Norme di Progetto

Gruppo SWEet BIT — ***Progetto SWEDesigner***

Informazioni sul documento

Versione	2.0.0
Redazione	Bodian Malick
	Santimaria Davide
	Pilò Salvatore
Verifica	Santimaria Davide
	Massignan Fabio
Approvazione	Salmistraro Gianmarco
Uso	Interno
Distribuzione	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Gruppo SWEet BIT

Descrizione

Documento riguardante le norme stabilite dal gruppo SWEet BIT per la realizzazione di SWEDesigner

Versioni del documento

Versione	Data	Persone coinvolte	Descrizione
2.0.0	2017/04/26	Salmistraro Gianmarco	Approvazione documento
1.3.0	2017/04/26	Santimaria Davide	Verifica documento
1.3.0	2017/04/25	Massignan Fabio	Verifica documento
1.2.5	2017/04/24	Bodian Malick	Aggiunti strumenti di verifica
1.2.4	2017/04/23	Pilò Salvatore	Stesura definizione metriche
1.2.3	2017/04/22	Santimaria Davide	Aggiunta capitolo codifica
1.2.2	2017/04/22	Santimaria Davide	Stesura appendice lista di controllo
1.2.1	2017/04/21	Santimaria Davide	Ristrutturazione documento secondo *best practice*
1.2.0	2017/03/01	Pilò Salvatore	Approvazione documento
1.1.1	2017/03/01	Salmistraro Gianmarco	Verifica documento
1.1.0	2017/02/28	Bodian Malick	Verifica documento
1.0.2	2017/02/26	Massignan Fabio	Stesura capitoli: Codifica dei file e documentazione, Glossario, Protocollo per lo sviluppo dell'applicazione e Ambiente di lavoro
1.0.1	2017/02/24	Bertolin Sebastiano	Stesura capitoli: Comunicazioni, Riunioni, Documenti e Analisi dei requisiti
1.0.0	2017/02/23	Santimaria Davide	Creazione scheletro del documento e stesura introduzione

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

In questo documento sono definite le norme che i membri del gruppo SWEet BIT adotteranno durante lo sviluppo del progetto *SWEDesigner*. Tutti i membri sono tenuti a leggere e seguire le norme per migliorare l'uniformità del materiale prodotto, migliorare l'efficienza e ridurre il numero di errori. In particolare verranno definite norme riguardanti:

- Interazioni fra membri del gruppo;
- Stesura e convenzioni dei documenti;
- Modalità di lavoro durante le fasi di sviluppo del progetto;
- Ambiente di lavoro.

1.2 Struttura del documento

Il presente documento è diviso in tre sezioni principali:

- La sezione che regola i due procssi primari di:
  - **fornitura**
  - **sviluppo**
- La sezione che regola i processi di supporto:
  - **documentazione**
  - **configurazione**
  - **verifica**
  - **validazione**
- La sezione riguardante i processi organizzativi:
  - **amministrazione**

1.3 Scopo del Prodotto

Lo scopo del progetto è la realizzazone di una *Web App*$_G$ che fornisca all'*Utente*$_G$ un *UML*$_G$ *Designer*$_G$ con il quale riuscire a disegnare correttamente *Diagrammi*$_G$ delle

Classi$_G$ e descrivere il comportamento dei *Metodi*$_G$ interni alle stesse attraverso l'utilizzo di *Diagrammi*$_G$ delle attività. La *Web App*$_G$ permetterà all'*Utente*$_G$ di generare *Codice*$_G$ *Java*$_G$ dall'insieme dei *diagrammi classi*$_G$ e dei rispettivi *metodi*$_G$.

1.4 Glossario

Con lo scopo di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v*2.0.0. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo.

1.5 Riferimenti

1.5.1 Informativi

- **Specifiche *UTF-8*$_G$:**
  http://unicode.org/faq/utf_BOM.html
- **ISO 8601:2004:**
  https://www.iso.org/standard/40874.html
- **Licenza MIT:**
  https://opensource.org/licenses/MIT
- **GitHUB:**
  https://GitHub.com/
- **UML:**
  http://www.UML.org/
- **Atom:**
  https://atom.io/
- **TexLive:**
  https://www.tug.org/texlive/
- ***LaTeX*$_G$:**
  https://www.latex-project.org/
- **Telegram:**
  https://Telegram.org/
- **Piano di progetto:** *Piano di progetto v*2.0.0
- **Piano di qualifica:** *Piano di qualifica v*2.0.0

1.5.2 Normativi

- **Capitolato di appalto SWE*Designer*$_G$ (C6):**
  http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6.pdf

2 Processi primari

2.1 Fornitura

2.1.1 Studio di fattibilità

Alla pubblicazione dei *capitolati*$_G$ è compito del *Responsabile di Progetto* convocare un numero di riunioni tale da consentire al gruppo un confronto su tutti i *capitolati*$_G$ disponibili.
Tali riunioni saranno d'aiuto agli *Analisti* per farsi un'idea delle conoscenze e preferenze di ogni membro del gruppo così da poter redigere uno *Studio di Fattibilità* dei *capitolati*$_G$ disponibili basandosi su:

- **Dominio tecnologico e applicativo:** Conoscenza delle tecnologie richieste, esperienze precedenti con le problematiche poste dal *capitolato*$_G$, conoscenza del *Dominio*$_{GG}$ applicativo;
- **Rapporto Costi/Benefici:** Competitori e prodotti simili già presenti sul mercato, quantità di requisiti obbligatori, costo della realizzazione rapportato al risultato previsto;
- **Individuazione dei rischi:** Comprensione dei punti critici della realizzazione, individuazione di eventuali lacune tecniche o di conoscenza del *Dominio*$_G$ applicativo dei membri del gruppo, analisi delle difficoltà nell'individuazione dei requisiti e loro verificabilità.

Un'ulteriore riunione, a *Studio di Fattibilità* concluso, determinerà la scelta del *capitolato*$_G$.

2.2 Sviluppo

2.2.1 Analisi dei requisiti

La stesura del documento di *Analisi dei Requisiti* è compito degli *Analisti* e si divide nelle fasi di seguito riportante.
Per semplificare il tracciamento degli *Use Case*$_G$ e dei Requisiti si è scelto di utilizzare **Trender**, uno strumento *Open-source*$_G$ che permette di gestire al meglio entrambi gli elementi e le realizioni fra gli stessi.
Il tool permette l'esportazione di questi direttamente in *LaTeX*$_G$ così da semplificare la scrittura degli stessi all'interno della documentazione. Il documento dovrà seguire, inoltre, le norme specificate di seguito.

2.2.1.1 Classificazione dei requisiti È compito degli *Analisti* stilare una lista dei requisiti emersi dal *capitolato*$_G$ e da eventuali riunioni con il *Proponente*$_G$. Questi dovranno essere classificati per tipo e per importanza utilizzando la seguente codifica:

R[importanza][tipo][codice]

- **Importanza** può assumere i seguenti valori:

  0 : Requisito obbligatorio;

  1 : Requisito desiderabile;

  2 : Requisito opzionale.

- **Tipo** può assumere i seguenti valori:

  F : Funzionale;

  Q : Di Qualità;

  P : Prestazionale;

  V : Vincolo.

- **Codice** è il codice univoco di ogni requisito espresso in modo gerarchico.

Ogni requisito è poi esplicato nel seguente modo:

- **Relazioni** di dipendenza con altri requisiti;
- **Descrizione** sintetica del requisito.

2.2.1.2 Modellazione concettuale del sistema e Allocazione Successivamente al riconoscimento e definizione del requisiti emersi dal *capitolato*$_G$ si procede all'analisi dei casi d'uso, denominati anche come *Use case*$_G$ o con l'acronimo UC.
È richiesta agli analisti l'identificazione dei vari casi d'uso, procedendo dal generale al particolare che verranno inseriti nel software di tracciamento ***Trender***$_G$.
Per ogni UC è richiesto l'inserimento, all'interno del software, di:

- **Titolo:** Nome del Caso d'Uso;
- **Descrizione:** Descrizione breve e coincisa dell'UC;
- **Precondizione:** Condizione d'accesso al Caso d'Uso;
- **Postcondizione:** Condizione d'uscita al Caso d'Uso.

Gli altri campi da compilare, da non ritenere obbligatori ma desiderabili, sono:

- **Padre:** Indicare codice univoco del Caso d'Uso padre;
- **Tipo:** Può essere di tre tipi differenti:
  - **Inclusione**;

  - **Estensione**;
  - **Gerarchia**.

- **Scenario:** Descrizione dello scenario rappresentato;
- **Scenario alternativo:** Descrizione scenari alternativi, se presenti;
- **Percorso immagine:** Il percorso dell'immagine rappresentate l'*UML*$_G$;
- **Descrizione immagine:** Descrizione breve e sommaria dell'immagine.

Una volta creato l'UC, selezionarlo dalla lista per:

- Modificare i campi dati inseriti;
- Osservare i figli dell'UC corrente;
- Associare un **Attore** selezionandolo dal menù a tendina;
- Associare un **Requisito**.

Il caso d'uso dovrà essere accompagnato da un grafico riassuntivo in *UML*$_G$2.x, titolato come il caso d'uso in questione.
È compito del software di tracciamento tracciare gli UC con un *Codice*$_G$ univoco e gerarchico nella forma:

UC[codice univoco del padre].[codice univoco del figlio]

Il software provvederà a generare anche i *File*$_G$ *LaTeX*$_G$ corrispondenti.

2.2.2 Progettazione

2.2.2.1 Attività progettuali Il lavoro dei progettisti consiste nel progettare struttura e comportamento del software in modo da implementare tutti i requisiti obbligatori individuati dagli analisti; inoltre, i progettisti dovranno prevedere anche il massimo della copertura dei requisiti desiderabili. Nello specifico, il lavoro dei progettisti si articola in due attività:

- **Progettazione architetturale** (descritta in §5.3.5 di ISO/IEC 12207:1995): definire l'architettura del sistema, cioè individuare e descrivere le componenti del sistema, esplicitandone ruoli e relazioni tra essi; ogni progettista documenta i prodotti di questa attività nel documento di *Specifica Tecnica*.
- **Progettazione di dettaglio** (descritta in §5.3.6 di ISO/IEC 12207:1995): definire, in dettaglio, il comportamento delle componenti del sistema; ogni progettista documenta i prodotti di questa attività nel documento di *Definizione di Prodotto*.

Per ognuna di queste due attività, ogni progettista deve definire dei test che permettano la verifica del sistema:

- durante la progettazione architetturale, i progettisti definiscono i test di integrazione, cioè dei test che verificano il corretto interfacciarsi delle componenti del sistema;
- durante la progettazione di dettaglio, la specifica del comportamento di una componente dev'essere accompagnata dalla definizione di un test di unità che verifichi il corretto funzionamento della componente isolata.

2.2.2.2 Diagrammi Per descrivere l'architettura e il comportamento del sistema i progettisti utilizzano i diagrammi *UML*$_G$ descritti qui di seguito, seguendo lo standard *UML*$_G$ 2.0:

- **Diagramma dei package:** raggruppa un numero di elementi *UML*$_G$ in una sola unità di livello più alto.
- **Diagramma delle classi:** descrive le interfacce delle componenti del sistema e le relazioni tra esse.
- **Diagramma delle attività:** descrive i passi di una procedura.
- **Diagramma di sequenza:** descrive uno scenario, dove le azioni sono disposte in sequenza e le varie scelte sono già state prese.

Per disegnare i diagrammi appena esposti, i progettisti devono usare il software gratuito Astah ed esportare i diagrammi in formato *PNG*$_G$. Astah supporta la generazione di tutti i diagrammi che vengono usati dal gruppo (classi, attività, sequenza e package). Le alternative analizzate sono state *Dia*, *LuchiChart* e *Papyrus* che si sono rivelati però troppo deboli in confronto ad **Astah** che offre diverse funzionalità aggiuntive raggruppandone diverse dei tre software citati al suo interno.

2.2.2.3 Stile di progettazione L'architettura di SWEDesigner segue lo stile dell'orientazione agli oggetti; perciò, le componenti software descritte nei diagrammi delle classi sono:

- prototipi di oggetti *JavaScript*$_G$;

Al fine di aumentare la manutenibilità i progettisti devono seguire le seguenti regole:

- La progettazione dovrà usare quanto più possibile *design pattern*$_G$;
- L'architettura deve rispettare le regole di progettazione che ne garantiscono la correttezza rispetto allo standard *UML*$_G$;
- Suddividere il progetto in *moduli*$_G$, in accordo con lo stile di progettazione dell'ambiente *Node.js*$_G$;
- ogni componente deve avere un compito ben precisio, ricavabile dal proprio nome e dai metodi che costituiscono l'interfaccia;

2.2.2.4 Nomenclatura I progettisti devono osservare le seguenti norme, al fine di uniformare il codice che verrà derivato dalla progettazione:

- le componenti vanno raggruppate in *moduli*$_G$. Il nome di tale *modulo*$_G$ deve:
  - contenere solo caratteri alfanumerici;
  - essere in lingua inglese;
  - avere il nome del *modulo*$_G$ "padre" che lo contiene preposto al nome del metodo, seguito da una coppia di due punti (::);
  - essere rappresentativo del ruolo che rappresenta.
- ogni nome di componente di un *modulo*$_G$ deve:
  - contenere solo caratteri alfanumerici;
  - essere in lingua inglese;
  - se composto da più termini, l'iniziale di ogni termine deve essere maiuscola;
  - non può essere un termine riservato del linguaggio *JavaScript*$_G$.
- ogni nome di funzione deve:
  - contenere solo caratteri alfanumerici;
  - essere in lingua inglese;
  - se composto da più termini, l'iniziale di ogni termine deve essere maiuscola;
  - essere rappresentativo per il servizio che offre;
  - non può essere un termine riservato del linguaggio *JavaScript*$_G$.

2.2.2.5 Specifica Tecnica I Progettisti devono descrivere la progettazione ad alto livello dell'architettura dell'applicazione e dei singoli componenti nella "Specifica Tecnica 1.0.0". Devono inoltre provvedere alla progettazione di opportuni test di integrazione.

- *Diagrammi UML*$_G$
  Devono essere realizzati i seguenti diagrammi:
  - diagrammi delle classi;
  - diagrammi dei *package*$_G$;
  - diagrammi di attività;
  - diagrammi di sequenza.

- *Design pattern*$_G$
  I Progettisti devono descrivere i design pattern utilizzati per realizzare l'architettura. Di tali design pattern, si deve includere una breve descrizione e un diagramma che ne esemplifichi il funzionamento e la struttura.
- *Tracciamento componenti*
  Ogni requisito deve essere tracciato al componente che lo soddisfa. L'applicazione web *Trender*$_G$ genera automaticamente le tabelle di tracciamento. In questo modo sarà possibile garantire che ogni requisito venga soddisfatto e, al tempo stesso, misurare il progresso nell'attività di progettazione.
- *Test di integrazione*
  I Progettisti devono definire delle classi di verifica. Tali classi sono necessarie per verificare che i componenti del sistema funzionino nella maniera prevista.

2.2.2.6 Definizione di Prodotto I Progettisti devono produrre la "Definizione di Prodotto". In essa viene descritta la progettazione di dettaglio del sistema, ampliando quanto scritto nella "Specifica Tecnica 1.0.0".

- *Diagrammi UML*
  Devono essere redatti i seguenti diagrammi:
  - diagrammi delle classi;
  - diagrammi di attività;
  - diagrammi di sequenza.
- *Definizione di classe*
  Ogni classe progettata deve essere descritta all'interno della "Definizione di Prodotto". Tale descrizione deve comprendere una spiegazione sullo scopo della classe e deve specificare quale funzionalità essa modella. Nella descrizione devono inoltre essere presenti l'elenco di metodi e attributi della classe.
- *Tracciamento classi*
  Ogni requisito deve essere tracciato alle classi che lo soddisfano. Il software Trender genera in automatico le tabelle di tracciamento. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni classe soddisfi almeno un requisito.
- *Test di unità*
  I Progettisti devono definire i test di unità necessari per verificare che i componenti del sistema funzionino nel modo previsto.

2.2.3 Codifica dei file e documentazione

2.2.3.1 Codifica e convenzioni Di seguito vengono elencate le convenzioni stabilite; è contemplata la modifica delle stesse, previa autorizzazione del *Project Manager*;

- Tutti i file contenenti codice o documentazione dovranno essere in codifica *UTF-8*$_G$ senza *BOM*$_G$;
- È ammessa la possibilità di effettuare modifiche alle convenzioni stabilite in seguito ad una decisione del *Responsabile di Progetto*;
- L'unica lingua ammessa per i nomi di variabili, classi e funzioni è l'inglese;
- I commenti devono essere fatti in italiano;
- I nomi delle classi devono avere l'iniziale maiuscola;
- I nomi di variabili e metodi devono avere la prima lettera minuscola, mentre devono essere maiuscole le iniziali delle eventuali altre parole che compongono il nome stesso;
- Identazione effettuata tramite tabs;
- I comandi if / else / for / while / try vengono sempre seguiti le parentesi graffe e bisogna sempre andare a capo dopo la loro apertura;
- Nessun spazio di riempimento in costruzioni vuote (ad esempio, , [], fn ());
- I nomi delle variabili e dei metodi devono essere il più descrittivi possibili, ad eccezioni degli iteratori;
- I nomi di metodi e funzioni devono essere adottati quelli definiti nel documento *Specifica Tecnica*;
- I programmatori possono introdurre funzioni ausiliarie non presenti nella progettazione.

2.2.3.2 Documentazione del codice I file contenti codice dovranno essere provvisti di un'intestazione contenente:

```
/*!
* \file Nome del file
* \author Autore (indirizzo e-mail dell'autore)
* \date Data di creazione
* \brief Breve descrizione del file
*
* Descrizione dettagliata del file
*/
```

- Nome: è il nome del file, estensione compresa;
- Autore: è l'indirizzo e-mail dell'autore;
- Data di creazione: è la data di creazione del file;
- Breve descrizione del file: qui viene inserita una descrizione sintetica del file;
- Descrizione dettagliata del file: qui viene inserita una descrizione più approfondita di quella precedente, illustrando tutti i dettagli che compongono il file.

Prima di ogni classe dovrà esserci un commento contenente:

```
/*!
* \class Nome della classe
* \brief Breve descrizione della classe
*/
```

Prima di ogni metodo dovrà essere inserito un commento contenente:

```
/*!
* \brief Breve descrizione della funzione
* \param Nome del primo parametro
* \param Nome del secondo parametro
* \return Valore ritornato dalla funzione
*/
```

2.2.4 Strumenti e ambienti di sviluppo

2.2.4.1 Stesura del codice Per la stesura del codice è consigliabile usare uno dei seguenti editor:

- gedit (http://projects.gnome.org/gedit);
- *Microsoft Visual Studio*$_G$ (https://www.visualstudio.com/it/).

Microsoft Visual Studio è un *IDE*$_G$ per lo sviluppo di applicazioni per tablet, smartphone e computer, oltre a siti e servizi web. E' fortemente consigliato per lo sviluppo di questo progetto dato che supporta numerosi linguaggi di programmazione tra cui *Node.js*$_G$, *HTML*$_G$, *Java*$_G$ e *Javascript*$_G$, tutti linguaggi fondamentali per la realizzazione del capitolato richiesto.

2.2.4.2 *Framework*$_G$ Per lo sviluppo del progetto è previsto l'utilizzo dello stack *MEAN*$_G$, contenente le seguenti tecnologie:

- *Mongodb*$_G$;
- *express*$_G$;
- *Angular js*$_G$;
- *Node js*$_G$.

3 Processi di supporto

Questo capitolo descrive tutte le convenzioni scelte ed adottate da SWEet BIT riguardo alla stesura, verifica e approvazione della documentazione da produrre.

3.1 Documentazione di progetto

La documentazione del progetto è contenuta in un *repository* distinto da quello del codice sorgente. Tale *repository* contiene i seguenti documenti:

- **Studio di Fattibilità:** Documento interno che valuta i pro e contro di ogni capitolato d'appalto;
- **Norme di Progetto:** Documento interno che descrive le norme utilizzate da SWEet BIT per la gestione dei processi;
- **Verbali:** Documento interno che riporta i verbali delle riunioni che interessano il gruppo;
- **Piano di Progetto:** Documento esterno che dichiara come il gruppo intende gestire le risorse umane e temporali;
- **Piano di Qualifica:** Documento esterno che descrive come il gruppo gestisce la qualità del prodotto;
- **Analisi dei Requisiti:** Documento esterno che elenca, descrive e traccia i requisiti ed i casi d'uso del prodotto;
- **Specifica Tecnica:** Documento esterno che descrive l'architettura logica del sistema, mostrandone l'interfaccia di ogni componente utilizzato;
- **Glossario:** Documento esterno che riporta la definizione di termini che possono portare ad ambiguità;

3.1.1 Ambiente di lavoro

3.1.1.1 Stesura documenti

3.1.1.1.1 LaTeX

Per la stesura dei documenti si è scelto di utilizzare il sistema *LaTeX*$_G$ poiché permette una netta separazione fra contenuti e formattazione; con *LaTeX*$_G$ è possibile definire i *Template*$_G$ di layout in *File*$_G$ condivisi da ogni documento rendendo la lavorazione degli stessi altamente più flessibile e ottimale.

$LaTeX_G$ consente poi l'utilizzo di funzioni e variabili locali definiti dall'*Utente*$_G$, in maniera tale da semplificare ulteriormente il lavoro di stesura dei documenti, una volta definita la struttura degli stessi.
Per la scrittura dei documenti $LaTeX_G$ l'editor utilizzato è **TexMaker**.

3.1.1.1.2 Strumentazione esterna

Per ridurre al minimo gli errori di calcolo di alcuni indici o di formattazione del testo si è scelto di utilizzare alcuni strumenti automatici che si occupano di svolgere alcuni semplici compiti di calcolo e formattazione:

- **Script *Perl*$_G$ per il calcolo dell'indice di Gulpease:** All'interno della directory `tools\gulpease` dentro la *Repository*$_G$ è presente uno *Script*$_G$ in *Perl*$_G$ per il calcolo automatico dell'indice di Gulpease che segue la seguente formula:

$$GulpeaseIndex = 89 + \frac{300 * (numerodellefrasi) - 10 * (numerodellelettere)}{numerodelleparole} \quad (1)$$

- **Script *Perl*$_G$ per la glossarizzazione dei termini:** Per ridurre al minimo gli errori nell'inserimento di un termine all'interno del *Glossario* si è deciso di utilizzare uno *Script*$_G$ in *Perl*$_G$, reperibile nella directory `tools\glossary`, che si occupa di inserire il comando $LaTeX_G$ \glossaryItem su tutti i termini presenti all'interno del *Glossario* dati in input un *Glossario* e un documento;
- **Aspell:** si tratta di un tool offerto dall'editor stesso o presente in forma *Standalone*$_G$ che effettua un controllo ortografico su tutto il documento.

3.1.2 Template

Per agevolare la redazione di un documento è stato prodotto un *Template*$_G$ e delle regole da seguire per la stesura degli stessi.
Tale modello e tali regole sono inseriti all'interno di una cartella `documents\template` sulla *repository*$_G$.

3.1.3 Formattazione generale delle pagine

L'intestazione di ogni pagina contiene:

- Logo del gruppo;
- Nome del gruppo;
- Nome del progetto;

- Sezione corrente del documento;

A piè di pagina invece è presente:

- Nome e versione del documento;
- Pagina corrente nel formato N di T dove N è il numero di pagina corrente e T è il numero di pagine totali.

3.1.4 Struttura del documento

3.1.4.1 Prima pagina Ogni documento è caratterizzato da una prima pagina che contiene le seguenti informazioni sul documento:

- Nome del gruppo;
- Nome del progetto;
- Logo del gruppo;
- Titolo del documento;
- Versione del documento;
- Cognome e nome dei redattori del documento;
- Cognome e nome dei verificatori del documento;
- Cognome e nome del responsabile approvatore del documento;
- Destinazione d'uso del documento;
- Lista di distribuzione del documento;
- Breve descrizione del documento.

3.1.4.2 Diario delle modifiche La seconda pagina di ogni documento contiene il diario delle modifiche.
Ogni riga del diario delle modifiche contiene:

- Un breve sommario delle modifiche svolte;
- Cognome e nome dell'autore;
- Data della modifica;
- Versione del documento dopo la modifica.

La tabella è ordinata per data in ordine decrescente, in modo che la prima riga corrisponda alla versione attuale del documento.

3.1.4.3 Indici In ogni documento è presente un indice delle sezioni, un indice delle figure e un indice delle tabelle. Nel caso non siano presenti figure o tabelle i rispettivi indici verranno omessi.

3.1.5 Ciclo di vita

Ogni documento prodotto segue un preciso iter che scandisce le fasi in cui si trova in ogni istante. Un documento può trovarsi in tre stati diversi:

- **In lavorazione:** Un documento entra in questa fase nel momento della sua creazione e vi rimane per tutto il periodo della sua stesura o per eventuali successive modifiche;
- **Da verificare:** Un documento entra in questa fase alla fine della sua stesura quando entra in possesso dei verificatori che avranno il compito di individuare e correggere eventuali errori sintattici o semantici;
- **Approvato:** Un documento entra in questa fase una volta che il *Responsabile di Progetto* lo ha approvato dopo la fase di verifica.
  L'approvazione sancisce la fine del ciclo di vita del documento per la data versione.

Ogni fase del ciclo di vita può essere affrontata anche più volte da parte di un documento.

3.1.6 Glossario

Il Glossario conterrà tutte le parole presenti negli altri documenti che fanno parte del contesto dell'applicazione o che possono essere fraintese. Le definizioni, presentate in ordine alfabetico, dovranno essere concise e comprensibili.
I termini verranno inseriti nel glossario parallelamente al processo di stesura degli altri documenti, in modo da limitare l'errore umano.
È preferibile inserire un termine inizialmente privo di definizione, piuttosto che rimandare la stesura del glossario.

3.1.7 Norme tipografiche

Questa sezione racchiude le convenzioni riguardanti tipografia, ortografia e uno stile uniforme per tutti i documenti.

3.1.7.1 Punteggiatura

- **Parentesi:** Il testo racchiuso tra parentesi non deve aprirsi o chiudersi con un carattere di spaziatura e non deve terminare con un carattere di punteggiatura;

- **Punteggiatura:** Un carattere di punteggiatura non deve mai esser preceduto da un carattere di spaziatura;
- **Lettere maiuscole:** Le lettere maiuscole vanno poste solo dopo il punto, il punto di domanda, il punto esclamativo e all'inizio di ogni elemento di un elenco puntato ed oltre a dove sia previsto dalla lingua italiana. È inoltre utilizzata l'iniziale maiuscola nel nome del team, del progetto, dei documenti, dei ruoli di progetto, delle fasi di lavoro e nelle parole *Proponente*$_G$ e *Committente*$_G$.

3.1.7.2 Stile di testo

- **Corsivo:** Il corsivo deve essere utilizzato nei seguenti casi:
  - **Citazioni:** Quando si deve citare una frase questa sarà scritta in corsivo;
  - **Nomi particolari:** Il corsviso deve essere utilizzato quando ci si rierisce a figure particolari (es. *Analista*);
  - **Documenti:** Il corsivo deve essere utilizzato quando ci si riferisce a documenti particolari (es. *Glossario*);
  - **Altri casi:** Il corsivo sarà utilizzato in tutte quelle situazioni in cui è necessario dare rilievo ad una parola o passaggio significativo;
- **Grassetto:** Il grassetto deve essere utilizzato nei seguenti casi:
  - **Elenchi puntati:** In questo caso il grassetto può essere utilizzato per mettere in evidenza i punti sviluppati nella loro continuazione;
  - **Altri casi:** Il grasstto dovrà essere sempre utilizzato per evidenziare passaggi o parole chiave;
- **\path:** Il comando \path deve essere utilizzato per indicare i percorsi all'interno di directory;
- **Maiuscolo:** L'utilizzo di parole completamente in maiuscolo è riservato solo ed esplusivamente alle sigle o alle macro LaTeX$_G$ riportate nei documenti;
- **LaTeX$_G$:** Ogni riferimento a LaTeX$_G$ deve essere scritto utilizzano la macro \LaTeX$_G$;

3.1.7.3 Composizione del testo

- **Elenchi puntati:** Ogni punto dell'elenco puntato deve essere scritto in grassetto e con la prima lettera in maiuscolo.
  Nella definizione del punto la prima lettera dovrà essere maiuscola ad eccezione di casi isolati (es. nome di file) e dovrà terminare sempre con un ";" ; mentre l'ultimo elemento terminerà con un "." ;

- **Note a piè di pagina:** Ogni nota dovrà cominciare con l'iniziale della prima parola maiuscola e non deve essere preceduta da alcun carattere di spaziatura. Ogni nota deve terminare con un punto.

3.1.7.4 Formati

- **Percorsi:** Per tutti gli indirizzi e-mail e web completi dovrà essere utilizzato il comando *LaTeX*$_G$ \url mentre per i percorsi relativi si utilizzerà il comando \path;
- **Date:** Tutte le date presenti all'interno della documentazione devono seguire la notazione definiti nello standard *ISO*$_G$ 8601:2004:

  AAAA-MM-GG

  dove:

  - AAAA: rappresenta l'anno utilizzando quattro cifre;
  - MM: rappresenta il mese utilizzando due cifre;
  - GG: rappresenta il giorno utilizzando due cifre.
- **Nomi propri:** L'utilizzo dei nomi propri dei membri del team (e non) deve seguire la notazione "Cognome Nome";
- **Nome gruppo:** Ci si riferirà al gruppo solo come "SWEet BIT";
- **Nome del *Proponente*$_G$:** Ci si riferirà al *Proponente*$_G$ come "Zucchetti s.r.l" o semplicemente come "*Proponente*$_G$";
- **Nome del *Committente*$_G$:** Ci si riferià al *Committente*$_G$ come "prof. Vardanega Tullio" o semplicemente come "*Committente*$_G$";
- **Nome del progetto:** Ci si riferirà al progetto solo come "SWEDesigner".

3.1.7.5 Sigle Le sigle dei documenti potranno essere utilizzate solo ed esclusivamente all'interno di tabelle o diagrammi. Sono previste le seguenti sigle:

- **AdR** = Analisi dei Requisiti;
- **GL** = Glossario;
- **NdP** = Norme di Progetto;
- **PdP** = Piano di Progetto;
- **PdQ** = Piano di Qualifica;
- **SdF** = Studio di Fattibilità;
- **ST** = Specifica Tecnica;

- **RA** = Revisione d'Accettazione;
- **RP** = Revisione di Progettazione;
- **RQ** = Revisione di Qualifica;
- **RR** = Revisione dei Requisiti.

3.1.7.6 Componenti grafiche

3.1.7.6.1 Tabelle

Ogni tabella presente all'interno dei documenti dev'essere accompagnata da una didascalia, in cui deve comparire un numero identificativo incrementale per la tracciabilità della stessa all'interno del documento.

3.1.7.6.2 Immagini

Le immagini da includere all'interno del documento devono avere preferibilmente il formato Portable Network Graphics (PNG_G).

3.1.8 Classificazione dei documenti

3.1.8.1 Documenti formali Un documento viene definito formale quando viene approvato dal *Responsabile di Progetto* ed è quindi pronto per essere inviato ai richiedenti. Per raggiungere questo stato il documento deve seguire l'iter descritto nel *Norme di Progetto* e nel paragrafo 3.1.5 riguardante il ciclo di vita dei documenti.

3.1.8.2 Documenti informali Un documento è definito informale fino a quando non approvato dal *Responsabile di Progetto*, fino ad allora il suo uso è da considerarsi unicamente interno.

3.1.8.3 Versionamento La documentazione prodotta deve essere corredata dal numero di versione attuale utilizzando la codifica:

$$v.X.Y.Z$$

dove:

- X: indica il numero crescente di uscite formali del documento;
- Y: indica il numero crescente di modifiche sotanziali al documento;
- Z: indica il numero crescente di modifiche minori apportate al documento;

3.2 Configurazione

3.2.1 Versionamento

Dopo aver preso in considerazione diverse opzioni per il versionamento, alla fine si è scelto di utilizzare ***GitHub***$_G$ per via della sua enrome flessibilità e per via delle esperienze pregresse di tutti i membri del gruppo che hanno manifestato una certa familiarità con tale strumento.
È stata creata una sola *Repository*$_G$, alla quale si aggiungeranno le altre legate alle fasi successive, contenente tutte le cartelle necessarie alla stesura dei documenti *LaTeX*$_G$.
Una volta terminato il lavoro di redazione dei documenti sarà creato un *branch* di verifica per permettere ai *Verificatori* di lavorare in parallelo agli altri membri del gruppo.

3.2.2 Controllo della configurazione

I cambiamenti nella configurazione del software e della documentazione vanno controllati e tracciati, possono nascere dal *Piano di Progetto* o dall'iniziativa di un membro.

3.2.2.1 Cambiamenti pianificati Nascono dal *Piano di Progetto*, il cambiamento è già controllato, e quindi basta registrarlo aggiornando la versione dell'elemento che è stato cambiato.

3.2.2.2 Cambiamenti non pianificati Nascono dall'iniziativa di un membro, va proposto agli altri membri del gruppo tramite *Telegram*, nel caso la maggioranza sia d'accordo, il responsabile di progetto incarica un amministratore di implementarlo e motivarlo nella documentazione.

3.3 Verifica

Vengono qui elencati, e sommariamente descritti, gli strumenti automatizzati e le politiche adottate per effettuare la verifica dei documenti redatti e del *Codice*$_G$ prodotto.

3.3.1 Verifica dei documenti

Nel verificare i documenti, i verificatori dovranno seguire le seguenti direttive:

- Controllare il diario delle modifiche per vedere cos'è cambiato nel documento dall'ultima verifica;
- Rileggere le parti modificate del documento, concentrandosi maggiormente nella ricerca degli errori più frequenti, riportati in appendice nella lista di controllo;

- Nel caso di un errore o un'incongruenza nel contenuto, il verificatore dovrà segnalare all'autore l'errore riscontrato, che provvederà a correggerlo;
- Infine, il verificatore aggiornerà il diario delle modifiche registrando la verifica e aggiornerà la versione del documento;
- Nel caso si siano riscontrati frequentemente errori dello stesso tipo, tali errori dovranno essere inseriti nella lista di controllo.

3.3.2 Verifica dei requisiti

I verificatori, nel controllare i requisiti, dovranno prestare attenzione a questi elementi:

- Correttezza ortografica e lessicale del testo;
- Correttezza del codice identificativo prodotto da *Trender*$_G$;
- Atomicità dei requisiti foglia.

3.3.3 Verifica dei diagrammi UML

I verificatori, nel controllare i diagrammi UML, dovranno prestare attenzione a questi elementi:

- Correttezza ortografica e lessicale del testo;
- Correttezza del formalismo grafico utilizzato;
- Correttezza logica del diagramma.

3.3.4 Analisi Statica

L'analisi statica è il processo di verifica del sistema o di un suo componente, senza che esso debba necessariamente poter essere eseguito. Questa tecnica permette di verificare sia la documentazione che il *Codice*$_G$, individuandone errori ed anomalie. Questa tecnica di analisi può essere svolta in due modi distinti e complementari.

3.3.4.1 Walkthrough Si svolge effettuando una lettura critica e a largo spettro del documento. È una tecnica utilizzata soprattutto nelle prime attività del progetto, quando ancora non è presente un'esperienza tale da permettere una verifica più mirata e precisa. Il verificatore genererà, infatti, una lista di controllo con gli errori più frequenti in modo da favorire il miglioramento della verifica nei periodi successivi. Il walkthrough è un'attività onerosa e collaborativa che richiede l'intervento di più persone per essere efficace. Questa tecnica si svolge in tre fasi principali:

- **Fase uno:** lettura dei documenti ed individuazione degli errori;
- **Fase due:** discussione dei errori riscontrati e delle correzioni da applicare;
- **Fase tre:** correzione degli errori rilevati, e stesura di un rapporto che ne elenchi le modifiche effettuate

La lista di controllo risultante, contenente le tipologie di errori più frequenti è riportata in appendice.

3.3.4.2 Inspection Consiste nell'analisi mirata di alcune parti dei documenti o del codice ritenute fonti maggiori di errore. Deve essere seguita una lista di controllo per svolgere efficacemente questa attività; tale lista deve essere redatta anticipatamente ed è sostanzialmente frutto dell'esperienza maturata dai membri del team con tecniche di walkthrough. L'inspection è dunque più rapida del walkthrough, in quanto il documento viene analizzato solo in alcune sue parti e con una lista di controllo ben precisa. In questa attività sono coinvolte solo i verificatori che, dopo aver individuato gli errori, procedono alla loro correzione e alla redazione di un rapporto che tenga traccia del lavoro svolto.

Per aiutare l'identificazione automatica di errori, vengono usati i seguenti strumenti:

- **JSHint:** uno strumento che aiuta ad identificare gli errori e i potenziali problemi nel codice *JavaScript*$_G$;
- **JSLint:** uno strumento usato per verificare che il codice *JavaScript*$_G$ compila secondo le regole del linguaggio;
- **Closure Compiler:** strumento utilizzato per migliorare il codice, controllando la sintassi ed i riferimenti alle variabili.

3.3.5 Analisi Dinamica

L'analisi dinamica è il processo che verifica il prodotto software mentre esso è in esecuzione. Viene effettuata tramite test che devono essere coerentemente pianificati in modo da evitare spreco di risorse. L'obiettivo dei test sul software è la realizzazione di un prodotto il più possibile esente da errori. Il principale ostacolo alla fase di test è sintetizzato nella tesi di Dijkstra, che afferma che il test può indicare la presenza di errori, ma non ne può garantire l'assenza. Affinché tale attività sia utile e generi risultati attendibili è necessario che i test effettuati siano ripetibili: dato un certo input deve essere prodotto sempre uno stesso output in uno specifico ambiente. Di conseguenza, i tre elementi fondamentali di un test sono:

- **Ambiente:** sistema hardware e software sui quali è stato pianificato l'utilizzo del prodotto software sviluppato. Su di essi deve essere specificato uno stato iniziale dal quale poter eseguire il test;

- **Specifica:** definizione di quali input sono necessari per l'esecuzione del test e quali output sono attesi;
- **Procedure:** definizione di come devono essere svolti i test, in che ordine devono essere eseguiti e come devono essere analizzati i risultati.

Organizzazione delle attività di test:

- **Test di unità**: l'obiettivo di questo test è verificare che ogni unità del $Codice_G$ si comporti esattamente come previsto. Un unità di $Codice_G$ è la più piccola parte di software che è utile verificare singolarmente e che viene prodotta da un singolo programmatore. Attraverso tali test si vuole verificare il corretto funzionamento dei moduli che compongono l'intero sistema, in modo da esaminare possibili errori di implementazione da parte dei programmatori. Vengono identificati tramite la seguente notazione:

  TI[CodiceTest]

- **Test di integrazione**: l'obiettivo di questo test è verificare che le componenti del sistema vengano aggiunte incrementalmente al prodotto e analizzare che la combinazione di due o più unità software funzioni come previsto. Una componente è l'aggregazione di più unità software. Questo tipo di test serve ad individuare errori residui nella realizzazione dei singoli moduli, modifiche delle interfacce e comportamenti inaspettati di componenti software preesistenti forniti da terze parti che non si conoscono a fondo. Per effettuare tali test devono essere aggiunte delle componenti fittizie al posto di quelle che non sono ancora state sviluppate, in modo da non influenzare negativamente l'esito dell'analisi. Vengono identificati tramite la seguente notazione:

  TI[CodiceTest]

- **Test di sistema**: l'obiettivo di questo test è verificare il comportamento del prodotto software finale. Lo scopo è quello di capire se esso rispetta i requisiti individuati nella fase di Analisi. Vengono identificati tramite la seguente notazione:

  TS[CodiceProgressivoRequisito]

- **Test di regressione**: l'obiettivo di questo test è stabilire se le modifiche apportate al software hanno compromesso componenti software precedentemente funzionanti. Questi test vengono eseguiti ogni volta che viene apportata una modifica al software. Tale operazione è aiutata dal tracciamento, che permette di individuare e ripetere facilmente i test di unità, integrazione ed eventualmente sistema che sono stati potenzialmente influenzati dalla modifica. Non vengono attualmente implemantati.

- **Test di accettazione**: l'obiettivo di questo test è la realizzazione del collaudo del prodotto software eseguito in presenza del proponente. Se l'esito risulta positivo,

si può procedere al rilascio ufficiale del prodotto. Vengono identificati tramite la seguente notazione:

TA[CodiceTest]

Per verificare tramite test se il prodotto software non abbia errori, viene utilizzato:

- **Mocha:** una libreria ricca di funzionalità per l'esecuzione di test *JavaScript*$_G$.

3.3.6 Metriche

Durante l'attività di verifica dei processi vengono definite delle metriche, di seguito riportate, utili alla misurazione quantitativa degli obiettivi di qualità prefissati dal documento *Piano di Qualifica v*2.0.0. Per ognuna delle metriche viene definito il nome ed il range o il valore di accettazione stabilito. In appendice viene specificato il significato e come calcolare tali metriche.

Metrica	**Range**
Schedule Variance	[≥ -141]
Budget Variance	[≥ -282.5]
Indice Gulpease	[40-100]
Complessità Ciclomatica	[0-15]
Numero metodi per file	[3-10]
Variabili non utilizzate e/o non definite	[0-0]
Numero argomenti per funzione	[0-6]
Linee di codice per linee di commento	[≥0.25]
Copertura del codice	[70%-100%]

Tabella 2: Metriche - Range accettazione

4 Processi Organizzativi

I processi normatizzati in questa sezione, servono all'organizzazione che li istanzia a progetto, quindi servono al gruppo stesso.

4.1 Gestione di progetto

4.1.1 Ruoli

Verranno assegnati dei ruoli corrispondendi a quelli professionali per lo sviluppo del progetto. Ogni membro del gruppo ricoprirà tutti i ruoli almeno una volta durante ogni periodo di sviluppo.
Ogni attività di un processo, verrà svolta solo ed esclusivamente dal membro del gruppo che ricopre il ruolo di tale attività, rispettando la pianificazione specificata all'interno del *Piano di Progetto*2.0.0.

4.1.1.1 Responsabile di Progetto Il *Responsabile di Progetto* detiene la resposabilità di tutto il team e il potere decisionale su ogni attività. Si occuperà inoltre delle comunicazioni esterne ed avrà il pieno accesso alla mail del gruppo. In particolare è di responsabilità del *Responsabile di Progetto*:

- Coordinamento, pianificazione e controllo delle attività;
- Approvazione dei documenti;
- Comunicazioni esterne;
- Assegnazione compiti ai vari individui;
- Convocazione riunioni interne/esterne.

Nello specifico i suoi compiti sono:

- Redazione *Piano di Progetto*;
- Collaborazione alla stesura del *Piano di Qualifica*;
- Assicurarsi che le attività svolte siano conformi alle *Norme di Progetto*;
- Garantire il rispetto dei ruoli;
- Assegnare e gestire task agli altri membri del gruppo;
- Approvare definitivamente i documenti.

4.1.1.2 Amministratore L'*Amministratore* è responsabile dell'ambiente di lavoro. È sua responsabilità:

- Gestione ambiente di lavoro attrezzando il team di strumenti necessari;
- Aggiungere all'ambiente di lavoro strumenti di automazione del lavoro;
- Gestione del versionamento della documentazione;
- Controllo delle versioni del prodotto;
- Risoluzione dei problemi in merito alla gestione di risorse e processi.

L'Amministratore, inoltre, redige le *Norme di Progetto*, collabora alla stesura del *Piano di Progetto* e al `Piano di Qualifica`.

4.1.1.3 Analista L'*Analista* è il responsabile di tutte le analisi del problema. È sua responsabilità:

- Studiare a fondo la natura del prodotto;
- Classificare i requisiti;
- Redigere diagramma dei Casi d'Uso;
- Produrre una specifica di progetto precisa in ogni sui punto e comprensibile dal proponente, dal committente e dai progettisti;
- Redigere lo *Studio di Fattibilità* e l'*Analisi dei Requisiti*.

4.1.1.4 Progettista Il *Progettista* è il responsabile delle attività di progettazione. È sua responsabilità:

- Effettuare scelte progettuali volte ad applicare al prodotto soluzioni ottimali;
- Effettuare scelte procedurali che ne garantiscano la manutenibilità;
- Produrre una soluzione soddisfacente per il Committente.

Il Progettista, inoltre, redige la *Specifica Tecnica*, la *Definizione di Prodotto* e collabora alla stesura del *Piano di Qualifica*.

4.1.1.5 Programmatore Il *Programmatore* è responsabile delle attività di codifica e delle commenti ausiliarie necessari per il processo di verifica. È sua responsabilità:

- Implementare le soluzioni descritte dal *Progettista*;
- Scrivere codice documentato che rispetti le metriche stabilite;
- Implementare i test da eseguire sul codice scritto durante i processi di verifica.

Il Programmatore ha, inoltre, il compito di redigere il *Manuale Utente*.

4.1.1.6 Verificatore Il *verificatore* è responsabile delle attività di verifica. È sua responsabilità:

- Controllare la conformità del prodotto ad ogni stadio del ciclo di vita;
- Garantire che le attività seguano le norme stabilite.

Il *Verificatore*, inoltre, collabora alla stesura del *Piano di Qualifica*.

4.1.2 Coordinamento

Il coordinamento del gruppo avviene tramite:

- *Repository*$_G$ su ***GitHub***$_G$;
- *Google Drive*$_G$;
- *Telegram*$_G$
- *Hangout*$_G$

4.1.2.1 Repository Sulla *Repository*$_G$ di ***GitHub***$_G$, raggiungibile all'indirizzo `https://GitHub.com/SWEetBIT`, sono caricati i vari *Template*$_G$ da utilizzare durante la stesura dei documenti e la strumentazione da utilizzare per la formattazione dei termini del *Glossario*.

4.1.2.2 Google Drive Lo strumento di *Cloud storage*$_G$ di *Google*$_G$, è stato utilizzato principalmente per tenere traccia di verbali interni e di documentazione interna non formale, che può essere utile a tutti i membri del gruppo, durante le varie fasi di lavorazione del progetto.

4.1.2.3 Telegram Il servizio di messaggistica istantanea *Telegram*$_G$, è stato utilizzato per il coordinamento delle riunioni e per l'inoltro di informazioni d'interesse per tutti i membri del gruppo.

4.1.2.4 Hangout Questo servizio di messaggistica istantanea che che supporta le videochiamate multiutente, è stato utilizzato per effettuare riunioni senza il bisogno di riunire fisicamente tutti i membri del gruppo.

4.1.3 Pianificazione

4.1.3.1 Strumenti di Pianificazione Per pianificare le attività legate allo sviluppo del progetto e la gestione delle risorse si è scelto di utilizzare **ProjectLibre**.
Si tratta di un ottimo software *Open-source*$_G$ basato su *Java*$_G$ per il project management.
La scelta è ricaduta su questo software principlamente per quattro motivi:

- Si tratta di un software portabile essendo basato su *Java*$_G$;
- È *Open-source*$_G$;
- Genera automaticamente digrammi di *Gannt*$_G$;
- Il salvataggio dei *File*$_G$ è in *XML*$_G$, quindi un formato testuale che permette di utilizzare i *Merge*$_G$ senza causare troppi conflitti.

4.1.3.2 Ticketing La piattaforma che è stata scelta per la gestione del progetto è **Redmine** che fornisce:

- Un sistema flessibile di gestione dei *Ticket*$_G$;
- Il grafico *Gantt*$_G$ delle attività;
- Un calendario per l'organizzazione e la distribuzione dei compiti;
- La visualizzazione della *Repository*$_G$ relativa al progetto;
- Un sistema di rendicontazione del tempo.

Sono state analizzate altre altenrative a **Redmine** che, dopo una fase di analisi iniziale, non sono risultate idonee allo scopo:

- **Teamworks:** si tratta del software probabilmente più adatto per il project management vista la sua grande versatilità e la strumentazione offerta. Purtroppo i suoi costi non hanno permesso un suo utilizzo in ambito universitario;
- **Zohoo:** A differenza di Redimine o di Teamworks, questa piattaforma non offre un servizio di rendicontazione del tempo e la generazione di grafici *Gantt*$_G$;

Tutti i digrammi di *Gantt*$_G$ presenti sono stati invece disegnati tramite il sotware **Gantt-project** sulla base dei dati inseriti e formiti da **Redmine** per permettere una stesura più corretta e una maggiore flessibilità nella stessa.

4.1.3.3 Protocollo di Sviluppo Per procedere con una stesura controllata dei documenti e con uno sviluppo controllato del *Codice*$_G$ si è scelto di adottare il sistema di *Ticketing*$_G$ **Redmine**.
In questa sezione si faranno molti riferimenti impliciti al *Piano di Progetto* e al *Piano di Qualifica*.

4.1.3.4 Creazione di un nuovo progetto La creazione di un progetto è un compito del *Responsabile di Progetto*.
Un nuovo progetto è una macro-attività formata da molte sotto-attività coordinate da un responsabile.
Per la creazione di un nuovo progetto la prassi da seguire è la seguente:

- Aprire **Progetti**;
- Selezionare **Nuovo Progetto**;
- Assegnare un **Nome** breve ma significativo;
- Nel caso in cui sia necessario creare un sotto-progetto, indicare il nome del progetto padre nell'omonimo campo;
- **Identificativo:** scrivere in minuscolo ed indicare il nome della fase a cui si riferisce (es. SdF-rr).

4.1.3.5 Creazione ticket I *Ticket*$_G$ vengono creati da:

- ***Responsabile di Progetto*:** crea i *Ticket*$_G$ più importanti che rappresentano le macro-fasi evideziate all'interno della pianificazione;
- ***Responsabile di Sotto-progetto*:** crea i *Ticket*$_G$ per i processi non pianificati inizialmente ma che si rivelano necessari per l'avanzamento del sotto-progetto assegnato;
- ***Verificatore*:** crea i *Ticket*$_G$ per segnalare errori emersi durante il processo di verifica.

I *Ticket*$_G$ possono essere di tre categorie:

- **Ticket di pianificazione:** rappresentano le macro-attività di maggiore importanza e sono organizzati in una gerarchia basata sul livello di importanza.
  Tali attività vengono create da:
  - *Responsabile di Progetto* che durante la pianificazione individua le attività più importanti da svolgere;
  - *Responsabile di Sotto-progetto* che durante lo svolgimento dell'attività principale può scomporla in sotto-problemi.
- **Ticket di realizzazione e controllo:** ogni documento, durante la sua stesura, passa attraverso due stadi:
  - **Realizzazione:** un redattore realizzerà la prima stesura dell'interno documento;
  - **Controllo:** un redattore, diverso dal precedente, eseguirà un primo controllo di tutta la parte scritta.

- **Ticket di verifica:** rappresentano gli errori evidenziati dai *Verificatori* durante l'operazione di controllo dell'intero documento.

4.1.4 Ticket di pianificazione

- Selezionare **Nuova segnalazione** dal menù principale;
- **Tracker:** Indicare la natura del $Ticket_G$:
  - **Documento:** Attività legata alla stesura di un documento;
  - **Codifica:** Attività legata alla codifica del software;
  - **Verifica:** Macro-attività legata alla verifica del prodotto delle macro-attività.
- **Oggetto:** Descrizione breve e significativa della natura del $Ticket_G$;
- **Descrizione:** Descrizione comprensibile dell'attività da svolgere;
- **Stato:** Plan;
- **Attività principale:** Se si vuole identificare una **sotto-attività** indicare l'id del $Ticket_G$ padre;
- **Categoria:** $PDCA_G$ se e solo se il $Ticket_G$ viene generato dal *Responsabile di Progetto*;
- **Assegnato a:** Indicare il nome del responsabile;
- **Osservatori:** Aggiungere eventuali collaboratori.

4.1.4.1 Ticket di realizzazione e controllo

- Selezionare **Nuova segnalazione** dal menù principale;
- **Tracker:** Indicare la natura del $Ticket_G$:
  - **Documento:** Attività legata alla stesura di un documento;
  - **Codifica:** Attività legata alla codifica del software;
  - **Verifica:** Macro-attività legata alla verifica del prodotto delle macro-attività;
- **Oggetto:** Descrizione breve e significativa della natura del $Ticket_G$;
- **Descrizione:** Descrizione comprensibile dell'attività da svolgere;
- **Stato:** New;
- **Attività principale:** Se si vuole identificare una **sotto-attività** indicare l'id del $Ticket_G$ padre;

- **Inizio:** Dare una presunta data di inizio;
- **Scadenza:** Dare una presunta data di fine;
- **Assegnato a:** Indicare il nome del responsabile;
- **Osservatori:** Aggiungere eventuali collaboratori.

4.1.4.2 Ticket di verifica

Un *Verificatore* per creare un *ticket di verifica* deve:

1. Assicurarsi che esista all'interno del progetto l'attività *Verifica*.
   Su questa attività devono essere presenti due sotto-attività: *Verifica - realizzazione* e *Verifica - approvazione*.
   Tutti i $Ticket_G$ devono essere creati come sotto-attività di *Verifica - realizzazione*.
2. Creare il $Ticket_G$ seguendo le seguenti direttive:
   - Selezionare **Nuova segnalazione** dal menù principale;
   - **Tracker:** Bug_G;
   - **Oggetto:** Descrizione breve e significativa della natura del $Ticket_G$;
   - **Descrizione:** Descrizione comprensibile dell'attività da svolgere;
   - **Stato:** New;
   - **Attività principale:** Se si vuole identificare una **sotto-attività** indicare l'id del $Ticket_G$ padre;
   - **Assegnato a:** Indicare il nome del responsabile.

Tutti i campi non segnalati sono da lasciarsi vuoti.
Il compito di assegnare la correzione dell'errore è dato al Responsabile del progetto padre.

4.1.4.3 Dipendenze temporali

Dopo la creazione dei $Ticket_G$ è necessario assegnare le **dipendenze temporali** fra gli stessi.
La procedura da seguire è la seguente:

- Spostarsi su **Segnalazioni**;
- Aprire il link alla segnalazione a cui aggiungere la dipendenza;
- Nella sezione **Segnalazioni correlate** premere su **Aggiungi**;
- Scegliere **segue** ed indicare il numero della segnalazione bloccante con eventuali giorni di $Slack_G$.

Tutti i campi non segnalati sono da lasciarsi vuoti.

4.1.4.4 Aggiornamento ticket L'aggiornamento dei *Ticket*$_G$ avviene tramite il cambiamento del loro stato da:

- **In Progress:** segnala che uno o più membri del gruppo stanno lavorando al completamento di quel *Ticket*$_G$.
  In questo caso la percentuale di completamento deve essere compresa fra 0% e 99%;
- **Closed:** segnala che l'attività è stata conclusa.
  La percentuale di completamento in questo caso è fissata a 100%;

4.2 Riunioni

4.2.1 Frequenza

Tutte le riunioni interne, salvo casi eccezionali, si svolgeranno settimanalmente mentre tutte quelle esterne saranno convocate solo quando sorgerà la necessità di contattare il *Proponente*$_G$ o con il *Committente*$_G$.

4.2.2 Interna

Il *Responsabile di Progetto* ha il compito di convocare le riunioni generali valutando, di volta in volta, la possibilità di anticipare o posticipare la data designata per la riunione del gruppo.
Qualora un membro del gruppo lo ritenesse necessario potrà fare richiesta, attraverso il gruppo ***Telegram***$_G$, di una riunione extra.
È auspicabile, infine, che diversi membri del gruppo possano organizzarsi fra di loro per svolgere alcuni compiti che non richiedono la presenza del gruppo di lavoro al completo. Ad esempio, è interessante e utile la collaborazione fra *Progettista* e *Analista*, senza che vengano coinvolte altre persone esterne ai compiti da loro svolti.
Il responsabile avviserà tutti i membri del gruppo attraverso un messaggio sul gruppo ***Telegram***$_G$ che verrà fissato in alto e conterrà luogo, data ed ora della riunione.
Ogni membro del gruppo è tenuto a confermare o meno la sua presenza nelle 24h successive. Il responsabile è tenuto ad avvertire telefonicamente tutti i membri che non hanno ancora risposto al messaggio.
Ogni cambiamento nell'orario di convocazione deve essere comunicato per tempo dal responsabile attraverso le modalità sopra elencate.

4.2.3 Esterna

Concordando con gli altri membri del gruppo la necessità di effettuare una riuinione con il *Proponente*$_G$ o con il *Committente*$_G$, il *Responsabile di Progetto* si metterà in contatto

con i diretti interessati per fissare una data consona a tutti.

4.2.4 Svolgimento riunione

All'apertura della riunione, verificata la presenza dei membri previsti, viene scelto un segretario che avrà il compito di annotare ogni argomento trattato e di redigere il verbale dell'assemblea, che dovrà poi essere inviato ai restanti componenti del gruppo.
Tutti i partecipanti devono osservare un comportamento consono al miglior svolgimento della riunione e al raggiungimento degli obbiettivi della stessa. Il segretario deve inoltre controllare che venga seguito l'ordine del giorno in modo da non tralasciare alcun punto.

4.2.5 Verbale

4.2.5.1 Riunione interna Il verbale di una riunione interna è un documento informale che traccia semplicemente tutti gli argomenti trattati all'interno della riunione.
Verrà redatto dal segretario della riuione, ruolo scelto di volta in volta e a rotazione fra i presenti, e dovrà essere condiviso attraverso il gruppo ***Telegram***$_G$ per essere a disposizione, in qualsiasi momento (***Telegram***$_G$ offre la possibilità di tracciare istantaneamente i media condivisi) da ogni membro del gruppo.
Il verbale dovrà essere inoltre inviato via e-mail ad ogni componente del gruppo il quale avrà cura di mantenerlo localmente al fine di avere sempre a disposizione gli argomenti trattati nel corso di una riunione.

4.2.5.2 Esterna Il verbale generato da una riunione esterna con il *Proponente*$_G$ o il *Committente*$_G$ è un documento ufficiale che può assumere il valore di normativo, quindi deve essere redatto seguendo dei criteri specifici.
Per agevolarne la scrittura è stato creato un *Template*$_G$ con *LaTeX*$_G$ che ne definisce la struttura. Vi è, ovviamente, l'obbligo di seguire tale schema per la stesura del verbale di una riunione esterna che dovrà poi essere inviato a tutti i membri del gruppo seguendo le stesse regole del verbale per una riunione interna.

4.3 Comunicazioni

4.3.1 Comunicazioni esterne

Per le comunicazioni esterne è stata creata un'apposita casella di posta elettronica:

sweet.bit.group@gmail.com

Tale indirizzo deve essere l'unico canale di comunicazione esistente tra il gruppo di lavoro e l'esterno.

L'unico membro del gruppo ad avere accesso alla mail, e quindi alle comunicazioni con il *Committente*$_G$, è il *Responsabile di Progetto*. Suo anche il compito di informare i membri del gruppo delle discussioni avvenute tramite la casella di posta eletronica e inoltrare, qualora dovesse ritenerlo necessario, tali discussioni alle caselle postali dei vari membri del gruppo.

4.4 Comunicazioni interne

Per tutte le comunicazioni interne è stato creato un gruppo ***Telegram***$_G$ in cui ogni membro può comunicare i propri impegni e il proprio stato dei lavori.
Il gruppo serve, inoltre, per la comunicazione delle date delle riuinioni, per comunicare tempestivamente le decisioni prese e per scambiare materiale utile al progetto.

4.5 Composizione e-mail

4.5.1 Destinatario

Il destinatario delle mail esterne può variare a seconda che ci si debba riferire al *Proponente*$_G$ del progetto, al Prof. Vardanega Tullio o al Prof. Cardin Riccardo.

4.5.2 Mittente

L'unico indirizzo utilizzabile è sweet.bit.group@gmail.com e deve essere usato solamente dal *Responsabile di Progetto*.

4.5.3 Oggetto

L'oggetto di una comunicazione deve essere chiaro, stringato e possibilmente diverso da altri oggetti preesistenti.
Nel caso in cui si dovesse comporre un messaggio di risposta, vi è l'obbligo di utilizzare la particella "RE:" prima dell'oggetto in modo tale da identificare il livello di risposta; Qualora si trattasse invece di un inoltro è obbligatorio utilizzare la particella "I:".
In ogni caso l'oggetto di una comunicazione già avviata non potrà essere cambiato.

4.5.4 Corpo

Il corpo di un messaggio deve contenere tutte le informazioni necessarie a rendere facilmente comprensibile l'argomento trattato a tutti i destinatari. Se alcune parti del messaggio hanno uno o più destinatari specifici, il loro nome dovrà essere aggiunto all'inizio del paragrafo tramite la seguente segnatura: *@destinatario*. In caso di risposta

od inoltro del messaggio, il contenuto aggiunto deve essere sempre messo in testa (per non costringere gli altri membri a dover scorrere tutta la mail). È caldamente consigliato, inoltre, evitare la cancellazione delle precedenti parti del messaggio in modo tale da rendere ogni partecipante alla discussione consapevole del contesto.

4.5.5 Allegati

L'uso di allegati è permesso qualora si ritenga necessario.
Un esempio è il resoconto di un incontro con il *Proponente*$_G$ o il *Committente*$_G$.

A Lista di controllo

Durante la verifica tramite walkthrough, definita dal *Piano di Qualifica v*2.0.0, sono stati rilevati numerosi errori. I principali e più comuni vengono riportati di seguito.

- **Norme stilistiche:**
  - Elenchi puntati non terminano con il punto e virgola o con il punto in caso di ultimo elemento;
  - Nomi propri di persona non cominciano con le iniziali maiuscole;
  - Inizio delle frasi non cominciano con la maiuscola;
  - Riferimenti ad altri documenti non vengono scritti in corsivo, non viene utilizzata la macro predisposta;
  - Parole Proponente e Committente vengono scritte con la minuscola iniziale.
- **Italiano:**
  - Accenti nelle parole;
  - Frasi troppo lunghe rendono difficile la lettura;
  - Doppie negazioni rendono difficile la comprensione;
  - Errori di battitura.
  - Periodo e Fase, evitare di confonderne il significato;
- **LaTeX:**
  - Evitare che compaiano elenchi di tabelle o figure vuoti;
  - Evitare che escano delle parole dalle tabelle;
  - Mancanza del carattere di spaziatura dopo l'utilizzo di macro.

B Definizione delle metriche

Il processo di verifica, per essere informativo, deve esse quantificabile. Le misure rilevate dal processo di verifica devono quindi essere basate su metriche stabilite a priori. Per automatizzare il più possibile il lavoro di misurazione saranno utilizzati degli strumenti automatizzati, con lo scopo di avere un resoconto affidabile e quantitativo che permetta di assicurare il grado di qualità voluto.

B.1 Schedule Variance (SV)

È un indice che dà informazioni necessarie a determinare se ci si trova in anticipo, in ritardo o in linea alle tempistiche delle attività di progetto. Si tratta di un indicatore di efficacia nei confronti del cliente. Se il risultato di tale metrica risulta positivo significa che il progetto sta procedendo con maggior velocità rispetto a quanto pianificato, viceversa se negativo. Alla fine del progetto questo indice assumerà il valore 0, perchè in quel momento tutte le attività saranno state realizzate. La seguente formula dà SV in termini di costo:

$$SV = BCWP - BCWS$$

Dove:

- BCWP = costo totale del lavoro svolto al momento della misurazione;
- BCWS = costo totale del lavoro pianificato al momento della misurazione.

SV ha tre significativi risultati:

- SV>0 indica che si è avanti rispetto alle pianificazione temporale del lavoro;
- SV=0 indica che si è in linea alle tempistiche delle attività di progetto;
- SV<0 indica che si è in ritardo rispetto alla pianificazione temporale delle attività.

B.2 Budget Variance (BV)

È un indice che dà informazioni necessarie a determinare se vi sono state più o meno spese rispetto al previsto. Si tratta di un indicatore che ha un valore contabile e finanziario. Se il risultato di tale metrica risulta positivo significa che l'attuazione del progetto sta consumando il proprio budget con minor velocità rispetto a quanto pianificato, viceversa se negativo.

La seguente formula dà BV in termini di costo:

$$BV = BCWS - ACWP$$

Dove:

- BCWS = costo pianificato per realizzare il lavoro al momento della misurazione;
- ACWP = costo totale richiesto per il completamento del lavoro al momento della misurazione.

BV ha tre significativi risultati:

- BV>0 indica che il progetto sta avendo un costo inferiore rispetto a quanto preventivato;
- BV=0 indica che il progetto ha un costo in linea a quanto preventivato;
- BV<0 indica che il progetto ha superato il costo preventivato.

B.3 Indice Gulpease

È un indice di leggibilità di un testo per la lingua italiana. Rispetto ad altri ha il vantaggio di utilizzare la lunghezza delle parole in lettere anziché in sillabe, semplificandone il calcolo automatico. Permette di misurare la complessità dello stile di un documento.

Questo indice considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere. La formula per il calcolo dell'indice Gulpease è:

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

Il risultato è un numero nell'intervallo [0-100], generalmente risulta che:

- inferiore a 80 il testo è difficile da leggere per chi ha la licenza elementare;
- inferiore a 60 il testo è difficile da leggere per chi ha la licenza media;
- inferiore a 40 il testo è difficile da leggere per chi ha un diploma superiore.

B.4 Complessità ciclomatica

Una metrica per la misura della complessità di funzioni, moduli, metodi o classi di un programma; la quale è calcolata mediate il grafo di controllo di flusso relativo al $Codice_G$. I nodi del grafo sono gruppi di istruzioni indivisibili. Se nel $Codice_G$ non sono presenti punti decisionali o cicli, allora la complessità ciclomatica sarà pari a 1. Alti valori di complessità ciclomatica implicano una ridotta manutenibilità del codice. Valori bassi di complessità ciclomatica potrebbero però determinare scarsa efficienza dei metodi.

La formula per il calcolo della complessità ciclomatica è:

$$Cc = a - n + 2p$$

Dove:

- Cc = indice di complessità ciclomatica;
- a = numero di archi nel grafo del $Codice_G$;
- n = numero di nodi nel grafo del $Codice_G$;
- p = numero di componenti connesse (per un singolo programma, $Metodo_G$ o funzione p è sempre 1).

B.5 Variabili non utilizzate e/o non definite

Rappresenta il numero di variabili che vengono definite, ma non utilizzate, o viceversa. Questo viene considerato pollution, e pertanto considerato inaccettabile. La misurazione avviene mediante un'analisi dell'AST (Abstract Syntax Tree).

$$VNU = numero\ variabili\ non\ utilizzate\ e/o\ non\ definite$$

B.6 Numero di argomenti per funzione

Rappresenta il numero di argomenti di una funzione. Una funzione con troppi argomenti risulta complessa e poco mantenibile; pertanto è necessario che tale numero sia contenuto.

$$NAF = numerio\ argomenti\ per\ funzione$$

B.7 Linee di codice per linee di commento

Rappresenta il rapporto tra le linee di codice e linee di commento, utile per stimare la manutenibilità del codice.

$$LCC = \frac{numerolineedicodice}{numerolineedicommento}$$

B.8 Copertura del codice

Rappresenta la percentuale di istruzioni che sono eseguite durante i test. Maggiore è la percentuale di istruzioni coperte dai test eseguiti, maggiore sarà la probabilità che le componenti testate abbiano una ridotta quantità di errori. Il valore di tale indice può essere abbassato da metodi molto semplici che non richiedono testing.

$$PCC = \frac{numerotesteseguiti}{numerotestpianificati} * 100$$