

# SEVEN CLICKERS

## You Wish We Click

[7clickersgroup@gmail.com](mailto:7clickersgroup@gmail.com)

## Norme di Progetto

<b>Versione</b>	0.2.0
<b>Stato</b>	Verificato
<b>Uso</b>	Interno
<b>Approvazione</b>	-
<b>Redazione</b>	Mirko Stella Giacomo Mason Gabriele Mantoan Marco Brigo
<b>Verifica</b>	Giacomo Mason Tommaso Allegretti Elena Pandolfo Mirko Stella
<b>Distribuzione</b>	<i>Seven Clickers</i> Prof. Vardanega Tullio Prof. Cardin Riccardo

### Descrizione

Questo documento contiene le norme di progetto seguite dal gruppo Seven Clickers per

il progetto ShowRoom3D

## Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
0.2.2	01-02-23	Mirko Stella	Programmatore	Aggiunta sezione Studio di Fattibilità, modificata sezione Verifica della documentazione
0.2.1	31-01-23	Mirko Stella	Programmatore	Aggiunta sezione Lettera di Presentazione, aggiunta sezione Diario di Bordo, modificata sezione Glossario, spostato sezione Verbale da Convenzioni generali a Tipi di documento
0.2.0	06-01-23	Elena Pandolfo Mirko Stella	Verificatore e Responsabile	Verifica Documento
0.1.3	06-01-23	Marco Brigo	Verificatore	Inserita sezione standard ISO/IEC 9126
0.1.2	10-12-22	Mirko Stella	Verificatore	Modifica a sezione Milestone, Projects Board, Issue Tracking System
0.1.1	03-12-22	Marco Brigo	Analista	Prima stesura sezione Approvazione e Strumenti per la stesura, aggiornate norme su Branching, aggiunta sezione su Pull Request, Milestone, Projects Board, aggiornata sezione Issue Tracking System, tolta sezione Jira <sub>g</sub>
0.1.0	22-11-22	Giacomo Mason Tommaso Allegretti	Verificatori	Verifica documento
0.0.3	16-11-22	Gabriele Mantoan	Verificatore	Modificate norme riguardo verifica <sub>g</sub> e vita delle issue <sub>g</sub> ; aggiunte norme riguardo Jira <sub>g</sub> e indici di glossario
0.0.2	13-11-22	Gabriele Mantoan	Verificatore	Aggiunte norme riguardo Issue tracking system <sub>g</sub> , ruoli e nomi dei branch liberi
0.0.1	06-11-22	Mirko Stella Giacomo Mason	Analista Verificatore	Creazione documento

## Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Scopo del documento . . . . .	4
1.2	Scopo del prodotto . . . . .	4
<b>2</b>	<b>Processi Primari</b>	<b>5</b>
2.1	Fornitura . . . . .	5
2.1.1	Scopo . . . . .	5
2.1.2	Descrizione . . . . .	5
2.1.3	Rapporti con il proponente . . . . .	5
2.2	Sviluppo . . . . .	5
2.2.1	Scopo . . . . .	5
2.2.2	Descrizione . . . . .	5
2.2.3	Analisi dei requisiti . . . . .	5
2.2.3.1	Struttura del documento . . . . .	6
2.2.3.2	Casi d'uso . . . . .	6
2.2.3.3	Requisiti . . . . .	6
2.2.4	Progettazione . . . . .	7
2.2.4.1	Requirments and Tecnology Baseline . . . . .	7
2.2.4.2	Product Baseline . . . . .	7
2.2.5	Codifica . . . . .	7
<b>3</b>	<b>Processi di Supporto</b>	<b>8</b>
3.1	Documentazione . . . . .	8
3.1.1	Convenzioni generali . . . . .	8
3.1.1.1	Versionamento . . . . .	8
3.1.1.2	Struttura Generale . . . . .	8
3.1.1.3	Date . . . . .	9
3.1.1.4	Nomi di persona . . . . .	9
3.1.1.5	Elenchi puntati . . . . .	9
3.1.1.6	Stile del testo . . . . .	9
3.1.2	Tipi di documento . . . . .	9
3.1.2.1	<i>Diario di Bordo</i> . . . . .	9
3.1.2.2	<i>Verbale</i> . . . . .	10
3.1.2.3	<i>Studio di Fattibilità</i> . . . . .	10
3.1.2.4	<i>Lettera di Presentazione</i> . . . . .	11
3.1.2.5	<i>Piano di Progetto</i> . . . . .	13
3.1.2.6	<i>Glossario</i> . . . . .	13
3.1.3	Strumenti per la stesura . . . . .	13
3.1.4	Metriche . . . . .	13
3.1.4.1	Indice di Gulpease . . . . .	14
3.1.5	Approvazione . . . . .	14
3.2	Verifica . . . . .	15
3.2.1	Verifica della documentazione . . . . .	15
3.2.2	Metriche . . . . .	15
3.2.2.1	Code coverage . . . . .	16
<b>4</b>	<b>Processi Organizzativi</b>	<b>17</b>
4.1	Gestione Organizzativa . . . . .	17
4.1.1	Ruoli . . . . .	17
4.1.1.1	Responsabile . . . . .	17
4.1.1.2	Analista . . . . .	17
4.1.1.3	Amministratore . . . . .	17

4.1.1.4	Progettista . . . . .	17
4.1.1.5	Programmatore . . . . .	18
4.1.1.6	Verificatore . . . . .	18
4.1.2	Metriche . . . . .	18
4.1.2.1	Rischi non previsti . . . . .	18
4.2	Gestione Infrastrutture . . . . .	19
4.2.1	GitHub <sub>g</sub> . . . . .	19
4.2.1.1	Repository <sub>g</sub> . . . . .	19
4.2.1.2	Branching . . . . .	19
4.2.1.3	Commits . . . . .	19
4.2.1.4	Pull Requests . . . . .	20
4.2.1.5	Milestone <sub>g</sub> . . . . .	20
4.2.1.6	Projects Board <sub>g</sub> . . . . .	21
4.2.1.7	Issue Tracking System <sub>g</sub> . . . . .	22
4.2.2	Metriche . . . . .	22
<b>5</b>	<b>Standard di qualità ISO/IEC 9126</b>	<b>23</b>
5.1	Qualità esterne . . . . .	23
5.2	Qualità interne . . . . .	23
5.3	Modello di qualità . . . . .	23
5.3.1	Funzionalità . . . . .	23
5.3.2	Affidabilità . . . . .	24
5.3.3	Efficienza . . . . .	24
5.3.4	Usabilità . . . . .	24
5.3.5	Manutenibilità . . . . .	24
5.3.6	Portabilità . . . . .	25
5.4	Qualità in uso . . . . .	25
<b>6</b>	<b>Standard di qualità ISO/IEC 12207:1995</b>	<b>26</b>
6.1	Processi primari . . . . .	26
6.1.1	Acquisizione . . . . .	26
6.1.2	Fornitura . . . . .	26
6.1.3	Sviluppo . . . . .	27
6.1.4	Gestione operativa . . . . .	27
6.1.5	Manutenzione . . . . .	27
6.2	Processi di supporto . . . . .	28
6.2.1	Documentazione . . . . .	28
6.2.2	Gestione della configurazione . . . . .	28
6.2.3	Accertamento della qualità . . . . .	28
6.2.4	Verifica . . . . .	28
6.2.5	Validazione . . . . .	29
6.2.6	Revisione congiunta . . . . .	29
6.2.7	Audit . . . . .	29
6.2.8	Risoluzione dei problemi . . . . .	29
6.3	Processi organizzativi . . . . .	29
6.3.1	Gestione organizzativa . . . . .	29
6.3.2	Gestione delle infrastrutture . . . . .	30
6.3.3	Miglioramento del processo . . . . .	30
6.3.4	Formazione del personale . . . . .	30

# 1 Introduzione

## 1.1 Scopo del documento

Lo scopo del documento è quello di stabilire le regole che ogni componente del gruppo SevenClickers deve rispettare per mantenere un ambiente di lavoro che mira a massimizzare l'economicità dei processi durante il ciclo di vita del prodotto ShowRoom3D.

Le norme verranno inserite in modo incrementale per regolamentare le attività di progetto imminenti rimandando quelle meno urgenti a quando se ne presenterà la necessità.

Inoltre tali norme potranno subire modifiche nel tempo in modo da garantire un miglioramento continuo della qualità del lavoro svolto. Il responsabile di progetto ha il compito di comunicare l'aggiunta di una nuova norma o la modifica di una già esistente a tutti i componenti del gruppo e di assicurarsi che siano comprese a pieno.

## 1.2 Scopo del prodotto

Il prodotto in questione nasce dalla necessità dell'azienda SanMarco Informatica di fornire una soluzione agli sprechi derivati dall'adozione di uno ShowRoom tradizionale proponendo uno ShowRoom3D che sia ugualmente o ancora più immersivo.

## 2 Processi Primari

### 2.1 Fornitura

#### 2.1.1 Scopo

Lo scopo del processo di fornitura è quello di stabilire quali sono le attività, le risorse e i compiti necessari allo svolgimento del progetto.

#### 2.1.2 Descrizione

Nella seguente sezione sono riportate le norme che il gruppo si impegna a rispettare al fine di creare e mantenere un rapporto attivo e vantaggioso con il proponente SanMarco Informatica.

#### 2.1.3 Rapporti con il proponente

Il gruppo si manterrà in contatto con il proponente regolarmente, organizzando incontri su Google Meet o comunicando in modo asincrono con email o Google Chat. Di seguito un elenco dei principali temi che possono essere motivo di confronto con il proponente:

- Chiarimento di eventuali dubbi o incomprensioni;
- Riscontro sul lavoro svolto;
- Definizione dei requisiti obbligatori e opzionali;
- Opinioni riguardanti tecnologie da utilizzare;
- Proposte da parte del gruppo di soluzioni alternative.

### 2.2 Sviluppo

#### 2.2.1 Scopo

Lo scopo del processo di sviluppo è quello di stabilire le attività necessarie allo sviluppo del prodotto software.

#### 2.2.2 Descrizione

Nelle seguenti sezioni verranno descritte più nel dettaglio le principali attività, relative al processo di sviluppo:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

#### 2.2.3 Analisi dei requisiti

L'analisi dei requisiti è un'attività fondamentale che precede lo sviluppo. Si concretizza nel documento *Analisi dei Requisiti*, redatto dagli analisti. Lo scopo di quest'attività è:

- Definire le funzionalità che il prodotto andrà ad offrire;
- Porre le basi per la fase di progettazione del software;
- Fare una stima della mole di lavoro;
- Facilitare la verifica.

### 2.2.3.1 Struttura del documento

Il documento è strutturato nel modo seguente:

- **Introduzione:** contiene una breve descrizione del documento e specifica gli attori dei casi d'uso;
- **Casi d'uso:** vengono specificati i casi d'uso individuati;
- **Requisiti:** sono classificati i requisiti che il prodotto finale dovrà soddisfare.

### 2.2.3.2 Casi d'uso

I casi d'uso sono identificati seguendo la seguente struttura:

UC [Numero caso d'uso].[Numero sottocaso d'uso]-[Titolo caso d'uso]

Vengono poi indicati:

- **Diagramma UML:** viene mostrato il diagramma solo per i casi più complessi, in cui è ritenuto necessario per una maggiore comprensione. Per i casi d'uso facoltativi il diagramma è colorato di azzurro (non standard UML) per una maggiore differenziazione;
- **Attore primario:** utente esterno al sistema che svolge il caso d'uso;
- **Descrizione:** breve descrizione del caso d'uso;
- **Precondizioni:** indica la condizione del sistema prima del verificarsi del caso d'uso;
- **Postcondizioni:** indica la condizione del sistema dopo che si è verificato il caso d'uso;
- **Scenario principale:** descrive l'interazione tra l'attore primario e il sistema;
- **Estensioni:** casi d'uso alternativi che possono verificarsi al posto di quello a cui sono collegati.

### 2.2.3.3 Requisiti

I requisiti sono classificati secondo la seguente struttura:

R.[Tipologia][Numero seriale]

con:

- **Tipologia:**
  - **F:** funzionale;
  - **Q:** qualitativo;
  - **D:** di dominio;
  - **P:** prestazionale.
- **Descrizione:** breve descrizione del requisito;
- **Classificazione:** un requisito può essere facoltativo o obbligatorio;
- **Fonti:** possono essere:
  - Casi d'uso;
  - Capitolato;
  - Decisione interna.



## 2.2.4 Progettazione

: L'attività di progettazione è assegnata ai progettisti. Seguendo quanto indicato nell'*Analisi dei Requisiti*, l'obiettivo è quello di progettare l'architettura del sistema, prima con la creazione di un Proof of Concept per la Requirments and Tecnology Baseline e poi andando nel dettaglio per la Product Baseline.

### 2.2.4.1 Requirments and Tecnology Baseline

In questa fase vengono fissati i requisiti che il gruppo si impegna a soddisfare, in accordo con il proponente; si studiano le tecnologie, i framework e le librerie utili alla realizzazione del prodotto finale e si crea di conseguenza un Proof of Concept. I materiali da mostrare sono:

- Tecnologie, framework, librerie utilizzate, motivando la scelta;
- Proof of Concept;
- *Analisi dei Requisiti*.

La documentazione da mostrare:

- *Piano di Progetto*;
- *Piano di Qualifica*;
- *Norme di Progetto*;
- *Verbali* interni ed esterni.

### 2.2.4.2 Product Baseline

In questa fase viene definito il design definitivo del prodotto, coerentemente con quello che è stato mostrato nella Requirments and Tecnology Baseline.

I materiali da mostrare sono:

## 2.2.5 Codifica

## 3 Processi di Supporto

### 3.1 Documentazione

#### 3.1.1 Convenzioni generali

Le convenzioni di seguito riportate vengono applicate a tutti i documenti. Esse rendono i documenti stilati omogenei tra loro contribuendo a rendere il progetto professionale.

##### 3.1.1.1 Versionamento

Il numero di versione permette di capire lo stato in cui si trova un documento. Un documento può trovarsi nei seguenti stati:

- **Approvato:** il documento è verificato ed approvato dal Responsabile;
- **Verificato:** il documento risulta verificato ma non ancora visionato dal Responsabile;
- **In Sviluppo:** sono presenti delle modifiche che non sono state verificate.

Il numero di versione ha il formato **X.Y.Z** dove:

- **X:** indica una versione approvata dal Responsabile, la numerazione parte da 0 e la prima versione approvata è la 1.0.0;
- **Y:** indica una versione verificata dal Verificatore, la numerazione inizia da 0 e si azzerà ad ogni incremento di X. La prima versione verificata è la 0.1.0;
- **Z:** indica una versione in fase di modifica da parte dei redattori che ne incrementano il numero ad ogni modifica, la numerazione parte da 1 e si azzerà ad ogni incremento di X o Y. La prima versione modificata è la 0.0.1.

##### 3.1.1.2 Struttura Generale

Ogni documento deve presentare le seguenti sezioni nell'ordine in cui vengono presentate:

- **Intestazione:** contiene:
  - Logo compreso di motto;
  - Indirizzo email di gruppo;
  - Titolo;
  - Tabella contenente le informazioni generali:
    - \* Versione;
    - \* Stato;
    - \* Uso;
    - \* Approvazione: indica il responsabile di progetto che ha approvato il documento;
    - \* Redazione: elenco dei collaboratori che hanno partecipato alla stesura del documento;
    - \* Verifica: elenco dei verificatori che hanno verificato il documento;
    - \* Distribuzione: elenco delle persone o organizzazioni a cui è destinato il documento.
  - Breve descrizione del documento .
- **Registro delle modifiche:** tabella che identifica ogni versione del documento indicandone:
  - Versione;
  - Data;
  - Autore;

- Ruolo;
- Descrizione.

- **Indice:** elenco ordinato dei titoli dei capitoli, ovvero delle varie parti di cui si compone il documento;
- **Contenuto:** varia a seconda del tipo di documento.

### 3.1.1.3 Date

Le date devono rispettare il seguente formato: **dd-mm-yyyy** All'interno delle tabelle il formato deve essere il seguente: **dd-mm-yy**

### 3.1.1.4 Nomi di persona

All'interno dei documenti i nomi di persona rispetteranno l'ordine nome seguito dal cognome della persona menzionata.

### 3.1.1.5 Elenchi puntati

Gli elenchi puntati devono rispettare le seguenti regole:

- Ogni elemento dell'elenco deve iniziare con la lettera maiuscola;
- Ogni elemento dell'elenco deve terminare con ";" ad eccezione dell'ultimo elemento che deve terminare con ".";
- Dopo i due punti la frase deve iniziare con la lettera minuscola.

### 3.1.1.6 Stile del testo

- **Grassetto:** stile utilizzato per i titoli delle sezioni e per i primi termini degli elenchi puntati;
- **Corsivo:** viene utilizzato per citare il nome dei documenti, ad esempio *Piano di Progetto*.

## 3.1.2 Tipi di documento

### 3.1.2.1 Diario di Bordo

Il *Diario di Bordo* è un documento ad uso esterno e permette di interagire con il Professore (committente) in modo da aggiornarlo sullo stato di avanzamento del progetto settimanalmente ed è usato anche per chiedere chiarimenti sui temi in cui riscontriamo dubbi o domande. Fino al termine delle lezioni il *Diario di Bordo* veniva redatto settimanalmente dal Responsabile di Progetto che presentava in classe prima dello svolgimento della lezione. Dal termine delle lezioni il *Diario di Bordo* viene redatto settimanalmente dal Responsabile di Progetto e caricato in una cartella denominata diari\_di\_bordo presente nel drive di gruppo.

La cartella diari\_di\_bordo si trova al link:

[https://drive.google.com/drive/folders/1a0kAZuOSsDEp\\_AaQub6OQJ4XQyZRCN](https://drive.google.com/drive/folders/1a0kAZuOSsDEp_AaQub6OQJ4XQyZRCN)

È stato fornito l'accesso in lettura alla cartella al Professore che viene notificato tramite mail (deve contenere il link al *Diario di Bordo* di interesse) al caricamento di un *Diario di Bordo*. Il formato dei diari di bordo condivisi nella cartella è .pdf. Inoltre tutti i diari di bordo una volta redatti vengono inviati anche al canale discord "diari-di-bordo-file" in modo da avere una duplice copia a fronte di qualsiasi imprevisto dato che i diari di bordo non vengono inseriti all'interno del repo di gruppo.

### Struttura *Diario di Bordo*

La struttura del *Diario di Bordo* non rispetta quella indicata nella sezione 3.1.1.2 relativa alla struttura generale della documentazione. Sono state fissate delle regole per la stesura dei diari di bordo per

fornire delle presentazioni il più possibile simili tra loro senza troppi vincoli superflui per tali documenti.

Il **font** utilizzato per la stesura del documento é Helvetica Neue con dimensione 22pt.

Il *Diario di Bordo* si divide in 4 slides che possono essere raggruppate in 3 se l'elenco degli obiettivi raggiunti e quelli futuri ci stanno in una singola slide. Le slides che compongono il *Diario di Bordo* sono:

1. Intestazione
2. Obiettivi raggiunti
3. Obiettivi futuri
4. Domande

Descrizione slides:

- La slide di **Intestazione** presenta nella parte centrale il logo del gruppo compreso di slogan, in basso centrale l'anno accademico e in alto a destra la data del *Diario di Bordo* (data a cui fa riferimento non stesura).
- La slide **Obiettivi raggiunti** presenta in alto centrale il titolo "Obiettivi raggiunti", a sinistra un elenco puntato con gli obiettivi raggiunti della settimana ed in alto a destra il logo del gruppo compreso di slogan.
- La slide **Obiettivi futuri** presenta in alto centrale il titolo "Obiettivi futuri", a sinistra un elenco puntato con gli obiettivi raggiunti della settimana ed in alto a destra il logo del gruppo compreso di slogan.
- La slide **Domande** presenta in alto centrale il titolo "Domande", a sinistra un elenco puntato con le domande da porre al Professore ed in alto a destra il logo del gruppo compreso di slogan.

### 3.1.2.2 Verbale

Rispettano tutta la struttura generale. In aggiunta presentano:

- **Informazioni Generali** contengono:
  - Luogo;
  - Data;
  - Ora;
  - Partecipanti.
- **Tabella tracciamento temi affrontati:** tabella che riassume i punti salienti della riunione indicandone:
  - Codice: ha il formato **VX Y.Z** dove X indica la tipologia di *Verbale*, Y indica il numero di *Verbale* (incrementale rispetto agli altri verbali); e Z indica il numero dell'argomento trattato (incrementale rispetto agli altri argomenti del *Verbale*);
  - Descrizione: breve descrizione di uno specifico argomento trattato.

### 3.1.2.3 Studio di Fattibilità

Lo *Studio di Fattibilità* ha lo scopo di valutare i pro ed i contro dei capitolati proposti in modo da scegliere il più vantaggioso al quale candidarsi. La scelta del capitolato viene fatta sulla base di diverse considerazioni. Un capitolato potrebbe essere vantaggioso per alcuni aspetti ma svantaggioso per altri. Inoltre potrebbero presentarsi capitolati irrealizzabili per mancanza di risorse o per scarse conoscenze del contesto applicativo. Oltre a risorse in termini di budget, tempo e personale un capitolato potrebbe

venire scartato per scarso guadagno netto al termine del lavoro commissionato (non é il nostro caso ma andrebbe tenuto in considerazione in ambito lavorativo). Avendo la possibilità di scegliere tra capitolati (fatto che capita raramente in un reale ambito lavorativo) di pari interesse siamo portati a scegliere il contesto applicativo che ci entusiasma maggiormente. Questo aspetto per lo più psicologico potrebbe avere un impatto positivo sulla produttività del gruppo. Il documento tiene in considerazione tutti i capitolati proposti per non scartare un capitolato per le sensazioni soggettive dei componenti del gruppo. Non tenendo in considerazione ogni capitolato, infatti, si potrebbe scartare un capitolato di forte interesse senza rendersene conto.

#### **3.1.2.4 Lettera di Presentazione**

Una *Lettera di Presentazione* serve a manifestare la volontà da parte del gruppo di prendere un impegno con il committente. L'impegno può essere la candidatura per un capitolato di interesse o per una revisione di avanzamento. Dopo aver ricevuto una *Lettera di Presentazione* sta al committente la decisione di accettare o rifiutare l'impegno che ne consegue.

#### **Struttura Lettera di Presentazione**

Una *Lettera di Presentazione* é formata dai seguenti campi:

- Header
- Mittente e data
- Destinatari
- Contenuto
- Riferimenti a documenti
- Conclusioni e saluti
- Nome,cognome,firma responsabile

**Header** contiene il logo del gruppo (con slogan) a sinistra ed il logo dell'Università di Padova a destra.

**Mittente e data** contiene il nome,email del gruppo seguito dal nome del corso e dalla data di invio della lettera al destinatario.

**Destinatari** contiene i destinatari della lettera compresi di titoli e luogo della loro sede.

**Contenuto** contiene il contenuto della lettera scritto in modo formale dichiarando lo scopo della lettera.

**Riferimenti a documenti** contiene un elenco puntato di tutti i documenti (e loro versioni) a cui si vuole sottoporre l'attenzione dei destinatari.

**Conclusioni** contiene le considerazioni finali seguite dai saluti rivolti al destinatario.

**Nome,cognome,firma responsabile** contiene il nome,cognome,titolo e firma digitale del responsabile.

Di seguito viene riportata un'immagine che illustra visivamente i campi appena descritti...











HEADER	
<div><div><b>SEVEN CLICKERS</b>  You Wish We Click</div><div></div></div>	
<div>Gruppo <i>Seven Clickers</i> E-mail: <a href="mailto:7clickersgroup@gmail.com">7clickersgroup@gmail.com</a> Corso di Ingegneria del Software AA 2022/2023 28 Ottobre 2022</div>	MITTENTE E DATA
DESTINATARI	
<div>Prof.  Prof.  Università degli Studi di Padova Dipartimento di Matematica Via Trieste, 63 35121 Padova</div>	
CONTENUTO	
<div>Egregio Prof.  Egregio Prof.   Con la presente, il gruppo <i>Seven Clickers</i> intende comunicarVi ufficialmente l'impegno alla realizzazione del prodotto da Voi commissionato, denominato:  <b><i>ShowRoom 3D</i></b>  proposto dall'azienda </div>	
RIFERIMENTI A DOCUMENTI	
<div>Si allegano i documenti redatti fino ad ora:<ul style="list-style-type: none"><li>• Piano di progetto v1.1.0</li><li>• Studio di fattibilità v1.0.0</li><li>• Verbale interno 19-10-2022</li><li>• Verbale interno 25-10-2022</li><li>• Verbale interno 26-10-2022</li><li>• Verbale esterno 25-10-2022</li><li>• Verbale esterno 26-10-2022</li></ul></div>	
<div>Come descritto nel <i>Piano di progetto v1.1.0</i>, il gruppo <i>Seven Clickers</i> si impegna a consegnare il prodotto entro <b>14 Aprile 2022</b> preventivando un costo complessivo di  Cordiali saluti,</div>	
CONCLUSIONI E SALUTI	
<div> <i>Responsabile di Progetto</i> </div>	
NOME, COGNOME, FIRMA RESPONSABILE	

Figura 3.1.1: Esempio di *Lettera di Presentazione*

### 3.1.2.5 Piano di Progetto

Il Piano di progetto ha lo scopo di pianificare il lavoro del gruppo e di stimare il consumo di risorse durante il periodo pianificato. Alla fine di ogni pianificazione viene fatto un consultivo che ci permette di valutare se le aspettative sono state mantenute e quindi di rendersi conto della quantità di risorse ancora disponibili prima di intraprendere le prossime pianificazioni.

### 3.1.2.6 Glossario

All'interno della documentazione si possono trovare dei termini che possono risultare ambigui a seconda del contesto, o non conosciuti dagli utilizzatori.

Per ovviare ad incomprensioni si è deciso di stilare un elenco di termini di interesse accompagnati da una descrizione del loro significato.

I termini presenti all'interno dei documenti che necessitano di una descrizione vengono indicati con il pedice 'g' come nell'esempio seguente: termine. È quindi possibile consultare il *Glossario* per reperire tale descrizione.

Ogni componente del gruppo all'inserimento di un termine ritenuto ambiguo deve preoccuparsi di aggiornare il *Glossario*.

Per aggiornare il *Glossario* si devono inserire i nuovi termini nel file .tex nella cartella corrispondente all'iniziale del termine situata al percorso *latex/esterni/doc\_esterna/Glossario/res/sections/alphabet/*. È stato creato uno script che scansiona un documento di interesse per inserire in automatico il pedice sui termini contenuti nel *Glossario*.

Il componente del gruppo che inserisce all'interno del *Glossario* un nuovo termine deve aggiungere nel file .tex il segnaposto %parola% dopo la subsection (senza spazi) che racchiude il termine per permetterne il riconoscimento da parte dello script.

Il *Glossario* ordina i termini in ordine alfabetico in modo da permetterne una facile e veloce ricerca.

### 3.1.3 Strumenti per la stesura

- **LaTeX<sub>g</sub>**: è un linguaggio di marcatura per la preparazione di testi, basato sul programma di composizione tipografica TEX.

Nel branch documentation si possono trovare i file .pdf prodotti e la cartella "latex<sub>g</sub>". La cartella latex<sub>g</sub> contiene tre cartelle interne:

- **esterni e interni**: contengono file .tex di documentazione esterna ed interna come ad esempio i *Verbalì* o altra documentazione esterna/interna:
  - \* La cartella config contiene i file .tex con le parti fisse dei documenti (intestazione, registro delle modifiche, tracciamento dei temi affrontati) che vengono modificati con i dati del documento specifico;
  - \* La cartella res/sections contiene i file .tex con il contenuto vero e proprio (sezioni del documento) che viene redatto in maniera libera dal redattore;
  - \* Un file col nome del documento pdf con estensione .tex che viene compilato per produrre il file pdf.
- **template**: contiene file .tex di base utilizzati secondo necessità per comporre i documenti:
  - \* changelox.tex è il file di template che serve per scrivere il registro delle modifiche;
  - \* package.tex è il file che contiene tutti gli usepackage;
  - \* titlepage.tex è il file di template che contiene la configurazione della pagina iniziale di ogni documento;
  - \* tracking.tex è il file di template che contiene il tracciamento dei temi affrontati nel documento.

### 3.1.4 Metriche

Per perseguire la qualità sulla documentazione prodotta si è deciso di adottare le seguenti metriche:

### 3.1.4.1 Indice di Gulpease

Si tratta dell'indice di leggibilità di un testo tarato sulla lingua italiana. I risultati sono compresi tra 0 e 100, dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. Ai seguenti valori si associano i seguenti significati:

- inferiore a 80 sono difficili da leggere per chi ha la licenza elementare
- inferiore a 60 sono difficili da leggere per chi ha la licenza media
- inferiore a 40 sono difficili da leggere per chi ha un diploma superiore

Viene adottata la seguente formula per calcolarlo:

$$89 + \frac{300 * (\text{numero delle frasi}) - 10 * (\text{numero delle lettere})}{\text{numero delle parole}} \quad (1)$$

Questi sono i valori da noi ritenuti opportuni:

*Valore minimo:*

$$\geq 50$$

*Valore ottimo:*

$$\geq 80$$

### 3.1.5 Approvazione

L'approvazione viene svolta dal responsabile.

L'approvazione consiste nell'aprire una Pull Request di approvazione. Nel caso in cui il responsabile riscontrasse ulteriori problematiche, segnalerà ai Verificatori le eventuali modifiche da apportare. I Verificatori apporteranno tali modifiche prima di chiudere la Pull Request di approvazione e quindi di integrare i cambiamenti nel branch documentation.

Per i *Verballi* si effettua questa pratica non appena il file prodotto viene verificato e quindi passa tutte le eventuali reviews create dai Verificatori mentre per tutti gli altri documenti prima di una consegna.

Ad esito positivo di approvazione, il responsabile creerà una nuova riga e compilerà i campi nelle colonne corrispondenti del registro delle modifiche inserendo: l'ultima versione secondo le norme di versionamento, la data, il proprio nome, il suo ruolo e la voce "Approvazione" nell'ultima colonna.



## 3.2 Verifica

### 3.2.1 Verifica della documentazione

La verifica viene svolta da due verificatori prima del merge con il branch `documentation` all'apertura di una pull request. Consiste nell'esaminare i file prodotti da chi ne ha fatto la stesura e segnalarne la presenza di errori nei concetti esposti.

Un verificatore dovrà verificare il documento a partire dalle modifiche fatte dopo l'ultima versione verificata. Le modifiche da verificare quindi possono essere dedotte dal registro dei cambiamenti presente in ogni documento oltre che dallo storico fornito dal sistema di versionamento utilizzato. Una volta controllato il documento, il primo verificatore segnalerà eventuali errori e successivamente dovrà spuntare come approvata la Pull Request nella sezione dedicata su GitHub<sub>g</sub>.

A questo punto se un secondo verificatore noterà la necessità di qualche altro cambiamento da apportare, chi dovrà apportare le modifiche farà una pull in locale per allineare il proprio branch con quello in remoto e continuare con il proprio lavoro.

Dopo il push delle modifiche se i file risultano corretti anche dal secondo verificatore esso aggiungerà i nomi dei verificatori all'intestazione modificando il file `titlepage.input.tex` e creerà la riga nel registro delle modifiche, nel file `changelog.input.tex`, inserendo la nuova versione secondo le regole di versionamento e scrivendo nella colonna Autore sia il suo nome, che quello del primo verificatore, e in descrizione "Verifica". Dopo aver eseguito un commit sul ramo da integrare approva la Pull Request e conferma il merge secondo le *Norme di Progetto* descritte nella sezione dedicata. La verifica di un documento prevede di seguire la seguente checklist da parte di ogni verificatore:

1. Lettura esplorativa del testo;
2. Seconda lettura del testo per capirne meglio i concetti esposti;
3. Segnalare i concetti poco chiari o errati;
4. Valutare se il testo è organizzato in maniera opportuna ed eventualmente suggerire un'alternativa migliore;
5. Valutare se ci sono parti mancanti ed eventualmente segnalarne la mancanza;
6. Segnalare errori grammaticali;
7. Segnalare le parti che non rispettano le *Norme di Progetto* riguardanti la stesura del documento;
8. Segnalare i termini che sono contenuti anche nel *Glossario* a cui non sono stati inseriti i pedici "g";
9. Segnalare i nomi dei documenti che non sono stati scritti seguendo la convenzione;
10. Segnalare le parti del registro delle modifiche mancanti o che non sono state compilate correttamente seguendo le *Norme di Progetto*;
11. Segnalare errori nell'intestazione del documento;
12. In caso non ci sia nulla da segnalare:
  - (a) se le modifiche da accettare sono state accettate da tutti gli altri verificatori allora compilare il registro delle modifiche e l'intestazione del documento seguendo le *Norme di Progetto* e fare il merge con il branch di destinazione;
  - (b) se altri verificatori devono ancora verificare il documento limitarsi a dare il proprio consenso alle modifiche nella sezione dedicata su GitHub<sub>g</sub>.

### 3.2.2 Metriche

Per mantenere la qualità nella verifica è stata scelta la seguente metrica.

### 3.2.2.1 Code coverage

Viene definita come la percentuale di linee di codice del progetto che sono state eseguite dai test dopo un'esecuzione.

Questi sono i valori da noi ritenuti opportuni:

*Valore minimo:*

$\geq 70\%$

*Valore ottimo:*

$\geq 100\%$

## 4 Processi Organizzativi

### 4.1 Gestione Organizzativa

#### 4.1.1 Ruoli

I componenti del gruppo si suddivideranno nei seguenti ruoli per periodi di circa 2-3 settimane (dipendentemente dalle esigenze del periodo) e al termine del periodo i ruoli verranno risuddivisi. Visto che nelle varie fasi di sviluppo del progetto le attività da svolgere variano, non sempre sarà necessario coprire tutti i ruoli.

Inoltre sarà necessario tenere traccia delle ore che ogni componente dedica al progetto ed il ruolo associato a quelle ore, in modo da andare a rispettare la tabella degli impegni individuali.

I ruoli e le loro competenze sono i seguenti:

##### 4.1.1.1 Responsabile

Deve avere la visione d'insieme del progetto e coordinare i membri, inoltre si occupa di rappresentare il gruppo con le interazioni esterne (proponente, committente ecc...). Le sue competenze specifiche sono:

- Ad ogni iterazione<sub>g</sub> c'è un solo responsabile;
- Presentare il diario di bordo in aula;
- Redarre l'ordine del giorno prima di ogni meeting interno del gruppo;
- Suddivide le attività del gruppo in singole issue (ma non le assegna ai membri del gruppo);
- In fase di release<sub>g</sub> si occupa di approvare<sub>g</sub> tutti i documenti che necessitano approvazione.

##### 4.1.1.2 Analista

Si occupa di trasformare i bisogni del proponente nelle aspettative che il gruppo deve soddisfare per sviluppare un prodotto professionale. Le sue competenze specifiche sono:

- Interrogare il proponente riguardo allo scopo del prodotto e le funzionalità che deve avere;
- Studiare le risposte del proponente per identificare i requisiti<sub>g</sub> e redarre l' *Analisi dei Requisiti*.

##### 4.1.1.3 Amministratore

Si occupa del funzionamento, mantenimento e sviluppo degli strumenti tecnologici usati dal gruppo. Le sue competenze specifiche sono:

- Ad ogni iterazione<sub>g</sub> basta un solo amministratore;
- Gestione delle segnalazioni e problemi dei membri del gruppo riguardanti problemi e malfunzionamenti con gli strumenti tecnologici;
- Valuta l'utilizzo di nuove tecnologie e ne fa uno studio preliminare per poter presentare al gruppo i pro e i contro del suo utilizzo.

##### 4.1.1.4 Progettista

Si occupa di scegliere la modalità migliore per soddisfare le aspettative del committente che gli analisti hanno ricavato dall'analisi dei requisiti<sub>g</sub>. Le sue competenze specifiche sono:

- Scegliere eventuali pattern architetturali da implementare;
- Sviluppare lo schema UML<sub>g</sub> delle classi<sub>g</sub>.

#### 4.1.1.5 Programmatore

Si occupa di implementare le scelte e i modelli fatti dal progettista. Le sue competenze specifiche sono:

- Scrivere il codice atto a implementare lo schema delle classi;
- Scrivere eventuali test;
- Scrivere la documentazione per la comprensione del codice che scrive.

#### 4.1.1.6 Verificatore

Si occupa di controllare che ogni file che viene caricato in un branch protetto della repository<sub>g</sub> sia conforme alle norme di progetto. Le sue competenze specifiche sono:

- Controllare i file modificati o aggiunti durante una pull request tra un ramo non protetto e un ramo protetto siano conformi alle norme di progetto e cercano errori di altra natura (ortografici, sintattici, logici, build ecc...).

### 4.1.2 Metriche

Abbiamo considerato questa metrica come rilevante per mantenere la qualità.

#### 4.1.2.1 Rischi non previsti

Il valore è identificato con un numero intero e indica il numero di rischi non preventivati. Questi sono i valori da noi ritenuti opportuni: *Valore minimo*: 10

*Valore ottimo*: 5

## 4.2 Gestione Infrastrutture

### 4.2.1 GitHub<sub>g</sub>

Servizio di hosting per progetti software che implementa uno strumento di controllo versione distribuito Git<sub>g</sub>. Oltre alla copia in remoto del repository<sub>g</sub> di progetto ogni componente del gruppo ha una propria copia in locale.

Per ottenere una copia del repository<sub>g</sub> ogni componente ha scaricato lo strumento Git<sub>g</sub> ed eseguendo il comando 'git clone' da git<sub>g</sub> bash viene creata una cartella collegata alla repository<sub>g</sub> di progetto.

Non sono state imposte modalità specifiche sull'interazione con il repository<sub>g</sub> remoto in modo da non sconvolgere le abitudini di lavoro di ciascun componente.

I componenti del gruppo abituati ad interagire con GitHub<sub>g</sub> da interfaccia grafica possono continuare a farne uso.

#### 4.2.1.1 Repository<sub>g</sub>

Il repository<sub>g</sub> si può trovare all'indirizzo <https://github.com/7clickers/ShowRoom3D> ed è pubblico. I collaboratori sono i componenti del gruppo SevenClickers che utilizzano il proprio account GitHub<sub>g</sub> personale per collaborare al progetto.

#### 4.2.1.2 Branching

**Branches protetti:**

- **main:** contiene le versioni di release<sub>g</sub> del software;
- **documentation:** contiene i template latex<sub>g</sub> e rispettivi pdf della documentazione.

*documentation:* i documenti presenti in documentation sono stati approvati dal responsabile o almeno verificati dai verificatori.

Per integrare delle modifiche da un branch protetto ad uno libero si utilizza un branch d'appoggio creato in locale partendo dall'ultimo commit di documentation e facendone il merge con il branch che necessita delle integrazioni. In seguito il branch di appoggio verrà eliminato.

**Branches liberi:** vengono utilizzati per creare nuove funzionalità e gli sviluppatori possono effettuare i commit senza l'approvazione degli altri componenti del gruppo in quanto ciascun componente sviluppa su un solo branch alla volta salvo casi eccezionali. Un branch<sub>g</sub> libero avrà il nome del documento che si sviluppa su quel branch<sub>g</sub>, oppure della feature<sub>g</sub> che va ad implementare.

Non appena i file nel branch sono stati verificati ed il merge è stato fatto, il branch libero verrà eliminato. I branch di approvazione saranno chiamati con la sintassi `appr_nomefile1_nomefile2_nomefile3....` a seconda dei file che verranno approvati durante il ciclo di vita del branch.

#### 4.2.1.3 Commits

È preferibile che ogni commit abbia una singola responsabilità per cambiamento.

I commits non possono essere effettuati direttamente sui branch protetti ma per integrare delle aggiunte o modifiche sarà necessario aprire una Pull Request. All'approvazione di una Pull Request tutti i commit relativi al merge verranno raggruppati in un unico commit che rispetti la struttura sintattica descritta in seguito.

I commit dovranno essere accompagnati da una descrizione solo se ritenuta indispensabile alla comprensione del commit stesso.

I messaggi di commit sui **BRANCH PROTETTI** dovranno seguire la seguente struttura sintattica:

`<label><#n.issue><testo>`

dove:

- **label** può assumere i seguenti valori:
  - **feat**: indica che è stata implementata una nuova funzionalità;
  - **fix**: indica che è stato risolto un bug;
  - **update**: indica che è stata apportata una modifica che non sia fix o feat;
  - **test**: qualsiasi cosa legata ai test;
  - **docs**: qualsiasi cosa legata alla documentazione.
- **n\_issue**: indica il numero della issue a cui fa riferimento il commit (se non fa riferimento a nessuna issue viene omesso);
- **testo**: indica con quale branch è stato effettuato il merge e deve rispettare la forma: merge from <nome branch da integrare> to <nome branch corrente>;
- **descrizione**: se aggiunta ad un commit deve rispondere alle domande WHAT?, WHY?, HOW? ovvero cosa è cambiato, perché sono stati fatti i cambiamenti, in che modo sono stati fatti i cambiamenti.

I messaggi di commit sui **BRANCH LIBERI** dovranno seguire la struttura sintattica dei branch protetti ad eccezione del testo. Il testo dei commit sui branch liberi non è soggetto a restrizioni particolari a patto che indichi in maniera intuitiva i cambiamenti fatti in modo che possano essere compresi anche dagli altri collaboratori.

#### 4.2.1.4 Pull Requests

Per effettuare un merge su un branch protetto si deve aprire da GitHub<sub>g</sub> una Pull Request. La Pull Request permette di verificare il lavoro svolto prima di integrarlo con il branch desiderato.

Alla creazione di una Pull Request bisogna associare:

- I verificatori in carica hanno il compito di trovare eventuali errori o mancanze e fornire un feedback riguardante il contenuto direttamente su GitHub<sub>g</sub> richiedendo una review con un review comment sulla parte specifica da revisionare o con un commento generico.  
Non sarà possibile effettuare il merge finché tutti i commenti di revisione non saranno stati risolti e la Pull Request approvata da due verificatori;
- L'issue associata nell'opzione "Development" che verrà chiusa alla risoluzione della Pull Request;
- La Projects Board di cui fa parte;
- Gli assegnatari che hanno il compito di apportare le modifiche necessarie in fase di verifica;
- Le labels associate.

Per i commit relativi alle Pull Requests seguire le regole descritte nella sottosezione Commits per i branch protetti.

#### 4.2.1.5 Milestone<sub>g</sub>

Una milestone indica un traguardo intermedio significativo per il progetto. Ad essa possono venire assegnate delle issues per verificarne il raggiungimento. Ogni milestone ha una scadenza che viene discussa e fissata da tutto il gruppo. Oltre alle 2 milestone + 1 milestone facoltativa fissate dal committente il gruppo ne creerà ulteriori per scandire più nel dettaglio i passi che ci porteranno ad ottenere i risultati prefissati. Una delle milestone create dal gruppo durante il completamento della technology baseline relativa alla prima milestone imposta dal committente (Requirements and Technology Baseline) riguarda l'implementazione di un PoC (Proof of concept).

#### 4.2.1.6 Projects Board<sub>g</sub>

Vengono utilizzate due project board per tracciare le issues della repo. Una project board principale utilizzata da tutti i membri del gruppo e una project board per le approvazioni utilizzata solo dal responsabile di progetto per approvare i file che richiedono l'approvazione prima di una consegna.

- La projectboard<sub>g</sub> principale è suddivisa in queste sezioni:
  - **Todo:** issue<sub>g</sub> che non sono ancora state iniziate o che non sono ancora state assegnate;
  - **In Progress:** issue<sub>g</sub> che sono state assegnate e a cui almeno un membro a cui è stata assegnata ha iniziato a lavorarci;
  - **Pull Request:** issue<sub>g</sub> che è in fase di integrazione e necessita della verifica dei verificatori. Corrisponde all'inizio di una pull request;
  - **Done:** issue<sub>g</sub> che sono state chiuse e che sono state verificate (se necessitano di verifica<sub>g</sub>);
  - **Approved:** issue<sub>g</sub> con label "Da Approvare" che hanno ottenuto l'approvazione<sub>g</sub> del responsabile subito dopo la verifica; Per tutte le issues che richiedono un'approvazione solo al momento della consegna è stata creata la project board dedicata alle approvazioni.

Inoltre nella project board principale vengono registrate delle issue che non richiedono verifica, approvazione o neanche integrazione, con lo scopo di monitorare meglio il lavoro di ogni membro del team.

Queste issue verranno chiuse e archiviate manualmente una volta che avranno terminato la loro utilità, un esempio può essere la seguente issue:

**diario di bordo 21-11-22;** questa issue non necessita verifica, approvazione o integrazione perchè non è di interesse caricare il file nella repo, però è utile tracciare lo svolgimento della issue.

- La projectboard<sub>g</sub> riservata alle approvazioni è suddivisa in queste sezioni:
  - **Todo:** issue<sub>g</sub> che non sono ancora state iniziate dal responsabile;
  - **In Progress:** issue<sub>g</sub> che sono state prese in carico per essere approvate dal responsabile;
  - **Pull Request:** issue<sub>g</sub> che è in fase di integrazione e necessita della verifica dei verificatori. Corrisponde all'inizio di una pull request; in questo caso i verificatori dovranno solo controllare che l'intestazione e il registro delle modifiche siano stati compilati correttamente dal responsabile di progetto in quanto tutto il resto del contenuto è già stato verificato in precedenza;
  - **Approved:** issue<sub>g</sub> che sono state approvate dal responsabile.

Le issue che si trovano nella project board approvazioni sono state create come continuazione ad issues che hanno in precedenza attraversato il work flow della project board principale fino allo stato "Done". In questo modo il Responsabile di progetto dovrà approvare solamente file che si trovano nei branches protetti. Vengono ricreate in questa project board per mantenerne la tracciabilità e permetterne una più facile reperibilità futura per l'approvazione. Questo permette anche di pulire la project board principale da tutte quelle issues che rimarrebbero inattive per molto tempo.

Esempio di work flow issues:

1. Viene creata l'issue ed inserita all'interno della sezione "To Do" della project board principale;
2. Quando viene presa in carico l'issue viene spostata nella sezione "In Progress" fino al suo completamento;
3. L'incaricato alla risoluzione dell'issue apre una pull request a cui assegna l'issue in modo che venga chiusa in automatico al momento dell'integrazione con il branch di destinazione;
4. Vengono fatte le verifiche opportune dai verificatori e le conseguenti modifiche prima di accettare la pull request;

5. Dopo l'accettazione la pull request viene postata insieme alle issues associate nella colonna "Done";
6. A questo punto si possono presentare due casi:
  - (a) I file relativi alla pull request richiedono approvazione immediata. Viene archiviata la pull request (e le issue associate) e creata una issue a cui viene aggiunta la label "Da approvare" con il nome del file da approvare. Il responsabile creerà un branch per approvare le issues con approvazione immediata aprirà una pull request per integrare nel branch di destinazione l'approvazione;
  - (b) I file relativi alla pull request richiedono l'approvazione prima di una consegna. Il responsabile archivia la pull request e ne crea una issue con il nome dei file da approvare nella project board dedicata alle approvazioni. NB: se nella project board Approvazioni è già presente un'issue di approvazione per il file da approvare non viene creata una nuova issue di approvazione ma si tiene la precedente. L'issue segue il work flow della project board approvazioni.

#### 4.2.1.7 Issue Tracking System<sub>g</sub>

Il gruppo utilizza l'issue tracking system<sub>g</sub> di Github<sub>g</sub> per tenere traccia delle issue<sub>g</sub>. Le issues<sub>g</sub> verranno determinate dal responsabile, ma la loro assegnazione verrà effettuata dai membri del gruppo, in base alla priorità delle issues<sub>g</sub>, i loro ruoli e alle loro disponibilità temporali.

Per marciare le issues secondo criteri di interesse (come ambito o priorità) vengono utilizzate le labels.

##### Labels:

- **Da approvare:** indica una issue o pull request che necessita di approvazione;
- **Da verificare:** indica una issue o pull request che necessita di verifica;
- **Documentazione:** indica una issue o pull request riguardante la documentazione;
- **P=Bassa:** indica una issue o pull request a priorità bassa (scadenza lontana o non limita il lavoro altrui);
- **P=Media:** indica una issue o pull request a priorità media (scadenza di almeno due settimane o non limita il lavoro altrui);
- **P=Alta:** indica una issue o pull request a priorità alta (scadenza breve o limita il lavoro altrui);
- **Presentazione:** indica una issue riguardante la redazione di una presentazione in classe o al proponente.

Nel caso un membro del gruppo dovesse rendersi conto che l'issue<sub>g</sub> che sta svolgendo potrebbe essere suddiviso in ulteriori issue<sub>g</sub>, dovrà rivolgersi al responsabile, che è l'unico che può aggiungere, modificare o eliminare le issue<sub>g</sub>.

Per raggruppare più issue si marca ciascuna con la label di raggruppamento. Il nome di una label di raggruppamento viene preceduto da una G maiuscola (che sta per gruppo) seguito dal nome dell'attività da svolgere. esempio: "G Piano di progetto" Al completamento di tutte le issues di un gruppo si potrà scegliere se eliminare la label o tenerla per raggruppare issues future dello stesso gruppo. indica un gruppo di labels relative al documento *Piano di Progetto*.

##### Utilizzo delle checkbox:

Una issue può essere suddivisa in più attività tramite delle checkbox in base alla grandezza o alla necessità di suddividere il lavoro. Al completamento di un'attività si spunta la checkbox corrispondente in modo da avvisare il gruppo sullo stato di avanzamento di quella issue.

#### 4.2.2 Metriche

Il processo di gestione delle infrastrutture non utilizza metriche qualitative particolari.



## 5 Standard di qualità ISO/IEC 9126

Questa sezione descrive in dettaglio lo standard ISO/IEC 9126 utilizzato dal gruppo. Le norme di questo standard descrivono un modello di qualità del software, definiscono le caratteristiche che lo determinano e propongono metriche per la misurazione. Le norme constano di quattro parti:

- Parte 1: Metriche per la qualità esterna
- Parte 2: Metriche per la qualità interna
- Parte 3: Modello della qualità del software
- Parte 4: Metriche per la qualità in uso

### 5.1 Qualità esterne

Le metriche esterne misurano i comportamenti del prodotto software rilevabili dai test, dall'operatività, dall'osservazione durante la sua esecuzione. L'esecuzione del prodotto software è fatta in base al suo utilizzo in funzione degli obiettivi di business stabiliti ed in un contesto organizzativo e tecnico rilevante.

### 5.2 Qualità interne

Le metriche interne si applicano alla parte non eseguibile del software (come le specifiche tecniche o il codice sorgente) durante le fasi di progettazione e codifica. Durante le fasi di sviluppo del software, quindi, i prodotti intermedi sono valutati tramite metriche interne che misurano le proprietà intrinseche del prodotto. Le metriche interne permettono ad utenti e sviluppatori di individuare eventuali problemi che potrebbero inibire la qualità finale del prodotto.

### 5.3 Modello di qualità

Il modello definisce le caratteristiche di qualità. A ciascuna di esse sono associate sotto-caratteristiche. La tabella successiva descrive le caratteristiche e le sue sotto-caratteristiche del software proposti dal modello. Il modello presenta le seguenti caratteristiche:

#### 5.3.1 Funzionalità

Si intende quella capacità di un prodotto software di determinare le esigenze richieste tramite delle funzioni, sotto precise condizioni. Le sue sotto-caratteristiche sono rispettivamente:

- **Appropriatezza:** rappresenta la capacità di fornire un appropriato insieme di funzioni che permettano agli utenti di svolgere determinate task e di raggiungere gli obiettivi prefissati
- **Accuratezza:** rappresenta la capacità del software di fornire i risultati o gli effetti attesi con il livello di precisione richiesta
- **Interoperabilità:** rappresenta la capacità del software di interagire con uno o più sistemi specificati
- **Conformità:** rappresenta la capacità del software di aderire a standard, convenzioni e regolamenti di carattere legale o prescrizioni simili che abbiano attinenza con la funzionalità
- **Sicurezza:** rappresenta la capacità del software di proteggere le informazioni ed i dati in modo che, persone o sistemi non autorizzati, non possano accedervi e quindi non possano leggerli o modificarli.

### 5.3.2 Affidabilità

Si intende quella capacità di un prodotto software di mantenere il livello di prestazione quando utilizzato sotto certe condizioni specifiche. Le sue sotto-caratteristiche sono rispettivamente:

- **Maturità:** rappresenta la capacità del software di evitare che si verifichino errori o siano prodotti risultati non corretti in fase di esecuzione
- **Tolleranza agli errori:** rappresenta la capacità del software di mantenere il livello di prestazioni in caso di errori nel software o di violazione delle interfacce specificate
- **Recuperabilità:** rappresenta la capacità del software di ripristinare il livello di prestazioni e di recuperare i dati direttamente coinvolti in caso di errori o malfunzionamenti
- **Aderenza:** rappresenta la capacità del software di aderire a standard, convenzioni e regole relative all'affidabilità

### 5.3.3 Efficienza

Si intende quella capacità di un prodotto software di realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie. Le sue sotto-caratteristiche sono rispettivamente:

- **Comportamento rispetto al tempo:** rappresenta la capacità del software di fornire appropriati tempi di risposta, tempi di elaborazione e quantità di lavoro eseguendo le funzionalità previste sotto determinate condizioni di utilizzo
- **Utilizzo delle risorse:** rappresenta la capacità del software di utilizzare un appropriato numero e tipo di risorse in maniera adeguata
- **Conformità:** rappresenta la capacità del software di aderire a standard e convenzioni relative all'efficienza

### 5.3.4 Usabilità

Si intende quella capacità di un prodotto software di essere comprensibile e appreso dall'utente, quando usato sotto condizioni specificate. Le sue sotto-caratteristiche sono rispettivamente:

- **Comprensibilità:** rappresenta la capacità del software di permettere all'utente di capire le sue funzionalità e come poterla utilizzare con successo per svolgere particolari task in determinate condizioni di utilizzo.
- **Apprendibilità:** rappresenta la capacità del software di permettere all'utente di imparare l'applicazione
- **Operabilità:** rappresenta la capacità del software di permettere all'utente di utilizzarlo e di controllarlo
- **Attrattività:** rappresenta la capacità del software di risultare "attraente" per l'utente
- **Conformità:** rappresenta la capacità del software di aderire a standard, convenzioni o regole relative all'usabilità

### 5.3.5 Manutenibilità

Si intende quella capacità di un prodotto software di essere modificato, includendo correzioni, miglioramenti o adattamenti. Le sue sotto-caratteristiche sono rispettivamente:

- **Analizzabilità:** rappresenta la capacità del software di poter effettuare la diagnosi sul software ed individuare le cause di errori o malfunzionamenti

- Modificabilità: rappresenta la capacità del software di consentire lo sviluppo di modifiche al software originale. Si tratta quindi di modifiche al codice o alla progettazione ed alla sua documentazione
- Stabilità: rappresenta la capacità del software di evitare effetti non desiderati a seguito di modifiche al software
- Testabilità: rappresenta la capacità del software di consentire la verifica e validazione del software modificato, cioè di eseguire i test

#### 5.3.6 Portabilità

Si intende quella capacità di un prodotto software di poter essere trasportato da un ambiente di lavoro ad un altro. Le sue sotto-caratteristiche sono rispettivamente:

- Adattabilità: rappresenta la capacità del software di essere adattato a differenti ambienti senza richiedere azioni specifiche diverse da quelle previste dal software per tali attività
- Installabilità: rappresenta la capacità del software di essere installato in un determinato ambiente
- Conformità: rappresenta la capacità del software di aderire a standard, convenzioni o regole relative alla portabilità
- Sostituibilità: rappresenta la capacità del software di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente

### 5.4 Qualità in uso

Le metriche delle qualità in uso misurano il grado con cui il prodotto software permette agli utenti di svolgere le proprie attività con efficacia, produttività, sicurezza e soddisfazione nel contesto operativo previsto.

- Efficacia: la capacità del software di permettere all'utente di svolgere le funzioni specificate con accuratezza e completezza
- Produttività: la capacità di far utilizzare all'utente un quantitativo adeguato di risorse in relazione al caso d'uso
- Soddisfazione: la capacità del software di soddisfare l'utente
- Sicurezza: la capacità del prodotto software di avere livelli accettabili di rischio per dati e persone

## 6 Standard di qualità ISO/IEC 12207:1995

Questa sezione descrive in dettaglio lo standard ISO/IEC 12207:1995 scelto dal gruppo per garantire la qualità dei processi del ciclo di vita di un software. I processi dello standard contengono attività e compiti che devono essere applicati durante l'acquisizione di un sistema software. Esso contiene tre tipologie di processi che sono:

- Processi primari
- Processi di supporto
- Processi organizzativi

### 6.1 Processi primari

Questi processi sono essenziali in quanto un progetto è detto tale se e solo se è attivo almeno un processo primario. I processi di supporto definiti dallo standard sono i seguenti.

#### 6.1.1 Acquisizione

Il processo di acquisizione contiene le attività ed i compiti dell'acquirente. Le attività del processo sono le seguenti:

- Iniziazione
- Preparazione della richiesta di proposta
- Preparazione e aggiornamento del contratto
- Monitoraggio dei fornitori
- Accettazione e completamento

#### 6.1.2 Fornitura

Il processo di fornitura contiene le attività e le mansioni del fornitore. Il processo può essere. Le attività del processo sono le seguenti:

- Iniziazione
- Preparazione alla risposta
- Contratto
- Pianificazione
- Esecuzione e controllo
- Revisione e valutazione
- Rilascio e completamento

### 6.1.3 Sviluppo

Il processo di sviluppo contiene le attività e i compiti dello sviluppatore. Il processo contiene le attività per l'analisi dei requisiti, la progettazione, la codifica, l'integrazione, il test, l'installazione e l'accettazione relative al sistema se previsto dal contratto. Le attività del processo sono le seguenti:

- Implementazione dei processi
- Analisi dei requisiti di sistema
- Progettazione architettura di sistema
- Analisi requisiti software
- Progettazione architettura software
- Progettazione dettagliata del software
- Codifica e test del software
- Integrazione software
- Test di qualificazione del software
- Integrazione del sistema
- Test di qualificazione del sistema
- Installazione software
- Supporto per l'accettazione del software

### 6.1.4 Gestione operativa

Il processo di gestione operativa contiene le attività e i compiti dell'operatore. Il processo riguarda il funzionamento del prodotto software e il supporto operativo agli utenti. Le attività del processo sono le seguenti:

- Implementazione dei processi
- Collaudo operativo
- Operazione di sistema
- Supporto all'utente

### 6.1.5 Manutenzione

Il processo di manutenzione contiene le attività e i compiti del manutentore. Questo processo viene attivato quando il prodotto software subisce modifiche al codice e alla documentazione associata a causa di un problema o necessità di miglioramento o adattamento. Le attività del processo sono le seguenti:

- Implementazione di processi
- Analisi dei problemi e delle modifiche
- Implementazione della modifica
- Revisione/Accettazione della manutenzione
- Migrazione
- Ritiro del software

## 6.2 Processi di supporto

I processi di supporto aiutano le attività di tutti gli altri processi dell'organizzazione a garantire il successo e la qualità del progetto. Questi processi possono essere attivati da un processo primario o da un altro processo di supporto. Le attività e i compiti in un processo di supporto sono di responsabilità dell'organizzazione che esegue tale processo. Questa organizzazione garantisce che il processo sia in atto e funzionante. I processi di supporto definiti dallo standard sono i seguenti.

### 6.2.1 Documentazione

Il processo di documentazione è un processo per la registrazione delle informazioni prodotte da un processo o attività del ciclo di vita. Il processo contiene l'insieme delle attività che pianificano, progettano, sviluppano, producono, modificano, distribuiscono e mantengono quei documenti necessari a tutti gli interessati. Le attività del processo sono le seguenti:

- Implementazione di processo
- Produzione
- Progettazione e sviluppo
- Manutenzione

### 6.2.2 Gestione della configurazione

Il processo di gestione della configurazione è un processo di applicazione di procedure amministrative e tecniche durante tutto il ciclo di vita del software per mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto. Le attività del processo sono le seguenti:

- Implementazione del processo
- Identificazione della configurazione
- Controllo della configurazione
- Valutazione dello stato di configurazione
- Gestione del rilascio e distribuzione

### 6.2.3 Accertamento della qualità

Il processo di accertamento della qualità è un processo per fornire un'adeguata garanzia che i prodotti ed i processi software nel ciclo di vita del progetto siano conformi ai requisiti specificati e aderiscano ai piani stabiliti. Per essere imparziale, la garanzia della qualità deve avere libertà organizzativa e autorità da parte delle persone direttamente responsabili dello sviluppo del prodotto software o dell'esecuzione del processo nel progetto. Le attività del processo sono le seguenti:

- Implementazione del processo
- Accertamento di prodotto
- Accertamento di processo
- Accertamento della qualità di sistema

### 6.2.4 Verifica

Il processo di verifica è un processo per determinare se i prodotti software di un'attività soddisfano i requisiti o le condizioni loro imposti nelle attività precedenti. Le attività del processo sono le seguenti:

- Implementazione del processo
- Verifica

### 6.2.5 Validazione

Il processo di validazione è un processo per determinare se i requisiti e il sistema finale soddisfano l'uso specifico previsto. Le attività del processo sono le seguenti:

- Implementazione del processo
- Validazione

### 6.2.6 Revisione congiunta

Il processo di revisione congiunta è un processo per valutare lo stato ed i prodotti di un'attività di un progetto in modo appropriato. Le revisioni congiunte avvengono sia a livello di gestione del progetto che a livello tecnico.

- Implementazione del processo
- Revisioni della gestione del progetto
- Revisioni tecniche

### 6.2.7 Audit

Il processo di Audit ha lo scopo di determinare in maniera indipendente la conformità di prodotti e processi selezionati ai requisiti, piani e accordi. Le attività del processo sono le seguenti:

- Implementazione del processo
- Audit

### 6.2.8 Risoluzione dei problemi

Il Processo di risoluzione dei problemi è un processo per l'analisi e la risoluzione dei problemi, indipendentemente dalla loro natura o origine, che vengono scoperti durante l'esecuzione di processi di sviluppo, funzionamento, manutenzione o altri. L'obiettivo è fornire un mezzo tempestivo, responsabile e documentato per garantire che tutti i problemi scoperti siano analizzati e risolti e che le tendenze siano riconosciute.

- Implementazione del processo
- Risoluzione del problema

## 6.3 Processi organizzativi

Le attività e i compiti in un processo organizzativo sono di responsabilità dell'organizzazione che utilizza il processo. L'organizzazione garantisce che il processo sia attivo e funzionante. I processi organizzativi definiti nello standard sono i seguenti.

### 6.3.1 Gestione organizzativa

Il processo di gestione organizzativa ha lo scopo di organizzare e monitorare i processi per il raggiungimento dei loro obiettivi. Il processo è stabilito da una organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti. Le attività del processo sono le seguenti:

- Inizio e definizione dello scopo
- Pianificazione
- Esecuzione e controllo

- Revisione e valutazione
- Chiusura

### **6.3.2 Gestione delle infrastrutture**

Il processo di gestione delle infrastrutture è un processo per stabilire e mantenere l'infrastruttura necessaria per qualsiasi altro processo. L'infrastruttura può includere hardware, software, strumenti, tecniche, standard e strutture per lo sviluppo, il funzionamento o la manutenzione. Le attività del processo sono le seguenti:

- Implementazione del processo
- Realizzazione dell'infrastruttura
- Manutenzione dell'infrastruttura

### **6.3.3 Miglioramento del processo**

Il processo di miglioramento è un processo per stabilire, valutare, misurare, controllare e migliorare un processo del ciclo di vita del software. Le attività del processo sono le seguenti:

- Stabilimento del processo
- Valutazione del processo
- Miglioramento del processo

### **6.3.4 Formazione del personale**

Il processo di formazione del personale è un processo per fornire e mantenere personale qualificato. L'acquisizione, la fornitura, lo sviluppo, il funzionamento o la manutenzione di prodotti software dipende in gran parte da personale esperto e qualificato. Le attività del processo sono le seguenti:

- Implementazione del processo
- Sviluppo materiale didattico
- Realizzazione piano formativo