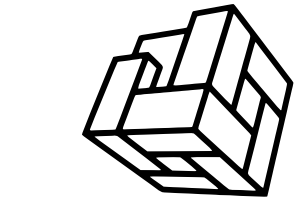
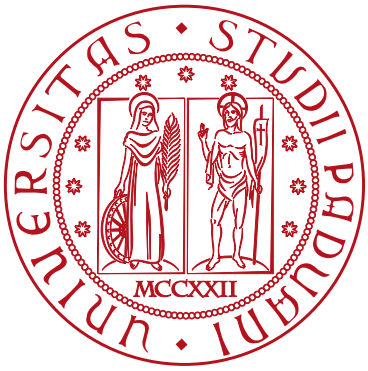



**PEBKAC**
Gruppo: 11
Email: pebkacswe@gmail.com
Docs: https://pebkac-swe-group-11.github.io
GitHub: https://github.com/PEBKAC-SWE-Group-11




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

# Norme di progetto

**Informazioni sul documento**:

| | |
|---|---|
| **Verificatori** | Alessandro Benin<br>Tommaso Zocche<br>Derek Gusatto<br>Davide Martinelli |
| **Redattori** | Derek Gusatto<br>Matteo Piron<br>Matteo Gerardin |
| **Uso** | Interno |
| **Destinatari** | Tullio Vardanega<br>Riccardo Cardin<br>Gruppo PEBKAC |

# Registro delle versioni

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 1.0.0 | 2025-01-26 | Derek Gusatto | Responsabile | Approvazione e rilascio |
| 0.4.0 | 2025-01-26 | Davide Martinelli | Amministratore | Verifica intero documento |
| 0.3.1 | 2025-01-23 | Matteo Gerardin | Amministratore | Inserimento redattori e verificatori nella prima pagina |
| 0.3.0 | 2025-01-15 | Derek Gusatto | Verificatore | Verifica paragrafi aggiunti da V0.2.0 |
| 0.2.2 | 2025-01-11 | Matteo Gerardin | Amministratore | Configuration Control (fino a §4.2.5.2) |
| 0.2.1 | 2024-12-28 | Matteo Gerardin | Amministratore | Inizio §4.2.2 Configuration Control (fino a §4.2.2.3.5) |
| 0.2.0 | 2024-12-17 | Tommaso Zocche | Verificatore | Verifica parziale |
| 0.1.5 | 2024-12-17 | Tommaso Zocche | Verificatore | Correzioni “strutturali” e typo |
| 0.1.4 | 2024-12-5 | Matteo Piron | Amministratore | §6 Metriche di qualità |
| 0.1.3 | 2024-11-25 | Derek Gusatto | Amministratore | §4.2 Configuration Management (struttura e scopo) |
| 0.1.2 | 2024-11-22 | Derek Gusatto | Amministratore | §3.2.2 Analisi dei requisiti |
| 0.1.1 | 2024-11-20 | Derek Gusatto | Amministratore | §3.1 Fornitura |
| 0.1.0 | 2024-11-19 | Alessandro Benin | Verificatore | Verifica parziale |
| 0.0.3 | 2024-11-17 | Derek Gusatto | Amministratore | §5.1 Gestione organizzativa |
| 0.0.2 | 2024-11-16 | Derek Gusatto | Amministratore | §4.1 Documentazione |
| 0.0.1 | 2024-11-14 | Derek Gusatto | Amministratore | §1 Introduzione, §2 Standard ISO/IEC 12207:1195 |

# Indice

# Elenco delle tabelle

# 1 Introduzione

## 1.1 Scopo del documento

Il presente documento ha l'obiettivo di definire le *best practices*$_{\text{G}}$ e il *way of working*$_{\text{G}}$ che i componenti del team *PEBKAC* hanno l'obbligo di rispettare per l'intero svolgimento del progetto. L'intento è di garantire un metodo di lavoro omogeneo, verificabile e migliorabile nel tempo. La creazione delle norme è progressiva e incrementale nel tempo per consentire al team di apportare aggiornamenti continui in risposta alle esigenze e alle problematiche incorse durante lo sviluppo dell'intero progetto.

## 1.2 Scopo del prodotto

Il progetto "Vimar GENIALE" mira a sviluppare un'applicazione intelligente che supporti installatori elettrici nell'uso di dispositivi *Vimar*$_{\text{G}}$, facilitando l'accesso alle informazioni tecniche sui prodotti, rispondendo a domande poste in linguaggio naturale. La tecnologia alla base prevede l'uso di modelli di *LLM*$_{\text{G}}$ e di tecniche *RAG*$_{\text{G}}$, con una struttura di gestione basata su *container*$_{\text{G}}$ e integrata in un ambiente *cloud*$_{\text{G}}$. Il sistema include tre componenti principali: una *applicativo web responsive*$_{\text{G}}$, un *applicativo server*$_{\text{G}}$ e un'*infrastruttura cloud-ready*$_{\text{G}}$.

## 1.3 Glossario

Per evitare ambiguità relative al linguaggio utilizzato nei documenti, viene fornito il Glossario V1.0.0, nel quale si possono trovare tutte le definizioni di termini che hanno un significato specifico che vuole essere disambiguato. Tali termini sono marcati con una G a pedice.

## 1.4 Riferimenti

### 1.4.1 Riferimenti normativi

- Regolamento del progetto didattico
  https://www.math.unipd.it/ tullio/IS-1/2024/Dispense/PD1.pdf
  (Ultimo accesso 2024-11-14)
- ISO/IEC 12207:1995 Information technology - Software life cycle processes
  https://www.math.unipd.it/ tullio/IS-1/2010/Approfondimenti/A03.pdf
  (Ultimo accesso 2024-11-14)

### 1.4.2 Riferimenti informativi

- Capitolato C2
  https://www.math.unipd.it/ tullio/IS-1/2024/Dispense/PD1.pdf
  (Ultimo accesso 2024-11-14)
- Capitolato C2 - slides
  https://www.math.unipd.it/ tullio/IS-1/2024/Dispense/PD1.pdf
  (Ultimo accesso 2024-11-14)

- Documentazione$_{G}$ GitHub$_{G}$
  https://docs.github.com/en
  (Ultimo accesso 2024-11-14)

# 2 Standard ISO/IEC 12207:1995

Il gruppo ha deciso di applicare nelle proprie modalità di lavoro e quindi nel presente documento di adottare lo *standard ISO/IEC 12207:1995 Information technology - Software life cycle processes*. In questa sezione del documento si possono trovare i criteri di applicazione e i processi definiti nell'ambito di questo standard.

## 2.1 Processi del ciclo di vita del software$_G$

Questo documento è usato per normare il *way of working*$_G$ del gruppo, in particolare l'organizzazione dei processi del ciclo di vita del *software*$_G$ secondo lo *standard ISO/IEC 12207:1995 Information technology - Software life cycle processes*, questi sono organizzati in una organizzazione gerarchica in cui ogni processo è costituito da un insieme di attività, le quali possono prevedere delle procedure e un elenco di strumenti usati per lo svolgimento.

### 2.1.1 Processi primari

Lo standard adottato presenta cinque processi primari (*Acquisizione, Fornitura, Sviluppo, Operazione, Manutenzione*), ma all'interno del contesto del progetto universitario in atto, il gruppo non ritiene i processi di *Operazione* e *Manutenzione* pertinenti, mentre il processo di *Acquisizione* è di competenza del *committente*$_G$, pertanto, il gruppo decide di escluderli dalla presentazione nel documento. I processi primari presentati nel presente documento sono:

- **Fornitura**: definisce le attività del fornitore, l'organizzazione che fornisce il prodotto *software*$_G$ all'acquirente;
- **Sviluppo**: definisce le attività dello sviluppatore, l'organizzazione che definisce e sviluppa il prodotto *software*$_G$.

### 2.1.2 Processi di supporto

I processi di supporto presentati nel presente documento sono:

- **Documentazione**$_G$: definisce le attività per la registrazione delle informazioni prodotte da un processo del ciclo di vita;
- **Configuration Management**$_G$: definisce le attività di gestione della configurazione;
- **Accertamento di qualità**$_G$: definisce le attività per assicurare in modo oggettivo che i prodotti e i processi *software*$_G$ siano conformi ai requisiti$_G$ specificati;
- **Verifica**$_G$: definisce le attività per verificare il prodotto *software*$_G$;
- **Validazione**$_G$: definisce le attività per validare il prodotto *software*$_G$;
- **Risoluzione dei problemi**: definisce un processo per analizzare e risolvere i problemi di qualsiasi natura o origine, sorti durante l'esecuzione di processi.

### 2.1.3 Processi organizzativi

I processi organizzativi presentati nel presente documento sono:

- **Gestione organizzativa**: definisce le attività dell'acquirente, l'organizzazione che acquisisce un prodotto *software*$_{\mathrm{G}}$;
- **Infrastruttura**: definisce le attività del fornitore, l'organizzazione che fornisce il prodotto *software*$_{\mathrm{G}}$ all'acquirente;
- **Miglioramento**: definisce le attività dello sviluppatore, l'organizzazione;
- **Formazione**: definisce le attività atte a provvedere una adeguata formazione del gruppo.

### 2.1.4 Ruoli

I ruoli definiti all'interno di questo progetto didattico universitario sono:

- **Docente del corso**: *committente*$_{\mathrm{G}}$;
- **Azienda proponente**: cliente e mentore;
- **Gruppo di lavoro**: fornitore.

# 3 Processi Primari

## 3.1 Fornitura

### 3.1.1 Scopo

La fornitura è il processo che descrive le attività svolte dal fornitore, coinvolge pianificazione, acquisizione e gestione delle risorse necessarie. Il processo determina le procedure e le risorse necessarie per gestire e garantire il progetto. L'obiettivo di questo processo è garantire l'*efficienza*$_{\text{G}}$ e la conformità ai *requisiti*$_{\text{G}}$ del progetto per raggiungere gli obiettivi stabiliti dal proponente.

### 3.1.2 Implementazione

Il processo di fornitura è composto delle seguenti fasi:

1. **Risposta alla richiesta**: il fornitore, dopo aver analizzato i *requisiti*$_{\text{G}}$ di una richiesta del proponente (il *Capitolato*$_{\text{G}}$) prepara in risposta una proposta;
2. **Negoziazione**: il fornitore negozia e stipula un contratto con il proponente;
3. **Pianificazione**: il fornitore rivede i *requisiti*$_{\text{G}}$ e valuta le opzioni per lo sviluppo del prodotto *software*$_{\text{G}}$ in base ad un'analisi dei rischi associati alle varie opzioni per definire la struttura di un piano di gestione del progetto al fine di garantire la qualità del prodotto finale;
4. **Esecuzione e controllo**: il fornitore esegue il piano di gestione del progetto, monitorando il progresso e la qualità del prodotto per tutto il ciclo di vita del prodotto;
5. **Revisione**: il fornitore coordina le comunicazioni con il proponente e partecipa a riunioni e revisioni. Il fornitore *verifica*$_{\text{G}}$ e convalida il processo per dimostrare che i prodotti e i processi soddisfano i *requisiti*$_{\text{G}}$;
6. **Consegna**: il fornitore consegna il prodotto finale, fornendo assistenza al proponente a supporto del prodotto consegnato.

### 3.1.3 Gestione

Al fine di identificare e comprendere i bisogni del Proponente, per poter individuare i *requisiti*$_{\text{G}}$ e i vincoli del progetto, deve essere mantenuta costante comunicazione con il Proponente, mediante riunioni SAL periodiche calendarizzate, in presenza o su *Microsoft Teams*$_{\text{G}}$ e con scambio di messaggi su *Microsoft Teams*$_{\text{G}}$ e mail qualora fosse necessario. Il dialogo continuo permette anche una valutazione costante dell'operato del fornitore, in modo da apportare correzioni, integrazioni e miglioramenti in modo tempestivo, incrementale e costruttivo.

### 3.1.4 Documentazione fornita

Sono di seguito elencati i documenti che PEBKAC si impegna a consegnare ai *Committenti*$_{\text{G}}$ e al Proponente:

#### 3.1.4.1 Piano di Progetto

Il Piano di Progetto V1.0.0, redatto dal *Responsabile*$_{\text{G}}$ con l'aiuto degli Amministratori, offre una guida per la pianificazione l'esecuzione e il controllo del progetto e viene utilizzato come punto di partenza principale per il monitoraggio del progresso del progetto, la gestione dei rischi e la comunicazione tra proponente e fornitore. Il Piano di Progetto comprende:

- Calendario di Progetto;
- Stima dei costi di realizzazione;
- Rischi e relativa mitigazione;
- Pianificazione e modello di sviluppo;
- Preventivo e *consuntivo*$_{\text{G}}$;
- Retrospettiva.

#### 3.1.4.2 Analisi dei requisiti

L'Analisi dei Requisiti V1.0.0, redatto degli *Analisti*$_{\text{G}}$, è un documento fondamentale che ha l'obiettivo principale di definire nel dettaglio le funzionalità che il prodotto deve necessariamente avere per soddisfare a pieno le richieste del Proponente. Il documento di Analisi dei Requisiti è formato da una serie di definizioni essenziali:

- **Attori**$_{\mathbf{G}}$: vengono definite entità e persone che interagiscono col *sistema*$_{\text{G}}$;
- **Casi d'uso**: vengono descritti narrativamente degli scenari specifici che descrivono come gli attori interagiscono col *sistema*$_{\text{G}}$. Lo scopo dei casi d'uso è offrire una visione semplice e chiara delle azioni eseguibili all'interno del sisitema e delle interazioni degli utenti con lo stesso. Per ciascun caso d'uso viene fornito un elenco delle azioni dell'*attore*$_{\text{G}}$ per attivare il caso d'uso, facilitando la comprensione dei requisiti corrispondenti;
- **Requisiti**: vengono individuati i requisiti obbligatori, desiderabili e opzionali e la loro categorizzazione in:
  - **Requisiti**$_{\mathbf{G}}$ **funzionali**: specificano le operazioni che il *sistema*$_{\text{G}}$ deve essere in grado di eseguire;
  - **Requisiti**$_{\mathbf{G}}$ **di qualità**: definiscono gli standard e gli attributi che il *software*$_{\text{G}}$ deve possedere per garantire prestazioni, affidabilità, sicurezza e usabilità ottimali;
  - **Requisiti**$_{\mathbf{G}}$ **di vincolo**: definiscono vincoli e limitazioni che il *sistema*$_{\text{G}}$ deve rispettare. Possono includere restrizioni tecnologiche, normative o di risorse.

#### 3.1.4.3 Piano di Qualifica

Il Piano di Qualifica V1.0.0, redatto dall'*amministratore*$_{\text{G}}$, descrive gli approcci e le strategie che il gruppo ha adottato per garantire la qualità del prodotto. Lo scopo di questo documento è quello di definire le modalità di *verifica*$_{\text{G}}$e *validazione*$_{\text{G}}$, oltre che gli standard e le procedure di qualità che il gruppo ha deciso di adottare per il ciclo di vita del progetto.
Si compone delle sezioni riguardanti:

- **Qualità di processo**: vengono definiti standard e procedure adottate per garantire la qualità durante tutto lo sviluppo del progetto. Vengono incluse anche informazioni sulle attività di gestione della qualità, i metodi utilizzati e le misurazioni dei processi stessi;
- **Qualità di prodotto**: vengono definiti standard, specifiche e caratteristiche che il prodotto deve soddisfare per essere considerato di qualità. Vengono incluse anche metriche e criteri di valutazione utilizzati per misurare la qualità del prodotto;
- **Specifiche dei test**: vengono definite specifiche dettagliate dei test che verranno condotti durante lo sviluppo del progetto;
- **Cruscotto delle metriche**: viene fatto un resoconto delle attività di valutazione effettuate durante il progetto per tracciare l'andamento dello stesso rispetto a obiettivi e aspettative e per identificare eventuali azioni correttive necessarie.

#### 3.1.4.4 Glossario

Il Glossario V1.0.0 serve come un catalogo completo dei termini tecnici impiegati all'interno del progetto, fornendo definizioni chiare e precise. L'obiettivo di questo documento previene fraintendimenti a favore di una comprensione condivisa della terminologia specifica, migliorando la coerenza e la qualità della *documentazione*$_{\text{G}}$ prodotta dal gruppo.

### 3.1.5 Strumenti

Gli strumenti utilizzati per il processo di fornitura sono:

- Google Calendar;
- *Google Sheets*$_{\text{G}}$;
- Microsoft PowerPoint;
- *Microsoft Teams*$_{\text{G}}$.

## 3.2 Sviluppo

### 3.2.1 Scopo

Il processo di sviluppo rappresenta la serie di attività svolte dal team PEBKAC al fine di implementare il prodotto *software*$_{\text{G}}$, rispettando le scadenze e i *requisiti*$_{\text{G}}$ concordati col Proponente. Il processo è suddiviso nelle seguenti attività:

- Analisi dei requisiti,
- Progettazione;
- Codifica;
- Testing;
- Integrazione *software*$_{\text{G}}$.

### 3.2.2 Analisi dei Requisiti

#### 3.2.2.1 Scopo

Lo scopo dell'analisi dei requisiti è comprendere e definire in modo chiaro e completo le necessità e le aspettative del Proponente e degli utenti relativamente al prodotto *software*$_{\text{G}}$.

#### 3.2.2.2 Implementazione

L'analisi dei requisiti, raccolta nel documento Analisi dei Requisiti V1.0.0, viene svolta secondo le seguenti fasi:

1. Studio del *capitolato*$_{\text{G}}$ e delle esigenze del Proponente;
2. Individuazione dei casi d'uso e dei *requisiti*$_{\text{G}}$;
3. Confronto con il Proponente su quanto prodotto;
4. Divisione dei *requisiti*$_{\text{G}}$ nelle categorie individuate e applicazione dei quanto emerso nella discussione col Proponente.

L'attività di analisi può essere svolta in modo incrementale, quindi le sue fasi possono essere svolte più volte durante lo sviluppo del progetto.
L'Analisi dei Requisiti V1.0.0 contiene:

- **Introduzione**: descrive lo scopo del documento, del prodotto e i riferimenti utilizzati;
- **Descrizione**: esplicita le funzionalità attese del prodotto;
- **Attori$_{\text{G}}$**: descrive gli utilizzatori del prodotto;
- **Casi d'uso**: individua le possibili interazioni tra gli *attori*$_{\text{G}}$ e il *sistema*$_{\text{G}}$;
- **Requisiti$_{\text{G}}$**: elenca le caratteristiche da soddisfare;

#### 3.2.2.3 Casi d'uso

I casi d'uso sono strutturati nel seguente modo:

- **Attore**$_{\mathrm{G}}$: l'*attore*$_{\mathrm{G}}$ che intende compiere lo scopo rappresentato dal caso d'uso;
- **Precondizioni**: stato in cui il *sistema*$_{\mathrm{G}}$ si deve trovare prima dell'avvio della funzionalità rappresentata dal caso d'uso;
- **Postcondizioni**: stato in cui il *sistema*$_{\mathrm{G}}$ si troverà dopo che l'utente avrà terminato lo scopo rappresentato dal caso d'uso;
- **Scenario principale**: descrizione della funzionalità rappresentata dal caso d'uso;
- **Scenari secondari** (se necessario);
- **Estensioni** (se presenti);
- **Specializzazioni** (se presenti).

##### 3.2.2.3.1 Notazione

i casi d'uso seguono la seguente notazione: **UC[Codice] - [Titolo]** in cui:

- **UC** sta per Use Case;
- **[Codice]** è l'identificativo univoco del caso d'uso. Si tratta di un numero intero progressivo assegnato in base all'ordine di descrizione, se il caso d'uso non ha padre, altrimenti se si tratta di un sottocaso d'uso si segue la notazione **[Codice_padre]-[Numero_figlio]**, ricorsivamente senza porre limite alla profondità della gerarchia;
- **[Titolo]** è il titolo del caso d'uso.

##### 3.2.2.3.2 Diagrammi UML$_{\mathrm{G}}$

Un *diagramma dei casi d'uso*$_{\mathrm{G}}$ è uno strumento di modellazione che rappresenta visivamente le funzionalità di un *sistema*$_{\mathrm{G}}$ e le modalità con cui gli utenti interagiscono con esso. È particolarmente utile nella progettazione di sistemi poiché offre una rappresentazione intuitiva delle dinamiche operative e delle interazioni tra *attori*$_{\mathrm{G}}$ e *sistema*$_{\mathrm{G}}$, senza entrare nei dettagli implementativi. I componenti principali di un *diagramma dei casi d'uso*$_{\mathrm{G}}$ sono:

1. **Attori**$_{\mathrm{G}}$: gli *attori*$_{\mathrm{G}}$ rappresentano entità esterne (umane o meno) che interagiscono con il *sistema*$_{\mathrm{G}}$ e sono raffigurati con un'icona stilizzata e un'etichetta identificativa. Possono essere generalizzati: un *attore*$_{\mathrm{G}}$ generico può avere *attori*$_{\mathrm{G}}$ più specifici che ne ereditano le funzionalità e aggiungono comportamenti contestuali;
2. **Casi d'uso**: un caso d'uso descrive un'operazione che un utente può compiere attraverso il *sistema*$_{\mathrm{G}}$. Ogni caso d'uso ha un'identificazione univoca e una breve descrizione della funzione. Può includere sequenze di azioni che illustrano le possibili interazioni con il *sistema*$_{\mathrm{G}}$ ed è collegato agli *attori*$_{\mathrm{G}}$ autorizzati tramite linee continue.

Nei diagrammi in questione poi possono comparire delle relazioni:

1. **Generalizzazioni**: le generalizzazioni possono riguardare sia gli *attori*$_{\text{G}}$ che i casi d'uso. Gli *attori*$_{\text{G}}$ o i casi figli ereditano le funzionalità dei genitori, aggiungendo aspetti specifici. La relazione è rappresentata con una freccia continua e un triangolo vuoto bianco;

2. **Inclusioni**: si verificano quando un caso d'uso ne richiama un altro in modo obbligatorio. Questo favorisce la riduzione della duplicazione e il riutilizzo delle strutture. La relazione è indicata con una freccia tratteggiata e l'etichetta "include";

3. **Estensioni**: rappresentano relazioni condizionali in cui un caso d'uso aggiuntivo viene eseguito solo in circostanze particolari, interrompendo temporaneamente il flusso principale. La relazione è raffigurata con una freccia tratteggiata e l'etichetta "extend".

#### 3.2.2.4 Requisiti

##### 3.2.2.4.1 Notazione

Ogni *requisito*$_{\text{G}}$ analizzato sarà identificato univocamente da una sigla del tipo **R[Tipo].[Importanza].[Codice]** nella quale:

- [**R**] sta per *Requisito*$_{\text{G}}$;
- [**Tipo**] può essere:
  - **F** per Funzionale;
  - **Q** per Qualità;
  - **V** per Vincolo.
- [**importanza**] classifica i *requisiti*$_{\text{G}}$ in:
  - **O** per Obbligatorio;
  - **D** per Desiderabile;
  - **P** per Opzionale.
- [**Codice**] identifica univocamente i *requisiti*$_{\text{G}}$ per ogni tipologia. È un numero intero progressivo univoco assegnato in ordine di importanza se il *requisito*$_{\text{G}}$ non ha padre, se invece si tratta di un sotto-*requisito*$_{\text{G}}$ segue il formato [**Codice_padre**].[**Numero_figlio**] e trattandosi di una struttura ricorsiva non c'è limite alla profondità della gerarchia.

##### 3.2.2.4.2 Suddivisione

1. **Requisiti$_{\text{G}}$ Funzionali**: descrivono le funzionalità del *sistema*$_{\text{G}}$, le azioni che il *sistema*$_{\text{G}}$ può compiere e le informazioni che il *sistema*$_{\text{G}}$ può fornire. Seguendo la notazione sopra riportata, si possono partizionare in:
   - RF.O - *Requisito*$_{\text{G}}$ Funzionale Obbligatorio;
   - RF.D - *Requisito*$_{\text{G}}$ Funzionale Desiderabile;
   - RF.P - *Requisito*$_{\text{G}}$ Funzionale Opzionale;

2. **Requisiti$_G$ di Qualità**: descrivono come un *sistema*$_G$ deve essere, o come il *sistema*$_G$ deve essere visualizzato, per soddisfare le esigenze dell'utente. Seguendo la notazione sopra riportata, si possono partizionare in:

    - RQ.O - *Requisito*$_G$ di Qualità Obbligatorio;
    - RQ.D - *Requisito*$_G$ di Qualità Desiderabile;
    - RQ.P - *Requisito*$_G$ di Qualità Opzionale;

3. **Requisiti$_G$ Funzionali**: descrivono i limiti e le restrizioni normative/legislative che un *sistema*$_G$ deve rispettare per soddisfare le esigenze dell'utente. Seguendo la notazione sopra riportata, si possono partizionare in:

    - RV.O - *Requisito*$_G$ di Vincolo Obbligatorio;
    - RV.D - *Requisito*$_G$ di Vincolo Desiderabile;
    - RV.P - *Requisito*$_G$ di Vincolo Opzionale;

# 4 Processi di Supporto

## 4.1 Documentazione

### 4.1.1 Scopo

Il processo di *documentazione*$_{\mathrm{G}}$ procede sempre di pari passo con tutte le attività di sviluppo, con l'obiettivo di fornire tutte le informazioni necessarie, sotto forma di testo scritto facilmente consultabile, inerenti al prodotto e alle attività stesse. Oltre a svolgere un *ruolo*$_{\mathrm{G}}$ essenziale nella descrizione del prodotto per coloro che lo sviluppano, lo distribuiscono e lo utilizzano, la *documentazione*$_{\mathrm{G}}$ svolge un *ruolo*$_{\mathrm{G}}$ di storicizzazione e di supporto alla manutenzione.

### 4.1.2 Documenti

In questa sezione viene descritto il piano che identifica i documenti da produrre durante il ciclo di vita del prodotto *software*$_{\mathrm{G}}$. Tutti i documenti da redigere sono presentati nella tabella che segue, vengono esclusi i documenti presentati per la candidatura per il progetto didattico, quali *Lettera di presentazione*, *Preventivo dei costi e assunzione degli impegni* e *Analisi dei capitolati*.

| **Nome** | **Scopo** | **Redattore** | **Destinatari** | **Consegne** |
|---|---|---|---|---|
| Analisi dei requisiti | Definizione dei requisiti utente | *Analista*$_{\mathrm{G}}$ | Azienda proponente, Docenti | $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |
| Norme di progetto | Regolamento normativo del gruppo | *Amministratore*$_{\mathrm{G}}$, *Responsabile*$_{\mathrm{G}}$ | Docenti | $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |
| Piano di Progetto | Definizione temporale scadenze e progressi | *Responsabile*$_{\mathrm{G}}$ | Docenti | $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |
| Piano di qualifica | Definizione qualità e testing | *Amministratore*$_{\mathrm{G}}$ | Docenti | $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |
| Verbali esterni | Tracciamento riunioni esterne | *Responsabile*$_{\mathrm{G}}$, *Amministratore*$_{\mathrm{G}}$ | Azienda proponente, Docenti | Candidatura, $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |
| Verbali interni | Tracciamento riunioni interne | *Responsabile*$_{\mathrm{G}}$, *Amministratore*$_{\mathrm{G}}$ | Docenti | Candidatura, $RTB_{\mathrm{G}}$, $PB_{\mathrm{G}}$ |

Tabella 1: Documenti del ciclo di vita del prodotto *software*$_{\mathrm{G}}$.

### 4.1.3 Progettazione e sviluppo

In questa sezione vengono presentati gli standard e le regole (nello specifico di stile) a cui i membri di PEBKAC si devono attenere per la stesura dei documenti relativi al progetto.

#### 4.1.3.1 Template

Per la stesura dei documenti il gruppo ha creato un template in formato $LaTeX_{\text{G}}$. Il template fornisce una struttura e un formato predefinito per semplificare la creazione di documenti, al fine di garantire coerenza, efficienza e standardizzazione della presentazione. Il template è progettato per essere facile da usare, dovendo inserire solo con piccole modifiche per rispecchiare le specificità di ciascun tipo di documento.
In particolare nel template è definite la pagina di copertina con intestazione contenente logo informazioni del gruppo e dell'Università di Padova, titolo del documento, informazioni sul documento (uso, destinatari) e un breve abstract del contenuto, oltre che altre specifiche di stile come il titolo dell'indice in italiano e il numero di pagina come `X di Tot`, dove `X` è il numero della pagina e `Tot` è il numero totale di pagine.

##### 4.1.3.1.1 Parametri

Nel principale file $LaTex_{\text{G}}$ del template sono definiti una serie di comandi personalizzati per l'inserimento automatico delle informazioni come titolo, data, uso, destinatari e abstract.
Sono inoltre già presenti ma commentate le voci necessarie solo per i verbali (vedi §4.1.3.3 Verbali)

#### 4.1.3.2 Struttura del documento

Tutti i documenti prodotti da PEBKAC presentano la medesima struttura, alla quale ogni membro si deve attenere durante la procedura di stesura e modifica.

- **Pagina di copertina**: come nella sezione Template precedente;
- **Registro delle versioni**: questo registro è utilizzato per tenere traccia delle varie versioni per permettere di comprendere velocemente chi ha realizzato o modificato determinate sezioni della *documentazione*$_{\text{G}}$ e quando. Il registro presenta le versioni ordinate a partire dalla versione più recente;
- **Indice**: presente per facilitare la consultazione del documento, dotato di sezioni. Il suo scopo è di facilitare e agevolare l'accesso ad un determinato contenuto all'interno nel documento;
- **Contenuto**: il contenuto vero e proprio del documento.

#### 4.1.3.3 Verbali

I verbali differiscono dalla struttura precedentemente esposta in quanto ad essi prevedono delle sezioni aggiuntive ed obbligatorie:

- **Pagina di copertina**: nel caso di un verbale tra le informazioni sul documento compaiono anche i nominativi con i rispettivi ruoli dei membri che hanno lavorato alla loro produzione;
- **informazioni generali**: la prima sezione di un verbale deve sempre essere quella nominata "Informazioni generali" che prevede, sotto forma di elenco puntato, le seguenti informazioni:
  - Tipo di riunione,

  - Luogo in cui si è tenuta la riunione (anche se telematica),
  - Data in cui si è tenuta la riunione,
  - Ora di inizio della riunione,
  - Ora di fine della riunione,
  - Membri presenti ed eventuali altre persone alla riunione,
  - Membri assenti dalla riunione;

- **Todo**: l'ultima sezione di un verbale deve sempre essere quella che elenca i *task*$_{\text{G}}$ emersi durante la riunione da aggiungere al *backlog*$_{\text{G}}$. Questi vengono presentati sotto forma di tabella a due colonne:

  - **Assegnatario**: il membro a cui quel *task*$_{\text{G}}$ è stato assegnato, nel caso in cui non ve ne sia uso ma il *task*$_{\text{G}}$ possa essere autoassegnato da uno dei membri si scriverà "autoassegnazione" in corsivo;
  - **Task$_{\text{G}}$ Todo**: denominazione del *task*$_{\text{G}}$.

#### 4.1.3.4 Nomenclatura

La nomenclatura per i documenti si ottiene unendo il nome del file in *Snake_Case* quindi con le parole separate da un underscore (_) (`Nome_del_File`), un underscore (_) e la sua versione (`1.2.3`), ottenendo per esempio `Norme_di_Progetto_1.2.3.pdf`. Nel caso di documenti il cui nome contiene una data, essa si inserisce dopo il nome, ma prima della versione, sempre usando gli underscores come separatori, nella forma YYYY-MM-DD: YYYY rappresenta l'anno, MM il mese e DD il giorno, sempre scritto in due cifre.

##### 4.1.3.4.1 Verbali

Per quanto riguarda i verbali, per facilitarne l'ordinamento) il loro nome è la data in cui la riunione di è tenuta nella forma YYYY-MM-DD: YYYY rappresenta l'anno, MM il mese e DD il giorno, sempre scritto in due cifre. Nel caso si tratti di un verbale esterno viene aggiunta una `E`, sempre separata da underscores tra la data e la versione.

#### 4.1.3.5 Versionamento

La versione di un documento è del tipo [**x**].[**y**].[**z**]:

- **z**: è un numero intero che incrementato dal Redattore ad ogni modifica;
- **y**: è un numero intero incrementato dal *Verificatore*$_{\text{G}}$ ad ogni *verifica*$_{\text{G}}$;
- **x**: è un numero intero che viene incrementato dal *Responsabile*$_{\text{G}}$ dopo la sua approvazione (versione di produzione).

#### 4.1.3.6 Convenzioni stilistiche

- **Date**: tutte le date nella *documentazione*$_{\text{G}}$ prevedono il seguente formato YYYY-MM-DD, dove DD indica il giorno a due cifre, MM il mese a due cifre e YYYY l'anno a 4 cifre;
- **Elenchi**: elenchi puntati o numerati, ogni punto inizia con la lettera maiuscola e termina con ";" ad eccezione dell'ultimo che termina con ".";

- **Menzioni**: ogni menzione ad una persona, interna o esterna, avviene nel formato Nome Cognome;
- **Riferimenti interni**: i riferimenti a sezioni interne allo stesso documento devono essere riportati seguendo la notazione `§1.2 Nome sezione`, dove `§1.2` è il numero della sezione. Inoltre questi riferimenti devono essere opportunamente collegati tramite link al paragrafo indicato, senza alterare lo stile del testo;
- **Riferimenti esterni**: i riferimenti a sezioni di documenti esterni devono essere riportati seguendo la notazione `Nome Documento (versione di riferimento), Nome sezione`;
- **Link URL**: possono essere estesi o avere una visualizzazione abbreviata, ma sempre visualizzati di colore blu;
- **Caratteri maiuscoli**: devono essere utilizzati per
  - Le iniziali dei nomi;
  - Le lettere che compongono degli acronimi e le iniziali delle rispettive definizioni;
  - Le iniziali dei ruoli svolti dai componenti del gruppo;
  - Le iniziali dei ruoli definiti all'interno del progetto didattico;
  - La prima lettera di ogni elenco puntato.
- **Grassetto**: devono essere visualizzati in grassetto
  - I titoli di sezioni/sottosezioni/paragrafi di un documento;
  - Le parole che meritano enfasi;
  - Le definizioni negli elenchi puntati.
- **Caption**: ogni immagine o tabella deve avere una caption, utile a fornire una breve descrizione o spiegazione del contenuto visivo.

#### 4.1.4 Ciclo di vita dei documenti

Ogni documento segue le fasi del seguente *workflow*$_{\text{G}}$:

1. **Assegnazione**: il gruppo assegna un documento a uno o più redattori, affiancati da uno o più verificatori;
2. **Branch$_{\text{G}}$**: si crea un *branch*$_{\text{G}}$ per lo sviluppo del documento nell'apposita *repository*$_{\text{G}}$ Docs;
3. **Template**: si copia il Template all'interno della cartella appropriata;
4. **Stesura**: si redige il documento o una sua sezione. Qualora serva un elevato parallelismo di lavoro è possibile usare Google Drive per la prima stesura e successivamente caricare il documento all'interno del *branch*$_{\text{G}}$;
5. **Commit$_{\text{G}}$**: si esegue la *commit*$_{\text{G}}$ sul *branch*$_{\text{G}}$ creato;

6. **Pull Request$_{\mathbf{G}}$**: si apre una *pull request*$_{\mathrm{G}}$ dal *branch*$_{\mathrm{G}}$ appena creato verso il *branch*$_{\mathrm{G}}$ develop: se il documento non è pronto per la *verifica*$_{\mathrm{G}}$, ma ha bisogno di ulteriori modifiche, si apre la *pull request*$_{\mathrm{G}}$ in modalità draft, per marcarla successivamente come "Ready to Review", altrimenti in modalità normale;

7. **Verifica$_{\mathbf{G}}$**: se il *verificatore*$_{\mathrm{G}}$ richiede modifiche si ripete, in ordine, dal punto 3 al punto 5;

8. **Chiusura branch$_{\mathbf{G}}$**: si elimina, quando la *pull request*$_{\mathrm{G}}$ viene chiusa o risolta, il *branch*$_{\mathrm{G}}$ creato.

Per la versione finale di un documento spetta al *Responsabile*$_{\mathrm{G}}$ conferire l'approvazione definitiva, annotando opportunamente nel registro delle versioni la versione `x.0.0` e la sua approvazione finale.

### 4.1.5 Strumenti

- *LaTex*$_{\mathrm{G}}$
- Visual Studio Code
- *GitHub*$_{\mathrm{G}}$

## 4.2 Configuration Management

### 4.2.1 Scopo

In questa sezione vengono presentate le attività svolte da PEBKAC per il processo di *Configuration Management*$_{\mathrm{G}}$. Il processo in questione consiste nell'applicazione di procedure amministrative e tecniche per l'intero ciclo di vita del *software*$_{\mathrm{G}}$, al fine di:

- Identificare, definire e stabilire una base per gli elementi *software*$_{\mathrm{G}}$ di un *sistema*$_{\mathrm{G}}$;
- Controllare le modifiche e le release degli elementi;
- Registrare lo stato degli elementi e delle richieste di modifica;
- Garantire la completezza, la coerenza e la correttezza degli elementi.

### 4.2.2 Configuration control

#### 4.2.2.1 Descrizione

Il processo di configuration control è finalizzato a garantire il controllo e la coerenza delle configurazioni del *sistema*$_{\mathrm{G}}$, assicurando che tutte le modifiche apportate a *software*$_{\mathrm{G}}$, artefatti e documenti siano tracciate, gestite e allineate agli obiettivi e ai *requisito*$_{\mathrm{G}}$ del progetto.

#### 4.2.2.2 Scopo

Il configuration control mira al raggiungimento dei seguenti punti:

- **Gestire le modifiche**: assicurare un controllo e una gestione corretti e sistematici per qualsiasi modifica nel progetto o nel *sistema*$_{\text{G}}$;
- **Documentare le richieste**: registrare tutte le richieste di modifica per mantenere una cronologia accurata e completa;
- **Valutare l'impatto**: analizzare tutte le conseguenze tecniche, economiche e operative di ogni modifica proposta;
- **Decidere sull'approvazione**: stabilire criteri chiari e inequivocabili per l'approvazione o il rifiuto delle modifiche con gli *stakeholder*$_{\text{G}}$ rilevanti;
- **Assicurare la tracciabilità**: creare *audit trail*$_{\text{G}}$ dettagliati per tracciare le modifiche e garantire la conformità alle politiche del progetto;
- **Evitare conflitti**: prevenire modifiche non autorizzate o che possano entrare in conflitto con il *sistema*$_{\text{G}}$;
- **Mantenere la qualità**: garantire che le modifiche non compromettano l'integrità, la funzionalità o gli obiettivi generali del progetto.

#### 4.2.2.3 ITS$_{\text{G}}$

Per conseguire l'obiettivo di assicurare la tracciabilità delle modifiche è necessario creare degli *audit trail*$_{\text{G}}$ dettagliati, ovvero dei registri che tracciano tutte le attività e le modifiche all'interno di un *sistema*$_{\text{G}}$. Per la creazione, la gestione ed il tracciamento di questi *audit trail*$_{\text{G}}$, PEBKAC utilizza l'Issue Tracking System *Jira*$_{\text{G}}$, sviluppato da *Atlassian*$_{\text{G}}$.

##### 4.2.2.3.1 Ticket

Un *ticket*$_{\text{G}}$ è una voce che rappresenta una singola attività, problema, richiesta o *task*$_{\text{G}}$ all'interno di un progetto.
Esistono varie tipologie di ticket:

- **Task$_{\text{G}}$**: questa tipologia di ticket rappresenta una comune attività che deve essere completata all'interno del progetto;
- **Sub-task**: questa tipologia di ticket rappresenta una parte di un ticket più grande (come un *task*$_{\text{G}}$) che viene suddivisa in azioni più piccole e gestibili;
- **Story**: questa tipologia di ticket rappresenta un *requisito*$_{\text{G}}$ ed è generalmente scritto in un formato che descrive il risultato atteso dal punto di vista dell'utente;
- ***bug*$_{\text{G}}$**: questa tipologia di ticket rappresenta un errore o difetto nel *sistema*$_{\text{G}}$ che necessita di correzione;

Ogni ticket è dotato di campi per riportare i dettagli relativi all'attività, al problema, alla richiesta o alla *task*$_{\text{G}}$ che rappresenta:

- **Summary**: un riassunto breve in una sola riga del ticket;

- **Key**: un identificatore unico per ogni ticket, nella forma si SW-Key;
- **Epic**: *epic*$_{\text{G}}$ a cui il ticket è associato;
- **Links**: un elenco di link a ticket correlati;
- **Assignee**: la persona o le persone a cui il ticket è attualmente assegnato;
- **Description**: una descrizione dettagliata del ticket;
- **Due**: la data entro cui questo ticket è programmato per essere completato;
- **Reporter**: la persona che ha inserito il ticket nel *sistema*$_{\text{G}}$;
- **Links**: un elenco di link alle *commit*$_{\text{G}}$ e alle *pull request*$_{\text{G}}$ effettuati nella *repository*$_{\text{G}}$ di *GitHub*$_{\text{G}}$ correlate al ticket;
- **Status**: la fase in cui si trova attualmente il ticket nel suo ciclo di vita, che può essere "To Do", "In Process", "Verify" ed infine "Approve & Release";
- **Sprint**: *sprint*$_{\text{G}}$ a cui il ticket è associato;
- **Fix Version**: la versione del progetto in cui il ticket è stato (o sarà) risolto;
- **Priority**: l'importanza del ticket rispetto ad altri ticket.

#### 4.2.2.3.2 Epic

Un'*epic*$_{\text{G}}$ è una raccolta di *ticket*$_{\text{G}}$ che rappresenta uno degli obiettivi più ampi e significativi verso cui è diretto l'intero progetto. Si tratta di un concetto che aiuta a gestire e strutturare il lavoro più complesso, suddividendolo in parti più piccole e gestibili. Le *epic*$_{\text{G}}$ sono utili per monitorare i progressi rispetto a funzionalità che richiedono tempo o che coinvolgono diverse aree del progetto. Le *epic*$_{\text{G}}$ sono particolarmente utili nei processi *Agile*$_{\text{G}}$, poiché offrono una visione a lungo termine del progetto, anche mentre si adatta e si pianifica in modo incrementale, fornendo strumenti di tracciamento, come una scorebord che presenta le percentuali di ticket presenti in ogni stato, che monitorano lo stato generale di ogni *epic*$_{\text{G}}$, misurano i progressi e identificano eventuali ritardi. Inoltre, un'*epic*$_{\text{G}}$, oltre ai ticket che comprende, possiede tutti i campi precedentemente elecati per i ticket.

#### 4.2.2.3.3 Versioni

Le versioni sono la modalità di organizzazione, pianificazione e monitoraggio del lavoro in base alle specifiche milestone di un progetto. Ogni versione ha a che fare con le funzionalità, rappresentate da *epic*$_{\text{G}}$ e relativi *ticket*$_{\text{G}}$ ad essa associati, da realizzare entro una scadenza. In pratica, permette di sapere chiaramente quali *requisito*$_{\text{G}}$ devono essere soddisfatti per la specifica milestone che rappresenta. Ciò rende semplice la tracciabilità dei progressi di ciascuna versione, oltre a ritardi o modifiche, usando una *scoreboard*$_{\text{G}}$ che adotta la stessa logica di quella utilizzata dalle *epic*$_{\text{G}}$. In seguito al completamento di tutti i ticket associati ad una determina versione, questa può essere rilasciata. Inoltre, una versione, oltre ai ticket che comprende, possiede dei campi che specificano la data di inizio, la data di fine ed una breve descrizione.

#### 4.2.2.3.4 Backlog e Sprint

In *Jira*$_{\text{G}}$ sono integrati diversi strumenti per lo sviluppo secondo il metodo *Agile*$_{\text{G}}$, tra i quali è importante evidenziare *backlog*$_{\text{G}}$ e *sprint*$_{\text{G}}$.
Il *backlog*$_{\text{G}}$ contiene una lista di *ticket*$_{\text{G}}$ da completare dal team, ed è ordinata in base alla priorità: i più importanti sono posti in cima, mentre i meno importanti sono disposti verso il fondo. La lista non è statica, ma è uno spazio dinamico all'interno del quale il team può aggiungere, eliminare, aggiornare e riorganizzare le priorità dei ticket al suo interno. Il *backlog*$_{\text{G}}$ è inteso come punto di partenza per pianificare il lavoro: prima di ogni *sprint*$_{\text{G}}$, il team esamina il *backlog*$_{\text{G}}$ per selezionare il lavoro da svolgere durante quello *sprint*$_{\text{G}}$.
Uno *sprint*$_{\text{G}}$ è un periodo di tempo predefinito in cui vengono completati i ticket selezionati dal *backlog*$_{\text{G}}$ prima del suo inizio. Ogni *sprint*$_{\text{G}}$ è dotato di una data di inizio, una data di fine e di uno stato, che può essere "In Corso" o "Terminato".

#### 4.2.2.3.5 Timeline

La *timeline*$_{\text{G}}$ messa a diposizione in *Jira*$_{\text{G}}$ è uno strumento realizzato tramite un *diagramma di Gantt*$_{\text{G}}$ che aiuta a gestire le scadenze, le dipendenze e l'andamento del progetto, fornendo una panoramica d'insieme dello stato di avanzamento.
Essa mostra tutti i *ticket*$_{\text{G}}$ associati ad una *epic*$_{\text{G}}$ che sono stati inseriti al suo interno, evidenziandone le date di inizio e fine e le dipendenze con altri ticket. Ogni ticket viene visualizzato come un blocco che si estende lungo la *timeline*$_{\text{G}}$ in base alla durata prevista. Le dipendenze tra i vari ticket possono essere visualizzate tramite linee di collegamento, mostrando come il completamento di un'attività dipenda da un'altra. Inoltre, la *timeline*$_{\text{G}}$ mostra chiamente anche lo stato delle attività, ovvero quali attività sono in corso, quali attività sono state completate e quali attività sono in ritardo.
Un'altra informazione mostrata nella *timeline*$_{\text{G}}$ sono le versioni e, in particolare, quando sono state fissate le loro date di scadenza. Le versioni sono rappresentate graficamente come delle linee verticali posizionate proprio sulla data di scadenza corrispondente.
É inoltre possibile visualizzare gli *sprint*$_{\text{G}}$ definiti all'interno di *Jira*$_{\text{G}}$ nell'area superiore della *timeline*$_{\text{G}}$, permettendo così di determinare quali attività sono state svolte durante ciascuno *sprint*$_{\text{G}}$.
Infine, è utile notare che nella *timeline*$_{\text{G}}$ e possibile effettuare delle operazioni di filtraggio dei ticket visualizzati, permettendo così di visualizzare anche l'organizzazione di specifici gruppi di attività.

### 4.2.2.4 Pull Request

Le *pull request*$_{\text{G}}$ sono un meccanismo per la gestione controllata delle modifiche nei rilasci del prodotto. Quando un membro del team propone modifiche al *repository*$_{\text{G}}$, esse vengono raggruppate in una *pull request*$_{\text{G}}$, che consente ai verificatori di analizzarle prima di integrarle nel *repository*$_{\text{G}}$.
Il processo può essere riassunto nei seguenti passaggi:

- **Creazione della *pull request*$_{\text{G}}$**: Non appena le modifiche vengono apportate in un *branch*$_{\text{G}}$ dedicato, il contributore apre una *pull request*$_{\text{G}}$ per proporre la loro integrazione nel *branch*$_{\text{G}}$ develop o in un altro *branch*$_{\text{G}}$ di destinazione;

- **Revisione**: I verificatori esaminano le modifiche proposte direttamente all'interno della *pull request*$_{\text{G}}$, visualizzando i dettagli di ciò che è stato modificato e avendo la

possibilità di commentare su una singola riga o su più righe/sezioni. Se le modifiche richieste sono minime, il *Verificatore*$_{\text{G}}$ può correggerle direttamente eseguendo una *commit*$_{\text{G}}$ sul *branch*$_{\text{G}}$ in cui sono state suggerite le modifiche;

- **Feedback e suggerimenti**: Se le modifiche sono significative, i verificatori solitamente forniscono suggerimenti per miglioramenti, evidenziano possibili errori ed elencano tutte le modifiche necessarie per allineare il lavoro agli standard e alle linee guida del progetto;
- **Applicazione delle correzioni**: Il contributore che ha creato la *pull request*$_{\text{G}}$ risponde al *feedback*$_{\text{G}}$ ricevuto ed apporta le modifiche necessarie. Successivamente, la *pull request*$_{\text{G}}$ viene aggiornata con le modifiche revisionate;
- **Decisione finale**: Quando i verificatori approvano la proposta, o quando tutte le questioni sollevate da questi ultimi sono state risolte, la *pull request*$_{\text{G}}$ può essere considerata accettata e viene "unita" al *branch*$_{\text{G}}$ di destinazione, garantendo che solo contributi di alta qualità facciano parte del progetto.

### 4.2.3 Configuration status accounting

#### 4.2.3.1 Scopo

Il *configuration status accounting*$_{\text{G}}$ è il processo di *documentazione*$_{\text{G}}$ e monitoraggio di verifiche e cambiamenti alle caratteristiche del prodotto *software*$_{\text{G}}$. Grazie ad esso si ha una visione complessiva della sua evoluzione e della sua aderenza ai *requisito*$_{\text{G}}$ e agli standard.

#### 4.2.3.2 Version control

Il *sistema*$_{\text{G}}$ di version control adottato dal gruppo per tracciare l'evoluzione del *software*$_{\text{G}}$ segue la seguente convenzione di numerazione delle versioni: [x].[y].[z]([build]). Si tratta di un *sistema*$_{\text{G}}$ di versionamento a quattro componenti, ognuna delle quali ha un significato specifico:

- **x**: Rappresenta la versione principale del *software*$_{\text{G}}$, e il suo valore viene incrementato per ogni fase significativa di revisione o avanzamento del progetto;
- **y**: Rappresenta cambiamenti significativi o aggiunte di nuove funzionalità, e il suo valore viene incrementato ogni volta che vengono apportati cambiamenti considerati rilevanti per il prodotto;
- **z**: Rappresenta piccole modifiche o correzioni di *bug*$_{\text{G}}$, e viene incrementato quando vengono svolte azioni come l'aggiornamento della *documentazione*$_{\text{G}}$ o la correzione di errori minori;
- **build**: Indica il numero delle build eseguite per una determinata versione, e viene incrementato ogni volta che vengono apportate delle modifiche alla *documentazione*$_{\text{G}}$.

Il *sistema*$_{\text{G}}$ prevede che la numerazione inizi dalla versione "0.0.1(0)". Ogni volta che il valore di x, y o z aumenta, tutti i valori alla sua destra vengono resettati a "0".

### 4.2.4 Configuration evaluation

#### 4.2.4.1 Scopo

Il processo di configuration evaluation serve a garantire la correttezza, la coerenza e la conformità della configurazione del *sistema*$_{\text{G}}$, del *software*$_{\text{G}}$ e dell'infrastruttura rispetto ai *requisito*$_{\text{G}}$ definiti.

#### 4.2.4.2 Tracciamento dei requisiti

Il team PEBKAC ha deciso di tracciare i *requisiti*$_{\text{G}}$ direttamente nel codice del prodotto *software*$_{\text{G}}$. Questo approccio offre un collegamento diretto e verificabile tra i *requisiti*$_{\text{G}}$ di progettazione e il codice che soddisfa tali *requisiti*$_{\text{G}}$. Il tracciamento viene effettuato includendo un commento specifico prima di ogni blocco di codice che implementa un determinato *requisito*$_{\text{G}}$. Il commento contiene l'ID univoco del *requisito*$_{\text{G}}$, in modo da facilitare l'associazione tra i *requisiti*$_{\text{G}}$ e le corrispondenti implementazioni nel codice.

### 4.2.5 Release Management

#### 4.2.5.1 Scopo

Il processo di *release management*$_{\text{G}}$ serve a pianificare, coordinare e gestire il rilascio, per far sì che qualsiasi nuova versione di un prodotto venga distribuita dal *sistema*$_{\text{G}}$ in modo controllato.

#### 4.2.5.2 Automazione compilazione documenti

Il gruppo si è dotato di una *GitHub Action*$_{\text{G}}$ che provvede all'*automazione*$_{\text{G}}$ della generazione e trasferimento di documenti *LaTeX*$_{\text{G}}$ in PDF.
Essa si attiva quando viene unita una *pull request*$_{\text{G}}$ nel *branch*$_{\text{G}}$ develop e funziona nel seguente modo:

1. Prepara l'ambiente ed installa gli strumenti necessari;
2. Esplora la directory contenente i documenti *LaTeX*$_{\text{G}}$;
3. Converte eventuali immagini in PDF e compila i file *LaTeX*$_{\text{G}}$;
4. Carica i PDF generati in una cartella organizzata;
5. Trasferisce la cartella sul *branch*$_{\text{G}}$ main;
6. Sposta i PDF all'interno della cartella corretta nel *branch*$_{\text{G}}$ main mediante un *commit*$_{\text{G}}$ automatico.

Il componente del gruppo che si è occupato della redazione di un documento dovrà semplicemente occuparsi di aprire una *pull request*$_{\text{G}}$ verso il *branch*$_{\text{G}}$ develop. Essa verrà verificata dal membro del gruppo che ricopre attualmente il ruolo di *Verificatore*$_{\text{G}}$ e, in seguito, verrà approvata dal membro del gruppo che ricopre attualmente il ruolo di *responsabile*$_{\text{G}}$, che, infine, chiuderà la *pull request*$_{\text{G}}$, eseguendo l'*automazione*$_{\text{G}}$.
Questa *automazione*$_{\text{G}}$ permette di eliminare il lavoro manuale ripetitivo e permette al team di concentrarsi sul contenuto invece che sulla gestione tecnica dei file.

# 5 Processi Organizzativi

## 5.1 Gestione organizzativa

### 5.1.1 Scopo

Lo scopo di questo processo è esporre le modalità e gli strumenti di coordinamento usati dal gruppo per la comunicazione, interna ed esterna, e normare l'assegnazione di *ruoli*$_{\text{G}}$ e compiti, oltre che la gestione dei rischi.

### 5.1.2 Ruoli

Per ottimizzare la gestione delle attività e dei compiti da svolgere vengono definiti sei *ruoli*$_{\text{G}}$ distinti, ciascuno con mansioni e responsabilità specifiche. Ogni componente del gruppo dovrà assumere ciascun *ruolo*$_{\text{G}}$ per un numero di ore significativo.

#### 5.1.2.1 Responsabile

Il *responsabile*$_{\text{G}}$ è il punto di riferimento per tutto il gruppo e anche per le comunicazioni con il *committente*$_{\text{G}}$ e con l'azienda proponente. Inoltre il *responsabile*$_{\text{G}}$ è la figura che ha il compito di coordinare le azioni dei membri del gruppo, perciò deve avere competenze tecniche in ogni ambito del progetto. Le responsabilità di questo *ruolo*$_{\text{G}}$ sono:

- Coordinamento tra gruppo ed enti esterni;
- Gestione delle comunicazioni interne;
- Pianificazione di progetto,
- Gestione dei *task*$_{\text{G}}$ e delle risorse;
- Gestione dell'avanzamento del progetto.

#### 5.1.2.2 Amministratore

L'*amministratore*$_{\text{G}}$ è la figura che definisce, gestisce e mantiene l'ambiente e l'infrastruttura necessari per lo sviluppo del progetto facendo in modo che siano affidabili e sicuri. Si occupa della gestione della configurazione, del versionamento, delle varie *automazioni*$_{\text{G}}$ e della *documentazione*$_{\text{G}}$. Si occupa di:

- Selezionare e abilitare risorse informatiche a supporto del *way of working*$_{\text{G}}$;
- Gestire errori e malfunzionamenti nei meccanismi nell'infrastruttura.

#### 5.1.2.3 Analista

La funzione dell'*analista*$_{\text{G}}$ è quella di analizzare il problema per definire i *requisiti*$_{\text{G}}$ del prodotto, per questo deve avere buona conoscenza del dominio del problema. L'*analista*$_{\text{G}}$ raccoglie le sue produzioni nel documento Analisi dei Requisiti. Si tratta di un *ruolo*$_{\text{G}}$ fondamentale all'inizio del progetto, ma la cui utilità cala nelle seguenti fasi del progetto.

#### 5.1.2.4 Progettista

Al progettista spettano le scelte realizzative e le specifiche architetturali del prodotto. Deve avere buone competenze tecniche e tecnologiche. Durante il processo di sviluppo la sua utilità è massima, ma tende a calare dalla fase di manutenzione in poi.

#### 5.1.2.5 Programmatore

Quello del programmatore è un *ruolo*$_{\text{G}}$ chiave nella fase di sviluppo. In particolare si occupa di :

- Codificare ciò che è stato definito dai progettisti;
- Implementare i test;
- Redigere il Manuale utente.

#### 5.1.2.6 Verificatore

Ha il compito di verificare il lavoro degli altri e per questo deve avere competenze tecniche ed essere presente per l'intera durata del progetto. Questa figura deve controllare che tutto ciò che viene prodotto sia conforme alle norme e alle aspettative di qualità del gruppo.

### 5.1.3 Attività

Ogni membro del gruppo può proporre attività da svolgere, ma è compito del *responsabile*$_{\text{G}}$ stabilire la fattibilità rispetto alle risorse usufruibili.

#### 5.1.3.1 Pianificazione

Le attività definite devono poi essere pianificate in termini di tempo e risorse dal *responsabile*$_{\text{G}}$, stabilendo quindi:

- La tempistica prevista per il completamento dell'attività;
- Il membro che dovrà eseguire l'attività, in base a *ruolo*$_{\text{G}}$ e risorse disponibili;
- Il *verificatore*$_{\text{G}}$;
- Il rischio associato.

##### 5.1.3.1.1 Strumenti

- Trello

#### 5.1.3.2 Esecuzione

L'esecuzione delle attività avviene obbligatoriamente per mano dell'assegnatario, definito dal *responsabile*$_{\text{G}}$. L'esecuzione deve essere obbligatoriamente conforme alla *documentazione*$_{\text{G}}$ associata precedentemente redatta. L'esecutore dovrà proporre la sua soluzione con una *pull request*$_{\text{G}}$.

#### 5.1.3.3 Revisione

La revisione dell'attività è effettuata dal *verificatore*$_{\text{G}}$ prima dell'effettivo inserimento delle modifiche nel *repository*$_{\text{G}}$ *GitHub*$_{\text{G}}$: la *pull request*$_{\text{G}}$ aperta dell'esecutore viene accettata o rifiutata, riportando le parti non valide ed eventuali accorgimenti possibili, a seconda dell'esito della *verifica*$_{\text{G}}$.

##### 5.1.3.3.1 Strumenti

- *GitHub*$_{\text{G}}$

#### 5.1.3.4 Chiusura

Solo nel caso dell'esito positivo della *verifica*$_{\text{G}}$, con l'accettazione della *pull request*$_{\text{G}}$, viene chiuso il branch di cui è stato effettuato il merge e l'attività viene segnata come completata.

#### 5.1.3.5 Tracciamento orario

Il gruppo utilizza *Google Sheets*$_{\text{G}}$ per avere un foglio di calcolo condiviso in cui tenere conto del tempo speso per svolgere le attività. Ogni membro è tenuto a registrare, alla fine di ogni sessione lavorativa, il numero di ore effettive di lavoro e il *ruolo*$_{\text{G}}$ ricoperto.

##### 5.1.3.5.1 Strumenti

- *Google Sheets*$_{\text{G}}$

### 5.1.4 Comunicazione

#### 5.1.4.1 Comunicazioni interne

##### 5.1.4.1.1 Comunicazioni sincrone

Le riunioni interne si svolgeranno sulla piattaforma *Slack*$_{\text{G}}$ oppure in presenza, dovranno in ogni caso decise e organizzate alcuni giorni prima per consentire la presenza di tutti i membri. In ogni riunione il gruppo predilige un approccio libero alla discussione, incentrato sulla crescita e allo scambio di opinioni.
La gestione delle riunioni interne viene affidata al *responsabile*$_{\text{G}}$ che, coadiuvato dall'*Amministratore*$_{\text{G}}$, ha il compito di:

1. Fissare data, ora e luogo della riunione;
2. Stabilire un ordine del giorno per le riunioni;
3. Fare da moderatore durante la discussione, per garantire a tutti l'opportunità di esprimersi;
4. Se necessario, comunicare con l'esterno in base alle decisioni prese dal gruppo.

#### 5.1.4.1.2 Comunicazioni asincrone

Per le comunicazioni asincrone il gruppo utilizzerà:

- *Slack*$_{\text{G}}$: in un area di lavoro sono stati creati:
  - **Canali**: uno principale con tutti i membri e altri, alla necessità, in cui i componenti possono organizzarsi e lavorare su attività collaborative;
  - **Canvas**: uno per ogni canale in cui sia necessario, per fissare messaggi importanti, documenti e risorse che richiedono facile accesso.
- Whatsapp: solo per comunicazioni immediate e poco formali.

#### 5.1.4.1.3 Strumenti

- *Slack*$_{\text{G}}$
- Whatsapp

### 5.1.4.2 Comunicazioni esterne

#### 5.1.4.2.1 Comunicazioni sincrone

Per quanto riguarda gli incontri con l'azienda proponente, cruciali per discutere in modo semplice e immediata di argomenti anche complessi, potranno essere in presenza (presso la sede R&D di Vimar S.p.A.) oppure da remoto sulla piattaforma *Microsoft Teams*$_{\text{G}}$. Per quanto riguarda le riunioni con l'azienda proponente, anche chiamate SAL - Stato di Avanzamento Lavori.

- Il gruppo ha concordato con l'azienda un calendario di incontri bisettimanali della durata di 60 minuti fino alla prima revisione, poi settimanali della durata di 30 minuti;
- Il gruppo si impegna a presenziare in maniera assidua agli incontri, segnalando per tempo eventuali assenze o modifiche a quanto precedentemente concordato;
- Il gruppo si impegna a redigere un verbale per ogni incontro per documentarne il contenuto e farlo approvare all'azienda.

#### 5.1.4.2.2 Comunicazioni asincrone

Per le comunicazioni asincrone con il proponente o altri soggetti esterni vengono utilizzati:

- **Microsoft Teams**$_{\mathbf{G}}$: per domande corte o piccoli chiarimenti si potrà usare la chat condivisa creata dall'azienda;
- **Posta elettronica**: per comunicare con soggetti esterni e con l'azienda proponente per domande articolate o modifiche agli appuntamenti fissati si utilizzerà la mail del gruppo: pebkacswe@gmail.com.

#### 5.1.4.2.3 Strumenti

- *Microsoft Teams*$_{\text{G}}$
- Google Gmail

#### 5.1.4.3 Norme comportamentali

I membri del gruppo, per garantire il rispetto delle norme e degli altri membri del gruppo, sono obbligati a:

- Essere sempre puntuali o almeno comunicare tempestivamente al *responsabile*$_{\mathrm{G}}$ eventuali problemi;
- Partecipare attivamente alla discussione;
- Mantenere un atteggiamento rispettoso, disciplinato, aperto alle discussione e disponibile.

Inoltre, per le riunioni SAL con l'azienda proponente i membri hanno concordato di rispettare le seguenti regole:

- Tenere i telefoni spenti o silenziati (a meno di particolati motivi);
- Non utilizzare PC o Tablet, fatta eccezione per il *responsabile*$_{\mathrm{G}}$ o chi deve raccogliere degli appunti.

# 6 Metriche di qualità

## 6.1 Metriche di qualità del processo

### 6.1.1 Fornitura

- **CV (Cost Variance):** Misura la deviazione dei costi rispetto al $budget_{\mathrm{G}}$, se il costo è negativo significa che si è sforato il limite del $budget_{\mathrm{G}}$($SPI_{\mathrm{G}} > 1$: in anticipo rispetto ai tempi pianificati, $SPI_{\mathrm{G}} < 1$: in ritardo rispetto ai tempi pianificati).
  CV = $EV_{\mathrm{G}}$ - AC.

- **PV (Planed Value):** Il valore pianificato, ovvero il costo stimato del lavoro previsto in un determinato momento del progetto.

- $\boldsymbol{EV_{\mathrm{G}}}$ **(Earned Value):** Rappresenta il valore guadagnato, che rappresenta il costo stimato del lavoro effettivamente completato in quel momento.

- **AC (Actual Cost):** Rappresenta il costo effettivo, cioè quanto è stato realmente speso fino a quel punto.

- **CPI (Cost Performance Index):** Indica se il progetto sta spendendo meno o più del previsto (CPI $> 1$: sotto $budget_{\mathrm{G}}$, CPI $< 1$: sopra $budget_{\mathrm{G}}$).
  La formula è CPI = $\frac{EV_{\mathrm{G}}}{\mathrm{AC}}$.

- **SPI (Schedule Performance Index):** Rappresenta l'efficienza temporale con cui il lavoro pianificato è stato completato rispetto a quanto programmato.
  La formula è $SPI_{\mathrm{G}} = \frac{EV_{\mathrm{G}}}{PV_{\mathrm{G}}}$.

- $\boldsymbol{BAC_{\mathrm{G}}}$ **(Budget At Completion):** Rappresenta il $budget_{\mathrm{G}}$ totale pianificato per il completamento del progetto.

- **EAC (Estimated At Completion):** Rappresenta l'aggiornamento della stima del valore per la realizzazione del progetto, ovvero il $BAC_{\mathrm{G}}$ ricalcolato in base allo stato attuale del progetto. La formula è $EAC_{\mathrm{G}} = \frac{BAC_{\mathrm{G}}}{\mathrm{CPI}}$.

- **VAC (Variance At Completion):** Rappresenta la differenza tra il $budget_{\mathrm{G}}$ previsto e quello attuale alla fine del progetto.
  La formula è VAC = $BAC_{\mathrm{G}}$ - $EAC_{\mathrm{G}}$.

- **ETC (Estimated To Completion):** Rappresenta la valutazione del costo supplementare richiesto per portare a termine il progetto.
  La formula è ETC = $EAC_{\mathrm{G}}$ - AC.

- **SV (Schedule Variance):** Indica se le attività pianificate del progetto sono in linea, anticipate o in ritardo rispetto alla programmazione.
  La formula è SV = $EV_{\mathrm{G}}$ - $PV_{\mathrm{G}}$.

- **BV (Budget Variance):** Indica se, alla data attuale, le spese sostenute sono superiori o inferiori rispetto a quanto originariamente previsto nel $budget_{\mathrm{G}}$.
  La formula è BV = $PV_{\mathrm{G}}$ - AC.

### 6.1.2 Sviluppo

- **SC (Statement Coverage):** Rappresenta la percentuale di istruzioni nel codice che vengono eseguite durante i test. La formula è SC = $\frac{\text{N Statement eseguiti}}{\text{N Statement totali}} * 100$.

### 6.1.3 Documentazione

- **IG (Indice Gulpease):** Rappresenta un indicatore per analizzare la facilità di lettura di un testo scritto in italiano. L'Indice Gulpease si basa su due variabili linguistiche principali: la lunghezza delle parole e quella delle frasi.
  La formula per determinarlo è: IG = $89 + \frac{300*\text{NF-NL}}{\text{NP}}$, dove:
  - **NF:** Indica il numero delle frasi.
  - **NL:** Indica il numero delle lettere.
  - **NP:** Indica il numero di parole.

  Questo indice fornisce un punteggio che varia da 0 a 100. I possibili punteggi possono essere:
  - **0-55:** Testo incomprensibile.
  - **56-70:** Testo molto difficile.
  - **71-80:** Testo difficile.
  - **81-95:** Testo facile.
  - **95-100:** Testo molto facile.

  Per calcolarlo viene utilizzato un *software*$_{\text{G}}$ online: https://farfalla-project.org/readability_static/

### 6.1.4 Gestione delle qualità

- **MNS (Metriche Non Soddisfatte):** Rappresenta le quantità di metriche che il progetto non riesce a soddisfare o mantenere.

## 6.2 Metriche per la qualità del prodotto

### 6.2.1 Funzionalità

- **ROS (Requisiti Obbligatori Soddisfatti):** Rappresenta la percentuale di requisiti obbligatori che sono stati soddisfatti durante la creazione del prodotto.
  La formula è ROS = $\frac{\text{requisiti obbligatori soddisfatti}}{\text{requisiti obbligatori totali}} * 100$.

- **RDS (Requisiti Desiderabili Soddisfatti):** Rappresenta la percentuale di requisiti desiderabili che sono stati soddisfatti durante la creazione del prodotto.
  La formula è RDS = $\frac{\text{requisiti desiderabili soddisfatti}}{\text{requisiti desiderabili totali}} * 100$.

- **RPS (Requisiti Opzionali Soddisfatti):** Rappresenta la percentuale di requisiti opzionali che sono stati soddisfatti durante la creazione del prodotto.
  La formula è RPS = $\frac{\text{requisiti opzionali soddisfatti}}{\text{requisiti opzionali totali}} * 100$.

### 6.2.2 Affidabilità

- **PTCP (Passed Test Cases Percentage):** Rappresenta la percentuale di casi di test completati con successo rispetto al numero totale di casi di test pianificati.
  La formula è PTCP = $\frac{\text{test superati}}{\text{test totali}} * 100$.

- **CC (Code Coverage):** Rappresenta il numero di linee di codice Verificate con esito positivo all'interno di un processo di test.
  La formula è CC = $\frac{\text{linee di codice scritte}}{\text{linee di codice totali}} * 100$.

### 6.2.3 Manutenibilità

- **SFIN (Structure Fan IN):** Rappresenta la quantità di moduli o componenti che interagiscono direttamente o dipendono da un modulo o una funzione specifica. Un valore elevato suggerisce che molte parti del $sistema_{\text{G}}$ fanno affidamento su quel particolare modulo.

- **SFOUT (Structure Fan Out):** Rappresenta la quantità di connessioni o relazioni che un componente o modulo ha con altri elementi del $sistema_{\text{G}}$. Questa misura riflette il numero di moduli che interagiscono o su cui si basa un determinato modulo. Un fan-out elevato può segnalare che un modulo è fortemente dipendente da molti altri.

### 6.2.4 Efficienza

- **TDE (Tempo di Elaborazione):** Rappresenta il tempo di risposta dal momento in cui vengono inseriti dati all'interno del prodotto $software_{\text{G}}$ al momento in cui vengono visualizzati dall'utente in questo caso l'installatore.