gammardx@gmail.com

Norme di progetto

Versione 1.0.0

Informazioni documento

Redattore	A. Zanella M. Cossi D. Erba M. Stevanin
Verificatore	M. Stevanin G. Bottacin D. Erba M. Cossi
Destinatari	T. Vardanega R. Cardin

Registro delle modifiche

Versione	Data	Autore	Verificatore	Dettaglio
1.0.0	02/02/2025	M. Stevanin	G. Bottacin	Correzioni minori, approvazione da parte di M. Cossi
0.9.0	23/01/2025	M. Cossi	G. Bottacin	Scrittura sezione “Sviluppo” (sez. 2.2).
0.8.0	21/01/2025	D. Erba	G. Bottacin	Completamento sezione “Metriche di Qualità del Prodotto” (sez. 7).
0.7.0	12/01/2025	M. Stevanin	M. Cossi	Scrittura sezione “Processo di Miglioramento” (sez. 4.2).
0.6.0	30/12/2025	M. Cossi	D. Erba	Scrittura delle metriche di qualità (sez. 5-6-7).
0.5.0	29/12/2025	D. Erba	M. Cossi	Scrittura della sezione “Processi di Qualifica” (sez. 3.3).
0.4.0	04/12/2025	M. Cossi	D. Erba	Scrittura della sezione “Processi Organizzativi” (sez. 4).
0.3.0	03/12/2025	M. Stevanin	D. Erba	Scrittura della sezione “Gestione delle Configurazioni” (sez. 3.2).
0.2.1	21/11/2025	M. Cossi	M. Stevanin	Correzioni minori.
0.2.0	20/11/2025	D. Erba	M. Stevanin	Scrittura della sezione “Processi di supporto” (sez. 3).
0.1.0	20/11/2025	M. Cossi	G. Bottacin	Scrittura della sezione “Processi primari” (sez. 2).
0.0.1	10/11/2025	A. Zanella	M. Stevanin	Scrittura introduzione (sez. 1)

Indice

GammardX

Contents

1 Introduzione

1.1 Scopo del documento

Il presente documento ha l'obiettivo di descrivere il *Way of Working*$_G$ adottato dal gruppo **GammardX** durante lo svolgimento del progetto didattico, stabilendo le linee guida e le best practice che ciascun membro deve seguire per garantire un approccio efficiente, coerente e professionale nello sviluppo del progetto.
Per la definizione del *Way of Working*$_G$, il gruppo ha preso come riferimento lo Standard ISO/IEC 12207:1995, che individua tre principali tipologie di processi:

- **Primari**: indispensabili per la realizzazione del progetto;
- **Di supporto**: che affiancano i processi primari nello svolgimento delle rispettive attività;
- **Organizzativi**: volti a gestire e coordinare le attività di carattere più generale.

Il documento è strutturato in base ai processi del ciclo di vita del software, ognuno dei quali si articola in una serie di attività definite da obiettivi, scopi e strumenti specifici. Inoltre, vengono descritte le convenzioni e le modalità di utilizzo degli strumenti adottati durante lo sviluppo del prodotto.
La redazione del documento segue un approccio incrementale, ovvero soggetto a revisioni, aggiornamenti e ottimizzazioni continue, al fine di migliorare progressivamente il metodo di lavoro. Tutti i membri del gruppo si impegnano a consultare regolarmente il documento e ad attenersi alle regole in esso definite, assicurando così uniformità e qualità nell'intero processo di sviluppo.

1.2 Scopo del prodotto

Il capitolato C6 "Second Brain" di **Zucchetti S.p.A.** propone lo sviluppo di un applicativo web che consenta la redazione di documenti di testo in linguaggio *Markdown*, offrendo all'utente la possibilità di visualizzare contemporaneamente sia l'editor che il rendering del documento. Inoltre, è previsto l'inserimento di un assistente virtuale basato su *Large Language Model* (*LLM*$_G$), progettato per supportare la scrittura del testo. L'assistente dovrà eseguire operazioni che spaziano da compiti semplici — come il riassunto, la riscrittura e la traduzione — a funzionalità più complesse, quali la stesura autonoma di un documento a partire da un prompt o la fornitura di feedback critici personalizzati in base alle esigenze dell'utente.
L'obiettivo che si è posto il gruppo è quello di realizzare il progetto entro il **20 Marzo 2026** con un budget di: **12.565,00 €.**

1.3 Glossario

La realizzazione di un *sistema*$_G$ software richiede, ancor prima della scrittura del codice, un'importante operazione di confronto, analisi e progettazione: per supportare e facilitare il lavoro asincrono tutte le informazioni derivate da questa attività saranno appositamente documentate.
È completamente ragionevole tuttavia pensare che tali documenti potrebbero contenere parole e terminologie complesse o comunque non direttamente comprensibili: è stato deciso dunque di realizzare un Glossario, nella quale saranno contenuti le spiegazioni relative a tali termini. Tale documento è in costante aggiornamento ed è reperibile, nella sua versione attuale, al seguente indirizzo.
Le parole che possiedono un riferimento nel Glossario saranno indicate nel modo che segue: *parola*$_G$.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Capitolato C6 - Second Brain**
  - https://www.math.unipd.it/~tullio/IS-1/2025/Progetto/C6.pdf
    Ultimo accesso: 02/02/2026
- **Regolamento Progetto IS**
  - https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/PD1.pdf
    Ultimo accesso: 02/02/2026
- **Standard ISO/IEC 12207:1995**
  - `www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf`
    Ultimo accesso: 02/02/2026

1.4.2 Riferimenti informativi

- **Glossario**
  - https://gammardx.github.io/Documents/Documenti interni/Glossario.pdf
    Ultimo accesso: 02/02/2026

2 Processi Primari

La semplice scrittura del codice e l'esecuzione di alcuni *Test*$_G$ non sono sufficienti a garantire lo sviluppo di un *sistema*$_G$ software di alta qualità e destinato a perdurare nel tempo. Per raggiungere tale obiettivo, è necessario adottare un approccio strutturato basato su modelli e processi definiti. Tra i processi primari identificati dallo standard ISO/IEC 12207, si distinguono i processi di:

- **Fornitura**
- **Sviluppo**

2.1 Fornitura

La fornitura è il processo primario di competenza del fornitore, volto a determinare e analizzare le azioni necessarie per la realizzazione del prodotto finale. Questa fase prevede uno studio preliminare dei requisiti che il software dovrà soddisfare. L'output di tale analisi costituisce la base per la negoziazione con il proponente e permette di elaborare una pianificazione delle attività lavorative, inclusa una stima della data di consegna.

2.1.1 Strumenti a supporto

Per supportare le attività di fornitura e coordinamento, sono stati selezionati i seguenti strumenti:

- **GitHub**: utilizzato per la gestione del *Backlog*$_G$ e il *sistema*$_G$ di ticketing, essenziali per monitorare l'avanzamento dei lavori. Inoltre, le funzionalità di project management permettono la visualizzazione dei diagrammi di Gantt per la pianificazione;
- **Discord e WhatsApp**: canali dedicati alla comunicazione rapida e alle riunioni interne;
- **Google Mail**: per le comunicazioni formali scritte verso l'esterno;
- **Google Meet**: piattaforma utilizzata per le riunioni svolte in remoto con l'azienda proponente.

2.1.2 Attività previste

Il processo di fornitura si articola nelle seguenti attività, descritte in dettaglio secondo la sequenza logica di esecuzione:

- **Inizializzazione**: Tale attività preliminare contempla l'analisi, a carico del fornitore, delle richieste avanzate dal proponente. La valutazione tiene conto dei vincoli organizzativi, tecnologici o di altra natura. In questa fase, il fornitore accerta la propria capacità realizzativa e determina gli eventuali requisiti che saranno oggetto di successiva contrattazione con il proponente.
- **Preparazione delle risposte**: Consiste nell'elaborazione di una contro-proposta formale indirizzata al proponente. Questo documento sintetizza le analisi e le conclusioni derivate dall'attività di Inizializzazione.
- **Contrattazione**: Rappresenta la fase di dialogo ufficiale con il proponente, durante la quale vengono presentate e discusse le risposte precedentemente elaborate. L'obiettivo primario di questa interazione è pervenire a un accordo definitivo e alla formalizzazione del contratto sui requisiti.
- **Pianificazione**: Una volta stabiliti i requisiti finali, il fornitore è tenuto a definire l'organizzazione e il metodo di lavoro atti a garantire la qualità del $sistema_G$. Tale attività include la selezione del modello di ciclo di vita del Software (ove non vincolato da contratto), l'identificazione delle risorse umane e strumentali, la scelta delle tecnologie di sviluppo e la valutazione dei rischi potenziali correlati.
- **Esecuzione e controllo**: In ottemperanza ai documenti di pianificazione, il fornitore procede alla realizzazione del prodotto. Contestualmente, si attiva il monitoraggio continuo della qualità degli artefatti prodotti e della progressione del progetto rispetto al cronoprogramma stabilito.
- **Revisione e valutazione**: Il fornitore mantiene un canale di comunicazione periodico con il proponente durante l'intero ciclo di sviluppo. Tale interazione è necessaria per acquisire feedback sul lavoro svolto e per effettuare una valutazione aggiornata dello stato di avanzamento.
- **Consegna e completamento**: A conclusione del progetto e a seguito delle attività di qualifica, il fornitore provvede alla fornitura del prodotto finale al proponente, impegnandosi a fornire il supporto post-consegna necessario.

2.1.3 Documentazione fornita

Di seguito vengono elencati i documenti prodotti da **GammardX**, specificandone la tipologia d'uso (Interno o Esterno).

2.1.3.1 Analisi dei Requisiti

Documento volto a definire in modo preciso i requisiti funzionali e non funzionali del *sistema*$_G$ *Second Brain* (*Capitolato*$_G$ C6). I requisiti sono individuati principalmente attraverso una descrizione approfondita dei Casi d'Uso, supportata dai relativi diagrammi per illustrare le interazioni tra attori e *sistema*$_G$. Include le matrici di tracciamento per garantire la copertura dei requisiti.
Tipologia: Uso Esterno

2.1.3.2 Preventivo costi e assunzione impegni

Documento che formalizza l'impegno dei membri del gruppo, fissato a un monte ore produttivo di 91 ore ciascuno. Dettaglia la stima dei costi e pianifica la rotazione dei ruoli, garantendo che ogni componente ricopra tutte le posizioni per un tempo equo, assicurando un bilanciamento del carico di lavoro.
Tipologia: Uso Esterno

2.1.3.3 Glossario

Strumento fondamentale per la standardizzazione del linguaggio, creato per prevenire ambiguità terminologiche. Raccoglie in ordine alfabetico le definizioni dei termini tecnici, contrassegnati nei documenti tramite il pedice "*G*", assicurando uniformità e chiarezza comunicativa tra tutti gli stakeholder del progetto.
Tipologia: Uso Interno

2.1.3.4 Lettera di Candidatura

Documento ufficiale con cui il gruppo **GammardX** comunica formalmente l'intenzione di candidarsi per il capitolato C6 *"Second Brain"*, proposto dall'azienda **Zucchetti S.p.A.** Il documento riassume le motivazioni della scelta e fornisce i riferimenti ai repository del gruppo.
Tipologia: Uso Esterno

2.1.3.5 Lettera di Presentazione

Documento formale che accompagna le consegne principali (*RTB*$_G$ e *PB*$_G$), ufficializzando la candidatura alle relative revisioni.

2.1.3.6 Norme di Progetto

Documento che definisce il *Way of Working* del gruppo, basato sullo standard ISO/IEC 12207:1995. Descrive i processi (Primari, di Supporto e Organizzativi), le convenzioni e gli strumenti adottati. Segue un approccio incrementale per ottimizzare progressivamente le

metodologie di lavoro e garantire la qualità del prodotto.
Tipologia: Uso Interno

2.1.3.7 Piano di Progetto

Documento di pianificazione strategica gestito con approccio incrementale. Espone le attività svolte e da svolgere, analizzando tempi previsti ed effettivi e gestendo i rischi di progetto. È essenziale per il monitoraggio dello stato di avanzamento e per la consuntivazione periodica di ore e costi.
Tipologia: Uso Esterno

2.1.3.8 Piano di Qualifica

Descrive i metodi di qualifica (*Verifica*$_G$ e *Validazione*$_G$) adottate, nonché i test effettuati sul prodotto e i rispettivi esiti.
Tipologia: Uso Esterno

2.1.3.9 Valutazione dei Capitolati

Analisi comparativa redatta per illustrare le motivazioni che hanno condotto alla scelta del capitolato C6. Il documento esamina tutti i capitolati proposti, evidenziando per ciascuno punti di forza e criticità, giustificando così la decisione finale del gruppo.
Tipologia: Uso Esterno

2.1.3.10 Verbale Interno

Resoconto delle riunioni organizzative e tecniche svolte esclusivamente tra i membri del gruppo.
Tipologia: Uso Interno

2.1.3.11 Verbale Esterno

Documentazione ufficiale delle riunioni tenutesi con soggetti esterni al gruppo (es. proponenti, committenti).
Tipologia: Uso Esterno

2.2 Sviluppo

Il **Processo di Sviluppo** stabilisce le attività che hanno come scopo quello di *Analisi dei Requisiti*$_G$, la progettazione, la codifica del Software, l'installazione e l'accettazione di quanto prodotto.

2.2.1 Attività previste

Le attività previste dal **Processo di Sviluppo** in base allo standard *ISO/IEC 12207:1995* sono le seguenti:

- **Implementazione del processo:** ovvero la scelta del Ciclo di Vita del Software più appropriato in base allo scopo, l'importanza e la complessità del progetto;
- **Analisi dei Requisiti:** consiste nell'identificazione e nella definizione delle necessità dell'utente finale in relazione alle funzionalità che il Software deve offrire. Un'*Analisi dei Requisiti*$_G$ completa deve descrivere le funzionalità del *Sistema*$_G$, i bisogni degli utilizzatori finali e vincoli imposti dal proponente;
- **Progettazione dell'architettura:** ovvero l'individuazione degli elementi *Hardware* e *Software* del prodotto finale, affinché tutti i requisiti individuati siano soddisfatti (a questo proposito, fondamentale è il tracciamento dei requisiti stessi);
- **Analisi dei Requisiti Software:** ovvero l'analisi del modo in cui il Software soddisfa i requisiti lato utente. Deve includere anche le caratteristiche di qualità: caratteristiche funzionali (includendo anche eventuali requisiti prestazionali), le interfacce di ogni elemento Software e requisiti di sicurezza;
- **Progettazione dell'architettura Software:** consiste nel definire le diverse componenti del *Sistema*$_G$ e il loro funzionamento, ponendo l'attenzione sulla struttura generale, non nel dettaglio implementativo;
- **Progettazione in dettaglio del Software:** ovvero la progettazione in dettaglio delle singole componenti Software, fino ad individuare le singole unità di ciascuna;
- **Codifica e *testing* del Software:** ovvero la produzione di tutte le unità di tutte le componenti individuate. Ciascuna di queste parti dovrà essere adeguatamente testata per assicurare il suo corretto funzionamento;
- **Integrazione del Software:** ovvero l'integrazione delle varie parti di una componente nella sua componente completa. Essenziali sono i *Test*$_G$ per assicurare il corretto funzionamento;
- **Test di qualifica del Software:** ovvero la realizzazione di appositi *Test*$_G$ per assicurare la conformità del Software agli obiettivi di qualità attesi;
- **Integrazione del Sistema:** ovvero l'integrazione di tutte le componenti realizzati nel *Sistema*$_G$ finale;
- **Test di qualifica del Sistema:** ovvero il *Test*$_G$ dell'intero *Sistema*$_G$ per assicurarne il corretto funzionamento;

- **Installazione del Software:** ovvero la fornitura di quanto realizzato al cliente finale nell'ambiente concordato;
- **Supporto per approvazione del Software:** ovvero l'attività per cui il fornitore dovrà supportare l'utilizzatore finale al fine di comprendere se nell'effettivo tutti i requisiti richiesti siano effettivamente soddisfatti.

2.2.2 Analisi dei Requisiti

L'*Analisi dei Requisiti*$_G$ è tra le attività cardine della *Requirements and Technology Baseline*$_G$ e ha come fine l'individuazione di tutti i requisiti che il *Sistema*$_G$ da noi sviluppato dovrà soddisfare. Tale analisi, espone nel dettaglio tutte le informazioni necessarie, che saranno poi fondamentali per supportare il lavoro dei progettisti e dei programmatori nelle rispettive attività di progettazione dell'architettura e codifica della stessa.
In particolar modo, il documento raggruppa tutti i Casi d'Uso rilevati e i requisiti ad essi associati. Per una più rapida consultazione sarà ora discussa la nomenclatura nel dettaglio.

2.2.2.1 Casi d'Uso

Per i casi d'Uso viene utilizzata la seguente nomenclatura:

UCPrincipale.Secondario

dove:

- **UC** è un acronimo stante per **U**se **C**ase, ovvero la traduzione inglese di *Caso d'Uso*$_G$
- **Principale** è un numero crescente stante ad indicare che il *Caso d'Uso*$_G$ non è correlato ad altri Casi d'Uso oppure è utilizzato da più Casi d'Uso mediante inclusione.
- **Secondario** è un numero crescente stante ad indicare che il *Caso d'Uso*$_G$ è correlato esclusivamente al *Caso d'Uso*$_G$ identificato dal valore **Principale**

Si noti che **Principale** è univoco tra tutti i Casi d'Uso, dunque non può sussistere l'esistenza di due Casi d'Uso aventi stesso valore **Principale** mentre è possibile che uno stesso valore **Secondario** sia ripetuto, ma mai per la stesso valore **Principale**.
Si noti, inoltre, che un *Caso d'Uso*$_G$ **Secondario** può avere a sua volta delle inclusioni: in tal caso la nomenclatura **Principale.Secondario** sarà la parte **Principale** di tale inclusione e seguirà le regole sopra esposte.
La nomenclatura utilizzata è volta ad assicurare l'unicità di ogni *Caso d'Uso*$_G$.
Ogni *Caso d'Uso*$_G$ è inoltre accompagnato da un nome che ne riassume lo scopo e una descrizione.

2.2.2.2 Requisiti

Identificati i Casi d'Uso, il documento si concentra sull'individuazione dei requisiti deducibili dagli stessi. I requisiti sono anch'essi identificati da una nomenclatura:

RTipologia-Numero

dove:

- **R** è per abbreviare la parola **R**equisito;
- **Tipologia** indica il tipo di requisito. I possibili valori sono:
  - **F** per **F**unzionale;
  - **Q** per **Q**ualità;
  - **V** per **V**incolo;
- **Numero** è un valore univoco che identifica il requisito;

3 Processi di supporto

3.1 Documentazione

Il processo di documentazione nasce a supporto di un qualsiasi processo primario che necessita di memorizzare ed esporre le decisioni e i dettagli riguardanti qualsiasi aspetto del progetto: architetturale, organizzativo, analitico, progettuale, distributivo e di *manutenzione*$_G$. A seconda dello scopo del documento, il target di riferimento varia dallo sviluppatore all'utente finale.

3.1.1 Strumenti a supporto

- **Redazione**: lo strumento scelto e utilizzato dal gruppo per redigere ognuno di questi documenti è *LaTeX*$_G$. Si tratta di un linguaggio di markup con un'elevata qualità tipografica, scelto per l'ottima efficienza di stesura di un documento formale. Tramite semplici comandi viene infatti assicurata l'uguaglianza sintattica e strutturale degli stessi. Garantisce anche una facile integrazione di più sezioni nel tempo grazie all'autostesura dell'indice, dunque si rivela ottimo nei documenti quale il presente, destinati ad essere soggetti a continue modifiche nel corso dello sviluppo del progetto. Per facilitare la redazione da parte di più membri, si è scelto di utilizzare una repository privata *GitHub*$_G$ apposita al solo codice *LaTeX*$_G$ sorgente del documento, chiamata "Sorgente_documents", dove sono memorizzati anche i template per i vari tipi di documenti che devono seguire una struttura precisa e ripetuta, come per esempio i *verbali*$_G$;
- **Organizzazione e distribuzione**: per la memorizzazione, organizzazione e distribuzione degli stessi è invece stato scelto di creare una *Repository*$_G$ apposita sulla piattaforma *GitHub*$_G$, chiamata "Documents". Questa è suddivisa in due cartelle, denominate "Documenti esterni" e "Documenti interni" che, come intuibile dal nome, organizzano rispettivamente i documenti esterni, dunque con target esterno al gruppo, ed interni, dunque destinati e riguardanti il gruppo stesso. È stato scelto *GitHub*$_G$ per la sua possibilità di versionare i documenti, importante quando si lavora su di essi lungo il corso del progetto, incrementando a mano a mano il numero di sezioni nei documenti. Inoltre, rendendo pubblica la repository, è possibile condividere i documenti al suo interno con un semplice link che porta alla versione dinamica del documento, e non alla versione attuale al momento di invio come succederebbe con un semplice PDF.

3.1.2 Attività previste

Il processo di documentazione prevede le seguenti attività:

- **Produzione**: si tratta dell'attività che si occupa di instaurare un processo di redazione per i vari documenti, descritta nel dettaglio nella sezione 3.1.5;
- **Manutenzione**: si tratta dell'attività che si occupa di gestire l'aggiornamento di quei documenti che ne necessitano, perché le informazioni di cui trattano sono cambiate o aumentate, descritta nel dettaglio nella sezione 3.1.6.

3.1.3 Verbali

I *verbali*$_G$ sono quei documenti nati per la necessità di ufficializzare e memorizzare le decisioni avvenute durante un meeting. A seconda se quest'ultimo sia stato interno al gruppo o esterno (dunque con la partecipazione di elementi che non fanno parte del gruppo), il *verbale*$_G$ prende il nome rispettivamente di "*Verbale*$_G$ Interno" o "*Verbale*$_G$ Esterno". Entrambi hanno una struttura simile ma diversa, infatti posseggono ognuno il proprio template nella repository privata "Sorgente_documents". La struttura comune è la seguente:

- **Frontespizio**: rappresenta la pagina iniziale del documento. Mostra il logo del gruppo seguito dalla mail tramite cui è possibile contattarci, oltre alla data in cui l'incontro è avvenuto. Più in basso sono visibili le informazioni più importanti che distinguono il documento, ovvero:
  - il luogo (fisico o virtuale) e l'orario di inizio e fine della riunione;
  - il nome del *redattore*$_G$ e del *verificatore*$_G$ del *verbale*$_G$;
  - i destinatari del documento, ovvero coloro che potranno accedere al documento, oltre al gruppo stesso;
  - se il *verbale*$_G$ è esterno, vengono specificati i membri partecipanti interni e quelli esterni;
  - il *responsabile*$_G$ del gruppo al momento del meeting.
- **Registro delle modifiche**: gestito tramite una tabella la cui struttura è riportata nella sezione 3.1.4;
- **Indice**: generato automaticamente da *LaTeX*$_G$, contiene, numerate, tutte le sezioni e sottosezioni del documento, che sono:
  - **Revisione del periodo precedente**: sezione che riporta la discussione del gruppo riguardante le attività che erano state aperte durante il precedente incontro, atto a verificare che siano state completate, e se no, le motivazioni che l'hanno reso infattibile;

  - **Ordine del giorno**: sezione che riporta le varie tematiche discusse nella riunione;
  - **Attività pianificate**: tabella che riporta le attività scaturite dalle decisioni prese durante la riunione. Presentano il nome dell'attività, il numero di riferimento della corrispondente *Issue*$_G$ su *GitHub*$_G$, ed il nome di colui che verificherà il *verbale*$_G$.

- **Approvazione esterna**: presente solo se si tratta di un *verbale*$_G$ esterno.

3.1.4 Altri documenti

Le norme generiche che tutti i documenti devono seguire sono le seguenti:

- il frontespizio deve contenere il logo del gruppo, la sua mail, i *redattori*$_G$, i *verificatori*$_G$, i destinatari e il *responsabile*$_G$;
- la tabella delle modifiche deve essere presente nel momento in cui si modifica un documento, e deve seguire la seguente struttura:
  - **Versione**: questa colonna è dedicata alla memorizzazione della versione a cui è stato portato il documento in seguito alla modifica apportata;
  - **Data**: riferita alla data di modifica, nel formato DD/MM/YYYY;
  - **Autore**: nome dell'autore delle modifiche;
  - **Verificatore**: nome del *verificatore*$_G$ delle modifiche;
  - **Dettaglio**: si tratta di una breve descrizione di che cosa si è effettivamente cambiato all'interno del documento.
- se soggetto a versionamento, sotto al nome del documento deve essere presente il numero di versione, che, come riportato nel *verbale*$_G$ del 13 Novembre 2025, dovrà seguire il formato X.Y.Z, dove:
  - il valore X deve essere incrementato al raggiungimento dei traguardi principali (*RTB*$_G$, *PB*$_G$);
  - il valore Y deve essere incrementato in caso di modifiche significative alla struttura o ai contenuti del documento;
  - il valore Z deve essere incrementato per modifiche minori o aggiornamenti marginali.
- deve essere presente l'indice, generato automaticamente da *LaTeX*$_G$;
- le pagine vanno numerate;
- le abbreviazioni sono utilizzabili, ma vanno dichiarate nel Glossario.

- il grassetto è da utilizzare quando vengono scritti nel contenuto del documento delle date, dei prezzi, il nome del gruppo o dell'azienda a cui si fa riferimento;
- l'italico è da utilizzare quando ci si riferisce a nomi specifici di componenti del progetto, o tecnologie/piattaforme utilizzate.

3.1.5 Produzione

I documenti vengono inizialmente prodotti seguendo questi passaggi:

- **Origine**: la necessità di un documento nasce da una decisione presa durante un incontro interno, e dichiarata nella sezione "Attività pianificate" del rispettivo *verbale*$_G$, assieme al *verificatore*$_G$ destinato a verificarlo;
- **Creazione della issue**: sulla repository "Documents" di *GitHub*$_G$ viene creata la *issue*$_G$ corrispondente al documento, e se redatto a parti, essa si riferirà solo al pezzo concordato. Viene *poi*$_G$ assegnata a colui che verificherà quella parte di documento;
- **Verifica**: quando si è terminata la redazione del documento (o parte di esso), il codice sorgente *LaTeX*$_G$ dello stesso viene caricato sulla repository privata "Sorgente_documents", così che sia distribuito a tutti i componenti del gruppo, in particolare al *verificatore*$_G$ assegnato. Una volta che quest'ultimo ha verificato il codice sorgente, sarà lui a doverlo scaricare in formato PDF e caricare sulla repository "Documents". Questa azione porterà all'attivazione della *GitHub*$_G$ Action incaricata di segnalare i termini presenti nel Glossario tramite una G a pedice della parola;

3.1.5.1 Denominazione e datazione dei documenti

Non differisce a seconda che si tratti del titolo del PDF o di quello del documento, ed avrà questa struttura: il titolo del documento, seguito da una "v" ed il numero di versione attuale. Oltre a queste regole, i *verbali*$_G$ devono seguire una nomenclatura più specifica, differente se si tratta del titolo del PDF o di quello del file visualizzato nella repository:

- **titolo PDF**: va scritto il tipo di *verbale*$_G$, esterno o interno, per esteso, seguito dalla data nella quale è avvenuto l'incontro, nel formato DD/MM/YYYY;
- **titolo file in repository**: è nella forma TIPO_YYYY_MM_DD, dove TIPO sta per l'abbreviazione del tipo di *verbale*$_G$, se esterno sarà VE, se interno VI, e i restanti campi sono rispettivamente l'anno, il mese e il giorno nel quale si è svolto l'incontro.

3.1.6 Manutenzione

Per quei documenti che hanno bisogno di modifiche, che possono essere o integrative o migliorative, si attiva un processo che le possa permettere, ed è il seguente:

- **Origine**: se durante un incontro si ritiene necessaria la modifica di un documento già verificato;
- **Creazione issue**: a quel punto, sempre sulla repository "Documents", viene aperta una nuova *issue*$_G$ apposita, assegnata a colui che verificherà la modifica al documento;
- **Modifica**: una volta terminata la modifica al documento, questa andrà segnata sull'apposita tabella che le registra, dove verrà anche modificato il numero di versione, secondo le regole precedentemente descritte nella sezione 3.1.4;
- **Verifica**: il codice sorgente *LaTeX*$_G$ viene dunque caricato sulla repository privata "Sorgente_documents", dove il *verificatore*$_G$ potrà svolgere il suo ruolo, in seguito al quale potrà caricare il documento modificato sulla repository "Documents" in formato PDF.

3.2 Gestione delle configurazioni

Si tratta del processo di supporto dedicato all'identificazione, attuazione e registrazione delle versioni di un qualsiasi elemento del progetto soggetto a modifiche.

3.2.1 Strumenti a supporto

- **Github**: il gruppo ha scelto questa piattaforma sia come ITS che come VCS. Tramite la creazione di una *issue*$_G$ si può infatti tener traccia delle modifiche avvenute e di quelle previste, mentre tramite il controllo di versione si può tornare ad uno stato precedente ad una qualsiasi commit (ovvero modifica approvata) nella repository.

3.2.2 Controllo di configurazione

Le attività da eseguire secondo lo standard ISO/IEC 12207:1995 sono le seguenti:

- **Identificazione della configurazione**: definisce i Configuration Items e la loro struttura all'interno della repository;
- **Controllo della configurazione**: definisce una struttura di creazione e approvazione o rifiuto delle Change Request;
- **Registrazione dello stato di configurazione**: monitora e traccia i Configuration Items;
- **Verifica della configurazione**: determina la comformità e qualità della configurazione.

3.2.3 Identificazione della configurazione

Un Configuration Item (CI) è un particolare artefatto del progetto soggetto a controllo formale, dunque a continue modifiche nel tempo, o che risulta critico ai fini del progetto stesso. Analizzando il capitolato scelto dal nostro gruppo, sono attualmente stati individuati i seguenti CI, strutturati nella repository "Documents" nella seguente struttura:

- **Documenti Esterni**:
  - *Analisi dei Requisiti*$_G$;
  - *Piano di Progetto*$_G$;
  - Preventivo costi e assunzione impegni;
- **Documenti Interni**:
  - Glossario;
  - *Norme di Progetto*$_G$.

è stata omessa la documentazione non identificata come CI.

3.2.4 Controllo della configurazione

Una Change Request è una richiesta formale di modifica di un qualsiasi CI. Può essere proposta da un qualsiasi componente del gruppo, che stia ricoprendo un qualsiasi ruolo, ogni qualvolta che lo ritiene opportuno. Questa Change Request viene analizzata e discussa da tutto il gruppo, e può essere rifiutata o approvata. Se approvata, sulla repository di Github viene aperta una *issue*$_G$, che presenta le seguenti caratteristiche:

- **Nome**: la targhetta della *issue*$_G$, descrive brevemente lo scopo che si vuole raggiungere completandola;
- **Numero**: numero univoco che identifica la *issue*$_G$, preceduto da un "#". Se è N, significa che quella è stata l'n-esima *issue*$_G$ aperta su quella repository;
- **Label**: etichette che rappresentano le sfere semantiche di appartenenza della modifica a cui è associata la *issue*$_G$. Ad esempio, se si tratta di aggiungere una sezione alla documentazione, sicuramente il label della *issue*$_G$ associata sarà "Documentazione";
- **Milestone**: una *milestone*$_G$ è un obiettivo significativo del progetto associato ad una data entro il quale si vuole raggiungere. Una *issue*$_G$ può essere associata ad una particolare *milestone*$_G$ per indicare che è parte dei progressi da eseguire per raggiungere quell'obiettivo;

- **Verificatore**: colui a cui è stata assegnata quella *issue*$_G$. Sarà il *verificatore*$_G$ della modifica richiesta.

Le modifiche implementate verranno *poi*$_G$ approvate dal *verificatore*$_G$ reperibile al *verbale*$_G$ dell'incontro in cui si è approvata la Change Request, ed infine validate dall'attuale *responsabile*$_G$. Per ufficializzare il tutto, si procede con un commit alla repository dedicata. Un commit è il caricamento di un qualsiasi numero di file, modificati in locale, in una repository, e presenta le seguenti proprietà:

- **Descrittiva**: ad ogni commit si può associare un messaggio che ne spiega l'importanza e i cambiamenti effettuati. Il gruppo ha anche scelto che sia il modo per chiudere una *issue*$_G$, dunque considerarla completata: scrivendo nel messaggio della commit "close #n", si può infatti chiudere automaticamente l'n-esima *issue*$_G$.
- **Reversibilità**: si può sempre tornare ad uno stato della repository precedente ad un commit, essenziale per tener traccia dei cambiamenti e delle varie versioni di ogni CI.

3.2.5 Registrazione dello stato di configurazione

Per tener traccia di tutte le modifiche apportate ad un determinato documento si è deciso di usare la tabella delle modifiche, come descritto nella sezione 3.1.4. Per tenere traccia della versione del documento, invece, è stata utilizzato lo standard Semantic Versioning, dunque nel seguente formato: **MAJOR.MINOR.PATCH**, strutturato nel seguente modo:

- **MAJOR**: una modifica incompatibile con lo stato precedente, o che porta il documento in uno stato importante;
- **MINOR**: una modifica retrocompatibile;
- **PATCH**: una modifica o correzione minima retrocompatibile.

La versione di un documento è reperibile nel nome del PDF.

3.2.6 Verifica della configurazione

Per determinare se il prodotto è conforme ai requisiti, vengono innanzitutto tracciati i requisiti individuati nel capitolato scelto, così che si possa determinare se il requisito implementato rispetti le specifiche tracciate. Il prodotto si dice sufficiente se almeno rispetta tutti i requisiti obbligatori individuati. Per controllare sia conforme alle norme, si *verifica*$_G$ che ogni CI rispetti le regole descritte in questo documento.

3.3 Processi di Qualifica

Sono tutti quei processi atti a fornire evidenze oggettive sulla qualità del software che si sta producendo, domostrandone la conformità agli standard adottati. Le attività previste da questo processo sono il processo di *Verifica*$_G$ e quello di *Validazione*$_G$, descritti in seguito.

3.3.1 Verifica

Fornisce evidenze oggettive che gli output di un particolare segmento del software incontri tutti i requisiti specifici dello stesso, controllandone consistenza, completezza, e correttezza. Vengono descritti e tracciati nel *Piano di Qualifica*$_G$.

3.3.1.1 Attività previste

- **Implementazione del processo**: si definisce che cosa si intende verificare, come, e con quali strumenti, se risultano essere necessari;
- **Verifica**: è l'attuazione della *verifica*$_G$ stessa sui diversi segmenti individuati nell'attività precendente. Comprende:
  - **Analisi Statica**: studia documentazione e codice, accertandone conformità, assenza di difetti e presenza di proprietà attese;
  - **Analisi Dinamica**: un'insieme di test applicati al codice del software.
- **Valutazione dei risultati**: i dati originati dalle attività precedenti vengono raccolti sotto forma di risultato così che possano essere esposti, confrontati e analizzati. Indispensabile per l'individuazione di problematiche o non conformità.

3.3.1.2 Implementazione del processo

Il gruppo ha ritenuto opportuno passare per il processo di *verifica*$_G$ ogni elemento del progetto destinato alla repository Documents. Dunque ad ogni documento viene controllata la correttezza ortografica (automaticamente), sintattica e di contenuto (ruolo del *verificatore*$_G$), mentre per il codice saranno sviluppati test automatici che dovranno produrre risultati positivi. Verranno discussi più nel dettaglio durante lo sviluppo dell'*RTB*$_G$ vista la leggera presenza di codice fino a quel momento.

3.3.1.3 Verifica

- **Documentazione**: come descritto nella sezione 3.2, ad ogni modifica viene assegnato un *Verificatore*$_G$, il cui ruolo sarà quello di controllare la correttezza ortografica, sintattica e di contenuto del nuovo pezzo di documento redatto. Per velocizzare il processo si è

automatizzato il controllo della correttezza ortografica e quello sulla leggibilità del testo tramite l'indice di Gulpease;

- **Codice**: per quanto riguarda il codice, il gruppo affronterà l'argomento nella prossima fase del progetto, dove effettivamente verrà sviluppato il software.

Questa attività può essere suddivisa in:

- **Analisi Statica**;
- **Analisi Dinamica**.

di seguito descritte.

3.3.1.4 Analisi Statica

Forma di *Verifica*$_G$ che non ha bisogno di eseguire dell'oggetto di *verifica*$_G$. Questa sua caratteristica la rende applicabile tempestivamente, in qualsiasi momento di produzione del codice, così da individuare gli errori sintattici, a volte anche difficili da individuare durante l'esecuzione, o da prevenirne altri. Rende il codice più leggibile.
Può essere applicata non solo al codice, ma anche alla documentazione, concentrandosi sempre sull'aspetto formale e sintattico.
Può essere eseguita tramite metodi di lettura, o metodi formali, ovvero realizzati con tecniche matematiche. I metodi di lettura sono metodi che si basano sulla lettura umana o automatizzata dell'oggetto di *verifica*$_G$, dunque può dipendere anche dall'esperienza del *verificatore*$_G$. Sono differenziati in:

- **Walkthrough**: usato quando non si conosce l'area dove l'errore possa verificarsi, dunque si controlla tutto, leggendo come leggerebbe un compilatore per individuare l'errore;
- **Inspection**: si ha una supposizione su dove l'errore si possa essere verificato, dunque si definiscono le aree di controllo e si analizzano solo quelle. Per questo, è maggiormente automatizzabile.

3.3.1.5 Analisi Dinamica

Come strumento utilizza i test, che devono essere:

- **ripetibili**: per poter verificare che, dopo che il test è fallito a causa di un errore, l'errore sia stato corretto, è essenziale poter ripetere lo stesso test affinchè questo possa andare a buon fine;
- **automatizzati**: per poter essere eseguiti molte volte senza perdere troppo tempo. Si ottiene mediante strumenti che simulano l'ambiente del test.

La nomenclatura utilizzata è la seguente:

T-#-t

dove T sta per *Test*$_G$, # è il numero identificativo dello stesso, t è il tipo di test, uno tra i seguenti:

- **Test di Integrazione** (I);
- **Test di Unità** (U);
- **Test di Accettazione** (A);
- **Test di Sistema** (S).

Lo stato del singolo test viene invece indicato in questa maniera:

- **NI**: Non Implementato;
- **S**: Superato;
- **NS**: Non Superato.

3.3.1.6 Test di Unità

Test$_G$ che analizzano la parte più piccola del Software, ovvero l'Unità definita per quel linguaggio di programmazione, a responsabilità singola. Si differenziano ulteriormente in:

- **Funzionali**: test di tipo Black-box, ovvero che non si riferiscono all'implementazione del test, ma solo alla correttezza dell'imput e dell'output;
- **Strutturali**: test di tipo White-box, controllando il cammino di esecuzione.

3.3.1.7 Test di Integrazione

Test$_G$ che controllano come le varie parti del sowfware comunichino tra loro. Utili per l'assemblaggio incrementale: si effettuano i test incrementali ogni volta che si vuole far comunicare una parte di *sistema*$_G$ con un'altra. In questo modo, si ottiene la *Continuous Integration*. Per questo devono essere reversibili, in caso la comunicazione non funziona bisogna poter tornare ad uno stato funzionante.
Vi sono due strategie di integrazione:

- **Bottom-up**: si sviluppano e si integrano prima le componenti con minori dipendenze d'uso e maggiore utilità interna;
- **Top-down**: si sviluppano e si integrano prima le componenti con maggiori dipendenze d'uso e quindi maggiore valore aggiunto esterno.

3.3.2 Validazione

Controlla che il prodotto risponda davvero ai bisogni dell'utente o del cliente

3.3.2.1 Attività previste

Le attività previste da questo processo sono:

- **Implementazione del processo**;
- **Esecuzione dei test**;
- **Raccolta dei risultati**.

3.3.2.2 Implementazione del processo

Tramite il **Tracciamento dei requisiti** che il gruppo ha svolto, individuando e raggruppando i requisiti tracciati nell'*Analisi dei Requisiti*$_G$, è possibile controllare quanto e come il prodotto soddisfi questi requisiti. I test utilizzati per questo sono:

- **Test di Sistema**: test di tipo black-box, Controlla come l'intero *sistema*$_G$ sia concorde con i requisiti concordati inizialmente;
- **Test di Accettazione**: Accerta il soddisfacimento dei requisiti utente.

3.3.2.3 Esecuzione dei test

I test vengono implementati, in base ad uno specifico requisito o scenario, sempre dal punto di vista dell'utente, ed eseguiti, in maniera automatizzata.

3.3.2.4 Raccolta dei risultati

I risultati di questi test vengono *poi*$_G$ raccolti nel *Piano di Qualifica*$_G$, dove in un'apposita sezione verranno anche analizzati e confrontati per scoprire l'andamento del *sistema*$_G$ in base ad uno o più requisiti.

4 Processi Organizzativi

I *Processi Organizzativi* sono dei processi a supporto del progetto che assicurano il buon andamento dell'intero progetto. Le attività previste assicurano la buona esecuzione di tutti i processi adottati e l'adozione di eventuali miglioramenti, la gestione delle infrastrutture utilizzate e la formazione del team nei compiti da seguire.

4.1 Gestione dei Processi

La *Gestione dei Processi* ha l'obiettivo di individuare i compiti da svolgere e i ruoli ai quali questi saranno assegnati, nonché permettere una comunicazione interna ed esterna efficace e altresì garantire lo svolgimento delle varie attività in maniera efficace mediante un'opportuna pianificazione.

4.1.1 Attività previste

Le attività principali, osservate da **GammardX**, previste da questo processo sono:

- **Inizializzazione**: è necessario stabilire i requisiti delle attività da svolgere, cercando di comprendere quali risorse richiedano. Esse vengono discusse e determinate durante le riunioni interne.
- **Pianificazione**: è necessario, stabiliti i requisiti, comprendere il tempo necessario per completare le attività. Per facilitare questo compito, è stato scelto di dividere le responsabilità in vari ruoli, vedi sezione 4.1.2.
- **Esecuzione e controllo**: l'esecuzione delle attività è affidata quindi ai vari ruoli e il *responsabile*$_G$ dovrà costantemente monitorare lo stato di progresso e avanzamento complessivo.
- **Revisione e valutazione**: una volta completata l'attività, è necessario verificarne la correttezza: tale *verifica*$_G$ è svolta dal *verificatore*$_G$. Per un approfondimento sulle attività di revisione si rimanda alla Sezione 3.1.6 per la documentazione.
- **Finalizzazione**: un'attività è ritenuta conclusa solo nel momento in cui viene approvata. Questa operazione viene eseguita al pari di una normale attività di *verifica*$_G$, con l'unica differenza che essa comporta anche la chiusura della *issue*$_G$ associata.

Di seguito vengono descritti i ruoli e le rispettive responsabilità, definite soprattutto nell'ambito della pianificazione. A seguire, viene presentata una parte fondamentale per il gruppo: i metodi di coordinamento.

4.1.2 Ruoli

Ruolo	Descrizione
Responsabile$_G$	Al *responsabile*$_G$ compete valutare quanto è già stato realizzato e pianificare ciò che resta da sviluppare nei periodi successivi, identificando attività, costi e rischi, e assegnando ogni compito ai membri del team che, in quel momento, ricoprono il ruolo più adeguato. Il *responsabile*$_G$ è anche colui che ha il compito, a nome di tutto il gruppo, di dialogare con le parti esterne a **GammardX**, quali, a mero titolo esemplificativo, l'azienda proponente del capitolato. In ultima istanza, è sempre compito di questo ruolo approvare i vari documenti prodotti da **GammardX**.
Amministratore$_G$	L'*amministratore*$_G$ ha l'importante compito di gestire e migliorare le infrastrutture a supporto del progetto. Ha inoltre la responsabilità di risolvere il prima possibile eventuali problematiche legate ad esse.
Analista$_G$	Questo ruolo è *responsabile*$_G$ dell'individuazione dei requisiti obbligatori, desiderabili e opzionali del progetto, considerando quanto discusso nelle riunioni esterne con la proponente.
Progettista$_G$	Il *progettista*$_G$ ha il compito di trasformare i requisiti in design architetturale, definendo le scelte tecnologiche.
Programmatore$_G$	Il *programmatore*$_G$ è *responsabile*$_G$ dello sviluppo del software, trasformando il design architetturale in codice funzionante. Lavora insieme al *progettista*$_G$ in modo da assicurare che tutte le funzionalità vengano implementante correttamente. Il suo lavoro richiede una buona conoscenza delle tecnologie utilizzate. É suo compito realizzare i test automatici necessari a verificare il corretto funzionamento del software.
Verificatore$_G$	Ha il compito di garantire che ogni prodotto, dalla documentazione fino alla più piccola attività, sia realizzato a regola d'arte. È sempre di questo ruolo la responsabilità di verificare la correttezza dei vari documenti ad ogni modifica effettuata.

4.1.3 Coordinamento

Una componente fondamentale del progetto è rappresentata dalla capacità di coordinamento del gruppo, sia al suo interno che verso l'esterno. Per garantire un coordinamento efficace è necessario programmare e svolgere riunioni dedicate, oltre a

disporre di canali di comunicazione affidabili e adeguati alle esigenze del progetto.

4.1.3.1 Riunioni

Le riunioni sono di due tipi: interne ed esterne.
GammardX realizza periodicamente, una volta a settimana, riunioni interne per discutere dell'avanzamento del gruppo e per allinearsi sulle attività svolte, quelle ancora da completare e possibili difficoltà riscontrate. Ogni due settimane, ad ogni fine sprint, si discutono e approvano le modifiche fatte ai vari documenti durante lo sprint e si effettua la rotazione dei ruoli.
Durante lo svolgimento del progetto si avranno riunioni con **Zucchetti**: riunioni esterne per discutere l'avanzamento del progetto con la proponente e di chiarire eventuali dubbi sorti. Contrariamente alle riunioni interne non è prevista periodicità ma verranno effettuate ogni qual volta ne sia ritenuto necessario.
L'esito degli incontri dovrà sempre essere documentato mediante la redazione di appositi *verbali*$_G$, rispettivamente interni ed esterni.

4.1.3.2 Comunicazioni

In merito ai metodi comunicativi, **GammardX** utilizza WhatsApp e *Discord*$_G$ per, rispettivamente, comunicazioni asincrone e sincrone (riunioni). In genere una comunicazione urgente ma che non richiede approfondita discussione o per comunicazioni di servizio minori i componenti utilizzeranno l'apposito gruppo realizzato su WhatsApp o messaggi diretti ad un componente specifico sempre utilizzando la medesima piattaforma, mentre la realizzazione di riunioni interne e la discussione di criticità complesse richiederanno lo svolgimento di una riunione presso il server *Discord*$_G$ appositamente realizzato. Per quanto riguarda invece le comunicazioni esterne, queste verranno sempre realizzate dal *responsabile*$_G$ mediante l'utilizzo dell'indirizzo di posta elettronica di **GammardX**: *gammardx@gmail.com*

4.2 Infrastruttura

Il processo di *infrastruttura* è *responsabile* della creazione e del mantenimento dei componenti necessari per permettere tutti gli altri processi.

4.2.1 Attività previste

Si compone delle seguenti attività

- Implementazione
- Creazione
- Mantenimento

4.2.1.1 Implementazione

GammardX ha utilizzato, durante il progetto, vari strumenti per permettere il lavoro asincrono ai membri del gruppo.

Strumento	Descrizione
Discord$_G$	Programma di messaggistica usata da **GammardX** per svolgere le riunioni interne virtuali.
WhatsApp	App usata dai membri del gruppo per comunicare in maniera veloce
Google Mail	Il servizio di posta elettronica utilizzato da **GammardX** per tutte le comunicazioni verso l'esterno
Google Drive	Servizio di archiviazione cloud utilizzato da **GammardX** per condividere file e poterci collaborare in tempo reale.
Google Sheets	Servizio di Google che permette di creare fogli di calcolo e tabelle. Viene utilizzato da **GammardX** per la rendicontazione ore.
GitHub$_G$	*GitHub*$_G$ è un prodotto che permette principalmente la memorizzazione su dispositivo remoto di repository Git.
LaTeX$_G$	*LaTeX*$_G$ viene utilizzato da **GammardX** per la redazione di tutti i documenti

4.2.2 Creazione

4.2.2.1 Discord

Per utilizzare *Discord*$_G$ è stato creato un server dedicato con un canale vocale per le riunioni interne virtuali e vari canali testuali per una migliore organizzazione.

4.2.2.2 WhatsApp

Per utilizzare WhatsApp è stato creato un gruppo per permettere ai membri del gruppo di comunicare in modo veloce e diretto.

4.2.2.3 Google Mail

La casella di poste di Google Mail non ha richiesto particolari configurazioni.

4.2.2.4 Google Drive

La cartella drive su Google Drive non ha richiesto particolari configurazioni.

4.2.2.5 Google Sheets

È stato creato un foglio di calcolo che calcola in modo automatico le ore ancora disponibili per ogni ruolo e il budget rimanente associato.

4.2.2.6 GitHub

All'interno del repository *GitHub*$_G$ dedicato alla documentazione è possibile trovare varie directories:

- Documenti esterni, contenente i documenti ad utilizzo esterno
- Documenti interni, contenente i documenti ad utilizzo interno
- Website, contenente assets e codice riguardante il sito: `https://gammardx.github.io/Documents/`

4.2.2.7 Latex

Per i vari tipi di documenti sono stati redatti vari template per facilitarne la redazione.

4.3 Processo di Miglioramento

Il Processo di Miglioramento, in accordo con lo standard ISO/IEC 12207:1995, è finalizzato all'analisi e all'ottimizzazione dei processi utilizzati durante tutto il ciclo di vita del software, garantendo un progressivo incremento della qualità del prodotto e dell'efficienza del gruppo di lavoro.

4.3.1 Attività previste

Il processo si articola in:

- **Analisi**: Identificare le aree di miglioramento dei processi;
- **Miglioramento**: Applicare le modifiche necessarie per migliorare i processi di sviluppo del software.

4.3.2 Analisi

Questa attività deve essere svolta con regolarità, secondo intervalli di tempo appropriati e costanti. L'analisi consente di valutare l'effettiva efficacia e correttezza dei processi implementati, permettendo di individuare tempestivamente quelli che necessitano di miglioramenti. Durante ogni riunione di fine sprint, il team dedica inizialmente del tempo a una retrospettiva sulle attività svolte nel periodo precedente. Tale pratica prevede una

riflessione approfondita sul lavoro svolto, coinvolgendo tutti i membri nell'individuazione dei punti di forza e delle aree di possibile miglioramento. L'obiettivo principale è definire azioni correttive da applicare nello sprint successivo, favorendo un feedback continuo e un adattamento progressivo volto a migliorare nel tempo le prestazioni complessive del team.

4.3.3 Miglioramento

Il team implementa le azioni correttive definite durante la retrospettiva e ne valuta successivamente l'efficacia, sottoponendole a una nuova analisi nel corso della retrospettiva successiva.

4.4 Processo di Formazione

Il processo di formazione nasce dall'esigenza di allineare le competenze tecniche di tutti i membri del gruppo GammardX.

4.4.1 Attività previste

Il Processo di Formazione definisce e disciplina le seguenti attività:

- **Analisi delle competenze**: valutazione delle conoscenze e delle capacità del personale in relazione ai ruoli assegnati e ai requisiti del progetto, al fine di individuare eventuali lacune formative;
- **Formazione individuale**: svolgimento di attività di studio e autoformazione finalizzate all'acquisizione delle competenze necessarie, integrate nella pianificazione temporale del progetto.

4.4.2 Analisi delle competenze

In assenza di vincoli implementativi mandatori definiti inizialmente dal proponente **Zucchetti**, il gruppo di sviluppo ha condotto un'attività di analisi per identificare lo stack tecnologico più idoneo al soddisfacimento dei requisiti di *sistema*$_G$.
L'efficacia delle attività formative viene periodicamente verificata attraverso l'osservazione delle prestazioni del team, la riduzione delle criticità tecniche emerse durante lo sviluppo e la capacità dei membri di svolgere in autonomia le attività assegnate.

5 Metriche e standard per la Qualità

GammardX ha deciso di seguire lo *standard ISO/IEC 9126* per definire le metriche e i parametri che determinano la qualità di quanto realizzato. Lo standard identifica sei caratteristiche generali:

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

5.1 Funzionalità

In merito alla **Funzionalità**, lo scopo è misurare quanto un prodotto soddisfa le esigenze del proponente. Nello specifico misura:

- **Adeguatezza**, ovvero se il prodotto offre le funzioni necessarie a svolgere i compiti prefissati dal proponente;
- **Accuratezza**, ovvero se il prodotto fornisce i precisi effetti richiesti;
- **Interoperabilità**, ovvero se il prodotto riesce ad interagire con altri *Sistemi*$_G$ definiti;
- **Conformità**, ovvero se il prodotto aderisce a determinati standard richiesti dal dominio d'uso del prodotto;
- **Sicurezza**, ovvero se il prodotto mette in sicurezza i dati degli Utenti, impedendone il reperimento ad individui non autorizzati.

5.2 Affidabilità

In merito all'**Affidabilità**, è la capacità del prodotto di mantenere un determinato livello di prestazioni in qualsiasi momento, anche in periodi d'uso particolarmente intensi. Nello specifico misura:

- **Maturità**, ovvero l'arginare la possibilità che si verifichino errori o malfunzionamenti;

- **Tolleranza agli errori**, ovvero l'obiettivo di mantenere la prestazionalità del prodotto anche in caso di malfunzionamenti;
- **Recuperabilità**, ovvero la capacità del prodotto di ritornare al normale comportamento prestazionale a seguito di un malfunzionamento;
- **Aderenza**, ovvero la capacità di aderire a standard di affidabilità.

5.3 Efficienza

L'**Efficienza** misura la capacità di fornire delle prestazioni che siano direttamente proporzionate alle risorse utilizzate. Nello specifico misura:

- **Comportamento temporale**, ovvero la capacità di fornire una risposta in tempi adeguati;
- **Utilizzo delle risorse**, ovvero un uso proporzionale delle risorse in rapporto all'utilizzo;
- **Conformità**, ovvero l'aderenza del prodotto a standard riguardanti l'efficienza.

5.4 Usabilità

L'**Usabilità** consiste nel misurare quanto l'Utente finale è in grado di apprendere la modalità di utilizzo del prodotto. Si misura:

- **Comprensibilità**, ovvero quanto è facile comprendere i concetti del prodotto;
- **Apprendibilità**, ovvero quanto è semplice apprendere l'uso delle funzionalità del prodotto;
- **Operabilità**, ovvero se è semplice per gli Utenti utilizzare il prodotto;
- **Attrattività**, ovvero se l'Utente trova positivo l'uso del prodotto;
- **Conformità**, ovvero se il prodotto aderisce a standard relativi all'usabilità.

5.5 Manutenibilità

La **Manutenibilità** misura quanto il prodotto è semplice da modificare per permetterne l'evoluzione o la correzione. Nello specifico misura:

- **Analizzabilità**, ovvero la facilità d'ispezione del codice con lo scopo di cercare un errore;
- **Modificabilità**, ovvero quanto è difficile apportare una modifica;

- **Stabilità**, ovvero se il prodotto è in grado di arginare effetti indesiderati determinati da modifiche non corrette;
- **Testabilità**, ovvero la semplicità con cui il prodotto può essere testato.

5.6 Portabilità

La **Portabilità** è la facilità con cui il prodotto può essere spostato da un ambiente di esecuzione ad un altro. Specificatamente si misura:

- **Adattabilità**, ovvero con che facilità il prodotto può essere adattato ad un diverso ambiente d'esecuzione;
- **Installabilità**, ovvero con che facilità il prodotto può essere installato su un altro ambiente d'esecuzione;
- **Conformità**, ovvero se il prodotto aderisce a convenzioni sulla portabilità;
- **Sostituibilità**, ovvero quanto è semplice sostituire un prodotto esistente con il prodotto da noi sviluppato.

5.7 Nomenclatura delle Metriche

La nomenclatura utilizzata per le metriche è la seguente:

MType##

dove:

- **M** = **M**etrica
- **Type** può assumere uno tra questi valori:
  - **PC** = **P**rocesso
  - **PD** = **P**rodotto
- **##** è un numero crescente da 00. Il conteggio per il tipo **PC** e **PD** è separato.

Per tutte le definizioni, acronimi e abbreviazioni utilizzati in questo documento, si faccia riferimento al **Glossario**, fornito come documento separato, che contiene tutte le spiegazioni necessarie per garantire una comprensione uniforme dei termini tecnici e dei concetti rilevanti per il progetto.

6 Metriche di Qualità del Processo

6.1 Processi primari

6.1.1 Fornitura

6.1.1.1 Earned Value (EV)

- **Codice**: MPC01
- **Formula**:
  $Earned\ Value = Budget\ at\ Completion * Percentuale\ di\ lavoro\ completato\ nello\ sprint$
- **Descrizione**: L'indicatore *Earned Value*$_G$ rappresenta il valore del lavoro *completato* rispetto al budget totale previsto.
  L'indicatore è utile per monitorare l'andamento del progetto e valutare se il lavoro svolto è in linea con le aspettative.

6.1.1.2 Planned Value (PV)

- **Codice**: MPC02
- **Formula**:
  $Planned\ Value = Budget\ at\ Completion * Percentuale\ di\ lavoro\ pianificato\ nello\ sprint$
- **Descrizione**: L'indicatore *Planned Value*$_G$ rappresenta il valore del lavoro *pianificato* rispetto al budget totale previsto.
  L'indicatore è utile per monitorare l'andamento del progetto e valutare se la pianificazione è rispettata. Il valore pianificato non può essere negativo e deve essere inferiore al *BAC*$_G$.

6.1.1.3 Actual Cost (AC)

- **Codice**: MPC03
- **Formula**: $Actual\ Cost = Costo\ effettivo\ sostenuto\ nello\ sprint$
- **Descrizione**: L'indicatore *Actual Cost*$_G$ rappresenta il costo effettivo sostenuto per completare il lavoro nello sprint.
  L'indicatore è utile per monitorare l'andamento del progetto e valutare se i costi sono in linea con le aspettative.

6.1.1.4 Cost Performance Index (CPI)

- **Codice**: MPC04
- **Formula**: $Cost\ Performance\ Index = \frac{\text{Earned Value}}{\text{Actual Cost}}$
- **Descrizione**: Il *Cost Performance Index*$_G$ rappresenta il rapporto tra il valore del lavoro completato e il costo effettivo sostenuto.
  Un valore maggiore di 1 indica che il progetto sta rispettando il budget, un valore minore di 1 indica che il progetto sta superando il budget.

6.1.1.5 Schedule Performance Index (SPI)

- **Codice**: MPC05
- **Formula**: $Schedule\ Performance\ Index = \frac{\text{Earned Value}}{\text{Planned Value}}$
- **Descrizione**: Lo *Schedule Performance Index*$_G$ rappresenta il rapporto tra il valore del lavoro completato e il valore del lavoro pianificato.
  Un valore maggiore di 1 indica che il progetto sta rispettando la pianificazione, un valore minore di 1 indica che il progetto sta accumulando ritardi.

6.1.1.6 Estimate At Completion (EAC)

- **Codice**: MPC06
- **Formula**: $Estimate\ At\ Completion = \frac{\text{Budget at Completion}}{\text{Cost Performance Index}}$
- **Descrizione**: La metrica Estimate At Completion rappresenta una proiezione del costo finale totale del progetto basata sulla performance attuale. Utilizza il CPI_G come indicatore di efficienza per correggere la stima iniziale (BAC_G). Se $CPI_G < 1$, EAC sarà maggiore del BAC_G, indicando un probabile sforamento del budget.

6.1.1.7 Estimate To Complete (ETC)

- **Codice**: MPC07
- **Formula**: $Estimate\ To\ Complete = Estimate\ At\ Completion - Actual\ Cost$
- **Descrizione**: La metrica Estimate To Complete stima quanto costerà completare il lavoro rimanente del progetto. Si calcola sottraendo i costi già sostenuti (AC_G) dalla stima del costo finale totale (EAC). Utile per la pianificazione del budget residuo necessario.

6.1.1.8 Time Estimate At Completion (TEAC)

- **Codice**: MPC08

- **Formula**: $Time\ Estimate\ At\ Completion = \frac{\text{Durata pianificata}}{\text{Schedule Performance Index}}$
- **Descrizione**: La metrica Time Estimate At Completion proietta la durata finale del progetto basandosi sulla performance temporale attuale. Utilizza SPI_G come indicatore di efficienza temporale per correggere la stima iniziale. Se $SPI_G < 1$, TEAC sarà maggiore della durata pianificata, indicando un probabile ritardo.

6.1.2 Sviluppo

6.1.2.1 Requirements Stability Index

- **Codice**: MPC09
- **Formula**: $Requirements\ Stability\ Index = \left(\frac{NINIZ-(NAGG+NCAM+NCAN)}{NINIZ}\right) * 100$
- **Descrizione**:
  - **NINIZ**: Numero **I**niziali di Requisiti
  - **NCAM**: Numero **C**ambiamenti di Requisiti
  - **NCAN**: Numero **C**ancellati di Requisiti
  - **NAGG**: Numero **A**ggiunti di Requisiti

  Permette di misurare il numero di cambiamenti apportati ai requisiti nel corso del tempo.

6.2 Processi di supporto

6.2.1 Documentazione

6.2.1.1 Indice di Gulpease

- **Codice**: MPC10
- **Formula**: $Indice\ Gulpease = 89 - \frac{\text{numero di lettere}}{\text{numero di parole}} * 100 + \frac{\text{numero di frasi}}{\text{numero di parole}} * 300$
- **Descrizione**: L'Indice Gulpease è un indice di leggibilità del testo. È utile per capire quanto un testo sia facile o difficile da leggere per un lettore medio. La formula tiene conto del numero di lettere, parole e frasi nel testo.

  Intervalli e interpretazioni

  - Indice $\geq$ 80: Il testo è molto facile da leggere, comprensibile per chi ha completato solo la scuola elementare.
  - Indice compreso 60 e 80: Il testo è di media difficoltà, adatto a chi ha completato la scuola dell'obbligo.

- Indice compreso 40 e 60: Il testo è abbastanza difficile, comprensibile per chi ha almeno un'istruzione di livello superiore.
- Indice < 40: Il testo è molto difficile da leggere, comprensibile per lettori con un'istruzione universitaria.

6.2.1.2 Correttezza ortografica

- **Codice**: MPC11
- **Formula**: $Correttezza\ ortografica = numero\ di\ errori\ ortografici$
- **Descrizione**: La correttezza ortografica è un indicatore della qualità della documentazione. I documenti devono essere privi di errori ortografici.

6.2.2 Verifica

6.2.2.1 Code Coverage

- **Codice**: MPC12
- **Formula**: $Code\ Coverage = \left(\frac{\text{Linee di codice testate}}{\text{Linee di codice totali}}\right) * 100$
- **Descrizione**: Percentuale di codice coperto da *Test*$_G$ automatizzati. Si raccomanda un coverage minimo del 80%.

6.2.2.2 Test Success Rate

- **Codice**: MPC13
- **Formula**: *Test*$_G$ $Success\ Rate = \left(\frac{\text{Test passati}}{\text{Test totali}}\right) * 100$
- **Descrizione**: Percentuale di *Test*$_G$ automatizzati che passano con successo. Dovrebbe mantenersi al 100% data la natura del progetto.

6.2.3 Gestione della Qualità

6.2.3.1 Quality metrics satisfied

- **Codice**: MPC14
- **Formula**: $Quality\ metrics\ satisfied = \left(\frac{\text{Numero metriche soddisfatte}}{\text{Numero metriche totali}}\right) * 100$
- **Descrizione**: Percentuale di soddisfacimento delle metriche.

6.3 Processi organizzativi

6.3.1 Gestione dei Processi

6.3.1.1 Time Efficiency

- **Codice**: MPC15
- **Formula**: $Time\ Efficiency = \left(\frac{\text{Ore produttive}}{\text{Ore totali}}\right) * 100$
- **Descrizione**: Valutazione del rapporto tra le ore utilizzate e quelle effettivamente produttive.

7 Metriche di Qualità del Prodotto

7.1 Funzionalità

7.1.1 Requisiti obbligatori soddisfatti

- **Codice**: MPD01
- **Formula**: $Requisiti\ obbligatori\ soddisfatti = \frac{\text{Numero di requisiti obbligatori soddisfatti}}{\text{Numero di requisiti obbligatori totali}} * 100$
- **Descrizione**: L'indicatore Requisiti obbligatori soddisfatti rappresenta la percentuale di requisiti obbligatori soddisfatti rispetto al totale dei requisiti obbligatori.
  L'indicatore è utile per monitorare il grado di soddisfacimento dei requisiti essenziali del progetto.

7.1.2 Requisiti desiderabili soddisfatti

- **Codice**: MPD02
- **Formula**: $Requisiti\ desiderabili\ soddisfatti = \frac{\text{Numero di requisiti desiderabili soddisfatti}}{\text{Numero di requisiti desiderabili totali}} * 100$
- **Descrizione**: L'indicatore Requisiti desiderabili soddisfatti rappresenta la percentuale di requisiti desiderabili soddisfatti rispetto al totale dei requisiti desiderabili.
  L'indicatore è utile per monitorare il grado di soddisfacimento dei requisiti opzionali del progetto.

7.1.3 Requisiti opzionali soddisfatti

- **Codice**: MPD03
- **Formula**: $Requisiti\ opzionali\ soddisfatti = \frac{\text{Numero di requisiti opzionali soddisfatti}}{\text{Numero di requisiti opzionali totali}} * 100$
- **Descrizione**: L'indicatore Requisiti opzionali soddisfatti rappresenta la percentuale di requisiti opzionali soddisfatti rispetto al totale dei requisiti opzionali.
  L'indicatore è utile per monitorare il grado di soddisfacimento dei requisiti aggiuntivi del progetto.

7.2 Affidabilità

7.2.1 Branch Coverage

- **Codice**: MPD04

- **Formula**: $Branch\ Coverage = \left(\frac{\text{Rami testati}}{\text{Rami totali}}\right) * 100$
- **Descrizione**: Percentuale di rami del codice coperti da *Test*$_G$ automatizzati. Si raccomanda un coverage minimo del 60%.

7.2.2 Statement Coverage

- **Codice**: MPD05
- **Formula**: $Statement\ Coverage = \left(\frac{\text{Istruzioni testate}}{\text{Istruzioni totali}}\right) * 100$
- **Descrizione**: Percentuale di istruzioni del codice coperte da *Test*$_G$ automatizzati. Si raccomanda un coverage minimo del 70%.

7.2.3 Failure Density

- **Codice**: MPD06
- **Formula**: $Failure\ Density = \left(\frac{\text{Numero di difetti rilevati}}{\text{KLOC}}\right)$
- **Descrizione**: Numero di failure per 1000 linee di codice (**KLOC**, *Kilo Lines Of Code*). Un valore superiore a 0.5 indica possibili problemi di affidabilità.

7.3 Usabilità

7.3.1 Click on Task

- **Codice**: MPD07
- **Formula**: $Click\ on\ Task = Numero\ di\ click\ medio\ per\ completare\ un'attivita'$
- **Descrizione**: Numero di click medio per completare un'attività. Si considera accettabile un valore pari o inferiore a 6, mentre l'obiettivo ottimale è un valore pari o inferiore a 4.

7.3.2 Error Rate

- **Codice**: MPD08
- **Formula**: $Error\ Rate = \left(\frac{\text{Errori totali}}{\text{Azioni totali}}\right) * 100$
- **Descrizione**: Percentuale di errori commessi durante l'utilizzo del prodotto. Dovrebbe essere inferiore al 5%.

7.4 Efficienza

7.4.1 Response Time

- **Codice**: MPD09
- **Formula**: $Response\ Time = Tempo\ medio\ di\ risposta$
- **Descrizione**: Tempo medio impiegato per rispondere a una richiesta. Indica l'efficienza del prodotto. Un tempo di risposta inferiore a 2 secondi è considerato accettabile, mentre un tempo inferiore a 1 secondo è considerato ottimo.

7.5 Manutenibilità

7.5.1 Code Smells

- **Codice**: MPD10
- **Formula**: $Code\ Smells = \left(\frac{\text{Numero di code smells}}{\text{KLOC}}\right)$
- **Descrizione**: Numero di code smells per 1000 linee di codice. Un valore superiore a 10 indica possibili problemi di manutenibilità.

7.5.2 Coefficient of Coupling (CoC)

- **Codice**: MPD11
- **Formula**: $Coefficient\ of\ Coupling = \left(\frac{\text{Numero di dipendenze}}{\text{Numero di componenti}}\right)$
- **Descrizione**: Numero medio di dipendenze tra le componenti del sistema. Un valore superiore a 0.4 indica un accoppiamento eccessivo tra le componenti.

7.5.2.1 Cyclomatic Complexity

- **Codice**: MPD12
- **Formula**: $Cyclomatic\ Complexity = E - N + P$
- **Descrizione**:
  - E = numero di archi nel grafo di controllo
  - N = numero di nodi nel grafo di controllo
  - P = numero di componenti connesse da ogni arco

  Misura la complessità del codice contando i percorsi linearmente indipendenti. Un valore superiore a 10 indica codice complesso che potrebbe richiedere refactoring.