Progetto: ***HD Viz***

codebusterswe@gmail.com

Norme di Progetto

Informazioni sul documento

Versione	1.0.0
Approvatori	Sassaro Giacomo
Redattori	Pirolo Alessandro Zenere Marco Rago Alessandro Safdari Hossain
Verificatori	Baldisseri Michele Scialpi Paolo
Uso	Interno
Distribuzione	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	Verificatore	Descrizione
0.0.11	2020-12-21	Rago Alessandro	Analista	-	Stesura §3.5
0.0.10	2020-12-21	Pirolo Alessandro	Analista	-	Stesura §3.1.7.5
0.0.9	2020-12-20	Pirolo Alessandro	Analista	-	Stesura §3.1.8
0.0.8	2020-12-20	Rago Alessandro	Analista	-	Stesura §3.4
0.0.7	2020-12-20	Pirolo Alessandro	Analista	-	Stesura §3.1.7
0.0.5	2020-12-19	Rago Alessandro	Analista	-	Stesura §3.2 e in parte §3.3
0.0.5	2020-12-19	Zenere Marco	Analista	-	Processi primari.
0.0.4	2020-12-18	Safdari Hossain	Analista	-	Processi organizzativi.
0.0.3	2020-12-17	Pirolo Alessandro	Analista	-	Iniziata stesura §3 fino a §3.1.6.3
0.0.2	2020-12-15	Pirolo Alessandro	Analista	-	Stesura §1.
0.0.1	2020-12-14	Zenere Marco	Analista	-	Creazione scheletro documento, introduzione e paragrafi.

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire le linee guida di tutti i processi istanziati dal gruppo *CodeBusters*, inoltre contiene l'organizzazione e l'uso di tutte le risorse di sviluppo e le convenzioni che il gruppo decide di attuare sull'uso delle tecnologie, sullo stile di codifica e di scrittura. Ogni membro del gruppo 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 e di conseguenza serve un software in grado di eseguire un'analisi e una interpretazione delle informazioni.
Il capitolato[G] C4 ha come obiettivo quello di creare un'applicazione di visualizzazione di dati con numerose dimensioni in un formato comprensibile dall'occhio umano. A questo scopo è necessario utilizzare algoritmi di intelligenza artificiale, o nel caso svilupparne di nuovi, che, agendo sulla distanza dei vari punti del grafico, riescano a sviluppare un modello semplificato che ne evidenzi i cluster[G]. L'applicazione dovrà inoltre agire su questi grafici creati evidenziando i dati ottenuti.

1.3 Glossario

Per evitare ambiguità relative alle terminologie utilizzare, è stato compilato il *Glossario 1.0.0*. 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
- **Guide to the Software Engineering Body of Knowledge(SWEBOK), 2014**

- **Software Engineering - Ian Sommerville - 10 th Edition (2010)**:
  (formato cartaceo);
- **Slide T3 del corso Ingegneria del Software - Ciclo di vita del software**:
  https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L03.pdf

2 Processi Primari

2.1 Fornitura

2.1.1 Descrizione

In questa sezione si presenteranno le regole a cui si atterranno i membri del gruppo Codebusters nei processi che comprendono lo studio del capitolato fino alla candidatura a fornitore del progetto HD Viz del proponente Zucchetti e dei committenti Prof. Vardanega Tullioe Prof. Cardin Riccardo.

2.1.2 Scopo della fornitura

Il fine del processo di fornitura è di scegliere 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.

2.1.3 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;
- Stimare i costi;
- Su vincoli e requisiti.

2.1.4 Documenti

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

2.1.4.1 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 C4 *HD Viz* a discapito degli altri sei proposti.

2.1.4.2 Piano di qualifica

Il piano di qualifica, redatto dal verificatore, contiene tutte le misure da adottare per garantire la qualità del prodotto. É suddiviso nelle seguenti parti:

- da inserire

2.1.4.3 Piano di progetto

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

- Analisi dei rischi;
- Modello di sviluppo;
- Pianificazione;
- Preventivo;
- Consuntivo;
- Organigramma.

2.1.5 Strumenti

2.1.5.1 Excel

Strumento utilizzato per creare grafici, fare calcoli e presentare tabelle organizzative.

2.1.5.2 Microsoft Planner

Per gestire le risorse, le task che ciascun membro del gruppo deve completare e verificare se queste sono in svolgimento, i responsabili hanno utilizzato questo strumento. Permette di assegnare attività a risorse e di verificare che vengano completate in linea con i tempi previsti.

2.2 Sviluppo

2.2.1 Descrizione

Lo scopo del processo di sviluppo è descrivere i task e le attività di analisi, progettazione, codifica, integrazione, test,installazione ed accettazione, relative al prodotto software da sviluppare.

2.2.2 Aspettative

Le aspettative sono le seguenti:

- 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 Attività

Le attività del processo di sviluppo sono tre:

- Studio di fattibilità;
- Progettazione;
- Codifica.

2.2.4 Studio di fattibilità

Questo documento, redatto dagli analisti, contiene:

- la descrizione degli attori del sistema;
- i casi d'uso individuati a partire dai requisiti;
- una visione del problema più chiara possibile per i progettisti;
- fornire ai verificatori riferimenti per l'attività di controllo dei test;
- calcolare la mole di lavoro per tracciare dei riferimenti per una stima dei costi.

2.2.4.1 Finalità

Lo scopo dell'attività è redigere tutti i requisiti in un documento.

2.2.4.2 Requisiti

I requisiti si raccolgono tramite:

- Lettura del capitolato;
- Visione della presentazione del capitolato;
- Confronto con l'azienda;

2.2.4.3 Casi d'uso

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

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

2.2.4.4 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.

2.2.4.5 Struttura dei requisiti

- **Codice identificativo:**

  R[Importanza][Tipologia][Codice]

  Per importanza si 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 si intende una lettera che rappresenta la natura del requisito:

V : Vincolo

P : Prestazionale

Q : Qualitativo

F : Funzionale

- **classificazione:** 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:** come scritto in precedenza, vi sono diverse fonti da cui possono derivare i requisiti. L'origine dei requisiti viene quindi riportata in questa sezione.

2.2.4.6 UML

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

2.2.5 Progettazione

2.2.5.1 Scopo

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

2.2.5.2 Aspettative

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

2.2.5.3 Descrizione

É formata da due parti:

Tecnology Baseline: motiva le tecnologie, i framework, e le librerie selezionate per la realizzazione del prodotto;
Product Baseline: illustra la baseline architetturale (design e coding) del prodotto, coerente con la Tecnology Baseline.

2.2.5.4 Tecnology Baseline

Sarà il progettista ad occuparsene e dovrà contenere:

- Diagrammi UML delle classi, di attività, di sequenza e dei package;
- Tecnologie adottate, motivando i motivi di tali scelte;

- Design pattern 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, ovvero un dimostratore eseguibile per dimostrare che ogni componente funzioni nel modo voluto.

2.2.5.5 Product Baseline

Il progettista dovrà occuparsi anche di questa parte che conterrà:

- una definizione delle classi, evitando nomi e funzionalità ridondanti;
- tracciamento delle classi, ovvero ciascun requisito deve essere soddisfatto da una classe;
- test di unità su ogni componente in modo da verificare il corretto funzionamento.

2.2.6 Codifica

2.2.6.1 Scopo

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

2.2.6.2 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 poter facilmente intervenire in seguito nelle attività di manutenzione, modifica, verifica e validazione.

2.2.6.3 Stile della codifica

- **Indentazione**: i blocchi di codice innestati dovranno avere una 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, la seconda dovrà iniziare con lettera maiuscola. É preferibile mantenere metodi brevi, con poche righe di codice;
- **Classi**: il nome delle classi dovrà sempre iniziare con la lettera maiuscola;
- **Costanti**: dovranno essere scritte utilizzando solo il carattere maiuscolo;
- **Univocità dei nomi**: tutti i costrutti dovranno avere nomi univoci e significativi;

- **Commenti**: i commenti dovranno essere inseriti prima dell'inizio del costrutto;
- **File**: dovranno avere un nome che inizia per lettera maiuscola che ne specifichi il contenuto;
- **Lingua**: i commenti al codice dovranno essere scritti in italiano, i nomi delle variabili possono essere scritti in inglese o italiano.

2.2.6.4 Ricorsione

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

2.3 Qualità

2.3.1 Descrizione

In questa sezione si tratterà dei modi in cui si cercherà di mantenere la qualità nelle varie fasi del progetto. Queste sono delle dichiarazioni di intenti, dunque potrebbero subire aggiunte, in caso se ne ritenesse necessario, nelle prossime versioni delle norme di progetto.

2.3.2 Analisi dei requisiti

Questa attività ha lo scopo di elencare e tracciare i requisiti e i casi d'uso richiesti dal proponente. Dopo averli formulati e approvati è necessario tracciare i loro cambiamenti.
Per verificare che il software sia di qualità è necessario che tutti i requisiti, almeno quelli obbligatori, siano rispettati.

2.3.3 Progettazione

Al termine della progettazione si dovrà avere un'architettura che ha tradotto i requisiti in unità di codice. É importante che ciascuna componente di codice si riferisca ad un requisito in modo da verificare facilmente che venga soddisfatto. I compiti del programmatore dovranno essere organizzati in modo che ciascuno si occupi di un singolo modulo e non si creino interferenze.
É buona prassi scomporre le varie componenti in piccole parti per favorire la manutenibilità.

2.3.4 Codifica

In questa fase di elaborazione vera e propria del prodotto dovranno essere seguite delle norme per rendere il codice più leggibile e favorire la manutenibilità:

- **Linee di commento**: un rapporto basso tra linee di codice e linee di commento può essere spia di carenza di informazioni;
- **Profondità della gerarchia**: avere una gerarchia molto profonda aumenta il numero di dipendenze, meglio limitarla;
- **Complessità dei moduli**: è preferibile avere moduli più semplici e scorporare moduli con più funzionalità per rendere il codice più leggibile e favorire le modifiche.
- **Numero dei parametri in un metodo**: un numero troppo elevato di parametri in un metodo potrebbe denotare un grado di complessità troppo elevato.

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 puo ciclo di vita;
- avere delle norme e convenzioni ben precise che coprono tutti gli aspetti della stesura di un documento, in modo che tutti 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

Ogni documento passa per queste fasi:

- **Pianificazione**: il documento viene pensato e vengono organizzate le varie parti. Questo accade soprattutto quando le informazioni sono numerose e complesse;
- **Impostazione**: viene creata la bozza e la struttura del documento;
- **Realizzazione**: viene redatto il contenuto del documento;
- **Verfica**: ogni sezione del documento è soggetta a revisioni da parte dei verificatori per correggere e, di conseguenza, sistemare;
- **Approvazione**: l'approvatore stabilisce che il documento è stato completato ed è pronto per essere rilasciato.

3.1.5 Template

Il gruppo ha deciso di creare un template con l'utilizzo di LaTeX, grazie al quale viene standardizzata la struttura del documento. In questo modo i componenti del gruppo si occupano unicamente di redigere il contenuto dei singoli testi senza doversi di . 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* sta ad indicare 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 prorpio. Se una sezione contiene numerose sottosezioni, allora il file avrà il compito di includere i file delle varie sottosezioni ;
- il registro delle modifiche, che contiene una lista delle modifiche effettuate al documento così da rendere tracciabili queste modifiche;
- "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.

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 la 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];
  - **Verificatori**: nomi dei componenti del gruppo che svolgono il ruolo di verificatore[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.

3.1.6.1 Registro delle modifiche

Il registro delle modifiche è una tabella che riporta ogni modifica effettuata al documento in questione. 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;
- **Verificatore**: il nome del componente che si è occupato di verificare la parte modificata;
- **Descrizione**: una breve descrizione sulla modifica effettuata.

3.1.6.2 Indice

L'indice rappresenta uno strumento di consultazione che viene usato per trovare rapidamente e facilmente un'informazione. Oltre ad elencare le sezioni in cui sono divise le diverse parti del documento, l'indice riporta prima le tabelle e successivamente le immagini presenti. L'indice viene posto dopo il registro delle modifiche.

3.1.6.3 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;

3.1.6.4 Verbali

I *Verbali* sono i documento scritti per attestare discorsi, dichiarazioni effettuate durante un incontro con o senza esterni. 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 tra sezioni, la prima è l'itroduzione 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;
- **Durata**: il tempo che il gruppo ha impiegato per terminare la riunione;
- **Partecipanti**: l'elenco dei partencipanti 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 desicisioni 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 ad eccezione di alcune cartelle:

- 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 maisucola 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, ai titoli e alle parole particolarmente importanti;
- **Corsivo**: lo stile corsivo si applica al nome del gruppo, al nome del progetto, al nome dei documenti e al nome del proponente;
- **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.

3.1.7.3 Glossario

Le norme relativa al *Glossario* sono:

- Ogni parola del *Glossario* è contrasegnata da una 'G' ad apice;
- Non vengono segnate come termini del *Glossario* le parole che fanno parte di un titolo, che sono presenti nelle tabelle e nelle didascalie;

3.1.7.4 Elenchi puntati

Gli elenchi puntati seguono le seguenti norme:

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

3.1.7.5 Sigle

Le sigle presenti nei documenti rappresentano i ruoli dei componenti:

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

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 tabela è affiancata da una didascalia descrittiva posizionata la tabella corrispondente;
- Ongi tabelle deve identificata da un numero composto dalla sezione a cui fa riferimento e da un numero progressivo che lo identifica all'interno della sezione;

3.1.8.2 Immagini

Le immagini sono centralizzato e hanno una didascalia posta sopra la figura corrispondente

3.1.8.3 Diagrammi

Sia i diagrammi UML[G] che i diagrammi di Gantst[G] vengono riportati come immagini e quindi sono soggetti alle regole esposte alla sezione 3.1.7.3.

3.1.9 Strumenti

3.1.9.1 LaTeX

Per la stesura dei documenti, il gruppo utilizza LaTeX, un linguaggio di markup per la preparazione di testi, basato sul programma di composizione tipografica TeX.

3.1.9.2 Texmaker

Per scrivere il codice LaTeX, il gruppo ha utilizzato Texmaker. *CodeBusters* ha deciso di utilizzare questo editor perché è gratis 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].

3.1.9.3 Draw.io

Un diagramma UML[G] viene prodotto con l'utilizzo di Draw.io. 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

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 possbili 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. Il particolare si vuole gestire la struttura e la disposizione delle varie parti di ogni file all'interno di un repository^{G} facilmente accessibile e navigabile. Inoltre il processo si occupa anche di mantenere ordinati i repository^{G} utilizzati, favorendo lo sviluppo di un senso dell'orientamento.

3.2.4 Versionamento

3.2.4.1 Codice di versione

La storia di ogni documento avanza nel tempo attraverso una successione di modifiche, verifiche e approvazioni. Ciascuna di queste operazioni comporta il cambiamento della versione del documento stesso. Ogni versione è identificata tramite un codice numerico di tre cifre:

X.Y.Z

Singolarmente esse rappresentano:

X : versione stabile, ossia sottoposta ad approvazione del responsabile del documento;

Y : versione controllata, ossia sottoposta a verifica di un verificatore del documento;

Z : versione modificata, ossia sottoposta a modifica di un redattore del documento.

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 0.2.6 a 1.0.0);

Se la cifra Y è modificata : la cifra Z ritorna a 0 (per esempio da 0.2.6 a 0.3.0).

3.2.4.2 Sistemi software utilizzati

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

- Possibilità di lavorare in locale senza il supporto del nodo centrale remoto;
- Possibilità di creare diversi flussi di lavoro (branch[G]) in cui 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/Panz99/CodeBusters-Docs`

??? : per il versionamento del codice. Il suo utilizzo è successivo alla revisione dei requisiti.
Riferimento: `???`

In questa prima versione delle *Norme di Progetto s*i tratta la struttura del solo repository[G] CodeBusters-Docs.

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 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 templete al momento del merge dei rami minori nel develop.

Nel *main* i file sono contenuti nella cartella RR (Revisione dei Requisiti), con l'obiettivo di crearne una nuova per ogni revisione futura.
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].
Gli altri file sono invece organizzati in ulteriori cartelle dentro la RR:

Doc_esterna : contiene l'*Analisi dei Requisiti ,* il *Piano di Progetto ,* il *Glossario ,* i *verbali esterni* ;

Doc_interna : 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 il logo del gruppo.

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 e quindi delle modifiche da apportare per errore. Il punto focale è che il ramo mian rimanga pulito da ogni tipo di errore, per cui non è utilizzabile da nessun membro del gruppo fino a che ciascun responsbile non abbia dato l'approvazione al corrispettivo documento. Solo in quel momento è permesso il merge 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 che potrà accettare o declinare la proposta di modifica.

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 r*acchiude gli standard di qualità che il gruppo si è posto di mantenere, le misurazioni oggettive che descrivono gli stati di avanzamneto. In generale per perseguire la qualità di un prodotto bisogna agire in modo sistematico[G], fornendo per esempio 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

Le attività del processo sono principalmente tre:

Quality Control (QC) : rappresenta tutti i controlli di qualità *product-oriented* che iniziano insieme all'inizio effettivo di produzione di documenti o codice;

Quality Assurance (QA) : rappresenta tutti gli accertamenti preventivi *process-oriented* che valutano il way of working[G] adottato, senza rallentare i processi di sviluppo;

Quality Planning (QP) : rappresenta una visione generale sugli obiettivi di qulità e sulle risorse necessarie a conseguirli. Nel nostro caso questo corrisponde alla redazione del *Piano di Qualifica .*

Nello specifico 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 v 1.0.0*.

3.3.5 Strumenti utili

Per le attività del processo di gestione della qualità gli strumenti sono stati scelti in base....

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 atomatici o condotta a mano (desk check[G]) con, in questo caso, due possibili metodi di lettura:

Tabella 1: 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.

3.4.5 Verifica del codice

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

Tabella 2: Analisi del codice

Analisi	Obiettivo	Attori	Esempi
Statica	Verificare che siano rispettate le regole di buona programmazione preimposte dal gruppo.	Verificatori Programmatori	Analisi di flusso dei dati, verifica formale del codice ecc.
Dinamica	Trovare bug e errori eseguendo il prodotto software.	Verificatori Programmatori	Test di unità, di integrazione, di sistema, di regressione, di accettazione.

3.4.5.1 Analisi di flusso dei dati

Tipo di analisi statica del codice che accerta che il programma in verifica non acceda in nessuna sua parte a variabili prive di valore, quindi non ancora sritte. Controlla l'assenza di dati globali.

3.4.5.2 Verifica formale

Tipo di 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.

3.4.5.3 Test

La scrittura dei test è la parte essenziale dell'analisi dinamica del codice. Tale attività ha lo scopo di rivelare al programmatore errori o bug[G] riscontrabili a run-time. Di conseguenza i test sono utili solo se falliscono.
L'esecuzione dei test deve essere ripetibile. Ottima pratica è renderla il più possibile automatica.
L'obiettivo del gruppo è rispettare gli standard di qualità dettati nel *Piano di Qualifica* .
I test si distinguono in:

Test di unità (TU) : verificano l'unità, ossia la più piccola parte di codice verificabile indipendentemente dalle altre (singolo metodo o singola classe);

Test di integrazione (TI) : verificano le componenti che compongono il sistema, rilevando errori in fase di progettazione;

Test di sistema (TS) : verificano il comportamento dell'intero sistema, controllando se rispetta le specifiche tecniche;

Test di regressione (TR) : verificano che nuove modifiche apportate non abbiano creato errori in altri parti del programma;

Test di accettazione (TA) : verificano errori a livello di UI[G], relativi a requisiti o casi d'uso concordati con il proponente.

3.4.6 Strumenti

Per il processo di verifica applicato ai documenti scritti in LaTeX il gruppo ha deciso di utilizzare il correttore automatico dell'editor TeXmaker, che individua le parole grammaticalmente scorrette sottolineandole in rosso. I verificatori, attraverso i metodi di lettura sopra citati, dovono occuparsi di rilevare possibili ripetizioni, errori di lessico o sintassi.

Per il processo di verifica applicato al codice...

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].

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Scopo

In questa sezione vengono esposte le modalità di coordinamento adottate dal gruppo che regolano gli incontri(interni o esterni) e le comunicazioni.

- **comunicazione**: interna con i membri del gruppo, quella esterna con l'azienda;
- **incontri**: incontri interni con in membri del gruppo, esterno con l'azienda.

4.1.2 Aspettative

Le attese, riguardo al processo in questione sono i seguenti:

- ottenere una pianificazione ragionevole delle attività da seguire;
- coordinamento dell'attività del gruppo, assegnando loro i ruoli, i compiti e semplificando la comunicazione tra loro;
- adoperare processi per 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 .

4.2 Ruoli di Progetto

Ogni membro del gruppo deve, a turno ricoprire almeno una volta ciascun ruolo di progetto che corrisponde alle figure aziendali. I ruoli che ogni membro del gruppo è tenuto a rappresentare sono descritti di seguito.

4.2.1 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,

4.2.2 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;

4.2.3 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 la sua complessità .

4.2.4 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.

4.2.5 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.

4.2.6 Verificatore

Il verificatore è presente durante tutta l'attività del progetto, deve controllare il lavoro svolto dai membro del gruppo. Il verificatore deve:

- controllare i prodotti in fase di revisione, utilizzando le tecniche e gli strumenti definiti nelle Norme di Progetto;
- evidenziare gli errori e segnalarli all'autore del prodotto in questione.

4.2.7 Formazione

La formazione di ogni membro del gruppo avviene attraverso studio autonomo delle tecnologie proposte dal proponente in occasione della presentazione del capitolato e incontri.

4.3 Procedure

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

- **comunicazione interna**: coinvolgeranno tutti i membri del team;
- **comunicazione esterna**: avverrà con il proponente e committente.

4.3.1 Gestione delle comunicazioni

4.3.1.1 Comunicazioni interne

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

4.3.1.2 Comunicazioni esterne

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

- tramite la posta elettronica, dove viene utilizzato il seguente indirizzo **codebusterswe@gmail.com**;
- attraverso skype per colloqui con azienda Zucchetti.

4.3.2 Gestione degli incontri

4.3.2.1 Incontri interni

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

4.3.2.2 Verbali di riunioni interni

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

4.3.2.3 Incontri esterni

Il responsabile del progetto organizza gli incontri esterni con il proponente. Il proponente o committente 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. L'incontri esterno avvengo tra i membri del gruppo e il proponente e viene riportato nel verbale esterno corrispondente.

4.3.2.4 Verbali di riunioni esterni

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

4.3.3 Gestione degli strumenti e di coordinamento

4.3.3.1 Tickecting

Il tickecting permette ai membri del gruppo di rimanere aggiornati sullo stato delle attività in corso. Attraverso il quale il responsabile assegna compiti ai membri del gruppo e controlla l'andamento dei task^{G} assegnati. Lo strumento di ticketing scelto è **Planner**: si tratta di un'applicazione multipiattaforma^{G}, in cui sono visibili a tutti i membri del team lo stato di avanzamento dei compiti assegnati e con la possibilità di aggiungere nuovi.

4.3.4 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 sono:

- individuazione nuovi problemi e monitorare quelli già presenti;
- menzionare i rischi individuati nel *Piano di Progetto*;
- ridefinizione le strategie dei rischi in caso di necessità.

4.3.4.1 Codifica dei rischi

I rischi sono codificate nel modo seguente:

- **RT**: rischi tecnologici
- **RO**: rischi organizzativi
- **RI**: rischi interpersonali

4.3.5 Strumenti

Nel durante dello sviluppo del progetto, il gruppo utilizzerà i seguenti strumenti:

- **Telegram**^{G}: un'applicazione di messaggistica per la comunicazione rapida e gestione del gruppo;
- **Github**^{G}: permette la condivisione in remoto di tutti i file del progetto e versionamento;
- **Git**^{G}: il sistema di controllo di versioni;
- **Discord**: un'applicazione multipiattaforma utilizzata per le riunioni interni;
- **Planner**: è una piattaforma per la gestione dei compiti assegnati;
- **Skype**: un'applicazione consente di effettuare le videoconferenze, utilizzato per comunicazione con il proponente;
- **Google Drive**: server per la condivisione rapida delle documentazioni che riguardano l'attività del gruppo.