Ingegneria del Software 1: Requisiti di Progetto

Srđan Krstić and Claudio Menghi

Contents

1 Introduzione

Il progetto consiste nello sviluppo di una versione software del gioco da tavolo Council of Four (CoF).
Il progetto finale dovrà includere:

- diagramma UML iniziale dell'applicazione (al alto livello)
- diagrammi UML finali che mostrino come è stato progettato il software;
- implementazione funzionante del gioco conforme alle regole del gioco e alle specifiche presenti in questo documento
- codice sorgente dell'implementazione
- codice sorgente dei test di unità

La data di consegna: 3, Giugno 2016, AoE
La data di valutazione: TBD.

2 Requisiti di Progetto

I requisiti del progetto consistono in due gruppi di requisiti:

- requisiti game-specific
- requisiti game-agnostic

2.1 Game-specific requirements

2.1.1 Regole base

Le regole del gioco sono descritte nel file **GameRules.pdf** (Inglese) e **RegoleGioco.pdf** (Italiano), caricati su BeeP. Il file definisce **regole base** di gioco. In oltre, vengono aggiunte le seguenti **regole addizionali**.

2.1.2 Regole Addizionali

Nel seguito ci riferiamo con il termine **regole complete** all'unione delle regole base e regole addizionali.

- **Gioco configurabile**: il gioco deve essere progettato al fine di poter creare:
  1. mappa configurabile
     (a) connessioni arbitrarie tra le cittá;
     (b) bonus arbitrari sulle carte permit e sulle cittá e sulla "percorso della nobiltá";
     (c) numero di bonus arbitrari sulle carte permit e sulle cittá e sulla "percorso della nobiltá";
  2. numero arbitrario di giocatori.
  3. una lista di mappe (punti a-c per ogni mappa) deve essere caricata da un opportuno file di configurazione. Punto 2 deve essere passato come parametro alla creazione del gioco;
- **Market**: dopo che tutti i giocatori hanno giocato il loro turno (dopo ogni giro) viene lanciata la fase di Market. Partendo dal primo giocatore fino all'ultimo, ogni giocatore puó scegliere le carte (di permesso e politiche) e gli assistenti che intende vendere e il prezzo richiesto per ognuno di essi. Quando TUTTI i giocatori hanno scelto quali elementi vendere, seguendo un ordine casuale i giocatori possono comprare gli elementi messi in vendita dagli altri giocatori.

2.2 Game-agnostic requirements

In questa sezione vengono presentati i requisiti tecnici dell'applicazione. Deve essere un sistema distributivo composto da un lato gioco che puó gestire molti giochi simultanei e multipli lati giocatore che possono partecipare nel un solo gioco. Si raccomanda l'utilizzo del pattern **MVC** (Model-View-Controller) per progettare l'intero sistema.

2.2.1 I Componenti del Gioco

Il gioco consiste di "Lato Gioco" e "Lato giocatore".

Lato Giocatore

- Il lato giocatore deve poter essere istanziato più volte.
- Il lato giocatore deve essere sviluppato obbligatoriamente utilizzando JavaSE.
- L'interfaccia grafica deve essere mediante Swing o altre tecnologie (tuttavia in questo caso non sarete supportati dai responsabili),
- Il lato giocatore deve supportare RMI e Socket, in relazione al numero di studenti del gruppo, come specificato in tabella 1.
- Nel caso in cui sia richiesta sia l'implementazione RMI che quelle per mezzo di socket, l'applicazione, all'avvio, deve permettere all'utente di selezionare il metodo utilizzato per la comunicazione.
- Nel caso in cui sia richiesta sia l'implementazione CLI che quelle per mezzo di GUI, l'applicazione, all'avvio, deve permettere all'utente di selezionare il metodo utilizzato per la visualizzazione.

Lato Gioco

Questo componente deve gestire le partite e deve poter essere istanziato una sola volta. Deve permettere di:

- creare una nuova partita, inizializzarla, giocarla e concluderla secondo le regole del gioco.
- Deve essere in grado di gestire più partite contemporaneamente.
- Deve essere implementato secondo la logica JavaSE.
- Nel caso in cui sia l'implementazione via socket che quelle via RMI sia richiesta, quando un giocatore si connette al server, il server deve comunicare utilizzando la connessione selezionata.

2.2.2 Avvio della partita

L'assunzione base è che ogni giocatore che voglia partecipare a una partita conosca l'indirizzo (numerico o simbolico) del lato gioco. Quando un giocatore si connette,

- se cè una partita in fase di avvio, il giocatore viene automaticamente aggiunto alla partita
- **Regole base:**[1] la partita inizia non appena si raggiungono i 4 giocatori. Quando 2 giocatori si connettono a una partita viene inizializzato un timeout di 20 secondi. Non appena il timeout scatta la partita inizia anche se non sono raggiunti i 4 giocatori.

[1] se non sono richeste le regole addizionali

- **Regole complete:**[2] quando 2 giocatori si connettono a una partita viene inizializzato un timeout di 20 secondi. Non appena il timeout scatta la partita inizia. Il numero di giocatori puó essere arbitrario, anche maggiore di 4.
- se non ci sono partite in fase di avvio, viene creata una nuova partita.

Si precisa che una nuova partita viene creata solamente quando un utente si connette e non ci sono partite in attesa, altrimenti l'utente entra automaticamente a far parte della partita in fase di avvio.

2.2.3 Corso della partita

Il lato gioco consente ai vari giocatori di svolgere i propri turni secondo le regole di gioco. E' necessario gestire il caso in cui i lati giocatore si disconnettano.

- ogni giocatore ha un periodo di tempo fissato passato come parametro al momento della creazione del server per eseguire le mosse
- se un giocatore va offline il lato gioco attende per il periodo di cui sopra, dopo il quale sospende il giocatore (nota il giocatore non esegue mosse ma viene comunque considerato nel conteggio dei punti etc.)
- tutti i giocatori vengono notificati della mancanza di un giocatore
- il gioco continua, saltando i giri del giocatore sospeso
- il lato giocatore può riconnettersi e continuare il gioco se si sceglie di implementare la funzionalità "Ripristino sessione giocatore".

2.2.4 Funzionalita Avanzate

Di seguito sono proposte alcune funzionalità avanzate che concorrono alla valutazione. Attenzione: il loro contributo non è necessariamente additivo. Design e codice verranno comunque valutati in quanto tali e contribuiranno al giudizio globale.

- **Gestione degli utenti**. Realizzare un sistema di gestione degli utenti che supporti il login dei giocatori, conservi per ciascuno le statistiche di gioco (numero di vittorie, tempo di gioco, numero di sconfitte, etc...) e produca una classifica ordinata, prima per numero di vittorie e quindi per minimo tempo di gioco cumulativo. Inoltre, per ogni parita si desidera memorizzare, in un frame laterale, la sequenza di mosse effettuate dai vari giocatori in ordine di occorrenza.
- **Lato Gioco persistente**. Implementare la possibilità di salvare lo stato del lato gioco su disco e di ricaricarlo all'avvio successivo nel modo che si ritiene più idoneo.
- **Ripristino sessione giocatore**. Ai giocatori è permesso abbandonare temporaneamente la partita per via della perdita di connettività. Se il giocatore si riconnette prima della fine della partita può ricominciare a giocare, come se nulla fosse successo (ovviamente perde i turni durante i quali non ha giocato).
- **Generazione automatica delle configurazioni di gioco**. Il lato gioco deve essere in grado di generare automaticamente una configurazione del gioco partendo da alcuni parametri dati.
- **Lobby** questo requisito rimpiazza quelli specificati in sezione 2.2.2. Una volta avviato il server i giocatori entrano una lobby (area gestione giochi) dove possono:

[2] se sono richeste le regole addizionali

1. vedere la lista dei partite create (la mappa, il numero massimo di giocatori, il numero di giocatori attuale, ...)
2. creare partite (definire mappa, timeout per le mosse...)
3. partecipare alle partite

- **Chat** i giocatori devono poter comunicare tra di loro mediante chat
- **AI** deve essere possibile giocare contro il computer (un agente intelligente). Questo é una funzionalitá molto complessa. Per informazioni addizionali contattare i responsabili.

3 Valutazione

I gruppi di studenti devono implementare le specifiche descritte in tabella 1 in relazione al punteggio desiderato. Nota: in tabella sono rappresentati i **massimi** punteggi ottenibili in relazione alle feature implementate. Se un gruppo decide di implementare solo le regole del gioco (la prima linea) l'applicazione è centralizzata e utilizza un CLI. Altrimenti, l'applicazione deve comprendere dei diversi componenti distribuiti che comunicano in rete.
La seguente è la spiegazione delle abbreviazioni nella tabella.

- **Command Line Interface (CLI)**: è implementato come un interfaccia testuale e i vari giocatori che si alternano nei turni utilizzeranno la tastiera.
- **Graphical User Interface (GUI)**: consiste in un interfaccia grafica
- **RMI:** la comunicazione avviene mediante "Remote method invocation"
- **Socket**: la comunicazione avviene mediante messaggi scambiati attraverso socket. Lo studente deve autonomamente definire e implementare un protocollo di comunicazione tra i componenti distribuiti.

Per esempio, per un gruppo di 3 studenti e un punteggio desiderato di al massimo 24 punti è necessario implementare le regole complete (base e addizionali) del gioco, comunicazione con RMI, Socket e il Command Line Interface (CLI). All Tech significa che il gruppo deve implementare due modi di comunicazione (RMI e Socket) e due tipi di interfaccia (CLI e GUI). Abbiamo ottimizzato il carico di lavoro per il gruppi di 3 persone e vi consigliamo quindi di costituire gruppi di questa dimensione.
Gli studenti di telecomunicazioni, possono fare i gruppi di un massimo di 4 persone ed i requisiti applicati corrisponderanno il gruppo con una persona in meno.
Nota: Non è possibile fare i gruppi di 1 persona, la tabella 1 contiene le informazioni solo come riferimento agli studenti di telecomunicazioni.

Saranno oggetto di valutazione

Punteggio	Numero studenti		
Max	1	2	3
18	Regole base	Regole complete	Regole complete
22	CLI + GUI	RMI + CLI	Socket + CLI
24	RMI + CLI	Socket + CLI	RMI + Socket + CLI
26	Socket + CLI	RMI + Socket + CLI	RMI + Socket + GUI
28	RMI + Socket + CLI	RMI + Socket + GUI	All Tech
30L	RMI + Socket + GUI	All Tech	1 Funzionalità Avanzata

Table 1: Tabella di valutazione

- la qualità della **progettazione** con particolare riferimento a un uso appropriato di interfacce, ereditarietà, composizione tra classi, uso dei design pattern;
- la qualità della progettazione dell'**architettura dell'applicazione**; divisione della responsabilitá; progettazione della comunicazione;
- la stabilità dell'implementazione e la **conformità alle specifiche** date;
- la **qualità** e la **leggibilità** del codice scritto, con particolare riferimento a nomi di variabili/metodi/-classi/package, all'inserimento di commenti e documentazione JavaDoc (preferibilmente in inglese), la mancanza di codice ripetuto e metodi di eccessiva lunghezza;
- la **qualità** e la **copertura** dei casi di test: il nome e i commenti di ogni test dovranno chiaramente specificare le funzionalità testate e i componenti coinvolti;
- il corretto utilizzo degli strumenti (Eclipse, Git, Maven, ...);
- il livello di autonomia e impegno nello svolgimento del progetto.

Le funzionalità avanzate possono essere implementate da tutti i gruppi e comportano dei punteggi aggiuntivi. Ovviamente per implementare queste funzionalità è necessario che TUTTO il resto del progetto sia implementato in maniera COMPLETA e ADEGUATA (copertura con test, ben commentata etc).