Norme di Progetto
Progetto Piattaforma di Localizzazione Testi

submarines.g4@gmail.com

Informazioni sul documento

Responsabile	Davide Marzaro
Redattori	Michael Amista'
	Samuel Scarabottolo
Verificatori	Niccolò Fasolo
	Matteo Cescon
	Davide Marzaro
	Michael Amista'
Uso	Interno
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo *Submarines*
Versione	2.0.0

Sommario

Questo documento contiene le regole interne adottate dal gruppo *Submarines* riguardo le attività del progetto Piattaforma di Localizzazione Testi.

Registro delle Modifiche

Versione	Data	Autore	Ruolo	Descrizione
2.0.0	2023/05/15	Davide Marzaro	*Responsabile*	Approvazione e rilascio del documento
1.2.0	2023/05/12	Michael Amista'	*Verificatore*	Verifica delle sotto-sezioni *Test* e *Nominazione dei test*
1.1.1	2023/05/11	Samuel Scarabottolo	*Amministratore*	Stesura sotto-sezioni *Test* e *Nominazione dei test*
1.1.0	2023/05/01	Davide Marzaro	*Verificatore*	Verifica delle sezioni *Codifica* (con annesse sotto-sezioni), *Verifica del codice* e *Validazione del codice*
1.0.1	2023/04/30	Michael Amista'	*Amministratore*	Aggiunte sezioni *Codifica* (con annesse sotto-sezioni), *Verifica del codice* e *Validazione del codice*
1.0.0	2023/01/28	Davide Marzaro	*Responsabile*	Approvazione e rilascio del documento
0.3.0	2023/01/25	Matteo Cescon	*Verificatore*	Verifica del documento
0.2.4	2023/01/18	Michael Amista'	*Amministratore*	Aggiunte sezioni *Diario di bordo*, *Gestione della Configurazione*, *Gestione della Qualità* e aggiornata sezione sulle *Modalità* della Formazione del Personale
0.2.3	2023/01/15	Samuel Scarabottolo	*Amministratore*	Inserite sezioni *Metriche per la qualità di prodotto* e *Metriche per la qualità di processo*
0.2.2	2023/01/15	Michael Amista'	*Amministratore*	Aggiunte sezioni *Standard ISO/IEC 15504 - SPICE* e *Standard ISO/IEC 9126*
0.2.1	2023/01/10	Michael Amista'	*Amministratore*	Aggiunte sezioni *Altri strumenti a supporto della gestione organizzativa* e *Standard ISO/IEC 12207:1995*
0.2.0	2023/01/09	Matteo Cescon	*Verificatore*	Verifica del documento
0.1.5	2023/01/08	Michael Amista'	*Amministratore*	Rinnovata sezione *Casi d'uso* (Flusso di eventi) e *Classificazione dei casi d'uso* (Descrizione eliminata)
0.1.4	2023/01/06	Michael Amista'	*Amministratore*	Sviluppo della sezione *Formazione del Personale*
0.1.3	2023/01/04	Michael Amista'	*Amministratore*	Sviluppo della sezione *Processi Organizzativi* e aggiornamento generale alla struttura del documento
0.1.2	2022/12/29	Michael Amista'	*Amministratore*	Aggiornate sezioni *Riferimenti* e *Registro delle modifiche*

Versione	Data	Autore	Ruolo	Descrizione
0.1.1	2022/12/27	Michael Amista'	*Amministratore*	Sviluppo sezione *Analisi dei Requisiti*, aggiunte sezioni *Informazioni sui documenti* e *Registro delle modifiche*, sviluppata sezione *Gestione Organizzativa*
0.1.0	2022/12/24	Niccolò Fasolo	*Verificatore*	Verifica del documento
0.0.5	2022/12/20	Michael Amista'	*Amministratore*	Sviluppo della sezione *Processi Primari*
0.0.4	2022/12/15	Michael Amista'	*Amministratore*	Rinnovata l'intera struttura del documento e aggiunta sezione *Riferimenti*
0.0.3	2022/12/12	Michael Amista'	*Amministratore*	Aggiornate le sezioni *date dei meeting*, *verbali*, *verbali interni*, *verbali esterni* e *organizzazione GitHub*
0.0.2	2022/12/09	Michael Amista'	*Amministratore*	Definizione del versionamento, revisione e approvazione della documentazione
0.0.1	2022/12/06	Michael Amista'	*Amministratore*	Creazione del documento e primo aggiornamento ai contenuti

Contenuti

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è di definire le norme, le convenzioni e le procedure adottate da tutti i membri di *Submarines*, in modo da poter definire un metodo di lavoro comune. Tutti i membri sono tenuti a visionare periodicamente tale documento e a rispettare tutte le norme in esso presenti. Per la stesura viene adottata una filosofia incrementale, quindi il documento allo stato attuale è incompleto, con l'aspettativa di avere un processo normato prima del suo avvio, considerando che, in generale, ogni norma può essere soggetta a cambiamenti.

1.2 Scopo del prodotto

L'obiettivo dell'azienda Zero12 è la creazione di un sistema software costituito da una webapp. Lo scopo del prodotto è quello di fornire un sistema di localizzazione dei testi per app e webapp. Molti applicativi moderni devono essere multi-lingua in modo da essere disponibili in un mercato internazionale, diventa quindi necessario capire come gestire le traduzioni ed è esattamente questo l'obiettivo che l'applicativo si pone di raggiungere.

1.3 Glossario

Per evitare di generare dubbi o ambiguità, è stato creato un documento denominato *Glossario v1.0.0*, che contiene tutti i termini per i quali il gruppo ha ritenuto adeguato fornire una spiegazione. All'interno di questa documentazione, tali termini sono contrassegnati da una ^{G} all'apice della parola.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- Regolamento del progetto didattico - Corso di Ingegneria del Software, UniPD a.a 2022-2023
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/PD02.pdf
- Presentazione del capitolato^{G} 4 - Zero12 Progettazione e sviluppo di una Piattaforma di Localizzazione Testi
  https://www.math.unipd.it/ tullio/IS-1/2022/Progetto/C4.pdf

1.4.2 Riferimenti informativi

- Slide T02 - Corso di Ingegneria del Software, UniPD a.a. 2022-2023 - Processi di ciclo di vita
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T02.pdf
- Slide T03 - Corso di Ingegneria del Software, UniPD a.a. 2022-2023 - Il ciclo di vita del SW
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T03.pdf
- Slide T04 - Corso di Ingegneria del Software, UniPD a.a 2022-2023 - Gestione di progetto
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T04.pdf
- Slide T10 - Corso di Ingegneria del Software, UniPD a.a 2022-2023 - Verifica e validazione: introduzione
  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T10.pdf

- Slide T11 - Corso di Ingegneria del Software, UniPD a.a 2022-2023 - Verifica e validazione: analisi statica

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T11.pdf

- Slide T12 - Corso di Ingegneria del Software, UniPD a.a 2022-2023 - Verifica e validazione: analisi dinamica

  https://www.math.unipd.it/ tullio/IS-1/2022/Dispense/T12.pdf

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

In questa sezione vengono illustrati i documenti e i metodi adoperati per realizzare il processo[G] di fornitura, definito secondo lo standard ISO/IEC/IEEE 12207:1995.

2.1.2 Descrizione

Il processo di fornitura determina ogni compito e attività necessaria allo svolgimento del progetto[G]. Tale processo permette di risolvere e sanare ogni dubbio legato a ciò che il proponente[G] si aspetta di vedere, verrà avviato solamente in seguito alla comprensione delle richieste del proponente. Seguiranno uno studio di fattibilità e la sigla del contratto con l'azienda interessata, la quale impegnerà il gruppo nella progettazione e realizzazione del prodotto software finale.

2.1.3 Rapporti con il proponente

2.1.3.1 Azienda

L'azienda Zero12 accompagna le aziende clienti in un percorso di innovazione che si pone i seguenti obiettivi: digitalizzare i processi, creare soluzioni software di engagement, semplici e indispensabili, per i clienti (interni o esterni). Tutto quello che l'azienda svolge è *Cloud Native*[G] e basato su tecnologia *Amazon Web Services*[G] (AWS).

2.1.3.2 Materiale fornito

Zero12 ha fornito, oltre al capitolato, una lista delle tecnologie da utilizzare e consigliate per lo sviluppo del prodotto finale e una serie di corsi di formazione sulle tecnologie AWS, le cui date saranno definite dall'azienda stessa. In particolare, i corsi di formazione si pongono l'obiettivo di fornire una base solida su: AWS e infrastruttura, front-end[G], back-end[G].

2.1.3.3 Supporto

L'azienda ha fornito una lista di tecnologie consigliate per lo sviluppo del progetto e nel caso si rivelassero quelle scelte dal gruppo, Zero12 si mette a disposizione per un supporto in caso di difficoltà. Essendo tali tecnologie quelle utilizzate dall'azienda il supporto si rivela molto più semplice rispetto a tecnologie da loro non utilizzate.

2.1.3.4 Comunicazione

Per la comunicazione con il proponente, Zero12 ha aperto un canale Slack[G] al quale sono stati invitati tutti i membri di *Submarines*. Tale canale viene utilizzato per le comunicazioni dirette con il Project Manager di Zero12, Michele Massaro, che si mette a disposizione per eventuali dubbi emersi o questioni che richiedono un suo riscontro.

2.1.4 Documenti

In questa sezione sono riportati i documenti prodotti in questa fase e i relativi strumenti a supporto della stesura.

2.1.4.1 Piano di Progetto

Il piano di progetto è un documento che contiene la pianificazione dei tempi, l'analisi dei rischi, il consuntivo di periodo, la data di consegna e i costi previsti.

2.1.4.2 Piano di Qualifica

Il piano di qualifica deve contenere tutte le modalità adottate dal *Verificatore* durante la verifica e validazione assicurando che la qualità dei processi e delle risorse rispetti le aspettative.

2.1.4.3 Strumenti

Per gli strumenti adottati per realizzare i processi di questa fase si fa riferimento alle sezioni 3.1.5 (Strumenti per la stesura e template) e 4.2.3 (Strumenti collaborativi) di questo documento.

2.2 Sviluppo

2.2.1 Scopo

In questa sezione vengono presentati i compiti e le attività da svolgere relative al processo di sviluppo del prodotto software, definito secondo lo standard ISO/IEC/IEEE 12207:1995. Di seguito vengono quindi elencate le norme e le convenzioni adottate per la composizione di questo processo.

2.2.2 Descrizione

Di seguito sono elencate le attività che compongo il processo di sviluppo:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.3 Analisi dei Requisiti

2.2.3.1 Scopo

Il documento *Analisi dei requisiti* definisce nel dettaglio l'analisi dei requisiti e i possibili scenari e casi d'uso che tratta. L'obiettivo è quello di analizzare accuratamente quanto presente nel capitolato al fine di individuare tutti i requisiti e i casi d'uso che il proponente richiede per la realizzazione del prodotto.

2.2.3.2 Requisiti

I requisiti vengono raccolti attraverso le seguenti fasi:

- Lettura attenta del capitolato;
- Confronto interno con il gruppo;
- Confronto con il proponente.

Tali fasi permettono di analizzare nel modo più corretto possibile i requisiti ottenendo infine una conferma fondamentale e necessaria da parte del proponente.

2.2.3.3 Classificazione dei requisiti

Rappresentano dei requisiti che deve soddisfare il prodotto che si vuole realizzare. I requisiti saranno organizzati in forma tabellare. La tabella avrà le seguenti quattro colonne:

- Codice identificativo
- Classificazione
- Descrizione
- Fonti

Ogni codice identificativo di un requisito sarà strutturato come segue:

- Codice identificativo: **R[Importanza][Tipologia][Codice]**
  - Importanza:
    - 1: Requisito obbligatorio, ovvero irrinunciabile per almeno uno degli stakeholder;
    - 2: Requisito desiderabile, ovvero non strettamente necessario ma che porta valore aggiunto riconoscibile;
    - 3: Requisito opzionale, ovvero relativamente utile oppure contrattabile più avanti nel progetto.
  - Tipologia:
    - F: Funzionale, definisce una funzione di un sistema di uno o più dei suoi componenti;
    - Q: Qualitativo, definisce un requisito per garantire la qualità per un certo aspetto del progetto;
    - P: Prestazionale, definisce un requisito che garantisce efficienza prestazionale nel prodotto;
    - V: Vincolo, definisce un requisito volto a far rispettare un dato vincolo.
  - Codice: Composto da:
    - C* se il requisito viene dal capitolato, * sarà un numero progressivo univoco;
    - I* se il requisito proviene da una decisione interna al gruppo, * sarà un numero;
    - Z* se il requisito proviene da un incrontro con l'azienda.
  - Descrizione: descrizione sintetica del requisito.
  - Classificazione: identifica l'importanza del requisito.

2.2.3.4 Casi d'uso

Il caso d'uso consiste nel valutare ogni requisito focalizzandosi sugli attori che interagiscono col sistema, valutandone le varie interazioni. Vengono rappresentati graficamente tramite schemi UML^{G}.

Ciascun caso d'uso è costituito da:

- *Attore primario*;
- *Precondizione*;
- *Postcondizione*;
- *Scenario principale*;
- *Flusso di eventi* (dove ritenuto necessario);
- *Estensioni* (dove ritenute necessarie).

2.2.3.5 Classificazione dei casi d'uso

La struttura adottata per la classificazione dei casi d'uso è la seguente:

UC[Identificatore][CodiceCasoBase](.[CodiceSottoCaso])*

Composta da:

- **UC**: acronimo di "Use Case";
- **Identificatore**: che può assumere le diverse espressioni letterali:
  - **W**: identifica un caso d'uso relativo alla WebApp[G];
  - **L**: identifica un caso d'uso relativo alla libreria[G];
  - **E**: identifica un caso d'uso d'errore.
- **CodiceCasoBase**: ID del caso d'uso generico;
- **CodiceSottoCaso**: ID opzionale per i sottocasi di un caso d'uso.

Ogni caso d'uso è descritto da:

- **Id**: codice identificativo del caso d'uso, stabilito come enunciato sopra;
- **Nome**: stringa titolo del caso d'uso posta dopo l'id;
- **Diagramma UML**: diagramma per rappresentare graficamente il caso d'uso (dove ritenuto necessario);
- **Attori**: entità esterne al sistema che interagiscono con esso. Ne esistono due varianti:
  - **Primario**: interagisce con il sistema per raggiungere un obiettivo;
  - **Secondario**: aiuta il primario a raggiungere l'obiettivo.
- **Precondizione**: descrive lo stato del sistema prima del verificarsi del caso d'uso;
- **Postcondizione**: descrive lo stato del sistema dopo che si è verificato il caso d'uso;
- **Scenario principale**: descrive il flusso degli eventi del caso d'uso;
- **Scenario secondario/alternativo**: descrive il flusso degli eventi del caso d'uso dopo un evento imprevisto che lo ha deviato dal caso principale. Può non esserci o possono essercene più di uno;
- **Estensioni**: utilizzate negli scenari alternativi. Se si verifica una determinata situazione, il caso d'uso collegato all'estensione viene interrotto.

2.2.4 Progettazione

Lo scopo della progettazione è quello di determinare le caratteristiche che il prodotto finale deve avere, incorporando le parti ottenute durante l'analisi dei requisiti.

2.2.4.1 Scopo

Lo scopo di questo processo è quello di pensare a come le parti del sistema possano essere realizzate, e come esse debbano interagire tra loro. É bene notare che in questo processo non si scrive codice ma si idealizza l'architettura di sistema.

2.2.4.2 Descrizione

Il fine della progettazione è la realizzazione dell'architettura di sistema inizialmente realizzata da: un *Proof of Concept*[G] sviluppato come una versione prototipale del sistema; la *Requirements and Technology Baseline*; e tramite il documento tecnico *Product Baseline*.

2.2.4.3 Requirements and Technology Baseline

In accordo con il proponente vengono motivate le tecnologie, i framework[G] e le librerie selezionate per la realizzazione del prodotto. Dimostra la fattibilità degli obiettivi prestabiliti. Il *Proof of Concept* raggruppa tutte le tecnologie adottate per la realizzazione del prodotto finale e ne dimostra la compatibilità e adeguatezza alle richieste del proponente.

2.2.4.4 Product Baseline

Rapppresenta la baseline del prodotto, coerente con la Requirements and Technology Baseline e definisce quindi il design effettivo del prodotto.

2.2.5 Codifica

2.2.5.1 Scopo

Questo processo dovrà avere come output un prodotto software avente le caratteristiche e i requisiti concordati con il proponente. Il codice generato dovrà rispettare alcune norme per poter essere leggibile e per poter facilmente intervenire in seguito nelle attività di manutenzione, modifica, verifica e validazione.

2.2.5.2 Descrizione

L'obiettivo del processo di codifica è la realizzazione del prodotto software, svolto dal programmatore, secondo l'architettura di sistema definita nella fase di progettazione.

2.2.5.3 Stile di codifica

- **Indentazione**: i blocchi di codice innestati dovranno avere un'indentazione opportuna al loro livello di innesto;
- **Parentesi**: la parentesi aperta dovrà essere inserita nella stessa riga di dichiarazione del costrutto mentre la parentesi chiusa dovrà essere inserita con la giusta indentazione alla riga immediatamente successiva all'ultima riga di codice del costrutto;
- **Metodi**: il nome dei metodi dovrà iniziare con lettera minuscola e, se composto da più parole, le successive dovranno iniziare con lettera maiuscola. Risulta preferibile mantenere metodi brevi, con poche righe di codice;
- **Variabili**: il nome delle variabili deve sempre essere scritto in minuscolo. Se il nome è composto da più parole, la seconda dovrà iniziare con la lettera maiuscola;
- **Costanti**: il nome delle costanti deve sempre iniziare con una lettera maiuscola. Se il nome è composto da più parole, la seconda dovrà iniziare con la lettera maiuscola;
- **Univocità dei nomi**: tutti i costrutti dovranno avere nomi univoci e significativi;
- **Commenti**: i commenti dovranno essere inseriti prima dell'inizio del costrutto e presentati in lingua italiana. Risulta preferibile commentare solo dove sono necessarie ulteriore informazioni non ricavabili dal blocco di codice in questione;

- **File**: dovranno avere un nome che inizia per lettera maiuscola che ne specifichi il contenuto.

2.2.5.4 Strumenti

Gli strumenti che verranno adottati durante il processo di sviluppo sono:

- **Visual Studio Code**: IDE utilizzato dal gruppo per la stesura del codice;
- **Postman**: piattaforma API che consente agli sviluppatori di progettare, costruire, testare e iterare le proprie API;
- **AWS Dashboard**: permette di gestire con facilità tutti i servizi AWS necessari per il corretto funzionamento dell'applicativo da realizzare.

3 Processi di supporto

3.1 Documentazione

3.1.1 Informazioni sul documento

La prima pagina di ogni documento, fatta eccezione per i verbali, riporta le seguenti informazioni:

- *Responsabile*: indica il nome del responsabile che ha approvato quel documento;
- *Redattori*: indica chi si è occupato della redazione del documento;
- *Verificatori*: indica chi sono stati i verificatori in quel preciso documento;
- *Uso*: indica se il documento è destinato a uso interno o esterno;
- *Destinatari*: indica a chi è destinato il documento.

3.1.2 Registro delle modifiche

Il registro delle modifiche è presente in ogni singolo documento, eccezione fatta per i verbali. Il registro delle modifiche tiene traccia delle attività effettuate al documento (es. modifica, verifica, approvazione) e, in particolare, è caratterizzato da:

- *Versione* attuale al momento della modifica;
- *Data* della modifica;
- *Autore* della modifica;
- *Ruolo* dell'autore del cambiamento effettuato;
- *Descrizione* su ciò che è stato aggiunto o modificato.

Tale registro è sempre aggiornato all'ultima attività effettuata sul documento.

3.1.3 Verbali

Si è deciso di creare un template per i verbali interni ed esterni in modo da facilitare notevolmente la loro compilazione. Tale template viene messo a disposizione nello spazio di archiviazione Google Drive[G] collegato alla mail del gruppo, in modo che possa essere reperito da chi incaricato per la stesura di un nuovo verbale. I verbali hanno il compito di riassumere quanto discusso nel relativo meeting in modo che possa essere recuperato anche da chi non era presente a quell'incontro. La persona incaricata della stesura del verbale dovrà riportare i punti salienti del meeting accompagnati da una breve descrizione. I verbali, interni ed esterni, sono gli unici documenti che non verranno sottoposti alle regole di versionamento, revisione ed approvazione. I verbali verranno stilati durante gli incontri interni ed esterni e saranno soggetti di una verifica e di una approvazione istantanea subito dopo tali incontri.

3.1.3.1 Verbali interni

I verbali interni sono dedicati alle discussioni dei *Submarines*, effettuate al termine di ogni settimana. Prima di ogni meeting verrà scelta la persona responsabile della redazione del verbale.

3.1.3.2 Verbali esterni

I verbali esterni sono dedicati agli incontri con il proponente. Tali incontri potranno essere richiesti dai *Submarines*, qualora fosse ritenuto necessario esporre a Zero12 dei quesiti riguardo il progetto, oppure potranno essere richiesti dal proponente stesso.

3.1.4 Diario di bordo

Il documento consiste in una presentazione settimanale nella quale il gruppo *Submarines* illustra il lavoro condotto durante l'ultima settimana. La presentazione segue un template comune che si suddivide in quattro punti:

- *Cose fatte nel periodo trascorso*: viene illustrato l'operato settimanale del gruppo;
- *Cose da fare nel prossimo periodo/in corso*: viene illustrata la pianificazione delle attività future del gruppo;
- *Difficoltà incontrate*: vengono illustrate, se presenti, eventuali criticità che hanno compromesso il lavoro del gruppo;
- *Dubbi su come procedere su specifiche questioni*: vengono poste, se presenti, alcuni dubbi del gruppo su come procedere su specifiche questioni.

Tale documento dovrà essere inoltrato entro ogni lunedì mattina al Prof. Tullio Vardanega secondo le modalità da lui indicate.

3.1.5 Strumenti per la stesura e template

Per quanto riguarda la stesura della documentazione si sono adottati diversi strumenti in base allo specifico documento e alle necessità che esso presenta. Si riportano qui di seguito i diversi software utilizzati in questo ambito:

- *Google Docs*^{G}, riservato alla stesura dei verbali interni ed esterni;
- *Google Slides*^{G}, riservato alla produzione di tutte le presentazioni effettuate dal gruppo (ad esempio i diari di bordo);
- *LaTeX*^{G} per la stesura della documentazione principale, tra cui: *Norme di progetto*, *Piano di progetto*, *Analisi dei requisiti*, *Piano di qualifica* e *Glossario*.

Per ciascuno degli strumenti sopra citati sono stati definiti dei template per la stesura che dovranno essere considerati la base comune per la creazione di un nuovo documento dei *Submarines*.

3.2 Gestione della configurazione

3.2.1 Scopo

Lo scopo del processo è quello di gestire e controllare la produzione di documenti e codice attraverso un opportuno sistema di tracciamento.

3.2.2 Descrizione

Per poter tenere traccia di documenti e codice si deve poter memorizzare la storia di ciascun elemento prodotto, documento o codice che sia. Per fare ciò essi devono essere memorizzati in un repository; tale sistema di archiviazione permette una facile archiviazione e accessibilità degli elementi in esso memorizzati.

3.2.3 Versionamento

I documenti e il codice vengono sviluppati attraverso un sistema di versionamento che illustra il relativo stato di avanzamento. In questa sezione vengono illustrate le regole di versionamento relative ai documenti e al codice sviluppato.

3.2.3.1 Versionamento della documentazione

La versione permette di capire in che stato si trova il documento. Il numero di versione presenta il seguente formato: **X.Y.Z**, dove:

- **X** rappresenta una versione approvata dal responsabile di progetto; il documento risulta pronto per il suo rilascio non appena raggiunge la versione 1.0.0;
- **Y** rappresenta una versione del documento verificata da uno dei verificatori in carica; il numero incrementa ad ogni verifica del documento e tale cifra si azzera ad ogni aumento di X;
- **Z** rappresenta una versione in fase di modifica da parte dei redattori del documento, essa aumenta ad ogni modifica e si azzera all'aumentare di Y.

3.2.4 Sistemi software utilizzati

Per gestire i repository si è scelto il sistema offerto da GitHub che tramite un'interfaccia web consente di accedere a tutti i contenuti in esso archiviati. Consente inoltre di tenere traccia delle loro versioni nel tempo. Per i dettagli su come i *Submarines* devono utilizzare questo sistema si rimanda alla sezione "4.2.3.1 - GitHub" di questo documento.

3.3 Gestione della qualità

3.3.1 Scopo

Lo scopo del processo è quello di definire il modo con cui i *Submarines* si impegnano nel garantire la qualità dei processi.

3.3.2 Descrizione

Il *Piano di qualifica* è un documento volto a illustrare le metriche e le modalità che il gruppo deve utilizzare per valutare la qualità di prodotti e processi.

3.3.3 Attività

Per far si che quanto prodotto dai *Submarines* raggiunga gli standard di qualità imposti, ogni membro deve:

- attenersi al *Piano di qualifica*;
- non smettere di autoformarsi sugli strumenti e tecnologie da utilizzare creando un miglioramento continuo nella formazione del personale.

3.4 Verifica

3.4.1 Scopo

Lo scopo di questo processo è garantire che ogni attività dei processi svolti non introduca errori nel prodotto e che soddisfi i requisiti o le condizioni necessarie per essere considerata accettabile.

3.4.2 Descrizione

Durante questo processo, il compito dei verificatori è quello di effettuare l'analisi dei prodotti del team *Submarines*. Tale analisi si differenzia in due diverse tipologie:

- **Analisi statica**: processo di valutazione di un sistema o di un suo componente. Questo tipo di analisi viene generalmente svolto tramite ispezioni e revisioni e può essere svolta sui documenti così come sul software o su parti di esso;
- **Analisi dinamica**: processo di valutazione di un sistema software o di un suo componente basato sull'osservazione del suo comportamento in esecuzione. Questo tipo di analisi viene generalmente svolta sul prodotto o sulle sue componenti.

3.4.3 Verifica della documentazione

Ogni sezione del corpo del documento è rivista da un verificatore del gruppo che non sia il redattore della parte in verifica. Una volta verificata la sezione in esame dovrà essere incrementato il numero Y, secondo il sistema di versionamento illustrato nella sezione 3.2.3.1 di questo documento.

3.4.4 Verifica del codice

Il codice viene revisionato da un verificatore del gruppo che non sia stato il redattore del codice in questione. Una volta verificato che il codice rispetti le buone norme di codifica e sia funzionante questo potrà essere approvato e passare quindi alla fase di validazione.

3.4.4.1 Test

I test sono l'attività fondamentale per quanto riguarda l'analisi dinamica. Essi servono a dimostrare che il programma funzioni e svolga ciò per cui è stato sviluppato. I test si suddividono in 4 principali categorie, in base all'oggetto della verifica e allo scopo:

- **Test di unità:** sono la più piccola parte che ha senso controllare di un programma (es. una procedura). Verificano in sostanza una singola unità di codice;
- **Test d'integrazione:** verificano la correttezza delle interfacce. Si vuole stabilire il corretto funzionamento delle varie componenti, successivamente ad aver passato il test d'unità, aggregandole man mano e verificando il funzionamento nel complesso;
- **Test di sistema:** verifica l'intera applicazione. Questo test viene dopo il test d'integrazione, il quale scopo è quello di verificare che le componenti siano sia compatibili, che lo scambio di dati tra interfacce e le varie interazioni siano conformi. Inoltre, procedendo così si controlla che i requisiti siano stati soddisfatti;
- **Test di regressione:** verifica l'applicazione dopo aver apportato modifiche al sistema. Lo scopo di questo test è quello di controllare nuove funzionalità non testate, al tempo stesso, anche di garantire che il codice già presente e testato precedentemente non subisca alterazioni al suo comportamento.

3.4.4.2 Nominazione dei test

Ogni test verrà identificato con un codice avente questo pattern: **T[Tipologia][Id]** Nel dettaglio:

- **Tipologia:** indica la tipologia del test che si deve eseguire
  - **U:** unità;
  - **I:** integrazione;
  - **S:** sistema;
  - **R:** regressione.
- **Id:** codice numerico per l'identificazione di test dello stesso tipo, a partire da 1.

3.4.5 Metriche

Per la valutazione del processo di verifica verranno utilizzate le seguenti metriche:

- MQP04 - Code coverage;
- MQP10 - Branch coverage;
- MQP11 - Successo dei test.

3.5 Validazione

3.5.1 Scopo

Lo scopo di questo processo è determinare in maniera oggettiva che il prodotto esaminato sia conforme ai requisiti richiesti e che soddisfi il compito per cui è stato creato.

3.5.2 Descrizione

Tale processo consiste nell'esaminare il prodotto dopo la fase di verifica e assicurarsi che esso sia in linea con i requisiti concordati con il proponente e che soddisfi i bisogni del committente. Sarà il responsabile di progetto ad avere la responsabilità nell'approvare o meno uno specifico aspetto del prodotto. Se il responsabile non dovesse approvare un documento o un componente sarà necessaria un'ulteriore attività di verifica per correggere quanto riscontrato.

3.5.3 Validazione della documentazione

Per poter essere approvato, uno specifico documento deve raggiungere quantomeno la versione 1.0.0 (secondo il sistema di versionamento illustrato nella sezione 3.2.3.1 di questo documento) che indica che il documento è stato approvato e può essere rilasciato per la prima volta.

3.5.4 Validazione del codice

Per poter essere approvato il codice deve rispettare quanto deciso e approvato nel periodo di progettazione. Al programmatore non sarà concesso modificare o alterare le decisioni dei progettisti ma si dovrà solamente limitare a codificare queste scelte progettuali. Il codice inoltre deve rispettare le *buone norme di codifica* presenti nella sezione 2.2.5.3 di questo documento (Stile di codifica).

4 Processi organizzativi

4.1 Gestione organizzativa

4.1.1 Scopo

Lo scopo di questa sezione è quello di normare le modalità di gestione interna del gruppo.

4.1.2 Descrizione

La gestione organizzativa definisce il *way of working* adottato dai *Submarines* durante tutte le attività di progetto.

4.1.3 Ruoli di progetto

Ogni membro dei *Submarines* è tenuto a occupare almeno una volta tutti i ruoli, come definito nel preventivo dei costi, e a farsi carico delle responsabilità che ne derivano. I ruoli ruotano in corrispondenza delle necessità del momento.

4.1.3.1 Responsabile di progetto (Re)

Il suo compito è quello di rappresentare il gruppo all'esterno e di assicurarsi che tutte le attività vengano svolte correttamente ed entro i tempi prestabiliti. Ha piena responsabilità di decisione e approvazione sulle parti che compongono il prodotto da realizzare. Pianifica le attività e redige il *Piano di progetto*.

4.1.3.2 Amministratore (Am)

Il suo compito è quello di occuparsi del funzionamento e sviluppo degli strumenti tecnologici utilizzati dal gruppo durante le attività di progetto. Redige le *Norme di progetto*.

4.1.3.3 Analista (An)

Il suo compito è quello di individuare, analizzare e stilare i diversi servizi che il prodotto finale deve presentare, individuando il problema. Redige l'*Analisi dei requisiti*.

4.1.3.4 Progettista (Pt)

Il suo compito è quello di formulare una soluzione al problema evidenziato dagli analisti, definendo l'architettura del sistema e i diagrammi delle classi.

4.1.3.5 Programmatore (Pr)

Il suo compito è la scrittura del codice necessario al funzionamento del prodotto software. Il suo ruolo è quello di codificare quanto deciso dai progettisti e non può prendere decisioni proprie.

4.1.3.6 Verificatore (Ve)

Il suo compito è quello di verificare quanto prodotto durante le attività di progetto, tra cui: documentazione e codice. Deve inoltre controllare che tali attività vengano svolte seguendo le direttive del documento *Norme di progetto*.

4.1.4 Comunicazione

4.1.4.1 Piattaforma

Come luoghi di incontro vengono utilizzate due piattaforme in base alla tipologia di incontri da effettuare; Discord[G] per gli incontri interni e, generalmente, Google Meet[G] per gli incontri con il proponente.

4.1.4.2 Date

Tenendo conto degli impegni individuali di ciascun membro dei *Submarines* si è deciso di effettuare quantomeno un incontro interno settimanale, con data e ora da decidersi, per discutere dei progressi fatti e per fissare ciò che si dovrà fare di settimana in settimana. Tale incontro verrà fissato tramite il canale Telegram[G] del gruppo; se per qualche motivo alcuni membri non riuscissero a parteciparvi, saranno informati attraverso Telegram e per entrare nel dettaglio potranno recuperare il relativo verbale.

Per quanto riguardo gli incontri esterni, essi verranno svolti quando ritenuti necessari dal gruppo o dal proponente e la data verrà concordata rispettivamente tra i membri del gruppo o tra i *Submarines* e il proponente.

4.1.4.3 Argomenti

Ogni incontro interno viene suddiviso in tre parti principali: la prima in cui si aggiorna il gruppo su quanto svolto individualmente fino a quel momento; la seconda in cui si decide cosa fare nel prossimo periodo, assegnando delle priorità alle diverse *"cose da fare"*; e la terza dedicata a domande o dubbi su come procedere.

Gli argomenti degli incontri esterni verranno invece discussi in seduta comune in modo da illustrare in maniera ottimale eventuali problematiche o idee al proponente.

4.2 Gestione infrastrutture

4.2.1 Scopo

Lo scopo di questo processo è quello di fornire una guida chiara all'utilizzo delle infrastrutture utilizzate per le attività di progetto.

4.2.2 Descrizione

L'utilizzo di determinate infrastrutture permette di incrementare notevolmente la qualità di realizzazione dei processi primari e di supporto, creando un ambiente comune dove coesiste un continuo scambio di informazioni tra i membri del gruppo.

4.2.3 Strumenti collaborativi

4.2.3.1 GitHub

Si è deciso di utilizzare il gestore di repository[G] messo a disposizione da GitHub[G] come base di archiviazione di tutti gli elementi che compongono il progetto didattico. È stata creata un'organizzazione visitabile al seguente link: https://github.com/SubmarinesG4. Ciascun membro del team presenta l'etichetta di *owner* e pertanto dispone di tutte le autorizzazioni legate a tale etichetta; in particolare quella di popolare ciascuna repository.
Le repository sono organizzate in diverse cartelle, ciascuna delle quali è dedicata ad un elemento o sottogruppo specifico (es. cartella *Documentazione interna*) che, tramite il nome, indica cosa vi si trova all'interno.

Ogni repository si presenta con una struttura a n branch[G], tutti ad uso interno, volti ad effettuare caricamenti di versioni di documentazione e codice ancora non verificati e approvati. Il branch principale è denominato *main* e contiene tutta la documentazione e il codice approvato e verificato. Una volta che un ramo (feature) viene verificato e approvato lo si potrà unire al ramo main tramite l'operazione di merge.

4.2.3.2 YouTrack

Per la gestione dei task del progetto software si è deciso di utilizzare YouTrack[G]. Questo strumento permette di gestire in modo centralizzato tutte le attività connesse alla gestione di un progetto software. L'elemento chiave è la Issue[G] a cui è associato un task da eseguire. All'interno di Youtrack sono presenti i Projects che rappresentano macro task da svolgere (ad esempio : un project viene creato per ogni tipologia di documentazione richiesta). Per avere una visione d'insieme dell'avanzamento del progetto software sono presenti le *Agile boards*[G] e *Gantt charts*[G]. I quali insieme all'integrata funzione di tracciamento del tempo, a colpo d'occhio, si è in grado di verificare l'avanzamento del progetto. Tutte queste funzioni sono automatizzate e gestite dalla piattaforma. Questo permette ai componenti del gruppo di concentrarsi nello sviluppo invece che nella gestione del progetto.

4.2.3.3 Altri strumenti a supporto della gestione organizzativa

Si riportano di seguito altri strumenti e tool utilizzati dal gruppo *Submarines* per le attività di progetto:

- **Slack**: strumento utilizzato dal gruppo per la comunicazione interna con il proponente;
- **Telegram**: strumento utilizzato dal gruppo per la comunicazione interna tra i membri;
- **Google Drive**: strumento utilizzato dal gruppo per la stesura di documenti che richiedono modifiche in tempo reale;
- **Gmail**: servizio per la posta elettronica scelto dal gruppo;
- **Discord**: strumento utilizzato dal gruppo per le riunioni interne.

4.3 Formazione del personale

4.3.1 Scopo

Lo scopo di questo processo è quello di definire come avviene il processo di formazione dei membri del gruppo *Submarines*, in particolare per lo studio delle tecnologie utilizzate per produrre la documentazione e il prodotto richiesto.

4.3.2 Descrizione

Questo processo è di fondamentale importanza e permette di ottenere un bagaglio di conoscenza sull'uso di tecnologie che fino ad ora non si erano mai affrontate o comunque non in questo modo.

4.3.3 Modalità

La formazione avviene in maniera autonoma e ciascun membro del gruppo *Submarines* dovrà provvedere a possedere buone conoscenze riguardo a:

- **GitHub**
- **LaTeX**
- **API Gateway[G]**

- **AWS Lambda**[G]
- **Amazon Cognito**[G]
- **Amazon DynamoDB**[G]
- **Linguaggio Typescript**[G]
- **Libreria React**[G]

Nell'eventualità che uno o più componenti del gruppo possiedano già una propria conoscenza riguardo uno degli argomenti sopra elencati, potranno colmare le mancanze degli altri membri attraverso delle loro spiegazioni.

5 Standard ISO/IEC 12207:1995

Questa sezione descrive in dettaglio lo standard internazionale ISO/IEC 12207:1995 scelto dal gruppo per garantire la qualità dei processi di ciclo di vita del software. Secondo tale standard i processi sono suddividi in tre categorie:

- **Processi Primari**
- **Processi di Supporto**
- **Processi Organizzativi**

5.1 Processi primari

Questi processi sono essenziali, in quanto un progetto si definisce tale se in esso è attivo almeno un processo primario; sono inoltre quelli maggiormente legati allo sviluppo del software.

5.1.1 Acquisition

Questo processo ha lo scopo di ottenere il prodotto richiesto dal cliente ed è suddiviso nelle seguenti attività:

- Preparazione dell'acquisizione;
- Scelta del fornitore;
- Monitoraggio dei fornitori;
- Accettazione del cliente.

5.1.2 Supply

Questo processo ha lo scopo di fornire il prodotto richiesto dal cliente ed è suddiviso nelle seguenti attività:

- Preparazione alla proposta;
- Contratto;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Rilascio e completamento.

5.1.3 Development

Questo processo contiene tutte le attività volte allo sviluppo del prodotto software, dall'analisi dei requisiti all'installazione del prodotto. Tali attività sono:

- Analisi dei requisiti;
- Progettazione dell'architettura di sistema;
- Analisi requisiti software;

- Progettazione dell'architettura software;
- Codifica del software e test;
- Integrazione software;
- Installazione del software.

5.1.4 Operation

Questo processo è svolto simultaneamente alla *Maintenance* ed è volto al corretto funzionamento del software e al supporto degli utenti. Le attività di questo processo sono:

- Uso operativo;
- Supporto agli utenti.

5.1.5 Maintenance

Questo processo è svolto simultaneamente alla *Operation* ed è volto a modificare il prodotto software dopo il suo rilascio per correggere difetti riscontrati, migliorare le sue prestazioni o adattarlo a cambiamenti nel sistema operativo. Le attività del processo sono:

- Difetto o richiesta di analisi del cambiamento;
- Implementazione del cambiamento;
- Revisione/Accettazione delle modifiche;
- Migrazione;
- Ritiro del software.

5.2 Processi di supporto

Questi processi aiutano le attività di tutti gli altri processi dell'organizzazione a garantire il successo e la qualità del progetto. Questi processi possono essere attivati da un processo primario o da un altro processo di supporto.

5.2.1 Documentation

Questo processo garantisce lo sviluppo e la manutenzione delle informazioni prodotte e registrate relativamente al prodotto software.

5.2.2 Configuration management

Questo processo ha lo scopo di definire e mantenere l'integrità di tutti i componenti della configurazione (Configuration Items) e di renderli accessibili a chi ne ha diritto.

5.2.3 Quality Assurance

Questo processo ha lo scopo di assicurare che tutti i prodotti di fase (work product) siano conformi con i piani e gli standard definiti.

5.2.4 Verification

Questo processo ha lo scopo di confermare che ciascun work product o servizio realizzato da un processo soddisfi i requisiti specificati. Il processo di Verifica deve essere integrato nei processi di Sviluppo, Fornitura e Manutenzione.

5.2.5 Validation

Questo processo ha lo scopo di confermare che i requisiti siano rispettati quando uno specifico work product (o componente software) sia utilizzato nell'ambiente destinatario.

5.2.6 Joint review

Questo processo ha lo scopo di rivedere con le parti interessate (stakeholders) i processi eseguiti rispetto agli obiettivi definiti negli accordi e le cose da fare per assicurare lo sviluppo di un prodotto che soddisfi i requisiti concordati. La revisione congiunta è svolta tra gli stessi componenti del team quando si revisioni un componente del prodotto oppure tra fornitore e committente quando si revisioni l'intero prodotto.

5.2.7 Audit

Questo processo ha lo scopo di determinare in maniera indipendente la conformità di prodotti e processi selezionati ai requisiti, piani e accordi.

5.2.8 Problem resolution

Questo processo ha lo scopo di assicurare che tutti i problemi individuati siano analizzati e risolti secondo andamenti (trend) riconosciuti.

5.2.9 Usability

Questo processo ha lo scopo di assicurare che siano prese in considerazione, ed opportunamente indirizzate, le considerazioni espresse dalle parti interessate (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.

5.2.10 Product evaluation

Questo processo ha lo scopo principale di assicurare, tramite esami e misure, che il prodotto soddisfi le necessità esplicite ed implicite degli utilizzatori del prodotto stesso.

5.3 Processi organizzativi

Questi processi eseguono funzioni a livello organizzativo aziendale, per supportare altri processi primari, di supporto o organizzativi. I processi organizzativi aiutano nel definire, controllare e migliorare gli altri processi.

5.3.1 Management

Questo processo garantisce lo sviluppo e la manutenzione delle informazioni prodotte e registrate relativamente al prodotto software.

5.3.2 Infrastructure

Questo processo ha lo scopo di definire e mantenere l'integrità delle infrastrutture utilizzata da qualsiasi altro processo, quali: hardware, software, strumenti e tecniche per lo sviluppo del processo stesso.

5.3.3 Improvement

Questo processo ha lo scopo di stabilire, valutare, misurare, controllare e migliorare il ciclo di vita del software.

5.3.4 Human resources

Questo processo ha lo scopo di fornire all'organizzazione risorse umane adeguate e di mantenere le loro competenze consistenti con le necessità del business.

6 Standard ISO/IEC 9126

Questa sezione descrive in dettaglio lo standard ISO/IEC 9126 scelto dal gruppo per garantire la qualità del prodotto software. Tale standard definisce una serie di linee guida volte a descrivere un modello di qualità del software che ha l'obiettivo di migliorare l'organizzazione e i processi per giungere alla qualità del prodotto. Tale standard è composto da quattro parti:

- **Modello della qualità del software**
- **Metriche per la qualità esterna**
- **Metriche per la qualità interna**
- **Metriche per la qualità in uso**

6.1 Modello della qualità del software

Il modello di qualità stabilito dallo standard è composto da sei caratteristiche generali (funzionalità, affidabilità, efficienza, usabilità, manutenibilità, portabilità) e varie sottocaratteristiche misurabili attraverso delle metriche di qualità. Di seguito vengono elencate le sei caratteristiche sopracitate:

- **Funzionalità**: La Funzionalità è la capacità di un prodotto software di fornire funzioni che soddisfano esigenze stabilite, necessarie per operare sotto condizioni specifiche. Le sue sottocaratteristiche sono:
  - **Appropriatezza**: rappresenta la capacità del prodotto software di fornire un appropriato insieme di funzioni per gli specificati compiti ed obiettivi prefissati all'utente.
  - **Accuratezza**: la capacità del prodotto software di fornire i risultati concordati o i precisi effetti richiesti.
  - **Interoperabilità**: è la capacità del prodotto software di interagire ed operare con uno o più sistemi specificati.
  - **Conformità**: la capacità del prodotto software di aderire a standard, convenzioni e regolamentazioni rilevanti al settore operativo a cui vengono applicate.
  - **Sicurezza**: la capacità del prodotto software di proteggere informazioni e dati negando in ogni modo che persone o sistemi non autorizzati possano accedervi o modificarli, e che a persone o sistemi effettivamente autorizzati non sia negato l'accesso ad essi.
- **Affidabilità**: L'Affidabilità è la capacità del prodotto software di mantenere uno specificato livello di prestazioni quando usato in date condizioni per un dato periodo. Le sue sottocaratteristiche sono:
  - **Maturità**: è la capacità di un prodotto software di evitare che si verificano errori, malfunzionamenti o siano prodotti risultati non corretti.
  - **Tolleranza agli errori**: è la capacità di mantenere livelli predeterminati di prestazioni anche in presenza di malfunzionamenti o usi scorretti del prodotto.
  - **Recuperabilità**: è la capacità di un prodotto di ripristinare il livello appropriato di prestazioni e di recupero delle informazioni rilevanti, in seguito a un malfunzionamento. A seguito di un errore, il software può risultare non accessibile per un determinato periodo di tempo, questo arco di tempo è valutato proprio dalla caratteristica di recuperabilità.
  - **Aderenza**: è la capacità di aderire a standard, regole e convenzioni inerenti all'affidabilità.
- **Efficienza**: L'efficienza è la capacità di fornire appropriate prestazioni relativamente alla quantità di risorse usate. Le sue sottocaratteristiche sono:

  - **Comportamento rispetto al tempo**: è la capacità di fornire adeguati tempi di risposta, elaborazione e velocità di attraversamento, sotto condizioni determinate.
  - **Utilizzo delle risorse**: è la capacità di utilizzo di quantità e tipo di risorse in maniera adeguata.
  - **Conformità**: è la capacità di aderire a standard e specifiche sull'efficienza.

- **Usabilità**: L'usabilità è la capacità del prodotto software di essere capito, appreso, usato e benaccetto dall'utente, quando usato sotto condizioni specificate. Le sue sottocaratteristiche sono:
  - **Comprensibilità**: esprime la facilità di comprensione dei concetti del prodotto, mettendo in grado l'utente di comprendere se il software è appropriato.
  - **Apprendibilità**: è la capacità di ridurre l'impegno richiesto agli utenti per imparare ad usare la sua applicazione.
  - **Operabilità**: è la capacità di mettere in condizione gli utenti di farne uso per i propri scopi e controllarne l'uso.
  - **Attrattiva**: è la capacità del software di essere piacevole per l'utente che ne fa uso.
  - **Conformità**: è la capacità del software di aderire a standard o convenzioni relativi all'usabilità.

- **Manutenibilità**: La manutenibilità è la capacità del software di essere modificato, includendo correzioni, miglioramenti o adattamenti. Le sue sottocaratteristiche sono:
  - **Analizzabilità**: rappresenta la facilità con la quale è possibile analizzare il codice per localizzare un errore nello stesso.
  - **Modificabilità**: la capacità del prodotto software di permettere l'implementazione di una specificata modifica (sostituzioni componenti).
  - **Stabilità**: la capacità del software di evitare effetti inaspettati derivanti da modifiche errate.
  - **Testabilità**: la capacità di essere facilmente testato per validare le modifiche apportate al software.

- **Portabilità**: La portabilità è la capacità del software di essere trasportato da un ambiente di lavoro ad un altro. Le sue sottocaratteristiche sono:
  - **Adattabilità**: la capacità del software di essere adattato per differenti ambienti operativi senza dover applicare modifiche diverse da quelle fornite per il software considerato.
  - **Installabilità**: la capacità del software di essere installato in uno specificato ambiente.
  - **Conformità**: la capacità del prodotto software di aderire a standard e convenzioni relative alla portabilità.
  - **Sostituibilità**: è la capacità di essere utilizzato al posto di un altro software per svolgere gli stessi compiti nello stesso ambiente.

6.2 Metriche per la qualità esterna

Le metriche esterne misurano i comportamenti del software sulla base dei test, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti.

6.3 Metriche per la qualità interna

Le metriche interne si applicano al software non eseguibile (ad esempio il codice sorgente) durante le fasi di progettazione e codifica. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale, poiché gli attributi interni influiscono su quelli esterni e quelli in uso.

6.4 Metriche per la qualità in uso

La qualità in uso rappresenta il punto di vista dell'utente sul software. Il livello di qualità in uso è raggiunto quando si è giunti a un buon livello di qualità esterna e interna. La qualità in uso permette di abilitare specificati utenti ad ottenere specificati obiettivi con efficacia, produttività, sicurezza e soddisfazione. Le sue sottocaratteristiche sono dunque:

- **Efficacia**: la capacità del software di mettere in grado gli utenti di raggiungere gli obiettivi specificati con accuratezza e completezza.
- **Produttività**: la capacità di mettere in grado gli utenti di spendere una quantità di risorse appropriate in relazione all'efficacia ottenuta in uno specificato contesto d'uso.
- **Soddisfazione**: è la capacità del prodotto software di soddisfare gli utenti.
- **Sicurezza**: rappresenta la capacità del prodotto software di raggiungere accettabili livelli di rischio di danni a persone, al software, ad apparecchiature o all'ambiente operativo d'uso.

7 Standard ISO/IEC 15504 - SPICE

Lo standard SPICE (Software Process Improvement and Capability Determination) è un insieme di standard tecnici per i processi di sviluppo software. Nello standard SPICE i processi sono suddivisi in 5 categorie:

- Cliente/fornitore;
- Ingegneristico;
- Supporto;
- Gestionale;
- Organizzativo.

7.1 Livelli di capibility e attributi di processo

La capability definisce una scala di maturità a cinque livelli (più il livello base, detto "livello 0") così definiti:

- **Level 0: Incomplete process**: processo incompleto, o il processo non viene eseguito o raggiunge solo parzialmente il suo obiettivo;
- **Level 1: Performed process**: processo svolto, è implementato e gli obiettivi sono raggiunti;
- **Level 2: Managed processs**: processo gestito, la sua attuazione è pianificata, monitorata e adattata;
- **Level 3: Established process**: processo consolidato, si basa su pratiche documentate ed è in grado di raggiungere i propri obiettivi;
- **Level 4: Predictable process**: processo prevedibile, la sua attuazione è condizionata da obiettivi di performance definiti, si basa su un approccio quantitativo;
- **Level 5: Optimizing process**: processo di ottimizzazione, per raggiungere gli obiettivi attuali e futuri, è costantemente migliorato.

La capability di un processo è misurata tramite gli attributi di processo, lo standard definisce 9 attributi (dove il codice rappresenta il livello alla quale vengono applicati):

- **1.1 Process Performance**: numero di obiettivi raggiunti;
- **2.1 Performance Management**: livello di organizzazione degli obiettivi fissati;
- **2.2 Work Product Management**: livello di organizzazione dei prodotti rilasciati;
- **3.1 Process Definition**: livello di adesione agli standard prefissati;
- **3.2 Process Deployment**: livello di ripetibilità del processo;
- **4.1 Process Measurement**: livello di efficacia di applicazione delle metriche al processo;
- **4.2 Process Cotrol**: livello di predicibilità delle valutazioni;
- **5.1 Process Innovation**: misura gli aspetti positivi generati dei cambiamenti attuati dopo una fase di analisi;

- **5.2 Process Optimization**: misura l'efficienza del processo, il rapporto tra i risultati ottenuti e le risorse impegnate.

Ciascun attributo di processo consiste di una o più pratiche generiche che a loro volta sono elaborate in "Indicatori della pratica" che aiutano nella fase di valutazione delle prestazioni. Ogni attributo del processo è valutato secondo una scala a quattro valori (N-P-L-F):

- **N** Not achieved (0 - 15%);
- **P** Partially achieved (>15% - 50%);
- **L** Largely achieved (>50% - 85%);
- **F** Fully achieved (>85% - 100%).

8 Metriche per la qualità di prodotto

8.1 MQP01 - Indice di Gulpease

Indice di leggibilità di un testo, calcola la lunghezza delle parole e delle frasi rispetto al numero totale di lettere. Per fare ciò considera: la lunghezza della parola e quella della frase rispetto al numero di lettere. Questo valore è un numero compreso tra 0 e 100, per calcolarlo si esegue la seguente formula:

$$IDG = 89 + \frac{(300 * NDF) - (10 * NDL)}{NDP}; \quad (1)$$

- **NDF** = Numero di frasi;
- **NDL** = Numero di lettere;
- **NDP** = Numero di parole.

I risultati si dividono per scaglioni:

- < 80 : testi difficili da leggere per chi ha la licenza elementare;
- < 60 : testi difficili da leggere per chi ha la licenza media;
- < 40 : testi difficili da leggere per chi ha la licenza superiore.

8.2 MQP02 - Profondità di una gerarchia

Valore che definisce quanto può essere profonda la gerarchia di una classe. Nel caso avesse una sola classe allora il suo valore sarà pari a 1.

8.3 MQP03 - Numero di parametri per metodo

Indica il numero di parametri che un metodo può avere, un numero elevato di essi indica che è necessario ridurre le funzionalità associate al metodo.

8.4 MQP04 - Code coverage

Indica con una misura percentuale il numero di righe di codice coperte da test rispetto al totale di linee di codice.

8.5 MQP05 - Percentuale dei requisiti obbligatori soddisfatti

Indica la quantità percentuale dei requisiti obbligatori soddisfatti in rapporto al totale dei requisiti obbligatori. Questo valore è dato dalla seguente formula:

$$RS = \frac{ROS}{TRO} * 100; \quad (2)$$

- **RS** = Percentuale requisiti obbligatori soddisfatti;
- **ROS** = Requisiti obbligatori soddisfatti;
- **TRO** = Totale requisiti obbligatori.

8.6 MQP06 - Complessità ciclomatica

La complessità ciclomatica fornisce una misura della complessità di un programma. Essa è identificata dal numero di cammini linearmente indipendenti presenti nel grafo del flusso di controllo del programma. Un valore elevato indica che il software possiede un comportamento poco predicibile, ciò causa grandi rischi. La formula è:

$$CC = E - N + P; \tag{3}$$

- **CC** = Complessità ciclomatica;
- **E** = Numero di congiunzioni tra statement (archi del grafo);
- **N** = Numero di statement (nodi del grafo);
- **P** = Numero di componenti connesse da ogni nodo.

8.7 MQP07 - Numero di bug

Numero di righe di codice che potrebbero comportare ad un risultato diverso da quello che ci si spetta.

8.8 MQP08 - Numero di code smell

Indica una serie di caratteristiche che il codice sorgente può avere. Sono generalmente riconosciute come probabili indicazioni di un difetto di programmazione.

8.9 MQP09 - Linee di Commento per Linee di Codice

$$LCC = \frac{LDC}{LCD} * 100 \tag{4}$$

- **LCC** = Percentuale rappresentante il rapporto tra linee di commento e linee di codice;
- **LDC** = Linee di commento;
- **LCD** = Linee di codice.

8.10 MQP10 - Branch coverage

Percentuale di copertura dei rami condizionati durante i test. Questo valore viene calcolato tramite la seguente formula:

$$BC = \frac{RCC}{RCT} * 100; \tag{5}$$

- **BC** = Percentuale di copertura dei rami condizionati durante i test;
- **RCC** = Rami condizionati coperti dai test;
- **RCT** = Rami condizionati totali.

8.11 MQP11 - Successo dei test

Percentuale di test superati con successo rispetto a quelli implementati. Questo valore è calcolato tramite la formula:

$$SDT = \frac{NTS}{NTT} * 100; \tag{6}$$

- **SDT** = Percentuale di successo dei test;
- **NTS** = Numero di test superati;
- **NTT** = Numero di test totali.

8.12 MQP12 - Numero di vulnerabilità

Indica il valore di vulnerabilità presenti nel codice sorgente del prodotto.

9 Metriche per la qualità di processo

9.1 MPC01 - SPICE

Questa metrica viene utilizzata per misurare la qualità dei processi impiegati e per fornire una valutazione ad essi. Lo standard è illustrato all'interno del capitolo 7.

9.2 MPC02 - Budgeted cost of work scheduled

Indica una previsione del budget speso dal gruppo per il lavoro svolto dall'inizio del progetto fino alla data corrente. Il suddetto valore è reperibile all'interno del file Piano di Progetto.

9.3 MPC03 - Actual cost of work performed

Indica il totale di tutte le spese sostenute dal team, questo valore deve essere minore o uguale a quello presente nel file Piano di Progetto.

9.4 MPC04 - Budgeted cost of work performed

Indica il valore effettivo del prodotto ottenuto fino al momento in cui l'indice viene calcolato.

9.5 MPC05 - Schedule variance

Indica la distanza percentuale tra la programmazione preventivata e quella effettiva. La formula per il calcolo di questo valore è:

$$SV = \frac{100 * (BCWP - BCWS)}{BCWS}; \tag{7}$$

- **SV** = Schedule variance;
- **BCWP** = Budgeted cost of work performed;
- **BCWS** = Budgeted cost of work scheduled.

9.6 MPC06 - Budget variance

Indica la percentuale delle spese sostenute dall'inizio del progetto fino al momento in cui viene calcolato, rientrando nel budget di spesa previsto per la data corrente. La formula per il calcolo di questo valore è:

$$BV = \frac{100 * (BCWS - ACWP)}{BCWS}; \tag{8}$$

- **BV** = Budget variance;
- **BCWS** = Budgeted cost of work scheduled;
- **ACWP** = Actual cost of work performed.