Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica

Titolo della tesi

Tesi di laurea triennale

Relatore
Prof. Claudio Enrico Palazzi

Laureando
Maurizio Biancucci

Anno Accademico 2013-2014

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Dedicato a ...

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Pinco Pallino presso l'azienda Azienda S.p.A. Gli obbiettivi da raggiungere erano molteplici.
In primo luogo era richiesto lo sviluppo di ... In secondo luogo era richiesta l'implementazione di un ... Tale framework permette di registrare gli eventi di un controllore programmabile, quali segnali applicati Terzo ed ultimo obbiettivo era l'integrazione ...

"Life is really simple, but we insist on making it complicated"

— Confucius

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. NomeDelProfessore, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Ho desiderio di ringraziare poi i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Set 2014 Maurizio Biancucci

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Introduzione al contesto applicativo.

Esempio di utilizzo di un termine nel glossario
Application Program Interface (API).

Esempio di citazione in linea
site:agile-manifesto

Esempio di citazione nel pie' di pagina
citazione[1]

1.1 L'azienda

Descrizione dell'azienda.

1.2 L'idea

Introduzione all'idea dello stage.

1.3 Organizzazione del testo

Il secondo capitolo descrive ...

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;

[1] **womak:lean-thinking**.

- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Processi e metodologie

Brevissima introduzione al capitolo

2.1 Processo sviluppo prodotto

Capitolo 3

Descrizione dello stage

Breve introduzione al capitolo

3.1 Introduzione al progetto

3.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Performance del simulatore hardware
Descrizione: le performance del simulatore hardware e la comunicazione con questo potrebbero risultare lenti o non abbastanza buoni da causare il fallimento dei test.
Soluzione: coinvolgimento del responsabile a capo del progetto relativo il simulatore hardware.

3.3 Requisiti e obiettivi

3.4 Pianificazione

Capitolo 4

Multiplatform File Analyzer

In questo capitolo è presente una analisi completa del tool Multiplatform Fyle Analyzer

4.1 Introduzione

4.1.1 Premessa

Tutti i giochi multipiattaforma prodotti da Milestone sono saggiamente organizzati in modo che ogni qual volta debbano ricercare un file dato un percorso, prima eseguono la ricerca nella cartella specifica della piattaforma di esecuzione e poi, se non presente, nel percorso originale dato[1]. Grazie a questa organizzazione si riesce facilmente a differenziare gli asset in maniera molto semplice e veloce ogni qual volta sia necessario. La differenziazione degli asset nasce principalmente dalle differenti specifiche tecniche e capacità di calcolo di ciascuna piattaforma che costringono a creare versioni apposite. Ovviamente la specializzazione è un costo aggiuntivo in termini di tempo di creazione e di manutenzione e va eseguita fatto il meno possibile.

4.1.2 Lo scopo

Inizialmente il metodo adottato funzionava senza problemi evidenti ma, al veloce crescere del numero dei file e delle piattaforme target[2], si è notata l'esigenza di avere un maggior controllo.
Multiplaform File Analyzer è stato realizzato appositamente per rimediare a questo problema, offrendo una panoramica semplice ma completa riguardo le differenti versioni di ogni file specializzato per almeno una piattaforma.

4.2 Casi d'uso

Per meglio capire e tracciare l'esperienza d'uso che un developer di un gioco multipiattaforma avrà con il tool sono stati creati dei diagrammi di casi d'uso gerarchici.
I diagrammi di casi d'uso fanno parte della famiglia dei diagrammi UML e descrivono le funzionalità offerte dal prodotto così come sono percepite dagli attore che interagiscono con il sistema.
Ogni caso d'uso ha un codice univoco gerarchico, nella forma:

UC[codice univoco del padre].[codice progressivo di livello]

[1] Lo stesso sistema è utilizzato anche per il caricamento dei file contenenti i testi specifici per ogni differente localizzazione del gioco.

[2] Le piattaforme target sono attualmente sei: *Playstation Vita, Playstation 3, Xbox 360, Playstation 4 e Xbox One.*

Il codice progressivo può includere diversi livelli di gerarchia separati da un punto.

4.2.1 UC 1: Caso d'uso principale

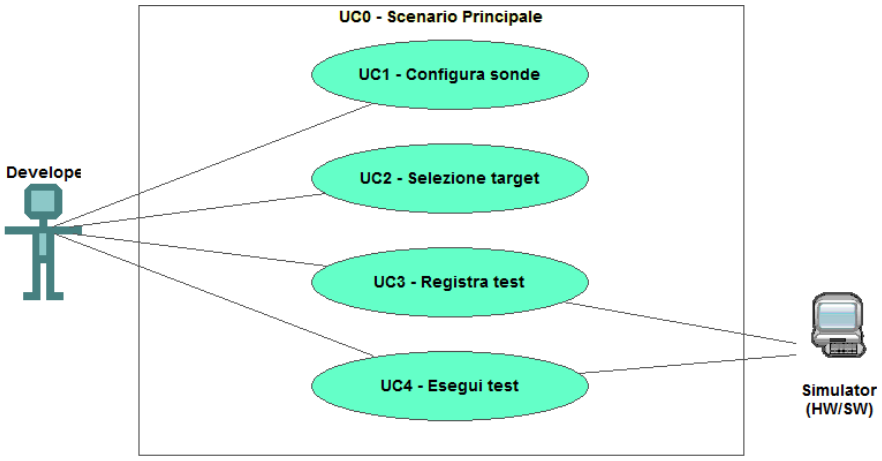

figura 4.1: Use Case - UC0: Scenario principale

- **Attori**: developer.
- **Descrizione**: un utente può eseguire un'analisi dei file di un gioco multipiattaforma.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi**:
  1. *Analisi dei file di un gioco multipiattaforma* (UC 1.1).
- **Postcondizione**: il sistema ha erogato le funzionalità richieste dal developer.

4.2.2 UC 1.1: Analisi dei file di un gioco multipiattaforma

- **Attori**: developer.
- **Descrizione**: un developer deve poter inserire i dati necessari all'analisi, visualizzare eventuali errori oppure il risultato dell'analisi.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi**:
  1. *Inserimento dei dati in input all'analisi* (UC 1.1.1);
  2. *Avvio dell'analisi* (UC 1.1.2);
  3. *Visualizzazione del risultato dell'analisi* (UC 1.1.4).
- **Estensioni**
  1. *Visualizzazione errori sui dati in input* (UC 1.1.3).
- **Postcondizione**: il sistema ha erogato le funzionalità richieste dal developer.

4.2.3 UC 1.1.1: Inserimento dei dati in input all'analisi

- **Attori**: developer.
- **Descrizione**: un developer deve poter inserire i dati necessari all'analisi composti da un percorso di base, i nomi delle piattaforme e il tempo di warning.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi**:
  1. *Inserimento percorso radice dei file da analizzare* (UC 1.1.1.1);
  2. *Inserimento dei nomi delle cartelle delle piattaforme* (UC 1.1.1.2);
  3. *Inserimento del warning time* (UC 1.1.1.3).
- **Postcondizione**: il sistema ha preso in carico i dati che verranno usati per la prossima analisi.

4.2.4 UC 1.1.1.1: Inserimento percorso radice dei file da analizzare

- **Attori**: developer.
- **Descrizione**: il developer deve poter inserire il percorso di base dell'analisi.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Scenario principale**: il developer sceglie il percorso di base da cui l'analisi partirà.
- **Postcondizione**: il sistema ha preso in carico il percorso di base inserito che verrà usato per la prossima analisi.

4.2.5 UC 1.1.1.2 Inserimento dei nomi delle cartelle delle piattaforme

- **Attori**: developer.
- **Descrizione**: il developer deve poter inserire i nomi delle piattaforme da analizzare.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Scenario principale**: il developer sceglie i nomi delle piattaforme.
- **Postcondizione**: il sistema ha preso in carico i nomi delle piattaforme inserite che verranno usate per la prossima analisi.

4.2.6 UC 1.1.1.3 Inserimento del warning time

- **Attori**: developer.
- **Descrizione**: il developer deve poter inserire il tempo di warning.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.

- **Scenario principale**: il developer il tempo superato il quale vuole essere avvertito.
- **Postcondizione**: il sistema ha preso in carico il tempo di warning che verrà usato per la prossima analisi.

4.2.7 UC 1.1.2 Avvio dell'analisi

- **Attori**: developer.
- **Descrizione**: il developer deve poter avviare la ricerca.
- **Precondizione**: il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Scenario principale**: il developer sceglie di avviare la ricerca.
- **Postcondizione**: il sistema inizia a eseguire l'analisi.

4.2.8 UC 1.1.3 Visualizzazione errori sui dati in input

- **Descrizione**: l'utente ha commesso uno dei seguenti errori durante l'inserimento dei dati in input all'analisi:
  - il percorso inserito non è valido;
  - l'utente non ha inserito nessuna piattaforma.
- **Precondizione**: il developer abbia avviato la ricerca.
- **Scenario principale**: viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione**: l'errore commesso è stato visualizzato e spiegato, l'analisi non è stata avviata.

4.2.9 UC 1.1.4 Visualizzazione dei risultati dell'analisi

- **Attori**: developer.
- **Descrizione**: un developer deve poter inserire i dati necessari all'analisi composti da un percorso di base, i nomi delle piattaforme e il tempo di warning.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Flusso principale degli eventi**:
  1. *Visualizzazione di tutti i file presenti in almeno una cartella di piataforma* (UC 1.1.4.1);
  2. *Visualizzazione del percorso per il file visualizzato* (UC 1.1.4.2);
  3. *Per ogni file visualizzato, visualizzazione del tempo di ultima modifica per ogni piattaforma per cui è presente* (UC 1.1.4.3);
  4. *Per ogni file visualizzato, visualizzazione della massima differenza fra i tempi di ultima modifica delle piattaforme in cui è presente* (UC 1.1.4.4);
  5. *Visualizzazione dei file che hanno la massima differenza fra i tempi di ultima modifica delle piattaforme in cui è presente, maggiore del tempo di warning* (UC 1.1.4.5);
  6. *L'utente può aprire l'explorer del sistema operativo per visualizzare dove questo si trovi* (UC 1.1.4.6);
- **Postcondizione**: i dati dell'analisi sono stati visualizzati.

4.2.10 UC 1.1.4.1 Visualizzazione di tutti i file presenti in almeno una cartella di piataforma

- **Attori**: developer.
- **Descrizione**: il developer deve poter visualizzare tutti i file che sono stati trovati in almeno una piattaforma.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Scenario principale**: vengono visualizzati i file presenti in almeno una piattaforma.
- **Postcondizione**: il risultato dell'analisi è stato visualizzato.

4.2.11 UC 1.1.4.2 Visualizzazione del percorso per il file visualizzato

- **Attori**: developer.
- **Descrizione**: il developer deve poter visualizzare il percorso del file visualizzato.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Scenario principale**: il developer visualizza il percorso del file visualizzato.
- **Postcondizione**: il percorso del file è stato visualizzato.

4.2.12 UC 1.1.4.3 Per ogni file visualizzato, visualizzazione del tempo di ultima modifica per ogni piattaforma per cui è presente

- **Attori**: developer.
- **Descrizione**: il developer deve poter visualizzare il tempo di ultima modifica per ciascuna piattaforma in cui il file è stato trovato.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Scenario principale**: viene visualizzata la data di ultima modifica per ciascuna piattaforma.
- **Postcondizione**: il risultato dell'analisi è stato visualizzato.

4.2.13 UC 1.1.4.4 Per ogni file visualizzato, visualizzazione della massima differenza fra i tempi di ultima modifica delle piattaforme in cui è presente

- **Attori**: developer.
- **Descrizione**: il developer deve poter visualizzare la massima differenza fra i tempi di ultima modifica delle piattaforme in cui è presente il file.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.

- **Scenario principale**: viene visualizzata la massima differenza fra i tempi di ultima modifica.
- **Postcondizione**: la massima differenza fra i tempi di ultima modifica è stata visualizzata.

4.2.14 UC 1.1.4.5 Visualizzazione dei file che hanno la massima differenza fra i tempi di ultima modifica delle piattaforme in cui è presente, maggiore del tempo di warning

- **Attori**: developer.
- **Descrizione**: il developer deve poter visualizzare e riconoscere i file per i quali la massima differenza fra i tempi di ultima modifica è maggiore del tempo di warning.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Scenario principale**: vengono visualizzati tutti i file che superano il tempo di warning.
- **Postcondizione**: i file che superano il tempo di warning sono stati visualizzati.

4.2.15 UC 1.1.4.6 L'utente può aprire l'explorer del sistema operativo per visualizzare dove questo si trovi

- **Attori**: developer.
- **Descrizione**: il developer deve poter aprire il programma che permettere l'esplorazione del file system nel punto in cui è presente il file correntemente selezionato.
- **Precondizione**: il developer abbia avviato l'analisi e che questa abbia terminato senza comunicare nessun errore sui dati in input.
- **Scenario principale**: viene visualizzato il file nella sua posizione all'interno del file system.
- **Postcondizione**: il file viene visualizzato nella sua posizione all'interno del file system.

4.3 Tracciamento dei requisiti

Partendo dai casi d'uso si è provveduto a stilare una precisa analisi dei requisiti per il tool in questione. I requisiti trovati sono stati inoltre tracciati in relazione al caso d'uso di origine.

Di seguito vengono riportati tutti i requisiti individuati. Per essere più leggibili verranno separati in tabelle a seconda della loro categoria. Di ogni requisito verranno indicati: tipologia, priorità e provenienza.
I requisiti dovranno essere classificati per tipo e importanza e utilizzeranno la seguente sintassi:

R[Importanza][Tipo][Codice]

- **Importanza**: può assumere solo uno fra i seguenti valori:

  - *0*: requisito obbligatorio;
  - *1*: requisito desiderabile;
  - *2*: requisito opzionale.

- **Tipo**: può assumere solo uno fra i seguenti valori:
  - *F*: funzionale;
  - *Q*: di qualità;
  - *P*: prestazionale;
  - *V*: vincolo.
- **Codice**: è il codice gerarchico univoco di ogni vincolo espresso in numeri (esempio: 1.3.2).

Per ogni requisito vengono inoltre specificati:

- **descrizione**: breve ma completa ed il meno ambigua possibile;
- **fonte**: può essere soltanto una o più tra le seguenti:
  - *caso d'uso*: il requisito è stato estrapolato da un caso d'uso. In questo caso va indicato il codice univoco del caso d'uso. È possibile indicare come fonte più di un caso d'uso.
  - *interno*: è stato ritenuto giusto aggiungere questo requisito per completezza.
  - *tutor aziendale*: il requisito è stato espressamente richiesto dal tutor aziendale.

4.3.1 Requisiti funzionali

Requisito	Descrizione	Fonti
R0F1	Un utente può effettuare l'analisi dei file di un gioco multipiattaforma.	UC 1.1
R0F1.1	L'analisi richiede l'inserimento di dati in input.	UC 1.1.1
R0F1.1.1	L'analisi richiede l'inserimento di un percorso assoluto dal quale far partire l'analisi.	UC 1.1.1.1
R0F1.1.2	L'analisi richiede l'inserimento dei nomi delle cartelle identificative di file di piattaforma.	UC 1.1.1.2
R0F1.1.2.1	Il programma interpreta i nomi delle piattaforme in modo case-insensitive.	UC 1.1.1.2
R0F1.1.2.2	Il programma richiede l'inserimento dei nomi delle piattaforme come unica stringa utilizzando il carattere ';' come separatore.	UC 1.1.1.2
R0F1.1.3	L'analisi richiede l'inserimento di un tempo di warning, superato il quale, a fine l'analisi, verrà segnalato il problema se presente.	UC 1.1.1.3

R0F1.1.3.1	Il formato del tempo di warning accettato prevede l'utilizzo esclusivo di numeri in formato inglese e con massimo due cifre decimali.	UC 1.1.1.3
R0F1.1.4	Il programma deve memorizzare e preinserire i dati della precedente analisi (se esistenti) negli appositi campi per l'input dei dati, anche se tra un'analisi e la successiva il programma è stato chiuso.	Interno
R0F1.3	Il programma mostra gli eventuali errori sui dati in input all'analisi trovati durante l'avvio della stessa.	UC 1.1.3
R0F1.3.1	Il programma ferma l'avvio dell'analisi se il percorso di base non viene fornito.	UC 1.1.3
R0F1.3.2	Il programma ferma l'avvio dell'analisi se il percorso di base inserito non è un percorso assoluto.	UC 1.1.3
R0F1.3.3	Il programma ferma l'avvio dell'analisi se il percorso di base inserito non esiste nel file system.	UC 1.1.3
R0F1.3.4	Il programma ferma l'avvio dell'analisi se il contenuto del percorso di base inserito non è leggibile.	UC 1.1.3
R0F1.4	Il programma permette la visualizzazione dei dati output dell'analisi.	UC 1.1.4
R0F1.4.1	Nell'output dell'analisi è presente ciascun file presente in almeno un cartella di piattaforma.	UC 1.1.4.1
R0F1.4.2	Per ciascun file presente nell'output è visibile la data di ultima modifica per ciascuna piattaforma in cui è presente.	UC 1.1.4.2
R0F1.4.3	Per ciascun file presente nell'output è indicato il percorso di relativo a partire dal percorso di base inserito in input all'analisi.	UC 1.1.4.3
R0F1.4.4	Per ciascun file presente nell'output è indicato il nome del file completo di estensione.	UC 1.1.4.4
R0F1.4.5	Per ciascun file presente nell'output è indicata la massima differenza tra le date di ultima modifica delle piattaforme per cui è presente	UC 1.1.4.5

R0F1.4.5.1	Se il tempo massimo per ciascuno file è superiore al tempo di warning inserito in input allora questo viene evidenziato come warning.	UC 1.1.4.5
R0F1.4.5.2	Il tempo è mostrato nella stessa unità di misura in cui è stato inserito il tempo di warning.	UC 1.1.4.5
R0F1.4.6	Il programma permette di aprire il programma per esplorare il file system di default del sistema e visualizzare la cartella dove è presente il file	UC 1.1.4.6

tabella 4.1: Requisiti funzionali

4.3.2 Requisiti di qualità

Requisito	Descrizione	Fonti
R0Q1	Viene fornito insieme al programma un l'help con la guida alla compilazione e al deploy per sistemi Windows.	Tutor interno
R0Q2	Viene fornito un l'help che spiega come utilizzare il programma.	Tutor interno

tabella 4.2: Requisiti di vincolo

4.3.3 Requisiti di vincolo

Requisito	Descrizione	Fonti
R0V1	Il programma deve funzionare su Windows 7 SP1 32 e 64 bit.	Tutor interno
R0V2	Il programma deve essere scritto utilizzando il linguaggio C++ 98.	Tutor interno

tabella 4.3: Requisiti di vincolo

4.4 Tecnologie e strumenti

Di seguito viene fornita una panoramica delle tecnologie e strumenti utilizzati.

4.4.1 Qt Framework

Per lo sviluppo del tool è stato usato Qt Framework versione 5.3.1. Qt è un potente Framework multipiattaforma che offre una vastissima serie di librerie di utilità, grazie alle quali lo sviluppo diviene agevole e veloce, permettendo quasi sempre di non legarsi a specifiche implementazioni per piattaforma.
In particolare è stata usata la libreria per costruire le interfacce grafiche e la libreria per accedere al file system e leggerne le informazioni.

4.4.2 Qt Creator

Come IDE per lo sviluppo ci si è affidati a Qt Creator 3.1.2. IDE semplice e performante che si integra perfettamente con il mondo Qt, offrendo addirittura la consultazione veloce della documentazione direttamente nell'IDE.

4.5 Ciclo di vita del software

Essendo Multiplatform File Analyzer uno strumento di piccole dimensioni si è adottato un semplice modello di ciclo di vita incrementale, con un singolo incremento corrispondente alla consegna finale.

4.6 Progettazione

Il software è stato progettato per offrire una semplice espandibilità, allo scopo la strutturazione generale segue il collaudato Design Pattern *MVC*, dove trovano luogo i seguenti pacchetti: `controller`, `view` e `data`. É inoltre presente il pacchetto `core` in cui è inserita l'implementazione degli algoritmi di business del tool, ad esempio l'algoritmo che esegue l'analisi.

Il pacchetto core è implementato con il Design Pattern *Strategy* allo scopo di permettere una facile estensione o sostituzione dell'algoritmo che esegue l'analisi, anche a run-time.

Per permettere una facile e veloce connessione tra gli stati descritti dall'*MVC* è stato adottato il Design Pattern *Obsever*, implementato nell'omonimo pacchetto *observer*.

Allo scopo di rendere il programma estendibile è stato scelto di utilizzare il Design Pattern *Strategy* per rendere modificabile facilmente l'implementazione dell'algoritmo che effettua l'analisi, all'interno del pacchetto `core`.

È stato inoltre cercato di utilizzare le classi del Framework Qt in modo isolato alle sole parti strettamente necessarie, ovvero la GUI e l'analisi del file system. Questo allo scopo di rendere il tool slegato dalle implementazioni specifiche, in favore di una più semplice espandibilità.

4.6.1 Diagramma delle classi

Di seguito è presente il diagramma delle classi espresso tramite il formalismo UML 2.0. Nel diagramma è inoltre possibile vedere le principali dipendenza verso le classi del Framework Qt.

todo inserire il diagramma delle classi, aggiornarlo poi anche su github e alla milestone

4.6.2 Pacchetto controller

Il pacchetto è composto dalla sola classe `Controller` che svolge il compito di controllore del sistema. È questo il solo pacchetto utilizzato dall funzione `main`.

Classe Controller

La classe `Controller` svolge il ruolo di starter del sistema. Essa provvede ad allocare tutti gli oggetti necessari e a rilasciarli in fase di chiusura. Provvede inoltre a selezionare la view e a connetterla con le classi che rappresentano i dati. Raccoglie i comandi utente provenienti dalla view e li esegue grazie al pacchetto `core`.

4.6.3 Pacchetto observer

Questo pacchetto rappresenta l'implementazione del Design Pattern *observer* ed è stata utilizzata per tenere aggiornata la view al cambiamento dei dati.

Interfaccia IObserver

`IObserver` è una classe astratta che rappresenta gli oggetti che osservano le modifiche di altri oggetti. Essa contiene il metodo virtuale puro `Update` che i soggetti osservati chiamano per invocare l'aggiornamento al cambiamento dei dati. Le classi concrete che desidereranno poter osservare un soggetto deriveranno da queste e implementeranno il metodo `Update`.

Classe Subject

La classe `Subject` rappresenta un soggetto che può essere osservato. Essa contiene la base di codice per notificare tutti gli osservatori dell'avvenuto cambiamento, aggiungere e rimuovere osservatori. Le classi che desiderano poter essere osservate deriveranno da questa.

4.6.4 Pacchetto data

Il pacchetto `data` contiene tutte le classi che rappresentano i dati di business del programma. Allo scopo di rendere il programma eseguibile anche su un semplice terminale e quindi slegato dal mondo Qt, le classi di dati sono state implementate con la STL invece delle classi collezione offerte dal Framework in uso.

Classe InputData

La classe `InputData` raccoglie tutti i dati che sono di input all'analisi. Contiene quindi il percorso di partenza, i nomi delle piattaforme ed il tempo di warning. É questa classe che si occupa della trasformazione della stringa contenente tutte le piattaforme in un comodo `Vector`.

Classe FileOccurrence

La classe `FileOccurrence` rappresenta un file trovato in almeno una cartella di piattaforma durante l'analisi. Per il file trovato, la classe contiene il percorso assoluto, il nome, tutte le piattaforme in cui è stato trovato e, per ciascuna, la data di ultima modifica.

ResultData

La classe `ResultData` contiene tutti i dati presenti come output di una analisi. Contiene quindi una collezione di `FileOccurrence` e tutti i dati di input all'analisi.

Essa eredita dalla classe `Subject` per permettere ad un osservatore, tipicamente una View di aggiornarsi al variare dei risultati. Permettendo di mostrare i risultati dell'analisi in modo interattivo durante l'inserimento di ogni nuovo `FileOccurence` trovato.

4.6.5 Pacchetto view

Questo pacchetto si occupa delle interfacce utente, derivando e personalizzando le classi offerte dal Framework Qt.

Iterfaccia IView

`IView` è una classe astratta che rappresenta una generica view.

Classe ViewMainWindow

La classe `ViewMainWindow` implementa `IView` tramite una finestra. Eredita e specializza la classe `QMainWindow` di Qt. Raccoglie i comandi utente e ne delega l'esecuzione al controller.

Classe ResultWidgetBase

La classe astratta `ResultWidgetBase` rappresenta un generico widget integrabile in una interfaccia grafica che sfrutta le classi del Framework Qt per la visualizzazione dell'output dell'analisi.

Classe ResultTableWIdget

La classe `ResultWidget` implementa la classe base astratta `ResultWidgetBase` mostrando l'output dell'analisi in forma tabellare, dedicando una riga a ciascun file trovato.

4.6.6 Core

Il pacchetto si occupa della realizzazione dell'analisi e di tutti gli algoritmi di business del tool. Utilizza il Pattern *Strategy* per l'organizzazione delle classe contenute.

Interfaccia ICoreMultiplatformFileAnalyzer

`ICoreMultiplatformFileAnalyzer` rappresenta l'algoritmo di business del programma, ovvero quello che effettua l'analisi e riempie un'istanza della classe `ResultData` con il risultato.

Classe QtCoreMultiplatformFIleAnalyzer

La classe `QtCoreMultiplatformFIleAnalyzer` implementa l'interfaccia `ICoreMultiplatformFileAnalyzer` utilizzando il Framework Qt per esplorare il file system.

4.7 Codifica

Tutto il codice del tool è stato pubblicato sulla piattaforma GitHub, accedibile tramite il seguente indirizzo: https://github.com/Mauxx91/Multiplatform-File-Analyzer. Tutto il codice è disponibile gratuitamente sotto licenza GNU GPL v3[3]

[3]Un'approfondimento sulla licenza può essere trovato al seguente indirizzo: http://www.gnu.org/copyleft/gpl.html.

Tutto il codice è stato scritto in lingua inglese, compresi i commenti, dei quali si è cercato di scriverne il più possibile. La codifica ha seguito alcune convenzioni della famosa notazione Ungherese. Di seguito sono elencati i formalismi utilizzati nel codice:

- **prefissi:**
  - **m**: usato per le variabili membro (esempio: `m_keyName`);
  - **p**: usato per le variabili puntatore (esempio: `m_pkeyNameList`);
  - **o**: usato per le variabili input alla funzione che sono usate come output (esempio: `o_foundKeys`);
  - **I**: usato per le classi interfacce (esempio: `IObserver`).
- **suffissi:**
  - **Base**: usato per le classi astratte (esempio: `ResultWidgetBase`);
- **capitalizzazione:**
  - **variabili e parametri**: iniziano sempre in minuscolo e usano una lettera minuscola per ogni parola (esempio: `m_parentWidget`, `returnValue`);
  - **macro ed enum**: completamente in maiuscolo con le diverse parole separate dal carattere '_' (esempio: `UPDATE_CODE`);
  - **classi, funzioni e metodi**: iniziano sempre in maiuscolo e ogni parola diversa parte in maiuscolo (esempio: `FileOccurrence`);

todo -> include del codice o no?

4.8 Conclusioni

Tutti i requisiti sono stati soddisfatti come pianificato e nei tempi prestabiliti.
La realizzazione del tool ha messo di fronte lo studente a una problematica reale presente durante lo sviluppo di giochi multipiattaforma. Contemporaneamente gli ha permesso di interfacciarsi ai comuni problemi legati alla manutenzione dei moltissimi file presenti nei grandi progetti, lasciandogli la libertà di progettare e realizzare una soluzione.
Lo studente ha inoltre approfondito le tematiche della creazione di interfacce usabili e degli algoritmi per l'esplorazione del file system.

Capitolo 5

Progettazione e codifica

Breve introduzione al capitolo

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

Tecnologia 1

Descrizione Tecnologia 1.

Tecnologia 2

Descrizione Tecnologia 2

5.2 Ciclo di vita del software

5.3 Progettazione

Namespace 1

Descrizione namespace 1.

Classe 1: Descrizione classe 1

Classe 2: Descrizione classe 2

5.4 Design Pattern utilizzati

5.5 Codifica

Capitolo 6

Verifica e validazione

Capitolo 7

Conclusioni

7.1 Consuntivo finale

7.2 Raggiungimento degli obiettivi

7.3 Conoscenze acquisite

7.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia

Riferimenti bibliografici

Siti Web consultati