NoTIP

NO TESTS IN PRODUCTION

Norme di Progetto

Versione	1.1.0
Data Modifica	2026-02-26
Utilizzo	Interno

Abstract dei contenuti

Raccolta delle norme atte a regolare lo sviluppo del progetto affidato a NoTIP

Changelog$_G$

Versione	Data	Autori	*Verificatore*$_G$	Descrizione
1.1.0	2026-02-26	Alessandro Contarini, Leonardo Preo	Valerio Solito	Adozione struttura "Norme e Strumenti + Attività" per processi primari e di supporto
1.0.0	2026-02-12	Valerio Solito (responsabile)		Approvazione per ingresso in *baseline*$_G$ *RTB*$_G$
0.15.2	2026-02-08	Francesco Marcon	Alessandro Mazzariol	Correzione typo, aggiunta tabelle descrittive per i documenti, aggiunta norma al processo di "Documentazione"
0.15.1	2026-02-08	Alessandro Contarini	Francesco Marcon	Allineamento metriche di qualità con quelle del piano di qualifica
0.15.0	2026-02-07	Mario De Pasquale	Francesco Marcon	Stesura metriche e standard per la qualità
0.14.0	2026-02-07	Francesco Marcon	Alessandro Contarini	Aggiunta sezione di "Processo di Formazione"
0.13.0	2026-02-06	Alessandro Contarini	Francesco Marcon	Stesura metriche di qualità del prodotto
0.12.0	2026-02-04	Valerio Solito	Alessandro Mazzariol	Aggiunte sezioni parte 4 e sistemata struttura
0.11.0	2026-02-01	Alessandro Contarini	Leonardo Preo	Stesura metriche di qualità di processo
0.10.0	2026-01-30	Alessandro Mazzariol	Matteo Mantoan	Completamento sezione fornitura e sviluppo dei processi primari
0.9.0	2026-01-28	Francesco Marcon	Matteo Mantoan	Aggiunta sezione "Gestione di configurazioni" in Processi di Supporto
0.8.0	2026-01-25	Valerio Solito	Alessandro Mazzariol	Aggiunta sezione Fornitura e Sviluppo nei processi primari
0.7.0	2026-01-24	Valerio Solito	Alessandro Mazzariol	Aggiunta sezione *Jira*$_G$, modificato anche lib.typ
0.6.0	2026-01-12	Matteo Mantoan	Alessandro Mazzariol	Cambiamento alla struttura del documento, espansione sezione Documentazione
0.5.0	2025-11-21	Francesco Marcon	Matteo Mantoan	Aggiornamento in seguito al cambiamento di versione e struttura della repo
0.4.1	2025-11-16	Leonardo Preo, Mario De Pasquale	Matteo Mantoan	Aggiustamento ore produttive per *sprint*$_G$
0.4.0	2025-11-16	Leonardo Preo, Mario De Pasquale	Matteo Mantoan	Aggiunta specifica criteri rotazione dei ruoli
0.3.1	2025-11-05	Leonardo Preo	Francesco Marcon	Sostituzione Figura 1 "Ciclo di vita dei documenti"

Versione	Data	Autori	*Verificatore*$_G$	Descrizione
0.3.0	2025-11-04	Leonardo Preo	Matteo Mantoan	Modifica spiegazione *commit*$_G$ e verifica
0.2.0	2025-11-04	Leonardo Preo	Francesco Marcon	Modifica ciclo di vita del documento - nuova tecnica di *versionamento*$_G$
0.1.0	2025-10-31	Leonardo Preo, Matteo Mantoan	Alessandro Mazzariol	Prima stesura
0.0.1	2025-10-30	Francesco Marcon, Leonardo Preo, Matteo Mantoan	Alessandro Mazzariol	Prima bozza delle Norme di Progetto

Indice

Indice delle figure

Indice delle tabelle

1. Introduzione

Questo documento descrive il Way of Working$_G$ adottato dal gruppo NoTIP per il progetto didattico, definendo le procedure, gli strumenti e le responsabilità che guidano il team.

Il testo è da considerarsi un ***Living Document***$_G$: la sua stesura è incrementale e seguirà l'evoluzione del metodo di lavoro, perfezionandosi iterativamente. Ogni membro del gruppo è tenuto a visionare ogni nuova versione e a rispettare quanto descritto, per garantire uniformità e professionalità.

1.1. Scopo del prodotto

Il progetto ha l'obiettivo di realizzare il sistema oggetto del *capitolato*$_G$ **C7**, proposto dall'azienda **M31 S.r.l.**.

Il contesto applicativo riguarda il contesto *IoT*$_G$, dove l'acquisizione e la gestione centralizzata di dati provenienti da sensori distribuiti è fondamentale. L'obiettivo primario è sviluppare una piattaforma **Cloud** distribuita, scalabile e sicura, in grado di acquisire dati provenienti da sensori ***BLE***$_G$ tramite *Gateway*$_G$. È inoltre fondamentale andare a garantire la segregazione dei dati in modalità **multi-*tenant***$_G$ ed esporre API per la consultazione storica e lo streaming in tempo reale dei dati.

Poiché i dispositivi fisici (sensori e *gateway*$_G$) non sono oggetto di fornitura, sarà necessario andare a sviluppare un **simulatore di *gateway***$_G$ capace di generare traffico dati verosimile per validare l'infrastruttura e i flussi di comunicazione.

L'obiettivo che si è posto il gruppo è andare a sviluppare questo progetto entro il **21 marzo 2026** rispettando il costo previsto per la realizzazione del progetto di **12.940 Euro**.

1.2. Riferimenti e *Tailoring*$_G$

Tutti i processi descritti fanno riferimento allo standard ***ISO/IEC 12207:1995***$_G$. Attraverso un'operazione di tailoring$_G$, il team ha selezionato i processi e le attività pertinenti al contesto del progetto, classificandoli nelle tre macro-categorie previste dallo standard:

- **Processi Primari**: attività fondamentali per la realizzazione e la fornitura del prodotto software;
- **Processi di Supporto**: attività trasversali che garantiscono la qualità, la tracciabilità e la corretta gestione del ciclo di vita;
- **Processi Organizzativi**: attività di gestione, infrastruttura e miglioramento del progetto.

1.3. Organizzazione dei contenuti

Il documento è strutturato per Processi. Per distinguere chiaramente le norme di riferimento («con cosa e secondo quali regole lavoriamo») dalle procedure operative («come eseguiamo i compiti»), ogni sezione di Processo è suddivisa sistematicamente in due parti fondamentali:

1. **Norme e strumenti** Raccoglie le specifiche tecniche, le convenzioni e gli standard che regolano il processo. Definisce sia l'infrastruttura da utilizzare, sia le norme che il team deve rispettare. Costituisce la base di conoscenza normativa necessaria prima di avviare qualsiasi operazione.
2. **Attività** Descrive il workflow operativo vero e proprio. Dettaglia la sequenza temporale di azioni necessarie per raggiungere gli obiettivi del processo.
   - *Nota:* Ogni attività indica esplicitamente in apertura quali elementi della sezione «Norme e strumenti» sono prerequisiti per la sua esecuzione.

In aggiunta, dove necessario, sono presenti box informativi che spiegano le motivazioni dietro le scelte adottate.

1.4. Glossario

Lo sviluppo di un progetto di questa entità necessita della redazione di un Glossario, affinché sia possibile evitare certe ambiguità che potrebbero sorgere durante la lettura della documentazione.

La nomenclatura utilizzata per segnalare che la definizione di una parola è contenuta nel Glossario è la seguente:

$$parola_G$$

NoTIP si impegna a visionare ed aggiornare periodicamente il Glossario, per permettere la più completa comprensione dell'intera documentazione riguardante il progetto.

2. Processi primari

In conformità alla norma ***ISO/IEC 12207:1995***$_G$, i processi primari rappresentano l'insieme delle attività fondamentali che definiscono il ciclo di vita del software. Essi comprendono, in generale, i processi di **acquisizione**, **fornitura**, **esercizio** e **manutenzione**.

Nel contesto del progetto, risultano rilevanti esclusivamente i processi di fornitura e sviluppo:
- Il **processo di fornitura** disciplina le attività relative alla pianificazione, definizione e gestione dell'impegno tra il fornitore e il committente;
- Il **processo di sviluppo** comprende le attività di analisi, progettazione, implementazione, verifica e validazione del prodotto software.

2.1. Fornitura

La fornitura è il processo primario adottato dal fornitore del futuro prodotto finale che si occupa di analizzare le azioni da intraprendere per la sua realizzazione. Questo processo prevede un primo studio dei requisiti che il progetto dovrà, nelle componenti prodotte, soddisfare. Ciò produce il materiale necessario per poter iniziare la fase di contrattazione dei requisiti con il proponente, e poter comunicare allo stesso una possibile pianificazione del lavoro da svolgere con probabile data di consegna prevista.

2.1.1. Norme e strumenti del processo di fornitura

2.1.1.1. Strumenti a supporto

Per procedere allo svolgimento delle attività, il gruppo ha deciso di usare i seguenti strumenti per le comunicazioni interne:
- ***Jira***$_G$: per la gestione del *backlog*$_G$ e del tracciamento delle *task*$_G$, offre una visualizzazione di diagrammi di qualsiasi tipologia per facilitare la pianificazione;
- ***GitHub***$_G$: per il *versionamento*$_G$ dei documenti. Utile anche a fini di verifica dei documenti e approvazione degli stessi;
- ***Discord***$_G$: usato principalmente come luogo per riunioni interne e sessioni di sviluppo sincrone;
- **Telegram**: usato come canale principale di comunicazione testuale all'interno del gruppo.

Per quanto concerne la comunicazione con l'azienda proponente, vengono utilizzati i seguenti strumenti:
- **Microsoft Teams**: usato come canale di comunicazione sincrono tra committente e gruppo;
- **Google Mail**: usato come canale testuale tra committente e gruppo.

2.1.1.2. Documentazione prodotta

Nella sezione seguente si elencano i documenti che il gruppo NoTIP consegnerà al committente *M31* e ai proponenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

Dichiarazione di impegni

La Dichiarazione di impegni v1.0.0 è il documento in cui il gruppo ha stimato i costi del progetto, dall'impegno orario per persona e per ruolo, al costo complessivo del progetto e dei ruoli che i componenti del gruppo ricopriranno.

Voce	Dettaglio
Redattore	Responsabile
Destinatari	NoTIP, M31, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Lettera di presentazione
La Lettera di presentazione è il documento con il quale il gruppo conferma la volontà di candidarsi per una determinata *Baseline*$_G$. Il gruppo durante lo sviluppo del progetto presenterà ai proponenti tre lettere di presentazione:
- La Lettera di presentazione per la **Candidatura all'*appalto*$_G$ del *capitolato*$_G$ C7**;
- La Lettera di presentazione per la **Requirements and Technology *Baseline*$_G$ (*RTB*$_G$)**;
- La lettera di presentazione per la **Product *Baseline*$_G$ (PB)**.

Voce	Dettaglio
Redattore	Responsabile
Destinatari	NoTIP, M31, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Analisi dei capitolati
L'Analisi dei capitolati v1.0.1 è il documento in cui il gruppo fornisce un analisi dettagliata di ogni *capitolato*$_G$ evidenziando diversi punti, in particolare l'analisi suddivide ogni *capitolato*$_G$ in diverse sezioni:
- **Panoramica**: che indica l'azienda proponente, il nome del *capitolato*$_G$ e delle

informazioni generali sul prodotto da realizzare;
- **Pro**;
- **Contro**;
- **Considerazione finale**: motivazioni sull'eventuale scelta o non di candidarsi al *capitolato*$_G$

Voce	Dettaglio
Redattore	Responsabile
Destinatari	NoTIP, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Analisi dei Requisiti
L'Analisi dei Requisiti v1.1.0 definisce nel dettaglio i requisiti obbligatori, desiderabili e opzionali del progetto. Il documento mira a risolvere le ambiguità derivanti dalla lettura del *Capitolato*$_G$ C7, fornendo una base solida per la progettazione attraverso:

- **Descrizione del prodotto**: analisi puntuale del sistema richiesto dal committente.
- **Casi d'uso**: identificazione degli scenari d'uso e delle interazioni tra utenti e sistema. Ogni caso d'uso include una descrizione dettagliata degli scenari principali, permettendo ai progettisti di comprendere il comportamento atteso del software in ogni situazione.
- **Lista dei Requisiti**: rappresenta l'insieme dettagliato delle funzionalità, dei vincoli e delle qualità del sistema, derivanti dalle richieste del proponente o identificato dal gruppo durante l'attività di analisi.

Voce	Dettaglio
Redattore	Analista
Destinatari	NoTIP, M31, Prof. Vardanega, Prof. Cardin
Uso	Esterno

I dettagli riguardanti la redazione del documento possono essere trovati nelle norme Sezione 2.2.1.1 Nomenclatura dei Casi d'Uso, Sezione 2.2.1.2 Struttura dei Casi d'Uso e Sezione 2.2.1.3 Nomenclatura dei Requisiti del processo di sviluppo.

Piano di Progetto

Il Piano di Progetto v1.0.0 definisce e organizza la pianificazione strategica e operativa del gruppo, fornendo una roadmap dettagliata delle attività e gestione delle risorse. Il documento si compone delle seguenti sezioni:

- **Analisi dei rischi**: identifica e qualifica le criticità che potrebbero manifestarsi durante il ciclo di vita del progetto. A ogni rischio è associata una strategia di mitigazione, volta a ridurne l'impatto o la probabilità che accada.
- **Pianificazione**: definisce la sequenza temporale dei periodi di lavoro ($Sprint_G$). Per ogni $Sprint_G$ sono riportate le attività da completare, il preventivo orario per componente e il consuntivo delle ore effettivamente impiegate, con il relativo aggiornamento del budget residuo.

Voce	Dettaglio
Redattore	Responsabile
Destinatari	NoTIP, M31, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Piano di Qualifica

Descrive i metodi di qualifica (Verifica e Validazione) che sono state adottate dal gruppo. Sono inclusi i test effettuati sul prodotto e i rispettivi esiti.

Voce	Dettaglio
Redattore	Amministratore
Destinatari	NoTIP, M31, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Verbali Esterni
Sono i documenti che riassumono i contenuti trattati nelle riunioni tenute con soggetti esterni al gruppo (es. azienda proponente). Hanno lo scopo di formalizzare le decisioni prese, i chiarimenti ottenuti e gli accordi stipulati durante gli incontri ufficiali.

Voce	Dettaglio
Redattore	Responsabile
Destinatari	M31, NoTIP, Prof. Vardanega, Prof. Cardin
Uso	Esterno

Verbali Interni
Sono i documenti che riassumono i contenuti trattati nelle riunioni interne al gruppo, senza la partecipazione di soggetti esterni. Servono a tracciare l'avanzamento dei lavori, la suddivisione dei compiti e le decisioni tecniche o organizzative prese dal team.

Voce	Dettaglio
Redattore	Responsabile
Destinatari	NoTIP, Prof. Vardanega, Prof. Cardin
Uso	Interno

Glossario
Il Glossario v2.0.0 raccoglie e definisce in modo univoco i termini tecnici e gli acronimi utilizzati nella documentazione. Il suo scopo è eliminare le ambiguità linguistiche, garantendo una comunicazione uniforme sia tra i membri del gruppo sia verso gli stakeholder esterni.

Voce	Dettaglio
Redattore	Amministratore
Destinatari	M31, NoTIP, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.2. Attività del processo

2.1.2.1. Inizializzazione

Ruoli Responsabile

Riferimenti • Sezione 2.1.1.1 Strumenti a supporto

Input *Capitolato*$_G$ o proposta del committente

Output Decisione di procedere o controproposta

Procedura
1. **Analisi delle richieste**: Il fornitore analizza le richieste del committente, tenendo conto di eventuali vincoli organizzativi o di altra natura. Questa è l'attività in cui il fornitore decide se proseguire con quanto proposto o di preparare una controproposta.

2.1.2.2. Risposta

Ruoli Responsabile

Riferimenti • Sezione 2.1.1.1 Strumenti a supporto

Input Analisi delle richieste completata

Output Risposta formale o controproposta al committente

Procedura
1. **Elaborazione della risposta**: Viene elaborata e presentata una risposta che può essere una controproposta dei requisiti, oppure una proposta su come soddisfare i requisiti.

2.1.2.3. Contrattazione

Ruoli Responsabile

Riferimenti • Sezione 2.1.1.1 Strumenti a supporto

Input Risposta accettata da entrambe le parti

Output Accordo formale su costi, tempi e criteri di accettazione

Procedura
1. **Incontro con il committente**: Si effettua un incontro con il committente con l'obiettivo di arrivare ad un accordo (sancito solitamente da un contratto formale) definendo costi, tempi e criteri di accettazione.

2.1.2.4. Pianificazione

Ruoli Responsabile

Riferimenti • Sezione 2.1.1.2 Documentazione prodotta

Input Requisiti finali concordati

Output Piano di Progetto

Procedura
1. **Stesura del Piano di Progetto**: Stabiliti i requisiti finali, il fornitore pianifica l'organizzazione e un metodo di lavoro in grado di assicurare la qualità del sistema da realizzare. La pianificazione comprende la stesura del Piano di Progetto, dove vengono indicate le risorse richieste per realizzare il prodotto, considerando anche i rischi che potrebbero accadere durante lo sviluppo.

2.1.2.5. Esecuzione e controllo

Ruoli Responsabile, Amministratore, Analista, Progettista, Programmatore, *Verificatore*$_G$

Riferimenti
- Sezione 2.1.1.2 Documentazione prodotta

Input Piano di Progetto approvato

Output Prodotto in lavorazione, avanzamento monitorato

Procedura
1. **Realizzazione e monitoraggio**: Il fornitore realizza il prodotto, monitorando nel frattempo la qualità di quanto fatto e il progresso raggiunto.

2.1.2.6. Revisione

Ruoli Responsabile

Riferimenti
- Sezione 2.1.1.1 Strumenti a supporto

Input Incremento di prodotto realizzato

Output Feedback del proponente integrato

Procedura
1. **Raccolta feedback**: Il fornitore si tiene in contatto con la proponente, in modo da ottenere feedback su quanto realizzato.

2.1.2.7. Consegna e completamento

Ruoli Responsabile

Riferimenti
- Sezione 2.1.1.2 Documentazione prodotta
- Sezione 2.1.1.1 Strumenti a supporto

Input Prodotto completato e approvato

Output Consegna formale al committente

Procedura
1. **Consegna**: Il fornitore, completato il progetto, fornisce quanto prodotto al committente.

2.2. Sviluppo

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

2.2.1. Norme e strumenti del processo di sviluppo

2.2.1.1. Nomenclatura dei Casi d'Uso

Per garantire univocità e tracciabilità, i casi d'uso adottano la seguente nomenclatura:

```
UC[Codice].[Sottocaso] - [Titolo]
```

dove:
- **UC**: acronimo di Use Case;
- **[Codice]**: numero identificativo univoco del caso d'uso principale;
- **[Sottocaso]**: numero identificativo progressivo gerarchico per identificare scenari derivati o specifici (ci possono essere sottocasi derivanti da altri sottocasi);
- **[Titolo]**: titolo sintetico ed esplicativo dell'azione.

Per la parte B (Simulatore), la nomenclatura viene estesa in **UCS (Use Case Simulatore)**.

2.2.1.2. Struttura dei Casi d'Uso

Ogni caso d'uso viene dettagliato secondo la seguente struttura:

- **Attori Primari**: utenti e attori che avviano l'interazione;
- **Attori Secondari**: destinatari di notifiche o sistemi esterni coinvolti passivamente;
- **Precondizioni**: stato del sistema o condizioni necessarie per l'attivazione del caso d'uso;
- **Postcondizioni**: stato garantito del sistema a seguito del completamento con successo;
- **Scenario Principale**: sequenza di azioni atomiche in linguaggio naturale, inclusi eventuali:
  - Punti di Inclusione (`Include: UC[ID] - Titolo`);
  - Punti di Estensione (`Descrizione passo. [EP: NOME]`).
- **Estensioni**: gestione di scenari alternativi o eccezioni, definiti da una condizione di guardia e dal relativo caso d'uso esteso.

2.2.1.3. Nomenclatura dei Requisiti

Ogni requisito è identificato dalla seguente nomenclatura:

`R-[Numero]-[Tipologia] [Priorità]`

dove:

- **R**: abbreviazione di Requisito;
- **Numero**: valore univoco che identifica il requisito;
- **Tipologia**: indica la natura del requisito:
  - **F** per **Funzionale**;
  - **Q** per **Qualità**;
  - **V** per **Vincolo**;
  - **S** per **Sicurezza**.
- **Priorità**: indica l'importanza strategica del requisito:
  - **Obbligatorio**: indispensabile per la validità del progetto;
  - **Desiderabile**: non indispensabile, ma con valore aggiunto;
  - **Opzionale**: funzionalità aggiuntive a bassa priorità.

Per la parte B (Simulatore), la nomenclatura viene estesa in **RS (Requisito Simulatore)**.

2.2.1.4. Stack Tecnologico

Le tecnologie adottate per lo sviluppo del progetto sono:

- ***Backend***$_G$: Linguaggio ***Go***$_G$ (*Golang*$_G$);
- ***Frontend***$_G$: Framework ***Angular***$_G$ con linguaggio ***TypeScript***$_G$;
- ***Automazione***$_G$/***Scripting***$_G$: Linguaggio **Python** per la gestione della documentazione.

2.2.1.5. Strumenti di Qualità del Codice

È richiesta la configurazione dell'*editor*$_G$ per l'applicazione automatica al salvataggio («format on save») e i seguenti strumenti per formattare il codice:

Ambito	Tool	Descrizione
Go$_G$	`gofmt`	Standard per la formattazione.

Ambito	Tool	Descrizione
***TypeScript*$_G$**	`ESLint` `Prettier`	Analisi per prevenire errori comuni e best-practices (ESLint) e formattazione stilistica (Prettier).

2.2.1.6. Convenzioni di Scrittura e Nomenclatura

Tutti i membri del gruppo sono tenuti a rispettare le seguenti convenzioni stilistiche:

- **Lingua**: Il codice, inclusi i nomi di variabili, funzioni e i commenti, deve essere scritto interamente in **Inglese**;
- **File**: Mantenere i file snelli con responsabilità singola. Se un file cresce eccessivamente, deve essere rifattorizzato;
- **Nomenclatura**:
  - **Variabili**: Utilizzare la notazione `PascalCase` (es. `MyVariable`);
  - **Funzioni/Metodi**: Utilizzare la notazione `camelCase` (es. `myFunction`);
  - **Costanti**: Evitare costanti globali; prediligere configurazioni o variabili contestualizzate.

2.2.2. Attività del processo

2.2.2.1. Analisi dei Requisiti di Sistema

Ruoli Analista

Riferimenti
- Sezione 2.2.1.1 Nomenclatura dei Casi d'Uso
- Sezione 2.2.1.2 Struttura dei Casi d'Uso
- Sezione 2.2.1.3 Nomenclatura dei Requisiti

Input *Capitolato*$_G$ C7, verbali degli incontri con il committente

Output Analisi dei Requisiti v1.1.0 (casi d'uso e lista requisiti)

Procedura
1. **Modellazione dei Casi d'Uso**: Individuazione degli attori e delle interazioni con il sistema. Ogni caso d'uso viene redatto rispettando la struttura e la nomenclatura definite nelle norme.
2. **Individuazione dei Requisiti**: Derivazione dei requisiti dal *capitolato*$_G$ e dagli incontri con il committente. Ogni requisito viene classificato per tipologia e priorità secondo la nomenclatura definita nelle norme.

2.2.2.2. Workflow di Sviluppo del Codice

Ruoli Programmatore

Riferimenti
- Sezione 2.2.1.4 Stack Tecnologico
- Sezione 2.2.1.5 Strumenti di Qualità del Codice
- Sezione 2.2.1.6 Convenzioni di Scrittura e Nomenclatura

Input Specifica architetturale, *Task*$_G$ *Jira*$_G$ assegnato

Output Codice sorgente committato, Unit Tests

Procedura
1. **Setup e Branching**: Creazione del *feature branch*$_G$ dedicato al *task*$_G$ corrente a partire dal ramo di sviluppo aggiornato.

2. **Codifica**: Scrittura del codice sorgente rispettando le convenzioni di nomenclatura definite. Il codice deve essere auto-esplicativo; eventuali commenti in inglese devono chiarire una sezione complessa.
3. ***Commit*$_G$**: Salvataggio delle modifiche sul *repository*$_G$ remoto seguendo le convenzioni dei messaggi di *commit*$_G$.

3. Processi di supporto

Secondo quanto definito dallo standard ***ISO/IEC 12207:1995***$_G$, i **processi di supporto** non concorrono direttamente alla realizzazione del prodotto software, ma svolgono un ruolo essenziale nel garantirne il controllo, la tracciabilità e la qualità durante tutte le fasi del suo ciclo di vita.

3.1. Processo di documentazione

Il **processo di documentazione** ha lo scopo di registrare informazioni che derivano da un processo o ciclo di vita. La documentazione ha l'obiettivo di tracciare storicamente ed in modo preciso tutte le decisioni intraprese durante lo svolgimento del progetto.

3.1.1. Norme e strumenti del processo di documentazione

3.1.1.1. Ciclo di vita e *versionamento*$_G$ dei documenti

Il ciclo di vita di un documento prevede tre stati fondamentali mappati sul *branch*$_G$ in cui si trova il file:

- **In lavorazione** (*Branch*$_G$ di feature): Il documento è in fase di stesura o modifica. La versione è provvisoria;
- **Verificato** (*Branch*$_G$ `main`$_G$): Il documento ha superato la revisione tecnica (*PR*$_G$) ed è stato integrato. È stabile ma non necessariamente rilasciato;
- **Approvato** (*Branch*$_G$ `main`$_G$): Il documento corrisponde a una *Baseline*$_G$ ufficiale. Richiede l'approvazione esplicita del Responsabile.

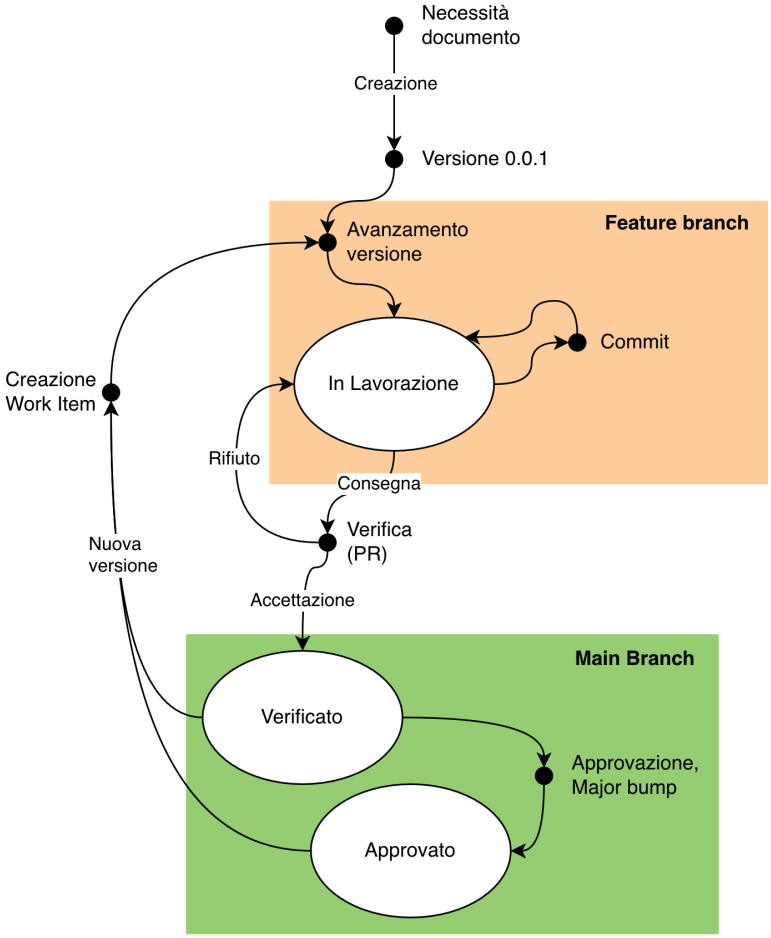

Figura 1: Diagramma degli stati del ciclo di vita documentale

Figura 1 raffigura un diagramma del ciclo di vita dei documenti.

Standard di *versionamento*$_G$ (*SemVer*$_G$)
L'avanzamento di versione segue lo standard `X.Y.Z` in base alla natura delle modifiche:

Componente	Tipo	Descrizione e Criterio
Z	*Patch*$_G$	Correzione di refusi o formattazione. *Azione:* Verifica (Peer Review).
Y	*Minor*$_G$	Aggiunta o modifica di contenuti sostanziali. *Azione:* Verifica (Peer Review).
X	*Major*$_G$	Approvazione del documento come versione stabile. *Azione:* Approvazione del Responsabile.

3.1.1.1.1. *Note*

Differenza tra verifica e approvazione: La verifica controlla la correttezza tecnica puntuale. L'approvazione è un atto formale che dichiara il documento «pronto» e stabile per gli scopi della *Milestone*$_G$ corrente. Separare questi momenti permette di tracciare chi ha controllato la correttezza (*Verificatore*$_G$) e chi si è assunto la responsabilità del rilascio formale (Responsabile).

3.1.1.2. Struttura del *repository*$_G$

Il *repository*$_G$ segue il paradigma *Convention Over Configuration*$_G$. La posizione dei file nel file system fornisce *metadati*$_G$ impliciti necessari alla validazione e pubblicazione.

Alberatura del progetto
Di seguito è riportata la struttura di riferimento del *repository*$_G$:

```
.
├── .schemas/                # schemi JSON_G per validazione metadati_G
├── docs/                    # sorgenti della documentazione
│   ├── 00-common_assets/    # risorse condivise (loghi, icone)
│   └── {milestone_G}/        # raggruppamento per baseline_G (es. 12-rtb_G)
│       ├── 00-templates/    # template Typst_G per baseline_G
│       ├── {subgroup}/      # classificazione (docint, docest, verbali...)
│       │   └── {doc}/       # directory del singolo documento
│       │       ├── assets/          # (opzionale) immagini specifiche
│       │       ├── {subfiles_G}/     # (consigliato) capitoli/sezioni .typ
│       │       ├── {doc}.typ        # entry point di compilazione
│       │       └── {doc}.meta.yaml_G # metadati_G del documento
├── notipdo/                 # tool Python per automazione_G e CI
└── site/                    # sorgenti del sito statico
```

Dettaglio directory `docs`

- `00-common_assets/`: *Asset*$_G$ condivisi (es. font, loghi). Vietato duplicare questi file nelle cartelle locali.
- **Livello *Milestone*$_G$ (`docs/xx-nome`)**: Le directory sono numerate progressivamente (`xx >= 11`) per mantenere l'ordine cronologico delle *Baseline*$_G$. Questo livello è anche chiamato *group* all'interno della documentazione;
- **Livello Subgroup**: Classificazione funzionale dei documenti:
  - `docint/`: Documentazione interna (e.g. Norme di Progetto);

  - `docest/`: Documentazione esterna (e.g. Analisi dei Requisiti, Piano di Progetto);
  - `verbint/` / `verbest/`: Verbali interni ed esterni;
  - `slides/`: Presentazioni e Diari di Bordo.
- **Livello Documento**: Ogni documento deve risiedere nella propria cartella `{nome_documento}` in `snake_case`:
  - `{nome_documento}.typ`: File «master». Non scrivere contenuto qui se il documento è lungo; in tal caso, deve contenere solo gli `include` dei moduli e la configurazione del *template*$_G$;
  - `{subfiles}`$_G$`/`: Directory che contiene i file `.typ` parziali (es. `cap1_intro.typ`), Obbligatorio per documenti lunghi e/o collaborativi per limitare i conflitti di *merge*$_G$;
  - `{nome_documento}.meta.yaml`$_G$: *Metadati*$_G$ (titolo, *changelog*$_G$) conformi allo schema;
  - *Nota*: Per i documenti periodici (`ddb`, `verbint`, `verbest`), è obbligatorio suffissare il nome con la data in formato ISO (`yyyy-mm-dd`) per garantire l'ordinamento cronologico (es. verbint_2025-11-21).

Integrità storica delle *baseline*$_G$
È severamente vietato modificare i file appartenenti a directory di *baseline*$_G$ passate. Eventuali correzioni a documenti storici devono essere discusse con il Responsabile.

3.1.1.2.1. *Note*

Perché dividere un documento in *subfiles*$_G$? Separare il contenuto in file più piccoli all'interno di una cartella riduce drasticamente la probabilità di conflitti *Git*$_G$ quando più persone lavorano sullo stesso documento.

Perché l'organizzazione semantica? La struttura delle cartelle rispecchia la struttura del sito web pubblicato, facilitando la navigazione e permettendo agli script di generare il sito statico senza configurazioni aggiuntive.

3.1.1.3. Norme tipografiche e stilistiche

Per garantire la qualità dei documenti, ogni autore è tenuto a rispettare le seguenti convenzioni. Il *verificatore*$_G$ utilizzerà queste tabelle come **checklist** durante la revisione.

Convenzioni Tipografiche

Elemento	Utilizzo	Esempio
Grassetto	Concetti chiave o formattazione.	• **Concetto A**: [...] ; • **Concetto B**: [...] .
Corsivo	Enfasi su parole specifiche, termini di glossario.	[...] perciò si decide di *non* intraprendere questa via. Il team adotta un approccio *Agile*$_G$.
`Monospazio`	Codice, nomi di file, percorsi (path), *branch*$_G$, nomi di variabili/funzioni.	Eseguire lo script `do.py`. Il file si trova nella directory `notipdo/`.
Glossario	Termini ambigui, poco conosciuti o malinterpretabili. La formattazione è automatica nei documenti *Typst*$_G$.	La suddetta *baseline*$_G$...

Elemento	Utilizzo	Esempio
Maiuscole	Nomi di ruoli, titoli dei documenti, espansione di acronimi.	Il Responsabile è tenuto all'approvazione delle Norme di Progetto.
Elenchi puntati	Ogni punto elenco inizia con lettera maiuscola e termina con un punto e virgola (;), eccetto l'ultimo che termina con il punto (.). Se il punto contiene più frasi, usare la punteggiatura standard interna.	• Fase di planning; • Fase di azione; • Fase di retrospettiva.

Stile e Linguaggio
Il registro linguistico deve adattarsi al destinatario del documento secondo lo schema seguente:

Destinatario	Registro e Caratteristiche
Interno (Team)	**Tecnico e Sintetico.** È ammesso l'uso di gergo tecnico condiviso e una struttura schematica per massimizzare la velocità di lettura. L'obiettivo è l'efficienza operativa.
Esterno - Tecnico (Committente/Professori)	**Formale e Professionale.** Uso rigoroso della terminologia standard. Le frasi devono essere complete, oggettive e non ambigue. In caso di ambiguità, porre i termini nel Glossario. Privilegiare la forma impersonale (es. «Si è scelto di...») o la prima persona plurale («Abbiamo analizzato...»).
Esterno - Utente (Manuali utente)	**Semplice e Accessibile.** Evitare tecnicismi. Spiegare i concetti complessi con esempi pratici. Il tono deve essere guidato e rassicurante.

3.1.1.4. *Versionamento*$_G$ dei riferimenti
Al fine di garantire la massima tracciabilità e la riproducibilità delle decisioni, ogni volta si inserisce un riferimento ad un documento interno o a una fonte esterna, è obbligatorio esplicitare la versione di quest'ultima.
In questo modo è possibile prevenire eventuali ambiguità derivanti dall'evoluzione dei contenuti nel tempo e assicura che qualsiasi lettore o *verificatore*$_G$ possa consultare l'esatta versione del documento utilizzata dall'autore al momento della stesura.

3.1.1.5. Nomenclatura Work Items di documentazione
I work items di documentazione presenti su *Jira*$_G$ devono seguire la seguente nomenclatura:

```
{nome_dir_documento}-v{nuova_versione_documento} - {Breve descrizione}
```

Per l'approvazione di un documento (ossia scatti di *Major*$_G$ version), la descrizione è sostituita da "Approvazione".

Nota: Non esagerare con la lunghezza dei nomi dei work items, in quanto determinano il nome dei relativi *branch*$_G$.

Esempi:
- "norme_progetto-v0.6.0 - Ristrutturazione e espansione processo documentazione";
- "piano_qualifica-v0.4.0 - *Sprint*$_G$ 4";
- "piano_progetto-v1.0.0 - Approvazione".

3.1.1.6. Gestione *Git*$_G$: Branching e *Commit*$_G$

Il *repository documentale*$_G$ segue una strategia di branching Trunk-Based, dove l'unica fonte di verità persistente è il *branch*$_G$ `main`$_G$. Ogni *feature branch*$_G$ è effimero e deve essere collegato univocamente a un Work Item registrato su *Jira*$_G$.

Nomenclatura *Branch*$_G$

È obbligatorio mantenere la nomenclatura generata automaticamente dal Work Item, rimuovendo la sezione `- {Breve descrizione}` dallo schema definito in Sezione 3.1.1.5. Esempio: `NT-67-norme_progetto-v0-6-0`.

Messaggi di *Commit*$_G$

I messaggi devono seguire tassativamente lo standard *Conventional Commits*$_G$ per garantire la leggibilità della *commit*$_G$ history.
- **Struttura:** `tipo(ambito): descrizione imperativa minuscola`
- **Tipi principali:** `docs` (cambiamento contenuti), `style` (formattazione dei file), `chore` (configurazioni/*asset*$_G$).
- **Esempi:**
  - `docs(norme_progetto): aggiunta sezione git`$_G$
  - `style(norme_progetto): formattazione file con typstyle`$_G$
  - `chore: inserimento font Noto Sans`

3.1.1.7. Co-authoring singole versioni di documento

Il co-authoring di un documento deve sempre essere guidato da un Work Item per ogni attività assegnata ad un componente.

Il titolo dei work items differisce solo per la parte di `{Breve descrizione}` (vedi Sezione 3.1.1.5). Esempio:
- "analisi_requisiti-v0.14.0 - aggiunta UC45-UC53", assegnato alla persona A;
- "analisi_requisiti-v0.14.0 - aggiunta UC54-57", assegnato alla persona B.

Comunicazione

La comunicazione tra gli autori del documento deve essere **sincrona** per evitare sovrapposizioni.

***Subfiles*$_G$**

Le diverse parti di documento su cui ognuno lavora devono essere divise in *subfiles*$_G$ *Typst*$_G$ per evitare conflitti.

Branching

Il caso del co-authoring è l'unico in cui viene meno la corrispondenza 1:1 tra *branch*$_G$ e Work Item. Viene creato un *branch*$_G$ per la versione (e.g. `NT-67-analisi_requisiti-v0-14-0`), slegato da qualunque work item, a partire da `main`$_G$. I *feature branch*$_G$ relativi ai work item di *Jira*$_G$ partono e vengono chiusi da e su quel *branch*$_G$ di versione con delle Pull Request. Il nome

di tali *branch*$_G$ è quello generato da *Jira*$_G$ comprensivo della parte di `- {Breve descrizione}` illustrata in Sezione 3.1.1.5. Una volta completata la stesura collaborativa, viene eseguito il *merge*$_G$ del *branch*$_G$ di versione in `main`$_G$ ed eliminato, preservando la pulizia del trunk.

Verifica
Una volta terminati tutti i work items relativi alla versione, *un* (solo) *verificatore*$_G$ può procedere a verificare tutte le *PR*$_G$, eseguire il *merge*$_G$ sul *branch*$_G$ di versione, ed eseguire un *merge*$_G$ finale in `main`$_G$.

3.1.1.8. Utilizzo del tool di *automazione*$_G$ `notipdo`

Il tool di *automazione*$_G$ interno alla *repository*$_G$ di documentazione si chiama `notipdo`.

Esso permette di
- *Validare* la struttura della *repository*$_G$ e la correttezza sintattica dei file;
- *Compilare* i documenti;
- Compilare i documenti in modalità *hot reload*;
- *Generare il sito* web del team contenente tutti i documenti;
- *Formattare i file* Typst$_G$ secondo le regole di formattazione del team;
- *Verificare* che un set di modifiche rispetti le norme del progetto. Ad esempio:
  - Verificare che non siano stati modificati documenti di vecchie *baseline*$_G$;
  - Verificare

Il tool è accompagnato dalla sua documentazione d'uso e da messaggi di errore chiari. Per iniziare ad utilizzarlo, seguire le istruzioni presenti in `README.md` e digitare `notipdo --help` sulla propria shell. `notipdo` supporta Linux, MacOS e Windows.

3.1.1.9. *Template Typst*$_G$

La redazione dei documenti deve partire esclusivamente dai *template*$_G$ ufficiali presenti in `docs/{milestone}/00-templates/`:

Template$_G$	Utilizzo
`base_document.typ`	Documenti generici (Analisi, Piano di Progetto, Norme)
`base_verbale.typ`	Verbali interni ed esterni. Include sezioni per ordine del giorno e decisioni
`base_slides.typ`	Presentazioni generiche
`base_ddb.typ`	Diari di bordo con struttura fissa

3.1.1.10. Matrice Documento - Ruolo autore

La seguente tabella rappresenta quale ruolo è responsabile della scrittura di ogni documento.

Documento	Ruolo autore
Norme di Progetto	Amministratore
Piano di Qualifica	Amministratore
Piano di Progetto	Amministratore
Analisi dei Requisiti	Amministratore
Verbali	Responsabile
Diari di Bordo	Responsabile
Analisi dei Requisiti	Analista
Analisi dello Stato dell'Arte	Progettista

Ognuno, durante la scrittura dei documenti, è tenuto ad inserire nel glossario i termini che lo richiedono.

3.1.1.11. Specifiche di contenuto per documenti periodici

Sebbene la struttura sia guidata dai *template*$_G$, deve essere garantita la presenza dei seguenti contenuti minimi, mappati sui parametri delle funzioni *Typst*$_G$ dedicate.

Verbali (Interni ed Esterni)

I verbali devono essere redatti tramite il *template*$_G$ `base_verbale.typ` e includere i seguenti elementi:

- **Informazioni di base (`front-info`):** Elenco dettagliato di tutti i presenti;
- **Dettagli logistici:** Indicazione esplicita della data, della piattaforma di comunicazione (es. Microsoft Teams, *Discord*$_G$) e della fascia oraria dell'incontro;
- **Ordine del Giorno (`odg`):** Sintesi dei punti previsti per la trattazione durante la riunione;
- **Svolgimento (`discussion`):** Il contenuto deve essere organizzato in blocchi tramite la funzione `report-point`, assicurando che ogni sezione contenga:
  - Un titolo chiaro per il punto di discussione (`discussion_point`);
  - La sintesi del dibattito intercorso (`discussion`);
  - Il riepilogo delle decisioni prese (`decisions`);
  - Eventuali azioni da intraprendere (`actions`), corredate da descrizione e URL diretto al relativo work item su *Jira*$_G$;
- **Approvazione Aziendale:** Per i verbali esterni, è obbligatorio includere la sezione finale dedicata alla firma dei referenti per la validazione formale dei contenuti.

Diari di Bordo (DdB)

Ogni *Diario di Bordo*$_G$ deve essere redatto tramite la funzione `apply-base-ddb` , indicando il numero dello *Sprint*$_G$ di riferimento. I contenuti devono seguire rigorosamente la suddivisione in tre blocchi posizionali:

1. **Risultati raggiunti:** Elenco puntato delle attività completate e dei traguardi raggiunti rispetto a quanto pianificato;
2. **Obiettivi per il periodo successivo:** Pianificazione dei *task*$_G$ previsti per lo *sprint*$_G$ seguente, inclusi avanzamenti documentali o tecnici;
3. **Criticità:** Esposizione di dubbi, difficoltà organizzative incontrate (es. durante le festività) o rischi occorsi durante lo svolgimento delle attività.

Per garantire l'integrità dei dati, ogni documento deve caricare il proprio registro delle modifiche tramite il file di *metadati*$_G$ `{nome_documento}.meta.yaml`$_G$.

3.1.2. Attività del processo

3.1.2.1. Pianificazione del documento

Ruoli Amministratore, Analista, Progettista, Programmatore, *Verificatore*$_G$, Responsabile

Riferimenti
- Sezione 3.1.1.5 Nomenclatura Work Items di documentazione
- Sezione 3.1.1.9 *Template Typst*$_G$
- Sezione 3.1.1.2 Struttura del *repository*$_G$

Input Necessità identificata di produrre o modificare documentazione

Output Work Item *Jira*$_G$, documento pronto alla lavorazione

Procedura
1. **Creazione Work Item**: Chi individua la necessità del nuovo documento apre un work-item su *Jira*$_G$:
   - Il titolo deve seguire rigorosamente la nomenclatura definita nelle norme (es. `doc-v0.1.0 - Stesura`);
   - Nella descrizione vanno specificati gli obiettivi della nuova versione del documento.
   - Il Work Item deve contenere stime del tempo richiesto dai *task*$_G$ di esecuzione e verifica.
2. **Setup (Solo nuovi documenti)**: Se il documento non esiste:
   - Creare la cartella e i file iniziali utilizzando il ***Template***$_G$ corretto (vedi *Standard dei* Template$_G$);
   - Inizializzare il *Changelog*$_G$ con la prima entry in versione `0.0.1` e descrizione "Creazione documento";
   - Aprire una *PR*$_G$ verso `main`$_G$.

3.1.2.2. Stesura del documento

Ruoli Autore (vedi Sezione 3.1.1.10 Matrice Documento - Ruolo autore)

Riferimenti
- Sezione 3.1.1.6 Gestione *Git*$_G$: Branching e *Commit*$_G$
- Sezione 3.1.1.3 Norme tipografiche e stilistiche
- Sezione 3.1.1.4 *Versionamento*$_G$ dei riferimenti
- Sezione 3.1.1.2 Struttura del *repository*$_G$
- Sezione 3.1.1.9 *Template Typst*$_G$
- Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`
- Sezione 3.1.1.7 Co-authoring singole versioni di documento

Input Work Item *Jira*$_G$ non assegnato

Output Pull Request nuova versione documento aperta

Procedura

1. **Presa in carico**: Auto-assegnarsi al Work Item relativo alla nuova versione del documento.
2. **Branching**: Creare un *branch*$_G$ locale a partire da `main`$_G$ aggiornato.
   - Nome *branch*$_G$: deve rispettare rigorosamente la nomenclatura definita (es. `NT-67-norme_progetto-v0-6-0`);
   - È severamente vietato lavorare direttamente su *main*$_G$ (azione tecnicamente inibita dalle policy del *repository*$_G$);
   - In caso di *co-authoring*, vedi Sezione 3.1.1.7 Co-authoring singole versioni di documento.
3. **Aggiornamento *Changelog*$_G$**: Creare la nuova versione nel *changelog*$_G$ del file `{nome_doc}.meta.yaml`$_G$, settando `TBD`$_G$ (To Be Defined) come *verificatore*$_G$.
4. **Redazione**: Scrivere i contenuti nel file `.typ` (o nei *subfiles*$_G$).
   - Consultare le *Norme tipografiche e stilistiche* per formattazione e registro linguistico;
   - Utilizzare `notipdo watch doc {doc_dir}` per lavorare al documento in modalità hot-reload.
5. **Formattazione**: Utilizzare `notipdo format doc {doc_dir}` per formattare correttamente i file sorgenti.
6. **Check delle modifiche**:
   - Assicurarsi che il documento compili correttamente;
   - Utilizzare `notipdo check pr`$_G$ e assicurarsi che le modifiche passino i controlli di formattazione e spellcheck.
7. ***Commit*$_G$ e Push (loop)**:
   - Eseguire *commit*$_G$ atomici usando i Conventional Commits$_G$ (es. `docs(norme_progetto): stesura introduzione`);
   - Eseguire il push sul *repository*$_G$ remoto.
8. **Apertura *PR*$_G$**: Una volta completata la stesura, aprire una Pull Request verso `main`$_G$.
9. **Submission Work Item**: Spostare il Work Item in verifica dopo aver segnato il tempo speso effettivamente sul *task*$_G$.

3.1.2.3. Verifica del documento

Ruoli *Verificatore*$_G$

Riferimenti

- Sezione 3.1.1.1 Ciclo di vita e *versionamento*$_G$ dei documenti
- Sezione 3.1.1.2 Struttura del *repository*$_G$
- Sezione 3.1.1.3 Norme tipografiche e stilistiche
- Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`
- Sezione 3.1.1.7 Co-authoring singole versioni di documento

Input Pull Request aperta

Output Documento verificato (su `main`$_G$) o richiesta modifiche

Procedura

1. **Presa in Carico**: Auto-assegnarsi al Work Item di verifica relativo alla nuova versione del documento.
   - In caso di versione realizzata in co-authoring, vedi Sezione 3.1.1.7 Co-authoring singole versioni di documento.

2. **Controlli Automatici**: Attendere l'esito della *pipeline*$_G$ di CI.
   - Se fallisce: La verifica è bloccata. L'autore deve risolvere gli errori tecnici.
3. **Revisione Umana**: Scaricare il PDF di anteprima (*artifact*$_G$), controllare il codice e verificare:
   - Rispetto delle norme tipografiche e stilistiche;
   - Completezza e correttezza del contenuto rispetto al Work Item.
4. **Feedback (Loop)**:
   - Se ci sono errori: Richiedere modifiche tramite commenti sulla *PR*$_G$. L'autore corregge e si riparte dal punto 1;
   - Se è tutto corretto: Procedere al punto successivo.
5. **Firma (Approvazione Tecnica)**: Modificare il *Changelog*$_G$: sostituire il placeholder `TBD`$_G$ (To Be Done) con il proprio Nome e Cognome. Questo sancisce la responsabilità della verifica.
6. **Controllo pre-*merge*$_G$**: Assegnare la label `merge`$_G$`-ready` alla *PR*$_G$. Questo scatenerà un controllo automatico che si assicura dell'avvenuta firma al punto precedente.
7. ***Merge***$_G$: Approvare la *PR*$_G$ su *GitHub*$_G$ ed eseguire lo Squash and Merge$_G$ sul *branch*$_G$ `main`$_G$.
8. **Chiusura Work Item**: Spostare il Work Item in «Done» dopo aver segnato il tempo speso effettivamente sul *task*$_G$.

3.1.2.4. Approvazione del documento

Ruoli Responsabile

Riferimenti
- Sezione 3.1.1.1 Ciclo di vita e *versionamento*$_G$ dei documenti
- Sezione 3.1.1.5 Nomenclatura Work Items di documentazione
- Sezione 3.1.1.6 Gestione *Git*$_G$: Branching e *Commit*$_G$

Input Documento in stato *Verificato* (su `main`$_G$), Rilascio di *Baseline*$_G$ imminente

Output Nuova *Major*$_G$ Version

Procedura

1. **Presa in carico**: Auto-assegnarsi al Work Item relativo all'approvazione del documento.
2. **Branching**: Aprire il *branch*$_G$ relativo al Work Item, come nel caso di redazione del documento.
3. **Valutazione**: Valutare la maturità del documento per l'ingresso in *Baseline*$_G$.
   - I contenuti del documento rispecchiano gli scopi della *baseline*$_G$?
   - La forma del documento è corretta?

   Se la risposta è negativa, viene annullata l'attività di approvazione e creato il Work Item relativo alla soluzione dei problemi riscontrati.
4. **Version Bump**:
   - Incrementare la versione da `X.Y.Z` a `X+1.0.0`;
   - Aggiornare la data del *Changelog*$_G$ alla data di rilascio corrente.
   - *Verificatore*$_G$: *TBD*$_G$
5. **Submission, *PR*$_G$, Verifica, Chiusura**: La chiusura del *task*$_G$ avviene esattamente come ogni altra versione.
   Unica differenza: il *verificatore*$_G$ è tenuto solo a controllare la spunta verde dei check automatici, non a controllare manualmente il contenuto.

3.1.2.4.1. *Note*

Questa attività sancisce il passaggio ufficiale da una versione di sviluppo a una versione stabile (*Baseline*$_G$).

3.1.2.5. Distribuzione della documentazione

Ruoli Amministratore

Riferimenti
- Sezione 3.1.1.2 Struttura del *repository*$_G$
- Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`

Input Documenti relativi alla *Baseline*$_G$ approvati

Output *Baseline*$_G$ pubblicata sul sito Web

Procedura
1. **Rilascio**: Lanciare l'*automazione*$_G$ di rilascio del sito su *GitHub*$_G$ Pages dal *branch*$_G$ `main`$_G$.
2. **Modifica configurazione `notipdo`**: Modificare le impostazioni di `notipdo` presenti nel file `notipdo/lib/configs.py`.

3.2. Gestione delle Configurazioni

Il processo di **Gestione delle Configurazioni** ha lo scopo di identificare, definire e tracciare gli elementi che compongono il prodotto software e la documentazione, controllandone le modifiche e registrandone lo stato nel corso del progetto.

3.2.1. Norme e strumenti della gestione di configurazione

3.2.1.1. Strumenti a supporto

Il gruppo utilizza i seguenti strumenti per garantire l'integrità e la tracciabilità dei prodotti:

Strumento	Descrizione e Utilizzo
Git$_G$	Sistema di controllo di versione, utilizzato per tracciare la storia delle modifiche di documenti, codice sorgente e script di *automazione*$_G$.
GitHub$_G$	Gestisce le Pull Request e le *pipeline*$_G$ di verifica create tramite *GitHub*$_G$ Actions.
Jira$_G$	Ogni modifica alla configurazione viene tracciata da un *Task*$_G$ su *Jira*$_G$ Board, contenente stime temporali, consuntivi e ruoli coinvolti.
`notipdo`	Automatizza la validazione, la compilazione dei documenti *Typst*$_G$ e la generazione del sito statico, garantendo build coerenti.

3.2.1.2. Elementi di Configurazione

Vengono sottoposti a controllo di versione e configurazione i seguenti elementi:
- **Documentazione**: Tutti i file sorgente `.typ` e gli *asset*$_G$ contenuti nella directory `docs/`;
- **Sito Web**: I *template*$_G$ e gli script per la generazione del sito statico contenuti nella directory `site/`;
- **Dizionari**: I file di dizionario personalizzati (es. `.hunspell/it_IT.dic`) utilizzati per il controllo ortografico;
- **Schemi di validazione**: File contenuti all'interno della directory `.schemas/`;
- ***Automazione***$_G$: Il codice sorgente del tool `notipdo` e i workflow di *GitHub*$_G$ (`.github`$_G$`/workflows/`) utilizzati nella chiusura delle *PR*$_G$ e rilascio del sito web.

3.2.1.3. Strategia dei *Repository*$_G$

Il gruppo adotta una strategia Multi-repo per garantire una netta separazione delle responsabilità e mantenere lineare la cronologia dei contributi. La struttura è così suddivisa:

- ***Repository Documentale***$_G$: Contiene la documentazione di progetto, il sito web e il tool `notipdo`;
- ***Repository***$_G$ ***PoC***$_G$: Contiene il codice sorgente per la Technology *Baseline*$_G$ (Proof of Concept);
- ***Repository***$_G$ ***MVP***$_G$: Conterrà il codice sorgente del prodotto richiesto dall'azienda proponente (Minimum Viable Product).

3.2.1.4. *Baseline*$_G$

Una *Baseline*$_G$ rappresenta una versione stabile e approvata della configurazione. Nella *repository documentale*$_G$, le *baseline*$_G$ sono definite a livello strutturale tramite le directory di primo livello in `docs/`:

- `11-candidatura`: *Baseline*$_G$ di Candidatura;
- `12-rtb`$_G$: Requirements and Technology *Baseline*$_G$.

Ogni modifica a una *baseline*$_G$ passata (conclusa) è vietata se non previa autorizzazione esplicita del Responsabile e bloccata dalle automazioni di verifica.

3.2.1.5. Configurazione *Task*$_G$ *Jira*$_G$

Ogni modifica alla configurazione deve essere associata a un *Task*$_G$ su *Jira*$_G$. È obbligatorio compilare i seguenti campi per garantire un tracciamento delle risorse corretto:

- **Autore**: Chi esegue la modifica;
- **Ruolo**: Il ruolo ricoperto durante l'attività (es. Analista, *Verificatore*$_G$, ...);
- **Time Tracking**:
  - *Original Estimate*: Tempo stimato prima di iniziare l'attività;
  - *Time Spent*: Tempo effettivamente impiegato (da aggiornare a fine attività).

3.2.1.6. Restrizioni e Check

Per garantire che solo configurazioni verificate confluiscano nel ramo principale, vengono applicate le seguenti regole su *GitHub*$_G$):

- ***Merge***$_G$ **Restriction**: Il *merge*$_G$ è consentito solo tramite Pull Request approvata da almeno un membro del team che in quel momento sta ricoprendo il ruolo di ***Verificatore***$_G$.
- **Status Check**: Il *merge*$_G$ è bloccato se i check automatici (`pr`$_G$`-check-n-build`) falliscono, indicando cosa correggere per allinearsi alle norme finora descritte.

3.2.2. Attività del processo

3.2.2.1. Identificazione della Configurazione

Ruoli	Amministratore, Responsabile
Riferimenti	• Sezione 3.2.1.2 Elementi di Configurazione • Sezione 3.1.1.2 Struttura del *repository*$_G$ • Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`
Input	Nuovi artefatti da produrre

Output Struttura di directory conforme

Procedura
1. **Collocazione**: Ogni nuovo documento o script deve essere collocato nella directory corretta secondo la struttura definita dal processo precedente.
2. ***Metadati*$_G$**: I documenti devono essere accompagnati dal relativo file `.meta.yaml`$_G$ conforme allo schema `.schemas/meta.schema.json`$_G$ per garantire la validazione automatica.
3. **Verifica Strutturale**: Eseguire `notipdo check` per verificare che la nuova struttura sia conforme alle regole del progetto.

3.2.2.2. Controllo della Configurazione

Ruoli Autore (vedi Sezione 3.1.1.10 Matrice Documento - Ruolo autore), *Verificatore*$_G$

Riferimenti
- Sezione 3.2.1.5 Configurazione *Task*$_G$ *Jira*$_G$
- Sezione 3.1.1.6 Gestione *Git*$_G$: Branching e *Commit*$_G$
- Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`

Input Necessità di modifica (*Task*$_G$ o Bug)

Output Configurazione aggiornata su `main`$_G$

Procedura
1. **Creazione *Task*$_G$**: Apertura di un ticket su *Jira*$_G$ Board specificando la stima temporale e il ruolo.
2. **Branching e Implementazione**: Creazione del *branch*$_G$ secondo nomenclatura (es. `NT-10-descrizione`) e implementazione della modifica.
3. **Verifica Automatica**: All'apertura della Pull Request, il workflow `pr`$_G$`-check-n-build` esegue automaticamente:
   - `notipdo check pr`$_G$: Validazione sintattica, controllo ortografico (Hunspell) e validazione schemi;
   - `notipdo build changes`: Compilazione di anteprima dei soli artefatti modificati.
4. **Review**: Un *Verificatore*$_G$ controlla le modifiche e richiede eventuali correzioni.
5. **Approvazione Tecnica**: Se la verifica è superata, viene applicata la label `merge`$_G$`-ready`. Questo innesca il workflow `pr`$_G$`-merge`$_G$`-ready` che conferma la mergeability della *PR*$_G$.
6. **Consuntivazione**: Registrazione delle ore effettive su *Jira*$_G$ (*Time Spent*) e chiusura del *Task*$_G$.

3.2.2.3. Registrazione dello Stato della Configurazione

Ruoli Amministratore

Riferimenti
- Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`
- Sezione 3.2.1.4 *Baseline*$_G$

Input Decisione di pubblicazione

Output Sito Web aggiornato, *Changelog*$_G$

Procedura
1. ***Automazione*$_G$ e Snapshot**: L'Amministratore avvia manualmente il workflow `deploy`$_G$`.yml` (dispatch). L'operazione è volutamente manuale in quanto la registrazione dello stato deve essere una operazione voluta e non esegui-

bile accidentalmente. Il sistema esegue `notipdo generate site`, generando la documentazione statica pubblicandoli su *GitHub*$_G$ Pages come snapshot ufficiale e immutabile della configurazione in quel preciso momento.

3.2.2.3.1. *Note*

Questa attività garantisce la visibilità immediata e pubblica dello stato corrente del progetto a tutti gli stakeholder tramite *automazione*$_G$.

3.2.2.4. Valutazione della Configurazione

Ruoli	Responsabile, Analista
Riferimenti	• Sezione 3.2.1.4 *Baseline*$_G$ • Sezione 3.2.1.5 Configurazione *Task*$_G$ *Jira*$_G$
Input	Rilascio di *Baseline*$_G$ imminente
Output	Matrice di Tracciamento, Report di *Audit*$_G$
Procedura	1. ***Audit*$_G$ dei Configuration Item**: Verifica l'integrità e la presenza fisica di tutti i componenti previsti per il rilascio, assicurandosi che ogni elemento sia correttamente versionato e coerente con lo stato descritto.

3.3. Accertamento della Qualità

Il processo di **Accertamento della Qualità** (QA) ha lo scopo di garantire che i processi di lavoro e i prodotti realizzati siano conformi agli standard stabiliti e soddisfino gli obiettivi di qualità. In particolare, l'Accertamento della Qualità si concentra sul monitoraggio continuo e sul miglioramento dei processi per prevenire l'introduzione di difetti, ponendo maggiore attenzione nei confronti del processo rispetto al prodotto.

3.3.1. Norme e strumenti per l'accertamento della qualità

3.3.1.1. Modello *PDCA*$_G$

Il gruppo adotta il modello iterativo **Plan-Do-Check-Act** per la gestione della qualità:

- **Plan**: Pianificare gli obiettivi di qualità e i processi per raggiungerli (es. definizione delle metriche nel Piano di Qualifica);
- **Do**: Eseguire i processi secondo le norme stabilite;
- **Check**: Monitorare e misurare i processi e i prodotti rispetto agli obiettivi tramite l'ausilio di *Dashboard*$_G$;
- **Act**: Adottare azioni correttive per colmare i gap rilevati e migliorare le prestazioni future.

3.3.1.2. Gestione delle Metriche

La qualità viene valutata quantitativamente tramite metriche definite nel **Piano di Qualifica**. Le metriche si dividono in:

- **Metriche di Processo**: Misurano l'efficienza del metodo di lavoro in utilizzo in un dato momento (es. Earned Value). La fonte dati principale risulta essere ***Jira*$_G$**;
- **Metriche di Prodotto**: Misurano la qualità degli artefatti (es. Indice Gulpease). Le metriche vengono raccolte e storicizzate al termine di ogni *Sprint*$_G$.

3.3.1.3. Strumenti di Monitoraggio

Il monitoraggio della qualità è supportato dai seguenti strumenti:

- ***Jira*$_G$ *Dashboard*$_G$**: Per la visualizzazione in tempo reale delle metriche di processo che permettono di avere un'idea precisa di come il gruppo stia procedendo. In particolare tramite l'utilizzo di: Burndown Chart, Velocity Chart, distribuzione del carico di lavoro;
- ***GitHub*$_G$ Actions & `notipdo`**: Per la raccolta automatica delle metriche di prodotto. I log delle *pipeline*$_G$ costituiscono la prova oggettiva del superamento dei controlli implementati fino a quel momento.

3.3.2. Attività del processo

3.3.2.1. Definizione e Evoluzione del Sistema di Qualità

Ruoli Amministratore

Riferimenti
- Sezione 3.3.1.1 Modello *PDCA*$_G$
- Sezione 3.3.1.2 Gestione delle Metriche

Input Avvio progetto, esiti delle *Retrospective*$_G$, non conformità rilevate

Output Norme di Progetto revisionate, *pipeline*$_G$ di verifica aggiornate

Procedura
1. **Formalizzazione degli Standard**: Definire e documentare le convenzioni per la codifica, la stesura dei documenti e le modalità operative del gruppo, aggiornando il presente documento.
2. ***Automazione*$_G$ dei Controlli**: Predisporre verifiche automatiche cercando di prevenire gli errori alla fonte, riducendo quindi le possibilità di un mancato rispetto delle norme.

3.3.2.2. Accertamento della Qualità del Prodotto

Ruoli *Verificatore*$_G$

Riferimenti
- Sezione 3.3.1.2 Gestione delle Metriche
- Sezione 3.3.1.3 Strumenti di Monitoraggio

Input Prodotti in rilascio (Codice, Documenti), Report di Verifica

Output Report di Qualità del Prodotto, Non conformità rilevate

Procedura
1. **Verifica delle Metriche**: Confrontare le metriche di prodotto raccolte automaticamente (es. Code Coverage, Indice Gulpease) con le soglie di accettabilità definite nel Piano di Qualifica.
2. **Controllo di Conformità**: Accertarsi che tutti gli artefatti siano stati sottoposti alle attività di Verifica obbligatorie e che non vi siano difetti bloccanti aperti.

3.3.2.3. Accertamento della Qualità del Processo

Ruoli Responsabile, Amministratore

Riferimenti
- Sezione 3.3.1.1 Modello *PDCA*$_G$
- Sezione 3.3.1.2 Gestione delle Metriche

Input Dati di processo (*Jira*$_G$, *Git*$_G$ logs), Svolgimento delle attività

Output Report di Qualità del Processo, Azioni correttive

Procedura
1. ***Audit*$_G$ dei Processi**: Verificare periodicamente che gli strumenti siano usati correttamente.
2. **Analisi delle Performance**: Valutare le metriche di processo (es. Earned Value, Velocity) per identificare inefficienze nel metodo di lavoro.
3. **Miglioramento Continuo**: Attuare le azioni correttive (Act) emerse dalle retrospettive per aggiornare i processi e prevenire il ripetersi delle eventuali non conformità rilevate.

3.3.2.3.1. *Note*
Attività prevista dallo standard. Si assicura che il team stia lavorando secondo le regole definite nelle Norme di Progetto.

3.4. Processo di Verifica
Il processo di **Verifica** ha lo scopo di confermare che ogni prodotto del ciclo di vita soddisfi i requisiti specificati e sia conforme agli standard di qualità interni fissati. L'obiettivo è quindi cercare di rispondere alla domanda: *«Stiamo costruendo il prodotto nel modo giusto?»* («Did we build the system right?»).

Gli esiti di questo processo, inclusi i risultati dei test e le metriche raccolte, saranno riportati nel **Piano di Qualifica**.

3.4.1. Norme e strumenti di verifica

3.4.1.1. Analisi Statica (Ispezione)
L'analisi statica viene effettuata senza l'esecuzione del prodotto software. Il metodo adottato è l'**Ispezione**:
- **Documentazione**: Verifica della correttezza ortografica, sintattica e contenutistica. Per la parte maggiormente strutturale e ortografica, l'ispezione è supportata dal tool `notipdo`.
- **Codice**: Revisione del codice tramite Pull Request. Il *verificatore*$_G$ controlla l'aderenza agli standard di codifica e l'assenza di difetti logici evidenti. Riguardo questa sezione seguiranno aggiornamenti dopo il raggiungimento della Requirements and Technology *Baseline*$_G$.

3.4.1.1.1. *Note*
Perché l'Ispezione? Rispetto al *Walkthrough*, l'Ispezione è un metodo più rigoroso che utilizza liste di controllo (checklist) per individuare errori specifici.

3.4.1.2. Analisi Dinamica (Testing)
L'analisi dinamica richiede l'esecuzione del codice per rilevare malfunzionamenti (*failure*) causati da difetti (*fault*). I test devono essere **ripetibili** e, ove possibile, **automatizzati**. Il gruppo adotta la classificazione dei test definita nel Piano di Qualifica:
- **Test di Unità (*Unit Testing*$_G$)**: Verifica delle singole unità di codice in isolamento. Si suddividono in:
  - *Funzionali (Black-box)*: Verifica input/output senza dare alcun peso alla struttura interna del codice.
  - *Strutturali (White-box)*: Verifica della logica interna e copertura dei percorsi da esso intrapresi.

- **Test di Integrazione**: Verifica il modo in cui collaborano le unità del nostro sistema.
- **Test di Sistema**: Verifica del comportamento dell'intero sistema rispetto ai requisiti funzionali e non funzionali.
- **Test di Regressione**: Riesecuzione dei test esistenti dopo delle modifiche per garantire che non siano stati introdotti nuovi difetti non previsti.

3.4.1.3. Nomenclatura dei Test

Ogni test definito nel Piano di Qualifica deve essere identificato univocamente secondo la nomenclatura:

`T-{Tipo}-{ID}`

Dove:

- `{Tipo}` indica la categoria:
  - `U`: Test di Unità;
  - `I`: Test di Integrazione;
  - `S`: Test di Sistema;
- `{ID}`: Numero progressivo univoco per tipologia.

Ogni test deve avere uno stato tracciato: *Non Implementato (NI)*, *Implementato (I)*, *Superato (S)*.

3.4.2. Attività del processo

3.4.2.1. Esecuzione della Verifica Documentale

Ruoli	*Verificatore*$_G$
Riferimenti	• Sezione 3.4.1.1 Analisi Statica (Ispezione) • Sezione 3.1.1.8 Utilizzo del tool di *automazione*$_G$ `notipdo`
Input	Documento in stato di «Verifica» (*PR*$_G$ aperta)
Output	Documento approvato o Richiesta modifiche
Procedura	1. **Check Preliminare**: L'autore si assicura che il documento superi i controlli automatici di `notipdo`, validando quindi struttura e spellcheck. 2. **Ispezione**: Il *Verificatore*$_G$ effettua una lettura completa del documento (o delle differenze nella *PR*$_G$) verificando: • Chiarezza espositiva; • Coerenza con quanto riportato all'interno del *changelog*$_G$ della modifica; • Integrità dei riferimenti: tutti i link a documenti esterni o interni sono funzionanti, puntano a versioni appropriate e contengono effettivamente l'informazione a cui il testo rimanda; • Assenza di errori residui non rilevati dagli strumenti automatici.

3.5. Processo di Validazione

Il processo di **Validazione** ha lo scopo di confermare che il sistema software realizzato soddisfi le esigenze specifiche dell'utente e sia idoneo all'utilizzo nel suo contesto operativo reale. L'obiettivo è rispondere alla domanda: *«Stiamo costruendo il prodotto giusto?»* («Did we build the right system?»).

Mentre la Verifica si concentrava sulla correttezza tecnica cercando di assicurare l'assenza di errori, la Validazione si concentra sulla corrispondenza con le aspettative del proponente (M31) e sui requisiti funzionali e qualitativi definiti nell'Analisi dei Requisiti.

3.5.1. Norme e strumenti di validazione

3.5.1.1. Tracciamento dei Requisiti

Per garantire la completezza del prodotto, deve essere mantenuta una «tracciabilità bidirezionale» nella Matrice di Tracciamento:

- **Fonte - Requisito**: Ogni esigenza espressa nel *capitolato*$_G$ o nei verbali esterni deve essere mappata su uno o più requisiti formali.
- **Requisito - Test**: Ogni Requisito implementato (sia esso Obbligatorio, Desiderabile o Opzionale) deve essere collegato a uno specifico Test di Sistema o di Accettazione che ne certifichi il soddisfacimento.

3.5.1.2. Test di Accettazione

I Test di Accettazione verificano il comportamento del sistema completo simulando scenari d'uso reali. Possiamo distinguerli in test:

- Eseguiti internamente dal team di sviluppo prima del rilascio, simulando l'utente finale per validare i flussi operativi.
- Eseguito congiuntamente al proponente durante le revisioni di avanzamento per ottenere l'approvazione formale del lavoro svolto.

3.5.2. Attività del processo

3.5.2.1. Validazione dei Requisiti (Analisi)

Ruoli	Analista, Responsabile
Riferimenti	• Sezione 3.5.1.1 Tracciamento dei Requisiti
Input	Specifica dei Requisiti, Codice Testato
Output	Matrice di Tracciamento aggiornata
Procedura	1. **Verifica Copertura**: Controllare che tutti i requisiti previsti per la *Baseline*$_G$ corrente siano marcati come «Soddisfatti» nella matrice di tracciamento e che non vi siano funzionalità richieste mancanti. 2. **Associazione Test**: Verificare che per ogni requisito soddisfatto esista almeno un test con esito positivo, garantendo che ogni funzionalità dichiarata sia effettivamente dimostrabile.

4. Processi organizzativi

I **processi organizzativi**, secondo quanto definito dallo standard ***ISO/IEC 12207:1995***$_G$, comprendono le attività di supporto e coordinamento che, pur non partecipando direttamente alla produzione del software, contribuiscono in modo significativo al buon esito e alla gestione complessiva del progetto.

Essi garantiscono che il lavoro del gruppo sia strutturato, monitorato e migliorato in modo continuo, favorendo la qualità del prodotto e l'efficienza dei processi.

All'interno del presente progetto, si individuano i seguenti processi organizzativi principali:
- **Gestione dei Processi**;
- **Processo di Infrastruttura**;
- **Processo di Miglioramento**;
- **Processo di Formazione**.

4.1. Gestione dei processi

In conformità alla norma ***ISO/IEC 12207:1995***$_G$, la Gestione dei Processi ha l'obiettivo di pianificare, controllare e coordinare le attività del gruppo durante l'intero ciclo di vita del progetto. Essa comprende, in particolare, la definizione dei ruoli, la pianificazione delle attività e il monitoraggio dell'avanzamento.

4.1.1. Norme e strumenti del processo di gestione

4.1.1.1. Definizione dei ruoli

Al fine di separare le attività da svolgere per ambito di competenza, il gruppo ha definito i seguenti ruoli:
- **Responsabile**: coordina il gruppo di lavoro per tutta la durata del progetto, pianificando le attività, gestendo tempi, costi e rischi, e rappresentando il team NoTIP verso l'esterno;
- **Amministratore**: gestisce e mantiene l'ambiente di lavoro e gli strumenti operativi del team, assicurando efficienza, la corretta configurazione del prodotto e la conformità alle norme di progetto;
- **Analista**: interpreta le esigenze del committente, raccoglie e formalizza i requisiti, redigendo lo Studio di Fattibilità e l'Analisi dei Requisiti che guidano la progettazione;
- **Progettista**: trasforma i requisiti definiti dall'analista in un'architettura tecnica coerente, definendo componenti, interazioni e tecnologie adottate, motivandole;
- **Programmatore**: implementa le soluzioni progettate, scrivendo codice conforme alle specifiche, realizzando test di verifica e contribuendo alla documentazione tecnica e utente;
- ***Verificatore***$_G$: controlla la qualità e la correttezza dei prodotti e dei processi, assicurando che le attività rispettino gli standard stabiliti e documentando i risultati.

4.1.1.2. Criteri di assegnazione dei ruoli

L'assegnazione dei ruoli viene definita durante ogni *Sprint Planning*$_G$, in base ai seguenti principi:
- ***Sprint Planning***$_G$: stima del tempo richiesto per ruolo dai *task*$_G$ in base agli obiettivi dello *sprint*$_G$, dipendenti dalla componente del progetto in sviluppo;
- **Disponibilità personale**: numero di ore produttive che la persona potrà dedicare allo *sprint*$_G$, generalmente tra le 7 e le 15 produttive;

- **Monitoraggio ore produttive**: il Responsabile aggiorna la rotazione dei ruoli valutando il numero di ore ricoperte da ogni persona per ciascun ruolo, monitorando il rispetto dei principi qui elencati e ripartendo eventuali carichi distribuiti non equamente;
- **Assegnazione basata sulle competenze**: per velocizzare il progresso nelle fasi iniziali, è possibile assegnare ruoli in cui i componenti hanno più esperienza, dando priorità a chi possiede competenze pregresse. Tutti i componenti dovranno comunque svolgere, a parità di ruolo, un numero simile di ore, come riportato nella Dichiarazione di Impegni v1.0.0.

4.1.1.3. Riunioni

Le riunioni rappresentano i momenti formali di sincronizzazione e aggiornamento del team. Si distinguono in:

- **Riunioni interne**: si svolgono periodicamente (di norma all'apertura e alla chiusura di ogni *Sprint*$_G$) per coordinare le attività, verificare lo stato di avanzamento e gestire la rotazione dei ruoli. Consentono al Responsabile di monitorare la situazione del gruppo, mitigare criticità e preparare le interazioni con gli stakeholder esterni;
- **Riunioni esterne**: seguono una pianificazione periodica, coincidendo con i ricevimenti dei docenti e gli incontri con l'azienda proponente *M31*. Il gruppo, rappresentato dal Responsabile, espone l'avanzamento del progetto e risolve dubbi o ambiguità sui requisiti;
- **Riunioni straordinarie**: convocate qualora emergano necessità impreviste o urgenza di chiarimenti immediati.

A garanzia della tracciabilità, i temi trattati e le decisioni assunte devono essere documentati tramite appositi verbali (interni o esterni). Per gli strumenti utilizzati si rimanda alla sezione Sezione 4.2.

4.1.1.4. Comunicazioni

I flussi comunicativi sono strutturati per garantire efficienza nel coordinamento interno e nelle relazioni con gli stakeholder.

Per la **comunicazione interna** vengono adottati due canali:

- ***Discord***$_G$: designato per le attività sincrone, incluse le riunioni formali di inizio e fine *Sprint*$_G$, sfruttando le funzionalità di condivisione audio/video;
- **Telegram**: utilizzato per il coordinamento operativo giornaliero, notifiche rapide e comunicazioni a carattere informale.

Per la **comunicazione esterna**, il canale ufficiale è la posta elettronica. Il Responsabile gestisce i contatti con l'azienda proponente e i docenti tramite l'indirizzo istituzionale del gruppo: `notip.swe@gmail.com`.

Per le specifiche tecniche relative agli strumenti citati si rimanda alla sezione Sezione 4.2.

4.1.2. Attività del processo

4.1.2.1. *Sprint Planning*$_G$

Ruoli Responsabile

Riferimenti
- Sezione 4.1.1.1 Definizione dei ruoli
- Sezione 4.1.1.2 Criteri di assegnazione dei ruoli

Input *Backlog*$_G$ di progetto, consuntivo dello *sprint*$_G$ precedente

Output Ruoli assegnati, *task*$_G$ distribuiti, stime aggiornate

Procedura
1. **Stima dei *task*$_G$**: Il Responsabile analizza gli obiettivi dello *sprint*$_G$ e stima il tempo richiesto per ruolo, tenendo conto della disponibilità personale dichiarata da ciascun membro.
2. **Assegnazione dei ruoli**: I ruoli vengono assegnati applicando i criteri definiti nelle norme, con attenzione al bilanciamento delle ore cumulative per ruolo e alle competenze dei componenti.
3. **Apertura work items**: I *task*$_G$ vengono aperti su *Jira*$_G$ e assegnati ai rispettivi componenti, con stime di tempo e descrizione degli obiettivi.

4.1.2.2. Gestione delle riunioni

Ruoli Responsabile

Riferimenti
- Sezione 4.1.1.3 Riunioni

Input Necessità di sincronizzazione o scadenza di *Sprint*$_G$

Output Verbale interno o esterno

Procedura
1. **Convocazione**: Il Responsabile convoca la riunione comunicando data, ora, piattaforma e ordine del giorno ai partecipanti tramite i canali definiti nelle norme di comunicazione.
2. **Svolgimento**: La riunione si svolge seguendo l'ordine del giorno. Il Responsabile modera la discussione e raccoglie le decisioni prese.
3. **Verbalizzazione**: Al termine, il Responsabile redige il verbale (interno o esterno) seguendo le norme del processo di documentazione.

4.1.2.3. Monitoraggio e controllo

Ruoli Responsabile

Riferimenti
- Sezione 4.1.1.2 Criteri di assegnazione dei ruoli

Input Avanzamento dei work items su *Jira*$_G$, ore consuntivate

Output Consuntivo di *sprint*$_G$, eventuali azioni correttive

Procedura
1. **Monitoraggio avanzamento**: Il Responsabile verifica periodicamente lo stato dei *task*$_G$ su *Jira*$_G$, controllando che le attività procedano nei tempi previsti e che il budget residuo sia allineato alle stime.
2. **Gestione delle criticità**: In caso di ritardi o problemi, il Responsabile attiva le strategie di mitigazione definite nel Piano di Progetto e, se necessario, convoca una riunione straordinaria.
3. **Aggiornamento rotazione ruoli**: A fine *sprint*$_G$, il Responsabile aggiorna il conteggio delle ore per ruolo e prepara le indicazioni per il successivo *Sprint Planning*$_G$.

4.2. Processo di infrastruttura

Il processo di Infrastruttura ha lo scopo di definire, predisporre e mantenere l'ambiente tecnico necessario per abilitare e supportare l'esecuzione di tutti gli altri processi di progetto.

Per supportare le attività di gestione, sviluppo e coordinamento, il gruppo ha adottato il seguente set di strumenti: Telegram, Google Mail, Microsoft Teams, Zoom, *Jira*$_G$, *Git*$_G$, *GitHub*$_G$, *Discord*$_G$, *Typst*$_G$ e Google Sheets. Di seguito si trovano le norme relative a ciascuno.

4.2.1. Norme e strumenti del processo di infrastruttura

4.2.1.1. Strumenti di comunicazione

Il gruppo adotta i seguenti strumenti per la comunicazione:

- **Telegram**: programma di messaggistica utilizzato per aggiornarsi giornalmente sui progressi del progetto e per qualsiasi tipo di comunicazione, nel quale è anche possibile fissare i messaggi più importanti in un determinato periodo;
- **Google Mail**: servizio di posta elettronica utilizzato per gestire le comunicazioni esterne al gruppo. A tal proposito è stata creata una mail dedicata al team chiamata `notip.swe@gmail.com`. All'interno di Google Mail è anche collegato un Calendario che registra in maniera autonoma a partire dalle mail ricevute i prossimi incontri a cui il team dovrà partecipare;
- **Microsoft Teams**: piattaforma adottata per lo svolgimento degli incontri a distanza con l'azienda proponente *M31*;
- **Zoom**: piattaforma designata per i ricevimenti e la discussione dei **Diari di Bordo** con il docente referente (Prof. Vardanega). Al fine di garantire l'indipendenza operativa dai singoli membri, il gruppo utilizza un'utenza condivisa, registrata direttamente con l'indirizzo e-mail ufficiale del progetto;
- ***Discord*$_G$**: programma di messaggistica utilizzato per svolgere riunioni interne in modalità virtuale e anche per lavorare insieme in chiamata sugli stessi documenti.

4.2.1.2. Strumenti di gestione e *versionamento*$_G$

Il gruppo adotta i seguenti strumenti per la gestione del progetto e il *versionamento*$_G$:

- ***Jira*$_G$**: piattaforma di Project Management selezionata per adottare correttamente la metodologia *Agile*$_G$. Lo strumento viene utilizzato per la pianificazione degli *Sprint*$_G$, la gestione del *Backlog*$_G$ e il tracciamento delle *Task*$_G$. Grazie all'integrazione con *Git*$_G$, *Jira*$_G$ funge da punto centrale di controllo per garantire la tracciabilità tra le attività e il codice prodotto;
- ***Git*$_G$**: strumento adottato per gestire il *versionamento*$_G$. Esso funge da «Single Source of Truth» per l'intero ciclo di vita del prodotto, garantendo l'integrità, la tracciabilità e la disponibilità storica di ogni artefatto. L'utilizzo di *Git*$_G$ è strettamente vincolato all'hosting remoto su ***GitHub*$_G$**, che funge da *repository*$_G$ centrale autoritativo;
- ***GitHub*$_G$**: piattaforma selezionata per la gestione del *repository*$_G$ *Git*$_G$ e il supporto allo sviluppo collaborativo. Permette la sincronizzazione del lavoro tra i membri del team, garantendo la disponibilità e l'integrità degli artefatti di progetto.

4.2.1.3. Utilizzo di *GitHub*$_G$

L'adozione di *GitHub*$_G$ è trasversale a diversi processi di supporto. Di seguito si riporta la mappatura delle funzionalità della piattaforma sulle norme di progetto:

- **Collaborazione e Verifica (Pull Request):** *GitHub*$_G$ abilita il lavoro asincrono tramite il meccanismo delle **Pull Request** (*PR*$_G$). Le *PR*$_G$ costituiscono il punto di controllo obbligatorio (*Quality Gate*$_G$) per la revisione del codice e la risoluzione dei conflitti prima

dell'integrazione nel ramo stabile. Il processo operativo di verifica tramite *PR*$_G$ è dettagliato nella sezione Sezione 3.1.2.3;
- **Organizzazione degli Artefatti:** La struttura delle directory ospitate sulla piattaforma non è arbitraria, ma deve rispecchiare fedelmente l'architettura informativa definita in Sezione 3.1.1.2;
- **Gestione della Configurazione:** Le politiche di interazione con il *repository*$_G$ remoto, incluse le strategie di **branching** e la sintassi dei ***commit*$_G$ messages**, devono conformarsi rigorosamente a quanto stabilito in Sezione 3.1.1.6 e nella norma di integrazione Sezione 4.2.1.9 Integrazione *Jira*$_G$-*Git*$_G$.

4.2.1.4. Strumenti di documentazione

Il gruppo adotta ***Typst*$_G$** come sistema per la redazione di tutta la documentazione. L'approccio **Docs as Code** garantisce coerenza stilistica e validazione strutturale tramite l'uso di librerie e *template*$_G$ condivisi.

4.2.1.5. Strumenti per la raccolta e l'analisi delle metriche

Per il monitoraggio quantitativo del progetto, la valutazione della qualità e la stesura dei grafici, il gruppo si affida a:
- **Google Sheets**: utilizzato come centro dell'inserimento, la classificazione e la storicizzazione di tutte le metriche (di processo, di prodotto e di qualità). Questo ci consente di elaborare i dati raccolti tramite script automatici e generando automaticamente dei grafici per i report;
- ***Git*$_G$ e *GitHub*$_G$**: oltre alla gestione del codice, vengono attivamente interrogati per estrarre metriche fondamentali legate allo sviluppo (come la qualità dei *commit*$_G$ o i tempi di risoluzione delle Pull Request), i cui risultati vengono poi aggregati, inviati ed analizzati su Google Sheets.

4.2.1.6. Identificazione *Jira*$_G$

Ogni attività censita su *Jira*$_G$ è rappresentata da una ***Issue*$_G$** dotata di una chiave univoca assegnata automaticamente dal sistema (es. `NT-XXX`). Al fine di garantire una corretta rendicontazione temporale e la qualità del prodotto, ogni *Task*$_G$ Madre deve essere scomposta in due sub-*task*$_G$ distinte:
- **Sub-*task*$_G$ di Esecuzione:** Utilizzata per tracciare le ore produttive impiegate nello svolgimento dell'attività;
- **Sub-*task*$_G$ di Verifica:** Utilizzata per tracciare le ore impiegate nelle attività di controllo.

4.2.1.6.1. *Note*

Sequenzialità degli ID: La creazione contestuale delle sub-*task*$_G$ garantisce che esse ricevano identificativi immediatamente successivi alla *Task*$_G$ Madre (es. Madre `NT-220` → Esecuzione `NT-221`, Verifica `NT-222`), preservando l'ordine logico e visivo nel *backlog*$_G$.

4.2.1.7. Workflow e Ciclo di vita *Jira*$_G$

Il ciclo di vita di ogni attività (***Task*$_G$ Madre**) è governato da una macchina a stati finiti che ne traccia l'evoluzione dalla presa in carico fino al rilascio. Il diagramma di riferimento è riportato in Figura 2.

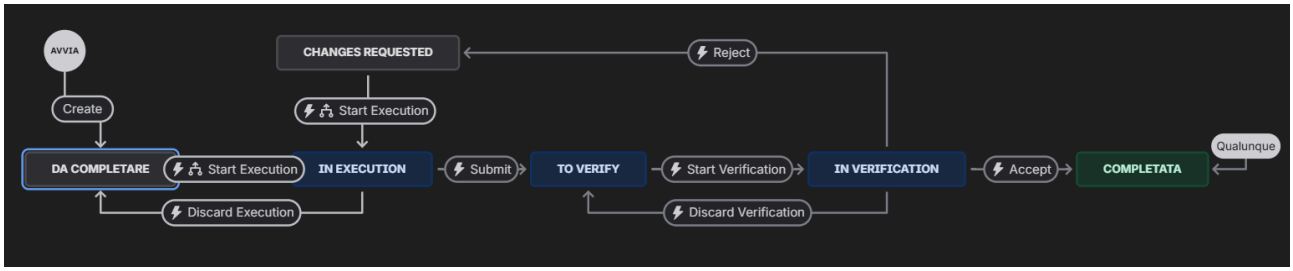

Figura 2: Workflow operativo della *Task*$_G$ Madre

Il percorso da seguire prevede le seguenti fasi:
- **Da Completare**: Dove la *task*$_G$ è stata creata ma non ancora avviata;
- **In Execution**: La *task*$_G$ è in fase di sviluppo;
- **To Verify**: L'attività attende di essere verificata;
- **In Verification**: L'attività è stata presa in carico dal *verificatore*$_G$;
- **Completata**: L'attività ha superato la verifica e può considerarsi conclusa;
- **Waiting**: Stato iniziale della Sub-*Task*$_G$ di Verifica, che passa allo stato **Da Completare** solo quando la Sub-*Task*$_G$ di Esecuzione passa allo stato **In Verification**;
- **Changes requested**: Stato della Sub-*Task*$_G$ di Esecuzione raggiungibile soltanto se il *Verificatore*$_G$ ha richiesto dei cambiamenti al lavoro svolto selezionando **Reject** nella Sub-*Task*$_G$ di Verifica quando essa era nello stato **In Verification**. Richiede che l'Autore del documento ritorni alla fase **In Execution** per poter così apportare le modifiche richieste.

Le transizioni invece sono:
- **Create**: La *task*$_G$ viene creata;
- **Start Execution**: Inizia lo sviluppo della *task*$_G$;
- **Discard Execution**: Regressione dallo stato "In Execution" allo stato "Da Completare";
- **Submit**: Passaggio dalla fase esecutiva a quella di verifica;
- **Start Verification**: Inizia l'attività di verifica del lavoro svolto;
- **Discard Verification**: Regressione dallo stato "In Verification" allo stato "To Verify";
- **Reject**: In caso di modifiche da apportare al lavoro svolto, la *task*$_G$ retrocede dallo stato "In Verification" allo stato "Changes Requested", richiedendo un intervento da parte dell'autore prima di poter ritornare alla fase di verifica.

4.2.1.7.1. *Note*

Sincronizzazione Automatica: L'evoluzione della *Task*$_G$ Madre è guidata automaticamente dallo stato delle sue Sub-*task*$_G$ (Esecuzione e Verifica), garantendo l'allineamento tra il lavoro svolto e lo stato riportato nel Project Management.

Quality Gate$_G$: Il workflow impone un vincolo bloccante: nessuna attività può raggiungere lo stato *Completata* senza aver superato con successo la fase di verifica formale, garantendo la qualità del prodotto in uscita.

4.2.1.8. Gestione delle Risorse *Jira*$_G$

Per la gestione delle risorse su *Jira*$_G$ abbiamo deciso di avere:
- **Assegnazione Singola:** Ogni entità (*Task*$_G$ Madre o Sub-*task*$_G$) deve avere sempre un solo assegnatario. Non è corretto lasciare *task*$_G$ in lavorazione (stati attivi diversi da "Da Completare") senza un responsabile assegnato;

- **Specifica del Ruolo:** È obbligatorio indicare il ruolo che il membro del gruppo ricopre durante lo svolgimento di quella specifica attività (es. `Programmatore`, `Verificatore`$_G$, `Analista`).

4.2.1.8.1. *Note*

L'assegnazione singola e specifica del ruolo è indispensabile per la rendicontazione delle ore. Poiché ogni membro dovrà ricoprire più ruoli all'interno del progetto (o anche all'interno dello stesso *sprint*$_G$), l'etichetta permette di calcolare correttamente il consumo delle risorse rispetto al preventivo, e di rendicontare nella maniera corretta le ore di lavoro svolte.

4.2.1.9. Integrazione *Jira*$_G$-*Git*$_G$

Jira$_G$ riceve aggiornamenti diretti dal *repository*$_G$ remoto. Per garantire il funzionamento del tracciamento, è necessario rispettare rigorosamente la nomenclatura:

- **Branching:** Il nome di ogni *branch*$_G$ deve contenere l'id del *Task*$_G$ (es. `NT-67-norme-di-progetto-v-1-0-0`);
- ***Commit*$_G$ Message:** I messaggi devono seguire lo standard ***Conventional Commits***$_G$ e includere l'id del *task*$_G$.

4.2.1.9.1. *Note*

L'inclusione dell'id nel messaggio di *commit*$_G$ crea un collegamento automatico tra la *Issue*$_G$ su *Jira*$_G$ e il codice su *GitHub*$_G$ (*Smart Commits*$_G$). Questo garantisce la tracciabilità bidirezionale: dalla *Task*$_G$ è possibile risalire al codice che lo soddisfa, e dal codice è possibile risalire alla *task*$_G$ che lo ha generato. Per ulteriori informazioni o esempi sui *Conventional Commits*$_G$ si consiglia di visionare la seguente pagina ***Conventional Commits***$_G$ o di rileggere la Sezione 3.1.1.6.

4.2.1.10. *Dashboard*$_G$ e Metriche *Jira*$_G$

Jira$_G$ è configurato per tracciare automaticamente le metriche di processo attraverso una ***Dashboard*$_G$ di Progetto** condivisa. I principali indicatori monitorati sono:

- **Distribuzione delle Ore:** Grafici a torta e tabelle che mostrano le ore assegnate per persona e per ruolo e in quali *sprint*$_G$ sono state svolte. Questo permette di verificare il rispetto della rotazione dei ruoli e anche per capire quali *sprint*$_G$ sono stati i più produttivi;
- **Velocity Chart e Burndown Chart:** Il **Burndown Chart** confronta visivamente l'andamento ideale di consumo delle ore rispetto alle ore effettivamente consumate dal team, permettendo di identificare ritardi o anticipi rispetto alla scadenza. Il **Velocity Chart**, invece, storicizza la quantità di lavoro completata in ogni *Sprint*$_G$ precedente, fornendo un dato medio fondamentale per stimare con precisione la capacità produttiva futura del team.

4.2.1.11. Configurazione dell'Ambiente Locale *Git*$_G$

Ogni membro del team è tenuto a configurare il proprio ambiente locale prima del primo *commit*$_G$, rispettando i seguenti vincoli:

Il ramo principale deve essere denominato `main`$_G$ (e non `master`) per conformità con le policy del *repository*$_G$ remoto.

4.2.1.12. Politiche di Esclusione (.gitignore)

È severamente vietato effettuare *commit*$_G$ di file derivati o specifici dell'ambiente locale. Il *repository*$_G$ deve contenere nella radice un file `.gitignore` condiviso che escluda tassativamente:

- Cartelle di build e dist (es. `bin/`, `build/`, `dist/`);
- Dipendenze esterne (es. `node_modules/`, `venv/`, `target/`);
- File di configurazione (es. `.vscode/`, `.idea/`);
- Credenziali o file `.env`.

Nota: Non utilizzare mai `git`$_G$ `add --force` per aggirare queste regole senza previa approvazione del Responsabile.

4.2.1.12.1. *Note*

Il *versionamento*$_G$ di file binari generati, dipendenze scaricate o file di configurazione locali appesantisce il *repository*$_G$ e crea conflitti non risolvibili.

4.2.1.13. Libreria dei Processi (lib.typ)

Il file `lib.typ` espone le primitive fondamentali per la stesura delle Norme di Progetto. È obbligatorio utilizzare le seguenti funzioni:

- `#norm`: Definisce regole statiche o vincoli di progetto.
  - `title`: Identificativo univoco della norma;
  - `label`: Etichetta per i riferimenti incrociati (es. `<etichetta>`);
  - `rationale`: (Opzionale) Giustificazione o note esplicative che appariranno in un blocco «Note».
- `#activity`: Definisce procedure operative e flussi di lavoro.
  - `roles`: Elenco dei ruoli coinvolti, da selezionare esclusivamente dal dizionario `ROLES` (es. `ROLES.anal`, `ROLES.ver`);
  - `input` / `output`: Descrizione degli artefatti in ingresso e uscita;
  - `norms`: Lista delle label delle norme citate (es. `("norma-1", "norma-2")`);
  - `procedure`: Array di oggetti contenenti `name` (nome del passo) e `desc` (descrizione operativa).

4.2.1.13.1. *Note*

Standardizzazione: L'uso di funzioni dedicate per norme e attività vincola gli autori a definire tutti i *metadati*$_G$ necessari (ruoli, input/output, tracciabilità), rendendo la documentazione conforme agli standard di qualità.

4.2.1.14. Adozione dei *Template Typst*$_G$

Ogni tipologia di documento deve estendere il relativo **base *template***$_G$ per garantire la presenza delle sezioni obbligatorie:

- Documenti generici: Utilizzare `base_document.typ` per Analisi, Piani e Norme;
- Verbali: Utilizzare `apply-base-verbale` da `base_verbale.typ`. La discussione deve essere strutturata tramite la funzione `report-point`, definendo esplicitamente `discussion`, `decisions` e `actions`;
- Diari di Bordo: Utilizzare `apply-base-ddb` da `base_ddb.typ`, compilando le sezioni di risultati, obiettivi e difficoltà;
- Presentazioni: Utilizzare `base_slides.typ` per le slide di avanzamento (SAL).

4.2.1.14.1. *Note*

I *template*$_G$ astraggono la formattazione e la struttura obbligatoria (front-matter, *changelog*$_G$, indici), permettendo agli autori di concentrarsi sul contenuto.

4.2.1.15. Specifica tecnica dei Casi d'Uso

La definizione dei Casi d'Uso è vincolata all'utilizzo della funzione `#uc` (libreria `uc_lib.typ`), che richiede obbligatoriamente `id` e `title` univoci, l'uso di costanti tipizzate per gli attori, la specifica degli scenari (`main`$_G$`-scen` e `alt-scen` con `cond`), il contratto (`preconds`/`postconds`) e l'importazione dei diagrammi UML tramite `#uml-schema`.

4.2.1.16. Organizzazione dei Canali *Discord*$_G$

Il server è strutturato in diverse categorie:

- **Discussions:** Categoria dedicata alle decisioni asincrone.
  - `tech`: Per dubbi su tecnologie, librerie e condivisione di snippet di codice;
  - `management`: Per scadenze, organizzazione informale o per raggruppare in un unico luogo considerazioni su unico argomento.
- **Meetings:** Categoria per le riunioni.
  - Il canale testuale `meeting-notes` è riservato a brevi appunti o link condivisi durante la call.
- ***Cowork*$_G$:** Canali vocali dedicati al lavoro di gruppo informale.

4.2.1.17. Manutenzione dell'infrastruttura

A causa del continuo avanzamento del progetto, il gruppo è consapevole che l'infrastruttura subirà nel tempo cambiamenti e potrebbe causare possibili problemi. Per questo spetta all'Amministratore il compito della Manutenzione, aggiornando le funzionalità qualora errori o cambiamenti lo rendano necessario.

4.2.2. Attività del processo

4.2.2.1. Creazione e Pianificazione *Task*$_G$ su *Jira*$_G$

Ruoli Autore (vedi Sezione 3.1.1.10 Matrice Documento - Ruolo autore), *Verificatore*$_G$

Riferimenti
- Sezione 4.2.1.6 Identificazione *Jira*$_G$
- Sezione 4.2.1.8 Gestione delle Risorse *Jira*$_G$

Input Attività da svolgere da trasformare in *Task*$_G$

Output *Task*$_G$ correttamente realizzate e assegnate allo *Sprint*$_G$ *Backlog*$_G$

Procedura
1. **Creazione *Task*$_G$ Madre**: Chiunque debba svolgere una qualsiasi attività è tenuto a creare la relativa *Task*$_G$ madre nel *Backlog*$_G$ e assegnarla allo *Sprint*$_G$ corrente.
2. **Sub-*Task*$_G$**: Immediatamente per ogni *Task*$_G$ madre verranno create e associate in maniera automatica le Sub-*Task*$_G$ di Esecuzione e Verifica.
3. **Assegnazione**: A questo punto l'autore della *task*$_G$ deve assegnare a se stesso la sub-*task*$_G$ di Esecuzione e un altro membro del team dovrà assegnarsi alla sub-*task*$_G$ di Verifica. Tutto questo deve sempre essere fatto assegnando correttamente la Label del ruolo che varia in base all'attività da svolgere.

4. **Stima temporale**: Bisogna anche inserire una stima realistica del tempo necessario per completare l'attività. La stima deve essere fatta distintamente sia per la Sub-*Task*$_G$ di Esecuzione che per quella di Verifica.
5. **Avvio**: Adesso la *Task*$_G$ sarà stata correttamente inizializzata e assegnata e nello stato **Da Completare**.

4.2.2.2. Ciclo di Avanzamento *Task*$_G$ su *Jira*$_G$

Ruoli Autore (vedi Sezione 3.1.1.10 Matrice Documento - Ruolo autore), *Verificatore*$_G$

Riferimenti
- Sezione 4.2.1.7 Workflow e Ciclo di vita *Jira*$_G$
- Sezione 4.2.1.9 Integrazione *Jira*$_G$-*Git*$_G$

Input *Task*$_G$ in stato **Da Completare**

Output *Task*$_G$ in stato **Completata**

Procedura
1. **Sviluppo**: L'autore porta la Sub-*Task*$_G$ di esecuzione nello stato **In Execution**, crea il *branch*$_G$ ed effettua i *commit*$_G$ secondo le norme precedentemente descritte, fa avanzare la Sub-*Task*$_G$ allo stato **In Verification** e apre una *Pull-Request*$_G$ su *GitHub*$_G$. Verifica che la *Task*$_G$ Madre sia passata allo stato **To Verify** grazie alle automazioni.
2. **Verifica**: Prende in carico la Sub-*Task*$_G$ di Verifica e fa avanzare allo stato **In Corso**. Se la verifica va a buon fine, anche su *GitHub*$_G$ viene approvata la *Pull-Request*$_G$ e questa Sub-*Task*$_G$ viene fatta avanzare allo stato **Completata**. Nel caso in cui siano richieste delle modifiche, allora il *verificatore*$_G$ farà la Reject del lavoro svolto, mandando la Sub-*Task*$_G$ di Esecuzione allo stato **Changes Requested**, e la Sub-*Task*$_G$ di Verifica allo stato **Waiting**.
3. **Completamento**: Il *verificatore*$_G$ completerà la *Pull-Request*$_G$ su *GitHub*$_G$. La Sub-*Task*$_G$ di Verifica è **Completata**, e quindi anche la Sub-*Task*$_G$ di Esecuzione passerà allo stesso stato, rendendo infine anche la *Task*$_G$ madre completata.

4.2.2.2.1. *Note*

Le transizioni della *Task*$_G$ Madre sono automatiche. È tuttavia necessario attendere il completamento dell'*automazione*$_G$ (visibile dall'aggiornamento dell'interfaccia) prima di considerare conclusa l'operazione, per evitare problemi di allineamento dovuti alla latenza di *Jira*$_G$.

4.2.2.3. Setup iniziale dell'ambiente di *versionamento*$_G$

Ruoli Autore (vedi Sezione 3.1.1.10 Matrice Documento - Ruolo autore)

Riferimenti
- Sezione 4.2.1.11 Configurazione dell'Ambiente Locale *Git*$_G$
- Sezione 4.2.1.12 Politiche di Esclusione (.gitignore)

Input Nuova postazione di lavoro o nuovo membro del team

Output Ambiente *Git*$_G$ configurato e *repository*$_G$ clonato

Procedura
1. **Installazione**: Verificare l'installazione dell'ultima versione stabile di *Git*$_G$ tramite il comando `git --version`.
2. **Configurazione completa e Autenticazione**: Eseguire in sequenza le configurazioni di identità, tecniche e di sicurezza:

- Impostare nome e mail (uguale a *GitHub*$_G$): `git`$_G$ `config --global user.name "Nome Cognome"` `git`$_G$ `config --global user.email "email@dominio.it"`;
- Generazione della chiave SSH.

3. **Cloning**: Clonare il *repository*$_G$ utilizzando la stringa di connessione SSH.

4.2.3. Manutenzione

A causa del continuo avanzamento del progetto, il gruppo è consapevole che l'infrastruttura subirà nel tempo cambiamenti e potrebbe causare possibili problemi. Per questo spetta all'Amministratore il compito della Manutenzione, aggiornando le funzionalità qualora errori o cambiamenti lo rendano necessario.

4.3. Processo di Miglioramento

In conformità alla norma ***ISO/IEC 12207:1995***$_G$, il Processo di Miglioramento ha lo scopo di stabilire, valutare e ottimizzare i processi utilizzati durante l'intero ciclo di vita del software.

4.3.1. Norme e strumenti del processo di miglioramento

4.3.1.1. Ciclo *PDCA*$_G$

Il miglioramento di un processo giudicato inefficace o inefficiente avviene secondo il ciclo *PDCA*$_G$ (Plan-Do-Check-Act):

1. Identificazione del Problema;
2. Modifica della documentazione relativa al problema;
3. Allineamento del Team sulle modifiche effettuate;
4. Monitoraggio delle metriche per verificare il successo delle modifiche apportate al processo.

4.3.2. Attività del processo

4.3.2.1. Inizializzazione del Processo

Ruoli	Responsabile, Amministratore
Riferimenti	
Input	Necessità di definire i processi organizzativi
Output	Norme di Progetto
Procedura	1. **Definizione dei processi**: In questa fase si stabiliranno tutti i processi organizzativi che guideranno il progetto. Lo scopo del documento delle Norme di Progetto è proprio quello di fornire una base solida per la comprensione e l'istanziazione dei processi.

4.3.2.2. Valutazione del Processo

Ruoli	Responsabile, Amministratore
Riferimenti	
Input	Processi definiti
Output	Metriche di qualità del processo (vedi Sezione 6)

Procedura
1. **Definizione delle metriche**: Una volta che i processi saranno stati definiti, sarà necessario definire delle metriche appropriate e, sulla base dei dati prodotti da queste, valutare l'efficacia e l'efficienza dei processi. Tali metriche verranno esplicitate nella Sezione 6, dedicata alle metriche di qualità del processo.

4.3.2.3. Miglioramento del Processo

Ruoli Responsabile, Amministratore

Riferimenti
- Sezione 4.3.1.1 Ciclo *PDCA*$_G$

Input Dati raccolti nella fase di Valutazione

Output Documentazione aggiornata, Team allineato sulle modifiche

Procedura
1. **Applicazione del ciclo *PDCA*$_G$**: Sulla base dei dati raccolti nella fase di Valutazione, bisognerà individuare i processi risultati inadatti e applicare il ciclo *PDCA*$_G$ come definito nelle norme.

4.4. Processo di Formazione

Il **Processo di Formazione** ha lo scopo di mantenere i membri del gruppo aggiornati sulle tecnologie adottate e sulle procedure interne, facendo in modo che ogni componente acquisisca le competenze necessarie per operare efficacemente all'interno del progetto. Il gruppo ha individuato le seguenti tecnologie come oggetto di formazione:

- *Backend*$_G$: *Go*$_G$, *NestJS*$_G$, *NATS*$_G$ *JetStream*$_G$;
- *Frontend*$_G$: *Angular*$_G$;
- Infrastruttura: *Docker*$_G$;
- Documentazione: *Typst*$_G$;
- Gestione: *Jira*$_G$, *Git*$_G$, *GitHub*$_G$;
- Comunicazione: *Discord*$_G$, Telegram.

4.4.1. Norme e strumenti del processo di formazione

4.4.1.1. Auto-formazione e Consolidamento

La fonte primaria di formazione per le procedure è il presente documento (**Norme di Progetto**).
Ogni membro è tenuto a:
- Studiare autonomamente la documentazione ufficiale degli strumenti adottati (vedi Tabella Risorse);
- Consultare le guide interne (es. `README.md` dei *repository*$_G$);
- Consolidare le competenze tramite prove pratiche in ambiente locale prima di operare sul *repository*$_G$ remoto.

4.4.1.2. Apprendimento tra Pari

Per favorire la condivisione della conoscenza e gestire al meglio la rotazione dei ruoli, il gruppo adotta il confronto diretto:

- **Sessioni di spiegazione**: Quando viene introdotta una nuova tecnologia o procedura, il membro più esperto organizza una breve sessione di spiegazione sincrona (su *Discord*$_G$) per il resto del team;
- **Supporto alla rotazione**: Chi subentra in un ruolo deve confrontarsi con chi lo ha ricoperto precedentemente per ricevere consigli e «best practices» non codificabili formalmente.

4.4.1.3. Risorse per la formazione

Il gruppo ha selezionato le seguenti risorse di riferimento per lo studio:

Ambito	**Risorse Selezionate**
Go$_G$	• *Go*$_G$ by Example • Effective *Go*$_G$
NestJS$_G$	• *NestJS*$_G$ Documentation • *NestJs*$_G$ Course for Beginners
NATS$_G$	• *NATS*$_G$ by Example • *JetStream*$_G$ Documentation
Angular$_G$	• *Angular*$_G$ Docs • *Angular*$_G$ Playground
Docker$_G$	• *Docker*$_G$ Get Started
Typst$_G$	• Reference Guide
Git$_G$ & *GitHub*$_G$	• *GitHub*$_G$ Docs
Jira$_G$	• Atlassian *Jira*$_G$ Software Guide

Tabella 10: Risorse per la formazione tecnica

4.4.2. Attività del processo

4.4.2.1. Sviluppo di materiale per la formazione

Ruoli Amministratore

Riferimenti • Sezione 4.4.1.1 Auto-formazione e Consolidamento

Input Adozione di un nuovo strumento o necessità operativa

Output Materiale didattico o Lista risorse aggiornata

Procedura

1. **Selezione delle fonti**: L'Amministratore o un delegato identifica la documentazione ufficiale più pertinente per lo strumento scelto (es. Tutorial *Angular*$_G$ o *Go*$_G$).
2. **Produzione guide interne**: Qualora non sia presente documentazione (nel caso di tool proprietari) oppure la documentazione ufficiale sia troppo vasta, viene prodotta una guida sintetica con i comandi specifici per il workflow del gruppo.
3. **Diffusione**: Le risorse selezionate vengono aggiunte alla Tabella Risorse delle Norme di Progetto.

4.4.2.1.1. *Note*

Attività necessaria per centralizzare la conoscenza e ridurre il tempo di ricerca delle informazioni per i singoli membri.

4.4.2.2. Implementazione del piano per la formazione

Ruoli Tutto il Team

Riferimenti
- Sezione 4.4.1.1 Auto-formazione e Consolidamento
- Sezione 4.4.1.2 Apprendimento tra Pari

Input Necessità di formazione (Nuovo strumento o Rotazione Ruolo)

Output Membro/Team competente

Procedura
1. **Studio Individuale**: Il membro studia le risorse per lo sviluppo materiale delle attività o rilegge le sezioni pertinenti delle Norme di Progetto.
2. **Sessione di Spiegazione (Nuovi Strumenti)**: In caso di nuove tecnologie, viene organizzata una call su *Discord*$_G$ in cui il membro più esperto mostra esempi pratici di utilizzo.
3. **Passaggio di Consegne (Rotazione Ruoli)**: Il membro che sta assumendo un nuovo ruolo contatta uno dei membri che lo hanno precedentemente svolto per ricevere consigli operativi.

5. Metriche e standard per la Qualità

Le metriche e standard per la qualità fanno riferimento allo standard *ISO/IEC 12207:1995*$_G$. Attraverso un'operazione di *tailoring*$_G$, il team ha selezionato gli standard della qualità pertinenti al contesto del progetto, classificandoli in tre macro-categorie:

1. Standard di Processo;
2. Standard di Prodotto;
3. Standard di Documentazione.

5.1. Standard del Processo

Il gruppo adotta le norme dettate dallo standard *ISO/IEC 12207:1995*$_G$, ed in particolare la quality_assurance definita nel medesimo standard. In particolare il gruppo si impegna a rispettare:

5.1.1. Pianificazione dell'Assicurazione Qualità

Ogni attività di verifica deve essere definita nel documeto Piano di Qualifica, il quale specifica lo standard , le procedure e gli strumenti da utilizzare.

5.1.2. Indipendenza e Autorità

Per garantire imparzialità nella verifica, chi svolge il ruolo di *Verificatore*$_G$ deve essere libero di poter segnalare ogni tipo di anomalia senza subire conseguenze o influenza da altri membri del gruppo, per garantira imparzialità da esso. La persona che svolge l'attività di *Verificatore*$_G$ per un determinato prodotto non può essere l'Autore dello stesso.

5.1.3. Product Assurance

Il gruppo si impegna a controllare e verificare che ogni prodotto sviluppato e la relativa documentazione sia conforme ai piani ed ai requisiti prima della consegna.

5.1.4. Process Assurance

Il gruppo si impegna ad assicurare che i processi di fornitura, sviluppo e supporto siano conformi alle norme stabilite in questo documento.

5.2. Standard di Prodotto

5.2.1. Idoneità Funzionale

Capacità del prodotto di rispettare e soddisfare le funzioni richieste.

5.2.2. Affidabilità

Capacità di mantenere le prestazioni.

5.2.3. Manutenibilità

Facilità con cui il software può essere modificato.

5.3. Standard di Documentazione

5.3.1. Standard per il formato data-ora

Lo standard di rappresentazione della data e ora è definito nello standard ***ISO 8601***$_G$ con il formato AAAA-MM-GG.

5.3.2. Indice di Gulpease

Standard di riferimento per la leggibilià della linguia italiana nei documenti tecnici. L'obiettivo minimo del gruppo è fissato a 60.

6. Metriche di Qualità del Processo

6.1. Processi Primari

6.1.1. Fornitura

6.1.1.1. MP01 - Earned Value

Descrizione: Misura il valore del lavoro effettivamente completato rispetto alla pianificazione.

Formula:

$$\text{EV} = (\text{Budget at Completion} \times \text{Percentuale di lavoro completato nello sprint}_{\text{G}})$$

6.1.1.2. MP02 - Planned Value

Descrizione: Indica il valore del lavoro pianificato fino a una certa data.

Formula:

$$\text{PV} = (\text{Budget at Completion} \times \text{Percentuale di lavoro pianificato nello sprint}_{\text{G}})$$

6.1.1.3. MP03 - Actual Cost

Descrizione: Rappresenta il costo reale sostenuto fino a un determinato momento.

Formula:

$$\text{AC} = \sum(\text{costi reali sostenuti})$$

6.1.1.4. MP04 - Cost Performance Index (CPI)

Descrizione: Valuta l'efficienza dei costi del progetto.

Formula:

$$\text{CPI} = \frac{\text{EV}}{\text{AC}}$$

6.1.1.5. MP05 - Schedule Performance Index (SPI)

Descrizione: Misura l'avanzamento temporale rispetto alla pianificazione.

Formula:

$$\text{SPI} = \frac{\text{EV}}{\text{PV}}$$

6.1.1.6. MP06 - Estimate At Completion (EAC)

Descrizione: Stima il costo totale previsto alla fine del progetto.

Formula:

$$\text{EAC} = \text{AC} + \frac{\text{BAC} - \text{EV}}{\text{CPI}}$$

6.1.1.7. MP07 - Estimate To Complete (ETC)

Descrizione: Stima il costo necessario per completare il progetto a partire da un certo punto.

Formula:

$$\text{ETC} = \text{EAC} - \text{AC}$$

6.1.1.8. MP08 - Time Estimate At Completion (TEAC)

Descrizione: Stima il tempo totale previsto alla fine del progetto.

Formula:

$$\text{TEAC} = \frac{\text{durata pianificata}}{\text{SPI}}$$

6.1.1.9. MP09 - Budget Burn Rate

Descrizione: Misura la velocità con cui il budget viene consumato.

Formula:

$$\text{Budget Burn Rate} = \frac{\text{AC}}{\text{giorni trascorsi}}$$

6.1.2. Sviluppo

6.1.2.1. MP10 - Requirements Stability Index

Descrizione: Misura la stabilità dei requisiti durante lo sviluppo.

Formula:

$$\text{RSI} = \frac{\text{Requisiti modificati}}{\text{Requisiti totali}}$$

6.2. Processi di Supporto

6.2.1. Documentazione

6.2.1.1. MP11 - Indice di Gulpease

Descrizione: Valuta la leggibilità della documentazione tecnica.

Formula:

$$\text{Indice di Gulpease} = 89.5 - \frac{\text{Numero di lettere}}{\text{Numero di parole}} \times 100 + \frac{\text{Numero di frasi}}{\text{Numero di parole}} \times 300$$

6.2.1.2. MP12 - Correttezza Ortografica

Descrizione: Misura la percentuale di parole senza errori ortografici nella documentazione.

Formula:

$$\text{Correttezza Ortografica} = \left(1 - \frac{\text{Parole con errori}}{\text{Parole totali}}\right) \times 100$$

6.2.2. Verifica

6.2.2.1. MP13 - Code Coverage

Descrizione: Misura la percentuale di codice sorgente coperto dai test.

Formula:

$$\text{Code Coverage} = \frac{\text{Linee di codice testate}}{\text{Linee di codice totali}} \times 100$$

6.2.2.2. MP14 - Test Success Rate

Descrizione: Misura la percentuale di test superati rispetto al totale dei test eseguiti.

Formula:

$$\text{Test Success Rate} = \frac{\text{Test superati}}{\text{Test eseguiti}} \times 100$$

6.2.2.3. MP15 - Test Automation Percentage

Descrizione: Misura la percentuale di test automatizzati rispetto al totale dei test eseguiti.

Formula:

$$\text{Test Automation Percentage} = \frac{\text{Test automatizzati}}{\text{Test totali}} \times 100$$

6.2.2.4. MP16 - Defect Discovery Rate

Descrizione: Misura la velocità con cui vengono scoperti i difetti durante la fase di verifica.

Formula:

$$\text{Defect Discovery Rate} = \frac{\text{Difetti scoperti}}{\text{Tempo di verifica}} \times 100$$

6.2.3. Gestione della Configurazione

6.2.3.1. MP17 - *Commit*$_G$ Message Quality Score

Descrizione: Valuta la qualità dei messaggi di *commit*$_G$ in base a criteri predefiniti.

Formula:

$$\text{Commit}_\text{G}\text{ Message Quality Score} = \frac{\text{Commit}_\text{G}\text{ con messaggi conformi}}{\text{Commit}_\text{G}\text{ totali}} \times 100$$

6.2.4. Gestione della Qualità

6.2.4.1. MP18 - Quality Metrics Satisfied

Descrizione: Misura la percentuale di metriche di qualità soddisfatte rispetto al totale delle metriche definite.

Formula:

$$\text{Quality Metrics Satisfied} = \frac{\text{Metriche soddisfatte}}{\text{Metriche totali}} \times 100$$

6.2.4.2. MP19 - *Quality Gate*$_G$ Pass Rate

Descrizione: Misura la percentuale di passaggio attraverso i gate di qualità definiti durante il processo di sviluppo.

Formula:

$$\text{Quality Gate}_\text{G}\text{ Pass Rate} = \frac{\text{Gate di qualità superati}}{\text{Gate di qualità totali}} \times 100$$

6.3. Processi Organizzativi

6.3.1. Gestione dei Processi

6.3.1.1. MP20 - Time Efficiency

Descrizione: Misura l'efficienza produttiva delle ore.

Formula:

$$\text{Time Efficiency} = \frac{\text{Ore produttive}}{\text{Ore totali}} \times 100$$

6.3.1.2. MP21 - *Sprint*$_G$ Velocity Stability

Descrizione: Misura la stabilità della velocità di completamento degli *sprint*$_G$ nel tempo.

Formula:

$$\text{Sprint}_\text{G}\text{ Velocity Stability} = 1 - \frac{\text{Deviazione standard della velocità degli sprint}_\text{G}}{\text{Velocità media degli sprint}_\text{G}}$$

6.3.1.3. MP22 - Meeting Efficiency Index

Descrizione: Valuta l'efficienza delle riunioni in termini di tempo speso rispetto agli obiettivi raggiunti.

Formula:

$$\text{Meeting Efficiency Index} = \frac{\text{Tempo speso in riunioni}}{\text{Obiettivi raggiunti nelle riunioni}}$$

6.3.1.4. MP23 - *PR_G* Resolution Time

Descrizione: Misura il tempo medio impiegato per risolvere le Pull Request aperte.

Formula:

$$\text{PR}_\text{G}\ \text{Resolution Time} = \frac{\text{Tempo totale per risolvere PR}_\text{G}}{\text{Numero di PR}_\text{G}\ \text{risolte}}$$

7. Metriche di Qualità del Prodotto

7.1. Funzionalità

7.1.1. MQ01 - Requisiti Obbligatori Soddisfatti

Descrizione: Misura la percentuale di requisiti obbligatori soddisfatti rispetto al totale dei requisiti obbligatori definiti.

Formula:

$$\text{Requisiti Obbligatori Soddisfatti} = \frac{\text{Requisiti obbligatori verificati}}{\text{Requisiti obbligatori totali}} \times 100$$

7.1.2. MQ02 - Requisiti Desiderabili Soddisfatti

Descrizione: Misura la percentuale di requisiti desiderabili soddisfatti rispetto al totale dei requisiti desiderabili definiti.

Formula:

$$\text{Requisiti Desiderabili Soddisfatti} = \frac{\text{Requisiti desiderabili verificati}}{\text{Requisiti desiderabili totali}} \times 100$$

7.1.3. MQ03 - Requisiti Opzionali Soddisfatti

Descrizione: Misura la percentuale di requisiti opzionali soddisfatti rispetto al totale dei requisiti opzionali definiti.

Formula:

$$\text{Requisiti Opzionali Soddisfatti} = \frac{\text{Requisiti opzionali verificati}}{\text{Requisiti opzionali totali}} \times 100$$

7.1.4. MQ04 - Requirements Test Coverage

Descrizione: Misura la percentuale di requisiti coperti da test rispetto al totale dei requisiti definiti.

Formula:

$$\text{Requirements Test Coverage} = \frac{\text{Requisiti coperti da test}}{\text{Requisiti totali}} \times 100$$

7.2. Affidabilità

7.2.1. MQ05 - *Branch*$_G$ Coverage

Descrizione: Misura la percentuale di rami del codice sorgente coperti dai test.

Formula:

$$\text{Branch}_\text{G}\ \text{Coverage} = \frac{\text{Rami coperti da test}}{\text{Rami totali}} \times 100$$

7.2.2. MQ06 - Statement Coverage

Descrizione: Misura la percentuale di istruzioni del codice sorgente coperti dai test.

Formula:

$$\text{Statement Coverage} = \frac{\text{Istruzioni coperte da test}}{\text{Istruzioni totali}} \times 100$$

7.2.3. MQ07 - Failure Density

Descrizione: Misura la densità di difetti rilevati durante la fase di verifica rispetto alla dimensione del software.

Formula:

$$\text{Failure Density} = \frac{\text{Difetti rilevati}}{\text{Linee di codice totali}} \times 1000$$

7.2.4. MQ08 - Modified Condition/Decision Coverage (MC/DC)

Descrizione: Misura la copertura dei test in base alla combinazione di condizioni e decisioni nel codice.

Formula:

$$\text{MC/DC Coverage} = \frac{\text{Condizioni/Decisioni coperte da test}}{\text{Condizioni/Decisioni totali}} \times 100$$

7.3. Usabilità

7.3.1. MQ09 - Time On $Task_G$

Descrizione: Misura il tempo medio impiegato dagli utenti per completare un'attività specifica utilizzando il software.

Formula:

$$\text{Time On Task}_\text{G} = \frac{\text{Tempo totale per completare l'attività}}{\text{Numero di utenti che hanno completato l'attività}}$$

7.3.2. MQ10 - Error Rate

Descrizione: Misura la percentuale di errori commessi dagli utenti durante l'utilizzo del software.

Formula:

$$\text{Error Rate} = \frac{\text{Errori commessi dagli utenti}}{\text{Interazioni totali degli utenti}} \times 100$$

7.4. Efficienza

7.4.1. MQ11 - Response Time

Descrizione: Misura il tempo medio di risposta del software a una richiesta dell'utente.

Formula:

$$\text{Response Time} = \frac{\text{Tempo totale di risposta}}{\text{Numero di richieste}}$$

7.5. Manutenibilità

7.5.1. MQ12 - Code Smells

Descrizione: Misura la presenza di «code smells» nel codice sorgente, che indicano potenziali problemi di manutenibilità.

Formula:

$$\text{Code Smells} = \frac{\text{Numero di code smells}}{\text{Linee di codice totali}} \times 1000$$

7.5.2. MQ13 - Coefficient of Coupling

Descrizione: Misura il grado di accoppiamento tra i moduli del software, indicando la dipendenza tra di essi.

Formula:

$$\text{Coefficient of Coupling} = \frac{\text{Numero di dipendenze tra moduli}}{\text{Numero totale di moduli}}$$

7.5.3. MQ14 - Cyclomatic Complexity

Descrizione: Misura la complessità del codice sorgente in base al numero di percorsi indipendenti attraverso il codice.

Formula:

$$\text{Cyclomatic Complexity} = \text{Numero di rami} - \text{Numero di nodi} + 2$$

7.5.4. MQ15 - Code Duplication Percentage

Descrizione: Misura la percentuale di codice duplicato rispetto al totale del codice sorgente.

Formula:

$$\text{Code Duplication Percentage} = \frac{\text{Linee di codice duplicate}}{\text{Linee di codice totali}} \times 100$$

7.6. Portabilità

7.6.1. MQ16 - Container Image Size

Descrizione: Misura la dimensione dell'immagine del container utilizzato per distribuire il software, indicando l'efficienza della distribuzione.

Formula:

$$\text{Container Image Size} = \text{Dimensione dell'immagine del container in MB}$$

7.6.2. MQ17 - Application Success Rate

Descrizione: Misura la percentuale di distribuzioni del software che sono state eseguite con successo senza errori.

Formula:

$$\text{Application Success Rate} = \frac{\text{Distribuzioni riuscite}}{\text{Distribuzioni totali}} \times 100$$

7.6.3. MQ18 - Encryption Coverage

Descrizione: Misura la percentuale di dati sensibili protetti da crittografia rispetto al totale dei dati sensibili gestiti dal software.

Formula:

$$\text{Encryption Coverage} = \frac{\text{Dati sensibili crittografati}}{\text{Dati sensibili totali}} \times 100$$

8. Riferimenti

8.1. Riferimenti normativi

- **Standard *ISO 8601*$_G$ - Formata data e ora**
  - *https*$_G$://www.iso.org/iso-8601-date-and-time-format.html — ultimo accesso: 2026-03-06
  - *https*$_G$://it.wikipedia.org/wiki/ISO_8601 — ultimo accesso: 2026-03-06

8.2. Riferimenti informativi

- **Standard *ISO/IEC 12207:1995*$_G$ - Processi del ciclo di vita del software** (versione 1995)
  - *https*$_G$://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf — ultimo accesso: 2026-03-06
- **Documentazione *Git*$_G$**
  - *https*$_G$://*git*$_G$-scm.com/docs — ultimo accesso: 2026-03-06
- **Documentazione *GitHub*$_G$**
  - http://docs.*github*$_G$.com/en — ultimo accesso: 2026-03-06
- **Documentazione *Typst*$_G$**
  - *https*$_G$://*typst*$_G$.app/docs/reference — ultimo accesso: 2026-03-06
- ***Conventional Commits*$_G$ v1.0.0**
  - *https*$_G$://www.conventionalcommits.org/en/v1.0.0/