Università degli Studi di Padova
Laurea: Informatica
Corso: Ingegneria del Software
Anno Accademico: 2024/2025

Gruppo: SWEg Labs
Email: gruppo.sweg@gmail.com

Norme di Progetto

Versione 1.0.0

Stato	Approvato
Redazione	Federica Bolognini Michael Fantinato Giacomo Loat Filippo Righetto Riccardo Stefani Davide Verzotto
Verifica	Federica Bolognini Michael Fantinato Giacomo Loat Filippo Righetto Riccardo Stefani Davide Verzotto
Proprietario	Davide Verzotto
Uso	Interno
Destinatari	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo *SWEg Labs*

Registro delle modifiche

Versione	Data	Descrizione	Autore	Verifica
1.0.0	23-01-2025	Approvazione del documento	Davide Verzotto	Davide Verzotto
0.3.0	23-01-2025	Verifica del documento	Giacomo Loat	Giacomo Loat
0.2.5	14-01-2025	Continuazione stesura delle sezioni §**2** e §**4**	Federica Bolognini	Davide Verzotto
0.2.4	02-01-2025	Stesura della sezione §**3.4**	Filippo Righetto	Federica Bolognini
0.2.3	02-01-2025	Stesura della sezione §**3.5**	Davide Verzotto	Filippo Righetto
0.2.2	01-01-2025	Continuazione stesura della sezione §**2**	Federica Bolognini	Davide Verzotto
0.2.1	31-12-2024	Stesura della sezione §**3.3**	Davide Verzotto	Riccardo Stefani
0.2.0	06-12-2024	Verifica del documento allo stato attuale	Federica Bolognini	Federica Bolognini
0.1.7	19-11-24	Correzioni sezione §**2**	Davide Verzotto	Giacomo Loat
0.1.6	17-11-24	Stesura della sezione §**2**	Davide Verzotto	Giacomo Loat
0.1.5	17-11-24	Stesura della sezione §**4**	Giacomo Loat	Riccardo Stefani
0.1.4	16-11-24	Stesura delle sezioni §**A**, §**B** e §**C**	Riccardo Stefani	Filippo Righetto
0.1.3	10-11-24	Stesura della sezione §**1** introduzione	Davide Verzotto	Riccardo Stefani
0.1.2	10-11-24	Scrittura della sezione §**3.2** sulla gestione della configurazione	Riccardo Stefani	Davide Verzotto
0.1.1	09-11-24	Scrittura della sezione §**3.1** sulla documentazione	Riccardo Stefani	Federica Bolognini
0.1.0	04-11-24	Creazione del documento	Riccardo Stefani	Federica Bolognini

Tabella 1: Registro delle modifiche

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire il *way of working*$_G$ che il gruppo *SWEg Labs* dovrà rispettare al fine di completare il progetto BuddyBot secondo i principi di *efficacia*$_G$ ed *efficienza*$_G$. Vengono inoltre specificati gli strumenti utilizzati, insieme al loro scopo e alla spiegazione dietro la loro scelta.

1.2 Scopo del prodotto

Al giorno d'oggi, per effetto della digitalizzazione è sempre più importante l'accesso a un numero crescente di informazioni eterogenee per mantenere la produttività. I dati richiesti provengono spesso da *fonti*$_G$ differenti, risultando talvolta difficili da identificare e *rintracciare*$_G$ con precisione. Inoltre la digitalizzazione, pur semplificando l'*accessibilità*$_G$ delle informazioni rispetto ai formati analogici, ha reso necessaria un'organizzazione più rigorosa di questi ultimi, aumentando l'importanza di una ricerca efficiente e mirata.
Il capitolato si propone quindi di migliorare il processo di ricerca e recupero delle informazioni tramite l'intelligenza artificiale, riducendo quindi i tempi di ricerca e incrementando la produttività.
Per raggiungere questi obiettivi, il gruppo *SWEg Labs* realizzerà un software basato sull'intelligenza artificiale. Il quale, integrato con *API*$_G$ di terze parti, riceverà le richieste in linguaggio naturale e ne fornirà le risposte adattate alle esigenze dell'utente.

1.3 Glossario

Al fine di evitare incomprensioni dovute all'utilizzo di termini ambigui, sarà redatto un glossario ordinato, contenente i termini utilizzati nella documentazione che potrebbero trarre in inganno. Ogni termine incluso nel glossario sarà evidenziato in corsivo e contrassegnato con il simbolo "G" in pedice alla sua prima occorrenza.

2 Processi primari

2.1 Fornitura

2.1.1 Descrizione

Questa sezione riporta tutte le norme, gli strumenti e i metodi che ogni membro del gruppo *SWEg Labs* si impegna a rispettare al fine di preservare al meglio i rapporti con il proponente *AzzurroDigitale*$_G$.

2.1.2 Scopo

Il *processo*$_G$ di fornitura intende occuparsi della gestione dei rapporti con il proponente *AzzurroDigitale* con l'obbiettivo di evitare qualsiasi tipo di ostacolo alla comunicazione ed avere un a buona qualità nella stessa.

2.1.3 Aspettative

Durante il rapporto il nostro gruppo desidera mantenere una comunicazione disponibile e con *AzzurroDigitale*, in particolare con i referenti Martina Daniele, Camilla Picello, Nicola Boscaro e Mattia Gottardello, così da poter:

- Discutere *requisiti*$_G$ chiave necessari da soddisfare nel prodotto finale;
- Stabilire tempistiche di lavoro;
- Ricevere *feedback*$_G$ sul lavoro in corso;
- Ottenere chiarimenti relativi a dubbi e incomprensioni;
- Stabilire i *vincoli*$_G$ riguardanti i processi intermedi.

2.1.4 Rapporti col proponente

Il proponente mette a disposizione un canale Discord al fine di rispondere a domande/dubbi occasionali, e una mail di riferimento per comunicazioni più formali.
Inoltre il proponente si rende disponibile a degli incontri da remoto, la cui frequenza è di una volta ogni 2 settimane. E in aggiunta due incontri in presenza, di cui il primo ad inizio progetto e il secondo nella parte finale. Ad ognuno di questi incontri la discussione verterà su 2 punti:

- *Revisione*$_G$ sul lavoro svolto nel precedente sprint;
- Raccolta delle nuove *specifiche*$_G$ e richieste da soddisfare per lo sprint successivo.

Durante tali incontri, l'azienda non richiede della specifica documentazione, ma gradisce strumenti per visualizzare, anche graficamente, lo stato di avanzamento del lavoro appena svolto. Per ciascun incontro verrà compilato dal nostro gruppo un verbale esterno, contenente gli argomenti di discussione e le decisioni prese, e sarà firmato dalla rappresentanza del proponente. Tutti i verbali saranno visibili nella *repository*$_G$ dedicata alla documentazione.

2.1.5 Strumenti

Di seguito sono riportati gli strumenti utilizzati per realizzare il processo di fornitura:

- ***Git***$_G$: software per il *controllo di versione*$_G$;
- ***GitHub***$_G$: servizio di *hosting*$_G$ per progetti software;
- ***Jira***$_G$: è un sistema software utilizzato per la gesitone delle attività, l'assegnazione delle risorse, la verifica dei tempi del progetto e l'analisi del lavoro svolto e da svolgere. Questo strumento è utile anche per generare i *diagrammi di Gantt*$_G$ presenti nel Piano di Progetto;
- ***Discord***$_G$: *piattaforma*$_G$ che mette a disposizione dei canali vocali con la possibilità di condivisione dello schermo; Utilizzata non solo dai membri del gruppo per comunicazioni interne ma anche per le comunicazioni rapide con il proponente;

- ***Google Meet***$_G$: piattaforma che permette di effettuare videoconferenze online. Utilizzata per organizzare incontri con il proponente;
- ***LaTeX***$_G$: linguaggio di *markup*$_G$ scelto dal gruppo per la produzione della documentazione.

2.2 Sviluppo

2.2.1 Scopo

La fase di sviluppo si occupa di definire le attività che il team compie per soddisfare i requisiti delineati con il proponente.

2.2.2 Descrizione

Il processo di sviluppo consiste nello strutturare, suddividere ai membri del team e completare le attività relative alla *codifica*$_G$. L'obbiettivo è che il software soddisfi le *aspettative*$_G$ del proponente. Nel processo di sviluppo saranno effettuate le seguenti attività.

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.3 Aspettative

Il gruppo *SWEg Labs* intende ottenere tramite il processo di sviluppo un prodotto software in grado di superare i test e soddisfare i requisiti del proponente *AzzurroDigitale*.

2.2.4 Analisi dei requisiti

2.2.4.1 Descrizione

L'analisi dei requisiti è un'attività svolta dall'analista, e produce il documento denominato "Analisi dei requisiti". Tale documento descrive:

- Lo scopo del prodotto;
- Le sue *funzionalità*$_G$;
- Gli *attori*$_G$ e le loro caratteristiche;
- I *casi d'uso*$_G$;
- I requisiti;
- La stima del lavoro necessario.

2.2.4.2 Scopo

L'analisi dei requisiti intende raccogliere, chiarire e precisare tutti i requisiti necessari da completare per soddisfare il cliente. Per poter fare ciò è necessario aver letto e compreso al meglio le specifiche del progetto, e poter comunicare al meglio con il proponente.

2.2.4.3 Casi d'uso

Un *caso d'uso*$_G$ è un insieme di *scenari*$_G$ che hanno in comune uno scopo finale per un *attore*$_G$.
Gli elementi che lo compongono sono i seguenti:

- l'attore;
- il sistema;
- precondizioni;
- postcondizioni;

- *Scenario principale*$_G$;
- *Scenario alternativo*$_G$;
- inclusione/i;
- estensione/i;
- specializzazione/i;
- commento/i;
- descrizione.

I casi d'uso sono identificati nel seguente modo:

UC[Numero]+(UC[Numero sottocaso])-[Titolo]

dove:

- UC: acronimo di "use case";
- Numero: numero identificativo del caso d'uso;
- Numero sottocaso: numero identificativo del sottocaso (se presente);
- Titolo: titolo assegnato al caso d'uso.

2.2.4.4 Struttura dei requisiti

I requisiti sono identificati da un codice univoco strutturato nel seguente modo:

R[Importanza][Tipologia] [Codice](+[Codice figlio])

dove:

- R: acronimo di "Requisito";
- Importanza: indica l'importanza del requisito e può assumere i seguenti valori:
  - O: requisito obbligatorio, cioè deve essere soddisfatto necessariamente per garantire la realizzazione del prodotto corrispondente agli accordi col *proponente*$_G$;
  - D: requisito desiderabile, cioè porterebbe al prodotto ulteriori funzionalità e completezza qualora fosse soddisfatto;
  - Z: requisito opzionale, cioè che potrebbe essere implementato solo se ci sono risorse, tempo e budget sufficienti, senza che la sua mancanza impatti negativamente il prodotto.
- Tipologia:
  - F: requisito funzionale che delinea gli obiettivi e le azioni chiave che l'utente deve essere in grado di compiere;
  - Q: requisito qualitativo che delinea le specifiche qualitative che devono essere rispettate per garantire la qualità del sistema;
  - V: requisito di vincolo che rappresenta le restrizioni e le condizioni che devono essere soddisfatte durante lo sviluppo e l'implementazione del sistema;
  - I: requisito implementativo che delinea le specifiche tecniche e operative che indicano come un sistema o una funzionalità deve essere sviluppato per soddisfare i requisiti del progetto;
  - P: requisito prestazionale che definisce le metriche di performance che il sistema deve soddisfare.
- Codice: identificatore numerico univoco per quella tipologia di vincolo;
- Codice figlio: numero identificativo del sottorequisito (se presente).

2.2.5 Progettazione

2.2.5.1 Scopo

L'attività di progettazione ha l'obiettivo di definire l'architettura del prodotto in modo da soddisfare le esigenze di tutti gli *stakeholder*$_G$, identificate dall'analisi dei requisiti. Identificando e documentando le soluzioni che rispondono ai requisiti evidenziati si garantisce al contempo una chiara suddivisione delle responsabilità di sviluppo e manutenzione. Questa attività fornisce una documentazione completa e dettagliata sulla struttura del prodotto, incluse le specifiche tecniche e le scelte tecnologiche adottate.

2.2.5.2 Descrizione

La progettazione si articola in più livelli per assicurare una completa copertura delle funzionalità e della struttura del prodotto:

- **Progettazione logica**: Questa fase definisce le tecnologie, i *framework*$_G$ e le librerie scelti per la realizzazione del prodotto, motivando l'adeguatezza delle scelte e dimostrando la fattibilità tecnica attraverso un *Proof of Concept*$_G$(PoC);
- **Progettazione di dettaglio**: In questa fase si definisce l'*architettura*$_G$ di dettaglio, seguendo quanto stabilito nella progettazione logica e sviluppando una rappresentazione completa delle componenti software.

2.2.6 Codifica

2.2.6.1 Scopo

L'attività di *codifica*$_G$ è mirata allo sviluppo del prodotto software da parte dei programmatori, il quale deve soddisfare le esigenze concordate con il *proponente*.

2.2.6.2 Descrizione

Durante questa attività, lo sviluppatore si impegna a soddisfare i requisiti implementando il codice nel linguaggio di programmazione scelto. Il codice deve rispettare le linee guida definite nella documentazione del progetto. Contestualmente, tutte le nuove unità software sviluppate e le modifiche apportate devono essere adeguatamente documentate.

2.2.6.3 Stile della codifica

Per garantire lo sviluppo del codice di qualità useremo i seguenti criteri:

- Backend:
  - **Variabili, attributi, funzioni e metodi:** *Snake Case*$_G$.
  - **Classi:** *Pascal Case*$_G$.
  - **Nomi di file:**
    - Camel Case se il file contiene una classe;
    - Snake Case se non contiene una classe.
  - **Docstring**: Ogni metodo o funzione deve avere un commento descrittivo sotto alla firma, scritto in inglese.
- Frontend:
  - **Variabili, attributi, funzioni e metodi:** *Camel Case*$_G$.
  - **Classi:** Pascal Case.
  - **Nomi dei file:** *Kebab Case*$_G$.
- **Lunghezza delle righe di codice**:
  - La lunghezza massima di una riga di codice non deve superare i 100 caratteri.
- **Indentazioni**:
  - I blocchi annidati del codice devono seguire un'indentazione con un carattere di tabulazione equivalente a 4 spazi.

2.2.6.4 Tecnologie Utilizzate

- ***Python***$_G$: un linguaggio di programmazione ad alto livello orientato ad oggetti. Si utilizza per realizzare la logica dell'applicazione.

  `https://www.python.org/`

- ***Angular***$_G$: un framework usato per lo sviluppo dell'interfaccia grafica dell'applicazione.

  `https://angular.dev/`

- ***Chroma***$_G$: un database vettoriale che memorizza e ricerca dati basandosi sulla loro somiglianza semantica.

  `https://www.trychroma.com/`

- ***GPT-4o***$_G$: un *LLM*$_G$, sviluppato da OpenAI, capace di comprendere e generare testo in modo contestuale.

  `https://openai.com/index/hello-gpt-4o/`

- ***Docker***$_G$: una piattaforma per eseguire applicazioni in container, ambienti isolati che includono tutto il necessario per funzionare ovunque.

  `https://www.docker.com`

- ***Python-Crontab***$_G$: una libreria Python che permette di leggere, scrivere e gestire *cron job*$_G$ direttamente da codice Python.

  `https://pypi.org/project/python-crontab/`

3 Processi di Supporto

3.1 Documentazione

3.1.1 Scopo

Il *processo*$_G$ di documentazione mira a raccogliere le informazioni prodotte da un processo o da un'attività nel *ciclo di vita*$_G$, le decisioni prese dal gruppo e gli standard adottati per lo svolgimento del progetto. È obbligatorio per tutti i membri del gruppo il rispetto di queste regole.

3.1.2 Descrizione

La documentazione costituisce una componente essenziale del progetto, poichè consente di registrare ogni aspetto del lavoro svolto e delle decisioni prese. In particolare, questa sezione raccoglie tutte le norme relative alla creazione, all'aggiornamento e al mantenimento della documentazione (interna ed esterna) prodotta dal gruppo *SWEg Labs* per ciascuna fase del ciclo di vita del software.

3.1.3 Aspettative

Per quanto riguarda il processo di documentazione, il team ha le seguenti aspettative:

- Definire delle procedure ripetibili che permettano di standardizzare il *way of working*$_G$ e la documentazione prodotta dal gruppo;
- Dichiarare le norme che i membri del gruppo sono tenuti a seguire per semplificare la redazione dei documenti.

3.1.4 Ciclo di vita dei documenti

Il ciclo di vita di ogni documento si compone delle seguenti fasi, visibili in Figura 1:

- **Redazione**: documenti vengono scritti seguendo un approccio incrementale, e sono considerati redatti soltanto una volta completata la loro stesura;
- **Verifica**$_G$: ogni volta che i documenti vengono modificati necessitano di una verifica. Ogni sezione coinvolta nella modifica di un documento viene verificata da una o più persone, chiamate verificatori. Il documento è considerato verificato una volta completate le verifiche da parte di tutti i verificatori incaricati;
- **Approvazione**: in questa fase, il Responsabile di Progetto dichiara che il documento è pronto per essere rilasciato, cioè che la sua stesura è completata ed è stato verificato. A questo punto, il documento viene marcato come approvato.

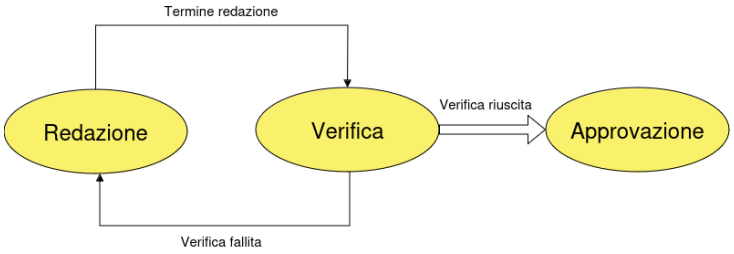

Figura 1: Ciclo di vita della documentazione

3.1.5 Struttura dei documenti

Ogni documento che verrà prodotto dovrà seguire una precisa struttura per garantire omogeneità e coesione.

3.1.5.1 Numerazione di pagine

La numerazione delle pagine del documento segue uno schema ben definito. Dalle pagine iniziali e fino alla pagina precedente l'inizio del primo capitolo, viene utilizzata la numerazione romana. Questa scelta mira a differenziare chiaramente la sezione introduttiva dal resto del testo principale. Dopo questa fase preliminare, la numerazione prosegue con numeri arabi, partendo da 1. Questo sistema offre una chiara progressione nel corpo principale del lavoro.
Nel caso di appendici, la numerazione ritorna all'uso dei numeri romani, assegnando il numero I a ciascuna appendice. Se ci sono più appendici, ogni volta che ne viene completata una, si ricomincia la numerazione da I per la successiva. Questo approccio fornisce un'organizzazione chiara e logica per le appendici, garantendo che ciascuna sia distintamente identificata. L'uso coerente di numeri romani e arabi in diverse sezioni del documento contribuisce a una struttura ordinata e comprensibile per il lettore.

3.1.5.2 Intestazione e piè di pagina

In ciascuna pagina del documento, escludendo il frontespizio, sono presenti sia un'intestazione che un piè di pagina. Nell'intestazione, sono inclusi i seguenti elementi:

- Sul lato sinistro: il numero e il titolo del capitolo corrente;
- Sul lato destro: il logo del gruppo.

Nel piè di pagina, invece, sono indicati i seguenti dettagli:

- Sul lato sinistro: il nome del file;
- Al centro: il numero della pagina attualmente in consultazione;
- Sul lato destro: il numero di versione del file.

3.1.5.3 Frontespizio

Il frontespizio, ovvero la prima pagina del documento, strutturato nel seguente modo:

- **Logo UniPD**: il logo universitario è posizionato in alto a sinistra;
- **Informazioni sul corso**: le informazioni relative al corso di Ingegneria del Software sono in alto a destra
- **Logo del gruppo**: il logo del gruppo è posizionato in alto a sinistra subito sotto al logo dell'Università;
- **Nome gruppo e recapito**: le informazioni sul gruppo *SWEg Labs* sono posizionate in alto a destra, subito sotto alle info sul corso;
- **Titolo**: il titolo del documento è posizionato al centro della pagina, in grassetto;
- **Versione**: la versione del documento è posizionata al centro della pagina, appena sotto il titolo;
- **Tabella descrittiva**: posizionata centralmente sotto la versione del documento, riporta le seguenti informazioni:
  - **Stato**: lo stato del documento nel suo ciclo di vita;
  - **Redazione**: elenco dei membri del gruppo (nome e cognome) che hanno svolto la redazione del documento;
  - **Verifica**: elenco dei membri del gruppo (nome e cognome) che hanno svolto la verifica del documento;
  - **Approvazione**: elenco dei membri del gruppo (nome e cognome) che hanno svolto l'approvazione del documento;
  - **Proprietario**: il proprietario del documento, nel nostro caso tutto il gruppo *SWEg Labs*;
  - **Uso**: destinazione d'uso del documento (interno o esterno);
  - **Destinatari**: destinatari del documento.

3.1.5.4 Registro delle modifiche

Dopo la prima pagina si trova il registro delle modifiche. Tale registro va aggiornato ad ogni modifica effettuata, specificando per ognuna:

- **Versione**: la versione del documento in seguito alla modifica;
- **Data**: la data in cui è stata effettuata la modifica;
- **Descrizione**: una breve descrizione della modifica apportata;
- **Autore**: il nome e cognome dell'autore della modifica;
- **Verificatore**: il nome e cognome del verificatore della modifica, cioè colui che effettua la verifica del contenuto che è stato aggiungo, modificato o eliminato.

Le lettere di presentazione e i verbali, sia interni che esterni, non sono dotati del registro delle modifiche in quanto in seguito alla prima redazione poi non sono soggetti a modifiche future.

3.1.5.5 Indice

Tutti i documenti (tranne le lettere di presentazione per ragione di brevità) devono contenere un indice, collocato nella pagina successiva al registro delle modifiche. L'indice è utile per agevolare la consultazione del documento, riportando per ogni titolo di sezione (e sottosezione) del contenuto effettivo del documento la sua pagina iniziale. Ciascuna delle pagine del contenuto è identificata da un numero progressivo, partendo da 1. In caso di sottosezioni si segue il formato:

[numero sezione].[numero sottosezione]

Lo stesso formato vale per qualsiasi livello di annidamento delle sottosezioni.

3.1.5.6 Elenco delle figure

Nella pagina successiva all'indice è presente l'elenco delle figure. Esso riporta, per ogni figura che compare all'interno del documento, il suo titolo e la pagina in cui si trova. Come avviene per le sezioni, ciascuna figura è identificata da un numero progressivo, partendo da 1.

3.1.5.7 Elenco delle tabelle

Nella pagina successiva a quelle dedicate all'elenco delle figure è presente l'elenco delle tabelle. Esso riporta, per ogni tabella che compare all'interno del documento, il suo titolo e la pagina in cui si trova. Come avviene per le sezioni e per l'elenco delle figure, ciascuna tabella è identificata da un numero progressivo, partendo da 1.

3.1.5.8 Contenuto

Si tratta del contenuto effettivo del documento. Il contenuto deve essere suddiviso in sezioni e sottosezioni, ciascuna con il suo titolo in grassetto e numerato secondo i criteri descritti in precedenza.

3.1.5.9 Verbali

I verbali sono documenti che riportano le discussioni e le decisioni prese durante incontri. I verbali devono contenere le seguenti sezioni:

- **Informazioni generali**: contiene le informazioni riguardanti l'incontro, come la data, l'ora, il luogo e i partecipanti;
- **Ordine del giorno**: elenco degli argomenti che si era pianificato di trattare durante l'incontro;
- **Diario della riunione**: riassunto delle discussioni e delle azioni operate durante l'incontro;
- **Decisioni**: elenco delle decisioni prese durante l'incontro, ciascuna munita di codice identificativo per consentirne il tracciamento;
- **Todo**: elenco delle cose da fare emerse durante l'incontro, ciascuna collegata ad una o più decisioni. Ogni voce include un codice identificativo per il tracciamento nel backlog, specifica la/e decisione/i di origine e il responsabile assegnato.

3.1.6 Convenzioni

In seguito vengono riportate tutte le convenzioni che i documenti devono rispettare.

3.1.6.1 Nomi dei file

Tutti i nomi dei file devono seguire la convenzione dello *snake case*$_G$, cioè devono iniziare con una lettera minuscola e in caso di più parole ciascuna di esse deve essere separata tramite il carattere di underscore ("_"), tranne la data presente nei verbali, le cui parti devono essere separate da un trattino ("-"). Dopo il nome vero e proprio del file segue il numero della versione di quest'ultimo.

3.1.6.2 Stile del testo

Qui sono descritti tutti i diversi tipi di formattazione del testo usati nei documenti e i contesti nei quali vengono impiegati:

- **Grassetto**: viene utilizzato per evidenziare termini negli elenchi puntati e per i titoli delle sezioni;
- **Corsivo**: viene utilizzato per parole di particolare rilevanza all'interno del documento, per indicare il nome del gruppo (*SWEg Labs*), il nome dell'azienda *proponente*$_G$ (*AzzurroDigitale Srl*) e per le parole che si riferiscono al glossario (seguite da $_G$);
- **Link**: i link sono i collegamenti ipertestuali, ovvero collegamenti a fonti esterne al documento. Essi sono mostrati sottolineati e in grassetto, mentre vengono evidenziati in giallo al momento del passaggio del cursore.

3.1.6.3 Elenchi puntati

Gli elenchi puntati sono utilizzati per gli elenchi oppure per esprimere concetti in modo più diretto. Ciascuna voce di un elenco puntato è identificata da un simbolo, che varia a seconda del livello di profondità in cui si trova. In particolare:

- Un pallino per il primo livello;
  - Un trattino per il secondo livello;
    - Un asterisco per il terzo livello;
      - Un punto per il quarto livello.

Ogni voce inizia con la lettera maiuscola e termina con un punto e virgola (";"), eccezione fatta per l'ultima voce che termina con un punto (".").

3.1.6.4 Formato delle date

Per le date, viene adottato il seguente formato:

DD/MM/YYYY

dove:

- **DD**: indica il giorno con 2 cifre;
- **MM**: indica il mese con 2 cifre;
- **YYYY**: indica l'anno con 4 cifre.

3.1.6.5 Sigle

Una lista di sigle presente all'interno dei documenti è la seguente:

- **Ruoli**:
  - **Re**: Responsabile di Progetto;
  - **Am**: Amministratore di Progetto;
  - **An**: Analista;

  - **Pt**: Progettista;
  - **Pr**: Programmatore;
  - **Ve**: Verificatore.

- **Revisioni di progetto**:
  - **RTB**: *Requirements and Technology Baseline*$_G$;
  - **PB**: *Product Baseline*$_G$.

3.1.6.6 Tabelle

Le tabelle di ogni documento devono rispettare le seguenti convenzioni:

- Devono essere centrate orizzontalmente all'interno della pagina;
- Dopo ogni tabella segue una breve didascalia descrittiva accompagnata da un numero identificativo della stessa, incrementale e univoco all'interno del documento;
- Nelle celle che contengono solo uno 0 (zero), esso viene sostituito con un trattino per aumentarne la leggibilità.

3.1.6.7 Immagini

Ciascuna immagine, come le tabelle, deve essere centrata orizzontalmente all'interno della pagina ed è seguita da una breve didascalia descrittiva comprensiva di un numero che le identifica univocamente, incrementale all'interno del documento. Anche i grafici, *diagrammi di Gantt*$_G$ e *diagrammi UML*$_G$ sono inseriti nei documenti come immagini, pertanto seguono le stesse regole.

3.1.7 Strumenti

Per la redazione dei documenti, il gruppo *SWEg Labs* ha scelto di utilizzare i seguenti strumenti:

- ***LaTeX***$_G$: per la redazione dei documenti il gruppo ha scelto il linguaggio LaTeX. Si tratta di un linguaggio di markup basato su TeX per la produzione di documenti tecnici e scientifici di alta qualità;
- ***Visual Studio Code***$_G$: come editor di testo, il gruppo utilizza Visual Studio Code, preferito per la sua leggerezza, versatilità e il supporto per estensioni personalizzabili;
- ***GitHub***$_G$: GitHub è utilizzato come piattaforma di versionamento e repository per la documentazione. Consente una gestione collaborativa, permettendo il tracciamento delle modifiche e la revisione del testo in tempo reale;
- ***draw.io***$_G$: il team ha optato per Draw.io per disegnare i diagrammi UML necessari alla realizzazione del progetto didattico.
- ***Fogli Google***$_G$: i Fogli di calcolo di Google sono utilizzati dal team per creare e aggiornare i grafici richiesti dal progetto.

3.2 Gestione della configurazione

3.2.1 Scopo

L'obiettivo di questa sezione è delineare l'approccio adottato dal gruppo *SWEg Labs* nella gestione della configurazione, ossia la strategia scelta dal team per tenere traccia della documentazione redatta e del codice sviluppato.

3.2.2 Aspettative

Per quanto riguarda la gestione della configurazione, il gruppo ha le seguenti aspettative:

- Possibilità di tracciare tutte le modifiche apportate ai documenti o al codice;
- Possibilità di condivisione dei file tra i vari membri del gruppo;
- Possibilità di individuare e risolvere eventuali conflitti o errori;
- Possibilità di tornare ad una versione precedente.

3.2.3 Descrizione

L'obiettivo del processo di gestione della configurazione è garantire l'organizzazione e la tracciabilità della documentazione e del codice, creando una storia per ogni file prodotto. In particolare, si intende disporre i vari file all'interno di *repository*$_G$ facilmente accessibili e navigabili.

3.2.4 Versionamento

3.2.4.1 Codice di versione

Ad ogni modifica apportata ad un documento viene generata automaticamente una nuova versione per quest'ultimo. Ogni versione è identificata dal suo codice, nel formato

X.Y.Z

dove:

- **X**: rappresenta la versione dell'ultima approvazione da parte del responsabile;
- **Y**: rappresenta la versione dell'ultima approvazione generale da parte di un verificatore;
- **Z**: rappresenta la versione dell'ultima modifica, verificata da un verificatore.

Ogni approvazione comporta un incremento del numero di versione, che assume un peso diverso a seconda della posizione della cifra incrementata: i cambiamenti più importanti implicano un incremento della cifra più significativa.
Inoltre, un incremento ad una determinata cifra implica l'azzeramento di tutte le cifre alla sua destra.

3.2.4.2 Sistemi software utilizzati

Per la gestione delle repository *Git*$_G$ si è deciso di utilizzare la piattaforma online *GitHub*$_G$, in quanto tutti i membri del gruppo hanno già familiarizzato in precedenza con questo sistema software. Per quanto riguarda la gestione del *backlog*$_G$, si è deciso di utilizzare il servizio offerto da *Jira*$_G$, dal momento che permette di usare strumenti avanzati per la pianificazione, tracciamento e gestione dell'assegnazione delle varie attività.

3.2.5 Struttura delle repository

Con l'intento di organizzare al meglio il lavoro, si è optato per la creazione di due distinte repository pubbliche, al fine di mantenere separati i documenti organizzativi e il software:

- **Documentazione**: per il versionamento della documentazione;
- **BuddyBot**: per il versionamento del codice.

3.2.5.1 Documentazione

All'interno di questa repository sono presenti le cartelle relative alle principali *milestone*$_G$ del progetto, contenenti le rispettive *baseline*$_G$:

- **Candidatura**: in questa cartella sono presenti i documenti prodotti per la candidatura d'appalto per il Capitolato C9. Il contenuto di questa cartella è il seguente:

  - **Lettera di presentazione (v1.0.0)**: documento con cui il gruppo *SWEg Labs* si candida formalmente per la realizzazione del progetto commissionato;
  - **Preventivo dei costi e degli impegni orari (v1.0.0)**: documento che contiene l'impegno orario per ciascun membro del gruppo, il costo totale preventivato per la realizzazione del progetto e la scadenza di consegna del prodotto software. Tali conclusioni sono tratte da alcune considerazioni preliminari sui ruoli che i membri del gruppo dovranno svolgere e da previa consultazione con il proponente *AzzurroDigitale*$_G$;
  - **Valutazione dei Capitolati (v1.0.0)**: documento in cui il gruppo valuta le criticità riscontrate per ogni capitolato d'appalto e motiva la scelta del capitolato selezionato;
  - **Verbali**: cartella contenente tutti i verbali interni ed esterni (raccolti in apposite sottocartelle) relativi al periodo di candidatura;

- **RTB**: in questa cartella sono presenti i documenti prodotti per la candidatura alla *RTB*$_G$. Il contenuto è composto da:
  - **§Lettera di Presentazione**;
  - **Documentazione esterna**: sono contenuti i seguenti documenti:
    * **§Analisi dei Requisiti (v1.0.0)**;
    * **§Glossario**;
    * **§Piano di Progetto (v1.0.0)**;
    * **§Piano di Qualifica(v1.0.0)**;
    * **Verbali esterni**: cartella che raccoglie tutti i verbali esterni relativi al periodo RTB;
  - **Documentazione interna**: al suo interno sono contenuti:
    * **§Norme di Progetto (v1.0.0)**;
    * **Verbali interni**: cartella che raccoglie tutti i verbali interni relativi al periodo RTB;

3.2.5.2 BuddyBot

Questa repository contiene il codice sorgente relativo al *Proof of Concept* del progetto e le istruzioni per scaricarlo e testarlo in locale (*README.md*). Inoltre, è presente un file chiamato prompt_engineering.txt dove sono riportati tutti i tentativi operati per raggiungere un miglioramento dell'efficacia informativa delle risposte di BuddyBot, tra cui varie domande poste al chatbot e le corrispondenti risposte ottenute nel corso del tempo.

3.2.5.3 Gestione delle modifiche

Una buona suddivisione del lavoro tra i diversi documenti da redigere e le varie *feature*$_G$ da sviluppare contribuisce significativamente a ridurre i conflitti. L'obiettivo è mantenere il *branch*$_G$ principale (*main*) privo di errori, rendendolo inaccessibile a qualsiasi membro del gruppo fino all'approvazione del Responsabile di Progetto. Solo in seguito di tale approvazione è consentito effettuare il *merge*$_G$ di uno dei rami minori nel *main*.

3.3 Gestione della Qualità

3.3.1 Scopo

Questa sezione ha l'obbiettivo di descrivere le modalità con cui il gruppo intende misurare la qualità dei processi e del prodotto finale, al fine di soddisfare le aspettative del proponente.

3.3.2 Descrizione

Il processo di gestione della qualità stabilisce le norme e gli strumenti per misurare della qualità dei processi e del prodotto finale. Sono specificati gli obbiettivi prefissati, i metodi per raggiungerli e per verificarne il compimento.

3.3.3 Aspettative

- Soddisfare i requisiti di qualità richiesti dal proponente;
- Migliorare la qualità dei processi svolti dal gruppo;
- Valutare lo stato di avanzamento del progetto.

3.3.4 Piano di Qualifica

Per stabilire gli obbiettivi di questo processo utilizziamo il Piano di Qualifica. Ossia un documento contenente:

- Gli obbiettivi qualitativi da raggiungere;
- Le definizioni e descrizioni delle metriche utilizzate per la misurazione della qualità dei processi, del prodotto, della documentazione e dei software utilizzati;
- La descrizione dei test da svolgere e le relative norme da seguire;
- Lo stato attuale della qualità del progetto, misurato tramite le metriche descritte.

3.3.5 Strumenti

Gli strumenti utilizzati per la gestione della qualità sono le metriche. Queste sono le norme che regolano le metriche adottate, le quali rappresentano ognuna un conseguente obiettivo prefissato. Le metriche adottate sono presentate e descritte nel documento piano di qualifica, ognuna con i seguenti parametri:

- **Nome**: il nome della metrica
- **Codice**: ogni metrica ha un codice identificativo formato da:

  M[Tipologia][Id numerico]

  dove
  - **Tipologia** può essere:
    - **PC** per processo;
    - **PD** per prodotto.
  - **Id numerico** indica un numero univoco incrementale separato per le due tipologie.
- **Processo**: indica a che processo fa riferimento la metrica;
- **Formula**: come la metrica viene matematicamente calcolata (opzionale);
- **Descrizione**: spiegazione di cosa rappresenta la metrica e di come valutarla e interpretarla;
- **Valore di accettabilità**: indica il valore della metrica considerato accettabile;
- **Valore di preferibilità**: indica il valore ideale che dovrebbe essere assunto.

3.4 Verifica

3.4.1 Scopo

Questa sezione si propone di illustrare come il gruppo ha pianificato e implementato il processo di *verifica*$_G$, finalizzato a valutare il prodotto software. Tale processo è essenziale per garantire che il software soddisfi i requisiti definiti e rispetti le aspettative.

3.4.2 Aspettative

- Garantire l'ottenimento del prodotto finale in linea con le aspettative;
- Individuare tempestivamente eventuali errori, riducendo sprechi di risorse e tempo;
- Consentire al gruppo di progredire nello sviluppo in maniera più sicura e accurata.

3.4.3 Descrizione

Il processo di *verifica*$_G$ si basa su *test*$_G$ e analisi per individuare e correggere eventuali errori emersi durante lo sviluppo del software o la redazione della documentazione, assicurando la conformità ai requisiti. Per una verifica efficace, è fondamentale applicare procedure definite, criteri affidabili e una sequenza di fasi che preparino al passaggio successivo, ossia la validazione. Di seguito sono illustrate le principali attività svolte in questo processo.

3.4.4 *Analisi statica*$_G$

L'*analisi statica* prende il nome dal fatto che non richiede l'esecuzione del codice e si presta anche all'applicazione sulla documentazione. Questa attività consiste in una revisione dettagliata del codice e dei documenti per assicurarsi che rispettino i vincoli stabiliti, siano privi di difetti e soddisfino le caratteristiche richieste Il successo di questa analisi dipende in modo significativo dalla competenza e dall'attenzione dei verificatori coinvolti.

3.4.5 *Analisi dinamica*$_G$

L'*analisi dinamica* comprende tecniche che richiedono l'esecuzione del codice da verificare, rendendola applicabile esclusivamente al software, e non alla documentazione. Questa attività prevede l'esecuzione di *test*$_G$ per verificare il corretto funzionamento del software e identificare eventuali discrepanze tra i risultati attesi e quelli effettivi. Questo processo deve essere automatizzato e ripetibile. Per garantire l'efficacia dei test, ciascuno di essi deve essere deterministico, cioè, dati gli stessi input, deve produrre sempre lo stesso output. Inoltre, ogni test deve avere parametri ben definiti, tra cui:

- descrizione dei parametri di input;
- descrizione dell'output;
- comportamento atteso del software;
- condizioni di esecuzione del test.

Al fine di agevolare l'automazione dei test, saranno utilizzati i seguenti strumenti:

- **Driver:** una componente che fornisce l'ambiente di esecuzione necessario per testare un componente del prodotto software. Un driver è progettato per "attivare" il componente da testare e fornirgli i dati di input necessari;
- **Stub:** una componente passiva che simula il comportamento di alcune parti mancanti o non ancora implementate nell'applicazione. Questo consente di testare in anticipo le parti del software che dipendono da altre parti non ancora disponibili;
- **Logger:** uno strumento che registra i risultati ottenuti durante l'esecuzione del test, consentendo un'analisi accurata dei dati prodotti.

3.4.6 Test di sistema

I *Test di sistema*$_G$ hanno l'obiettivo di misurare la copertura dei requisiti specificati nel capitolato d'appalto. Essi sono definiti durante l'*Analisi dei Requisiti* e vengono svolti dopo i *test di integrazione*, dunque quando tutte le componenti del sistema software sono state integrate. Questo tipo di test è svolto anche durante il processo di validazione.

3.4.7 Test di accettazione

I *Test di accettazione*$_G$ servono per dimostrare che i requisiti individuati sono stati soddisfatti. Questo tipo di test deve essere svolto obbligatoriamente alla presenza del *committente*.

3.4.8 Codici relativi ai test

Il gruppo ha deciso di classificare i *test* che andranno svolti nell'attività di verifica associando un codice identificativo a ciascuno di essi nel formato

T[tipo][codice]

T[tipo][codice] dove:

- **[tipo]** rappresenta il tipo di test, in particolare:
  - S per i test di sistema;
  - A per i test di accettazione.
- **[codice]** è un numero progressivo associato al test all'interno del suo tipo.

3.4.9 Stato dei test

Ad ogni test verrà associato uno stato che ne rappresenterà l'esito. Lo stato può essere:

- N-I: il test non è ancora stato implementato;
- Superato: il test riporta esito positivo;
- Non superato: il test riporta esito negativo.

3.5 Validazione

3.5.1 Scopo

Questa sezione contiene le norme atte a regolare il processo di validazione, al fine di garantirne l'efficacia. Il processo di validazione permette di confermare l'effettivo conseguimento delle aspettative specificate dal proponente.

3.5.2 Aspettative

Tramite il processo di validazione il gruppo intende assicurarsi la conformità del prodotto software rispetto ai requisiti specificati nel documento di Analisi dei Requisiti.

3.5.3 Descrizione

La validazione si articola in più attività:

- **Istanziazione delle procedure**: vengono stabilite le procedure e le strategie da attuare per garantire una validazione efficace;
- **Istanziazione tei test di accettazione**: un test di accettazione consiste in una serie di operazioni eseguite sul prodotto software finale che rispondono positivamente solo se il software soddisfa tutti i requisiti specificati, garantendo che sia conforme alle aspettative del proponente. Tali test devono essere riportati nel documento *Piano di qualifica*;
- **Collaudo**: i test di accettazione vengono effettuati sul prodotto software in presenza del proponente, ma non prima di aver svolto con successo i test di sistema in un ambiente identico a quello di installazione.

4 Processi organizzativi

4.1 Scopo

Lo scopo di questa sezione è esporre le modalità e gli strumenti di coordinamento usati dal gruppo per la comunicazione interna ed esterna, definire e normare l'assegnazione di ruoli e compiti, e stabilire procedure e metodologie che il team seguirà durante il ciclo di vita del progetto.

4.2 Aspettative

Le aspettative in questa fase sono le seguenti:

- Redazione del documento *Piano di Progetto*$_G$;
- Definire i ruoli assunti dai membri del gruppo;
- Definizione di un piano per l'esecuzione dei compiti programmati.

4.3 Descrizione

Le attività previste da tale processo sono raccolte nel *Piano di Progetto*. In dettaglio viene trattata la gestione dei seguenti argomenti:

- Ruoli;
- Incontri;
- Comunicazioni;
- Metodo di sviluppo e gestione delle attività;
- Miglioramento
- Formazione;

4.4 Gestione dei Processi

4.4.1 Gestione dei ruoli

Il *Responsabile di Progetto* ha il compito di suddividere i ruoli e l'assegnazione oraria per i membri del gruppo, garantendo che ognuno di essi assuma, nel corso del progetto, almeno una volta ogni ruolo. Visto che nelle varie fasi di sviluppo del progetto le attività da svolgere variano, non sempre sarà necessario coprire tutti i ruoli. I ruoli richiesti dal progetto sono descritti qui di seguito.

4.4.1.1 Responsabile di Progetto

È il punto di riferimento per tutto il gruppo e per le comunicazioni con il *committente* e l'*azienda proponente*. Si assume la responsabilità delle scelte del gruppo dopo averle approvate e coordina le azioni dei vari membri del team. È responsabile della pianificazione, dell'esecuzione e del monitoraggio delle attività del progetto secondo le specifiche previste dal gruppo. Le sue principali responsabilità includono:

- **Pianificazione e Definizione degli Obiettivi**: pianificare e definire obiettivi, scadenze e risorse necessarie per il progetto;
- **Coordinamento del Team e Gestione delle Risorse Umane**: coordinare il lavoro del team, gestire le risorse umane, assegnare i compiti agli altri membri del gruppo e garantire che ogni membro del gruppo sappia cosa ci si aspetta da lui;
- **Monitoraggio del Progresso e Gestione dei Rischi**: monitorare il progresso del progetto, assicurandosi che le attività pianificate vengano svolte entro i tempi e le modalità previste, e identificare, studiare e gestire i rischi associati al progetto;
- **Comunicazione**: comunicare con il team, le parti interessate, il *committente* e l'*azienda proponente*. Gestire le comunicazioni e gli incontri interni, nonché le relazioni esterne;

- **Approvazione della Documentazione**: approvare la documentazione prodotta dal gruppo, l'offerta economica sottoposta al committente e qualsiasi *task* completata e verificata;
- **Suddivisione delle Attività**: suddividere le attività del gruppo in singole *issue*$_G$, e gestire la loro assegnazione ai membri del gruppo.

4.4.1.2 Amministratore

È incaricato del controllo e dell'amministrazione di tutto l'ambiente di lavoro. Le sue principali responsabilità includono:

- **Gestione della Documentazione**: salvaguardare la documentazione di progetto e gestire il sistema di archiviazione e versionamento di documentazione e codice;
- **Gestione degli Strumenti e dell'Infrastruttura**: gestire l'infrastruttura e gli strumenti utilizzati, mantenere efficiente l'ambiente di sviluppo e fornire strumenti adeguati ai membri del gruppo. Gestire errori e segnalazioni di malfunzionamenti con gli strumenti tecnologici;
- **Automazione e Miglioramento dei Processi**: automatizzare i processi, individuare punti di miglioramento nei processi e mettere in opera risorse per migliorare l'ambiente di lavoro;
- **Gestione della Configurazione e Versionamento**: effettuare il controllo di versioni e configurazioni del prodotto software, gestire il sistema di configurazione e *versionamento*$_G$ del prodotto;
- **Gestione della Qualità**: redigere e attuare i piani e le procedure per la gestione della qualità, garantendo l'*efficacia*$_G$ e l'*efficienza*$_G$ dei processi;
- **Supporto e Comunicazione**: gestire le comunicazioni interne, supportare la gestione delle risorse e delle comunicazioni del progetto.

4.4.1.3 Analista

L'analista è la figura responsabile dell'analisi dei requisiti del progetto e della definizione delle specifiche. Le sue principali responsabilità includono:

- **Raccolta e Analisi dei Requisiti**: raccogliere e analizzare i requisiti del cliente, trasformando i bisogni del *proponente* nelle aspettative che il gruppo deve soddisfare per sviluppare un prodotto professionale;
- **Definizione delle Specifiche**: definire le *specifiche funzionali*$_G$ e *tecniche*$_G$ del prodotto, modellare concettualmente il sistema e suddividere i requisiti del progetto in categorie;
- **Collaborazione con il Progettista**: collaborare con il progettista per creare soluzioni che soddisfino i requisiti definiti;
- **Documentazione**: redigere l'*Analisi dei Requisiti*$_G$, documentando i servizi che il sistema deve fornire e garantendo che i requisiti siano chiari e completi;
- **Studio del Problema e del Contesto Applicativo**$_G$: studiare il problema e il relativo *contesto applicativo*, comprendere la complessità del problema e definire i requisiti impliciti ed espliciti;
- **Gestione della Qualità**: assicurare che i requisiti siano ben definiti per evitare problemi durante la fase di progettazione e sviluppo.

4.4.1.4 Progettista

È responsabile di modellare i requisiti individuati nella fase di analisi e ricomporli in un'*architettura*$_G$ che possa soddisfarli. Le sue principali responsabilità includono:

- **Definizione dell'Architettura**: creare un'*architettura* del sistema che soddisfi i requisiti richiesti, con un alto livello di manutenibilità e un basso grado di *accoppiamento*$_G$ tra i componenti;
- **Scelte Tecniche e Tecnologiche**: effettuare scelte riguardanti gli aspetti tecnici e tecnologici del progetto, favorendo l'*efficacia* e l'*efficienza*. Scegliere eventuali *pattern architetturali*$_G$ da implementare e sviluppare lo *schema UML*$_G$ delle classi;

- **Qualità del Prodotto**: garantire la qualità del prodotto, producendo una soluzione economica e mantenibile che rientri nei costi stabiliti nel preventivo;
- **Collaborazione e Coordinamento**: collaborare con gli analisti per comprendere i requisiti e con i programmatori per l'implementazione delle soluzioni tecniche. Coordinare le attività di progettazione per assicurare che il prodotto finale soddisfi tutti i requisiti;
- **Ricerca e Innovazione**: approfondire le conoscenze tecniche e ricercare strumenti tecnologici utili nell'ambito di applicazione, migliorando continuamente l'*architettura* del sistema.

4.4.1.5 Programmatore

Il programmatore è responsabile della scrittura del codice e dell'implementazione delle soluzioni tecniche definite dal progettista. Le sue principali responsabilità includono:

- **Scrittura del Codice**: scrivere codice che soddisfi le specifiche di progettazione, applicando le "*best practices*$_G$" note riguardanti la scrittura di codice;
- **Risoluzione dei Problemi Tecnici**: risolvere i problemi tecnici che emergono durante lo sviluppo;
- **Test del Codice**: testare il proprio codice per garantire che funzioni correttamente e soddisfi i requisiti;
- **Manutenibilità del Codice**: scrivere codice documentato, mantenibile e versionato, rendendo l'attività di verifica semplice e agevole;
- **Documentazione**: redigere le documentazioni necessarie per la comprensione e l'uso del codice;
- **Collaborazione**: collaborare con il progettista e altri membri del team per implementare l'architettura prodotta nella fase di progettazione.

4.4.1.6 Verificatore

Il verificatore è responsabile del controllo del lavoro svolto dagli altri componenti del gruppo, assicurando che sia conforme agli standard di qualità e alle norme di progetto. Le sue principali responsabilità includono:

- **Controllo della Conformità**: verificare che ogni file caricato in un *branch*$_G$ protetto della *repository*$_G$ sia conforme alle *Norme di Progetto*$_G$, controllando i file modificati o aggiunti durante una *pull request*$_G$ per errori ortografici, sintattici, logici e di *build*$_G$;
- **Esame dei Prodotti**: esaminare i prodotti in fase di revisione utilizzando le tecniche e gli strumenti definiti nelle *Norme di Progetto*, assicurandosi che siano conformi ai requisiti funzionali e di qualità;
- **Segnalazione degli Errori**: segnalare eventuali errori riscontrati durante la *verifica*, fornendo *feedback*$_G$ completi e chiari a chi ha prodotto il lavoro revisionato;
- **Assicurazione della Qualità**: assicurare che la qualità di quanto prodotto sia conforme agli standard imposti, verificando la conformità di ogni stadio del *ciclo di vita*$_G$ del prodotto;
- **Documentazione**: redigere la parte retrospettiva del *Piano di Qualifica*$_G$, descrivendo le verifiche e le prove effettuate;
- **Sorveglianza Continua**: essere presente durante l'intera durata del progetto, garantendo che le attività svolte rispettino il livello di qualità atteso.

4.4.2 Gestione degli Incontri

Le riunioni si dividono in interne ed esterne. Per ogni riunione verrà redatto un verbale per tenere traccia degli argomenti trattati e delle decisioni prese. La pratica di redigere verbali favorisce la trasparenza e la tracciabilità delle decisioni, costituendo uno strumento prezioso per mantenere un registro chiaro e condiviso delle attività svolte.

4.4.2.1 Riunioni Interne

Le riunioni interne si svolgono esclusivamente tra i membri del gruppo e avvengono almeno una volta a settimana. Un incontro può essere richiesto da qualsiasi membro del gruppo al *Responsabile di Progetto*, che lo organizzerà in base alla disponibilità di tutti i membri. La modalità di incontro è prevalentemente virtuale, utilizzando come piattaforma di riferimento *Discord*$_G$. Per mantenere una buona efficienza e ottimizzare il tempo, ogni riunione interna segue una linea guida:

- **Preparazione**: Prima di ogni incontro, viene preparata una scaletta dei principali punti da discutere;
- **Discussione**: Discussione del lavoro svolto da ogni membro del gruppo dall'ultimo incontro e dei punti prefissati nella scaletta iniziale, con confronto su eventuali dubbi;
- **Pianificazione**: Pianificazione delle attività da svolgere per ogni membro del gruppo fino al prossimo incontro.

Alla fine di ogni incontro viene scelto un membro del gruppo incaricato di redigere il verbale interno con una breve descrizione dei punti focali dell'incontro.

4.4.2.2 Riunioni Esterne

Le riunioni esterne coinvolgono i membri del gruppo e i referenti aziendali. Questi incontri sono previsti di giovedì pomeriggio (dalle 17.00 alle 18.00) con cadenza bisettimanale nella prima parte di sviluppo del progetto, indicativamente fino al superamento della revisione *RTB*$_G$, per poi passare ad una cadenza settimanale fino alla consegna definitiva del progetto. È possibile per i rappresentanti di entrambe le parti richiedere incontri aggiuntivi se necessario. Le riunioni avvengono principalmente in modalità virtuale, utilizzando come piattaforma di riferimento *Google Meet*$_G$. Alla fine di ogni incontro esterno, viene redatto un verbale per documentare i momenti salienti e le decisioni prese e viene sottoposto ai referenti aziendali per approvazione.

4.4.3 Gestione delle Comunicazioni

Le comunicazioni per tutta la durata del progetto si dividono in interne ed esterne. Ogni canale di comunicazione è dedicato a specifiche esigenze per garantire efficienza e chiarezza.

4.4.3.1 Comunicazioni Interne

Le comunicazioni interne riguardano esclusivamente i membri del gruppo e avvengono attraverso i seguenti strumenti:

- ***Telegram***$_G$: Utilizzato per comunicazioni rapide e istantanee, sia testuali che vocali, e per la pianificazione degli incontri interni;
- **Discord**: Utilizzato per le riunioni interne e per comunicazioni veloci tra i membri del gruppo. *Discord* permette una comunicazione sincrona vocale affidabile e comoda per tutti i membri del gruppo.

4.4.3.2 Comunicazioni Esterne

Le comunicazioni esterne sono gestite dal *Responsabile di Progetto* e coinvolgono il proponente e i committenti. Avvengono attraverso i seguenti canali:

- **Google Meet**: Utilizzato per le riunioni esterne con il proponente e i committenti. Queste riunioni possono essere richieste da entrambe le parti e al termine viene redatto un verbale;
- **Discord**: Utilizzato per comunicazioni veloci con i referenti aziendali;
- **Email**: Utilizzata per comunicazioni formali e dettagliate. L'indirizzo email del gruppo è accessibile a tutti i membri del team per garantire una comunicazione chiara e trasparente.

4.4.3.3 Compiti del Responsabile del Progetto

Il *Responsabile di Progetto* si occupa di:

- Pianificare l'ordine del giorno delle riunioni;
- Comunicare tempestivamente eventuali variazioni orarie;
- Verificare la presenza dei membri durante le riunioni;
- Guidare le discussioni in modo ordinato;
- Nominare un segretario per redigere il verbale;
- Approvare formalmente il verbale.

4.4.3.4 Doveri dei partecipanti

I Partecipanti si impegnano a:

- Partecipare puntualmente alle riunioni;
- Comunicare tempestivamente eventuali ritardi o assenze;
- Partecipare attivamente durante le riunioni;
- Mantenere un comportamento consono durante le riunioni.

4.4.4 Gestione compiti e task

La gestione dei compiti e task è cruciale per garantire un'organizzazione efficace delle attività, mantenendo un alto livello di tracciabilità, qualità e aderenza alle tempistiche pianificate. Questa sezione descrive le metodologie adottate dal team e il ciclo di vita di ciascun task, che ne scandisce lo svolgimento dall'ideazione alla chiusura.

4.4.4.1 Metodologie e pratiche

Il team utilizza il metodo di sviluppo *Agile*$_G$ per organizzare e pianificare le attività progettuali. Questo approccio consente una gestione flessibile e iterativa, promuovendo una costante collaborazione tra i membri del team e una comunicazione trasparente con il committente. Le principali pratiche includono:

- **Organizzazione attraverso Sprint**$_G$: ogni ciclo di lavoro è scandito da sprint di durata predefinita, con obiettivi chiari e misurabili;
- **Collaborazione strutturata**: i task vengono definiti, pianificati e assegnati utilizzando *Jira*$_G$, cioè uno strumento di *Issue Tracking System*$_G$ che facilita la condivisione delle responsabilità e il lavoro di gruppo;
- **Monitoraggio continuo**: board visive, roadmap e *diagrammi di Gantt*$_G$ offrono una visione d'insieme sullo stato delle attività, permettendo di identificare rapidamente eventuali blocchi o ritardi;
- **Revisione e miglioramento iterativo**: al termine di ogni sprint, il team valuta i risultati ottenuti e pianifica miglioramenti per il ciclo successivo.

Questo approccio garantisce flessibilità nell'adattarsi a cambiamenti nei requisiti e favorisce il coinvolgimento attivo di tutti i partecipanti.

4.4.4.2 Ciclo di vita di un Task

Il *ciclo di vita*$_G$ dei task si articola in sei fasi principali, gestite attraverso *Jira*, che funge da strumento digitale configurato per garantire efficienza e coerenza.

- **Creazione**: il *Responsabile* definisce i task su *Jira*, completando i campi essenziali (titolo, descrizione, priorità, assegnatari, stima temporale) e assegnandoli ad una sprint; checklist e tag possono essere aggiunti per migliorare l'organizzazione;
- **Assegnazione**: i task vengono distribuiti ai membri in base alle competenze e al carico di lavoro. Nei casi in cui un task rimanga non assegnato, i membri possono prendersene carico autonomamente;
- **Esecuzione**: l'esecutore del task ne aggiorna lo stato (es. da "To Do" a "In Progress") e lavora su un branch dedicato per garantire una gestione separata e tracciabile delle modifiche;
- **Revisione**: una volta completato, il task passa in revisione. Il *Verificatore* esamina il lavoro, segnala eventuali modifiche necessarie tramite commenti o review e valida il task se conforme agli standard;
- **Accettazione**: dopo la revisione, il *Responsabile* approva il task, esegue il merge del *branch* (se pertinente) e aggiorna lo stato del task a "Completato".

Questa gestione strutturata dei task consente al team di affrontare le attività con precisione, tracciando ogni modifica e garantendo una visione chiara e condivisa dell'avanzamento del progetto.

4.4.5 Miglioramento

Scopo: durante lo svolgimento delle attività e la successiva elaborazione della documentazione, ci poniamo l'obiettivo di seguire costantemente il principio di miglioramento continuo. L'obiettivo principale è identificare proattivamente le attività, i ruoli e le opportunità di miglioramento, cercando nuove soluzioni per affrontare sfide emergenti o passate.

Descrizione: il miglioramento continuo è un ciclo iterativo che ci consente di adattarci dinamicamente alle esigenze mutevoli del progetto, garantendo una crescita continua e una maggiore efficienza nelle nostre attività. Questo approccio contraddistinguerà ogni fase del processo, dalla pianificazione all'esecuzione, compresa la documentazione.

Implementazione: durante lo svolgimento delle attività e la stesura dei documenti, il team si impegna a seguire il principio di miglioramento continuo. Le problematiche riscontrate e le soluzioni adottate vengono documentate e confrontate per garantire una gestione consapevole e mirata delle sfide incontrate durante il processo di sviluppo. Questo approccio formale è stato adottato per evitare di ripetere errori già fatti e fornire soluzioni efficaci.

4.4.6 Formazione

Scopo: l'obiettivo di questa sezione è stabilire le regole relative al processo di istruzione dei membri del team, che include lo studio delle tecnologie impiegate per la produzione dei documenti e la realizzazione del prodotto richiesto. **Aspettative**: le aspettative per il processo di istruzione includono:

- Acquisire una solida conoscenza del linguaggio LaTeX$_G$;
- Ottenere una buona dimestichezza con i diversi linguaggi di programmazione, le librerie e gli strumenti necessari per la realizzazione del prodotto software assegnato dal proponente;
- Sviluppare una buona familiarità con l'ambiente in cui si sta lavorando relativamente al capitolato di interesse.

Piano di Formazione: ogni componente del gruppo è libero di formarsi nel modo che ritiene più opportuno, sostenendo il processo di apprendimento "learning by doing". Questa libertà permette di ampliare la visione del gruppo riguardo le tecnologie utilizzate, aumentando la consapevolezza e migliorando complessivamente le scelte implementative.

A Standard ISO/IEC 12207

L'*ISO/IEC/IEEE 12207*$_G$ definisce un framework completo per il ciclo di vita del software, identificando processi, attività e compiti necessari per sviluppare, gestire e mantenere software in modo sistematico. Implementare questo standard permette di mantenere un alto livello di qualità, tracciabilità e coerenza nei processi di sviluppo software. Fornisce un quadro di riferimento robusto per gestire i progetti, favorendo il miglioramento continuo e il rispetto delle tempistiche e dei requisiti iniziali.
Lo standard si articola in tre principali categorie di processi:

- Processi primari
- Processi di supporto
- Processi organizzativi

A.1 I processi primari

I processi primari comprendono quelli necessari alla gestione e allo sviluppo del software, includendo sia la gestione del progetto sia il supporto tecnico.

- **Processo di acquisizione**: Coinvolge tutte le attività di acquisizione del software o di servizi legati al software, come la definizione dei requisiti dell'acquirente, la selezione del fornitore, la gestione del contratto e il monitoraggio del progresso.
- **Processo di fornitura**: Descrive le attività del fornitore per sviluppare, modificare e consegnare il software in base alle specifiche di contratto. Include la pianificazione, lo sviluppo, la consegna e la gestione del software.
- **Processo di sviluppo**: Definisce le attività per creare o modificare il software. Le fasi principali sono:
  - **Analisi dei requisiti**: Identificazione delle specifiche funzionali e non funzionali.
  - **Progettazione del sistema e del software**: Progettazione dell'architettura e della struttura del software.
  - **Implementazione**: Programmazione del codice sorgente.
  - **Integrazione**: Assemblaggio delle diverse parti del software.
  - **Testing**: Verifica del software per assicurare che rispetti i requisiti definiti.
  - **Installazione e accettazione**: Rilascio del software e conferma della sua aderenza ai requisiti.
- **Processo di gestione operativa**: Si occupa delle attività di gestione del software nel suo ambiente di produzione, incluse la manutenzione, il supporto utente, l'operatività continua e il monitoraggio delle performance.
- **Processo di manutenzione**: Comprende tutte le attività per correggere, migliorare o adattare il software dopo il rilascio, al fine di mantenerne o aumentarne l'efficienza e la rilevanza.

A.2 I processi di supporto

Questi processi assistono i processi primari e garantiscono che il software sia conforme agli standard di qualità.

- **Processo di documentazione**: Definisce la creazione, la gestione e la manutenzione della documentazione di progetto.
- **Processo di configurazione**: Gestisce le versioni del software, tenendo traccia delle modifiche e assicurando che ogni versione sia stabile e rintracciabile.
- **Processo di verifica**: Garantisce che ogni fase dello sviluppo rispetti i requisiti iniziali attraverso attività di verifica e review, che coinvolgono sia il codice che la documentazione.
- **Processo di validazione**: Assicura che il software finale soddisfi le esigenze dell'utente, attraverso attività di testing e collaudo in condizioni reali.

- **Processo di qualità**: Definisce le attività di controllo qualità per monitorare la conformità agli standard e migliorare continuamente i processi di sviluppo.
- **Processo di revisione e audit**: Prevede revisioni periodiche e audit di progetto per identificare eventuali problemi o non conformità rispetto agli standard definiti.

A.3 I processi organizzativi

I processi organizzativi mirano a sostenere e migliorare i processi aziendali nel loro insieme, creando un ambiente di supporto che faciliti l'attività dei team.

- **Processo di gestione**: Include tutte le attività di pianificazione, coordinamento e monitoraggio del progetto, come l'assegnazione delle risorse, la definizione del budget e la gestione del rischio.
- **Processo di miglioramento**: Riguarda le attività di analisi e ottimizzazione dei processi, come la raccolta di feedback, l'identificazione delle aree di miglioramento e la messa in atto di strategie di ottimizzazione.
- **Processo di formazione**: Definisce la formazione continua per il personale, migliorando le competenze tecniche e manageriali necessarie a svolgere i vari processi.
- **Processo di gestione delle risorse umane**: Include le attività di selezione, valutazione e gestione delle risorse umane impiegate nei vari processi.

B Standard di qualità ISO/IEC 9126

Lo standard per la valutazione della qualità che il gruppo *SWEg Labs* ha adottato è l'*ISO/IEC 9126*$_G$, il quale definisce diverse caratteristiche per garantire che il prodotto finale soddisfi le esigenze dell'utente e del destinatario finale.
I criteri per la valutazione e misurazione della qualità si basano sui seguenti punti:

- Funzionalità;
- Affidabilità;
- Usabilità;
- *Efficienza*$_G$;
- Manutenibilità;
- Portabilità.

B.1 Funzionalità

Questa caratteristica fa riferimento alle prestazioni del software rispetto alle specifiche funzionali previste, cioè la capacità del *prodotto software*$_G$ di fornire funzionalità che soddisfino i requisiti stabiliti (o implicitamente dedotti). La Funzionalità si articola nelle seguenti sotto-caratteristiche:

- **Adeguatezza**: capacità del software di fornire un insieme di funzioni che soddisfino i requisiti specificati e le esigenze dell'utente;
- **Accuratezza**: capacità del software di fornire i risultati attesi con il grado di precisione richiesto;
- **Interoperabilità**: capacità del software di interagire con uno o più sistemi specificati;
- **Sicurezza**: capacità del software di proteggere informazioni e dati da accessi non autorizzati;
- **Aderenza alla funzionalità**: capacità del software di aderire a standard, convenzioni e regolamentazioni specificate.

B.2 Affidabilità

Questa caratteristica si riferisce alla capacità del software di operare con affidabilità nel corso del tempo, cercando di ridurre il più possibile interruzioni o errori in circostanze specifiche. L'affidabilità si compone delle seguenti sotto-caratteristiche:

- **Maturità**: capacità del software di evitare errori, malfunzionamenti o guasti in condizioni specificate;
- **Tolleranza agli errori**: capacità del software di mantenere un livello specificato di prestazioni in caso di errori software o di violazioni della sua specifica;
- **Recuperabilità**: capacità del software di ripristinare il livello appropriato di prestazioni e di recupero dei dati in caso di malfunzionamenti;
- **Aderenza all'affidabilità**: capacità del software di aderire a standard, convenzioni e regolamentazioni specificate.

B.3 Usabilità

L'usabilità è un aspetto cruciale che influenza l'esperienza dell'utente nell'interazione con il software. Le sotto-caratteristiche di usabilità delineate nell'*ISO/IEC 9126*$_G$ contribuiscono a valutare l'*efficacia*$_G$, l'*efficienza*$_G$ e la soddisfazione dell'utente. Queste sotto-caratteristiche sono:

- **Comprensibilità**: la comprensibilità indica la facilità con cui gli utenti possono capire e comprendere il funzionamento del software. Un'interfaccia chiara e intuitiva favorisce la rapida comprensione delle funzionalità, riducendo la curva di apprendimento per gli utenti;

- **Curva d'apprendimento**: riflette la facilità con cui gli utenti possono imparare ad utilizzare il software. Un software con una buona curva d'apprendimento consente agli utenti di acquisire familiarità con le sue funzionalità in modo rapido ed efficiente;
- **Operabilità**: l'operabilità valuta la facilità con cui gli utenti possono interagire con il software per eseguire le operazioni desiderate. Un'interfaccia ben progettata e accessibile contribuisce a un'esperienza utente più efficiente e soddisfacente;
- **Apparenza**: l'apparenza si riferisce all'aspetto visivo del software e alla sua capacità di suscitare interesse ed essere piacevole per gli utenti. Un'interfaccia accattivante può migliorare la percezione complessiva del software e influenzare positivamente l'esperienza utente;
- **Conformità**: la conformità riguarda la coerenza del software rispetto alle convenzioni di usabilità e ai principi di design. Un'applicazione conforme alle *best practices*$_G$ di usabilità tende a fornire un'esperienza più uniforme e prevedibile agli utenti.

B.4 Efficienza

L'efficienza rappresenta la capacità del software di eseguire le proprie funzioni in modo rapido ed economico, utilizzando in modo ottimale le risorse disponibili. L'efficienza la possiamo individuare mediante:

- **Comportamento rispetto al tempo**: questa sotto-caratteristica valuta la tempestività con cui il software risponde alle richieste degli utenti. Un'applicazione efficiente è in grado di fornire risposte rapide, riducendo al minimo i tempi di attesa;
- **Utilizzo delle risorse**: l'efficienza nelle risorse misura quanto il software sia in grado di utilizzare in modo ottimale le risorse di sistema, come CPU, memoria e spazio di archiviazione. Un'applicazione efficiente è in grado di eseguire le proprie funzioni senza spreco eccessivo di risorse, garantendo prestazioni elevate e affidabili.

B.5 Manutenibilità

La manutenibilità è una caratteristica chiave nel valutare la qualità del software, concentrandosi sulla facilità con cui è possibile apportare modifiche al sistema, correggere errori e migliorare le prestazioni nel tempo. Per permettere una facile manutenzione del software, alcuni aspetti fondamentali da mantenere in considerazione sono:

- **Analizzabilità**: l'analizzabilità valuta quanto sia agevole comprendere la struttura del codice sorgente e individuare eventuali errori. Un software che già presenta un'approfondita Analisi dei Requisiti semplifica il processo di ispezione e correzione, accelerando le attività di manutenzione;
- **Modificabilità**: questa sotto-caratteristica riflette la facilità con cui è possibile apportare modifiche al software, aggiungere nuove funzionalità o adattarlo a nuovi requisiti. Un sistema altamente modificabile è più flessibile e reattivo agli aggiornamenti;
- **Stabilità**: la stabilità indica la capacità del software di mantenere la coerenza delle prestazioni anche dopo l'introduzione di modifiche. Un'applicazione stabile minimizza gli effetti collaterali delle modifiche, garantendo che le nuove funzionalità non compromettano l'integrità del sistema;
- **Testabilità**: la testabilità permette di misurare quanto sia agevole verificare e validare le modifiche apportate al software. Un'applicazione con un'elevata copertura dei test semplifica il processo di identificazione e risoluzione di problemi, facilitando il mantenimento del software nel tempo.

B.6 Portabilità

La portabilità si riferisce alla capacità del software di essere trasferito o adattato facilmente a diversi ambienti, sistemi operativi o architetture senza perdere le sue funzionalità e prestazioni. Queste capacità possono essere individuate come:

- **Adattabilità**: l'adattabilità indica quanto il software può essere modificato per adattarsi a nuovi ambienti o requisiti. Un'applicazione adattabile è in grado di operare senza problemi su diverse piattaforme, consentendo una maggiore flessibilità operativa;

- **Installabilità**: questa sotto-caratteristica riflette la facilità con cui il software può essere installato in diversi ambienti. Un'applicazione facilmente installabile semplifica il processo di distribuzione e implementazione su varie configurazioni di sistema;
- **Conformità**: la conformità si riferisce alla capacità del software di rispettare gli standard e i protocolli di interoperabilità. Un'applicazione conforme è in grado di interagire in modo coerente con altri sistemi e applicazioni, facilitando l'integrazione in ambienti eterogenei;
- **Sostituibilità**: la sostituibilità valuta quanto sia agevole sostituire il software con un'altra soluzione equivalente. Un'applicazione sostituibile agevola il processo di migrazione verso nuove tecnologie o soluzioni senza eccessivi sforzi di adattamento.

C Metriche per la qualità

La valutazione della qualità del software è un *processo*$_G$ complesso che coinvolge diverse categorie di *metriche*$_G$. Esse sono suddivise principalmente in quattro macro-aree: metriche interne, metriche esterne, metriche della qualità in uso e metriche per la qualità di processo. Ogni categoria ha il suo scopo specifico nel valutare aspetti diversi del *ciclo di vita del software*$_G$.

C.1 Metriche interne

Le metriche interne sono utilizzate per valutare il codice sorgente e la documentazione intermedia durante lo sviluppo del software. Esse forniscono indicazioni sulla qualità interna del prodotto e possono aiutare a prevenire potenziali problemi futuri. Queste metriche sono spesso utilizzate durante il processo di sviluppo per migliorare la qualità del codice e facilitare la manutenzione.

C.2 Metriche esterne

Le metriche esterne sono orientate verso gli utenti finali e valutano il comportamento del software in esecuzione. Esse misurano le caratteristiche che sono visibili agli utenti, come le prestazioni e la facilità d'uso. Queste metriche sono essenziali per garantire che il software soddisfi le aspettative degli utenti e fornisca un'esperienza positiva.

C.3 Metriche della qualità in uso

Le metriche della qualità in uso valutano come gli utenti effettivamente interagiscono e sfruttano il software nell'ambiente operativo. Misurano l'*efficacia*$_G$, l'*efficienza*$_G$, la soddisfazione dell'utente e altri aspetti che influenzano direttamente l'esperienza dell'utente durante l'utilizzo del software.

C.4 Metriche per la qualità di processo

Le metriche per la qualità di processo valutano la qualità dei processi adottati durante lo sviluppo del software. Esse includono metriche per miglioramento, fornitura, *codifica*$_G$ e documentazione, fornendo indicazioni su come i processi possono essere ottimizzati per migliorare la qualità del prodotto finale:

- **Miglioramento**: queste metriche valutano l'efficacia dei processi di miglioramento continuo implementati nel ciclo di vita dello sviluppo del software.
- **Fornitura**: le metriche di fornitura misurano la qualità del processo di consegna del software, compreso il rispetto dei tempi, la gestione delle risorse e la conformità agli standard.
- **Codifica**: queste metriche valutano la qualità del processo di scrittura del codice, inclusa la correttezza sintattica, la chiarezza e la conformità agli standard di codifica.
- **Documentazione**: misurano la qualità della documentazione associata al software, fornendo indicazioni sulla chiarezza e completezza della documentazione.

C.5 Metriche di qualità di prodotto

Le metriche per la qualità di prodotto valutano le caratteristiche del software come funzionalità, usabilità, manutenibilità e altre. Queste metriche forniscono indicazioni specifiche sulla qualità del prodotto software in termini di conformità agli standard e soddisfazione degli utenti.

- **Funzionalità**: queste metriche valutano la completezza e la correttezza delle funzionalità del software, assicurando che risponda adeguatamente ai requisiti
- **Usabilità**: misurano la facilità con cui gli utenti possono interagire con il software, considerando aspetti come la comprensibilità, l'apprendibilità e l'operabilità.

C.6 Manutenibilità

La manutenibilità è una categoria specifica che riflette la facilità con cui il software può essere modificato, corretto e adattato nel tempo. Essa include metricheG di affidabilità, valutando la capacità del software di mantenere prestazioni stabili e coerenti anche dopo le modifiche.

- **Affidabilità**: Questa metrica valuta la capacità del software di mantenere la coerenza delle prestazioni anche dopo l'introduzione di modifiche, assicurando che nuove funzionalità non compromettano l'integrità del sistema.