torchlight

torchlight.swe2324@outlook.com

Norme di Progetto

Versione 1.0.0

Redattori	Agafitei Ciprian
	Cappellari Marco
	De Laurentis Arianna Pia
	Filippini Giovanni
	Meneghini Fabio
	Pluzhnikov Dmitry
	Ye Tao Ren Federico
Verifica	Agafitei Ciprian
	Cappellari Marco
	De Laurentis Arianna Pia
	Filippini Giovanni
	Meneghini Fabio
	Pluzhnikov Dmitry
	Ye Tao Ren Federico
Approvazione	Filippini Giovanni
Uso	Interno
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo *torchlight*

Registro delle Modifiche

Ver.	Data	Descrizione	Autore	Verifica
1.0.0	2024/01/15	Verifica finale e convalida del documento	Filippini Giovanni	Filippini Giovanni
0.4.7	2024/01/09	Controllati i termini del glossario e piccole correzioni ortografiche	Filippini Giovanni	Cappellari Marco
0.4.6	2023/12/13	Completata la sezione §2.2.6	Ye Tao Ren Federico	Meneghini Fabio
0.4.5	2023/12/10	Migliorata la sezione §3.2.5.1	Pluzhnikov Dmitry	Filippini Giovanni
0.4.4	2023/11/6	Completata la sezione §2.2.5, aggiunta la sezione §2.2.6	Pluzhnikov Dmitry	Filippini Giovanni
0.4.3	2023/12/4	Completata la sezione §2.2.4, aggiunta la sezione §2.2.5	Pluzhnikov Dmitry	Filippini Giovanni
0.4.2	2023/12/3	Completata la sezione §3.4, aggiunta la sezione §3.1.6.4	Meneghini Fabio	Cappellari Marco
0.4.1	2023/12/2	Completate le sezioni §2.2, §2.2.1, §2.2.2,§2.2.3, aggiunta la sezione §2.2.4	Meneghini Fabio	Cappellari Marco
0.4.0	2023/11/30	Revisione e verifica della documentazione prodotta	De Laurentis Arianna Pia	De Laurentis Arianna Pia

Ver	Data	Descrizione	Autore	Verifica
0.3.4	2023/11/29	Migliorata la sezione §3.4, aggiunte le sezioni §3.4.4.1, §3.4.4.2, §3.4.5.1, §3.4.5.2, §3.4.5.3, §3.4.5.4, §3.4.5.5, completata la sezione §3.5	Meneghini Fabio	De Laurentis Arianna Pia
0.3.3	2023/11/28	Completata la sezione §2.1.6	Meneghini Fabio	Cappellari Marco
0.3.2	2023/11/28	Completata la sezione §3.3	Meneghini Fabio	De Laurentis Arianna Pia
0.3.1	2023/11/27	Migliorata la sezione §2.1.1, aggiunta la sezione §2.1.4, completata la sezione §2.1.5	Meneghini Fabio	De Laurentis Arianna Pia
0.3.0	2023/11/26	Verifica della documentazione prodotta	Cappellari Marco	Cappellari Marco
0.2.6	2023/11/26	Migliorate le sezioni §3.1.5.5, §3.1.5.6, §3.1.5.7, §3.1.5.8, §3.1.7, §3.2.4.1, §3.2.5.1, §3.3	Filippini Giovanni	Cappellari Marco
0.2.5	2023/11/25	Migliorate le sezioni §3.1.1, §3.1.2, §3.1.3, §3.1.5.1, §3.1.6.2, §3.2.2	Filippini Giovanni	Agafitei Ciprian

Ver	Data	Descrizione	Autore	Verifica
0.2.4	2023/11/21	Completata la stesura di §4	Filippini Giovanni	Agafitei Ciprian
0.2.3	2023/11/21	Stesura appendice §C	Filippini Giovanni	Cappellari Marco
0.2.2	2023/11/21	Completata la stesura di §A e iniziata la stesura di §B	Filippini Giovanni	Cappellari Marco
0.2.1	2023/11/20	Iniziata la stesura di §A	Filippini Giovanni	Agafitei Ciprian
0.2.0	2023/11/19	Verifica di §3 e §4	Ye Tao Ren Federico	Ye Tao Ren Federico
0.1.7	2023/11/18	Continuata la stesura di §4	De Laurentis Arianna Pia	Ye Tao Ren Federico
0.1.6	2023/11/17	Continuata la stesura di §4	De Laurentis Arianna Pia	Agafitei Ciprian
0.1.5	2023/11/16	Continuata la stesura di §3	De Laurentis Arianna Pia	Agafitei Ciprian
0.1.4	2023/11/10	Continuata la stesura di §3 e §4	Cappellari Marco	Pluzhnikov Dmitry
0.1.3	2023/11/09	Continuata la stesura di §3	Cappellari Marco	Ye Tao Ren Federico
0.1.2	2023/11/08	Continuata la stesura di §3 e iniziata la stesura di §4	Cappellari Marco	Ye Tao Ren Federico
0.1.1	2023/11/07	Sviluppo di §1, iniziata la stesura di §2 e §3, definita la struttura preliminare del documento	Cappellari Marco	Pluzhnikov Dmitry

Ver	Data	Descrizione	Autore	Verifica
0.1	2023/11/06	Creazione del documento	Cappellari Marco	Pluzhnikov Dmitry

Tabella 1: Registro delle modifiche

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento ha l'obiettivo di identificare le *best practices*$_G$ di progetto e stabilire in modo chiaro e dettagliato il *Way of Working*$_G$ dell'attività produttiva, con lo scopo di ottenere omogeneità e coesione nel lavoro svolto dal gruppo *torchlight* nell'ambito del progetto "ChatSQL".

1.2 Scopo del prodotto

Nell'ultimo anno vi è stato un cambiamento repentino nello sviluppo e nell'applicazione dell'*Intelligenza Artificiale*$_G$, passando dall'elaborazione e raccomandazione dei contenuti alla generazione di essi, come immagini, testi e tracce audio.

Il *capitolato*$_G$ C9, "ChatSQL", pone come obiettivo la realizzazione di un'applicazione che permetta, partendo dalla descrizione di un *database*$_G$, la generazione di un *prompt*$_G$ in risposta ad un'interrogazione in linguaggio naturale. Questo processo dovrà essere gestito da un *LLM*$_G$ (Large Language Model) specificatamente addestrato in questo scopo.

Tale applicazione sarà fruibile attraverso una *web app*$_G$, dove l'utente potrà caricare un file in formato *JSON*$_G$ e richiedere, con linguaggio naturale, la produzione di prompt riguardanti il documento caricato.

1.3 Glossario

Al fine di evitare possibili ambiguità relative al linguaggio utilizzato nei documenti, viene fornito un *Glossario* (attualmente alla sua versione *1.0.0*), nel quale sono contenute le definizioni di termini aventi uno specifico significato.
Tali termini sono segnati in corsivo e marcati con il simbolo "$_G$" a pedice (per esempio: *Way of Working*$_G$) ogni prima occorrenza nel capitolo.

1.4 Miglioramenti al documento

La maturità e i miglioramenti sono aspetti fondamentali nella stesura di un documento. Questo permette di apportare agevolmente modifiche in base alle esigenze concordate tra i membri del gruppo e il *proponente*$_G$, nel corso del tempo. Di conseguenza, questa versione del documento non può essere considerata definitiva o completa, poiché è soggetta a evoluzioni future.

1.5 Riferimenti

1.5.1 Riferimenti normativi

- Glossario (v 1.0.0)
- Capitolato d'appalto C9 - ChatSQL
- Standard ISO/IEC 12207:1995 - Processi del ciclo di vita del software

1.5.2 Riferimenti informativi

- Documentazione Jira
- Documentazione Overleaf

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Il *processo*$_G$ di Fornitura ha l'obiettivo di stabilire e gestire le regole, le risorse e i requisiti che i membri del team *torchlight* devono seguire per svolgere in modo adeguato il ruolo di *fornitore*$_G$ nei confronti dell'azienda *proponente*$_G$, in questo caso *Zucchetti S.p.A*$_G$, e dei *committenti*$_G$.
Il fornitore dovrà stabilire un contratto con il proponente in cui vengono concordati i requisiti che il prodotto finale dovrà soddisfare e viene definita una data di consegna di quest'ultimo. Una volta raggiunto l'accordo sarà possibile passare alla fase esecutiva, redigendo il documento *Piano di Progetto*$_G$.

2.1.1.1 Fasi della fornitura

Secondo la norma *ISO/IEC 12207:1995*$_G$, il *processo*$_G$ di fornitura si compone delle seguenti fasi:

1. Avvio;
2. Approntamento di risposte alle richieste;
3. Contrattazione;
4. Pianificazione;
5. Esecuzione e controllo;
6. Revisione e valutazione;
7. Consegna e completamento.

2.1.2 Descrizione

Questa sezione raccoglie le norme che i membri del gruppo *torchlight* sono tenuti a rispettare durante lo sviluppo del progetto.

2.1.3 Aspettative

Durante l'intero progetto, il nostro gruppo mira a stabilire un rapporto di collaborazione continua con *Zucchetti S.p.A.*$_G$, in particolare con il referente Dr. Gregorio Piccoli, in modo da:

- Identificare gli aspetti chiave per soddisfare le esigenze del *proponente*$_G$;
- Rilevare eventuali vincoli relativi ai requisiti e ai *processi*$_G$;
- Effettuare stime dei costi, sia in termini di tempo che di denaro;
- Garantire che il prodotto soddisfi le richieste, concordando i criteri di qualifica.

2.1.4 Rapporti col proponente

Il *proponente*$_G$ mette a disposizione la e-mail come canale di comunicazione tramite i quali è possibile stabilire nuovi incontri telematici, che avverranno nella piattaforma *Zoom*$_G$. La frequenza dei meeting online non è regolare, essi infatti vengono concordati in base alle necessità del gruppo o del proponente.
I principali motivi di discussione sono:

- Chiarimenti riguardo i vincoli del *capitolato*$_G$;
- Dubbi riguardo le tecnologie da utilizzare per lo sviluppo del progetto;
- Richiesta di feedback su quanto prodotto.

Per ciascun incontro con il proponente verrà compilato un verbale esterno, che riporterà gli argomenti di discussione tra il gruppo ed esso. Tutti i verbali potranno essere visionati nella *repository*$_G$ relativa alla documentazione del progetto (si veda il paragrafo 3.2.5.1).

2.1.5 Documentazione fornita

Di seguito viene riportato l'elenco completo dei documenti che il gruppo *torchlight* consegnerà al *proponente*$_G$ *Zucchetti S.p.A*$_G$ e ai *committenti*$_G$ Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.5.1 Analisi dei Requisiti

L'*Analisi dei Requisiti*$_G$ è un documento che descrive dettagliatamente i *casi d'uso*$_G$, i requisiti e le funzionalità che il *prodotto software*$_G$ deve offrire. Tale documento ha lo scopo di chiarire eventuali dubbi ed ambiguità che possono presentarsi dopo la lettura del *capitolato*$_G$. Nella sezione 2.2.4 sono presenti maggiori dettagli riguardo l'Analisi dei Requisiti.
Il documento conterrà:

- **Descrizione del prodotto**;
- **Lista dei casi d'uso**: elenca tutti i possibili scenari di utilizzo del sistema software da parte degli utenti finali, ovvero le azioni o attività che essi possono svolgere con il sistema. Tutti i casi d'uso sono dotati di una descrizione dettagliata delle azioni che l'utente compie, in modo da far emergere tutti i requisiti che non erano ovvi dopo la sola lettura del capitolato;
- **Lista dei requisiti**: elenca tutti i vincoli richiesti dal *proponente*$_G$ o dedotti in seguito all'individuazione dei casi d'uso ad essi collegati.

2.1.5.2 Piano di Progetto

Il *Piano di Progetto*$_G$ è un documento che tratta i seguenti punti:

- **Analisi dei rischi**: vengono analizzati eventuali rischi che potrebbero emergere durante lo sviluppo del progetto. Ad ogni rischio individuato viene associata una strategia di mitigazione, così da risolvere o ridurre l'entità del problema in caso di occorrenza del rischio in questione;
- **Modello di sviluppo**: viene delineato l'approccio metodologico impiegato durante il *processo*$_G$ di sviluppo del *prodotto software*$_G$;
- **Pianificazione**: viene pianificato il periodo temporale relativo a ciascuna attività da svolgere, con relativa descrizione;
- **Preventivo**: dalla durata dei periodi per poter completare tutte le attività si ricava il preventivo. Al termine di ogni periodo verrà redatto il consuntivo che metterà a

confronto il preventivo con quanto realmente realizzato, così da visualizzare lo stato di avanzamento del progetto;

- **Consuntivo**: in questo capitolo sono indicate le spese effettive nelle diverse fasi del progetto.

2.1.5.3 Piano di Qualifica

Il *Piano di Qualifica*$_G$ è un documento che descrive le attività e le strategie adottate per garantire la qualità del *prodotto software*$_G$. In esso vengono descritte le metodologie, le tecniche e gli strumenti che verranno utilizzati per far sì che il prodotto software sia in linea con le aspettative del *proponente*$_G$. Tale documento è utile per la gestione del *processo*$_G$ di sviluppo: in particolare, permette di monitorare lo stato di avanzamento del progetto in relazione agli obiettivi di qualità prefissati.
Il Piano di Qualifica tratta i seguenti punti:

- **Qualità di processo**: definisce i parametri e le *metriche*$_G$ che ciascun membro del team è tenuto a rispettare al fine di garantire processi di alta qualità;
- **Qualità di prodotto**: definisce i parametri e le metriche che ciascun membro del team è tenuto a rispettare al fine di garantire un prodotto finale di alta qualità;
- **Strategie di testing**: descrive il piano di Testing, con l'obiettivo di garantire la correttezza del prodotto finale;
- **Obiettivi di qualità**: definisce quali sono i valori che dovranno assumere le metriche per essere ritenute accettabili o pienamente soddisfatte;
- **Valutazioni per il miglioramento**: analizza le criticità rilevate durante il processo di sviluppo e le azioni intraprese per agevolare tale processo;
- **Cruscotto delle metriche**: presenta il *cruscotto*$_G$ delle metriche utilizzato durante il periodo dello sviluppo del progetto.

2.1.5.4 Lettera di Presentazione

Ad ogni revisione di avanzamento del progetto è associata una *Lettera di Presentazione*. È un documento con il quale si formalizza la consegna della revisione di avanzamento in questione. In essa è presente l'elenco della documentazione che verrà messa a disposizione del *proponente*$_G$ e dei *committenti*$_G$. Il gruppo si impegna a rispettare i requisiti minimi e a consegnare il *prodotto software*$_G$ entro la data prestabilita.

2.1.5.5 Glossario

Il *Glossario* consiste in un elenco di termini che compaiono nei documenti, e relative definizioni, il cui significato può non essere immediato. È utile per evitare potenziali ambiguità ed agevolare la comunicazione tra i membri del gruppo.

2.1.6 Strumenti

Di seguito sono riportati gli strumenti utilizzati per la realizzazione del *processo*$_G$ di fornitura:

- **Git**: software per il controllo di versione;
- **Github**: servizio di hosting per progetti software;
- **Pixelmator Pro**: software per la creazione e modifica di foto, video e grafica vettoriale, utilizzato nello specifico per lo sviluppo delle grafiche;
- **Jira**: 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*;
- **Zoom**: è una piattaforma che permette di effettuare videoconferenze online. Il gruppo utilizza la piattaforma Zoom per organizzare gli incontri con il proponente e quelli interni del team;
- **Microsoft Outlook**: casella di posta elettronica per la comunicazione col *proponente*$_G$;

- **LaTeX**: 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;
- **Overleaf**: i membri del gruppo utilizzano la piattaforma online *Overleaf* per scrivere e modificare i documenti in LaTeXin modo collaborativo e in tempo reale;
- **Diagrams.net**: software di disegno grafico per creare diagrammi;
- **Telegram**: per communicazioni brevi è stato creato un gruppo in questo messenger.

2.2 Sviluppo

2.2.1 Scopo

Il *processo*$_G$ è mirato a definire le *attività*$_G$ e i compiti che lo sviluppatore esegue e supporta per soddisfare i requisiti definiti precedentemente nel contratto.

2.2.2 Descrizione

Il *processo*$_G$ di Sviluppo ha l'obiettivo di delineare le responsabilità e le attività necessarie affinché il nostro team possa procedere con la creazione del software richiesto, assegnando specifici ruoli ai vari membri. Il fine ultimo è garantire che il *prodotto software*$_G$ soddisfi le aspettative del *proponente*$_G$.
Nel processo di Sviluppo saranno effettuate le seguenti *attività*$_G$:

- *Analisi dei Requisiti*$_G$;
- *Progettazione*$_G$;
- *Codifica*$_G$.

2.2.3 Aspettative

Le aspettative del *processo*$_G$ di Sviluppo consistono nel determinare gli obiettivi di sviluppo, i vincoli tecnologici e quelli di design, realizzare un prodotto che riesca a superare i *test*$_G$ e a soddisfare i requisiti del *proponente*$_G$.

2.2.4 Analisi dei Requisiti

2.2.4.1 Scopo

L'*attività*$_G$ di *Analisi dei Requisiti*$_G$ è mirata a:

- Definire lo scopo finale del *prodotto*$_G$;
- Individuare i requisiti del prodotto;
- Definire gli *attori*$_G$ del sistema;
- Definire i *casi d'uso*$_G$ del sistema;
- Presentare una visone chiara del prodotto ai progettisti;
- Favorire l'attività di *verifica*$_G$;

2.2.4.2 Descrizione

L'*attività*$_G$ di *Analisi dei Requisiti*$_G$ viene effettuata dall'analista e come risultato ha la produzione del documento *Analisi dei Requisiti*. Tale documento contiene le informazioni che descrivono:

- lo scopo del *prodotto*$_G$;
- gli obiettivi del prodotto;
- le funzionalità del prodotto;
- le caratteristiche degli *attori*$_G$;
- i *casi d'uso*$_G$;
- gli attori;
- i requisiti.

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*$_{\boldsymbol{G}}$;
- *Scenario alternativo*$_{\boldsymbol{G}}$;
- inclusione/i;
- estensione/i;
- specializzazione/i;
- commento/i;
- descrizione.

I casi d'uso sono identificati nel seguente modo:

$$UC[Numero](.[Sottocaso])^* - [Titolo]$$

dove:

- **UC**: acronimo di *use case*;
- ***Numero***: numero identificativo del caso d'uso;
- ***Sottocaso***: numero identificativo del sottocaso;
- ***Titolo***: titolo assegnato al caso d'uso.

2.2.4.4 Requisiti

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

$$R[Importanza][Tipologia]\ \ [Codice](.[Codicefiglio])^*$$

dove:

- **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;

- **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;
- **Codice**: identificatore numerico univoco;
- **Codice figlio**: numero identificativo del sottorequisito.

2.2.5 Progettazione

2.2.5.1 Scopo

Lo scopo dell'*attività*$_G$ di progettazione è di definire l'*architettura*$_G$ del prodotto capace di soddisfare le necessità di tutti gli *stakeholder*$_G$ e fornire una documentazione dettagliata sulla sua struttura, sulle specifiche tecniche e sulle scelte tecnologiche che verranno adottate nella sua realizzazione.

2.2.5.2 Descrizione

Gli obiettivi della *progettazione*$_G$ si raggiungono tramite dominazione della complessità del prodotto, organizzazione e ripartizione delle responsabilità di realizzazione. Questa *attività*$_G$ va effettuata su diversi livelli: di sistema, di software e di dettaglio.
Nella fase *Requirements and Technology Baseline*$_G$ va prodotto il *proof of concept*$_G$ per dimostrare la fattibilità tecnologica del prodotto atteso. Nella fase *Product Baseline*$_G$ si raggiunge il design definitivo che rispecchia l'architettura logica del prodotto, per componenti,

ruoli, connessioni e interazioni. Inoltre, si definiscono architettura di *deployment*$_G$, *design pattern*$_G$ architetturali e *idiomi*$_G$ utilizzati.

2.2.5.3 Documentazione

Requirements and Technology Baseline

- **struttura dei componenti**: definisce i componenti del prodotto e loro relazione con i requisiti che soddisfano;
- **scelte tecnologiche**: presentano le tecnologie utilizzate ed argomentano la loro scelta;
- ***proof of concept***$_G$: insieme di artefatti che dimostrano la fattibilità di realizzazione delle funzionalità principali individuate con il proponente;

2.2.6 Codifica

2.2.6.1 Scopo

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

2.2.6.2 Descrizione

Durante questa *attività*$_G$ lo sviluppatore si impegna a soddisfare i requisiti scrivendo il codice nel linguaggio di programmazione prescelto. Tale codice deve corrispondere alle linee guida prescritte nella documentazione del progetto. Parallelamente, tutte le nuove unità del software sviluppate e le modifiche apportate vanno documentate.

2.2.6.3 Stile della codifica

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

- **Variabili**: devono avere un nome univoco, significativo, scritto in minuscolo in lingua inglese. Se il nome contiene più parole, quelle si separano con un trattino basso;
- **Funzioni**: devono avere un nome univoco, significativo, in lingua inglese che comincia con lettera minuscola. Se il nome contiene più parole, ogni parola dopo la prima comincia con lettera maiuscola;

- **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 seguono indentazione con carattere di tabulazione di lunghezza equivalente a 4 spazi;
- **Commenti**: devono essere scritti prima del componente a cui fanno riferimento in lingua italiana.

2.2.6.4 Tecnologie utilizzate

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

  https://www.python.org/

- **Txtai**: un framework che realizza un *embedded database*$_G$, usato per la ricerca e l'immagazzinamento delle informazioni del *dizionario dati*$_G$ in forma vettoriale

  https://neuml.github.io/txtai/

- **Streamlit**: un framework usato per lo sviluppo dell'interfaccia grafica dell'applicazione

  https://streamlit.io/

- **paraphrase-multilingual-MiniLM-L12-v2**: un LLM_G usato per la generazione della rappresentazione vettoriale delle informazioni del dizionario dati

  https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2

- **JSON schema**: una libreria usata per il controllo di consistenza e validità dei files di dizionari dati

  https://json-schema.org/

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, poiché 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 *torchlight* per ciascuna fase del ciclo di vita del software.

3.1.3 Aspettative

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

- Definire delle procedure ripetibili che permettano di standardizzare il *way of working* 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*$_G$ di vita di ogni documento si compone delle seguenti fasi:

- **Redazione**: i documenti vengono scritti seguendo un approccio incrementale, e sono considerati redatti soltanto una volta completata la loro stesura;

- **Verifica**: 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.

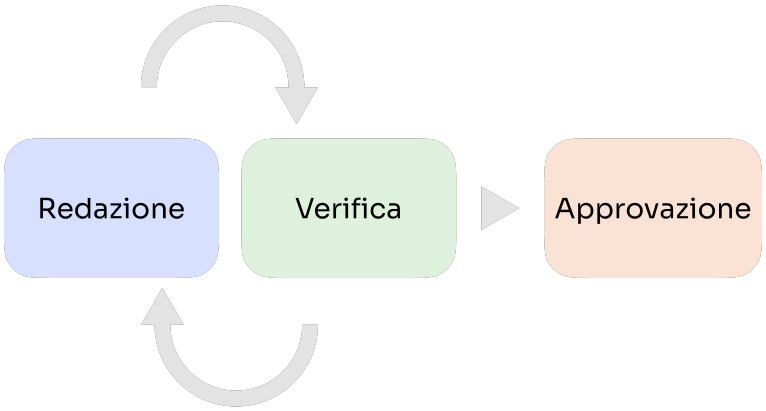

Figura 1: Ciclo di vita dei documenti

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 delle 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 uno a ciascuna appendice. Se ci sono più appendici, ogni volta che ne viene comple-

tata una, si ricomincia la numerazione da uno 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 di versione del file;
- Sul lato destro: il numero della pagina attualmente in consultazione.

3.1.5.3 Frontespizio

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

- **Logo del gruppo**: il logo del gruppo è posizionato in alto a centro pagina;
- **Recapito**: il recapito mail del gruppo è posizionato centralmente, appena sotto il nome dello stesso;
- **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:
  - **Redattori**: elenco dei membri del gruppo (cognome e nome) che hanno svolto la redazione del documento;

- **Verifica**: elenco dei membri del gruppo (cognome e nome) che hanno svolto la verifica del documento;
- **Approvazione**: elenco dei membri del gruppo (cognome e nome) che hanno svolto l'approvazione del documento;
- **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**: versione del documento in seguito alla modifica;
- **Data**: data in cui è stata effettuata la modifica;
- **Descrizione**: breve descrizione della modifica apportata;
- **Verificatore**: sta ad indicare il verificatore del periodo che effettua la verifica del contenuto che è stato aggiungo, modificato o eliminato;
- **Autore**: elenco dei membri del gruppo coinvolti.

I verbali, sia interni che esterni, non sono dotati del registro delle modifiche in quanto sono redatti direttamente durante gli incontri e non sono soggetti a modifiche future.

3.1.5.5 Indice

Tutti i documenti 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 di esse 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 l'indice, 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 l'indice 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 devono contenere le seguenti sezioni:

- **Informazioni generali**, che si compone delle sottosezioni:
  - **Svolgimento dell'incontro**: riporta alcune informazioni riguardanti data, ora e luogo dell'incontro. Nello specifico, indica il luogo (che può essere fisico o virtuale), la data, l'ora di inizio e l'ora di fine dell'incontro;
  - **Registro delle presenze**: è una lista di tutte le persone presenti seguita da un'eventuale lista delle persone assenti (se non ci sono persone assenti, al suo posto si trova la dicitura *Nessun membro del gruppo è assente alla riunione*).
- **Temi affrontati**: è una descrizione dei temi di discussione dell'incontro. Ciascuna tematica definisce una sottosezione.

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* ("_"). 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 (*torchlight*), il nome dell'azienda *proponente*$_G$ (*Zucchetti S.p.A*) 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 saranno mostrati in colore arancione e lievemente evidenziati in giallo al momento del passaggio del cursore al di sopra degli stessi.

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:

YYYY/MM/DD

dove:

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

3.1.6.5 Sigle

Una lista di sigle presenti 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;
  - **PB**: Product Baseline;
  - **CA**: Customer Acceptance.

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

- **LaTeX**: 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;
- **Overleaf**: i membri del gruppo utilizzano la piattaforma online Overleaf per scrivere e modificare i documenti in LaTeX in modo collaborativo e in tempo reale;
- **Diagrams.net**: il team ha optato per Diagrams.net per disegnare i diagrammi *UML*$_G$ necessari alla realizzazione del progetto didattico.

3.2 Gestione della configurazione

3.2.1 Scopo

L'obiettivo di questa sezione è delineare l'approccio adottato dal gruppo *torchlight* 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 delle attività, invece, 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*$_G$.

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*$_G$ pubbliche, al fine di mantenere separati i documenti organizzativi e il software:

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

3.2.5.1 Documentazione

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

- ***1 - 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:
  - **Verbali**: cartella contenente tutti i verbali interni (raccolti in una sotto-cartella) e un verbale esterno relativi al periodo di candidatura;
  - **Lettera di presentazione (v1.0)**: documento con cui il gruppo *torchlight* si candida formalmente per la realizzazione del progetto commissionato;
  - **Preventivo dei costi ed impegni orari (v1.0)**: 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;

- **Valutazione capitolati d'appalto (v1.0)**: documento in cui il gruppo valuta le criticità riscontrate per ogni capitolato d'appalto e motiva la scelta del capitolato selezionato;

- ***2 - RTB***: in questa cartella sono presenti i documenti prodotti per la candidatura alla RTB_G. Il contenuto è composto da:
  - **Documentazione esterna**: sono contenuti i seguenti documenti:
    - **Verbali esterni**: cartella che raccoglie tutti i verbali esterni relativi al periodo RTB;
    - **Analisi dei Requisiti (v1.0.0)**: si veda la sezione §2.1.5.1
    - **Piano di Progetto (v1.0.0)**: si veda la sezione §2.1.5.2
    - **Piano di Qualifica (v1.0.0)**: si veda la sezione §2.1.5.2
    - **Glossario (v1.0.0)**: si veda la sezione §2.1.5.5
  - **Documentazione interna**: al suo interno sono contenuti:
    - **Verbali interni**: cartella che raccoglie tutti i verbali interni relativi al periodo RTB;
    - **Norme di Progetto (v1.0.0)**: si veda la sezione §1.1;
    - **Lettera di Presentazione**: si veda la sezione §2.1.5.4.

3.2.5.2 ChatSQL

Questa $repository_G$ contiene il codice sorgente relativo al $Proof\ of\ Concept_G$ del progetto e le istruzioni per scaricarlo e testarlo in locale (*README.md*). Inoltre, è presente una cartella chiamata *JSON* che contiene alcuni dizionari dati usati per testare l'applicazione (cioè dei file in formato $JSON_G$ che descrivono la struttura di un $database_G$).

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

Lo scopo di questa sezione è delineare le norme che il gruppo *torchlight* adotta per la gestione della qualità.

3.3.2 Descrizione

Il processo di gestione della qualità si pone l'obiettivo di garantire che i processi adottati dal *fornitore*$_G$ e i prodotti sviluppati rispettino specifici obiettivi di qualità, al fine di soddisfare le aspettative del *proponente*$_G$ e degli utenti finali.

3.3.3 Aspettative

Le aspettative del gruppo sono le seguenti:

- Realizzazione di un prodotto di qualità, in linea con le richieste del *proponente*$_G$;
- Ottenere un miglioramento continuo dei processi, perseguendo gli standard di qualità.

3.3.4 PDCA

Il *PDCA*$_G$ (*Plan Do Check Act*) è una metodologia iterativa che permette di realizzare il controllo e il miglioramento continuo dei *processi*$_G$. A tal fine è necessario svolgere costantemente tutte e quattro le fasi di cui il PDCA si compone. Queste fasi sono:

- **Plan**: consiste nello svolgere le attività di pianificazione con lo scopo di stabilire quali processi debbano essere attivati e in quale ordine, in modo da raggiungere gli obiettivi specificati;
- **Do**: consiste nell'esecuzione effettiva di ciò che è stato deciso nella fase *Plan*, misurando i risultati ottenuti durante lo svolgimento della stessa;
- **Check**: consiste nell'analisi dei risultati ottenuti durante la fase *Do*. Questi dati vengono valutati sulla base di alcune *metriche*$_G$ prefissate e confrontati con i risultati previsti nella fase *Plan*;
- **Act**: consiste nel consolidare quanto di buono è emerso nella fase *Check* e nell'attuazione di strategie correttive per migliorare ciò che non è risultato conforme con quanto previsto, analizzandone a fondo le cause.
  Ciò permette di raffinare il ciclo PDCA ad ogni iterazione, portando di volta in volta del valore aggiunto.

3.3.5 Strumenti

Gli strumenti utilizzati per la gestione della qualità sono le *metriche*$_G$.
In questa sezione del documento saranno illustrate le norme che regolano le metriche adottate e i conseguenti obiettivi prefissati.

3.3.5.1 Struttura delle metriche

Le *metriche*$_G$ adottate verranno presentate in elenchi strutturati in questo modo:

- **Codice**: codice identificativo della metrica nel formato:

  M[numero][sigla]

  dove:

  - **M**: indica "metrica";
  - [**numero**]: numero che identifica univocamente ciascuna metrica;
  - [**sigla**]: sigla composta dalle iniziali del nome della metrica.

- **Nome**: specifica il nome della metrica;
- **Descrizione**: breve descrizione della metrica in questione;
- **Scopo**: motivo per cui viene adottata tale metrica;
- **Formula**: come viene calcolata (opzionale);
- **Strumento**: strumento che viene usato per calcolarla (opzionale).

3.3.5.2 Struttura degli obiettivi

Gli obiettivi per ciascuna *metrica*$_G$ sono descritti nel dettaglio nel documento *Piano di Qualifica*$_G$ in tabelle strutturate (per colonne) nel seguente modo:

- **Metrica**: codice che identifica la metrica in questione;
- **Nome**: nome della relativa metrica;
- **Valore di accettazione**: valore considerato accettabile relativo alla metrica in questione;
- **Valore preferibile**: valore ideale che dovrebbe essere assunto dalla metrica.

3.4 Verifica

3.4.1 Scopo

Lo scopo di questa sezione è descrivere come il gruppo ha deciso di attuare il processo di *verifica*$_G$, ovvero il processo di valutazione del prodotto software. Esso serve per garantire che il software sia conforme alle aspettative e ai requisiti specificati, basandosi su criteri di consistenza, completezza e correttezza dei prodotti ottenuti. Il processo di verifica viene eseguito durante tutto il *ciclo di vita del software*$_G$, dalla fase di *progettazione*$_G$ alla fase di manutenzione.

3.4.2 Aspettative

Le aspettative per questo processo sono le seguenti:

- Ottenere il prodotto finale atteso;

- Individuare preventivamente eventuali errori in modo da risparmiare risorse e tempo;
- Permettere al gruppo di avanzare nello sviluppo in modo più sicuro e corretto.

3.4.3 Descrizione

Per attuare il processo di *verifica*$_G$ si fa uso di *test*$_G$ e analisi che permettono di accertarsi che non siano stati introdotti errori durante lo sviluppo del software o la stesura dei documenti, in modo tale che i prodotti soddisfino i requisiti specificati.
Per garantire il corretto svolgimento del processo di verifica è necessario rispettare alcuni punti chiave, tra cui l'applicazione di procedure finite, l'adozione di criteri affidabili per la verifica, l'implementazione di una sequenza di fasi successive per poi permettere l'avvio della prossima fase, cioè quella di *validazione*.
Di seguito vengono elencate e descritte le possibili attività che possono essere svolte durante il processo di verifica.

3.4.4 Analisi statica

Questo tipo di analisi è chiamato *statica* perché non richiede l'esecuzione del codice da testare, e per questo motivo può anche essere applicata alla documentazione. Quest'*attività*$_G$ prevede la revisione critica del codice e della relativa documentazione con lo scopo di verificarne la conformità ai vincoli, l'assenza di difetti e la presenza delle proprietà desiderate. Il successo di quest'attività dipende dall'esperienza dei verificatori coinvolti.
L'*analisi statica*$_G$ può avvalersi di diversi metodi di lettura, descritti in seguito:

3.4.4.1 Walkthrough

Questa tecnica consiste in una lettura "a pettine" della documentazione o del codice. Il verificatore e l'autore del prodotto in esame collaborano seguendo questi passi:

- **Pianificazione**: è un dialogo tra autori e verificatori che ha lo scopo di individuare tutte le proprietà e i vincoli che il prodotto deve rispettare per essere considerato corretto;
- **Lettura**: il verificatore legge il documento o il codice per cercare errori e discrepanze con i vincoli imposti in precedenza;

- **Discussione**: è un dialogo tra autori e verificatori che avviene dopo la lettura per confrontarsi su quanto rilevato;
- **Correzione**: gli autori del prodotto in esame correggono gli errori che sono stati trovati.

Questa tecnica non è automatizzabile in quanto i criteri possono subire frequenti variazioni. Il metodo del *Walkthrough*$_G$ sarà utilizzato soprattutto nei primi periodi del progetto, dal momento che i documenti in esame saranno relativamente brevi e il costo di tale operazione sarà contenuto.

3.4.4.2 Inspection

A differenza del *Walkthrough*$_G$, questa tecnica ha l'obiettivo di rilevare la presenza di difetti nel prodotto in analisi andando a compiere una lettura mirata anziché "a pettine", specificando prima cosa sarà oggetto di verifica. L'insieme delle caratteristiche da verificare selettivamente definiscono una *lista di controllo*, ovvero una *checklist*$_G$ utile a determinare se l'attività di ispezione sia stata svolta in modo corretto o meno.
I passi che compongono questa attività sono i seguenti:

- **Pianificazione**: vedi §3.4.4.1;
- **Definizione della lista di controllo**: si determina ciò che deve essere verificato in modo selettivo;
- **Lettura**: vedi §3.4.4.1;
- **Correzione**: vedi §3.4.4.1.

3.4.5 Analisi dinamica

L'*analisi dinamica*$_G$ è una categoria di tecniche di analisi che richiedono l'esecuzione del codice in esame. Per questo motivo l'analisi dinamica può essere applicata soltanto al codice e non alla documentazione.
Essa prevede l'esecuzione di un insieme di *test*$_G$ volti a verificare la corretta esecuzione del software e di rilevare eventuali errori, cioè differenze tra risultati ottenuti e risultati attesi. Questo tipo di analisi deve essere automatizzata e ripetibile.

Per garantire l'efficacia dei test, ciascuno di essi deve essere decidibile e ripetibile, ovvero, dati gli stessi input, deve produrre sempre lo stesso output. Inoltre, tutti i test devono presentare dei parametri ben definiti, tra cui:

- Descrizione dei parametri di input;
- Descrizione dell'output;
- Comportamento atteso del software;
- Condizioni di esecuzione del test.

Per garantire l'efficacia dei test è opportuno scrivere del codice di qualità e identificare correttamente i requisiti funzionali e non del sistema.
Per agevolare l'automazione dei test, verrà fatto uso dei seguenti strumenti:

- **Driver**: è 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**: è una componente passiva che simula il comportamento di alcune parti mancanti o non ancora implementate nell'applicazione. In questo modo consente di testare in anticipo le parti del software che dipendono da altre parti che non sono ancora disponibili;
- **Logger**: è uno strumento attraverso il quale il test registra i risultati ottenuti durante la sua esecuzione.

In seguito sono elencate e descritte le varie tipologie di test che il gruppo *torchlight* andrà ad utilizzare, in ordine crescente di grandezza dell'oggetto del test:

3.4.5.1 Test di unità

I *test di unità*$_G$ agiscono individualmente sulle più piccole unità del prodotto software. Essi vengono definiti durante la progettazione di dettaglio.
Ci sono due tipi di test di unità:

- **Test funzionali**: hanno l'obiettivo di verificare se, dato un certo input, l'output prodotto corrisponde con quello atteso. I test funzionali producono *requirement coverage*$_G$. Sono molto veloci, ma non assicurano la completa copertura del codice dell'unità;
- **Test strutturali**: hanno l'obiettivo di testare la copertura di tutti i possibili cammini dell'unità. Vengono costruiti ad hoc per testare un determinato cammino, quindi è necessario creare una batteria di test per coprire completamente il codice dell'unità. Essi producono *structural coverage*$_G$. Assicurano che ogni cammino venga eseguito, ma in alcuni casi potrebbe essere piuttosto onerosi da costruire.

3.4.5.2 Test di integrazione

I *test di integrazione*$_G$ hanno l'obiettivo di verificare la corretta integrazione di ciò che viene testato con i test di unità, ovvero le singole unità architetturali, in modo da rilevare e correggere eventuali difetti. Questi test sono definiti durante la fase di *progettazione*$_G$ architetturale.
Essi sono inoltre reversibili, cioè è possibile tornare indietro ad uno stato sicuro in caso si verifichino errori durante lo svolgimento di questi test.
L'integrazione può avvenire tramite due approcci differenti:

- **Top-down**: l'integrazione inizia dalle componenti del sistema che presentano più dipendenze e con più valore esterno, rendendo disponibili da subito le funzionalità ad alto livello. Permettono di testare fin da subito le funzionalità cardine del prodotto, ma richiedono molti *stub*;
- **Bottom-up**: l'integrazione inizia dalle componenti del sistema che presentano meno dipendenze, ma meno visibili dall'utente. Il vantaggio di questo approccio è che richiede pochi *stub*, tuttavia così viene ritardata l'implementazione delle funzionalità utente.

3.4.5.3 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 *testi di integrazione*$_G$, dunque quando tutte le componenti del sistema software sono state integrate. Questo tipo di test è svolto anche durante il *processo* di validazione.

3.4.5.4 Test di regressione

I *test di regressione*$_G$ servono per assicurare che aggiornamenti a specifiche unità architetturali non danneggino il resto del sistema. Questi test consistono nel ripetere i test di unità, di integrazione e di sistema necessari al fine di poter confermare che tali cambiamenti non intacchino le funzionalità precedentemente implementate e verificate.
Questo tipo di test coinvolge i processi di *Problem Resolution* e *Change Management*, valutando se e in che modo effettuare le modifiche desiderate.

3.4.5.5 Test di accettazione

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

3.4.5.6 Codici relativi ai test

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

T[tipo][codice]

dove:

- [**tipo**] rappresenta il tipo di test, in particolare:
  - **U** per i test di unità;
  - **I** per i test di integrazione;
  - **S** per i test di sistema;
  - **R** per i test di regressione;
  - **A** per i test di accettazione.
- [**codice**] è un numero progressivo associato al test all'interno del suo tipo. Se il test in questione ha un padre, si segue il formato

  [codice padre].[codice figlio]

dove [**codice figlio**] è un numero progressivo che identifica univocamente un test figlio del test con codice pari a [**codice padre**].

3.4.5.7 Stato dei test

Ad ogni test verrà associato uno stato che ne rappresenterà l'esito. L'insieme degli esiti dei test verrà riportato nel *cruscotto*$_G$ dei test. Lo stato può essere:

- **N-I**: il test non è ancora stato implementato;
- **Passato**: il test riporta esito positivo;
- **Non passato**: il test riporta esito negativo.

3.5 Validazione

3.5.1 Scopo

L'obiettivo di questa sezione è delineare l'approccio adottato dal gruppo nel gestire il *processo*$_G$ di *validazione*$_G$. Tale processo mira a garantire che il prodotto sviluppato sia conforme ai requisiti stabiliti con il *proponente*$_G$ e che soddisfi le esigenze del *committente*$_G$.

3.5.2 Aspettative

L'aspettativa che il team di sviluppo si propone di raggiungere attraverso l'implementazione di questo *processo*$_G$ è quella di garantire la conformità del prodotto software ai requisiti specificati nel documento di *Analisi dei Requisiti*$_G$.

3.5.3 Descrizione

Il *processo*$_G$ di *validazione*$_G$ prenderà in ingresso i *test*$_G$ che saranno svolti durante l'*attività*$_G$ di *verifica*$_G$ (normati nella corrispondente sezione di questo documento) e successivamente verrà svolto il *test di accettazione*$_G$, ovvero l'operazione che porterà effettivamente alla validazione del prodotto.

3.5.4 Test di accettazione

Il *test di accettazione*$_G$ coincide con l'operazione di collaudo e deve essere effettuato in presenza del *proponente*$_G$. L'obiettivo è dimostrare che il prodotto soddisfi tutti i requisiti (espressi e impliciti) del *capitolato*$_G$ d'appalto.
La figura principale coinvolta in questo processo è il Verificatore, che dovrà prima occuparsi di effettuare i *test di sistema*$_G$ in un ambiente identico a quello in cui il prodotto verrà rilasciato. Una volta completati tali test con esito positivo, si potrà passare al test di accettazione.

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Scopo

Questa sezione è definita in conformità allo standard *ISO/IEC 12207:1997*$_G$. Il *processo*$_G$ di gestione racchiude *attività*$_G$ e compiti generici, applicabili da qualsiasi entità che debba gestire il proprio processo specifico. Il responsabile è incaricato della gestione del prodotto, del progetto e delle attività, quali il processo di acquisizione, fornitura, sviluppo, funzionamento, manutenzione o supporto.

4.1.2 Descrizione

Le *attività*$_G$ di gestione di *processo*$_G$ definite dallo standard sono:

- Inizio e definizione dello scopo;
- Pianificare: stimare i tempi, le risorse ed i costi;
- Esecuzione e controllo;
- Revisione e valutazione periodica delle attività.

4.1.3 Pianificazione

4.1.3.1 Scopo

In conformità con lo standard *ISO/IEC 12207:1995*$_G$, il responsabile predispone i piani per l'esecuzione del *processo*$_G$. I piani dovranno contenere la descrizione delle *attività*$_G$ e dei compiti associati e l'identificazione dei prodotti software che saranno forniti. Tali piani includeranno, ma non si limiteranno a quanto segue:

a) Programmi per il tempestivo completamento delle attività;

b) Stima dello sforzo;

c) Risorse adeguate necessarie per eseguire i compiti;

d) Attribuzione dei compiti;

e) Attribuzione delle responsabilità;

f) Quantificazione dei rischi associati alle attività o al processo stesso;

g) Misure di controllo della qualità da impiegare durante tutto il processo;

h) Costi connessi all'esecuzione del processo;

i) Fornitura di ambiente e infrastrutture.

4.1.3.2 Descrizione

L'attività di pianificazione verrà articolata come segue nella presente sezione:

1. Assegnazione dei ruoli (§4.1.3.3)
2. Ticketing (§4.1.3.4)

4.1.3.3 Assegnazione ruoli

Il Responsabile di Progetto si occupa di suddividere i ruoli e di assegnare questi ultimi ai membri del gruppo, garantendo che ognuno di essi assuma, nel corso del progetto, almeno una volta ogni ruolo. La designazione dei ruoli sarà effettuata su base settimanale e sarà formalizzata ogni sabato, attraverso la discussione e l'assegnazione che avverranno nel corso della riunione interna programmata per tale giorno. Di seguito la descrizione dei ruoli:

Responsabile

È responsabile della pianificazione, dell'esecuzione e del monitoraggio delle attività del progetto secondo le specifiche previste dal gruppo. Le sue principali responsabilità sono:

- Pianificare e definire obiettivi, scadenze e risorse;
- Coordinare il lavoro del team;
- Monitorare il progresso del progetto;
- Gestire i rischi;
- Comunicare con il team e le parti interessate.

Amministratore

Svolge un ruolo di supporto nella gestione della documentazione, delle risorse e delle comunicazioni del progetto. Le sue principali responsabilità sono:

- Mantenere la documentazione del progetto;
- Gestire risorse come strumenti e infrastrutture;
- Gestire le comunicazioni interne.

Analista

È responsabile dell'*Analisi dei Requisiti*$_G$ del progetto e della definizione delle specifiche. Le sue principali responsabilità sono:

- Raccogliere e analizzare i requisiti del cliente;
- Definire specifiche funzionali e tecniche;
- Collaborare con il progettista per creare soluzioni.

Progettista

Traduce i requisiti in soluzioni tecniche e architetturali, definendo la struttura del sistema. Le sue principali responsabilità sono:

- Creare un'*architettura*$_G$ del sistema;
- Definire il design dettagliato;
- Collaborare con i programmatori per l'implementazione.

Programmatore

È responsabile della scrittura del codice e dell'implementazione delle soluzioni tecniche definite dal progettista. Le sue principali responsabilità sono:

- Scrivere codice in base alle specifiche;
- Risolvere i problemi tecnici;

- Testare il proprio codice.

Verificatore

È responsabile della *verifica*$_G$ e della *validazione*$_G$ del lavoro svolto, assicurandosi che il prodotto soddisfi gli standard di qualità e i requisiti del cliente. Le sue principali responsabilità sono:

- Effettuare test di verifica e validazione;
- Identificare e segnalare difetti;
- Assicurarsi che il prodotto rispetti gli standard di qualità.

4.1.3.4 Ticketing

Il gruppo *torchlight* adotta *Jira*$_G$ come *Issue Tracking System*$_G$ (ITS) per una gestione efficiente dei compiti. Durante le riunioni interne, il responsabile collabora con il team per creare e assegnare compiti in modo concordato, assicurando che ciascun membro sia d'accordo sulle proprie responsabilità e le relative scadenze. Questo approccio promuove una partecipazione attiva e consapevolezza all'interno del gruppo. Ogni membro può monitorare i progressi dei compiti direttamente dalla *Board* di Jira, garantendo tracciabilità e comprensione dello stato del progetto.

Ogni volta che è necessario completare un compito, viene seguita la seguente procedura:

1. **Assegnazione del compito**: il responsabile crea e assegna il task utilizzando la piattaforma Jira;
2. **Creazione del *branch*$_G$ su GitHub**: l'incaricato apre un branch su GitHub, seguendo la denominazione consigliata da Jira. Ciò collega il branch direttamente al task in questione;
3. **Avanzamento del lavoro**: al momento dell'avvio del lavoro di produzione, il task viene segnalato come "*in corso*" su Jira;

4. **Apertura della *pull request*$_G$**: una volta completato il lavoro di produzione, viene aperta una pull request su GitHub, e viene assegnato il verificatore. Simultaneamente, vengono aggiornate le ore di lavoro svolte su Jira;
5. **Marcatura del task in revisione**: il task viene contrassegnato come "*in revisione*" su Jira;
6. **Verifica del lavoro svolto**: il verificatore si assicura della qualità del lavoro eseguito.

Se la verifica ha esito positivo:

a) Il verificatore conferma la validità dei cambiamenti su GitHub e approva la pull request;

b) Il task viene contrassegnato come "*approvato*" su Jira;

c) Il responsabile esegue il *merge*$_G$ del branch;

d) Il responsabile elimina il branch;

e) Il task viene contrassegnato come "*completato*" su Jira.

In caso di un esito negativo della verifica, il processo viene adeguato nel modo seguente:

a) Il verificatore fornisce una lista di suggerimenti per le modifiche tramite GitHub. In questo modo, è sempre tracciabile quali modifiche sono state richieste;

b) L'incaricato implementa le modifiche richieste e il processo ritorna al punto 4.

I task sono creati dal responsabile e hanno:

- **Titolo**;
- **Descrizione**;
- **Assegnatario**: membro del gruppo che ha il compito di occuparsi di questo task;
- **Data di inizio**: data in cui il task in questione viene aperto;
- **Data di fine**: data entro la quale il task deve essere completato;
- **Etichetta**: parola/e chiave o tag che vengono assegnati ai task per organizzarli e categorizzarli in base a determinati criteri;
- ***Sprint*$_G$**: sprint a cui è associato il task;

- **Versione**: versione del software a cui è associato il task;
- **Epic**: è usato per raggruppare un insieme di task. Una *Epic* rappresenta un lavoro più ampio o una caratteristica significativa che può essere suddivisa in task di minori dimensioni;
- **Reporter**: indica colui che ha originariamente creato il task.

4.1.4 Coordinamento

4.1.4.1 Scopo

Il coordinamento, fondamentale all'interno di un progetto di Ingegneria del Software, si configura come un'attività cruciale per la gestione efficace delle comunicazioni e la pianificazione degli incontri tra le diverse parti interessate. Tale ambito di responsabilità include la gestione tanto delle comunicazioni interne, fra i membri del team di progetto, quanto delle comunicazioni esterne con il proponente e i committenti, al fine di assicurare un avanzamento efficiente del progetto.

4.1.4.2 Descrizione

Nella presente sezione, sarà esplicitato in maniera dettagliata il protocollo relativo alle riunioni interne ed esterne adottato dal gruppo *torchlight*. Si procederà all'analisi delle modalità di coordinamento e delle aspettative nei confronti dei partecipanti, con particolare enfasi sui doveri attribuiti a ciascun membro. Contestualmente, sarà condotta un'indagine approfondita sui verbali interni ed esterni, illustrando il processo di registrazione e documentazione delle discussioni e delle decisioni emerse durante le riunioni. L'obiettivo è instaurare una chiara comprensione di tali pratiche, fondamentali per il successo delle attività progettuali, e promuovere un ambiente di lavoro coeso e informato, in linea con gli standard e le esigenze organizzative.

4.1.4.3 Comunicazioni

Comunicazioni interne

All'interno del nostro team, le comunicazioni interne sono facilitate attraverso l'uso di diverse applicazioni e strumenti, che contribuiscono a una gestione efficace del progetto e alla collaborazione nei seguenti modi:

- **Telegram**: per comunicazioni rapide e istantanee. Ci consente di scambiare informazioni e aggiornamenti cruciali, nonché di discutere di questioni urgenti in tempo reale;
- **Zoom**: rendiamo ampio uso di Zoom per condurre riunioni e conferenze virtuali. Questo strumento ci permette di avere discussioni più approfondite, presentazioni e interazioni visive con i membri del team, il che è particolarmente utile per incontri significativi e sessioni di lavoro collettive;
- **Google Sheets**: costituisce la piattaforma primaria per la nostra gestione della rotazione dei ruoli, per la contabilizzazione delle ore produttive di lavoro e per monitorare il budget rimanente nel contesto del nostro progetto;
- **Overleaf**: ci permette di consentire il lavoro simultaneo di tutti i membri del team, in modo tale da evitare problemi di duplicazione e ridondanza dei file, facilitando così una collaborazione senza intoppi.

Comunicazioni esterne

Il responsabile di progetto ha la responsabilità delle comunicazioni esterne, che avvengono attraverso il seguente indirizzo di posta elettronica del gruppo:

torchlight.swe2324@outlook.com

accessibile a tutti i membri del team.
Inoltre, un'altra forma di comunicazione esterna, che si rivela essere più immediata ed inclusiva, avviene attraverso Zoom, nel quale il gruppo comunica con il *proponente*$_{\boldsymbol{G}}$.
Il coordinamento e la documentazione delle comunicazioni con le parti esterne sono fondamentali per mantenere una comunicazione chiara e trasparente all'interno del team e per

assicurare che tutti i membri siano informati sugli sviluppi e le interazioni con le entità esterne.

Compiti del responsabile

Il responsabile si occupa di:

1. **Pianificazione dell'ordine del giorno**: prima delle riunioni, sarà compito del responsabile stabilire in modo preciso l'ordine del giorno, dettagliando gli argomenti specifici da affrontare e la pianificazione delle attività da svolgere;
2. **Comunicazioni sulle variazioni orarie**: qualora si verifichino variazioni orarie come anticipazioni, ritardi o cancellazioni, il responsabile sarà incaricato di comunicare tempestivamente tali cambiamenti a tutti i membri del gruppo;
3. **Controllo della presenza**: durante le riunioni, sarà sua responsabilità verificare la presenza della maggioranza dei componenti del gruppo, assicurando una partecipazione adeguata;
4. **Conduzione ordinata delle discussioni**: il responsabile guiderà le discussioni in modo ordinato, assicurandosi di raccogliere i pareri dei membri in modo equo e strutturato;
5. **Nomina del segretario**: prima dell'inizio della riunione, il responsabile nominerà un segretario incaricato di redigere il verbale.
6. **Assegnazione di compiti su *Jira*$_G$**: il responsabile sarà responsabile di aprire e assegnare i task su Jira per la stesura, la *verifica*$_G$ e l'approvazione del verbale della riunione;
7. **Approvazione del verbale**: infine, il responsabile effettuerà l'approvazione formale del verbale, garantendo che il resoconto rifletta accuratamente quanto discusso e deciso durante la riunione.

Doveri dei partecipanti

I partecipanti prestano attenzione alla:

1. **Puntualità**: i partecipanti si impegnano ad arrivare puntuali alle riunioni;
2. **Comunicazione tempestiva**: è previsto che i partecipanti avvisino tempestivamente il responsabile in caso di ritardi o assenze;
3. **Partecipazione attiva**: è atteso che i partecipanti partecipino attivamente durante la trattazione degli argomenti elencati nell'ordine del giorno;
4. **Comportamento adeguato**: si richiede ai partecipanti di mantenere un comportamento consono durante lo svolgimento delle riunioni.

4.1.4.4 Gestione delle riunioni

Nel contesto di un'efficace gestione delle riunioni, emerge la necessità di documentare in modo sistematico le discussioni e le decisioni prese durante gli incontri. Al termine di ciascuna riunione, un elemento cruciale è rappresentato dalla redazione del verbale, nel quale vengono riportati i punti salienti della discussione e l'esito delle votazioni. Questo registro, indipendentemente dalla tipologia della riunione, sia essa interna o esterna, assume un ruolo fondamentale nel fornire un resoconto accurato e dettagliato degli avvenimenti, consentendo ai partecipanti di rivedere in seguito i contenuti e le conclusioni emerse durante l'incontro. La pratica di redigere verbali non solo favorisce la trasparenza e la tranciabilità delle decisioni prese, ma costituisce anche uno strumento prezioso per mantenere un chiaro e condiviso registro delle attività svolte.

Riunioni interne

Gli incontri interni solitamente vengono svolti almeno una volta a settimana, di sabato. Un incontro può essere richiesto da qualsiasi membro del gruppo al Responsabile di Progetto, che poi lo organizzerà a in base alla disponibilità di tutti i membri. Dopo aver fissato definitivamente la data e l'ora dell'incontro, questi vengono comunicati a tutti i partecipanti tramite il gruppo Telegram. La modalità di incontro che il nostro gruppo ha scelto è prevalentemente virtuale, viste la difficoltà che sussistono nell'organizzare un incontro di persona. Per mantenere una buona efficienza e per ottimizzare il tempo limitato di ogni incontro interno, viene seguita una linea guida in ogni riunione sincrona:

- Prima di ogni incontro, si stila una scaletta dei principali punti oggetto di discussione durante la riunione;
- Discussione del lavoro che ogni membro del gruppo ha svolto in seguito all'ultimo incontro;
- Discussione di tutti i punti prefissati nella scaletta iniziale, con un confronto su eventuali dubbi;
- Pianificazione delle attività da svolgere per ogni membro del gruppo, fino al prossimo incontro.

Approvazione delle decisioni

Una decisione è ritenuta approvata quando si verificano le seguenti condizioni:

- L'argomento in questione è stato trattato e discusso durante la riunione;
- La maggioranza dei componenti del gruppo è d'accordo sulla decisione da prendere;
- Alla riunione sono presenti almeno 5 dei 7 componenti del gruppo.

Riunioni esterne

La partecipazione congiunta dei membri del gruppo *torchlight* alle riunioni esterne è essenziale per agevolare un dialogo efficace con il *proponente*$_{\boldsymbol{G}}$. Per la gestione di tali incontri con il proponente, il nostro team si avvale della piattaforma *Zoom*$_{\boldsymbol{G}}$, le cui informazioni di accesso saranno comunicate tempestivamente al team.

La dedizione dei membri del gruppo si manifesta attraverso un impegno prioritario a essere costantemente presenti durante le riunioni esterne. Qualora si rendesse necessario, siamo disposti a riassegnare altri obblighi al fine di garantire la partecipazione a questi importanti incontri. Nel caso di eventuali impegni irrinunciabili che potrebbero impedire la presenza di alcuni membri, il responsabile agirà prontamente per comunicare al proponente la situazione e, ove possibile, concordare la riconvocazione della riunione in una data successiva.

Verbali interni ed esterni

Alla fine di ogni incontro:

- Il Responsabile di Progetto si occuperà di scrivere un breve resoconto sulle attività svolte durante la riunione sincrona, per permettere alle persone non presenti di poter rimanere al passo su ciò che è stato svolto, e su ciò che dovranno fare. Questo resoconto verrà diffuso tramite Telegram, così da avere informazioni istantanee senza aspettare la stesura del verbale;
- Stesura del Verbale interno da parte di un membro del gruppo con una breve descrizione dei punti focali dell'incontro svolto.

4.2 Miglioramento

4.2.1 Scopo

Secondo lo standard *ISO/IEC 12207:1997*$_G$ il *processo*$_G$ di miglioramento è un processo per stabilire, valutare, misurare, controllare e migliorare il *ciclo di vita del software*$_G$.

4.2.2 Descrizione

Il processo di miglioramento è composto da 3 fasi:

1. **Stabilimento del Processo (Process Establishment)**: definizione e implementazione del *processo*$_G$ di miglioramento, compreso l'assegnamento di responsabilità e risorse necessarie;
2. **Valutazione del Processo (Process Assessment)**: analisi delle attività correnti e delle relative prestazioni per identificare aree di forza e debolezza;
3. **Miglioramento del Processo (Process Improvement)**: implementazione di azioni correttive o preventive per ottimizzare l'*efficacia*$_G$ e l'*efficienza*$_G$ del processo.

4.2.2.1 Stabilimento e valutazione

L'analisi fornisce un mezzo per ottenere un riscontro sulla reale *efficienza*$_G$, *efficacia*$_G$ e correttezza dei *processi*$_G$ implementati, facilitando l'identificazione tempestiva dei processi

che richiedono miglioramenti.

4.2.2.2 Miglioramento

Nel corso delle *attività*$_G$ e durante la redazione dei documenti, ci si è impegnati a seguire il principio di miglioramento continuo. L'obiettivo è stato individuare con facilità attività, ruoli e potenziali aree di miglioramento, cercando soluzioni innovative o alternative per affrontare le problematiche emerse. Le valutazioni effettuate sono dettagliatamente documentate e confrontabili nelle sezioni dedicate all'interno del *Piano di Qualifica*$_G$ (v1.0.0). In queste sezioni, sono evidenziati i problemi riscontrati dal gruppo e le soluzioni adeguate, personalizzate in base alla specifica casistica. Tale approccio formale è stato adottato per garantire una gestione consapevole e mirata delle sfide incontrate durante il *processo*$_G$ di sviluppo.

4.3 Formazione del personale

4.3.1 Scopo

L'obiettivo di questa sezione è stabilire le regole relative al processo di istruzione dei membri del team *torchlight*, che include lo studio delle tecnologie impiegate per la produzione dei documenti e la realizzazione del prodotto richiesto.

4.3.2 Aspettative

Le aspettative per il processo di istruzione includono:

- Acquisire una solida conoscenza del linguaggio LaTeX;
- 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.

4.3.3 Formazione dei membri del gruppo

L'istruzione di ciascun membro del gruppo avviene in modo indipendente, attraverso la consultazione della documentazione reperibile online e l'utilizzo di materiale fornito sia dal proponente che dai docenti.

A Standard ISO/IEC 12207

ISO/IEC 12207$_G$ è uno standard internazionale che specifica tutti i *processi*$_G$ necessari per lo sviluppo e la manutenzione dei sistemi software. Per ciascun processo, tale standard definisce le *attività*$_G$ da svolgere e i risultati da ottenere. I processi sono suddivisi in tre categorie:

- **Processi Primari**: ne fanno parte le attività direttamente legate allo sviluppo del software;
- **Processi di Supporto**: comprendono la gestione dei documenti e dei processi di controllo della qualità;
- **Processi Organizzativi**: includono gli aspetti manageriali e di gestione delle risorse.

A.1 Processi Primari

A.1.1 Acquisizione

Questo *processo*$_G$ descrive le attività dell'acquirente. Esso ha inizio con la necessità di un *prodotto software*$_G$ o un servizio che soddisfa le necessità del cliente. Il processo inizia con l'individuazione dei requisiti e termina con l'accettazione della fornitura. È composto dalle seguenti attività:

- Acquisition preparation;
- Supplier selection;
- Supplier monitoring;
- Customer acceptance.

A.1.2 Fornitura

Questo *processo*$_G$ descrive le *attività*$_G$ del *fornitore*$_G$. Esso ha lo scopo di fornire al cliente il prodotto software o servizio che soddisfa i requisiti pattuiti. Il processo di Fornitura si compone delle seguenti attività:

- Proposal preparation;
- Contract;
- Planning;
- Execution and control;
- Review and evaluation;
- Release and completion.

A.1.3 Sviluppo

Questo *processo*$_G$ descrive le *attività*$_G$ dello sviluppatore. Esso ha il compito di normare lo sviluppo di un *prodotto software*$_G$ che soddisfi le esigenze del cliente. Le attività di cui si compone tale processo sono le seguenti:

- Requirements elicitation;
- System requirements analysis;
- System architecture design;
- Software requirements analysis;
- Software architecture design;
- Software construction (code and unit test);
- Software integration;
- Software testing;
- System integration;
- System testing;
- Software installation.

A.1.4 Operation

Questo *processo*$_G$ è svolto in concomitanza al processo di Manutenzione. Il suo scopo è di mantenere operativo il sistema e di fornire supporto agli utenti. Si compone delle seguenti *attività*$_G$:

- Operational use;
- Customer support.

A.1.5 Manutenzione

Questo *processo*$_G$ è svolto in concomitanza al processo di Operation. Il suo scopo è di modificare il *prodotto software*$_G$ dopo il suo rilascio per correggerne difetti, migliorarne le prestazioni o adattarlo ad eventuali cambiamenti nell'ambiente operativo. È suddiviso nelle seguenti *attività*$_G$:

- Defect or request for change analysis;
- Change implementation;
- Review/acceptance of changes;
- Migration;
- Software withdraw.

A.2 Processi di supporto

Lo scopo dei *processi*$_G$ di supporto consiste nel supportare gli altri processi in atto. Questi possono essere attivati da un processo primario oppure da un altro processo di supporto.

A.2.1 Gestione della documentazione

Questo *processo*$_G$ ha lo scopo di registrare tutte le informazioni prodotte da un'*attività*$_G$ o da un processo del *ciclo di vita*$_G$ di un *prodotto software*$_G$.

A.2.2 Gestione della configurazione

Questo *processo*$_G$ ha lo scopo di definire e mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto.

A.2.3 Accertamento della qualità

Questo *processo*$_G$ ha lo scopo di assicurare che tutti i prodotti e i processi del *ciclo di vita*$_G$ siano conformi con gli standard ed i requisiti definiti.

A.2.4 Verifica

Questo *processo*$_G$ ha lo scopo di verificare se tutti i *prodotti software*$_G$ realizzati soddisfano i requisiti e le condizioni imposte dalle attività precedenti. Il processo di Verifica deve essere integrato nei processi di Sviluppo, Fornitura e Manutenzione.

A.2.5 Validazione

Questo *processo*$_G$ ha lo scopo di accertarsi che il *prodotto software*$_G$ rispetti tutti i requisiti stabiliti.

A.2.6 Revisione congiunta

Questo *processo*$_G$ mira ad esaminare, insieme alle parti coinvolte, i processi eseguiti rispetto agli obiettivi stabiliti negli accordi, nonché le azioni necessarie per garantire lo sviluppo di un prodotto in linea con i requisiti concordati. Le revisioni sono condotte lungo l'intero *ciclo di vita*$_G$, sia a livello di progetto che tecnico. Le revisioni congiunte coinvolgono i membri interni del team quando si esamina un componente specifico del prodotto, mentre coinvolgono il *fornitore*$_G$ e il *committente*$_G$ quando si esamina l'intero prodotto.

A.2.7 Audit

L'obiettivo di questo *processo*$_G$ è valutare in modo indipendente la conformità di prodotti e processi rispetto ai requisiti, piani e accordi stabiliti. L'*attività*$_G$ di Audit viene condotta da personale che non è stato coinvolto direttamente nello sviluppo dei prodotti, servizi o sistemi soggetti alle revisioni.

A.2.8 Risoluzione dei problemi

Questo *processo*$_G$ ha lo scopo di assicurare che tutti i problemi individuati vengano analizzati e risolti in modo responsabile e documentato.

A.2.9 Usabilità

L'obiettivo di questo *processo*$_G$ è garantire che vengano attentamente valutate e opportunamente gestite le considerazioni e le esigenze espresse dalle parti coinvolte in merito alla facilità d'uso del prodotto finale da parte degli utenti a cui è destinato, al supporto che ne riceverà, alla formazione, al miglioramento della produttività, alla qualità del lavoro e all'accettazione complessiva del prodotto.

A.2.10 Valutazione del prodotto

L'obiettivo primario di questo *processo*$_G$ è garantire, attraverso valutazioni e analisi, che il prodotto risponda adeguatamente alle esigenze esplicite e implicite degli utenti che lo utilizzano.

A.3 Processi organizzativi

I *processi*$_G$ organizzativi sono gestiti dall'organizzazione che li attiva. Il loro scopo è stabilire e mettere in atto una struttura per la gestione dei processi del *ciclo di vita*$_G$ e del personale. In aggiunta, sono utili per la gestione del miglioramento continuo dei processi e della struttura stessa.

A.3.1 Gestione organizzativa

Lo scopo di questo *processo*$_G$ è di organizzare, monitorare e controllare l'avvio e le prestazioni di un processo per conseguire i suoi obiettivi in conformità con quelli di business dell'organizzazione. Il processo è attivato da un'organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti.

A.3.2 Gestione delle infrastrutture

Questo *processo*$_G$ ha lo scopo di istituire e manutenere un'infrastruttura stabile ed affidabile necessaria a supportare tutti gli altri processi. L'infrastruttura può comprendere hardware, software, strumenti, tecniche, standard e funzionalità per lo sviluppo, operazioni o manutenzione.

A.3.3 Miglioramento

Questo *processo*$_G$ ha lo scopo di istituire, valutare, misurare, controllare e migliorare il *ciclo di vita del software*$_G$.

A.3.4 Formazione

Questo *processo*$_G$ ha lo scopo di fornire all'organizzazione risorse umane adeguate e di mantenere il personale adeguatamente istruito.

B Standard di qualità ISO/IEC 9126

Lo standard per la valutazione della qualità che il gruppo *torchlight* 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**: è la capacità del software di fornire un insieme di funzionalità che soddisfino opportunamente le esigenze dell'utente;
- **Accuratezza**: è la capacità del software di fornire risultati corretti, graditi ed in linea con le aspettative;
- **Interoperabilità**: è la capacità del software di interagire con gli altri sistemi;
- **Sicurezza**: è la capacità del software di proteggere le informazioni e i dati sensibili, prevenendo accessi non autorizzati al sistema;
- **Aderenza alle funzionalità**: è la capacità del software di aderire agli standard, convenzioni e regolamenti legali.

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**: è la capacità del software di mantenere alto il livello di stabilità durante il suo utilizzo;
- **Tolleranza ai guasti**: è la capacità del software di gestire gli errori in modo appropriato;
- **Recuperabilità**: è la capacità del software di ristabilire un alto livello di prestazioni in caso di malfunzionamenti.

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 accattivan-

te 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*$_G$ 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:

- ***Analisi dei Requisiti***$_G$: L'Analisi dei Requisiti valuta quanto sia agevole comprendere la struttura del codice sorgente e individuare eventuali errori. Un software con 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;
- **Copertura dei *test*$_G$**: L'utilizzo di test 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;
- **Compatibilità**: 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*$_G$ 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*$_G$ di sviluppo per migliorare la qualità del codice e facilitare la manutenzione.

C.2 Metriche esterne

Le *metriche*$_G$ 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*$_G$ 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*$_G$ per la qualità di *processo*$_G$ valutano la qualità dei processi adottati durante lo sviluppo del software. Esse includono metriche per miglioramento, fornitura, *codifi-*

ca$_G$ e documentazione, fornendo indicazioni su come i processi possono essere ottimizzati per migliorare la qualità del prodotto finale.

C.4.1 Miglioramento

Queste *metriche*$_G$ valutano l'*efficacia*$_G$ dei processi di miglioramento continuo implementati nel *ciclo di vita*$_G$ dello sviluppo del software.

C.4.2 Fornitura

Le *metriche*$_G$ di fornitura misurano la qualità del *processo*$_G$ di consegna del software, compreso il rispetto dei tempi, la gestione delle risorse e la conformità agli standard.

C.4.3 Codifica

Queste *metriche*$_G$ valutano la qualità del *processo*$_G$ di scrittura del codice, inclusa la correttezza sintattica, la chiarezza e la conformità agli standard di *codifica*$_G$.

C.4.4 Documentazione

Misurano la qualità della documentazione associata al software, fornendo indicazioni sulla chiarezza e completezza della documentazione.

C.5 Metriche per la qualità di prodotto

Le *metriche*$_G$ 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.

C.5.1 Funzionalità

Queste *metriche*$_G$ valutano la completezza e la correttezza delle funzionalità del software, assicurando che risponda adeguatamente ai requisiti.

C.5.2 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 $metriche_{\boldsymbol{G}}$ di affidabilità, valutando la capacità del software di mantenere prestazioni stabili e coerenti anche dopo le modifiche.

C.6.1 Affidabilità

Questa $metrica_{\boldsymbol{G}}$ 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.