TECH MINDS

Norme di progetto

techminds.unipd@gmail.com

Sommario

Documento che descrive le norme di progetto del gruppo Tech Minds.

Changelog

Versione	Data	Descrizione	Autore	Verificatore
1.0.0	12/02/2025	Fix grafici termini glossario e UML	Tutino Giuseppe	Squarzoni Matteo
0.11.0	11/02/2025	Descritto il sito web, diagramma di Gantt	Tutino Giuseppe	Squarzoni Matteo
0.10.0	11/02/2025	Gestione link e fix issue	Vallotto Caterina	Lazzarin Tommaso
0.9.1	10/02/2025	Piccoli fix	Corradin Samuele	Lazzarin Tommaso
0.9.0	09/02/2025	Documentato l'aggiornamento del cruscotto	Tutino Giuseppe	Squarzoni Matteo
0.8.0	05/02/2025	Redazione sezioni revisioni congiunte, risoluzione di problemi, PDCA e vari ampliamenti	Vallotto Caterina	Lazzarin Tommaso
0.7.2	05/02/2025	Fix indice	Vallotto Caterina	Squarzoni Matteo
0.7.1	04/02/2025	Modifica identificativo delle metriche	Vallotto Caterina	Lazzarin Tommaso
0.7.0	12/01/2025	Redazione sezione accertamento qualità	Squarzoni Matteo	Tutino Giuseppe
0.6.2	08/01/2025	Modifica sezione verifica e validazione (processi di supporto)	Corradin Samuele	Salviato Leonardo
0.6.1	06/01/2025	Modifica sezione processi di supporto	Corradin Samuele	Squarzoni Matteo
0.6.0	06/01/2025	Redazione sezione verifica e validazione (processi di supporto)	Corradin Samuele	Salviato Leonardo
0.5.0	30/12/2024	Redazione sezione processi primari di sviluppo	Corradin Samuele	Squarzoni Matteo
0.4.0	18/12/2024	Redazione sezione processi organizzativi	Vallotto Caterina	Salviato Leonardo
0.3.0	12/12/2024	Redazione sezione processi primari di fornitura	Vallotto Caterina	Corradin Samuele
0.2.1	20/11/2024	Evidenziata solo la prima occorrenza dei termini del glossario	Lazzarin Tommaso	Vallotto Caterina
0.2.0	19/11/2024	Specifica tipografica elenchi nei verbali. Specifica tipografica termini nel glossario	Lazzarin Tommaso	Bressan Alessandro
0.1.0	13/11/2024	Struttura iniziale e redazione di introduzione, documentazione e gestione di configurazione	Squarzoni Matteo	Vallotto Caterina

Indice

Lista delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento nasce con l'intenzione di documentare il *way of working* del gruppo al fine di gestire ed organizzare le attività di *progetto* nella maniera più efficiente possibile. Tutte le modifiche effettuate all'interno dei documenti saranno riportate nel registro delle modifiche (*changelog*).

Abbiamo deciso di seguire lo standard ISO/IEC 12207:1995 che definisce i *processi di ciclo di vita*, ovvero l'insieme dei *processi* che descrivono e organizzano il software dalla sua concezione iniziale fino al suo ritiro. La scelta di aderire ad uno standard è dovuta al fatto che ci fornisce delle regole comuni ottimali, le quali ci permetteranno di raggiungere l'*economicità* (combinazione di *efficienza* ed *efficacia*).

Il documento è organizzato secondo tali processi, ognuno dei quali presenta delle attività composte da procedure. Queste indicano gli strumenti, le convenzioni e gli obiettivi che ci impegniamo a seguire e adottare.

1.2 Glossario

Uno dei documenti interni prodotti dal gruppo è il così detto *glossario*, ovvero una lista di termini inerenti alle attività progettuali con la relativa definizione. Il suo scopo è quello di garantire che tutti i membri del gruppo abbiano la stessa base di conoscenza e, per sua natura, è in continuo aggiornamento. I termini che sono presenti all'interno del glossario, verranno scritti *in questo stile*.

1.3 Scopo del prodotto

Il progetto ha lo scopo di realizzare un prodotto che, utilizzando l'*intelligenza artificiale* generativa, vada ad automatizzare molte delle routine digitali che gli utenti svolgono manualmente.
Il prodotto sarà una web app che permetterà di costruire dei *workflow* i cui nodi sono dei servizi esterni (ad esempio un servizio Mail o un programma di videoscrittura) e gli *archi* indicheranno l'automazione da effettuare in linguaggio naturale. Successivamente, un *agente* prenderà il workflow, interpreterà le istruzioni fornite in linguaggio naturale e le eseguirà.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- *Capitolato* C3, tutte le slides: https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C3.pdf [visitato il: 10/02/2025];
- ISO/IEC 12207:1995: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf [visitato il: 10/02/2025];
- Regolamento progetto didattico, in particolare le slide «Avvio delle attività», «Revisioni di avanzamento» e «Obblighi documentali» : https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/PD1.pdf [visitato il: 10/02/2025];
- *Casi d'uso*, in particolare le sezioni «Specifica Use Case» e «Diagrammi dei Casi d'Uso»: https://www.math.unipd.it/~rcardin/swea/2022/Diagrammi%20Use%20Case.pdf [visitato il: 10/02/2025].

1.4.2 Riferimenti informativi

- T2 - Processi di ciclo di vita, tutte le slides: https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T02.pdf [visitato il: 10/02/2025];
- Docs as Code: https://www.writethedocs.org/guide/docs-as-code/ [visitato il: 10/02/2025];
- T5 - *Analisi dei requisiti*, in particolare le slide «Documentazione dell'analisi»: https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T05.pdf [visitato il: 10/02/2025];
- T9 - *Verifica* e *validazione*, tutte le slide: https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T09.pdf [visitato il: 10/02/2025];

- Glossario: https://techminds-unipd.github.io/docs/RTB/documenti_interni/glossario/glossario.pdf [versione 1.0.0].

2 Processi primari

Un progetto è definito tale se e solo se in esso sono attivi processi primari.

Lo standard ISO/IEC 12207:1995 definisce i processi primari come i processi relativi alle attività fondamentali per la creazione e la gestione del software.

Lo standard, all'interno dei processi primari, definisce 5 categorie:
1. Acquisizione;
2. Fornitura;
3. Sviluppo;
4. Gestione operativa;
5. Manutenzione.

I processi di acquisizione, manutenzione e gestione operativa non saranno descritti a causa della natura di questo progetto.

2.1 Fornitura

2.1.1 Scopo

Il processo primario di fornitura è focalizzato sulle attività che coinvolgono noi e il proponente durante la fornitura del prodotto software e sulle risorse necessarie per completare il progetto con successo.

Questa sezione elenca tutte le regole che dobbiamo seguire e rispettare per mantenere una relazione efficace e trasparente con il proponente e i committenti durante l'intero progetto.

Per ottenere una visione costante e aggiornata dello stato del progetto e del bilancio andremo quindi a monitorare, documentare e valutare il lavoro svolto, stimando le attività ancora da completare e confrontando il loro stato attuale con i *requisiti* definiti dal proponente.

2.1.2 Attività

Come descrive lo standard ISO/IEC 12207:1995, il processo primario di fornitura è composto dalle seguenti attività:
1. Avvio: eseguiamo una revisione dei requisiti studiando a fondo i capitolati, decidendo poi di fare una proposta per un determinato capitolato;
2. Preparazione della risposta: definiamo e prepariamo la proposta, una sorta di «risposta alla richiesta avanzata»;
3. Contrattazione: accordo sui requisiti e consegna del progetto;
4. Pianificazione: revisioniamo i requisiti di acquisizione per definire la gestione del progetto e per assicurare la *qualità* del prodotto offerto. Tale pianificazione contiene anche le esigenze di risorse e il coinvolgimento del proponente;
5. Esecuzione e controllo: implementiamo il piano di gestione del progetto, controllando i progressi e la qualità del prodotto software e della relativa documentazione durante tutte le fasi del progetto;
6. Revisione e valutazione: effettuiamo *verifiche* periodiche per dimostrare che i prodotti software e i processi soddisfino pienamente i requisiti individuati;
7. Consegna e completamento: consegniamo il prodotto software al proponente secondo quanto accordato in precedenza.

2.1.3 Rapporti con il proponente

I rapporti con il proponente sono un elemento che riteniamo fondamentale per il progetto in quanto:
- Guidano il corretto svolgimento dei processi;

- Facilitano lo scambio di feedback e dubbi, da entrambi i lati (noi nei confronti del proponente e viceversa);
- Garantiscono il rispetto di quanto pattuito.

Sono stati quindi concordati dei canali di comunicazione che permettono il raggiungimento degli obiettivi sopra descritti, per ulteriori approfondimenti vedi Sezione 4.1.3.

2.1.4 Documentazione fornita

La documentazione che forniamo al proponente e al committente è la seguente:

- **Documentazione esterna**:
  - lettera di presentazione: documento di presentazione per ogni revisione del progetto;
  - *piano di progetto*: documento che ha lo scopo di raccogliere la pianificazione delle attività progettuali, compresi la gestione dei ruoli e il bilancio di *sprint* in sprint;
  - *piano di qualifica*: documento che contiene le metriche e le normative che abbiamo individuato e adottato;
  - *analisi dei requisiti*: documento che raccoglie i requisiti del prodotto software.
- **Documentazione interna al gruppo**:
  - studio dei capitolati: documento che racchiude l'analisi approfondita di ogni proposta, con relativi vantaggi e svantaggi, tecnologie richieste e considerazioni;
  - glossario: documento utile per definire termini rilevanti nell'ambito del progetto, al fine di uniformare la conoscenza dei membri del gruppo ed evitare incomprensioni e ambiguità;
  - *norme di progetto*: il presente documento, creato per determinare il way of working che ci guida nel corso del progetto.

2.1.5 Strumenti e tecnologie

A supporto del processo di fornitura abbiamo deciso di utilizzare i seguenti strumenti e tecnologie:

- Canva per la realizzazione dei diari di bordo;
- Microsoft Teams per gli incontri esterni;
- *Slack* per le comunicazioni testuali con il proponente;
- *Typst* per la stesura della documentazione, compresi i vari diagrammi (*casi d'uso*, *diagrammi di gantt* ecc.) e le tabelle.

2.2 Sviluppo

2.2.1 Scopo

Questa sezione stabilisce le linee guida e le regole che dobbiamo seguire per il processo di sviluppo del prodotto software. L'obiettivo è quello di fornire una struttura ben definita per la realizzazione del prodotto, in modo tale da garantire un'implementazione corretta e coerente con i requisiti del proponente.

Come descrive lo standard ISO/IEC 12207:1995, il processo primario di sviluppo include le seguenti attività:

1. Analisi dei requisiti;
2. Progettazione;
3. Codifica e testing;
4. Integrazione;
5. Installazione.

2.2.2 Analisi dei requisiti

L'attività di analisi dei requisiti è fondamentale per:

- La corretta comprensione delle esigenze del proponente;

- Identificare, documentare e validare i requisiti funzionali e non funzionali;
- Evitare il rischio che il prodotto software non converga mai verso le aspettative del proponente;
- Facilitare la comprensione dei requisiti a tutti gli *stakeholder*;
- Garantire che la progettazione riceva dei requisiti chiari e semplici da comprendere.

In questa fase il nostro compito è quindi quello di raccogliere, analizzare e documentare i requisiti del prodotto (requisiti software).

In generale, l'analisi studia a fondo i bisogni, con particolare attenzione al «cosa» deve succedere nel prodotto che verrà sviluppato, ovvero alla sua struttura funzionale. I requisiti riflettono la prospettiva dell'utente nel passaggio dalla situazione senza, a quella con il prodotto. Uno dei modi più efficaci per individuare i requisiti è attraverso la scrittura dei casi d'uso.

2.2.2.1 Casi d'uso

Un caso d'uso è un insieme di scenari (sequenze di azioni) che hanno in comune uno scopo finale (obiettivo) per un utente (*attore*).

Ogni caso d'uso è composto da:
- Identificativo e nomenclatura:

UC.a.b Nome

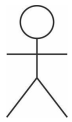
dove «a» è il numero del caso d'uso, «b» è il numero del sotto caso d'uso, se presente, e «Nome» è il nome del caso d'uso (esplicativo del suo scopo);
- Diagramma: rappresentazione grafica del caso d'uso;
- Descrizione: breve descrizione del caso d'uso;
- Attori principali: elenco degli attori principali (entità esterne che interagiscono attivamente con il sistema) coinvolti;
- Attori secondari: elenco degli attori secondari (entità esterne che interagiscono passivamente con il sistema) coinvolti, se presenti;
- Scenario principale: sequenza di azioni che descrive il comportamento dell'attore e del sistema;
- Pre-condizioni: condizioni che devono essere sempre verificate prima dello scenario descritto;
- Post-condizioni: condizioni che devono essere sempre verificate una volta concluso lo scenario descritto;
- Estensioni: elenco delle estensioni, se presenti;
- Generalizzazioni: elenco delle generalizzazioni, se presenti.

Diagrammi dei casi d'uso

I diagrammi dei casi d'uso sono strumenti utili a visualizzare in modo immediato e intuitivo la composizione del caso d'uso e le sue relazioni interne, in modo tale da avere una prima descrizione visiva di quello che successivamente verrà spiegato in versione testuale.

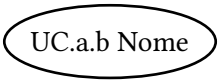
I diagrammi corrispondono a dei grafi orientati, dove i nodi rappresentano gli attori e i casi d'uso, mentre gli archi rappresentano le relazioni tra essi. Nel diagramma viene anche delineato il confine del sistema in quel determinato scenario, in modo tale da avere una visione chiara e precisa delle interazioni tra attori esterni e il sistema stesso.

I componenti principali di un diagramma dei casi d'uso sono:
- Attori: entità esterne che interagiscono con il sistema;



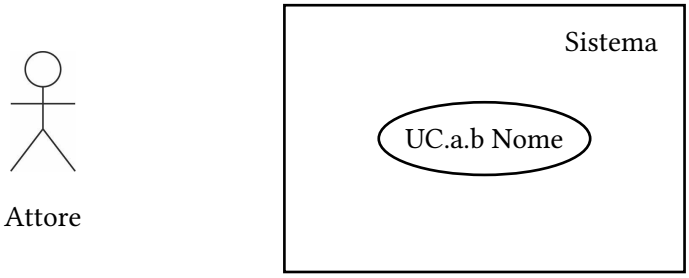
Attore

- Casi d'uso: rappresentano le funzionalità offerte dal sistema, con cui l'attore può interagire;

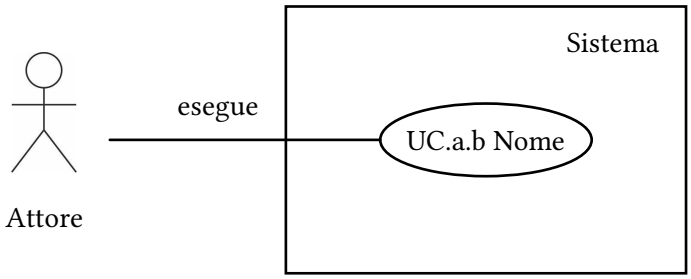


Caso d'uso

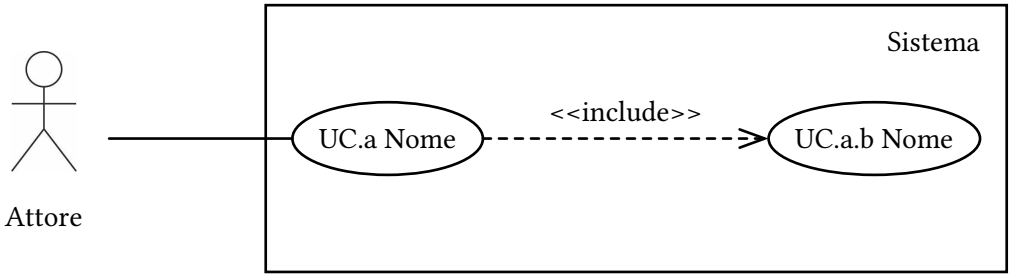
- Sistema: rappresenta il «dominio» di un particolare caso d'uso, serve a stabilire il confine tra ciò che è esterno e quindi non gestito dal sistema e ciò che invece è interno e quindi gestito dal sistema. Gli attori per definizione saranno rappresentati esternamente al sistema, mentre i casi d'uso al suo interno;



- Relazioni: rappresentano le interazioni tra attori e casi d'uso, e tra casi d'uso stessi. Sono di quattro tipi:
  - associazione: collegamento tra attore e caso d'uso, indica che l'attore è coinvolto nel caso d'uso;

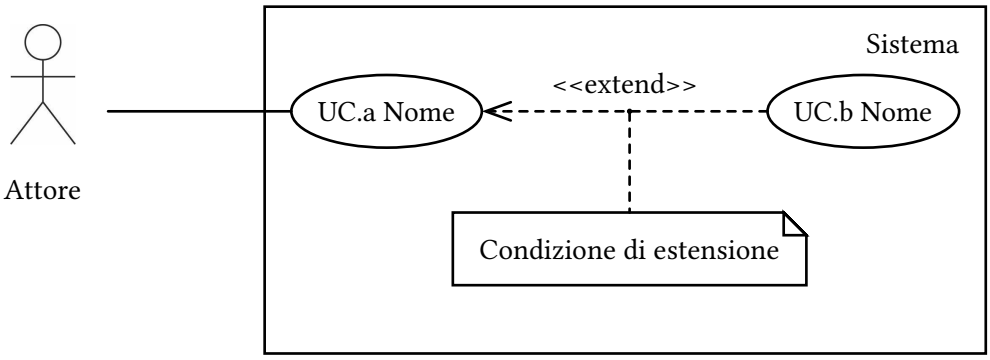


  - inclusione: relazione tra due casi d'uso, indica che un caso d'uso ne include un altro, può essere una funzionalità comune a più casi d'uso. Ogni istanza del caso d'uso «che include» esegue sempre il caso d'uso «incluso», con lo stesso attore principale. Nel sotto caso d'uso si utilizza l'identificativo del caso d'uso «che include» seguito da un punto e un numero progressivo;

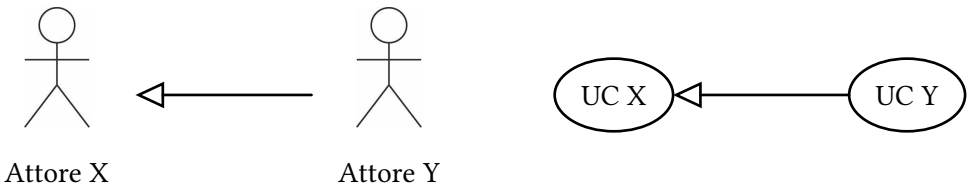


  - estensione: relazione tra due casi d'uso, indica che un caso d'uso può estendere uno o più casi d'uso, aumentandone le funzionalità. Ogni istanza del caso d'uso «base» esegue il caso d'uso che estende in modo condizionato (a differenza dell'inclusione), e questo dipende dalla condizione

di estensione, la quale determina quando l'estensione deve essere utilizzata. Con le estensioni si vogliono descrivere variazioni dalla funzionalità del caso d'uso «base», ad esempio in caso di errore/eccezione;



- generalizzazione: relazione tra due attori o due casi d'uso, indica che un attore o un caso d'uso è una generalizzazione di un altro attore o caso d'uso, quindi condivide le funzionalità del primo e ne può aggiungere di nuove o modificare quelle già esistenti (Y generalizza X);



2.2.2.2 Requisiti

Per costruire un sistema efficace, efficiente e in linea con le aspettative del proponente è necessario individuare:

- Requisiti funzionali: funzionalità e comportamenti che il software deve supportare per soddisfare le esigenze del proponente;
- Requisiti non funzionali: definiscono come il sistema deve comportarsi, cioè le qualità, i vincoli e le caratteristiche tecniche che influenzano l'esperienza d'uso e le performance del software.

Ogni requisito è costituito da:

1. Codice: un codice identificativo univoco, nel formato

   [Tipologia][Importanza]Requirement N

   dove «Tipologia» corrisponde a:

   - «F», functional, requisito funzionale, descrive una funzionalità del sistema;
   - «Q», quality, requisito di qualità, descrive una caratteristica del sistema;
   - «C», constraint, requisito di vincolo, descrive una limitazione imposta al sistema;

   «Importanza» corrisponde a:

   - «M», mandatory, requisito obbligatorio, irrinunciabile per qualcuno degli *stakeholder*;
   - «D», desirable, requisito desiderabile, non strettamente necessario ma a valore aggiunto riconoscibile;
   - «O», optional, requisito opzionale, relativamente utile e contrattabile anche quando il progetto è in uno stato avanzato;

   «N» è un numero progressivo (es. QMR2 indica un requisito di qualità obbligatorio numero 2);

2. Descrizione: una breve descrizione del requisito;
3. Fonti: le fonti da cui è stato identificato il requisito.

2.2.2.3 Strumenti e tecnologie

In linea con la nostra filosofia «Docs as Code» (si veda Sezione 3.1.2.1), tutti i diagrammi dei casi d'uso sono creati con Typst.

2.2.3 Progettazione

L'attività di progettazione è fondamentale per la corretta realizzazione del prodotto software. Questa fase viene naturalmente svolta in seguito a quella di analisi, in quanto si basa sui requisiti individuati in precedenza, per definire come fare ciò di cui c'è bisogno. L'obiettivo è quello di fornire una soluzione realizzativa che stabilisca l'architettura per la successiva attività di codifica e che soddisfi le esigenze di tutti gli stakeholder coinvolti nel progetto.

Secondo lo standard ISO/IEC 12207:1995, i criteri per riconoscere una buona progettazione sono:
- Tracciabilità rispetto ai requisiti dell'elemento software;
- Coerenza esterna con i requisiti dell'elemento software;
- Coerenza interna tra i componenti software;
- Adeguatezza dei metodi e degli standard di progettazione utilizzati;
- Fattibilità della progettazione dettagliata (se è realizzabile con le risorse disponibili);
- Fattibilità delle operazioni e della manutenzione.

A supporto dell'attività di progettazione verranno utilizzati i diagrammi delle classi, vista la loro utilità nel documentare in modo conciso l'architettura per la futura fase di implementazione.

2.2.3.1 Diagrammi delle classi

I diagrammi delle classi sono uno strumento molto utile nella fase di progettazione del software, poiché permettono di rappresentare graficamente la struttura statica del sistema, mostrando le classi che lo compongono, i loro attributi, i metodi e le relazioni tra di esse.

I vantaggi di utilizzare i diagrammi delle classi sono molteplici:
- Aiutano a identificare le responsabilità delle singole classi e a definire chiaramente i confini tra di esse;
- Consentono di individuare potenziali problemi di progettazione prima di iniziare la fase di codifica, riducendo così di molto il costo di modifiche o future correzioni;
- Forniscono una visione d'insieme del sistema, facilitando la comprensione e la comunicazione interna tra i membri del gruppo e con gli stakeholder;
- Offrono un linguaggio comune (*UML*) che permette di seguire uno standard riconosciuto a livello internazionale.

2.2.4 Codifica e testing

L'attività di codifica e testing consiste nella realizzazione effettiva del prodotto software. Durante questa fase, il codice viene scritto e testato per garantire che il prodotto soddisfi i requisiti individuati in fase di analisi e rispetti la progettazione definita al passo precedente. L'obiettivo è quello di creare il prodotto software richiesto dal committente, rispettando gli accordi stipulati in fase di fornitura.

Secondo lo standard ISO/IEC 12207:1995, i criteri per riconoscere una buona codifica/testing sono:
- Tracciabilità rispetto ai requisiti e alla progettazione dell'elemento software;
- Coerenza esterna con i requisiti e la progettazione dell'elemento software;
- Coerenza interna tra i requisiti delle unità;
- Copertura dei test delle unità;
- Adeguatezza dei metodi e degli standard di codifica utilizzati;

- Fattibilità dell'integrazione e del testing del software;
- Fattibilità delle operazioni e della manutenzione.

2.2.5 Integrazione

L'attività di integrazione consiste nell'unione delle parti di software sviluppate in precedenza, per formare un'unica entità funzionante. A ogni nuovo sviluppo corrisponde una nuova integrazione in modo tale da verificare il prima possibile che l'elemento appena integrato sia conforme alle aspettative. L'obiettivo è quello di verificare che le singole parti del software funzionino correttamente nel loro insieme.

2.2.6 Installazione

L'attività consiste nello sviluppo di un piano per installare il prodotto software nell'ambiente di destinazione. In questo piano devono essere fornite le risorse e le informazioni necessarie per l'installazione e la configurazione del software. L'obiettivo è quello di fornire istruzioni chiare e dettagliate all'utente finale, in modo tale da consentirgli di utilizzare il prodotto software.

3 Processi di supporto

I processi di supporto rappresentano un insieme di attività trasversali che aiutano a garantire la qualità e l'efficacia dei processi attivi durante lo sviluppo software. Le attività descritte all'interno di questa sezione seguono un processo di adattamento e miglioramento in base alle esigenze che sorgono e alla necessità di garantire qualità.

3.1 Documentazione

3.1.1 Caratteristiche e finalità

La documentazione è l'insieme di tutte le informazioni utili al gruppo raccolte sotto forma di testo, diagrammi o immagini.

Secondo lo standard ISO/IEC 12207:1995, il processo di documentazione è un processo finalizzato alla registrazione delle informazioni prodotte da un processo o attività del ciclo di vita del software. Questo processo comprende un insieme di attività che pianificano, progettano, sviluppano, producono, modificano, distribuiscono e mantengono i documenti necessari per tutte le parti coinvolte nel progetto.

In particolare lo standard definisce quattro attività principali:
1. Implementazione del processo: definisce un piano per identificare i documenti da produrre durante il ciclo di vita del software e stabilisce gli aspetti che ogni documento deve contenere;
2. Progettazione e sviluppo: definisce la struttura del documento e l'origine delle informazioni da includere al suo interno. Crea e sviluppa del contenuto in base agli standard definiti;
3. Produzione: produce e distribuisce il documento finale nel formato prestabilito;
4. Manutenzione: aggiorna il documento nel tempo in modo che sia sempre accurato e pertinente.

Per la natura di questo progetto, la prima e ultima attività non verranno svolte interamente.

3.1.2 Progettazione e sviluppo

3.1.2.1 Filosofia «Docs as code»

Abbiamo scelto di usare la filosofia «Docs as code» che si prescrive di trattare tutta la documentazione come se fosse del vero e proprio codice sorgente.
Questo porta i seguenti vantaggi:
1. Supporto agli *issue tracking system*;
2. Controllo di versione con *git*;
3. Processi di revisione del codice;

4. Test e *integrazione continua*.

3.1.2.2 Tecnologie utilizzate per la documentazione

Il team ha scelto di usare le seguenti tecnologie per redigere i documenti:

- *Typst*: linguaggio simile a *LaTeX* ma che permette di comporre documenti in modo più semplice e intuitivo.

3.1.2.3 Organizzazione dei documenti

- Nomi dei file: i file iniziano sempre con lettera minuscola o con una data del tipo yyyy-mm-dd[-info] nel caso di *verbali* (info è un campo opzionale che contiene informazioni aggiuntive);
- Organizzazione delle cartelle: nel caso in cui il documento sia composto da più file, questi sono contenuti in una cartella chiamata `/include`;
- Elenchi: in tutti gli elenchi gli elementi terminano con «;» tranne l'ultimo che termina con «.». Inoltre la prima lettera di ogni punto deve essere maiuscola, mentre se si tratta di un sottoelenco deve essere minuscola;
- Sezioni: viene seguita la convenzione di Typst, ovvero la sezione principale avrà un numero «X» e tutte le sottosezioni aggiungeranno ricorsivamente un altro «.X»;
- Figure: ogni figura deve essere dotata di caption che ne spieghi brevemente il contenuto;
- Tabelle: ogni tabella deve essere dotata di caption che ne spieghi il contenuto;
- Termini nel glossario: si evidenziano le parole presenti nel glossario solamente nelle loro prime occorrenze nel documento;
- Link: quando un link viene inserito come risorsa da consultare, deve essere indicata la data dell'ultima volta in cui la risorsa è stata visitata (nel formato [visitato il: dd/mm/aaaa]). Se invece un link porta ad un documento con ciclo di vita, è necessario inserirne la versione (nel formato [versione X.Y.Z]). Per i link relativi alle *issue* non vale quanto appena descritto. I link possono essere scritti:
  - per esteso: il link deve essere cliccabile e portare alla risorsa;
  - come testo cliccabile: deve essere aggiunta una footnote con il link per esteso.

3.1.2.4 Struttura dei documenti

L'impaginazione e la struttura generale dei documenti è descritta nel file `template.typ`.
Tutti i documenti sono sempre divisi in almeno queste parti:

1. Intestazione:
   - logo;
   - mail del gruppo;
   - nome del documento;
   - sommario (breve descrizione del contenuto del documento).
2. Changelog, tabella che contiene tutti i cambiamenti effettuati al documento:
   - versione del documento;
   - data della modifica nel formato gg/mm/aaaa;
   - autore della modifica;
   - verificatore.
3. Indice;
4. Lista delle figure;
5. Contenuto.

3.1.2.5 Struttura dei verbali

I verbali hanno informazioni aggiuntive oltre a quelle descritte precedentemente. Tali informazioni sono già presenti nella funzione `verbale` presente file `template.typ`. Invocando la funzione e impostando i parametri indicati è possibile ottenere un verbale già formattato.

- Nei verbali interni vengono aggiunti:

1. partecipanti;
2. data, ora e luogo;
3. ordine del giorno;
4. riassunto con le decisioni prese.

- Nei verbali esterni vengono aggiunti:
  1. tutte le informazioni dei verbali interni;
  2. lo spazio per la firma di chi non fa parte del gruppo.
- Scelte tipografiche specifiche dei verbali:
  - gli elenchi e le liste possono non terminare col punto e virgola;
  - i termini presenti del glossario presenti nei verbali verranno evidenziati dalla sezione «Contenuto della riunione» in poi.

3.1.3 Sito web

Utilizziamo un sito web per esporre pubblicamente la nostra documentazione, in modo da fornire un'interfaccia più adatta ad un pubblico non tecnico. Il sito (https://techminds-unipd.github.io/docs) è generato automaticamente con delle GitHub Action. Ogni volta che avviene un cambiamento nel branch main tutti i documenti vengono compilati e poi viene generata una pagina web che sarà messa online dalla action delle GitHub Pages.

3.2 Gestione della configurazione

3.2.1 Caratteristiche e finalità

La gestione della configurazione è un *processo* fondamentale per monitorare e controllare le modifiche ai componenti software durante il loro ciclo di vita. Il prodotto SW non è mai un monolite, infatti è composto da più componenti, ognuno dei quali è caratterizzato da un proprio ciclo di vita. Ogni singolo componente ha una storia individuale che viene gestita tramite controllo di versione, che aiuta a non perdere traccia di questa storia e a poter avanzare o retrocedere nelle versioni. Il processo di gestione della configurazione, dunque, aiuta a mantenere la coerenza e l'integrità del software, facilitando la collaborazione tra i membri del team.

Secondo lo standard ISO/IEC 12207:1995, questo processo include le seguenti attività:

1. Implementazione del processo: definisce un piano per la gestione della configurazione, che descrive le varie attività e l'organizzazione di queste attività;
2. Identificazione della configurazione: definisce uno schema per l'identificazione degli elementi relativi al progetto e delle loro versioni;
3. Controllo della configurazione: gestisce le richieste di modifica e il loro tracciamento;
4. Registrazione dello stato della configurazione: prepara i registri di gestione, i quali mostrano lo stato e la storia completa degli elementi relativi al progetto;
5. Valutazione della configurazione: valuta la completezza degli elementi relativi al progetto rispetto ai requisiti;
6. Gestione e consegna delle versioni: assicura che tutte le versioni degli elementi software e della documentazione siano correttamente gestite, conservate e distribuite in modo tracciabile.

3.2.2 Versionamento (Identificazione della configurazione)

La procedura di versionamento in un progetto software è fondamentale per gestire le modifiche e garantire la collaborazione tra sviluppatori. In ogni documento è possibile trovare, prima ancora dell'indice, un registro delle modifiche (changelog), necessario per comprenderne il ciclo di vita dal suo concepimento fino all'ultima versione disponibile. All'interno del changelog si fa riferimento alla versione del documento rappresentato da un numero di versione così composto:

vX.Y.Z

spiegazione:
- X rappresenta la versione principale (*major version*), che cambia con aggiornamenti significativi o modifiche radicali;
- Y indica la sottoversione (*minor version*), che viene aggiornata per miglioramenti o nuove funzionalità non critiche;
- Z è il numero di *patch*, utilizzato per correzioni di bug o piccole modifiche.

All'interno del codice non è presente un vero e proprio changelog, in questo caso l'intero versionamento del codice viene gestito dal software git sulla piattaforma *GitHub*.

3.2.3 Gestione Repository (Controllo della configurazione e registrazione dello stato)

La struttura del *repository* è composta da:
- **main**: è il ramo predefinito dove risiede la versione stabile del prodotto, su questo *branch* è stata impostata una regola di protezione per evitare che venga introdotto del codice non funzionante o non approvato;
- **branch di feature** creati secondo la pratica *GitHub Flow*.

Il GitHub Flow è un flusso di lavoro semplice e leggero, considerato ideale dal nostro team per lo sviluppo. Si compone di cinque fasi principali:
1. **Creare un branch di feature**: iniziare creando un nuovo branch dal main per lavorare su nuove funzionalità senza influenzare il codice principale;
2. **Modificare i file nel nuovo branch**: effettuare modifiche e aggiungere *commit* per tenere traccia dei progressi;
3. **Creare una *Pull Request***: una volta effettuati i commit e il *push*, aprire una pull request per avviare la revisione del codice;
4. **Unire la Pull Request**: dopo la revisione, unire il branch al main facendo il merge;
5. **Eliminare il branch**: infine, eliminare il branch di feature per mantenere il repository pulito.

All'interno del branch main si trova un file README.md dove è possibile visionare la struttura delle cartelle del repository. Inoltre, per il repository della documentazione è disponibile una pagina web statica ospitata da *GitHub Pages*, che permette di visionare tutti i documenti appartenenti alla documentazione generati attraverso delle *GitHub Actions*.

3.3 Accertamento della qualità

3.3.1 Caratteristiche e finalità

L'accertamento della qualità è un processo che ha lo scopo di garantire che il prodotto software soddisfi i requisiti concordati con il proponente e che sia conforme agli standard di qualità prefissati. Questo processo è fondamentale per garantire che il prodotto finale sia affidabile, sicuro e soddisfi le aspettative del cliente.

Secondo lo standard ISO/IEC 12207:1995, l'accertamento della qualità include le seguenti attività:
- Implementazione del processo: definisce un piano per l'accertamento della qualità, che descrive metodologie, procedure e strumenti da utilizzare. È necessario definire anche il coordinamento con le attività di verifica e validazione;
- Garanzia di prodotto: definisce le garanzie di qualità che il prodotto software deve soddisfare;
- Garanzia di processo: definisce le garanzie di qualità che i processi di ciclo di vita del software devono soddisfare;

3.3.2 PDCA

Il ciclo *PDCA*, noto anche come ciclo di Deming, è stato scelto come approccio per il miglioramento continuo di processi e prodotti, con l'obiettivo di attuare manutenzione migliorativa al proprio way of working. Per ottenere risultati concreti, è fondamentale seguire con attenzione le quattro fasi che lo compongono:

1. **Plan**: in questa fase si definiscono le attività necessarie per identificare quali processi avviare e in quale ordine, con l'obiettivo di raggiungere risultati specifici. Si stabiliscono obiettivi di miglioramento chiari e si progettano le azioni da intraprendere. Non riguarda pianificazione di progetto, ma bensì pianificazione di miglioramento;
2. **Do**: si passa all'azione, mettendo in pratica quanto pianificato. Durante l'esecuzione si raccolgono dati e si monitorano i risultati per valutare l'efficacia delle attività svolte. Non è sviluppo, ma dispiegamento (esplorativo) di azioni di miglioramento;
3. **Check**: qui si analizzano i dati raccolti nella fase di esecuzione, confrontandoli con gli obiettivi prefissati. Si utilizzano metriche specifiche per interpretare i risultati e individuare eventuali discrepanze o aree di miglioramento. Si verifica quindi l'esito delle azioni di miglioramento rispetto alle attese;
4. **Act**: sulla base delle valutazioni precedenti, si consolidano le pratiche che hanno prodotto risultati positivi, inserendole nel way of working. Si implementano poi azioni correttive per affrontare eventuali criticità. Si analizzano le cause dei problemi e si apportano miglioramenti, favorendo così un'evoluzione continua del processo.

Ogni ciclo PDCA rappresenta un'opportunità per raggiungere gli obiettivi di qualità fissati, creando un processo di miglioramento costante e progressivo.

3.3.3 Piano di qualifica

Il piano di qualifica è un documento che definisce le strategie e le metodologie che il team intende adottare per garantire la qualità del prodotto software. All'interno del piano di qualifica vengono descritti i processi di verifica e validazione con i relativi obiettivi di qualità. Inoltre, vengono fissati gli standard di qualità da rispettare e le metriche da utilizzare per misurare la qualità del prodotto. All'interno del piano di qualifica viene definito il cruscotto di controllo, che raccoglie un insieme di misurazioni per ogni metrica adottata, i dati rilevati da queste misurazioni poi vengono sottoposti a criteri di accettazione. Il cruscotto di controllo dunque consente di monitorare lo stato del progetto, rilevare problemi critici e prendere decisioni migliorative basate sui dati a disposizione.

3.3.4 Struttura metriche di qualità

- **Codice:** identificativo univoco della metrica;

  M[abc][numero]

  dove:
  - M: indica che si tratta di una metrica;
  - abc: indica le prime 3 lettere della categoria di appartenenza della metrica (se ritenuto necessario è possibile inserire le prime 4 lettere);
  - numero: numero progressivo della metrica.
- **Nome:** specifica il nome della metrica;
- **Descrizione:** breve descrizione della metrica;
- Relativi criteri di accettazione:
  - **Valore accettabile**: valore minimo accettabile della metrica per essere conforme agli standard di qualità stabiliti;

- **Valore ottimale:** valore ideale della metrica;

3.3.5 Aggiornamento cruscotto

Il cruscotto è interamento gestito e generato in Typst. Ogni volta che viene compilato, il documento riprende tutti i valori aggiornati degli sprint da una funzione apposita (definita in costi.typ nella directory contenente il piano di progetto). I valori ottenuti vengono usati per calcolare le varie metriche, che saranno successivamente visualizzate attraverso dei grafici, sempre generati in Typst con la libreria CeTZ. C'è da precisare che questa è una procedura semi-automatica, ad esempio alcuni dati per le metriche devono essere inseriti a mano ad ogni sprint. I dati in questione riguardano l'indice di Gulpease, rischi non previsti e l'*Earned Value*.

3.4 Verifica e validazione

3.4.1 Caratteristiche e finalità

La verifica e la validazione sono due processi fondamentali per garantire la qualità e la correttezza del prodotto finale. La verifica viene eseguita su tutte le componenti del prodotto, siano esse documentazione o codice, durante tutta la durata del progetto, in modo tale da confermare che il lavoro svolto fino a quel momento sia corretto e rispetti gli standard prefissati. La validazione, invece, agisce solo a fine progetto, poiché si occupa di confermare che il prodotto finale nel suo complesso soddisfi le aspettative e i requisiti concordati con il proponente. La validazione rispecchia quindi l'insieme di tutte le verifiche effettuate durante il progetto, e per questo motivo la verifica è ciò che permette alla validazione di essere eseguita con successo.

Per il processo di verifica lo standard ISO/IEC 12207:1995 include i seguenti compiti:
- Verifica della documentazione, considerando i criteri di completezza, correttezza e coerenza;
- Verifica del design, considerando i criteri di correttezza, coerenza e tracciabilità rispetto ai requisiti;
- Verifica del codice, considerando i criteri di correttezza, tracciabilità rispetto al design e ai requisiti, testabilità e conformità agli standard di codifica.

La verifica può essere eseguita in due forme: analisi statica e analisi dinamica.

3.4.2 Analisi statica

L'analisi statica è una forma di verifica che non richiede l'esecuzione dell'oggetto di verifica e per questo può essere applicata a ogni componente del prodotto. Viene utilizzata per accertare il rispetto di regole di scrittura e di stile, l'assenza di difetti e la presenza di proprietà desiderate. Questa forma di verifica può essere eseguita tramite due modalità principali:
- Walkthrough: tecnica di lettura in cui l'oggetto in esame viene controllato nella sua totalità, senza svolgere una ricerca di un errore/proprietà specifica al suo interno, ad esempio la lettura di un documento per rilevare errori grammaticali. Questo tipo di esame è privo di assunzioni o presupposti;
- Inspection: tecnica di lettura in cui l'oggetto in esame viene controllato per verificare la presenza di difetti o proprietà specifiche, ad esempio la ricerca di termini particolari in un documento. L'esame è basato su presupposti e per questo permette una maggiore possibilità di automazione rispetto al Walkthrough.

3.4.3 Analisi dinamica

L'analisi dinamica è una forma di verifica che richiede l'esecuzione dell'oggetto di verifica e quindi può essere applicata solo a codice eseguibile. Viene utilizzata per accertare il corretto funzionamento dell'oggetto tramite test (prove), i quali verificano che l'oggetto produca i risultati attesi e individuano eventuali anomalie durante l'esecuzione. I test possono essere di vario tipo e per questo è necessario classificarli a seconda del loro scopo e del loro ambito di applicazione.

Classificazione dei test

Le tipologie di test principali sono, in ordine di esecuzione:

1. *Test di unità*: verificano il corretto funzionamento di singole unità di codice, come funzioni, metodi o classi. Poiché verificano piccole porzioni di codice, questi test devono essere eseguiti per primi, in modo tale da evitare l'introduzione di errori una volta che queste unità vengono integrate tra loro;
2. *Test di integrazione*: verificano il corretto funzionamento delle unità di codice integrate tra loro, con l'obiettivo di verificare che le unità funzionino correttamente anche una volta integrate;
3. *Test di sistema*: verificano il corretto funzionamento del sistema nel suo complesso, con l'obiettivo di verificare che il sistema soddisfi i requisiti software concordati con il proponente e stabiliti nel documento di analisi dei requisiti;
4. *Test di regressione*: verificano che le modifiche apportate al codice non abbiano introdotto difetti in altre parti del sistema prima funzionanti. Questi test vengono eseguiti ogni volta che viene apportata una modifica al codice e non sono altro che la ripetizione selettiva di test già eseguiti in precedenza;
5. *Test di accettazione*: verificano che il prodotto finale soddisfi i requisiti utente concordati con il proponente. Sono l'ultima fase di test prima del possibile rilascio del prodotto.

3.4.4 Processo di verifica

In generale, il processo di verifica include le seguenti fasi:

1. **Apertura della pull request e assegnazione dei verificatori**: il redattore o sviluppatore apre una pull request per sottoporre il proprio lavoro a revisione, aggiungendo un titolo e una breve descrizione del lavoro svolto. Successivamente, assegna i due verificatori scelti durante la pianificazione dello sprint in corso. Se necessario, è possibile aggiungere un terzo verificatore o sostituire uno dei verificatori già assegnati;
2. **Conflitti**: se sono presenti conflitti con il ramo di destinazione, è compito del redattore risolverli prima di procedere con la verifica;
3. **GitHub Actions**: se sono presenti automazioni, queste vengono eseguite automaticamente una volta aperta la pull request. È compito del redattore verificare che tutte abbiano successo prima di procedere con la verifica. In caso di fallimento di una o più automazioni, il redattore deve correggere gli errori e ripetere l'operazione fino a quando tutte le automazioni non avranno esito positivo;
4. **Verifica**: ogni verificatore esegue un'attenta fase di verifica del lavoro svolto, seguendo norme precise a seconda del tipo di lavoro sottoposto a revisione. In caso di errori o dubbi, il verificatore può richiedere modifiche al redattore, aggiungendo commenti che citino parti specifiche del lavoro, in modo da facilitare la discussione e l'eventuale correzione;
5. **Discussione**: se necessario, il redattore e i verificatori possono discutere le modifiche richieste o i dubbi tramite la sezione apposita della pull request. Quando un dubbio viene chiarito, è compito del redattore marcare la conversazione come risolta, in modo da mantenere ordine nelle conversazioni ancora aperte;
6. **Correzione**: il redattore apporta le modifiche richieste dai verificatori, se presenti, e richiede nuovamente la verifica. In caso di necessità o di correzioni banali, anche il verificatore può modificare il lavoro, in modo da velocizzare il più possibile questa fase. Questo processo può essere ripetuto più volte, fino a quando i verificatori non approvano la pull request;
7. **Chiusura della pull request**: una volta che tutti i verificatori hanno approvato la pull request, è loro compito apportare l'ultima modifica al changelog, in modo da tracciare il lavoro svolto. Successivamente la pull request viene chiusa e il ramo di feature viene unito al ramo destinazione, concludendo così il processo di verifica.

3.4.5 Verifica della documentazione

Ogni documento relativo al progetto deve essere sottoposto a verifica. Questo compito è affidato al verificatore, che esegue un'analisi statica tramite Walkthrough.

In generale, la verifica dovrà controllare i seguenti aspetti:

- Correttezza tecnica: le informazioni contenute nel documento devono essere accurate e basate su fonti attendibili;
- Coerenza: le informazioni contenute nel documento devono essere coerenti tra loro e con gli altri documenti;
- Chiarezza: il documento deve essere facilmente comprensibile, senza ambiguità o termini tecnici non spiegati;
- Conformità agli standard: il documento deve rispettare le norme stilistiche e strutturali prestabilite;
- Correttezza ortografica e grammaticale: il documento deve essere privo di errori ortografici e grammaticali.

A supporto del verificatore, per il controllo di alcune norme strutturali, sono stati sviluppati degli script che effettuano un'analisi statica del documento mediante il metodo Inspection. Questi script vengono eseguiti automaticamente attraverso GitHub Actions, contribuendo a rendere la fase di verifica il più efficace ed efficiente possibile. Gli script in questione si occupano di controllare:
- che le parole da glossario siano ben indicate nei documenti;
- che tutti i documenti ottengano il punteggio accettabile per l'indice di Gulpease;
- che le parole presenti nel glossario siano in ordine alfabetico.

La verifica si conclude quando almeno due verificatori hanno approvato il documento, che può quindi essere esposto pubblicamente all'interno del repository.

3.4.6 Validazione

Secondo lo standard ISO/IEC 12207:1995, la validazione è il processo che serve a determinare se i requisiti e il sistema/prodotto software finale, per come è stato costruito, soddisfano l'uso specifico previsto. Il prodotto finale deve quindi soddisfare tutti i requisiti concordati con il proponente e funzionare correttamente nel suo ambiente finale.

3.5 Revisioni congiunte

Secondo lo standard ISO/IEC 12207:1995, il processo di revisione congiunta ha l'obiettivo di valutare lo stato e i prodotti di un'attività di un progetto. Tali revisioni congiunte si svolgono per tutta la durata del rapporto con il proponente. Questo processo può essere impiegato da tutte le parti coinvolte, dove una «parte revisionante» esamina una «parte revisionata».

Questo processo si compone delle seguenti attività:
- Implementazione del processo;
- Revisioni della gestione del progetto;
- Revisioni tecniche.

3.5.1 Implementazione del processo

Vengono programmate delle revisioni con il proponente quando lo svolgimento del lavoro ha raggiunto un buon grado di avanzamento rispetto all'incontro precedente. Durante tali revisioni rappresentiamo la parte revisionata, mentre il proponente rappresenta la parte revisionante. Oltre a concentrarsi sugli avanzamenti raggiunti, si discutono eventuali problematiche e le corrispettive azioni correttive necessarie.

I risultati di tali revisioni sono poi documentati tramite i verbali esterni, i quali verranno firmati dalla parte revisionante per garantire l'approvazione di quanto revisionato e discusso.

3.6 Risoluzione dei problemi

Mira ad analizzare e risolvere i problemi (incluse le non conformità), qualunque sia la loro natura o origine. Tali problemi possono essere scoperti durante l'esecuzione dello sviluppo, dell'operatività, della manutenzione o di altri processi. L'obiettivo è fornire un mezzo tempestivo, responsabile e

documentato per garantire che tutti i problemi scoperti siano analizzati e risolti, riconoscendo le cause scatenanti.

Il processo di risoluzione dei problemi ha l'obiettivo di garantire un approccio rapido, responsabile e ben documentato per analizzare e risolvere le criticità riscontrate durante l'intero ciclo di vita del prodotto. Questo processo non si limita a gestire i problemi nell'immediato, ma punta anche a riconoscere eventuali tendenze e a comprendere le cause profonde delle non conformità, adottando misure preventive per evitarne il ripetersi in futuro.

L'intento principale è affrontare ogni problematica in modo efficace, promuovendo al contempo una cultura del miglioramento continuo. L'esperienza derivante dall'analisi degli errori passati diventa così una risorsa preziosa per ottimizzare i processi e favorire la crescita organizzativa.

Per garantire risultati concreti, è fondamentale adottare metodologie strutturate e strumenti adeguati, come la raccolta sistematica dei dati, l'analisi delle cause, la valutazione degli impatti e la definizione di piani d'azione correttivi e preventivi.

Infine, la gestione accurata della documentazione relativa ai problemi riscontrati e alle soluzioni adottate è cruciale per assicurare trasparenza, tracciabilità e la possibilità di effettuare revisioni periodiche, contribuendo così a un costante miglioramento della qualità.

3.6.1 Gestione dei rischi

All'interno del Piano di Progetto è presente una sezione dedicata all'individuazione dei rischi (sezione *Analisi dei rischi*[1] [versione 1.0.0]). Tale compito è assegnato al responsabile, che andrà quindi a scovare possibili cause di problemi, indicando inoltre la loro probabilità di occorrenza e le tecniche di mitigazione.

3.6.1.1 Codifica dei rischi

Per identificare i rischi adottiamo la seguente struttura:

- **Codice**:

R[a][numero]

dove:

  - R: indica che si tratta di un rischio;
  - a: indica la prima lettera della categoria di appartenenza del rischio (se ritenuto necessario è possibile inserire le prime 2 lettere);
  - numero: numero progressivo del rischio.

- **Descrizione**: descrizione del rischio;
- **Probabilità di occorrenza**: numero da 1 a 5, dove 5 indica altissima probabilità di occorrenza;
- **Pericolosità**: alta, media o bassa;
- **Tecniche di mitigazione**: azioni che permettono di arginare le possibili conseguenze del rischio.

4 Processi organizzativi

L'organizzazione garantisce che un processo esista e sia funzionale. In questa sezione infatti descriviamo le attività di mantenimento e miglioramento delle capacità organizzative per gestire il ciclo di vita del software in modo efficace.

[1] https://techminds-unipd.github.io/docs/RTB/documenti_esterni/piano_progetto/piano-di-progetto.pdf#analisi-dei-rischi

Anche in questo caso lo standard ISO/IEC 12207:1995 definisce i processi che concorrono alla formazione dei processi organizzativi. La loro attuazione diventa punto cruciale per la buona riuscita del progetto software.

4.1 Gestione dei processi

La gestione dei processi è essenziale per garantire che tutte le attività relative al ciclo di vita del software siano condotte in modo strutturato, supervisionato e orientato agli obiettivi. Questo processo fornisce una struttura per pianificare, monitorare e controllare lo svolgimento del progetto.

Obiettivi principali di tale processo sono:
- Assicurare che il progetto resti allineato agli obiettivi e soddisfi i requisiti del proponente;
- Minimizzare i rischi e le incertezze, attraverso una supervisione continua e azioni preventive;
- Ottimizzare l'uso delle risorse, migliorando l'efficienza;
- Mantenere la qualità del prodotto software, rispettando vincoli di tempo e costo.

I risultati di tutto ciò si concretizzano nel piano di progetto, documento utile per il gruppo ma anche per il committente e per il proponente in quanto funge da presentazione dei processi organizzativi del team.

4.1.1 Organizzazione dei ruoli

In conformità a quanto descritto nel regolamento del progetto didattico offerto dal committente e come indicato in Tabella 1, ogni persona che svolge un determinato ruolo deve assumersi diverse responsabilità.

Ruoli	Responsabilità
Responsabile	Coordina l'elaborazione di piani e scadenze. Approva il rilascio di prodotti parziali o finali (SW, documenti). Coordina le attività del gruppo.
Amministratore	Assicura l'efficienza di procedure, strumenti e tecnologie a supporto del way of working.
Analista	Svolge le attività di analisi dei requisiti.
Progettista	Svolge le attività di progettazione (design).
Programmatore	Svolge le attività di codifica.
Verificatore	Svolge le attività di verifica.

Tabella 1: Responsabilità ruoli.

4.1.1.1 Rotazione dei ruoli

Lo scopo che ci poniamo riguardo la rotazione dei ruoli è permettere ed assicurare a tutti di assumere almeno una volta ogni ruolo durante l'arco temporale in cui si svolge il progetto. Questo ci permette di apprendere le basi di ogni singola posizione e le relative responsabilità.

La politica di rotazione è determinata in accordo tra tutto il gruppo, tenendo ovviamente conto dei ruoli che non sono ancora stati coperti da ogni membro. Poniamo particolare attenzione al fatto che l'assegnazione dei ruoli è fluida, ovvero che privilegiamo le attività da compiere piuttosto che il mantenimento fisso del ruolo. Questo significa che in caso di necessità potremmo svolgere compiti afferenti a un ruolo diverso da quello assegnato per lo sprint corrente.

Infine, durante la fase di scambio dei ruoli, chi lascia un ruolo deve condividere la conoscenza acquisita per facilitare la transizione. Per fare questo è fondamentale:
- Utilizzare canali vocali (privilegiando i canali *Discord*) per uno scambio più immediato;
- Utilizzare documenti informali per velocizzare il passaggio di conoscenza;

- Documentare in modo chiaro e dettagliato ogni azione che è reputata non banale.

4.1.1.2 Responsabile

Il *responsabile* è una figura di riferimento non solo per il team, ma anche per il committente e il proponente in quanto svolge il ruolo di mediatore tra le parti.

La sua posizione centrale è dovuta al fatto che svolge l'attività di **pianificazione**, componente fondamentale perché programma un piano d'azione organizzato che mira al raggiungimento degli obiettivi posti per un determinato periodo.

In particolare:
- Coordina la pianificazione:
  - alloca le risorse specificando le tecnologie da utilizzare;
  - definisce i tempi e i costi;
  - redige piani dettagliati da inserire nel piano di progetto.
- Verifica la fattibilità del piano proposto;
- Gestisce le relazioni con gli altri *stakeholder*;
- Controlla i progressi del progetto;
- Gestisce il bilancio.

4.1.1.3 Amministratore

Funge da guida per quanto concerne le norme di progetto e predispone l'ambiente di lavoro utilizzato dal gruppo. In caso di necessità, affronta e risolve le problematiche relative alla gestione dei processi e collabora con il responsabile per garantire la qualità del prodotto offerto. Inoltre, si occupa della stesura delle norme di progetto.

4.1.1.4 Analista

Ha lo scopo di redigere l'analisi dei requisiti andando quindi a identificare, documentare e studiare a fondo le esigenze e le specifiche del progetto, traducendole in requisiti dettagliati e non ambigui.

In particolare:
- Definisce il problema;
- Stabilisce gli obiettivi analizzando il contesto;
- Definisce i requisiti per raggiungere gli obiettivi;
- Comprende a fondo i bisogni impliciti ed espliciti.

4.1.1.5 Progettista

Partendo dal lavoro dell'analista effettua scelte progettuali per definire il «come» devono essere implementati i requisiti individuati durante la fase di analisi. Effettua quindi decisioni di natura tecnica e tecnologica, andando in particolare a:
- Effettuare scelte tecnologiche e strutturali per soddisfare i requisiti;
- Progettare l'architettura del prodotto.

4.1.1.6 Programmatore

Scrive il codice software con l'obiettivo che esso rispecchi i requisiti individuati con l'analisi e le specifiche fornite dai progettisti.

Il codice scritto deve:
- Contenere gli strumenti per la verifica e la *validazione*;
- Rispecchiare l'architettura ideata dai progettisti;
- Essere conforme ai requisiti di qualità stabiliti.

4.1.1.7 Verificatore

Ruolo chiave in quanto ha lo scopo di controllare la qualità del lavoro svolto dagli altri ruoli e che tale lavoro sia stato eseguito secondo quanto concordato dal gruppo e prefissato dalle specifiche tecniche.

Ha quindi il compito di:
- Verificare la conformità del lavoro svolto rispetto alle norme di progetto;
- Evidenziare eventuali errori e richiederne la correzione;
- Redigere il piano di qualifica definendo le metriche e i corrispettivi risultati.

4.1.1.8 Strumenti e tecnologie

Per la gestione dei ruoli e per le attività che ogni ruolo deve svolgere abbiamo deciso di utilizzare i seguenti strumenti e le seguenti tecnologie:
- GitHub Board[2] [visitato il: 10/02/2025] per le attività di pianificazione;
- Typst: per la documentazione e la creazione di diagrammi, tabelle e grafici;
- Ambiente Google Drive: per scambiare file utili in modo veloce e condiviso;
- Telegram e Discord: per scambiare consigli sui ruoli in modo immediato.

4.1.2 Coordinamento interno

Per raggiungere una buona efficienza nello svolgimento del progetto è fondamentale un ottimo coordinamento interno, che permetta di collaborare e comunicare in modo rapido e semplice.

4.1.2.1 Reperibilità dei membri

Per far sì che tutti i membri siano a conoscenza degli impegni altrui, ciascun membro ha indicato i propri orari di disponibilità durante l'arco della settimana in un foglio Google condiviso. Il documento è modificabile in caso vi siano nuovi impegni sorti tra l'inizio del progetto e la fase attuale.

Ogni membro inoltre si assume l'impegno di portare a termine le proprie task principalmente in modo asincrono e, in caso di necessità collaborative, anche in modo sincrono. La gestione individuale di tali attività è totalmente libera, tenendo comunque in considerazione l'impegno preso nei confronti del gruppo.

Infine, in caso di impegni personali, accademici o imprevisti è necessario che l'interessato comunichi il prima possibile al responsabile l'impedimento nel portare a termine i propri compiti secondo i termini previsti.

4.1.2.2 Comunicazioni testuali

Per consentire al team di comunicare agilmente ci siamo dotati dei seguenti canali testuali:
- Discord: abbiamo creato un server per facilitare sia le comunicazioni testuali che quelle vocali. In particolare per le comunicazioni testuali abbiamo creato più canali:
  - generale: per comunicazioni generali;
  - diario-di-bordo: raccolta di dubbi da esprimere durante il diario di bordo;
  - analisi-dei-requisiti: canale relativo a dubbi o discussioni riguardo l'analisi dei requisiti e relativi use case;
  - *poc*: canale per dubbi o discussioni riguardo l'implementazione del PoC;
  - proponente: raccolta di dubbi da sottoporre al proponente.
- Telegram: il gruppo Telegram viene utilizzato per uno scambio rapido di informazioni di una certa rilevanza. Se invece sorgono dei dubbi o degli aspetti urgenti che possono essere risolti da un gruppo ristretto si predilige la comunicazione tra i singoli individui, per non congestionare e rallentare il lavoro degli altri membri. Anche su Telegram abbiamo creato vari canali:
  - General: comunicazioni generali;

[2] https://github.com/orgs/techminds-unipd/projects/1

- Daily Scrum: ogni membro indica le cose fatte il giorno prima, cosa farà durante il giorno, eventuali problemi sorti;
- Domande: utilizzato per raccogliere domande generiche.

4.1.2.3 Incontri

Per quanto concerne i meeting interni, che solitamente si tengono da remoto, utilizziamo il canale vocale di Discord, suddiviso in 3 canali (Generale, Generale 1 e Generale 2) in caso vi sia la necessità di lavorare contemporaneamente divisi in gruppi.

Gli incontri interni si verificano periodicamente per effettuare la sprint *retrospective* e lo *sprint planning* tra la fine di uno sprint e l'inizio del successivo. Si possono fissare anche dei meeting nel mezzo dello sprint per necessità di allineamento interno o per risolvere problematiche urgenti.

Ogni incontro interno verrà guidato e mediato dal responsabile. Oltre a questo, il responsabile deve stabilire una bozza dell'ordine del giorno, eventualmente da ampliare tenendo conto delle difficoltà e delle questioni emerse dal precedente incontro. Per ogni incontro interno effettuato il responsabile deve infine redigere il relativo verbale, compito delegabile all'amministratore nel caso in cui sorgano dei problemi personali o degli elementi di maggiore urgenza da risolvere.

4.1.2.4 Strumenti e tecnologie

Come evidenziato sopra, gli strumenti e tecnologie di supporto scelte sono:
- Fogli Google;
- Discord;
- Telegram.

4.1.3 Coordinamento con il proponente

Un frequente coordinamento tra il gruppo e il proponente è fondamentale per assicurare che il prodotto evolva coerentemente con le richieste del proponente. Inoltre, ricevere dei feedback periodici sul lavoro svolto ci indica se è conforme agli obiettivi fissati.

4.1.3.1 Comunicazioni testuali

Nella fase iniziale abbiamo utilizzato il servizio Gmail per le comunicazioni testuali; dopo l'aggiudicazione dell'appalto, invece, il proponente ha creato un canale Slack in cui possiamo fare delle domande e scambiare file, risorse e materiali. Gli scopi di questo canale sono rispondere ai dubbi in modo veloce e mirato e facilitare lo scambio di informazioni.

Il proponente ha aggiunto in questo canale diverse figure professionali esperte nelle tecnologie da utilizzare nel progetto, per poter rispondere in maniera esaustiva ai diversi tipi di dubbi e problematiche che possono sorgere.

4.1.3.2 Incontri

I meeting esterni si svolgono su Microsoft Teams. Nella fase iniziale del progetto non sono stati svolti degli incontri di revisione in quanto il primo periodo è stato dedicato allo studio delle tecnologie e del capitolato. Sono però stati svolti degli incontri di formazione in presenza, offerti dal proponente. Dal periodo natalizio in poi invece vengono svolti degli incontri quando una delle due parti (noi in quanto gruppo o il proponente) lo ritengono necessario, ovvero quando sono stati raggiunti degli obiettivi di avanzamento considerevoli.

Anche in questo caso è il responsabile a mediare l'incontro e, quando necessario, lascia la parola agli altri membri del gruppo. Per ogni incontro esterno redige il relativo verbale (delegabile all'amministratore) e lo invia al proponente per l'approvazione tramite una firma.

4.1.3.3 Strumenti e tecnologie

Per i rapporti con il proponente utilizziamo le seguenti tecnologie e strumenti:

- Slack;
- Microsoft Teams;
- Gmail.

4.1.4 Organizzazione delle attività

In questa sezione sono presenti le metodologie scelte per organizzare e gestire le varie attività da svolgere nel corso del progetto.

4.1.4.1 Metodologie utilizzate

Abbiamo deciso di adottare un approccio agile allo sviluppo del progetto prendendo ispirazione dal modello *Scrum*, metodologia ben consolidata e utilizzata nel contesto lavorativo. Questa scelta è stata guidata dai numerosi vantaggi che comporta, per esempio:

- Maggiore flessibilità;
- Maggiore produttività;
- Maggiore trasparenza e comunicazione.

Alla base delle strategie di tipo agile vi è l'adozione della pratica di Continuous Integration (CI) che:

- Favorisce il lavoro di gruppo promuovendo la collaborazione e la condivisione;
- Promuove il miglioramento continuo, attraverso la retrospettiva, identificando e risolvendo le problematiche in modo mirato e rapido;
- Garantisce una distribuzione efficace delle responsabilità e delle attività;
- Assicura trasparenza facilitando l'analisi e la comprensione del processo di sviluppo al proponente e al committente.

Dividiamo quindi le attività da svolgere in sprint, periodi di tempo che durano mediamente due settimane (salvo eccezioni) e durante i quali il gruppo si impegna a svolgere le seguenti attività:

- Daily scrum: ogni membro del gruppo ogni giorno comunica agli altri le attività svolte il giorno precedente, quelle che ha intenzione di svolgere nel giorno corrente ed eventuali dubbi. Questo aiuta la comunicazione interna del gruppo e consente un minimo allineamento interno giornaliero;
- Sprint planning: all'inizio di ogni sprint, guidati dal responsabile, definiamo cosa dovremo consegnare al termine del tempo programmato e predisponiamo il repository GitHub con le *issue* necessarie per svolgere le attività individuate. Inoltre preventiviamo le ore che ognuno pensa saranno necessarie per portare a termine i propri compiti, informazioni che poi il responsabile inserisce nel piano di progetto;
- Sprint retrospective: avviene al termine dello sprint ed è finalizzata ad analizzare l'andamento dello sprint. Ispezionamo cosa non ha funzionato e cosa è andato bene per apprendere e migliorare. É anche un evento propedeutico allo sprint successivo in quanto volto a migliorare le performance future.

4.1.4.2 Milestone e sprint

Il progetto è suddiviso in periodi chiamati *milestone* e sprint. Tenendo conto della scelta di un modello agile, abbiamo definito questi due concetti:

- Milestone: rappresentano le revisioni di progetto. Ogni milestone, per essere raggiunta, richiede il completamento di periodi di tempo chiamati sprint;
- Sprint: periodi di lavoro con durata fissa, generalmente due settimane, in cui completiamo le attività pianificate rispettando le scadenze. La durata dello sprint è comune a tutto il gruppo e aiuta a mantenere disciplina e produttività. Variazioni sono consentite solo in alcuni casi e di comune accordo. Gli sprint sono definiti nel piano di progetto a cura del responsabile.

4.1.4.3 Issue

Utilizziamo le issue di GitHub per tracciare le attività da svolgere all'interno di ogni sprint.

4.1.4.3.1 Creazione

Le issue vengono solitamente create dall'amministratore (ad eccezione di alcuni casi come lo sprint planning o un'elevata urgenza), specificando i seguenti attributi:

- Titolo: descrive in modo breve e conciso lo scopo della issue;
- Descrizione: breve spiegazione testuale di cosa dovrà essere fatto per poter portare a termine la issue, eventualmente anche con una checklist;
- Label: tag che identifica la categoria (esempi: Norme di progetto, Allenamento, Glossario, bug, fix). In caso di necessità può essere creata una nuova label;
- Project: GitHub Projects a cui la issue è associata. Se sono presenti delle dashboard (vedi Sezione 4.1.4.4 per approfondimenti) associate ad un progetto, le issue associate saranno visibili anche da lì;
- Priority: priorità della issue rispetto alle altre, scelta tra alta, media e bassa;
- Size: dimensione della issue in termini di giornate di lavoro, scelta tra XS, S, M, L, XL;
- Sprint: ad ogni issue associamo lo sprint in cui deve essere completata;
- Giorni previsti: giorni previsti per il completamento della issue, includendo la verifica;
- Milestone: alla issue associamo la milestone di riferimento.

Nel caso in cui una issue sia assegnabile fin da subito ad un membro del gruppo si procede completando anche il campo «Assignees». Se invece è una issue che potrebbe essere svolta da più individui perchè due o più persone svolgono lo stesso ruolo, allora si lascia il campo «Assignees» vuoto e gli interessati andranno a suddividersi le issue di competenza in separata sede.

4.1.4.3.2 Ciclo di vita

Di seguito descriviamo il ciclo di vita di una issue:

1. L'amministratore (o chi di competenza) accede al repository GitHub e crea la issue seguendo quanto descritto in Sezione 4.1.4.3.1;
2. Accedendo poi alla dashboard di progetto sposta la issue dalla colonna «No Status» alla colonna «Sprint *Backlog*»;
3. L'assegnatario, quando prende in carico la issue, deve svolgere le seguenti azioni:
   - accedere alla dashboard e spostare la issue nella colonna «In progress». In alternativa può cambiare lo stato della issue modificando il campo «Status»;
   - inserire la data di inizio della issue;
   - inserire la stima del termine della issue.
4. Una volta che la issue è considerata terminata, l'assegnatario apre una Pull Request su GitHub scrivendo sulla descrizione close numero_issue e aggiornando lo stato della issue da «In progress» a «In review»;
5. I verificatori seguono la procedura di verifica;
6. Se e quando l'esito della verifica è positivo, la issue viene ufficialmente spostata nella colonna «Done» della dashboard.

4.1.4.4 Board

Board è un sistema visivo che permette di organizzare e tracciare il progresso delle attività. La motivazione per cui abbiamo scelto questo sistema è che permette a tutti di avere una panoramica chiara ed immediata del lavoro, migliorando anche la comunicazione e la trasparenza. Possiamo facilmente individuare se ci sono delle attività bloccate e intervenire tempestivamente. Inoltre ci consente di visualizzare l'avanzamento delle issue in modo veloce ed intuitivo.

Per le motivazioni sopra descritte abbimo deciso di utilizzare gli strumenti di GitHub Projects, in particolare:

- GitHub Board: https://github.com/orgs/techminds-unipd/projects/1/views/1

  Dashboard suddivisa in 5 colonne che corrispondono alle fasi del ciclo di vita delle issue:
  - No Status: vengono inserite le issue che non hanno una particolare urgenza; solo una volta terminate le issue relative allo sprint verranno completate anche quelle presenti in questa colonna;
  - Sprint Backlog: presenta tutte le issue che devono essere completate entro fine sprint, organizzate in ordine di priorità;
  - In progress: raccoglie le issue che sono state iniziate ma non ancora terminate;
  - In review: raccoglie le issue che si trovano in fase di revisione;
  - Done: raccoglie le issue terminate.

  Ogni membro del gruppo è tenuto a controllare frequentemente la dashboard per avere una panoramica della situazione dello sprint e a gestire le attività che ha preso in carico;

4.1.4.5 Diagrammi di Gantt

Per pianificare le attività di ogni sprint utilizziamo dei diagrammi di Gantt generati in Typst con la libreria timeliney. Il diagramma ha una riga per ogni issue dello sprint con relativa linea del tempo di fianco che indica con dei colori i tempi preventivati ed effettivi dell'attività. Per ottenere i tempi effettivi di una issue abbiamo uno script *Python* che, utilizzando le *API* di GitHub, recupera la data di inizio della prima commit associata alla issue e la data di chiusura della Pull Request ad essa associata. Queste due date precedentemente citate corrispondono alla data di inizio e di fine della issue, che pur non essendo sempre esatte sono una buona approssimazione della realtà.

4.1.4.6 Strumenti e tecnologie

Per la gestione dell'organizzazione delle attività ci siamo dotati dei seguenti strumenti/tecnologie:

- Ambiente GitHub: GitHub Projects, GitHub Boards.

4.2 Gestione delle infrastrutture

La gestione delle infrastrutture ha lo scopo di stabilire e mantenere l'infrastruttura a supporo di qualsiasi altro processo. L'infrastruttura può includere: hardware, software, strumenti, tecniche, standard e strutture per lo sviluppo, il funzionamento o la manutenzione.

4.2.1 Attività

Secondo lo standard ISO/IEC 12207:1995 questo processo è formato dalle seguenti attività, che ci impegniamo a rispettare:

1. Attuazione del processo: definizione e documentazione dell'infrastuttura, tenendo in considerazione le procedure descritte per ogni attività e gli standard da adottare;
2. Realizzazione dell'infrastruttura: l'infrastuttura adottata deve essere installata, configurata o attuata (in base alla sua natura) in tempo per l'esecuzione del processo pertinente, tenendo conto dei suoi costi, della sicurezza e dell'eventuale spazio richiesto;
3. Manutenzione dell'infrastruttura: è importante che monitoriamo, manteniamo adatta e modifichiamo in caso di necessità l'architettura scelta, in modo da soddisfare i requisiti del processo in corso.

4.2.2 Strumenti e tecnologie

In questa sezione approfondiamo le tecnologie e gli strumenti che formano la nostra architettura a supporto dei processi organizzativi.

Elenco di tecnologie e strumenti:

- GitHub e strumenti annessi: piattaforma per il versionamento del codice, gestione di repository Git e gestione di un project attraverso GitHub Board;

- Telegram: app di messaggistica istantanea per chat di gruppo interne;
- Discord: piattaforma di comunicazione con chat vocali e testuali per scopi interni al gruppo;
- Ambiente condiviso di Google: suite collaborativa online per editing e condivisione di documenti in tempo reale;
- Slack: strumento di messaggistica aziendale utilizzato con il proponente;
- Microsoft Teams: piattaforma per collaborazione aziendale con chat e videochiamate utilizzata per le riunioni con il proponente;
- Gmail: servizio di posta elettronica di Google utilizzato per le comunicazioni con committente e proponente;
- Typst: linguaggio per creare documenti con formattazione avanzata, utilizzato per la redazione della documentazione e per la creazione di diagrammi, tabelle e grafici.

4.3 Miglioramento del processo

Il miglioramento del processo ha lo scopo di stabilire, valutare, misurare, controllare e migliorare il processo del ciclo di vita del software (standard ISO/IEC 12207:1995).

4.3.1 Attività

Questo processo consiste nelle seguenti attività:

1. Definizione del processo: con l'organizzazione abbiamo definito una serie di processi organizzativi da applicare alle varie attività e la loro applicazione deve essere documentata;
2. Valutazione del processo: sviluppare, documentare ed applicare una procedura di valutazione del processo. Tale valutazione deve essere registrata. È necessario pianificare ed effettuare revisioni dei processi a intervalli regolari per garantire la loro continua idoneità ed efficacia alla luce dei risultati della valutazione;
3. Miglioramento dei processi: a seguito della valutazione e revisione del processo è fondamentale apportare gli eventuali miglioramenti che riteniamo necessari. La documentazione del processo deve essere aggiornata per rispecchiare il suo miglioramento. Inoltre è importante svolgere le seguenti azioni:
   - raccogliere dati storici, tecnici e di valutazione per ottenere una maggiore comprensione dei punti di forza e delle debolezze dei processi impiegati;
   - utilizzare l'analisi come feedback per migliorare i processi e raccomandare cambiamenti in una direzione più efficiente;
   - documentare i costi per il miglioramento dell'organizzazione.

4.3.2 Impegni per il miglioramento

Con lo scopo di adattarci alle esigenze dei vari processi, il gruppo si impegna a:

- Seguire costantemente il principio di miglioramento continuo durante ogni fase del progetto;
- Individuare in modo proattivo le attività, i ruoli e altre opportunità di miglioramento, cercando in caso nuove alternative;
- Condurre una retrospettiva alla fine di ogni sprint, andando quindi ad approfondire ogni aspetto che non ha funzionato e individuando le possibili aree di miglioramento;
- Attuare azioni correttive da implementare in modo immediato.

4.4 Formazione del personale

L'acquisizione, la fornitura, lo sviluppo, il funzionamento o la manutenzione dei prodotti software dipendono in gran parte da personale esperto e qualificato. È importante che ognuno acquisisca una competenza adeguata per utilizzare le tecnologie scelte.

4.4.1 Attività

1. Attuazione del processo: analisi dei requisiti del progetto per individuare risorse e competenze necessarie, creazione di un piano di auto-formazione;
2. Sviluppo del materiale di formazione: ricerca e ordinamento di guide, manuali o mini-progetti di prova da mettere a disposizione di tutto il gruppo;
3. Attuazione del piano di formazione: realizzazione del piano di formazione.

Questo processo di formazione è importante si svolga per l'intera durata del progetto, ovviamente con obiettivi diversi di fase in fase. Per il primo sprint la formazione si è concentrata sullo studio delle tecnologie da utilizzare ed è stata organizzata nel seguente modo:

1. Attuazione del processo: abbiamo individuato, in accordo con il proponente, le tecnologie da studiare attraverso un approfondimento del capitolato. Abbiamo poi creato un piano di formazione;
2. Sviluppo del materiale di formazione: il proponente ci ha fornito delle risorse specifiche per ogni tecnologia e abbiamo trovato alcuni mini-progetti di prova per iniziare a prendere confidenza con le prime tecnologie;
3. Attuazione del piano di formazione: ogni individuo ha attuato il piano di formazione, contattando gli altri membri del gruppo in caso di dubbi. Inoltre, abbiamo partecipato a 3 corsi di formazione offerti dal proponente.