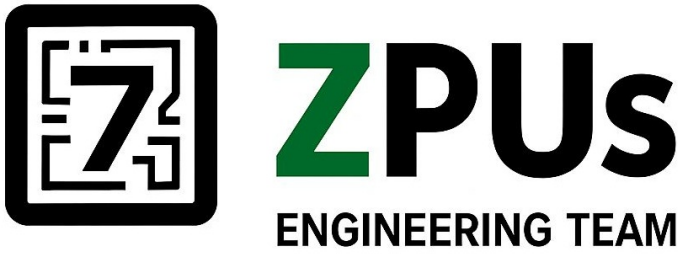

7zpus.swe@gmail.com

Norme di Progetto

Indice

Elenco delle figure

Elenco delle tabelle

Tabella di Versionamento

Versione	Data	Autore	Verificatore	Descrizione
0.9	07/02/2025	Aaron Gingillino	Vigolo Davide	Scrittura paragrafi 2.2, 4.1
0.8	25/01/2025	Aaron Gingillino	Fattoni Antonio	Scrittura paragrafi 3.3,3.4
0.7	15/12/2025	Rocco Matteo A.	Georgescu Diana	Stesura paragrafo 5
0.6	10/12/2025	Georgescu Diana	Soligo Lorenzo	Stesura sottosezioni 2.3, 2.4
0.5	1/12/2025	Soligo Lorenzo	Rocco Matteo A.	Aggiornamento nuovo standard per gestione branch, convenzioni sui nomi, date e versioni. Sezioni 3.1.2, 3.1.3, 3.2.1.1.1, 3.2.4, 4.2.3.2.1
0.4.1	28/11/2025	Soligo Lorenzo	Fattoni Antonio	Correzione nella Procedura di Revisione Paragrafo 3.1.3.1 e aggiornamento immagini
0.4	26/11/2025	Soligo Lorenzo	Fattoni Antonio	Ristrutturazione completa Processi (ISO 12207: Attività-Procedure-Strumenti)
0.3	25/11/2025	Soligo Lorenzo	Fattoni Antonio	Creazione e stesura sezioni Processi di Infrastruttura e sottosezioni 4.2.1-4.3. Nuova struttura Paragrafi e sottoParagrafi
0.2	22/11/2025	Soligo Lorenzo	Fattoni Antonio	Creazione e stesura sezioni Documentazione e sottosezioni 3.1-3.1.5
0.1	16/11/2025	Rocco Matteo A.	Soligo Lorenzo	Creazione e stesura sezioni Introduzione e Processo di fornitura

1 Introduzione

1.1 Scopo

Questo documento ha l'obiettivo di definire e normare il *Way of Working*$_G$, ovvero le regole di lavoro che ogni membro del gruppo deve rispettare durante lo svolgimento delle *attività di progetto*$_G$ volte allo sviluppo dell'applicativo software ***DIPReader***$_G$, proposto dall'azienda *Sanmarco Informatica*$_G$. A ciascun membro è richiesto di seguirle integralmente per poter lavorare in maniera efficace, efficiente e omogenea. Data la natura incrementale della redazione del documento, il *responsabile di progetto*$_G$ ha il compito di mantenere aggiornate le presenti norme e gli eventuali riferimenti ad altri documenti in esse contenuti.

1.2 Glossario

Ogni termine tecnico o con un significato particolare nell'ambito dell'*Ingegneria del Software*$_G$, utilizzato nella documentazione di progetto, è definito nell'apposito documento Glossario 1.0 (ultimo accesso: 17/11/2025).

1.3 Riferimenti

Il gruppo ha redatto il presente documento in conformità con lo *standard*$_G$ ISO/IEC 12207:1995, integrandolo occasionalmente con approfondimenti tratti dalla sua versione più recente, ISO/IEC/IEEE 12207:2017, per includere dettagli aggiuntivi sugli approcci *agili*$_G$ e *iterativi*$_G$ che caratterizzano lo *sviluppo software*$_G$ moderno.

1.3.1 Riferimenti Normativi

- Standard ISO/IEC 12207:1995 (ultimo accesso: 17/11/2025)
- Standard ISO/IEC/IEEE 12207:2017
- Standard ISO/IEC/IEEE 24765:2017
- Capitolato C3: DIPReader (ultimo accesso: 13/11/2025)
- Regolamento di Progetto Didattico a.a. 2025/2026 (ultimo accesso: 17/11/2025)

1.3.2 Riferimenti Informativi

- Dispense del corso di Ingegneria del Software 2025/2026:
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T01.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T02.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T03.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T04.pdf (ultimo accesso: 17/11/2025)

  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T05.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T06.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T07.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T08.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T09.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T10.pdf (ultimo accesso: 17/11/2025)
  - https://www.math.unipd.it/ tullio/IS-1/2025/Dispense/T11.pdf (ultimo accesso: 17/11/2025)
- Linee Guida Sviluppo Sicuro AGID (Agenzia per l'Italia Digitale)
- Linee Guida sulla formazione, gestione e conservazione dei documenti informatici AGID
- Documentazione LaTeX by Lorenzo Pantieri (ultimo accesso: 17/11/2025)
- Documentazione Jira

2 Processi Primari

2.1 Processo di Fornitura

Il *processo*$_G$ di fornitura contiene le attività e i compiti svolti dal *fornitore*$_G$. Per implementare correttamente il processo il gruppo si impegna a svolgere le seguenti attività.

2.1.1 Attività di processo

2.1.1.1 Avvio

Il fornitore analizza i *requisiti*$_G$ necessari alla proposta di fornitura, tenendo in considerazione eventuali vincoli organizzativi e normativi.

2.1.1.2 Preparazione della proposta di fornitura

Il fornitore prepara la proposta di fornitura in risposta alle richieste del committente e definisce i termini in cui si articola la proposta.

2.1.1.3 Accordo

Proponente e fornitore entrano nella fase di definizione dell'accordo di fornitura del prodotto software, prevedendo possibilità di negoziazione della fornitura da parte del fornitore.

2.1.1.4 Pianificazione

Il fornitore rielabora l'analisi dei requisiti fondamentali per definire il *framework*$_G$ entro il quale il prodotto verrà sviluppato e gestito, in modo tale da garantire un processo di qualità durante lo sviluppo. Si impegna inoltre a definire il modello del ciclo di vita del prodotto adatto alla complessità del progetto e ai relativi rischi che potrebbero insorgere. Tutte queste decisioni convergono nel Piano di Progetto.

2.1.1.5 Esecuzione e controllo

Il fornitore si impegna a sviluppare il prodotto secondo il Piano di Progetto, avendo cura di controllare che i processi siano stati eseguiti correttamente.

2.1.1.6 Verifica e validazione

Il fornitore stabilisce con la proponente le modalità di rendicontazione dello stato di avanzamento del prodotto e rende disponibili i documenti che dimostrino la verifica e validazione dei processi secondo i requisiti precedentemente individuati.

2.1.1.7 Consegna e terminazione

Il fornitore consegna il prodotto finale al proponente e ne espone le funzionalità.

2.1.1.8 Accordi con l'azienda proponente

I capitolati presentati dalle proponenti vengono analizzati e viene redatto il documento di Analisi dei capitolati, nel quale sono delineati i bisogni e i principali vincoli a cui attenersi per la fornitura del prodotto finale. Il fornitore espone ai committenti di fornitura, ovvero i Professori Vardanega Tullio e Cardin Riccardo, la Lettera di Presentazione della proposta di fornitura che descrive il preventivo di costi, cronogramma di sviluppo, suddivisione del lavoro e i ruoli coinvolti. La *proponente*$_G$, in qualità di *stakeholder*$_G$, esercita il diritto di ricevere la rendicontazione professionale e approfondita del lavoro svolto dal gruppo fornitore, perciò si instaura un accordo per delineare le modalità di comunicazione e il contenuto di tale rendicontazione. È previsto l'aggiornamento costante e tempestivo della proponente per quanto riguarda la pianificazione degli obiettivi e delle tempistiche di sviluppo individuate dal fornitore. Ogni qualvolta vi siano modifiche di notevole interesse esterno dal gruppo fornitore verranno comunicate all'azienda proponente attraverso appositi canali di comunicazione sincrona o asincrona. Il fornitore e la proponente hanno accordato lo svolgimento di un incontro di verifica dello stato di avanzamento lavori (*SAL*$_G$) in modalità sincrona ogni due settimane, in cui discutere l'andamento del lavoro e chiarire eventuali dubbi da parte del fornitore o segnalazioni di difformità dai requisiti iniziali della proponente. È inoltre sempre disponibile la comunicazione via email per questioni minori e di facile risoluzione. La consegna del prodotto è suddivisa in due *milestone*$_G$ principali: *RTB*$_G$ (Requirements and Technology Baseline) e *PB*$_G$ (Product Baseline).

2.1.2 Documentazione fornita

2.1.2.1 Analisi dei requisiti

Nel documento di Analisi dei requisiti (ultimo accesso: 17/11/2025) sono riportati i bisogni e i vincoli a cui attenersi per la realizzazione del prodotto finale. L'obiettivo è definire in maniera non ambigua i *casi d'uso*$_G$ (*Use Cases*) e i requisiti (*Requirements*) del software. Il documento è diviso nelle seguenti sezioni:

1. Introduzione
2. Descrizione
3. Definizione dei casi d'uso
4. Definizione dei requisiti

2.1.2.2 Glossario

Il Glossario è il documento che raccoglie ogni termine di carattere tecnico, nomenclature e acronimi con particolare significato nell'ambito dell'Ingegneria del Software utilizzato nella documentazione di progetto. La definizione dei termini di glossario è coadiuvata dal contenuto dello standard ISO/IEC/IEEE 24765/2017.

2.1.2.3 Piano di progetto

Il Piano di progetto v1.0 (ultimo accesso: 17/11/2025) è il documento che espone all'esterno il lavoro di sviluppo svolto seguendo le procedure delineate all'interno di questo documento. Fornisce una guida dettagliata alla pianificazione, esecuzione e consuntivo delle attività completate in ciascuna *sprint*$_G$. Il documento è diviso nelle seguenti sezioni:

1. Introduzione
2. Analisi dei rischi e mitigazione
3. Modello di sviluppo
4. Pianificazione dei costi e suddivisione ruoli
5. Preventivo di periodo
6. Consuntivo di periodo
7. Retrospettiva

2.1.2.4 Piano di qualifica

Il piano di qualifica descrive gli obiettivi di qualità dei processi che il fornitore si impegna a soddisfare per consegnare un prodotto finale di qualità. Le metriche di valutazione vengono determinate dall'analisi dei requisiti e dalle indicazioni date dalla proponente, suddivise in base all'applicazione sui processi o sul prodotto. Le metriche stabilite vengono poi misurate attraverso opportuni test e verifiche, di cui vengono riportate le specifiche. Il documento include una sezione di rendicontazione per la valutazione dei processi e la valutazione del prodotto, in cui riportare l'attinenza alle metriche ottenuta rispetto agli obiettivi e di conseguenza valutare azioni correttive in caso si verifichino eventuali problemi (*cruscotto di qualità*$_G$). Il documento è diviso nelle seguenti sezioni:

1. Qualità dei processi
2. Qualità del prodotto
3. Specifiche di test e verifica
4. Cruscotto di qualità

2.1.2.5 Lettera di presentazione

La lettera di presentazione è il documento necessario alla candidatura per la milestone di revisione di avanzamento *RTB*$_G$ (*Requirements and Technology Baseline*). Essa contiene le informazioni sul repository di progetto, il puntatore al *Proof of Concept*$_G$, il consuntivo di spesa e preventivo a finire del progetto.

2.1.3 Strumenti

- *GitHub*$_G$ per la gestione della documentazione di progetto e mezzo comunicativo nella fase di fornitura
- *Jira*$_G$ per la suddivisione e il monitoraggio delle attività di progetto
- Discord per la comunicazione sincrona tra i membri del gruppo
- Gmail per la comunicazione asincrona con l'azienda proponente
- VSCode con estensione con IDE di riferimento.

2.2 Processo di sviluppo

Il processo di sviluppo prevede l'insieme di attività che definiscono lo svolgersi dell'Analisi dei Requisiti.

2.2.1 Attività di processo

L'elenco di attività previste è basato sullo standard ISO/IEC 12207:1995, descritte di seguito:

- **Definizione del processo di sviluppo**: selezione del Ciclo di Vita del Software più idoneo tenendo conto degli obiettivi, della rilevanza e della complessità del progetto;
- **Raccolta e analisi dei requisiti**: attività volta a individuare e specificare le esigenze dell'utente finale rispetto alle funzionalità richieste al Software. Un'analisi esaustiva deve includere le funzioni del Sistema, i bisogni degli utilizzatori e i vincoli stabiliti dal committente;
- **Progettazione dell'architettura di Sistema**: identificazione delle componenti hardware e software necessarie a garantire il soddisfacimento di tutti i requisiti definiti, supportata da un adeguato tracciamento degli stessi;
- **Analisi dei requisiti Software**: studio di come il Software risponde ai requisiti lato utente, includendo anche gli aspetti di qualità quali funzionalità (comprensive di requisiti prestazionali), interfacce tra componenti e requisiti di sicurezza;
- **Definizione dell'architettura Software**: progettazione delle principali componenti del sistema e delle loro interazioni, con particolare attenzione alla struttura complessiva piuttosto che ai dettagli implementativi;
- **Progettazione dettagliata del Software**: sviluppo del progetto delle singole componenti Software fino all'individuazione delle unità elementari;
- **Sviluppo e verifica del Software**: realizzazione delle unità che costituiscono le componenti progettate, accompagnata da test specifici per verificarne il corretto funzionamento;
- **Integrazione delle componenti Software**: assemblaggio delle diverse parti in componenti complete, supportato da test di integrazione per garantirne il comportamento corretto;
- **Test di qualificazione del Software**: esecuzione di test dedicati per verificare che il Software soddisfi i requisiti e gli obiettivi di qualità prefissati;
- **Integrazione del Sistema**: combinazione di tutte le componenti realizzate nel Sistema finale;
- **Test di qualificazione del Sistema**: verifica dell'intero Sistema attraverso test complessivi per accertarne il corretto funzionamento;
- **Installazione del Software**: consegna e messa in opera del prodotto presso il cliente finale nell'ambiente precedentemente concordato;
- **Supporto all'approvazione del Software**: attività di assistenza all'utente finale per verificare che tutti i requisiti richiesti siano stati effettivamente soddisfatti.

2.2.2 Analisi dei Requisiti

L'*Analisi dei Requisiti* $_G$rappresenta una delle attività fondamentali all'interno della ***Requirements and Technology Baseline****(RTB)*$_G$ e ha l'obiettivo di identificare in modo completo l'insieme dei requisiti che il sistema sviluppato dovrà soddisfare.

I risultati di tale attività sono raccolti nel documento *Analisi dei Requisiti*, nel quale sono riportate in maniera dettagliata tutte le informazioni necessarie. Questo documento costituisce un riferimento essenziale per le successive fasi di progettazione dell'architettura e di codifica, supportando il lavoro dei progettisti e degli sviluppatori.

Un ulteriore elemento di riferimento è il *Piano di Qualifica* che, includendo l'elenco dei test e il loro stato di avanzamento, consente di verificare quali requisiti risultano soddisfatti e quali siano ancora da validare.

In particolare, il documento di Analisi dei Requisiti organizza i *casi d'uso*$_G$ individuati e i relativi requisiti associati. Al fine di agevolarne la consultazione, viene di seguito illustrata nel dettaglio la nomenclatura adottata.

2.2.3 Casi d'uso

I casi d'uso utilizzano la nomenclatura seguente:

UC-[Primario](-[Secondario]-[Terziario]-ecc.)

UC sta per Use Case, ovvero caso d'uso in inglese. Gli UC sono identificati univocamente tramite un enumerazione crescente. *Primario* è il numero dello UC principale. È possibile avere dei *sotto-UC*, in tal caso verrà aggiunto il numero crescente del sotto-UC come pendice allo UC genitore. Ogni UC è anche dotato di un nome descrittivo.

2.2.4 Requisiti

I requisiti utilizzano la nomenclatura seguente:

R-[ID]-[Tipo]-[Priorità]

Dove:

- **ID**: numero progressivo del requisito
- Tipo:
  - **F** (Requisiti Funzionali): descrivono le funzionalità del sistema
  - **Q** (Requisiti di Qualità): descrivono le caratteristiche qualitative del sistema
  - **V** (Requisiti di Vincolo): descrivono i vincoli tecnologici e normativi
- Priorità: **Ob** (Obbligatorio), **De** (Desiderabile), **Op** (Opzionale)

2.3 Processo operativo

Il processo operativo comprende quell'insieme di attività trasversali che sono necessarie a garantire il corretto coordinamento tra i membri del gruppo e il raggiungimento degli obiettivi del progetto. Tale processo assicura comunicazioni efficaci, nonchè una distribuzione ottimale dei compiti, al fine di mantenere alta la qualità del software e

rispettare le tempistiche stabilite. Il processo operativo si integra naturalmente con tutti gli altri processi del progetto.

2.3.1 Pianificazione operativa

La pianificazione operativa rappresenta l'organizzazione delle attività quotidiane e settimanali del gruppo. Il Responsabile, all'inizio di ogni sprint, coordina la distribuzione dei compiti tra i membri.

Procedure di pianificazione operativa:

1. Svolgimento di una riunione di pianificazione all'inizio di ogni sprint;
2. Discussione degli obiettivi dello sprint e le attività necessarie per raggiungerli;
3. Assegnamento delle task ai vari membri;
4. Scelta delle scadenze intermedie.

2.3.2 Gestione dei rischi operativi

La gestione dei rischi operativi mira a identificare in maniera proattiva le potenziali problematiche che potrebbero minare il normale svolgimento delle attività e a pianificare le dovute azioni di mitigazione. I principali rischi operativi identificati sono:

- Sforamento dei costi preventivati;
- Calo di produttività del team;
- Mancata comunicazione e collaborazione tra i membri del team;
- Mancata comunicazione con l'azienda proponente;
- Problemi tecnici con gli strumenti di sviluppo;
- Mancato rispetto delle norme e documenti di progetto interni.

Le strategie adottate per ogni tipologia di rischio sono rispettivamente:

- Monitoraggio costante dell'allocazione delle ore rispetto alla pianificazione iniziale, svolgimento di stand-up meetings periodici e previsione margini temporali per imprevisti;
- Pianificazione anticipata delle attività più critiche prima del periodo di calo, e le restanti tenendo conto del periodo di ridotta attività;
- Adozione di canali di comunicazione chiari e regolari e una routine di aggiornamenti pianificati per garantire che tutti i membri del team siano allineati sugli obiettivi e le responsabilità;

- Scelta di un calendario di incontri regolari con l'azienda proponente per garantire un flusso costante di comunicazione e feedback;
- Impostazione di sessioni di formazione iniziali con il supporto occasionale dell'azienda proponente per familiarizzare con gli strumenti e le tecnologie;
- Ruolo attivo di amministratori e tester per garantire il rispetto delle norme e dei documenti di progetto interni.

2.4 Processo di manutenzione

Il processo di manutenzione definisce le modalità con cui il gruppo gestisce le modifiche, le correzioni e gli aggiornamenti del software e della corrispondente documentazione durante tutto il ciclo di vita del progetto. Questo processo garantisce che il prodotto rimanga conforme ai requisiti. Eventuali difetti devono essere corretti tempestivamente mentre, ove sorgesse la necessità di apportare migliorie, queste ultime vengano fornite in modo efficace ed efficiente.

2.4.1 Manutenzione correttiva

Si tratta di interventi finalizzati alla correzione di difetti o malfunzionamenti trovati nel software o nella documentazione. Sono inclusi:

- Correzioni di bug;
- Risoluzione di errori di varia natura nella documentazione;
- Correzione di qualsiasi comportamento non conforme ai requisiti.

2.4.2 Manutenzione adattiva

Sono le modifiche necessarie per adattare il prodotto a cambiamenti in itinere nell'ambiente operativo o nei requisiti. Sono inclusi:

- Adattamento a nuove tecnologie;
- Modifiche conseguenti a variazioni nei requisiti.

2.4.3 Manutenzione preventiva

Si risferisce alle attività proattive per ridurre la probabilità di problemi futuri. Sono inclusi:

- Aggiornamenti di sicurezza;
- Miglioramento della gestione degli errori.

2.4.4 Identificazione della necessità di manutenzione

La necessità di manutenzione può emergere da diverse fonti, come segnalazioni interne quali le attività di verifica e validazione, lo sviluppo del codice e le retrospettive; o segnalazioni esterne quali feedback della proponente.

2.4.5 Verifica e validazione

La verifica va effettuata rigorosamente prima dell'integrazione. Il verificatore assegnato deve esaminare il codice modificato verificando la conformità agli standard e l'assenza di introduzione di problematiche note. Deve inoltre eseguire i test, verificare che non siano presenti regressioni e controllare che la documentazione sia stata aggiornata coerentemente.

La validazione riguarda modifiche significative che possono impattare i requisiti o l'architettura. Tali modifiche vengono presentate illustrando il problema e la soluzione implementata e si accoglie il feedback della proponente.

2.4.6 Metriche

Per valutare l'efficacia della manutenzione, il gruppo monitora le seguenti metriche:

- Tempo medio di risoluzione: tempo medio tra l'identificazione di un problema e la sua risoluzione;
- Percentuale di regressioni: numero di nuovi difetti introdotti da attività di manutenzione rispetto al totale delle modifiche;
- Distribuzione per tipologia: suddivisione degli interventi di manutenzione per categoria.

3 Processi di Supporto

I processi di supporto sono volti a garantire l'efficacia e l'efficienza dei processi primari.

3.1 Processo di documentazione

Il processo di documentazione è parte integrante del Progetto in quanto permette il tracciamento delle decisioni prese, delle attività svolte e dei risultati ottenuti. Tutto ciò al fine di favorire il lavoro asincrono tra membri del gruppo e promuovere il principio *Agile*$_G$ di continuo miglioramento e adattamento tramite *feedback*$_G$.

3.1.1 Attività: Pianificazione della documentazione

La pianificazione della documentazione avviene contestualmente alla pianificazione delle attività di progetto.
Durante la pianificazione di ogni *sprint*$_G$, il *responsabile di progetto*$_G$ assegna le attività di documentazione ai membri del gruppo, tenendo conto delle competenze e della

disponibilità di ciascuno. Le scadenze per la consegna dei documenti sono stabilite in modo da garantire che la documentazione sia sempre aggiornata e disponibile per la consultazione da parte del gruppo e di eventuali attori esterni (Azienda *proponente*$_G$, *committente*$_G$).
Per una più efficiente scrittura dei documenti, soprattutto di tutti quei documenti periodici (Verbali Interni, Verbali Esterni, Diario di Bordo) sono presenti modelli standard approvati in /assets. L'aggiornamento di tali standard deve essere argomento di Verbali Interni e risultato di una discussione e successiva decisione presa in tale sede.

3.1.1.1 Procedure di Pianificazione

I seguenti passaggi guidano il Team nella pianificazione delle attività di documentazione:

1. Durante la pianificazione di ogni *sprint*$_G$, il *responsabile*$_G$ identifica le necessità di documentazione in base agli obiettivi dello sprint e alle attività previste.
2. Il responsabile assegna le attività di documentazione ai membri del gruppo, tenendo conto delle competenze e della disponibilità di ciascuno, nonchè della necessità di ruotare i ruoli, per dare la possibilità a tutti i membri di acquisire esperienza in diverse aree.
3. Vengono create le *issue*$_G$ in Jira per ogni attività di documentazione, specificando i dettagli del compito, le scadenze e i verificatori per ogni attività. Specifiche in 3.2.1.1.

3.1.1.2 Strumenti di Pianificazione

- *Jira*$_G$ per la gestione delle attività di progetto. In particolare con la *board*$_G$ *Scrum*$_G$ che viene aggiornata in automatico con i commit effettuati sui Work Item e può essere personalizzata con la creazione di *Sprint*$_G$.
- *DashBoard/Cruscotto*$_G$ di Jira per il monitoraggio delle attività assegnate per ogni membro del gruppo.

3.1.2 Attività: Produzione della documentazione

La produzione della documentazione, assegnata durante la pianificazione, è visibile all'assegnatario come *Work item*$_G$ grazie all'estensione Jira di VSCode 4.2.3.3. Grazie a quest'ultima è possibile creare direttamente il Branch di lavoro che si baserà sulla feature branch principale. Una volta completata la stesura, seguendo i modelli standard sopracitati, l'autore del documento crea una *(PR) Pull Request*$_G$ verso la feature branch principale, assegnando come revisore il membro del gruppo designato, diverso da se.
A questo punto:

- Se il revisore **approva la PR**, questo branch viene automaticamente eliminato, il work item viene marcato come completato in Jira e l'assegnatario può proseguire con gli altri compiti a lui assegnati.

- Se il revisore richiede modifiche **la PR viene rifiutata** e l'assegnatario deve procedere con le modifiche richieste. Una volta completate, l'assegnatario notifica il revisore che procederà con una nuova revisione. Questo ciclo si ripete fino a quando la PR non viene approvata.

L'integrazione con Jira permette di controllare lo stato di avanzamento dei Work Item, la rendicontazione delle ore lavorate e la gestione delle scadenze. Risulta quindi **obbligatorio** l'utilizzo di Smart Commit per tutti i commit, compresi quelli di Pull Request. Più in 3.2.3.1.

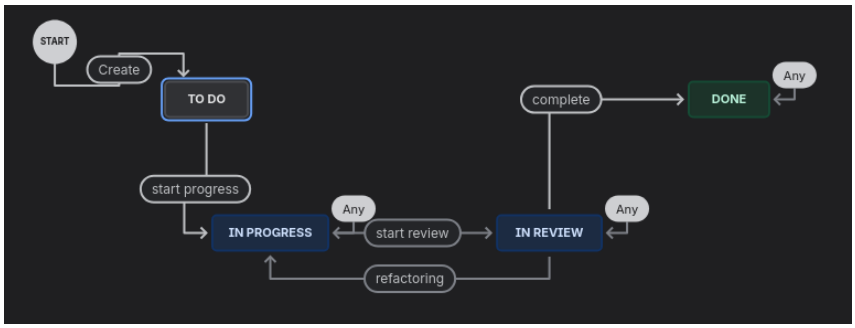

Figura 1: Flusso del processo di documentazione

3.1.2.1 Procedure di Produzione

I seguenti passaggi guidano il Team nella produzione di documenti:

1. Consultando l'estensione "Atlassian: Jira, Rovo Dev, Bitbucket" il membro del gruppo potrà avere accesso al Work Item assegnatogli e cliccando su "Start Work" potrà **creare il branch** di lavoro secondo le convenzioni stabilite (3.2.2.1). Nell'immagine sottostante sono indicati visivamente i passi.

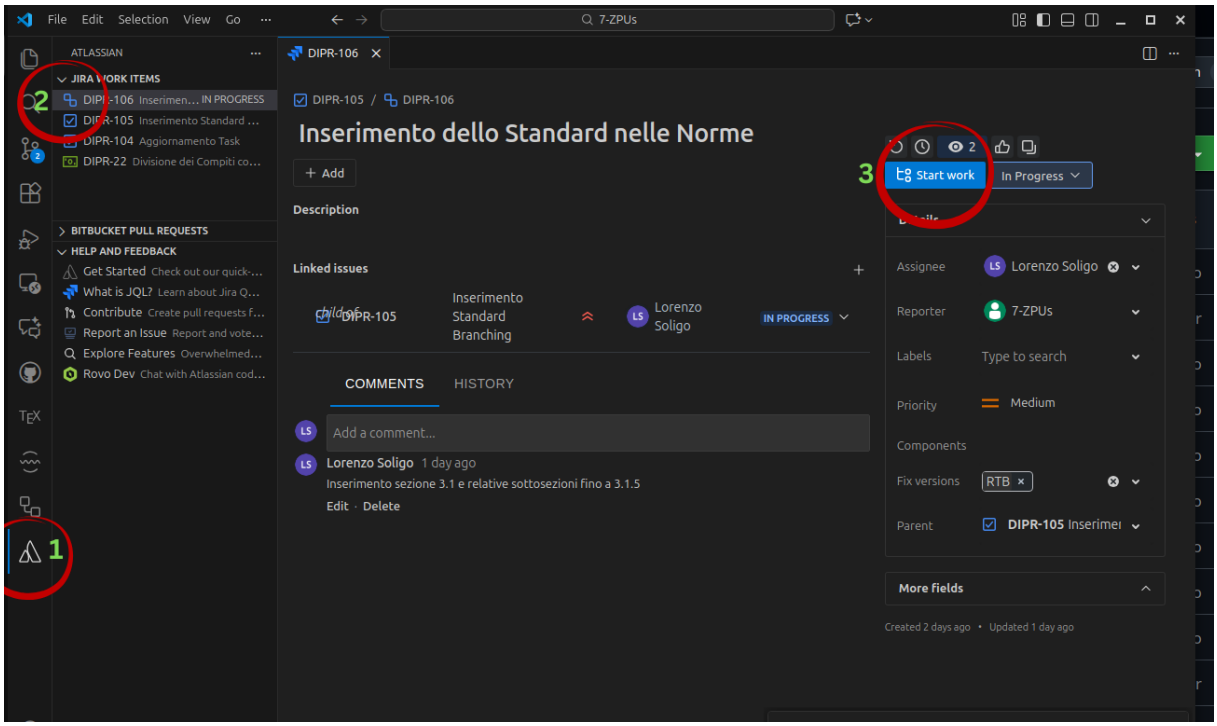

Figura 2: Creazione del branch di lavoro tramite estensione Jira in VSCode

2. Quando è necessario **effettuare un commit** è obbligatorio utilizzare lo Smart Commit. Più in 4.2.3.2.

3. Una volta completata la task, è necessario **creare la PR** verso la branch *"in_lavorazione"* di riferimento mettendo come revisore il membro del gruppo designato.
   Per esempio, una volta completata la stesura di un verbale viene creata una PR verso *verbali_in_lavorazione* e se questa viene approvata, il documento sarà pronto per la revisione finale del responsabile che si occuperà del merge al *main* in sede di milestone.

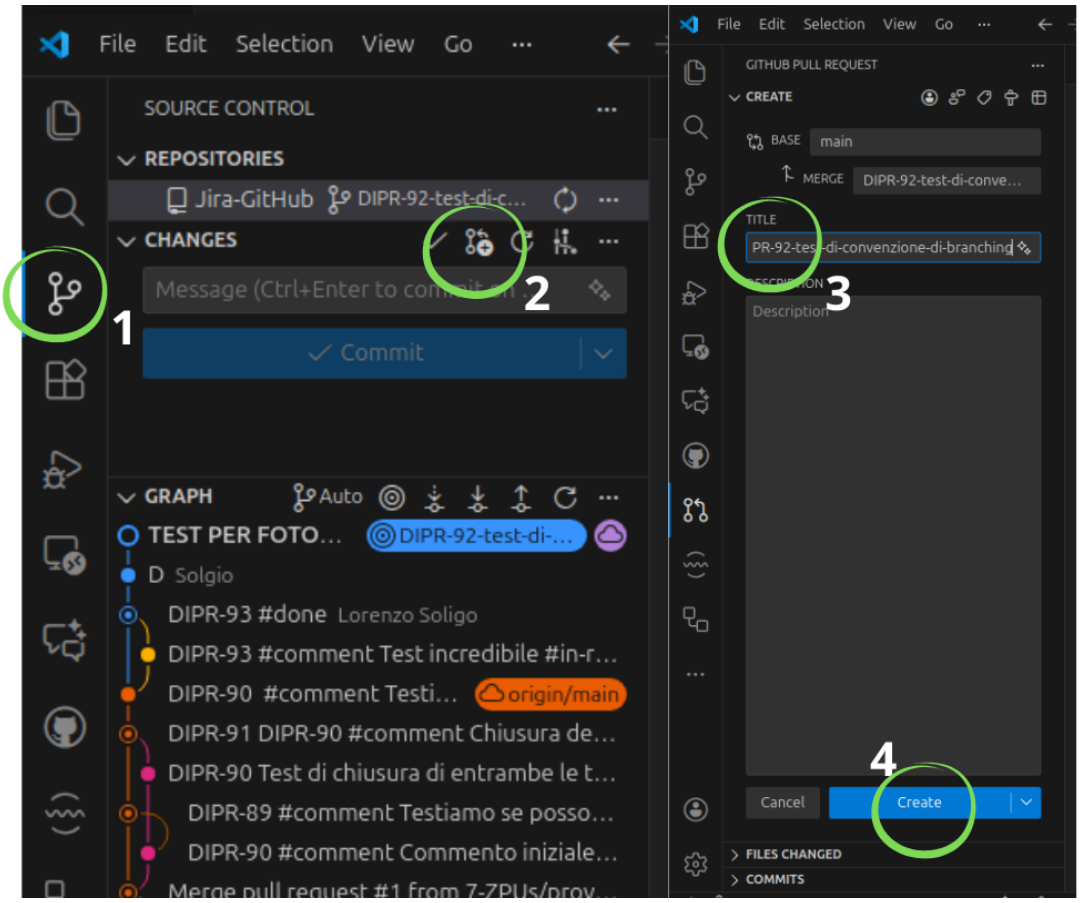

Figura 3: Creazione della PR verso l'issue branch *in_lavorazione*

3.1.2.1.1 Denominazione e datazione

Per una corretta archiviazione e reperibilità dei documenti, è necessario seguire le seguenti convenzioni per la datazione e denominazione.

- Tutti i file dei documenti devono seguire la regola del **Pascal Case**, ovvero devono essere scritti senza spazi e con la prima lettera di ogni parola in maiuscola.

- All'interno dei documenti, la data deve essere riportata nel formato **DD-MM-YYYY**, giorno-mese-anno.

La denominazione dei file sul sito necessita di seguire invece la datazione usata per il nome dei file su GitHub ovvero **YYYY-MM-DD**, con l'aggiunta del numero di versione alla fine del nome. Per le specifiche di versionamento, si rimanda alla sezione 3.2.4.1.

3.1.2.2 Strumenti di Produzione

- *VSCode*$_G$ come IDE principale per la stesura dei documenti in *LaTeX*$_G$.
- *Estensione Jira per VSCode*$_G$ per la gestione dei Work Item assegnati e la creazione automatica delle branch di lavoro.
- *GitHub*$_G$ per la gestione delle versioni della documentazione di progetto.

3.1.3 Attività: Revisione e Approvazione

Ogni documento redatto viene sottoposto a un processo di revisione interna che ne accerta la correttezza contenutistica, formale, e stilistica. La revisione viene effettuata da un membro del gruppo diverso dall'autore del documento seguendo la procedura definita a seguire

3.1.3.1 Procedure di Revisione e Approvazione

1. Una volta ricevuta la notifica della PR da revisionare, il revisore dovrà controllare l'aderenza ai modelli approvati, la correttezza formale e sostanziale del documento. Per velocizzare, oltre alla lettura attenta, si consiglia l'uso di LLM, in particolare per l'analisi grammaticale e stilistica.
2. Completata la revisione, il revisore può:
    - **Approvare la PR**, notificando all'autore l'approvazione. È necessario che nel testo del commit del merge siano chiuse tramite Smart Commit entrambe le issue correlate.
    - **Richiedere modifiche**, fornendo un feedback dettagliato all'autore, chiudendo la PR che sarà riaperta dall'autore una volta implementate le modifiche.

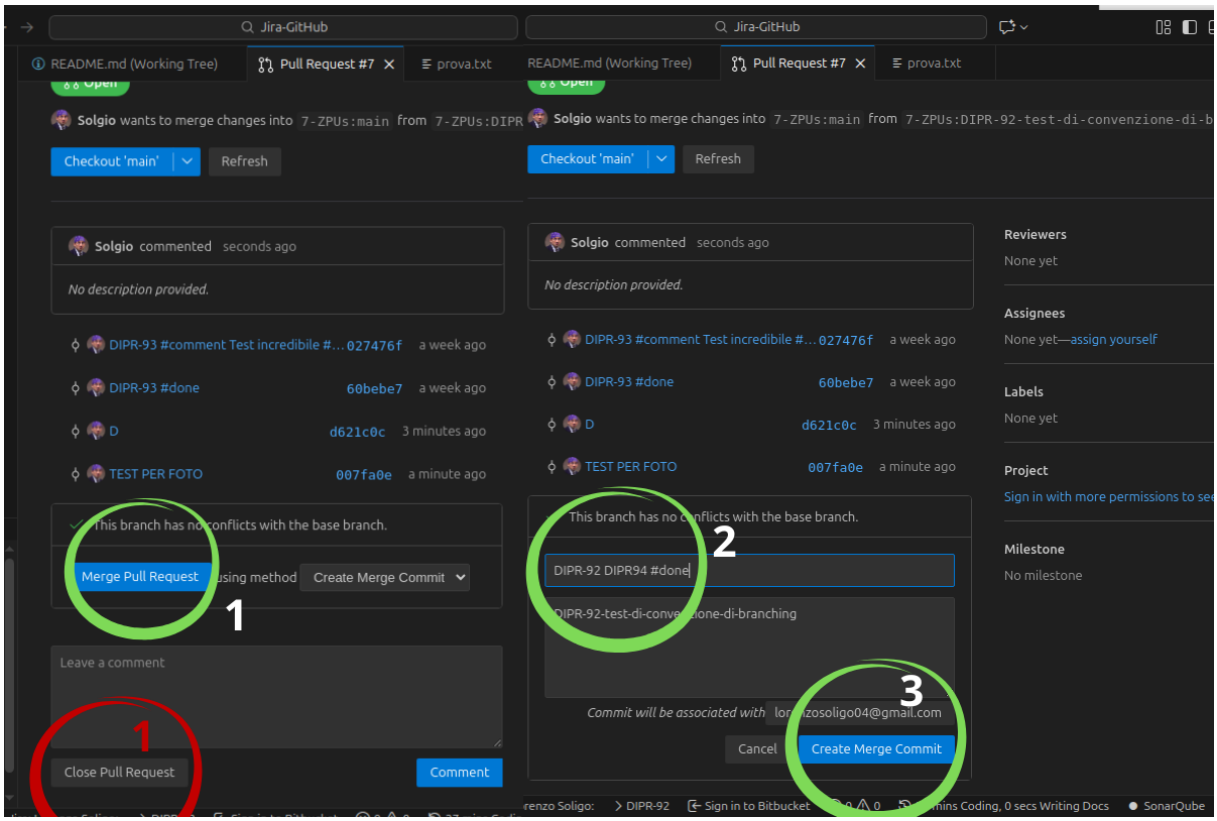

Figura 4: Commit della PR con Smart Commit verso le due issue

Per quanto riguarda l'approvazione finale del documento, questa spetta al *responsabile*$_G$, il quale effettua il merge da "*in_lavorazione*" nel ramo principale *main*, base per la

versione ufficiale di rilascio corrispondente alla *milestone*$_G$. Questo passaggio dovrebbe risultare puramente formale. Non di meno è il garante delle qualità del documento quindi deve impiegare il proprio tempo, il minimo possibile, per rileggere e confermare i contenuti.

3.1.3.2 Strumenti di Revisione e Approvazione

Per la gestione della documentazione di progetto il gruppo utilizza i seguenti strumenti:

- *GitHub*$_G$, in particolare integrato in VSCode per la gestione delle versioni e delle pull request, comodamente nell'ambiente di sviluppo di VSCode 4.2.3.3. Per una stesura efficiente dei documenti il Team si è dotato di modelli predefiniti (Decisione del 2025-11-07).
- *Jira*$_G$: strumento di gestione delle attività di progetto, utilizzato per tracciare le attività di documentazione e assegnarle ai membri del gruppo.
- *LLM*$_G$ per il supporto alla revisione formale e stilistica dei documenti.

3.2 Processo di Gestione della Configurazione

Il processo di gestione della configurazione ha lo scopo di identificare, definire e controllare gli elementi della configurazione software (*SCIs*$_G$) durante tutto il ciclo di vita del progetto, garantendo la tracciabilità delle modifiche e l'integrità dei rilasci. Gli SCIs in poche parole sono tutti gli artefatti prodotti e gestiti durante il progetto.

3.2.1 Attività: Identificazione della Configurazione

Questa attività prevede la definizione degli elementi della configurazione (SCIs) e la loro identificazione univoca.

3.2.1.1 Procedure: Identificazione degli SCIs

Gli SCIs sono sempre associati ad un Work Item di Jira, che ne garantisce la tracciabilità e la gestione delle modifiche. In particolare il processo di creazione deve seguire i seguenti passaggi:

1. Creazione di una issue in Jira per ogni nuovo artefatto da produrre (documento, componente software, etc).
2. Identificazione dell'ambito di appartenenza (Epic), della funzionalità (Feature) o di una specifica attività (Task) già presenti nel sistema o da aggiungere se non compatibile.
3. Assegnazione della issue al membro del gruppo responsabile della sua supervisione.
4. Aggiunta di label specifici per facilitare la ricerca e la categorizzazione degli SCIs, per esempio *DOCS*, *Formazione*, *Code* etc.

5. Aggiunta di Linked Issues per collegare SCIs correlati o dipendenti tra loro, con relazioni di *child/parent of* o *blocked by* per esempio.
6. Aggiunta di eventuali allegati.
7. Definizione delle scadenze e del *Time Estimate*$_G$ per la gestione del carico di lavoro.

È quindi necessaria una specificazione sulla struttura di un Work Item. Ogni Work Item deve contenere deve consistere in una fase produttiva e una fase di revisione, per garantire la qualità del prodotto finale.
A tal fine si adottano le seguenti convenzioni:

- Ogni Work Item deve presentare una sotto issue che rappresenta la fase di produzione.
- La sotto issue di produzione deve essere collegata alla issue principale tramite la relazione *child of*.
- La sotto issue di produzione deve essere assegnata al membro del gruppo responsabile della stesura o sviluppo dell'artefatto, mentre la issue principale deve essere assegnata al membro responsabile della supervisione.

Questo approccio divide chiaramente le responsabilità e le attività, mantenendo la correlazione tra produzione e supervisione e inoltre facilita il monitoraggio dello stato di avanzamento, la gestione delle revisioni e conteggio di *Ore produttive*$_G$ consumate. Di più nello specifico in **??**.

3.2.1.1.1 Issue e SubIssue

In generale, come già detto, ogni Work Item deve essere composto da una issue principale e una sotto issue di produzione. Nel caso però di Riunioni e Diari di Bordo, intesi come eventi ma anche come tipi di work item Jira, la struttura cambia:

- **Riunioni:** La Riunione rappresenta l'evento, simile al concetto di Epic o Feature, mentre le issue rappresentano i verbali interni ed esterni associati alla riunione stessa.
- **Diari di Bordo:** Similarmente, il Diario di Bordo rappresenta l'evento periodico, mentre la issue associata riguarda la preparazione delle slide.

Per entrambi, la creazione della issue di produzione avviene automaticamente una volta che l'evento viene assegnato al membro del team che si occuperà della verifica del materiale prodotto.

3.2.1.2 Strumenti di Identificazione

- **Jira:** Strumento di gestione delle attività di progetto.

3.2.2 Attività: Versionamento e Identificazione

Questa attività definisce le regole per l'identificazione univoca degli artefatti e la gestione delle ramificazioni nel repository.

3.2.2.1 Procedure: Standard per le Branch

Per garantire una gestione ordinata e coerente del codice e della documentazione, il Team adotta il seguente standard per la denominazione dei branch in GitHub:

- La creazione del branch deve avvenire preferibilmente in modo automatico tramite l'integrazione Jira-VSCode 4.2.3.3.
- Il formato obbligatorio è:

```
DIPR-<numero issue>-<descrizione-breve>
```

- È necessario scegliere una descrizione breve che identifichi chiaramente il contenuto della modifica.

3.2.2.2 Strumenti di Versionamento

- **GitHub:** Repository remoto per la memorizzazione delle versioni.
- **VSCode (Estensione Jira):** Per l'automazione della nomenclatura delle branch.

3.2.3 Attività: Controllo delle Configurazioni

Questa attività regolamenta il modo in cui le modifiche di avanzamento vengono registrate e tracciate rispetto ai task pianificati.

3.2.3.1 Procedure: Smart Commit

La necessità di tracciamento delle attività richiede l'adozione capillare degli Smart Commit per collegare i commit GitHub alle issue Jira. Questo permette di aggiornare automaticamente lo stato delle task e rendicontare il tempo.

La sintassi obbligatoria è:

```
DIPR-<numero issue> #time <nd nh nm> #<stato> #comment <Descrizione>
```

Regole di applicazione:

1. **Time Tracking:** L'inserimento del tempo (`#time`) deve seguire il formato *nd nh nm* (giorni, ore, minuti).
2. **Transizioni di Stato:** Utilizzare i tag `#start-progress`, `#start-review`, `#complete` per avanzare il workflow su Jira.

3. **Pull Request:** Nelle PR è vietato inserire il tag `#time` per evitare la doppia contabilizzazione delle ore lavorative.
4. **Issue Multiple:** È possibile agire su più issue in un singolo commit separandole con spazi (utile per chiudere issue di sviluppo e verifica contemporaneamente)[cite: 97, 98].

3.2.3.2 Strumenti

- **Jira Automation:** Interpreta gli Smart Commit per aggiornare i Work Item.
- **Git/GitHub:** Motore di versionamento sottostante.

3.2.4 Registrazione delle Configurazioni

Questa attività prevede la documentazione e la registrazione delle modifiche apportate agli SCIs.

3.2.4.1 Procedure: Registrazione delle Modifiche

Per ogni modifica apportata a uno SCI, è necessario documentare le seguenti informazioni, all'interno della sezione denominata "Tabella di Versionamento", situata all'inizio di ogni documento.
Al suo interno vi sono:

- Numero di versione
- Data della modifica (Di più in 3.1.2.1.1)
- Autore della modifica
- Verificatore della modifica
- Descrizione della modifica, con riferimento preciso, al paragrafo o sezione interessata.

Per la numerazione delle versioni si adotta lo standard di versionamento **MAJOR.MINOR.PATCH**, nel quale:

- **MAJOR** viene incrementato per modifiche sostanziali che introducono cambiamenti significativi o incompatibili con le versioni precedenti. Viene modificato in vista delle milestone.
- **MINOR** viene incrementato per l'aggiunta di nuove funzionalità o miglioramenti che non compromettono la compatibilità con le versioni precedenti.
- **PATCH** viene incrementato per correzioni di bug, miglioramenti minori o modifiche che non influenzano le funzionalità principali del documento.

3.3 Processo di verifica

La verifica serve a controllare che il software prodotto corrisponda ai requisiti richiesti dalla proponente. Per evitare problemi, a ogni task viene assegnato uno o più verificatori. A seconda del tipo di task, i verificatori useranno tecniche e pratiche adeguate per verificare che il prodotto rispetti gli standard di qualità. Dentro il Piano di Qualifica sono elencate le attività che i verificatori devono fare al completamento della task a loro assegnatali. Troviamo anche per ogni attività gli obbiettivi, gli esiti desiderabili e quelli ottenuti.

3.3.1 Verifica documentale

I passaggi che il verificatore deve svolgere per approvare un documento sono specificati nella sezione 3.1.2. Perché un documento possa essere approvato il verificatore deve controllare che sia corretto:

- Il contenuto.
- La grammatica.
- La presentazione, rispetto alle linee guida.
- La comprensibilità del testo.

Individuiamo due tipologie di verifica inerenti al codice, l'analisi statica e quella dinamica

3.3.2 Analisi Statica

L'analisi statica prevede di analizzare il codice in modo statico, ovvero senza che venga eseguito. Ci concentriamo sulla documentazione e sul codice, e ci accertiamo che non presentino errori che non li rendano conformi alle norme, o che non producano risultati attesi. Per fare ciò possiamo utilizzare dei metodi di lettura oppure dei metodi formali. Iniziamo concentrandoci sui metodi di lettura, possono essere svolti da esseri umani o tramite processi automatizzati, e si suddividono in:

- Walkthrough: a priori non conosciamo l'errore, ne sappiamo la sua locazione, quindi andiamo ad analizzare il testo in maniera esplorativa. I verificatori si occupano di quest'ultimo passo, la correzione sarà poi a carico degli autori del testo in questione. Questo processo potrebbe risultare particolarmente laborioso in proporzione alla quantità di dati da ispezionare.
- Inspection: sappiamo dove si potrebbe situare l'errore e quindi a nostro vantaggio possiamo essere selettivi rispetto alle porzioni di testo da esaminare. Come per il Walkthrough sono gli autori a correggere l'errore, mentre i verificatori lo cercano. Questo tipo di metodo ha come presupposto la conoscenza del problema a priori, questa conoscenza è ottenibile tramite esperienza.

3.3.3 Analisi Dinamica

L'analisi dinamica consiste nell'eseguire l'oggetto da verificare. Durante l'esecuzione si possono eseguire dei test, che devono essere **ripetibili** e **automatizzabili**. I test devono essere ripetibili per evitare regressioni del software e automatizzati per essere più efficienti. Per farlo si usano:

- **Driver**, che permette di invocare il componente da testare se ancora nessuno lo chiama;
- **Stub**, che evita le chiamate a moduli non testati;
- **Logger**, che traccia i test.

Vediamo le tipologie di test più comuni:

- **Unità**: verificano il singolo componente in isolamento. Si suddividono in funzionali e strutturali, i primi si basano sulle specifiche funzionali e non sull'implementazione, cioè verificano il comportamento dell'unità in base alle specifiche, i secondi invece sulla struttura interna del codice, come ad esempio la copertura dei rami.
- **Regressione**: verificano che modifiche o correzioni non introducano nuovi errori in parti del codice già verificate. Ripetiamo i test già eseguiti.
- **Integrazione**: verifichiamo che più unità già testate singolarmente possano essere integrate, cioè interagiscano correttamente.
- **Sistema**: verificano la totalità del sistema rispetto ai requisiti. Andiamo a collaudare il sistema con il committente, questo fa parte del processo di validazione, trattato nella sezione successiva

3.4 Processo di validazione

La validazione serve a verificare se i requisiti e le aspettative della proponente sono stati soddisfatti tramite test di accettazione. Questi test simulano uno scenario realistico, permettendo di mostrare il prodotto alla proponente e raccogliere il suo feedback.

3.5 Processo di revisione congiunta

3.6 Processo di risoluzione dei problemi

3.7 Gestione della qualità

4 Processi Organizzativi

4.1 Gestione

La gestione descrive le modalità di comunicazione interne ed esterne, descrivendo i vari ruoli e i loro compiti.

4.1.1 Ruoli di progetto

Durante lo sprint ogni membro del gruppo assume un singolo ruolo. Il ruolo può variare da sprint a sprint. Di seguito elenchiamo le responsabilità di ogni ruolo:

- Responsabile: gestisce la distribuzione di attività tra i membri, assegnando le attività basandosi sull'ammontare ore rimasto per ogni ruolo ed ogni membro. Si occupa inoltre della redazione dei verbali, della comunicazione con la proponente e dell'aggiornamento del piano di progetto.
- Amministratore: si occupa del corretto svolgimento del progetto, manutenendo l'infrastruttura, e controllando il corretto aderimento alle pratiche sprint. Si occupa anche della scrittura delle norme di progetto.
- Analista: concepisce i casi d'uso basandosi sui requisiti ricavati dalle richieste della proponente. Si occupa della scrittura dell'analisi dei requisiti e del piano di qualifica.
- Progettista: traduce i requisti individuati dagli analisti in un progetto, questo include la scelta delle tecnologie da utilizzare, la struttura interna del progetto, cioè le sue parti e sotto-parti. Si occupa anche della supervisione dello sviluppo, verificando il corretto aderimento al progetto.
- Programmatore: il suo ruolo è sviluppare il progetto generato dal progettista. Lavorando con quest'ultimo implementa tutte le funzionalità richieste tramite le tecnologie prestabilite. Si occupa anche della creazione di test automatizzati, utilizzati per verificare la correttezza del codice sviluppato.
- Verificatore: si occupa di verificare che il prodotto sia a regola d'arte, dunque aderisca alle norme di progetto. Si occupa di controllare la documentazione e il codice in cerca di possibili errori, che siano stilistici o di contenuto, e avvisare gli autori in questione del problema.

4.1.2 Attività

Appoggiandoci allo standard ISO/IEC 12207:1997 individuiamo le seguenti attività:

- Processo di gestione.
- Processo di infrastruttura.
- Processo di Miglioramento.
- Processo di Formazione. Le sezioni di seguito tratteranno i processi sopra scritti in dettaglio.

4.2 Infrastruttura

Il processo di Infrastruttura fornisce il supporto tecnico necessario per lo sviluppo e la gestione del progetto.

4.2.1 Attività di processo

Il processo si articola nelle seguenti attività principali:

- **Implementazione:** scelta, configurazione e gestione degli strumenti e delle tecnologie necessarie per supportare le attività di progetto.
- **Creazione:** sviluppo e manutenzione dell'infrastruttura tecnica, strumenti, procedure e ambienti di sviluppo.
- **Manutenzione:** aggiornamento, monitoraggio e risoluzione di eventuali problemi legati all'infrastruttura esistente.

4.2.2 Procedure di processo

Le seguenti procedure guidano il Team nell'utilizzo degli strumenti di infrastruttura, garantendo un uso coerente ed efficiente delle risorse disponibili.

4.2.3 Strumenti a supporto

4.2.3.1 GitHub

Piattaforma di hosting per il versionamento e la gestione dei contenuti di progetto. Il Team deve sfruttare appieno le potenzialità di GitHub, in particolare per l'integrazione con Jira. Vengo quindi descritti gli Smart Commit, lo standard per la scrittura di commit e la gestione dello stato di vita degli work item.

4.2.3.2 Jira

Strumento di gestione delle attività di progetto adottato per potenzialità e flessibilità del sistema. Permette di tracciare le attività di progetto, assegnarle ai membri del gruppo, monitorare lo stato di avanzamento e gestire le scadenze.

4.2.3.2.1 Automation

Il Team ha deciso di adottare alcune automazioni per facilitare la gestione delle attività di progetto. Le automazioni attualmente implementate sono:

- Make child work items inherit labels from parent work items
- When a commit is made $\rightarrow$ then move issue to in progress
- When all child work items are completed $\rightarrow$ then close parent
- When all sub-tasks are done $\rightarrow$ move parent to done
- When Item In Progress $\rightarrow$ Parent In Progress
- Diario Assegnato $\rightarrow$ Creo Preparazione Slide
- Riunione Assegnata $\rightarrow$ Creo Scrittura Verbale

4.2.3.3 VSCode

Ambiente di sviluppo integrato (IDE) utilizzato per la scrittura del codice e la gestione della documentazione di progetto. Grazie all'estensione Atlassian per **Jira** e **GitHub**, il Team può integrare direttamente le funzionalità di gestione delle attività e del versionamento all'interno dell'IDE, migliorando l'efficienza del flusso di lavoro e uniformando le pratiche di sviluppo.
Si presuppone che tutti i membri del gruppo si adattino allo standard comune. Le estensioni attualmente utilizzate sono:

- **Atlassian: Jira, Rovo Dev, Bitbucket**
- **GitHub Pull Requests and Issues**
- **GitHub Actions**
- **GitHub Codespaces**
- **LaTeX Workshop**

Altre estensioni come GitHub Copilot possono essere utilizzate a discrezione del membro del gruppo, al fine di velocizzare, per esempio, il processo di Documentazione 3.1.

4.3 Miglioramento

4.4 Formazione

5 Metriche della qualità

5.1 Attività per la qualità

Lo standard ISO/IEC 9126-1:1999 individua tre categorie fondamentali di requisiti per la qualità che concorrono alla creazione del prodotto e interconnesse tra loro secondo il Modello a V$_{\text{(Figura 5)}}$ (*V-Model*$_{G}$):

- **Bisogni utente**: sono i requisiti che il prodotto deve soddisfare per incontrare i bisogni dell'utente. L'output corrispondente e' la qualità in uso, verificata da apposite metriche esterne.
- **Requisiti di qualità esterni**: corrispondono ai requisiti che riguardano le caratteristiche proprie del prodotto da realizzare. Il loro soddisfacimento definisce la qualità esterna del prodotto, validata anch'essa da opportune metriche esterne.
- **Requisiti di qualità interni**: sono i requisiti legati strettamente ai processi di sviluppo software che quando rispettati risultano in codice di qualità, processi efficaci e organizzazione efficiente.

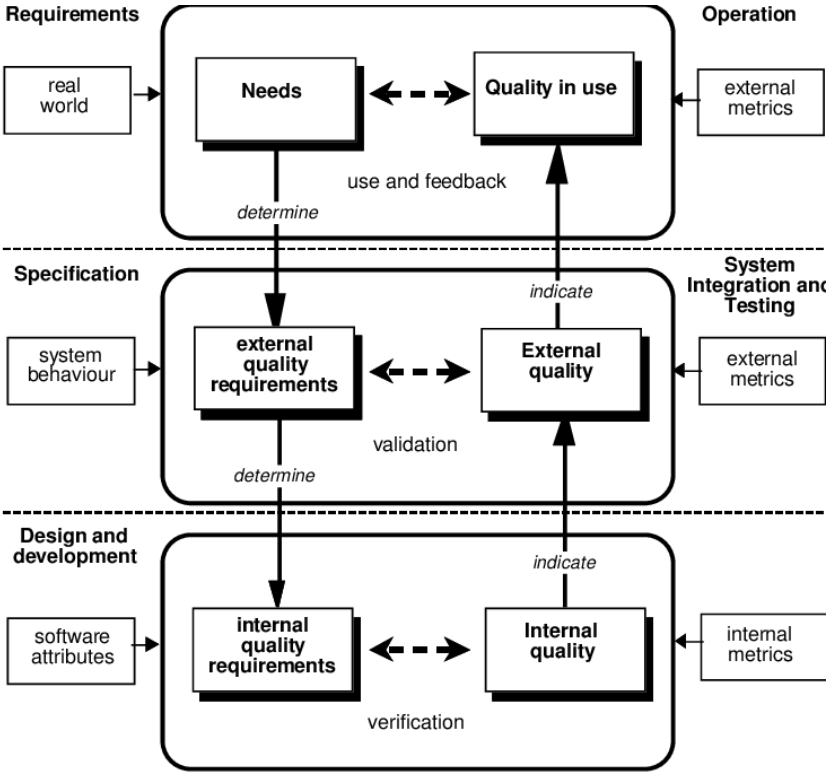

Figura 5: Modello a V

5.2 Procedure per la qualità

Di seguito vengono indicate le metriche utilizzate per valutare la qualità del prodotto e dei processi di sviluppo.

5.2.1 Qualità di Processo

In questa sezione vengono definite le metriche utilizzate per valutare l'efficienza e l'efficacia dei processi del ciclo di vita del software, in conformità con lo standard ISO/IEC 12207.

5.2.1.1 Processo di fornitura

MPC-1 - Earned Value (EV)

Descrizione: Valore del lavoro effettivamente completato in un determinato momento, espresso in termini di budget approvato.

Formula di calcolo:

$$EV = \sum_i (\%\text{Completamento}_i \times \text{Budget}_i)$$

MPC-2 - Planned Value (PV)

Descrizione: Valore del lavoro che si era pianificato di completare entro una certa data.

Formula di calcolo:

$$PV = \sum_i (\%\text{Pianificato}_i \times \text{Budget}_i)$$

MPC-3 - Actual Cost (AC)

Descrizione: Costo realmente sostenuto per il lavoro svolto fino alla data corrente.

Formula di calcolo:

$$AC = \sum \text{Costi Sostenuti}$$

MPC-4 - Cost Performance Index (CPI)

Descrizione: Indice di efficienza dei costi. Un valore < 1 indica che il progetto è fuori budget.

Formula di calcolo:

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

MPC-5 - Schedule Performance Index (SPI)

Descrizione: Indice di efficienza della schedulazione. Un valore < 1 indica che il progetto è in ritardo.

Formula di calcolo:

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

5.2.1.2 Processo di Sviluppo

MPC-11 - Work in Progress (WIP)

Descrizione: Numero di task o ticket su cui il team sta lavorando attivamente in contemporanea. Un valore alto indica colli di bottiglia.

Formula di calcolo:

$$WIP = \sum \text{Ticket in stato "In Progress"}$$

MPC-13 - Change Failure Rate (CFR)

Descrizione: Percentuale di deployment o merge in produzione che causano un guasto richiedendo un hotfix.

Formula di calcolo:

$$CFR = \left(\frac{\text{Numero di deployment falliti}}{\text{Numero totale di deployment}} \right) \times 100$$

5.2.1.3 Processo di Documentazione

MPC-14 - Indice di Gulpease

Descrizione: Indice che valuta la leggibilità di un testo in lingua italiana. Valori bassi indicano bassa leggibilità.

Formula di calcolo:

$$G = 89 + \frac{300 \times (\text{numero frasi}) - 10 \times (\text{numero lettere})}{\text{numero parole}}$$

MPC-15 - Correttezza Ortografica

Descrizione: Misura la densità di errori ortografici presenti nella documentazione.

Formula di calcolo:

$$C = 1 - \left(\frac{\text{Numero errori ortografici}}{\text{Numero parole totali}} \right)$$

5.2.1.4 Processo di Configurazione

MPC-16 - Build Success Rate (BSR)

Descrizione: Percentuale di build completate con successo dalla pipeline di CI/CD rispetto al totale delle build eseguite.

Formula di calcolo:

$$BSR = \left(\frac{\text{Builds con esito positivo}}{\text{Builds totali}} \right) \times 100$$

MPC-17 - Average Build Time

Descrizione: Tempo medio impiegato dalla pipeline per completare il processo di build e test automatici.

Formula di calcolo:

$$T_{avg} = \frac{\sum_{i=1}^{n} T_{build\_i}}{n}$$

5.2.1.5 Processo di Verifica

MPC-19 - Statement Coverage

Descrizione: Percentuale di istruzioni (statements) del codice sorgente eseguite durante i test automatici.

Formula di calcolo:

$$SC = \left(\frac{\text{Linee di codice eseguite}}{\text{Linee di codice totali}} \right) \times 100$$

MPC-20 - Branch Coverage

Descrizione: Percentuale di rami decisionali (if, switch, loop) percorsi durante i test.

Formula di calcolo:

$$BC = \left(\frac{\text{Rami percorsi}}{\text{Rami totali}} \right) \times 100$$

5.2.1.6 Processi Organizzativi)

MPC-21 - Retrospective Effectiveness

Descrizione: Misura la capacità del team di risolvere i problemi identificati durante le retrospettive.

Formula di calcolo:

$$E_{retro} = \left(\frac{\text{Action items risolti}}{\text{Action items identificati}} \right) \times 100$$

MPC-22 - Meeting Attendance

Descrizione: Percentuale di presenza dei membri del team alle riunioni obbligatorie.

Formula di calcolo:

$$MA = \left(\frac{\text{Presenze effettive}}{\text{Presenze attese totali}} \right) \times 100$$

5.2.2 Qualità di Prodotto

In questa sezione vengono definite le metriche per valutare la qualità intrinseca del prodotto software, basandosi sul modello ISO/IEC 9126.

5.2.2.1 Funzionalità

MPD-2 - Completezza dell'implementazione

Descrizione: Misura la percentuale di requisiti funzionali obbligatori implementati e testati.

Formula di calcolo:

$$CI = \left(\frac{\text{Requisiti funzionali implementati}}{\text{Requisiti funzionali totali}} \right) \times 100$$

MPD-3 - Accuratezza dei risultati

Descrizione: Percentuale di test funzionali che producono il risultato atteso senza deviazioni.

Formula di calcolo:

$$Acc = \left(\frac{\text{Test funzionali passati}}{\text{Test funzionali eseguiti}} \right) \times 100$$

5.2.2.2 Affidabilità

MPD-6 - Tasso Errori HTTP

Descrizione: Percentuale di richieste al server che restituiscono codici di errore (4xx o 5xx) rispetto al totale delle richieste.

Formula di calcolo:

$$E_{http} = \left(\frac{\text{Richieste con status } \geq 400}{\text{Richieste totali}} \right) \times 100$$

MPD-NEW - Availability (Disponibilità)

Descrizione: Percentuale di tempo in cui il sistema è operativo e accessibile agli utenti.

Formula di calcolo:

$$A = \left(\frac{\text{Tempo totale} - \text{Tempo di disservizio}}{\text{Tempo totale}} \right) \times 100$$

5.2.2.3 Usabilità

MPD-9 - Profondità di Navigazione

Descrizione: Numero massimo di click necessari per raggiungere una qualsiasi pagina di contenuto partendo dalla Home Page.

Formula di calcolo:

$$Clicks \leq 3$$

MPD-10 - Search Success Rate

Descrizione: Percentuale di ricerche effettuate dagli utenti che portano al clic su un risultato utile.

Formula di calcolo:

$$SSR = \left(\frac{\text{Ricerche con clic}}{\text{Ricerche totali}} \right) \times 100$$

5.2.2.4 Efficienza

MPD-11 - Time to First Byte (TTFB)

Descrizione: Tempo che intercorre tra l'invio della richiesta HTTP dal client e la ricezione del primo byte di risposta dal server.

Formula di calcolo:

$$TTFB = T_{ricezione\_primo\_byte} - T_{invio\_richiesta}$$

MPD-13 - Tempo di Rendering

Descrizione: Tempo necessario al browser per completare il rendering visivo della pagina (First Contentful Paint).

Formula di calcolo:

T_{render} (misurato tramite API del browser)

5.2.2.5 Manutenibilità

MPD-16 - Complessità Ciclomatica

Descrizione: Misura la complessità del flusso di controllo del codice. Valori superiori a 15 indicano codice difficile da testare e manutenere.

Formula di calcolo:

$$M = E - N + 2P \quad (\text{dove } E = \text{archi}, N = \text{nodi}, P = \text{componenti connessi})$$

MPD-17 - Comment Ratio

Descrizione: Rapporto tra le righe di commento e le righe di codice totali (LOC).

Formula di calcolo:

$$CR = \frac{\text{Linee di commento}}{\text{Linee di Codice (SLOC)}}$$

MPD-18 - Accoppiamento tra Classi (CBO)

Descrizione: Numero di classi a cui una determinata classe è accoppiata (usa o è usata da). Un valore alto indica scarsa modularità.

Formula di calcolo:

$CBO =$ Conteggio dipendenze esterne per classe

5.3 Strumenti

7-ZPUs