SWEG GROUP

Norme di Progetto

Versione 1.0.1
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.

1 Registro Modifiche

Modifica	Nome	Data	Ver.
Aggiunta procedura di verifica Piergiorgio Danieli	Piergiorgio Danieli 15/02/2017	22/02/2017 1.0.1	1.0.2 heightCorrezioni
Validazione	Pietro Lonardi	09/01/2017	1.0.0
Correzioni Verifica	Sebastiano Marchesini	08/01/2017	0.5.1
Verifica	Alberto Gelmi	06/01/2017	0.5.0
Aggiunti Punti nella Lista di Controllo	Sebastiano Marchesini	05/01/2017	0.1.2
Stesura Completa	Sebastiano Marchesini	02/01/2017	0.1.1
Stesura Iniziale	Sebastiano Marchesini	30/12/2016	0.1.0
Aggiunto Versionamento	Sebastiano Marchesini	15/12/2016	0.0.6
Aggiunto Ambiente di Lavoro	Piergiorgio Danieli	15/12/2016	0.0.5
Aggiunti Analisi dei Requisiti	Sebastiano Marchesini	14/12/2016	0.0.4
Aggiunte Procedure di Sviluppo	Piergiorgio Danieli	13/12/2016	0.0.3
Aggiunti Ruoli	Sebastiano Marchesini	12/12/2016	0.0.2
Creazione Documento	Piergiorgio Danieli	12/12/2016	0.0.1

Indice

2 Introduzione

2.1 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.

2.2 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.

2.3 Riferimenti

2.3.1 Normativi

- **Standard ISO/IEC 12207**.

2.3.2 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 v1.0.0".
- **Piano di Progetto**:
  "Piano di Progetto v1.0.0".

3 Comunicazioni e Riunioni

3.1 Comunicazioni Interne ed Esterne

3.1.1 Interne

Per le comunicazioni interne è stato creato un gruppo su *Telegram*, un'app di messaggistica istantanea. Questo canale sarà utilizzato solamente per le comunicazioni interne informali.

3.1.2 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.

3.2 Riunioni

3.2.1 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.

3.2.2 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 Documenti

4.1 Template

Per facilitare ed unificare la redazione della documentazione è stato creato un template apposito in LaTeX.

4.1.1 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 e nomi dei file;
- **Maiuscolo**: va usato per scrivere gli acronimi;
- **Latex**: ogni riferimento a LaTeXva scritto utilizzando il comando `\LaTex`.

4.2 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:

- `\renewcommand\arraystretch{1.2}` allarga ogni riga dello spazio indicato. La tabella va racchiusa tra parentesi graffe;
- `\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;
- `\hline` indica una linea separatrice orizzontale;
- `&` è il simbolo separatore tra le celle orizzontalmente;
- `\\` passa alla riga successiva della tabella.

Per le immagini:

- `\begin{figure} \end{figure}` inizio e fine di una sezione con figure;
- `\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 della tabella.

4.3 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".

4.4 Struttura del documento

4.4.1 Frontespizio

Il frontespizio di ogni documento dovrà essere così strutturato:

- Logo-nome del gruppo;
- Titolo del documento;
- Versione del documento;
- Nome e cognome dei redattori del documento;
- Nome e cognome dei revisori del documento;
- Uso del documento;
- Destinatari del documento.

4.4.2 Diario delle modifiche

La seconda pagina deve essere una tabella contenente i cambiamenti che sono stati effettuati nel documento. Deve essere così strutturata:

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

La tabella è ordinata per data in ordine decrescente, in modo che la prima riga sia l'ultima modifica eseguita, e quindi corrisponda alla versione attuale del documento.

4.4.3 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.
L'indice è autogenerato con un comando di LaTeX.

4.4.4 Formattazione generale

Tutte le altre pagine del documento 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: compare a piè di pagina, a sinistra;
- Numero di pagina: compare a piè di pagina, a destra.

Ogni pagina deve rispettare i seguenti margini:

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

4.5 Classificazione documenti

4.5.1 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 dal *Project Manager*.

4.5.2 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.

4.6 Componenti grafiche

4.6.1 Tabelle

Tutte le tabelle presenti nella documentazione necessitano di una didascalia che le descrive brevemente e che porta un numero progressivo in modo da identificarla univocamente.

4.6.2 Immagini

Le immagini come le tabelle dovranno essere accompagnate da una didascalia ed un numero progressivo che le identifica.

4.7 Intestazione file di documentazione

Ogni file di documentazione dovrà iniziare con la seguente intestazione:

- %FILE: nome del file;
- %PERCORSO: /PercorsoDelFile/NomeDelDocumento/;
- %DATA CREAZIONE: gg/mm/aa;
- %AUTORE: SWEg;
- %EMAIL: sweg.group@gmail.com.

5 Ruoli

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 Sistema rotativo dei ruoli

Il progetto a scopo didattico ha l'obbligo di far interpretare ogni ruolo a tutti i partecipanti del team. Per il momento assumiamo un modello agile di tipo *scrum* per eseguire i compiti (o *ticket*). Quindi dopo aver costruito una prima check list in gruppo ogni obbiettivo viene risolto da un sottogruppo in cui il primo che preferisce eseguirlo assume il ruolo di *Project Manager* e vaglia oggettivamente ogni sfaccettatura per risolvere il *backlog* richiedendo le risorse di cui ha bisogno.

5.3 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.

5.3.1 Tabella Costi

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

6 Analisi dei Requisiti

6.1 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.

6.2 Documento Analisi dei Requisiti

Sempre gli *Analisti* hanno 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.

6.2.1 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.

6.2.2 Allocazione e modellazione concettuale del sistema

Il team *Analisti*, dopo aver preso visione nello specifico del capitolato e aver fatto emergere i requisiti, procede con la costruzione dei 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;
- **Precondizione**;
- **Postcondizione**;
- **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 2.x e 1.x* con le funzioni principali che racchiudono le due versioni di linguaggio.

6.3 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.

È nostro compito automatizzare il processo con software adeguato.

7 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.

7.1 Acquisizione

L'acquisizione comprende tutte le attività coinvolte nel promuovere un progetto. La fase di acquisizione può essere suddivisa in diverse attività e risultati finali che sono stati completati in ordine cronologico.

7.1.1 Obbiettivo

In questa fase lo scopo da raggiungere è l'accordo tra le parti per eseguire il lavoro.

7.1.2 Iniziazione

Durante questa attività sono completati i seguenti compiti:

- Viene descritto il motivo per acquisire, sviluppare, o migliorare un prodotto;
- I requisiti di sistema sono definiti ed approvati;
- I requisiti software globali sono definiti;
- La valutazione di altre opzioni, come un acquisto prodotto già pronto o il miglioramento di un prodotto esistente;
- Viene sviluppato un piano di acquisizione, questo piano verrà utilizzato più avanti;
- I criteri di accettazione sono definiti.

7.1.3 Richiesta di preparazione delle proposte

Durante questa attività sono svolte le seguenti mansioni:

- Vengono definiti i requisiti di acquisizione come i requisiti di sistema e dei vincoli tecnici;
- Vengono definite le *milestone* del contratto per la revisione ed audit il progresso.

7.1.4 Preparazione contrattto

- Viene sviluppata una procedura di selezione per i fornitori;
- Vengono scelti i fornitori sulla base della procedura sviluppata.

7.1.5 Negoziare i cambiamenti

In questa fase si svolgono i negoziati con i fornitori scelti.

7.1.6 Accettazione e completamento

- Vengono sviluppati i test e le procedure di accettazione;
- Viene condotta la gestione della configurazione sul prodotto consegnato.

7.2 Fornitura

Formato dai:

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

7.2.1 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.

7.2.2 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 v1.0.0"*.

7.2.3 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.

7.2.4 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.

7.2.5 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.

7.2.6 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.

7.3 Sviluppo

Il metodo a *cascata sequenziale* è il modello utilizzato per ogni passaggio nel processo di sviluppo. Il *Responsabile di Progetto* granula ogni attività per poi rendere il modello agile di tipo *scrum*. Si rimanda al *"Piano di Progetto v1.0.0"* 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.

7.3.1 Analisi

È la prima attività di studio e di stesura dei documenti. Dopo un primo incontro con i membri del team e il team concorrente si stabiliscono i ruoli e si dividono i compiti come già menzionato o come descritto nel Piano di Progetto. È un processo lungo e lascia le basi per le future versioni.

7.3.2 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.

7.3.3 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. In particolare un documento fondamentale in questa fase è chiamato Specifica Tecnica, il quale deve dare le indicazioni necessarie al programmatore per poter scrivere nel modo corretto il software.

7.3.4 Progettazione di Dettaglio e Codifica

Ogni parte del progetto in questa fase è definita e progettata. Il sistema è modulato in ogni minima soluzione. Resta solo la codifica e la verifica del codice. In questa fase viene anche redatto il Manuale Utente, il quale dovrà spiegare il corretto utilizzo e funzionamento del software in modo dettagliato e preciso.

7.3.5 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.

7.4 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.

7.5 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.Questi, e le carenze, di solito sono documentati in via di sviluppo per consentire soluzioni future in ogni *release* di manutenzione future. Non vi è alcuna fase di smaltimento.

8 Procedure di sviluppo

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 v1.0.0*.

8.1 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.

8.1.1 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.

8.1.2 Coordinamento delle attività

Una volta pianificate le attività, il *Project Manager* le assegna ad ognuno dei membri attraverso l'ausilio offerto da *Asana*. Ogni componente può quindi verificare quali attività gli sono state assegnate e modificarne lo stato, facendo in modo che il *Project Manager* possa seguire con facilità i progressi effettuati.

8.1.3 Analisi dei rischi

Durante tutto l'avanzamento del progetto il *Project Manager* ha il compito di valutare eventuali rischi che possono venirsi a creare e che sono descritti nel *Piano di Progetto v1.0.0* attuando le adeguate contromisure descritte.

8.1.4 Elaborazione dei dati

Il *Project Manager* dovrà creare eventuali grafici esplicativi dell'andamento del progetto, e riportarli nel *"Piano di Progetto v1.0.0"*.

8.2 Verifica

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

8.2.1 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 di tutti i processi. 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 v1.0.0"*. 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 Misure e metriche.

8.2.2 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 la coerenza e la 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.

8.2.3 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 v1.0.0"* 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 v1.0.0"*;

8.2.4 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 v1.0.0*.
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**: 15/05/2017.

8.2.5 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*.

8.2.6 Risorse

Per realizzare il prodotto sono necessarie le seguenti risorse:

- **Risorse umane**: vengono descritte dettagliatamente nel documento *"Piano di Progetto v1.0.0"*. 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*;
  - automatizzare le verifiche;
  - sviluppare nei linguaggi di programmazione scelti;
  - analizzare il codice scritto;
  - gestire i test che effettueremo sul codice.
- **Risorse hardware**: sono necessari computer aventi applicativi descritti in questo documento.

8.2.7 Tecniche di analisi -Analisi statica

L'analisi statica è l'analisi eseguita senza una effettiva esecuzione dei programmi. Viene eseguita quindi 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.

8.2.8 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;
  - capo riga e spazi prima di ogni sezione, sottosezione e sotto-sottosezione.
- Italiano:
  - scorretto utilizzo di accenti acuti e gravi;
  - punteggiatura nelle liste;
  - fluidità delle frasi.
- LaTeX:
  - controllo spaziatura delle tabelle (prima, nel mezzo e dopo);
  - controllo delle liste;
  - termini non evidenziati (corsivo e/o grassetto).
- UML:
  - direzione delle frecce del diagramma errate;
  - specificare la versione;
  - 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;
  - i verbi devono indicare azioni;
  - non usare fork e join insieme.
- Design pattern:
  - deve essere sempre contestualizzato, la descrizione non basta.

Verifica documenti Il *Verificatore* nei documenti dovrà:

- Controllare nel diario delle modifiche i cambiamenti avvenuti dall'ultima volta;
- Cercare eventuali errori 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.

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

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

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 logica del programma.

8.2.9 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 nello stesso ambiente e con gli stessi input. 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.

8.2.10 Misure e metriche

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.
Vi possono essere due tipologie di range nell'accettazione dei risultati di un processo di verifica:

- **Accettazione**: è il livello minimo che il prodotto deve raggiungere per superare il test;
- **Ottimale**: è il livello in cui dovrebbe posizionarsi la misurazione. In caso questo valore non venga raggiunto sono necessarie verifiche aggiuntive.

8.2.11 Misure e metriche - Metriche per i processi

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.
Parametri utilizzati:

- **Range di accettazione**: $\geq$ -(preventivo fase*5%);
- **Range ottimale**: ≥ 0.

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.
Parametri utilizzati:

- **Range di accettazione**:$\geq$-(preventivo fase*10%);
- **Range ottimale**:≥ 0.

8.2.12 Misure e metriche - Metriche per i documenti

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. In generale risulta che testi con un indice:

- inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
- inferiore a 60 sono difficili da leggere per chi ha la licenza media;
- inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

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.
Parametri utilizzati:

- **Range di accettazione**: [40 - 100];
- **Range ottimale**: [50 - 100].

8.2.13 Misure e metriche - Metriche per il software

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.
Parametri utilizzati:

- **Range di accettazione**: [1 - 15];
- **Range ottimale**: [1 - 10].

Rapporto linee di commento su linee di codice Affinché il codice sia più facilmente comprensibile e manutenibile è necessario che sia adeguatamente commentato.
Parametri utilizzati:

- **Range di accettazione**: [>0.2];
- **Range ottimale**: [>0.3].

Numero di livelli di annidamento Indica il livello di annidamento delle strutture di controllo. Un numero troppo elevato può comportare difficoltà nella verifica e nella manutenibilità del codice.
Parametri utilizzati:

- **Range di accettazione**: [0-6];
- **Range ottimale**: [0-4].

Numero di campi dati per classe Una classe con troppi campi dati potrebbe essere poco specializzata e quindi mal progettata. Una classe poco specializzata inoltre è anche poco manutenibile.
Parametri utilizzati:

- **Range di accettazione**: [0-15];
- **Range ottimale**: [1-8].

Accoppiamento 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:

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

Validazione W3C 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.
Parametri utilizzati:

- **Range di accettazione**: [0-5];
- **Range ottimale**: [0-0].

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.
Parametri utilizzati:

- **Range di accettazione**: 6/10;
- **Range ottimale**: 10/10.

8.3 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 autoavverante.
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.

9 Ambiente di lavoro

9.1 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*.

9.2 Applicativo Guida

Si è scelto per il coordinamento:

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

9.3 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.

9.4 Documentazione

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

9.4.1 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 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.

9.4.2 Edior

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.

9.4.3 Diagrammi UML

Per la creazione dei diagrammi *UML* si è preso come punto di riferimento 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.

9.4.4 Verifica

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

9.5 Pianificazione

9.5.1 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 perchè 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);
- impostare dipendenze tra le attività.

9.5.2 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*;
- 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*. 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.

9.6 Versionamento

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

- È strutturato come un File System;
- 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 modifca 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.

9.7 Repository

9.7.1 Repository documentazione

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

- AnalisiDeiRequisiti;
- Glossario;
- NormeDiProgetto;
- PianoDiProgetto;
- PianoDiQualifica;
- StudioDiFattibilità;
- Verbali.

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

9.7.2 Repository progetto

Non avendo ancora iniziato il progetto non vi è ancora una sezione dedicata alle cartelle dei file.