Piano di Qualifica

Versione: 1.0.0 23/11/2024

Redattori	Malik Giafar Mohamed Stefano Baso
Verifica	Ion Cainareanu Maria Fuensanta Trigueros Hernandez Marco Perazzolo Malik Giafar Mohamed
Approvazione	Luca Parise
Uso	Esterno

nextsoftpadova@gmail.com

Registro dei cambiamenti

Versione	Data	Autore	Descrizione	Verifica
1.0.0	04/03/2025	Stefano Baso	Aggiunti grafici e fix nomeclatura	Malik Giafar Mohamed
0.5.0	26/02/2025	Malik Giafar Mohamed	Aggiunte formule per calcolo metriche e sezione valutazione lavoro	Ion Cainareanu
0.4.0	15/01/2025	Stefano Baso	Aggiunta test documenti	Ion Cainareanu, Marco Perazzolo
0.3.0	14/01/2025	Stefano Baso	Continuo aggiunta schema sezioni e tabelle	Marco Perazzolo
0.2.0	13/12/2024	Stefano Baso	Aggiunta schema sezioni	Ion Cainareanu
0.1.0	23/11/2024	Malik Giafar Mohamed	Creazione Documento	Ion Cainareanu, Maria Fuensanta Trigueros Hernandez

Indice

Elenco delle immagini

Elenco delle tabelle

1 Scopo del documento

Il *Piano di Qualifica*[G] è un documento soggetto a modifiche incrementali, finalizzate principalmente alla definizione delle *metriche*[G] di valutazione del prodotto. Tali metriche saranno stabilite in conformità ai requisiti e alle aspettative del proponente, con l'obiettivo di determinare correttamente la qualità del prodotto attraverso un processo di miglioramento continuo. Questo approccio tende ad evolversi nel tempo, in particolare una volta stabilita una linea guida.

Il presente documento si propone di:

- Definire le metriche e le metodologie di controllo e misurazione.
- Stabilire quantità, qualità dei *test*[G] e relative metriche.
- Descrivere l'applicazione dei test e documentarne i risultati, valutando la conformità rispetto alle attese e alle metriche definite.

Il documento sarà soggetto a modifiche e integrazioni durante il corso del progetto, in particolare durante le fasi di analisi e progettazione, e quindi non può essere considerato come definitivo.

1.1 Scopo del prodotto

Il prodotto, un plug-in per Visual Studio Code chiamato "Requirement Tracker", è progettato per automatizzare il tracciamento dei *requisiti*[G] nei progetti software complessi, con un focus particolare sull'ambito embedded. L'obiettivo principale è migliorare la qualità e la chiarezza dei requisiti, fornendo suggerimenti basati sull'analisi di un'intelligenza artificiale, riducendo al contempo i tempi e gli errori legati alla verifica manuale dell'implementazione nel codice sorgente. Il plug-in adotta un'architettura modulare che consente un'estensibilità semplice, rendendolo facilmente adattabile a nuove funzionalità o esigenze future. Inoltre, supporta gli sviluppatori avendo la capacità di utilizzare documenti tecnici come knowledge, ad esempio datasheet e manuali, permette di garantire una corretta copertura dei requisiti.

1.2 Glossario

I termini ambigui che necessitano di una spiegazione sono contrassegnati da una [G] come apice alla loro prima occorrenza nei documenti. Tutti i termini da glossario sono riportati in ordine alfabetico nell'omonimo documento.

1.3 Riferimenti

1.3.1 Riferimenti normativi

- Analisi dei Requisiti
- Norme di Progetto

1.3.2 Riferimenti informativi

Materiale didattico del corso

- Qualità di prodotto
  - https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T07.pdf
- Qualità di processo
  - https://www.math.unipd.it/~tullio/IS-1/2024/Dispense/T08.pdf
- Indice di Gulpease
  - https://www.ilc.cnr.it/dylanlab/apps/texttools/
- ISO/IEC 9126

2 Qualità di processo

2.1 Scopo ed obiettivi

La qualità di un sistema è determinata dai processi che lo costituiscono e viene misurata attraverso l'uso di metriche specifiche, atte a valutare tali processi e verificarne il raggiungimento degli obiettivi di qualità stabiliti. Il modello di riferimento è il *Ciclo di Deming*[G] o PDCA (Plan - Do - Check - Act), il quale consente di avere un miglioramento continuo tramite una gestione strutturata delle attività. Questo approccio si basa su una *pianificazione*[G] accurata, il monitoraggio mediante *metriche*[G] definite e l'integrazione dei risultati ottenuti nella fase di produzione operativa.

Di seguito, vengono presentati i processi identificati e i corrispondenti livelli di qualità prefissati. Per ciascuna metrica è fornita una descrizione che ne illustra le modalità di applicazione e definisce i valori considerati accettabili nel contesto delle verifiche di qualità.

2.2 Processi primari

2.2.1 Fornitura

In questa fase del processo vengono analizzate tutte le scelte effettuate durante lo sviluppo, verificandone la conformità con gli obiettivi stabiliti nelle diverse fasi del progetto. Vengono definite le misure da implementare, assicurando il rispetto dei termini e delle condizioni prestabiliti. L'obiettivo principale è garantire che la *fornitura*[G] sia allineata alle aspettative, sia in termini di risorse impiegate che di risultati ottenuti.

Un concetto chiave in questo contesto è l'MPC (Minimum Predictive Capability), una metrica fondamentale per valutare l'affidabilità di un modello di apprendimento automatico nel generare risultati accurati. L'MPC rappresenta il livello minimo di precisione che un modello deve raggiungere per essere considerato accettabile, contribuendo a garantire che le previsioni siano coerenti con gli standard richiesti.

Di seguito sono descritte le principali metriche e calcoli associati che verranno riportati nella tabella sottostante mettendo in relazione il valore plausibile e il valore ottimale:

- BAC (Budget At Completion): Costo totale preventivato per il completamento del progetto.

$$\mathrm{BAC} = \sum \text{costi previsti}$$

- EAC (Estimated At Completion): Valore stimato per i compiti rimanenti.

$$\mathrm{EAC} = \frac{\mathrm{BAC}}{\mathrm{CPI}}$$

- CPI (Cost Performance Index): Indice di prestazione dei costi, misura l'*efficienza*[G] con cui il *budget*[G] viene utilizzato. Un valore > 1 indica che il progetto sta spendendo meno del previsto, mentre un valore < 1 indica che sta spendendo di più del previsto.

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

- ETC (Estimated To Completion): Stima del costo finale aggiornato alla data di misurazione.

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

- EV (Earned Value): Valore ottenuto fino al momento attuale.

$$\mathrm{EV} = \left(\frac{\%\ \text{lavoro svolto}}{100}\right) * \mathrm{EAC}$$

- PV (Planned Value): Valore pianificato fino al momento attuale.

$$\mathrm{PV} = \left(\frac{\%\ \text{lavoro pianificato}}{100}\right) * \mathrm{BAC}$$

- AC (Actual Cost): Budget effettivamente speso fino al momento attuale.

$$\mathrm{AC} = \sum \text{costi effettivi}$$

- CV (Cost Variance): Differenza tra il valore ottenuto (EV) e il costo effettivo (AC).

$$\mathrm{CV} = \mathrm{EV} - \mathrm{AC}$$

- SV (Schedule Variance): Differenza tra il valore ottenuto (EV) e quello pianificato (PV). Un valore negativo indica un ritardo rispetto alla pianificazione.

$$\mathrm{SV} = \mathrm{EV} - \mathrm{PV}$$

- BV (Budget Variance): Differenza rispetto al budget preventivato.

$$\mathrm{BV} = \mathrm{AC} - \mathrm{CV}$$

Codice	Descrizione	Soglia accettabile	Ottimo
MPC01	Estimated at Completion	± 5% rispetto al *preventivo*[G]	Corrispondente al preventivo
MPC02	Estimated to Completion	>= 0	≤ EAC
MPC03	Earned Value	>= 0	≤ EAC
MPC04	Planned Value	>= 0	≤ BAC
MPC05	Actual Cost	>= 0	≤ EAC
MPC06	Cost Variance	≥ −5%	≥ 0%
MPC07	Schedule Variance	≥ −10%	≥ 0%
MPC08	Budget Variance	± 10%	≤ 0%

Table 1: Metriche di fornitura

Questi indicatori consentono di monitorare l'andamento del progetto in termini di costi, tempi e precisione delle previsioni, supportando una gestione efficiente e mirata.

2.2.2 Sviluppo

2.2.2.1 Progettazione di dettaglio

Indice per la media del numero di metodi presenti in ogni *package*[G], un indice alto potrebbe comportare il *refactoring*[G].

Codice	Descrizione	Soglia accettabile	Ottimo
MPC09	Number of Methods	3-11	3-8

Table 2: Metriche di progettazione di dettaglio

2.2.2.2 Codifica

- **BLC** (Bugs for Line of Code) = indice per il numero di righe di codice che possono contenere *bug*[G] o errori.
- **VNUD** (Variabili Non Utilizzate o non Definite) = indice per il numero di variabili utilizzate o non definite, queste sono a tutti gli effetti errori di programmazione che possono comportare bug. Variabili non utilizzate occupano spazio inutilmente in memoria e creano confusione all'interno del codice.

Codice	Descrizione	Soglia accettabile	Ottimo
MPC10	Bugs for Line of Code	0-70	0-25
MPC11	Variabili non utilizzate e non definite	0	0

Table 3: Metriche di codifica

2.3 Processi di supporto

2.3.1 Documentazione

La documentazione ha ruolo di supporto, in particolare definisce le norme da seguire durante lo sviluppo, la divisione delle risorse e responsabilità. E' necessario quindi definire una linea guida anche per la redazione dei documenti per evitare ambiguità e renderli chiari

Codice	Nome	Descrizione	Metriche associate
OPC01	Leggibilità dei documenti	Per mantenere una buona comprensione, il documento deve essere leggibile	MPC12
OPC02	Correttezza ortografica	Numero di errori grammaticali o ortografici per documento	MPC13

Table 4: Obiettivo di qualità della documentazione

2.3.1.1 Indice di leggibilità di Gulpease

L'indice di leggibilità di Gulpease è una metrica che valuta la semplicità di un testo in italiano, ideata per stimare quanto sia comprensibile da lettori con livelli diversi di istruzione. L'indice si basa su tre parametri: il numero di lettere, il numero di parole e il numero di frasi. La formula è:

$$\text{Gulpease} = 89 - \frac{\text{N° lettere}}{\text{N° parole}}10 + \frac{\text{N° frasi}}{\text{N° parole}}30$$

Il punteggio varia da 0 a 100, dove valori alti indicano maggiore leggibilità. Tipicamente, un testo comprensibile per chi ha una licenza elementare ha un punteggio sopra 80, mentre per chi possiede una licenza media è sufficiente un punteggio superiore a 60. Il metodo è particolarmente utile per valutare documenti destinati a un pubblico ampio, come testi scolastici o burocratici.

Codice	Descrizione	Soglia accettabile	Ottimo
MPC12	Indice di leggibilità di Gulpease	GULP ≥ 40	GULP ≥ 60

Table 5: Obiettivo di leggibilità

2.3.1.2 Indice errori ortografici

Per raggiungere l'ottimo anche nella documentazione bisogna raggiungere la massima correttezza in termini di grammatica e ortografia.

Codice	Descrizione	Soglia accettabile	Ottimo
MPC13	Numero errori ortografici	0	0

Table 6: Obiettivo di leggibilità

3 Qualità del prodotto

Per mantenere ed assicurare la qualità del prodotto software il gruppo ha adottato il modello di qualità stabilito dallo standard ISO/IEC 9126, adattandolo alle esigenze e requisiti del progetto. Tale standard propone una serie di metriche e regole per migliorare l'organizzazione dei processi e di conseguenza la qualità del prodotto. Di seguito verranno elencate e descritte le metriche che verranno utilizzate.

3.1 Efficienza

L'efficienza indica il tempo di elaborazione della richiesta da parte del software per raggiungere il risultato.

Codice	Descrizione	Soglia accettabile	Ottimo
MPDS01	Tempo di risposta medio	3 secondi	2 secondi

Table 7: Metriche di tempo medio

3.2 Usabilità

L'*usabilità*[G] riguarda l'esperienza dell'utente nell'interagire con il nostro prodotto, capirne il suo funzionamento e apprezzarne le sue funzioni.

Codice	Nome	Descrizione	Metriche associate
OPDS01	Usabilità del prodotto	Il prodotto deve essere facilmente usabile dall'utente in modo da raggiungere il più velocemente possibile quello che cerca	MPDS01

Table 8: Obiettivo di funzionalità

3.2.1 Facilità di utilizzo

Questa rappresenta la velocità con cui l'utente trova quello che sta cercando, calcolata in base al numero di click minimo che si deve effettuare per arrivare all'obiettivo.

Codice	Descrizione	Soglia accettabile	Ottimo
MPDS01	Facilità di utilizzo	FU ≤ 3	FU ≤ 5

Table 9: Obiettivo di usabilità

3.3 Manutenibilità

La manutenibilità del software è la facilità con cui può essere modificato, corretto, adattato o aggiornato.

Codice	Nome	Descrizione	Metriche associate
OPDS02	Analizzabilità del prodotto	Una facile analisi del codice permette di localizzare in tempi minimi il blocco di codice che riguarda l'errore o l'aggiornamento	MPDS03 MPDS04 MPDS06
OPDS03	Modificabilità del prodotto	Permette una manutenzione più agevolata per la correzione	MPDS05 MPDS02

Table 10: Obiettivo di manutenibilità

3.3.1 Metriche

- **Complessità ciclomatica**: misura la complessità strutturale di un programma basandosi sul grafo di controllo del flusso del codice. In particolare, rappresenta il numero di cammini *linearmente indipendenti*[G] attraverso il codice sorgente. Viene calcolata con la seguente formula:

$$V(G) = E - N + 2P$$

in cui:

- E è il numero di archi (transizioni tra i nodi),
- N è il numero di nodi (blocchi di codice o decision points),
- P è il numero di componenti connesse (tipicamente P = 1 per un singolo metodo o funzione)

- **Profondità della gerarchia**: indica il numero massimo di livelli di ereditarietà in una *gerarchia*[G] di classi. Una gerarchia più profonda può favorire il riuso del codice ma aumenta la complessità e il rischio di propagazione degli errori. Per un *design*[G] più manutenibile, è preferibile mantenere la profondità entro limiti ragionevoli (3-4 livelli), favorendo la composizione rispetto a una struttura gerarchica troppo profonda.
- **Parametri per metodo**: indica il numero di parametri per metodo. Un indice basso rappresenta un numero basso di parametri richiesti dal metodo, di conseguenza risulta di più facile comprensione e utilizzo.
- **Code Smell**: indice che rappresenta indicatori qualitativi di potenziali problemi nel codice. E' utile per valutare la leggibilità, la modificabilità, e la testabilità del codice. Si dividono in:
  - Duplicated Code: frammenti di codice identici o simili in più punti, che aumentano i costi di manutenzione perché le modifiche devono essere replicate ovunque.
  - Long Methods: metodi eccessivamente lunghi, che riducono la leggibilità e la comprensione del codice.
  - God Class: una classe con troppe responsabilità (violazione del *principio di Single Responsibility*[G]), difficile da testare e modificare.
  - High Coupling: una forte *dipendenza*[G] tra componenti, che rende il sistema rigido e suscettibile a errori quando una parte viene modificata.
  - Low Cohesion: componenti con funzionalità eterogenee che non si relazionano strettamente, rendendo il codice più complesso da comprendere.
- **Facilità di comprensione**: rappresenta il rapporto tra commenti presenti e codice totale per capirne il suo funzionamento.

Codice	Descrizione	Soglia accettabile	Ottimo
MPDS02	Profondità di gerarchia	PG <= 3	PG <= 2
MPDS03	Parametri per metodo	PPM <= 8	PPM <= 4
MPDS04	Complessità ciclomatica	CC <= 20	CC <= 10
MPDS05	Code smell	CS <= 50	CS <= 10
MPDS06	Facilità di comprensione	FC >= 0.10	FC >= 0.20

Table 11: Metriche di manutenibilità

3.4 Affidabilità

L'affidabilità riguarda il livello minimo di prestazioni da mantenere durante l'uso in determinate situazioni.

Codice	Nome	Descrizione	Metriche associate
OPDS04	Prodotto maturo	Evita errori o malfunzionamenti durante l'utilizzo	MPDS07 MPDS10
OPDS05	Tolleranza agli errori	Mantiene il livello di prestazioni anche durante un uso scorretto o in presenza di errori	MPDS11 MPDS08 MPDS09

Table 12: Obiettivo di affidabilità

3.4.1 Metriche

- **Code Coverage**: percentuale di codice eseguito nei test. Un indice di copertura del codice alto significa che è stato testato più codice, riducendo quindi la presenza di bug.
- **Branch Coverage**: percentuale di copertura di tutti i *branch*[G] all'esecuzione del codice. Il compito dei test è anche quello di verificare tutti i rami esistenti per verificarne la correttezza.
- **Presenza di vulnerabilità**: indice per il numero di vulnerabilità ancora presenti nel codice.
- **Presenza di bug**: indice per il numero di bug ancora presenti nel codice.
- **Successo dei test**: indice in percentuale relativo al successo dei test definiti dai *programmatori*[G].

Codice	Descrizione	Soglia accettabile	Ottimo
MPDS07	Code Coverage	CC >= 75%	100%
MPDS08	Presenza di vulnerabilità	VLN <= 2	0
MPDS09	Presenza di bug	BUG <= 20%	BUG <= 5%
MPDS10	Branch Coverage	BC >= 75%	100%
MPDS11	Successo dei test	>= 75%	100%

Table 13: Metriche di affidabilità

3.5 Funzionalità

La funzionalità è la capacità di fornire funzioni / azioni per ogni esigenza stabilita.

Codice	Nome	Descrizione	Metriche associate
OPDS06	Appropriatezza del prodotto	Fornire le funzioni richieste ed essere in linea con i requisiti fissati nell'Analisi dei Requisiti	MPDS12 MPDS13

Table 14: Obiettivo di funzionalità

3.5.1 Metriche

- **Requirement coverage**: indice della copertura dei requisiti descritti nell'*Analisi dei Requisiti*[G]. Viene calcolato con il rapporto percentuale tra numero di requisiti rispettati e numero di requisiti totali, con la formula:

$$\mathrm{RC} = \frac{R_{\mathrm{RISP}}}{R_{\mathrm{TOT}}} 100$$

- **Requisiti obbligatori soddisfatti**: indice della copertura dei *requisiti obbligatori*[G] descritti nell'Analisi dei Requisiti. Viene calcolato con il rapporto percentuale tra numero di requisiti rispettati e numero di requisiti totali, con la formula:

$$\mathrm{RC} = \frac{R_{\mathrm{ROS}}}{R_{\mathrm{ROT}}} 100$$

Codice	Descrizione	Soglia accettabile	Ottimo
MPDS12	*Requirement coverage*[G]	RC >= 75%	100%
MPDS13	Requisiti obbligatori soddisfatti	100%	100%

Table 15: Obiettivo di usabilità

4 Test e specifiche

Il seguente capitolo presenta in maniera dettagliata le strategie e scelte di testing, atte a garantire la correttezza del prodotto e facilitarne la *validazione*[G]. Viene adottato il Modello a V, in cui ad ogni fase di sviluppo corrisponde una fase di *verifica*[G] e validazione, garantendo quindi un controllo strutturato del processo.

Il modello è suddiviso in tre parti:

- **Fase di sviluppo** (lato sinistro): definisce e dettaglia i requisiti e la progettazione del sistema.
  - Requirements Gathering: Definizione dei requisiti.
  - System Analysis: Analisi funzionale e tecnica.
  - Software Design: Progettazione architetturale del sistema.
  - Module Design: Definizione dei singoli *moduli software*[G].
- **Coding**: avviene l'implementazione vera e propria del software.
- **Fase di testing e validazione** (lato destro): verifica che ogni fase di sviluppo soddisfi i requisiti stabiliti.
  - Unit Testing: Test sui singoli moduli.
  - Integration Testing: Verifica dell'integrazione tra i moduli.
  - System Testing: Validazione dell'intero sistema.
  - Acceptance Testing: Verifica finale rispetto ai requisiti del cliente.

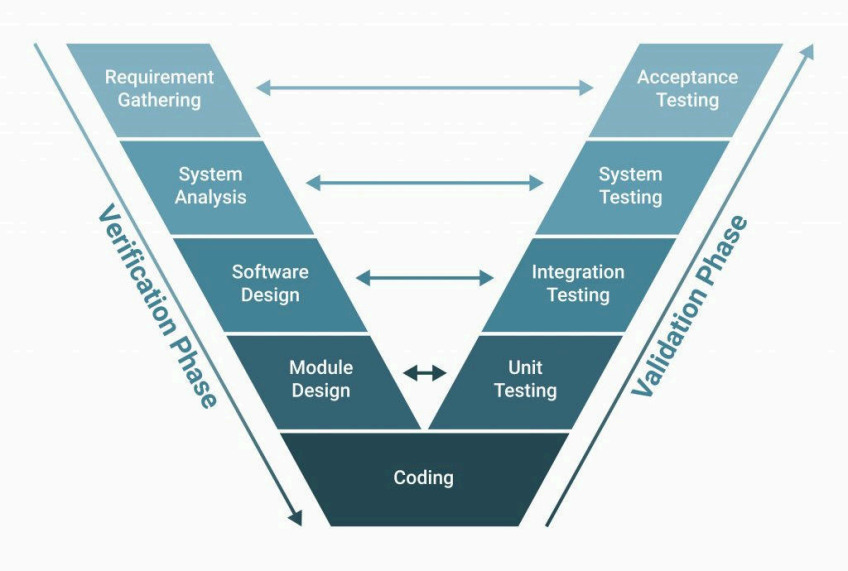

Figure 1: Modello a V

Questo modello prevede una stretta corrispondenza tra sviluppo e testing, assicurando che ogni fase sia verificata e validata, riducendo il rischio di errori.

4.1 Tipologie di test

4.1.1 Test di Unità

I *test di unità*[G] valutano il corretto funzionamento delle singole unità di codice all'interno del software. Un'unità di codice è una funzione, una classe o qualsiasi componente che svolge un'attività specifica in modo indipendente rispetto al resto del sistema. Attualmente, nella prima versione del Piano di Qualifica, né le unità né i relativi test corrispondenti sono stati definiti. La definizione delle unità avverrà con l'avvio del processo di progettazione e sviluppo software.

4.1.2 Test di Integrazione

I test di integrazione valutano il corretto funzionamento delle diverse componenti del software e il modo in cui vengono integrate tra loro, evidenziandone eventuali problemi. In questa stesura iniziale del Piano di Qualifica non sono state ancora individuate le componenti del prodotto, di conseguenza i test di integrazione non sono ancora stati definiti.

4.1.3 Test di Sistema

I test di sistema verificano il sistema completo del prodotto software, prendendo in considerazione tutti i componenti e interfacce con altri sistemi. Questi test controllano che il software rispetti tutti i requisiti prestabiliti e che sia adatto all'uso in produzione.

4.1.4 Test di Accettazione

I test di accettazione assicurano che il software soddisfi i requisiti e parametri stabiliti dal *capitolato*[G]. Sono svolti in presenza del *committente*[G] e verificano che il prodotto possa essere consegnato al committente o messo in produzione.

4.1.5 Test di Regressione

I test di regressione servono a testare che gli aggiornamenti / modifiche rilasciati nel software non incidano negativamente sulle funzioni già presenti. Ciò consiste nella ripetizione di test di unità, integrazione e sistema.

4.1.6 Sviluppo

Le specifiche riguardanti i test descritti verranno definite nelle successive versioni del Piano di Qualifica

5 Resoconto delle attività di verifica

5.1 MPC05 - MPC02: Actual Cost e Estimated to Completion

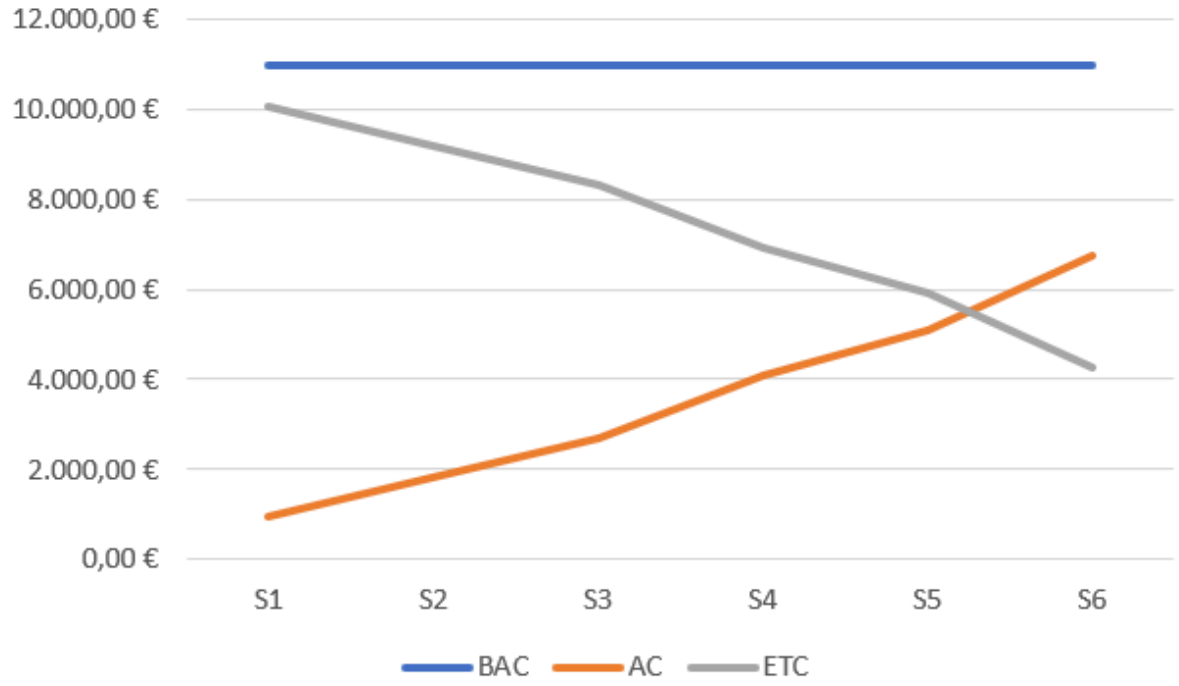

Figure 2: Grafo Actual Cost e Estimated to Completion

5.2 MPC03 - MPC04: Earned Value e Planned Value

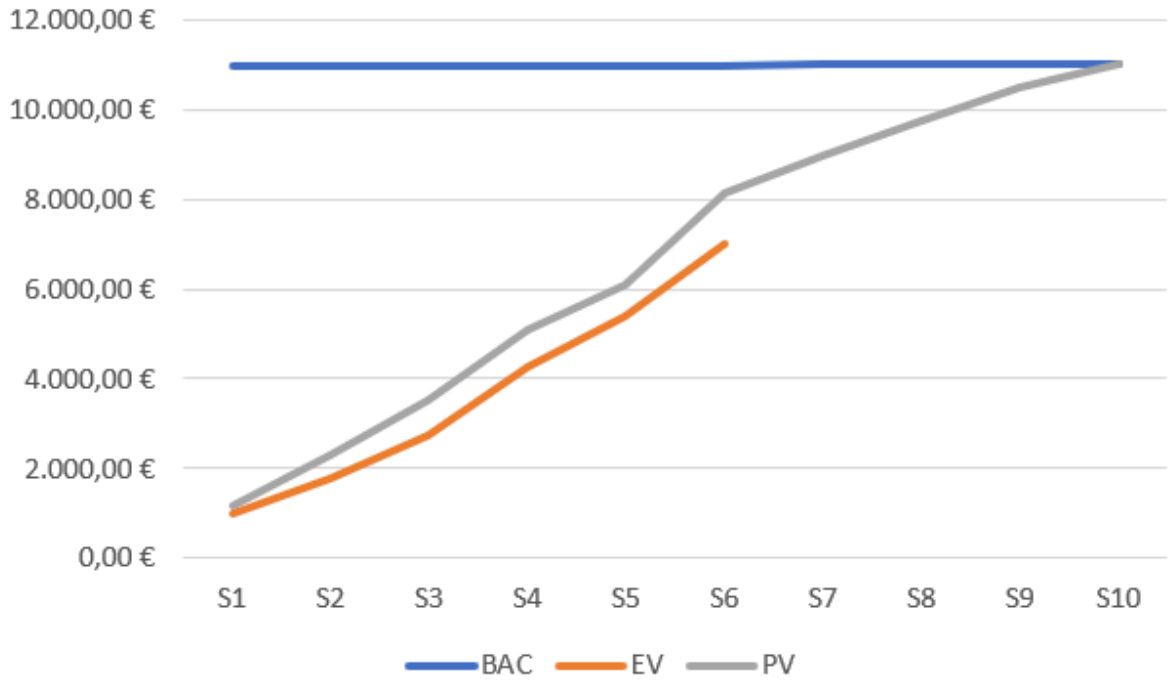

Figure 3: Grafo Earned Value e Planned Value

5.3 MPC07: Schedule Variance

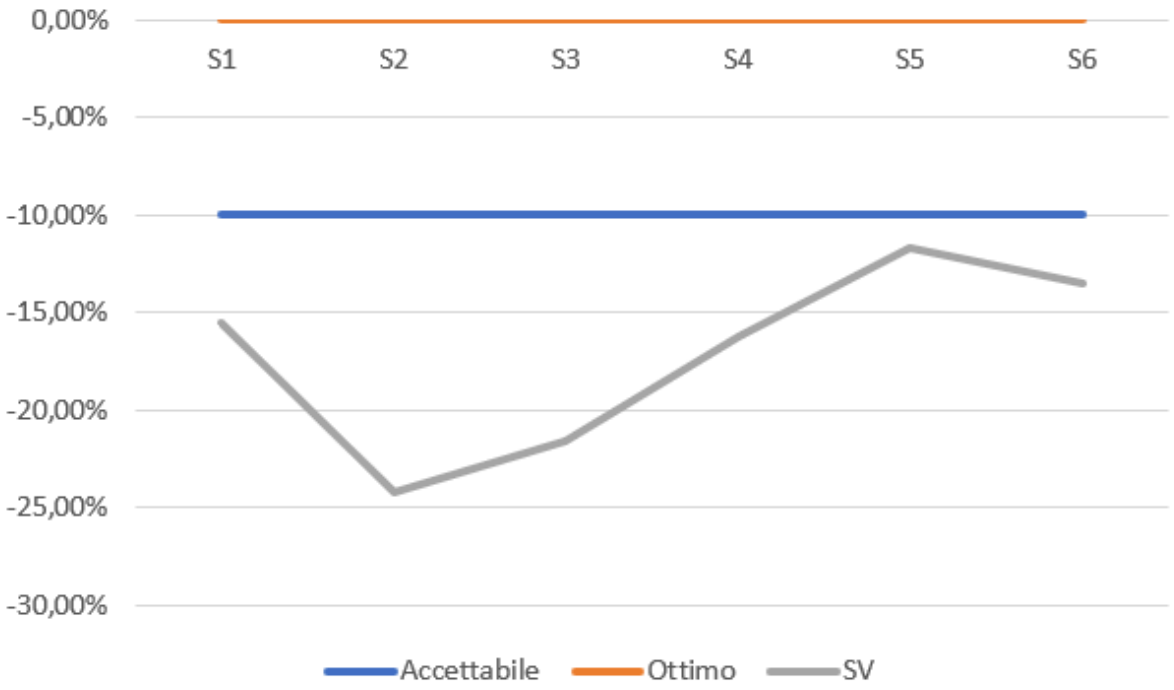

Figure 4: Grafo Schedule Variance

5.4 MPC06: Cost Variance

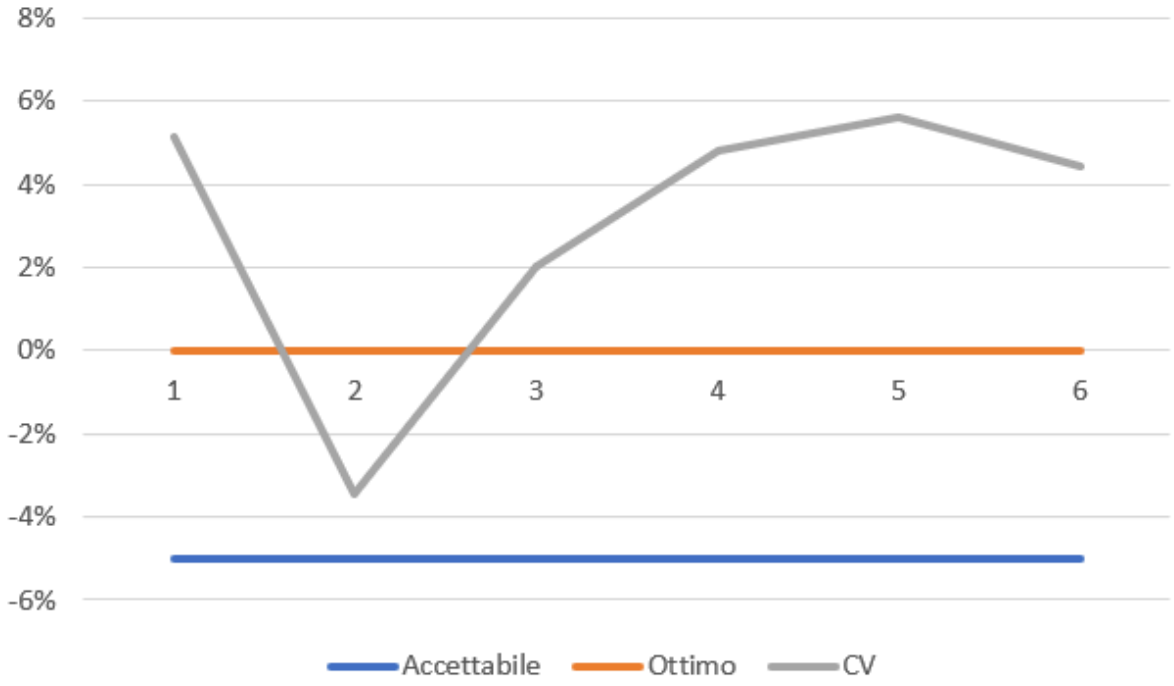

Figure 5: Grafo Cost Variance

5.5 MPC01: Estimated at Completion

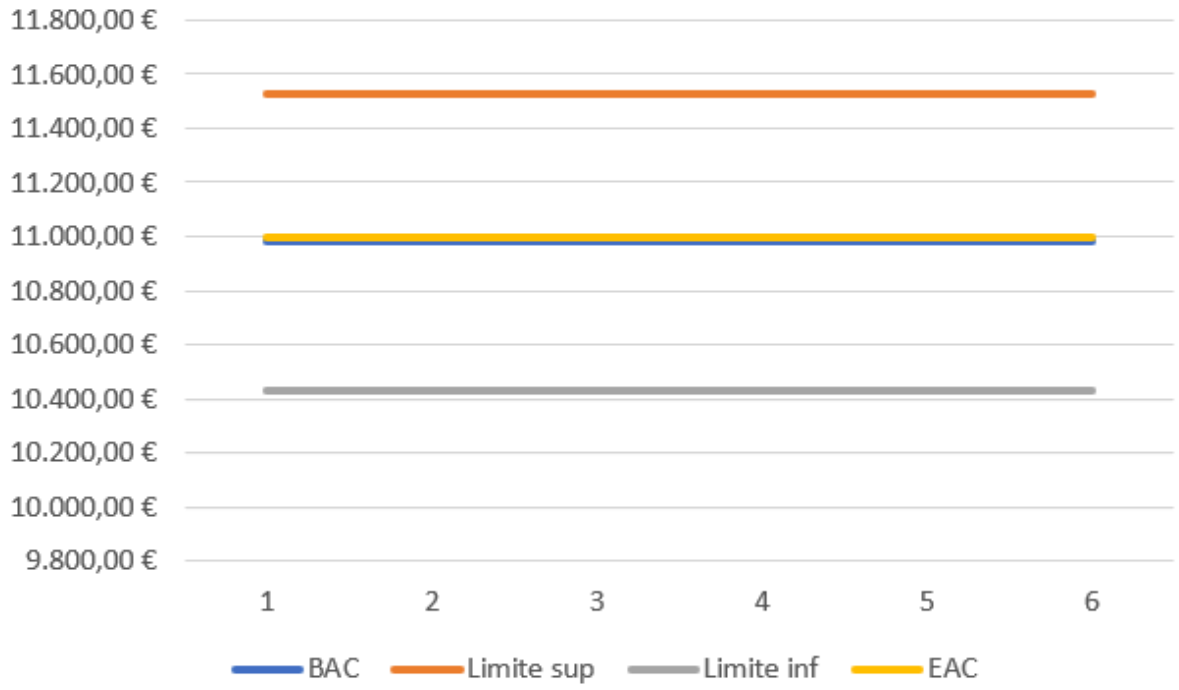

Figure 6: Grafo Estimated at Completion

5.6 MPC12: Indice di Gulpease

Di seguito la tabella con i risultati ottenuti dai documenti secondo l'indice di Gulpease. Come metro di valutazione del documento viene esclusa la prima pagina che, trattandosi dell'intestazione, potrebbe portare ad un risultato inesatto.

Documento	Risultato	Esito
Analisi dei Requisiti	83	Superato
Piano di qualifica	84	Superato
Piano di Progetto	80	Superato
Norme di Progetto	75	Superato
Glossario	83	Superato
2024-11-15	68	Superato
2024-11-24	66	Superato
2024-12-09	75	Superato
2024-12-18	66	Superato
2025-01-03	73	Superato
2025-01-10	65	Superato
2025-01-19	63	Superato
2025-02-08	65	Superato
2025-02-20	67	Superato
2025-03-07	65	Superato
2024-11-25	67	Superato
2024-12-24	64	Superato
2025-02-25	65	Superato

Table 16: Valutazione documenti fase 1

6 Valutazioni per il miglioramento

Nel seguente capitolo vengono riportate delle osservazioni sulle criticità incontrate con lo scopo di individuare i problemi e adottare dei miglioramenti.

6.1 Valutazione sull'organizzazione

Problema	Descrizione	Gravità	Soluzione
Riunione di gruppo	Incontri settimanali di durata molto lunga con ripetizione di argomenti	Bassa	Preparazione di una presentazione con punti da discutere e su cui focalizzarsi

Table 17: Problemi organizzativi

6.2 Valutazione sui ruoli

Problema	Descrizione	Gravità	Soluzione
Rotazione dei ruoli	Durante le prime fasi del lavoro la rotazione dei ruoli non era definita e a tratti assente	Media	Nuova ripartizione del carico di lavoro e definizione dei ruoli alle riunioni settimanali

Table 18: Problemi rotazione ruoli

6.3 Valutazione degli strumenti di lavoro

Problema	Descrizione	Gravità	Soluzione
Poca conoscenza delle tecnologie richieste	Durante lo sviluppo sono state richieste l'uso di tecnologie non conosciute dal gruppo	Alta	Investito tempo nello studio e formazione dei membri con prove e test pratici, uso di Notion per condividere le ricerche fatte

Table 19: Problemi con strumenti di lavoro