CODEBUSTERS

Progetto: ***HD Viz***

codebusterswe@gmail.com

Norme di Progetto

Informazioni sul documento	
Versione	2.0.0-0.2
Approvatori	Zenere Marco
Redattori	Pirolo Alessandro Zenere Marco Rago Alessandro Safdari Hossain
Verificatori	Sassaro Giacomo Scialpi Paolo Baldisseri Michele
Uso	Interno
Distribuzione	*Zucchetti* Prof. Vardanega Tullio Prof. Cardin Riccardo Gruppo *CodeBusters*

Descrizione

Questo documento racchiude le regole, gli strumenti e le convenzioni adottate dal gruppo nello svolgimento del progetto *HD Viz*.

Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
2.0.0-0.2	28-02-2021	Zenere Marco	Responsabile	Approvazione del documento
1.2.0-0.1	25-02-2021	Sassaro Giacomo	Verificatore	Revisione complessiva del documento
1.1.2-0.1	22-02-2021	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Aggiunti strumenti in §2.2.6, in §3.4.7 e verifica
1.1.1-0.1	18-02-2021	Rago Alessandro, Baldisseri Michele	Amministratore, Verificatore	Aggiunte regole a §2.2.4.3.3, aggiunta §3.2.5.4 e verifica
1.1.0-0.0	10-02-2021	Sassaro Giacomo	Verificatore	Revisione complessiva del documento
1.0.5-0.0	09-02-2021	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Modificate §3.4.5.4, §4.1.5.4.1, §3.1.7.2, §3.1.6.5 e verifica
1.0.4-0.0	08-02-2021	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Revisione sistema di versionamento §3.2.4.1
1.0.3-0.0	06-02-2021	Safdari Hossain, Scialpi Paolo	Amministratore, Verificatore	Aggiunte figure 1 e 2 e verifica
1.0.2-0.0	05-02-2021	Pirolo Alessandro, Baldisseri Michele	Amministratore, Verificatore	Aggiunte §3.4.7, §3.4.6, §3.5.4, §3.5.5, §4.2.5 e verifica
1.0.1-0.0	04-02-2021	Pirolo Alessandro, Baldisseri Michele	Amministratore, Verificatore	Aggiunte §4.2, §2.6.1, §2.2.6, §3.2.6, §3.2.7, §3.3.5, §3.3.6 e verifica
1.0.0	04-01-2021	Zenere Marco	Responsabile	Approvazione del documento
0.4.0	30-12-2020	Sassaro Giacomo	Verificatore	Verifica del documento
0.3.2	29-12-2020	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Sistemate §A e §B e verifica
0.3.1	28-12-2020	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Aggiunte metriche e verifica
0.3.0	28-12-2020	Scialpi Paolo	Verificatore	Revisione complessiva del documento
0.2.4	27-12-2020	Pirolo Alessandro, Sassaro Giacomo	Amministratore, Verificatore	Stesura §B e verifica

Versione	Data	Nominativo	Ruolo	Descrizione
0.2.3	23-12-2020	Pirolo Alessandro, Baldisseri Michele	Amministratore, Verificatore	Stesura §A e verifica
0.2.2	22-12-2020	Safdari Hossain, Baldisseri Michele	Amministratore, Verificatore	Stesura §4.1.5 e §4.1.6 e verifica
0.2.1	22-12-2020	Pirolo Alessandro, Sassaro Giacomo	Amministratore, Verificatore	Aggiunta normativa in materia di elenchi puntati e verifica
0.2.0	22-12-2020	Sassaro Giacomo	Verificatore	Revisione complessiva del documento
0.1.7	21-12-2020	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Stesura §2.3 e verifica
0.1.6	21-12-2020	Rago Alessandro, Baldisseri Michele	Amministratore, Verificatore	Stesura §3.5 e verifica
0.1.5	21-12-2020	Pirolo Alessandro, Baldisseri Michele	Amministratore, Verificatore	Stesura §3.1.7.5 e §3.1.7.6 e verifica
0.1.4	20-12-2020	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Stesura §3.1.9 e §3.1.10 e verifica
0.1.3	20-12-2020	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Stesura §3.1.8 e verifica
0.1.2	20-12-2020	Rago Alessandro, Baldisseri Michele	Amministratore, Verificatore	Stesura §3.4 e verifica
0.1.1	20-12-2020	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Stesura §3.1.7 e verifica
0.1.0	19-12-2020	Sassaro Giacomo	Verificatore	Revisione complessiva del documento
0.0.7	19-12-2020	Rago Alessandro, Scialpi Paolo	Amministratore, Verificatore	Stesura §3.2 e §3.3 e verifica
0.0.6	19-12-2020	Zenere Marco, Baldisseri Michele	Amministratore, Verificatore	Stesura §2.1 e §2.2 e verifica
0.0.5	19-12-2020	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Stesura §3.1.6.4 e §3.1.6.5 e verifica
0.0.4	18-12-2020	Safdari Hossain, Scialpi Paolo	Amministratore, Verificatore	Stesura §4 fino a §4.1.4 e verifica
0.0.3	17-12-2020	Pirolo Alessandro, Baldisseri Michele	Amministratore, Verificatore	Iniziata stesura §3 fino a §3.1.6.3 e verifica

Versione	Data	Nominativo	Ruolo	Descrizione
0.0.2	15-12-2020	Pirolo Alessandro, Scialpi Paolo	Amministratore, Verificatore	Stesura §1 e verifica
0.0.1	14-12-2020	Zenere Marco, Scialpi Paolo	Amministratore, Verificatore	Creazione scheletro documento, stesura introduzione e verifica

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento vuole definire le linee guida di tutti i processi istanziati dal gruppo *CodeBusters*. Sono presenti l'organizzazione e l'uso di tutte le risorse di sviluppo, oltre alle convenzioni che il gruppo decide di attuare sull'uso delle tecnologie, sullo stile di codifica e di scrittura. Ogni membro è obbligato a tenere in considerazione questo documento per garantire maggiore uniformità e coerenza del materiale prodotto.

1.2 Scopo del capitolato

Oggigiorno, anche i programmi più tradizionali gestiscono e memorizzano una grande mole di dati; di conseguenza servono software in grado di eseguire un'analisi e un'interpretazione delle informazioni.
Il capitolato[G] C4 ha come obiettivo quello di creare un'applicazione di visualizzazione di dati con numerose dimensioni in modo da renderle comprensibili all'occhio umano. Lo scopo del prodotto sarà quello di fornire all'utente diversi tipi di visualizzazioni e di algoritmi per la riduzione dimensionale in modo che, attraverso un processo esplorativo, l'utilizzatore del prodotto possa studiare tali dati ed evidenziarne degli eventuali cluster[G].

1.3 Glossario

Per evitare ambiguità relative alle terminologie utilizzate, è stato compilato il *Glossario v2.0.0-0.2*. In questo documento sono riportati tutti i termini di particolare importanza e con un significato particolare. Questi termini sono evidenziati da una 'G' ad apice.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Capitolato d'appalto C4 - HD Viz: visualizzazione di dati multidimensionali**: https://www.math.unipd.it/~tullio/IS-1/2020/Progetto/C4.pdf.

1.4.2 Riferimenti informativi

- **Standard ISO/IEC 12207:1995**: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  - Capitolo 5 - Primary life cycle processes (da pag. 10 a 24);
  - Capitolo 6 - Supporting life cycle processes (da pag. 28 a 41);
  - Capitolo 7 - Organizational life cycle processes (da pag. 42 a 47).
- **ISO/IEC 9126**: http://www.colonese.it/00-Manuali_Pubblicatii/07-ISO-IEC9126_v2.pdf
  - Capitolo 2 - Il modello ISO/IEC 9126 (da pag. 12 a 24);
  - Capitolo 4 - Utilizzo del modello:
    - Paragrafo 4.1 - Processo di sviluppo e qualità (da pag. 31 a 34);

    - Paragrafo 4.2 - Implementazione del modello in progetti reali (da pag. 34 a 37);
    - Paragrafo 4.3 - Esempi di metriche (pag. 37).

- **Metriche**:
  https://it.wikipedia.org/wiki/Metriche_di_progetto
- **Slide T3 del corso Ingegneria del Software - Processi di ciclo di vita**:
  https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L03.pdf
- **Slide FC1 del corso Ingegneria del Software - Amministrazione di progetto**:
  https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/FC1.pdf
- **Software Engineering - Ian Sommerville - 10 th Edition**:
  Parte 4 - Software management:
  - Capitolo 25 - Configuration management:
    - Paragrafo 25.1 - Version management (da pag. 735 a 740);
    - Paragrafo 25.2 - System building (da pag. 740 a 745).
- **Guida Latex**:
  http://www.lorenzopantieri.net/LaTeX_files/LaTeXimpaziente.pdf.

2 Processi Primari

2.1 Fornitura

2.1.1 Scopo

In questa sezione vengono descritti i documenti che compongono il processo di fornitura, la loro struttura e gli strumenti utilizzati.

2.1.2 Aspettative

Le aspettative nell'applicazione del processo di fornitura sono:

- Avere una chiara struttura dei documenti;
- Definire i tempi di lavoro;
- Chiarire dubbi e stabilire vincoli con il proponente.

2.1.3 Descrizione

Nel processo di fornitura si scelgono le procedure e le risorse atte a perseguire lo sviluppo del progetto. Dopo aver ricevuto le richieste del proponente, il gruppo redige uno studio di fattibilità e la fornitura può essere avviata per completare tali richieste.
Il proponente e il fornitore stipuleranno un contratto per la consegna del prodotto.
Si dovrà poi sviluppare un piano di progetto partendo dalla determinazione delle procedure e delle risorse necessarie. Da quel momento fino alla consegna del prodotto il *Piano di Progetto* scaglionerà le attività da svolgere.
Il processo di fornitura è composto dalle seguenti fasi:

1. Avvio;
2. Approntamento di risposte alle richieste;
3. Contrattazione;
4. Pianificazione;
5. Esecuzione e controllo;
6. Revisione e valutazione;
7. Consegna e completamento.

2.1.4 Proponente

CodeBusters vorrebbe avere un contatto costante con il proponente in modo da avere un riscontro:

- Sulle soluzioni utilizzate;
- Sulle tempistiche di consegna del prodotto;
- Su eventuali dubbi;

- Sulla stima dei costi;
- Su vincoli e requisiti.

2.1.5 Documenti

Di seguito sono descritti i documenti che sono redatti durante questa fase.

2.1.5.1 Studio di fattibilità

Documento che contiene la stesura dello studio di fattibilità riguardante i sette capitolati proposti; per ciascuno di essi vengono evidenziati i seguenti aspetti:

- Descrizione generale;
- Prerequisiti e tecnologie coinvolte;
- Vincoli;
- Aspetti positivi;
- Aspetti critici.

Infine, per ogni capitolato vengono esposte le motivazioni e le ragioni per cui il gruppo ha scelto come progetto il capitolato[G] *HD Viz* a discapito degli altri sei proposti.

2.1.5.2 Piano di qualifica

Il *Piano di Qualifica*, redatto dal verificatore[G], contiene tutte le misure da adottare per garantire la qualità del prodotto e dei processi. È suddiviso nelle seguenti parti:

- Qualità di processo;
- Qualità di prodotto;
- Specifica dei test;
- Resoconto attività di verifica.

2.1.5.3 Piano di progetto

Gli amministratori e il responsabile dovranno redigere questo documento che dovrà essere seguito durante tutto il corso del progetto. È suddiviso nelle seguenti sezioni:

- Analisi dei rischi;
- Modello di sviluppo;
- Pianificazione;
- Preventivo;

- Consuntivo;
- Organigramma;
- Attualizzazione dei rischi.

2.1.6 Metriche

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

2.1.7 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **Excel**: utilizzato per creare grafici, eseguire calcoli e presentare tabelle organizzative;

  https://www.microsoft.com/it-it/microsoft-365/excel

- **Microsoft Planner**: utilizzato per gestire le task[G] che ciascun membro del gruppo deve svolgere. Permette di assegnare attività a specifici membri, suddividerle per categorie ed applicare molte altre personalizzazioni. Grazie a questa applicazione è possibile verificare che tutte le attività siano completate in linea con i tempi previsti;

  https://www.microsoft.com/it-it/microsoft-365/business/task-management-software

- **GanttProject**: utilizzato per creare i grafici di Gantt relativi alla pianificazione.

  https://www.ganttproject.biz/

2.2 Sviluppo

2.2.1 Scopo

In questa sezione vengono descritte le attività che compongono il processo di sviluppo, le norme e le convenzioni adottate per ognuna di esse.

2.2.2 Aspettative

Le aspettative nell'applicazione del processo di sviluppo sono:

- Determinare vincoli tecnologici;
- Determinare gli obiettivi di sviluppo;
- Determinare vincoli di design;
- Realizzare un prodotto finale che superi i test e soddisfi requisiti e richieste del proponente.

2.2.3 Descrizione

Il processo di sviluppo contiene le attività e i compiti dello sviluppatore, tra cui le attività per l'analisi dei requisiti, la progettazione, la codifica, l'integrazione, il test, l'installazione e l'accettazione relative ai prodotti software.

2.2.4 Attività

Le attività del processo di sviluppo sono tre:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.4.1 Analisi dei requisiti

2.2.4.1.1 Scopo

Lo scopo dell'analisi dei requisiti è individuare e poi riportare in un documento tutti i requisiti individuati e i casi d'uso del sistema.

2.2.4.1.2 Aspettative

L'obiettivo dell'attività di analisi dei requisiti consiste nella creazione della documentazione formale contenente tutti i requisiti richiesti dal proponente.

2.2.4.1.3 Requisiti

I requisiti si raccolgono tramite:

- Lettura del capitolato;
- Visione della presentazione del capitolato;

- Confronto interno tra i membri del gruppo;
- Confronto con l'azienda.

2.2.4.1.4 Casi d'uso

È un diagramma che esprime un comportamento o un modo di utilizzare il prodotto. È costituito da:

- Codice identificativo e titolo;
- Attore primario;
- Precondizioni;
- Postcondizioni;
- Scenario principale;
- Generalizzazioni;
- Estensioni.

2.2.4.1.5 Codice identificativo casi d'uso

Un caso d'uso è così identificato:

UC[Numero caso d'uso].[Caso d'uso figlio] - [Titolo caso d'uso]

dove "caso d'uso figlio" è il sottocaso del caso d'uso principale.

2.2.4.1.6 Struttura dei requisiti

- **Codice identificativo:**

  R[Importanza][Tipologia][Codice]

  Per **Importanza** s'intende un numero da 1 a 3 che rappresenta:

  1. Requisito obbligatorio;
  2. Requisito desiderabile ma non essenziale per il funzionamento;
  3. Requisito opzionale.

  Per **Tipologia** s'intende una lettera che rappresenta la natura del requisito:

  - **V**: Vincolo;
  - **P**: Prestazionale;
  - **Q**: Qualitativo;
  - **F**: Funzionale.

  Per **Codice** s'intende un identificativo univoco del requisito espresso in forma gerarchica padre/figlio.

- **Classe:** per rendere la tabella più esplicativa viene riportata nuovamente l'importanza del requisito, nonostante sia già scritta nel codice identificativo;
- **Descrizione:** una sintetica descrizione del requisito;
- **Fonti:** vi sono diverse fonti da cui possono derivare i requisiti; l'origine dei requisiti viene quindi riportata in questa sezione.

2.2.4.1.7 UML

I diagrammi UML[G] devono essere realizzati usando la versione del linguaggio v2.0.

2.2.4.2 Progettazione

2.2.4.2.1 Scopo

Questa attività ha la funzione di definire una soluzione al capitolato proposto basandosi sull'analisi dei requisiti. Mentre quest'ultima divide il problema nei requisiti da soddisfare, la progettazione incorpora le parti specificando le funzionalità dei sottosistemi e riconducendo ad un'unica soluzione.

2.2.4.2.2 Aspettative

Riuscire ad arrivare, al termine di questa attività, ad una architettura di sistema.

2.2.4.2.3 Tecnology Baseline

Motiva le tecnologie, i framework[G] e le librerie selezionate per la realizzazione del prodotto. Sarà il progettista ad occuparsene e dovrà contenere:

- Diagrammi UML delle classi, di attività, di sequenza e dei package;
- Tecnologie adottate, motivando le scelte;
- Design pattern[G] accompagnato da una descrizione e un diagramma che ne esponga la struttura;
- La relazione tra ciascuna componente e il requisito che soddisfa, per avere un tracciamento;
- Proof of Concept[G], ovvero un prototipo per dimostrare le funzionalità del prodotto.

2.2.4.2.4 Product Baseline

Illustra la baseline architetturale (design e coding) del prodotto, coerente con la Tecnology Baseline. Il progettista dovrà occuparsi anche di questa parte, che conterrà:

- Una definizione delle classi, evitando nomi e funzionalità ridondanti;
- Tracciamento delle classi, in modo che ciascun requisito sia soddisfatto da una classe;
- Test di unità su ogni componente, in modo da verificare il corretto funzionamento.

2.2.4.3 Codifica

2.2.4.3.1 Scopo

L'attività di codifica ha il fine di concretizzare la progettazione con la programmazione del software vero e proprio.

2.2.4.3.2 Aspettative

Questa attività dovrà avere come risultato 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.

2.2.4.3.3 Stile della codifica

- **Indentazione**: i blocchi di codice innestati dovranno avere un'indentazione di quattro spazi;
- **Parentesi**: la parentesi aperta dovrà essere inserita nella stessa riga di dichiarazione del costrutto, separate da uno spazio;
- **Metodi**: il nome dei metodi dovrà iniziare con lettera minuscola e, se composto da più parole, le successive dovranno iniziare con lettera maiuscola. È preferibile mantenere metodi brevi, con poche righe di codice;
- **Classi**: il nome delle classi dovrà sempre iniziare con la lettera maiuscola e, come per i metodi, se composto da più parole, le successive dovranno iniziare con la lettere maiuscola;
- **Variabili**: il nome delle variabili deve sempre essere scritto in minuscolo e in inglese. Se il nome è composta da più parole, la seconda dovrà iniziare con la lettera maiuscola;
- **Costanti**: il nome deve essere sempre scritto in maiuscolo e in inglese. Se il nome è composto da più parole, queste dovranno essere separate dal carattere "_";
- **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;
- **File**: dovranno avere un nome che inizia per lettera maiuscola che ne specifichi il contenuto;
- **Altro**:
  - Utilizzare la method chaining[G] per l'invocazione dei metodi. Se la catena di metodi è formata da più di due metodi, scrivere un metodo per riga.

2.2.4.3.4 Ricorsione

Onde evitare un'eccessiva allocazione di memoria è preferibile, quando possibile, evitare la ricorsione.

2.2.5 Metriche

2.2.5.1 Metriche per la portabilità

Alcuni parametri per capire meglio la tabella seguente:

- **Browser[G] supportati (B_{sup})**: browser dove il software può essere eseguito;
- **Browser[G] richiesti (B_{ric})**: numero di browser compatibili con il programma richiesti dal proponente.

Codice	Nome	Descrizione	Formula
MPD10	Versioni del browser supportate	La percentuale di versioni di browser supportate dal prodotto.	$(\frac{B_{sup}}{B_{ric}}) \cdot 100$

Tabella 1: Metriche per garantire la portabilità del prodotto

2.2.5.2 Metriche per la funzionalità

Alcuni parametri per capire meglio la tabella seguente:

- **Funzionalità mancanti (N_{fm})**: numero di funzionalità non implementate;
- **Funzionalità individuate (N_{fi})**: numero di funzionalità individuate.

Codice	Nome	Descrizione	Formula
MPD3	Copertura dei requisiti (CDR)	Descrive quanti requisiti sono stati implementati nel prodotto software.	$(1 - \frac{N_{fm}}{N_{fi}}) \cdot 100$

Tabella 2: Metriche per garantire che i requisiti siano rispettati

2.2.5.3 Metriche per la manutenibilità

Alcuni parametri per comprendere la tabella seguente:

- **Linee di commento (N_{com})**: numero di righe di commento;
- **Linee di commento (N_{cod})**: numero di righe di codice.

Codice	Nome	Descrizione	Formula
MPD5	Average Cyclomatic complexity (ACC)	Indica in numero di cammini indipendenti presenti nel programma.	Misurabile attraverso il grafo di controllo di flusso.
MPD9	Comprensione del codice	Può essere dedotta dal rapporto tra linee di codice e linee di commenti.	$(\frac{N_{com}}{N_{cod}}) \cdot 100$

Tabella 3: Metriche per garantire manutenibilità del prodotto

2.2.5.4 Metrica per l'efficienza

Codice	Nome	Descrizione	Formula
MPD8	Tempo medio di risposta	Tempo medio impiegato dal software per rispondere a una richiesta utente o svolgere un'attività di sistema.	-

Tabella 4: Metrica per garantire efficienza del prodotto

2.2.5.5 Metriche per l'usabilità

Codice	Nome	Descrizione	Formula
MPD6	Facilità di utilizzo	Numero di click necessari con cui l'utente raggiunge la funzionalità cercata.	-
MPD7	Facilità apprendimento funzionalità	Numero di minuti necessari all'utente per apprendere le funzionalità del prodotto software.	-

Tabella 5: Metriche per garantire usabilità del prodotto

2.2.5.6 Metrica per l'affidabilità

Alcuni parametri per comprendere la tabella seguente:

- **Test falliti ($\mathbf{T_{fal}}$)**: test eseguiti sul programma ma falliti;
- **Test eseguiti ($\mathbf{T_{ese}}$)**: test totali eseguiti sul programma.

Codice	Nome	Descrizione	Formula
MPD4	Densità di failure	Indica l'affidabilità del prodotto e si può ricavare dalla percentuale di test falliti sui test eseguiti.	$(\frac{T_{fal}}{T_{ese}}) \cdot 100$

Tabella 6: Metrica per garantire affidabilità del prodotto

2.2.6 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **DruidJS**: libreria per la riduzione dimensionale sui dati caricati nella web app[G]. Essa fornisce diversi algoritmi a questo scopo, tra cui i più interessanti e utilizzati in ambienti reali di analisi dei dati;

  https://github.com/saehm/DruidJS

- **D3.js**: libreria per la manipolazione del DOM (Document Object Model). Permette la realizzazione di molti tipi diversi di grafici da dati reali;

  https://d3js.org/

- **Node.js**: runtime JavaScript[G] utilizzato per creare applicazioni di rete scalabili;

  https://nodejs.org/it/

- **Express.js**: framework[G] standard per le applicazioni che utilizzano Node.js[G] per la comunicazione tra front-end[G] e back-end[G];

  https://expressjs.com/it/

- **PostgreSQL**: database[G] scelto per la sua efficienza e affidabilità nel lungo periodo;

  https://www.postgresql.org/

- **Babel**: transcompiler JavaScript per compilare tutte le nuove istruzioni introdotte con ES6 in istruzioni retrocompatibili;

  https://babeljs.io/

- **Npm**: packet manager per il linguaggio JavaScript[G];

  https://www.npmjs.com/

- **Bootstrap**: framework[G] CSS[G] che velocizza la scrittura della parte di presentazione della web app[G] con classi predefinite per elementi HTML[G] specifici;

  https://getbootstrap.com/

- **React**: libreria JavaScript[G] per facilitare la creazione di interfacce utente, migliorare i tempi di renderizzazione della pagina, aumentare la manutenibilità e la facilità di testing;

  https://it.reactjs.org/

- **Visual Studio Code**: IDE versatile ed estendibile scelto per lo sviluppo del codice.

  https://code.visualstudio.com/

3 Processi di Supporto

3.1 Documentazione

3.1.1 Scopo

Tutti i processi e le attività di sviluppo devono essere documentate. Questa sezione ha lo scopo di definire le norme, le convenzioni e la struttura organizzativa riguardanti la documentazione, oltre che la definizione degli strumenti necessari alla sua stesura.

3.1.2 Aspettative

Le aspettative di questo processo sono:

- Avere una chiara struttura per i documenti, in modo da ottenere un risultato uniforme alla fine del suo ciclo di vita;
- Avere norme e convenzioni ben precise che coprono tutti gli aspetti della stesura di un documento, in modo che tutti i membri di *CodeBusters* possano lavorare senza dover interpellare il gruppo per prendere decisioni riguardo un generico aspetto.

3.1.3 Descrizione

La documentazione è un processo per registrare le informazioni prodotte da una attività del ciclo di vita. Il processo contiene una serie di attività che pianificano, progettano, sviluppano, producono, modificano, distribuiscono e mantengono quei documenti necessari a tutti gli interessati, come manager, ingegneri e utenti.

3.1.4 Ciclo di vita del documento

Figura 1: Fasi del ciclo di vita di un documento

3.1.5 Template

Il gruppo ha deciso di creare un template con l'utilizzo di LaTeX, grazie al quale viene standardizzata la struttura dei documenti. In questo modo i componenti del gruppo si occupano unicamente di redigere il contenuto dei singoli testi velocizzando la stesura degli stessi. Più precisamente, nel template vengono definite la prima pagina, la struttura del registro delle modifiche e l'indicizzazione delle sezioni e sottosezioni.

3.1.6 Struttura del documento

Ogni documento è formato da diverse sezioni, ognuna definita dal proprio file LaTeX. La parte principale è chiamata "*nomedoc*.tex" (dove *nomedoc* indica il nome del documento) ed ha il compito di includere le seguenti componenti:

- I file LaTeX delle sezioni, che contengono il contenuto del testo vero e proprio. Se presenti numerose sottosezioni in una sezione, quest'ultima deve includere i file di tutte le varie sottosezioni;
- Il registro delle modifiche, che contiene una lista delle modifiche effettuate al documento così da renderle tracciabili;
- "String.tex", che contiene una serie di comandi LaTeX personalizzati che facilitano la scrittura di parole frequentemente utilizzate;
- "Comandi.tex", che contiene una serie di comandi LaTeX personalizzati, diversi per ogni documento.

3.1.6.1 Prima pagina

La prima pagina di un documento è formata da:

- **Logo**: logo di *CodeBusters* posto in alto e centralizzato;
- **Progetto ed e-mail**: sotto il logo e centralizzato viene scritto il nome del progetto e la mail del gruppo *CodeBusters*;
- **Titolo**: il nome del documento;
- **Informazioni sul documento**: sotto al titolo è presente una tabella con le seguenti informazioni riguardanti il documento:
  - **Versione**: versione del documento;
  - **Approvatori**: nomi dei componenti del gruppo che svolgono il ruolo di approvatore[G];
  - **Redattori**: nomi dei componenti del gruppo che svolgono il ruolo di redattore[G];
  - **Uso**: specifica il tipo di utilizzo che viene fatto di questo documento;
  - **Distribuzione**: specifica a chi il documento verrà distribuito.
- **Descrizione**: una breve descrizione del documento posta sotto la tabella.

3.1.6.2 Registro delle modifiche

Il registro delle modifiche è una tabella che riporta ogni modifica effettuata al documento in cui è contenuta. Una modifica è rappresentata da una riga della tabella avente le seguenti voci:

- **Versione**: versione attuale del documento;
- **Data**: data della modifica;
- **Nominativo**: il nome del redattore[G] della modifica;
- **Ruolo**: il ruolo che il redattore[G] ha all'interno del gruppo;
- **Descrizione**: una breve descrizione sulla modifica effettuata.

3.1.6.3 Indice

L'indice deve elencare le sezioni in cui sono divise le diverse parti del documento, le tabelle e le immagini presenti nel documento. L'indice viene posto dopo il registro delle modifiche.

3.1.6.4 Struttura delle pagine

La singola pagina di contenuto è strutturata come segue:

- in alto a sinistra viene posto il logo del gruppo *CodeBusters*;
- in basso a sinistra viene scritto il nome del documento;
- in basso a destra viene indicato il numero della pagina corrente sul totale delle pagine;
- il contenuto della pagina è scritto tra l'intestazione e il piè di pagina che lo delimitano con una riga.

3.1.6.5 Verbali

I *Verbali* prevedono una singola stesura, dato che contengono delle decisioni che non possono essere modificate successivamente. I *Verbali* contengono la prima pagina e il registro delle modifiche come gli altri documenti; viene invece omesso l'indice che il gruppo ha ritenuto poco utile data la brevità del documento.
Il contenuto di un *Verbale* è formato da tre sezioni. La prima è l'introduzione che contiene le seguenti informazioni:

- **Motivo della riunione**: il motivo per cui il gruppo ha deciso di organizzare un incontro e che, di conseguenza, contiene le materie che verranno discusse;
- **Luogo riunione**: il luogo dove viene svolta la riunione;
- **Data**: la data che indica quando il gruppo ha effettuato la riunione;
- **Orario di inizio**;
- **Orario di fine**;
- **Partecipanti**: l'elenco dei partecipanti al meeting.

La seconda sezione è il resoconto ed ha la funzione di fornire un breve riassunto di quanto discusso e delle decisioni prese. I motivi di discussione vengono riportati in un elenco dove vengono spiegati uno per volta.
Alla fine di ogni *Verbale* è presente una tabella che ha la funzione di tenere traccia delle decisioni prese durante l'incontro. Ogni riga della tabella prevede una descrizione molto breve della decisione e un identificativo che segue questo formato:

[Destinazione]-X.Y

Dove:

- **[Destinazione]**: è **Interno**, se il verbale è interno, mentre è **Esterno**, se il verbale è esterno;
- **X.Y**: dove **X** è il numero del verbale e **Y** indica il numero della decisione all'interno del verbale.

3.1.7 Convenzioni

3.1.7.1 Nomi dei file

I nomi dei file e delle cartelle seguono tutte la stessa convenzione:

- I nomi dei file e delle cartelle iniziano per lettera maiuscola;
- Se il nome è composto da più parole, ognuna di queste inizia per lettera maiuscola e non è previsto alcun tipo di separazione tra una parola e l'altra.

Esempi corretti:

- NormeDiProgetto;
- AnalisiDeiRequisiti.

Esempi non corretti:

- Normediprogetto (non tutti le parole iniziano con lettera maiuscola);
- analisi_dei_requisiti (sono presenti dei caratteri separatori).

3.1.7.2 Stile di testo

- **Grassetto**: lo stile grassetto viene applicato ai termini negli elenchi puntati e ai titoli delle sezioni;
- **Corsivo**: lo stile corsivo si applica al nome del gruppo, al nome del progetto, al nome dei documenti, al nome del proponente e alle parole che hanno una particolare importanza nel contesto in cui sono inserite;
- **Nome dei documenti**: il nome dei documenti deve essere scritto in corsivo e deve iniziare per lettera maiuscola, mentre quando si fa riferimenti al documento vero e proprio viene aggiunta anche la versione, anch'essa in corsivo. Se il nome è composto da più parole, ogni parola inizia per lettera maiuscola, ad esclusione delle preposizioni. Se il nome viene usato come titolo, questi stili non si applicano, ma il redattore deve utilizzare lo stile grassetto.
- **Link**: il testo che indica un link esterno deve essere di colore blu e sottolineato.
- **Collegamenti interni**: le parole che si riferiscono ad una parte del documento stesso vanno sottolineate.

3.1.7.3 Glossario

Le norme relative al *Glossario* sono:

- Ogni parola del *Glossario* è contrassegnata da una 'G' ad apice;
- Non vengono segnate come termini del *Glossario* le parole che fanno parte di un titolo o che sono presenti nelle didascalie;
- Se un termine compare nella propria definizione all'interno del *Glossario*, essa non viene contrassegnata.

3.1.7.4 Elenchi puntati e numerati

Gli elenchi puntati e numerati seguono le seguenti norme:

- Ogni voce inizia per lettera maiuscola;
- Ogni voce termina con ';', escludendo l'ultima che termina con '.';
- Se una voce deve descrivere un concetto, un termine o un oggetto allora esso va scritto in grassetto seguito da ':'.

I componenti del gruppo devono preferire l'utilizzo degli elenchi puntati, ma se sussiste la necessità di fornire un ordine preciso all'elenco da redigere, allora possono utilizzare un elenco numerato.

3.1.7.5 Sigle

Le sigle presenti nei documenti rappresentano i ruoli dei componenti:

- **RE**: responsabile di progetto;
- **AM**: amministratore;
- **AN**: analista;
- **PT**: progettista;
- **PR**: programmatore;
- **VE**: verificatore.

Nel documento *Analisi dei Requisiti v2.0.0-0.2* vengono usate le seguenti sigle:

- **OB**: obbligatorio;
- **DE**: desiderabile;
- **OP**: opzionale.

Altri sigle che vengono utilizzate sono:

- **RR**: revisione dei requisiti;
- **RP**: revisione di progettazione;
- **RQ**: revisione di qualifica;
- **RA**: revisione di accettazione.

3.1.7.6 Data

I componenti del gruppo hanno deciso di adottare il seguente formato per la rappresentazione delle date:

DD-MM-YYYY

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

3.1.8 Elementi grafici

3.1.8.1 Tabelle

Ad eccezione del registro delle modifiche, le tabelle di un documento seguono le seguenti convenzioni:

- Ogni tabella è affiancata da una didascalia descrittiva posizionata sotto la tabella corrispondente;
- Ogni tabella viene identificata da un numero progressivo a partire da 1, che la identifica univocamente all'interno del documento.

3.1.8.2 Immagini

Le immagini sono centralizzate e presentano un numero e una didascalia esplicativa.

3.1.8.3 Diagrammi

Sia i diagrammi UML[G] che i diagrammi di Gantt[G] vengono riportati come immagini e quindi sono soggetti alle regole precedentemente esposte.

3.1.9 Metriche

Per poter ottenere una documentazione di buona qualità si è deciso di utilizzare le seguenti metriche. Alcuni parametri per comprendere la tabella seguente:

- **Numero frasi (N_{fr})**: numero di frasi nell'intero documento;
- **Numero lettere (N_{lett})**: numero di lettere nell'intero documento;
- **Numero parole (N_{par})**: numero di parole nell'intero documento.

Codice	Nome	Descrizione	Formula
MPD1	Indice Gulpease	Descrive la leggibilità del documento.	$89 + \frac{300 \cdot N_{fr} - 10 \cdot N_{lett}}{N_{par}}$
MPD2	Errori ortografici (EO)	Verifica sulla presenza di errori ortografici.	-

Tabella 7: Metriche per la qualità dei documenti

3.1.10 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **LaTeX**: linguaggio di markup per la preparazione di testi, basato sul programma di composizione tipografica TeX;

https://www.latex-project.org/

- **Texmaker**: editor per scrivere codice LaTeX. *CodeBusters* ha deciso di utilizzare questo editor perché gratis, perché prevede un compilatore integrato e un visualizzatore PDF; inoltre prevede delle funzionalità interessanti come l'autocompletamento dei comandi LaTeX, il controllo dell'ortografia e il code folding[G];

  https://www.xm1math.net/texmaker/

- **Draw.io**: programma per la produzione dei diagrammi UML[G]. Il gruppo ha scelto questo strumento perché non è richiesto scaricare alcun software per progettare un diagramma; inoltre rispetto a molti programmi simili offre maggiori funzionalità.

  https://www.diagrams.net/

3.2 Gestione della Configurazione

3.2.1 Scopo

Lo scopo di questa sezione è definire come il gruppo ha deciso di attuare il processo di supporto di gestione della configurazione sui file prodotti.

3.2.2 Aspettative

Le aspettative del gruppo *CodeBusters* nell'utilizzo di questo processo sono:

- Avere un vantaggio nell'individuazione e risoluzione di possibili conflitti o errori;
- Avere il tracciamento di ogni modifica;
- Avere il ripristino, se necessario, a una versione precedente;
- Avere la condivisione tra i membri del gruppo del materiale configurato.

3.2.3 Descrizione

Il processo di gestione della configurazione ha lo scopo di mantenere organizzata e tracciabile la documentazione redatta e il codice sviluppato, creando una storia per ogni file prodotto. In particolare si vuole gestire la struttura e la disposizione delle varie parti di ogni file all'interno di repository[G] facilmente accessibili e navigabili. Inoltre il processo si occupa anche di mantenere ordinati tali repository[G], favorendo lo sviluppo di un senso dell'orientamento.

3.2.4 Versionamento

3.2.4.1 Codice di versione

Ogni versione è identificata tramite un codice numerico di cinque cifre:

[X].[Y].[Z]-[A].[B]

Le prime tre rappresentano:

- **X**: versione stabile, ossia sottoposta ad approvazione del responsabile del documento;
- **Y**: versione controllata, ossia sottoposta a revisione complessiva da parte del verificatore;
- **Z**: versione modificata dal redattore con relativa verifica.

Dall'inizio della progettazione e codifica del prodotto software il gruppo ha deciso d'integrare ulteriori due cifre, in questo modo:

***Codice*-[A].[B]**

dove:

- **A**: indica una versione completa e funzionante del prodotto che supera tutti i test, soddisfa le metriche e implementa i requisiti obbligatori;

- **B**: cresce al raggiungimento degli obiettivi degli incrementi pianificati nel *Piano di Progetto v2.0.0-0.2*.

Tutte le cifre iniziano dal valore 0.
Ciascuna cifra aumenta di un'unità ogni volta che si compie un'operazione sul documento; inoltre:

Se la cifra X è modificata : le cifre Y e Z ritornano a 0 (per esempio da 1.2.6-0.0 a 2.0.0-0.0);

Se la cifra Y è modificata : la cifra Z ritorna a 0 (per esempio da 1.2.6-0.0 a 1.3.0-0.0).

Le cifre A e B non vengono mai annullate.

3.2.4.2 Sistemi software utilizzati

Per gestire le versioni è stato deciso di utilizzare il version control system (VCS[G]) distribuito[G] Git[G]. Le motivazioni di questa scelta si racchiudono nei vantaggi di utilizzo rispetto a VCS[G] centralizzati:

- Possibilità di lavorare in locale senza il supporto del nodo centrale remoto;
- Possibilità di creare diversi flussi di lavoro (branch[G]) per lavorare a documenti differenti;
- Miglior gestione dei conflitti, a favore di una migliore collaborazione.

Per gestire i repository[G] Git[G] è stata scelto il servizio GitHub[G] per i seguenti motivi:

- Integrazione di un issue tracking system(ITS)[G];
- Possibilità di utilizzarlo tramite browser, applicazione desktop, applicazione mobile o linea di comando;
- Buona conoscenza di quest'ultimo da parte di tutti i membri del gruppo.

3.2.5 Struttura dei repository

3.2.5.1 Repository utilizzati

Per favorire una migliore organizzazione e divisione del lavoro è stato deciso di creare due repository[G] pubblici distinti:

CodeBusters-Docs : per il versionamento dei documenti.
Riferimento: https://github.com/CodeBusterswe/CodeBusters-Docs.git

CodeBusters-HDviz : per il versionamento del codice.
Riferimento: https://github.com/CodeBusterswe/CodeBusters-HDviz.git

3.2.5.2 CodeBuster-Docs

L'organizzazione del lavoro collaborativo è così riassunta:

- Ramo principale main in cui è presente la sola documentazione pronta alla revisione;
- Ramo develop in cui effettuare periodicamente il merge[G] dai vari rami minori;

- Rami derivanti dal develop con nomi parlanti, ognuno dedicato alla stesura di uno specifico documento (l'idea fondante è quella del Git feature branch workflow[G]);
- Ramo fixTemplate per evitare errori sul template al momento del merge dei rami minori nel develop.

Nel *main* i file sono contenuti in diverse cartelle, a seconda della revisione per cui sono stati prodotti. Attualmente sono presenti le cartelle RR (*Revisione dei Requisiti*) e RP(*Revisione di Progettazione*). Il file *.gitignore* è l'unico esterno a cartelle e dichiara esplicitamente l'estensione dei file automaticamente generati da LaTeX da non tracciare, poiché poco utili allo scopo del repository[G].
I file nelle cartelle principali del main sono organizzati in sottocartelle:

- **DocEsterna**: contiene l'*Analisi dei Requisiti*, il *Piano di Progetto*, il *Piano di Qualifica*, il *Glossario*, i *verbali esterni*;
- **DocInterna**: contiene lo *Studio di Fattibilità*, le *Norme di Progetto*, i *verbali interni*;
- **Utility**: contiene file di utilità generale come il template dei documenti, comandi LaTeX per velocizzare la redazione e immagini utilizzate.

3.2.5.3 Gestione dei cambiamenti in CodeBuster-Docs

La separazione del flusso di lavoro tra i vari documenti da redarre permette una notevole diminuzione dei conflitti. Il punto focale è che il ramo main rimanga pulito da ogni tipo di errore, per cui non è utilizzabile da nessun membro del gruppo fino a che ciascun responsabile non abbia dato l'approvazione al corrispettivo documento. Solo in quel momento è permesso il merge[G] del ramo develop nel main.
I cambiamenti da gestire sui documenti possono essere:

- **Modifiche minori**: riguardano errori grammaticali, lessicali o di sintassi, che possono essere corretti dai redattori senza l'approvazione del responsabile;
- **Modifiche generali**: riguardano cambiamenti più generali come la struttura del documento o convenzioni da utilizzare e richiedono il consulto con il responsabile, il quale potrà accettare o declinare la proposta di modifica.

3.2.5.4 CodeBusters-HDviz

Il repository[G] è composto da un solo branch[G] main che contiene tutti i file della piattaforma che *CodeBusters* sta progettando. I file sono suddivisi nel seguente modo:

- **src**: la cartella che contiene i file sorgente del codice. È composta da:
  - **components**: la cartella contiene al suo interno una cartella per ogni componente del programma, dove al suo interno ci sarà il file sorgente e il file del test di unità;
  - **__snapshots__**: la cartella viene creata dall'utilizzo della React Testing Library e contiene gli snapshot[G] test dei singoli componenti. Ognuno di questi è nominato in questo modo:

    App.*test.js.snap*

    dove **App** è il nome del componente.

- **test**: la cartella contiene i file dei test di unità. È composta da:
  - **integrazione**: contiene i file dei test di integrazione;
  - **sistema**: contiene i file dei test di sistema.

I singoli file di sorgente sono denominati nel seguente modo:

- **Componenti**: il nome del file sorgente deve indicare la sua funzione o quello che rappresenta;
- **Test**: il nome del file sorgente del test segue la seguente nomenclatura:

  [Componente].[Categoria].*test.js*

  dove:
  - **Componente**: indica il nome del componente a cui il test si riferisce;
  - **Categoria**: la categoria del test che può essere:
    - **unit**: unità;
    - **int**: integrazione;
    - **syst**: sistema.

3.2.5.5 Gestione dei cambiamenti in CodeBuster-HDviz

I cambiamenti apportati alla repository[G] non devono includere codice "sporco", di debug o qualsiasi forma di codice provvisorio.

3.2.6 Metriche

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

3.2.7 Strumenti

Non sono stati identificati degli strumenti particolari per la gestione della configurazione.

3.3 Gestione della Qualità

3.3.1 Scopo

Lo scopo di questa sezione è definire gli obiettivi che il gruppo si è posto nel processo di supporto della gestione della qualità.

3.3.2 Aspettative

Le aspettative del gruppo *CodeBusters* nell'utilizzo di questo processo sono:

- Avere un continuo accertamento sulla qualità del prodotto, in modo che sia conforme con quella richiesta dal proponente[G];
- Avere un'organizzazione dei documenti qualitativa, al fine di velocizzare possibili modifiche e manutenzioni future;
- Avere un livello quantificabile della qualità dei processi attuati.

3.3.3 Descrizione

Il documento *Piano di Qualifica v2.0.0-0.2* racchiude i livelli di qualità che il gruppo si è posto di mantenere e le misurazioni oggettive che descrivono gli stati di avanzamento. Il gruppo vuole perseguire la qualità del prodotto agendo in modo sistematico[G], fornendo quindi un ruolo a ciascun componente del gruppo, gestendo risorse e procedure per ogni processo in atto, effettuando test statici e dinamici sul codice prodotto in modo non invasivo.

3.3.4 Attività

Ogni membro del gruppo deve:

- Comprendere fin da subito l'obiettivo del file che sta scrivendo, che sia documentale o software;
- Porsi obiettivi incrementali, per ridurre la possibilità d'errore e perseguire correttezza;
- Utilizzare conoscenze personali o di altri componenti del gruppo, creando un avanzamento continuo della propria formazione;
- Rispettare gli standard di qualità del *Piano di Qualifica*.

3.3.4.1 Classificazione delle metriche

Le metriche scelte ed utilizzate dal gruppo sono identificabili tramite un codice univoco così composto:

M[Utilizzo][IdNumerico]

Singolarmente ciascun campo rappresenta:

- **Utilizzo**: se la metrica è per:
  - **PC**: processo;
  - **PD**: prodotto.
- **IdNumerico**: codice numerico crescente che parte da 1 e distingue le metriche dello stesso sottoinsieme (PC o PD).

3.3.5 Metriche

Il processo di gestione della qualità non fa uso di metriche qualitative particolari.

3.3.6 Strumenti

Non sono stati identificati degli strumenti particolari per la gestione di qualità.

3.4 Verifica

3.4.1 Scopo

Lo scopo di questa sezione è definire come il gruppo ha deciso di attuare il processo di verifica[G]. Questo accerta che non siano stati introdotti errori durante lo sviluppo. Essa è svolta ripetutamente su tutti i processi in esecuzione (a ogni incremento della baseline[G]).

3.4.2 Aspettative

Le aspettative del gruppo *CodeBusters* nell'utilizzo di questo processo sono:

- Verificare ogni fase rispettando criteri precisi, consistenti e modificabili se necessario;
- Automatizzare il più possibile le attività del processo;
- Rispettare gli obiettivi di copertura indicati nel *Piano di Qualifica*;
- Verificare correttamente per ottenere successo in fase di validazione.

3.4.3 Descrizione

Il processo di verifica prevede due attività principali, svolte dai verificatori[G]:

- **Analisi statica**: non richiede l'esecuzione dell'oggetto di verifica. Per questo motivo è applicabile ad ogni prodotto, accertando la conformità agli standard e convenzioni di stile;
- **Analisi dinamica**: richiede l'esecuzione dell'oggetto di verifica. Per questo motivo è applicabile al codice sviluppato ma non ai documenti. Accerta, tramite test, il funzionamento di ogni unità del codice presa singolarmente ma anche dell'intero sistema nella sua complessità.

3.4.4 Verifica della documentazione

Il processo di verifica della documentazione consiste in un'analisi statica attraverso strumenti automatici o condotta a mano (desk check[G]) con, in questo caso, due possibili metodi di lettura:

Metodo	Obiettivo	Attori	Caratteristica
Walkthrough	Rilevare errori attraverso letture ad ampio spettro.	Verificatori Redattori	Un errore riscontrato dal verificatore comporta una discussione con il redattore "colpevole" riguardo a una possibile soluzione.
Inspection	Rilevare specifici errori attraverso letture mirate.	Verificatori Redattori	Il verificatore utilizza una lista di controllo, ossia un elenco di cosa va verificato in modo selettivo.

Tabella 8: Metodi di lettura

3.4.5 Verifica del codice

Il processo di verifica del codice rappresenta l'unione delle attività di analisi statica e dinamica.

Analisi	Obiettivo	Attori	Esempi
Statica	Verificare che siano rispettati i principi di buona programmazione preimpostati dal gruppo.	Verificatori Programmatori	Analisi di flusso dei dati, verifica formale del codice ecc.
Dinamica	Trovare bug[G] ed errori eseguendo il prodotto software.	Verificatori Programmatori	Test di unità, di integrazione, di sistema, di regressione, di accettazione.

Tabella 9: Analisi del codice

3.4.5.1 Principi di buona programmazione

Il gruppo si pone come obiettivo quello di scrivere codice facilmente verificabile, favorendo il parallelismo tra i processi di sviluppo e verifica. I principi da rispettare sono:

- Riflettere nel codice il design[G] progettato, utilizzando una programmazione strutturata;
- Separare le interfacce dalla loro implementazione (interfacce esposte, implementazioni nascoste);
- Massimizzare l'incapsulamento (information hiding[G]);
- Usare tipi specializzati per specificare dati, aumentando il potere espressivo del sistema.

3.4.5.2 Analisi di flusso dei dati

Il gruppo effettua un'analisi statica del codice accertando che il programma in verifica non acceda in nessuna sua parte a variabili prive di valore, quindi non ancora scritte. Controlla l'assenza di dati globali.

3.4.5.3 Verifica formale

Il gruppo effettua un'analisi statica del codice che ne prova la sua correttezza rispetto ai requisiti imposti dalle specifiche. Lo scopo è quello di esplorare tutti i rami possibili di esecuzione, senza doverlo necessariamente eseguire.

3.4.5.4 Test

Questa attività ha lo scopo di rivelare al programmatore errori o bug[G] riscontrabili a run-time. L'esecuzione dei test deve essere ripetibile ed è quindi buona pratica renderla il più possibile automatica. Il gruppo si pone l'obiettivo di rispettare gli standard di qualità dettati nel *Piano di Qualifica*.

3.4.5.4.1 Classificazione dei test

I test sono identificati tramite un codice formato dai seguenti campi:

T[TipologiaTest][ImportanzaRequisito]*[TipologiaRequisito]*[IdNumerico]

Singolarmente essi rappresentano:

- **TipologiaTest**: identifica la tipologia di test tra quelli prima citati:
  - **U** : unità;
  - **I** : integrazione;
  - **S** : sistema;
  - **R** : regressione;
  - **A** : accettazione.
- **ImportanzaRequisito & TipologiaRequisito**: sono seguiti da '*' perché presenti solo nei codici per i test di sistema e accettazione. Questi campi identificano il requisito che si vuole testare (o più requisiti con la stessa importanza e stessa tipologia), seguendo ciò che è riportato al paragrafo 2.2.4.1.6;
- **IdNumerico**: codice numerico crescente che parte da 1 e distingue i test dello stesso tipo (U, I, S, R, A).

3.4.6 Metriche

Alcuni parametri per comprendere la tabella seguente:

- **Budget at Completion (BAC)**: numero intero, corrisponde al budget totale allocato inizialmente per il progetto;
- **Number of Changed (NoC)**: numero di requisiti cambiati;
- **Number of Deleted (NoD)**: numero di requisiti eliminati;
- **Number of Added (NoA)**: numero di requisiti aggiunti;
- **Total Number of Initial Requirement (TNIR)**: numero totale dei requisiti iniziali;
- **Number of Satisfied (NoS)**: numero totale di requisiti soddisfatti;
- **Total number of Requirement (TnR)**: numero totale di requisiti;
- **Line of Code Executed (LCE)**: linee di codice eseguite dagli algoritmi di test;
- **Line of Code(LC)**: linee di codice totali;
- **Number of Quality Metrics Satisfied (NQMS)**: numero di metriche di qualità soddisfatte;
- **Total number of Quality Metrics (TQM)**: numero totale di metriche di qualità;
- **Passed test cases percentage (PTCP)**: numero totale di metriche di qualità;
- **Failed test cases percentage (FTCP)**: numero totale di metriche di qualità.

Codice	Nome	Descrizione	Formula
MPC1	Schedule Variance (SV)	Descrive se il gruppo sta rispettando o meno i tempi prestabiliti per i processi.	((EV / PV) - 1) ·100
MPC2	Budget Variance (BV)	Descrive se il gruppo sta rispettando o meno i costi prestabiliti per i processi.	((EV / AC) - 1) ·100
MPC3	Estimated at Completion (EAC)	Indica il budget totale allocato per il progetto.	AC + ETC
MPC4	Earned Value (EV)	Rappresenta il valore prodotto dal progetto fino al momento della misurazione in seguito alle attività svolte.	% completamento $\cdot EAC$
MPC5	Planned Value (PV)	Corrisponde al denaro che si dovrebbe guadagnare al momento della misurazione.	% lavoro pianificato $\cdot BAC$
MPC6	Actual Cost (AC)	soldi spesi per il progetto fino al momento del calcolo.	Numero intero
MPC7	Estimate to Complete (ETC)	Valore stimato delle attività necessarie per il completamento del progetto.	Numero intero
MPC8	Requirements stability index (RSI)	Indica quanto i requisiti variano nel tempo.	$(1 - \frac{NoC+NoD+NoA}{TNIR}) \cdot 100$
MPC9	Satisfied obligatory requirements (SOR)	Descrive se il gruppo ha soddisfatto i requisiti obbligatori o meno.	$\frac{NoS}{TnR} \cdot 100$
MPC10	Code Coverage (CC)	Descrive quanto il codice prodotto è coperto dalla suite di test dinamici.	$\frac{LCE}{LC} \cdot 100$
MPC11	Passed test cases percentage (PTCP)	Corrisponde alla percentuale di test passati rispetto alla suite di test dinamici.	Test passati / Test totali
MPC12	Failed test cases percentage (FTCP)	Corrisponde alla percentuale di test falliti rispetto alla suite di test dinamici.	Test falliti / Test Totali

MPC13	Quality Metrics Satisfied (QMS)	Descrive la percentuale di metriche di qualità soddisfatte.	$\frac{NQMS}{TQM} \cdot 100$
MPC14	Non-calculated Risk (NR)	Indica il numero di rischi non preventivati.	Numero intero

Tabella 10: Metriche per i processi

3.4.7 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **Texmaker**: mette a disposizione un correttore automatico che individua le parole grammaticalmente scorrette sottolineandole in rosso;
- **Jest**: framework[G] di testing utilizzato per testare il codice JavaScript[G] e le attività interne delle componenti React[G];

  https://jestjs.io/

- **ESLint**: programma utilizzato per l'analisi statica del codice. Nello specifico, il tipo di analisi effettuata è chiamata code linting[G]. Questo programma mette a disposizione regole sintattiche che il codice deve rispettare e permette di definirne di nuove;

  https://eslint.org/

- **SonarCloud**: servizio web che permette di eseguire controlli del codice presente all'interno di un repository[G] e di riportare i risultati delle analisi in un cruscotto accessibile tramite web. Può quindi essere integrato con il repository[G] di GitHub[G] del gruppo, in modo che ad ogni caricamento o modifica il software analizzi il codice. Il cruscotto dei risultati è disponibile nel profilo del team e viene riportato come badge sul repository[G] stesso;

  https://sonarcloud.io/

- **Codecov**: strumento utilizzato per stabilire la percentuale di codice coperto dai test implementati. Questo strumento viene integrato nel repository[G] di GitHub[G] del gruppo tramite GitHub Action[G] e ad ogni build[G] i risultati vengono aggiornati nel profilo. La percentuale di code coverage[G] viene riportata come badge sul repository[G] stesso;

  https://sonarcloud.io/

- **React Testing Library**: libreria utilizzata per testare e renderizzare[G] le componenti React[G] del prodotto.

  https://testing-library.com/

3.5 Validazione

3.5.1 Scopo

Lo scopo di questa sezione è fissare come il gruppo ha deciso di attuare il processo di validazione[G]. Questo comprende le attività di controllo mirate a confrontare che il prodotto sia conforme ai requisiti accordati con il proponente[G].

3.5.2 Aspettative

Le aspettative del gruppo *CodeBusters* nell'utilizzo di questo processo sono:

- Dimostrare la correttezza delle attività svolte in fase di verifica;
- Avere la certezza che il prodotto software rispetti i requisiti riportati nell'*Analisi dei Requisiti*.

3.5.3 Descrizione

Le attività di validazione seguono quelle di verifica e validano i risultati ottenuti dai test. Sono attività che possono essere svolte non solo a prodotto finito, ma anche integrate con il processo di sviluppo per validare i risultati ottenuti fino a quel momento. È compito del *Responsabile di progetto* accettarli solo se rispettano le aspettative del gruppo e del proponente[G].

In questa seconda versione delle *Norme di Progetto* non si ha ancora attivato questo processo non avendo ancora iniziato la fase di sviluppo del codice (se non per il Proof of Concept[G]).

3.5.4 Metriche

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

3.5.5 Strumenti

Non sono stati identificati degli strumenti particolari per la validazione.

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Scopo

In questa sezione vengono esposte le modalità di coordinamento adottate dal gruppo, che sono:

- Adottare un modello organizzativo per l'individuazione dei rischi che potrebbero verificarsi;
- Definire un modello di sviluppo da adottare;
- Pianificare il lavoro rispettando le scadenze;
- Calcolo del prospetto economico in base ai ruoli;
- Determinare un bilancio finale sulle spese.

4.1.2 Aspettative

Le aspettative nell'applicazione del processo di gestione organizzativa sono:

- Ottenere una pianificazione ragionevole delle attività da seguire;
- Avere un coordinamento delle attività, assegnando ruoli, compiti e semplificando la comunicazione tra i membri;
- Riuscire a regolare le attività e renderle economiche.

4.1.3 Descrizione

Le attività di gestione sono:

- Assegnazione dei ruoli e dei compiti;
- Inizio e definizione dello scopo;
- Istanziazione dei processi;
- Pianificazione e stima di tempi, risorse e costi;
- Esecuzione e controllo;
- Revisione e valutazione periodica delle attività.

4.1.4 Ruoli di progetto

Ogni membro del gruppo deve, a rotazione, ricoprire almeno una volta ciascun ruolo di progetto per comprendere le differenze tra le diverse figure aziendali. Tali ruoli sono descritti di seguito.

4.1.4.1 Responsabile di progetto

Il responsabile di progetto ricopre un ruolo fondamentale, in quanto si occupa delle comunicazioni con il proponente e committente. Inoltre, egli deve svolgere i seguenti compiti:

- Pianificare;
- Gestire;
- Controllare;
- Coordinare.

4.1.4.2 Amministratore di progetto

L'amministratore deve avere il controllo dell'ambiente di lavoro ed essere di supporto. Inoltre egli deve:

- Dirigere le infrastrutture di supporto;
- Controllare versioni e configurazioni;
- Risolvere i problemi che riguardano la gestione dei processi;
- Gestire la documentazione.

4.1.4.3 Analista

L'analista si occupa dell'analisi dei problemi e del dominio applicativo. Questa figura ha le seguenti responsabilità:

- Studio del dominio del problema;
- Redazione della documentazione: *Analisi dei Requisiti* e *Studio di Fattibilità*;
- Definizione dei requisiti e della sua complessità.

4.1.4.4 Progettista

Il progettista si occupa dell'aspetto tecnico e tecnologico del progetto, segue lo sviluppo e non la manutenzione del prodotto. Inoltre egli deve scegliere:

- Un'architettura adatta per il sistema del prodotto in base alle tecnologie scelte;
- Il modo più efficiente per ottimizzare l'aspetto tecnico del progetto.

4.1.4.5 Programmatore

Il programmatore si occupa della parte di codifica in base alle specifiche fornite dal progettista, operando con ottica di manutenibilità del codice. Inoltre egli deve creare e gestire componenti di supporto per la verifica e la validazione del codice.

4.1.4.6 Verificatore

Il verificatore[G] è presente durante tutta l'attività del progetto. Il verificatore deve:

- Controllare i prodotti in fase di revisione, utilizzando le tecniche e gli strumenti definiti nelle *Norme di Progetto v2.0.0-0.2*;
- Evidenziare gli errori e segnalarli all'autore del prodotto in questione.

4.1.5 Procedure

Per il coordinamento e le comunicazioni durante la realizzazione del progetto, il gruppo adotterà le seguenti procedure:

- **comunicazione interna**: coinvolge tutti i membri del team;
- **comunicazione esterna**: con proponente[G] e committente.

4.1.5.1 Gestione delle comunicazioni

4.1.5.1.1 Comunicazioni interne

Le comunicazioni interne avvengono attraverso il canale Discord[G]. Quest'applicazione favorisce la collaborazione a distanza e viene utilizzata anche in ambienti aziendali. Tale software permette al team di creare uno spazio di lavoro condiviso.

4.1.5.1.2 Comunicazioni esterne

Le comunicazioni con utenti esterni al gruppo sono gestite dal responsabile del progetto. Le modalità utilizzate sono le seguenti:

- Tramite posta elettronica (all'indirizzo **codebusterswe@gmail.com**);
- Attraverso Skype[G] per i colloqui con l'azienda Zucchetti.

4.1.5.2 Gestione degli incontri

4.1.5.2.1 Incontri interni

Il responsabile del progetto concorda con il team gli incontri interni. Egli ha il compito di specificare le date delle riunioni nel calendario e approvare i verbali redatti. I membri del gruppo sono tenuti a partecipare alle riunioni, interagendo nei dibattiti. Affinché una riunione sia ritenuta valida, devono essere presenti almeno cinque membri del gruppo.

4.1.5.2.2 Verbali di riunioni interne

In occasione di ogni incontro interno viene redatto un verbale dal segretario scelto dal responsabile. Il contenuto della riunione deve essere riportato nel verbale corrispondente e approvato dal responsabile.

4.1.5.2.3 Incontri esterni

Il responsabile del progetto organizza gli incontri esterni con il proponente[G] o committente. Questi potrebbero richiedere incontri con il team; il responsabile è tenuto a proporre una data in accordo con le parti e la comunica attraverso i canali sopra citati.

4.1.5.2.4 Verbali di riunioni esterne

In occasione di ogni incontro esterno viene redatto un verbale da un redattore scelto dal responsabile. Il contenuto della riunione deve essere riportato nel verbale esterno corrispondente e approvato dal responsabile.

4.1.5.3 Gestione degli strumenti e di coordinamento

4.1.5.3.1 Ticketing

Il ticketing permette ai membri del gruppo di rimanere aggiornati sullo stato delle attività in corso. Il responsabile assegna compiti ai membri del gruppo e controlla l'andamento delle tasks[G] assegnate. Lo strumento di ticketing scelto è Planner[G]: si tratta di un'applicazione multi-piattaforma[G], in cui è visibile a tutti i membri del team lo stato di avanzamento dei compiti assegnati, con la possibilità di aggiungerne di nuovi.

4.1.5.4 Gestione dei rischi

Il responsabile è tenuto a individuare i rischi e renderli noti; tale attività verrà documentata nel *Piano di Progetto*. La procedura per la gestione dei rischi è:

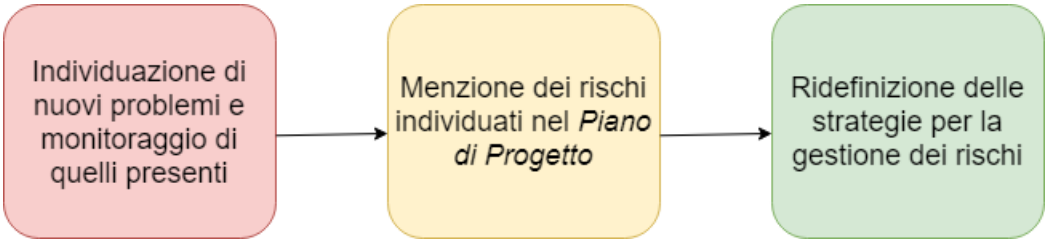

Figura 2: Procedura per la gestione dei rischi

4.1.5.4.1 Codifica dei rischi

I rischi sono codificati nel modo seguente:

R[Categoria][Numero]

Dove:

- **[Categoria]**: indica la classificazione del rischio e può assumere tre valori:
  - **T**: rischi tecnologici;
  - **O**: rischi organizzativi;
  - **I**: rischi interpersonali.
- **[Numero]**: è un numero progressivo che identifica univocamente il rischio all'interno di una categoria

4.1.6 Metriche

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

4.1.7 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **Github**: piattaforma che permette la condivisione in remoto di tutti i file prodotti e la creazione di una storia per ciascuno di questi file;

  https://github.com/

- **Git**: sistema di controllo delle versioni;

  https://git-scm.com/

- **Telegram**: applicazione di messaggistica per la comunicazione rapida e la gestione del gruppo;

  https://telegram.org/

- **Discord**: applicazione multipiattaforma[G] utilizzata per le riunioni interne;

  https://discord.com/

- **Skype**: applicazione che consente di effettuare delle videoconferenze, utilizzata per la comunicazione con il proponente[G];

  https://www.skype.com/it/

- **Google Drive**: server per la condivisione rapida di alcune documentazioni che possono essere utili riguardo l'attività in svolgimento;

  https://www.google.com/intl/it_it/drive/

- **Microsoft Planner**: piattaforma per la gestione dei compiti in svolgimento, svolti e da svolgere. Permette al gruppo di rispettare gli incrementi e le scadenze dettate dal *Piano di Progetto*.

  https://www.microsoft.com/it-it/microsoft-365/business/task-management-software

4.2 Formazione

4.2.1 Scopo

Lo scopo di questa sezione è definire le norme che riguardano la formazione dei membri del gruppo e quindi lo studio delle tecnologie utilizzate per produrre i documenti e costruire il prodotto richiesto.

4.2.2 Aspettative

Le aspettative sono:

- Ottenere una buona conoscenza di LaTeX;
- Ottenere una buona conoscenza delle librerie, degli strumenti e del linguaggio utilizzati per la codifica.

4.2.3 Descrizione

Il processo di formazione è un processo per fornire e mantenere i componenti del gruppo qualificati.

4.2.4 Modalità di formazione

La formazione di ogni membro del gruppo avviene attraverso studio autonomo delle varie tecnologie, sia quelle proposte dal proponente, che quelle prese in considerazione dal gruppo.

4.2.5 Metriche

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

4.2.6 Strumenti

Alcuni strumenti, librerie, framework[G] necessitano di una ampia formazione. In particolare:

- **React**: libreria JavaScript[G] per la creazione di interfacce utente;
- **D3.js**: libreria per la manipolazione del DOM (Document Object Model);
- **DruidJS**: libreria per la riduzione dimensionale sui dati caricati nella web app[G];
- **Node.js**: runtime JavaScript[G] utilizzato per creare applicazioni di rete scalabili.

A Standard ISO/IEC 12207

ISO/IEC 12207 è uno standard internazionale per i processi del ciclo di vita del software. Lo standard stabilisce i processi presenti nel ciclo di vita del software e, per ciascuno di essi, le attività da svolgere e i risultati da produrre. I processi sono suddivisi dalla norma in tre categorie:

- Processi primari;
- Processi di supporto;
- Processi organizzativi.

A.1 Processi primari

I processi primari definiti dallo standard ISO/IEC 12207 sono i seguenti.

A.1.1 Acquisizione

Definisce le attività dell'acquirente, l'organizzazione che acquisisce un sistema, un prodotto software o un servizio software. Le attività sono le seguenti:

- Iniziazione;
- Preparazione della richiesta di proposta;
- Preparazione e aggiornamento del contratto;
- Monitoraggio dei fornitori;
- Accettazione e completamento.

A.1.2 Fornitura

Definisce le attività del fornitore, l'ente che fornisce il sistema, il prodotto software o il servizio software. Le attività sono le seguenti:

- Iniziazione;
- Preparazione della risposta;
- Contratto;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Rilascio e completamento.

A.1.3 Sviluppo

Il processo ha lo scopo di sviluppare un prodotto software, o un sistema basato sul software, che indirizzi le esigenze del cliente. Le attività sono le seguenti:

- Iniziazione;
- Analisi dei requisiti di sistema;
- Progettazione architettura di sistema;
- Analisi dei requisiti software;
- Progettazione software;
- Codifica;
- Integrazione dei componenti;
- Collaudo del software;
- Integrazione del sistema;
- Collaudo di sistema;
- Installazione software.

A.1.4 Esercizio

Il processo di esercizio è svolto simultaneamente al processo di manutenzione. Il processo ha lo scopo di mantenere operativo il sistema e di fornire il supporto agli utenti. Le attività sono le seguenti:

- Implementazione dei processi;
- Operational testing;
- System operation;
- Supporto utente.

A.1.5 Manutenzione

Le fase di manutenzione è svolta simultaneamente alla fase di esercizio.
Il processo ha lo scopo di modificare il prodotto software dopo il suo rilascio per correggere i difetti, migliorare le sue prestazioni o altri attributi o adattarlo a cambiamenti nell'ambiente operativo. Le attività sono le seguenti:

- Analisi del problema e delle modifiche;
- Implementazione delle modifiche;
- Revisione/accettazione della manutenzione;
- Migrazione;
- Ritiro del software.

A.2 Processi di supporto

I processi di supporto 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.
I processi definiti dallo standard sono i seguenti.

A.2.1 Documentazione

Il processo di documentazione garantisce lo sviluppo e la manutenzione delle informazioni prodotte e registrate relativamente al prodotto software. Il processo contiene una serie di attività per gestire i documenti:

- Pianificazione;
- Progettazione e sviluppo;
- Produzione;
- Manutenzione.

A.2.2 Gestione della configurazione

Il processo di gestione della configurazione ha lo scopo di definire e mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto. Le sue attività sono:

- Identificazione;
- Controllo della configurazione;
- Controllo;
- Valutazione della configurazione;
- Gestione del rilascio e distribuzione.

A.2.3 Accertamento della qualità

Il processo ha lo scopo di assicurare che tutti i prodotti di fase siano conformi con i piani e gli standard definiti.
Le sue attività sono:

- Accertamento di prodotto;
- Accertamento di processo;
- Accertamento della qualità di sistema.

A.2.4 Verifica

Il processo di verifica è un processo per determinare se i prodotti software di un'attività soddisfano i requisiti o le condizioni a loro imposti nelle attività precedenti. Per l'efficacia dei costi e delle prestazioni, la verifica deve essere integrata, il prima possibile, con il processo che la utilizza. Questo processo include attività di analisi, revisione e test. Può essere eseguito con diversi gradi di indipendenza. Nel caso in cui il processo venga eseguito da un'organizzazione indipendente dal fornitore, sviluppatore, operatore o manutentore, viene chiamato processo di verifica indipendente.

A.2.5 Validazione

Il processo di validazione ha lo scopo di confermare che i requisiti siano rispettati quando uno specifico prodotto sia utilizzato nell'ambiente destinatario.

A.2.6 Revisione congiunta

Il processo di revisione congiunta ha lo scopo di rivedere con gli stakeholders[G] 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. Le revisioni sono svolte durante l'intero ciclo di vita, sia a livello di progetto che a livello tecnico. La revisione congiunta è svolta tra gli stessi componenti del team quando si revisiona un componente del prodotto oppure, tra fornitore e committente quando si revisiona l'intero prodotto. Le attività che la compongono sono:

- Revisioni di gestione del progetto;
- Revisioni tecniche.

A.2.7 Audit

Il processo di audit ha lo scopo di determinare in maniera indipendente la conformità di prodotti e processi selezionati ai requisiti, piani e accordi. L'attività di auditing è svolta da personale che non ha partecipato direttamente allo sviluppo dei prodotti, dei servizi o dei sistemi oggetto delle revisioni.

A.2.8 Risoluzione dei problemi

Il processo di risoluzione dei problemi è un processo per analizzare e risolvere i problemi, qualunque sia la loro natura o fonte, che vengono scoperti durante l'esecuzione di sviluppo, funzionamento, manutenzione o altri processi. L'obiettivo è fornire un mezzo tempestivo, responsabile e documentato per garantire che tutti i problemi rilevati siano analizzati e risolti.

A.3 Processi organizzativi

I processi organizzativi sono impiegati in una organizzazione per stabilire e implementare una struttura, per organizzare e gestire i processi del ciclo di vita e del personale e per il continuo miglioramento dei processi e della struttura stessa.

A.3.1 Gestione organizzativa

Il processo di gestione organizzativa ha lo scopo di organizzare, monitorare e controllare l'avvio e le prestazioni di un processo per il raggiungimento dei loro obiettivi in accordo con quelli di business dell'organizzazione. Il processo è stabilito da una organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti.
Questo processo è composto da una serie di attività:

- Inizializzazione e definizione dello scopo;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e Valutazione;
- Chiusura.

A.3.2 Gestione delle infrastrutture

Il processo ha lo scopo di mantenere una infrastruttura stabile ed affidabile necessaria a supportare le prestazioni di qualsiasi processo. L'infrastruttura può includere hardware, software, metodi, tools, tecniche, standard ed utilità per lo sviluppo, operatività o manutenzione.
Il processo è composto dalle seguenti attività:

- Implementazione del processo;
- Istituzione dell'infrastruttura;
- Mantenimento dell'infrastruttura.

A.3.3 Miglioramento del processo

Il processo ha lo scopo di stabilire, valutare, misurare, controllare e migliorare il ciclo di vita del software.
Le attività che lo compongono sono:

- Istituzione del processo;
- Valutazione del processo;
- Miglioramento del processo.

A.3.4 Formazione del personale

Il processo di formazione ha lo scopo di fornire e mantenere personale qualificato. L'acquisizione, la fornitura, lo sviluppo e molti altri processi dipendono in gran parte da personale esperto e competente.
Il processo è composto dalle seguenti attività:

- Implementazione del processo;
- Materiale per l'implementazione della formazione;
- Pianificazione per l'implementazione della formazione.

B Standard di qualità ISO/IEC 9126

ISO/IEC 9126 descrive un modello di qualità del software. Il modello propone un approccio alla qualità in modo tale che le società di software possano migliorare l'organizzazione e i processi e, quindi come conseguenza concreta, la qualità del prodotto sviluppato.
La norma tecnica relativa alla qualità del software si compone nelle seguenti quattro parti.

B.1 Modello di qualità

Il modello di qualità stabilito nella prima parte dello standard è classificato da sei caratteristiche generali e varie sottocaratteristiche misurabili attraverso delle metriche.
Il modello è articolato nel seguente modo:

B.1.1 Funzionalità

La funzionalità è la capacità di un prodotto software di fornire funzioni che soddisfino esigenze stabilite, necessarie per operare sotto condizioni specifiche.

- **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 negandone l'accesso e la modifica a persone e sistemi non autorizzati ma permettendola a chi è abilitato.

B.1.2 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.

- **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.

B.1.3 Efficienza

L'efficienza è la capacità di fornire appropriate prestazioni relativamente alla quantità di risorse usate.

- **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.

B.1.4 Usabilità

L'usabilità è la capacità del prodotto software di essere capito, appreso, usato e benaccetto dall'utente, quando usato sotto condizioni specificate.

- **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.

B.1.5 Manutenibilità

La manutenibilità è la capacità del software di essere modificato, includendo correzioni, miglioramenti o adattamenti.

- **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;
- **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.

B.1.6 Portabilità

La portabilità è la capacità del software di essere trasportato da un ambiente di lavoro ad un altro.

- **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.

B.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 in un contesto tecnico rilevante o di business.

B.3 Metriche per la qualità interna

La qualità interna, più precisamente le metriche interne, si applica al software non eseguibile 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. Le metriche interne permettono di individuare eventuali problemi che potrebbero influire sulla qualità finale del prodotto prima che sia realizzato il software eseguibile. Esistono metriche che possono simulare il comportamento del prodotto finale tramite simulazioni.

B.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 sia il livello di qualità esterna sia il livello di qualità interna sono raggiunti. La qualità in uso permette di abilitare specificati utenti ad ottenere specificati obiettivi con efficacia, produttività, sicurezza e soddisfazione.

- **Efficacia**: la capacità del software di permettere agli utenti di raggiungere gli obiettivi specificati con accuratezza e completezza;
- **Produttività**: la capacità di permettere agli 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.