Università degli Studi di Padova

Catch em All - ***CAPTCHA: Umano o Sovraumano?***
Email: catchemallswe3@gmail.com

Norme di progetto

Versione	2.0.0
Approvazione	Zhen Wei Zheng
Redazione	Gabriele Da Re, Zhen Wei Zheng, Luca Brugnera, Matteo Stocco
Verifica	Sinicato Nicola, Gabriele Da Re, Ana Lazic
Stato	Approvato
Uso	Interno
Distribuzione	Prof. Vardanega Tullio, Prof. Cardin Riccardo,
	Gruppo Catch Em All

Registro delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
2.0.0	1/07/2023	Approvazione documento	Matteo Stocco	Responsabile
1.1.0	30/06/2023	Verifica complessiva del documento	Nicola Sinicato, Ana Lazic	Verificatore, Verificatore
1.0.4	20/06/2023	Modifiche alle sezioni §3.5, §4.1.1 e verifica	Matteo Stocco, Zhen Wei Zheng	Amministratore, Verificatore
1.0.3	04/05/2023	Aggiunta sezione §2.2.4 e verifica	Luca Brugnera, Ana Lazic	Amministratore, Verificatore
1.0.2	03/05/2023	Modifiche alle sezioni §2.3.4.2, §3.1.4	Luca Brugnera, Ana Lazic	Amministratore, Verificatore
1.0.1	12/04/2023	Aggiunte le sezioni §2.2.3.4, §2.2.3.5, §2.2.3.7 e verifica	Gabriele Da Re, Matteo Stocco	Amministratore, Verificatore
1.0.0	09/03/2023	Approvazione documento	Zhen Wei Zheng	Responsabile
0.2.0	07/03/2023	Verifica complessiva del documento	Gabriele Da Re, Ana Lazic, Nicola Sinicato	Verificatore, Verificatore, Verificatore
0.1.5	01/03/2023	Revisione con relative modifiche sezione §3	Luca Brugnera	Amministratore
0.1.4	12/02/2023	Modifiche alle sezioni §2 e §3	Matteo Stocco	Amministratore
0.1.3	22/01/2023	Fine aggiunta e controllo delle pedici delle parole da glossario	Zhen Wei Zheng	Verificatore
0.1.2	11/01/2023	Aggiunto punto sezione §3.4.2	Gabriele Da Re	Verificatore
0.1.1	11/01/2023	Fine della verifica globale del documento	Sinicato Nicola, Gabriele Da Re	Verificatore, Verifcatore

0.1.0	10/01/2023	Inizio della verifica globale del documento	Sinicato Nicola	Verificatore
0.0.9	10/01/2023	Aggiunta sezione §3.4.2.2	Gabriele Da Re	Amministratore
0.0.8	02/01/2023	Aggiunte appendici	Matteo Stocco	Amministratore
0.0.7	28/12/2022	Aggiunte parti nella sezione §3 e §4	Matteo Stocco	Amministratore
0.0.6	24/12/2022	Completato sezione §2	Matteo Stocco	Amministratore
0.0.5	29/11/2022	Completato sezione §3.1 e revisione	Zhen Wei Zheng, Nicola Sinicato	Amministratore, Verificatore
0.0.4	23/11/2022	Definita e implementata la sezione §2 del documento	Luca Brugnera, Gabriele Da Re	Amministratore, Amministratore
0.0.3	22/11/2022	Stesura sezione §3.1	Zhen Wei Zheng	Amministratore
0.0.2	16/11/2022	Impostazione layout documento	Zhen Wei Zheng	Amministratore
0.0.1	15/11/2022	Creazione e prime definizioni del documento	Luca Brugnera	Amministratore

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento ha come obiettivo il fissare gli standard che permetteranno al gruppo *Catch Em All* di garantire qualità al prodotto$_G$ e ai processi durante l'intera durata del progetto. Verranno quindi definiti metodi di verifica$_G$ e validazione$_G$ continui che permetteranno al gruppo di agire in modo rapido e incisivo nel momento in cui si dovranno fare delle correzioni su eventuali errori o andamenti indesiderati. Questo allo scopo di sprecare meno risorse possibili e produrre un prodotto che sia facilmente manutenibile.

1.2 Scopo del prodotto

Dal proponente *Zucchetti S.p.A.* viene evidenziata, nel capitolato da loro proposto, una criticità negli attuali sistemi di sicurezza sulla rilevazione dei bot$_G$ rispetto agli esseri umani. Oggigiorno il meccanismo più utilizzato per risolvere questo problema è il test CAPTCHA$_G$.
Un bot$_G$ non è altro che una procedura automatizzata che, in questo caso, ha fini malevoli, come per esempio:

- Registrazione presso siti web;
- Creazione di spam$_G$;
- Violare sistemi di sicurezza.

I bot$_G$, grazie alle nuove tecnologie sviluppate con sistemi che utilizzano principalmente l'intelligenza artificiale, riescono a svolgere compiti che fino a poco tempo fa venivano considerati impossibili da svolgere per una macchina.
Ciò evidenzia che i CAPTCHA$_G$ attuali risultano sempre più obsoleti, non andando a individuare correttamente tutti i bot$_G$, se non quasi nessuno.
Un'altra criticità individuata dal proponente è il sistema di classificazione delle immagini che sta effettuando Google grazie al proprio reCAPTCHA$_G$, che attualmente è il sistema più diffuso.
Questa criticità nasce dal beneficio che questa big tech$_G$ ottiene dall'interazione degli utenti nel risolvere i task$_G$ proposte, che portano alla creazione di enormi dataset$_G$ di immagini classificate che possono essere utilizzate per l'apprendimento dei propri sistemi di machine learning, oppure possono essere vendute a terzi.
Il capitolato C1 richiede di sviluppare un' applicazione web costituita da una pagina di login provvista di questo sistema di rilevazione in grado di distinguere un utente umano da un bot$_G$.
L'utente quindi, dopo aver compilato il form in cui inserirà il nome utente e la password, dovrà svolgere una task$_G$ che sarà il cosiddetto test CAPTCHA$_G$.

1.3 Glossario

Per evitare ambiguità relative al linguaggio utilizzato nei documenti prodotti, viene fornito il **Glossario v 2.0.0**. In questo documento sono contenuti tutti i termini tecnici, i quali avranno una definizione specifica per comprenderne al meglio il loro significato.
Tutti i termini inclusi nel Glossario, vengono segnalati all'interno del documento Norme di Progetto con una G a pedice.

1.4 Riferimenti

1.4.1 Riferimenti normativi:

- Capitolato C1 "CAPTCHA: umano o sovrumano?" `https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C1.pdf`

1.4.2 Riferimenti informativi:

- Processi di ciclo di vita - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T02.pdf`;
- Il ciclo di vita del Software - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T03.pdf`;
- Gestione di progetto - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T04.pdf`;
- `https://it.wikipedia.org/wiki/ISO/IEC_12207`;
- Approfondimento standard ISO/IEC$_G$ 12207: `https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf`;
- Qualità di prodotto$_G$ - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T08.pdf`;
- Qualità di processo$_G$ - Materiale didattico del corso di Ingegneria del Software: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T09.pdf`;
- Standard SQuaRE: `http://www.iso25000.it/styled/`;
- Standard SPICE: `https://it.frwiki.wiki/wiki/ISO/CEI_15504`;
- Regolamento del progetto didattico – Materiale didattico del corso di Ingegneria del Software:: `https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/PD02.pdf`;

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Il processo di fornitura richiederà al gruppo di definire le norme che dovranno essere rispettate per poter diventare un adeguato fornitore dell'azienda proponente *Zucchetti S.p.A.* e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin. Di conseguenza verranno illustrati i prodotti e documenti che dovranno essere forniti per rispettare i requisiti$_G$ concordati.

2.1.2 Rapporto con il proponente

Durante il corso del progetto il gruppo ha intenzione di instaurare un rapporto di collaborazione con il proponente Dr. Gregorio Piccoli in modo da:

- Approfondire gli aspetti chiave del progetto per far fronte ai suoi bisogni;
- Chiarire i vari dubbi che emergeranno durante il progetto;
- Definire i requisiti$_G$ e vincoli da rispettare;
- Definire una stima dei costi, di tempo e denaro per la durata del progetto;
- Garantire che il prodotto soddisfi le richieste, accordandosi sulla qualifica di questo.

2.1.3 Prodotti e documenti forniti

2.1.3.1 Analisi_dei_requisiti_v.2.0.0

Questo documento, stilato dagli analisti del gruppo, contiene tutti i requisiti$_G$ e casi d'uso$_G$ individuati per il progetto. I requisiti sono ottenuti dal documento di presentazione del capitolato e in seguito integrati sia attraverso discussioni tra i membri del gruppo, sia organizzando incontri con il proponente.

2.1.3.2 Piano_di_progetto_v.2.0.0

Scopo
Questo documento stilato dal responsabile di progetto servirà ad individuare ed organizzare i vari periodi del progetto. Esso conterrà preventivi temporali e costi per ognuna di tali peridi, compiendo un'analisi dei rischi che si possono incontrare durante il corso del progetto.

Analisi dei rischi
In questa sezione si analizzano i diversi rischi in cui il team può incombere durante la durata del progetto. Ogni rischio appartiene ad una specifica categoria, ovvero:

- Rischi personali;
- Rischi tecnologici;
- Rischi organizzativi.

Ogni rischio è inoltre composto da:

- Nome;
- Descrizione;
- Identificazione;
- Precauzioni;
- Pericolosità;
- Stima di manifestazione;
- Conseguenze;
- Piano di contingenza.

Modello di sviluppo
In questa sezione viene specificato il modello di sviluppo che il team ha deciso di adottare, in questo caso il *modello AGILE*$_G$.

Pianificazione
In questa sezione sono contenute le pianificazioni temporali dei periodi in cui il responsabile di progetto ha deciso di suddividere taluni. Ogni periodo è contraddistinto da:

- Nome identificativo;
- Descrizione;
- Periodo;
- Precondizioni;
- Postcondizioni;
- Attività;
- Ruoli attivi.

Ogni periodo è inoltre suddiviso in altri vari periodi temporali per raggruppare al meglio le diverse attività che lo compongono.
Infine ogni periodo possiede un proprio diagramma di Gantt$_G$.

Preventivo
In questa sezione sono contenuti i preventivi sulla distribuzione oraria del lavoro assegnato ad ogni periodo. Inoltre viene preventivato il costo di ognuna ed il costo totale del progetto. Ogni preventivo sarà composto da:

- Due tabelle che indicano le ore e i costi necessari per lo svolgimento del periodo;
- Un istogramma che illustra come sono state distribuite le ore fra i vari membri del gruppo;
- Un grafico a torta che mostra quanto ogni ruolo abbia inciso nel determinato periodo.

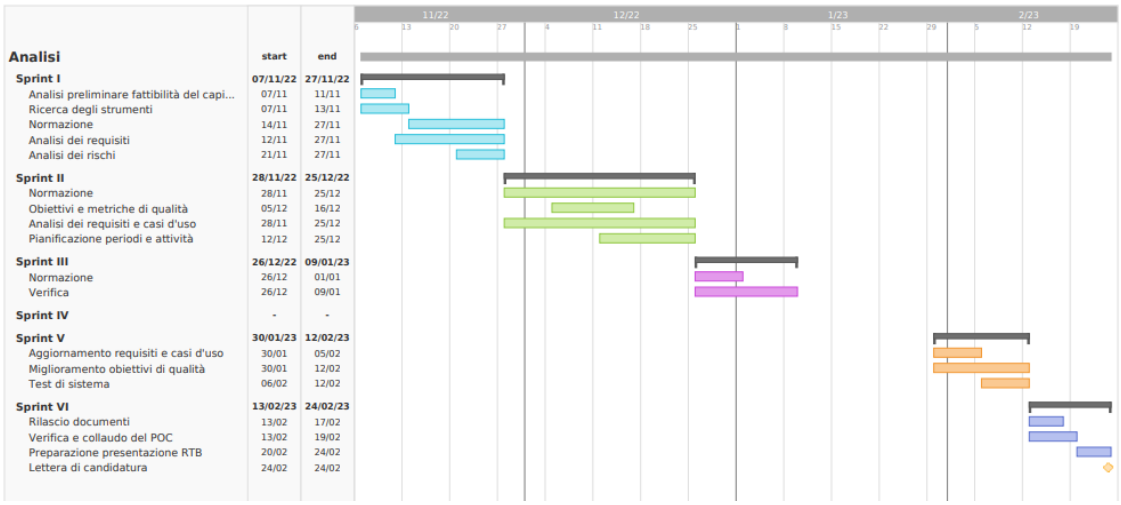

Figura 2.1: Esempio di diagramma di Gantt$_G$

Consuntivo
In questa sezione vengono indicati il numero di ore di lavoro impiegate e i relativi costi effettivi di ogni periodo. Questi vengono poi relazionati con i preventivi fatti nella sezione *preventivo*.

2.1.3.3 Piano_di_qualifica_v.2.0.0

Descrizione
Questo documento, stilato dai membri con il ruolo di analista e di verificatore, contiene i vari obiettivi e metriche che permettono di garantire la qualità della verifica$_G$ e della validazione$_G$ dei processi e dei prodotti del progetto.

Struttura documento
Questo documento è suddiviso in:

- **Obiettivi e metriche di qualità di**:
  - **Processo**: contiene i vari obiettivi generici e specifici e le metriche correlate ad essi che permettono di valutare la qualità di un processo;
  - **Prodotto**: contiene i vari obiettivi e le metriche correlate ad essi che permettono di valutare la qualità di un prodotto.
- **Specifiche dei test**: vengono definiti i vari test che dovranno essere eseguiti, ovvero:
  - Test di unità;
  - Test di integrazione;
  - Test di sistema;
  - Test di regressione;

  - Test di collaudo.

- **Resoconto delle attività di verifica**: vengono illustrati i risultati ottenuti rispetto alle metriche scelte;
- **Risultati dei test**: vengono inseriti i vari risultati ottenuti dai test definiti nella sezione *Specifiche dei test*;
- **Valutazione per il miglioramento**: sono inserite le varie osservazioni e valutazioni fatte dal gruppo sia per poter migliorare l'efficacia e l'efficienza delle varie attività di verifica$_G$ future, che per migliorare i risultati ottenuti nei resoconti delle verifiche.

Struttura obiettivi
Ogni obiettivo sarà contrassegnato da un codice univoco così composto:

`OQ<<Tipo di obiettivo>><<ID>>`

Dove:

- `OQ` sta per obiettivo di qualità;
- `<<Tipo di obiettivo>>` identifica se è di processo o prodotto (PC-PD);
- `<<ID>>` è un contatore correlato al tipo di obiettivo.

Struttura metriche
Ogni metrica sarà contrassegnata da un codice univoco così composto:

`MQ<<Tipo di metrica>><<ID>>`

Dove:

- `MQ` sta per metrica di qualità;
- `<<Tipo di metrica>>` identifica se è di processo o prodotto (PC-PD);
- `<<ID>>` è un contatore correlato al tipo di metrica.

2.1.3.4 Specifica_architetturale_v.1.0.0

Descrizione
Questo documento, stilato dai membri con il ruolo di progettista, conterrà informazioni sulle tecnologie scelte per la realizzazione del prodotto software, assieme all'architettura del sistema e le scelte progettuali effettuate.

Struttura documento
Questo documento è suddiviso in:

- **Tecnologie coinvolte**:
  - **Tecnologie per la codifica**: contiene informazioni sui linguaggi, strumenti, framework$_G$ adottati;
  - **Strumenti per l'analisi del codice**.
- **Architettura**:
  - **Diagrammi delle classi**: diagrammi che descrivono le classi progettate, con le proprietà di ciascuna classe e le relazioni tra le classi;
  - **Architettura di dettaglio**.

2.1.3.5 Manuale_utente_v.1.0.0

Descrizione
Questo documento, stilato dai membri con il ruolo di amministratore, conterrà le istruzioni utili per l'utilizzo del prodotto software, in modo che l'utente abbia una fonte di informazioni sempre consultabile in caso di necessità.

Struttura documento
Questo documento è suddiviso in:

- **Requisiti**: specifica i requisiti minimi per poter usufruire del prodotto software;
- **Installazione**: contiene brevi indicazioni per installare il prodotto software;
- **Istruzioni per l'uso**: contiene indicazioni chiare e schematiche per l'utilizzo del prodotto software.

2.1.3.6 Proof of Concept

Un esempio di software che va ad analizzare alcune sezioni critiche per lo sviluppo del progetto, individuate in seguito ad un'analisi del gruppo. Questo software permetterà di determinare la fattibilità pratica e l'applicabilità di alcuni concetti necessari per la progettazione e codifica del prodotto finale.

2.1.3.7 Prodotto software

Sviluppato successivamente al Poc_G, il prodotto software conterrà le funzionalità descritte nel documento *Analisi dei requisiti v 2.0.0*, e sarà consegato come prodotto software finale.

2.2 Sviluppo

2.2.1 Scopo

L'obiettivo del processo di sviluppo è definire le attività che il gruppo deve eseguire per realizzare il prodotto finale richiesto dal proponente.

2.2.2 Analisi dei requisiti

2.2.2.1 Scopo

Lo scopo di questa attività è quello di stilare tutti i requisiti$_G$ che saranno necessari per il successivo periodo di progettazione e quindi per lo sviluppo di un prodotto che risponda in maniera completa ai bisogni del proponente.
I requisiti individuati dovranno quindi:

- Fornire dei riferimenti affidabili e precisi per il periodo di progettazione;
- Fissare le funzionalità che il prodotto finale dovrà avere;
- Aiutare a definire dei test precisi e dettagliati per agevolare le verifiche future.

2.2.2.2 Contenuti

Il documento stilato dovrà contenere:

- Una descrizione generale del prodotto;
- La scelta delle tecnologie da utilizzare;
- L'analisi dettagliata dei casi d'uso$_G$;
- I requisiti$_G$ individuati tramite:
  - Documento di presentazione del capitolato;
  - Confronti tra i membri del gruppo;
  - Incontri con il proponente.

2.2.2.3 Struttura dei casi d'uso$_G$

Ogni caso d'uso$_G$ è identificato utilizzando la seguente convenzione di nomenclatura:

`UC<<ID_CasoBase>>.<<ID_SottoCaso>>`

Dove:

- `<<ID>>` identifica l'use case;
- `<<ID_SottoCaso>>` identifica eventuali sottocasi.

Ogni caso d'uso$_G$ è composto inoltre da:

- Descrizione: una breve descrizione dell'attività rappresentata dal caso d'uso$_G$;
- Attori$_G$: entità esterne al sistema che interagiscono con esso. Ne esistono di due tipologie:
  - Primario: interagisce con il sistema per raggiungere un obiettivo;
  - Secondario: aiuta l'attore primario a raggiungere l'obiettivo.
- Precondizione: descrive lo stato del sistema prima dell'attività svolta nel caso d'uso$_G$;

- Postcondizione: descrive lo stato del sistema dopo l'attività svolta nel caso d'uso$_G$;
- Scenario principale: elenco che descrive il flusso degli eventi dell'attività rappresentata dal caso d'uso$_G$;
- Scenari alternativi (se presenti): elenco che descrive gli eventi del caso d'uso$_G$ dopo un imprevisto che lo ha deviato dallo scenario principale;
- Scenari inclusi (se presenti): elenco di casi d'uso$_G$ che svolgono attività necessarie allo svolgimento dello scenario principale;
- Generalizzazioni (se presenti): elenco di casi d'uso$_G$ che generalizzano il caso d'uso$_G$ principale.

2.2.2.4 Struttura dei requisiti$_G$

Ogni requisito è identificato da un codice univoco così composto:

`R<<TIPOLOGIA DI REQUISITO>>-<<ID>>`

Dove:

- `<<TIPOLOGIA DI REQUISITO>>` identifica una classe tra le seguenti:
  - Funzionale {F};
  - Qualità {Q};
  - Vincolo {V};
  - Prestazionale {P}.
- `<<ID>>` identifica numericamente il requisito nella classe di appartenenza.

Nel documento i requisiti vengono raggruppati per categoria specificandone:

- Grado di obbligatorietà;
- Descrizione;
- Fonti, le quali possono essere:
  - Il capitolato d'appalto;
  - Verbali interni;
  - Verbali esterni;
  - I casi d'uso$_G$ identificati.

2.2.3 Progettazione

2.2.3.1 Scopo e descrizione

Lo scopo di questa attività è quello di individuare le varie caratteristiche che comporranno il prodotto richiesto dal proponente, avendo come riferimento i requisiti$_G$ e casi d'uso$_G$ definiti nel documento *Analisi_dei_requisiti v 2.0.0*. Le varie caratteristiche verranno poi messe insieme per costruire una singola soluzione che rispetti i vari obiettivi di qualità del prodotto. I vari periodi in cui sarà svolta l'attività di progettazione sono:

- **Proof of concept**;
- **Progettazione architetturale**;
- **Progettazione di dettaglio**.

2.2.3.2 Proof of concept

Scopo
In questo periodo viene prodotto un software esempio, il quale sarà anche la technology baseline$_G$ del prodotto finale. Questo andrà ad analizzare alcune sezioni critiche per lo sviluppo del progetto e servirà ad agevolare le successive scelte di progettazione del gruppo, aiutando a determinare la fattibilità e l'applicabilità di alcune scelte analizzate.

Suddivisione periodi
Questo periodo è diviso in due:

- Periodo nel quale vengono identificati i requisiti$_G$ del Poc_G e le tecnologie necessarie a svilupparlo, oltre che lo studio di queste ultime;
- Periodo di produzione del Poc_G.

2.2.3.3 Progettazione architetturale

Scopo
Lo scopo di questo periodo è il raffinamento della technology baseline$_G$ definita nel periodo di *Proof of Concept*, e discute ad alto livello l'architettura del prodotto e delle sue componenti. Le scelte che il gruppo effettua in questo periodo riguarderanno la struttura complessiva del sistema. Queste ne influenzeranno varie caratteristiche qualitative come per esempio l'efficienza, l'estensibilità e la manutenibilità.

2.2.3.4 Progettazione di dettaglio

Scopo
Lo scopo di questo periodo è definire le specifiche di dettaglio dell'architettura del prodotto e di tutte le sue componenti, scomposte in unità. Queste saranno correlate a diagrammi UML$_G$ che le descriveranno e ai test di verifica$_G$ per la qualità, i quali saranno indicati nel documento *Piano_di_qualifica v2.0.0*. Tali informazioni costituiranno la Product Baseline$_G$, la quale conterrà:

- Desing patterns utilizzati;

- Tecnologie utilizzate;
- Definizione delle classi;
- Tracciamento dei componenti;
- Diagrammi UML_G:
  - Diagrammi delle attività;
  - Diagrammi delle classi.
- Test di integrazione.

2.2.4 Codifica

2.2.4.1 Scopo e descrizione

Il periodo di *Codifica* è assegnato ai membri con il ruolo di programmatore, i quali dovranno realizzare il prodotto software richiesto dal proponente utilizzando ciò che i progettisti hanno definito nel periodo di *Progettazione*.
Per garantire un prodotto adeguato, il codice dovrà essere verificato in modo che rispetti le metriche che garantiscono gli obiettivi di qualità definiti nel documento *Piano_ di_ qualifica v 2.0.0*.

2.2.4.2 Sintassi

Al fine di rendere il codice più leggibile, e agevolare le attività di verifica menzionate nel paragrafo precedente, il gruppo si impegna a seguire le seguenti convenzioni prestabilite:

- Chiarezza dei nomi: i nomi di classi, metodi e variabili devono avere un nome che identifica il loro compito in maniera intuitiva;
- Abbreviazioni nei nomi: sono ammesse parole abbreviate solo se comunemente note, per esempio *Img* è una valida abbreviazione di *Image*;
- Classi: il nome delle classi deve essere scritto in stile *PascalCase*, ovvero iniziare con una lettera maiuscola e, se il nome contiene più parole, tutte le parole inizieranno con una lettera maiuscola;
- Metodi: il nome dei metodi deve essere scritto in stile *camelCase*, ovvero iniziare con una lettera minuscola e, se il nome contiene più parole, tutte le parole inizieranno con una lettera maiuscola;
- Variabili: il nome delle variabili deve essere scritto in stile *snake_ case*, ovvero iniziare con una lettera minuscola e, se il nome contiene più parole, tutte le parole inizieranno con una lettera minuscola e saranno separate dal simbolo _ .

2.2.4.3 Strumenti

Qui di seguito si elencano gli strumenti e tecnologie che sono stati utilizzate durante la codifica del prodotto:

- **Tecnologie**:
  - **Laravel**: Framework$_G$ PHP$_G$ utilizzato per sviluppare il back-end e front-end del prodotto;
  - **Javascript**: Linguaggio di programmazione utilizzato per la gestione dinamica del front-end e per il calcolo del proof of work$_G$;
  - **SQLITE$_G$**: Libreria che implementa un DBMS SQL$_G$ di tipo ACID incorporabile all'interno di applicazioni. Utilizzata per la creazione del database$_G$ utilizzato dall'API;
  - **Python**: Linguaggio di programmazione utilizzato per lo sviluppo dello script$_G$ per scontornare le immagini;
  - **HTML**: Linguaggio di markup utilizzato per definire il layout della pagina di login;
  - **CSS**: Linguaggio utilizzato per definire l'aspetto della pagina di login.
- **Strumenti di supporto**:
  - **Visual Studio Code**: Compilatore e editor di codice utilizzato per lo sviluppo del prodotto.

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

Lo scopo di questa sezione è definire gli standard necessari per la stesura di tutti i documenti del progetto.

3.1.2 Ciclo di vita del documento

Tutti i documenti prodotti dal team seguono il seguente ciclo di vita:

- **Stesura**: Il documento viene scritto utilizzando il modello AGILE$_{\text{G}}$;
- **Revisione**: I documenti modificati devono essere revisionati da un membro del gruppo diverso dal redattore. Solo dopo una revisione con esito positivo le modifiche e i nuovi contenuti possono essere integrati nel documento finale (implementazione attraverso branch protection rules$_{\text{G}}$);
- **Verifica$_{\text{G}}$**: La verifica dei documenti accompagna il progetto lungo tutta la sua durata. Tale attività viene svolta da almeno una persona. Il documento è considerato verificato quando i Verificatori dichiarano che le modifiche necessarie per renderlo coerente con tutte le norme sono state portate a termine;
- **Approvazione**: Il responsabile di progetto dichiara che il documento è completo in ogni sua parte e pronto per essere rilasciato, marcandolo come approvato.

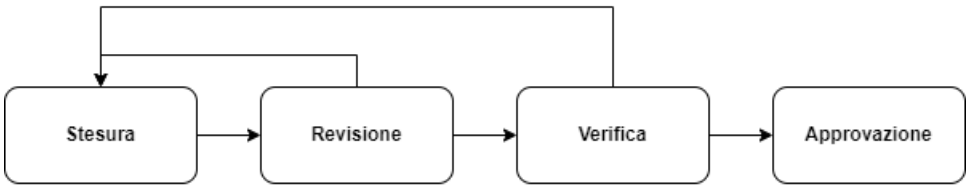

Figura 3.1: Ciclo di vita dei documenti

3.1.3 Struttura dei documenti

Tutti i documenti ufficiali seguono una struttura ben definita, così da mantenere l'omogeneità tra gli stessi. Più precisamente, ogni documento è composto da:

- **Frontespizio**;
- **Registro delle modifiche**;
- **Indice**;
- **Contenuto principale**.

3.1.3.1 Frontespizio

Rappresenta la pagina iniziale del documento ed è strutturato come segue:

- **Logo dell'università**: Logo dell'*Università di Padova*, posizionato nella parte centrale alta della pagina, seguito dalla nomenclatura "Università degli Studi di Padova";
- **Logo del gruppo**: Logo del gruppo, posizionato in centro, a seguito della nomenclatura dell'università;
- **Nome del gruppo e del progetto**: Il nome del gruppo e del capitolato scelto, seguito dal recapito email del gruppo;
- **Nome del documento**: Il titolo del documento, definito in grassetto e posizionato al centro della pagina;
- **Tabella di descrizione**: Tabella contenente le informazioni generali del documento.

3.1.3.2 Registro delle modifiche

I documenti che sono soggetti a modifiche periodiche sono dotati di un registro che ne memorizza lo storico. Questo è impostato come segue:

- **Versione**: Indica la versione del documento dopo una particolare modifica;
- **Descrizione**: Descrive brevemente la modifica apportata;
- **Data**: Indica la data in cui è stato modificato il documento.

3.1.3.3 Indice

Per agevolare la lettura, tutti i documenti sono dotati di un indice. Le sezioni sono rappresentate da un numero identificativo, seguito dal titolo della sezione. Ogni sottosezione deve riportare il numero della sezione genitore e poi il proprio numero identificativo. I numeri partono dall'uno.

3.1.3.4 Contenuto principale

Le varie pagine di contenuto sono costituite da:

- **Intestazione**: In alto a sinistra deve esserci il nome del gruppo *Catch em All*, mentre in alto a destra si trova il numero e il nome della sezione in cui ci si trova;
- **Piè di pagina**: In basso sinistra si trova il nome del documento e la sua versione attuale, mentre in basso a destra viene indicato il numero della pagina in cui ci si trova assieme al numero di pagine complessive del documento.

3.1.4 Classificazione dei documenti

Tutti i documenti prodotti sono divisi in:

- **Uso interno**: Sono documenti finalizzati ad un uso interno al gruppo, ovvero *Norme di progetto* e *Verbali interni*;
- **Uso esterno**: Sono documenti di interesse a tutti gli stakeholder, ovvero *Analisi dei requisiti*$_G$, *Verbali esterni*, *Piano di progetto*, *Piano di qualifica*, *Glossario*, *Specifica architetturale*, *Manuale utente*.

3.1.5 Norme tipografiche

3.1.5.1 Nome del file

Di seguito viene descritto il formato dei nomi dei documenti:

- Iniziano tutti con la lettera minuscola;
- Se il nome comprende più parole allora ognuna di esse è separata dal simbolo '_';
- Deve essere seguito dalla versione in cui si trova.

La sigla della versione deve essere così strutturata:

_v.X.Y.Z

Dove:

- **X**: Indicato da un numero che parte da 0, corrisponde al numero di approvazioni del documento da parte del responsabile;
- **Y**: Indicato da un numero che parte da 0, corrisponde al numero di verifiche del documento da parte dei verificatori, viene portato a 0 ad ogni incremento di **X**;
- **Z**: Indicato da un numero che parte da 0, corrisponde al numero di modifiche del documento da parte del redattore, viene portato a 0 ad ogni incremento di **X** o **Y**.

Esempi corretti: Norme_di_progetto_v.0.0.1, Norme_di_progetto_v.0.2.1, Norme_di_progetto_v.1.0.0.
Esempi non corretti: Norme_di_progetto, NormeDiProgetto.
I verbali non seguono questa norma e hanno una nomenclatura diversa, poiché non subiscono variazioni dopo la prima redazione e hanno il seguente formato:

`V<<Tipo di verbale>>_<<Data verbale>>`

Dove:

- `V` sta a indicare che si tratta di un verbale;
- `Tipo di verbale` indica se è interno (I) o esterno(E);
- `Data verbale` indica la data in cui è stato redatto, ed deve essere scritta nel formato YYYYMMDD.

3.1.5.2 Stile di testo

Di seguito vengono riportati i vari stili del testo e i loro utilizzi:

- **Grassetto**: Utilizzato per i termini da descrivere all'interno degli elenchi puntati e per i titoli delle varie sezioni del documento. Può essere anche utilizzato anche per evidenziare concetti particolarmente rilevanti;
- **Corsivo**: Utilizzato per il nome e l'email del gruppo, il nome del progetto, riferimenti ad altri documenti e sigle;
- **Link**: Sono collegamenti esterni al documento.

3.1.5.3 Glossario

Le norme da seguire relative al *Glossario* sono:

- Ogni parola presente nel documento *Glossario v 1.0.0* viene contrassegnata con una 'G' a pedice all'interno dei vari documenti dove è utilizzata;
- Se un termine compare nella sua stessa definizione all'interno del *Glossario*, esso viene contrassegnato.

3.1.5.4 Elenchi puntati e numerati

Di seguito viene descritto come utilizzare elenchi puntati e numerati:

- La frase introduttiva all'elenco deve terminare con **':'**;
- Ogni punto dell'elenco inizia con la lettera maiuscola;
- Alla fine di ogni punto vi è un **';'**;
- Dopo l'ultima voce vi è un **'.'**.

3.1.5.5 Sigle

Di seguito viene elencata una lista di sigle le quali si possono trovare nei documenti e i loro significati:

- **Documentazione**:
  - **AdR**: Indica l' *Analisi Dei requisiti*$_G$;
  - **NdP**: Indica le *Norme Di Progetto*;
  - **PdP**: Indica il *Piano Di Progetto*;
  - **PdQ**: Indica il *Piano Di Qualifica*;
  - **Gls**: Indica il *Glossario*.
- **Ruoli**:
  - **Re**: Indica il ruolo di *Responsabile di progetto*;
  - **Am**: Indica il ruolo di *Amministratore di progetto*;
  - **An**: Indica il ruolo di *Analista*;
  - **Pt**: Indica il ruolo di *Progettista*;
  - **Pr**: Indica il ruolo di *Programmatore*;
  - **Ve**: Indica il ruolo di *Verificatore*.
- **Revisioni di progetto**:
  - **RTB**: Indica la prima revisione di progetto, e comprende la Requirements and Technology baseline$_G$;
  - **PB**: Indica la seconda revisione, e comprende la Product Baseline$_G$;
  - **CA**: Indica la terza revisione, e comprende la Customer Acceptance.

3.1.5.6 Formato della data

Il team ha adottato il seguente formato per le date all'interno dei documenti:

DD-MM-YYYY

Dove **DD** indica il giorno, **MM** indica il mese e **YYYY** indica l'anno.

3.1.6 Elementi grafici

3.1.6.1 Tabelle

Ogni tabella del documento deve:

- Essere centrata orizzontalmente;
- Essere accompagnata da una didascalia che indichi il numero della tabella all'interno del documento - l'unica eccezione è la tabella delle modifiche che non necessita di didascalie.

3.1.6.2 Immagini

Ogni immagine del documento deve:

- Essere centrata orizzontalmente;
- Essere accompagnata da una didascalia che indichi il numero dell'immagine all'interno del documento.

3.1.7 Strumenti

Di seguito vengono elencati gli strumenti usati per redigere i documenti:

- **LaTeX**: Per la produzione dei documenti, il team ha deciso di usare il linguaggio di markup *LaTeX*;
- **Microsoft Word**: Per la stesura delle bozze di alcune parti di documenti;
- **Microsoft Excel**: Per la creazione delle tabelle con i preventivi delle ore e costi che verranno poi inserite nel *PdP*;
- **StarUML**: Per la creazione dei vari UML_G da inserire all'interno dei documenti;
- **LucidChart**: Per discussioni di gruppo sui vari UML_G creati, dato che lo strumento permette modifiche condivise.

3.2 Gestione della configurazione

3.2.1 Scopo

Lo scopo di questa sezione è descrivere come il team ha deciso di mantenere traccia delle attività e del loro risultato durante il corso del progetto.

3.2.2 Sistemi software utilizzati

La gestione delle versioni dei documenti viene effettuata utilizzando il sistema di controllo di versione Git, attraverso il servizio di GitHub_G. Per la scrittura dei verbali interni ed esterni si è deciso di utilizzare il servizio Confluence offerto da JIRA_G. Al fine di una migliore organizzazione sono stati creati 3 $\text{repository}_\text{G}$ per: documentazione, PoC_G e RTB.

3.2.3 Strutture dei $\text{repository}_\text{G}$

Di seguito vengono illustrati i contenuti dei $\text{repository}_\text{G}$:

- **Docs** è il $\text{repository}_\text{G}$ contenente documentazione riguardante:
  - Assegnazione appalto: Con al suo interno i documenti *lettera_candidatura.pdf*, *preventivo_ore_costi_rischi.pdf*, *motivazione_capitolati.pdf*;
  - Norme di progetto;
  - $\text{Analisi dei requisiti}_\text{G}$;

  - Piano di progetto;
  - Piano di qualifica;
  - Glossario;
  - Link dei verbali interni ed esterni;
  - Ricerche ed approfondimenti riguardanti nuove tecnologie da adottare.

- **PoC$_G$** è il repository$_G$ contenente il PoC$_G$, il quale è composto da:
  - PoC$_G$ Immagini;
  - PoC$_G$ Proof of work$_G$.
- **RTB** è il repository$_G$ contenente tutti i documenti (alla versione *1.0.0*) necessari per la revisione RTB, e il link ai verbali che si trovano su Confluence. I documenti contenuti sono pertanto:
  - *Norme_di_progetto v 1.0.0*;
  - *Piano_di_progetto v 1.0.0*;
  - *Analisi_dei_requisiti v 1.0.0*;
  - *Piano_di_qualifica v 1.0.0*;
  - *Glossario v 1.0.0*;
  - Link ai verbali.
- **Dev** è il repository$_G$ contenente il codice utilizzato per lo sviluppo del prodotto. Al suo interno è presente:
  - Script$_G$ per lo scontorno delle immagini;
  - Cartella contenente le parti frond-end e back-end del prodotto, sviluppate utilizzando il framework$_G$ Laravel$_G$.

3.2.4 Struttura delle cartelle su Confluence

Confluence, un servizio fornito da JIRA$_G$, è utilizzato dal gruppo per la scrittura e l'organizzazione di:

- **Verbali**: Interni ed esterni;
- **Sprint$_G$ retrospective**: Contenente le varie analisi retrospettive del gruppo sugli sprint$_G$ svolti.

Tale scelta è stata guidata dalla qualità dei template forniti dallo strumento, i quali facilitano la scrittura e la comprensione di questo tipo di documenti.

3.2.5 Gestione delle modifiche

Per evitare i conflitti tra le modifiche, mantenere in ordine i file e garantire che all'interno del branch$_G$ principale ci siano solo documenti verificati, il gruppo ha deciso che ogni qual volta sia necessaria una modifica in uno specifico documento dovrà essere creato un branch$_G$ per apportarla. La modifica in questione potrà essere integrata nel branch$_G$ principale soltanto se revisionata con successo dal *Ve*.

3.2.6 Tipi di file presenti

Nella repository$_G$ *Docs* sono presenti esclusivamente file *.tex*, *.txt*, *.png* e *.pdf*. Altri file prodotti durante la stesura dei documenti con estensioni diverse da quelle citate vengono escluse attraverso il file *.gitignore*.

3.3 Assicurazione della qualità

3.3.1 Scopo

L'assicurazione della qualità ha lo scopo di accertare che tutti i processi e i prodotti siano conformi con gli obiettivi e le metriche definiti dal gruppo nel documento *Piano_ di_ qualifica v 1.0.0*. Devono essere continuamente osservate sia la qualità di processo$_G$, per garantire una buona gestione del progetto, sia la qualità di prodotto$_G$, per assicurarsi di ottenere un prodotto finale conforme alle richieste del proponente.

3.3.2 Denominazione obiettivi di qualità

Ogni obiettivo sarà contrassegnato da un codice univoco così composto:

`OQ<<Tipo di obiettivo>><<ID>>`

Dove:

- `OQ` sta per obiettivo di qualità;
- `<<Tipo di obiettivo>>` identifica se è di processo o prodotto (PC-PD);
- `<<ID>>` è un contatore correlato al tipo di obiettivo.

3.3.3 Denominazione metriche di qualità

Ogni metrica sarà contrassegnata da un codice univoco così composto:

`MQ<<Tipo di metrica>><<ID>>`

Dove:

- `MQ` sta per metrica di qualità;
- `<<Tipo di metrica>>` identifica se è di processo o prodotto (PC-PD);
- `<<ID>>` è un contatore correlato al tipo di metrica.

3.3.4 Dettagli metriche di qualità

3.3.4.1 Metriche di processo

Le metriche di qualità a cui ogni processo deve essere conforme sono:

- **SPICE**: Riferito alla metrica per misurare il miglioramento continuo (MQPC01);

- **Costo pianificato di progetto**: Riferito alla metrica per misurare l'efficienza dell'utilizzo delle risorse (MQPC02);
- **Costo reale di progetto svolto**: Riferito alla metrica per misurare l'efficienza dell'utilizzo delle risorse (MQPC03);
- **Variazioni nella pianificazione**: Riferito alla metrica per misurare le variazioni dalla pianificazione (MQPC04);
- **Variazioni nei costi**: Riferito alla metrica per misurare le variazioni dei costi dalla pianificazione (MQPC05);
- **Indice di Gulpease**: Riferito alla metrica per misurare l'indice di Gulpease(MQPC06);
- **Correttezza ortografica**: Riferito alla metrica per controllare la correttezza ortografica (MQPC07).

SPICE
Questa metrica è di riferimento ai processi ed è stata scelta dal gruppo per verificare il grado di *capability*$_{\mathrm{G}}$ che ogni processo deve raggiungere. Lo standard definisce vari livelli di *capability*$_{\mathrm{G}}$:

- **Livello 0 - Incomplete process**: Il processo non è implementato oppure è incapace di raggiungere i suoi obiettivi;
- **Livello 1 - Performed process**: Il processo è attivo e può essere completato ma non è sottoposto a controlli;
- **Livello 2 - Managed process**: Il processo processo è attivo e pianificato, e può completare i suoi obbiettivi attraverso vari controlli;
- **Livello 3 - Established process**: Il processo è definito da degli standard;
- **Livello 4 - Predictable process**: Il processo è attivo secondo standard e viene controllato in modo dettagliato per renderlo in futuro prevedibile e ripetibile;
- **Livello 5 - Optimizing process**: Il processo è completamente definito e tracciato, e viene analizzato e migliorato in maniera continua.

Per misurare la *capability*$_{\mathrm{G}}$ si utilizzano i vari attributi di un processo:

- Process performance;
- Performance management;
- Work product management;
- Process definition;
- Process deployment;
- Process measurement;
- Process control;

- Process innovation;
- Process optimization.

Ogni attributo di processo viene valutato su una scala di valutazione NPLF_G.
Il gruppo si impegna a raggiungere un grado di $\textit{capability}_\text{G}$ minimo di 2 per ogni processo.

Costo pianificato di progetto
Questa metrica è di riferimento ai processi ed è stata scelta dal gruppo per indicare il costo totale di progetto pianificato alla data corrente. Il valore si può osservare nella sezione *Preventivo* del Piano di progetto. Questo valore deve essere ≥ 0 e minore del budget totale disponibile.

Costo reale di progetto svolto
Questa metrica è di riferimento ai processi ed è stata scelta dal gruppo per indicare il costo reale impiegato per svolgere il progetto fino alla data corrente. Il valore si può osservare nella sezione *Consuntivo* del Piano di progetto. Questo valore deve essere un intorno_G del BCWS con un errore non superiore al 20%.

Variazioni nella pianificazione
Questa metrica è di riferimento ai processi ed è stata scelta dal gruppo per misurare in che percentuale ci sono state variazioni rispetto alla pianificazione preventivata. Questa metrica si calcola come segue:

$$\textbf{VP} = \frac{100 * (BCWP - BCWS)}{BCWS}$$

Dove:

- **VP** sta per *Variazione pianificazione*;
- **BCWP** sta per *Budgeted Cost of Work Performed*;
- **BCWS** sta per *Budgeted Cost of Work Scheduled*.

Variazioni nei costi
Questa metrica è di riferimento ai processi ed è stata scelta dal gruppo per misurare in che percentuale ci sono state variazioni tra i costi di sviluppo pianificati e quelli reali. Questa metrica si calcola come segue:

$$\textbf{VC} = \frac{100 * (BCWS - ACWP)}{BCWS}$$

Dove:

- **VC** sta per *Variazione costi*;
- **ACWP** sta per *Actual Cost of Work Performed*;
- **BCWS** sta per *Budgeted Cost of Work Scheduled*.

Indice di Gulpease
L'indice di Gulpease è una metrica di riferimento ai prodotti di documentazione che il gruppo ha scelto di utilizzare per verificare la leggibilità della documentazione prodotta. L'indice è tarato sulla lingua italiana e si calcola in questo modo:

$$\mathbf{IG} = 89 + \frac{300 * Nfrasi - 10 * Nlettere}{Nparole}$$

Il gruppo ha scelto come valore minimo di accettabilità 40. Questo viene indicato come limite dato che un valore minore implica una difficoltà di lettura anche per chi ha conferito un diploma di scuola superiore.

Correttezza ortografica
Questa metrica è di riferimento ai prodotti di documentazione ed è utilizzata dal gruppo per assicurare la correttezza ortografica di ogni parola presente nei documenti. Non devono esserci errori grammaticali per far sì che un documento sia accettato.

3.3.4.2 Metriche di prodotto

Le metriche di qualità a cui il prodotto software finale dovrà essere conforme sono:

- **Copertura funzionale**: Che fa riferimento alla metrica *MQPD01*;
- **Tempo di risposta dei servizi all'utente**: Che fa riferimento alla metrica *MQPD02*;
- **Copertura dei test**: Che fa riferimento alla metrica *MQPD03*;
- **Robustezza agli errori**: Che fa riferimento alla metrica *MQPD04*;
- **Completezza di descrizione**: Che fa riferimento alla metrica *MQPD05*;
- **Completezza della guida utente**: Che fa riferimento alla metrica *MQPD06*;
- **Interfaccia utente auto-esplicativa**: Che fa riferimento alla metrica *MQPD07*;
- **Accoppiamento$_{\mathbf{G}}$ di componenti**: Che fa riferimento alla metrica *MQPD08*;
- **Adeguatezza della complessità ciclomatica$_{\mathrm{G}}$**: Che fa riferimento alla metrica *MQPD09*;
- **Completezza della funzione di test**: Che fa riferimento alla metrica *MQPD10*;
- **Browser supportati**: Che fa riferimento alla metrica *MQPD11*.

Copertura funzionale
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare che tutti i requisiti obbligatori del progetto siano stati integrati nel prodotto finale. Questa metrica è calcolata attraverso il rapporto tra il numero di requisiti soddisfatti e quello di requisiti obbligatori totali:

$$\mathbf{CF} = \frac{RqSoddisfatti}{RqTotali}$$

Dove **CF** sta per *Copertura funzionale*.

Tempo di risposta dei servizi all'utente
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per assicurare che i tempi di risposta del prodotto siano accettabili. Un tempo di risposta adeguato in un sistema CAPTCHA$_G$ è molto importante e per questo è un obiettivo fondamentale. Il valore accettabile verrà analizzato in una fase più avanzata di progetto.

Copertura dei test
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare che i test svolti sul prodotto finale coprano tutti i requisiti e casi d'uso identificati. Questa metrica è calcolata attraverso il rapporto tra il numero di requisiti e casi d'uso$_G$ testati e quello di requisiti e casi d'uso$_G$ totali da testare:

$$\textbf{CdT} = \frac{RqUCTestati}{RqUCTotali}$$

Dove **CdT** sta per *Copertura dei test*.

Robustezza agli errori
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare quale parte di tutti gli errori critici, ovvero quelli che possono determinare blocchi del sistema, è stata messa sotto controllo. Questa metrica è calcolata attraverso il rapporto tra il numero di errori critici gestiti e il numero totale di errori critici da gestire in totale:

$$\textbf{RaE} = \frac{ErrCritGestiti}{ErrCritTotali}$$

Dove **RaE** sta per *Robustezza agli errori*.

Completezza di descrizione Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare la percentuale degli scenari d'uso che è descritta nella documentazione rispetto al totale. Questo per poter garantire informazioni complete agli utilizzatori del prodotto. Questa metrica è calcolata attraverso il rapporto tra il numero di scenari descritti e il numero di scenari effettivamente presenti nel dominio$_G$:

$$\textbf{CdD} = \frac{ScenariDescritti}{ScenariPresenti}$$

Dove **CdD** sta per *Completezza di descrizione*.

Completezza della guida utente
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare la percentuale delle funzioni utilizzabili dall'utente che hanno una descrizione completa nei vari manuali. Questa metrica è calcolata attraverso il rapporto tra il numero di funzionalità descritte e il numero di funzionalità totali:

$$\textbf{CdGU} = \frac{FunzDescritte}{FunzTotali}$$

Dove **CdGU** sta per *Completezza della guida utente*.

Interfaccia utente auto-esplicativa
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare la percentuale degli elementi di informazione che sono presentati all'utente inesperto in modo che possa completare un'attività senza un addestramento preliminare o assistenza esterna. Questa metrica è calcolata attraverso il rapporto tra il numero di informazioni fornite all'utente rispetto a quelle di cui avrebbe bisogno per completare ogni passo dell'attività:

$$\mathbf{IUAE} = \frac{InfoFornite}{InfoRichieste}$$

Dove **IUAE** sta per *Interfaccia utente auto-esplicativa.*

Accoppiamento$_G$ di componenti
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per controllare quanti componenti del sistema sono strettamente indipendenti e quanti sono esenti da impatti conseguenti a cambiamenti negli altri componenti. I gradi di accoppiamento sono definiti su una scala di valori indicata qui di seguito:

- Message Coupling (livello ottimale);
- Data Coupling;
- Stamp Coupling;
- Control Coupling (accettabile);
- External Coupling;
- Common Coupling;
- Content Coupling.

Adeguatezza della complessità ciclomatica$_G$
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare quanti moduli software hanno una complessità ciclomatica$_G$ accettabile. Per verificarla il gruppo deciderà una soglia di accettabilità per i vari linguaggi di programmazione e per il tipo di modulo o di funzione utilizzati durante il progetto.
Nelle metriche software la complessità ciclomatica$_G$ è usata per valutare la complessità di un algoritmo$_G$ ed è basata sulla struttura del grafo che rappresenta l'algoritmo$_G$ da misurare. Per calcolarla si fa uso di questa formula:

$$\mathbf{v(G)} = L - N + 2 * P$$

Dove:

- **v(G)**: Numero ciclomatico relativo al grafo G;
- **L**: Numero di archi nel grafo;
- **N**: Numero di nodi del grafo;
- **P**: Numero dei componenti del grafo disconnessi.

Completezza della funzione di test
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare la percentuale di completezza delle funzioni di test implementate. Questa metrica è calcolata attraverso il rapporto tra il numero di test implementati e il numero di test totali da fare:

$$\textbf{CdFT} = \frac{TestImpl}{TestTot}$$

Dove **CdFT** sta per *Completezza della funzione di test*.

Browser supportati
Questa metrica è di riferimento ai prodotti software ed è utilizzata dal gruppo per verificare il numero di browser che supportano il prodotto sviluppato. Il gruppo definirà un grado di accettabilità per la percentuale di browser che deve supportare il prodotto.

$$\textbf{BRS} \geq 75\%$$

Dove **BRS** sta per *Browser supportati*.

3.4 Verifica$_G$

3.4.1 Scopo

La verifica$_G$ ha lo scopo di assicurare che ciascuna attività svolta non abbia introdotto errori, e che soddisfi i requisiti$_G$ e gli obiettivi specificati. A tal fine vengono utilizzate le metriche scelte dal gruppo e descritte in dettaglio nel documento *Piano_di_qualifica v 1.0.0*. La verifica$_G$ deve essere integrata nei processi di *Fornitura*, *Sviluppo* e *Manutenzione*.

Il processo di verifica$_G$ è suddiviso in due fasi:

- **Analisi statica**, la quale non richiede l'esecuzione dell'oggetto di verifica$_G$;
- **Analisi dinamica**, la quale richiede l'esecuzione dell'oggetto di verifica$_G$.

3.4.2 Analisi statica

L'analisi statica si occupa di analizzare la documentazione e il codice, e accerta la conformità alle regole introdotte, l'assenza di errori, e la completezza dei requisiti$_G$ desiderati. Inoltre, poiché non richiede l'esecuzione dell'oggetto di verifica$_G$, si può applicare ad ogni prodotto di processo.
Si utilizzano due metodi per svolgere analisi statica:

- **Walkthrough**;
- **Inspection**.

Questi metodi di analisi statica vengono effettuati tramite studio dell'oggetto di verifica$_G$ e lettura umana o automatizzata.

3.4.2.1 Walkthrough

Scopo
I verificatori, assieme agli sviluppatori quando necessario, utilizzano questo metodo per rilevare la presenza di errori attraverso una lettura critica ad ampio spettro del prodotto da analizzare. Questo metodo è molto oneroso dal punto di vista delle risorse utilizzate, e perciò si cercherà di utilizzare solo fino al momento in cui non sarà disponibile una checklist di controlli da effettuare.

Fasi:

1. Pianificazione, svolta da autori e verificatori;
2. Lettura, svolta dai verificatori;
3. Discussione, svolta da autori e verificatori;
4. Correzione degli errori, svolta dagli autori.

3.4.2.2 Inspection

Scopo
I verificatori che utilizzano questo metodo devono rilevare la presenza di errori eseguendo una lettura mirata dell'oggetto di verifica$_G$ attraverso l'utilizzo di una checklist. Si cerca quindi di immaginare in precedenza quali saranno le criticità dell'oggetto da analizzare e di elencarli.

Fasi:

1. Pianificazione;
2. Definizione di una checklist;
3. Lettura;
4. Correzione degli errori, svolta dagli autori.

Per garantire e mantenere un'elevata qualità dei documenti durante il loro intero sviluppo (quindi lungo l'intero arco di vita del progetto), il gruppo si è impegnato nello sviluppo di 2 programmi di analisi statica: uno riguardante i documenti e l'altro per le parole del Glossario.
Etrambi gli script$_G$ sono eseguibili da qualunque ambiente di sviluppo e sono stati sviluppati in Python$_G$.
La loro semplice e breve esecuzione permette un controllo frequente.
Sono stati, inoltre, integrati e resi obbligatori nel repository$_G$ pubblico attraverso le branch protection rules$_G$ di GitHub$_G$.

File correctness script$_G$
Questo script$_G$ permette di controllare la correttezza dei file di testo (ad esempio i file .tex) e delle cartelle.
Il programma permette di verificare determinati elementi:

- All'interno del file;
- Nomi dei file;
- Nomi delle cartelle;
- Strutture delle cartelle.

Queste verifiche sono per la maggior parte delegate ad un modulo, chiamato "specific_rules", il quale contiene tutte le verifiche da fare tramite espresssioni regolari (regex).
La struttura di questo programma prevede una checklist di macro controlli da effettuare (contenuta nel file principale "file_correctness.py"), i quali a loro volta contengono una serie di controlli più specifici (contenuti nel modulo "checks", il quale si avvale del modulo "specifics_rules").
I documenti analizzati sono:

- Analisi dei requisiti$_G$;
- Norme di progetto;
- Piano di progetto;
- Piano di qualifica;
- Glossario.

Le verifiche effettuate dallo script$_G$ sono le seguenti:

- Presenza delle cartelle contententi i file sopra citati;
- Correttezza nome cartella;
- Struttura delle cartelle:
  - Presenza di una sola cartella "src":
    - Presenza di una cartella "sections" (contenente i file LaTeX che compongono il documento);
    - Eventuale presenza della cartella "img" contenente immagini necessarie al documento.
  - Presenza di un unico file "pdf", il quale identifica il documento.

Di seguito vengono elencati i controlli specifici:

- Ogni documento deve avere

```
'font size': r'.*\\documentclass\[10pt\]\{article\}.*',
'packages file': r'.*\\input\{sections/packages\}.*',
'style file': r'.*\\input\{sections/style\}.*',
'title page': r'.*\\input\{sections/title_page\}.*',
'page roman numbering': r'.*\\pagenumbering\{roman\}.*',
'table of contents': r'.*\\tableofcontents.*',
```

- Controllo della presenza di almeno un file LaTeX nella cartella "sections", e di nessun altro tipo di file;
- Controllo della presenza dei file necessari nella cartella "sections", precedentemente inclusi nel file LaTeX in "src"; Ovvero:

```
necessary_sections_files = [
        'style.tex',
        'packages.tex',
        'title_page.tex',
        'modifiche.tex',
        ]
```

- Controllo della presenza di determinati parametri nel file title_page.tex, tra cui il titolo che coincida con il nome della cartella in cui si trova il file;
- Controllo della presenza di determinati parametri nel file style.tex, tra cui la correttezza del nome del file;
- Controllo della presenza di determinati parametri nel file modifiche.tex, tra cui il giusto ordine e l'assenza di ripetizioni nel numero delle versioni;
- Controllo della presenza dell'ultima versione nei file title_page.tex e in style.tex;
- Controllo del nome e della correttezza della versione del file pdf;
- Controllo del rispetto della norma per le gli elenchi puntati e numerati (3.1.5.4).

Glossario script$_G$
Lo script$_G$ per le parole del glossario verifica che queste ultime siano identificate correttamente all'interno dei documenti.
In caso contrario il programma segnala l'errore (o un semplice avviso nel caso in cui non sia in grado di analizzare con certezza un punto del documento), e consente di correggerlo.

3.4.3 Analisi dinamica

L'analisi dinamica si occupa di effettuare dei test sugli oggetti di verifica$_G$, che devono quindi essere eseguiti. Questo permetterà al gruppo di accertarsi dell'assenza di errori noti. I test dovranno essere:

- **Ripetibili**, ovvero devono garantire la correttezza dell'oggetto di verifica$_G$ e quindi la rimozione di eventuali errori;

- **Automatizzati**, ovvero svolti in maniera automatica da strumenti selezionati.

Inoltre verranno eseguiti diversi tipi di test:

- **Test di unità**;
- **Test di integrazione**;
- **Test di sistema**;
- **Test di regressione**;
- **Test di accettazione e collaudo**.

3.4.4 Denominazione test di verifica$_G$

Ogni test sarà contrassegnato da un codice univoco così composto:

`TV<<Tipo di test>><<ID>>`

Dove:

- `TV` sta per test di verifica$_G$;
- `<<Tipo di test>>` identifica il tipo di test che si vuole fare, ovvero:
  - UN, di unità;
  - IN, di integrazione;
  - ST, di sistema;
  - RG, di regressione;
  - AC, di accettazione e collaudo.
- `<<ID>>` è un contatore correlato al tipo di test.

Informazioni più dettagliate sui vari test, strumenti e metriche utilizzate per la verifica$_G$ si possono trovare nel documento *Piano_di_qualifica v 1.0.0*.

3.5 Validazione$_G$ e collaudo

3.5.1 Scopo

Lo scopo di questo processo è quello di confermare che tutti i requisiti$_G$ siano stati soddisfatti con la realizzazione del prodotto finale, il quale deve essere conforme alle richieste del proponente. Si arriva alla validazione$_G$ e al collaudo dopo aver svolto tutti i test di verifica$_G$. La verifica$_G$ è infatti strettamente legata alla validazione$_G$, in quanto un'attenta verifica$_G$ durante il corso del progetto permette di superare con successo anche il processo di validazione$_G$. Sarà il responsabile di progetto che avrà la responsabilità di controllare i risultati decidendo se:

- Accettare il prodotto;
- Rifiutarlo richiedendo una nuova verifica.

3.5.2 Aspettative

Il gruppo dovrà assicurarsi che i seguenti punti siano rispettati:

- Il prodotto software sia conforme con i requisiti riportati nel documento Analisi_dei_requisiti v 2.0.0;
- Dimostrare la correttezza delle attività svolte durante il periodo di verifica.

3.5.3 Validazione della documentazione

Al termine della stesura di un documento il responsabile si occupa di rileggerlo controllandone anche la correttezza grammaticale, ne calcola l'indice di Gulpease e si assicura che gli obiettivi stabiliti siano stati raggiunti. Se il documento soddisfa le attese allora viene approvato per il rilascio, invece se non le soddisfa il responsabile assegna il compito di sistemare le parti carenti e la procedura si ripete.

3.5.4 Validazione del codice

Il verificatore esegue i test di unità, integrazione, sistema e di accettazione e si assicura che il tasso di superamento sia conforme alle metriche definite.

3.6 Usabilità

3.6.1 Scopo

Questo processo ha lo scopo di assicurare che vengano prese in considerazione, ed opportunamente indirizzate, tutte le considerazioni espresse dalle parti interessate - gli stakeholders - relativamente alla facilità d'uso del prodotto finale da parte degli utenti cui è rivolto, al supporto che ne riceverà, alla formazione, all'incremento della produttività, alla qualità del lavoro, all'accettazione del prodotto stesso.

4 Processi organizzativi

4.1 Gestione progetto

4.1.1 Scopo

Lo scopo di questo processo è quello di fornire al gruppo delle linee guida su come gestire l'organizzazione delle varie fasi del progetto, con i punti a seguito elencati:

- Modello di sviluppo;
- Gestione dei ruoli;
- Gestione delle comunicazioni;
- Gestione degli incontri;
- Gestione per il controllo della versione;
- Gestione del GitHub Workflow$_G$;
- Gestione del tracciamento delle attività.

4.1.2 Modello di sviluppo

Il gruppo ha scelto di utilizzare il modello **AGILE$_G$** con framework scrum$_G$.

4.1.2.1 Modello AGILE$_G$

Il modello AGILE$_G$ con framework scrum$_G$ prevede di dividere il progetto in blocchi rapidi di lavoro (Sprint$_G$), alla fine di ciascuno dei quali viene realizzato un incremento nello sviluppo del prodotto. Esso indica come definire i dettagli del lavoro da fare nell'immediato futuro e prevede vari meeting con caratteristiche precise per creare occasioni di ispezione e controllo del lavoro svolto.

I cicli di scrum$_G$, detti anche sprint$_G$, avranno durata che variano da una a quattro settimane. All'inizio di ogni ciclo vi sarà una riunione nella quale si discuterà:

- Resoconto e status dei lavori del ciclo precedente;
- Riepilogare i problemi riscontrati durante il lavoro;
- Definire chiaramente la milestone$_G$ dello sprint$_G$;
- Pianificazione e assegnazione delle attività (task$_G$) da svolgere nel nuovo ciclo e le date delle riunioni intermedie usando Product Backlog Refinement$_G$.

Le riunioni intermedie si terranno ogni settimana e hanno funzione di:

- Comunicare ai membri del gruppo lo stato di avanzamento delle attività dello sprint$_G$;
- Assegnazione delle attività di revisione per le task$_G$ completate.

Verranno inoltre mantenuti costantemente monitorati gli avanzamenti delle attività attraverso JIRA_G e discussioni giornaliere svolte utilizzando gli strumenti di comunicazione scelti dal gruppo. Ogni Sprint_G inizierà il lunedì della settimana.
Si è deciso di dividere il progetto in vari periodi, divisi in sprint_G per gestire meglio i vari processi che comporranno gli vari stadi del progetto. Ogni sprint_G avrà uno scopo preciso e sarà considerato un punto importante da raggiungere per il corretto andamento del progetto.
Il gruppo inizialmente aveva optato per utilizzare sprint_G di durata settimanale per mantenere un controllo stretto sull'andamento del progetto. Ci si è accorti però che spesso uno sprint_G non era fine a se stesso e alcune attività si andavano a prolungare anche nello sprint_G successivo. Il gruppo ha quindi deciso di allungare la durata degli sprint_G definendone chiaramente per ognuno l'obiettivo, che sarà anche una $\text{milestone}_\text{G}$ per il progetto. In questo modo gli sprint_G indicano molto più chiaramente l'andamento del progetto. È stato comunque mantenuto l'incontro settimanale necessario per monitorare l'andamento delle task_G da svolgere.

4.1.3 Gestione ruoli

Durante il corso del progetto il gruppo dovrà gestire i vari ruoli garantendo una suddivisione equa tra i membri e pertinente rispetto al processo in svolgimento.
I sei ruoli possibili da assegnare sono:

- Responsabile;
- Amministratore;
- Analista;
- Progettista;
- Programmatore;
- Verificatore.

4.1.3.1 Responsabile

Il responsabile di progetto è la figura di riferimento con l'esterno, quindi garantisce una buona comunicazione con proponente e committente. Si assume inoltre la responsabilità delle scelte del gruppo, dopo averle approvate.
Il responsabile quindi gestisce:

- Elaborazione di piani e scadenze del progetto;
- Gestire la suddivisione dei ruoli all'interno del gruppo;
- Approvare il rilascio di prodotti parziali o finali, come documentazione o software;
- Analisi e gestione dei rischi.

4.1.3.2 Amministratore

L'amministratore di progetto ha lo scopo di controllare l'efficienza dell'ambiente di lavoro, assicurandosi quindi che gli strumenti di supporto alle norme di progetto siano usati correttamente da tutti i membri del gruppo.
L'amministratore quindi gestisce:

- La corretta applicazione delle norme di progetto;
- La ricerca di nuovi metodi per rendere più efficiente l'ambiente di lavoro;
- Le versioni dei vari prodotti durante il corso del progetto;
- L'analisi di metodi per la gestione della qualità.

4.1.3.3 Analista

A seguito di un'approfondita analisi del capitolato e dai diversi incontri effettuati con il proponente, si occuperà dell'individuazione dei vari requisiti.
L'analista quindi gestisce:

- Lo studio approfondito del dominio$_G$ del problema;
- Il documento *Analisi dei requisiti*$_G$;
- L'individuazione dei requisiti$_G$ del prodotto finale;
- L'analisi dei casi d'uso$_G$.

4.1.3.4 Progettista

Il progettista si occupa di trovare soluzioni tecniche e tecnologiche che possano permettere al gruppo di creare un prodotto che rispetti al meglio tutti i requisiti$_G$ individuati dagli analisti.
Il progettista quindi gestisce:

- La scelta degli aspetti tecnici e tecnologici per la realizzazione del prodotto;
- La scelta dei vari modelli da utilizzare nella definizione dell'architettura del prodotto;
- Di definire l'architettura del prodotto che verrà poi programmato.

4.1.3.5 Programmatore

Il programmatore si occupa di codificare le soluzioni individuate dai progettisti per la creazione del prodotto finale.
Il programmatore quindi gestisce:

- La scrittura del codice in modo che sia chiaro e facile da mantenere;
- Gli strumenti che si occupano dei test utilizzati per la verifica$_G$ e la validazione$_G$ del software;
- La redazione del Manuale Utente relativo alla codifica del prodotto.

4.1.3.6 Verificatore

Il verificatore si occupa di tutte le operazioni di verifica$_G$ e validazione$_G$ dei vari prodotti parziali e finali del progetto.
Il verificatore quindi gestisce:

- La verifica$_G$ e validazione$_G$ dei prodotti parziali o finali, in fase di revisione in modo che rispettino gli obiettivi di qualità imposti nel documento *Piano_di_qualifica v 1.0.0*. I seguenti prodotti se a norma verranno di conseguenza integrati;
- Segnalare gli eventuali errori riscontrati.

4.1.4 Gestione delle comunicazioni

4.1.4.1 Comunicazioni interne

Le comunicazioni interne:

- Riguardano solamente i componenti del team;
- Avvengono su *WhatsApp*;
- Utilizzate per:
  - Comunicazioni istantanee tra tutti i componenti;
  - Discussioni;
  - Pianificazione degli incontri;
  - *daily scrum$_G$ meeting*.

4.1.4.2 Comunicazioni esterne

Le comunicazioni esterne:

- Riguardano il gruppo e le altre figure (proponente e committente);
- Utilizzo del dominio$_G$ di gruppo (*catchemallswe3@gmail.com*) di posta elettronica;
- Utilizzate per comunicazioni ufficiali tra il team e le altre figure.

4.1.5 Gestione degli incontri

4.1.5.1 Incontri interni

Gli incontri interni sono necessari sia per una corretta adozione del *framework Scrum$_G$* sia per permettere al team di interagire direttamente, discutendo, proponendo e valutando idee, problematiche e possibili soluzioni: per questo si tratta di uno strumento largamente utilizzato.
Si predilige la modalità virtuale per comodità cercando di schedulare riunioni in cui tutti riescano a partecipare.
La piattaforma utilizzata è *discord*, la quale permette la creazione e l'utilizzo di:

- Canali testuali;

- Canali video (con possibilità di condivisione schermo).

Al termine degli incontri il responsabile di progetto assegna la task$_G$ di redazione dei verbali a un componente del gruppo nello sprint$_G$ corrente.

4.1.5.2 Incontri esterni

Gli incontri esterni sono schedulati in seguito alla presenza di dubbi (implementativi, riguardanti requisiti$_G$ o richieste di altro tipo) all'interno del team: questi incontri sono preceduti dallo svolgimento di una o più riunioni interne nelle quali si affrontano e si definiscono tali problematiche.
Per quanto riguarda l'organizzazione viene contattato tramite email il referente di progetto proponendogli diverse date e orari, affinchè si trovi quella più comoda per entrambe le parti.
Come per quelli interni gli incontri esterni sono tenuti in modalità virtuale ma a loro differenza si utilizza una riunione *Zoom* definita dal gruppo.
I verbali hanno lo scopo di documentare in maniera dettagliata tutti gli argomenti trattati affinchè si possa costruire uno storico, il quale è volto a identificare e motivare le decisioni prese.
Come per quelli interni il responsabile di progetto inserisce nello sprint$_G$ corrente il compito di redigere tali documenti.

4.1.6 Versionamento

GitHub$_G$ è lo strumento utilizzato dal gruppo per il versionamento del codice.
Il team è identificato in tale piattaforma come organizzazione (vedi). Inoltre, al fine di documentare il più possibile, ogni commit$_G$ che porta valore al progetto contiene il riferimento al ticket che completa (totalmente o anche solo parzialmente).

4.1.7 GitHub Workflow$_G$

Il workflow$_G$ scelto dal gruppo è anche conosciuto come feature workflow$_G$.
Tutti i titoli e le descrizioni dei commit$_G$ devono essere fatti in inglese per conformità tra essi.
Il Workflow viene gestito concorrentemente da GitHub$_G$ e JIRA$_G$.
In JIRA$_G$ vengono create ed organizzate le issue$_G$, una volta fatto ciò si procede attraverso github$_G$ alla creazione del branch$_G$ relativo alla issue$_G$ da risolvere.
Tale ramo ha nome codificato come:

CEA–num–titolo–della–issue\textsubscript{G}

Questo permette di identificare titolo e numero della issue$_G$ di appartenenza. Una volta fatto ciò viene creato un **primo commit$_G$**, nel cui messaggio è specificata l'avvenuta presa in carico della issue$_G$, la quale dovrà passare dallo stato "to do" allo stato "in progress"
Ciò è garantito dal suddetto commit$_G$ message contenente la stringa:

CEA–num #open <testo aggiuntivo>

Una volta fatto ciò è possibile lavorare liberamente sul proprio ramo di feature$_G$.
Ad ogni aggiornamento dell'attività svolta si dovrà fare riferimento alla issue$_G$ e specificare il tempo impiegato per lo svolgimento di tale attività includendo nella descrizione:

- Visual Studio Code;

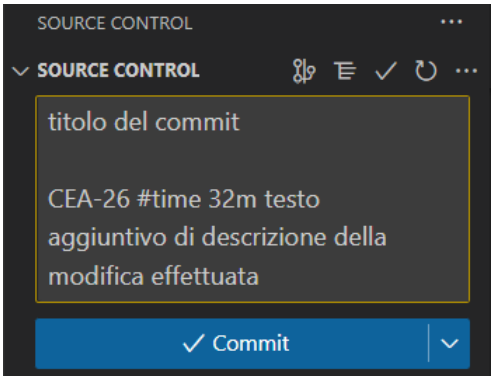

Figura 4.1: Immagine di come scrivere un commit$_G$ su Visual Studio Code

- Git Bash.

```
git commit -m "titolo del commit\textsubscript{G}" -m "CEA-26 #time 32m ...
```

Comando generico da aggiungere nel corpo del messaggio, non nel titolo del commit$_G$:

```
CEA-num #time ww dd hh mm <testo aggiuntivo>
```

Così facendo è permesso specificare a libera scelta le settimane, giorni, ore e minuti di lavoro. Ad esempio:

```
CEA-26 #time 1h aggiunto github Workflow\textsubscript{G}
```

Ciò aggiunge 1h alle ore di lavoro impiegate per la issue$_G$ con ID CEA-26, e come testo aggiuntivo per il commit$_G$ "aggiunto github Workflow$_G$", ignorato da JIRA$_G$.
Una volta terminata l'attività, sarà necessario passare allo stato di revisione, il quale permette di verificare il corretto svolgimento del compito eseguito. Questo è permesso da un ultimo commit$_G$ prima della revisione, con messaggio da includere nella descrizione(non titolo):

```
CEA-num #review #time ww dd hh mm <testo aggiuntivo>
```

Questo permette lo spostamento della issue$_G$ dallo stato "in progress" allo stato "in review".
Per permettere la revisione è necessario aprire una pull request. Il titolo deve corrispondere al nome del branch$_G$. Una volta revisionata la issue$_G$, nel caso in cui presenti qualche problema può essere spostata allo stato "in progress" dal pannello JIRA$_G$. Altrimenti attraverso una pull request nel ramo "main" e con il seguente comando posto **nel TITOLO del commit$_G$ di chiusura della pull request**, la issue$_G$ verrà chiusa e considerata completata:

```
CEA-num #close <testo aggiuntivo>
```

Una volta chiusa, sempre dalla pull request su github$_G$, **si elimina il ramo di feature$_G$** creato precedentemente.

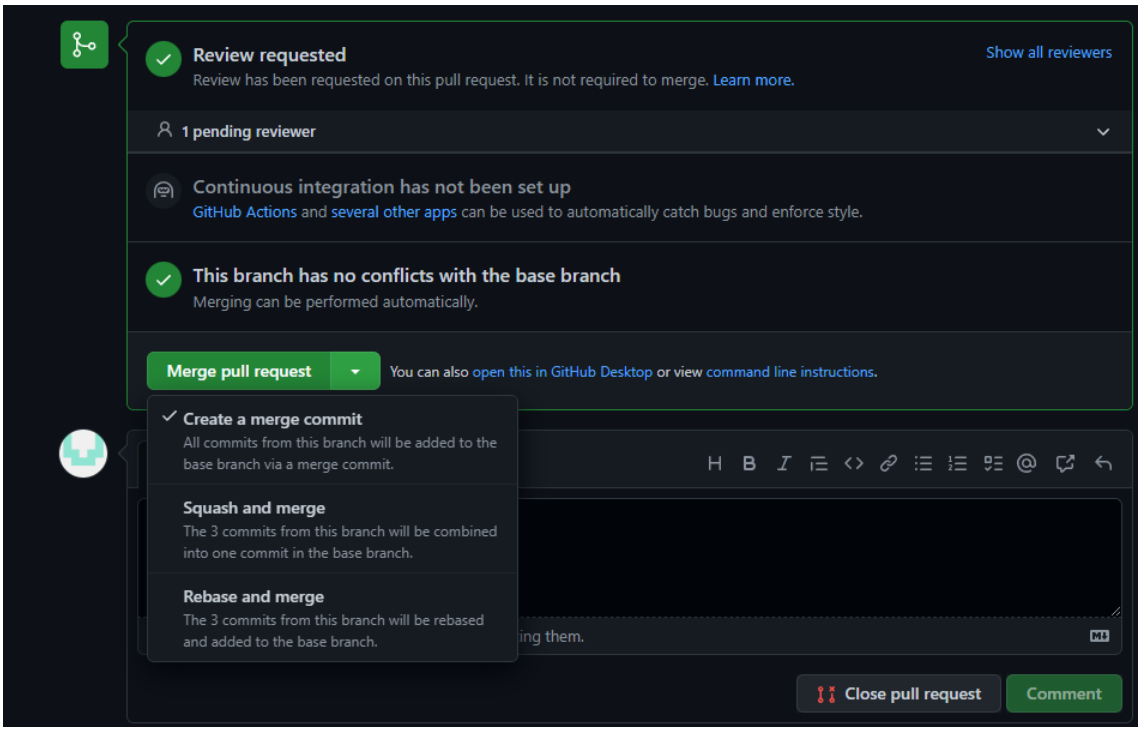

Figura 4.2: Merge$_G$ di una pull request su Github$_G$

La descrizione è a titolo esemplificativo, il contenuto non influenza gli smart commit$_G$ di JIRA$_G$, il **titolo del merge$_G$ commit$_G$** si.

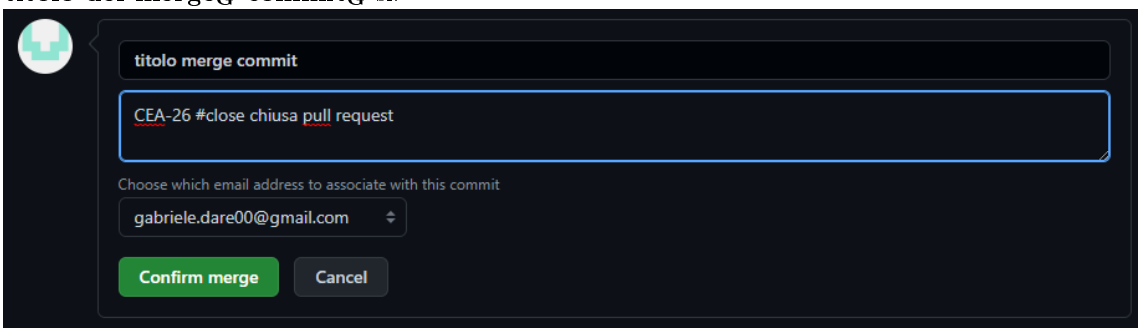

Figura 4.3: Messaggio esempio per effettuare il merge$_G$ di una pull request

4.1.8 Issue tracking

JIRA$_G$, piattaforma che offre un servizio di *Issue$_G$ Tracking*, è il supporto scelto vista la qualità ed il numero di servizi ed estensioni che offre.
La definizione dei ticket è regolata dalla seguente convenzione:

- Titolo e descrizione devono, oltre ad essere sempre presenti, esplicitare in maniera chiara il problema;
- Utilizzo di label;
- Stima del lavoro necessario al completamento;
- Corretto utilizzo di ereditarietà (rapporti di parentela).

Si è deciso di adottare il *framework Scrum$_G$* per la gestione del ciclo di sviluppo del progetto con le seguenti caratteristiche:

- Sprint$_G$ della durata generalmente di 2 o 3 settimane;
- Utilizzo di una board avente 4 stati. I quali sono:
  - To do;
  - In progress;
  - In review (ogni ticket deve essere validato da uno o più componenti del gruppo per essere considerato chiuso);
  - Done.

JIRA$_G$ dispone di un'integrazione con github$_G$ che fornisce un meccanismo chiamato *smart commit$_G$*, il quale permette la transizione dei ticket da uno stato ad un altro, attraverso comandi posti nei commit$_G$ stessi, la sintassi utilizzata è la seguente:

CEA–number #command <message body describing the commit\textsubscript{G}>

Tra i comandi troviamo:

- **Open**: permette di spostarsi da una issue$_G$ nello stadio "to do" oppure "in review" allo stadio "in progress";
- **Review**: permette lo spostamento della issue$_G$ dallo stadio "in progress" oppure "done" allo stadio "in review";
- **Close**: premette di spostarsi dallo stadio "in review" allo stadio "done";
- **Close-no-rev**: permette in casi eccezionali di passare direttamente dallo stadio "in progress" allo stadio "done".

4.1.9 Strumenti

I membri del gruppo lavorano sui seguenti sistemi operativi:

- Windows;
- Linux.

I membri del gruppo, nel corso del progetto, utizzano i seguenti strumenti come descritto nei paragrafi precedenti:

- **Whatsapp**;
- **Discord**;
- **Zoom**;
- **GitHub$_{G}$**;
- **Jira$_{G}$**.

4.1.10 Formazione

La formazione di ogni singolo componente del gruppo avviene in maniera autonoma, tramite documentazione trovata in rete e tramite materiale fornito dal proponente e dai docenti.

A Standard di riferimento

A.1 Standard ISO/IEC$_G$ 12207

A.1.1 Scopo

La norma ha lo scopo principale di definire una struttura comune in modo che i professionisti coinvolti nello sviluppo del software (committenti, fornitori, sviluppatori, manutentori, operatori, manager e tecnici) possano utilizzare un linguaggio comune. Tale linguaggio è basato su una struttura di processi, attività, compiti e risultati prodotti. Il modello è flessibile e modulare in modo che ciascuno possa personalizzarlo a seconda delle proprie esigenze organizzative dei singoli progetti software. Lo standard stabilisce i processi presenti nel ciclo di vita del software e, per ciascuno di essi, le attività da svolgere e i risultati da produrre.

A.1.2 Tipi di Processo

I processi sono suddivisi dalla norma in tre categorie:

- **Processi primari**: I quali comprendono le attività direttamente legate allo sviluppo del software;
- **Processi di supporto**: I quali includono la gestione dei documenti e dei processi di controllo della qualità;
- **Processi organizzativi**: I quali coprono gli aspetti manageriali e di gestione delle risorse.

Per ciascun processo la norma evidenzia chiaramente:

- **Obiettivo**;
- **Responsabilità**;
- **Lista delle attività che lo compongono**;
- **Singoli compiti nei quali è suddivisa ogni attività**.

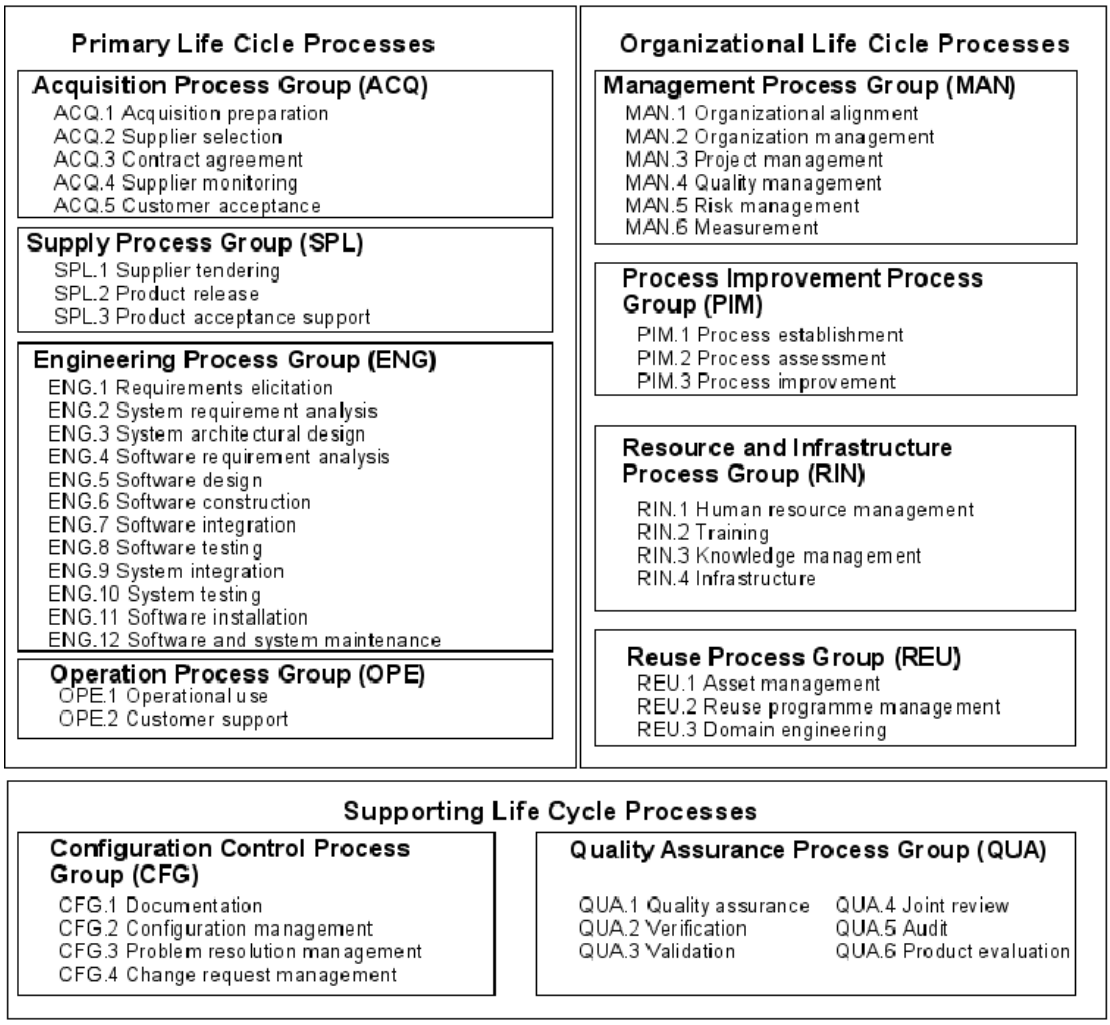

Figura A.1: Processi definiti dallo standard ISO-IEC-12207

A.2 Standard ISO/IEC$_G$ 15504 SPICE

A.2.1 Scopo

Lo standard ISO/IEC$_G$ 15504, chiamato anche SPICE (Software Process Improvement and Capability$_G$ Determination), fornisce un framework$_G$ per la valutazione dei processi di un'organizzazione. Questo framework$_G$ può essere utilizzato dalle organizzazioni coinvolte nella pianificazione, gestione, monitoraggio, controllo e miglioramento di acquisizione, consegna, sviluppo, implementazione, evoluzione e manutenzione di prodotti e servizi di supporto.

Dimensione del processo
La dimensione di processo del modello comprende 5 categorie di processi, rispettivamente composte da 4 a 10 processi:

- **Cliente-fornitore**: Raggruppa i processi messi in atto da un acquirente per identificare il suo bisogno, selezionare il suo fornitore e ricevere la fornitura. Dal punto di vista del fornitore, questa categoria comprende le attività necessarie per la fornitura, la messa in servizio, il funzionamento e il supporto dell'utente;
- **Engineering**: Rientrano in questa categoria le attività di sviluppo software, nell'ambito del proprio ambiente di sistema, dalla fase di definizione alla fase di manutenzione;
- **Supporto**: Raggruppa i processi che consentono l'implementazione nell'ambito di un altro processo come la documentazione o i processi di gestione della configurazione;
- **Gestione**: Questa categoria contiene i processi caratteristici delle attività di gestione, in particolare la gestione dei progetti e le attività di gestione della qualità e del rischio;
- **Organizzazione**: Questa categoria contiene i processi che riguardano l'intera organizzazione e non più il livello del singolo progetto.

Livelli di capacità e attributi di processo
Il livello di capacità della dimensione è stabilito dai seguenti 6 gradi:

- **Livello 0**: Processo incompleto o non eseguito, non raggiunge i suoi obiettivi;
- **Livello 1**: Processo svolto e implementato, gli obiettivi sono raggiunti ma non viene verificato;
- **Livello 2**: Processo gestito, la sua attuazione è pianificata, monitorata e adattata;
- **Livello 3**: Processo consolidato, si basa su pratiche documentate ed è in grado di raggiungere i propri obiettivi;
- **Livello 4**: Processo prevedibile e ripetibile, la sua attuazione è condizionata da obiettivi di performance definiti;
- **Livello 5**: Processo di ottimizzazione, per raggiungere gli obiettivi attuali e futuri, è costantemente migliorato.

La capacità di processo viene misurata utilizzando gli attributi di processo. Lo standard identifica nove attributi di processo:

- **1.1 - Prestazioni del processo**;
- **2.1 - Gestione delle prestazioni**;
- **2.2 - Gestione del prodotto di lavoro**;
- **3.1 - Definizione del processo**;
- **3.2 - Implementazione dei processi**;
- **4.1 - Misura di processo**;

- **4.2 - Controllo di processo**;
- **5.1 - Innovazione di processo**;
- **5.2 - Ottimizzazione del processo**.

In tutti i processi analizzati, vengono identificati una serie di forze e debolezze da cui si possono identificare potenziali di miglioramento. Le descrizioni del livello di maturità successivo mostrano le possibilità di miglioramento del processo. Questo standard richiede l'istituzione di una scala di valutazione. Gli attributi di ciascun processo sono valutati su una scala di valutazione suddivisa in quattro punti. I valori delle dimensioni dipendono dalla percentuale di raggiungimento degli attributi:

- **N**: Non implementato (0-15%);
- **P**: parzialmente implementato (> 15-50%);
- **L**: Ampiamente implementato (> 50-85%);
- **F**: completamente implementato (> 85%).

A.3 Standard ISO/IEC$_G$ 25000 SQuaRE

A.3.1 Scopo

L'ISO/IEC$_G$ 25000 vuole dare un contributo alla sicurezza, alla funzionalità e manutenibilità del prodotto software, all'accuratezza dei dati, al raggiungimento della soddisfazione dell'utente in un'ottica preventiva e di qualità misurabile. Esso propone quindi modelli di qualità di riferimento a priori rispetto a quelli dei sistemi basati solo sulla difettosità a posteriori o monitorata durante le fasi del ciclo di vita del prodotto.

A.3.2 ISO/IEC$_G$ 25010

Uno standard molto rilevante della serie 25000 SQuaRE e l'ISO/IEC$_G$ 25010 il quale si occupa di definire gli obiettivi di qualità che deve avere un prodotto software. Lo standard definisce quindi i termini di:

- **Qualità interna**: La quale è riferita alle proprietà statiche e strutturali del software;
- **Qualità esterna**: La quale è riferita alle proprietà dinamiche e comportamentali del software;
- **Qualità in uso**: La quale è riferita al comportamento del software in un ambiente di utilizzo reale e alle varie interazioni con gli utenti.

Qualità interna ed esterna
I modelli di qualità interna ed esterna comprendono 8 caratteristiche a loro volta sotto-categorizzate:

- **Idoneità funzionale**: Suddivisa in:
  - **Completezza**;
  - **Adeguatezza**;

  - **Correttezza**.
- **Prestazione ed efficienza**: Suddivisa in:
  - **Tempo**;
  - **Risorse**;
  - **Capacità**.
- **Usabilità**: Suddivisa in:
  - **Riconoscibilità**;
  - **Apprendibilità**;
  - **Operabilità**;
  - **Protezione errori**;
  - **Esteticità**;
  - **Accessibilità**.
- **Affidabilità**: Suddivisa in:
  - **Maturità**;
  - **Disponibilità**;
  - **Tolleranza**;
  - **Recuperabilità**.
- **Sicurezza**: Suddivisa in:
  - **Riservatezza**;
  - **Integrità**;
  - **Non ripudio**;
  - **Autenticazione**;
  - **Autenticità**.
- **Manutenibilità**: Suddivisa in:
  - **Modularità**;
  - **Riusabilità**;
  - **Analizzabilità**;
  - **Modificabilità**;
  - **Testabilità**.
- **Compatibilità**: Suddivisa in:
  - **Coesistenza**;
  - **Interoperabilità**.
- **Portabilità**: Suddivisa in:
  - **Adattabilità**;
  - **Installabilità**;
  - **Sostituibilità**.

Qualità in uso del prodotto
I modelli di qualità in uso del prodotto comprendono 5 caratteristiche a loro volta sotto-categorizzate:

- **Efficacia**;
- **Efficienza**;
- **Soddisfazione**; Suddivisa in:
  - **Utilità**;
  - **Fiducia**;
  - **Piacere**;
  - **Comodità**.
- **Assenza e attenuazione dei rischi**: Suddivisa in:
  - **Economicità**;
  - **Salute**;
  - **Ambiente**.
- **Copertura del contesto**: Suddivisa in:
  - **Completezza**;
  - **Flessibilità**.

A.3.3 ISO/IEC$_{G}$ 25023

Ad accompagnare lo standard 25010 troviamo lo standard 25023, il quale fornisce delle metriche di riferimento ai vari obiettivi di qualità di prodotto$_{G}$ presenti nello standard a cui riferisce.