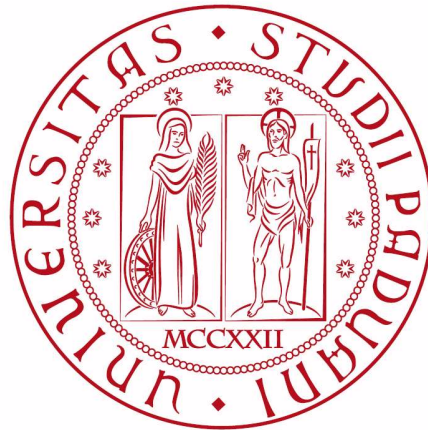


Università degli Studi di Padova

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, consectetur 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

1	Introduzione	1
1.1	L'azienda	1
1.2	L'idea	1
1.3	Organizzazione del testo	1
2	Processi e metodologie	3
2.1	Processo sviluppo prodotto	3
3	Descrizione dello stage	5
3.1	Introduzione al progetto	5
3.2	Analisi preventiva dei rischi	5
3.3	Requisiti e obiettivi	5
3.4	Pianificazione	5
4	Multiplatform File Analyzer	7
4.1	Introduzione	7
4.1.1	Premessa	7
4.1.2	Lo scopo	7
4.2	Casi d'uso	7
4.2.1	UC 1: Caso d'uso principale	8
4.2.2	UC 1.1: Analisi dei file di un gioco multiplatforma	8
4.2.3	UC 1.1.1: Inserimento dei dati in input all'analisi	10
4.2.4	UC 1.1.1.1: Inserimento percorso radice dei file da analizzare	10
4.2.5	UC 1.1.1.2 Inserimento dei nomi delle cartelle delle piattaforme	10
4.2.6	UC 1.1.1.3 Inserimento del warning time	10
4.2.7	UC 1.1.2 Avvio dell'analisi	11
4.2.8	UC 1.1.3 Visualizzazione errori sui dati in input	11
4.2.9	UC 1.1.4 Visualizzazione dei risultati dell'analisi	11
4.2.10	UC 1.1.4.1 Visualizzazione di tutti i file presenti in almeno una cartella di piattaforma	12
4.2.11	UC 1.1.4.2 Visualizzazione del percorso per il file visualizzato	13
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	13
4.2.13	UC 1.1.4.4 Per ogni file visualizzato, visualizzazione della massi- ma differenza fra i tempi di ultima modifica delle piattaforme in cui è presente	13
4.2.14	UC 1.1.4.5 Visualizzazione dei file che hanno la massima diffe- renza fra i tempi di ultima modifica delle piattaforme in cui è presente, maggiore del tempo di warning	14
4.2.15	UC 1.1.4.6 L'utente può aprire l'explorer del sistema operativo per visualizzare dove questo si trovi	14
4.3	Tracciamento dei requisiti	14
4.3.1	Requisiti funzionali	15



4.3.2	Requisiti di qualità	17
4.3.3	Requisiti di vincolo	17
4.4	Tecnologie e strumenti	17
4.4.1	Qt Framework	17
4.4.2	Qt Creator	18
4.5	Ciclo di vita del software	18
4.6	Progettazione	18
4.6.1	Diagramma delle classi	18
4.6.2	Pacchetto controller	20
4.6.3	Pacchetto observer	20
4.6.4	Pacchetto data	20
4.6.5	Pacchetto view	21
4.6.6	Core	21
4.7	Codifica	21
4.8	Conclusioni	22
5	XML Editor	23
5.1	Introduzione	23
5.1.1	Premessa	23
5.1.2	Lo scopo	23
5.2	Casi d'uso	23
5.2.1	UC 1: Caso d'uso principale	24
5.2.2	UC 1.1: Creazione di un nuovo file XML	25
5.2.3	UC 1.1.1: Inserimento del percorso dove memorizzare il nuovo XML	25
5.2.4	UC 1.1.2: Inserimento del nome del file del nuovo XML	25
5.2.5	UC 1.2: Apertura di un file XML preesistente	25
5.2.6	UC 1.3 Editing del file XML correntemente aperto	26
5.2.7	UC 1.3.1 Visualizzazione del nodo riferito dal nodo selezionato	27
5.2.8	UC 1.3.2 Visualizzazione degli errori occorsi durante la ricerca del nodo riferito	27
5.2.9	UC 1.3.3 Aggiunta di un nuovo nodo figlio al nodo selezionato	27
5.2.10	UC 1.3.4 Editing del nodo selezionato	28
5.2.11	UC 1.3.4.1 Modifica del valore dell'elemento	28
5.2.12	UC 1.3.4.2 Inserimento di un nuovo attributo	28
5.2.13	UC 1.3.4.3 Modifica di un attributo esistente	28
5.2.14	UC 1.3.4.4 Rimozione di un attributo esistente	29
5.2.15	UC 1.3.5 Duplicazione del nodo selezionato	29
5.2.16	UC 1.3.6 Copia del nodo selezionato	29
5.2.17	UC 1.3.7 Incollare il nodo precedentemente copiato	29
5.2.18	UC 1.3.8 Rimozione del nodo selezionato	30
5.2.19	UC 1.3.9 Rimozione di tutti i nodi figli del nodo selezionato	30
5.2.20	UC 1.3.10 Istanziamento di una nuova relazione	30
5.2.21	UC 1.3.11 Visualizzazione degli errori occorsi durante l'istanziamento di una nuova relazione	30
5.2.22	UC 1.3.12 Annullamento dell'ultima modifica eseguita	31
5.2.23	UC 1.3.13 Ripristino dell'ultima modifica annullata	31
5.2.24	UC 1.4: Apertura di un file XML preesistente	31
5.2.25	UC 1.5 Editing del filtro sul nome dell'attributo da visualizzare	32
5.2.26	UC 1.6 Editing dei file associati	32
5.2.27	UC 1.6.1 Aggiunta di un nuovo file associato	32
5.2.28	UC 1.6.2 Rimozione di un file precedentemente associato	33
5.2.29	UC 1.7 Editing delle relazioni	33
5.2.30	UC 1.7.1 Inserimento di una nuova relazione	33
5.2.31	UC 1.7.2 Modifica di una relazione precedentemente inserita	34



5.2.32	UC 1.7.3 Rimozione di una relazione precedentemente inserita .	34
5.2.33	UC 1.8 Importazione delle relazioni da file	34
5.2.34	UC 1.9 Esportazione delle relazioni su file	34
5.2.35	UC 1.10 Avvio del check sulle relazioni	35
5.2.36	UC 1.11 Visualizzazione del risultato del check sulle relazioni .	35
5.2.37	UC 1.11.1 Visualizzazione degli errori	35
5.2.38	UC 1.11.2 Selezione dei filtri sulle categorie di errori da visualizzare	36
5.2.39	UC 1.11.3 Visualizzazione dell'elemento causa dell'errore nell'al- bero XML	36
5.2.40	UC 1.12 Visualizzazione dell'about sul programma	36
5.2.41	UC 1.13 Visualizzazione degli errori occorsi durante l'apertura del file	36
5.2.42	UC 1.14 Visualizzazione degli errori occorsi durante il salvataggio del file	37
5.2.43	UC 1.15 Visualizzazione degli errori occorsi durante l'importazione	37
5.2.44	UC 1.16 Visualizzazione degli errori occorsi durante l'esportazione	37
5.3	Tracciamento dei requisiti	38
5.3.1	Requisiti funzionali	39
5.3.2	Requisiti di qualità	40
5.3.3	Requisiti di vincolo	41
5.4	Tecnologie e strumenti	41
5.4.1	Qt Framework	41
5.4.2	Qt Creator	41
5.5	Ciclo di vita del software	41
5.6	Progettazione	41
5.6.1	Diagramma delle classi	42
5.6.2	Pacchetto controller	44
5.6.3	Pacchetto observer	44
5.6.4	Pacchetto data	44
5.6.5	Pacchetto view	45
5.6.6	Core	45
5.7	Codifica	45
5.8	Conclusioni	46
6	Verifica e validazione	47
7	Conclusioni	49
7.1	Consuntivo finale	49
7.2	Raggiungimento degli obiettivi	49
7.3	Conoscenze acquisite	49
7.4	Valutazione personale	49
A	Appendice A	51
	Bibliografia	55

Elenco delle figure

4.1	Use Case - UC 1: Caso d'uso principale	8
4.2	Use Case - UC 1.1: Analisi dei file di un gioco multiplatforma	9
4.3	Use Case - UC 1.1.1: Inserimento dei dati in input all'analisi	9
4.4	Use Case - UC 1.1.4: Visualizzazione dei risultati dell'analisi	12
4.5	Diagramma delle classi di Multiplatform File Analyzer	19
5.1	Use Case - UC 1: Caso d'uso principale	24
5.2	Use Case - UC 1.1: Creazione di un nuovo file XML	25
5.3	Use Case - UC 1.3 Editing del file XML correntemente aperto	26
5.4	Use Case - UC 1.3.4 Editing del nodo selezionato	28
5.5	Use Case - UC 1.6 Editing dei file associati	32
5.6	Use Case - UC 1.7 Editing delle relazioni	33
5.7	Use Case - UC 1.11 Visualizzazione del risultato del check sulle relazioni	35
5.8	Diagramma delle classi di Multiplatform File Analyzer	43

Elenco delle tabelle

4.1	Requisiti funzionali	17
4.2	Requisiti di vincolo	17
4.3	Requisiti di vincolo	17
5.1	Requisiti funzionali	40
5.2	Requisiti di vincolo	41
5.3	Requisiti di vincolo	41

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
citazione¹

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;

¹womak:lean-thinking.



1.3. ORGANIZZAZIONE DEL TESTO

- 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 multiplatforma 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 dato¹. 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 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 target², si è notata l'esigenza di avere un maggior controllo.

Multiplatform 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 multiplatforma 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 attori 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]

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

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

4.2. CASI D'USO

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

4.2.1 UC 1: Caso d'uso principale

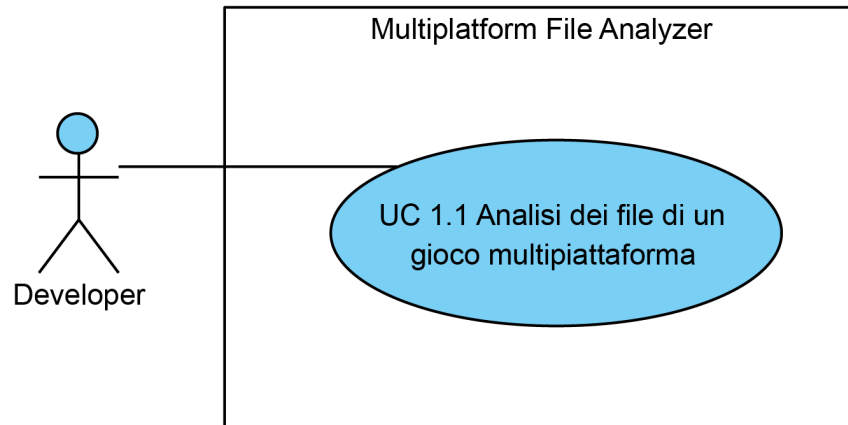


figura 4.1: Use Case - UC 1: Caso d'uso principale

- **Attori:** developer.
- **Descrizione:** un developer può eseguire un'analisi dei file di un gioco multiplatforma.
- **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 multiplatforma (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 multiplatforma

- **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.

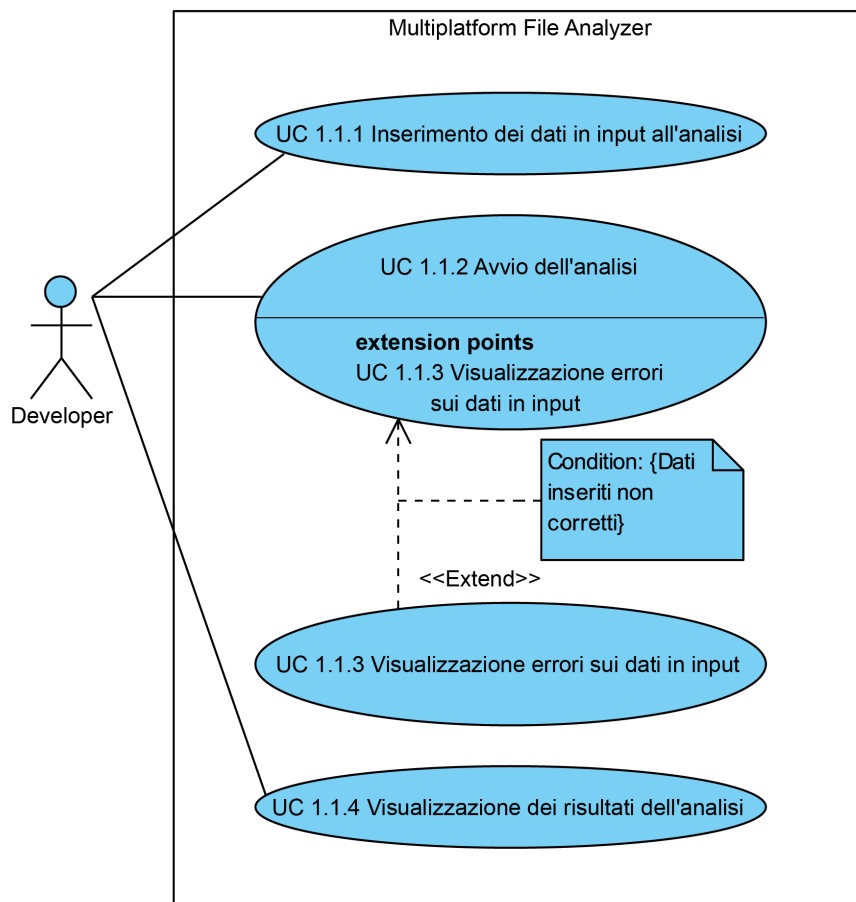


figura 4.2: Use Case - UC 1.1: Analisi dei file di un gioco multiplatforma

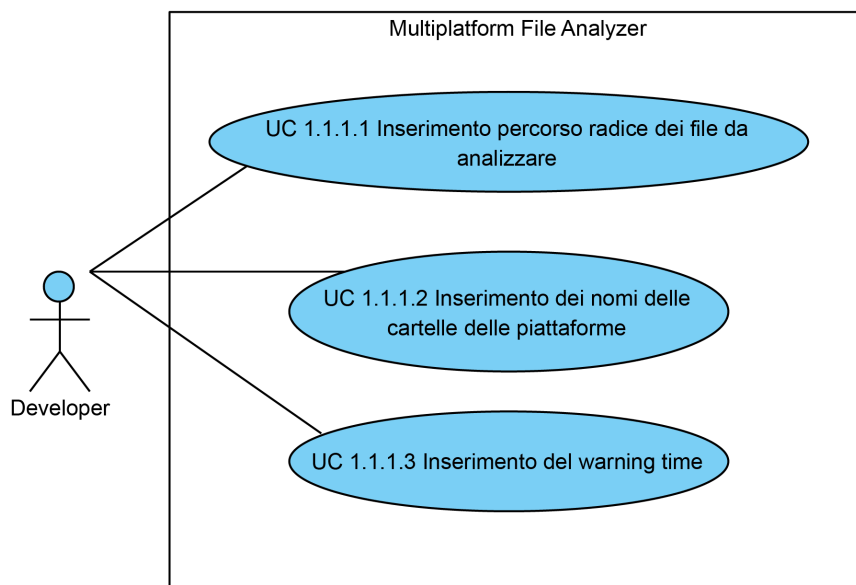


figura 4.3: Use Case - UC 1.1.1: Inserimento dei dati in input all'analisi



4.2. CASI D'USO

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:** il developer ha commesso uno dei seguenti errori durante l'inserimento dei dati in input all'analisi:
 - il percorso inserito non è valido;
 - il developer 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 piattaforma ([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. Il developer 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.

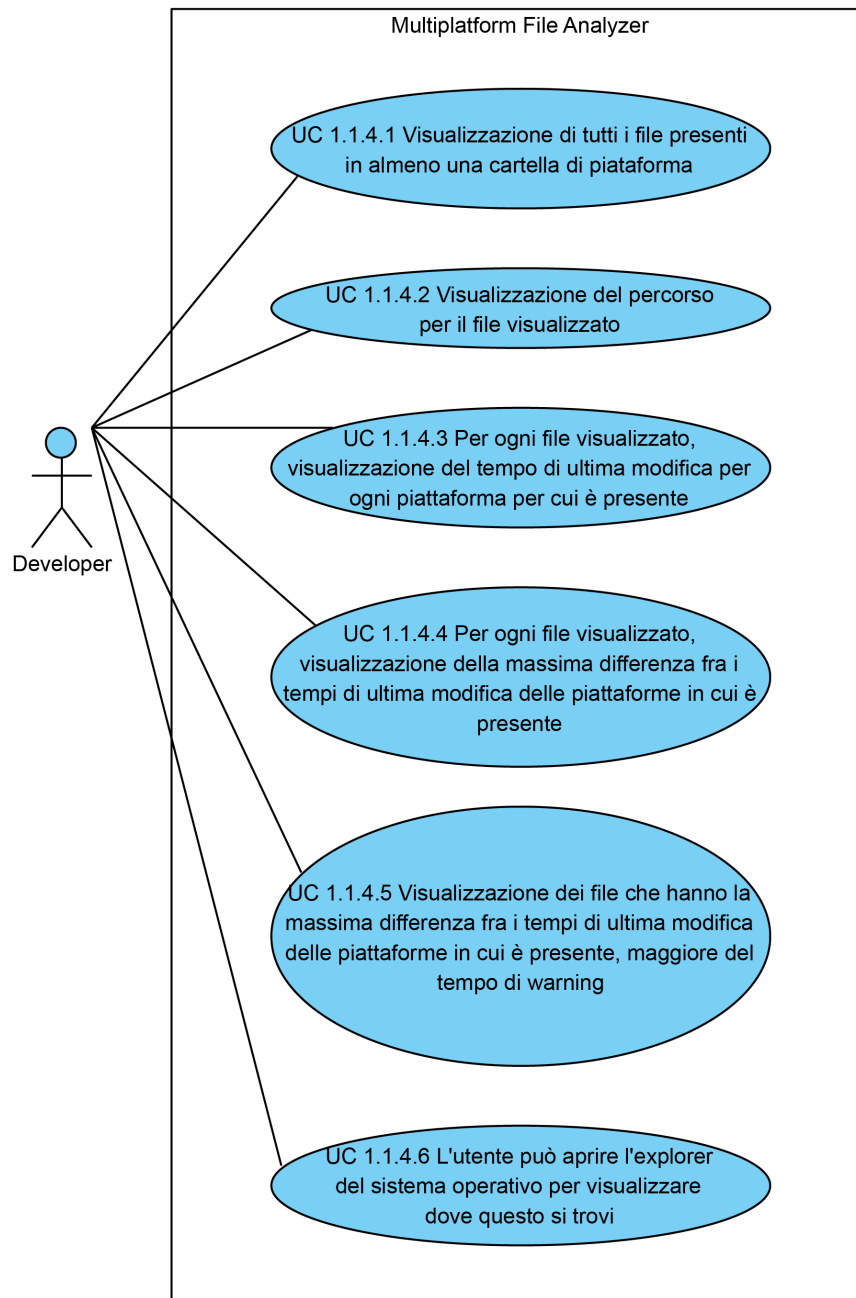


figura 4.4: Use Case - UC 1.1.4: Visualizzazione dei risultati dell'analisi

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

- **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.



4.3. TRACCIAMENTO DEI REQUISITI

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:



4.3. TRACCIAMENTO DEI REQUISITI

- 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 multiplatforma.	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



4.3. TRACCIAMENTO DEI REQUISITI

R0F1.1.4	Il programma deve memorizzare e pre-inserire 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



4.4. TECNOLOGIE E STRUMENTI

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 multiplatforma che offre una vastissima serie di librerie di utilità, grazie alle quali lo sviluppo diviene agevole e veloce, permettendo quasi sempre di non legarsi



4.5. CICLO DI VITA DEL SOFTWARE

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**. È 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.

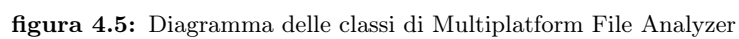
Per permettere una facile e veloce connessione tra gli stati descritti dall'*MVC* è stato adottato il Design Pattern *Observer*, implementato nell'omonimo pacchetto *observer*.

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 dipendenze verso le classi del Framework Qt.





4.6. PROGETTAZIONE

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** si occupa di avviare il sistema. Essa provvede ad allocare tutti gli oggetti necessari e al relativo rilascio al termine del programma. Inoltre seleziona la view e la connette con le classi che rappresentano i dati. Infine 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 stato utilizzato per tenere aggiornata la view al cambiamento dei dati.

Interfaccia IObserver

IObserver è l'interfaccia che rappresenta gli oggetti che osservano le modifiche di altri oggetti. Essa contiene il metodo virtuale puro **Update** che i soggetti osservati invocano per segnalare il fatto che sono stati modificati e che quindi un aggiornamento è necessario. Le classi concrete che avranno la necessità di osservare un soggetto deriveranno da **IObserver** implementando il metodo **Update**.

Classe Subject

La classe **Subject** rappresenta un soggetto che può essere osservato. Essa contiene il codice necessario per notificare tutti gli osservatori dell'avvenuto cambiamento. Mantiene inoltre una lista degli osservatori da notificare, i quali possono iscriversi se desiderano ricevere la notifica, oppure rimuoversi se non è più necessario ricevere aggiornamenti. Le classi che devono 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 disaccoppiato dal mondo Qt, le classi di dati sono state implementate con la Standard Template Library (STL), piuttosto che le 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. È questa classe che si occupa della trasformazione della stringa contenente tutte le piattaforme in un **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 deriva dalla classe **Subject** per permettere ad un osservatore, tipicamente un elemento dell'View, di aggiornarsi al variare dei risultati. Permettendo di mostrare i risultati dell'analisi in modo interattivo durante l'inserimento di ogni nuovo **FileOccurrence** trovato.

4.6.5 Pacchetto view

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

Interfaccia 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 Design 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, accessibile tramite il seguente indirizzo: <https://github.com/Mauxx91/Multiplatform-File-Analyzer>.



4.8. CONCLUSIONI

Tutto il codice è disponibile gratuitamente sotto licenza GNU GPL v3³

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 multiplatforma. 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.

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

Capitolo 5

XML Editor

In questo capitolo è presente una analisi completa del tool XML Editor

5.1 Introduzione

5.1.1 Premessa

Una buona architettura dell'engine di un videogioco, legge dai file di configurazione la maggior parte delle informazioni al posto di scriverle direttamente nel codice di gioco¹. I file di configurazione si possono specializzare per una specifica piattaforma come già visto nel capitolo 4 (todo crea collegamento).

I file di configurazione contengono praticamente di tutto: a partire dalla definizione dei percorsi delle pagine dei menu, alla descrizione degli oggetti grafici in relazione ai comportamenti e animazioni. Tutti questi file sono solitamente tutti scritti in linguaggio XML.

5.1.2 Lo scopo

Vista l'eterogeneità dei file di configurazione, è possibile che siano scritti anche da artisti e/o da persone non specializzate. Inoltre è possibile inserire relazioni tra elementi presenti in questi file. Ad esempio, per un oggetto grafico si specifica quali sono gli oggetti attaccati allo scheletro di animazione, elementi che sono definiti solitamente in un altro XML. Essendo questi file caricati a run-time il debugging diviene non banale, in quanto non è semplice capire da dove l'errore proviene essendo moltissime le variabili in gioco.

Il tool si pone l'obiettivo di rendere la scrittura e il debugging dei file di configurazione XML più semplice e veloce. Innanzitutto fornirà un'editor visuale in cui sarà possibile editare in tutti gli aspetti l'XML. Verrà inoltre fornita la possibilità di specificare le relazioni ed eventuali altri file coinvolti, dopodiché il programma sarà in grado di verificare le relazioni nei file aperti e di crearne di nuove tramite un semplice drag & drop dei nodi destinatari della relazione.

5.2 Casi d'uso

Per meglio capire e tracciare l'esperienza d'uso che un developer di un gioco avrà con il tool sono stati creati dei diagrammi di casi d'uso gerarchici.

¹La pratica di scrivere nel gioco le informazioni è comunemente chiamata *Hard-codding*.



5.2. CASI D'USO

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 attori 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]

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

5.2.1 UC 1: Caso d'uso principale

figura 5.1: Use Case - UC 1: Caso d'uso principale

- **Attori:** developer.
- **Descrizione:** un developer deve avere tutte le funzionalità a disposizione per editare ed verificare uno o più file XML.
- **Precondizione:** il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: creazione di un nuovo file XML (UC 1.1);*
 2. *l'utente ha la possibilità di: apertura di un file XML preesistente (UC 1.2);*
 3. *l'utente ha la possibilità di: editing del file XML correntemente aperto (UC 1.3);*
 4. *l'utente ha la possibilità di: salvare le modifiche di un file modificato (UC 1.4);*
 5. *l'utente ha la possibilità di: editing del filtro sul nome dell'attributo da visualizzare (UC 1.5);*
 6. *l'utente ha la possibilità di: editing dei file associati (UC 1.6);*
 7. *l'utente ha la possibilità di: editing delle relazioni (UC 1.7);*
 8. *l'utente ha la possibilità di: importazione delle relazioni da file (UC 1.8);*
 9. *l'utente ha la possibilità di: esportazione delle relazioni su file (UC 1.9);*
 10. *l'utente ha la possibilità di: avvio del check sulle relazioni (UC 1.10);*
 11. *l'utente ha la possibilità di: visualizzazione del risultato del check sulle relazioni (UC 1.11);*
 12. *l'utente ha la possibilità di: visualizzazione dell'about sul programma (UC 1.12).*
- **Estensioni**
 1. *Visualizzazione degli errori occorsi durante l'apertura del file (UC 1.13);*
 2. *Visualizzazione degli errori occorsi durante il salvataggio del file (UC 1.14);*
 3. *Visualizzazione degli errori occorsi durante l'importazione (UC 1.15);*
 4. *Visualizzazione degli errori occorsi durante l'esportazione (UC 1.16).*
- **Postcondizione:** il sistema ha erogato le funzionalità richieste dal developer.



figura 5.2: Use Case - UC 1.1: Creazione di un nuovo file XML

5.2.2 UC 1.1: Creazione di un nuovo file XML

- **Attori:** developer.
- **Descrizione:** un developer deve poter creare un nuovo file XML.
- **Precondizione:** il developer ha lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi:**
 1. *Inserimento del percorso dove memorizzare il nuovo XML (UC 1.1.1);*
 2. *Inserimento del nome del file del nuovo XML (UC 1.1.2).*
- **Postcondizione:** il sistema ha creato un nuovo file XML in memoria, esso sarà salvato su disco solo nel momento in cui l'utente lo salverà. Il sistema chiude il lavoro corrente e apre il nuovo file creato.

5.2.3 UC 1.1.1: Inserimento del percorso dove memorizzare il nuovo XML

- **Attori:** developer.
- **Descrizione:** un developer deve poter inserire il percorso dove memorizzare su disco il nuovo file XML.
- **Precondizione:** il developer ha selezionato l'azione per creare un nuovo file XML.
- **Scenario principale:** il developer sceglie il percorso dove memorizzare il nuovo file XML.
- **Postcondizione:** il sistema ha acquisito il percorso dove salvare il nuovo file XML.

5.2.4 UC 1.1.2: Inserimento del nome del file del nuovo XML

- **Attori:** developer.
- **Descrizione:** un developer deve poter inserire il nome del nuovo file XML.
- **Precondizione:** il developer ha selezionato l'azione per creare un nuovo file XML.
- **Scenario principale:** il developer inserisce il nome del nuovo file XML.
- **Postcondizione:** il sistema ha acquisito il nome del nuovo file XML.

5.2.5 UC 1.2: Apertura di un file XML preesistente

- **Attori:** developer.
- **Descrizione:** un developer deve poter aprire un file XML preesistente.
- **Precondizione:** il developer ha selezionato l'azione per aprire un file XML.
- **Scenario principale:** il developer inserisce il percorso ed il nome del file XML da aprire.



5.2. CASI D'USO

- **Postcondizione:** il sistema ha chiuso il lavoro precedentemente aperto e l'ha sostituito con il file XML selezionato. Se il file era stato precedentemente aperto ed erano stati settati alcuni file associati allora anche quei file saranno aperti.

5.2.6 UC 1.3 Editing del file XML correntemente aperto

figura 5.3: Use Case - UC 1.3 Editing del file XML correntemente aperto

- **Attori:** developer.
- **Descrizione:** un developer deve poter editare il file XML correntemente visualizzato.
- **Precondizione:** il developer abbia aperto con successo almeno un file XML. Di seguito è specificato quali azioni ha a disposizione l'utente.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: visualizzazione del nodo riferito dal nodo selezionato (UC 1.3.1);*
 2. *l'utente ha la possibilità di: aggiunta di un nuovo nodo figlio al nodo selezionato (UC 1.3.3);*
 3. *l'utente ha la possibilità di: editing del nodo selezionato (UC 1.3.4);*
 4. *l'utente ha la possibilità di: duplicazione del nodo selezionato (UC 1.3.5);*
 5. *l'utente ha la possibilità di: copia del nodo selezionato (UC 1.3.6);*
 6. *l'utente ha la possibilità di: incollare il nodo precedentemente copiato (UC 1.3.7);*
 7. *l'utente ha la possibilità di: rimozione del nodo selezionato (UC 1.3.8);*
 8. *l'utente ha la possibilità di: rimozione di tutti i nodi figli del nodo selezionato (UC 1.3.9);*
 9. *l'utente ha la possibilità di: istanziazione di una nuova relazione (UC 1.3.10);*
 10. *l'utente ha la possibilità di: annullamento dell'ultima modifica eseguita (UC 1.3.12);*
 11. *l'utente ha la possibilità di: ripristino dell'ultima modifica annullata (UC 1.3.13).*
- **Estensioni**
 1. *Visualizzazione degli errori occorsi durante la ricerca del nodo riferito (UC 1.3.2);*
 2. *Visualizzazione degli errori occorsi durante l'istanziazione di una nuova relazione (UC 1.3.11).*
- **Postcondizione:** il sistema ha modificato la versione caricata in memoria del file XML modificato.



5.2.7 UC 1.3.1 Visualizzazione del nodo riferito dal nodo selezionato

- **Attori:** developer.
- **Descrizione:** un developer visualizza il nodo destinatario della relazione che ha il nodo selezionato come punto di partenza.
- **Precondizione:** il developer ha selezionato l'azione seguire la relazione.
- **Scenario principale:** il developer seleziona un nodo e seleziona l'azione per seguire la relazione.
- **Postcondizione:** il sistema ha selezionato il nodo ed espanso l'albero fino al nodo destinatario della relazione, se sono presenti più nodi destinazione o più relazioni da seguire viene sempre usata quella inserita prima nel sistema.

5.2.8 UC 1.3.2 Visualizzazione degli errori occorsi durante la ricerca del nodo riferito

- **Attori:** developer.
- **Descrizione:** si è verificato uno dei seguenti errori durante la ricerca del nodo destinazione:
 - non esiste nessuna relazione per cui il nodo selezionato è riconoscibile come nodo partenza di una relazione;
 - non esiste nessun nodo di destinazione.
- **Precondizione:** il developer ha selezionato l'azione seguire la relazione.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, non viene visualizzato nessun nodo destinazione.

5.2.9 UC 1.3.3 Aggiunta di un nuovo nodo figlio al nodo selezionato

- **Attori:** developer.
- **Descrizione:** viene aggiunto un nuovo nodo vuoto come figlio del nodo correttamente selezionato.
- **Precondizione:** il developer ha selezionato l'azione aggiungere un nuovo nodo come figlio del nodo correttamente visualizzato.
- **Scenario principale:** il developer seleziona un nodo e seleziona l'azione inserire un nuovo figlio, dopodiché inserisce il tag name del nuovo nodo.
- **Postcondizione:** il sistema ha selezionato il nodo ed espanso l'albero fino al nuovo nodo creato.



figura 5.4: Use Case - UC 1.3.4 Editing del nodo selezionato

5.2.10 UC 1.3.4 Editing del nodo selezionato

- **Attori:** developer.
- **Descrizione:** un developer deve poter editare il file XML correntemente visualizzato.
- **Precondizione:** il developer ha selezionato l'azione per modificare l'elemento correntemente selezionato.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: modifica del valore dell'elemento* (UC 1.3.4.1);
 2. *l'utente ha la possibilità di: inserimento di un nuovo attributo* (UC 1.3.4.2);
 3. *l'utente ha la possibilità di: modifica di un attributo esistente* (UC 1.3.4.3);
 4. *l'utente ha la possibilità di: rimozione di un attributo esistente* (UC 1.3.4).
- **Postcondizione:** il sistema ha modificato la versione caricata in memoria del file XML modificato.

5.2.11 UC 1.3.4.1 Modifica del valore dell'elemento

- **Attori:** developer.
- **Descrizione:** il developer può modificare il value dell'elemento.
- **Precondizione:** il developer ha selezionato l'azione aggiungere per modificare un elemento.
- **Scenario principale:** il developer inserisce il nuovo value.
- **Postcondizione:** il sistema ha modificato il value dell'elemento.

5.2.12 UC 1.3.4.2 Inserimento di un nuovo attributo

- **Attori:** developer.
- **Descrizione:** il developer può aggiungere un attributo all'elemento.
- **Precondizione:** il developer ha selezionato l'azione aggiungere per modificare un elemento.
- **Scenario principale:** il developer inserisce il nome del nuovo attributo ed il corrispettivo valore.
- **Postcondizione:** il sistema ha aggiunto un nuovo attributo con il nome ed il valore inseriti dal developer.

5.2.13 UC 1.3.4.3 Modifica di un attributo esistente

- **Attori:** developer.
- **Descrizione:** il developer può modificare il valore e/o il nome di un attributo precedentemente inserito.
- **Precondizione:** il developer ha selezionato l'azione aggiungere per modificare un elemento.



- **Scenario principale:** il developer inserisce il nome e/o il valore dell'attributo da modificare.
- **Postcondizione:** il sistema ha modificato l'attributo selezionato con i dati inseriti dal developer.

5.2.14 UC 1.3.4.4 Rimozione di un attributo esistente

- **Attori:** developer.
- **Descrizione:** il developer può eliminare un attributo precedentemente inserito.
- **Precondizione:** il developer ha selezionato l'azione aggiungere per modificare un elemento.
- **Scenario principale:** il developer seleziona l'attributo da eliminare.
- **Postcondizione:** il sistema ha eliminato l'attributo selezionato dal developer.

5.2.15 UC 1.3.5 Duplicazione del nodo selezionato

- **Attori:** developer.
- **Descrizione:** viene aggiunta al padre del nodo selezionato una copia profonda di questo.
- **Precondizione:** il developer ha selezionato l'azione di duplicazione di un nodo.
- **Scenario principale:** il developer seleziona una nodo e successivamente l'azione di duplicazione.
- **Postcondizione:** il sistema ha aggiunto al padre del nodo selezionato una copia profonda di tale nodo, inoltre ha espanso l'albero fino al nuovo nodo creato.

5.2.16 UC 1.3.6 Copia del nodo selezionato

- **Attori:** developer.
- **Descrizione:** viene fatta una copia profonda del nodo selezionato e conservata in memoria, pronta per essere incollata.
- **Precondizione:** il developer ha selezionato l'azione di copia di un nodo.
- **Scenario principale:** il developer seleziona una nodo e successivamente l'azione di copia.
- **Postcondizione:** il sistema ha memorizzato in memoria una copia profonda del nodo selezionato.

5.2.17 UC 1.3.7 Incollare il nodo precedentemente copiato

- **Attori:** developer.
- **Descrizione:** viene aggiunto come figlio del nodo correttamente selezionato il nodo precedentemente copiato.
- **Precondizione:** il developer ha selezionato l'azione per incollare un nodo dopo averne copiato almeno uno.
- **Scenario principale:** il developer seleziona una nodo e successivamente l'azione per incollare.
- **Postcondizione:** il sistema ha aggiunto come figlio del nodo selezionato il nodo precedentemente copiato, inoltre ha espanso l'albero fino al nuovo nodo aggiunto.



5.2. CASI D'USO

5.2.18 UC 1.3.8 Rimozione del nodo selezionato

- **Attori:** developer.
- **Descrizione:** viene rimosso il nodo correntemente selezionato.
- **Precondizione:** il developer ha selezionato l'azione per rimuovere il nodo corrente.
- **Scenario principale:** il developer seleziona una nodo e successivamente l'azione di rimozione.
- **Postcondizione:** il sistema ha rimosso il nodo attualmente selezionato.

5.2.19 UC 1.3.9 Rimozione di tutti i nodi figli del nodo selezionato

- **Attori:** developer.
- **Descrizione:** vengono rimossi tutti i nodi figli del nodo correntemente selezionato.
- **Precondizione:** il developer ha selezionato l'azione per rimuovere tutti i nodi figli.
- **Scenario principale:** il developer seleziona una nodo e successivamente l'azione di rimozione.
- **Postcondizione:** il sistema ha rimosso tutti i nodi del nodo attualmente selezionato.

5.2.20 UC 1.3.10 Istanziatura di una nuova relazione

- **Attori:** developer.
- **Descrizione:** vengono creati in automatico tutti i nodi necessari per creare una relazione tra i nodi partenza selezionati e il nodo destinatario scelto.
- **Precondizione:** il developer ha selezionato l'azione per creare in automatico una relazione.
- **Scenario principale:** il developer seleziona uno o più nodi e successivamente sceglie il nodo che riferirà i nodi precedentemente selezionati.
- **Postcondizione:** il sistema ha creato in automatico tutti i nodi necessari ad esplicitare le relazioni richieste.

5.2.21 UC 1.3.11 Visualizzazione degli errori occorsi durante l'istanziatura di una nuova relazione

- **Attori:** developer.
- **Descrizione:** si è verificato uno dei seguenti errori durante l'istanziatura di una relazione:
 - non esiste nessuna relazione per cui il nodo selezionato è riconoscibile come nodo partenza di una relazione;
 - sono stati selezionati come nodi destinazione nodi non riconoscibili come destinazione di una relazione.



- non esiste nessun percorso di relazioni che lega il nodo partenza con il nodo destinazione selezionati;
- sono stati selezionati come nodi destinazione nodi di tipi differenti.

.

- **Precondizione:** il developer ha selezionato l'azione per creare una nuova relazione e specificato gli elementi da coinvolgere.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, non viene eseguita nessuna modifica al file XML.

5.2.22 UC 1.3.12 Annullamento dell'ultima modifica eseguita

- **Attori:** developer.
- **Descrizione:** viene annullata l'ultima modifica e ripristinato lo stato precedente del file.
- **Precondizione:** il developer ha selezionato l'azione per annullare l'ultima modifica ed è stata effettuata almeno una modifica al file XML.
- **Scenario principale:** il developer seleziona l'azione per annullare l'ultima modifica.
- **Postcondizione:** il sistema ha ripristinato lo stato del file prima dell'ultima azione di modifica.

5.2.23 UC 1.3.13 Ripristino dell'ultima modifica annullata

- **Attori:** developer.
- **Descrizione:** viene ripristinata l'ultima modifica annullata.
- **Precondizione:** il developer ha selezionato l'azione per ripristinare l'ultima modifica ed è stata annullata almeno una modifica al file XML.
- **Scenario principale:** il developer seleziona l'azione per ripristinare l'ultima modifica.
- **Postcondizione:** il sistema ha ripristinato lo stato del file prima dell'ultima azione di annullamento eseguita.

5.2.24 UC 1.4: Apertura di un file XML preesistente

- **Attori:** developer.
- **Descrizione:** un developer deve poter salvare le modifiche effettuate su un file XML precedentemente aperto.
- **Precondizione:** il developer ha selezionato l'azione per salvare un file XML e abbia almeno un file aperto e modificato.
- **Scenario principale:** il developer seleziona l'azione per salvare un file XML modificato.
- **Postcondizione:** il sistema ha salvato su disco le modifiche apportate al file XML.



5.2. CASI D'USO

5.2.25 UC 1.5 Editing del filtro sul nome dell'attributo da visualizzare

- **Attori:** developer.
- **Descrizione:** un developer deve poter modificare il filtro che seleziona l'attributo cui valore è mostrato nell'albero rappresentante il file XML.
- **Precondizione:** il developer abbia lanciato il tool sotto un sistema Windows 7 o superiore.
- **Scenario principale:** il developer inserisce il nuovo filtro.
- **Postcondizione:** il sistema ha salvato il nuovo filtro e ha aggiornato la visualizzazione di tutti i file aperti.

5.2.26 UC 1.6 Editing dei file associati

figura 5.5: Use Case - UC 1.6 Editing dei file associati

- **Attori:** developer.
- **Descrizione:** un developer deve poter aggiungere e rimuovere file associati.
- **Precondizione:** il developer ha caricato correttamente un file XML.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: aggiunta di un nuovo file associato (UC 1.6.1);*
 2. *l'utente ha la possibilità di: rimozione di un file precedentemente associato (UC1.6.2).*
- **Postcondizione:** il sistema ha modificato i file associati del file XML principale attualmente aperto.

5.2.27 UC 1.6.1 Aggiunta di un nuovo file associato

- **Attori:** developer.
- **Descrizione:** un developer deve poter aggiungere un nuovo file associato al corrente file principale.
- **Precondizione:** il developer abbia selezionato l'azione di modifica dei file associati.
- **Scenario principale:** il developer inserisce il percorso ed il nome del nuovo file associato.
- **Postcondizione:** il sistema ha salvato il nuovo file associato e tenta di caricarlo.



5.2.28 UC 1.6.2 Rimozione di un file precedentemente associato

- **Attori:** developer.
- **Descrizione:** un developer deve poter rimuovere un file associato dal corrente file principale.
- **Precondizione:** il developer abbia selezionato l'azione di modifica dei file associati.
- **Scenario principale:** il developer seleziona il file da rimuovere.
- **Postcondizione:** il sistema ha rimosso il file selezionato. La copia in memoria del file non è stata eliminata.

5.2.29 UC 1.7 Editing delle relazioni

figura 5.6: Use Case - UC 1.7 Editing delle relazioni

- **Attori:** developer.
- **Descrizione:** un developer deve poter editare le relazioni presenti.
- **Precondizione:** il developer ha lanciato il tool sotto un sistema Windows 7 o superiore.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: inserimento di una nuova relazione (UC 1.7.1);*
 2. *l'utente ha la possibilità di: modifica di una relazione precedentemente inserita (UC1.7.2);*
 3. *l'utente ha la possibilità di: rimozione di una relazione precedentemente inserita (UC1.7.3).*
- **Postcondizione:** il sistema ha modificato le relazione come desiderato dall'utente.

5.2.30 UC 1.7.1 Inserimento di una nuova relazione

- **Attori:** developer.
- **Descrizione:** un developer deve poter aggiungere una nuova relazione.
- **Precondizione:** il developer abbia selezionato l'azione di modifica delle relazioni.
- **Scenario principale:** il developer inserisce tutte le informazioni necessarie alla creazione di una relazione.
- **Postcondizione:** il sistema ha memorizzato la nuova relazione.



5.2. CASI D'USO

5.2.31 UC 1.7.2 Modifica di una relazione precedentemente inserita

- **Attori:** developer.
- **Descrizione:** un developer deve poter modificare una relazione precedentemente inserita.
- **Precondizione:** il developer abbia selezionato l'azione di modifica delle relazioni e la relazione da modificare.
- **Scenario principale:** il developer inserisce tutte le informazioni che vuole modificare della relazione.
- **Postcondizione:** il sistema ha memorizzato i cambiamenti nella relazione selezionata.

5.2.32 UC 1.7.3 Rimozione di una relazione precedentemente inserita

- **Attori:** developer.
- **Descrizione:** un developer deve poter rimuovere una relazione.
- **Precondizione:** il developer abbia selezionato l'azione di modifica delle relazioni.
- **Scenario principale:** il developer seleziona la relazione da rimuovere.
- **Postcondizione:** il sistema ha rimosso la relazione selezionata.

5.2.33 UC 1.8 Importazione delle relazioni da file

- **Attori:** developer.
- **Descrizione:** un developer deve poter importare e sostituire le relazioni correnti con le relazioni contenute in un file di configurazione precedentemente esportato.
- **Precondizione:** il developer ha selezionato l'azione per importare le relazioni da un file.
- **Scenario principale:** il developer inserisce il percorso ed il nome del file contenente le relazioni.
- **Postcondizione:** il sistema ha rimosso tutte le precedenti relazioni e le ha sostituite con tutte quelle presenti nel file.

5.2.34 UC 1.9 Esportazione delle relazioni su file

- **Attori:** developer.
- **Descrizione:** un developer deve poter esportare le relazioni attualmente inserite su file.
- **Precondizione:** il developer ha selezionato l'azione per esportare le relazioni su file.
- **Scenario principale:** il developer inserisce il percorso ed il nome del file dove esportare le relazioni.
- **Postcondizione:** il sistema esportato le relazioni nel file selezionato.



5.2.35 UC 1.10 Avvio del check sulle relazioni

- **Attori:** developer.
- **Descrizione:** un developer deve poter verificare la consistenza delle relazioni presenti nel file principale e in tutti i file associati sulla base delle relazioni inserite.
- **Precondizione:** il developer ha selezionato l'azione per verificare la consistenza delle relazioni. Almeno un file XML deve essere stato aperto con successo.
- **Scenario principale:** il developer seleziona l'azione per effettuare il check delle relazioni.
- **Postcondizione:** il sistema ha verificato le relazioni nel file principale e in tutti gli associati.

5.2.36 UC 1.11 Visualizzazione del risultato del check sulle relazioni

figura 5.7: Use Case - UC 1.11 Visualizzazione del risultato del check sulle relazioni

- **Attori:** developer.
- **Descrizione:** un developer deve visualizzare l'esito del check sulle relazioni.
- **Precondizione:** il developer ha avviato il check sulle relazioni e quest'ultimo è terminato.
- **Flusso principale degli eventi:**
 1. *l'utente ha la possibilità di: visualizzazione degli errori (UC 1.11.1);*
 2. *l'utente ha la possibilità di: selezione dei filtri sulle categorie di errori da visualizzare (UC1.11.2);*
 3. *l'utente ha la possibilità di: visualizzazione dell'elemento causa dell'errore nell'albero XML (UC1.11.3).*
- **Postcondizione:** il sistema ha permesso la visualizzazione dei risultati del check sulle relazioni.

5.2.37 UC 1.11.1 Visualizzazione degli errori

- **Attori:** developer.
- **Descrizione:** un developer deve poter visualizzare la finestra di about sul programma.
- **Precondizione:** il developer ha avviato il check sulle relazioni e quest'ultimo è terminato.
- **Scenario principale:** il developer visualizza per ogni relazione: la severità dell'errore, la tipologia, la descrizione, il nome del file in cui l'errore è stato trovato.
- **Postcondizione:** il developer ha visualizzato l'output della verifica delle relazioni.



5.2. CASI D'USO

5.2.38 UC 1.11.2 Selezione dei filtri sulle categorie di errori da visualizzare

- **Attori:** developer.
- **Descrizione:** un developer deve poter selezionare i filtri sulla severità degli errori.
- **Precondizione:** il developer ha avviato il check sulle relazioni e quest'ultimo è terminato.
- **Scenario principale:** il developer seleziona per ogni categoria di severità se desidera visualizzare gli errori di quella categoria.
- **Postcondizione:** la visualizzazione viene aggiornata e vengono mostrati solo gli errori per il quale corrispettivo filtro di severità è attivo.

5.2.39 UC 1.11.3 Visualizzazione dell'elemento causa dell'errore nell'albero XML

- **Attori:** developer.
- **Descrizione:** un developer deve poter visualizzare l'elemento causa dell'errore nel suo albero XML di appartenenza.
- **Precondizione:** il developer ha avviato il check sulle relazioni seleziona l'azione per mostrare l'elemento causa dell'errore.
- **Scenario principale:** il developer seleziona l'azione per visualizzare il nodo causa dell'errore nel suo albero di appartenenza.
- **Postcondizione:** il developer ha visualizzato il nodo nel contesto del suo albero di appartenenza.

5.2.40 UC 1.12 Visualizzazione dell'about sul programma

- **Attori:** developer.
- **Descrizione:** un developer deve poter visualizzare la finestra di about sul programma.
- **Precondizione:** il developer ha selezionato l'azione per visualizzare la finestra di about sul programma.
- **Scenario principale:** il developer visualizza l'about sul programma.
- **Postcondizione:** il sistema ha mostrato la finestra di about sul programma.

5.2.41 UC 1.13 Visualizzazione degli errori occorsi durante l'apertura del file

- **Descrizione:** si è verificato uno dei seguenti errori durante l'apertura di un file XML:
 - il percorso inserito non è valido;
 - il file indicato non esiste;
 - il file indicato non è leggibile;
 - si è verificato un errore non gestito durante l'apertura del file;
 - il parser XML ha trovato un errore sintattico nel file XML.



- **Precondizione:** il developer ha selezionato l'azione per aprire un file esistente e abbia inserito il percorso ed il nome del file.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, il file XML non è stato aperto. Il precedente lavoro è stato chiuso.

5.2.42 UC 1.14 Visualizzazione degli errori occorsi durante il salvataggio del file

- **Descrizione:** si è verificato uno dei seguenti errori durante il salvataggio su disco di un file XML:
 - il file indicato non è scrivibile;
 - si è verificato un errore non gestito durante l'apertura in scrittura del file.
- **Precondizione:** il developer ha selezionato l'azione per il salvataggio del file modificato.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, il file XML non è stato salvato. Il precedente lavoro non è stato perso.

5.2.43 UC 1.15 Visualizzazione degli errori occorsi durante l'importazione

- **Descrizione:** si è verificato uno dei seguenti errori durante l'importazione delle relazioni:
 - il percorso inserito non è valido;
 - il file indicato non esiste;
 - il file indicato non è leggibile;
 - si è verificato un errore non gestito durante l'apertura del file;
 - il parser XML ha trovato un errore sintattico nel file.
- **Precondizione:** il developer ha selezionato l'azione per l'importazione delle relazioni da file e ha inserito un percorso ed il nome del file.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, le relazioni non sono state importate. Le precedenti relazioni non sono state eliminate.

5.2.44 UC 1.16 Visualizzazione degli errori occorsi durante l'esportazione

- **Descrizione:** si è verificato uno dei seguenti errori durante l'esportazione delle relazioni su file:
 - il file indicato non è scrivibile;
 - si è verificato un errore non gestito durante l'apertura in scrittura del file.



5.3. TRACCIAMENTO DEI REQUISITI

- **Precondizione:** il developer ha selezionato l'azione per l'esportazione delle relazioni su file.
- **Scenario principale:** viene fornita una spiegazione dell'errore commesso e come risolverlo.
- **Postcondizione:** l'errore commesso è stato visualizzato e spiegato, le relazioni non sono state esportate.

5.3 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.



5.3.1 Requisiti funzionali

Requisito	Descrizione	Fonti
R0F1	Un utente può effettuare l'analisi dei file di un gioco multiplatforma.	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 pre-inserire 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



5.3. TRACCIAMENTO DEI REQUISITI

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 5.1: Requisiti funzionali

5.3.2 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 5.2: Requisiti di vincolo

5.3.3 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 5.3: Requisiti di vincolo

5.4 Tecnologie e strumenti

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

5.4.1 Qt Framework

Per lo sviluppo del tool è stato usato Qt Framework versione 5.3.1. Qt è un potente Framework multiplatforma 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.

5.4.2 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.

5.5 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.

5.6 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*. È inoltre presente il pacchetto *core*



5.6. PROGETTAZIONE

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.

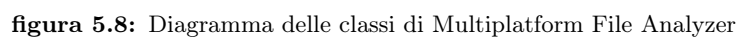
Per permettere una facile e veloce connessione tra gli stati descritti dall'*MVC* è stato adottato il Design Pattern *Observer*, implementato nell'omonimo pacchetto *observer*.

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.

5.6.1 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 dipendenze verso le classi del Framework Qt.





5.6. PROGETTAZIONE

5.6.2 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** si occupa di avviare il sistema. Essa provvede ad allocare tutti gli oggetti necessari e al relativo rilascio al termine del programma. Inoltre seleziona la view e la connette con le classi che rappresentano i dati. Infine raccoglie i comandi utente provenienti dalla view e li esegue grazie al pacchetto **core**.

5.6.3 Pacchetto observer

Questo pacchetto rappresenta l'implementazione del Design Pattern *Observer* ed è stato utilizzato per tenere aggiornata la view al cambiamento dei dati.

Interfaccia IObserver

IObserver è l'interfaccia che rappresenta gli oggetti che osservano le modifiche di altri oggetti. Essa contiene il metodo virtuale puro **Update** che i soggetti osservati invocano per segnalare il fatto che sono stati modificati e che quindi un aggiornamento è necessario. Le classi concrete che avranno la necessità di osservare un soggetto deriveranno da **IObserver** implementando il metodo **Update**.

Classe Subject

La classe **Subject** rappresenta un soggetto che può essere osservato. Essa contiene il codice necessario per notificare tutti gli osservatori dell'avvenuto cambiamento. Mantiene inoltre una lista degli osservatori da notificare, i quali possono iscriversi se desiderano ricevere la notifica, oppure rimuoversi se non è più necessario ricevere aggiornamenti. Le classi che devono essere osservate deriveranno da questa.

5.6.4 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 disaccoppiato dal mondo Qt, le classi di dati sono state implementate con la Standard Template Library (STL), piuttosto che le 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. È questa classe che si occupa della trasformazione della stringa contenente tutte le piattaforme in un **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 deriva dalla classe **Subject** per permettere ad un osservatore, tipicamente un elemento dell'View, di aggiornarsi al variare dei risultati. Permettendo di mostrare i risultati dell'analisi in modo interattivo durante l'inserimento di ogni nuovo **FileOccurrence** trovato.

5.6.5 Pacchetto view

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

Interfaccia 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.

5.6.6 Core

Il pacchetto si occupa della realizzazione dell'analisi e di tutti gli algoritmi di business del tool. Utilizza il Design 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.

5.7 Codifica

Tutto il codice del tool è stato pubblicato sulla piattaforma GitHub, accessibile tramite il seguente indirizzo: <https://github.com/Mauxx91/Multiplatform-File-Analyzer>.



5.8. CONCLUSIONI

Tutto il codice è disponibile gratuitamente sotto licenza GNU GPL v3²

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?

5.8 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 multiplatforma. 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.

²Un'approfondimento sulla licenza può essere trovato al seguente indirizzo: <http://www.gnu.org/copyleft/gpl.html>.

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