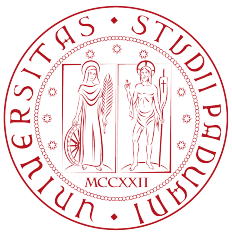
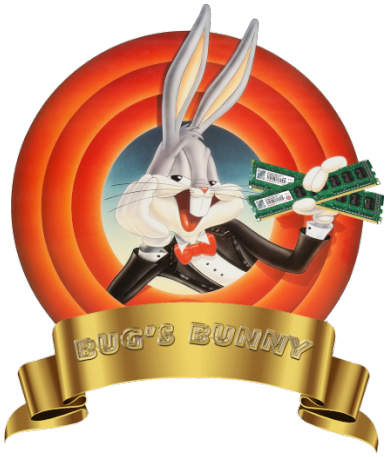


Università degli Studi di Padova
Ingegneria del Software
Anno Accademico: 2021/2022



Bug's Bunny

Norme di Progetto

bugsbunnyteam@protonmail.com

Versione: 2.0.0
Redazione: Angela Arena,
Matteo Tossuto,
Giulio Zanatta
Verifica: Angela Arena,
Marco Bellò
Approvazione: Tommaso Di Fant
Uso: Esterno

Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
2.0.0	04-09-2022	Tommaso Di Fant	Responsabile	Approvazione del documento
1.0.5	02-09-2022	Angela Arena	Amministratore	Modificate §3.2.2., §3.2.2.4 (Verificatore: Marco Bellò)
1.0.4	31-08-2022	Giulio Zanatta	Amministratore	Aggiornamento §2.1.6, §2.1.7 (Verificatore: Angela Arena)
1.0.3	17-08-2022	Matteo Tossuto	Amministratore	Aggiornamento §2.2.4, §2.2.4.1 (Verificatore: Angela Arena)
1.0.2	26-07-2022	Matteo Tossuto	Amministratore	Aggiornamento §3.1.6, §3.2.2 e §4.1.5.1 (Verificatore: Angela Arena)
1.0.1	26-07-2022	Marco Volpato	Responsabile	Aggiunta numero versione
1.0.0	24-06-2022	Angela Arena	Responsabile	Approvazione del documento
0.2.3	23-06-2022	Marco Volpato	Verificatore	Verifica e correzione errori minori
0.2.2	21-06-2022	Marco Bellò	Verificatore	Segnatura glossario e verifica
0.2.1	21-06-2022	Marco Bellò	Amministratore	Stesura convenzioni liste §3.1.7
0.2.0	20-06-2022	Marco Bellò	Analista	Stesura §5
0.1.9	20-06-2022	Giulio Zanatta	Analista	Stesura §3.3.2
0.1.8	16-06-2022	Matteo Tossuto	Analista	Stesura §7 e revisione generale documento
0.1.7	15-06-2022	Giulio Zanatta	Analista	Modifica registro modifiche, stesura §2.2.2.5, §2.2.3, §2.3
0.1.6	13-06-2022	Marco Bellò	Analista	Modifica registro modifiche, stesura §6

Versione	Data	Nominativo	Ruolo	Descrizione
0.1.5	10-06-2022	Marco Volpato	Verificatore	Revisione §3.2.2
0.1.4	10-06-2022	Matteo Tossuto	Analista	Stesura §8
0.1.3	29-05-2022	Matteo Tossuto	Verificatore	Revisione §3
0.1.2	26-05-2022	Marco Volpato	Verificatore	Revisione generale NdP
0.1.1	23-05-2022	Matteo Tossuto	Analista	Revisione §3
0.1.0	18-05-2022	Marco Volpato	Analista	Revisione servizi AWS
0.0.4	16-05-2022	Matteo Tossuto	Analista	Aggiunte §3
0.0.3	16-05-2022	Marco Volpato	Analista	Stesura §3
0.0.2	14-05-2022	Matteo Tossuto	Analista	Stesura §3.5
0.0.1	13-05-2022	*Bug's Bunny*	Analisti	Creazione bozza documento, introduzione e paragrafi

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Le *Norme di Progetto*$_G$ hanno l'obbiettivo di stabilire e normare i processi utili allo sviluppo del prodotto attraverso la definizione di *best practices*$_G$ e del *way of working*$_G$. Il presente documento deve essere visionato da ogni membro del gruppo, il quale dovrà attenersi alle norme qui descritte durante l'intera durata del progetto. Questo documento viene frequentemente aggiornato per assicurarsi che le norme descritte rispecchino la realtà operativa del gruppo rendendo il documento un'importante risorsa.

1.2 Scopo del prodotto

Il capitolato propone lo sviluppo di una piattaforma simile ad una guida Michelin$_G$, basandosi sulle esperienze che vengono condivise sui social network$_G$ Instagram$_G$ e TikTok$_G$. La richiesta prevede che la piattaforma sia in grado di ispezionare ed estrarre determinate informazioni quali immagini, audio o commenti relativi al contenuto analizzato, dalle storie dei relativi social network$_G$. L'obiettivo è quello di riuscire a formare una mappa di location e determinare se quest'ultime vengono recensite negativamente o positivamente, e a tal scopo stilare un ranking$_G$ di esse incrociando ciò che viene analizzato dalla piattaforma con altre classifiche per rendere omogeneo il risultato. Il progetto avrà un'architettura a microservizi.

1.3 Glossario

Nei documenti possono essere presenti termini sconosciuti o ambigui a seconda del contesto. Al fine di poter garantire una lettura priva di incomprensioni o interpretazioni differenti, viene fornito un *Glossario* contenente i suddetti termini e la loro spiegazione.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Capitolato d'Appalto C4**
- **Regolamento del progetto didattico**

1.4.2 Riferimenti informativi

- **ISO/IEC 12207:1997**
- **ISO/IEC 9126**
- **ISO/IEC 15504 - SPICE**
- **V - Model**
- **Gestione di Progetto**
- **Progettazione software**
- **Python PEP8**

- **Airbnb Javascript Style Guide**

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Il processo di fornitura viene svolto al fine di comprendere al meglio la richiesta di appalto del *proponente*: il documento che verrà creato include le norme a cui i membri del gruppo *Bug's Bunny* devono fare riferimento al fine di rivestire in modo adeguato il ruolo di *fornitore* dell'azienda Zero12 e dei committenti *Prof. Tullio Vardanega* e *Prof. Riccardo Cardin*.

2.1.2 Aspettative

Durante lo svolgimento del progetto il gruppo intende mantenere un rapporto di costante collaborazione con il *proponente* in modo da facilitare il lavoro svolto. In particolare, si vuole:

- Stimare costi e tempistiche del lavoro;
- Determinare aspetti chiave che il prodotto dovrà soddisfare;
- Determinare vincoli e requisiti sui processi;
- Accordarsi sulla qualifica del prodotto.

2.1.3 Valutazione dei capitolati

L'attività di Valutazione dei capitolati, che sfocia nell'omonimo documento, serve a spiegare le motivazioni per la scelta di un capitolato tra quelli proposti. Il documento *Valutazione dei Capitolati* viene redatto dopo un'attenta analisi svolta dai membri. Per ogni capitolato sono indicate:

- **Informazioni generali**: nome del progetto, il *proponente* e il committente;
- **Descrizione**: sintesi del prodotto, caratteristiche principali e obiettivi del progetto;
- **Tecnologie utilizzate**: elenco delle tecnologie richieste;
- **Vincoli Generali**: vincoli imposti dal *proponente* e da rispettare durante il progetto;
- **Aspetti positivi, criticità e fattori di rischio**: considerazioni fatte dal gruppo riguardanti gli aspetti positivi e sulle criticità del capitolato;
- **Conclusioni**: ragioni per le quali il gruppo accetta o rifiuta il capitolato.

2.1.4 Documentazione fornita

Il gruppo fornisce all'azienda Zero12 e ai committenti *Prof. Tullio Vardanega* e *Prof. Riccardo Cardin* i seguenti documenti, essenziali per tracciare le attività svolte e per iniziare la fase di implementazione:

- *Analisi dei Requisiti*: documento che contiene l'analisi approfondita del capitolato scelto, comprendente tutti i *requisiti* e i *casi d'uso*$_G$ individuati;
- *Piano di Progetto*: documento nel quale viene pianificato nel dettaglio il modo di lavorare del gruppo, contenente un preventivo riguardante tempistiche, l'analisi dei rischi, la pianificazione delle attività e il consuntivo;
- *Piano di Qualifica*: documento dove vengono stabilite e descritte le modalità di *verifica*$_G$ e *validazione*$_G$, in modo da garantire la qualità del prodotto.

I seguenti documenti, invece, sono forniti solo all'azienda Zero12:

- Piano di test di unità;
- Documentazione dettagliata di tutte le API$_G$ usate;
- Schema Design$_G$ relativo alla base dati;
- Diagrammi UML$_G$ relativi agli Use Cases$_G$ di progetto.

2.1.5 Metriche

Il processo di foritura non fa uso di metriche qualitative particolari.

2.1.6 Preparazione al collaudo del prodotto

In vista della fase di collaudo, il gruppo mira a garantire la qualità del prodotto software e si avvale delle seguenti tipologie di test verificarne la correttezza, presenti nel *Piano di Qualifica v.2.0.0*:

- Test di unità
- Test di integrazione
- Test di sistema
- Test di accettazione

Al momento del collaudo il gruppo potrà affermare che, se superati positivamente i testi, avrà un prodotto di qualità e corretto.

Al committente dovrà essere dimostrato che i requisiti obbligatori definiti nel documento *Analisi dei Requisiti* sono stati realizzati al meglio, ed infine che la suite di test effettuati ha portato ad un risultato in linea con le aspettative.

2.1.7 Collaudo e consegna

Il gruppo *Bug's Bunny* provvederà alla consegna del seguente materiale al proponente e ai committenti:

- Documentazione del prodotto;
- Codice sorgente;
- Specifica Architetturale *v.1.0.0*;
- Manuale Utente *v.1.0.0*;
- Glossario *v.2.0.0*;

2.2 Sviluppo

2.2.1 Descrizione e aspettative

Lo scopo del processo di sviluppo è descrivere i compiti e le attività da svolgere relative al prodotto *software* richiesto dal *proponente*. Le attività coinvolte riguardano l'*Analisi dei Requisiti*$_G$, la progettazione e la codifica.
Le aspettative sono le seguenti:

- Stabilire gli obiettivi di sviluppo;
- Stabilire i vincoli tecnologici e di design$_G$;
- Realizzare un prodotto finale che superi i *test*, che soddisfi i *requisiti* e le richieste del *proponente*.

2.2.2 Analisi dei requisiti

L'analisi dei requisiti$_G$ è un'attività, che sfocia nell'omonimo documento, dove vengono individuati tutti i *requisiti* che il *proponente* richiede per la realizzazione del prodotto.
Il documento *Analisi dei Requisiti*$_G$ va steso in maniera efficace ed è molto importante in quanto coinvolto in diverse fasi della realizzazione del prodotto: oltre che definire funzionalità e *requisiti* individuati e concordati col cliente, fornisce ai *Programmatori* riferimenti precisi e affidabili, ai *Verificatori* riferimenti per il processo di *verifica*$_G$ ed è la base dalla quale partire per eventuali raffinamenti successivi, garantendo un continuo miglioramento del prodotto. Ogni requisito può essere ricavato da diverse fonti:

- **Capitolato d'Appalto**: attraverso la lettura del capitolato;
- **Casi d'uso**: estrapolati da uno o più casi d'uso$_G$;
- **Verbali**: attraverso riunioni interne o incontri con l'azienda *proponente*.

2.2.2.1 Struttura

La struttura del documento *Analisi dei Requisiti*$_G$ è la seguente:

- **Introduzione**: contiene scopo del documento, introduzione al progetto e riferimenti;
- **Descrizione**: contiene informazioni riguardanti il prodotto, la piattaforma d'esecuzione e la descrizione degli utenti interessati;
- **Casi d'uso**: vengono identificati gli attori che interagiscono con le componenti del sistema e le interazioni tra sistema, attori ed elementi esterni;
- **Requisiti**: rappresentati in tabelle che riportano le seguenti informazioni:
  - **Codifica**: codice di riferimento del requisito;
  - **Classificazione**: obbligatorio o desiderabile;
  - **Descrizione**: breve descrizione del requisito;
  - **Fonti**: da dove è stato ricavato il requisito.

2.2.2.2 Classificazione dei requisiti

La classificazione requisiti verrà effettuata mediante la seguente codifica:

R[Tipo][Numero]

Nome	Descrizione
R	Abbrevia "Requisito"
Tipo	**F**: requisito funzionale, ossia la definizione di una particolare caratteristica che deve essere inclusa nel software
	V: requisito di vincolo che rappresenta dei vincoli avanzati dal *proponente*
	Q: requisito di qualità, relativo alle regole di qualità del software (efficienza ed efficacia)
	P: requisito di prestazione, relativo alle prestazioni del software
Numero	Codice identificativo

Tabella 2: Tabella degli elementi che classificano i requisiti

2.2.2.3 Classificazione dei casi d'uso

La codifica scelta per la rappresentazione dei casi d'uso$_G$ è la seguente:

UC[CodiceCaso][CodiceSottoCaso]

Nome	Descrizione
UC	Acronimo di "Use Case$_G$"
CodiceCaso	Identificativo del caso d'uso generico
CodiceSottoCaso	Identificativo opzionale per gli eventuali sotto casi del caso d'uso

Tabella 3: Tabella degli elementi che classificano i casi d'uso

2.2.2.4 Struttura dei casi d'uso

Nome	Descrizione
Codifica	Identificativo del caso d'uso$_G$ generico
Nome	Nome descrittivo del caso d'uso$_G$
Descrizione Grafica	Realizzata con UML$_G$
Attore	Interagisce col sistema per il raggiungimento di un obiettivo
Descrizione	Breve descrizione dell'obiettivo
Scenario	Rappresentato attraverso elenco numerato degli eventi
Estensioni	Opzionali, per modellare scenari alternativi. Se si verifica, il caso d'uso$_G$ ad essa collegata si interrompe
Precondizioni	Condizioni del sistema prima del verificarsi del caso d'uso$_G$
Postcondizioni	Condizioni del sistema dopo il verificarsi del caso d'uso$_G$

Tabella 4: Tabella della struttura dei casi d'uso

2.2.3 Progettazione

L'attività di progettazione definisce le caratteristiche che il prodotto richiesto deve avere in modo da fornire una soluzione che soddisfa i requisiti specificati nell'*Analisi dei Requisiti*$_G$. Il procedimento è infatti l'opposto rispetto a quello utilizzato nell'Analisi dei Requisiti$_G$: in quest'ultimo avviene una suddivisione del problema in parti per poter comprendere al meglio il dominio applicativo, mentre nella progettazione si ricostruisce il problema specificando ogni funzionalità di ogni parte.

2.2.3.1 Technology Baseline

Misura le specifiche della progettazione nelle tecnologie indivduate per realizzare l'architettura del prodotto. Vengono rese note:

- Tecnologie adottate;
- Relazioni e interazioni tra i vari componenti;
- *Proof of Concept*$_{\text{G}}$, un prototipo per dimostrare in modo pratico la corretta compatibilità dellle tecnologie utilizzate.

2.2.3.2 Product Baseline

Rappresenta la base architetturale definita coerentemente nella Technology Baseline$_{\text{G}}$. Deve comprendere:

- *Design Pattern*$_{G}$ che sono stati utilizzati e una loro descrizione;
- Diagrammi UML$_{\text{G}}$, in particolare diagrammi delle classi, sequenza, package;
- Tracciamento dei requisiti che devono essere soddisfatti da una classe.

2.2.4 Codifica

La codifica ha lo scopo di normare l'effettiva realizzazione del prodotto richiesto. I *Programmatori* dovranno attenersi a queste norme durante la fase di programmazione e implementazione. L'uso di norme e convenzioni è fondamentale per permettere la generazione di codice leggibile e uniforme, agevolare la manutenzione e i processi di *verifica*$_{\text{G}}$ e *validazione*$_{\text{G}}$.

2.2.4.1 Stile della codifica

- **Directory**: i nomi delle Directory dovranno seguire la convenzione *camelCase*;
- **Classi**: i nomi delle classi dovranno seguire la convenzione *snake_case*;
- **Parentesi**: le parentesi devono essere sempre utilizzate alla dichiarazione di un costrutto e all'inizio di esso, anche se esso è vuoto o contiene un'istruzione sola;
- **Variabili**: il nome delle variabili è scritto in inglese e in minuscolo, ad eccezione di un nome composto che dovrà avere il primo nome minuscolo e l'iniziale del secondo maiuscola;
- **Costanti**: il nome delle costanti deve essere scritto in inglese e in maiuscolo;
- **Univocità dei nomi**: classi, variabili, metodi e qualsiasi costrutto deve avere un nome, in inglese, significativo ed essere univocamente identificabile;
- **Commenti**: i commenti possono essere scritti in inglese o italiano, all'inizio di un blocco di codice;

2.2.5 Metriche

Metrica	Descrizione	Riferimento
MPR2	Percentuale Requisiti Obbligatori Soddisfatti	§6.2 MPR2
MPP6	Requirements Stabilty Index	§6.1 MPP6

Tabella 5: Metriche utilizzate per la valutazione di analisi dei requisiti

Metrica	Descrizione	Riferimento
MPR5	Accoppiamento tra classi	§6.2 MPR5
MPR6	Profondità delle gerarchie	§6.2 MPR6
MPR12	Facilità di utilizzo	§6.2 MPR12

Tabella 6: Metriche utilizzate per la valutazione della progettazione

Metrica	Descrizione	Riferimento
MPR6	Profondità Gerarchie	§6.2 MPR6
MPR7	Numero Attributi per Classe	§6.2 MPR7
MPR8	Numero Parametri per Metodo	§6.2 MPR8
MPR9	Linee Codice per Metodo	§6.2 MPR9
MPR10	Linee Commento per Codice	§6.2 MPR10
MPR14	Complessità Ciclomatica Media	§6.2 MPR14

Tabella 7: Metriche utilizzate per la valutazione della codifica

2.2.6 Strumenti e linguaggi di programmazione

Di seguito vengono elencati e descritti i vari strumenti e linguaggi di programmazione utilizzati durante il progetto, siano essi richiesti dal *proponente* o scelti autonomamente:

IDE

- *PyCharm*: IDE per lo sviluppo tramite linguaggio Python$_G$;

AWS

- *Fargate*: servizio serverless$_G$ per la gestione di container;

- *RDS*: database$_G$ relazionale in modalità serverless$_G$;
- *Lambda*: servizio di calcolo serverless$_G$, basato sugli eventi;
- *API Gateway*: servizio gestito che semplifica creazione, pubblicazione, manutenzione, monitoraggio e protezione delle API$_G$ dei vari microservizi sviluppati;
- *Rekognition*: servizio per l'estrazione di informazioni da immagini e video tramite tecniche di Machine Learning;
- *Comprehend*: servizio per l'estrazione di informazione da testi, tramite tecniche di Natural Language Processing;
- *Cognito*: servizio che permette di aggiungere strumenti di registrazione degli utenti, accesso e controllo degli accessi.

Linguaggi di programmazione

- *Python*;
- *Typescript*.

Librerie e framework

- *Svelte*: libreria per lo sviluppo web frontend$_G$.

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

Il processo di documentazione comprende le attività di stesura e aggiornamento di tutti i documenti creati durante il ciclo di vita del *software* in modo da renderli formalmente concordi. In particolare, in questa sezione ne vengono normate le attività, comprendenti stesura, *verifica*$_G$ e *validazione*$_G$.

3.1.2 Aspettative

Durante questo processo il *team* ha le seguenti aspettative:

- Ideare una struttura ben organizzata comune a tutti i documenti;
- Stesura di norme per facilitare tale processo.

3.1.3 Documenti prodotti

I documenti prodotti saranno di due tipi:

- **Formali**:
  - **interni**: riguardanti le dinamiche del gruppo;
  - **esterni**: di interesse ai committenti e al *proponente*.

  In particolare, i documenti formali prodotti saranno:

Nome	Descrizione	Uso
Norme di Progetto	Contiene tutte le regole stabilite dai membri alle quali attenersi durante l'intera durata del progetto	Interno
Valutazione dei Capitolati	Contiene l'analisi dei capitolati messi a disposizione, evidenziandone pregi e difetti. Contiene, inoltre, il capitolato scelto dal gruppo	Interno
Analisi dei Requisiti	Descrive i *requisiti* che il prodotto dovrà possedere per essere in linea con le richieste dei committenti	Esterno
Piano di Progetto	Contiene la pianificazione di tutte le attività previste, comprendente il preventivo delle spese e una previsione dell'impegno in ore per ogni membro del gruppo	Esterno
Piano di Qualifica	Descrive i criteri di valutazione della qualità impiegate dal gruppo	Esterno

Nome	Descrizione	Uso
Glossario	Elenco di tutti i termini presenti nella documentazione che, secondo i membri, necessitano di una definizione al fine di chiarirne il significato e rimuovere eventuali ambiguità	Esterno

Tabella 8: Tabella dei documenti formali prodotti

- **Informali**: In particolare, i documenti informali prodotti saranno:

Nome	Descrizione	Uso
Verbali interni	Contengono le informazioni e le decisioni prese durante gli incontri tra i membri del gruppo	Interno
Verbali esterni	Comprendono le informazioni ed i chiarimenti ricevuti durante gli incontri tra i membri ed il committente o tra i membri ed il *proponente*	Esterno

Tabella 9: Tabella dei documenti informali prodotti

3.1.4 Directory di un documento

Le directory prendono il nome dal documento contenuto, nella forma **Nome_Del_Documento**. Questa viene poi collocata, a seconda del tipo di contenuto, in una delle directory **Interni** o **Esterni**.

3.1.5 Ciclo di vita di un documento

Ogni documento passa per le seguenti fasi:

- **Stesura**: il documento viene creato, aggiornato e modificato fino ad *Approvazione*;
- **Verifica**: il documento viene verificato sia dal punto di vista grammaticale che contenutistico; la verifica viene effettuata da uno o più *Verificatori*;
- **Approvazione**: il documento viene approvato da un *Approvatore* che ne consente il rilascio.

3.1.6 Struttura dei documenti

3.1.6.1 Frontespizio

La prima pagina di ogni documento sarà così strutturata:

- **Logo Università e didascalia**: posizionato in alto a sinistra e al suo fianco, al centro, la scritta "Università degli studi di Padova", corso del progetto e anno accademico;
- **Logo gruppo e nome**: entrambi al centro, il nome del gruppo è sotto il logo;

- **Titolo del documento**;
- **Recapito email del gruppo**;
- **Informazioni del documento**: Redattore, Verificatore, Approvatore ed Uso(interno od esterno).

3.1.6.2 Registro delle modifiche

Ogni documento formale presenta subito dopo il frontespizio il registro modifiche, che viene aggiornato alla chiusura di una *Pull Request*$_G$ da parte di un verificatore. Esso è una tabella che traccia tutte le modifiche significative apportate al documento durante il suo ciclo di vita. Ogni voce della tabella riporta:

- **Versione**: numero versione documento dopo la modifica;
- **Data**: data in cui è stata effettuata la modifica;
- **Nominativo**: nome e cognome dell'autore della modifica;
- **Ruolo**: ruolo dell'autore che ha apportato la modifica;
- **Descrizione**: sintetica descrizione delle modifiche apportate e nominativo del verificatore delle modifiche;

3.1.6.3 Corpo del documento

Tutte le pagine del corpo del documento contengono un intestazione composta da:

- **Logo Gruppo**: a sinistra;
- **Titolo Documento**: a destra;

e un piè di pagina al cui centro c'è la pagina corrente del documento.

3.1.6.4 Verbali I verbali, sia interni che esterni, presentano una struttura fissa e a differenza degli altri documenti, essendo informali, non sono soggetti a versionamento e non riportano il registro delle modifiche. La struttura sarà quindi la seguente:

- **Logo Università e logo gruppo**;
- **Uso**: indica se il verbale è esterno o interno;
- **Data**: data della riunione;
- **Informazioni del documento**: Redattore, Verificatore ed Approvatore;
- **Orari:** ora di inizio e ora di fine della riunione;
- **Agenda:** contiene la motivazione per cui si è deciso di fissare la riunione;
- **Resoconto della riunione:** riporta l'esito della discussione dei singoli argomenti trattati;
- **Issue aperte:** vengono elencate le issue$_G$ che sono state aperte su Github$_G$ durante il verbale.

3.1.7 Norme tipografiche

3.1.7.1 Convenzioni di denominazione

I nomi dei file relativi alla documentazione presentano l'iniziale della prima e terza parola che lo compone in maiuscolo e presenta separazione tra le parole.

3.1.7.2 Stili di testo

- **Grassetto**: viene usato nei titoli delle sezioni e dei paragrafi e per enfatizzare parole;
- **Corsivo**: viene usato per i nomi propri (membri del gruppo, proponente e committenti) e per citare il nome di un documento.

3.1.7.3 Convenzioni scrittura liste puntante

In una lista senza ulteriori liste innestate gli elementi si scrivono con la prima lettera maiuscola, sia che siano in **grassetto**, sia che siano in *corsivo*, sia che non abbiano alcun tipo di formattazione. Gli elementi vanno separati con un *punto e virgola*, e l'ultimo segna la fine della lista con un *punto*. La descrizione va anticipata dai *due punti*.
Per quanto riguarda le liste innestate le differenze risiedono in due fattori: la lista a maggiore profondità ha gli elementi con la prima lettera in minuscolo, e gli elementi padre di una lista innestata trattano la suddetta lista come una descrizione, quindi introducendola con i *due punti*.
Con "maggiore profondità" si intende localmente, cioè si intende il blocco di elementi di una lista che non ha, per nessuno dei suoi elementi, una lista figlia. Nell'esempio sottostante ciò è reso in maniera più chiara, per capirne il funzionamento si può immaginare che la divisione dei blocchi della lista avvenga similmente all'indentazione del codice sorgente.
Possono esistere dei casi nei quali è accettato deviare da queste convenzioni per questioni di leggibilità o comprensione (ne sono un esempio le liste dei riferimenti normativi e informativi), si lascia quindi a discrezione di relatori e verificatori un certo grado di libertà. Di seguito viene mostrato un esempio per chiarezza.

3.1.7.4 Lista senza nested lists, diverse formattazioni

- **Elemento Livello 1 Grassetto**: descrizione;
- *Elemento Livello 1 Corsivo*: descrizione;
- Elemento Livello 1: descrizione.

3.1.7.5 Lista con nested lists

- Primo elemento livello 1:
  - Primo Elemento livello 2:
    - primo elemento livello 3;
    - secondo elemento livello 3;
    - terzo elemento livello 3.

  - Secondo elemento livello 2;
  - Terzo elemento livello 2.
- Secondo elemento livello 1;
- Terzo elemento livello 1:
  - quarto elemento livello 2;
  - quinto elemento livello 2.
- Quarto elemento livello 1.

3.1.8 Metriche

Metrica	Descrizione	Riferimento
MPR1	Indice di Gulpease	§6.2 MPR1
MPR13	Errori Ortografici	§6.2 MPR13

Tabella 10: Metriche utilizzate per la valutazione della documentazione

3.1.9 Strumenti

- **LaTeX$_G$:** stesura in bella copia dei documenti caricati poi sul repository$_G$ del gruppo;
- **Visual Studio Code$_G$:** usato per la scrittura di codice e documenti, e visione del versioning tramite collegamento a GitHub$_G$;
- **Google Docs$_G$:** cartella condivisa contenente documenti in brutta copia, permette scrittura collaborativa contemporanea da parte di tutti i membri con visualizzazione delle modifiche in live;

3.2 Processo di configurazione

3.2.1 Scopo

Il processo di gestione della configurazione ha lo scopo di gestire in maniera ordinata e sistematica la produzione di documenti. In particolare, un elemento sottoposto a configurazione ha una collocazione, una denominazione e uno stato definiti, oltre a norme e versionamento.

3.2.2 Versionamento

3.2.2.1 Tecnologie adottate

Per gestire il versionamento del codice sorgente, viene utilizzato il sistema di versionamento distribuito *Git$_G$*, con un *repository$_G$* remoto presente su *GitHub$_G$*.

3.2.2.2 Repository

Il gruppo *Bug's Bunny* ha scelto di creare i seguenti *repository*$_G$ per il progetto:

- **docs**: per il versionamento dei documenti.
- **poc**: per la gestione e il versionamento del *POC*$_G$.
- **bunnyfood**: per la gestione della piattaforma e del versionamento del codice.

3.2.2.3 Struttura dei repository

Il gruppo mantiene diversi repository$_G$ separati, ognuno con struttura e fini propri. Di base, ogni repository$_G$ conterrà i seguenti file:

- `README.md`: contiene istruzioni d'uso, installazione, e sviluppo proprie del repository$_G$. Ne viene inoltre descritta la struttura nel dettaglio. Redatto in formato *Markdown*.
- `.gitignore`: descrive quali file ignorare dal sistema di versionamento, deve essere utilizzato per evitare la proliferazione di file inutili e che non necessitano di versionamento. È consigliato utilizzare servizi di generazione per generare le regole.
- `.editorconfig`: descrive regole di formattazione, quali indentazione e stile dei fine riga. Viene utilizzato dai formattatori automatici degli *IDE* ed editor di testo. Segue il formato *EditorConfig*.

3.2.2.4 Norme di branching

Il *repository*$_G$ inerente alla documentazione sarà composto da diversi *branch*$_G$:

- `master`: *branch*$_G$ principale che viene aggiornato quando un documento è approvato o per introdurre nuove funzionalità;
- un *branch*$_G$ per ogni documento o tipologia di documento.

Il *repository*$_G$ inerente alla realizzazione della piattaforma sarà composto da diversi *branch*$_G$:

- `master`:*branch*$_G$ principale che viene aggiornato quando la piattaforma è realizzata o per introdurre nuove funzionalità;
- *frontend*$_G$: *branch*$_G$ è stato utilizzato il framework$_G$ Svelte per la realizzazione del Frontend$_G$;
- *dev*$_G$: *branch*$_G$ è stato utilizzato il linguaggio Python$_G$.

3.2.2.5 Modifiche alla repository del codice

Tutti i membri del gruppo possono apportare modifiche ai file elaborati, previa assegnazione della relativa *Issue*$_G$, salvo quelli presenti nel ramo *master* o nel ramo *dev*.
Il ramo *master* verrà aggiornato alle ultime modifiche tramite merge di una *Pull Request*$_G$ di approvazione dal ramo *dev* prima del rilascio di una versione finale.
Mentre il ramo *dev* verrà aggiornato con le ultime modifiche tramite merge, di una *Pull Request*$_G$ di verifica, dagli altri rami di lavoro, dopo la chiusura di una *Issue*$_G$.
Per entrambi i rami le *Pull Request*$_G$ vanno chiuse preferibilmente tramite sistema *"Squash and Merge"*.

3.2.2.6 Modifiche alla repository per la realizzazione della piattaforma

Tutti i membri del gruppo possono apportare modifiche ai file elaborati, salvo quelli presenti nel ramo *master*
Il ramo *master* verrà aggiornato alle ultime modifiche tramite merge dal ramo *dev* prima del rilascio di una versione finale.
Mentre il ramo *dev* verrà aggiornato con le ultime modifiche tramite merge, dall'altro ramo di lavoro.

3.2.3 Metriche

Il processo di configurazione non fa uso di metriche qualitative particolari.

3.2.4 Strumenti

Per la gestione della configurazione o versionamento il gruppo utilizza *Visual Studio Code*$_G$ o i seguenti client *Git*$_G$:

- **GitHub Desktop**: Client ufficiale di *GitHub*$_G$ utilizzato per la gestione delle *repository Git*$_G$;
- **GitKraken**.

3.3 Gestione della qualità

3.3.1 Scopo

Per la gestione della qualità è dedicato il documento *Piano di Qualifica*$_G$: il documento fissa i *requisiti* qualitativi individuati dagli *stakeholder*$_G$ e le metriche$_G$ per la *verifica* e *validazione* per garantire la qualità del prodotto finale.

3.3.2 Attività di processo

Si possono individuare delle attività principali nel processo di gestione della qualità:

- **Pianificazione**: attività volta a definire gli obiettivi e i metodi con i quali raggiungerli;
- **Esecuzione**: attività che si occupa dell'esecuzione di ciò che è stato pianificato nell'attività precedente;
- **Controllo**: attività per analizzare i risultati ottenuti con lo scopo di misurarli e capire se l'obiettivo è stato raggiunto.

Queste attività mirano soprattutto all'*Auto-Miglioramento* del processo, facendo un'analisi dei risultati ottenuti con lo scopo di migliorare e/o correggere eventuali attività che non raggiungono obiettivi prefissati.

3.3.3 Controllo della qualità

Ogni componente deve contribuire a rendere buona la qualità, uguagliandola a quella auspicata. Per farlo è necessario seguire questi punti:

- Gli obiettivi devono essere compresi, in caso di dubbi bisogna chiarirli con gli altri componenti;
- Trovare dei possibili errori, proponendo una soluzione;
- Riuscire a stimare la complessità di una task, organizzandosi per terminarla;
- Cercare di migliorare progressivamente la conoscenza dell'argomento;
- Rispettare le regole che vengono adotatte dal team;
- Produrre i risultati desiderati e concreti.

3.3.4 Aspettative

Le aspettative di questo processo sono:

- Conseguimento della qualità nel prodotto, secondo le richieste del *proponente*;
- Prova oggettiva della qualità del prodotto;
- Conseguimento della qualità nell'organizzazione delle attività del gruppo e dei processi;
- Raggiungimento della piena soddisfazione del *proponente*.

3.3.5 Metriche

Metrica	Descrizione	Riferimento
MPR18	Metriche di qualità soddisfatte	§6.2 MPR18
MPR11	Densità Errori	§6.2 MPR11
MPP7	Non-calculetd Risk	§6.1 MPP7

Tabella 11: Metriche utilizzate per la valutazione della qualità

3.3.6 Strumenti

- **LaTeX$_G$:** stesura in bella copia dei documenti caricati poi sul repository$_G$ del gruppo;
- **Visual Studio Code$_G$:** usato per la scrittura di codice e documenti, e visione del versioning tramite collegamento a GitHub$_G$;
- **Google Docs:** cartella condivisa contenente documenti in brutta copia, permette scrittura collaborativa contemporanea da parte di tutti i membri con visualizzazione delle modifiche in live.

3.4 Verifica

3.4.1 Scopo

Il processo di *verifica*$_{\text{G}}$ viene applicato per individuare eventuali errori introdotti nel prodotto durante la fase di sviluppo di un processo. La *verifica*$_{\text{G}}$ viene applicata sia alla documentazione che al codice.

3.4.2 Aspettative

Le aspettative di questo processo sono:

- Seguire procedure definite con criteri chiari ed affidabili;
- Verificare ad ogni fase;
- La verifica$_{\text{G}}$ deve garantire che il prodotto si trovi in uno stato stabile;
- La verifica$_{\text{G}}$ deve risultare più automatica possibile.

3.4.3 Verifica del codice

Il codice, derivante dai requisiti da soddisfare e dalla progettazione effetuata, verrà verificato tramite test automatici. Inoltre sono presenti all'interno del *Piano di Qualifica* le metriche e varie misurazioni.

3.4.4 Analisi

Il processo di analisi si suddivide in Analisi statica e Analisi dinamica:

3.4.4.1 Analisi statica

L'analisi statica permette di effettuare controlli su documenti e codice, verificando così l'assenza di errori e difetti. Esistono due metodologie per applicarla:

- **Walkthrough:** consiste nella lettura da parte del *team* dell'intero documento o codice in cerca di anomalie. Viene applicata quando non si conosce in modo chiaro la sorgente dei difetti. Questa tecnica risulta molto onerosa in termini di efficienza ed efficacia.
- **Inspection:** consiste in una lettura mirata del documento o del codice nei punti in cui si sa già che possano essere presenti degli errori. Risulta meno dispendiosa in termini di tempo ma richiede una buona conoscenza della situazione.

3.4.4.2 Analisi dinamica

L'analisi dinamica prevede l'esecuzione del prodotto *software* e la sua analisi tramite l'utilizzo di *test* che verificano se il prodotto funziona o se vi sono presenti anomalie

3.4.4.3 Test

L'attività di testing è la base dell'analisi dinamica. I *test* permettono di individuare tutti i possibili errori che possono essere stati commessi e tutti i casi limite che possono risultare problematici.
I test sono ben progettati e scritti se e solo se:

- Sono ripetibili;
- Sono automatici;
- Forniscono informazioni tramite artefatti di vario genere, quali file di log.

3.4.5 Metriche

Metrica	Descrizione	Riferimento
MPR3	Code Coverage	§6.2 MPR3
MPR4	Branch Coverage	§6.2 MPR4
MPR16	Percentuale test passati	§6.2 MPR16
MPR17	Percentuale test falliti	§6.2 MPR17
MPP1	Schedule$_G$ variance	§6.1 MPP1
MPP2	Budget variance	§6.1 MPP2
MPP3	Budgeted Cost of Work Performed	§6.1 MPP3
MPP4	Budgeted Cost of Work Scheduled	§6.1 MPP4

Tabella 12: Metriche utilizzate per la verifica

3.4.6 Strumenti

Per il processo di verifica applicato ai documenti scritti in LATEX il gruppo ha deciso di utilizzare il correttore automatico per i documenti LaTeXfornito dalle estensioni di VSC$_G$, che individua le parole non corrette grammaticalmente con una classica sottolineatura. Per il frontend scritto in javascript il gruppo usa jest per verificare il corretto funzionamento dell'interfaccia e pytest per il backend scritto in python.

3.5 Processo di validazione

3.5.1 Scopo

Il processo di *validazione*$_G$ prende in esame il prodotto ottenuto dalla fase di *verifica*$_G$ e stabilisce se esso rispetti i requisiti e le aspettative del *committente*.

3.5.2 Aspettative

Le aspettative di questo processo sono:

- Identificazione oggetti da validare;
- Valutazione dei risultati rispetto alle attese;
- Rendere tale processo automatico e riutilizzabile.

3.5.3 Metriche

Il processo di validazione non fa uso di metriche qualitative particolari

3.5.4 Strumenti

Non sono stati identificati degli strumenti particolari per la validazione.

4 Processi organizzativi

4.1 Gestione dei processi

4.1.1 Scopo

La presente sezione espone gli strumenti impiegati dal gruppo *Bug's Bunny* per quanto concerne le attività di comunicazione interna ed esterna, l'organizzazione e la gestione dei ruoli di ogni componente. A tale processo è dedicato il documento *Piano di Progetto*$_G$.

4.1.2 Aspettative

Le aspettative di questo processo sono:

- Definizione strumenti e modalità di comunicazione;
- Definizione strumenti, modalità e norme di organizzazione;
- Definizione norme per la gestione dei ruoli dei membri del gruppo.

4.1.3 Gestione degli incontri

4.1.3.1 Incontri interni Gli incontri interni si svolgono almeno una volta a settimana per gestire il lavoro settimanale del progetto e possono essere richiesti da un qualsiasi componente del gruppo. Sono un momento di confronto e risoluzione di eventuali problemi. Ad ogni riunione è auspicabile che partecipino tutti i componenti del gruppo.

4.1.3.2 Incontri esterni Per ogni incontro esterno il *Responsabile di Progetto* si occupa di comunicare con l'azienda, i committenti o altri enti esterni al gruppo di lavoro. Le vie utilizzate sono le due descritte nel paragrafo §4.2.2. Questo tipo di incontri avvengono saltuariamente a seconda delle necessità burocratiche o operative del progetto (quindi revisioni o necessità di supporto sulle tecnologie utilizzate). Ad ogni riunione partecipano tutti i membri del gruppo, salvo imprevisti inderogabili.

4.1.3.3 Verbali Per ogni incontro (interno o esterno) viene redatto un verbale da una persona scelta dal responsabile e successivamente verrà controllato dal verificatore e confermato dall'approvatore. Sia quest'ultimo che il verificatore, suggeriscono eventuali integrazioni o modifiche al redattore.

4.1.4 Ruoli di progetto

Il *Responsabile di Progetto* deve assicurarsi che ogni membro del *gruppo* ricopra almeno una volta ogni ruolo, che sono i seguenti:

- ***Responsabile di Progetto***: è la figura centrale del progetto, esso ha il compito di coordinare il *team* e il ruolo di rappresentante del progetto durante le comunicazioni con l'esterno;
- ***Amministratore***: ha il compito di gestire e configurare adeguatamente le piattaforme utilizzate dal gruppo per lo svolgimento del progetto. Da lui dipendono l'affidabilità e l'efficacia dei mezzi scelti per lo svolgimento del progetto;

- ***Analista***: segue il progetto principalmente nelle fasi iniziali ed è fortemente coinvolto nella stesura dell' *Analisi dei Requisiti*$_G$. Il suo ruolo è quello di analizzare i problemi posti dal progetto e chiarire le dipendenze e le ramificazioni di ogni attività necessaria alla consegna del prodotto;
- ***Progettista***: segue lo sviluppo del progetto e, a partire dai *requisiti*, definisce le scelte tecniche necessarie per lo sviluppo del prodotto;
- ***Programmatore***: ha il compito di codificare i modelli realizzati dal *Progettista*. Il codice prodotto dal *Programmatore* deve attenersi il più possibile alle specifiche elaborate dal *Progettista* e documentare opportunamente il codice creato per aumentarne la manutenibilità;
- ***Verificatore***: segue l'intero ciclo di vita del progetto. Egli si assicura che la qualità della documentazione prodotta aderisca alla norme stabilite.

4.1.5 Gestione dei rischi

Nel *Piano di Progetto* i rischi sono analizzati e suddivisi in 4 tipi:

- (RT): rischi tecnologici legati alla gestione e suddivisone del lavoro;
- (ROR): rischi organizzativi legati a comprensione ed uso delle tecnologie necessarie;
- (RI): rischi interpersonali legati specificatamente all'interazioe tra i membri del gruppo;
- (RO): rischi operativi legati alle possibilità di mettere in atto le pratiche di lavoro concordate.

Ogni rischio è seguito da un numero (esempio: RT1), il quale parte da 1 ed incrementa. Ogni volta che la tipologia cambia, il numero riparte da 1 e il ragionamento è analogo. Ogni rischio è strutturato nel seguente modo:

- Nome: nome del rischio per identificarlo;
- Descrizione: breve descrizione del rischio a cui si va incontro;
- Notifica: ogni membro è tenuto a notificare eventuali problemi;
- Occorrenza: viene distinta in Bassa, Media ed Elevata; quindi si classifica un rischio in base alla sua occorrenza;
- Pericolosità: viene sempre distinta in Bassa, Media ed Elevata; quindi si classifica un rischio in base alla sua pericolosità;
- Gestione: viene proposta una soluzione per risolvere il rischio.

4.1.5.1 Assegnazione dei compiti Per garantire l'efficienza e la flessibilità del processo di sviluppo, le attività necessarie al completamento di un progetto devono essere suddivise in compiti, che possono essere svolti sequenzialmente o in parallelo a seconda delle dipendenze tra di loro. Per ottimizzare tale processo il gruppo utilizza il sistema delle *Issues*$_G$ e delle *Pull Request*$_G$ offerto da *Github*$_G$. Una volta individuate dall'*Analista*, le attività vengono all'occorrenza suddivise in sotto attività e compiti. Il *Responsabile di Progetto* si occupa di assegnare ad ogni membro del team *Issue*$_\text{G}$ diverse per ottimizzare il progresso del progetto e nel farlo deve valutarne l'idoneità, tenendo conto di disponibilità, competenze tecniche e carico di lavoro attuale del membro in esame. A questo punto, il membro assegnato all'incarico dovrà svolgere il compito nei tempi definiti dalla *Milestone*$_G$ collegata alla *Issue*$_\text{G}$ e, nel caso di contrattempi, notificarlo quanto prima possibile al *Responsabile di Progetto*. Una volta che avrà concluso il proprio lavoro e fatto un commit (che riporterà il numero della *Issue*$_\text{G}$) con le relative modifiche, dovrà aprire una *Pull Request*$_G$ di verifica per il merge sul ramo *dev*. Un altro membro del gruppo effettuerà l'azione di verifica e controllo delle modifiche verificando la *Pull Request*$_G$. Se il verificatore accetta le modifiche effettuate aggiornerà il registro delle modifiche e la versione del relativo documento modificato e chiuderà la *Pull Request*$_\text{G}$. Verrà aperta in seguito, prima della versione di rilascio, una *Pull Request*$_G$ di approvazione per il merge sul ramo *master* che dovrà essere approvato da un altro membro del gruppo. Inoltre ogni membro associato a una *Issue*$_\text{G}$ o a una *Pull Request*$_G$ verrà notificato tramite email per ogni modifica relativa alla documentazione di cui si occupa la *Issue*$_\text{G}$.

4.1.6 Metriche

Il processo di gestione organizzativa non fa uso di metriche qualitative particolari.

4.1.7 Strumenti

Il gruppo *Bug's Bunny* utilizza vari strumenti per la comunicazione sia interna che esterna:

4.1.7.1 Comunicazione interna Le comunicazioni interne riguardano solamente i membri del gruppo *Bug's Bunny* attraverso l'utilizzo dei seguenti strumenti:

- **Telegram$_\text{G}$**: usato per pianificare incontri su *Discord*$_\text{G}$ e per comunicazioni rapide e non troppo importanti;
- **Discord$_\text{G}$**: usato per incontri di discussione e pianificazione sul lungo periodo; inoltre è stato integrato con *GitHub*$_G$ per la ricezione di messaggi riguardanti lo stato dei repository$_\text{G}$.

4.1.7.2 Comunicazioni esterne Le comunicazioni esterne riguardano le comunicazione tra i membri del gruppo *Bug's Bunny*e persone esterne al *gruppo*:

- **ProtonMail**: usato per la comunicazione scritta tramite email con l'azienda Zero12e i *committenti*;
- **Slack$_\text{G}$**: usato per la comunicazione con il rappresentante dell'azienda Zero12.

5 V-Model

5.1 Sintesi

Il modello a V_G è un sistema astratto atto a favorire lo sviluppo di software qualitativamente valido.
Struttura i vari step necessari secondo due rami, uno decrescente e uno crescente:
Nel primo si "scende" verso la fase di codifica decomponendo le necessità del cliente e degli stakeholders$_G$ in requisiti via via più specifici, per poi andare a progettarli e realizzarli.
Nel secondo ramo si "risale" dalla fase di codifica e si va a verificare e validare il lavoro fatto, concettualmente andando a ricomporre le parti precedentemente divise per ottenere il prodotto finale.
Il tutto si conclude con il collaudo, dove il cliente e gli stakeholders$_G$ controllano il software completo e si accertano che esso svolga i compiti previsti e voluti.

5.2 V&V

Verifica$_G$ e validazione$_G$ possono essere espressi con le seguenti due frasi: "did I build the system right?" e "did I build the right system?", ovvero la verifica$_G$ si occupa di controllare che ciò che è stato fatto segua le metriche$_G$ di qualità e rispetti le best practices$_G$ (di codifica, di documentazione, etc.), mentre la validazione$_G$ accerta che ciò che è stato prodotto sia conforme alle richieste e alle necessità degli stakeholders$_G$.

5.3 Due rami

Come precedentemente spiegato nella sintesi, il modello a V_G comprende due rami (in inlese chiamati anche "streams") che possiamo schematizzare come:

Specification Stream:

- Richieste utente;
- Analisi dei requisiti;
- Progettazione logica;
- Progettazione di dettaglio.

Codifica;

Testing Stream:

- Verifica progettazione di dettaglio;
- Verifica progettazione logica;
- Validazione analisi dei requisiti;
- Collaudo (validazione richieste utente).

5.4 Obiettivi

Il modello a V_G si propone come linea guida per le fasi di pianificazione e realizzazione di un progetto, nello specifico tenta di:

- **Minimizzare il rischio:** specificando un approccio standardizzato al lavoro migliora la trasparenza e rende più facile il controllo delle fasi di un progetto. Permette inoltre di riconoscere con anticipo le deviazioni dal piano e l'insorgere di rischi;
- **Miglioramento della qualità:** seguire con cura il modello a V_G permette di mantenere una qualità generale del prodotto più alta;
- **Riduzione dei costi:** l'impegno necessario nelle fasi di sviluppo, produzione, rilascio e manutenzione di un sistema può essere stimato con più precisione applicando un modello standardizzato;
- **Miglioramento della comunicazione:** una descrizione uniforme degli elementi rilevanti e dei termini contrattuali del progetto favoriscono una comunicazione migliore tra gli stakeholders$_G$.

6 Metriche

6.1 Metriche per qualità di processo

Alcuni parametri per comprendere le tabelle:

- ACWP : Actual Cost of Work Performed;
- NoC / NoD / NoA : Number of Changed / Added / Deleted *: Numero di requisiti cambiati / aggiunti / eliminati*;
- TNIR : Total Number of Initial Requirements.

Codice	Nome	Descrizione	Ottenimento
MPP1	Schedule_G variance	Variazione rispetto ai tempi pianificati	$\frac{100 \cdot (BCWP - BCWS)}{BCWS}$
MPP2	Budget variance	Variazione rispetto ai costi preventivati	$\frac{100 \cdot (BCWS - ACWP)}{BCWS}$
MPP3	Budgeted Cost of Work Performed	Valore effettivo del prodotto al calcolo dell'indice	$sum(\forall ruolo(oreRuolo \cdot costoOrario))$
MPP4	Budgeted Cost of Work Scheduled	Previsione costi	*Piano di Progetto*
MPP5	SPICE capability	Misura qualità processi	§5 di questo documento
MPP6	Requirements Stabilty Index	Indica variabilità dei requisiti nel tempo	$(1 - \frac{Noc+NoD+NoA}{TNIR}) \cdot 100$
MPP7	Non-calculated Risk	Numero di rischi non preventivati	Numero intero

Tabella 13: Tabella metriche per qualità di processo

6.2 Metriche per qualità di prodotto

Alcuni parametri per comprendere le tabelle:

- NdF / NdL / NdP : Numero di Frasi / Lettere / Parole;
- # : numero, inteso come "quantità", di una certa collezione di elementi;
- T_{pos} : numero tests eseguiti sul programma che rilevano errori;
- T_{neg} : numero tests eseguiti sul programma che non rilevano errori;
- T_{tot} : numero di tests eseguiti sul programma.

Codice	Nome	Descrizione	Ottenimento
MPR1	Indice di Gulpease	Indice di leggibilità del testo	$89 + \frac{300 \cdot NdF - 10 \cdot NdL}{NdP}$
MPR2	Percentuale requisiti obbligatori soddisfatti	Autoesplicativo	$100 \cdot \frac{\#\ requisiti\ soddisfatti}{\#\ requisiti\ totali}$
MPR3	Code coverage	% linee codice percorse dai tests	$100 \cdot \frac{linee\ codice\ percorse}{linee\ codice\ totali}$
MPR4	Branch coverage	% rami condizionali coperti dai tests	$100 \cdot \frac{rami\ condizionali\ percorsi}{rami\ condizionali\ totali}$
MPR5	Accoppiamento tra classi	Numero di dipendenze per classe	Numero Intero
MPR6	Profondità gerarchie	Rappresenta la quantità di super classi	Numero Intero
MPR7	Numero attributi per classe	Autoesplicativo	Numero Intero
MPR8	Numero parametri per metodo	Autoesplicativo	Numero Intero
MPR9	Linee codice per metodo	Autoesplicativo	Numero Intero
MPR10	Linee commento per codice	Autoesplicativo	$\frac{\#\ linee\ codice}{\#\ linee\ commento}$
MPR11	Densità errori	Percentuale che rappresenta la solidità del prodotto	$100 \cdot \frac{T_{\text{pos}}}{T_{\text{tot}}}$
MPR12	Facilità utilizzo	Numero di input necessari all'utente per ottenere il risultato voluto	Numero Intero
MPR13	Errori ortografici	Autoesplicativo	PdF checker e simili
MPR14	Complessità ciclomatica media	Numero cammini linearmente indipendenti nel programma	Grafo controllo di flusso
MPR15	Tempo medio risposta WebApp	Autoesplicativo	Misurato in secondi

Codice	Nome	Descrizione	Ottenimento
MPR16	Percentuale test passati	Autoesplicativo	$100 \cdot \frac{T_{\text{neg}}}{T_{\text{tot}}}$
MPR17	Percentuale test falliti	Autoesplicativo	MPR11
MPR18	Metriche di qualità soddisfatte	Percentuale di metriche che rientrano nei valori accettabili	$100 \cdot \frac{\#\, metriche\ soddisfatte}{\#\, metriche\ totali}$

Tabella 14: Tabella metriche per qualità di processo

7 Standard ISO/IEC 12207 - Software life cycle processes

7.1 Introduzione

È uno standard$_G$ per la gestione del ciclo di vita del software. Stabilisce un processo$_G$ di ciclo di vita del software, compreso processi ed attività relative alle specifiche ed alla configurazione di un sistema e ad ogni processo corrisponde un insieme di risultati. La struttura dello standard$_G$ è stata concepita per essere flessibile e modulare in modo che sia adattabile alle necessità di chiunque lo utilizzi.

7.1.1 Principi fondamentali

Lo standard$_G$ è basato su due principi fondamentali:

- **Modularità**: definire processi$_G$ con il minimo accoppiamento e la massima coesione;
- **Responsabilità**: stabilire un responsabile per ogni processo$_G$.

7.1.2 Tipi di processi

Esistono tre tipi di processi:

- Processi$_G$ primari;
- Processi$_G$ di supporto;
- Processi$_G$ di organizzazione.

8 Standard ISO/IEC 15504 - SPICE

8.1 Introduzione

Lo standard *SPICE* (Software Process Improvement and Capability Determination) è un insieme di standard tecnici per i processi di sviluppo software.

8.1.1 Classificazione processi

Nello standard *SPICE* i processi sono suddivisi in 5 categorie:

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

8.2 Livelli di capability e attributi di processo

La capability viene definita come la capacità di un processo di raggiungere il suo scopo. Per ogni processo, *SPICE* definisce un livello di capability, che sono i seguenti:

- **Livello 0 - Incomplete process**: processo$_G$ non implementato, incapace di raggiungere i suoi obiettivi;
- **Livello 1 - Performed process**: processo$_G$ implementato e in grado di raggiungere i suoi obiettivi, ma non è sottoposto a nessun tipo di controllo;
- **Livello 2 - Managed process**: processo$_G$ pianificato e sottoposto a controllo e correzione, gli obiettivi vengono raggiunti e sono tracciabili e verificati;
- **Livello 3 - Established process**: processo$_G$ definito da standard e quindi regolamentato;
- **Livello 4 - Predictable process**: processo$_G$ istanziato$_G$ entro limiti ben definiti, viene monitorato in modo dettagliato con lo scopo di renderlo prevedibile e ripetibile;
- **Livello 5 - Optimizing process**: processo$_G$ completamente definito e tracciato, soggetto ad analisi e miglioramento continui.

La capability di un processo è misurata tramite gli attributi di processo; lo standard$_G$ definisce i seguenti 9 attributi (riportati con la seguente codifica: [Livello di capability][Numero attributo]-[Nome attributo]):

- **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$_G$ al processo;
- **4.2 - Process control**: 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.

Ogni processo è valutato tramite la seguente scala di valori che esprimono numericamente il grado di soddisfacimento dell'attributo:

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

9 Standard ISO/IEC 25000 - SQuaRE: Systems and software Quality Requirements and Evaluation

9.1 Introduzione

È uno standard$_G$ che punta a creare un framework$_G$ per l'evoluzione della qualità del prodotto software ed è nato dall'evoluzione di due precedenti standard:

- **ISO/IEC 9126**: definisce un modello di qualità per la valutazione del prodotto software;
- **ISO/IEC 14598**: definisce il processo$_G$ per la valutazione del prodotto software.

9.2 Divisioni dello standard

Lo standard ISO/IEC 25000 è suddiviso in cinque divisioni:

- **ISO/IEC 2500n - Quality Management Division**: definisce modelli, termini e definizioni comuni per tutti gli standard$_G$ delle serie SQuaRE;
- **ISO/IEC 2501n - Quality Model Division**: presenta modelli di qualità dettagliati per sistemi informatici e prodotti software, qualità d'uso e dati;
- **ISO/IEC 2502n - Quality Measurement Division**: presenta un modello di riferimento per la misurazione della qualità del prodotto software, definizioni delle misure di qualità e linee guide pratiche per la loro applicazione;
- **ISO/IEC 2503n - Quality Requirements Division**: aiuta a specificare i requisiti di qualità da seguire e soddisfare;
- **ISO/IEC 2504n - Quality Evaluation Division**: presenta requisti, raccomandazioni e linee guida per la valutazione dei prodotti software.

9.3 Qualità perseguite dallo standard

Di seguito vengono riportate le qualità perseguite dallo standard:

- **Qualità di prodotto**:
  - Adeguatezza Funzionale;
  - Efficienza Prestazionale;
  - Compatibilità;
  - Usabilità;
  - Affidabilità;
  - Sicurezza;
  - Manutenibilità;
  - Portabilità.
- **Qualità in uso**:

  - Efficacia;
  - Efficienza;
  - Soddisfazione;
  - Mitigazione dei rischi;
  - Copertura.

9.4 Processo di valutazione

Lo standard prevede un processo di valutazione da seguire per valutare la qualità del software; tale processo si compone dei seguenti cinque passi:

- Stabilire i requisiti di valutazione;
- Specificare la valutazione;
- Progettare la valutazione;
- Eseguire la valutazione;
- Concludere la valutazione.