GeneraMessaggioErrore.
- UC9: RegistraVincitorePartita.
- UC10: TerminaTurno.

Caso d'uso	Attori Primari	Attori Secondari	Incl./Ext.	Requisiti Corrispondenti
UC1: Inizializza partita.	Amministratore	-	-	RF01, RF02
UC2: Effettua mossa	Giocatore	-	**Include** VerificaCoordinate e VerificaNumeroTentativiMassimi. **Estende** GeneraMessaggioErrore RegistraVincitorePartita TerminaTurno	RF03, RF04, RF05, RF06, RF07, RF12, RF15, RF16
UC3: Visualizza elenco posizioni.	Giocatore	-	-	RF13
UC4: Interroga stato partita.	Amministratore	-	-	RF14
UC5: Mostra elenco partite.	Giocatore	-	-	RF17
UC6: VerificaCoordinate.	Giocatore	-	**Incluso in** Effettua mossa	RF09
UC7: VerificaNumeroTentativiMassimo.	Giocatore	-	**Incluso in** Effettua mossa	RF08
UC8: GeneraMessaggioErrore.	Giocatore	-	**Estensione di** VerificaCoordinate e VerificaNumeroTentativiMassimo.	RF10, RF11
UC9: RegistraVincitorePartita	Giocatore	-	**Estensione di** Effettua mossa	RF18
UC10: TerminaTurno	Giocatore	-	**Estensione di** Effettua mossa	RF16

2.6 Diagramma dei casi d'uso



2.7 Scenari

Caso d'uso:	Inizializza partita
Attore primario	Amministratore
Attore secondario	-
Descrizione	L'Amministratore crea una nuova partita specificando i giocatori e le regole.
Pre-Condizioni	-
Sequenza di eventi principale	1. Il caso d'uso inizia quando l'amministratore inizializza una nuova partita. 2. L'amministratore specifica la dimensione della griglia, il numero e il nome dei giocatori. 2.1. Se il nome dei giocatori è uguale, il sistema restituisce un errore. 2.2. Se il numero dei giocatori è minore di uno o maggiore di tre, il sistema restituisce un errore. 3. L'amministratore specifica la posizione del bersaglio. 3.1. Se la posizione non è compresa nella griglia, il sistema restituisce un errore.

	4. L'amministratore specifica il numero di tentativi massimo consentiti per ogni giocatore. 4.1. Se il numero di tentativi massimo consentiti non è un multiplo del numero di giocatori, il sistema restituisce un errore. 5. Il sistema fornisce ad ogni giocatore un segnaposto per partecipare al gioco.
Post-Condizioni	Viene creata una nuova partita, con una griglia, un bersaglio posizionato su di essa inoltre viene assegnato ad ogni Giocatore un Segnaposto.
Casi d'uso correlati	*nessuno*
Sequenza di eventi alternativi	-

Caso d'uso:	**Effettua mossa**
Attore primario	Giocatore
Attore secondario	-
Descrizione	Il Sistema permette al giocatore di effettuare una mossa, spostando il suo segnaposto all'interno della griglia.
Pre-Condizioni	Il Sistema ha assegnato il turno al Giocatore che deve effettuare una mossa.
Sequenza di eventi principale	1. Il caso d'uso inizia quando il Giocatore deve effettuare una mossa. 2. Il Giocatore inserisce in input il numero di riga e colonna sulla griglia in cui posizionare il proprio segnaposto. 3. Il Sistema verifica che il numero di tentativi effettuati dal Giocatore non sia superiore al numero di tentativi massimi previsti per la partita. 4. Il Sistema verifica che la riga e la colonna inserite dal giocatore siano comprese all'interno della griglia. 5. Il Sistema sposta il Segnaposto del Giocatore nella posizione da lui fornita. 6. Il Sistema verifica se il Segnaposto ha raggiunto il Bersaglio. 6.1. Se tale condizione è verificata, il sistema registra la fine della partita e mostra in output il nome del vincitore.
Post-Condizioni	-
Casi d'uso correlati	Registra vincitore partita e Termina turno.
Sequenza di eventi alternativi	Al punto 3, se il numero di tentativi effettuati dal Giocatore è superiore al numero di tentativi massimo 3.1 Il Sistema mostra un messaggio di errore e passa il turno al Giocatore successivo. Al punto 4, se il numero di riga e di colonna inserito dal Giocatore non rientra nella Griglia 4.1 Il Sistema mostra un messaggio di errore e chiede al Giocatore di inserire le nuove coordinate. Al punto 6, se il Segnaposto non ha raggiunto il Bersaglio

	6.2 Il sistema incrementa di un un'unita il numero di tentativi svolti dal giocatore e passa il turno al Giocatore successivo.

Caso d'uso:	**Visualizza elenco posizioni**
Attore primario	Giocatore
Attore secondario	-
Descrizione	Il sistema permette ai giocatori, in ogni momento, di visualizzare l'elenco ordinato delle posizioni del proprio segnaposto.
Pre-Condizioni	Il giocatore ha effettuato almeno una mossa nel corso della partita.
Sequenza di eventi principale	1. Il caso d'uso inizia quando il giocatore decide di visualizzare l'elenco delle posizioni del suo segnaposto. 2. Il sistema restituisce, in output, al giocatore l'elenco richiesto.
Post-Condizioni	
Casi d'uso correlati	*nessuno*
Sequenza di eventi alternativi	-

Caso d'uso:	**Interroga stato partita**
Attore primario	Amministratore
Attore secondario	-
Descrizione	L' Amministratore può in qualunque momento interrogare lo stato della partita.
Pre-Condizioni	L'Amministratore deve aver inizializzato una partita.
Sequenza di eventi principale	1. Il caso d'uso inizia quando l'amministratore interroga lo stato di una partita. 2. Il sistema mostra il numero di tentativi già svolti e tentativi rimasti per ogni giocatore.
Post-Condizioni	
Casi d'uso correlati	*nessuno*
Sequenza di eventi alternativi	-

Caso d'uso:	**Mostra elenco partite**
Attore primario	Giocatore
Attore secondario	-

Descrizione	Il Giocatore può in qualunque momento chiedere al sistema di mostrare a video l'elenco delle partite e dei rispettivi vincitori.
Pre-Condizioni	Il sistema deve verificare che il giocatore che effettua la richiesta abbia partecipato ad almeno una partita.
Sequenza di eventi principale	1. Il caso d'uso inizia quando il Giocatore interroga il sistema per visualizzare l'elenco delle partite a cui ha partecipato. 2. Il sistema mostra le partite ed i rispettivi vincitori a cui ha partecipato il giocatore che ha effettuato la richiesta.
Post-Condizioni	-
Casi d'uso correlati	*nessuno*
Sequenza di eventi alternativi	-

2.8 Diagramma delle classi

Diagramma delle classi di analisi.

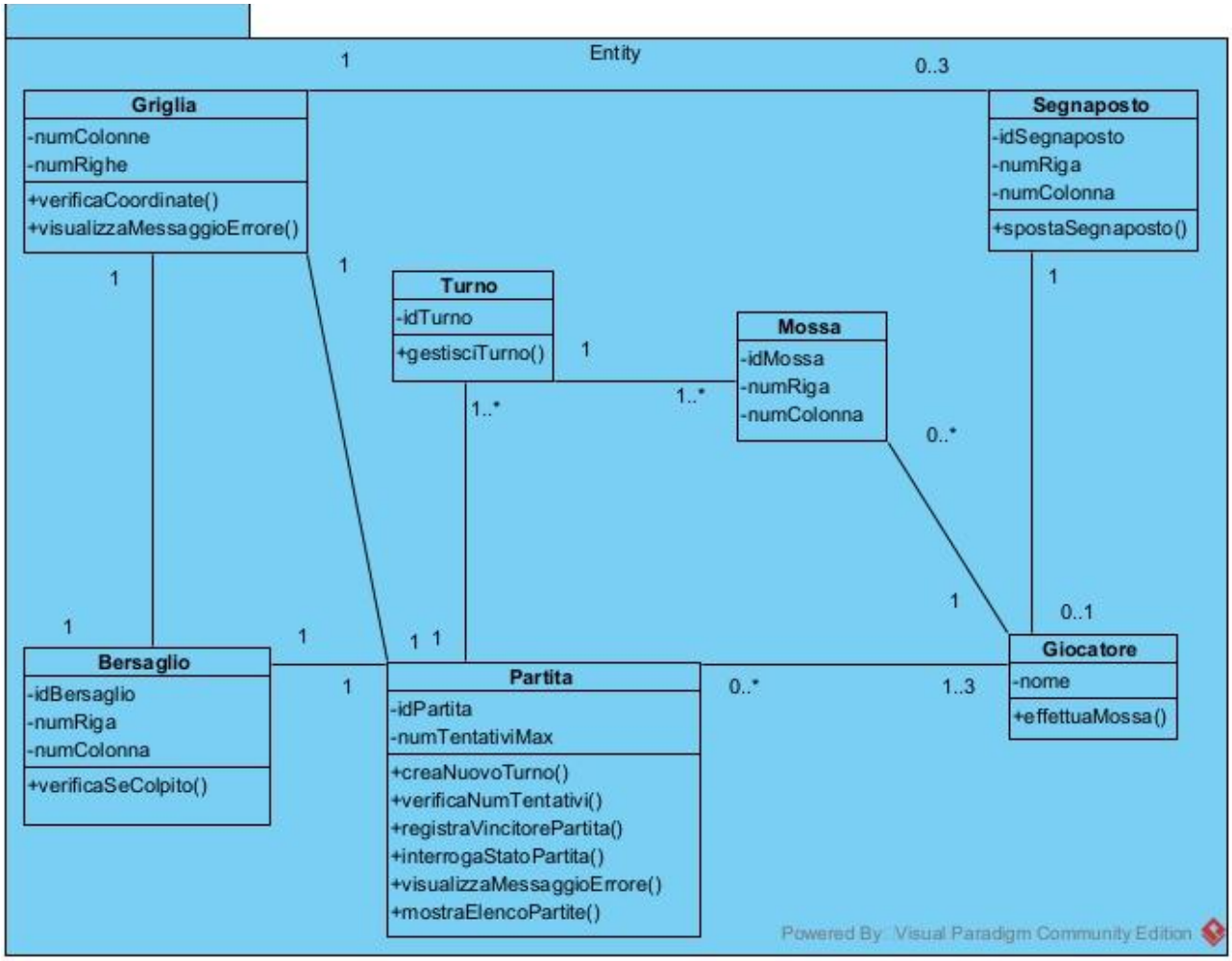
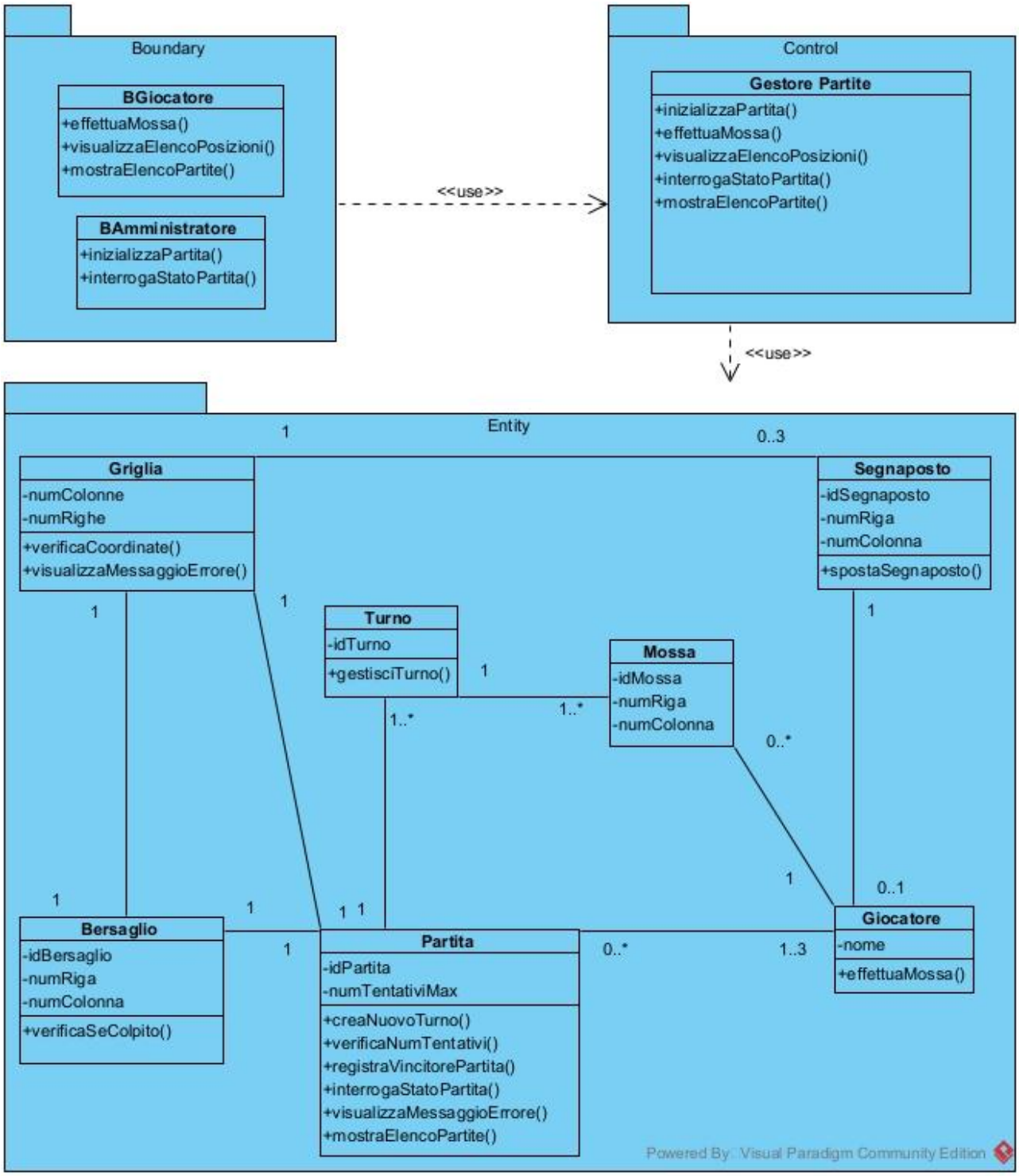
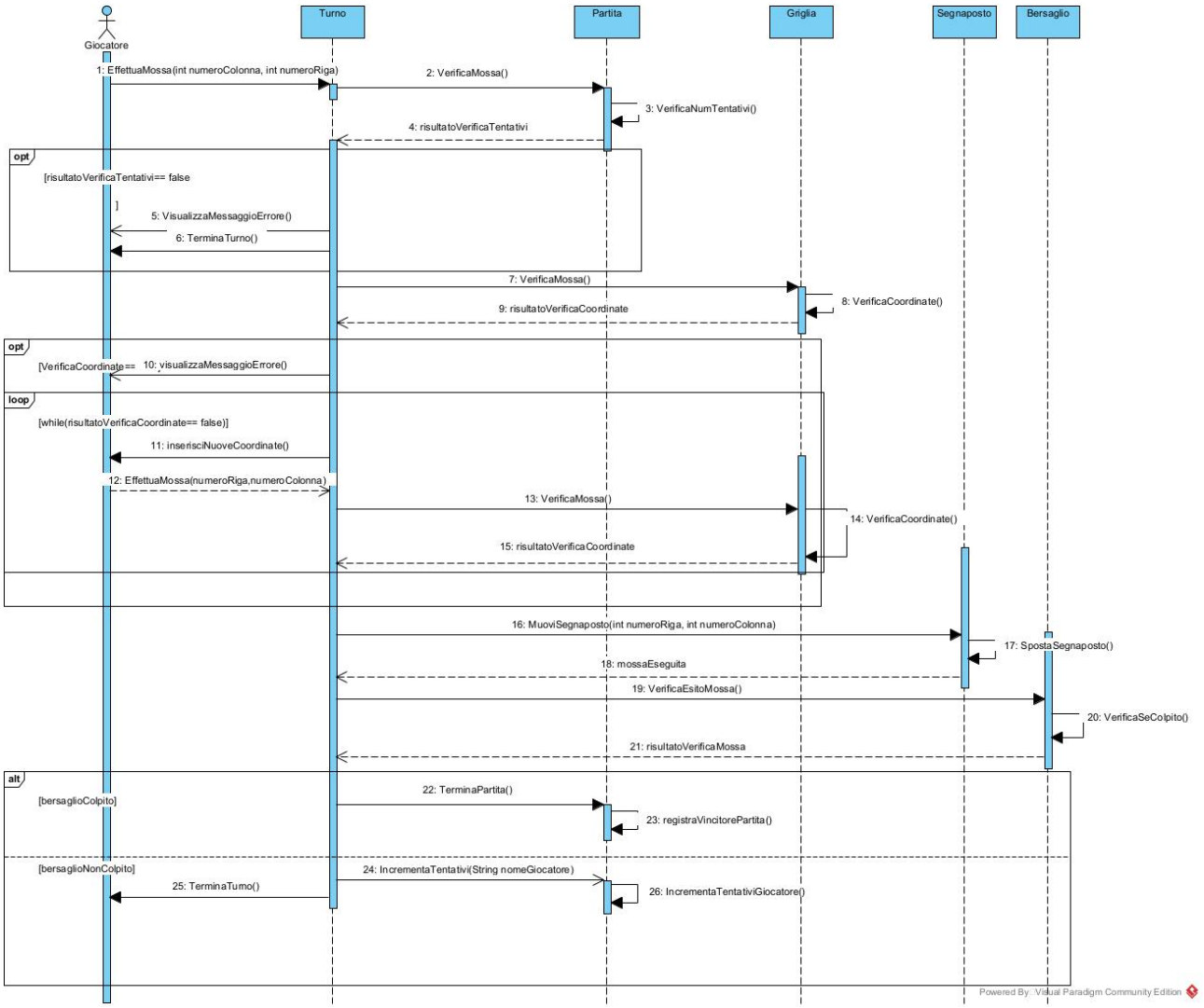


Diagramma delle classi raffinato (con classi Control e Boundary).

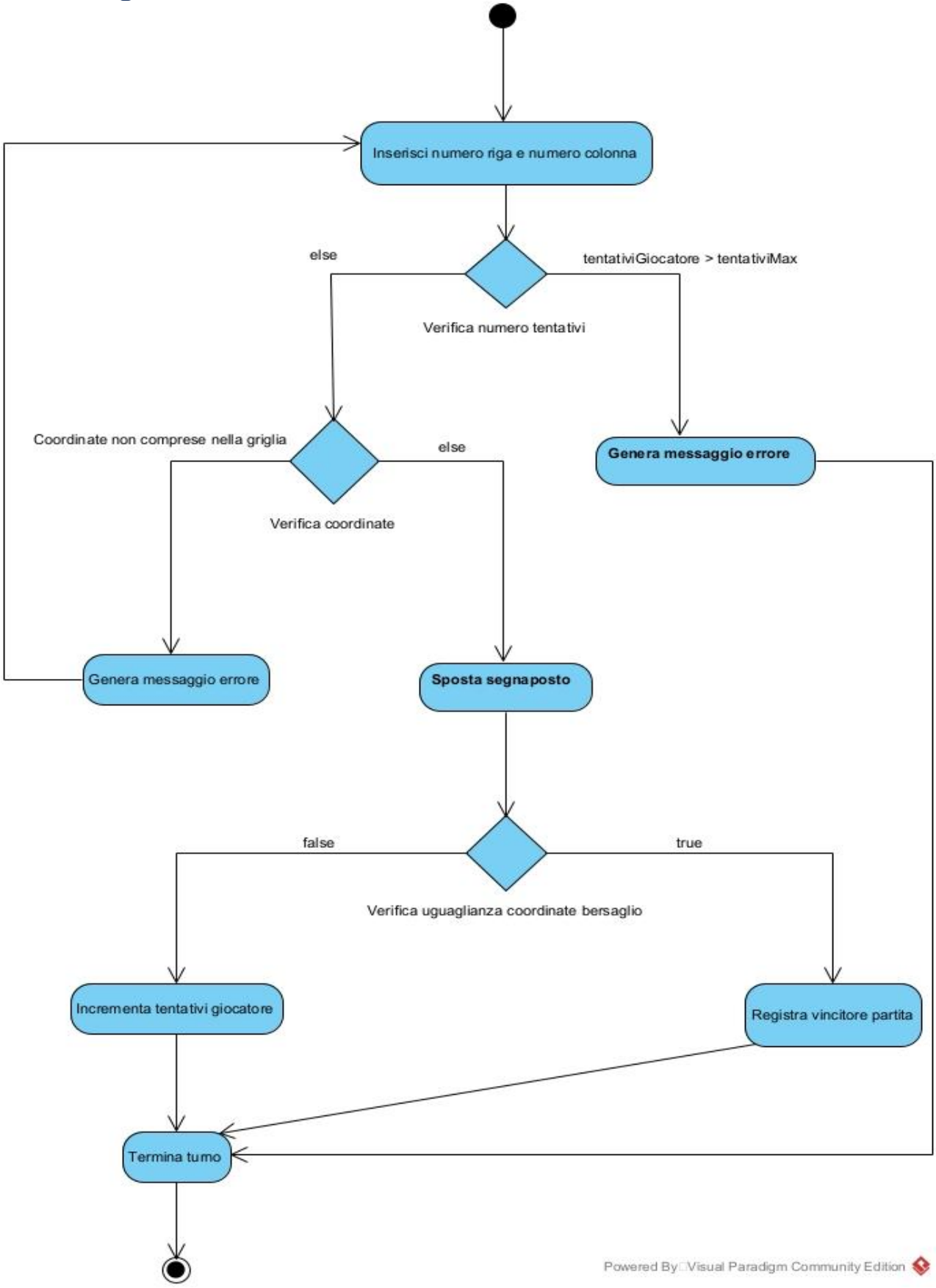


2.9 Diagrammi di sequenza

Diagrammi di sequenza di analisi per il caso d'uso "EffettuaMossa".



2.10 Diagramma di attività



2.11 Verifica della completezza dei requisiti

Legenda: UCD = Use Case Diagram, CD = Class Diagram, SD = Sequence Diagram

- **RF01** è modellato nell'UCD con l'attore "Amministratore" e con il caso d'uso UC1
- **RF02** è modellato nell'UCD con l'attore "Amministratore" e con il caso d'uso UC1
- **RF03** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF04** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF05** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF06** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF07** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC7
- **RF08** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC7
- **RF09** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC6
- **RF10** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC8
- **RF11** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC8
- **RF12** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF13** è modellato nell'UCD con l'attore "Amministratore" e con il caso d'uso UC3
- **RF14** è modellato nell'UCD con l'attore "Amministratore" e con il caso d'uso UC4
- **RF15** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF16** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC2
- **RF17** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC5
- **RF18** è modellato nell'UCD con l'attore "Giocatore" e con il caso d'uso UC9
- **RD01** è modellato nel CD con la cardinalità dell'associazione tra la classe "Segnaposto" e la classe "Giocatore"
- **RD02** è modellato nel CD con la classe "Griglia"
- **RD03 e RD04** sono modellati nel CD con l'attributo "nome" della classe "Giocatore"
- **RD05** è modellato nel CD con l'attributo "idPartita" della classe "Partita"
- **RD06** è modellato nel CD con la cardinalità dell'associazione tra la classe "Partita" e la classe "Segnaposto"
- **RD07** è modellato nel CD con l'attributo "idSegnaposto" della classe "Segnaposto"
- **RD08** è modellato nel CD con gli attributi "numRiga" e "numColonna" della classe "Segnaposto"
- **RD09** è modellato nel CD con le classi "Partita", "Griglia", "Giocatore" e "Bersaglio"
- **RD10** è modellato nel CD con l'attributo "numTentativiMax" della classe "Segnaposto"

3. Stima dei costi

Effettua Mossa

	VALORE	SEMPLICE	MEDIO	COMPLESSO	TOT
NILF	1			6	6
NEIF	1	4			4
NEI	2	3			6
NEO	2		5		10
NEQ	0				

UFP =	26
LLOC/FP =	1368

NILF: la lista delle mosse viene creata e aggiornata continuamente dal sistema, la identifichiamo come ILF. [1 complesso]

NEIF: la partita viene salvata sul database al suo termine. [1 semplice]

NEI: numero riga e numero colonna [2 semplici]

NEO: messaggio di errore e visualizza nome vincitore [2 medi]

FATTORI CORRETTIVI	
COMUNICAZIONE DATI	0
DISTRIBUZIONE ELABORAZIONE	0
PRESTAZIONI	4
UTILIZZO INTENSIVO CONFIGURAZIONE	2
FREQUENZA DELLE TRANSAZIONI	3
INSERIMENTO DATI INTERATTIVO	3
EFFICIENZA PER L'UTENTE FINALE	4
AGGIORNAMENTO INTERATTIVO	4
COMPLESSITA' ELABORATIVA	1
RIUSABILITA'	3
FACILITA' INSTALLAZIONE	3
FACILITA' GESTIONE OPERATIVA	1
MOLTECIPLITA' DI SITI	0
FACILITA' DI MODIFICA	3
	31

FP=	24,96
JAVA =	1.323

4. Piano di test funzionale

PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ "*EffettuaMossa*".

NUMERO RIGA	NUMERO COLONNA
• Numero intero >= 0 • Numero decimale [ERROR] • Numero < 0 [ERROR] • Stringa alfanumerica [ERROR] • Stringa con caratteri [ERROR]	• Numero intero >= 0 • Numero decimale [ERROR] • Numero < 0 [ERROR] • Stringa alfanumerica [ERROR] • Stringa con caratteri [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $5 * 5 = 25$
Introduciamo i vincoli [ERROR].
Il numero di test da eseguire per testare singolarmente i vincoli è 8 (4 per Numero Riga, 4 per Numero Colonna).
Il numero di test risultante è: (1*1) +8 = 9

PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ "*EffettuaMossa*".

NUMERO RIGA	NUMERO COLONNA	NUMERO TENTATIVI
• Numero riga <= Numero riga griglia • Numero riga > Numero riga griglia [ERROR] • Numero riga = Numero riga bersaglio • Numero riga != Numero riga bersaglio [ERROR]	• Numero colonna <= Numero colonna griglia • Numero colonna > Numero colonna griglia [ERROR] • Numero colonna = Numero colonna bersaglio • Numero colonna != Numero colonna bersaglio [ERROR]	• Numero tentativi < Numero tentativi massimo • Numero Tentativi >= Numero tentativi massimo [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 4 * 2 = 32$
Introduciamo i vincoli [ERROR].
Il numero di test da eseguire per testare singolarmente i vincoli è 9 (4 per Numero Riga, 4 per Numero Colonna, 1 per Numero Tentativi).
Il numero di test risultante è: (2*2*1) + 9 = 13

4.1 Test suite

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese
ID1	Tutti input validi.	Numero riga valido. Numero colonna valido.	La verifica del numero di tentativi effettuati e delle coordinate è andata a buon fine.	{Numero riga: “3”} {Numero colonna: “5”}	Mossa effettuata con successo.	Il giocatore ha spostato il suo segnaposto.
ID2	Numero riga decimale.	Numero riga decimale [ERROR] Numero colonna valido.		{Numero riga: “2.8”} {Numero colonna: “5”}	Formato del numero riga errato.	
ID3	Numero riga < 0	Numero riga negativo [ERROR] Numero colonna valido.		{Numero riga: “-3”} {Numero colonna: “5”}	Numero riga non ammesso.	
ID4	Stringa alfanumerica	Stringa riga alfanumerica [ERROR] Numero colonna valido.		{Numero riga: “aa2”} {Numero colonna: “5”}	Formato del numero riga errato.	
ID5	Stringa con caratteri	Stringa riga di caratteri [ERROR] Numero colonna valido.		{Numero riga: “abb@”} {Numero colonna: “5”}	Formato del numero riga errato.	
ID6	Numero decimale	Numero riga valido.		{Numero riga: “3”} {Numero colonna: “5,2”}	Formato del numero	

		Numero colonna decimale [ERROR].			colonna errato.	
ID7	Numero < 0	Numero riga valido. Numero colonna negativo [ERROR].		{Numero riga: “3”} {Numero colonna: “-5”}	Numero colonna non ammesso.	
ID8	Stringa alfanumerica	Numero riga valido. Stringa colonna alfanumerico [ERROR].		{Numero riga: “3”} {Stringa colonna: “ab5”}	Formato del numero colonna errato.	
ID9	Stringa con caratteri	Numero riga valido. Stringa colonna di caratteri [ERROR].		{Numero riga: “3”} {Stringa colonna: “abc”}	Formato del numero colonna errato.	
ID10	Coordinate del bersaglio individuate	Numero riga <= Numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna = Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		{Numero riga: “2”} {Numero colonna: “2”} {Numero tentativo: “2”} Nota: Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”} Grandezza griglia: {“4”,”4”}	Il nome del vincitore viene mostrato ai giocatori e termina la partita	Salvataggio delle entità nel database
ID11	Coordinate del bersaglio non individuate	Numero riga <= Numero riga griglia		{Numero riga: “3”} {Numero colonna: “3”} {Numero tentativo: “2”}	Il giocatore termina il proprio turno	Incremento del numero di tentativi del giocatore

		Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		Nota: Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”} Grandezza griglia: {“4”,”4”}		
ID12	Numero riga non compreso nella griglia	Numero riga > Numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		{Numero riga: “5”} {Numero colonna: “3”} {Numero tentativo: “2”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Coordinate non valide”	Il giocatore dovrà reinserire nuovamente le coordinate
ID13	Numero colonna non compreso nella griglia	Numero riga <= Numero riga griglia Numero colonna > Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio		{Numero riga: “3”} {Numero colonna: “5”} {Numero tentativo: “2”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Coordinate non valide”	Il giocatore dovrà reinserire nuovamente le coordinate

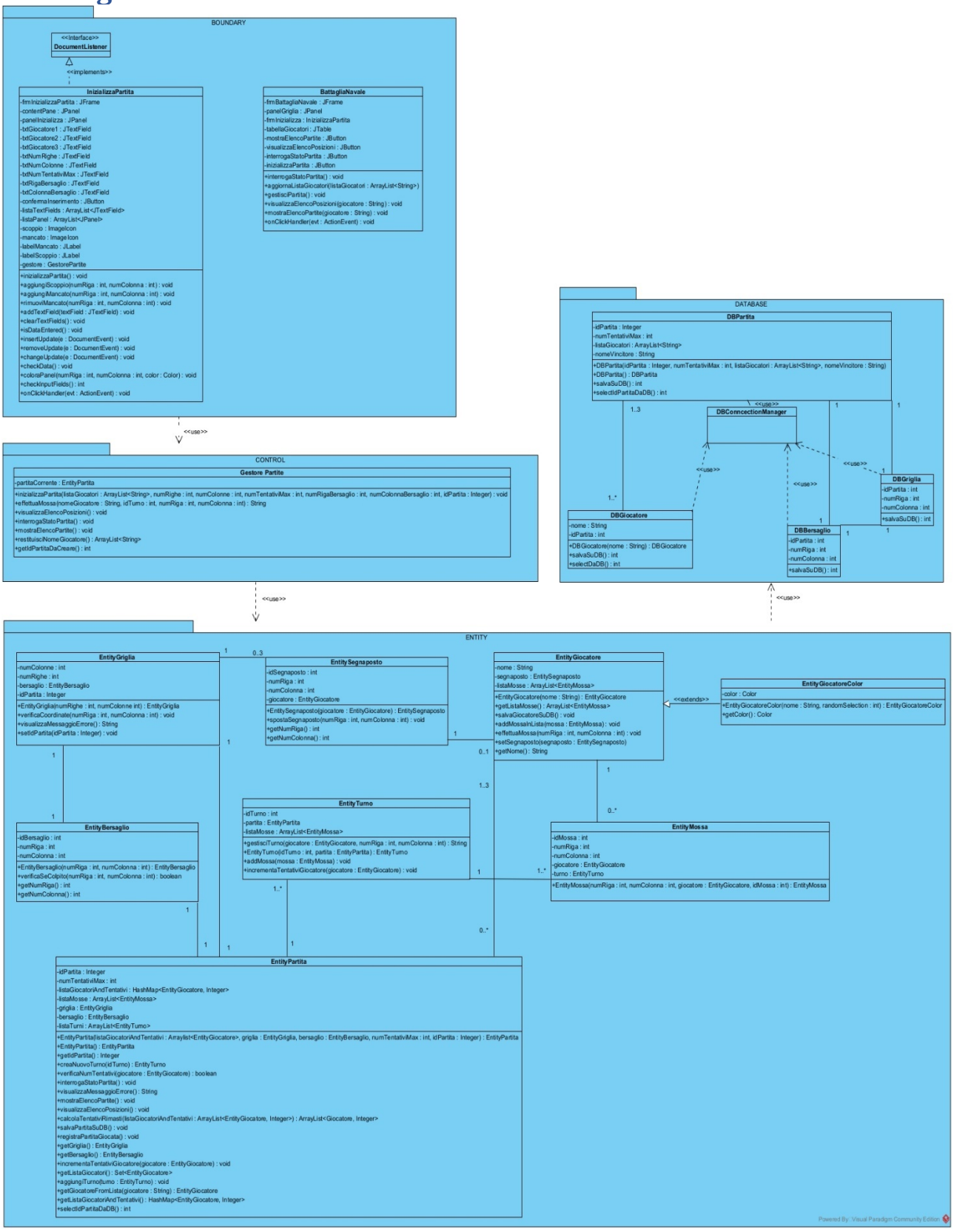
		Numero tentativi < Numero tentativi massimo				
ID14	Numero riga non compreso nella griglia e Numero tentativi maggiore o uguale al Numero tentativi massimo	Numero riga > numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi >= Numero tentativi massimo		{Numero riga: “5”} {Numero colonna: “3”} {Numero tentativo: “3”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Tentativi esauriti”	Il turno passa al prossimo giocatore
ID15	Numero colonna non compreso nella griglia e Numero tentativi maggiore o uguale al Numero tentativi massimo	Numero riga <= numero riga griglia Numero colonna > Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi >= Numero tentativi massimo		{Numero riga: “3”} {Numero colonna: “5”} {Numero tentativo: “3”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Tentativi esauriti”	Il turno passa al prossimo giocatore
ID16	Numero tentativi maggiore o uguale del	Numero riga o colonna <= numero riga e colonna griglia		{Numero riga: “4”} {Numero colonna: “3”} {Numero tentativo: “3”}	Viene mostrato un messaggio di errore:	Il turno passa al prossimo giocatore

	Numero tentativi massimo	Numero riga o colonna != Numero riga o colonna bersaglio Numero tentativi >= Numero tentativi massimo		Nota: Grandezza griglia: {“4”,”4”} Numero tentativi massimo: {“3”}	“Numero tentativi esauriti”	

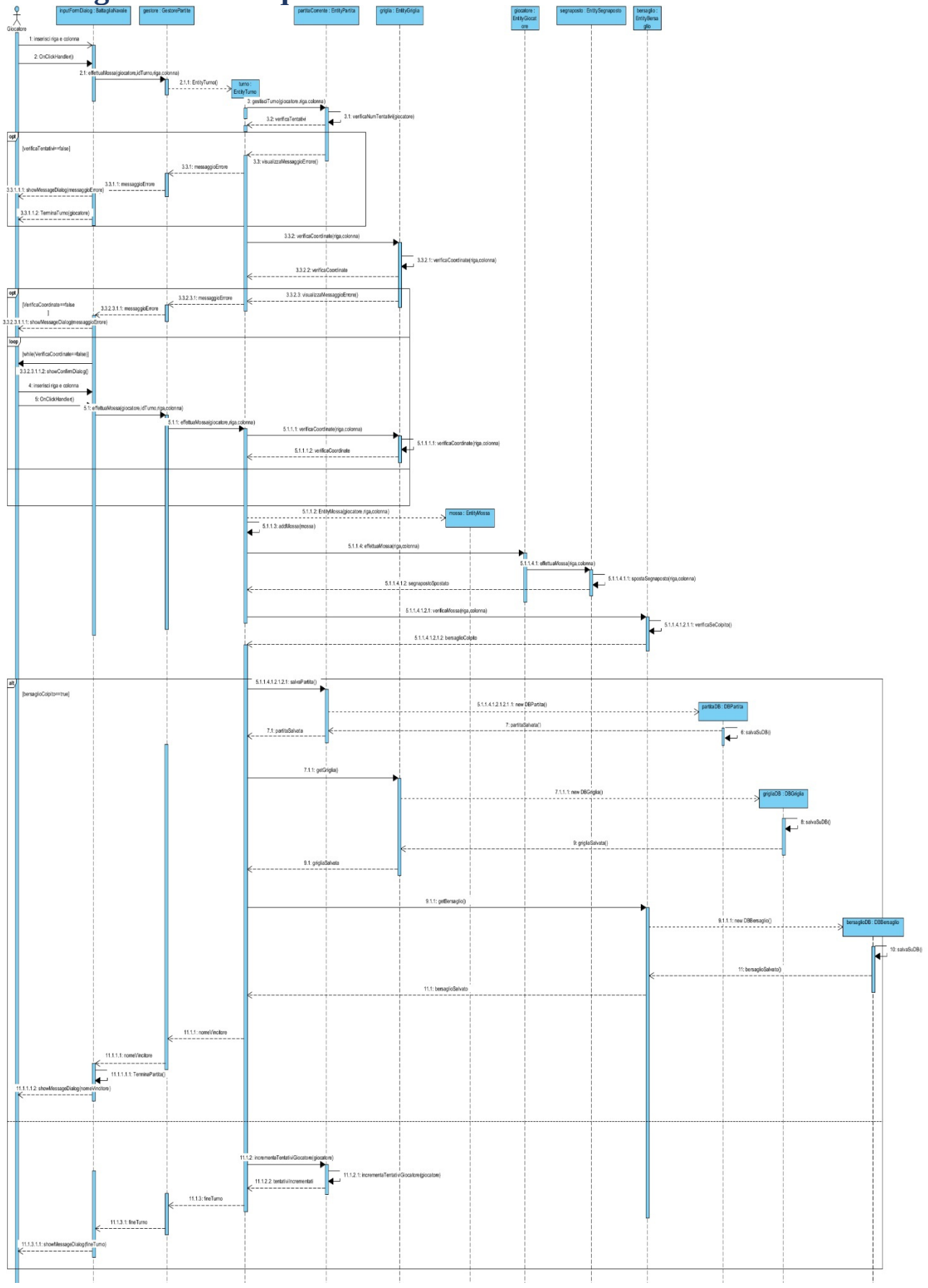
Nota: Nel caso in cui il numero di riga e colonna di Effettua Mossa sia più grande della dimensione della griglia, oppure il numero di tentativi del giocatore è superiore al massimo, si è evitato di considerare le due classi di equivalenza (Posizione del bersaglio) come distinte, quindi in un unico test sono state aggiunte entrambe le alternative. Il numero di test risultate era 13, ne abbiamo indicate 7 conseguenzialmente a ciò che è stato descritto sopra.

5. Progettazione

5.1 Diagramma delle classi



5.2 Diagramma di sequenza



6. Implementazione

PACKAGE ENTITY
EntityGiocatore: modella il Giocatore che partecipa ad una partita di battaglia navale. Presenta un nome, un'istanza della classe Segnaposto che, nel corso della partita, verrà spostato virtualmente dal giocatore, attraverso il metodo effettuaMossa(), e una lista di mosse riguardanti la partita in corso. La lista viene inizializzata e riempita ma non utilizzata nel codice visto che non era richiesto implementare il caso d'uso corrispondente a essa. Presenta inoltre un metodo che permette il salvataggio di un Giocatore sul database sfruttando i metodi di DBGiocatore.

EntityGiocatoreColor: estende la classe EntityGiocatore aggiungendo un attributo di tipo Color che viene settato randomicamente ogni volta che la classe GestorePartite invoca il costruttore di questa classe.

EntityPartita: modella la singola partita al gioco BattagliaNavale e presenta tutti i dati relativi ad una partita sotto forma di attributi: lista dei giocatori in gara con rispettivi tentativi effettuati, numero di tentativi massimo, la griglia, il bersaglio, una lista di turni e una lista di mosse.
Attraverso il metodo creaNuovoTurno() richiama il costruttore della classe Turno per creare una nuova istanza su cui poi si invocherà gestisciTurno().
Presenta metodi di utilità come metodi get o metodi per ottenere il giocatore dalla lista, e attraverso i metodi della classe DBPartita permette il salvataggio di una partita conclusa sul database.

EntityGriglia: modella la griglia, ossia il campo da gioco, della battaglia navale. Presenta una dimensione di riga e una di colonna e su di essa vengono posizionati virtualmente il bersaglio e i segnaposti dei Giocatori. Presenta un metodo per la verifica delle coordinate inserite dal Giocatore, che restituisce una stringa di controllo interpretata poi ai livelli superiori. Presenta inoltre un metodo per il salvataggio su database sfruttando i metodi di DBGriglia.

EntityBersaglio: modella il bersaglio del gioco battaglia navale e quindi rappresenta lo scopo del gioco: se colpito dal segnaposto di un Giocatore decreta la fine della partita. Presenta un numero di riga e di colonna (minori della dimensione di griglia) che lo posizionano virtualmente sulla griglia e presenta un metodo per verificare se il segnaposto l'ha raggiunto. Tale metodo garantisce l'esito del turno e viene utilizzato dalla classe EntityTurno per gestire la logica del turno.
EntitySegnaposto: modella il segnaposto del gioco battaglia navale. Presenta numero di riga e numero di colonna e un metodo spostaSegnaposto() che cambia questi attributi e quindi sposta virtualmente il segnaposto sulla griglia.

EntityTurno: modella il turno del gioco battaglia navale e si occupa di gestire la logica di effettua una mossa da parte di un Giocatore. Una nuova istanza di turno sarà creata dalla partita corrente al termine di ogni turno, se non c'è stato nessun vincitore.
Attraverso il metodo gestisciTurno() si gestisce la logica della mossa e si richiamano le classi che hanno il compito di effettuare le verifiche sugli input del Giocatore. Invia anche il segnale di termine partita e quindi salvataggio di essa sul database.

EntityMossa: modella la singola mossa effettuata dal Giocatore. Presenta un attributo partita che indica la partita in cui è stata effettuata la mossa, un attributo Giocatore,che indica il Giocatore che l'ha effettuata, e numero di riga e colonna che indicano la mossa.

PACKAGE CONTROL
GestorePartite: controller dell'applicazione. Presenta un attributo partitaCorrente riferito alla partita in corso per tenere traccia di tutti i dati utili. Presenta il metodo inizializzaPartita richiamato dalla classe boundary InizializzaPartita quando l'amministratore ha inserito correttamente i dati, tale metodo crea una nuova partita settando tutti i valori necessari. Inoltre presenta il metodo effettuaMossa che delega la classe Turno nella gestione della mossa da parte del Giocatore. Tale metodo viene invocato dalla classe BattagliaNavale quando il Giocatore inserisce in input numero di riga e di colonna della mossa da effettuare.
Si occupa anche di assegnare randomicamente un colore ad ogni giocatore in gara per la modellazione grafica del segnaposto.

PACKAGE BOUNDARY
BattagliaNavale: classe che gestisce l'andamento della partita. Notifica i giocatori uno alla volta invitandoli ad effettuare una mossa e attraverso i metodi di GestorePartite notifica il Giocatore di quello che sta accadendo. In particolare, interpreta una stringa di controllo per mostrare al giocatore il risultato della mossa.

InizializzaPartita: classe che si occupa dell'inizializzazione della partita. Mostra all'amministratore un form dove inserire i dati per creare una nuova partita. Sulle textfield è presente un DocumentListener che abilita il bottone di conferma solo quando i dati necessari sono stati inseriti. Si occupa di creare una griglia di JPanel sul frame BattagliaNavale per modellare il campo da gioco. Inoltre, presenta dei metodi per colorare e aggiungere icone sui suddetti JPanel per notificare visivamente al Giocatore l'esito della mossa.

PACKAGE DATABASE
DBGiocatore: classe che permette il salvataggio di un nuovo Giocatore sul database. Contiene gli attributi relativi al Giocatore.

DBPartita: classe che permette il salvataggio di una partita conclusa sul database.Contiene gli attributi relativi a partita. Fornisce attraverso selectDaDB() l'id della nuova partita da creare per evitare errori.

DBGriglia: classe che permette il salvataggio della griglia sul database. Contiene gli attributi relativi alla griglia. Il salvataggio avviene al termine della partita.

DBBersaglio: classe che permette il salvataggio del bersaglio sul database. Contiene tutti gli attributi relativi al bersaglio. Il salvataggio avviene al termine della partita.

DBConnectionManager: classe che permette la connessione al database e l'invio di query attraverso metodi di utilità.

Nella cartella condivisa sono stati generati i **javaDoc** relativi alle suddette classi.

NUMERO DI LOC: *1.965*
NUMERO DI LLOC: *1.300*

Il valore calcolato nella fase della stima dei costi, prima della fase di progettazione, risulta essere 1323, quindi, possiamo dedurre che la stima effettuata in quella fase, sia stata coerente con il risultato finale.

7. Testing

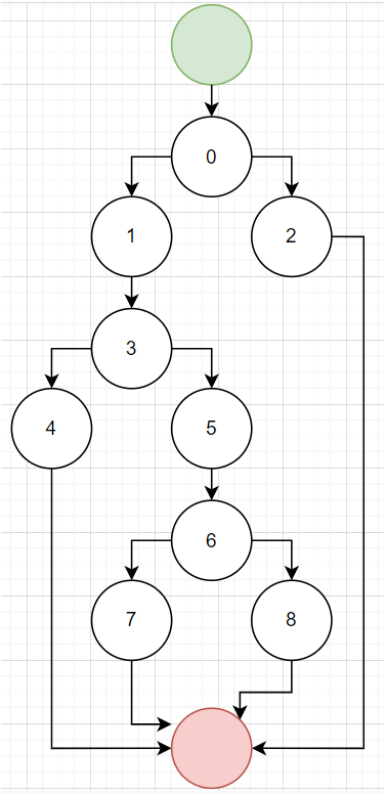
7.1 Test strutturale

7.1.1 Complessità ciclomatica

```
public String gestisciTurno(EntityGiocatore giocatore, int numRiga, int
numColonna) {
	if (!partita.verificaNumTentativi(giocatore))
		return partita.visualizzaMessaggioErrore();
	else {
		if (!(partita.getGriglia().verificaCoordinate(numRiga, numColonna)))
			return partita.getGriglia().visualizzaMessaggioErrore();
		else {
			EntityMossa mossa = new EntityMossa(numRiga, numColonna,
			giocatore, partita);
			addMossa(mossa);
			giocatore.effettuaMossa(numRiga, numColonna);
			if (partita.getBersaglio().verificaSeColpito(numRiga,
			    numColonna)) {
					partita.salvaPartitaSuDB(giocatore.getNome());
					partita.getGriglia().salvaGrigliaSuDB();
					partita.getBersaglio().salvaBersaglioSuDB();
					return giocatore.getNome();
			} else
				incrementaTentativiGiocatore(giocatore);

				return "Turno finito";
		}
	}
}
```

Control Flow Graph



NUMERO CICLOMATICO:

numero di regioni chiuse del grafo = 4

numero di nodi predicati (0,3,6) + 1 = 4

archi - # nodi +2 = (12-9) + 2 = 5

CAMMINI:

1. 0-2
2. 0-1-3-4
3. 0-1-3-5-6-7
4. 0-1-3-5-6-8

7.2 Test funzionale

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuti	Post-condizioni Ottenute	Esito (*FAIL*, *PASS*)
ID1	Tutti input validi	Numero riga valido Numero colonna valido	La verifica delle coordinate è andata a buon fine	{nuRiga: “3”} {numColonna: “5”}	Viene verificato l’esito della mossa	Il giocatore ha spostato il suo segnaposto			
ID2	Numero riga decimale	Numero riga decimale [ERROR] Numero colonna valido		{nuRiga: “2.2”} {numColonna: “5”}	Formato del numero riga errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID3	Numero riga < 0	Numero riga negativo [ERROR] Numero colonna valido		{nuRiga: “-2”} {numColonna: “5”}	Numero riga non ammesso		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID4	Stringa alfanumerica riga	Stringa riga alfanumerica [ERROR] Numero colonna valido		{nuRiga: “a2”} {numColonna: “5”}	Formato del numero riga errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID5	Stringa con caratteri riga	Stringa riga di caratteri [ERROR] Numero colonna valido		{nuRiga: “2@”} {numColonna: “5”}	Formato del numero riga errato		Messaggio di errore “l’input non ha un	Il giocatore reinserisce i dati in input	

							formato valido”		
ID6	Numero colonna decimale	Numero riga valido Numero colonna decimale [ERROR]		{nuRiga: “2”} {numColonna: “5.3”}	Formato del numero colonna errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID7	Numero colonna < 0	Numero riga valido Numero colonna < 0 [ERROR]		{nuRiga: “2”} {numColonna: “-4”}	Formato del numero colonna errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID8	Stringa alfanumerica colonna	Numero riga valido Stringa colonna alfanumerica [ERROR]		{nuRiga: “2”} {numColonna: “a3”}	Formato del numero colonna errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID9	Stringa con caratteri colonna	Numero riga valido Stringa colonna con caratteri [ERROR]		{nuRiga: “2”} {numColonna: “#3”}	Formato del numero colonna errato		Messaggio di errore “l’input non ha un formato valido”	Il giocatore reinserisce i dati in input	
ID10	Coordinate del bersaglio individuate	Numero riga <= Numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna = Numero		{Numero riga: “2”} {Numero colonna: “2”} {Numero tentativo: “2”} Nota:	Il nome del vincitore viene mostrato ai giocatori		Il nome del vincitore viene mostrato ai giocatori e	La partita termina e viene registrata nel database	PASS

		riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”} Grandezza griglia: {“4”,”4”}	e termina la partita		termina la partita		
ID11	Coordinate del bersaglio non individuate	Numero riga <= Numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		{Numero riga: “3”} {Numero colonna: “3”} {Numero tentativo: “2”} Nota: Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”} Grandezza griglia: {“4”,”4”}	Il giocatore termina il proprio turno		Il giocatore termina il proprio turno	Viene incrementato il numero di tentativi del giocatore e il controllo passa al giocatore successivo	PASS
ID12	Numero riga non compreso nella griglia	Numero riga > Numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		{Numero riga: “5”} {Numero colonna: “3”} {Numero tentativo: “2”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Coordinate non valide”		Viene mostrato un messaggio di errore: “Coordinate non valide”	Il giocatore dovrà reinserire il numero di riga e colonna	PASS

ID13	Numero colonna non compreso nella griglia	Numero riga <= Numero riga griglia Numero colonna > Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi < Numero tentativi massimo		{Numero riga: “3”} {Numero colonna: “5”} {Numero tentativo: “2”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Coordinate non valide”		Viene mostrato un messaggio di errore: “Coordinate non valide”	Il giocatore dovrà reinserire il numero di riga e colonna	PASS
ID14	Numero riga non compreso nella griglia e Numero tentativi maggiore o uguale al Numero tentativi massimo	Numero riga > numero riga griglia Numero colonna <= Numero colonna griglia Numero riga e colonna != Numero riga e colonna bersaglio Numero tentativi >= Numero tentativi massimo		{Numero riga: “5”} {Numero colonna: “3”} {Numero tentativo: “3”} Nota: Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Tentativi esauriti”		Viene mostrato un messaggio di errore: “Tentativi esauriti”	Il turno passa al giocatore successivo	PASS
ID15	Numero colonna non compreso nella griglia e Numero tentativi maggiore o	Numero riga <= numero riga griglia Numero colonna > Numero colonna griglia Numero riga e colonna != Numero		{Numero riga: “3”} {Numero colonna: “5”} {Numero tentativo: “3”} Nota:	Viene mostrato un messaggio di errore:		Viene mostrato un messaggio di errore: “Tentativi esauriti”	Il turno passa al giocatore successivo	PASS

	uguale al Numero tentativi massimo	riga e colonna bersaglio Numero tentativi >= Numero tentativi massimo		Grandezza griglia: {“4”,”4”} Posizione bersaglio: {“2”, “2”} Numero tentativi massimo: {“3”}	“Tentativi esauriti”				
ID16	Numero tentativi maggiore o uguale del Numero tentativi massimo	Numero riga o colonna <= numero riga e colonna griglia Numero riga o colonna != Numero riga o colonna bersaglio Numero tentativi >= Numero tentativi massimo		{Numero riga: “4”} {Numero colonna: “3”} {Numero tentativo: “3”} Nota: Grandezza griglia: {“4”,”4”} Numero tentativi massimo: {“3”}	Viene mostrato un messaggio di errore: “Numero tentativi esauriti”		Viene mostrato un messaggio di errore: “Numero tentativi esauriti”	Il turno passa al giocatore successivo	PASS