Norme di Progetto

FoxyByte - Guida Michelin @ social

foxybyte.swe@gmail.com

Informazioni sul documento

Versione	2.1.0
Redazione	Hida Denisa
	Fincato Alessandro
	Ferrari Gianluca
Verifica	Biasotto Luca
	Uderzo Marco
Responsabile	Bosinceanu Ecaterina
Uso	Interno
Distribuzione	Vardanega Tullio
	Cardin Riccardo
	FoxyByte

Descrizione

Il presente documento espone le norme, gli strumenti e le convenzioni a cui ogni componente del gruppo *FoxyByte* dovrà attenersi drante tutto l'intero sviluppo del progetto *Guida Michelin @ social*

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
v2.1.0	2022-09-07	Uderzo Marco	*Verificatore*	Verifica aggiornamenti del documento
v2.0.2	2022-09-05	Fincato Alessandro	*Amministratore*	Aggiunte norme sezione §2.2.4.3
v2.0.1	2022-08-19	Fincato Alessandro	*Amministratore*	Modifica sezioni §2, §3 e §4
v2.0.0	2022-08-12	Bosinceanu Ecaterina	*Responsabile*	Accettazione
v1.1.0	2022-08-11	Udero Marco	*Verificatore*	Verifica documento
v1.0.1	2022-08-10	Fincato Alessando	*Amministratore*	Aggiornamento sezione §4.1 e §4.2
v1.0.0	2022-07-10	Bosinceanu Ecaterina	*Responsabile*	Accettazione e rilascio ufficiale
v0.8.1	2022-07-08	Fincato Alessandro	*Amministratore*	Correzione degli errori
v0.8.0	2022-07-06	Luca Biasotto	*Verificatore*	Verifica complessiva
v0.7.2	2022-07-02	Denisa Hida	*Amministratore*	Aggiunti strumenti sezione §4
v0.7.2	2022-07-01	Gianluca Ferrari	*Amministratore*	Aggiornamento sezione §3.2 e §3.3
v0.7.1	2022-06-24	Fincato Alessandro	*Amministratore*	Aggiornamento ativitá sezione §2
v0.7.0	2022-06-16	Marco Uderzo	*Verificatore*	Verifica generale
v0.6.3	2022-06-10	Fincato Alessandro	*Amministratore*	Sezione §1
v0.6.2	2022-06-04	Gianluca Ferrari	*Amministratore*	Aggiunti tutti gli strumenti
v0.6.1	2022-05-31	Fincato Alessandro	*Amministratore*	Aggiornamento sezione §2, aggiunti gli strumenti
v0.6.0	2022-05-28	Luca Biasotto	*Verificatore*	Verifica generale

Versione	Data	Autore	Ruolo	Descrizione
v0.5.4	2022-05-23	Gianluca Ferrari	*Amministratore*	Sezioni §3.4 e §3.5
v0.5.3	2022-05-20	Gianluca Ferrari	*Amministratore*	Appendice §B
v0.5.2	2022-05-18	Denisa Hida	*Amministratore*	Appendice §A
v0.5.1	2022-05-17	Gianluca Ferrari	*Amministratore*	Correzioni varie
v0.5.0	2022-05-15	Marco Uderzo	*Verificatore*	Verifica generale
v0.1.3	2022-05-08	Denisa Hida	*Amministratore*	Completamento sezione §4
v0.1.2	2022-05-03	Gianluca Ferrari	*Amministratore*	Sezioni §3.2 e §3.3
v0.1.1	2022-05-01	Denisa Hida	*Amministratore*	Correzione errori
v0.1.0	2022-04-25	Luca Biasotto	*Verificatore*	Verifica generale
v0.0.5	2022-04-22	Denisa Hida	*Amministratore*	Sezione §4.1
v0.0.4	2022-04-18	Fincato Alessandro	*Amministratore*	Sezione §2.2
v0.0.3	2022-04-17	Ferrari Gianluca	*Amministratore*	Sezione §3.1
v0.0.2	2022-04-15	Fincato Alessando	*Amministratore*	Sezione §2.1
v0.0.1	2022-04-10	Ecaterina Bosinceanu	*Responsabile*	Creazione e definizione struttura del documento

Indice

1 Introduzione **6**
1.1 Scopo del documento . . . 6
1.2 Scopo del prodotto . . . 6
1.3 Maturità del Documento . . . 6
1.4 Riferimenti . . . 6
1.4.1 Normativi . . . 6
1.4.2 Informativi . . . 6

2 Processi primari **7**
2.1 Processo di Fornitura . . . 7
2.1.1 Scopo . . . 7
2.1.2 Aspettative . . . 7
2.1.3 Descrizione . . . 7
2.1.4 Attività . . . 7
2.1.4.1 Ricerca delle tecnologie . . . 7
2.1.4.2 Normazione . . . 7
2.1.4.3 Piano di Progetto . . . 8
2.1.4.4 Piano di Qualifica . . . 8
2.1.4.5 Preparazione in vista della revisione . . . 8
2.1.5 Strumenti . . . 9
2.1.5.1 Google Docs . . . 9
2.1.5.2 Google Sheets . . . 9
2.1.5.3 Online Gantt . . . 9
2.2 Processo di Sviluppo . . . 9
2.2.1 Scopo . . . 9
2.2.2 Aspettative . . . 9
2.2.3 Descrizione . . . 9
2.2.4 Attività . . . 9
2.2.4.1 Analisi dei Requisiti . . . 9
2.2.4.1.1 Denominazione dei requisiti . . . 9
2.2.4.1.2 Casi d'uso . . . 10
2.2.4.1.3 Metriche . . . 11
2.2.4.2 Progettazione . . . 11
2.2.4.2.1 Obiettivi . . . 11
2.2.4.2.2 Descrizione . . . 11
2.2.4.2.3 Diagrammi UML . . . 11
2.2.4.3 Codifica . . . 13
2.2.4.3.1 Aspettative . . . 14
2.2.4.3.2 Descrizione . . . 14
2.2.4.3.3 Stile di codifica . . . 14
2.2.4.3.4 Metriche . . . 15
2.2.5 Strumenti . . . 16
2.2.5.1 Draw.io . . . 16
2.2.5.2 Visual Studio Code . . . 16

3 Processi di supporto **17**
3.1 Processo di documentazione . . . 17
3.1.1 Scopo . . . 17
3.1.2 Descrizione . . . 17
3.1.3 Aspettative . . . 17
3.1.4 Ciclo di vita del documento . . . 17
3.1.5 Suddivisione dei documenti . . . 17
3.1.6 Template . . . 17

3.1.7 Struttura dei documenti . . . 17
3.1.7.1 Frontespizio . . . 17
3.1.7.2 Registro delle modifiche . . . 18
3.1.7.3 Indice ed elenchi . . . 18
3.1.7.4 Contenuto . . . 18
3.1.7.5 Verbali . . . 19
3.1.8 Norme tipografiche . . . 19
3.1.8.1 Convenzioni di denominazione . . . 19
3.1.8.2 Stile del testo . . . 19
3.1.8.3 Elenchi . . . 19
3.1.8.4 Formato di data e ora . . . 20
3.1.8.5 Elementi grafici . . . 20
3.1.8.6 Sigle . . . 20
3.1.9 Glossario . . . 21
3.1.10 Strumenti utilizzati . . . 21
3.1.11 Strumenti e tecnologie . . . 21
3.1.12 Metriche . . . 21
3.2 Gestione della configurazione . . . 21
3.2.1 Descrizione . . . 21
3.2.2 Versionamento . . . 22
3.2.3 Gestione delle repository . . . 22
3.2.4 Gestione del lavoro e sincronizzazione . . . 22
3.2.4.1 Branch e Issues . . . 22
3.2.4.2 Pullrequests . . . 22
3.3 Gestione della qualità . . . 22
3.3.1 Descrizione . . . 22
3.3.2 Scopo . . . 22
3.3.3 Ciclo di Deming . . . 23
3.3.4 Denominazione delle metriche . . . 23
3.3.5 Denominazione degli obiettivi . . . 24
3.4 Verifica . . . 25
3.4.1 Scopo . . . 25
3.4.2 Descrizione . . . 25
3.4.3 Analisi statica . . . 25
3.4.4 Analisi dinamica . . . 25
3.4.4.1 Test di unità . . . 25
3.4.4.2 Test di integrazione . . . 25
3.4.4.3 Test di sistema . . . 26
3.4.4.4 Test di regressione . . . 26
3.4.4.5 Test di accettazione . . . 26
3.4.5 Strumenti . . . 26
3.4.6 Codice identificativo dei test . . . 26
3.4.7 Metriche . . . 26
3.5 Validazione . . . 27
3.5.1 Descrizione . . . 27
3.5.2 Scopo . . . 27

4 Processi organizzativi **28**
4.1 Gestione dei processi . . . 28
4.1.1 Scopo . . . 28
4.1.2 Aspettative . . . 28
4.1.3 Descrizione . . . 28
4.1.4 Pianificazione . . . 28
4.1.4.1 Ruoli di progetto . . . 28
4.1.4.1.1 Responsabile di progetto . . . 29

4.1.4.1.2 Amministratore di Progetto . . . 29
4.1.4.1.3 Analista . . . 29
4.1.4.1.4 Progettista . . . 29
4.1.4.1.5 Programmatore . . . 30
4.1.4.1.6 Verificatore . . . 30
4.1.4.2 Rischi . . . 30
4.1.4.2.1 Gestione dei rischi . . . 30
4.1.4.2.2 Codifica dei rischi . . . 30
4.1.4.3 Metriche . . . 31
4.1.5 Coordinamento . . . 31
4.1.5.1 Gestione delle comunicazioni . . . 31
4.1.5.1.1 Comunicazioni interne . . . 31
4.1.5.1.2 Comunicazioni esterne . . . 32
4.1.5.2 Gestione delle riunioni . . . 32
4.1.5.2.1 Riunioni interne . . . 32
4.1.5.2.2 Riunioni esterne . . . 32
4.1.5.2.3 Verbali delle riunioni . . . 33
4.1.5.3 Ticketing . . . 33
4.2 Strumenti . . . 33
4.3 Formazione . . . 33

A Standard di qualità **34**
A.1 ISO/IEC 9126 . . . 34
A.1.1 Modello della qualità del software . . . 34
A.1.1.1 Funzionalità . . . 34
A.1.1.2 Affidabilità . . . 34
A.1.1.3 Usabilità . . . 35
A.1.1.4 Efficienza . . . 35
A.1.1.5 Manutenibilità . . . 35
A.1.1.6 Portabiltà . . . 36
A.1.2 Metriche per la qualità interna . . . 36
A.1.3 Metriche per la qualità esterna . . . 36
A.1.4 Metriche per la qualità in uso . . . 36

B ISO/IEC 15504 - SPICE **37**
B.1 Introduzione . . . 37
B.2 Classificazione dei processi . . . 37
B.3 Livello di Capability . . . 37
B.3.1 Concetto di Capability . . . 37
B.3.2 Livelli di Capability . . . 37

Elenco delle tabelle

1 Metriche di qualità . . . 23
2 Obiettivi di qualità . . . 24

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Il presente documento ha l'obiettivo di mettere in chiaro le *best practice*$_G$ e il *way of working*$_G$ che ogni componente del gruppo è tenuto a rispettare durante tutto lo svolgimento del progetto. In questo modo sarà possibile cercare di garantire omogeneità e coesione in ogni aspetto del suddetto.

1.2 Scopo del prodotto

I social nettwork stanno diventando sempre di più un luogo virtuale dove vengono condivise le proprie esperienze e le proprie idee. Queste testimonianze hanno un ruolo importante in quanto permettono di avere sia un esempio dell'esperienza per i possibili futuri clienti che di avere degli spunti di miglioramento per chi, invece, offre il prodotto.
Il capitolato C4, *Guida Michelin @ social*, pone l'obiettivo di raccogliere informazioni da storie e post di *Instagram*, incrociandone i dati e le recenzioni, in modo da creare una piattaforma che svolga il ruolo di *guida Michelin*$_G$. Questa guida deve inoltre permettere all'utente di:

- Creare una mappa di interesse;
- Indicare persone da seguire per la creazione di tale guida;
- Specificare un luogo dal quale monitorare le recensioni.

Tale applicazione sarà fruibile dall'utente finale tramite una *Webapp*$_G$.

1.3 Maturità del Documento

Il presente documento verrà redatto con un metodo incrementale al fine di poter trattare rapidamente, sulla base di decisioni concordate dall'intero gruppo, nuove questioni ed eventuali accorgimenti. Per cui non è da intendersi al pari un documento completo.

1.4 Riferimenti

1.4.1 Normativi

- Capitolato d'appalto C4-Guida Michelin @ social:
  https://www.math.unipd.it/~tullio/IS-1/2021/Progetto/C4p.pdf

1.4.2 Informativi

- Standard ISO/IEC 12207:1995:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- Standard ISO/IEC 9126:
  http://www.colonese.it/00-Manuali_Pubblicatii/07-ISO-IEC9126_v2.pdf
- Standard ISO/IEC 15504 - SPICE:
  https://en.wikipedia.org/wiki/ISO/IEC_15504

2 Processi primari

2.1 Processo di Fornitura

2.1.1 Scopo

Il processo di Fornitura ha lo scopo di definire le procedure e le risorse necessarie allo svolgimento del progetto. Il processo verrà avviato una volta ottenuto l'appalto e avrà il fine di soddisfare ciascuna delle richieste che verranno definite mediante accordi con il *proponente*$_{\text{G}}$ *Zero12*. Tale processo è composto dalle seguenti fasi:

- avvio;
- studio delle tecnologie e delle capacità per rispondere alle richieste;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- verifica e valutazione;
- rilascio e completamento.

2.1.2 Aspettative

Durante il corso dell'intero progetto, il gruppo ha intenzione di instaurare e mantenere un costante dialogo con il *proponente*$_{\text{G}}$ *Zero12* al fine di avere un rapporto collaborativo riguardo a:

- Determinare aspetti chiave, requisiti e vincoli;
- Promuovere una verifica continua del prodotto, concordandone la qualifica;
- Chiarire eventuali dubbi emersi.

2.1.3 Descrizione

La sezione corrente (§2.1) ha l'obiettivo di trattare le principali attività e le norme che il gruppo si impegna ad attuare e rispettare in tutte le fasi di progettazione, sviluppo e consegna del prodotto per il *proponente*$_{\text{G}}$ *Zero12*.

2.1.4 Attività

2.1.4.1 Ricerca delle tecnologie

In quest'attività ogni componente del gruppo approfondisce autonomamente la propria conoscenza riguardo le tecnologie ed i framework utilizzati, in modo da poter avere delle discussioni riguardo i migliori metodi di approccio alla progettazione e pianificazione del prodotto.
Questo permette, a seguito delle decisioni prese dall'intero gruppo, di condividere la conoscenza all'intero gruppo, facilitando l'apprendimento degli aspetti che solo alcuni componenti hanno approfondito.

2.1.4.2 Normazione

Tale attività prevede il costante incremento del documento corrente, con nuove regole o aggiornando quelle già definite, ogni qual volta lo si ritenga necessario.

2.1.4.3 Piano di Progetto

In tale attività *Responsabile* ed *Amministratori* hanno l'obiettivo di definire tutti gli aspetti di pianificazione del progetto, ai quali i componenti del gruppo dovranno fare riferimento durante l'intero sviluppo. Il prodotto principale atteso è il *Piano di Progetto*, redatto definendo la pianificazione delle attività di processo durante le diverse fasi di sviluppo e la relativa organizzazione delle risorse. Nello specifico, tale documento espone:

- **Analisi dei rischi**: viene fatta un'analisi dei rischi che possono avvenire durante lo sviluppo del progetto. Per ognuno di questi rischi vengono inoltre definite le tecniche e le misure di monitoraggio e mitigazione, oltre che alla probabilità di avvenimento e la relativa gravità;
- **Modello di sviluppo**: descrive il *modello di sviluppo*$_G$ scelto ed adottato dal gruppo;
- **Pianificazione**: seguendo il metodo di sviluppo deciso, viene svolta una pianificazione delle diverse fasi di lavoro fino al completamento dello sviluppo del progetto, definendone anche delle scadenze temporali;
- **Preventivo e consuntivo**: basandosi su quanto deciso nella pianificazione, viene una stima di lavoro per ciascuna fase per poi proporre un preventivo finale. In seguito, per ogni periodo, verrà tracciato un consuntivo relativo all'andamento rispetto a quanto preventivato;
- **Attualizzazione dei rischi**: descrive i rischi che si sono effettivamente presentati durante il progetto e la soluzione che il gruppo ha deciso di adottare per risolverli.

2.1.4.4 Piano di Qualifica

In questa attività i *Verificatori* avranno il compito di decidere le strategie da utilizzare per la *verifica*$_G$ e la *validazione*$_G$ del materiale prodotto e dei processi attuati, al fine di poterne garantirne la *qualità*$_G$. Il principale prodotto atteso è il *Piano di Qualifica*, un documento contenente le metriche ed i test che compongono il "cruscotto" di controllo della qualità del prodotto e de processi. Nello specifico, quest'ultimo espone:

- **Qualità di processo**: sono individuati i processi e le attività principali, per questi vengono escogitate strategie e metriche per controllarne quantitativamente la qualità, in modo da poterne comprendere l'andamento durante l'intero progetto e perseguirne un miglioramento continuo;
- **Qualità di prodotto**: sono identificati gli attributi più rilevanti con gli obiettivi da raggiungere. Per fare ciò vengono definite delle metriche per misurarli;
- **Specifica dei test**: definisce i vari test, basandosi sui requisiti analizzati, che saranno utilizzati per verificare il corretto funzionamento del prodotto rispetto alle aspettative;
- **Resoconto della verifica**: alla fine di ogni periodo viene eseguita un analisi delle metriche precedentemente definite, in modo da capire il livello di qualità attuale del progetto, ovvero sia quella del prodotto che quella del *way of working*$_G$ del gruppo.

2.1.4.5 Preparazione in vista della revisione

Questa attività avverrà in vicinanza alle revisioni fissate e consiste nella preparazione del materiale necessario ad ognuna di queste per il loro buon superamento. Tale materiale consiste in:

- Documenti interni ed esterni;
- Diagrammi e codice necessario;
- Diapositive di esposizione.

2.1.5 Strumenti

2.1.5.1 Google Docs

Strumento utile per la creazione veloce di documenti di testo, utili soprattutto per un uso temporaneo durante riunioni interne oppure esterne.

2.1.5.2 Google Sheets

Strumento utile per la creazione fogli elettronici, utili soprattutto per ottenere eventuali diagrammi, tabelle e grafici e per eseguire dei calcoli.

2.1.5.3 Online Gantt Per assistere i responsabili di progetto, durante la pianificazione, nell'assegnazione delle risorse e nella decisione delle scadenze da rispettare tramite la creazione di *diagrammi di Gantt*$_G$ è stato utilizzato Online Gantt, un software online che ne permette la creazione facile e veloce.

2.2 Processo di Sviluppo

2.2.1 Scopo

Il processo di sviluppo contiene compiti e attività da svolgere al fine di realizzare il prodotto finale richiesto e concordato con il *proponente*$_G$.

2.2.2 Aspettative

Le aspettative del gruppo sull'applicazione del processo di sviluppo sono:

- Determinazione dei vincoli tecnologici e di design;
- Determinazione degli obiettivi di sviluppo;
- Realizzazione di un prodotto che superi i test di *qualità*$_G$ e che sia conforme alle richieste del *proponente*$_G$.

2.2.3 Descrizione

La sezione corrente (§2.2) tratta le principali attività e norme che il gruppo si impegna ad attuare e rspettare in tutte le fasi di progettazione, sviluppo e consegna del prodotto per il *proponente*$_G$ *Zero12*.

2.2.4 Attività

2.2.4.1 Analisi dei Requisiti

In quest'attività gli *Analisti* si occuperanno di redigere l'*Analisi dei Requisiti* composto da:

- Descrizione generale del prodotto;
- Modellazione concettuale del sistema mediante la definizione dei casi d'uso;
- Definizione e classificazione dei requisiti individuati.

2.2.4.1.1 Denominazione dei requisiti

Ogni requisito che verrà individuato durante l'analisi sarà identificato mediante un codice univoco definito come:

R[classificazione][tipo][numero]

- **R:** si riferisce a requisito;
- **Classificazinoe:** indica l'importanza del requisito, può essere:

  - **O**: obbligatorio;
  - **D**: desiderabile.

- **Tipo:** definisce il tipo del requisito, questo può essere:
  - **F:** funzionalità che deve avere il sistema software;
  - **Q:** qualità rispetto delle tecniche apposite.

- **Numero:** indica il numero univoco del requisito. Un requisito può avere dei sottocasi nel caso in cui vi siano degli ulteriori requisiti affini al principale, in questo caso i sottocasi vengono indicati suddividendo con un punto (e.g. requisito 1 è il principale, i sottocasi sono: requisito 1.1, requisito 1.2, ecc) per un massimo di tre livelli della gerarchia (e.g. requisito 1.1.1).

2.2.4.1.2 Casi d'uso

Un *caso d'uso*$_{\text{G}}$ permette di identificare i requisiti funzionali del prodotto descrivendo le interazioni tra il sistema e un utente esterno. Ognuno videne presentato attraverso:

- **Codice** che lo identifica univocamente;
- **Titolo** per la denominazione;
- ***Attori***$_{\text{G}}$ **primari** coinvolti;
- **Attori secondari** coinvolti, se presenti;
- **Descrizione** più dettagliata delle azioni compiute;
- **Scenario principale** ovvero l'elenco numerato delle azioni da compiere;
- **Estensioni** per rappresentare azioni o scenari straordinari (eccezioni o errori);
- **Precondizione** che rappresenta lo stato del sistema all'istante precedente;
- **Postcondizione** che rappresenta lo stato del sistema all'istante successivo.

Il codice che identifica univocamente ogni caso d'uso viene definito come:

UC[numero]

- **UC** sta per "Use Case" ovvero caso d'uso;
- **Numero:** si riferisce esclusivamente ad un caso. Tale numero è singolo se il caso d'uso è principale (e.g. 1 per il primo caso d'uso), altrimenti è composto da più numeri separati da un punto per intendere che sono dei sotto-casi del principale (e.g. 1.1 indica il primo sottocaso del caso d'uso principale 1). L'annidamento può essere al massimo di tre livelli (e.g. 1.1.1).

I casi d'uso vengono rappresentati mediante lo strumento *Draw.io*$_{\text{G}}$ e il linguaggio *UML2.0*$_{\text{G}}$. Nello specifico sono composti da:

- Attore;
- Caso d'uso;
- Relazione che può essere:
  - **Associazione:** sempre presente, rappresenta la relazione diretta tra attore e caso d'uso;
  - **Inclusione:** quando una funzionalità coinvolge più casi d'uso;
  - **Estensione:** quando le funzionalità di un caso d'uso vengono aumentate;
  - **Generalizzazione:** quando si vuole aggiungere o modificare le caratteristiche base di un caso d'uso o di un attore.

2.2.4.1.3 Metriche

Le metriche utilizzate per la valutazione dell'attività di analisi dei requisiti sono:

- **MPDS12 - Requirement coverage**: Rappresenta la compertura dei requisiti definiti dal team mediante l'*Analisi dei Requisiti*. Tale indice si misura tramite la formula:

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

  ovvero il rapporto in percentuale del nuemro dei requisiti rispettati fin'ora dal prodotto ($\mathrm{R_{RISP}}$) ed il numero di requisiti totali ($\mathrm{R_{TOT}}$);

- **MPDS13 - Requisiti obbligatori soddisfatti**: Rappresenta la compertura dei requisiti obbligatori definiti nell'*Analisi dei Requisiti*. Tale indice si misura tramite la formula:

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

  ovvero il rapporto in percentuale del nuemro dei requisiti obbligatori soddisfatti fin'ora dal prodotto ($\mathrm{R_{ROS}}$) ed il numero di requisiti obbligatori totali ($\mathrm{R_{ROT}}$).

2.2.4.2 Progettazione

Successiva all'analisi, l'attività di progettazione ha l'obiettivo di definire una soluzione del problema che sia soddisfacente per gli *stakeholder*$_{\mathrm{G}}$ e che rispetti i requisiti specificati nel documento *Analisi dei Requisiti*. Il procedimento da seguire durante la progettazione è quello di riunire tutte le singole parti e funzionalità trovate durante l'attività di analisi dei requisiti in modo da ricondursi ad un'unica possibile soluzione.

2.2.4.2.1 Obiettivi

Gli obiettivi principali dell'attività di progettazione sono:

- Definire l'architettura software che soddisfi i requisiti definiti e che sia comprensibile e modulare in modo da facilitarne la manutenzione;
- Organizzare i ruoli e le rispetttive responsabilità della realizzazione;
- Perseguire la correttezza per costruzione in modo da garantire la qualità del prodotto.

2.2.4.2.2 Descrizione

L'attività principale di progettazione è suddivisa in più sotto-attività:

- **Technology Baseline:** dove sarà svolta un'analisi delle tecnologie che verranno usate nello sviluppo del prodotto, il risultato di quest'analisi saranno il *PoC*$_{\mathrm{G}}$ e la *Technology Baseline*$_{\mathrm{G}}$ del progetto;
- **Progettazione Architetturale:** dove saranno definiti sia l'architettura del prodotto e delle sue componenti ad alto livello che i test di integrazione;
- **Product Baseline:** dove sarà definita in dettaglio l'architettura del prodotto e delle componenti, integrando diagrammi e test di unità per la verifica.

2.2.4.2.3 Diagrammi UML

Per tutti i tipi di diagrammi verranno utilizzati lo strumento *Draw.io*$_{\mathrm{G}}$ e il linguaggio *UML2.0*$_{\mathrm{G}}$. Nello specifico, i tipi di diagrammi che verranno usati sono:

- **Diagrammi delle classi:** permettono di descrivere ad alto livello gli oggetti facenti parte del sistema e le relazioni tra loro, sono composti da:
  - **Nome** dell'oggetto;

  - **Attributi**, ciascuno definito da:
    * Visibilità: pubblica (+), privata (-) oppure protetta (#);
    * Nome;
    * Tipo;
    * Default nel caso all'attributo venisse assegnato il valore di default.
  - **Operazioni fornite:** ciascuna definita da:
    * Visibilità (coerentemente agli attributi);
    * Nome;
    * Lista dei parametri;
    * Tipo di ritorno.

  Tra le classi ci sono delle relazioni, queste possono essere di:

  - **Dipendenza:** quando la classe utilizza una funzione della classe puntata, viene indicata tramite una freccia tratteggiata;
  - **Associazione:** quando gli oggetti della classe collaborano con gli oggetti della classe puntata per un periodo di tempo prolungato, viene indicato con una freccia piena;
  - **Aggregazione:** quando la classe possiede un'istanza condivisibile della classe puntata tra i suoi attributi, viene indicata tramite un rombo vuoto ed una linea continua;
  - **Composizione:** quando abbiamo una relazione simile all'aggregazione ma non vogliamo che l'istanza della classe puntata sia condivisa, viene indicata tramite un rombo pieno ed una linea continua;
  - **Generalizzazione:** quando la classe eredita attributi e metodi della classe puntata, viene indicata tramite una freccia vuota.

  Infine, verranno usati i seguenti costrutti per le classi in alternativa a quelle concrete:

  - **Classi astratte:** sono classi che non possono essere istanziate, presentano almeno un'operazione astratta e possono avere attributi. Queste classi saranno collegate da delle associazioni di generalizzazione a classi che ne implementeranno le operazioni astratte e ne erediteranno le funzionalità non astratte e gli attributi. Saranno simili alle classi concrete con l'eccezione che il nome e le funzioni astratte saranno scritte in corsivo;
  - **Interfacce:** classi prive di implementazione, ovvero elencano solamente le operazioni che dovranno essere implementate dalle classi che ne saranno collegate (tramite generalizzazione). Vengono rappresentate tramite un pallino vuoto.

- **Diagrammi di attività:** descrivono gli aspetti dinamici dei casi d'uso, supportando anche l'elaborazione parallela. Il diagramma è composto da:

  - **Nodo iniziale:** un pallino pieno, rappresenta l'inizio di un'attività;
  - **Pin:** un box, rappresentano delle risorse prodotte da delle azioni e consumate da altre;
  - **Fork:** un tratto marcato, rappresenta l'inizio di elaborazioni parallele;
  - **Join:** un tratto marcato, rappresenta un punto di sincronizzazione tra elaborazioni parallele;
  - **Branch:** un rombo vuoto, rappresenta la possibilità di intraprendere uno solo dei percorsi possibili;
  - **Merge:** un rombo vuoto, rappresenta l'unione di più possibili elaborazioni non parallele ma in alternativa tra esse;
  - **Nodi finali:** un pallino pieno circondato da un cerchio, rappresenta il termine di un'attività con successo;
  - **Nodo di fine flusso:** un cerchio con una croce, rappresenta il non termine dell'attività (viene usato in casi eccezionali).

Una attività può anche essere composta da delle sotto-attività, in questo caso la indicheremo con un simbolo a forca nell'attività principale e da un rettangolo esterno che contiene le sotto-attività. Altri costrutti più specifici che possono essere usati sono inoltre:

- **Swimlanes:** riquadri che permettono la divisione delle responsabilità delle azioni presenti nelle attività (un'azione può far capo a più swimlanes);
- **Segnali:** indicano eventi dipendenti da processi esterni, la situazione cambia a seconda dell'evento scatenante:
  - **Eventi esterni:** l'attesa dell'evento viene rappresentata come un rettangolo con un lato a punta entrante mentre l'evento generato è rappresentato da un rettangolo con un lato a punta uscente;
  - **Eventi temporali:** rappresentato tramite una clessidra indicandone il tempo di ripetizione. Può sostituire un nodo iniziale.

Abbiamo inoltre la possibilità di ripetere delle attività su di una collezione. In questo caso possiamo utilizzare una regione di espansione, indicata con un rettangolo tratteggiato ad angoli arrotondati e da rettangoli più piccoli posti ai lati di inizio e fine regione che rappresentano la lista degli elementi processati.

- **Diagrammi di sequenza:** permettono di esporre il comportamento a *run-time*$_G$ degli oggetti delle classi descrivendo la collaborazione di un gruppo di oggetti che devono implementare collettivamente un comportamento. Il diagramma va letto dall'alto verso il basso e contiene i seguenti costrutti:
  - **Partecipanti:** rappresentano le entità che detengono il flusso del caso d'uso, devono sempre rappresentare dei tipi concreti. Ogni partecipante inoltre ha una barra di attivazione (rettangolo verticale vuoto) che indica il momento nel quale tale partecipante è attivo;
  - **Messaggi (segnali):** rappresentano i dati scambiati e le operazioni chiamate tra i partecipanti, questi messaggi possono essere di diverso tipo:
    - **sincroni:** quando il chiamante rimane in attesa passiva della risposta, vengono rappresentati da una freccia piena con il nome del messaggio ed eventuali parametri;
    - **asincroni:** quando il chiamante non deve attendere la risposta del messaggio, vengono rappresentati da una freccia vuota con il nome del messaggio ed eventuali parametri;
    - **ritorno:** quando il metodo chiamato manda dei dati di ritorno al chiamante, viene rappresentato da una freccia tratteggiata;
    - **creazione:** quando viene creata una nuova entità da quella chiamante, viene rappresentata da una freccia tratteggiata con la notazione ≪create≫;
    - **distruzione:** quando viene distrutta un'entità da parte di quella chiamante, viene rappresentata da una freccia con la notazione ≪destroy≫.

Per descrivere cicli e condizioni, che modificano la semantica del tempo, si possono utilizzare dei *frame di interazione*. Questi vengono rappresentati mediante un rettangolo che circonda le entità e le rispettive barre di attivazione coinvolte, con un label che indica il tipo di frame. Questa modalità sarà usata solamente se strettamente necessario, in quanto non è adeguata. Per tale scopo è preferibile utilizzare dei diagrammi di attività oppure del pseudocodice.

2.2.4.3 Codifica

L'attività di codifica norma l'effettiva realizzazione del prodotto software seguendo quanto definito nell'attività di progettazione. In questa fase i *Programmatori* avranno il compito di concretizzare la soluzione attenendosi alle seguenti norme:

2.2.4.3.1 Aspettative

- Realizzare il prodotto software definito dalle precedenti attività;
- Definizione ed uso di norme e convenzioni per generare del codice uniforme e leggibile, agevolandone di conseguenza la verifica, la validazione e la manutenzione;
- Garantire la qualità finale del prodotto tramite l'uso di metriche.

2.2.4.3.2 Descrizione

Il codice prodotto deve perseguire gli obiettivi di qualità definiti nel *Piano di Qualifica* e far si che il prodotto finale rispetti i requisiti definiti nell'*Analisi dei Requisiti*. Di seguito verranno descritte le norme che regolano l'attività di codifica; queste differiscono per ogni linguaggio di programmaizone impiegato nel progetto.

2.2.4.3.3 Stile di codifica

Norme di buona programmazione
Le norme fondamentali e generali da rispettare nonostante il linguaggio di programmazione utilizzato sono:

- Mantenere il codice il più leggibile e comprensibile possibile, commentando le parti più complesse;
- Strutturare il codice in maniera tale da rispettare l'architettura che si è deciso di utilizzare;
- Massimizzare, per quanto possibile, l'information hiding.
- Cercare di mantenere un buon livello di coesione delle procedure, evitando di riunire più compiti su singole procedure.

Norme di programmazione per Python

- **Indentazione**: si dovrà prestare molta attenzione all'indentazione del codice, in modo da evitare eventuali errori dati da un uso scorretto di questa;
- **Lunghezza massima**: ogni riga di codice non dovrà essere eccessivamente lunga, in modo tale da non intaccare la leggibilità generale;
- **Commenti**: saranno presenti nel codice ove si considera utile descrivere ciò che si sta facendo, cercando così di aumentare la comprensibilità del codice;
- **Univocità dei nomi**: tutti i nomi dovranno essere univoci ed il più autoesplicativi possibile, evitando al tempo stesso nomi eccessivamente lunghi;

Convenzioni sulla nomenclatura: in generale per i nomi dei costrutti verrà utilizzata la nomenclatura "camelCase", nella quale ogni parola è in minuscolo ed ogni parola successiva alla prima inizia con una lettera maiuscola, inoltre:

- Variabili e metodi avranno l'iniziale del nome minuscola;
- Classi e file avranno l'iniziale del nome maiuscola;
- Cartelle e package avranno l'iniziale del nome maiuscola.

Norme di programmazione per JavaScript

- **Indentazione**: verrà utilizzata per mantenere il codice il più ordinato e leggibile possibile;
- **Lunghezza massima**: ogni riga di codice non dovrà essere eccessivamente lunga, in modo tale da non intaccare la leggibilità generale;
- **Commenti**: saranno presenti nel codice ove si considera utile descrivere ciò che si sta facendo, cercando così di aumentare la comprensibilità del codice;
- **Univocità dei nomi**: tutti i nomi dovranno essere univoci ed il più autoesplicativi possibile, evitando al tempo stesso nomi eccessivamente lunghi;

Convenzioni sulla nomenclatura: in generale per i nomi dei costrutti verrà utilizzata la nomenclatura "camelCase", nella quale ogni parola è in minuscolo ed ogni parola successiva alla prima inizia con una lettera maiuscola, inoltre:

- Variabili e metodi avranno l'iniziale del nome minuscola;
- File e classi, esclusi quelli *.css*, avranno l'iniziale del nome maiuscola;
- Cartelle e package avranno l'iniziale del nome minuscola.

2.2.4.3.4 Metriche

Le metriche utilizzate per valutare la valutare l'attività di codifica sono:

- **MPDS01 - Facilità di utilizzo**: La facilità di utilizzo è data dalla velocità con la quale l'utente riesce a reperire le informazioni che vuole. Questa è rappresentata dal numero di click necessari per arrivare al contenuto richiesto.
  La misurazione utilizzata è, uindi, il numero di click necessari per aprire la scheda del ristorante richiesto;
- **MPDS02 - Profondità gerarchia**: Il numero di livelli di una gerarchia serve per indicarne la profondità. In generale, più una gerarchia è profonda più può rilevarsi complessa da analizzare, ma allo stesso tempo può essere più facile modificarne solo alcune specifiche parti, se ben progettata. Per questo è utile trovare il giusto equilibrio in modo tale da renderne semplice sia l'analisi che la modifica, rendendone quindi più efficiente la manutenibilità;
- **MPDS03 - Parametri per metodo**: Il numero di parametri per metodo aiuta ad indicare il livello di facilità di comprensione di tale metodo. In generale, meno parametri ha una funzione più è semplice e intuitiva, di conseguenza più è semplice da modificare;
- **MPDS04 - Complessità ciclomatica**: Calcola il numero di percorsi linearmente indipendenti in una unità. Inizialmente pari a 1, viene incrementata da branch, salti e iterazioni. Dato il grafo G del flusso di esecuzione all'interno dell'unità, la complessità ciclomatica si calcola come:

  $$v(G) = e - n + p$$

  dove:

  - **e**: indica il numero degli archi del grafo, ovvero il flusso tra i comandi dell'unità;
  - **n**: indica il numero dei nodi del grafo, ovvero le espressioni o i comandi dell'unità;
  - **p**: indica il numero delle componenti connesse da ogni arco.
- **MPDS05 - Code smell**: Indica il numero difetti di programmazione riconosciuti nel codice sorgente del prodotto. I *code smell* rappresentano delle debolezze di progettazione che riducono la qualità del software, a prescindere dall'effettiva correttezza del suo funzionamento. La loro individuazione è un comune metodo euristico usato principalmente come guida per l'attività di refactoring;
- **MPDS06 - Facilità di comprensione**: Un codice comprensibile consente di capire fin da subito cosa fa, permettendo quindi una più facile gestione. Tale comprensibilità viene misurata tramite la formula:

  $$R = \frac{N_{\mathrm{LCOM}}}{N_{\mathrm{LCOD}}}$$

  che indica il rapporto tra le linee di commento ($\mathrm{N_{LCOM}}$) e quelle di codice ($\mathrm{N_{LCOD}}$).
- **MPDS08 - Presenza di vulnerabilità**: Indica il numero di vulnerabilità presenti nel codice che non son ancora state sistemate;

- **MPDS09 - Presenza di bug**: Indica il numero di bug presenti nel codice che non son ancora stati sistemati;
- **MPDS12 - Requirement coverage**: Rappresenta la compertura dei requisiti definiti dal team mediante l'*Analisi dei Requisiti*. Tale indice si misura tramite la formula:

$$\text{RC} = \frac{R_{\text{RISP}}}{R_{\text{TOT}}} \times 100$$

  ovvero il rapporto in percentuale del nuemro dei requisiti rispettati fin'ora dal prodotto (R_{RISP}) ed il numero di requisiti totali (R_{TOT});
- **MPDS13 - Requisiti obbligatori soddisfatti**: Rappresenta la compertura dei requisiti obbligatori definiti nell'*Analisi dei Requisiti*. Tale indice si misura tramite la formula:

$$\text{RC} = \frac{R_{\text{ROS}}}{R_{\text{ROT}}} \times 100$$

  ovvero il rapporto in percentuale del nuemro dei requisiti obbligatori soddisfatti fin'ora dal prodotto (R_{ROS}) ed il numero di requisiti obbligatori totali (R_{ROT}).

2.2.5 Strumenti

2.2.5.1 Draw.io

Per la produzione di diagrammi UML è stato utilizzato Draw.io date le agevolazioni che offre nello sviluppo di diagrammi durante la progettazione, in quanto permette una produzione veloce dei diagrammi e risulta uno strumento semplice da apprendere e da usare.

2.2.5.2 Visual Studio Code

Questo IDE offre compatibilità tra i sistemi Linux, Windows e macOS oltre a fornire molte funzionalità utili ed essere un potente editor per la stesura di codice.

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Scopo

In questa sezione vengono definite le regole e gli standard da seguire durante l'attività di documentazione, in modo da perseguire chiarezza e professionalità.

3.1.2 Descrizione

Di seguito vengono specificate le norme e le convenzioni per la stesura, verifica e approvazione della documentazione ufficiale interna ed esterna, che ogni membro del gruppo si impegna ad adottare.

3.1.3 Aspettative

Ci si aspetta di individuare e raccogliere le norme comuni per la stesura dei documenti ufficiali da fornire ai membri del gruppo, in modo da ottenere una documentazione il più coesa e coerente possibile.

3.1.4 Ciclo di vita del documento

Il ciclo di vita di ogni documento è composto da tre tappe fondamentali:

- **Stesura del documento:** il documento viene scritto da uno o più redattori adottando un metodo incrementale. Per facilitarne la stesura si parte da un template contenuto nella repository;
- **Verifica del documento:** ogni sezione del documento viene controllato da uno o più membri del gruppo (diversi dai redattori). Una volta verificato l'intero documento e apportate le modifiche necessarie, il documento viene considerato verificato;
- **Approvazione:** il *Responsabile di Progetto* stabilisce la validità del documento e lo approva per il rilascio.

3.1.5 Suddivisione dei documenti

Per migliorare l'organizzazione dei documenti, essi verranno suddivisi in due categorie:

- **Documenti interni:** documenti legati all'organizzazione e al metodo di lavoro del team. In questa categoria rientrano le *Norme di Progetto* e i *Verbali Interni*. L'uso è limitato ai membri interni del gruppo ed ai proponenti;
- **Documenti esterni:** documenti di interesse a committenti e proponente. In questa categoria rientrano: il *Piano di Progetto*, l'*Analisi dei Requisiti*, il *Piano di Qualifica* e i *Verbali Esterni*.

3.1.6 Template

Per la stesura dei documenti il gruppo ha deciso di adottare il linguaggio $\LaTeX_{\text{G}}$; sono stati definiti dei template per uniformare la struttura dei documenti e facilitarne la stesura.

3.1.7 Struttura dei documenti

3.1.7.1 Frontespizio

La prima pagina di ogni documento è strutturata nel seguente modo:

- **Logo:** il logo del gruppo al centro è il primo elemento del documento;
- **Titolo:** sotto al logo al centro si trova il titolo del documento;
- **Nome:** nome del gruppo seguito dal nome del progetto;

- **Recapito:** indirizzo di posta elettronica del gruppo;
- **Informazioni sul documento:** tabella continente:
  - **Versione:** indica la versione corrente;
  - **Redazione:** indica i membri del gruppo responsabili della stesura del documento;
  - **Verifica:** indica i membri che si sono occupati della verifica del documento;
  - **Responsabile:** indica il *Responsabile di Progetto* che ha approvato il documento;
  - **Uso:** indica se il documento è destinato all'uso interno o esterno;
  - **Distribuzione:** indica i destinatari del documento.
- **Descrizione:** posizionato in fondo alla pagina al centro, contiene una descrizione sintetica del contenuto del documento.

3.1.7.2 Registro delle modifiche

Il registro delle modifiche è una tabella che segue il frontespizio e tiene traccia delle modifiche significative apportate al documento durante le fasi del suo ciclo di vita. Ogni voce della tabella riporta:

- **Versione:** versione del documento dopo la modifica;
- **Data:** data in cui è stata apportata la modifica;
- **Autore:** autore della modifica;
- **Ruolo:** ruolo dell'autore della modifica;
- **Descrizione:** breve descrizione della modifica apportata.

3.1.7.3 Indice ed elenchi

Ogni documento presenta l'indice dei contenuti posizionato dopo il diario delle modifiche, che permette di orientarsi tra i contenuti e individuare la posizione delle varie sezioni.
Nel caso all'interno del documento siano presenti immagini e tabelle, l'indice sarà seguito da un elenco delle tabelle e uno delle immagini.

3.1.7.4 Contenuto

Tutte le pagine del documento, esclusa la prima, hanno i seguenti elementi:

- **Intestazione:**
  - in alto a sinistra si trova il logo del gruppo a colori;
  - in alto a destra è indicato il titolo della sezione corrente.
- **Piè di pagina:**
  - in basso a sinistra si trova il nome del documento;
  - in basso a destra è indicato il numero della pagina sul totale di pagine del documento.

Tra l'intestazione ed il piè di pagina si trova il contenuto della pagina.

3.1.7.5 Verbali

I verbali sono documenti informali e la loro struttura differisce dai normali documenti precedentemente descritti. Si presentano nel seguente modo:

- **Informazioni generali:** elenco delle informazioni generali dell'incontro che comprende:
  - data della riunione;
  - ora dell'inizio;
  - ora della fine;
  - luogo di incontro;
  - partecipanti;
  - redattore.
- **Ordine del giorno:** argomenti trattati durante l'incontro;
- **Resoconto:** esito della discussione dei punti inseriti nell'ordine del giorno;
- **Riepilogo:** tabella che riassume le decisioni prese dal gruppo durante l'incontro; ogni decisione ha un codice identificativo nel formato `V[I/E]_[YYYY]-[MM]-[DD].[X]` dove:
  - `V[I/E]`: indica se il verbale è interno (I) o esterno (E);
  - `[YYYY]-[MM]-[DD]`: indica la data del verbale e segue il formato dello standard ISO 8601;
  - `[X]`: indica il numero della decisione all'interno del verbale.

3.1.8 Norme tipografiche

Di seguito verranno esposte le norme tipografiche a cui ogni componente del gruppo dovrà sottostare per evitare incongruenze e tra i vari file e ottenere uno stile uniforme per tutti i documenti.

3.1.8.1 Convenzioni di denominazione

I nomi dei file e delle cartelle adottano la convenzione Snake Case, ovvero il nome verrà scritto interamente in minuscolo e, se composto la più parole, queste verranno separate da un *underscore*. Il nome del documento comprenderà anche la versione come nel seguente esempio: *nome_file_X.Y.Z*. Nel caso dei verbali, la nomenclatura sarà la seguente: `V[I/E]_[YYYY]-[MM]-[DD]`, dove si specifica se è interno o esterno, e la data esatta del verbale.

3.1.8.2 Stile del testo

Nei documenti vengono utilizzati i seguenti stili di testo:

- **Grassetto:** utilizzato nei titoli e nelle liste di definizione;
- **Corsivo:** utilizzato per i nomi propri, nomi dei documenti, termini del glossario, termini specifici, ruoli, linguaggi e tecnologie;
- **Maiuscolo:** utilizzato per gli acronimi, nomi propri, iniziale dei nomi dei documenti e dei ruoli.

3.1.8.3 Elenchi

Gli elenchi puntati seguono le seguenti norme:

- il simbolo che scandisce ogni elemento dell'elenco è il pallino (•); al secondo e terzo livello si trovano rispettivamente il doppio trattino e l'asterisco;
- le voci iniziano per lettera minuscola;
- ogni voce si conclude col punto e virgola eccetto l'ultima, che termina col punto;
- le liste del tipo "*Termine:descrizione*" presentano il termine in grassetto con la prima lettera in maiuscolo.

3.1.8.4 Formato di data e ora

Per indicare le date e gli orari si segue lo standard ISO 8601, dunque seguiranno il seguente formato:

- **Data:** [YYYY-MM-DD], dove YYYY indica l'anno, MM indica il mese e DD indica il giorno;
- **Ora:** [HH-MM], dove HH indica le ore e MM indica i minuti.

3.1.8.5 Elementi grafici

Vengono seguite le seguenti norme per utilizzare immagini, grafici e tabelle:

- **Immagini:** sono centrate e accompagnate da una didascalia;
- **Grafici UML:** usati per rappresentare i casi d'uso, i diagrammi delle classi e i diagrammi di sequenza, verranno inseriti come immagini e trattati come tali;
- **Tabelle:** sono centrate e ognuna è numerata e ha una didascalia, esclusa la tabella relativa al registro delle modifiche. Le righe delle tabelle hanno colori alternati per semplificarne la lettura, nello specifico:
  - L'intestazione ha caratteri neri su sfondo blu (*HEX #6699cc*);
  - Le righe pari hanno caratteri neri su sfondo azzurro chiaro (*HEX #f0ffff*);
  - Le righe dispari hanno caratteri neri su sfondo azzurro (*HEX #bcd4e6*).

  Sono stati scelti colori che creino contrasto col nero e permettano una facile lettura dei caratteri.

3.1.8.6 Sigle

Nella stesura della documentazione verranno utilizzate le seguenti sigle per favorire maggiore scorrevolezza:

- **Sigle relative ai ruoli:**
  - **Re:** *Responsabile*;
  - **Am:** *Amministratore*;
  - **An:** *Analista*;
  - **Pg:** *Progettista*;
  - **Pr:** *Programmatore*;
  - **Ve:** *Verificatore*.
- **Sigle relative ai documenti:**
  - **AdR:** *Analisi dei Requisiti*;
  - **NdP:** *Norme di Progetto*;
  - **PdQ:** *Piano di Qualifica*;
  - **PdP:** *Piano di Progetto*;
  - **VI:** *Verbale Interno*;
  - **VE:** *Verbale Esterno*.
- **Sigle relative alle revisioni:**
  - **RTB:** *Requirement and Technology Baseline*;
  - **PB:** *Product Baseline*;
  - **CA:** *Customer Acceptance*.

3.1.9 Glossario

I termini ambigui che necessitano di una spiegazione sono contrassegnati da una "**G**" a pendice alla loro prima occorrenza nella sezione. Tutti i termini da glossario sono riportati in ordine alfabetico nel documento *Glossario*.

3.1.10 Strumenti utilizzati

Di seguito verranno riportati gli strumenti utilizzati nella stesura dei documenti:

- *LaTeX*$_G$: linguaggio di markup utilizzato per la stesura della documentazione;
- *Google Drive*$_G$: usato per la condivisione di materiale utile al gruppo, stesura di bozze e appunti;
- *Draw.io*$_G$: usato per la creazione dei diagrammi di casi d'uso;
- *StarUML*$_G$: usato per la creazione dei diagrammi *UML*.

3.1.11 Strumenti e tecnologie

Il software di versionamento distribuito *Git*$_G$ verrà utilizzato per il versionamento del codice sorgente della documentazione e del software. L'utilizzo di *Git* è facilitato dalla sua integrazione con *GitHub*$_G$. Ogni membro del team è libero di scegliere il client.

3.1.12 Metriche

Le metriche utilizzate per valutare l'attività di documentazione sono:

- **MPDD01 - Indice di Gulpease**: L'indice di Gulpease è un indice che rappresenta il grado di leggibilità di un testo scritto in lingua italiana. La formula utilizzata la seguente:

$$GULP = 89 + \frac{300*(numero\ delle\ frasi) - 10*(numero\ delle\ lettere)}{numero\ delle\ parole}$$

  Questo indice considera due variabili linguistiche: la lunghezza delle parole e la lunghezza della frase rispetto al numero delle lettere. I risultati sono compresi tra 0 (leggibilità più bassa) a 100 (leggibilità più alta), in generale i testi con indice:

  - **GULP<80**: sono difficili da leggere per chi ha la licenza elementare;
  - **GULP<60**: sono difficili da leggere per chi ha la licenza media;
  - **GULP<40**: sono difficili da leggere per chi ha un diploma di scuola superiore.

3.2 Gestione della configurazione

3.2.1 Descrizione

Il processo di Gestione della configurazione ha lo scopo di gestire in modo ordinato e sistematico la produzione di documenti e codice. Un elemento che è soggetto a configurazione ha una denominazione e uno stato definiti. Quello che che effettivamente fa questo processo è raggruppare e organizzare sia gli strumenti necessari alla configurazione degli strumenti adoperati per la produzione di documenti, codice e diagrammi, sia quelli necessari al versionamento e al coordinamento del gruppo. Implementando questo processo è così possibile:

- rendere la produzione di codice e documentazione un'attività sistematica;
- gestire lo stato degli strumenti usati durante lo svolgimento del progetto in maniera uniforme;
- classificare i prodotti dei vari processi implementati.

3.2.2 Versionamento

Ogni modifica apportata ad un documento genera una nuova versione il cui codice è nel formato **[X].[Y].[Z]** dove:

- **X:** rappresenta la versione approvata dal *Responsabile*, l'unico autorizzato ad incrementarla;
- **Y:** rappresenta la versione approvata dal *Verificatore*, l'unico autorizzato ad incrementarla;
- **Z:** rappresenta la versione dell'ultima modifica.

Ogni parte del codice di versione inizia da 0 e torna a 0 ogni volta che la componente alla sua sinistra viene incrementata.

3.2.3 Gestione delle repository

Per migliorare l'organizzazione dei documenti e del codice, si è deciso di utilizzare tre *repository*$_G$:

- **C4:** *repository*$_G$ pubblica contenente tutti i documenti ed i verbali, approvati dal *Responsabile* in formato *.pdf*;
- **C4-private:** *repository*$_G$ privata per il versionamento intermedio dei documenti interni ed esterni. Contiene il template, i documenti in formato *.pdf* e *.tex*, immagini e loghi;
- **Progetto:** *repository*$_G$ pubblica per il versionamento del codice del prodotto Software.

Nelle prime fasi del progetto vi saranno delle *repository*$_G$ private per sviluppare i singoli servizi che andranno poi a comporre il prodotto finale, ovvero la *repository*$_G$ *Progetto*. Queste implementazioni sono da considerarsi delle versioni di prova e soluzioni ancora embrionali.

3.2.4 Gestione del lavoro e sincronizzazione

3.2.4.1 Branch e Issues

Per gestire nel miglior modo possibile il lavoro sarà necessario suddividerlo in compiti minori e semplici, ovvero i *task*$_G$, ognuno dei quali verrà assegnato ad un membro del gruppo; verranno create Issues per ogni *task*, e ad ogni *Issue*$_G$ corrisponderà un *branch*$_G$ secondario. In questo modo ogni membro lavorerà su branch diversi e le modifiche non creeranno conflitti sul master.

3.2.4.2 Pullrequests

Una volta completato il *task*$_G$, il membro del gruppo che ha lavorato su tale *task*$_G$ effettuerà una *pull request*$_G$. Il codice nel branch verrà verificato e verrà fatto il *merge*$_G$ nel branch principale nel caso la verifica abbia avuto esito positivo.

3.3 Gestione della qualità

3.3.1 Descrizione

In questo paragrafo verranno discusse le norme sulla gestione della qualità, ovvero le metriche e le modalità attuate per verificare la qualità dei prodotti e dei processi.

3.3.2 Scopo

Lo scopo è garantire che il prodotto finale rispetti i requisiti di qualità stabiliti, così da ottenere una documentazione e un prodotto software efficienti ed efficaci. Più nello specifico si vogliono raggiungere i seguenti obiettivi:

- soddisfare le richieste del *proponente*$_G$ rispettandone la qualità;
- ottenere una prova oggettiva e quantificabile della qualità del prodotto;
- raggiungere un'organizzazione delle attività e dei processi qualitativa.

3.3.3 Ciclo di Deming

Per mantenere un'alta qualità di lavoro si è stabilito l'utilizzo del *ciclo di Deming*$_G$ (o *PDCA*), uno schema iterativo di auto-miglioramento che consiste di quattro punti:

- **Plan:** individuare obiettivi di miglioramento;
- **Do:** eseguire ciò che si è pianificato;
- **Check:** verificare i risultati della fase precedente (do) e confrontarli con gli obiettivi individuati nella prima fase(plan);
- **Act:** se la fase precedente (check) ha dimostrato che il piano (plan) implementato nella seconda fase (do) è migliore rispetto agli standard precedentemente usati, allora questo piano diventa il nuovo standard. Altrimenti, rimarrà in uso il vecchio standard.

3.3.4 Denominazione delle metriche

La denominazione delle metriche segue il seguente formato

M[Categoria][TipoProdotto][Numero]

dove:

- **M:** indica che si tratta di una metrica di qualità;
- **[Categoria]:** indica a quale categoria appartiene la metrica, e può assumere i seguenti valori:
  - **PD** per indicare i prodotti;
  - **PC** per indicare i processi;
  - **TS** per indicare i test.
- **[TipoProdotto]:** indica se si riferisce a documenti o software ed è presente solo nel caso sia una metrica di prodotto; assume i valori:
  - **D** per indicare i documenti;
  - **S** per indicare i prodotti software.
- **Numero:** rappresenta il codice numerico identificativo della metrica, inizia da 1.

Di seguito abbiamo un elenco di tutte le metriche di qualità:

Tabella 1: Metriche di qualità

Codice metrica	Nome
MPC01	SPICE
MPC02	Budgeted cost of work scheduled
MPC03	Actual cost of work perfomed
MPC04	Budgeted cost of work performed
MPC05	Schedule Variance
MPC06	Budget Variance
MPDS01	Facilità di utilizzo
MPDS02	Profondità gerarchia
MPDS03	Parametri per metodo
MPDS04	Complessità ciclomatica

Metriche di qualità

Codice metrica	Nome
MPDS05	Code smell
MPDS06	Facilità di comprensione
MPDS07	Code Coverage
MPDS08	Presenza di vulnerabilità
MPDS09	Presenza di bug
MPDS10	Branch Coverage
MPDS11	Successo dei test
MPDS12	Requirement coverage
MPDS13	Requisiti obbligatori soddisfatti
MPDD01	Indice di Gulpease

3.3.5 Denominazione degli obiettivi

La denominazione degli obiettivi di qualità segue il seguente formato

O[Categoria][Prodotto][X]

dove:

- **O:** indica che si tratta di un obiettivo di qualità;
- **[Categoria]:** indica a quale categoria appartiene la metrica, e può assumere i seguenti valori:
  - **PD** per indicare i prodotti;
  - **PC** per indicare i processi.
- **[TipoProdotto]:** indica se si riferisce a documenti o software ed è presente solo nel caso sia una metrica di prodotto; assume i valori:
  - **D** per indicare i documenti;
  - **S** per indicare i prodotti software.
- **X:** rappresenta il codice numerico identificativo della metrica e inizia da 1.

Di seguito abbiamo un elenco di tutti gli obiettivi di qualità:

Codice obiettivo	Nome
OPC01	Miglioramento continuo
OPC02	Rispetto pianificazione
OPC03	Consumo delle risorse efficiente
OPDS01	Usabilità
OPDS02	Manutenibilità
OPDS03	Affidabilità
OPDS04	Funzionalità
OPDD01	Leggibilità dei documenti

Tabella 2: Obiettivi di qualità

3.4 Verifica

3.4.1 Scopo

Il processo di verifica ha l'obiettivo di individuare e correggere gli errori nei documenti e nelle componenti del prodotto software.

3.4.2 Descrizione

Si compone di due attività principali, analisi e test, che permettono di verificare che il prodotto ottenuto sia conforme alle aspettative, e individuare e correggere eventuali anomalie.

3.4.3 Analisi statica

L'analisi statica si applica a tutti i processi attivi nel progetto e si distinguono due tecniche principali:

- **Walkthrough:** consiste in una ricerca generale degli errori senza presupposti e si articola nelle seguenti attività:
  - pianificazione;
  - lettura;
  - discussione;
  - correzione degli errori.
- **Inspection:** consiste nell'eseguire una lettura mirata, alla ricerca di errori noti e si suddivide nelle seguenti attività:
  - pianificazione;
  - definizione di una lista di controllo;
  - lettura;
  - correzione degli errori.

3.4.4 Analisi dinamica

L'analisi dinamica è applicabile solo al prodotto software in quanto prevede l'esecuzione di *test*, cioè prove sul codice in esecuzione. Un test definito correttamente deve:

- essere ripetibile: dato un determinato input si deve ottenere sempre lo stesso output per ogni prova effettuata;
- specificare l'ambiente di esecuzione;
- identificare input e output richiesti;
- fornire informazioni utili sui risultati dell'esecuzione.

Esistono diverse categorie di test, ognuno con uno scopo e oggetto di verifica diverso.

3.4.4.1 Test di unità

I test di unità verificano la correttezza di una piccola parte di software testabile, chiamata unità, per stabilirne il corretto funzionamento rispetto alle attese. Le unità vengono testate con l'ausilio di $driver_G$ e $stub_G$ che simulano rispettivamente un'unità chiamante e un'unità chiamata.

3.4.4.2 Test di integrazione

I test d'integrazione rilevano difetti di progettazione e si applicano alle componenti specificate durante la progettazione architetturale; l'integrazione delle componenti costituisce il sistema completo.
L'ideale è fare tanti test di integrazione quante sono le interfacce nell'architettura del sistema: i test d'integrazione devono accertare che i dati scambiati attraverso ciascuna interfaccia siano conformi alla propria specifica.

3.4.4.3 Test di sistema

I test di sistema verificano il comportamento dell'intero sistema rispetto ai suoi requisiti.

3.4.4.4 Test di regressione

I test di regressione vengono eseguiti a seguito di un test fallito o di una correzione; rappresentano l'insieme di test necessari ad accertare che la correzione non causi errori nelle parti del sistema che dipendono da essa.

3.4.4.5 Test di accettazione

Il test di accettazione o collaudo è un'attività esterna, supervisionata dal *committente*$_{\text{G}}$ e consiste nel dimostrare il soddisfacimento dei requisiti. Al collaudo segue il rilascio del prodotto.

3.4.5 Strumenti

Non sono ancora stati individuati strumenti per la verifica del codice.

3.4.6 Codice identificativo dei test

I test vengono identificati dal codice

T[Tipo]-[ID]

dove:

- **T:** indica che si tratta di un test;
- **Tipo:** indica indica il tipo di test e può assumere i seguenti valori:
  - **U** per i test di unità;
  - **I** per i test d'integrazione;
  - **S** per i test di sistema;
  - **R** per i test di regressione;
  - **A** per i test di accettazione.
- **ID**: codice numerico progressivo che inizia da 1.

3.4.7 Metriche

Le metriche utilizzate per valutare l'attività di verifica sono:

- **MPDS07 - Code Coverage**: Indica la percentuale di codice eseguito durante i test. Del codice con un'alta percentuale di copertura ha più codice effettivo testato, per questo è più probabile che non contenga bug nascosti rispetto a del codice con una percentuale di copertura minore;
- **MPDS10 - Branch Coverage**: Simile al concetto di code coverage, indica in percentuale la copertura di tutti i branch che si presentano nel codice quando eseguito. Un branch è un intero ramo di esecuzione. Questo cambia a seconda dei risultati delle condizioni che trova durante l'esecuzione, compito dei test è anche quello di esplorare ogni possibile ramo di esecuzione in modo da poterne verificare la correttezza;
- **MPDS11 - Successo dei test**: Percentuale di successo dei test definiti dai programmatori, maggiore è tale percentuale più è probabile che il codice sia corretto e opresenti meno errori.

3.5 Validazione

3.5.1 Descrizione

Successivo al processo di verifica, durante tale processo il prodotto finale verrà preso in esame dal *Responsabile di progetto* che deciderà se il prodotto sia conforme ai requisiti del committente basandosi sui risultati ottenuti nei test. A tal proposito è necessario:

- identificare gli oggetti da validare;
- identificare e applicare una strategia di validazione riutilizzabile;
- valutare i risultati rispetto alle attese.

3.5.2 Scopo

Lo scopo del processo di validazione è accertare che il prodotto realizzato rispetti i requisiti e gli obiettivi prestabiliti. Una validazione positiva attesta che il prodotto finale soddisfi quanto stabilito con il proponente.

4 Processi organizzativi

4.1 Gestione dei processi

4.1.1 Scopo

Lo scopo del processo è assicurare un'efficace comunicazione tra i membri del gruppo, con gli esterni, ovvero *proponente*$_G$ e *committenti*$_G$, l'assegnazione e la gestione dei ruoli dei componenti, tramite la creazione di un documento chiamato *Piano di Progetto*.

4.1.2 Aspettative

Dalla gestione dei processi organizzativi ci si aspetta:

- la redazione del documento denominato *Piano di Progetto*;
- la definizione dei ruoli aziendali assunti dai membri del gruppo;
- la pianificazione dell'esecuzione dei compiti assegnati.

4.1.3 Descrizione

Le attività previste da tale processo sono raccolte nel *Piano di Progetto*. In dettaglio, viene trattata la gestione dei seguenti argomenti:

- Ruoli nel Progetto;
- Gestione delle Comunicazioni;
- Incontri;
- Strumenti di Coordinamento;
- Strumenti di Versionamento;
- Rischi.

4.1.4 Pianificazione

4.1.4.1 Ruoli di progetto

I componenti del gruppo ricopriranno i seguenti ruoli:

- Responsabile di Progetto;
- Amministratore di Progetto;
- Analista;
- Progettista;
- Programmatore;
- Verificatore.

Tali ruoli corrispondono alle omonime figure aziendali. Il calendario che sarà stilato permetterà ad ogni membro di ricoprire ogni ruolo per un'omogenea quantità di tempo, compatibilmente con i propri e altrui impegni personali.

4.1.4.1.1 Responsabile di progetto

Il *Responsabile di Progetto* è la figura professionale di riferimento sia per il *committente*$_G$ che per il *fornitore*$_G$, e fa da intermediario tra i due. Coordina l'intera struttura e organizza il lavoro, assumendosi la responsabilità delle scelte di gruppo. In particolare, il *Responsabile di Progetto*:

- si occupa del coordinamento dei membri del gruppo, dei loro ruoli durante il progetto e dei compiti che devono portare a termine;
- gestisce la pianificazione delle attività di progetto da svolgere e le relative scadenze da rispettare;
- è responsabile della stima dei costi e dell'analisi dei rischi;
- cura le relazioni che il gruppo intrattiene con i soggetti esterni;
- approva l'emissione della documentazione.

4.1.4.1.2 Amministratore di Progetto

E' la figura professionale incaricata del controllo e dell'amministrazione dell'ambiente di lavoro che il gruppo utilizza. Garantisce l'affidabilità e l'efficacia dei mezzi scelti dal gruppo, secondo l'idea che un ambiente di lavoro ben gestito, in termini di regole, strumenti e servizi, favorisca la produttività. In particolare, l'*Amministratore di Progetto* deve:

- amministrare le infrastrutture e i servizi per i processi di supporto;
- automatizzare l'ambiente di lavoro ove possibile;
- gestire il versionamento e la configurazione dei prodotti;
- gestire la documentazione di progetto, assicurandosi che venga corretta, verificata ed approvata;
- risolvere i problemi inerenti alla gestione dei processi;
- redigere ed attuare le norme e le procedure per la gestione della qualità.

4.1.4.1.3 Analista

L'*Analista* è la figura professionale che ha maggiori conoscenze circa il dominio applicativo del problema,e partecipa al progetto soprattutto nelle fasi iniziali, ad esempio nella stesura dell'*Analisi dei Requisiti*. Il suo compito è quello di individuare i punti chiave del problema in tutte le sue sfacettature. E' perciò una figura fondamentale al fine della buona riuscita del lavoro, perchè una mancata o erronea individuazione dei requisiti da soddisfare compromette l'attività di Progettazione. In particolare, l'*Analista*:

- studia il problema e il suo contesto applicativo;
- analizza le richieste, definisce la complessità ed estrapola i requisiti, studiando i bisogni impliciti ed espliciti;
- redige lo l'*Analisi dei Requisiti*.

4.1.4.1.4 Progettista

Il *Progettista* è la figura professionale che si occupa di sviluppare una soluzione che soddisfi i bisogni individuati, a partire dal lavoro svolto dall'*Analista*. In particolare, il *Progettista*:

- applica i principi dell'Ingegneria del Software per produrre un'architettura solida, consistente e coerente, ed effettua scelte tecniche per favorire efficacia, efficienza, flessibilità e riusabilità;
- produce una soluzione economica, ovvero che rientri nei costi stabiliti nel preventivo, e manutenibile;
- si adopera per la creazione di una struttura che soddisfi tutti i requisiti;
- redige la *Specifica Architetturale* e la parte pragmatica del *Piano di Qualifica*.

4.1.4.1.5 Programmatore

Il *Programmatore* è la figura professionale responsabile della codifica del progetto. Deve implementare l'architettura frutto del lavoro del *Progettista*, in modo tale che aderisca alle specifiche, ed è responsabile della manutenzione del codice prodotto. In particolare, il *Programmatore*:

- codifica secondo le specifiche del Progettista, in modo da renderne agevole la manutenzione, rispettando le *Norme di Progetto*;
- realizza gli strumenti per verificare e validare il software;
- documenta e versiona il codice prodotto;
- redige il*Manuale Utente*..

4.1.4.1.6 Verificatore

Il *Verificatore* è la figura professionale responsabile, per tutta la durata del progetto, del controllo del lavoro svolto dagli altri componenti del gruppo, in termini di rispetto delle norme e raggiungimento delle attese prefissate. In particolare, il *Verificatore*:

- esamina i prodotti secondo le tecniche e gli strumenti definiti nelle *Norme di Progetto;*
- si accerta che l'esecuzione delle attività di processo non abbia causato errori, segnalandoli se presenti;
- redige la parte retrospettiva del *Piano di Qualifica*, descrivendo le verifiche e le prove effettuate.

4.1.4.2 Rischi

4.1.4.2.1 Gestione dei rischi

Il Responsabile di Progetto deve monitorare i rischi evidenziati nel *Piano di Progetto*. Inoltre, deve classificare e documentare nuovi rischi, in caso emergessero. Il processo di gestione dei rischi comprende:

- la verifica dello stato dei rischi presenti nel *Piano di Progetto*;
- l'individuazione dei problemi non considerati in precedenza;
- l'aggiunta al *Piano di Progetto* dei nuovi rischi individuati;
- la registrazione degli aggiornamenti su tutti i rischi, sia nuovi sia già presenti;
- la ridefinizione delle strategie di progetto per la loro gestione.

4.1.4.2.2 Codifica dei rischi

I rischi sono codificati nel modo seguente:

R[Iniziale categoria][Numero]

Dove:

- **Iniziale Categoria**: indica la classificazione del rischio, appartenente alla categoria:
  - **T**: rischi correlati alle tecnologie scelte;
  - **I**: rischi correlati ai rapporti interpersonali;
  - **O**: rischi correlati all'organizzazione del lavoro.
- **Numero**: indica il numero progressivo che identifica in modo univolo il rischio in una categoria.

4.1.4.3 Metriche

Le metriche utilizzate per valutare il processo di pianificazione sono:

- **MPC01 - SPICE**: definisce la maturità del processo. Questo standard viene presentato nel dettaglio in §B;
- **MPC02 - Budgeted Cost of work Scheduled**: Indica una previsione di costo del lavoro programmato fino alla data corrente. Questo valore è consultabile all'interno del preventivo nel *Piano di Progetto*;
- **MPC03 - Actual Cost of work Perfomed**: Indica la somma di tutte le spese effettivamente sostenute dal gruppo dall'inizio del progetto fino alla data corrente. Disponibile come consuntivo di periodo nel *Piano di Progetto*, il suo valore dovrebbe essere minore o uguale a quanto preventivato;
- **MPC04 - Budgeted Cost of work Performed**: Indica il valore effettivo del prodotto ottenuto dall'inizio del progetto alla data corrente. Viene calcolato come:

$$ACP = \%\text{Lavoro completato} * \text{Budget Totale} \quad ;$$

- **MPC05 - Schedule Variance**: Indica il discostamento, in percentuale, del programma tra quello preventivato durante la progettazione e quello effettivo. La formula per calcolarlo è:

$$SV = \frac{100*(BCP-BCS)}{BCS}$$

  dove:
  - **SV**: schedule variance;
  - **BCP**: costo preventivato del lavoro svolto;
  - **BCS**: costo preventivato del lavoro programmato.
- **MPC06 - Budget Variance**: Indica il discostamento, in percentuale, tra le spese sostenute a partire dall'inizio del progetto fino alla data corrente ed il budget di spesa preventivato per la data corrente. La formula è:

$$BV = \frac{100*(BCS-ACP)}{BCS}$$

  - **BV**: Budget variance;
  - **ACP**: costo effettivo del lavoro svolto;
  - **BCP**: costo preventivato del lavoro svolto.

4.1.5 Coordinamento

4.1.5.1 Gestione delle comunicazioni

4.1.5.1.1 Comunicazioni interne

Le comunicazioni interne riguardano i membri del gruppo e avvengono attraverso i seguenti strumenti:

- *Telegram*$_G$, il quale:
  - permette di comunicare istantaneamente con tutti i membri del gruppo;
  - è utilizzato per la pianificazione degli incontri interni e per ogni discussione di carattere organizzativo.
- *Google Meet*$_G$, il quale:
  - permette la creazione di videoconferenze e condivisione dello schermo.

4.1.5.1.2 Comunicazioni esterne

Le comunicazioni esterne sono affidate al *Responsabile di Progetto* che:

- utilizza la casella di posta elettronica ufficiale del gruppo: foxybyte.swe@gmail.com;
- tiene informati tutti i componenti del gruppo circa le corrispondenze pervenute dai *committenti*$_G$ e *proponenti*$_G$.

Inoltre, gli strumenti utilizzati sono:

- *Slack*$_G$, il quale:
  - permette una comunicazione più veloce con il *committente*$_G$ per organizzare eventuali chiamate oppure chiarire dei dubbi riguardo i requisiti del prodotto.
- *Gmail*$_G$, il quale:
  - permette di comunicare con i *committenti* per eventuali dubbi e per organizzare le revisioni.

4.1.5.2 Gestione delle riunioni

4.1.5.2.1 Riunioni interne

Le riunioni interne sono riservate ai soli membri del gruppo. Il *Responsabile di Progetto* deve:

- fissare preventivamente la data della riunione, compatibilmente con la disponibilità dei membri del gruppo;
- stabilire i punti salienti della riunione;
- comunicare data e argomenti della riunione, avvertire circa ritardi, modifiche o cancellazioni;
- approvare il verbale della riunione.

I partecipanti, invece, devono:

- avvertire in caso di assenze o ritardi con un adeguato anticipo;
- presentarsi puntuali all'ora e luogo stabiliti;
- partecipare attivamente alla riunione.

Ogni membro del gruppo avrà la possibilità di chiedere un incontro interno, indirizzando la domanda al *Responsabile di Progetto*, che, qualora fosse necessario, lo pianificherà. Le riunioni interne sono svolte principalmente in modalità:

- **Virtuale**: i componenti del gruppo svolgono videochiamate di gruppo tramite *Google Meet*$_G$ per discutere dubbi o difficoltà incontrate durante lo svolgimento delle attività.

4.1.5.2.2 Riunioni esterne

Le riunioni esterne prevedono la partecipazione, oltre che dei membri del gruppo, dei soggetti esterni, ovvero il proponente e i committenti. E' prevista la nomina di un segretario incaricato di redigere il verbale della riunione. Le riunioni esterne vengono svolte principalmente in modalità virtuale, utilizzando i seguenti strumenti:

- *Zoom*$_G$: piattaforma virtuale, alternativa a *Google Meet*$_G$, per svolgere da remoto le riunioni;
- *Google Meet*$_G$: utilizzato come piattaforma virtuale della riunione stessa.

4.1.5.2.3 Verbali delle riunioni

I membri del gruppo incaricati dal *Responsabile di Progetto* hanno il compito, dopo ogni incontro, di redarre il verbale (secondo norme alla sezione §3.1.8).

4.1.5.3 Ticketing

Il *Responsabile di Progetto* si impegna a suddividere il carico di lavoro in task da assegnare ai componenti del gruppo. Perché ciò avvenga il gruppo si affida agli strumenti:

- *Jira*$_G$: il quale:
  - permette l'assegnazione dei task ai singoli componenti;
  - peremtte di avere un controllo costante dell'andamento del lavoro;
  - permette di avere un controllo efficiente del lavoro di ciascun componente.

4.2 Strumenti

Tutti gli strumenti di supporto ai processi organizzativi utilizzati dal gruppo sono:

- ***Git***$_G$: utilizzato per il versionamento del prodotto e dei documenti;
- ***GitHub***$_G$: utilizzato per il versionamento ed il salvataggio di tutti i file prodotti;
- ***Gmail***$_G$: servizio di posta elettronica del gruppo;
- ***Google Drive***$_G$: utilizzato per la condivisione di file che eventualmente richiedono molte modifiche in tempo reale.
- ***Google Meet***$_G$: utilizzato per le chiamate del gruppo e con il proponente;
- ***Jira***$_G$: utilizzato per la gestione dei task ed il coordinamento del gruppo;
- ***Slack***$_G$: utilizzato per le comunicazioni con il committente;
- ***Telegram***$_G$: utilizzato per la comunicazione interna del gruppo;
- ***Zoom***$_G$: .

4.3 Formazione

I membri del gruppo sono responsabili della loro formazione autonoma circa le tecnologie e gli strumenti necessari al completamento del progetto.

A Standard di qualità

A.1 ISO/IEC 9126

Negli anni sono stati definiti molti modelli della qualità del software. Ciascuno di essi definisce un set di caratteristiche ed attributi propri, anche se molto simili tra di loro.
Le norme ISO/IEC 9126 descrivono un modello di qualità del software, definiscono le caratteristiche che la determinano e propongono metriche per la misurazione. Le norme relative alla qualità del software constano di quattro parti:

1. Modello della qualità del software;
2. Metriche per la qualità interna;
3. Metriche per la qualità esterna;
4. Metriche per la qualità in uso.

A.1.1 Modello della qualità del software

Il modello, si è detto, definisce le caratteristiche di qualità. A ciascuna di esse sono associate sottocaratteristiche, dette anche attributi.

A.1.1.1 Funzionalità

La *funzionalità* rappresenta la capacità del software di fornire le funzioni, espresse ed implicite, necessarie per operare in determinate condizioni, cioè in un determinato contesto. Gli attributi del software richiesti da questa caratteristica sono: adeguatezza, accuratezza, interoperabilità, sicurezza, aderenza.

- **Adeguatezza**: Rappresenta la capacità di fornire un appropriato insieme di funzioni che permettano agli utenti di svolgere determinati $task_G$ e di raggiungere gli obiettivi prefissati;
- **Accuratezza**: Rappresenta la capacità di fornire i risultati o gli effetti attesi con il livello di precisione richiesta;
- **Interoperabilità**: Rappresenta la capacità di integrare con uno o più sistemi specificati;
- **Sicurezza**: Rappresenta la capacità di proteggere le informazioni ed i dati in modo che, persone o sistemi non autorizzati, non possano accedervi e quindi non possano leggerli o modificarli;
- **Aderenza**:itemize Rappresenta la capacità di aderire a standard, convenzioni e regolamenti di carattere legale o prescrizioni simili che abbiano attinenza con le funzionalità.

A.1.1.2 Affidabilità

L'*affidabilità* rappresenta la capacità di un prodotto software di mantenere il livello di prestazione quando viene utilizzato in condizioni specificate. Possibili limitazioni all'affidabilità del software possono essere causate da errori nei requisiti, nella progettazione, nel codice. Le evidenze di tali errori possono essere rilevate a seconda delle condizioni in cui è utilizzato. Gli attributi richiesti da tale caratteristiche sono:

- **Maturità**: Rappresenta la capacità di evitare che si verifichino errori o siano prodotti risultati non corretti in fase di esecuzione;
- **Tolleranza ai guasti**: Rappresenta la capacità di un prodotto software a mantenere il livello di prestazioni in casi di errori nel software o di violazione delle interfacce specificate;
- **Recuperabilità**: Rappresenta la capacità di un prodotto software di ripristinare il livello di prestazioni e di recuperare i dati direttamente coinvolti in caso di errori o malfunzionamenti;
- **Aderenza**: L'aderenza di un prodotto software rappresenta la capacità di aderire a standard, convenzioni e regole relative all'affidabilità.

A.1.1.3 Usabilità

L'*usabilità* rappresenta la capacità di un prodotto software ad essere comprensibile, di poter essere studiato, di risultare attraente da parte di un utente sotto determinate condizioni. Alcuni aspetti della "funzionalità", "affidabilità" ed "efficacia" possono influire sull'usabilità del prodotto software anche se, per il modello ISO/IEC 9126, queste non sono classificate come usabilità. Tra gli utenti sono inclusi gli operatori, gli utenti finali ed altri utenti indiretti che sono influenzati dal software o che dipendono da esso. Gli attributi sono:

- **Comprensibilità**: Rappresenta la capacità di un prodotto software a permettere all'utente di capire la sua funzionalità e come poterla utilizzare;
- **Apprendibilità**: Rappresenta la capacità di un prodotto software a permettere all'utente di imparare l'applicazione;
- **Operabilità**: Rappresenta la capacità di un prodotto software a permettere all'utente di utilizzarlo e di controllarlo;
- **Attratività**: Rappresenta la capacità di risultare "attraente" per l'utente;
- **Aderenza all'usabilità**: Rappresenta la capacità di aderire a standard, convenzioni, guida allo stile e regole relative all'usabilità.

A.1.1.4 Efficienza

L'*efficienza* rappresenta la capacità di un prodotto software di realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie, quando opera in determinate condizioni. Le risorse includono altri prodotti software, la configurazione hardware e software del sistema, i mateziali (carta per stampanti, dischetti, ecc).
Gli attributi relativi all'efficienza del software sono:

- **Comportamento rispetto al tempo**: Rappresenta la capacità di fornire appropriati tempi di risposta, tempi di elaborazione e quantità di lavoro eseguendo le funzionalità previste sotto determinate condizioni di utilizzo;
- **Utilizzo delle risorse**: Rappresenta la capacità di utilizzare un appropriato numero e tipo di risorse quando esegue le funzionalità previste sotto determinate condizioni di utilizzo;
- **Aderenza all'efficienza**: Rappresenta la capacità di aderire a standard e convenzioni relative all'efficienza.

A.1.1.5 Manutenibilità

La *manutenibilita* rappresenta la capacità di un prodotto software di essere modificato. Le modifiche possono includere correzioni o adattamenti del software a modifiche negli ambienti, nei requisiti e nelle specifiche funzionali. In altre parole, *"II software segue l'evoluzione dell'organizzazione"*. Gli attributi previsti per la manutenibiliti del software sono:

- **Analizzabilità**: Rappresenta la capacità di poter effettuare la diagnosi sul software ed individuare le cause di errori o malfunzionamenti;
- **Modificabilità**: Rappresenta la capacità di consentire lo sviluppo di modifiche al software originale;
- **Stabilità**: Rappresenta la capacità di evitare effetti non desiderati a seguito di modifiche al software;
- **Provabilità**: Rappresenta la capacità di consentire la verifica e validazione del software modificato, cioè di eseguire i test;
- **Aderenza alla manutenibilità**: Rappresenta la capacità di aderire a standard e convenzioni relative alla manutenibilità.

A.1.1.6 Portabiltà

La *portabilità* rappresenta la capacità di un prodotto software di poter essere trasportato da un ambiente ad un altro. L'ambiente include aspetti organizzativi e tecnologici (hardware e software). Gli attributi sono:

- **Adattabilità**: Rappresenta la capacità di essere adattato a differenti ambienti senza richiedere azioni specifiche diverse da quelle previste dal software per tali attività;
- **Installabilità**: Rappresenta la capacità di essere installato in un determinato ambiente;
- **Coesistenza**: Rappresenta la capacità di coesistere con altre applicazioni indipendenti in ambienti e l'operatività;
- **Sostituibilità**: Rappresenta la capacità di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente;
- **Aderenza alla portabilità**: Rappresenta la capacità di aderire a standard e convenzioni relatove alla portabilità.

A.1.2 Metriche per la qualità interna

Le metriche interne si applicano al software non eseguibile durante le fasi di progettazione e codifica. Durante le fasi di sviluppo del software, quindi, i prodotti intermedi sono valutati tramite metriche interne che misurano le proprietà intrinseche del prodotto. Tali metriche includono quelle che possono misurare il comportamento del prodotto finale tramite simulazioni. Le metriche interne permettono ad utenti, valutatori e sviluppatori di indirizzare per tempo - prima che sia realizzato il software eseguibile - eventuali problemi che potrebbero inibire la qualità finale del prodotto.

A.1.3 Metriche per la qualità esterna

Le metriche esterne misurano i comportamenti del prodotto software rilevabili dai test, dall'operatività, dall'osservazione durante la sua esecuzione. L'esecuzione del prodotto software è fatta in base al suo utilizzo in funzione degli obiettivi di business stabiliti ed in un contesto organizzativo e tecnico rilevante.
Le metriche esterne sono scelte in base alle caratteristiche che il prodotto finale dovrà dimostrare durante la sua esecuzione in esercizio. La norma ISO/IEC 91262 presenta un numero consistente di metriche esterne su cui operare la scelta. Esse forniscono agli utenti, tester e sviluppatori elementi concreti per misurare e valutare le caratteristiche esterne e comportamentali del prodotto finale durante il suo utilizzo.

A.1.4 Metriche per la qualità in uso

Le metriche della "qualità in uso" misurano il grado con cui il prodotto software permette agli utenti di svolgere le proprie attività con efficacia, produttività, sicurezza e soddisfazione nel contesto operativo previsto. La valutazione della qualità finale del prodotto software è quindi fatta in specifici scenari d'uso.
La qualità in uso rappresenta la vista esterna che l'utente ha del prodotto ed è misurata in base ai risultati ottenuti dal suo utilizzo, piuttosto che da caratteristiche interne al software.

B ISO/IEC 15504 - SPICE

B.1 Introduzione

ISO / IEC 15504, noto anche come Software Process Improvement and Capability Determination (SPICE), è un insieme di standard tecnici per i processi di sviluppo del software. Costituisce un riferimento per la valutazione della maturità dei processi software rispetto al quale i valutatori possono misurare gli esiti raccolti dalle prove di qualità, in modo da ricavare una determinazione generale delle capacità dell'organizzazione di rilascio dei prodotti software. Il modello di riferimento SPICE, contenuto nello standar ISO/IEC 15504, definisce una *dimensione di processo* e una *dimensione di Capability*.

B.2 Classificazione dei processi

La *dimensione del processo* definisce i processi suddivisi nelle cinque categorie di processo di:

- Cliente-Fornitore;
- Ingegneristico;
- di Supporto;
- Gestionali;
- Organizzativi.

B.3 Livello di Capability

B.3.1 Concetto di Capability

La *Capability* è la capacità di un processo di essere capace di raggiungere il suo scopo. Un livello alto di capability denota che il processo è attuato da tutto il team in modo disciplinato e sistematico; un livello basso di capability, invece, indica che il processo viene istanziato poco professionalmente da parte del team.

B.3.2 Livelli di Capability

Lo standard definisce un *livello di Capability* su una scala di sei livelli, a cui vengono assegnati degli attributi col fine di valutare l'effettiva qualità del processo. I livelli stabiliti sono:

- **Livello 0 - *Incomplete***: il processo non è implementato, quindi non è in grado di raggiungere i propri obiettivi e di conseguenza non viene prodotto alcun output significativo;
- **Livello 1 - *Performed***: il processo è implementato e raggiunge gli obiettivi prestabiliti, producendo degli output identificabili. Però non è ancora sottoposto a controlli costanti ai fini di correzione e miglioramento.
  L'attributo associato al livello è:
  - **performance di processo (PP)**: riporta il numero di obiettivi raggiunti.
- **Livello 2 - *Managed***: il processo è gestito tramite pianificazione, controllo e correzione. L'output risultante raggiunge gli obiettivi fissati.
  Gli attributi associati al livello sono:
  - **gestione della performance (PM)**: indica il grado di organizzazione degli obiettivi fissati;
  - **gestione del prodotto di lavoro (WPM)**: indica il grado di organizzazione dei prodotti rilasciati.

- **Livello 3 - *Established***: il processo è definito a partire da uno standard, viene quindi regolamentato da principi dettati dall'Ingegneria del Software. L'output risultante impiega una quantità limitata di risorse.
  Gli attributi associati al livello sono:
  - **definizione di processo (PDEF)**: indica il grado di adesione del processo rispetto gli standard;
  - **rilascio di processo (PDEP)**: indica in che misura il processo può essere rilasciato garantendone la ripetibilità.
- **Livello 4 - *Predictable***: il processo è istanziato in modo coerente entro limiti definiti. È inoltre caratterizzato dalla raccolta e dal monitoraggio di misure dettagliate e funzionali al consolidamento della prevedibilità del processo.
  Gli attributi associati al livello sono:
  - **misurazioni di processo (PME)**: indica con quanta efficacia le metriche possono essere applicate al processo;
  - **controllo di processo (PC)**: indica il grado di predicibilità dei risultati delle valutazioni.
- **Livello 5 - *Optimizing***: il processo è correttamente definito e tracciato. Viene sottoposto a continue analisi con il fine di miglioramento.
  Gli attributi associati al livello sono:
  - **innovazione di processo (PI)**: indica quanto i cambiamenti attuati nel processo risultano innovativi e positivi;
  - **ottimizzazione di processo (PO)**: indica quanto la curva di miglioramento del processo sia lineare. Rappresenta una corretta gestione del rapporto tra risorse impiegate e risultati ottenuti.

Ognuno degli attributi associati ai vari livelli viene valutato con un giudizio su una scala di quattro livelli. A questo giudizio corrisponde inoltre una classe di valori percentuali che ne esprimono il grado di soddisfacimento in forma numerica. Questi quattro giudizi sono:

- **N - *not achieved* [0% - 15%]**: il processo non implementa l'attributo o presenta gravi lacune in merito;
- **P - *partially achieved* [15% - 50%]**: il processo implementa sistematicamente l'attributo, ma risulta ancora migliorabile e poco predicibile;
- **L - *largely achieved* [50% - 85%]**: il processo implementa ampiamente l'attributo in modo sistematico, ma il suo valore risulta ancora poco uniforme all'interno delle varie parti del processo;
- **F - *fully achieved* [85% - 100%]**: il processo implementa completamente l'attributo in modo sistematico e uniforme in ogni sua parte.