

# DON'T PANIC

3DMob: Grafica 3D su device mobili

---



## Norme di progetto

### Informazioni sul documento

---

Versione	5.2.0
Redazione	Busato Luca
Verifica	Lain Daniele Rampazzo Federico
Responsabile	Cesarato Fabio
Uso	Interno
Lista di distribuzione	Don't Panic

### Descrizione

Documento riguardante le norme stabilite dal gruppo Don't Panic per la realizzazione di 3DMob



## Diario delle modifiche

Descrizione modifica	Autore	Ruolo	Data	Versione
Approvazione documento	Cesarato Fabio	Responsabile	2013-03-8	5.2.0
Verifica documento	Rampazzo Federico	Verificatore	2013-03-7	5.1.1
Verifica documento	Lain Daniele	Verificatore	2013-03-7	5.1.0
Modificate le convenzioni tipografiche relative al documento <i>Definizione di Prodotto</i>	Busato Luca	Amministratore	2013-03-6	5.0.2
Aggiunta sezione relativa alla gestione dei bug	Busato Luca	Amministratore	2013-03-5	5.0.1
Aggiunta sezione relativa alla validazione degli output XML <sub>G</sub> e JSON <sub>G</sub>	Busato Luca	Amministratore	2013-03-14	5.0.1
Revisione norme in base a quanto segnalato in revisione di qualifica	Busato Luca	Amministratore	2013-03-14	5.0.0
Approvazione documento	Basaglia Mattia	Responsabile	2013-02-22	4.2.0
Verifica documento	Pezzutti Marco	Verificatore	2013-02-21	4.1.1
Verifica documento	Busato Luca	Verificatore	2013-02-11	4.1.0
Revisione norme per la progettazione di dettaglio	Cesarato Fabio	Amministratore	2013-02-07	4.0.0
Approvazione documento	Pezzutti Marco	Responsabile	2013-01-20	3.2.0
Verifica documento	Sciarrone Riccardo	Verificatore	2013-01-18	3.1.1
Verifica documento	Cesarato Fabio	Verificatore	2013-01-17	3.1.0
Definizione dettagliata utilizzo strumenti	Basaglia Mattia	Amministratore	2013-01-16	3.0.3
Stesura gestione cambiamenti	Rampazzo Federico	Amministratore	2013-01-15	3.0.2
Definizione ruoli con relative responsabilità e vincoli	Rampazzo Federico	Amministratore	2013-01-13	3.0.1
Correzione struttura e contenuti basata su errori segnalati dal committente	Basaglia Mattia	Amministratore	2013-01-12	3.0.0
Approvazione documento	Busato Luca	Responsabile	2013-01-07	2.2.0
Verifica documento	Lain Daniele	Verificatore	2013-01-05	2.1.0



Stesura diagrammi di flusso	Sciarrone Riccardo	Amministratore	2013-01-03	2.0.0
Approvazione documento	Cesarato Fabio	Responsabile	2012-11-25	1.2.0
Verifica documento	Lain Daniele	Verificatore	2012-11-24	1.1.1
Verifica documento	Busato Luca	Verificatore	2012-11-23	1.1.0
Modificata sezione versionamento	Pezzutti Marco	Amministratore	2012-11-23	1.0.11
Stesura ambiente di lavoro, progettazione, introduzione, codifica, glossario	Rampazzo Federico	Amministratore	2012-11-22	1.0.10
Stesura ambiente di lavoro	Rampazzo Federico	Amministratore	2012-11-22	1.0.9
Analisi dei requisiti	Sciarrone Riccardo	Amministratore	2012-11-22	1.0.8
Aggiunto albero del repository	Sciarrone Riccardo	Amministratore	2012-11-22	1.0.7
Correzioni norme di comunicazione e riunione	Pezzutti Marco	Amministratore	2012-11-22	1.0.6
Stesura norme di comunicazione e riunione	Sciarrone Riccardo	Amministratore	2012-11-22	1.0.5
Stesura norme di ciclo di vita dei documenti	Pezzutti Marco	Amministratore	2012-11-21	1.0.4
Descrizione software per il Project Management e norme utilizzo sistema di ticketing	Basaglia Mattia	Amministratore	2012-11-21	1.0.3
Stesura norme repository	Sciarrone Riccardo	Amministratore	2012-11-21	1.0.2
Stesura norme tipografiche	Basaglia Mattia	Amministratore	2012-11-21	1.0.1
Creazione scheletro del documento e stesura parziale di documenti	Rampazzo Federico	Amministratore	2012-11-20	1.0.0



## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del Prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti . . . . .	1
1.4.1	Informativi . . . . .	1
<b>2</b>	<b>Collaborazione</b>	<b>2</b>
2.1	Comunicazioni . . . . .	2
2.1.1	Comunicazioni esterne . . . . .	2
2.1.2	Comunicazioni interne . . . . .	2
2.1.3	Composizione e-mail . . . . .	2
2.2	Riunioni . . . . .	3
2.2.1	Frequenza . . . . .	3
2.2.2	Convocazione riunione . . . . .	3
2.2.3	Svolgimento riunione . . . . .	4
2.2.4	Verbale . . . . .	4
2.3	Repository e strumenti per la condivisione di file . . . . .	5
2.3.1	Repository . . . . .	5
2.3.2	Condivisione file . . . . .	5
<b>3</b>	<b>Documentazione</b>	<b>6</b>
3.1	Template . . . . .	6
3.2	Struttura del documento . . . . .	6
3.2.1	Prima pagina . . . . .	6
3.2.2	Diario delle modifiche . . . . .	6
3.2.3	Indici . . . . .	7
3.2.4	Formattazione generale delle pagine . . . . .	7
3.3	Norme tipografiche . . . . .	7
3.3.1	Punteggiatura . . . . .	7
3.3.2	Stile di testo . . . . .	7
3.3.3	Composizione del testo . . . . .	8
3.3.4	Formati . . . . .	8
3.3.5	Sigle . . . . .	9
3.4	Componenti grafiche . . . . .	10
3.4.1	Tabelle . . . . .	10
3.4.2	Immagini . . . . .	10
3.5	Classificazione documenti . . . . .	10
3.5.1	Documenti formali . . . . .	10
3.5.2	Documenti informali . . . . .	10
3.6	Versionamento documenti . . . . .	10
3.7	Ciclo di vita . . . . .	12
3.8	Glossario . . . . .	12



<b>4</b>	<b>Ruoli di progetto</b>	<b>13</b>
4.1	Responsabile di Progetto . . . . .	13
4.2	Amministratore . . . . .	13
4.3	Analista . . . . .	14
4.4	Progettista . . . . .	14
4.5	Verificatore . . . . .	14
4.6	Programmatore . . . . .	15
<b>5</b>	<b>Procedure a supporto dei processi</b>	<b>16</b>
5.1	Gestione di progetto . . . . .	16
5.1.1	Pianificazione delle attività . . . . .	16
5.1.2	Coordinazione e controllo delle attività . . . . .	16
5.1.3	Gestione e controllo delle risorse . . . . .	16
5.1.4	Analisi e Gestione dei rischi . . . . .	16
5.1.5	Gestione bug . . . . .	17
5.1.6	Elaborazione dati . . . . .	17
5.1.7	Delega . . . . .	17
5.1.8	Responsabilità di sotto-progetto . . . . .	17
5.2	Analisi dei Requisiti . . . . .	18
5.2.1	Studio di Fattibilità e Analisi dei Rischi . . . . .	18
5.2.2	Analisi dei requisiti . . . . .	18
5.3	Progettazione . . . . .	20
5.3.1	Specifica Tecnica . . . . .	20
5.3.2	Definizione di Prodotto . . . . .	20
5.4	Verifica . . . . .	22
5.4.1	Metriche per gli errori riscontrati e gestione dei cambiamenti . . . . .	22
5.4.2	Verifica dei processi . . . . .	23
5.4.3	Verifica dei documenti . . . . .	24
5.4.4	Verifica del tracciamento dei requisiti . . . . .	25
5.4.5	Verifica dei diagrammi UML . . . . .	25
5.4.6	Verifica della progettazione architeturale . . . . .	25
5.4.7	Verifica della progettazione di dettaglio . . . . .	25
5.4.8	Verifica del codice . . . . .	26
5.4.9	Resoconto bug . . . . .	27
5.4.10	Validazione output . . . . .	27
5.5	Codifica . . . . .	27
5.5.1	Codifica e convenzioni . . . . .	27
5.5.2	Documentazione . . . . .	28
<b>6</b>	<b>Ambiente di lavoro</b>	<b>29</b>
6.1	Coordinamento . . . . .	29
6.1.1	Software di gestione del progetto . . . . .	29
6.1.2	Software di versionamento . . . . .	29
6.1.3	Software di Integrazione Continua . . . . .	30
6.1.4	Condivisione dei file . . . . .	30
6.1.5	Google Calendar . . . . .	31
6.2	Pianificazione . . . . .	31
6.2.1	Modalità di utilizzo . . . . .	31
6.3	Strumenti per i documenti . . . . .	32
6.3.1	L <sup>A</sup> T <sub>E</sub> X . . . . .	32



6.3.2	Script . . . . .	32
6.3.3	Verifica . . . . .	33
6.3.4	Diagrammi UML . . . . .	33
6.3.5	Fogli di calcolo . . . . .	34
6.4	Tracy . . . . .	35
6.4.1	Analisi dei Requisiti . . . . .	35
6.4.2	Progettazione e codifica . . . . .	36
6.4.3	Grafico PDCA . . . . .	37
6.4.4	Test . . . . .	38
6.4.5	Tracciamento . . . . .	38
6.5	Protocollo per lo sviluppo dell'applicazione . . . . .	38
6.5.1	Creare un nuovo progetto . . . . .	38
6.5.2	Creazione ticket . . . . .	39
6.5.3	Aggiornamento ticket . . . . .	41
6.5.4	Consigli di utilizzo . . . . .	45
6.6	Strumenti per lo sviluppo dell'applicazione . . . . .	45
6.6.1	Framework . . . . .	45
6.6.2	Documentazione . . . . .	45
6.6.3	Validazione output . . . . .	46
6.7	Strumenti per la codifica . . . . .	46
6.7.1	Stesura . . . . .	46
6.7.2	Verifica . . . . .	46
6.7.3	Scrittura dei test . . . . .	46
<b>A Lista di controllo</b>		<b>48</b>
<b>B Screenshot Tracy</b>		<b>50</b>



## Elenco delle figure

1	Tracy - screenshot pagina principale . . . . .	35
2	Diagramma di attività - Casi d'uso / Requisiti . . . . .	36
3	Diagramma di attività - Classi / Metodi . . . . .	37
4	Diagramma attività - Creazione nuovo ticket . . . . .	42
5	Diagramma attività - Aggiornamento ticket esistente . . . . .	43
6	Tracy - screenshot gestione casi d'uso . . . . .	50
7	Tracy - screenshot gestione classi . . . . .	51
8	Tracy - screenshot gestione package . . . . .	52
9	Tracy - screenshot gestione test d'integrazione . . . . .	52
10	Tracy - screenshot lista comandi di esportazione in $\text{\LaTeX}$ . . . . .	53
11	Tracy - screenshot esportazione in $\text{\LaTeX}$ delle classi . . . . .	53



# 1 Introduzione

## 1.1 Scopo del documento

Questo documento definisce le norme che i membri del gruppo Don't Panic adotteranno nello svolgimento del progetto 3DMob.

Tutti i membri sono tenuti a leggere il documento e a seguire le norme per migliorare l'uniformità del materiale prodotto, migliorare l'efficienza e ridurre il numero di errori. In particolare verranno definite norme riguardanti:

- Interazioni tra membri del gruppo;
- Stesura documenti e convenzioni;
- Modalità di lavoro durante le varie fasi del progetto;
- Ambiente di lavoro.

## 1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un'applicazione in grado di convertire file prodotti da programmi di grafica 3D in file in formato JSON<sub>G</sub> in grado di essere visualizzati su dispositivi mobile senza perdita di informazione. L'obiettivo è quello di semplificare il workflow attuale necessario a rendere compatibili i file.

## 1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite, sono riportate nel documento *Glossario v5.2.0*.

Ogni occorrenza di vocaboli presenti nel *Glossario* è marcata da una "G" maiuscola in pedice.

## 1.4 Riferimenti

### 1.4.1 Informativi

- Specifiche UTF-8<sub>G</sub>:  
<http://www.unicode.org/versions/Unicode6.1.0/ch03.pdf>;
- Licenza GNU LGPL<sub>G</sub> 2.1: <http://www.gnu.org/licenses/lgpl-2.1.html>;
- Qt Coding Conventions<sub>G</sub>:  
<http://qt-project.org/wiki/Coding-Conventions>;
- Qt<sub>G</sub> Coding Style: [http://qt-project.org/wiki/Qt\\_Coding\\_Style](http://qt-project.org/wiki/Qt_Coding_Style);
- ISO<sub>G</sub> 8601: [http://it.wikipedia.org/wiki/ISO\\_8601](http://it.wikipedia.org/wiki/ISO_8601);
- Jenkins: [http://en.wikipedia.org/wiki/Jenkins\\_\(software\)](http://en.wikipedia.org/wiki/Jenkins_(software));
- Piano di Progetto: *Piano di Progetto v5.2.0*;
- Piano di Qualifica: *Piano di Qualifica v5.2.0*.





## 2 Collaborazione

### 2.1 Comunicazioni

#### 2.1.1 Comunicazioni esterne

Per le comunicazioni esterne è stata creata una casella di posta elettronica:

[dont.panic.swe@gmail.com](mailto:dont.panic.swe@gmail.com)

Tale indirizzo deve essere l'unico canale di comunicazione esistente tra il gruppo di lavoro e l'esterno. Solo il *Responsabile di Progetto* ha accesso all'indirizzo ed è quindi l'unico a poter comunicare con il committente del progetto. È compito del *Responsabile di Progetto* informare i membri del gruppo delle discussioni avvenute e inoltrare, qualora sia necessario, il messaggio alla mailing list<sub>G</sub> ufficiale.

#### 2.1.2 Comunicazioni interne

Per le comunicazioni interne è stata creata una mailing list<sub>G</sub>:

[dtp@myopenproject.it](mailto:dtp@myopenproject.it)

I membri del gruppo sono quindi tenuti ad utilizzare l'indirizzo [dtp@myopenproject.it](mailto:dtp@myopenproject.it) per comunicare tra loro. Ogni componente è quindi costantemente informato sullo scambio di informazioni interne. Questo strumento va utilizzato correttamente e solo per questioni inerenti al progetto. Un redirect provvederà ad inoltrare le mail destinate a [dtp@myopenproject.it](mailto:dtp@myopenproject.it) al fine di mantenere uno storico, consultabile online mediante la webmail<sub>G</sub>:

<http://webmail.myopenproject.it>

Con lo scopo di facilitare la comunicazione tra i membri del gruppo, viene adottato l'uso di sistemi di instant messaging<sub>G</sub> e videoconferenza quali Skype e Google Plus, con l'obbligo di redigere un verbale da inviare alla mailing list una volta terminata la conversazione. La verbalizzazione ha l'obiettivo di tenere traccia di ogni argomento discusso all'interno del gruppo, in quanto una comunicazione verbale non documentata non è accettabile per il corretto svolgimento del progetto.

È sconsigliato l'uso del telefono e degli SMS, ma in caso di necessità è possibile ricorrere a tali strumenti. Si chiede che la conversazione venga documentata tramite mail nel caso vengano prese decisioni o emergano contenuti utili allo sviluppo del progetto.

#### 2.1.3 Composizione e-mail

In questo paragrafo viene descritta la struttura che deve avere un messaggio sia per una comunicazione interna che esterna.

##### 2.1.3.1 Destinatario

- **Interno:** l'unico indirizzo utilizzabile è [dtp@myopenproject.it](mailto:dtp@myopenproject.it);
- **Esterno:** può variare a seconda che ci si debba riferire al Proponente, al Prof. Vardanega Tullio o al Prof. Cardin Riccardo.



### 2.1.3.2 Mittente

- **Interno:** l'indirizzo personale di chi scrive il messaggio;
- **Esterno:** l'unico indirizzo utilizzabile è [dont.panic.swe@gmail.com](mailto:dont.panic.swe@gmail.com) e deve essere usato solamente dal *Responsabile di Progetto*.

### 2.1.3.3 Oggetto

L'oggetto deve essere chiaro ed esaustivo, possibilmente stringato e non confondibile con altri preesistenti. Nel caso si debba comporre un messaggio di risposta vi è l'obbligo di aggiungere la particella "Re:" all'inizio dell'oggetto per essere in grado di distinguere il livello di risposta; se si dovesse trattare di un inoltro si deve usare invece la particella "I:". In ogni caso, l'oggetto di una comunicazione, una volta avviata, non deve mai essere cambiato.

### 2.1.3.4 Corpo

Il corpo di un messaggio deve contenere tutte le informazioni necessarie a rendere facilmente comprensibile l'argomento trattato a tutti i membri del gruppo.

Se alcune parti del messaggio hanno uno o più destinatari specifici, il loro nome dovrà essere aggiunto all'inizio del paragrafo tramite la seguente segnatura: *@destinatario*.

In caso di risposta od inoltro del messaggio, il contenuto aggiunto deve essere sempre messo in testa (per non costringere gli altri membri a dover scorrere tutta la mail). Si consiglia di non cancellare il resto del messaggio, per consentire una visione completa della discussione.

### 2.1.3.5 Allegati

Viene permesso l'uso di allegati qualora ve ne sia la necessità. Essi possono essere usati ad esempio per allegare il verbale di una comunicazione via instant messaging<sub>G</sub> tra componenti del gruppo di lavoro o per il resoconto di un incontro con il proponente o con il committente.

## 2.2 Riunioni

### 2.2.1 Frequenza

Le riunioni del gruppo di lavoro avranno una frequenza almeno quindicinale.

### 2.2.2 Convocazione riunione

Il *Responsabile di Progetto* ha il compito di convocare le riunioni generali, ovvero le riunioni in cui vengono convocati tutti i membri del gruppo, ed eventualmente di valutare se anticipare la naturale scadenza della successiva riunione.

Qualora ve ne sia la necessità, qualsiasi componente può richiedere la convocazione di una riunione, ma tale richiesta deve essere inoltrata al *Responsabile di Progetto* che può decidere se accettarla o respingerla. È inoltre possibile e auspicabile che possano essere necessarie riunioni tra specifici membri: ad esempio, nella fase di progettazione può essere utile la collaborazione tra *Progettista* e *Analista*, senza richiedere la presenza di persone non direttamente coinvolte, che verranno comunque informate delle decisioni prese tramite verbale inviato mediante posta elettronica.



Il responsabile deve convocare l'assemblea con almeno tre giorni di preavviso attraverso l'invio una email a [ntp@myopenproject.it](mailto:ntp@myopenproject.it) contenente

- **Oggetto:** Convocazione riunione n. X, dove X indica il numero crescente di riunioni effettuate;
- **Corpo:**
  - **Data:** data e ora prevista;
  - **Luogo:** luogo previsto;
  - **Tipo:** ordinaria/straordinaria;
  - **Ordine del giorno:** elenco numerato delle varie voci da esaminare.

Ogni componente del gruppo deve rispondere al messaggio nel minor tempo possibile confermando la presenza o motivando un'eventuale assenza. In caso di mancata risposta il *Responsabile di Progetto* ha il dovere di contattare anche telefonicamente colui che non ha fornito una risposta in tempo utile. Il *Responsabile di Progetto*, una volta ricevute le risposte e verificata la presenza o assenza dei membri richiesti, può decidere se confermare, annullare o spostare la riunione, per permettere la partecipazione agli eventuali assenti. La conferma, l'annullamento e lo spostamento dell'assemblea devono essere effettuati tramite email.

### 2.2.3 Svolgimento riunione

All'apertura della riunione, verificata la presenza dei membri previsti, viene scelto un segretario che avrà il compito di annotare ogni argomento trattato e di redigere il verbale dell'assemblea, che dovrà poi essere inviato ai restanti componenti del gruppo.

Tutti i partecipanti devono osservare un comportamento consono al miglior svolgimento della riunione e al raggiungimento degli obiettivi della stessa. Il segretario deve inoltre controllare che venga seguito l'ordine del giorno in modo da non tralasciare alcun punto.

### 2.2.4 Verbale

#### 2.2.4.1 Riunione interna

Il verbale di riunione interna è un documento interno informale che consente di tracciare gli argomenti discussi durante la riunione.

Verrà redatto dal segretario della riunione, ruolo scelto di volta in volta a rotazione tra i presenti.

Si consiglia di redigere il verbale di una riunione interna in un documento di testo da condividere con Google Drive<sub>G</sub>, in modo da rendere sempre disponibile il contenuto dello stesso.

Il verbale deve anche essere inviato a tutto il gruppo di lavoro tramite posta elettronica allegando il documento ad un messaggio di risposta all'email di convocazione dell'assemblea.

#### 2.2.4.2 Riunione esterna

In caso di riunione con il committente od il proponente, il verbale è un documento ufficiale che può avere valore normativo e quindi deve essere redatto seguendo criteri specifici.



Per agevolare la scrittura di tale documento è stato preparato un template  $\text{\LaTeX}$  che ne definisce la struttura e ne organizza i contenuti. Vi è quindi l'obbligo di seguire il sopraccitato schema per creare il verbale e di inviare il documento come allegato alla mailing list<sub>G</sub> in risposta all'email di convocazione dell'assemblea.

## 2.3 Repository e strumenti per la condivisione di file

### 2.3.1 Repository

Sono stati creati due repository<sub>G</sub> Git<sub>G</sub>, accessibili solamente tramite connessione sicura SSH<sub>G</sub> con scambio di chiave pubblica/privata con il server dedicato:

- **documents.git<sub>G</sub>**: disponibile all'indirizzo

```
ssh://sw-eng@ntp.myopenproject.it:documents.git
```

conterrà i sorgenti  $\text{\LaTeX}$  e gli script necessari alla stesura dei documenti;

- **src.git<sub>G</sub>**: disponibile all'indirizzo

```
ssh://sw-eng@ntp.myopenproject.it:src.git
```

conterrà i sorgenti dell'applicazione.

Il repository<sub>G</sub> **documents.git** è suddiviso secondo la seguente struttura:

```
documents
├── AnalisiDeiRequisiti
├── Glossario
├── LetteraDiPresentazione
├── NormeDiProgetto
├── PianoDiProgetto
├── PianoDiQualifica
├── SpecificaTecnica
├── StudioDiFattibilita
├── Verbali
└── template
```

Una volta terminata la fase di lavorazione di un documento, verrà creato un branch di verifica. In questo modo i *Verificatori* potranno lavorare parallelamente al resto del gruppo ed effettuare il merge<sub>G</sub> delle loro modifiche, una volta terminato il lavoro di verifica.

Il meccanismo di verifica e approvazione è descritto in dettaglio nella sezione 5.4.3.

Per ogni revisione verrà creato un tag così da identificare chiaramente la versione dei documenti presentati ad ogni revisione.

Eventuali suddivisioni del repository<sub>G</sub> **src.git** avverranno non prima della fase di **Progettazione di Dettaglio**.

### 2.3.2 Condivisione file

Per la condivisione informale di file e per il lavoro collaborativo su documenti di supporto, si usano le piattaforme di condivisione file online Dropbox<sub>G</sub> e Google Drive<sub>G</sub>. Trattandosi di strumenti informali, non si definiscono procedure rigorose d'uso e se ne lascia la descrizione alle sezioni 6.1.4.1 e 6.1.4.2.



## 3 Documentazione

Questo capitolo descrive tutte le convenzioni scelte ed adottate da Don't Panic riguardo alla stesura, verifica e approvazione della documentazione da produrre.

### 3.1 Template

Per agevolare la redazione della documentazione è stato creato un template  $\text{\LaTeX}$  contenente tutte le impostazioni stilistiche e grafiche citate in questo documento. Tale modello si può trovare nel repository<sub>G</sub> in `documents/template`.

### 3.2 Struttura del documento

#### 3.2.1 Prima pagina

Ogni documento è caratterizzato da una prima pagina che contiene le seguenti informazioni sul documento:

- Nome del gruppo;
- Nome del progetto;
- Logo del gruppo;
- Titolo del documento;
- Versione del documento;
- Cognome e nome dei redattori del documento;
- Cognome e nome dei verificatori del documento;
- Cognome e nome del responsabile approvatore del documento;
- Destinazione d'uso del documento;
- Lista di distribuzione del documento;
- Una breve descrizione del documento.

#### 3.2.2 Diario delle modifiche

La seconda pagina di ogni documento contiene il diario delle modifiche del documento. Ogni riga del diario delle modifiche contiene:

- Un breve sommario delle modifiche svolte;
- Cognome e nome dell'autore;
- Ruolo dell'autore;
- Data della modifica;
- Versione del documento dopo la modifica.

La tabella è ordinata per data in ordine decrescente, in modo che la prima riga corrisponda alla versione attuale del documento.



### 3.2.3 Indici

In ogni documento è presente un indice delle sezioni, un indice delle figure e un indice delle tabelle. Nel caso non siano presenti figure o tabelle i rispettivi indici verranno omessi.

### 3.2.4 Formattazione generale delle pagine

L'intestazione di ogni pagina contiene:

- Logo del gruppo;
- Nome del gruppo;
- Nome del progetto;
- Sezione corrente del documento.

A piè di pagina invece è presente:

- Nome e versione del documento;
- Pagina corrente nel formato  $N$  di  $T$  dove  $N$  è il numero di pagina corrente e  $T$  è il numero di pagine totali.

## 3.3 Norme tipografiche

Questa sezione racchiude le convenzioni riguardanti tipografia, ortografia e uno stile uniforme per tutti i documenti.

### 3.3.1 Punteggiatura

- **Parentesi:** il testo racchiuso tra parentesi non deve aprirsi o chiudersi con un carattere di spaziatura e non deve terminare con un carattere di punteggiatura;
- **Punteggiatura:** un carattere di punteggiatura non deve mai seguire un carattere di spaziatura;
- **Lettere maiuscole:** le lettere maiuscole vanno poste solo dopo il punto, il punto di domanda, il punto esclamativo e all'inizio di ogni elemento di un elenco puntato, oltre che dove previsto dalla lingua italiana. È inoltre utilizzata l'iniziale maiuscola nel nome del team, del progetto, dei documenti, dei ruoli di progetto, delle fasi di lavoro e nelle parole Proponente e Committente.

### 3.3.2 Stile di testo

- **Corsivo:** il corsivo deve essere utilizzato nei seguenti casi:
  - **Citazioni:** quando si deve citare una frase questa va scritta in corsivo;
  - **Abbreviazioni:** quando possibile si deve preferire una parola completa ad un'abbreviazione;
  - **Nomi particolari:** il corsivo deve essere utilizzato quando si parla di figure particolari (es. *Progettista*);



- **Documenti:** il corsivo deve essere utilizzato quando si parla di documenti (es. *Glossario*);
- **Altri casi:** in altre situazione, il corsivo va utilizzato per mettere in rilievo passaggi o parole significativi, evidenziare riferimenti ai documenti interni o esterni.
- **Grassetto:** il grassetto può essere utilizzato nei seguenti casi:
  - **Elenchi puntati:** in questi casi può essere utilizzato il grassetto per evidenziare il concetto sviluppato nella continuazione del punto;
  - **Altri casi:** per evidenziare particolari passaggi o parole chiave.
- **Monospace<sub>G</sub>:** il carattere monospace<sub>G</sub> serve per formattare il testo contenente parti di codice, nomi di classi, comandi, indirizzi web e percorsi;
- **Maiuscolo:** l'utilizzo di parole completamente in maiuscolo è riservato solo agli acronimi o alle macro  $\text{\LaTeX}$  riportate nei documenti;
- **$\text{\LaTeX}$ :** ogni riferimento a  $\text{\LaTeX}$  va scritto utilizzando il comando  $\text{\LaTeX}$ .

### 3.3.3 Composizione del testo

- **Elenchi puntati:** ogni punto dell'elenco deve terminare con un punto e virgola, tranne l'ultimo che deve terminare con un punto. La prima parola deve avere la lettera maiuscola, a meno di casi particolari (es. nome di un file);
- **Note a piè di pagina:** ogni nota dovrà cominciare con l'iniziale della prima parola maiuscola e non deve essere preceduta da alcun carattere di spaziatura. Ogni nota deve terminare con un punto.

### 3.3.4 Formati

- **Percorsi:** per gli indirizzi email e web completi deve essere utilizzato il comando  $\text{\LaTeX}$   $\text{\url}$ , mentre per gli indirizzi relativi va usato il formato monospace<sub>G</sub>;
- **Date:** le date presenti nei documenti devono seguire la notazione definita dallo standard ISO<sub>G</sub> 8601:2004:

$$AAAA - MM - GG$$

dove:

- *AAAA*: rappresenta l'anno scritto utilizzando quattro cifre;
- *MM*: rappresenta il mese scritto utilizzando due cifre;
- *GG*: rappresenta il giorno scritto utilizzando due cifre.
- **Ruoli di progetto:** è necessario riferirsi ai ruoli di progetto mediante il comando  $\text{\LaTeX}$   $\text{\role{Nome del ruolo}}$ , che garantisce la corretta scrittura, con la prima lettera di ogni parola che non sia una preposizione maiuscola (es. *Amministratore*);



- **Riferimenti ai documenti:** è necessario riferirsi ai documenti mediante il comando `\LaTeX \doc{Nome del documento}`, che garantisce la corretta scrittura, con la prima lettera di ogni parola che non sia una preposizione maiuscola (es. *Revisione dei Requisiti*).

Nel caso sia necessario riferirsi alla versione più recente del documento, è necessario l'uso dei comandi `\LaTeX` appositamente predisposti, quali ad esempio `\NormeDiProgetto`, che garantisce il corretto riferimento (es. *Norme di Progetto v5.2.0*);

- **Nomi dei file:** nel caso in cui ci si riferisca a un file senza utilizzare il suo percorso completo si deve utilizzare il formato `monospaceG`;
- **Codice e comandi:** per pezzi di codice e comandi presenti all'interno di frasi, le stesse vanno riferite nel documento con la macro ;
- **Revisioni:** è necessario riferirsi alle revisioni programmate con il comando `\LaTeX \rev{Nome Revisione}`, che garantisce la corretta scrittura, con la prima lettera di ogni parola che non sia una preposizione maiuscola (es. **Revisione dei Requisiti**);
- **Fasi del progetto:** è necessario riferirsi alle fasi del progetto con il comando `\LaTeX \fasi{Nome Fase}`, che garantisce la corretta scrittura, con la prima lettera di ogni parola che non sia una preposizione maiuscola (es. **Validazione**);
- **Nomi propri:** l'utilizzo dei nomi propri dei membri del team deve seguire la forma "Cognome Nome";
- **Nome del gruppo:** ci si riferirà al gruppo solo come "Don't Panic". Per la corretta scrittura è definita la macro `\LaTeX \GRUPPO`;
- **Nome del proponente:** ci si riferirà al proponente come "Mentis Srl" o con "Proponente". Per la corretta scrittura è definita la macro `\LaTeX \PROPONENTE`;
- **Nome del committente:** ci si riferirà al committente come "Prof. Vardanega Tullio" o con "Committente". Per la corretta scrittura è definita la macro `\LaTeX \COMMITTENTE`;
- **Nome del progetto:** ci si riferirà al progetto solo come "3DMob"; per la corretta scrittura è definita la macro `\LaTeX \PROGETTO`.

### 3.3.5 Sigle

Le sigle dei documenti e delle revisioni potranno essere utilizzate esclusivamente all'interno di tabelle o diagrammi. In particolare sono previste le seguenti sigle:

- **AdR** = Analisi dei Requisiti;
- **GL** = Glossario;
- **NdP** = Norme di Progetto;
- **PdP** = Piano di Progetto;
- **PdQ** = Piano di Qualifica;





- **SdF** = Studio di Fattibilità;
- **ST** = Specifica Tecnica;
- **RA** = Revisione di Accettazione;
- **RP** = Revisione di Progettazione;
- **RQ** = Revisione di Qualifica;
- **RR** = Revisione dei Requisiti.

### 3.4 Componenti grafiche

#### 3.4.1 Tabelle

Ogni tabella presente all'interno dei documenti dev'essere accompagnata da una didascalia, in cui deve comparire un numero identificativo incrementale per la tracciabilità della stessa all'interno del documento.

Per aumentare la leggibilità, in caso di tabelle contenenti molti numeri, le celle con molteplici valori uguali possono essere lasciate vuote. Tale scelta deve essere riportata all'inizio del documento che ne fa uso, specificando il valore che non verrà riportato.

#### 3.4.2 Immagini

Le immagini presenti all'interno dei documenti devono essere nel formato Scalable Vector Graphics ( $\text{SVG}_G$ ). In questo modo si garantisce una maggior qualità dell'immagine in caso di ridimensionamento. Per consentire l'inclusione delle immagini nei documenti, le immagini dovranno essere convertite nel formato  $\text{PDF}_G$ . Qualora non sia possibile salvare le immagini in formato vettoriale è preferito il formato Portable Network Graphics ( $\text{PNG}_G$ ).

### 3.5 Classificazione documenti

#### 3.5.1 Documenti formali

Un documento viene definito formale quando viene approvato dal *Responsabile di Progetto* ed è quindi pronto ad essere diffuso ai richiedenti. Questo momento coincide con una revisione con il committente. Per giungere a questo stato ogni documento deve aver seguito il percorso di verifica e validazione descritto nel *Piano di Qualifica v5.2.0* e nel paragrafo 3.7 riguardante il ciclo di vita dei documenti.

#### 3.5.2 Documenti informali

Un documento è ritenuto informale dal momento della sua creazione fino a quando non viene approvato dal *Responsabile di Progetto* ed è quindi da considerarsi esclusivamente ad uso interno.

### 3.6 Versionamento documenti

La documentazione prodotta deve essere corredata del numero di versione attuale tramite la seguente codifica:

$$vX.Y.Z$$



dove:

- **X**: indica il numero crescente di uscite formali del documento
  1. Fase di **Analisi** che si conclude con la **Revisione dei Requisiti**;
  2. Fase di **Analisi di Dettaglio** che si conclude con l'ingresso nella fase successiva;
  3. Fase di **Progettazione Architetture** che si conclude con la **Revisione di Progettazione**;
  4. Fase di **Progettazione di Dettaglio e Codifica** che si conclude con la **Revisione di Qualifica**;
  5. Fase di **Verifica e Validazione** che si conclude con la **Revisione di Accettazione**.

All'inizio di ogni fase il *Responsabile di Progetto* deve variare tale indice seguendo la numerazione progressiva indicata e impostare a 0 gli indici *Y* e *Z*; non sono ammessi indici diversi da quelli elencati;

- **Y**: indica il numero crescente di modifiche sostanziali al documento
  0. Stesura del documento;
  1. Verifica del documento;
  2. Approvazione del documento.

Nel momento in cui inizia l'attività di stesura il redattore del documento deve controllare che tale indice sia correttamente impostato a 0. All'inizio della verifica il *Verificatore* deve variare l'indice impostandolo a 1, dopo aver ricevuto il consenso dal *Responsabile*. Conclusa la verifica, il *Responsabile* provvede all'approvazione del documento e deve impostare l'indice a 2.

Il variare dell'indice sancisce un cambio di stato del documento. L'indice deve seguire la numerazione progressiva indicata e non sono ammessi indici diversi da quelli elencati;

- **Z**: indica il numero crescente di modifiche minori effettuate nelle varie fasi. Ad ogni modifica effettuata al documento che corrisponde ad un'aggiunta di una voce nel diario delle modifiche, il redattore o il *Verificatore* devono aggiornare l'indice seguendo una numerazione progressiva. Non viene fissato un limite superiore per tale indice.

Ogniqualevolta sia necessaria la citazione di una versione specifica di un documento, essa deve comprendere sia il nome che il numero di versione aderente al formato:

*Nome Documento vX.Y.Z*

Il formato da applicare ai nomi dei file alla loro creazione invece è:

NomeDocumento\_vX.Y.Z.pdf



### 3.7 Ciclo di vita

Ogni documento prodotto segue un preciso iter che scandisce le fasi in cui si trova in ogni istante. Un documento può trovarsi in tre stati diversi:

- **In lavorazione:** un documento entra in questa fase nel momento della sua creazione, e qui vi rimane per tutto il periodo necessario alla sua realizzazione, o per eventuali successive modifiche;
- **Da verificare:** una volta che il documento viene ultimato, esso deve essere preso in consegna dai verificatori che avranno il compito di rilevare e correggere eventuali errori e/o imprecisioni sintattici e semantici;
- **Approvato:** ogni documento, una volta ultimata la fase di verifica, deve essere approvato dal *Responsabile di Progetto*. L'approvazione sancisce lo stato finale del documento per la data versione.

Durante la sua vita, ogni documento può attraversare ogni fase più di una volta: nel momento in cui un documento approvato necessita di una revisione formale, esso inizia nuovamente il ciclo che, al suo termine, porterà ad una nuova versione dello stesso.

### 3.8 Glossario

Il Glossario conterrà tutte le parole presenti negli altri documenti che fanno parte del contesto dell'applicazione o che possono essere fraintese. Le definizioni, presentate in ordine alfabetico, dovranno essere concise e comprensibili.

I termini verranno inseriti nel glossario parallelamente al processo di stesura degli altri documenti, in modo da limitare l'errore umano. È preferibile inserire un termine inizialmente privo di definizione, piuttosto che rimandare l'inserimento dello stesso nel glossario.



## 4 Ruoli di progetto

Durante lo sviluppo del progetto vi saranno diversi ruoli che i membri del gruppo andranno a ricoprire. Tali ruoli rappresentano figure aziendali specializzate, indispensabili per il buon esito del progetto. Ciascun componente del gruppo dovrà ricoprire almeno una volta ogni ruolo<sup>1</sup>. Si deve inoltre certificare che non vi siano conflitti di interesse nello svolgimento delle attività di verifica e di approvazione.

Per garantire che la rotazione dei ruoli non provochi conflitti è necessario che le attività di stesura e verifica vengano pianificate dettagliatamente e che i soggetti interessati rispettino i compiti a loro assegnati. Sarà poi compito del *Verificatore* controllare attentamente il diario delle modifiche di ogni documento per individuare eventuali incongruenze.

Si descrivono ora i diversi ruoli di progetto, con le relative responsabilità e le modalità operative affinché essi possano svolgere i compiti assegnati con l'ausilio dei software scelti per il progetto.

### 4.1 Responsabile di Progetto

Il *Responsabile di Progetto* rappresenta il progetto, in quanto accentra su di sé le responsabilità di scelta ed approvazione, ed il gruppo, in quanto presenta al committente i risultati del lavoro svolto.

Detiene il potere decisionale, quindi la responsabilità su:

- Pianificazione, coordinamento e controllo delle attività;
- Gestione e controllo delle risorse;
- Analisi e gestione dei rischi;
- Approvazione dei documenti;
- Approvazione dell'offerta economica.

Di conseguenza, ha il compito di assicurarsi che le attività di verifica vengano svolte sistematicamente seguendo le *Norme di Progetto*, vengano rispettati i ruoli e le competenze assegnate nel *Piano di Progetto*, non vi siano conflitti di interesse tra redattori e verificatori. Egli è l'unico a poter decidere l'approvazione di un documento e a sancirne la distribuzione. Solo in casi particolari il *Responsabile* può delegare ad un verificatore l'approvazione di un documento come descritto nella sezione 5.1.7

Ha inoltre l'incarico di gestire la creazione e l'assegnazione dei ticket delle macro-fasi e di assegnare ad un membro del gruppo il ruolo di responsabile di quest'ultima.

Redige il *Piano di Progetto* e collabora alla stesura del *Piano di Qualifica*, in particolare nella sezione relativa alla pianificazione.

### 4.2 Amministratore

L'*Amministratore* è responsabile del controllo, dell'efficienza e dell'operatività dell'ambiente di lavoro. Le mansioni di primaria importanza che gli competono sono:

- Ricerca di strumenti che possano automatizzare qualsiasi compito che possa essere tolto all'umano;

---

<sup>1</sup>Tale regola deriva dai vincoli di organigramma



- Risoluzione dei problemi legati alle difficoltà di gestione e controllo dei processi e delle risorse. La risoluzione di tali problemi richiede l'adozione di strumenti adatti;
- Controllo delle versioni e delle configurazioni del prodotto;
- Gestione dell'archiviazione e del versionamento della documentazione di progetto;
- Fornire procedure e strumenti per il monitoraggio e la segnalazione per il controllo qualità.

Redige le *Norme di Progetto*, dove spiega e norma l'utilizzo degli strumenti, redige la sezione del *Piano di Qualifica* dove vengono descritti strumenti e metodi di verifica;

### 4.3 Analista

L'*Analista* è responsabile delle attività di analisi. Le responsabilità di spicco per tale ruolo sono:

- Produrre una specifica di progetto comprensibile, sia per il Proponente, sia per il Committente che per il *Progettista*, e motivata in ogni suo punto;
- Comprendere appieno la natura e la complessità del problema.

Redige lo *Studio di Fattibilità*, l'*Analisi dei Requisiti* e parte del *Piano di Qualifica*. Partecipa alla redazione del *Piano di Qualifica* in quanto conosce l'ambito del progetto ed ha chiari i livelli di qualità richiesta e le procedure da applicare per ottenerla.

### 4.4 Progettista

Il *Progettista* è responsabile delle attività di progettazione. Le responsabilità di tale ruolo sono:

- Produrre una soluzione attuabile, comprensibile e motivata;
- Effettuare scelte su aspetti progettuali che applichino al prodotto soluzioni note ed ottimizzate;
- Effettuare scelte su aspetti progettuali e tecnologici che rendano il prodotto facilmente manutenibile.

Redige la *Specifica Tecnica*, la *Definizione di Prodotto* e le sezioni inerenti le metriche di verifica della programmazione del *Piano di Qualifica*.

### 4.5 Verificatore

Il *Verificatore* è responsabile delle attività di verifica. Ha il compito di effettuare la verifica dei documenti utilizzando gli strumenti e i metodi proposti dal *Piano di Qualifica* e attenendosi a quanto descritto nelle *Norme di Progetto*. Le responsabilità di tale ruolo sono:

- Assicurare che l'attuazione delle attività sia conforme alle norme stabilite;
- Controllare la conformità di ogni stadio del ciclo di vita del prodotto.

Redige la sezione del *Piano di Qualifica* che illustra l'esito e la completezza delle verifiche e delle prove effettuate.



## 4.6 Programmatore

Il *Programmatore* è responsabile delle attività di codifica e delle componenti di ausilio necessarie per l'esecuzione delle prove di verifica e validazione. Le responsabilità di tale ruolo sono:

- Implementare rigorosamente le soluzioni descritte dal *Progettista*, da cui seguirà quindi la realizzazione del prodotto;
- Scrivere codice documentato, versionato, manutenibile e che rispetti gli standard stabiliti per la scrittura del codice;
- Implementare i test sul codice scritto, necessari per prove di verifica e validazione.

Redige il *Manuale Utente* e produce una abbondante documentazione del codice.



## 5 Procedure a supporto dei processi

Dopo aver descritto i ruoli di progetto e le relative funzioni, si procede ora ad elencare le procedure che essi devono seguire in modo rigoroso per convergere agli obiettivi posti nel *Piano di Qualifica*.

### 5.1 Gestione di progetto

Le responsabilità di gestione dell'intero progetto, dalla nascita alla conclusione, sono da attribuire al *Responsabile di Progetto*.

Quest'ultimo dovrà garantire un corretto sviluppo delle attività utilizzando, qualora sia possibile, degli strumenti che gli consentano di:

- Pianificare, coordinare e controllare le attività;
- Gestire e controllare le risorse;
- Analizzare e gestire i rischi;
- Elaborare i dati.

#### 5.1.1 Pianificazione delle attività

Per pianificare le attività il *Responsabile di Progetto* deve realizzare un diagramma di Gantt<sub>G</sub> per ciascuna fase indicata nella sezione Pianificazione del *Piano di Progetto v5.2.0*, utilizzando ProjectLibre come descritto nella sezione 6.2.

#### 5.1.2 Coordinazione e controllo delle attività

Per coordinare e controllare le attività il *Responsabile di Progetto* deve riportare la struttura creata con ProjectLibre in Redmine sfruttando il suo sistema di ticketing come descritto nella sezione 6.5.

In questo modo ciascun componente del gruppo sarà avvisato delle attività ad esso assegnate e potrà inserire lo stato delle stesse permettendo al *Responsabile* di verificare immediatamente l'avanzamento del progetto.

Infine può controllare le attività grazie al grafico PDCA prodotto da Tracy e descritto nella sezione 6.4.3.

#### 5.1.3 Gestione e controllo delle risorse

Per gestire e controllare le risorse il *Responsabile di Progetto* deve utilizzare Redmine come indicato nella sezione 6.5 che gli consente anche di verificare l'avanzamento di ogni processo come riportato nel *Piano di Progetto v5.2.0*.

#### 5.1.4 Analisi e Gestione dei rischi

Durante l'avanzamento del progetto il *Responsabile di Progetto* deve monitorare costantemente il verificarsi dei rischi descritti nel *Piano di Progetto v5.2.0* ed eventuali nuovi rischi, attuando le contromisure descritte e riportando gli effettivi riscontri.



### 5.1.5 Gestione bug

Per coordinare i *Programmatori* il *Responsabile di Progetto* deve controllare che non avvenga un aumento incontrollato del numero di bug. All'aumentare del numero di bug devono essere assegnate un maggior numero di risorse alla correzione ed alla verifica del codice.

Durante la validazione del prodotto il *Responsabile di Progetto* dovrà esportare i dati dal sistema di ticketing relativi ai bug utilizzando l'esportazione mediante foglio di calcolo nel formato OpenDocument<sub>G</sub> di phpMyAdmin. I dati esportati devono essere inseriti nel modello del foglio di calcolo per la generazione del grafico PDCA, presente nella cartella:

Dropbox/swe/qualità/template\_grafico\_Bug.odt

Grazie a tale foglio di calcolo verrà presentato il grafico temporaneo che descrive l'andamento della gestione dei bug del sistema. Il grafico non deve essere salvato, ma utilizzato unicamente con lo scopo di trarre le opportune considerazioni. Come leggere tale grafico è descritto nel *Piano di Progetto v5.2.0*.

### 5.1.6 Elaborazione dati

Il *Responsabile di Progetto* deve sfruttare il software Calc, come descritto nella sezione 6.3.5, per elaborare i dati raccolti durante lo sviluppo del progetto e riportarli nel *Piano di Progetto*.

### 5.1.7 Delega

Il *Responsabile di Progetto*, nel caso in cui abbia redatto una parte di un documento, può delegare l'approvazione di tale documento ad un *Verificatore*.

### 5.1.8 Responsabilità di sotto-progetto

Ogni macroattività può essere assegnata dal *Responsabile* ad un responsabile di sotto-progetto, i cui compiti saranno l'assegnazione delle singole attività alle risorse rese disponibili e la gestione dei cambiamenti.

#### 5.1.8.1 Assegnazione attività

Per assegnare attività alle risorse disponibili, il responsabile di sotto-progetto dovrà seguire le procedure di ticketing descritte in 6.5.2.2.

#### 5.1.8.2 Gestione dei cambiamenti

In caso di errori, in seguito alla notifica da parte del *Verificatore* tramite ticket, il responsabile di sotto-progetto dovrà assegnare la correzione mediante la procedura di ticketing descritta in 6.5.2.2. Al termine della correzione, sarà compito del responsabile di sotto-progetto accettare o respingere la modifica, e richiederne di conseguenza il rifacimento.

Nel caso la correzione riguardi un'attività di codifica, sarà compito del responsabile di sotto-progetto programmare una nuova esecuzione dei test di unità e di integrazione correlati al modulo modificato.





## 5.2 Analisi dei Requisiti

### 5.2.1 Studio di Fattibilità e Analisi dei Rischi

Alla pubblicazione dei capitolati, è compito del *Responsabile di Progetto* convocare quante riunioni saranno ritenute necessarie per consentire il confronto tra i membri del gruppo su ogni capitolato. Queste riunioni forniranno agli *Analisti* una base riguardante le conoscenze e preferenze di ogni membro del gruppo. È compito degli *Analisti*, sulla base di quanto deciso, redigere uno *Studio di Fattibilità* dei capitolati basandosi su:

- **Dominio tecnologico e applicativo:** conoscenza delle tecnologie richieste, esperienze precedenti con le problematiche poste dal capitolato, conoscenza del dominio applicativo;
- **Rapporto Costi/Benefici:** competitori e prodotti simili già presenti sul mercato, quantità di requisiti obbligatori, costo della realizzazione rapportato al risultato previsto;
- **Individuazione dei rischi:** Comprensione dei punti critici della realizzazione, individuazione di eventuali lacune tecniche o di conoscenza del dominio applicativo dei membri del gruppo, analisi delle difficoltà nell'individuazione dei requisiti e loro verificabilità.

Un'ultima riunione, a *Studio di Fattibilità* concluso, presenterà la decisione finale sul capitolato scelto.

### 5.2.2 Analisi dei requisiti

La stesura del documento denominato *Analisi dei Requisiti* è compito degli *Analisti*, e si divide nelle fasi illustrate in questa sezione.

#### 5.2.2.1 Classificazione dei requisiti

È compito degli *Analisti* stilare una lista dei requisiti emersi dal capitolato e da eventuali riunioni con il proponente. Questi requisiti dovranno essere classificati per tipo e importanza, utilizzando la seguente codifica:

R[importanza][tipo][codice]

- **Importanza** può assumere i seguenti valori:

0. Requisito obbligatorio;
1. Requisito desiderabile;
2. Requisito opzionale.

L'uso di numeri permette di ordinare facilmente i requisiti per importanza;

- **Tipo** può assumere i seguenti valori:

- F: Funzionale;
- Q: Di qualità;
- P: Prestazionale;
- V: Vincolo.



- **Codice** è il codice univoco di ogni requisito espresso in modo gerarchico.

Ogni requisito è poi esplicito nel seguente modo:

- **Relazioni** di dipendenza con altri requisiti;
- **Descrizione** sintetica ma chiara del requisito.

Tutti i requisiti devono essere inseriti in Tracy, software di tracciamento appositamente creato, secondo le modalità descritte in 6.4. Il software provvederà alla generazione del codice identificativo univoco.

#### 5.2.2.2 Modellazione concettuale del sistema e Allocazione

Successivamente al tracciamento dei requisiti emersi dal capitolato e da eventuali riunioni, si procede all'analisi dei casi d'uso, denominati nelle sezioni seguenti anche come *use case* o con l'acronimo UC. È richiesta agli *Analisti* l'identificazione dei casi d'uso, procedendo dal generale al particolare, e l'inserimento degli stessi nel software di tracciamento Tracy.

Di ogni caso d'uso è richiesto l'inserimento nel software di:

- **Titolo** sintetico del caso d'uso;
- **Attori** principali e secondari;
- **Descrizione** chiara e dettagliata del caso d'uso;
- **Precondizione** del caso d'uso;
- **Flusso principale degli eventi** specificando per ogni evento titolo, descrizione, attori coinvolti e se l'evento è descritto nel dettaglio da un altro caso d'uso;
- **Scenari alternativi** specificando per ogni scenario alternativo titolo, descrizione, attori coinvolti e se lo scenario è descritto nel dettaglio da un altro caso d'uso;
- **Postcondizione** del caso d'uso;
- **Requisiti** dedotti dal caso d'uso.

Il caso d'uso dovrà essere accompagnato da un grafico riassuntivo in UML<sub>G</sub> 2.4, titolato come il caso d'uso. È compito del software di tracciamento Tracy titolare il caso d'uso con 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. Il software provvederà a fornire le tabelle di tracciamento e il codice  $\text{\LaTeX}$  dei casi d'uso, formattati secondo convenzione.

#### 5.2.2.3 Tracciamento

Sebbene il software di tracciamento Tracy sia provvisto di strumenti automatici atti allo scopo, spetta agli *Analisti* controllare la corrispondenza biunivoca di ogni requisito con una o più fonti. Le fonti di requisiti sono: casi d'uso, verbale e capitolato.

In caso di anomalie è necessario correggerle e segnalarle all'*Amministratore*.



## 5.3 Progettazione

### 5.3.1 Specifica Tecnica

I *Progettisti* devono descrivere la progettazione ad alto livello dell'architettura dell'applicazione e dei singoli componenti nella *Specifica Tecnica* e provvedere alla progettazione di opportuni test di integrazione.

#### 5.3.1.1 Diagrammi UML

Devono essere realizzati i seguenti diagrammi:

- Diagrammi dei package<sub>G</sub>;
- Diagrammi delle classi;
- Diagrammi di sequenza;
- Diagrammi di attività.

seguendo lo standard UML<sub>G</sub> 2.0.

#### 5.3.1.2 Design pattern

I *Progettisti* devono descrivere i design pattern<sub>G</sub> utilizzati per realizzare l'architettura: di essi si deve includere una breve descrizione e un diagramma che ne esemplifichi il funzionamento e la struttura.

#### 5.3.1.3 Tracciamento componenti

Ogni requisito deve essere tracciato al componente che lo soddisfa. Il software Tracy genera automaticamente le tabelle di tracciamento come descritto nella sezione 6.4.5. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni requisito venga soddisfatto.

#### 5.3.1.4 Test di integrazione

I *Progettisti* devono definire delle classi di verifica necessarie per verificare che i componenti del sistema funzionino nella maniera prevista.

### 5.3.2 Definizione di Prodotto

I *Progettisti* devono produrre la *Definizione di Prodotto* dove viene descritta la progettazione di dettaglio del sistema ampliando quanto scritto nella *Specifica Tecnica*.

Lo scopo di questo documento è quello di definire dettagliatamente ogni singola unità di cui è composto il sistema in modo da semplificare l'attività di codifica e allo stesso tempo di non fornire alcun grado di libertà al *Programmatore*.

Parallelamente alla progettazione di dettaglio dei componenti software dovranno essere progettati i relativi test di unità che verranno descritti nel *Piano di Qualifica*.



### 5.3.2.1 Diagrammi UML

Devono essere aggiornati e affinati i seguenti diagrammi:

- Diagrammi delle classi;
- Diagrammi di sequenza;
- Diagrammi di attività.

### 5.3.2.2 Definizione di classe

Ogni classe progettata deve essere descritta all'interno della *Definizione di Prodotto*; tale descrizione deve comprendere l'elenco di metodi e attributi, una spiegazione sullo scopo della classe e deve specificare quale funzionalità essa modella.

La descrizione di ogni classe viene effettuata tramite il software Tracy, dal quale poi verrà generato automaticamente il codice  $\text{\LaTeX}$  da includere nel documento. Il formalismo tipografico scelto prevede le seguenti convenzioni:

- Gli **attributi** della classe sono indicati con l'indicazione dell'accessibilità dell'attributo secondo quanto previsto da  $\text{UML}_G$  2.4, seguita dal nome di colore verde e dalla descrizione dello stesso. Il colore è presente per aiutare la leggibilità e non ha significati particolari;
- I **metodi** della classe sono indicati di colore blu, con l'indicazione dell'accessibilità del metodo secondo quanto previsto da  $\text{UML}_G$  2.4, seguita dal nome del metodo e dai suoi parametri tra parentesi tonde, ad esempio: `# myMethod(argumentList)`. Viene infine descritto il metodo e si riporta la lista degli argomenti. Il colore è presente per aiutare la leggibilità e non ha significati particolari;
- I **parametri** del metodo racchiusi da parentesi tonde devono essere riportati con il nome, seguito dai due punti e dal tipo del parametro, ad esempio: `+ myMethod(myParam : Type1, myOtherParam : Type2*)`;
- Gli **argomenti** riportati per esteso devono essere riportati con la direzione tra parentesi quadre, il nome seguito dai due punti e dal tipo dell'argomento, ad esempio: `[in] myParam : myType*`. Segue infine la descrizione dell'argomento.

### 5.3.2.3 Formalismo di specifica dei metodi

Ogni classe progettata deve essere descritta all'interno della *Definizione di Prodotto*; tale descrizione deve comprendere l'elenco di metodi e attributi, una spiegazione sullo scopo della classe e deve specificare quale funzionalità essa modella.

La descrizione di ogni classe viene effettuata tramite il software Tracy, dal quale poi verrà generato automaticamente il codice  $\text{\LaTeX}$  da includere nel documento.

### 5.3.2.4 Tracciamento classi

Ogni requisito deve essere tracciato alle classi che lo soddisfano. Il software Tracy genera automaticamente le tabelle di tracciamento come descritto nella sezione 6.4.5. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni classe soddisfi almeno un requisito.



### 5.3.2.5 Test di unità

I *Progettisti* dovranno definire i test d'unità necessari per verificare che i componenti del sistema funzionino nella modo previsto.

## 5.4 Verifica

La verifica di processi, documenti e prodotti è un'attività da eseguire continuamente durante lo sviluppo del Progetto. Di conseguenza, servono modalità operative chiare e dettagliate per i *Verificatori*, in modo da uniformare le attività di verifica svolte ed ottenere il miglior risultato possibile. Si descrivono ora le modalità ordinate e puntuali di verifica di processi, documenti, attività e codice alle quali ci si riferirà in questo documento e alle quali i *Verificatori* dovranno attenersi.

### 5.4.1 Metriche per gli errori riscontrati e gestione dei cambiamenti

Si definiscono ora delle metriche per gli errori che i *Verificatori* potranno trovare, fornendo criteri per la quantificazione dell'impatto sul prodotto o sul processo e per la definizione delle priorità di intervento. In questo modo si potrà agire prima nella risoluzione di errori a gravità maggiore.

Errore trovato	Gravità	Priorità risoluzione	Modalità operative
Indici fuori range	Alta	Urgente	Ticket
Ritardi superiori a 2 giorni nelle attività	Alta	Urgente	Ticket
Errato tracciamento di requisiti e casi d'uso	Alta	Urgente	Ticket
Errore di progettazione	Alta	Urgente	Ticket

Tabella 2: Errori nei processi: gravità e procedure di gestione

Errore trovato	Gravità	Priorità risoluzione	Modalità operative
Errore ortografico o di formattazione	Bassa	Entro la Milestone <sub>G</sub>	Ticket o correzione immediata
Errore sistematico di ortografia o formattazione	Media	Breve	Ticket e aggiunta alla checklist
Compilazione fallita del documento	Alta	Urgente	Ticket o correzione immediata
Valori Gulpease fuori range	Media	Breve	Ticket
Errore di concetto nel testo	Alta	Urgente	Ticket
Errore di formalismo UML <sub>G</sub> 2.x	Bassa	Entro la Milestone <sub>G</sub>	Ticket
Mancata compilazione del codice	Alta	Urgente	Ticket o correzione immediata



Mancato rispetto delle norme di codifica	Medio	Breve	Ticket
--	-------	-------	--------

Tabella 3: Errori nei documenti e nel codice: gravità e procedure di gestione

La gravità dell'errore può essere:

- **Bassa** se l'errore ha impatto su aspetti marginali del prodotto o provoca un basso aumento dei costi o dei tempi del processo;
- **Media** se l'errore ha impatto significativo sul prodotto o provoca un aumento percepibile di tempi e costi;
- **Alta** se l'errore rende il prodotto inutilizzabile o provoca un forte aumento dei tempi o dei costi.

Ambito	Gravità bassa	Gravità media	Gravità alta
Errore nel prodotto	Impatto su aspetti marginali	Impatto su aspetti visibili	Prodotto inutilizzabile
Errore nei processi	Aumento costi o tempi < 10%	Aumento costi o tempi < 25%	Aumento costi o tempi > 25%

Tabella 4: Gravità dell'errore e impatto su processi e prodotti

La priorità di risoluzione può essere:

- **Entro la milestone<sub>G</sub>**: indica che l'errore deve essere risolto prima della milestone<sub>G</sub> successiva;
- **Breve**: indica che l'errore deve essere risolto entro 4-5 giorni;
- **Urgente**: indica che l'errore deve essere risolto appena possibile.

Le modalità operative per il verificatore sono le seguenti:

- **Ticket**: il *Verificatore* deve aprire un ticket assegnato al responsabile dell'attività secondo le modalità descritte in 6.5.2.3. Sarà il responsabile a designare modalità e persone addette alla correzione;
- **Correzione immediata**: è richiesto che il *Verificatore* proceda autonomamente alla correzione dell'errore;
- **Aggiunta alla checklist**: è richiesto che il *Verificatore* aggiunga l'errore riscontrato alla checklist appropriata.

#### 5.4.2 Verifica dei processi

Ai Verificatori è richiesto di effettuare quanto segue:

- **Controllo delle metriche**: Alla conclusione di ogni fase del progetto, per ogni macro-attività, definita nel *Piano di Progetto v5.2.0*, si calcolano gli indici definiti nella sezione *Metriche per i processi* del *Piano di Qualifica v5.2.0*. Al fine di avere



un indice complessivo di fase dovrà essere inoltre calcolato il valore medio di tali indici.

Tali indici verranno calcolati utilizzando in modo complementare il software per la pianificazione come descritto nella sezione 6.2;

- **Grafico PDCA:** Alla conclusione di ogni fase del progetto il *Verificatore* dovrà esportare i dati dal sistema di ticketing utilizzando l'esportazione mediante foglio di calcolo nel formato OpenDocument<sub>G</sub> di phpMyAdmin. I dati esportati devono essere inseriti nel modello del foglio di calcolo per la generazione del grafico PDCA, presente nella cartella:

Dropbox/swe/pianificazione/template\_grafico\_PDCA.odt

Dopo aver ottenuto il grafico il *Verificatore* con la supervisione del *Responsabile di Progetto* dovrà trarre delle conclusioni generali sulle velocità con cui sono stati portati avanti i processi.

### 5.4.3 Verifica dei documenti

Il processo di verifica viene istanziato nel momento in cui l'output di un processo passa dalla versione *X.0.Z* alla versione *X.1.0*. È compito del *Responsabile* notificare i *Verificatori* dell'inizio dell'attività di verifica. Una definizione esplicita del cambio di versione del documento è disponibile nella sezione 6.3.3. Attraverso il diario delle modifiche è possibile focalizzare l'attenzione maggiormente sulle sezioni che hanno subito dei cambiamenti dall'ultima verifica, riducendo il tempo necessario al controllo.

Per eseguire un'accurata verifica dei documenti redatti è necessario seguire il seguente protocollo:

1. **Controllo sintattico e del periodo:** Utilizzando TeXstudio e GNU Aspell vengono evidenziati e corretti gli errori di grammatica più evidenti. Gli errori di sintassi, di sostituzione di lettere che provocano la creazione di parole grammaticalmente corrette ma sbagliate nel contesto ed i periodi di difficile comprensione necessitano dell'intervento di un verificatore umano. Per questa ragione ciascun documento dovrà essere sottoposto ad un walkthrough da parte dei verificatori per individuare tali errori;
2. **Rispetto delle norme di progetto:** Sono state definite norme tipografiche di carattere generale. Impongono una struttura dei documenti che non può essere verificata in maniera automatica. Per la verifica di alcune norme si possono utilizzare degli strumenti automatici, descritti nella sezione 6.3.3. La verifica delle norme per cui non è stato definito uno strumento automatico richiede che i *Verificatori* eseguano inspection sul rispetto di quelle norme in ciascun documento;
3. **Lista di controllo:** Il *Verificatore* dovrà utilizzare la lista di controllo per i documenti, descritta nell'appendice A, e verificare che gli errori tipici non siano presenti;
4. **Verifica delle proprietà di glossario:** Nella redazione del *Glossario* è possibile automatizzare tale verifica. Utilizzando l'apposito script, vengono individuate tutte le segnalazioni di una parola del glossario in un documento in maniera automatica, ed applicato l'opportuno stile. Tale script è descritto nella sezione 6.3.2. Il *Verificatore* dovrà quindi controllare che l'output finale sia conforme a quanto precisato nelle *Norme di Progetto v5.2.0* relativamente al *Glossario*;



5. **Calcolo dell'indice Gulpease:** Su ogni documento redatto il *Verificatore* deve calcolare l'indice di leggibilità utilizzando lo strumento automatico fornito nel Makefile.  
Nel caso in cui l'indice risultasse troppo basso, sarà necessario eseguire un walk-through del documento alla ricerca delle frasi troppo lunghe o complesse;
6. **Miglioramento del processo di verifica:** Per avere un miglioramento del processo di verifica, quando i *Verificatori* eseguono walkthrough di un documento, dovranno riportare gli errori più frequentemente trovati. Grazie a tale pratica sarà possibile eseguire inspection su tali errori nelle verifiche future;
7. **Segnalazione degli errori riscontrati:** Attenendosi a quanto riportato nelle tabelle 3 e 2 il *Verificatore* deve generare ticket secondo quanto descritto nella sezione 6.5.3.3.

#### 5.4.4 Verifica del tracciamento dei requisiti

Al *Verificatore* è richiesto di controllare tutti i punti della checklist relativa al tracciamento dei requisiti riportata nell'appendice A, oltre al controllo dei diagrammi di caso d'uso prodotti.

#### 5.4.5 Verifica dei diagrammi UML

Al *Verificatore* è richiesto il controllo dei diagrammi  $UML_G$  prodotti:

- **Diagrammi di caso d'uso:** Il controllo dei diagrammi di caso d'uso deve avvenire manualmente, controllando il rispetto delle specifiche  $UML_G$  2.4 e il corretto uso delle relazioni di inclusione ed estensione. Il diagramma di caso d'uso deve rappresentare fedelmente quanto descritto dal caso d'uso;
- **Diagrammi delle classi:** Il controllo del rispetto del formalismo  $UML_G$  2.0 è automatico, ed avviene tramite WhiteStarUML come descritto nella sezione 6.3.4. Al *Verificatore* è chiesto di controllare la corrispondenza tra progettazione e diagrammi delle classi.  
Inoltre premendo il pulsante F9, il quale avvia lo strumento di verifica  $UML_G$ , il *Verificatore* deve controllare che i diagrammi siano ben formati secondo lo standard.

#### 5.4.6 Verifica della progettazione architetturale

Al *Verificatore* è richiesto il controllo del rispetto delle metriche di accoppiamento afferente e efferente delle classi. Entrambi i valori sono calcolati automaticamente per ogni classe e come media da *Tracy* come illustrato nella sezione 6.4.

#### 5.4.7 Verifica della progettazione di dettaglio

Al *Verificatore* è richiesto di controllare che il documento *Definizione di Prodotto* contenga l'architettura di dettaglio con i vari moduli del sistema aventi una dimensione e un accoppiamento che li renda realizzabili per un singolo *Programmatore*.

Ogni modulo deve essere associato ad almeno un requisito e il *Verificatore* dovrà controllare la bontà di tale tracciamento generato da *Tracy* come specificato nella sezione 6.4.2.





#### 5.4.8 Verifica del codice

Per la verifica del codice al *Verificatore* è richiesto l'avvio dei test statici e dinamici e l'analisi dei risultati. Di seguito un elenco degli strumenti da usare per l'analisi.

##### 5.4.8.1 Analisi Statica

Tutti gli strumenti di seguito riportati sono integrati in Jenkins ed è stata impostata la compilazione automatica e l'esecuzione dei test ad ogni *push*, per poter rendere la verifica il più automatica possibile. In questo modo, al *Verificatore* è richiesta solamente un'accurata analisi dei risultati della verifica automatica che viene avviata ad ogni *push* sul repository<sub>G</sub> del codice.

I tool integrati in Jenkins per l'**analisi statica** sono:

- **CppCheck**<sup>2</sup>: analizzatore statico per individuare errori comuni del codice C++;
- **CCCC**<sup>3</sup> (C and C++ Code Counter): Misura metriche riguardanti codice sorgente in C++. Le più rilevanti sono:
  - **Complessità ciclomatica (McCabe's complexity)**: utilizzata per misurare la complessità di funzioni, moduli, metodi o classi di un programma;
  - **Information flow measure (IF4)**: misura il flusso di informazioni come suggerito da Henry e Kafura. Il valore è calcolato come  $(lunghezzafunzione)^2 \times fan-in \times fan-out$ ;
  - **Linee di codice per linee di commento (L\_C)**: rapporto tra linee di codice e linee di commento;
  - **Metodi per classe (WMC)**: numero pesato di metodi e funzioni per classe. Vi sono due indici per tale parametro: WMC1 misura il numero nominale di funzioni e metodi, WMCv conta il numero di funzioni e metodi accessibili da altri metodi e non conta quelli privati;
  - **Profondità albero gerarchie(DIT)**: la lunghezza del percorso più lungo che termina con il modulo in esame;
  - **Numero di figli (NOC)**: numero di moduli che ereditano direttamente dal modulo in esame;
  - **Accoppiamento tra gli oggetti(CBO)**: numero di moduli che sono accoppiati al modulo in esame come clienti o come fornitori di servizi.
- **Clang**<sup>4</sup>: compilatore che segnala errori e warning in modo molto espressivo;
- **GCC**<sup>5</sup>: compilatore che segnala errori e warning in modo dettagliato e chiaro.

##### 5.4.8.2 Analisi Dinamica

I tool integrati in Jenkins per l'**analisi dinamica** sono:

- **Valgrind**<sup>6</sup>: strumento open source per il debug di problemi di memoria, la ricerca dei memory leak<sub>G</sub> ed il profiling<sub>G</sub> del software;

<sup>2</sup><http://cppcheck.sourceforge.net/>

<sup>3</sup><http://cccc.sourceforge.net/>

<sup>4</sup><http://clang.llvm.org/diagnostics.html>

<sup>5</sup><http://gcc.gnu.org/>

<sup>6</sup><http://valgrind.org/>



- **Unit testing:** Per assicurare il corretto funzionamento del programma durante lo sviluppo dell'applicazione, verranno scritti dei test usando **QtTest**<sup>7</sup>. QtTest è un framework<sub>G</sub> per test di unità ed è un modulo di Qt<sub>G</sub> SDK. Per scrivere un test sarà necessario creare una classe specifica per il testing e utilizzare le macro **QVERIFY** e **QCOMPARE** definite da QtTest;
- **Squish Coco:** strumento in grado di calcolare la copertura totale del codice da parte dei test di unità e integrazione e di mostrare le parti di codice che non vengono mai eseguite dai test. È ben integrato con Qt<sub>G</sub> ed è in grado di ignorare i file prodotti dal Meta Object Compiler, permettendo una misurazione più efficace.

#### 5.4.9 Resoconto bug

Alla conclusione del progetto il *Verificatore* dovrà esportare i dati dal sistema di ticketing reattivi ai bug utilizzando l'esportazione mediante foglio di calcolo nel formato OpenDocument<sub>G</sub> di phpMyAdmin. I dati esportati devono essere inseriti nel modello del foglio di calcolo per la generazione del grafico PDCA, presente nella cartella:

Dropbox/swe/qualità/template\_grafico\_Bug.odt

Dopo aver ottenuto ed esportato il grafico il *Verificatore* con la supervisione del *Responsabile di Progetto* dovrà trarre delle conclusioni generali sul numero di bug e sulla velocità con cui sono stati gestiti.

#### 5.4.10 Validazione output

Al *Verificatore* è richiesto il controllo che i file in forma testuale, XML<sub>G</sub> e JSON<sub>G</sub>, in output siano validi rispetto agli scemi descritti nella *Definizione di Prodotto v5.2.0*. Le procedure da seguire sono descritte nella sezione 6.6.3.

### 5.5 Codifica

#### 5.5.1 Codifica e convenzioni

Tutti i file contenenti codice o documentazione dovranno essere conformi alla codifica UTF-8<sub>G</sub> senza BOM<sub>G</sub>. Verrà usato LF<sub>G</sub> (U+000A) per andare a capo.

Gli sviluppatori dovranno attenersi alle Qt Coding Conventions<sub>G</sub> e seguire le convenzioni stilistiche definite dal Qt<sub>G</sub> Coding Style.

È ammessa la possibilità di effettuare modifiche alle convenzioni stabilite in seguito ad una decisione del *Responsabile di Progetto*.

##### 5.5.1.1 Nomi

I nomi di variabili, classi, funzioni, metodi e commenti dovranno essere in camel case<sub>G</sub> e in inglese. I nomi di variabili, metodi e funzioni dovranno avere la prima lettera minuscola, i nomi delle classi dovranno avere la prima lettera maiuscola.

<sup>7</sup><http://qt-project.org/doc/qt-4.8/qttest.html>



#### 5.5.1.2 Ricorsione

La ricorsione va evitata quando possibile. Per ogni funzione ricorsiva sarà necessario fornire una prova di terminazione e sarà necessario valutare il costo in termini di occupazione della memoria. Nel caso l'utilizzo di memoria risulti troppo elevato la ricorsione verrà rimossa.

#### 5.5.2 Documentazione

I file contenenti codice dovranno essere provvisti di un'intestazione contenente:

```
/*!  
 * \file Nome del file  
 * \author Autore (indirizzo email dell'autore)  
 * \date Data di creazione  
 * \brief Breve descrizione del file  
 *  
 * Descrizione dettagliata del file  
 */
```

Prima di ogni classe dovrà essere presente un commento contenente:

```
/*!  
 * \class Nome della classe  
 * \brief Breve descrizione della classe  
 */
```

Prima di ogni metodo dovrà essere presente un commento contenente:

```
/*!  
 * \brief Breve descrizione della funzione  
 * \param Nome del primo parametro  
 * \param Nome del secondo parametro  
 * \return Valore ritornato dalla funzione  
 */
```

La documentazione verrà generata insieme ai file delle classi dal sistema di generazione automatica del codice di Tracy.

Nel caso si verifichi la necessità di documentare del codice di difficile comprensione sarà possibile inserire un commento nelle righe precedenti, dopo aver verificato l'impossibilità di effettuare il refactoring<sub>G</sub> del codice esaminato.



## 6 Ambiente di lavoro

### 6.1 Coordinamento

È stato predisposto un server dedicato sul quale sono installate alcune applicazioni web che facilitano la gestione del progetto. Per connettersi al server seguire quanto descritto nella sezione 2.3.1.

#### 6.1.1 Software di gestione del progetto

Come piattaforma di gestione del progetto è stato scelto **Redmine**. Redmine fornisce:

- Un sistema flessibile di gestione dei ticket;
- Il grafico Gantt<sub>G</sub> delle attività;
- Un calendario per organizzare i compiti;
- La visualizzazione del repository<sub>G</sub> associato al progetto;
- Un sistema di rendicontazione del tempo.

Sono state valutate diverse alternative ma, dopo un'attenta fase di test, nessuna di queste è stata ritenuta all'altezza di Redmine per quantità e qualità delle caratteristiche proposte.

Sono stati provati i seguenti software:

- **Gitlab/Gitorious:** entrambe le piattaforme propongono una gestione del progetto minimale e inadeguata alle esigenze del gruppo;
- **ProjectPier:** la gestione dei ticket non è stata ritenuta all'altezza del sistema di ticketing di Redmine;
- **TRAC:** raggiunge la completezza di Redmine in quanto a caratteristiche tramite l'installazione di plugin non ufficiali ma non fornisce un'interfaccia altrettanto immediata, aumentando il tempo speso dal team nella gestione del progetto.

Le modalità di corretto utilizzo dello strumento sono descritte in dettaglio nella sezione 6.5.4.

#### 6.1.2 Software di versionamento

Sono state presi in considerazione diversi software di versionamento (Git<sub>G</sub>, SVN, Mercurial) prima di decidere di usare **Git<sub>G</sub>**. I motivi principali della scelta sono:

- **Flessibilità:** Git<sub>G</sub> è un repository<sub>G</sub> distribuito con la possibilità di *commit<sub>G</sub>* e *revert* locali;
- **Esperienza del team:** Git<sub>G</sub> è già stato usato da alcuni componenti del gruppo Don't Panic.

Si descrive ora la procedura di corretto utilizzo del repository<sub>G</sub>.

- **Clonare il repository<sub>G</sub>:** è possibile clonare il repository<sub>G</sub> remoto in locale attraverso il comando `git clone sw-eng@dtp.myopenproject.it:nome-repository.git`;



- **Sincronizzare il repository<sub>G</sub>:** per scaricare in locale le modifiche bisogna eseguire il comando `git pull` nella cartella che contiene il repository<sub>G</sub> locale;
- **Salvare una modifica in locale:** con il comando `git add NomeFile` si aggiunge un file al commit<sub>G</sub> corrente e usando il comando `git commit -m 'Messaggio di commit'` si salvano le modifiche effettuate ai file;
- **Inviare le modifiche al repository<sub>G</sub> remoto:** tramite il comando `git push` si inviano le modifiche al repository<sub>G</sub> remoto;
- **Gestione dei conflitti:** nel caso si verifichino conflitti sarà necessario usare il comando `git mergetool` e risolvere il conflitto tramite un tool di merge<sub>G</sub>.

### 6.1.3 Software di Integrazione Continua

Si è scelto di adottare **Jenkins** per applicare l'integrazione continua allo sviluppo del progetto.

Tale software permette di pianificare ed impostare la compilazione del codice e dei documenti. Mette inoltre a disposizione un cruscotto su cui è possibile visualizzare lo stato del codice prodotto. Esso è infatti in grado di interagire con il software di versionamento e, se disponibile, con software per l'esecuzione di test sul codice prodotto. Attualmente Jenkins viene utilizzato per la compilazione dei documenti  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ . Tale software è disponibile alla pagina:

<http://dtp.myopenproject.it:8070/>

### 6.1.4 Condivisione dei file

Si è inoltre scelto di utilizzare degli strumenti online che permettono di condividere file in modo semplice e veloce e che consentono di organizzare gli appuntamenti personali dei singoli componenti del gruppo.

#### 6.1.4.1 Dropbox

In questa piattaforma di condivisione devono essere caricati solo file che non necessitano di controllo di versione<sub>G</sub> e che sono comunque recuperabili da altre fonti (es. internet). Si caricheranno qui i file di installazione dei software utilizzati dal gruppo in modo da uniformare le versioni utilizzate da ogni componente.

Vengono resi disponibili anche i manuali (software, libri, PDF<sub>G</sub>) di consultazione così da avere uniformità di versione e di informazione. Altresì è presente una cartella dove il server esegue il salvataggio dei backup del database.

La possibilità di installare Dropbox<sub>G</sub> sul proprio PC dà a ogni componente del gruppo la possibilità di avere a disposizione documentazione e software anche offline.

#### 6.1.4.2 Google Drive

In questa piattaforma di condivisione file verranno salvati i documenti che:

- Non necessitano di controllo di versione<sub>G</sub>;
- Hanno bisogno di grande interattività tra i componenti del gruppo;
- Possono essere acceduti tramite l'uso di un semplice browser.



Questo strumento dovrebbe permettere a 2 o più componenti del gruppo di interagire lavorando sugli stessi documenti contemporaneamente. Google Drive<sub>G</sub> viene utilizzato come strumento di supporto allo sviluppo della documentazione e del software presente su Git<sub>G</sub>.

### 6.1.5 Google Calendar

Google Calendar<sub>G</sub> viene utilizzato all'interno del gruppo per gestire le risorse umane. In particolare tale strumento viene utilizzato per notificare in quali giorni un determinato membro non può essere disponibile e per segnalare date rilevanti per il gruppo, come ad esempio le date delle riunioni.

## 6.2 Pianificazione

Per pianificare le attività legate allo sviluppo del progetto e la gestione delle risorse si è scelto di utilizzare **ProjectLibre**.

ProjectLibre è un programma open source per il project management. È basato su Java<sub>G</sub> ed è quindi eseguibile su ogni sistema operativo. È il successore ufficiale di OpenProj. Tale software è stato scelto in quanto possiede le seguenti caratteristiche:

- Portabilità, essendo basto su Java<sub>G</sub>;
- Open-source;
- Genera diagrammi Gantt<sub>G</sub>;
- Genera automaticamente diagrammi Program Evaluation and Review Technique(PERT<sub>G</sub>), a partire dal Gantt<sub>G</sub>;
- Genera automaticamente la Work Breakdown Structure<sub>G</sub> (WBS<sub>G</sub>), a partire dal Gantt<sub>G</sub> con allocazione di risorse;
- Calcola i parametri Schedule Variance<sub>G</sub> (SV<sub>G</sub>) e Budget Variance<sub>G</sub> (BV<sub>G</sub>);
- Compatibile con Microsoft Project 2010;
- Salvataggio su file XML<sub>G</sub>: essendo un file testuale è possibile effettuare il merge<sub>G</sub> di più file in caso di conflitti sul repository<sub>G</sub>.

### 6.2.1 Modalità di utilizzo

Il *Responsabile di Progetto* deve creare un progetto per ogni fase indicata nella sezione Pianificazione del *Piano di Progetto v5.2.0* e procedere nel modo seguente:

1. Creare un calendario lavorativo per il progetto;
2. Inserire le attività da svolgere e le corrispondenti sotto-attività;
3. Inserire le dipendenze temporali tra le attività;
4. Inserire i periodi di slack dove previsto;
5. Inserire la milestone<sub>G</sub> per indicare il termine previsto delle attività;
6. Creare le risorse;



7. Assegnare ad ogni attività le risorse necessarie;
8. Salvare la  $baseline_G$ .

ProjectLibre viene anche utilizzato come strumento di controllo in quanto permette di calcolare automaticamente le metriche di  $Budget\ Variance_G$  e di  $Schedule\ Variance_G$ . Per far ciò il *Responsabile* deve modificare i progetti creati nel modo seguente:

1. Inserire le colonne  $SV_G$ , BCWS, ACWP;
2. Modificare la durata delle sotto-attività in base a quanto è stato consuntivato;
3. Modificare le risorse assegnate alle sotto-attività in base a quanto è stato consuntivato;
4. Aggiornare il progetto alla data corrente.

## 6.3 Strumenti per i documenti

### 6.3.1 $\text{\LaTeX}$

Per la stesura dei documenti si è scelto di utilizzare il linguaggio di markup  $\text{\LaTeX}$ . Il motivo principale che ha portato a questa scelta è la facilità di separazione tra contenuto e formattazione: con  $\text{\LaTeX}$  è possibile definire l'aspetto delle pagine in un file template condiviso da tutti i documenti. Altre soluzioni come Microsoft Office, LibreOffice o Google Docs non avrebbero consentito questa separazione, duplicando il lavoro di formattazione del testo e non garantendo un risultato uniforme.

Il grande numero di pacchetti esistenti consente di implementare funzionalità comuni in maniera semplice. L'estensibilità di  $\text{\LaTeX}$  può essere sfruttata per creare funzioni e variabili globali che rendono la scrittura del contenuto più corretta da un punto di vista semantico. Un esempio è dato dal comando `\role{Ruolo}` che identifica ogni ruolo all'interno del progetto.

Per la scrittura di documenti  $\text{\LaTeX}$ , l'editor consigliato è **TeXstudio**<sup>8</sup>.

### 6.3.2 Script

Per semplificare il lavoro di scrittura dei documenti sono stati creati alcuni script accessibili tramite il comando `make`. Lo script presente all'indirizzo `documents/Makefile` consente di:

- **Generare tutti i documenti:** eseguendo il comando `make all` tutti i documenti verranno generati;
- **Generare un archivio con tutti i documenti:** eseguendo il comando `make dist` tutti i documenti verranno generati e inseriti in un archivio;
- **Evidenziare le parole presenti nel Glossario:** eseguendo il comando `make glossary` tutti i file  $\text{\LaTeX}$  in nella cartella `documents` verranno esaminati e tutte le parole presenti nel *Glossario* verranno evidenziate tramite un tag  $\text{\LaTeX}$ . Tutte le parole evidenziate che non hanno una corrispondenza nel *Glossario* verranno riportate alla normalità.

---

<sup>8</sup><http://texstudio.sourceforge.net/>



Attraverso lo script presente in `documents/NomeDocumento/Makefile`, e condiviso da tutte le cartelle contenenti un documento, è possibile:

- **Generare il pdf<sub>G</sub> del documento:** il comando `make build` dev'essere utilizzato al solo scopo di visualizzare un'anteprima durante il lavoro. Per la generazione del documento è stato scelto Latexmk per la sua possibilità di iterare i comandi `pdflatex`, `makeindex` e `makeglossaries` fino ad ottenere un documento con tutti i riferimenti risolti;
- **Eliminare i file generati da compilazioni errate:** il comando `make clean` consente di eliminare i file generati precedentemente tramite il comando `make build`, in particolare in seguito ad un errore di sintassi;
- **Calcolare l'indice Gulpease:** attraverso il comando `make gulpease` lo script estrarrà dal documento corrente il testo ed effettuerà il calcolo dell'indice di leggibilità Gulpease. L'indice è calcolato secondo la seguente formula:

$$89 + \frac{300 * (\text{numero delle frasi}) - 10 * (\text{numero delle lettere})}{\text{numero delle parole}}$$

- **Controllo ortografico:** attraverso il comando `make aspell` verrà invocato il programma Aspell per il documento corrente.

Si prevede la possibilità di estendere ulteriormente i Makefile presentati in base alle esigenze che si incontreranno nelle fasi di sviluppo.

### 6.3.3 Verifica

Per la verifica dei documenti si utilizzano:

- **TeXstudio:** Per la scrittura di documenti è consigliato utilizzare l'ambiente grafico TeXstudio. Tale strumento integra i dizionari di OpenOffice.org e segnala i potenziali errori ortografici presenti nel testo durante la stesura del testo stesso;
- **Aspell:** Il software per il controllo ortografico è **Aspell**<sup>9</sup>. Il programma è accessibile tramite il `Makefile` oppure tramite interfaccia grafica nell'editor consigliato TeXstudio. In entrambi i casi l'interattività del programma non rende necessarie spiegazioni sull'utilizzo;
- **Tracy:** strumento realizzato dal gruppo Don't Panic, descritto nella sezione 6.3, permette di automatizzare molte operazioni di tracciamento e controllo;
- **Script:** i componenti del gruppo hanno scritto vari script per il calcolo dell'indice Gulpease e la marcatura dei termini presenti nel glossario del documento.

### 6.3.4 Diagrammi UML

Per quanto concerne la modellazione dei diagrammi dei caso d'uso, diagrammi di sequenza e diagrammi di attività è stato scelto l'editor **Visual Paradigm**<sup>10</sup>. Tale editor supporta l'UML<sub>G</sub> 2.4. Dopo aver valutato numerose alternative quali ArgoUML, Dia, LucidChart, StarUML, UMLet e Papyrus, Visual Paradigm è risultato l'unico software

<sup>9</sup><http://aspell.net/>

<sup>10</sup><http://www.visual-paradigm.com/>





in grado di coniugare il completo supporto a UML<sub>G</sub> 2.0 con un'interfaccia semplice e responsiva.

Per la realizzazione dei diagrammi delle classi e dei package<sub>G</sub> è stato scelto **WhiteStarUML**<sup>11</sup>. Questo software, a differenza di Visual Paradigm, supporta l'UML<sub>G</sub> 2.0 ma fornisce una funzionalità di importazione inversa delle modifiche effettuate: una volta definite le classi ed aver generato il codice, è in grado di importare eventuali modifiche rilevate nel codice nel diagramma. Inoltre, lo strumento interno Tracy permette di esportare file XMI compatibili con WhiteStarUML, rendendo così automatica l'importazione nel software di quanto progettato.

Per verificare la correttezza dei file XMI esportati da Tracy e i conseguenti diagrammi della classi, deve essere utilizzata una funzione propria di WhiteStarUML chiamata *Verifying Model*<sup>12</sup> che controlla che tali diagrammi siano ben formati attraverso una serie di regole che rispettano lo standard UML<sub>G</sub> 2.0.

#### 6.3.4.1 Creazione di un diagramma delle classi

Per la creazione di un diagramma delle classi, è necessario esportare da Tracy il file XMI contenente la definizione di tutte le classi. Per ottenere tale file seguire i seguenti passi:

- Selezionare *XMI* nella sezione *Generators* della pagina principale di Tracy;
- Selezionare *View Raw* dal menu *Operations*;
- Salvare il file con nome `ClassDefinition.xmi`.

#### 6.3.5 Fogli di calcolo

Per l'elaborazione dei dati si utilizza il software Calc del pacchetto LibreOffice in quanto tale prodotto è open source.

Calc viene utilizzato per elaborare i dati prodotti con ProjectLibre e produrre:

- Grafici a torta per l'utilizzo delle risorse;
- Grafici a torta per il costo dedicato a ciascuna risorsa;
- Istogrammi per le ore assegnate ad ogni componente del gruppo;
- Calcolo della metrica  $SV_G$  come somma delle singole righe della colonna  $SV_G$  di ProjectLibre;
- Calcolo della metrica  $BV_G$  come somma delle varie differenze tra la colonna BCWS e ACWP di ProjectLibre;
- Tabelle per il confronto tra preventivo e consuntivo;
- Istogrammi per il confronto tra ore preventivate e ore realmente impiegate da ciascuna risorsa.

<sup>11</sup><http://sourceforge.net/projects/whitestaruml/>

<sup>12</sup>[http://staruml.sourceforge.net/docs/user-guide\(en\)/ch09.html](http://staruml.sourceforge.net/docs/user-guide(en)/ch09.html)



## 6.4 Tracy

Per semplificare il tracciamento dei requisiti è stato realizzato internamente il software Tracy, un'applicazione web accessibile tramite browser ed ospitata sul server dedicato. Quest'applicazione consente di gestire il tracciamento e la gestione di fonti, casi d'uso, requisiti, package<sub>G</sub>, classi e metodi, automatizzando molte operazioni. Si interfaccia inoltre allo strumento di ticketing Redmine per generare il grafico PDCA, configurabile per periodi, revisioni e attività.

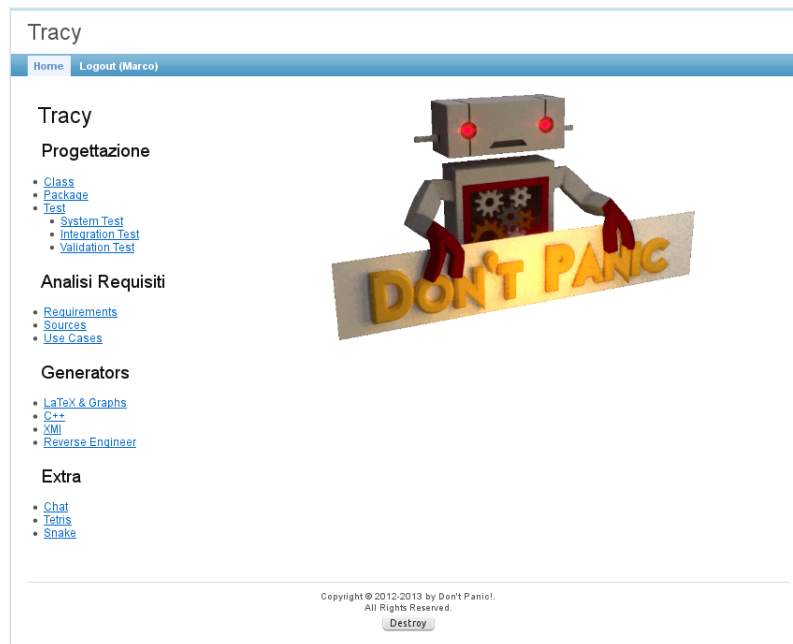


Figura 1: Tracy - screenshot pagina principale

### 6.4.1 Analisi dei Requisiti

Tracy permette la gestione dei requisiti e dei casi d'uso individuati dagli *Analisti*. Permette di automatizzare il tracciamento tra casi d'uso, requisiti e fonti, in modo da garantirne la copertura. Inoltre rende automatica la stesura di parti dell'*Analisi dei Requisiti v4.2.0*, grazie all'esportazione di codice L<sup>A</sup>T<sub>E</sub>X relativo a casi d'uso, requisiti e tabelle di tracciamento.

Per i casi d'uso genera un'anteprima indicativa del diagramma mostrando i casi d'uso del flusso di eventi principale (figura 6). Tale diagramma non è conforme ad UML<sub>G</sub>, il suo uso si limita a fornire un'idea visiva dei dati inseriti.

È inoltre possibile tracciare i requisiti ai test di validazione e di sistema definiti.

#### 6.4.1.1 Aggiunta di un nuovo UC

Per aggiungere un nuovo caso d'uso alla gerarchia dei casi d'uso è necessario eseguire i passi descritti nel diagramma in figura 2(a).



#### 6.4.1.2 Aggiunta di un nuovo requisito

Per aggiungere un nuovo requisito al progetto è necessario eseguire i passi descritti nel diagramma in figura 2(b).

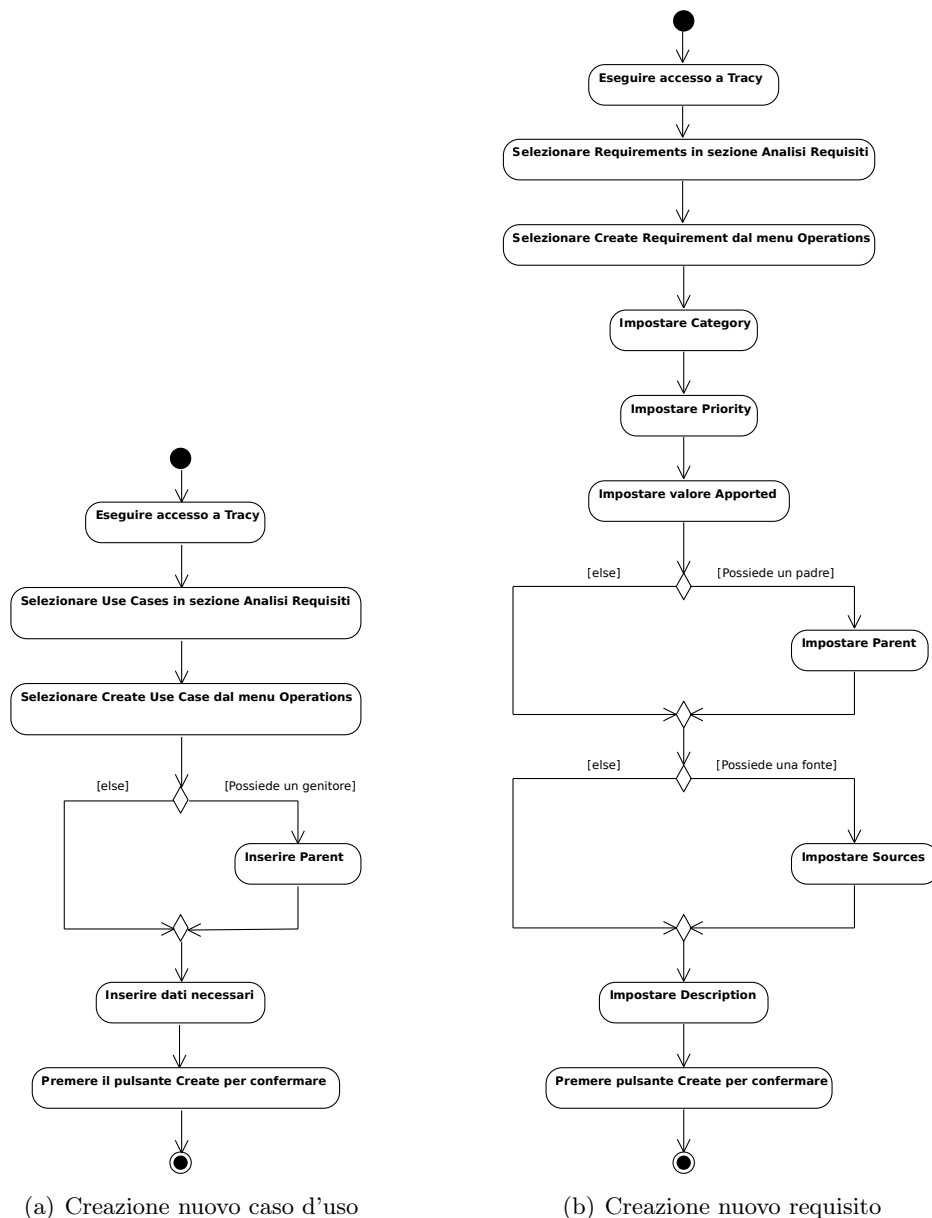


Figura 2: Diagramma di attività - Casi d'uso / Requisiti

#### 6.4.2 Progettazione e codifica

Tracy consente di gestire  $\text{package}_G$  (figura 8), classi (figura 7) e prototipi dei metodi delle classi, il cui inserimento nel software spetta ai *Progettisti*. Permette di automatizzare il tracciamento requisiti-componenti e componenti-package<sub>G</sub>, ed è in grado di esportare il codice C++ delle classi (figura 11) e dei prototipi dei metodi delle classi, garantendo uniformità tra progettazione, codifica e documentazione. Il codice C++ esportato contiene anche i commenti appropriati per la generazione automatica della documenta-



zione con Doxygen, aiutando a rispettare la corrispondenza tra documentazione e codice.

Permette inoltre di esportare file XMI, contenenti informazioni su classi e package<sub>G</sub>, compatibili con l'editor UML<sub>G</sub> WhiteStarUML. In questo modo, sarà necessario solamente il disegno dei diagrammi di classi e package<sub>G</sub> desiderati, ma non la ridefinizione delle classi e package<sub>G</sub> progettati all'interno dell'editor UML<sub>G</sub>.

Tracy può inoltre calcolare i parametri di accoppiamento afferente ed efferente di ogni classe, descritti nel *Piano di Qualifica*, permettendo di automatizzarne la verifica.

#### 6.4.2.1 Aggiunta di una nuova classe

Per aggiungere una nuova classe alla gerarchia è necessario eseguire i passi descritti nel diagramma in figura 3(a).

#### 6.4.2.2 Aggiunta di un nuovo metodo di classe

Per aggiungere un nuovo metodo ad una classe esistente è necessario eseguire i passi descritti nel diagramma in figura 3(b).

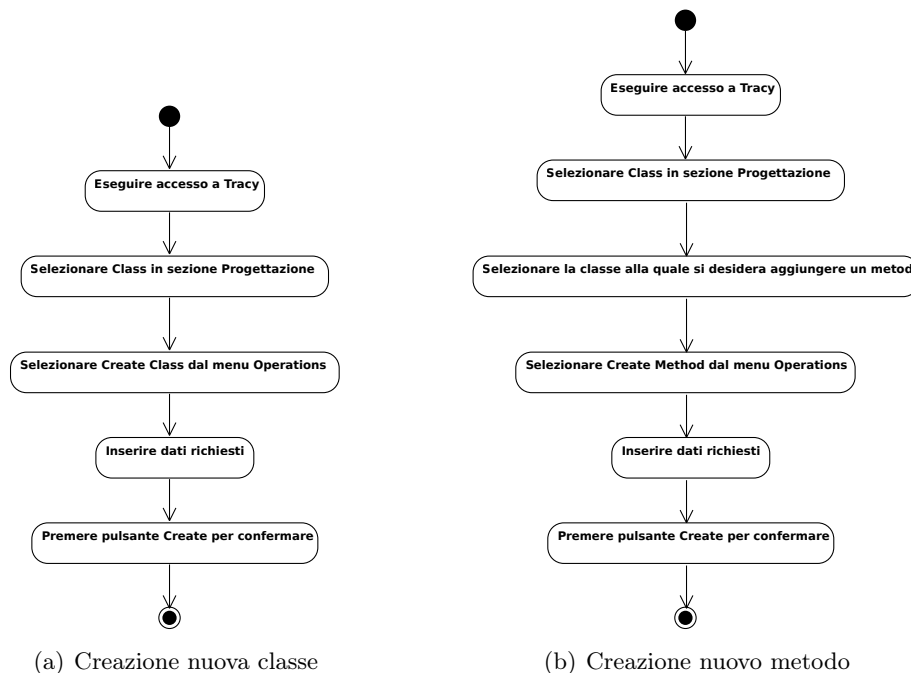


Figura 3: Diagramma di attività - Classi / Metodi

#### 6.4.3 Grafico PDCA

Attraverso il software Tracy è possibile visualizzare il grafico PDCA relativo alle attività in svolgimento:

- Selezionare *Latex & Graphs* dalla sezione *Generators* della pagina principale di Tracy;
- Selezionare *PDCA* dalla sezione relativa all'attività desiderata (es. Progettazione).



#### 6.4.4 Test

Tracy permette di descrivere e tracciare i test di unità, integrazione, sistema e validazione, assegnando identificativi univoci e permettendo di definirne la descrizione e le relazioni con classi, componenti o requisiti, a seconda della tipologia di test. È così possibile tracciare i test di unità alle classi, i test di integrazione ai vari componenti o macro componenti e i test di sistema e validazione ai requisiti.

Permette inoltre di tracciare i test al codice assegnato da Jenkins all'esecuzione dello stesso, e di conseguenza al risultato, permettendo il controllo delle attività di test automatico predisposte.

#### 6.4.5 Tracciamento

Il software Tracy è in grado di esportare le seguenti tabelle per il tracciamento:

- Requisiti  $\rightarrow$  Fonti;
- Fonti  $\rightarrow$  Requisiti;
- Requisiti  $\rightarrow$  Componenti;
- Componenti  $\rightarrow$  Requisiti;
- $\text{Package}_G \rightarrow$  Componenti  $\rightarrow$  Classi;
- Requisiti  $\rightarrow$  Test di sistema  $\rightarrow$  Test di validazione;
- Test di sistema  $\rightarrow$  Requisiti;
- Test di validazione  $\rightarrow$  Requisiti.

Il codice `LATEX` da includere nei relativi documenti viene generato da Tracy e aggiunto tramite lo script `download_from_tracy.sh`.

### 6.5 Protocollo per lo sviluppo dell'applicazione

Per procedere con uno sviluppo controllato dei documenti e del codice si è scelto di adottare il sistema di ticketing **Redmine**.

La scelta di tale software è descritta nella sezione 6.1.1.

In questa sezione vi saranno molti riferimenti impliciti a contenuti del *Piano di Progetto v5.2.0* e del *Piano di Qualifica v5.2.0*.

#### 6.5.1 Creare un nuovo progetto

La creazione di un progetto è compito del *Responsabile di Progetto*.

Un nuovo progetto rappresenta una macro-attività caratterizzata da molte sotto-attività supervisionate da un responsabile.

Per creare un nuovo progetto:

- Aprire **Progetti**;
- Selezionare **Nuovo progetto**;
- Assegnare un **Nome** breve ma significativo;



- Nel caso in cui si voglia creare un sotto-progetto indicare il nome del progetto padre dall'omonimo campo;
- **Identificativo:** scrivere in minuscolo ed indicare codice della fase a cui si riferisce (es. ndp-rr);
- Lasciare inalterati gli altri campi.

### 6.5.2 Creazione ticket

I ticket vengono creati da:

- **Responsabile di Progetto:** crea i ticket più importanti che rappresentano le macro fasi evidenziate dalla pianificazione (vedi *Piano di Progetto v5.2.0*);
- **Responsabile di Sotto-progetto:** crea i ticket per i processi non pianificati inizialmente, che si evidenziano necessari per l'avanzamento del sotto-progetto assegnato;
- **Verificatore:** crea i ticket per segnalare errori ed imprecisioni trovate durante il processo di verifica.

I ticket possono essere di tre tipologie:

- **Ticket di pianificazione:** rappresentano le macro-attività di maggiore importanza. Sono organizzate in una gerarchia con vari livelli di importanza. Tali attività vengono create da:
  - *Responsabile di Progetto* che durante la pianificazione identifica le attività più importanti e generali;
  - *Responsabile di Sotto-progetto* che durante lo svolgimento delle attività può scomporre in sotto-problemi l'attività indicata dal *Responsabile di Progetto*.
- **Ticket di realizzazione e controllo:** tutti i documenti redatti, durante la stesura attraversano due stadi:
  - **Realizzazione:** un redattore del documento effettua una prima stesura;
  - **Controllo:** un redattore, diverso da quello della precedente fase, esegue un primo controllo sui contenuti della parte scritta.
- **Ticket di verifica:** rappresentano gli errori identificati dai *Verificatori* durante il controllo che la realizzazione dell'attività sia conforme a quanto richiesto e che rispetti tutte le norme.

#### 6.5.2.1 Ticket di pianificazione

- Selezionare **Nuova segnalazione** da menù principale;
- **Tracker:** indicare la natura del ticket:
  - **Documento:** stesura di un documento. Il tipo di attività svolta dal redattore del documento viene definito durante la rendicontazione;
  - **Codifica:** stesura di codice;
  - **Verifica:** macro-attività di verifica sul prodotto dei sotto-processi.



- **Oggetto:** descrizione breve e significativa;
- **Descrizione:** descrizione comprensibile e con riferimenti esterni mediante link se necessario;
- **Stato:** Plan;
- **Attività principale:** se si vuole creare una **sotto-attività** indicare l'id del ticket padre;
- **Categoria:** PDCA, solo se il ticket viene creato dal *Responsabile di Progetto*;
- **Assegnato a:** inserire il nome del responsabile;
- **Osservatori:** aggiungere eventuali collaboratori.

#### 6.5.2.2 Ticket di realizzazione e controllo

- Selezionare **Nuova segnalazione** da menù principale;
- **Tracker:** indicare la natura del ticket:
  - **Documento:** stesura di un documento. Il tipo di attività svolta dal redattore del documento viene definito durante la rendicontazione;
  - **Codifica:** stesura di codice;
  - **Verifica:** attività di verifica sui prodotti dei processi.
- **Oggetto:** descrizione breve e significativa secondo il principio: nome ticket padre - attività da svolgere (realizzazione o controllo);
- **Descrizione:** descrizione comprensibile e con riferimenti esterni mediante link se necessario;
- **Stato:** New;
- **Attività principale:** se si vuole creare una **sotto-attività** indicare l'id del ticket padre;
- **Inizio:** dare una data di inizio presunta;
- **Scadenza:** dare una data di fine presunta;
- **Assegnato a:** inserire il nome del responsabile;
- **Osservatori:** aggiungere eventuali collaboratori.

#### 6.5.2.3 Ticket di verifica

Un *Verificatore* per creare un *ticket di verifica* deve:

1. assicurarsi che esista all'interno del progetto l'attività *Verifica*.  
Su tale attività vi devono essere due sotto-attività: "Verifica - realizzazione", "Verifica - approvazione".  
Tutti i ticket creati devono essere sotto-attività di: "Verifica - realizzazione";
2. Creare quindi il ticket secondo le seguenti direttive:



- Selezionare **Nuova segnalazione** da menù principale;
- **Tracker:** Bug;
- **Oggetto:** descrizione breve e significativa dell'errore incontrato;
- **Descrizione:** descrivere in modo dettagliato e chiaro: la natura e la posizione dell'errore;
- **Stato:** New;
- **Attività principale:** tutti i ticket devono essere figli del ticket "Verifica - realizzazione" del progetto su cui si sta eseguendo la verifica;
- **Assegnato a:** inserire il nome del responsabile del progetto padre (es. responsabile delle *Norme di Progetto*).

Tutti i campi non segnalati sono da lasciare vuoti.

Sarà poi compito del responsabile del progetto padre decidere a chi assegnare la correzione dell'errore. Nel caso in cui l'errore segnalato non sia considerato valido dal *Responsabile del sotto-progetto* verrà confermato il rifiuto dal *Responsabile di Progetto*.

#### 6.5.2.4 Dipendenze temporali

Dopo la creazione del ticket, per aggiungere **dipendenze temporali** tra i ticket:

- Andare su **segnalazioni**;
- Aprire il link alla segnalazione a cui aggiungere la dipendenza;
- Nella sezione **segnalazioni correlate** premere **aggiungi**;
- Scegliere **segue** e indicare il numero della segnalazione che lo blocca ed eventuali giorni di slack.

Tutti i campi non segnalati sono da lasciare vuoti.

#### 6.5.3 Aggiornamento ticket

Esistendo due tipologie di ticket, viene qui definito la procedura per effettuare l'aggiornamento di entrambe.

##### 6.5.3.1 Ticket di pianificazione

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket di interesse;
- Cliccare il link **Aggiorna**;
- Commentare ciò che si è fatto sulla form **Note**;
- Cambiare lo stato del ticket secondo la seguente logica:
  - **Do:** quando un ticket è in questo stato indica che una o più persone stanno lavorando su tale attività;
  - **Check:** quando un ticket è in questo stato indica che una o più persone stanno lavorando sulla verifica di tale attività;



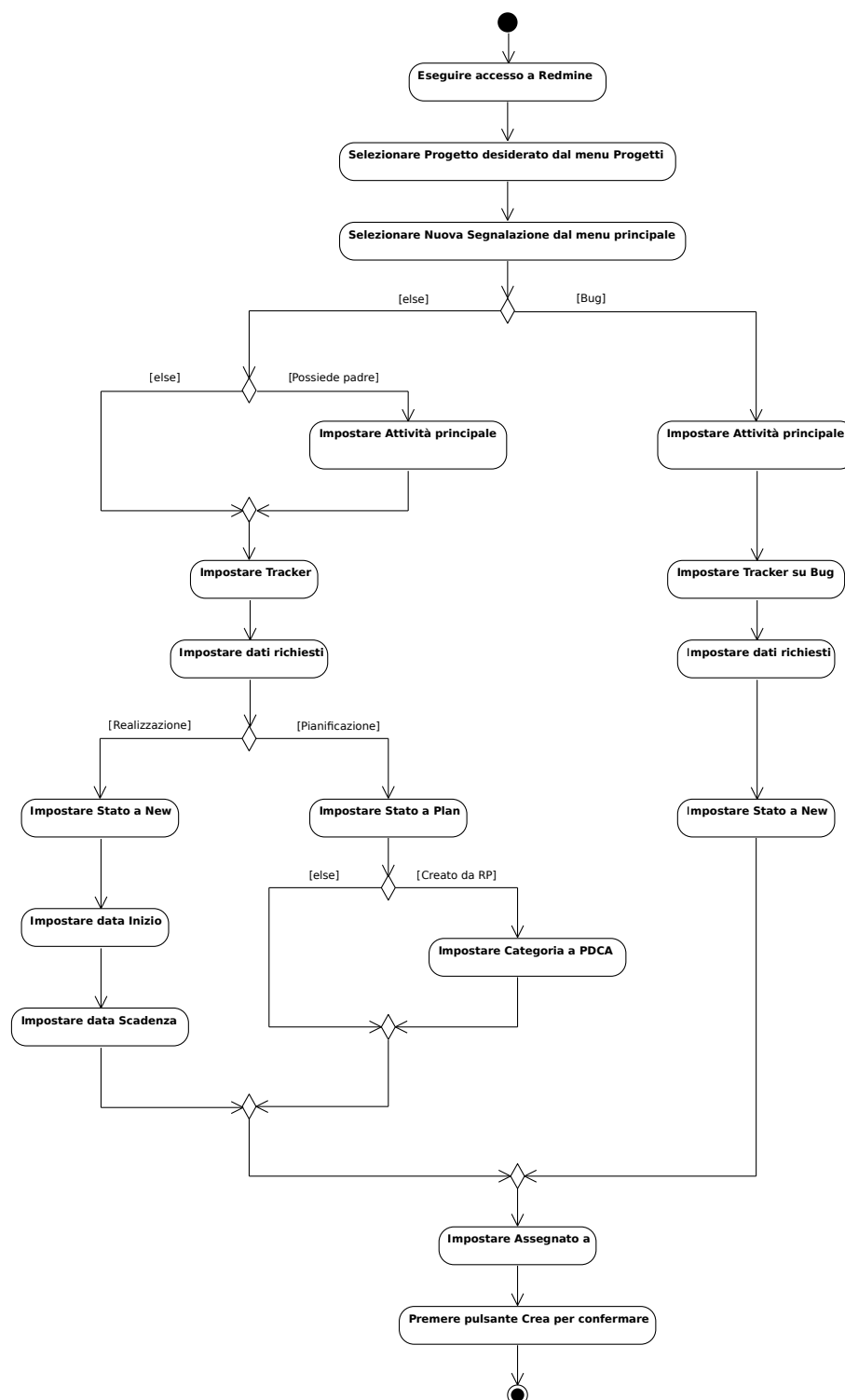


Figura 4: Diagramma attività - Creazione nuovo ticket

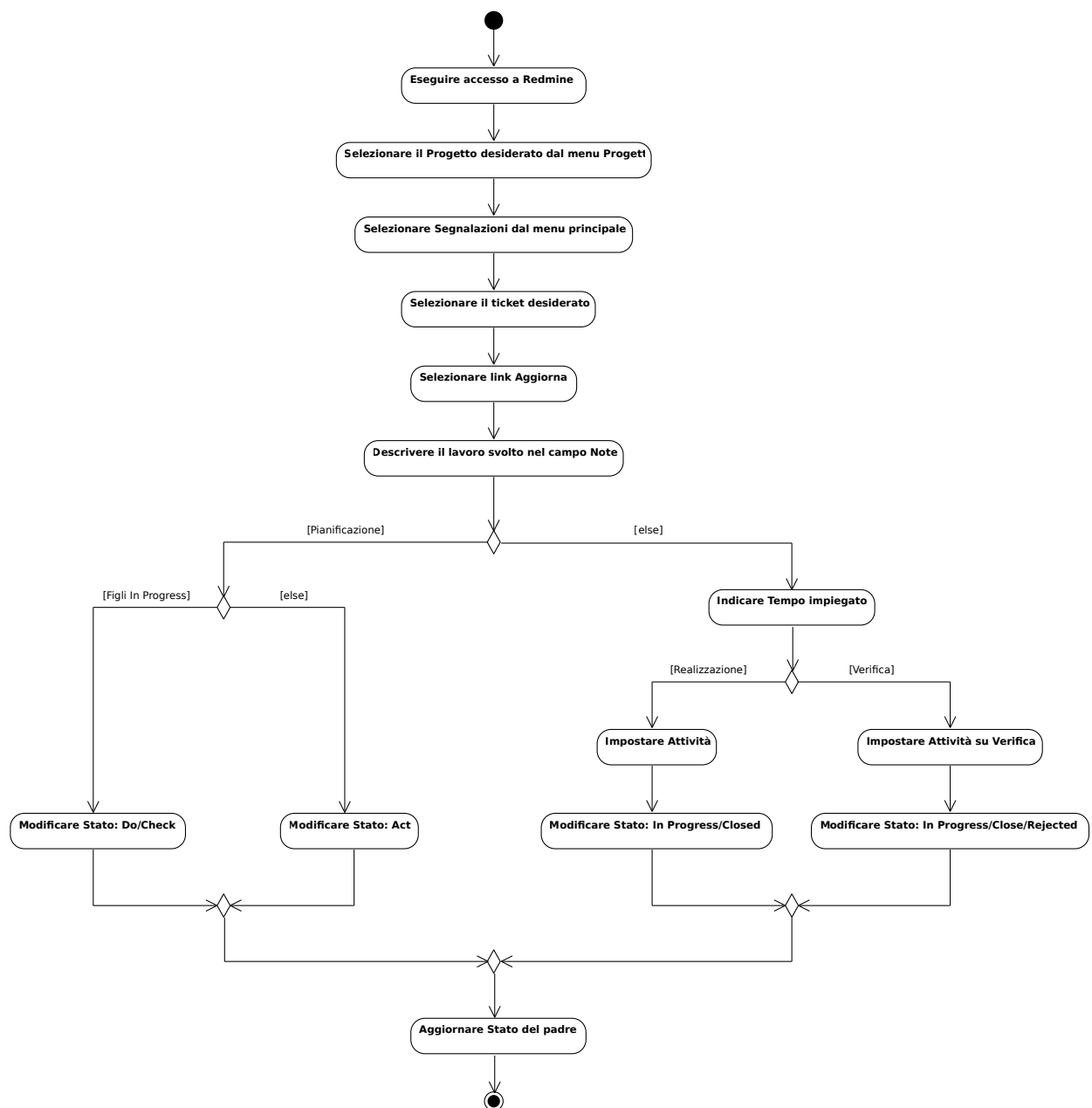


Figura 5: Diagramma attività - Aggiornamento ticket esistente



- **Act:** l'attività è stata conclusa e verificata, e ne sono state tratte le conclusioni adeguate.
- Se viene concluso, aggiornare lo stato del ticket di pianificazione padre.

#### 6.5.3.2 Ticket di realizzazione e controllo

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket di interesse;
- Cliccare il link **Aggiorna**;
- Indicare il tempo impiegato in ore;
- Indicare il tipo di attività svolta;
- Commentare ciò che si è fatto sulla form **Note**;
- Cambiare lo stato del ticket secondo la seguente logica:
  - **In Progress:** quando un ticket è in questo stato indica che una o più persone stanno lavorando su tale attività. La percentuale di completamento deve essere impostata tra lo 0% ed il 90%;
  - **Closed:** l'attività è stata conclusa. La percentuale di completamento dell'attività è al 100%.
- Aggiornare lo stato del ticket di pianificazione padre secondo tali principi:
  - ticket figlio passa da New a In Progress: il ticket padre passa da Plan a Do, o da Do a Check;
  - ticket figlio passa a Closed: il ticket padre deve essere in Do o Check;
  - tutti i ticket figli vengono chiusi: il ticket padre passa ad Act.

#### 6.5.3.3 Ticket di verifica

- Andare sul menù **Segnalazioni**;
- Selezionare il ticket di interesse;
- Cliccare il link **Aggiorna**;
- Indicare il tempo impiegato in ore;
- Indicare Verifica come tipo di attività svolta;
- Commentare le correzione nella form **Note**;
- Cambiare lo stato del ticket secondo la seguente logica:
  - **In Progress:** quando un ticket è in questo stato indica che una o più persone stanno lavorando su tale attività. La percentuale di completamento deve essere impostata tra lo 0% ed il 90%;
  - **Closed:** l'attività è stata conclusa. La percentuale di completamento dell'attività è al 100%;



- **Rejected:** l'attività di verifica è stata rifiutata dal *Responsabile del sotto-progetto* in accordo con il *Responsabile di Progetto*.
- Aggiornare lo stato del ticket di pianificazione padre secondo tali principi:
  - ticket figlio passa da New a In Progress: il ticket padre passa da Plan a Do, o da Do a Check;
  - ticket figlio passa a Closed: il ticket padre deve essere in Do o Check;
  - tutti i ticket figli vengono chiusi: il ticket padre passa ad Act.

#### 6.5.4 Consigli di utilizzo

**Pagina personale** Per avere una immediata visualizzazione dei ticket assegnati, è consigliato personalizzare la pagina personale:

- Andare alla **Pagina personale**;
- Cliccare il link **Personalizza la pagina**;
- Dal menù a tendina **La mia pagina di blocco**, selezionare **Le mie segnalazioni** e premere il pulsante verde +;
- Ripetere il punto precedente per aggiungere **Segnalazioni osservate**.

**Visualizzare segnalazioni** Per avere una visualizzazione più chiara delle segnalazioni si consiglia di ordinarle per oggetto. Tale risultato può essere ottenuto premendo **Oggetto** dalla pagina **Segnalazioni**.

### 6.6 Strumenti per lo sviluppo dell'applicazione

Si è deciso di preparare una macchina virtuale tramite **Virtualbox**<sup>13</sup> contenente tutto il software necessario per sviluppare l'applicazione. Uniformando l'ambiente di sviluppo viene facilitata la comunicazione e la riproduzione delle segnalazioni.

#### 6.6.1 Framework

Per la realizzazione del progetto è stato scelto il framework<sub>G</sub> **Qt**<sub>G</sub><sup>14</sup>.

QtCreator è l'IDE<sub>G</sub> che verrà utilizzato per lo sviluppo. È stato scelto, oltre che per l'ottima integrazione con Qt<sub>G</sub>, perché combina in un'unica interfaccia strumenti di editing, documentazione ufficiale, strumenti di debugging, strumenti di profiling<sub>G</sub> e strumenti di versionamento. La libreria di riferimento attuale è la "Qt<sub>G</sub> libraries 5.0.0".

#### 6.6.2 Documentazione

È stato scelto il sistema di documentazione **Doxygen**<sup>15</sup> che consente di inserire descrizioni di classi e metodi tramite commenti all'interno del codice. In questo modo la lettura del codice diventa un processo più immediato e la scrittura della documentazione viene incorporata nel codice, diminuendo il rischio di inconsistenze tra codice e

<sup>13</sup><https://www.virtualbox.org/>

<sup>14</sup><http://qt-project.org/>

<sup>15</sup><http://www.stack.nl/~dimitri/doxygen/>



documentazione. Doxygen consente di generare la documentazione in formato HTML e in formato  $\text{\LaTeX}$ .

La generazione dei commenti con il formato accettato da Doxygen è resa automatica da Tracy che, basandosi sui dati inseriti dai *Progettisti* nella descrizione delle classi e dei metodi, provvederà a generare i file sorgente con appropriati commenti. Il vantaggio di questa scelta, oltre alla garanzia di corrispondenza tra progettazione e documentazione, è che permette al *Programmatore* di avere sempre la descrizione puntuale di quanto previsto dal *Progettista* presente nel sorgente. Non è quindi richiesta la stesura di alcuna documentazione relativa alla progettazione al *Programmatore*, al quale permane però l'obbligo di commentare il più possibile il codice con il quale implementa i metodi.

### 6.6.3 Validazione output

Per validare l'output del programma in riferimento agli schemi associati vengono seguite procedure differenziate, in base al tipo di file esportato.

#### 6.6.3.1 XML

Per validare un file  $\text{XML}_G$  utilizzare l'IDE<sub>G</sub> Eclipse<sup>16</sup>.

Installare il plugin: Eclipse XML<sub>G</sub> Editors and Tools.

Creare un nuovo progetto, importare il file  $\text{xml}_G$  da validare ed il file contenente lo schema.

Utilizzare la funzionalità *Validate* utilizzabile mediante click destro sul file da validare.

#### 6.6.3.2 JSON

Per validare un file  $\text{JSON}_G$  utilizzare il servizio messo disponibile al sito:

<http://json-schema-validator.herokuapp.com/>

## 6.7 Strumenti per la codifica

### 6.7.1 Stesura

Per la scrittura del codice il *Programmatore* dovrà utilizzare l'IDE<sub>G</sub> QtCreator, la cui versione di riferimento 2.6.1 è presente nella macchina virtuale fornita dall'*Amministratore*, ed aprire il progetto presente nel repository<sub>G</sub> *src.git*.

### 6.7.2 Verifica

L'analisi statica e l'analisi dinamica integrate in Jenkins vengono lanciate automaticamente ad ogni push di codice nel repository<sub>G</sub>. Al verificatore è richiesto quindi soltanto di analizzare l'output del tool, visualizzabile dalla dashboard<sub>G</sub> di Jenkins.

### 6.7.3 Scrittura dei test

Per scrivere un test è necessario creare una classe specifica per il testing e utilizzare le macro `QVERIFY` e `QCOMPARE` definite da QtTest. Ogni slot privato della classe viene interpretato come un test dal framework<sub>G</sub> di testing.

<sup>16</sup><http://www.eclipse.org/>



Passando a `QVERIFY` o a `QCOMPARE` una condizione falsa il test fallisce. `QCOMPARE` aggiunge al log anche delle informazioni aggiuntive sugli oggetti.  
Segue un esempio di test:

```
#include <QtTest/QtTest>

class TestTriangle: public QObject
{
    Q_OBJECT
    private slots:
        void rotation();
};

void TestTriangle::rotation()
{
    Triangle triangle(Point(0,0), Point(4,1), Point(3,3));
    triangle.rotate(Point(0,0), 90);

    QVERIFY(triangle.vertex(1) == Point(0,0));
    QVERIFY(triangle.vertex(2) == Point(1,-4));
    QVERIFY(triangle.vertex(3) == Point(3,-3));
}
```



## A Lista di controllo

Durante l'applicazione del walkthrough ai documenti, sono state riportate le tipologie di errori più frequenti. La lista di controllo risultante è la seguente:

- **Norme stilistiche:**

- Elenco puntato: non inizia con la lettera maiuscola;
- Elenco puntato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
- Elenco numerato: non termina con il punto e virgola oppure con il punto se è l'ultimo elemento;
- Nome proprio di persona: non rispetta la norma Cognome Nome;
- Nome ruolo di progetto: non viene utilizzata la macro predisposta;
- Nome documento: non viene utilizzata la macro predisposta;
- Parole Proponente e Committente: non vengono scritte con la maiuscola iniziale.

- **Italiano:**

- Parola destinatario: viene erroneamente scritta come destinatario;
- Virgola tra soggetto e verbo: rende di difficile comprensione la frase;
- Carattere È: non viene scritto correttamente utilizzando il comando apposito;
- Periodi: frasi troppo lunghe rendono i concetti di difficile comprensione;
- Doppie negazioni: evitare l'utilizzo di doppie negazioni perché complicano la comprensione della frase;
- Punto e virgola: evitare l'uso del punto e virgola quando è necessario usare il punto;
- Proponente e Committente: non si deve confondere il loro significato.

- **L<sup>A</sup>T<sub>E</sub>X:**

- Lettere accentate nelle variabili: non viene utilizzato il comando apposito;
- Carattere di spaziatura: non deve essere utilizzato all'interno dei tag;
- Macro L<sup>A</sup>T<sub>E</sub>X: non viene scritta usando l'apposito comando `\LaTeX{}`.

- **UML<sub>G</sub>:**

- Il sistema non deve mai essere un attore;
- Controllo ortografico: deve essere effettuato in modo dettagliato a causa dell'impossibilità di automatizzare i controlli sui diagrammi;
- Direzione delle frecce non corrette;
- Consistenza della nomenclatura tra i diagrammi e le descrizioni testuali nei documenti.

La seguente lista di controllo vuole riassumere invece gli errori più frequenti rilevati durante il walkthrough del tracciamento requisiti effettuato mediante il software Tracy:



- **Tracciamento requisiti:**

- Ad ogni caso d'uso deve corrispondere almeno un requisito;
- Ad ogni requisito deve corrispondere almeno una fonte;
- La fonte "Capitolato" non deve comparire nei requisiti interni;
- Deve esserci copertura totale del capitolato nei requisiti;
- La checkbox appointed deve essere spuntata solo nei requisiti opzionali e desiderabili presi in carico;
- Devono essere impostate le corrette relazioni di parentela tra requisiti in Tracy;
- Devono essere impostate le corrette relazioni di parentela tra casi d'uso in Tracy;
- In Tracy deve essere indicato almeno un attore in ogni caso d'uso;
- Controllare in Tracy che le fonti dei requisiti siano le fonti corrette;
- I codici dei casi d'uso nei diagrammi e in Tracy devono corrispondere.





## B Screenshot Tracy

Tracy

[Home](#) [Logout \(Mattia\)](#)

[Home](#) » [Use Cases](#) » 32

### View UseCase #32 (UC1.2.2)

<b>Id Use Case</b>	32
<b>Title</b>	Modifica della luce
<b>Parent</b>	<a href="#">UC1.2 - Modifica della scena</a>
<b>Description</b>	L'utente può effettuare varie operazioni sulla luce selezionata, dall'effettuare una traslazione al modificarne le caratteristiche
<b>Pre</b>	Il sistema presenta una fonte di luce selezionata
<b>Post</b>	La fonte di luce selezionata è stata modificata

#### Operations


[List Sources](#)  
[Create ExternalSource](#)  
[Create Use Case](#)  
[Update UseCase](#)  
[Delete UseCase](#)  
[View Image](#)

### Event Flow

#### Main Scenario

[Add new Event to Main Scenario](#)

Displaying 1-2 of 2 results.

Id Event	Category	Description	Refers To	Primary Actor	Order	
45	Principale	Traslazione luce	<a href="#">UC1.2.2.1</a>	Utente	1	  
46	Principale	Modifica caratteristiche luce	<a href="#">UC1.2.2.2</a>	Utente	2	  

#### Alternate

[Add new Event to Alternate Scenario](#)

Id Event	Category	Description	Refers To	Primary Actor	Order	
----------	----------	-------------	-----------	---------------	-------	--

No results found.

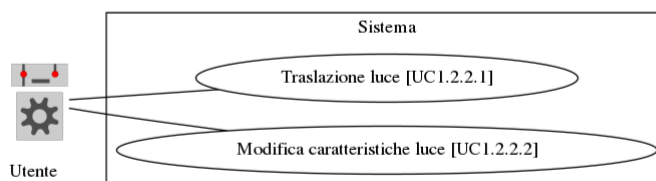


Figura 6: Tracy - screenshot gestione casi d'uso



# Tracy

[Home](#) [Logout \(Marco\)](#)

[Home](#) » [Class](#) » [SceneObject](#)

## View Class

### [3DMob::Model::CBasicEditor::C3DObject::SceneObject](#)

Id Class_Prog	21
Name	SceneObject
Type	interface
Description	Astrazione di oggetti che è possibile trovare nella scena 3D
Usage	Un oggetto possiede caratteristiche di materiale e trasformazione, modifica a tali caratteristiche vengono notificate tramite signal
Qobject	0
Library Class	0
Include	
Extra Declaration	

#### Operations







- List Classes
- Associations
- View Source
- Create Class
- Update Class
- Delete Class
- Create Method
- Create Attribute

#### Parents

name	Package	type	description	qobject	library
No results found.					

Add as parent  Add

#### Children

name	Package	type	description	qobject	library
<a href="#">Light</a>	<a href="#">3DMob::Model::CBasicEditor::C3DObject</a>	class	Luce presente nella scena 3D	0	0   
<a href="#">Mesh</a>	<a href="#">3DMob::Model::CBasicEditor::C3DObject</a>	class	Mesh poligonale presente nella scena 3D	0	0   

Add as child  Add

#### Attributes










Name	Type	Access	Const	Static	Description
<a href="#">diffusionColor</a>	QColor	private	0	0	Colore di diffusione   
<a href="#">emission</a>	double	private	0	0	Emissione   
<a href="#">emissionColor</a>	QColor	private	0	0	Colore di emissione   

Figura 7: Tracy - screenshot gestione classi



Tracy

Home Logout (Marco)

Home » Packages » CSceneController

### View Package `3DMob::Controller::CSceneController`

Id Package	19
Name	CSceneController
Parent	8
Description	Componente del Controller che si occupa dei Command
Virtual	0

Children

Name	Description
------	-------------

No results found.

[Add child](#)

Operations

- Create Package
- Update Package
- Delete Package
- Manage Package

#### Classes

Name	Package	Type	Description	Qobject	Library Class
<a href="#">CommandTransform</a>	<code>3DMob::Controller::CSceneController</code>	abstract	Base per i comandi di trasformazione	0	0
<a href="#">CommandAddLight</a>	<code>3DMob::Controller::CSceneController</code>	class	Comando di aggiunta luce	0	0
<a href="#">CommandSelectObject</a>	<code>3DMob::Controller::CSceneController</code>	class	Comando per selezionare l'oggetto	0	0
<a href="#">CommandEditMesh</a>	<code>3DMob::Controller::CSceneController</code>	abstract	Comando per modificare la mesh	0	0
<a href="#">CommandEditLight</a>	<code>3DMob::Controller::CSceneController</code>	abstract	Comando per modificare la luce	0	0
<a href="#">CommandScene</a>	<code>3DMob::Controller::CSceneController</code>	abstract	Rappresenta un comando da eseguire sulla scena	0	0
<a href="#">CommandAddElement</a>	<code>3DMob::Controller::CSceneController</code>	abstract	Comando di aggiunta di un elemento	0	0
<a href="#">CommandPosition</a>	<code>3DMob::Controller::CSceneController</code>	class	Comando per cambiare la posizione del modello	0	0
<a href="#">CommandScale</a>	<code>3DMob::Controller::CSceneController</code>	class	Comando per cambiare la dimensione dell'Oggetto	0	0

Figura 8: Tracy - screenshot gestione package

Tracy

Home Logout (Marco)

Home » Tests » Integration

### Manage Integration Tests

View: 20 50 100 infinity tests per page

Operations

Create Test

Id LaTeX	Status	Description	Jenkins	Refers to	
Tl.3DMob	unimplemented	Test di integrazione finale per Model, View e Controller		<a href="#">3DMob</a>	
Tl.CHelpModel	unimplemented	\textbf{Sistema di aiuto}: verifica il corretto funzionamento delle operazioni del sistema di aiuto		<a href="#">3DMob::Model::CHelpModel</a>	
Tl.CSettingsModel	unimplemented	\textbf{Impostazioni}: verifica il corretto funzionamento delle impostazioni dell'applicazione		<a href="#">3DMob::Model::CBasicEditor::CConverter::CSettingsModel</a>	
Tl.CExportModel	unimplemented	\textbf{Esportazione della scena}: verifica che l'operazione di esportazione nei vari formati produca i risultati attesi		<a href="#">3DMob::Model::CBasicEditor::CConverter::CExportModel</a>	
Tl.CLoaderModel	unimplemented	\textbf{Caricamento della scena}: verifica che l'operazione di caricamento della scena avvenga correttamente		<a href="#">3DMob::Model::CBasicEditor::CConverter::CLoaderModel</a>	

Figura 9: Tracy - screenshot gestione test d'integrazione

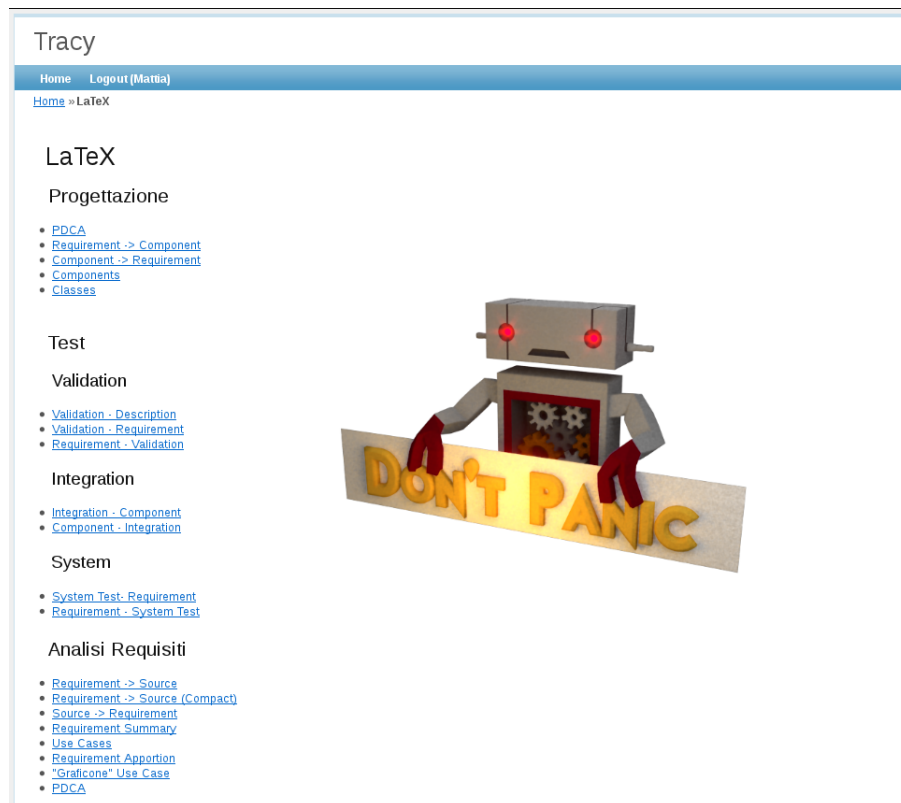


Figura 10: Tracy - screenshot lista comandi di esportazione in  $\text{LaTeX}$

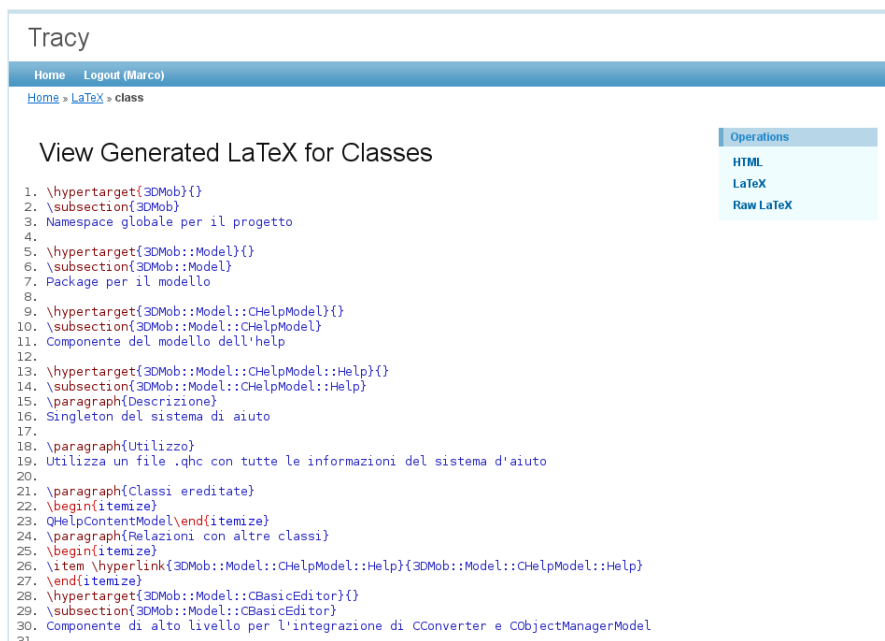


Figura 11: Tracy - screenshot esportazione in  $\text{LaTeX}$  delle classi