SWEG GROUP

Norme di Progetto

Versione 3.0.0
Data di rilascio 22/06/2017
Redazione Sebastiano Marchesini
Piergiorgio Danieli
Verifica Alberto Gelmi
Validazione Pietro Lonardi
Responsabile Sebastiano Marchesini
Uso Interno
Destinato SWEg Group

Sommario
Questo documento ha l'obiettivo di definire le regole che i membri del gruppo seguiranno nello sviluppo del progetto.

Registro Modifiche

Ver.	Modifica	Nome	Data
3.5.6	Aggiunte: Tutte le metriche mancanti cap.3.8.14 Descritte più in dettaglio	Sebastiano Marcheisni	20/06/2017
3.5.5	Aggiunti: Manuale Amministratore cap.2.3.14 Codifica cap.2.3.16 Ciclo di vita di un documento cap.3.4.6 Capitoli 3.4.1 , 3.4.2, 3.4.4 Processo di miglioramento dei Processi cap.4.11 Processo di formazione cap.4.12	Sebastiano Marchesini	19/06/2017
3.5.4	Rettificati completamente: Manuale Utente cap.2.3.13 Specifica Tecnica cap.2.3.8 Definizione di Prodotto cap.2.3.11 Processi di fornitura cap.2 Capitoli 3.4.1 , 3.4.2, 3.4.4 Sistema rotativo dei ruoli 4.1	Sebastiano Marchesini	18/06/2017
3.5.4	Riorganizzazione capitoli in riferimento allo standard ISO/IEC 12207:2008	Sebastiano Marchesini	19/06/2017
3.5.3	Aggiunti elementi alla Lista di Controllo cap.4.8.8	Sebastiano Marchesini	14/06/2017
3.5.2	Rettifica: Versione tutti i documenti di riferimento Pianificazione strategica e temporale cap.4.8.4	Sebastiano Marchesini	14/06/2017
3.5.1	Arricchimento: Convenzioni tipografiche cap.4.2 Per le Tabelle e Immagini	Sebastiano Marchesini	14/06/2017
3.5.0	Verificato intero documento Correzione ortografica e sintattica	Sebastiano Marchesini	14/06/2017
3.0.1	Riorganizzato capitolo 5 Migliorato 4.8 Sistemato 4.10	Piergiorgio Danieli	07/06/2017
3.0.0	Validazione	Sebastiano Marchesini	08/05/2017
2.0.2	Riorganizzato capitolo processi primari	Piergiorgio Danieli	21/04/2017
2.0.1	Appendice A, aggiunto foglio di Google	Piergiorgio Danieli	20/04/2017
2.0.0	Verifica delle aggiunte	Lonardi Pietro	22/02/2017
1.0.1	Aggiunta procedura di verifica	Piergiorgio Danieli	22/02/2017
1.0.0	Correzioni	Piergiorgio Danieli	15/02/2017
1.0.1	Validazione	Pietro Lonardi	09/01/2017
0.5.1	Correzioni Verifica	Sebastiano Marchesini	08/01/2017
0.5.0	Verifica	Alberto Gelmi	06/01/2017
0.1.2	Aggiunti Punti nella Lista di Controllo	Sebastiano Marchesini	05/01/2017
0.1.1	Stesura Completa	Sebastiano Marchesini	02/01/2017
0.1.0	Stesura Iniziale	Sebastiano Marchesini	30/12/2016
0.0.6	Aggiunto Versionamento	Sebastiano Marchesini	15/12/2016
0.0.5	Aggiunto Ambiente di Lavoro	Piergiorgio Danieli	15/12/2016

0.0.4	Aggiunti Analisi dei Requisiti	Sebastiano Marchesini	14/12/2016
0.0.3	Aggiunte Procedure di Sviluppo	Piergiorgio Danieli	13/12/2016
0.0.2	Aggiunti Ruoli	Sebastiano Marchesini	12/12/2016
0.0.1	Creazione Documento	Piergiorgio Danieli	12/12/2016

Indice

1 Introduzione

1.1 Scopo del Documento

Questo documento ha l'obiettivo di definire le regole che i membri del gruppo *SWEG* seguiranno nello sviluppo del progetto.
Ogni componente del team é tenuto a leggere e seguire le norme qui contenute per ottimizzare il lavoro, uniformare tutti i documenti e minimizzare gli errori.
In particolare verranno specificate le norme per:

- interazioni tra i membri del gruppo e componenti esterni;
- stesura dei documenti e relative convenzioni;
- ambiente di lavoro;
- stesura del codice.

1.2 Glossario

Al fine di evitare ambiguità e ottimizzare la comprensione dei documenti, viene incluso un Glossario, nel quale saranno inseriti i termini tecnici, acronimi e parole che necessitano di essere chiarite.
Un glossario è una raccolta di termini di un ambito specifico e circoscritto. In questo caso per raccogliere termini desueti e specialistici inerenti al progetto.

1.3 Riferimenti

1.3.1 Normativi

- **Standard ISO/IEC 12207**.

1.3.2 Informativi

- **Manuale Latex**:
  http://www.guit.sssup.it/downloads/LaTeX-facile.pdf.
  http://www.guit.sssup.it/downloads/fig_tut.pdf.
  http://mnnugm.altervista.org/latex/Tabelle.pdf.
- **Piano di Qualifica**:
  "Piano di Qualifica v4.0.0".
- **Piano di Progetto**:
  "Piano di Progetto v4.0.0".

2 Processi Primari

Si è scelto di seguire e reinterpretare per i processi primari lo **Standard ISO/IEC 12207**. Tenendo delle basi comuni ma essendo più flessibile su alcuni processi, raggruppandoli e rinominandoli.
I processi primari esistono all'esistenza del processo, contengono il nocciolo dei processi coinvolti nella creazione di un prodotto software. Sono divisi in cinque processi principali:

- Acquisizione;
- Fornitura;
- Sviluppo;
- Gestione operativa (utilizzo);
- Manutenzione.

2.1 Processo di Acquisizione

L'acquisizione comprende tutte le attività coinvolte nel promuovere un progetto. L'obiettivo di questa fase è raggiungere l'accordo tra le parti. La descrizione nel dettaglio di questa fase esula dallo scopo di questo documento.

2.2 Processo di Fornitura

Formato dai:

- Bandi del Fornitore;
- Accordo Contrattuale;
- Fornitura di Rilascio del Prodotto;
- Accettazione del Prodotto.

2.2.1 Obbiettivo

Il processo riguarda le azioni che deve svolgere il *Fornitore*, a partire dalla stesura dell'offerta tecnico-economica dopo aver scelto il capitolato adeguato, sino alla realizzazione ed alla consegna della fornitura.

2.2.2 Attivazione del processo

Il processo è attivato dal *Fornitore* a fronte della decisione di predisporre un'offerta per rispondere alla richiesta di presentazione di una proposta da parte di un'Amministrazione. Il processo prosegue quindi con la sottoscrizione del contratto con il team, a cui segue la determinazione delle procedure e delle risorse necessarie a gestire ed assicurare la corretta esecuzione del capitolato, con lo svolgimento di tutte le attività previste nell'*"Analisi dei Requisiti v3.0.0"*.

2.2.3 Attività e prodotti

Nello schema che segue si fornisce una rappresentazione delle attività del processo di Fornitura che riguardano la fase precedente la sottoscrizione del contratto. Le attività che il gruppo deve svolgere a valle della sottoscrizione del contratto, ricadono nell'ambito degli altri processi primari, organizzativi e di supporto del ciclo di vita della fornitura.

- (Superamento delle materie che creano propedeuticità);
- Studio personale della materia Ingegneria del Software;
- Registrazione e creazione del gruppo tramite documento condiviso;
- Analisi del capitolato scelto in comune accordo;
- Realizzazione dello studio di fattibilità.

2.2.4 Preparazione della Risposta

In risposta alla richiesta di una proposta da parte di un'azienda, il gruppo predispone un'offerta in cui definisce modi, tempi e costi per realizzare il progetto, nel rispetto dei requisiti specificati (offerte tecnico-economiche, organigramma, ecc.).
Il risultato delle attività è l'offerta tecnico-economica che è vincolante per il team e diviene parte integrante della baseline di contratto. Questo avviene attuando uno studio specifico sui vari capitolati e realizzando un documento dedicato: lo *Studio di Fattibilità v2.0.0*. Il documento è realizzato tramite i seguenti principi:

- **Introduzione**: standard con descrizione dello
  scopo del documento,
  scopo del prodotto,
  riferimenti e introduzione al glossario,
  riferimenti normativi e informativi.
- **Valutazione del capitolato scelto**: nel nostro caso il capitolato C1 - API Market. Con una
  descrizione semplice del prodotto idealizzato,
  studio del dominio, sia applicativo sia tecnologico,
  valutazione del capitolato generale e delle potenziali criticità.
- **Confronto con altri capitolati**: in cui verranno effettivamente comparati il progetto scelto con gli altri proposti.

2.2.5 Riesame e sottoscrizione del contratto

In caso di superamento della Revisione dei Requisiti, il team deve negoziare e sottoscrivere il contratto con l'ente.
È parte integrante dell'attività un riesame del contratto da sottoscrivere, al fine di assicurare che i requisiti siano adeguatamente definiti e documentati, che eventuali scostamenti tra i requisiti riportati nel capitolato e quelli riportati nell'offerta siano risolti. Fare un calcolo delle risorse e stimare un prezzo ragionevole per l'impegno preso.

2.2.6 Pianificazione

Per avere delle linee guida e una stima effettiva dell'impegno persona al interno del progetto viene stilato il documento *Piano di Progetto v4.0.0* contenente la pianificazioni a periodi di tempo racchiusi tra revisioni concordate tra il team e rese accessibili dal committente. Lo strumento utilizzato principalmente in questa attività è il *Gantt Chart*$_g$ che descriveremo più dettagliatamente nelle sezione 5.4.

2.2.7 Chiusura del processo

Il processo di Fornitura si conclude con il completamento da parte del gruppo di tutte le attività previste in esecuzione del contratto.

2.3 Processo di Sviluppo

Il metodo *incrementale* è il modello utilizzato per ogni passaggio nel processo di sviluppo. Il *Responsabile di Progetto* granula ogni attività per poi rendere il modello attuabile secondo le risorse per produrre in modo efficiente ed efficace. Si rimanda al *"Piano di Progetto v4.0.0"* per i dettagli.
La specifica data che indica il passaggio tra il medesimo processo ed il successivo è regolata dai giorni di consegna e rilascio della correzione del capitolato. Tranne nella presentazione dei capitolati e inizio delle attività, vi sarà certamente un integrazione su ogni documento dettata dalle correzioni riportate dal *Committente*.
Per processi di sviluppo si è scelto di rielaborare quelli descritti nello standard ISO. Di seguito indichiamo la dichiarazione e le rielaborazioni.
Ricordiamo inoltre che ogni processo ha inevitabilmente una o più fasi di verifica.

2.3.1 Analisi

È la prima attività di studio e di stesura dei documenti.In questo periodo la prima cosa da fare è redigere il documento *Studio di Fattibilità*, ed effettuare una corretta e specifica *Analisi dei Requisiti*. In seguito dopo un incontro con i membri del team, ed il team concorrente, si sono stabiliti i ruoli e si dividono i compiti come descritto nel *Piano di Progetto v4.0.0*. È un processo lungo e getta le basi per le future versioni.

2.3.2 Studio di Fattibilità e Analisi dei Rischi

Dopo aver indetto le riunioni e stabilito i vari ruoli gli *Analisti* valuteranno capacità e preferenze del progetto che gioveranno alla completa realizzazione degli obbiettivi. È quindi compito di questi ultimi, sulla base di quanto deciso, redigere uno *Studio di Fattibilità* con:

- Motivi della scelta del capitolato;
- Ragioni per le quali sono stati scartati gli altri capitolati;
- Contributi agli obbiettivi dell'organizzazione;
- Ingegnerizzare con la tecnologia corrente entro il *budget*;
- Integrazione con altri sistemi usati.

2.3.3 Analisi in dettaglio

Questo processo si concentra sul documento di *Analisi dei Requisiti*, ed è il punto focale per la stipulazione del contratto con il *Committente*.
Il gruppo provvederà a una scrematura dei requisiti, con una divisione in accordo con il gruppo concorrente. Potrà inoltre porre delle domande all'azienda per migliorare la comprensione del capitolato e diminuire le possibilità di fallimento.

2.3.4 Analisi dei Requisiti

Gli *Analisti* hanno il compito di redigere l'*Analisi dei Requisiti*. Comunicheranno poi al *Project Manager* il bisogno di negoziare e chiarire con il committente punti meno chiari del capitolato.

2.3.5 Classificazione dei requisiti

Ogni requisito può essere provvisto di più sotto-requisiti. I requisiti vengono organizzati in forma gerarchica.
Al compimento di tutti i moduli che compongono il requisito padre questo viene soddisfatto.
La notazione da utilizzare è quella che segue:

<0|1|2><F|P|Q|V>-X<.Y<.Z> >

Dove le seguenti sigle indicano:

0: obbligatorio;

1: desiderabile;

2: opzionale.

Hanno diverso tipo:

F: requisito funzionale;

P: requisito prestazionale;

Q: requisito di qualità;

V: requisito dichiarativo(vincoli).

Tale parte indica il codice identificativo di ogni requisito.
È espresso in modo gerarchico ed univoco.

X: requisito di primo livello;

Y: sotto-requisito;

Z: sotto-requisito di un sotto-requisito.

I campi Y e Z possono essere assenti.
Si ricorda che per ogni requisito c'è bisogno di una descrizione e di riportare le dipendenze che ha verso altri. I requisiti non devono comunque essere in conflitto tra loro.

2.3.6 Allocazione e modellazione concettuale del sistema

Il team di *Analisti*, dopo aver preso visione nello specifico del capitolato e aver fatto emergere i requisiti, procede con la costruzione di diagrammi dei casi d'uso (*UC*).
Nello specifico è richiesto:

- **Titolo** del caso d'uso sintetico;
- **Attori Principali**;
- **Attori Secondari**, nel caso vi fossero.

Descrizione del caso con:

- **Scenario principale**;
- **Scenari/o alternativo**, se presenti;
- **Pre-Condizione**;
- **Post-Condizione**;
- **Requisiti del caso d'uso ricavati**.

Il caso d'uso sarà sempre associato da un grafico specializzato. Si è scelto uno standard ibrido tra UML_g *2.x e 1.x* con le funzioni principali che racchiudono le due versioni di linguaggio.

2.3.7 Progettazione Architetturale

Progettazione non specifica e soluzioni dei principali requisiti accordati.
Redazione di nuovi documenti volti a illustrare scelte progettuali che il prodotto deve avere.

Lo strumento principale usato per la progettazione, in particolare per la creazione dei diagrammi è $Astah_{\text{g}}$. Tale strumento è descritto più specificatamente nella sezione 5.3.3.

2.3.8 Specifica Tecnica

I progettisti devono descrivere in questo documento la progettazione ad alto livello dell'architettura dell'applicazione e dei singoli componenti, e provvedono alla progettazione di opportuni test di integrazione. In questo documento devono essere descritti i *package*, i diagrammi delle classi, i diagrammi di sequenza e quelli di attività. Inoltre deve essere presente una sezione dedicata ai *Design Pattern* utilizzati, e per ognuno di essi deve esserci una breve descrizione ed un diagramma che ne esemplifichi il funzionamento e la struttura. E una sezione che descrive le tecnologie utilizzate e prese in carico per completare il prodotto software.

Il documento è realizzato seguendo le linee guida qui presenti:

- **Introduzione** standard con descrizione dello
  scopo del documento,
  scopo del prodotto,
  riferimenti e introduzione al glossario,
  riferimenti normativi e informativi.
- **Tecnologie Utilizzate** in cui vengono descritti i
  linguaggi,
  librerie e
  framework.

- **Descrizione generale dell'architettura**: senza essere specifici ma dando un idea di quello che troveremo all'interno del prodotto a livello software.
- **Architettura Front-End** e **Architettura Back-End** con la descrizione dei package, delle classi e della relazione tra loro.
- **Architettura del Database** dove vi sarà la consistenza dei nostri dati. Essa sarà compresa di :

  progettazione concettuale,

  progettazione logico-relazionale.
- **Design Pattern** principali utilizzati nel progetto. Preventivamente studiati ed applicati a seconda delle tecnologie e dello scopo di questi. Suddivisi per design pattern:

  architetturali,

  creazionali,

  strutturali,

  comportamentali.
- **Diagrammi di attività** dei principali algoritmi che muovono il sistema.
- **Tracciamento** nello specifico dei:

  requisiti verso i componenti,

  componenti verso i requisiti.
- **Un appendice** che riporta :

  Descrizione dei Design Pattern più teorica. Anche questa suddivisa per categorie.

  Mock-Up in cui viene rappresentato un prototipo ideologico del progetto a livello di grafica utente.

2.3.9 Design Pattern

I *Progettisti* devono descrivere i *Design Pattern* utilizzati per realizzare l'architettura: di essi si deve includere una breve descrizione ed un diagramma che ne esemplifichi il funzionamento e la struttura. Questa descrizione deve essere inserita all'interno del documento precedentemente descritto, la *Specifica Tecnica v3.0.0*.
Non è stato riportato il diagramma specifico del pattern *Dependency Injection*$_g$ concordando con il committente in quanto è impossibile la sue riproduzione in standard *UML*.

2.3.10 Progettazione di Dettaglio e Codifica

Ogni parte del progetto, a fine i questo periodo, è definita e progettata. Il sistema è modulato in ogni minima soluzione. Resta solo codifica e verifica del codice. In questo periodo viene anche redatto il *Manuale Utente*, il quale dovrà spiegare il corretto utilizzo e funzionamento del software in modo dettagliato e preciso per un possibile user. I documenti da redigere sono nello specifico: la *Definizione di Prodotto*, che va nel dettaglio di ogni singola classe e funzione del prodotto software, ed il *Manuale Utente* precedentemente descritto.

2.3.11 Definizione di Prodotto

I *Progettisti* producono tale documento dove viene descritta la progettazione di dettagli del sistema ampliando quanto già scritto nella *Specifica Tecnica v3.0.0*. Lo scopo di questo documento è quello di definire dettagliatamente ogni singola unità di cui è composto il sistema in modo da semplificare l'attività di codifica ed allo stesso tempo evita di fornire libertà al *Programmatore*. Parallelamente alla progettazione di dettaglio dei componenti software dovranno essere progettati i relativi test di unità, integrazione e sistema, che verranno descritti nel *Piano di Qualifica v4.0.0*.

Il documento deve presentare le seguenti sezioni:

- **Introduzione** con descrizione dello

  scopo del documento,

  scopo del prodotto,

  riferimenti e introduzione al glossario,

  riferimenti normativi e informativi,

  link alla repository, unica aggiunta che modifica lo standard di tale sezione.

- **Standard di Progetto** in cui vengono brevemente descritta

  la progettazione architetturale,

  documentazione del codice,

  denominazione di entità e relazioni,

  codifica e

  strumenti di lavoro.

- **Specifica architettura di Back-End** e **Specifica architettura di Front-End** nel dettaglio rispetto al documento precedente. Fino ad arrivare a descrivere nel particolare:

  Descrizione della classe,

  utilizzo,

  classi che eredita,

  attributi con descrizione (nel caso di Jolie suddivisi ulteriormente per *Location*, *Protocol*, *Interfaces*),

  funzioni e la loro descrizione.

- **Diagrammi di Sequenza** dei principali algoritmi che muovono il sistema specificando attori e una descrizione testuale.

- **Tracciamento** nello specifico dei:

  requisiti verso le classi,

  classi verso i requisiti.

2.3.12 Classificazione dei package

I package, e gli elementi contenuti, vengono descritti univocamente a seconda del percorso e dell documento specifico:

Nella documento *Specifica Tecnica v3.0.0* vengono il percorso deve essere riportato tramite "punti":

PackagePadre.PackageSpecifico.Classe

Nella documento *Definizione di Prodotto v2.0.0* invece il percorso deve essere riportato tramite "due punti":

PackagePadre::PackageSpecifico::Classe

La classe è inserita solo se è presente nella descrizione dell'elemento. Inoltre va ricordato che *Jolie*$_g$ è un linguaggio in cui le classi hanno una forma più astratta rispetto i linguaggi a oggetti classici

2.3.13 Manuale Utente

Questo documento deve contenere tutte le istruzioni necessarie all'utente per poter utilizzare senza problemi il sistema.
È compreso di immagini , prima *Mock-Up*, poi *snapshot*$_g$ di quello che l'utente ha la possibilità di interfacciarsi.
Deve inoltre contenere in appendice un glossario personalizzato.

2.3.14 Manuale Amministratore

Questo documento similare al *Manuale Utente* conterrà invece le specifiche funzioni che è possibile attuare da utente amministratore.
Redatto nel periodo finale presenterà, nella sua prima versione, direttamente degli *snapshot*$_g$ delle pagine personalizzate ad uso amministrativo.
Anche questo documento deve inoltre contenere in appendice un glossario personalizzato.

2.3.15 Manuale Sviluppatore

2.3.16 Codifica

L'attività di codifica ha come obiettivo quello di trasformare, in maniera efficiente, dalla soluzione descritta nella *Definizione di Prodotto* realizzata dai Progettisti a quella finale eseguibile da un calcolatore. I fautori di questo processo sono i Programmatori che seguiranno delle semplici regole per una maggior fluidità del codice e comprensione.

- **Formattazione**: è stata presa nello specifico l'indentazione data da il tasto *"tab"* e standardizzata dal software *Sublime Text*$_g$.
- **Suddivisione:** del codice in maniera logica e in file e package adeguati.
- **Incapsulamento:** il più possibile creando così un ambiente più pulito e sano di codifica.
- **Convenzioni di Layout:**

  scrivere una sola istruzione per riga;

  una sola dichiarazione per riga;

  se le righe di continuazione non sono rientrate automaticamente, impostare un rientro con un punto di tabulazione (quattro spazi);

  aggiungere almeno una riga vuota tra le definizioni di metodo e proprietà;

- **Convenzioni relative ai commenti:**

  iniziare il commento con una lettera maiuscola;

  il commento deve essere di facile comprensione generale;

  evitare commenti non necessari con linguaggio inadatto e tenere sempre in considerazione che possono essere letti da terzi;

  terminare il commento con un punto finale.

- **Parentesi**: graffe di apertura di una classe o funzione sulla riga di dichiarazione (metodo *inline*).
- **Convenzioni sui nomi:**

  per le classi si è scelto di utilizzare la prima lettera maiuscola;

  nel caso di classe con nome composto da più vocaboli, questi saranno uniti senza alcuno spazio con l'iniziale del secondo anch'essa maiuscola;

  evitare nomi uguali tra le classi e/o attributi e/o funzioni;

  attributi iniziali in minuscolo;

  interfacce con idealizzato il nome standard, mentre la classe estesa riporterà precedentemente il termine "*Conc*".

- **Rami di Sviluppo:** si sono utilizzati solamente 3 rami per la codifica.

  Master che comprende tutti i documenti, il database e vengono riuniti successivamente i rami con certezza di esecuzione corretta del codice;

  Front-End con i lavori della parte frontale dell'applicazione;

  Back-End dove riporta invece il codice scritto in *Jolie* e $Java_g$

2.3.17 Validazione e Collaudo

La ricomposizione di ogni parte del progetto e la prova tramite *Analisi Dinamica*. Revisione completa di ogni documento e parte. Collaudo e validazione di ogni struttura e componente progettuale daranno il via libera alla consegna finale del prodotto.
Realizzazione del *Manuale Amministratore v1.0.0* e *Manuale Sviluppatore v1.0.0* in aggiunta alla rettifica del *Manuale Utente v2.0.0*

2.3.18 Manuale Amministratore

Tale documento illustra le operazione che può eseguire l'amministratore in uno scenario di login particolare e governativo.

2.3.19 Manuale Sviluppatore

Il prodotto software essendo progettato con architettura a microservizi e quindi scalabile, implementa tra i prodotti finali uno strumento documentale in cui è possibile il richiamo a componenti software di pertinenza.

2.4 Gestione operativa

Le fasi di funzionamento e di manutenzione avvengono simultaneamente. Il funzionamento consiste in attività come assistere gli utenti nel lavorare con il prodotto software realizzato.

2.5 Manutenzione

La fase di manutenzione consiste di tutte quelle attività che servono per mantenere il prodotto installato e funzionante. La manutenzione include qualsiasi miglioramento generale, modifiche ed aggiunte che potrebbero essere richieste dagli utenti finali. Questo tipo di lavoro non è richiesto in questa sede, ma il team *SWEg Group* si riserva di lasciare la propria mail (*sweg.group@gmail.com*) in caso il proponente abbia necessità di chiedere della manutenzione al prodotto.

2.6 Tracciamento

Il tracciamento dimostra completezza ed economicità della soluzione. Se tracciati tutti i requisiti conseguentemente:

- Possiamo soddisfarli;
- Nessuna funzionalità è superflua;
- Nessun comportamento è ingiustificato.

La facilità d'uso del tracciamento manuale e la continua comunicazione tra i membri del gruppo ha fatto si che un membro si è preso la responsabilità di tenere sotto controllo e dirigere il tracciamento in tutti i documenti delle componenti software con i requisiti. Questo processo è stato effettuato passo passo, in modo da essere sempre aggiornati e poter tenere sotto controllo questo aspetto.

3 Processi di supporto

In questa sezione verranno descritte le procedure che tutti i componenti del gruppo sono tenuti a seguire per ottenere un risultato soddisfacente degli obiettivi stabiliti dal *Piano di qualifica v4.0.0*. In particolare di seguito verranno elencati gli standard da seguire per la compilazione e la formattazione dei documenti.

3.1 Documenti

Per facilitare ed unificare la redazione della documentazione è stato creato un template apposito in LaTeX. Utilizzato all'inizio della creazione di ogni nuovo prodotto documentale.
Questo facilità sia efficacemente sia efficientemente un buon risultato qualitativo per lo sviluppo. Inoltre è utile a qualsiasi componente per un ripasso semplificato e schematico delle principali funzioni di LaTeXindicate all'inizio di questo tra commenti.

3.1.1 Stile del testo

- **Grassetto**: Deve essere applicato alle parole significative che devono essere messe in risalto;
- **Corsivo**: va utilizzato per indicare termini in lingua inglese, nomi dei file e testo che indica un documento o un ruolo all'interno del progetto;
- **Maiuscolo**: va usato per scrivere gli acronimi;
- **Latex**: ogni riferimento a LaTeXva scritto utilizzando il comando `\LaTex`.

3.2 Convenzioni tipografiche

I principali comandi utilizzati in LaTeXsono indicati sotto per argomento.
Per le liste:

- `\begin{itemize} \end{itemize}` dove tra i due comandi posso mettere i vari punti di una lista;
- gli oggetti della lista si indicano col comando `\item`;
- vi possono essere vari tipi di liste: `itemize` puntata, `enumerate` numerata e `trivlist` vuota.

Per le tabelle:

- `\begin{center}` inizialmente e `\end{center}` a chiusura della sezione dove riportare la tabella. Questo per avere una tabella centrata nella pagina;
- `\renewcommand\arraystretch{1.2}` allarga ogni riga dello spazio indicato. La tabella va racchiusa tra parentesi graffe;
- `\begin{table}[H] ... \end{table}` racchiude la sezione di tabella utile successivamente per riportare la capitolazione ed altre informazioni. `[H]` indica il posizionamento della tabella e nello specifico vuole essere inserita nel punto esatto dove è stata dichiarata dal codice;
- `\begin{tabular}{|c|c|c|} \end{tabular}` indica una tabella con 3 colonne e testo centrato. La barra verticale (|) indica che vi è una linea divisoria verticale tra le celle;
- per tabelle particolarmente lunghe e complesse (come le tabelle utilizzate per il tracciamento) si è utilizzato un pacchetto esterno per la loro gestione. Tale pacchetto è attuabile tramite il comando `\begin{longtable}contenuto...\end{longtable}`. Utilizzabile in sostituzione di `tabular`

- `\footnotesize \small \normal` per indicare la dimensione del testo all'interno della tabella. Sappiamo che se essa è ricca di informazioni avremo bisogno di un testo più piccolo per avere una visione completa nella pagina;
- `\caption{...}` all'interno delle parentesi graffe è possibile indicare un testo che verrà riportato sotto la tabella alla compilazione. Tale verbo va inserito all'esterno della zona *table* ma internamente a *tabular*;
- `\label{...}` permette di indicare con una stringa la tabella singolarmente.
- `\hline` indica una linea separatrice orizzontale;
- `&` è il simbolo separatore tra le celle orizzontalmente;
- `\\` passa alla riga successiva della tabella;
- `\minitab[c]{...}` è un comando implementato direttamente dal gruppo utile nel creare sottotabelle in particolare per dividere il testo.

Per le immagini:

- `\begin{figure}[H] \end{figure}` inizio e fine di una sezione con figure. `[H]` è similare all'uso per tabelle;
- `\centering` imposta centrale la figura;
- `\includegraphics{filegrafico}` comando per includere le immagini, è importante controllare il formato;
- `\caption{didascalia}` inserisce una didascalia sotto l'immagine;
- `\label{nome}` imposta il nome univoco dell'immagine.

3.3 Formati

- **Date**: saranno espresse nel formato italiano gg/mm/aaaa dove:
  - **gg**: indica il giorno, va scritto sempre con 2 cifre;
  - **mm**: rappresenta il mese, va scritto sempre con 2 cifre;
  - **aaaa**: rappresenta l'anno, va scritto sempre con 4 cifre.
- **Anni accademici**: si userà il formato aaaa1-aaaa2 dove:
  - **aaaa1**: indica l'anno solare di inizio, va scritto sempre con 4 cifre;
  - **aaaa2**: indica l'anno solare di fine, va scritto sempre con 4 cifre.
- **Orari**: verranno espressi secondo lo standard *ISO 8601* HH:MM dove:
  - **HH**: indica le ore, va scritto sempre con 2 cifre;
  - **MM**: indica i minuti, va scritto sempre con 2 cifre.
- **URL**: collegamento ad un indirizzo web deve essere scritto con un font di colore blu;
- **Sigle**: i nomi dei documenti potranno essere sostituiti dalle rispettive sigle:
  - **AdR** ad indicare il documento "Analisi dei Requisiti";

- **PdP** ad indicare il documento "Piano di Progetto";
- **PdQ** ad indicare il documento "Piano di Qualifica";
- **NdP** ad indicare il documento "Norme di Progetto";
- **Gl** ad indicare il documento "Glossario";
- **ST** ad indicare il documento "Specifica Tecnica";
- **SdF** ad indicare il documento "Studio di Fattibilità".

3.4 Struttura del documento

3.4.1 Frontespizio

Il frontespizio di ogni documento dovrà essere così strutturato:

- Logo-nome del gruppo;
- Titolo del documento;
- Versione del documento;
- Data di Rilascio;
- Nome e cognome dei redattori del documento;
- Nome e cognome dei revisori del documento;
- Nome e cognome di colui che valida il documento;
- Responsabile del documento, cioè uno dei redattori che si fa carico dell'interno iter;
- Uso del documento;
- Destinatari del documento;
- Sommario generico descrittivo.

3.4.2 Diario delle modifiche

La seconda pagina deve essere una tabella contenente i cambiamenti che sono stati effettuati nel documento.Tale sezione è esterno alla numerazione poiché appartiene alla sua parte strutturale.
Si intende specificare la localizzazione delle modifiche specificandole in modo numerico e non narrativo.
Deve essere così strutturata:

- **Versione**: la versione del documento dopo la modifica;
- **Modifica**: descrizione delle modifiche effettuate;
- **Nome**: nome e cognome dell'autore delle modifiche;
- **Data**: il giorno nel quale sono state apportate le modifiche.

La tabella è ordinata per versione in ordine decrescente, in modo che la prima riga sia l'ultima modifica eseguita, e quindi corrisponda alla versione attuale del documento.
La versione dei documenti aumenta di una unità a ogni modifica del documento della terza cifra. Al momento di verifica la terza cifra viene azzerata, incrementando la seconda di cinque unità. Nel caso di cambiamento importante, non verificato, invece di una sola unità.

3.4.3 Indice

La pagina successiva al diario delle modifiche deve essere l'indice del documento. Ogni documento tranne i verbali deve contenere l'indice che serve a dare una visione globale degli argomenti trattati.
Vi sono più indici specifici all'interno di un documento completo rappresentati e auto-generati con i comandi LaTeX:

- `\tableofcontents` per i capitolo principali e appendici;
- `\listoftables` per l'indicizzazione tabelle;
- `\listoffigures` per l'indicizzazione delle figure;

3.4.4 Formattazione generale

Tutte le altre pagine del documento, a partire dall'introduzione, dovranno rispettare la struttura di base che segue:

- Logo del gruppo: sarà posizionato sempre in alto a sinistra;
- Sezione corrente: nell'intestazione deve comparire il numero ed il nome della sezione in cui ci si trova. Questa informazione sarà in alto a destra;
- Nome del documento e versione: compare a piè di pagina, a sinistra;
- Numero di pagina e totali: compare a piè di pagina, a destra.

Non verranno quindi conteggiate la prima pagina, la sezione dedicata al diario delle modifiche e l'indice.
Particolari attenzioni vanno poste per il *Glossario* in cui le sezioni non saranno numerate e vengono indicate con una lettera.
Tutti i documenti che riportano un *Appendice* utilizzeranno il comando `\appendix` per indicarne l'inizio in cui comincia, fino a fine documento, le sezioni aggiunte.
Ogni pagina deve rispettare i seguenti margini:

- Superiore: 2cm;
- Inferiore: 2cm;
- Destro: 1cm;
- Sinistro: 1cm.

3.4.5 Sezioni del Documento

Si è scelto di seguire per la maggior parte le sezioni indicate negli esempi forniti dal committente. Nello specifico si utilizza:

- `\section{...}` per le sezioni principali;
- `\subsection{...}` per le sottosezioni;
- `\subsubsection{...}` per sotto-capitoli non influenti e che non vengono riportati nell'indice;
- `\section*{...}` l'elemento " * " prima della definizione di capitolo o sottocapitolo esclude la sezione dall'indicizzazione;

- `\appendix` è il comando che indica l'inizio dell'appendice, i cuoi capitoli successivi sono indicizzati in lettere.

Bisogna sottolineare il fatto che per documenti molto complessi in cui tutto il gruppo si è cimentato nella stesura le singole sezioni sono state spezzate in più file. È infatti grazie al comando `\newcommand{\docs}{./sezi` che per noi è possibile importare un file da una sotto-cartella sezioni e quindi lavorare su più file contemporaneamente senza preoccupazione di sovrascrittura. I documenti in cui è stata usata questa procedura sono la *Specifica Tecnica v3.0.0* e la *Definizione di Prodotto v2.0.0*.

3.4.6 Ciclo di vita di un documento

Ogni documento ha tre stadi in cui può trovarsi:

- In scrittura;
- Da verificare;
- Da validare.

Ogni stadio è seguito da uno specifico ruolo. Nel primo, la scrittura, a seconda del documento può essere svolto da tutti i ruoli che sono interessati al team e nello specifico al documento.
La verifica viene attuata solamente dal *verificatore* a seguito di un importante incremento del contenuto del documento.
Infine la validazione è congiunta tra un *verificatore* designato ad ogni documento e il *redattore* dello stesso. L'*amministratore* insieme al *Project manager* nella fase precedente alla revisione organizzeranno la consegna del documento finito. Molto spesso questa parte è compresa di ricompilazione e controllo ortografico automatizzato.

3.5 Classificazione documenti

3.5.1 Documenti formali

Tutti i documenti che sono stati approvati dal *Project Manager* sono da ritenersi formali.
Se un documento formale viene modificato, deve essere successivamente approvato nuovamente tramite processo di verifica in accordi con il responsabile del documento. Preventivamente scelto, insieme al validatore. Tali due ruoli non possono essere ricoperti dallo stesso membro del gruppo.

3.5.2 Documenti informali

Sono quei documenti che non sono stati approvati del Project Manager, poiché non lo necessitano e quindi sono ad esclusivo uso interno, o perché sono ancora in fase di sviluppo e verranno approvati successivamente.

3.6 Componenti grafiche

3.6.1 Tabelle

Tutte le tabelle presenti nella documentazione necessitano di una didascalia che le descrive brevemente e una stringa identificativa. Tramite comando precedentemente descritto gli viene auto-assegnato anche un numero identificativo.

3.6.2 Immagini

Le immagini come le tabelle dovranno essere accompagnate da una didascalia e parallelamente avranno stringa e numero identificativo.

3.7 Gestione di progetto

Il *Project Manager* ha la responsabilità di gestire l'intero progetto. Dovrà utilizzare gli strumenti in suo possesso per:

- Coordinare e pianificare le attività;
- Coordinare e gestire le risorse umane;
- Analizzare i rischi;
- Elaborare e monitorare i dati d'avanzamento.

3.7.1 Pianificazione delle attività

Per ogni periodo di lavoro il *Project Manager* deve stabilire le seguenti attività:

- Creare un calendario lavorativo;
- Inserire le attività da svolgere;
- Inserire eventuali dipendenze delle attività;
- Inserire le *milestone* indicante il termine previsto delle attività;
- Assegnare ad ogni attività le risorse necessarie.

3.7.2 Coordinamento delle attività

Una volta pianificate le attività, il *Project Manager* le assegna ad ognuno dei membri attraverso l'ausilio offerto da *Asana*. In principio viene prodotto un digramma $Gantt_g$ e poi popolato delle attività specifiche. Ogni componente può quindi verificare quali di questi compiti gli sono stati assegnati e modificarne lo stato, facendo in modo che il *Project Manager* possa seguire con facilità i progressi effettuati. Viene comunque tramite comunicazione informale tenuto sotto controllo lo stato dei processi genericamente.

3.7.3 Analisi dei rischi

Durante tutto l'avanzamento del progetto il *Project Manager* ha il compito di valutare e riportare eventuali rischi che possono venirsi a creare e che sono descritti nel *Piano di Progetto v4.0.0* attuando le adeguate contromisure descritte. Vengono nello specifico descritti genericamente nel *Registro dei Rischi* e in seguito riportati nello specifico nell'*Attuazione dei Rischi*.

3.7.4 Elaborazione dei dati

Il *Project Manager* è tenuto a creare grafici di pianificazione esplicativi dell'andamento delle attività e delle ore ricoperte da ogni ruolo e membro del gruppo. Tali grafici sono riportati nel *"Piano di Progetto v4.0.0"*.
Inoltre la collaborazione tra *Amministratore* e *Verificatori* riporteranno in forma tabellare i dati ricavati dai vari test statici e dinamici all'interno del *"Piano di Qualifica v4.0.0"*.

3.8 Processo di verifica qualità

In questa sezione verranno descritti gli standard minimi che dovranno rispettare processi, prodotti e documenti per essere adeguati. Le tecniche e strategie che dovranno utilizzare i *Verificatori* durante il loro lavoro per ottenere tali standard, in modo da uniformare le attività ed ottenere il risultato più efficace e efficiente possibile.

3.8.1 Procedure di controllo di qualità del processo

La qualità dei processi verrà garantita dall'applicazione del principio PDCA. Grazie a questo principio sarà possibile garantire non solo il controllo e la correttezza, ma anche il miglioramento costante della qualità. Come conseguenza diretta si otterrà il miglioramento del prodotto.
Per avere controllo dei processi, e di conseguenza qualità, è necessario che:

- vi sia controllo sull'operato dei membri del team e sui processi;
- i processi siano pianificati in modo dettagliato;
- le risorse siano ripartite in modo chiaro.

L'attuazione di questi punti è descritta nel documento *"Piano di progetto v4.0.0"*. Inoltre l'analisi costante della qualità del prodotto permette di monitorare in modo indiretto la qualità dei processi: se il prodotto è di bassa qualità sicuramente il processo è migliorabile.
Per valutare la qualità di un processo è fondamentale che questa sia quantificata, le metriche per fare ciò sono descritte nella sezione Descrizione.

3.8.2 Procedure di controllo di qualità di prodotto

Il controllo di qualità dei prodotti verrà garantito dai seguenti processi:

- **Software Quality Assurance (SQA)**: assicura che i processi siano appropriati per il progetto, che siano correttamente implementati. Prevede l'attuazione di tecniche di analisi statica e dinamica, descritte in seguito;
- **Verifica**: è il procedimento che controlla coerenza e correttezza dei prodotti dei processi. La verifica verrà eseguita costantemente durante tutta la durata del progetto;
- **Validazione**: ovvero la conferma che il sistema soddisfi i requisiti e sia conforme alle attese.

3.8.3 Organizzazione

Il processo di verifica inizia nel momento in cui la versione del prodotto di un processo cambia. Il diario delle modifiche aiuta a monitorare solo le sezioni che sono state modificate, ottimizzando così i tempi di verifica.
Viene verificata sia la qualità del processo che del prodotto da esso compiuto.
Ogni fase del progetto, descritta nel documento *"Piano di progetto v4.0.0"* produce un prodotto diverso, quindi necessita di diverse attività di verifica:

- **Analisi e Analisi di Dettaglio**: in questa fase si verifica che i processi e la documentazione prodotta rispettino le norme definite in questo documento e i tempi prestabiliti nel *"Piano di Progetto v4.0.0"*.

3.8.4 Pianificazione strategica e temporale

È essenziale che l'attività di verifica sia sistematica ed organizzata, al fine di evitare la propagazione di errori ed ottimizzare i tempi di sviluppo, per rispettare le scadenze elencate nel documento *Piano di Progetto v4.0.0*.
Ogni fase di redazione di documenti e di codifica deve essere preceduta da una fase di studio preliminare, al fine di ridurre la possibilità di errori di natura concettuale o tecnica, e favorendo l'attività dei verificatori.
In seguito vengono riportate le scadenze fissate:

- **Revisioni formali**:

  - **Revisione dei Requisiti**: 24/01/2017;
  - **Revisione di Accettazione**: 27/06/2017.

- **Revisioni di progresso**:
  - **Revisione di Progettazione**: 13/03/2017;
  - **Revisione di Qualifica**: 08/05/2017.

3.8.5 Responsabilità

Al fine di garantire un processo di verifica efficace ed efficiente vengono attribuite delle responsabilità all'interno del gruppo di progetto.
I ruoli che intervengono nel processo di verifica sono il *Project Manager* e il *Verificatore*.

3.8.6 Risorse

Per realizzare il prodotto sono necessarie le seguenti risorse:

- **Risorse umane**: vengono descritte dettagliatamente nel documento *"Piano di Progetto v4.0.0"*. E sono:
  - Project Manager;
  - Amministratore;
  - Analista;
  - Progettista;
  - Programmatore;
  - Verificatore.
- **Risorse software**: sono necessari strumenti software. In particolare per:
  - scrivere la documentazione in formato LaTeX;
  - creare diagrammi *UML* tramite applicativo $Asana_{g}$;
  - automatizzare le verifiche (ortografiche);
  - sviluppare nei linguaggi di programmazione scelti tramite IDE preinstallati nelle macchine personali o uso universitario;
  - gestire i test automatici che effettuiamo sul codice.
- **Risorse hardware**: sono necessari computer aventi applicativi descritti in questo documento.

3.8.7 Tecniche di analisi -Analisi statica

L'analisi statica è il processo eseguito senza una effettiva esecuzione dei programmi e quindi non automatizzato. Viene eseguita sia sul codice che sui documenti durante tutta la fase di sviluppo.
Può essere applicata nei seguenti modi:

- **Walkthrough**: è una lettura del documento da verificare cercando errori senza un'idea precisa di cosa si sta cercando. È utile nelle prime fasi di sviluppo per stilare una lista degli errori più frequenti da condividere con il gruppo;
- **Inspection**: è una ricerca mirata degli errori che sono stati evidenziati nella lista di controllo, che può essere espansa qualora se ne palesi la necessità.

3.8.8 Lista di controllo

Durante la fase di *Walkthrough* sono stati rilevati questi come errori più frequenti:

- Norme Stilistiche:
  - un elemento dell'elenco inizia con lettera minuscola;
  - usare descrizioni opportune;
  - tabelle con assenza di descrizione;
  - testo non indentato;
  - tabelle e immagini non centrate;
  - riferimenti non aggiornati di versione;
  - non segnalati termini da glossario;
  - non segnati in corsivo inglesismi, ecc;
  - capo riga e spazi prima di ogni sezione, sottosezione e sotto-sottosezione.
- Italiano:
  - scorretto utilizzo di accenti acuti e gravi;
  - punteggiatura nelle liste;
  - ortografia generica;
  - fluidità delle frasi.
- LaTeX:
  - controllo spaziatura delle tabelle (prima, nel mezzo e dopo);
  - controllo delle liste;
  - controllo celle delle tabelle;
  - termini non evidenziati (corsivo e/o grassetto).
- UML:
  - direzione delle frecce del diagramma errate;
  - specificare la versione;
  - tipologia di frecce;
  - attenzione particolare a dipendenze circolari;
  - creazione di grafici chiari e con minor sovrapposizione di frecce;
  - evitare interfacce con funzioni.
- Diagrammi dei Package:
  - relazioni tra package;
  - diagrammi troppo vasti e con troppe specifiche;
  - corretti tipi di relazioni;
  - utilizzare solo notazioni standard.
- Interfacce:

  - nomi delle classi specifici;
  - colori per gruppi di classi omogenee.

- Diagrammi di attività:
  - uso dettagliato;
  - diagrammi consoni alla tipologia per cui sono realizzati oltre che al documento indirizzato;
  - i verbi devono indicare azioni;
  - non usare fork e join insieme.

- Design pattern:
  - deve essere sempre contestualizzato, la descrizione non basta.

- Tecnologie:
  - Descrizione pro e contro più dettagliate;
  - Descrizioni specifiche e all'ottica di progetto.

3.8.9 Verifica documenti

Il *Verificatore* nei documenti dovrà:

- Controllare nel diario delle modifiche i cambiamenti avvenuti dall'ultima volta;
- Controllare nelle specifiche revisioni gli errori principali;
- Cercare eventuali discrepanze nelle parti che sono state modificate del documento;
- Fare un resoconto degli errori riscontarti, in modo che il *Project Manager* possa creare un *ticket* per la correzione;
- Aggiornare il registro delle modifiche e la versione del documento.

3.8.10 Verifica requisiti

Il verificatore, nel controllare i requisiti dovrà porre la propria attenzione su:

- Correttezza ortografica e lessicale del testo;
- Errori grafici generici;
- Atomicità dei requisiti foglia.

3.8.11 Verifica diagrammi UML

Nel controllare i diagrammi *UML*, il *Verificatore* dovrà prestare attenzione a questi elementi:

- Correttezza ortografica e lessicale del testo;
- Correttezza del formalismo grafico utilizzato;
- Correttezza grafica e di esposizione;
- Correttezza logica del programma.

3.8.12 Tecniche di analisi - Analisi dinamica

L'analisi dinamica viene eseguita sulle componenti software, testando il comportamento dell'applicativo. I test, per essere attendibili, devono essere ripetibili. Devono cioè produrre lo stesso risultato se provati nello stesso ambiente e con gli stessi input. Questa condizione è fondamentale per poter considerare attendibile un test che viene effettuato. Un test deve inoltre poter essere riprodotto da persone che non hanno scritto e che quindi non conoscono il codice. Questo garantisce efficienza e leggibilità del codice. I tipi di test che verranno eseguiti saranno:

- **Test di unità**: consiste nel controllo dei moduli che costituiscono l'applicativo;
- **Test di integrazione**: consiste nella verifica che i moduli, una volta collegati tra loro, diano il risultato previsto. Verifica inoltre il corretto funzionamento di librerie e framework esterni;
- **Test di sistema**: consiste nella validazione del prodotto software, una volta giunto ad uno stadio definitivo;
- **Test di regressione**: consiste nell'eseguire a cascata tutti i test collegati ad una componente appena modificata;
- **Test di accettazione**: consiste nel collaudo del software con il proponente prima del rilascio.

3.9 Metriche per il processo di Verifica

3.9.1 Descrizione

Qui di seguito vengono elencate le misure e le metriche che verranno adottate nel processo di verifica. Il processo di verifica deve essere verificabile e quantificato secondo metriche stabilite a priori.
La verifica attiene alla coerenza, completezza e correttezza del prodotto. È un processo che si applica ad ogni "segmento" temporale di un progetto (ad ogni prodotto intermedio) per accertare che le attività svolte in tale segmento non abbiano introdotto errori nel prodotto.

3.9.2 Misure e metriche - Metriche per i processi

MP1 - SPICE Il modello ISO/IEC 15504, conosciuto anche come SPICE (Software Process Improvement and Capability Determination), è lo standard di riferimento per valutazione della qualità dei processi con il fine di migliorarli e permette di misurare oggettivamente e indipendentemente l'attitudine di ogni processo tramite le loro caratteristiche. Si va quindi a prendere in esame i risultati di ogni valutazione. Queste devono essere :

- ripetibili;
- oggettive, quindi con una visione obbiettiva;
- comparabili;
- utili al miglioramento dei processi.

Per maggiori dettagli si consiglia di visionare l'appendice del *Piano di Qualifica v4.0.0*.

Schedule Variance (SV) Indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate nella baseline. È un indicatore di efficacia soprattutto nei confronti del Cliente. Se $SV > 0$ significa che il progetto sta producendo con maggior velocità a quanto pianificato, viceversa se negativo.

Budget Variance (BV) Indica se alla data corrente si è speso di più o di meno rispetto a quanto previsto a budget alla data corrente. È un indicatore che ha un valore unicamente contabile e finanziario. Se BV > 0 significa che il progetto sta spendendo il proprio budget con minor velocità di quanto pianificato, viceversa se negativo.

3.9.3 Misure e metriche - Metriche per i documenti

MPPD1 - Gulpease L'indice Gulpease è un indice, sviluppato per valutare la lingua italiana, che valuta la complessità e la leggibilità del testo. Rispetto ad altri ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. L'indice di Gulpease considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere.
L'indice è calcolato con la seguente formula:

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

I risultati sono compresi tra 0 e 100, dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. Va notato che questo indice non considera che il testo sia comprensibile o meno, in effetti anche una frase con parole casuali potrebbe avere un ottimo indice Gulpease. I documenti perciò vanno valutati da un essere umano per rendere le frasi semplici ma comprensibili.

MPPD2 - Errori ortografici corretti Il termine ortografia deriva dal greco e significa: retta maniera di scrivere e comprende l'insieme complesso delle regole che servono a scrivere bene. Si riferisce al corretto modo di usare le consonanti, le lettere maiuscole o minuscole, la giusta accentazione, ecc.
Nel nostro caso avendo installato un vocabolario *open source*$_{\text{g}}$ e inserendo all'interno del software per la gestione di file LaTeXè possibile un'individuazione automatica degli errori ortografici. Così da poter controllare tutti i termini errati e avviare una correzione completa.

3.9.4 Misure e metriche - Metriche per il software

MPPS1 - Copertura Requisiti Obbligatori Monitora costantemente la percentuale di requisiti obbligatori soddisfatti.
Per requisiti obbligatori si intente tutti quelli che rientrano all'interno delle *Analisi dei Requisiti v3.0.0* che hanno un grado di urgenza massimo (*0*). Essi sono necessari ai fini del progetto del funzionamento del prodotto e richiesti esplicitamente dal proponente. Vengono infatti chiamati requisiti esterni (cioè volti al dominio di pertinenza) ed interni (cioè ricavati dal capitolato).
La metrica è calcolata tramite la seguente formula:

$$\frac{\#\ \text{requisiti obbligatori soddisfatti}}{\#\ \text{totale requisiti obbligatori}}$$

Tale calcolo viene effettuato contando manualmente i requisiti. Si intende raggiungere un risultato massimo.

MPPS2 - Copertura Requisiti Accettati Monitora costantemente la percentuale di requisiti desiderabili e facoltativi soddisfatti.
Per requisiti accettati si intendono la somma dei di tutti i requisiti, non solo quelli obbligatori. Tali non sono estesi anche ai requisiti opzionali e desiderabili. Quindi non si intende coprire totalmente il totale di questi.

Sarà attraverso una combinazione tra *Piano di Progetto v4.0.0* e *Definizione di Prodotto v2.0.0* che deciderà se è possibile consegnare quanti più possibili requisiti, senza andare oltre le risorse prestabilite. La metrica è calcolata tramite la seguente formula:

$$\frac{\#\text{ requisiti accettati soddisfatti}}{\#\text{ totale requisiti accettati}}$$

Il calcolo dovrà essere effettuato con aritmetica manuale senza l'uso di strumenti automatizzati.

MPPS3 - Numero di campi dati per classe Considera il numero totale di attributi presenti all'interno di una classe. Un valore elevato può indicare che tale classe si fa carico di una quantità eccessiva di responsabilità; potrebbe quindi convenire scorporare una parte di essa in una seconda classe tramite incapsulamento, ridefinendone le interfacce.
Il numero di campi dati per classe è calcolato manualmente con uno studio specifico dei package e delle classi implementate. Ai fini di una maggior flessibilità e per distribuire il lavoro a più membri si è scelto di suddividere l'attività di verifica ai due principali blocchi del progetto : *Front-End* e *Back-End*.ù

MPPS4 - Grado di Accoppiamento Indica quanto le classi sono dipendenti tra loro rispetto al sistema totale. È utile per valutare l'instabilità del software.
Il grado di accoppiamento è misurabile interpretando le componenti di un sistema come i nodi di un grafo orientato dove gli archi sono dipendenze di una componente nei confronti di un'altra. Il numero di archi entranti in una componente (*fan-in*) è indice di utilità, mentre il numero di archi uscenti (*fan-out*) è indice di dipendenza da altre componenti. Riguardo alla riusabilità, infine, è bene notare che costituisce un guadagno soltanto nel lungo termine, mentre nel breve termine è un puro costo. L'indice di accoppiamento è il risultato di due indici:

- **Accoppiamento afferente**: è il numero di classi esterne ad un package che dipendono da classi interne ad esso. Un valore troppo elevato indica un altro grado di dipendenza, che potrebbe portare a criticità nel caso di modifiche;
- **Accoppiamento efferente**: è il numero di classi interne al package dipendenti da classi esterne. Un valore troppo alto è sintomo di scarsa progettazione.

Il calcolo dell'indice di stabilità, ottenuto unendo Accoppiamento afferente e Accoppiamento efferente avviene secondo la seguente formula:

$$I = \frac{Ce}{Ca + Ce}$$

MPPS5 - Complessità ciclomatica La complessità ciclomatica è una metrica software utilizzata per misurare la complessità di un programma. Misura direttamente il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso.
La complessità ciclomatica è calcolata utilizzando il grafo di controllo di flusso del programma: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni, mentre gli archi connettono due nodi se il secondo gruppo di istruzioni può essere eseguito immediatamente dopo il primo gruppo.
Alti valori di questo indice indicano che il codice è poco manutenibile e difficile da testare, mentre valori bassi potrebbero indicare una scarsa efficienza nei metodi.

MPPS6 - Numero di livelli di annidamento Numero di livelli di annidamento va ridotto il più possibile in modo da limitare l'accoppiamento. Con le tecnologie adottate è difficile utilizzare classi astratte ma ci siamo concentrati sulla costruzione di interfacce.

È fortemente consigliato utilizzare l'indentazione per segnalare visivamente la presenza di strutture annidate una dentro l'altra: la quantità di indentazione da usare è proporzionale al livello di annidamento (in questo esempio vengono usati due spazi per ciascun livello).
Nel caso di cicli annidati, il ciclo più interno viene interamente ripetuto ad ogni iterazione del ciclo che lo contiene, per cui il numero totale di iterazioni eseguite è il prodotto del numero di iterazioni di ciascun ciclo.
Un numero troppo elevato può comportare difficoltà nella verifica e nella manutenibilità del codice.

MPPS7 - Rapporto linee di commento su linee di codice Affinché il codice sia più facilmente comprensibile e manutenibile è necessario che sia adeguatamente commentato. Un rapporto troppo basso indica una carenza di informazioni necessarie alla comprensione del codice scritto.
Durante il processo di compilazione queste istruzioni sono ignorate e di conseguenza non pesano computazionalmente sulla grandezza dell'eseguibile prodotto.
Queste particolari notazioni hanno molta importanza, soprattutto se il programma è sviluppato da persone diverse e in tempi diversi, aumentandone la leggibilità/intelligibilità al lettore e favorendone così la manutenzione.
I commenti vengono anche usati per inibire l'esecuzione di alcune porzioni di codice, solitamente lo si fa in previsione di una successiva riabilitazione del codice inibito.
L'impegno a inserire linee di commento è lasciato al *Programmatore* che deve stendere un più alto possibili di linee di commento per codice.

MPPS8 - Statement Coverage Indica la percentuale di istruzioni che vengono eseguite durante i test rispetto al totale di linee di codice. Maggiore è il numero di linee testate, più alta sarà la possibilità che eventuali errori vengano individuati e risolti.
Un valore troppo basso indica che troppe istruzioni non vengono testate e non si può sapere se esse funzionino correttamente.

MPPS9 - Branch Coverage Indica la percentuale del numero di rami del flusso di controllo che vengono attraversati complessivamente almeno una volta. Anche in questa metrica più alti saranno i $branch_{\text{g}}$ percorsi, più alta sarà la possibilità di trovare eventuali errori e fallimenti.
Un valore minore al range di accettazione l'applicazione non potrà essere considerata abilitata e quindi utilizzabile.

MPPS10 - Validazione W3C Per validazione si intende confrontare il codice *HTML* e *CSS* di una pagina o applicazione web con degli standard, delle specifiche, codificate ed approvate dal $W3C_{\text{g}}$.
Una pagina web per essere accessibile ed usabile deve essere valida secondo le norme del W3C. L'applicativo web deve superare tutti i test del validatore automatico W3C per quanto riguarda gli errori gravi, deve inoltre evitare quanti più warning possibili.
Lo strumento di valutazione offerto dall'ente W3C che si trova alla pagina `https://validator.w3.org/`.

MPPS11 - Euristica di Nielsen Le euristiche di Nielsen sono 10 euristiche studiate per controllare in modo metodico l'usabilità di un sito web, derivano dall'applicazione di tecniche di analisi fattoriale su 249 problemi di usabilità. Sono semplici da applicare e mostrano benefici nelle fasi iniziali di un progetto, ma sono informali e non sono automatizzabili in quanto richiedono un giudizio umano. Essendo parametri non completamente oggettivi, può capitare che alcune pagine siano usabili anche non rispettando tutte le euristiche, si è quindi posto un limite minimo di 6 euristiche rispettate.
Qui in seguito vengono descritte tali norme:

1. **Visibilità dello stato del sistema**: il sistema deve sempre tenere informato l'utente su cosa sta facendo, fornendo un adeguato feedback in un tempo ragionevole.

   sapere se un oggetto è un link e dove porta;

   icona o testo sotto-intensificato significa che la funzione non è disponibile;

   presenza di un segnale di attività in corso (clessidra, barra di caricamento, messaggio testuale, etc.).

2. **Corrispondenza tra sistema e mondo reale**: il sistema deve parlare il linguaggio dell'utente, con parole, frasi e concetti a lui familiari;

   uso di messaggi testuali, icone, azioni dal significato condiviso da tutti ("salva con nome", icona "cestino", azione "copia e incolla";

   garantire l'associazione tra oggetti e informazione.

3. **Controllo e libertà**: l'utente deve avere il controllo del contenuto informativo e muoversi liberamente tra i vari argomenti.

   evitare procedure costrittive troppo lunghe (iscrizioni);

   evitare percorsi predefiniti senza possibili scorciatoie.

   evitare azioni non volute dall'utente (apertura automatica di pagine non richieste);

4. **Consistenza e standard**: l'utente deve aspettarsi che le convenzioni del sistema siano valide per tutta l'interfaccia.

   riportare in ogni pagina alcuni elementi di riconoscimento (logo, stile grafico, etc.);

   dare la sensazione di essere sempre nello stesso ambiente.

5. **Prevenzione dell'errore**: evitare di porre l'utente in situazione ambigue, critiche e che possono portare all'errore.

   dare la possibilità di tornare indietro;

   evitare che la non comprensione induca in errore.

6. **Riconoscimento anziché ricordo**: le istruzioni per l'uso del sistema devono essere ben visibili e facilmente recuperabili.

   produrre *layout* semplici e schematici;

   non contare sulla capacità dell'utente di ricordare il posizionamento degli oggetti che caratterizzano le pagine;

   evitare che l'utente riscopra ogni volta l'interfaccia.

7. **Flessibilità d'uso**: offrire all'utente la possibilità di un uso differenziale (a seconda della sua esperienza) dell'interfaccia.

   offrire una navigazione gerarchica per i meno esperti;

   offrire delle scorciatoie per i più esperti.

8. **Design e estetica minimalista**: dare maggior importanza al contenuto che all'estetica.

   evitare di accentuare oggetti irrilevanti o raramente necessari (immagini grandi, etc.);

   evitare che il contenuto informativo della pagina sia messo in secondo piano;

   evitare che l'utente si distragga o si confonda.

9. **Aiuto all'utente**: aiutare l'utente a riconoscere, diagnosticare e recuperare l'errore.

   i messaggi di errore devono essere espressi in linguaggio comprensibile (senza codici);

   i messaggi di errore devono indicare in modo preciso il problema e suggerire una soluzione;

   chiedere conferma per un'azione importante;

10. **Documentazione**: anche se il sistema dovrebbe essere usabile senza documentazione è preferibile che essa sia disponibile.

    deve essere facile da reperire;

    focalizzata sul compito dell'utente;

    strutturata in un insieme di passi comprensibili.

3.10 Validazione

Il principio assoluto è scrivere documenti e programmi verificabili. Dopo una serie di verifiche che seguono tutto il ciclo di vita del prodotto la validazione è il verdetto finale che lo rende pubblicabile senza timore. Quindi vorremo che la validazione sia auto-avverante
Il *testing*, negli approcci dinamici, prevede esecuzione del software o prototipi al fine di scoprire difetti e assicura quindi la validazione del programma.
Al momento, non avendo prodotto software e quindi impossibile eseguire dell'analisi dinamica, si è scelto come regola generale di utilizzare una doppia revisione. Per doppia revisione si intende:

- Temporale: cioè il documento è revisionato più volte nel corso del tempo se importante, anche se non completato;
- Umana: cioè al completamento ogni atto è controllato da due membri del team diversi e che non l'abbiano redatto (per evitare conflitti d'interesse).

Questa scelta è al momento il metodo di validazione del gruppo.

4 Processi organizzativi

In questa sezione verranno esposti in particolare gli ambienti di lavoro che abbiamo scelto di utilizzare e come verranno gestite le varie attività.

4.1 Sistema rotativo dei ruoli

Il progetto a scopo didattico ha l'obbligo di far interpretare ogni ruolo a tutti i partecipanti del team. Quindi dopo aver costruito una prima lista di attività e pianificato queste ultime ogni obbiettivo viene risolto da un sottogruppo (con ruoli specifici). Scelto il *Project Manager* di periodo i seguenti ruoli vengono scelti per preferenza e per attitudini personali, in quanto aiuta a lavorare efficientemente e con risultati migliori. Il *Project Manager* vaglia oggettivamente ogni sfaccettatura per risolvere i *backlog* richiedendo le risorse di cui ha bisogno e rendere la formazione il più produttiva possibile.
Ruoli e divisione delle mansioni sono descritti a pieno e nello specifico nel *Piano di Progetto v4.0.0*.

4.2 Processi di coordinamento

4.2.1 Comunicazioni Interne ed Esterne

- **Interne**: Per le comunicazioni interne è stato creato un gruppo su *Telegram*$_{\text{g}}$, un'app di messaggistica istantanea. Questo canale sarà utilizzato solamente per le comunicazioni interne informali.

- **Esterne**: Per le comunicazioni esterne è stata creata la casella di posta *sweg.group@gmail.com*. Questo può essere l'unico canale utilizzabile. Solo il *Project Manager* può avere delle interazioni con l'esterno, sarà poi compito suo riferire il contenuto delle discussioni agli altri componenti del gruppo.

4.2.2 Riunioni

- **Interne**: Le convocazioni alle riunioni sono notificate dal *Project Manager* e confermate informalmente dai componenti del gruppo. Ogni membro è tenuto a partecipare a meno di una giustificazione valida. Una riunione è fissata per discutere di argomenti di interesse generale di tutti i componenti. È però possibile che ci sia la necessità di effettuare delle riunioni alle quali non è richiesta la presenza di tutti i membri ma solo di alcuni di essi; in questo caso il *Project Manager* autorizza l'incontro e poi vi prenderanno parte solo le persone interessate, le quali dovranno poi informare gli altri componenti del gruppo delle decisioni prese.

- **Esterne**: È compito del *Project Manager* fissare degli incontri con il proponete ed il committente. Tutti i membri del gruppo ne devono essere informati. Ad ogni incontro verrà scelto un segretario il quale avrà il compito di redigere un verbale che verrà poi inviato a tutti i componenti.
  Ogni verbale redatto dovrà avere la seguente composizione:

  1. Data e ora;
  2. Luogo;
  3. Partecipanti interni;
  4. Partecipanti esterni;
  5. Argomenti trattati;
  6. Domande e risposte.

4.3 Sistema operativo

Il progetto e l'insieme delle applicazioni di supporto non sono progettate su un particolare sistema operativo. I componenti del team avranno quindi la possibilità di installare i propri *client* dei mezzi in qualunque macchina: *Windows*, *Mac OS X*, *Linux*.

4.4 Applicativo Guida

Si è scelto per il coordinamento:

- Applicativo web Asana;
- Repository Git;
- Comunicazioni tramite Telegram.

4.5 Documentazione

Per la documentazione inizialmente si redige una prima stesura su un documento semplice in qualsiasi text editor per poi elaborarlo in linguaggio LaTeX.

4.5.1 Latex

È stato scelto di scrivere i documenti in linguaggio LaTeXperché:

- Inerente al nostro percorso di studi essendo un linguaggio compilabile. Si usa di più la mente logica che serve a costruire un programma informatico;
- I documenti scritti con il *mark-up*$_{\text{g}}$ di Latex posso venire composti direttamente in pdf, oltre ad altri formati;
- Sono più precisi gli automatismi di creazione di indice e l'impaginazione;
- È distribuito con una licenza di software libero;
- Permette di definire template;
- Consente di suddividere il documento in file separati.

4.5.2 Editor

L'editor di testo utilizzato multi-piattaforma è *TexStudio*. Programma efficiente con compilazione rapida. Offre una insieme di mezzi per velocizzare e facilitare la creazione della documentazione.
È stato aggiunto il dizionario italiano per una correzione immediata e automatica di errori ortografici.

4.5.3 Diagrammi UML

Per la creazione dei diagrammi *UML* abbiamo deciso di utilizzare due diversi software. Come punto di riferimento abbiamo l'applicazione usata dal professore nelle ore di lezione. Inserendo la mail concessaci dell'università è possibile avere una versione gratuita del software *Astah*.
Progettato per il design dei principali grafici a uso informatico. Oltre a questo abbiamo deciso di utilizzare il software *Draw.io* che e' disponibile gratuitamente online. Abbiamo deciso di scegliere questo software per i seguenti motivi:

- Essendo un'applicazione web e' usufruibile sulla maggior parte dei dispositivi;

- Consente di utilizzare solo questa applicazione per la costruzione di vari modelli grafici;
- Permette di utilizzare dei template già esistenti a secondo dello scopo (informatica, ingegneria ad esempio);
- Ha disponibili molte immagini vettoriali;
- Si può esportare in diversi formati compreso HTML.

4.5.4 Verifica

Tutti i documenti creati con LaTeXsono automaticamente controllati con uno script per il controllo ortografico.

4.6 Processo di Pianificazione

4.6.1 Diagrammi

Per la pianificazione delle attività è stato scelto il software *Instagantt*. Software web per la pianificazione delle attività tramite diagramma Gantt.
È stata scelta tale applicazione perché compatibile con il sistema di ticketing offerto da *Asana*. Infatti ogni attività e il suo responsabile è direttamente collegato al diagramma.
Le principali funzioni sono:

- data di inizio e fine dell'attività;
- gruppi e sottogruppi di attività;
- milestone e date importanti;
- segnare lo stato di un'attività;
- colorare attività per suddividerle (es. urgenza o tipologia);
- segnare date di vacanza e ferie;
- impostare dipendenze tra le attività.

4.6.2 Attività

Asana è un'applicazione web e mobile che permette la suddivisione delle attività in modo strutturato. Offre un'organizzazione ad albero suddivisa in questo modo:

- Creare più *Workspace*$_g$;
- Creare *Project* all'interno di *Workspace*;
- Creare *Task* all'interno di un *Project*;
- Creare *Sub-Task* all'interno di un *Task*.

Alcune delle funzionalità offerte sono:

- Assegnare un *Task* ad uno o più individui;
- Assegnare una data di scadenza;

- Vedere le attività in scadenza sul calendario;
- Seguire lo stato di avanzamento dei *Task*;
- Avere un riepilogo dei *Task*;
- Riceve e-mail automatiche con l'aggiornamento dei *Task* che si seguono.

I nomi delle etichette relativi ai Task dovranno seguire la seguente forma [elemento di interesse][messaggio]:

- **Elemento di interesse**: si riferisce al documento o all'ambito al quale appartiene il Task. Nel caso un cui si tratti di un documento, dovranno essere messe le iniziali del suo nome.
  Ad esempio NdP, per indicare *Norme di Progetto v4.0.0*. Nel secondo caso si userà il nome completo dell'ambito che si sta trattando. In entrambi i casi viene utilizzata la lettera maiuscola;
- **Messaggio**: breve messaggio quanto più esaustivo possibile per indicare la problematica da risolvere o il compito da eseguire.

4.6.3 Controllo costi

Per avere una visione generale dei costi di questo progetto è stato scelto il foglio di Google. E' stato scelto questo strumento per i seguenti motivi:

- Permette di tenere sotto controllo i costi ed i tempi;
- Crea i grafici che vengono inseriti nel documento *Piano di Progetto v4.0.0*;
- Permette di impostare le formule desiderate e di lavorare in modo automatico.

4.7 Versionamento

Il software di versionamento scelto è *Git*. Le motivazioni a base della scelta è perché *Git*:

- È strutturato come un File System$_g$;
- Salva uno snapshot del progetto a ogni commit;
- È veloce e distribuito avendo in locale la copia della repository per ogni utente;
- Sfrutta facilmente le operazioni di merge e branch;
- È già stato usato da alcuni membri del team.

All'interno della *repository* vi sono due macro sezioni: la sezione documentazione e la sezione progetto.
Ogni documento è composto dai numeri X.Y.Z. Convenzionalmente abbiamo deciso che il numero Z aumenta di uno ogni qual volta viene effettuata una modifica da parte di un componente del gruppo, il numero Y viene maggiorato di uno tutte le volte che il documento viene verificato, e viene azzerato Z; X invece viene aumentato quando si ritiene che il documento sia pronto per essere sottoposto alla revisione.

4.8 Tracciamento

Il tracciamento dimostra completezza ed economicità della soluzione. Se tracciati tutti i requisiti conseguentemente :

- Possiamo soddisfarli;
- Nessuna funzionalità è superflua;
- Nessun comportamento è ingiustificato;

Nel documento di *Specifica Tecnica* deve essere presente anche il tracciamento dei componenti; mentre nella *Definizione di Prodotto* deve essere presente il tracciamento dei test. In questo modo ogni requisito è tracciato al componente che lo soddisfa, e sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni requisito venga soddisfatto.

4.9 Ticket-ing

Il web applicativo Asana è stato scelto per la comunicazione e il ticketing. È stata scelta questa piattaforma per la sua esistenza sia in versione web sia mobile. Inoltre è collegata alle email personali dei membri del gruppo così da avere una notifica sui compiti seguiti.
Oltre ovviamente a essere gratuito Asana offre la possibilità di collegare direttamente file e link su ogni singolo ticket.
Cosa infine forse più importante è l'associazione al calendario così da avere sotto controllo migliore gli eventi datati.

4.10 Repository

4.10.1 Repository documentazione

La *repository* della documentazione è composta dalle varie cartelle :

- AnalisiDeiRequisiti;
- DefinizioneDiProdotto;
- Glossario;
- ManualeUtente;
- ManualeAmministratore;
- ManualeSviluppatore
- NormeDiProgetto;
- PianoDiProgetto;
- PianoDiQualifica;
- SpecificaTecnica;
- StudioDiFattibilità;
- Verbali.

I file all'interno possono essere di qualsiasi genere, anche appunti. Nella capitolo della documentazione è stato trattato il modo in cui crearli ed elaborarli.

4.10.2 Repository progetto

La *repository* del codice che compone il nostro progetto è composta dalle seguenti cartelle:

- APIM;
- Templates;
- Backend;

4.11 Processo di miglioramento dei Processi

Per processo di miglioramento *SWEg Group* intende utilizzare i tre principali standard di perfezionamento dei processi, in primis per poterli stimare e rendere misurabili, così da potersi elevare.
Tali strumenti sono spiegati in maniera teorica e più dettagliata nell'*Appendice* del *Piano di Qualifica v 4.0.0*.

4.11.1 PDCA o Ciclo di Deming

È l'acronimo di *Plan-Do-Check-Act*$_{\text{g}}$, le quattro fasi per il miglioramento del processo che intendiamo applicare ai finire di perfezionare la qualità del processo. Nello specifico intendiamo:

- **Plan**: pianificare e stabile gli obbiettivi per il miglioramento di singole attività;
- **Do**: eseguiamo ciò che abbiamo pianificato ai fini di perfezionamento;
- **Check**: attraverso metriche precedentemente descritte, confrontiamo i risultati riportati dell'esecuzione del miglioramento (eseguiti nella fase *Do*). Se tali obbiettivi sono stati raggiunti è possibile passare alla fase successiva. Altrimenti, segnando le cause che hanno portato all'insuccesso, si ripete il ciclo.
- **Act**: vengono applicate le azioni correttive prima implementate. La soluzione prodotta diventa la nuova *baseline*$_{\text{g}}$ ed ora è possibile ripetere il ciclo con un nuovo processo ad obbiettivo.

4.11.2 ISO/IEC 9126

Il modello ISO/IEC 9126 è uno standard che valuta la qualità del software che il nostro team ha voluto prendere come esempio per migliorare il processo.
Esso definisce degli spunti per ogni parte, cioè:

- **Qualità interna:** applicabile al codice e si attua in particolare un analisi statica.
- **Qualità esterna:** vieni rivelata tramiti analisi dinamica. Si attua sul software in esecuzione tramite processi automatizzati (*test*$_{\text{g}}$)
- **Qualità in uso:** metriche in uso solo a prodotto finito che misurano quantitativamente caratteristiche qualitative.

E sette principali caratteristiche :

1. funzionalità;
2. affidabilità;
3. efficienza;

4. usabilità;
5. manutenibilità;
6. portabilità.

4.11.3 Standard ISO/IEC 15504 (SPICE)

Questo standard ha il fine di migliorare i processi software, nello specifico ci da degli strumenti per misurare oggettivamente e indipendentemente l'attitudine di ogni processo tramite le loro caratteristiche.
I processi vengono valutati utilizzando la scala:

- **Incompleto (Livello 0);** il processo è incompleto in quanto non è stato implementato o fallisce nel raggiungere il proprio obiettivo;
- **Eseguito (Livello 1):** il processo è stato implementato e ha successo nel raggiungere il proprio obiettivo.
- **Gestito (Livello 2):** il processo è implementato in maniera organizzata tramite pianificazione, controllo e correzioni; i suoi prodotti sono sicuri.
- *Stabilito (Livello 3):* il processo opera entro limiti definiti per raggiungere i propri risultati.
- **Predicibile (Livello 4):** il processo è oggetto di miglioramento continuo per raggiungere gli obiettivi di progetto/aziendali.
- **Ottimizzato (Livello 5):** il processo è oggetto di miglioramento continuo per raggiungere gli obiettivi di progetto/aziendali.

4.12 Processo di formazione

Il processo di formazione consiste nel fornire i mezzi per i membri del team per essere pronti e in grado di lavorare efficientemente ed efficacemente nello sviluppo e nella gestione del progetto.
Nel nostro particolare caso la formazione del gruppo è avvenuta in maniera automatica in quanto la scelta dei tali è avvenuta per cause di forza maggiore. Alcuni non accettati in altri gruppi già formati, altri impreparati alla selezione.
Lo *SWEg Group* è l'unica figura ad avere cinque soli membri sotto la soglia minima di partecipazione al progetto. La causa è in particolare il ritiro da parte di un membro essendo studente in $erasmus_{g}$ e non il linea con i requisiti del progetto.
Il materiale di formazione e riferimenti degni di nota sono stati consigliati in aula nelle lezioni teoriche della materia *Ingegneria del Software*. Altre strumenti citati nel *Riferimenti Normativi* e *Riferimenti Informativi* sono ricavati tramite ricerca *web*. Inoltre la rete studentesca ha creato un gruppo specifico per la materia su piattaforma $Telegram_{g}$ in cui la comunicazione ha permesso lo scambio di informazioni utili per il raggiungimento di un obbiettivo comune.

5 Appendice A

5.1 Ruoli all'interno del progetto

Ad ogni componente è assegnato un ruolo all'interno del progetto che varia a seconda di regole che sono riportate in seguito. È importante che non vi sia conflitto di interesse, infatti non è possibile validare il processo progettato o realizzato. Può però capitare che si svolgano più compiti nello stesso periodo causa carenza di personale. Importante è la flessibilità all'interno del team per diminuire i tempi di scambio mansione.
Ogni membro, qualunque ruolo svolga, ha il diritto di rafforzare la sua funzione con corsi ed il dovere di specializzarsi con ricerca ed esperienze personali.

5.1.1 Project Manager (PM)

Titolare di responsabilità decisionale. Rappresenta il progetto di fronte al fornitore e al committente. Tale figura è garante unico dell'avvio, pianificazione, svolgimento, controllo e chiusura di un progetto. In particolare :

- Pianifica riunioni e compiti;
- Gestisce i ruoli;
- Ruolo preventivo e consuntivo;
- Capacità tecniche e conosce bene gli strumenti utili;
- Controllo, coordinamento con relazioni esterne;
- Approvazione documenti.

5.1.2 Amministratore (AM)

Responsabilità di cura dei processi. In dettaglio:

- Gestione delle risorse e delle infrastrutture;
- Ispettore di versioni e configurazioni;
- Risolutore di difficoltà di processi.

5.1.3 Analista (AN)

Il ruolo principale dell'analista è l'attuazione dei requisiti, quindi comprendere il problema in ogni sfaccettatura, senza l'obbligo di trovare una soluzione. Tra le sue competenze:

- Cogliere il problema;
- Conoscere il dominio del quesito;
- Vasta esperienza professionale.

5.1.4 Progettista (PL)

Il compito del progettista è quello di realizzare e seguire la manutenzione del prodotto. È di fatto un attuatore:

- Risolve il problema;
- Definisce cosa viene costruito e come;
- Ampia conoscenza nei linguaggi e delle risorse utilizzate;
- Esegue una soluzione attuabile e comprensibile.

5.1.5 Programmatore (PR)

Il programmatore è la mano che crea il progetto. Il suo compito principale è infatti la realizzazione e la manutenzione del prodotto. Come il progettista anche nel suo ambito è un attuatore :

- Segue i piani progettati e attua il disegno previsto;
- Implementa test di verifica automatica per il suo codice;
- Il codice che scrive deve essere versionato, documentato e comprensibile.

5.1.6 Verificatore (VR)

Al termine del lavoro eseguito dagli altri membri del team il verificatore esegue un controllo tramite analisi statica del ticket. Questa può essere attuata tramite *Walkthrough* o *Inspection*. Nel secondo caso la check list è concordata insieme al *Project Manager*.

- Controlla siano rispettati gli standard aderiti nel progetto;
- Assicura la conformità del prodotto in ogni fase del ciclo di vita.

5.1.7 Responsabile Qualità

Non è richiesto tale ruolo all'interno del team e non sarà interpretato nella rotazione. Molto spesso è incaricato un ente esterno all'azienda. In questo caso è compito del committente.

5.2 Costi Ruoli

Ciascun ruolo e mansione ha un peso diverso nel progetto e quindi un costo, che deve rientrare nel *budget* pattuito dal committente $9300,00$[1]. Non specificato nel nostro caso.

[1] Il calcolo del budget per noi gruppo composto da 5 persone è stato calcolato proporzionalmente al costo di 7 persone arrotondato per eccesso.

5.2.1 Tabella Costi

Ruolo	Costo
Responsabile/Project Manager	€30,00
Amministratore	€20,00
Analista	€25,00
Progettista	€22,00
Programmatore	€15,00
Verificatore	€15,00

Tabella 2: Costo Ruoli