>Three Way Milkshake_

# Norme di Progetto

**Three Way Milkshake - Progetto "PORTACS"**

threewaymilkshake@gmail.com

| | |
|---|---|
| **Versione** | 3.0.0 |
| **Stato** | Approvato |
| **Uso** | Interno |
| **Approvazione** | De Renzis Simone |
| **Redazione** | Greggio Nicolò |
| | Tessari Andrea |
| **Verifica** | Zuccolo Giada |
| | Crivellari Alberto |
| **Destinatari** | Three Way Milkshake |
| | Prof. Vardanega Tullio |
| | Prof. Cardin Riccardo |

**Descrizione**

Questo documento contiene tutte le norme di progetto$_G$, definite inizialmente o aggiunte in seguito, viene quindi aggiornato per incrementi successivi.

# Registro delle modifiche

| Vers. | Descrizione | Data appr. | Approvazione |
|---|---|---|---|
| 3.0.0 | Approvazione del documento | 2021-04-27 | De Renzis Simone |

| Vers. | Descrizione | Redazione | Data red. | Verifica | Data ver. |
|---|---|---|---|---|---|
| 2.2.1 | Incremento § 2.2.4.1.2 e 3.1.5.3 | Greggio Nicolò | 2021-04-21 | Zuccolo Giada | 2021-04-22 |
| 2.2.0 | Redazione § 2.2.4.3 | Greggio Nicolò | 2021-04-03 | Zuccolo Giada | 2021-04-05 |
| 2.1.0 | Redazione appendice § A | Crivellari Alberto | 2021-04-01 | Greggio Nicolò | 2021-04-02 |
| 2.0.1 | Modifiche § 3 | Crivellari Alberto | 2021-04-01 | Greggio Nicolò | 2021-04-02 |

| Vers. | Descrizione | Data appr. | Approvazione |
|---|---|---|---|
| 2.0.0 | Approvazione del documento | 2021-03-08 | Chiarello Sofia |

| Vers. | Descrizione | Redazione | Data red. | Verifica | Data ver. |
|---|---|---|---|---|---|
| 1.1.1 | Modifica § 3.4.3 | Tessari Andrea | 2021-03-02 | Crivellari Alberto | 2021-02-08 |
| 1.1.0 | Modifica e stesura § 4.1.3, 3.1.3 | Tessari Andrea | 2021-02-13 | Crivellari Alberto | 2021-02-08 |
| 1.0.2 | Modifica § 3.1.8.5 | Tessari Andrea | 2021-02-13 | Crivellari Alberto | 2021-02-08 |
| 1.0.1 | Modifica § 1.3, 3.1.5.1, 3.1.5.7, 3.1.7 | Tessari Andrea | 2021-02-13 | Crivellari Alberto | 2021-02-08 |

| Vers. | Descrizione | Data appr. | Approvazione |
|---|---|---|---|
| 1.0.0 | Approvazione del documento | 2021-01-10 | De Renzis Simone |

| Vers. | Descrizione | Redazione | Data red. | Verifica | Data ver. |
|---|---|---|---|---|---|
| 0.5.1 | Terminata stesura § 4 | Greggio Nicolò | 2021-01-03 | Crivellari Alberto | 2021-01-09 |
| 0.5.0 | Stesura § 4.1, 4.2 | Greggio Nicolò | 2021-01-02 | Crivellari Alberto | 2021-01-09 |
| 0.4.1 | Aggiunta § 3.6 | Greggio Nicolò | 2020-12-30 | Zuccolo Giada | 2021-01-09 |

| | | | | | |
|---|---|---|---|---|---|
| 0.4.0 | Stesura § 3.1, 3.2 | Greggio Nicolò | 2020-12-28 | Zuccolo Giada | 2021-01-09 |
| 0.3.0 | Termine stesura § 2 | Greggio Nicolò | 2020-12-26 | Zuccolo Giada | 2021-01-09 |
| 0.2.2 | Stesura § 2.2 | Greggio Nicolò | 2020-12-22 | Zuccolo Giada | 2021-01-09 |
| 0.2.1 | Stesura § 2.1 | Greggio Nicolò | 2020-12-18 | Zuccolo Giada | 2021-01-09 |
| 0.2.0 | Stesura § 1 | Greggio Nicolò | 2020-12-15 | Zuccolo Giada | 2021-01-09 |
| 0.1.0 | Creazione e strutturazione documento nelle sue | Greggio Nicolò | 2020-12-13 | Zuccolo Giada | 2021-01-09 |

# Indice

# Elenco delle figure

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento ha lo scopo di fissare e definire tutte le regole, convenzioni e buone pratiche utili a formare un way of working condiviso alla base da tutti i componenti del gruppo per assicurare una collaborazione efficiente$_G$ ed efficace$_G$. Si discuteranno inoltre i vari strumenti che verranno adottati per facilitare lo sviluppo del progetto$_G$ e per promuovere un'organizzazione adeguata. Si ritiene inoltre che la definizione ed il mantenimento per incremento di un documento condiviso all'interno del gruppo di lavoro, che definisca e raccolga quanto descritto in maniera formale e centralizzata, possa favorire, in un contesto dove i membri possano variare, l'inserimento di nuovi componenti facilitandone l'ambientamento. Pur non essendo questo il contesto di lavoro attuale, è comunque una buona pratica da sperimentare e consolidare.

## 1.2 Scopo del prodotto

Il capitolato$_G$ C5 propone un progetto$_G$ in cui viene richiesto lo sviluppo di un software per il monitoraggio in tempo reale di unità che si muovono in uno spazio definito. All'interno di questo spazio, creato dall'utente per riprodurre le caratteristiche di un ambiente reale, le unità dovranno essere in grado di circolare in autonomia, o sotto il controllo dell'utente, per raggiungere dei punti di interesse posti nella mappa. La circolazione è sottoposta a vincoli di viabilità e ad ostacoli propri della topologia dell'ambiente, deve evitare le collisioni con le altre unità e prevedere la gestione di situazioni critiche nel traffico.

## 1.3 Termini, abbreviazioni ed altri documenti

Tutti i termini che necessitano di una spiegazione, per fornire un'adeguata comprensione, o perché possono causare ambiguità nel contesto, sono definiti nel glossario. A dispetto di tutti gli altri documenti presenti, il glossario è sotto la veste di una pagina web, più nello specifico in una wiki$_G$ di GitHub, accessibile a questo indirizzo, con vocaboli suddivisi tra "Acronimi" e "Termini", disposti in ordine alfabetico al fine di facilitare la navigazione. All'interno dei documenti le voci di glossario saranno seguite da una G pedice mentre gli acronimi da una A (e.g.: voce di glossario$_G$ ; acronimo$_A$). Inoltre quando si farà riferimento ad un altro documento o al documento stesso, il nome di questo sarà in maiuscoletto (e.g.: ESEMPIO NOME DOCUMENTO).

## 1.4 Riferimenti

### 1.4.1 Riferimenti Normativi

- Standard ISO 12207:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf;
- Standard UML$_A$ 2.0:
  https://www.omg.org/spec/UML/2.0/Superstructure/PDF;
- Diagrammi dei casi d'uso$_G$:
  https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20Use%20Case_4x4.pdf;

- Diagrammi delle classi:
https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20delle%20Classi_4x4.pdf;
- Diagrammi dei package:
https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20dei%20Package_4x4.pdf;
- Diagrammi di attività$_G$:
https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20di%20Attivit%c3%a0_4x4.pdf;
- Diagrammi di sequenza:
https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20di%20Sequenza_4x4.pdf;
- Standard ISO 8601:
https://www.iso.org/iso-8601-date-and-time-format.html.

### 1.4.2 Riferimenti Informativi

- Glossario:
per la definizione dei termini (pedice G) e degli acronimi (pedice A) evidenziati nel documento;
- https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L03.pdf
- https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L06.pdf
- https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/FC2.pdf
- https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf
- https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L09.pdf
- https://docs.github.com/en/free-pro-team@latest/actions
- https://www.latex-project.org/
- http://www.texstudio.org/
- https://www.xm1math.net/texmaker/
- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- https://github.com/about
- https://www.atlassian.com/software/jira
- https://www.atlassian.com/
- https://meet.google.com/
- https://slack.com/intl/en-it/about
- https://telegram.org/
- https://workspace.google.com/products/chat/

# 2 Processi primari

## 2.1 Fornitura

### 2.1.1 Scopo

Il processo di fornitura sostanzialmente si occupa della gestione dei rapporti con il cliente. Il suo scopo quindi è quello di determinare strumenti e competenze utili e necessari alla realizzazione del prodotto e di assicurare la conformità di questo con le richieste del proponente. Si rende necessaria quindi la produzione di documenti che descrivano le intenzioni e le modalità che il gruppo si prefigge di seguire al fine di soddisfare il cliente.

### 2.1.2 Aspettative

Il confronto diretto e frequente tra fornitore e proponente è senza dubbio utile ad entrambe le parti, affinché ambedue soddisfino i loro obiettivi in tempi desiderabili.

### 2.1.3 Descrizione

Il processo di fornitura si compone [1] di 7 attività$_G$, definite come segue:

### 2.1.4 Attività

#### 2.1.4.1 Inizializzazione

Il gruppo dovrà effettuare collettivamente una valutazione di tutti i capitolati$_G$ proposti e formalizzarla in uno STUDIO DI FATTIBILITÀ, il quale, per ogni capitolato$_G$, darà una breve descrizione dello stesso, delle finalità, delle tecnologie, degli aspetti positivi e delle criticità, proponendo poi delle conclusioni che saranno il frutto delle riflessioni interne ed indicando la scelta definitiva dei membri.

#### 2.1.4.2 Preparazione della risposta

Il gruppo preparerà una lettera di presentazione indirizzata al committente ed al proponente del capitolato$_G$ scelto, per candidarsi alla fornitura del prodotto indicando un sunto del preventivo dei costi.

#### 2.1.4.3 Pianificazione

Il gruppo dovrà fornire dei documenti che illustrino la gestione del lavoro e mostrino come verranno assicurati qualità e conformità del prodotto. Nello specifico realizzerà:

- un PIANO DI PROGETTO, contenente [2]:
  - **pianificazione macroscopica (a lungo periodo):**
    - scadenze, fissate all'indietro;
    - analisi dei rischi;
    - preventivo dei costi.
  - **pianificazione dettagliata (a breve):**
    - attività$_G$, fissate in avanti;

[1] Standard ISO 12207 § 5.2
[2] https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/FC2.pdf § 3

* preventivo minuto, alla luce del consuntivo di periodo$_G$ precedente;
* riscontro dei rischi ed aggiornamento delle misure di mitigazione.

che andrà così strutturato: [3]:

– Introduzione (scopo e struttura);
– Organizzazione del progetto$_G$;
– Analisi dei Rischi;
– Risorse disponibili (tempo e persone);
– Suddivisione del lavoro (work breakdown);
– Calendario delle attività$_G$;
– Meccanismi di controllo e di rendicontazione.

- un PIANO DI QUALIFICA, contenente [4]:
  – obiettivi quantitativi di qualità;
  – cruscotto$_G$ di misurazione;
  – analisi degli scostamenti e misure correttive.

#### 2.1.4.4 Esecuzione e controllo

In questa attività$_G$ il gruppo Three Way Milkshake dovrà implementare ed eseguire i piani delineati al punto 2.1.4.3 e sviluppare il prodotto in accordo con il processo di sviluppo.

#### 2.1.4.5 Revisione e valutazione

Il gruppo dovrà coordinare le revisione$_G$ delle attività$_G$ svolte e gestire la comunicazione con il committente ed il proponente. Si dovrà inoltre aver cura di operare in accordo con quanto scritto negli altri processi.

#### 2.1.4.6 Consegna e completamento

Il fornitore dovrà consegnare il prodotto in accordo con quanto specificato nel contratto. A seguito della consegna del prodotto, il gruppo Three Way Milkshake non si farà carico delle mansioni di supporto ed assistenza.

[3] https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L06.pdf § 25
[4] https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/FC2.pdf § 4

## 2.2 Sviluppo

### 2.2.1 Scopo

Il processo di sviluppo comprende tutte quelle attività$_G$ che portano alla costruzione del prodotto finale.

### 2.2.2 Aspettative

Questo processo dev'essere attuato secondo quanto pattuito con proponente e committente, rispettando i loro requisiti$_G$ e soddisfacendo le loro aspettative. Tutto ciò naturalmente rispettando le norme definite in questo documento.

### 2.2.3 Descrizione

Nel processo di sviluppo si individuano [5] le seguenti attività$_G$:

### 2.2.4 Attività

#### 2.2.4.1 Analisi dei requisiti

##### 2.2.4.1.1 Descrizione

Gli analisti devono stabilire, raccogliere e documentare tutti i requisiti$_G$ stilando un documento che fornirà una base precisa su cui i progettisti si potranno fondare. Dovrà contenere, in accordo con quanto richiesto dal cliente, la raccolta dei casi d'uso$_G$, rappresentati anche tramite diagrammi UML$_A$, ed il tracciamento di tutti i requisiti$_G$ individuati.

##### 2.2.4.1.2 Nomenclatura casi d'uso

Ogni caso d'uso$_G$ è univocamente identificato da un codice, secondo lo schema:

UC[caso].[sottoCaso].[sottoSottoCaso].[sottoSottoSottoCaso]

dove caso ed i successivi sono numeri progressivi che partono da 1, solo il primo è obbligatorio. Segue poi un breve nome. Inoltre ogni caso dovrà avere le seguenti sezioni (quelle indicate tra parentesi (...) sono opzionali):

- attori$_G$ primari;
- (attori secondari);
- descrizione;
- (estensioni);
- (inclusioni);
- precondizione$_G$;
- post condizione$_G$;
- scenario$_G$ principale, uno solo per caso d'uso$_G$, rappresenta il normale flusso degli eventi;

[5] Standard ISO 12207 § 5.3

- (scenari alternativi).

Tutti i casi d'uso$_G$ non banali verranno accompagnati da diagrammi secondo lo standard UML$_A$ 2.0 [6] [7]. Questo è un esempio di caso d'uso$_G$ conforme alle regole appena definite:

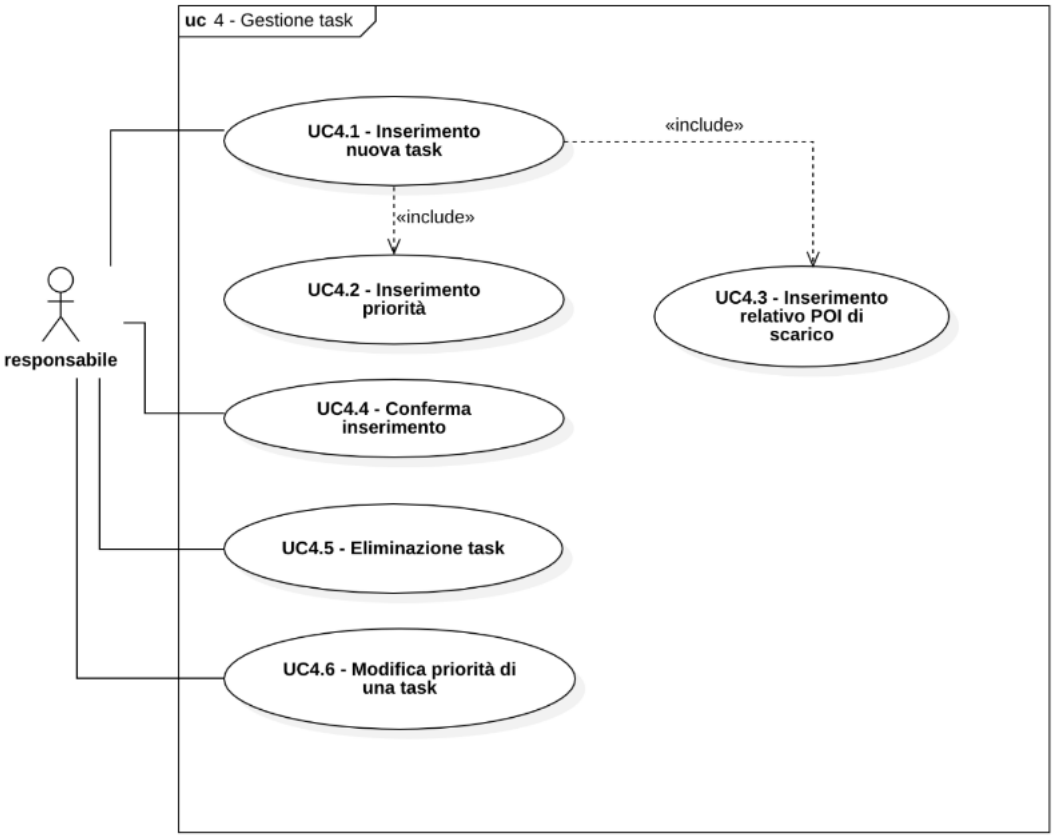


Figura 2.2.1: Esempio di caso d'uso

#### 2.2.4.1.3 Classificazione dei requisiti

Per favorire l'organizzazione ed il tracciamento dei requisiti$_G$, questi saranno classificati secondo la convenzione che segue:

R[tipo]-[numero progressivo]-[importanza]

dove tipo è uno tra:

- **V**→ vincolo;
- **F**→ funzionale;
- **Q**→ qualità;

[6] https://www.omg.org/spec/UML/2.0/Superstructure/PDF § 16
[7] https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20Use%20Case_4x4.pdf

- **P**→ prestazionale;

mentre importanza può essere:

- **O**→ obbligatorio, irrinunciabile per qualcuno degli stakeholder$_G$;
- **D**→ desiderabile, non strettamente necessario, ma se ne riconosce il valore aggiunto;
- **F**→ facoltativo, relativamente utile oppure contrattabile in futuro.

Il numero progressivo parte da 1 ed i sotto requisiti$_G$ si indicano in maniera analogo ai sotto casi d'uso$_G$.

#### 2.2.4.2 Progettazione

##### 2.2.4.2.1 Descrizione

I progettisti hanno il compito di tradurre il problema in una possibile soluzione, descritta come architettura, da dividere in parti che possono essere trattate individualmente. Sarà necessario:

- svolgere le attività$_G$ coerentemente con quanto individuato nell'analisi dei requisiti$_G$;
- adottare design pattern$_G$ opportuni quando si individuano strutture che ne possono trarre beneficio;
- seguire i principi SOLID[8];
- utilizzare sempre una nomenclatura significativa, parlante e consistente.

##### 2.2.4.2.2 Obiettivi di qualità

Affinché un'architettura si possa definire buona, deve perseguire i seguenti obiettivi di qualità: [9]

- **sufficienza:** soddisfa tutti i requisiti$_G$;
- **comprensibilità:** per tutti gli stakeholder$_G$;
- **modularità:** suddivisa in parti chiare e ben distinte;
- **robustezza:** capace di sopportare ingressi diversi da utenti ed ambienti differenti;
- **flessibilità:** permette modifiche a costo contenuto al variare dei requisiti$_G$;
- **riusabilità:** le sue parti possono essere impiegate in altre applicazioni;
- **efficienza:** nel tempo (CPU), nello spazio (RAM), nelle comunicazioni (rete);
- **affidabilità:** svolge bene il suo compito quando utilizzata con un'alta probabilità;
- **disponibilità:** la sua manutenzione la rende indisponibile per un tempo ridotto;
- **safety$_G$** ;

[8] https://www.math.unipd.it/~rcardin/swea/2021/SOLID%20Principles%20of%20Object-Oriented%20Design_4x4.pdf

[9] https://www.math.unipd.it/~tullio/IS-1/2020/Dispense/L09.pdf § 17..27

- **security$_G$** ;
- **semplicità:** ogni parte contiene solo il necessario e niente di superfluo;
- **incapsulazione:** l'interno delle componenti non è visibile all'esterno;
- **coesione:** le parti che stanno insieme hanno gli stessi obiettivi;
- **basso accoppiamento:** parti distinte hanno una dipendenza il più possibile ridotta.

#### 2.2.4.2.3 Attività

La progettazione porterà in seguito alla produzione di una Technology Baseline$_G$ e di una Product Baseline$_G$. A supporto di queste, ci saranno diversi diagrammi, sempre secondo lo standard UML$_A$ 2.0:

- **delle classi**[10]**:** per rappresentare oggetti e relazioni tra questi in un sistema;
- **dei package**[11]**:** per rappresentare dipendenze tra classi e package ad un livello di astrazione più alto rispetto a quello dei diagrammi delle classi;
- **delle attività**[12]**:** per descrivere processi o algoritmi;
- **di sequenza**[13]**:** la collaborazione di un gruppo di oggetti che devono implementare collettivamente un comportamento.

Si dovranno indicare le tecnologie che si andranno ad utilizzare, descrivendo il loro impiego e le motivazioni che hanno portato alla loro scelta.

### 2.2.4.3 Codifica

Qui si definiscono le norme che i programmatori dovranno seguire, di modo da:

- perseguire: leggibilità, uniformità e consistenza;
- agevolare: verifica, validazione e manutenzione.

#### 2.2.4.3.1 Convenzioni

Per uniformare lo stile di codifica all'interno del gruppo si seguono le linee guida di Google per i vari linguaggi adottati:

- java:
  https://google.github.io/styleguide/javaguide.html;
- javascript:
  https://google.github.io/styleguide/jsguide.html;
- typescript:
  https://google.github.io/styleguide/tsguide.html;
- JSON:
  https://google.github.io/styleguide/jsoncstyleguide.xml;
- HTML/CSS:
  https://google.github.io/styleguide/htmlcssguide.html.

[10] https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20delle%20Classi_4x4.pdf
[11] https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20dei%20Package_4x4.pdf
[12] https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20di%20Attivit%c3%a0_4x4.pdf
[13] https://www.math.unipd.it/~rcardin/swea/2021/Diagrammi%20di%20Sequenza_4x4.pdf

#### 2.2.4.3.2 Strumenti

Viene adottato l'editor di testo Visual Studio Code [14] in quanto molto leggero ma allo stesso tempo ricco e potente grazie all'ampio ventaglio di estensioni e plugin a disposizione. Inoltre è ugualmente disponibile per Linux, MacOS e Windows.
Per il rispetto delle regole sopracitate e per automatizzare la formattazione uniforme del codice si adottano gli strumenti:

- spotless per java, configurabile come plugin gradle:
  - https://github.com/diffplug/spotless/tree/master/plugin-gradle;
- prettier per tutti gli altri linguaggi, installabile ed utilizzabile tramite $npm_A$:
  - https://prettier.io/docs/en/index.html.

[14] Documentazione VS Code: https://code.visualstudio.com/docs

# 3 Processi di Supporto

## 3.1 Documentazione

### 3.1.1 Scopo

Lo scopo del processo di documentazione è regolamentare la creazione e la gestione dei documenti e fissare le modalità di stesura ed approvazione degli stessi.

### 3.1.2 Aspettative

Avere un approccio condiviso ed uniforme per la stesura e l'aggiornamento dei documenti all'interno del gruppo di lavoro è fondamentale per rendere la documentazione uno strumento costruttivo e di supporto, e non un mera formalità aggiuntiva. Inoltre fornire un aspetto uniforme attraverso tutti i documenti facilita qualunque lettore.

### 3.1.3 Descrizione

Il gruppo Three Way Milkshake si doterà di due categorie di documentazione:

- **formale:** documenti interni o esterni che rispetteranno strettamente i vincoli descritti in seguito e che saranno interamente pubblici, realizzati in LaTeX aderendo ad un template condiviso;
- **informale:** documenti interni che potranno svolgere diverse funzioni, tra cui:
  - raccolta appunti e ordini del giorno per riunioni;
  - raccolta argomenti delle discussioni delle riunioni, per tracciare l'evoluzione delle stesse e favorire la stesura dei verbali in seguito;
  - creazione di wiki$_G$, workflow e how-to per condivisione di materiale utile riguardo l'uso di tecnologie o strumenti a supporto di qualsiasi attività$_G$;
  - raccolta di varie annotazioni, catalogate per documento .

Questi documenti saranno realizzati sfruttando Confluence per garantire semplicità, accentramento e condivisione real-time.

### 3.1.4 Ciclo di vita dei documenti

Ogni documento formale si redige ed incrementa tramite queste attività$_G$:

- **stesura:** la scrittura del documento in sé, riguarda sia la creazione di nuove parti che l'aggiornamento di queste. Uno o più redattori si occupano di ciò;
- **verifica:** eseguita da uno o più verificatori, necessariamente diversi dai redattori, consiste nel controllo della correttezza sintattica, semantica, grammaticale ed ortografica e della conformità del documento rispetto alle suo scopo. Nel caso in cui si rendano necessarie modifiche sostanziali, i verificatori notificheranno il responsabile che provvederà a riportare il documento in stesura e solleciterà i redattori affinché apportino le correzioni richieste. I verificatori sono autorizzati ad apportare piccole modifiche correttive quali:
  - correzione di errori:

    * sintattici;
    * ortografici;
    * grammaticali;
  - riformulazione di frasi per dare aspetto più formale e consistente nel caso non sia già sufficiente;

- **approvazione:** quando i verificatori riscontreranno la completa correttezza ed aderenza ai requisiti$_G$ del documento, il responsabile provvederà all'approvazione finale ed al rilascio di una nuova versione dello stesso.

Adottando un approccio incrementale$_G$, queste attività$_G$ possono ripetersi.

### 3.1.5 Struttura dei documenti formali

#### 3.1.5.1 Organizzazione in file e cartelle

Ogni documento, ha una sua cartella dedicata, all'interno della quale ci devono essere:

- **file principale del documento:** (nome_documento.tex) che contiene l'impostazione della struttura, le dichiarazioni per importare i package LaTeX aggiuntivi e le dichiarazioni di inclusione delle sezioni;
- **cartella config:** contenente file di configurazione relativi al documento, che consentono l'impostazione del frontespizio e del registro delle modifiche;
- **cartella res:** per contenere le risorsa$_G$ del documento, a sua volta questa contiene:
  - **cartella images:** per le eventuali immagini;
  - **cartella sections:** che contiene tutte le sezioni, ognuna in un file diverso, i quali seguono la convenzione di nomenclatura preceduta da un identificativo numerico progressivo ad indicare la posizione della stessa nel documento (e.g.: 01_introduzione.tex, 02_sezione1.tex)

#### 3.1.5.2 Frontespizio

Fornisce delle informazioni generali e di introduzione al documento, ed è così composto:

- logo esteso del gruppo;
- nome del documento;
- nome del gruppo e del progetto$_G$ relativo al capitolato$_G$ scelto;
- indirizzo email di riferimento del gruppo;
- versione del documento, secondo la convenzione;
- stato, in accordo con quanto descritto nel ciclo di vita dei documenti;
- elenco dei redattori;
- elenco dei verificatori;
- nome del responsabile che ha effettuato l'ultima approvazione;
- destinatari del documento;
- breve descrizione.

#### 3.1.5.3 Registro delle modifiche

Raccoglie in forma tabellare tutte le modifiche e conseguentemente la storia del documento, dove ogni riga può essere:

- **modifica:** per modifiche seguite da verifiche, con questa sequenza:
  - versione;
  - descrizione;
  - redattore;
  - data redazione;
  - verificatore;
  - data verifica;
- **approvazione:** per le approvazioni dei documenti da parte del responsabile, queste seguono la struttura:
  - versione;
  - descrizione;
  - data approvazione;
  - approvatore.

Queste strutture possono essere richiamate rispettivamente tramite i comandi:

- `\changelogTable{...}`
- `\approvingTable{...}`

definite nel template LaTeXper i documenti formali. Le date sono sempre scritte secondo lo standard ISO 8601;

#### 3.1.5.4 Indice

Riassume i contenuti del documento raccolti per sezioni ed intestazioni, indicando la pagina di inizio e collegando alla stessa

#### 3.1.5.5 Elenchi delle figure e tabelle

Nel caso in cui un documento presenti una o più figure o tabelle, queste sezioni le indicheranno raccogliendole rispettivamente ed indicando la pagina di apparizione.

#### 3.1.5.6 Sezione di introduzione

La prima sezione di contenuto di ogni documento formale è l'introduzione che si articola in:

- una sezione che descrive lo scopo del documento;
- due sezioni condivise fra tutti i documenti, non verbali, che illustrano rispettivamente scopo del prodotto del capitolato$_G$ scelto e riferimenti ad altri documenti;
- una sezione che contiene tutti i riferimenti, di natura normativa o informativa.

#### 3.1.5.7 Ulteriori sezioni e appendici

Come indicato precedentemente ogni sezione di primo livello deve avere un suo file dedicato e si svilupperà in seguito all'introduzione. Al termine di tutte le sezioni vi può essere un'appendice.

### 3.1.6 Verbali

I verbali possono essere interni, se relativi ad una riunione dei soli membri del gruppo, o esterni, se relativi ad un qualche tipo di incontro con persone esterne al gruppo. Come gli altri documenti formali, hanno un frontespizio ed un registro delle modifiche, poi si articolano in questo modo:

- informazioni generali
  - dettagli sull'incontro
    - luogo, data[15], orari di inizio e fine e partecipanti
  - ordine del giorno;
- verbale della riunione
  - contiene tutte le sottosezioni necessarie a descrivere lo svolgimento dell'incontro;
- conclusioni
  - nel caso di verbale interno, una tabella di tracciamento delle decisioni;
  - nel caso di verbale esterno, un riassunto delle decisioni prese o delle risposte alle domande poste.

Ogni verbale avrà come nome "verbale_[tipo]_[num]" dove tipo può essere interno o esterno, e num e il progressivo rispetto al tipo.

#### 3.1.6.1 Tracciamento delle decisioni nei verbali interni

Ogni riga della tabella di tracciamento delle decisioni si compone di:

- **codice:** identificativo univoco della decisione, così formato: VI_numRiunione.numDecisione (e.g.: VI_2.3 indica la terza decisione presa durante la riunione interna 2);
- **decisione:** breve riassunto che indichi in maniera chiara la decisione presa.

### 3.1.7 Glossario e acronimi

Ogni documento può contenere dei termini ambigui o degli acronimi che necessitano di una definizione, scritta all'interno della wiki$_G$ in GitHub.

#### 3.1.7.1 Definzione ed utilizzo delle voci

Gli acronimi ed i termini di glossario vanno definiti ed inseriti in ordine alfabetico all'interno della pagina web "Glossario" di GitHub, istruzioni precise su come utilizzare correttamente questo documento si trovano nella wiki$_G$ HOW-TO GLOSSARIO. Una volta che una voce è stata aggiunta, la si può usare liberamente in qualsiasi sezione tramite il suo nome, nel caso di termine nel glossario, o abbreviazione nel caso di acronimo.

[15] Standard ISO 8601

#### 3.1.7.2 Funzionamento

Ad ogni azione di $git_G$ $push_G$ su un branch feature, un'automazione GitHub Action si occupa di eseguire uno script $bash_G$, glossary_builder.sh il quale:

- controlla tutti i documenti che hanno almeno un termine definito nella pagina "Glossario" di GitHub;
- aggiunge a tutte le occorrenze delle voci di glossario e acronimi rispettivamente pedice G o A;
- se ci sono stati cambiamenti, ricompila il documento per ottenere un pdf aggiornato;
- se ci sono stati dei documenti aggiornati, effettua $commit_G$ e $git_G$ $push_G$;
- effettua l'upload dei pdf su una cartella condivisa in Google Drive.

Questo permette di avere sempre i file pdf dei documenti aggiornati e con le voci di glossario correttamente compilate, in maniera automatica e rapida. Si consiglia quindi ai membri di effettuare sempre una $git_G$ $pull_G$ prima di apportare nuove modifiche ai documenti se si ha precedentemente eseguito una $git_G$ $push_G$, così da avere il documento aggiornato ed evitare problemi di conflitti.

### 3.1.8 Norme tipografiche

#### 3.1.8.1 Nomi di file e cartelle

Ogni file e cartella dovrà avere un nome:

- che sia il più possibile conciso ed esplicativo;
- composto di sole lettere minuscole, numeri, '_' e '-', ad eccetto del '.' per separare nome da estensione e per suddividere i numeri di versione, se indicata;
- che abbia alla fine un'estensione coerente col suo tipo;
- non contenere spazi o caratteri diversi da quelli indicati sopra, parole diverse si separano con '_'.

#### 3.1.8.2 Stile del testo

Ai seguenti stili si attribuisce una specifica funzione semantica:

- **corsivo:** per denotare termini tecnici appartenenti ad una particolare tecnologia che si sta trattando;
- **grassetto:** per evidenziare termini rilevanti o dei quali viene dato un significato esteso immediatamente in seguito, come questi in un elenco puntato;
- **maiuscoletto:** per indicare altri documenti (e.g.: Piano di Progetto);
- **pedice:** G e A, per indicare rispettivamente una voce di glossario o un acronimo.

#### 3.1.8.3 Elenchi

Ogni elemento deve terminare con un punto e virgola, eccetto quella finale che va seguito da un punto, quindi ogni prima parola deve avere iniziale minuscola a meno che non indichi qualcosa per la quale la capitalizzazione è importante. Gli elenchi puntati non numerati avranno come simbolo di primo livello un puntino pieno, come secondo un trattino, come terzo un asterisco. Gli elenchi numerati come primo livello numeri, come secondo lettere minuscole, come terzo numeri romani rappresentati con lettere minuscole

- elenco puntato non numerato
- secondo elemento
  - secondo livello
    - terzo livello

1. elenco puntato numerato
2. secondo elemento
   (a) secondo livello
      i. terzo livello

(Esempi elenchi puntati numerati e non corretti)

#### 3.1.8.4 Data e ora

In ogni documento si adotta lo Standard ISO 8601 , quindi aaaa-mm-gg (e.g.: 5 Gennaio 2021 = 2021-01-05) e hh:mm. Se si scrive il mese in caratteri questo va con l'iniziale maiuscola e vale lo stesso per i giorni della settimana.

#### 3.1.8.5 Codici di versioni

Si adotta la convenzione $x.y.z$ dove x, y e z sono numeri progressivi che partono da 0, x non si azzera mai, y si azzera ad ogni incremento di x e z si azzera ad ogni incremento di y o x. Gli incrementi avvengono:

- **per x:** quando avviene un accertamento da parte del responsabile di modifiche sostanziali;
- **per y:** quando viene apportato e verificato un insieme discreto di innovazioni;
- **per z:** per ogni modifica verificata minore.

Nel caso si voglia specificare la versione nel nome di un file, per esempio per condivisioni esterne al gruppo, la nomenclatura dovrà essere:

[nome_file]__v[x.y.z].[estensione]

(e.g.: norme_di_progetto__v1.0.3.pdf)

### 3.1.9 Strumenti

Come già citato, per la scrittura dei documenti formali si usa LaTeX[16] e come editor non c'è una preferenza assoluta, ma si consigliano TexStudio o TexMaker. Per quanto riguarda i documenti informali invece, Confluence può essere usato in qualunque browser.

[16] https://www.latex-project.org/

## 3.2 Gestione della configurazione

### 3.2.1 Scopo

Lo scopo del processo di gestione della configurazione è quello di applicare procedure amministrative e tecniche al fine di controllare e registrare modifiche e rilasci ed assicurare la correttezza, consistenza, completezza ed archiviazione di ogni elemento.

### 3.2.2 Aspettative

Avere uno spazio di lavoro versionato, condiviso ed altamente orientato alla collaborazione è fondamentale in un gruppo di lavoro con molteplici componenti. La storicizzazione, tracciabilità e reversibilità di ogni cambiamento permettono di concentrarsi sulla produzione di nuovo valore invece che sulle problematiche di versionamento, conflitti e condivisione.

### 3.2.3 Descrizione

Il gruppo Three Way Milkshake ha un organizzazione su GitHub[17] all'interno della quale gestisce tutte le repo$_A$ di cui necessita.

### 3.2.4 Workflow di versionamento

Il gruppo adotta il workflow gitflow all'interno di ciascuno repo$_A$. Per semplificare le operazioni si può usare il pacchetto di estensioni gitflow. Maggiori informazioni ed istruzioni per l'uso di git$_G$ e GitHub si trovano nella wiki$_G$ GIT, GitHub & Jira

#### 3.2.4.1 Pull requests

Per l'implementazione delle feature dai branch di lavoro, bisogna aprire una pull-request$_G$ su develop. A questo punto il responsabile o l'amministratore provvederanno a fare i controlli necessari ed effettueranno il merge[18] tramite *squash*.

### 3.2.5 Integrazione con Jira

GitHub è stato configurato in maniera che ad ogni commit$_G$ si scatenino delle azioni che coinvolgono Jira. Usando una specifica sintassi[19] [20] è possibile collegare ogni commit$_G$ con una task$_G$ o sottotask di Jira, aggiungendo direttamente su questa commenti, tracciamento del tempo di lavoro ed apportando modifiche allo stato di progresso.

### 3.2.6 Strumenti

Come già detto, le repo$_A$ sono tenute su GitHub, per quanto riguarda la gestione locale, ogni membro è libero di adottare lo strumento che preferisce.

[17] Three Way Milkshake su Github: https://github.com/Three-Way-Milkshake
[18] Il merge in git$_G$: https://www.atlassian.com/git/tutorials/using-branches/git-merge
[19] https://support.atlassian.com/bitbucket-cloud/docs/use-smart-commits/
[20] wiki GIT, GitHub & Jira

## 3.3 Risoluzione dei problemi

### 3.3.1 Scopo

Questo processo ha lo scopo di definire un insieme di procedure atte alla risoluzione dei problemi.

### 3.3.2 Aspettative

Identificazione, segnalazione e risoluzione dei problemi devono avvenire in maniera standardizzata all'interno del gruppo così che tutto il processo possa essere svolto nei tempi più brevi possibili.

### 3.3.3 Descrizione

Quando un verificatore riscontra dei problemi o degli errori, aggiunge dei commenti che iniziano con "TODO", così da facilitare la ricerca poi, che indichino brevemente i problemi individuati e, tramite il meccanismo degli *smart commits*, riporta la task$_G$ o sottotask relativa nello stato *in progress* ed aggiunge un commento riassuntivo. Nel caso sia necessario aggiungere più informazioni, il verificatore si può recare nella board ed aggiungere ulteriori commenti. Il verificatore viene quindi sollevato automaticamente dal compito, il responsabile controlla il problema segnalato, se necessario imposta una priorità diversa da *medium*, aggiunge eventuali commenti o descrizioni e riassegna la task$_G$ ad un altro membro. Il responsabile può sfruttare le sezioni *history* e *comments* di ogni task$_G$ per controllare chi ha lavorato in passato a quel compito di modo da facilitare la scelta di assegnazione.

## 3.4 Accertamento della qualità

### 3.4.1 Scopo

Lo scopo del processo di accertamento della qualità è di garantire che il prodotto e i servizi offerti rispettino gli obiettivi di qualità e che i bisogni del proponente siano soddisfatti.

### 3.4.2 Aspettative

Gli obiettivi che si desidera raggiungere tramite la gestione della qualità sono i seguenti:

- qualità nell'organizzazione e nei suoi processi;
- qualità del prodotto;
- soddisfazione finale sul prodotto e il lavoro svolto da parte del proponente;
- qualità provata oggettivamente.

### 3.4.3 Descrizione

Per trattare approfonditamente la gestione della qualità si fa riferimento all'appendice Standard di Qualità di questo documento e al Piano di Qualifica v4.0.0 dove sono descritte le modalità impiegate per garantire la qualità nello sviluppo del progetto$_G$.
In particolare, in tale documento:

- sono presentati gli standard utilizzati;

- sono individuati i processi di interesse negli standard;
- sono individuati gli attributi del software più significativi per il progetto$_G$.

Per ogni processo vengono descritti:

- gli obiettivi prefissati di qualità;
- le metriche da utilizzare;
- un intervallo, un valore di minimo o un valore di massimo che servono ad identificare uno standard minimo di qualità secondo la specifica metrica.

In questo caso si mira ad ottenere software e documentazione di qualità soddisfacente.

### 3.4.4 Attività

Le tre attività$_G$ principali del processo sono:

- **pianificazione:** porsi degli obiettivi di qualità, definire strategie per raggiungerli e disporre di conseguenza persone e risorse nel modo migliore;
- **valutazione:** mettere in atto la pianificazione, applicando i criteri, misurando e monitorando i risultati;
- **reazione:** sulla base dei risultati, adottare le proprie strategie e criteri.

### 3.4.5 Strumenti

Gli strumenti predefiniti per la qualità sono:

- forniti dallo standard ISO 12207;
- le metriche.

## 3.5 Verifica

### 3.5.1 Scopo

Il processo di verifica ha per scopo la realizzazione di prodotti corretti, funzionali e completi. Sono soggetti a verifica il software ed i documenti.

### 3.5.2 Aspettative

Il corretto svolgimento del processo di verifica rispetta i punti seguenti:

- viene effettuata seguendo le procedure definite;
- vi sono criteri chiari e affidabili da seguire per verificare;
- ogni fase$_G$ del ciclo di vita attraverso cui i prodotti passano deve essere verificata;
- dopo la fase$_G$ di verifica, il prodotto è in uno stato stabile;
- rende possibile la validazione del prodotto.

### 3.5.3 Descrizione

Il processo di verifica prende in input ciò che è già stato prodotto e lo restituisce in uno stato conforme alle aspettative.
Per ottenere tale risultato, il processo si divide in due fasi$_G$: una di analisi e una di test.

### 3.5.4 Attività

#### 3.5.4.1 Analisi
La fase$_G$ di analisi si divide in analisi statica e analisi dinamica.

#### 3.5.4.2 Analisi statica
L'analisi statica è un processo che esegue controlli sulla correttezza di documenti e codice, senza alcuna esecuzione dello stesso.
I metodi di analisi statica verificano la conformità dei prodotti con le regole fissate, tramite metodi manuali oppure metodi automatizzati.

#### 3.5.4.3 Analisi dinamica
L'analisi dinamica richiede l'esecuzione del codice per valutarne la qualità.
Vengono realizzati dei test che verificano la funzionalità dei frammenti di codice analizzati.

#### 3.5.4.4 Test
Per approfondimenti sui test, si fa riferimento al Piano di Qualifica v4.0.0. L'attività di testing è necessaria per assicurarsi del corretto funzionamento del prodotto.
Ogni test verrà indicato nel seguente modo:

**T[Tipo_test][Tipo_Requisito]-[Codice]-[Importanza]**

Dove:

- **Tipo_test:** indica il livello di astrazione a cui va operare il test, può essere:
  - **A:** per i test di accettazione;
  - **I:** per i test di integrazione;
  - **S:** per i test di sistema;
  - **U:** per i test di unità.
- **Tipo_requisito:** indica il tipo di requisito$_G$ a cui il test è associato, può essere:
  - **F:** per i requisiti$_G$ funzionali;
  - **V:** per i requisiti$_G$ di vincolo;
  - **Q:** per i requisiti$_G$ di qualità;
  - **P:** per i requisiti$_G$ prestazionali.
- **Importanza:** indica l'importanza del requisito$_G$, può essere:
  - **O:** per i requisiti$_G$ obbligatori;
  - **D:** per i requisiti$_G$ desiderabili;

– **F:** per i requisiti$_G$ facoltativi.

Dei test ben scritti devono:

- essere ripetibili;
- specificare i valori dei parametri sopraelencati: tipo_test,tipo_requisito,importanza;
- avvertire di esiti inattesi.

#### 3.5.4.5 Tipi di test

Di seguito una breve descrizione dei tipi di test:

- **di sistema:** effettuato sulla totalità del sistema, verificandone la corretta integrazione tra le varie parti e corretto comportamento;
- **di unità:** effettuati sulle più piccole parti analizzabili individualmente all'interno del prodotto;
- **di integrazione:** verificano la corretta collaborazione e integrazione tra le varie parti del sistema;
- **di accettazione:** verificano che il prodotto realizzato nel suo complesso soddisfi i criteri di accettazione decisi con il cliente.

## 3.6 Validazione

### 3.6.1 Scopo

La validazione è necessaria per controllare che il prodotto sia conforme alle pretese del cliente ed ai requisiti$_G$ stabiliti all'inizio del progetto$_G$.

### 3.6.2 Aspettative

L'obiettivo della fase$_G$ di validazione è portare i prodotti ad un livello soddisfacente.

### 3.6.3 Descrizione

Il processo di validazione prende in input ciò che è stato prodotto e si assicura che sia ad un livello atteso.

### 3.6.4 Attività

Anche la fase$_G$ di validazione segue un processo diviso in più fase$_G$:

- identificare i prodotti completati che devono essere validati;
- identificare una strategia di validazione adatta e riutilizzabile;
- validare i documenti in esame con la strategia individuata e valutare gli esiti ottenuti.

# 4 Processi Organizzativi

## 4.1 Gestione dei processi

### 4.1.1 Scopo

Qui lo scopo è la gestione delle attività$_G$ e dei compiti per ogni altro processo.

### 4.1.2 Aspettative

L'organizzazione e la gestione dei compiti all'interno del gruppo di lavoro deve avvenire in modo sistematico$_G$, disciplinato$_G$ e quantificabile$_G$ così da favorire uno svolgimento dei lavori ordinato, che rispetti i tempi e che non sia vulnerabile a variazioni e problemi che si possono certamente riscontrare.

### 4.1.3 Descrizione

Il responsabile e l'amministratore si sono occupati della creazione e della configurazione di tutti gli ambienti necessari:

- account Gmail;
- organizzazione GitHub;
- calendario condiviso su Google Calendar;
- workspace Slack;
- workspace Confluence con spazio dedicato ai verbali e spazio dedicato alle wiki$_G$;
- progetto$_G$ Jira con board condivisa per l'organizzazione, gestione e tracciamento dei compiti;
- organizzazione del cruscotto$_G$ contenete varie informazioni sull'andamento del progetto$_G$, presente su Google Sheets;
- organizzazione di un sito per la visualizzazione del cruscotto$_G$, tramite Google Sites;
- organizzazione di Google Drive.

#### 4.1.3.1 Jira

Il gruppo adotta Jira per la gestione dei compiti, così da avere:

- cruscotto$_G$ di lavoro che mostri lo stato di ogni compito:
  - ogni task$_G$ ha diversi campi oltre al nome (descrizione, data di scadenza, assegnatari, eventuali sottotask), comprende una sezione *history* che raccoglie tutte le modifiche avvenute sulla stessa, una sezione *comments* che raccoglie commenti apportati manualmente o automaticamente importati dagli *smart commits* ed una sezione per il tracciamento del tempo, nel quale si può impostare un tempo stimato e tracciare quello effettivo mano a mano che si prosegue;
- integrazione con Confluence, essendo entrambi prodotti di Atlassian;
- automazioni per azioni con gli *smart commits*;

- automazioni di notifiche su Slack.

Come spiegato nella sezione Integrazione con Jira, è stato configurato un processo automatico di collegamento fra commit$_G$ su GitHub e task$_G$ in Jira. Inoltre ad ogni azione, manuale o automatica, su Jira, viene generata una notifica nel canale dedicato di Slack.

#### 4.1.3.2 Workflow dei compiti

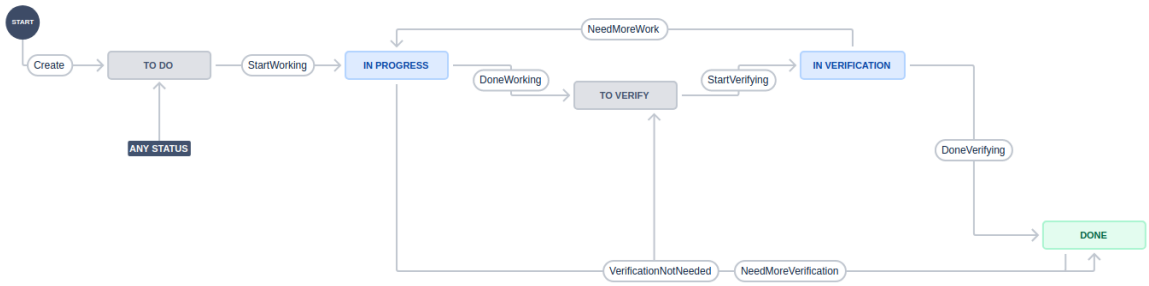


Figura 4.1.1: Diagramma degli stati e delle transizioni del workflow

Come si può vedere dall'immagine, il workflow di task$_G$ e sottotask è stato configurato di modo che ci siano delle precise transizioni da uno stato all'altro e che solo quelle siano permesse. I nomi delle transizioni sulle frecce corrispondono ai comandi da usare nella sintassi degli smart commit$_G$. Maggiori dettagli si trovano nella wiki$_G$ Git, GitHub & Jira.

#### 4.1.3.3 Google Sheets

Si fa uso dei fogli di calcolo Google Sheets per tenere traccia di tutte le informazioni relative alle metriche usate nel Piano di Qualifica. Contiene dati essenziali al controllo dello svolgimento del progetto$_G$, come il tempo per ogni lavoro individuale svolto dal singolo soggetto o il tempo effettivo e preventivato di una specifica riunione. Parte di questo foglio è stato bloccato in maniera tale da favorire l'inserimento dei soli dati da parte dei componenti del gruppo ed evitare modifiche accidentali.

#### 4.1.3.4 Google Sites

Collegato direttamente con i grafici del cruscotto$_G$ presenti in Google Sheets, il sito interattivo creato tramite Google Sites permette di avere una rapida visualizzazione dell'avanzamento del progetto$_G$. Questo sito, diviso in periodi, propone delle "fotografie" ritraenti degli specifici grafici utili a verificare velocemente metriche essenziali del Piano di Qualifica e la loro variazione tra periodi diversi.

#### 4.1.3.5 Google Drive

Viene utilizzato uno spazio Google Drive per contenere tutti i file utili al gruppo, come presentazioni, materiale utile alla comprensione delle tecnologie da utilizzare, un foglio elettronico contenente le varie metriche e diagrammi Draw.io

#### 4.1.3.6 Workflow per l'inizio di una nuova task

1. Si prende il codice della task$_G$ da Jira: PCS-### e si va nel foglio di calcolo *Sincronizzazione Dati* alla riga corrispondente **TEMPO (minuti) PREVENTIVATO** → tempo durata preventivato (relativo al lavoro);
2. in **DATA #StartWorking → #DoneWorking:**
   - Settare date: AVVIO e SCADENZA PREFISSATA;
   - Ogni qualvolta si esegue un commit$_G$ con 'git commit -m "PCS-### [#<eventuale transizione>] #comment <commento> #time <tempo>" ' (quindi usando tracciamento temporale di Jira) e si incrementa nel foglio di calcolo;
3. al termine del lavoro della task$_G$ si dovrà andare ad inserire la data di fine effettiva della colonna **DATA #StartWorking → #DoneWorking FINE EFFETTIVA**;
4. aggiungere il **TEMPO (minuti) EFFETTIVO** con relativo **Ruolo**;

#### 4.1.3.7 Workflow per la verifica di una task

1. Si prende il codice della task$_G$ da Jira: PCS-### e si va nel foglio di calcolo *Sincronizzazione Dati* nella riga corrispondente **TEMPO (minuti) PREVENTIVATO** → tempo durata preventivato (relativo alla verifica);
2. settare **SCADENZA PREFISSATA** in **DATA #StartVerifying → #DoneVerifying**;
3. Al termine della verifica si dovrà andare ad inserire la data di fine effettiva della colonna **DATA #StartVerifying → #DoneVerifying AVVIO (=EFFETTIVA FINE DI #TOVERIFY) FINE EFFETTIVA**.

#### 4.1.3.8 Workflow per le riunioni

Chi dovrà redigere il verbale dovrà:

1. nel foglio *Durata Riunioni - Dati* inserire la data ed i tempi prevista ed effettivi sotto **Durata riunioni interne / esterne**;
2. creare un branch feature/<data-iso>_verbale_<T>_<n>, dove:
   (a) **data-iso** è la data di svolgimento della riunione;
   (b) **T** indica il tipo, che può essere quindi interno o esterno;
   (c) **n** il numero progressivo nella sua categoria;
3. Terminata la stesura, oltre a settare tempi come per le task$_G$ normali, controllare la wiki$_G$ Sequenza Verbali su Confluence e:
   - spuntare la propria box;
   - una volta che la task$_G$ è in toVerify con l'assignee cancellato (si cancella da solo tramite automazione) assegnare a chi tocca la verifica;
4. spostare note su Confluence → Trascritti.

Il verificatore terminato il suo lavoro dovrà aprire una pull-request$_G$ del branch relativo al verbale con develop come base.

## 4.2 I ruoli di progetto

La suddivisione in ruoli è necessaria per favorire la parallelizzazione e distribuzione del lavoro e delle responsabilità.

### 4.2.1 Analista

Ha il compito specifico di comprendere il problema. Ha un ruolo attivo fin quando la comprensione non è adeguata. Adottando un modello incrementale$_G$, rimane più a lungo ed evolve. E auspicabile che ci siano più analisti per portare punti di vista diversi e sommare le conoscenze. Redige Studio di Fattibilità ed Analisi dei Requisiti.

### 4.2.2 Progettista

Traduce il problema in una possibile soluzione, descritta come architettura, divisa in parti per agevolare lo sviluppo individuale di queste. Dati i vincoli ha il compito di trovare una buona soluzione. Formalizza le scelte architetturali tramite diagrammi UML$_A$.

### 4.2.3 Verificatore

Agisce su ogni attività$_G$ ed attua una verifica oggettiva, segnalando eventuali problemi secondo il processo rispettivo.

### 4.2.4 Programmatore

Implementa in codice ciò che i progettisti hanno definito come design. Un'adeguata divisione dei compiti tra diversi programmatori consente un alto parallelismo, ideale per avanzare rapidamente. Deve svolgere compiti piccoli, che siano facilmente verificabili.

### 4.2.5 Responsabile

Gestisce il controllo sul progetto$_G$ e tramite un cruscotto$_G$ di controllo aggiorna, revisiona ed adatta lo svolgimento dei lavori. Elabora piani e scadenze, assegna compiti, approva documenti, gestisce i problemi, redige l'organigramma ed il Piano di Progetto ed approva l'offerta prima di sottoporla al committente.

### 4.2.6 Amministratore

Ha in carico la definizione, modifica ed agevolazione del way of working. È un percorso in logica JiT$_A$ e procede per incrementi. È responsabile dell'efficacia e dell'efficienza nell'ambiente di lavoro, gestisce la documentazione, organizza il glossario, collabora alla redazione del Piano di Progetto e redige le Norme di Progetto.

## 4.3 Gestione delle comunicazioni

Di seguito si definiscono le procedure e gli strumenti adottati per standardizzare ed organizzare le comunicazioni all'interno ed all'esterno del gruppo.

### 4.3.1 Comunicazioni interne

Riguardano i soli componenti del gruppo Three Way Milkshake e in base allo scopo si articolano tramite strumenti diversi.

#### 4.3.1.1 Google Meet

Piattaforma di videoconferenze[21] di Google che permette la condivisione di più schermi e non richiede l'installazione di programmi aggiuntivi in quanto fruibile da browser. Integrata con Google Calendar permette la creazione di meeting collegati ad eventi direttamente da quest'ultimo strumento, semplificando così il processo di creazione ed organizzazione di eventi comuni. Viene utilizzata per le riunioni, le quali devono essere effettuate con una frequenza non inferiore a quella settimanale. Per ogni riunione il gruppo segue l'ordine del giorno che si sarà venuto a formare nel documento Confluence dedicato allo specifico incontro, al quale tutti possono collaborare. Le discussioni e le decisioni verranno raccolte dal segretario, che cambierà ad ogni riunione, per distribuire equamente questo compito a rotazione, nello stesso documento dal quale poi produrrà il verbale seguendo le regole definite in 3.1.6.

#### 4.3.1.2 Slack

Strumento di messaggistica fortemente orientato alla collaborazione in gruppi di lavoro [22]. Il gruppo Three Way Milkshake ha un workspace dedicato nel quale sono presenti diversi canali, per suddividere ed organizzare le comunicazioni. Durante lo svolgimento del progetto$_G$ si possono creare tanti canali quanti se ne rendono necessari, il responsabile provvederà alla gestione di questi. Oltre ai canali dedicati alle attività$_G$ appena citati ci sono i seguenti:

- **generale:** per le comunicazioni che non ricadono in nessun canale specifico già esistente e per le quali non si rende necessaria la creazione di un nuovo canale;
- **capitolato-c5-portacs:** all'interno del quale l'amministratore configura le integrazioni automatizzate utili a notificare i componenti del gruppo:
  - Jira, invierà una notifica ogni qualvolta che si intraprenderà un'azione, manuale o automatica (*smart commits*) sul cruscotto$_G$ di progetto$_G$;
  - GitHub, notificherà ogni azione intrapresa sui branch *master* e *develop*;
  - Google Calendar, che provvederà a notificare la programmazione di eventi e ricorderà quelli imminenti allegando codice Google Meet per accedere direttamente alla riunione.

#### 4.3.1.3 Telegram

Software di messaggistica [23] nel quale il gruppo Three Way Milkshake ha un gruppo dedicato utilizzato solamente per le comunicazioni informali e non strettamente legate al progetto$_G$ di lavoro.

[21] https://meet.google.com/
[22] https://slack.com/intl/en-it/about
[23] https://telegram.org/

### 4.3.2 Comunicazioni esterne

Possono avvenire tra il gruppo Three Way Milkshake ed il proponente e/o con il committente. Il gruppo si adatterà allo strumento preferito da questi in ogni momento nel quale si renda necessario il contatto. Per le comunicazioni rapide e dirette che non richiedono un incontro sincrono fra le parti, fornitore e proponente si sono accordati per l'utilizzo di uno spazio Google Chat[24].

## 4.4 Gestione dei Rischi

### 4.4.1 Descrizione

I rischi previsti o identificati devono essere prontamente documentati nel Piano di Progetto. Ogni rischio dovrà essere così descritto:

- nome;
- **codice**: secondo la classificazione in 4.4.2;
- **occorrenza e pericolosità:** possono assumere i valori bassa, media alta;
- descrizione;
- metodologie di rilevamento;
- piano di contingenza.

### 4.4.2 Classificazione dei Rischi

Per facilitare raccolta e riferimenti successivi, il codice identificativo di ogni rischio deve seguire la seguente convenzione:

RIS_[tipo]-[num]

dove tipo può essere:

- **T** → tecnologico;
- **O** → organizzativo;
- **I** → interpersonale;

mentre num è un numero progressivo che parte da 1 per ogni categoria, ed è univoco all'interno della stessa.

[24] https://workspace.google.com/products/chat/

| | | |
|---|---|---|
| **Nome**: Novità del problema e delle tecnologie | **Codice**: RIS_T - 1 | **Occorrenza**: Alta<br>**Pericolosità**: Media |
| **Descrizione**: Il capitolato non pone vincoli sull'utilizzo delle tecnologie da adottare. Se da un lato questo permette libertà nell'implementazione, dall'altro può causare disorientamento nei membri meno esperti. Vista la novità del problema da trattare, le tecnologie da impiegare potranno risultare nuove per molti. | **Rilevamento**: Il responsabile si occuperà di censire le conoscenze e competenze dei membri del gruppo, al fine di individuare particolari lacune. I membri, qualora dovessero riscontrare difficoltà, lo comunicheranno al resto del gruppo. | **Piano di contingenza**: Dopo un'esplorazione generale delle tecnologie che si prestano a risolvere il problema richiesto, ci si confronterà con il proponente per confermare la bontà delle scelte adottate. I membri che hanno più esperienza guideranno lo studio di queste tecnologie. |

Figura 4.4.1: Esempio corretto di raccolta rischio.

## 4.5 Formazione

I membri del gruppo Three Way Milkshake sono tenuti a formarsi individualmente in quanto si ritiene che i tempi e le modalità di questo processo siano altamente individuali. Per quanto riguarda tecnologie e strumenti di interesse al gruppo, ogni membro è libero di creare una wiki$_G$ nello spazio dedicato di Confluence per raccogliere risorse$_G$ e documentazioni utili, raccolte online o prodotte dallo stesso, così da condividere le proprie conoscenze apprese e velocizzarne l'assorbimento da parte degli altri componenti.
Per i compiti attuali e per quanto discusso fino a questo momento con il proponente, si rimandano i membri del gruppo alle seguenti fonti ufficiali:

- **LaTeX:**
  - https://www.latex-project.org/;
  - https://www.overleaf.com/learn;
- **Github:** https://docs.github.com/en;
- **Confluence:** https://confluence.atlassian.com/alldoc/;
- **Java:** https://docs.oracle.com/en/java/;
- **Nodejs:** https://nodejs.org/en/docs/.

# A Standard di qualità

## A.1 ISO/IEC 12207

ISO/IEC 12207 è uno standard ISO per la gestione del ciclo di vita del software.
Di tutti questi processi, elenchiamo quelli su cui ci siamo concentrati maggiormente.

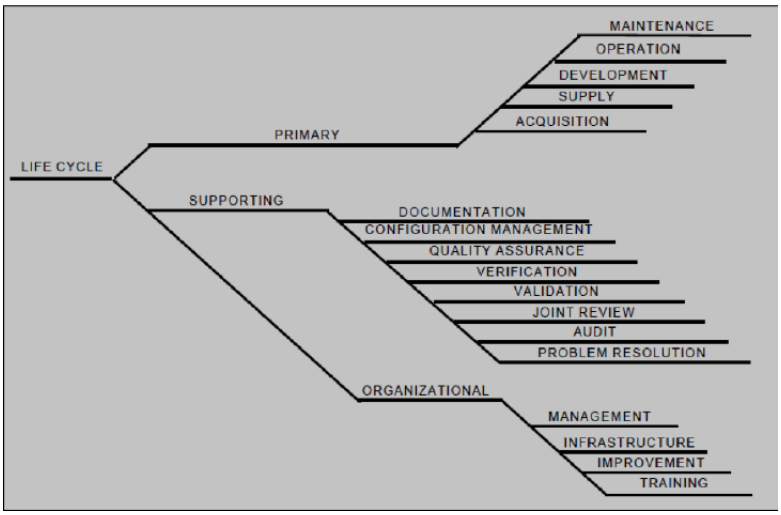


Figura A.1.1: Processi del ciclo di vita del software, secondo lo standard ISO/IEC 25010:2011

- Tra i **processi primari**:
  - Sviluppo;
  - Fornitura.
- Tra i **processi di supporto**:
  - Documentazione;
  - Gestione della configurazione;
  - Accertamento della qualità;
  - Verifica;
  - Validazione;
  - Risoluzione dei problemi.
- Tra i **processi organizzativi**:
  - Gestione (dei processi, comunicazione e rischi);
  - Infrastruttura.

## A.2 ISO/IEC 25010:2011

In questo standard troviamo la parte di qualità del software, sostituisce l'ISO/IEC 9126 dal 2011 in poi.
In particolare aggiunge il modello della qualità in uso del software.

1. **Efficacia:** precisione e completezza con cui gli utenti raggiungono i risultati desiderati.
2. **Efficienza:** risorse spese in relazione agli obiettivi raggiunti (e in relazione all'efficacia).
3. **Soddisfazione:** soddisfazione dell'utente, relativo ai suoi bisogni soddisfatti dal software.
   Solitamente la soddisfazione dipende dal soddisfacimento di *utilità*, *fiducia nel software*, *gradimento* e *comfort*.
4. **Libertà da rischi:** grado con cui il software mitiga i possibili rischi.
   In particolare:
   - mitigazione rischi economici;
   - mitigazione rischi di salute e sicurezza;
   - mitigazione rischi ambientali.
5. **Context coverage:** grado con cui il software può essere usato con efficacia$_G$, efficienza$_G$, soddisfazione e libertà da rischi, in qualunque contesto.

Consultare la sezione ISO/IEC 9126 per altre informazioni sulla qualità del software.

## A.3 ISO/IEC 9126

ISO/IEC 9126 è uno standard internazionale per valutare la qualità del software.
Questo standard fornisce un modello di qualità e 3 tipologie di metriche, queste 4 sezioni vengono riportate di seguito.

### A.3.1 Metriche per la qualità interna

Definisce metriche applicabili al codice sorgente non eseguibile. Idealmente, la qualità interna determina la qualità esterna.
Viene rilevata tramite **analisi statica**.

### A.3.2 Metriche per la qualità esterna

Definisce metriche applicabili al software in esecuzione che ne misurano i comportamenti tramite test. Idealmente, la qualità esterna determina la qualità in uso.
Viene rilevata tramite **analisi dinamica**.

### A.3.3 Metriche per la qualità in uso

Definisce metriche applicabili solo quando il prodotto è finito e utilizzato in condizioni reali.


Figura A.3.1: Figura esplicativa del modello della qualità software esterna ed interna dello standard ISO/IEC 9126

### A.3.4 Modello della qualità del software

1. **Funzionalità:** il software deve fornire funzioni che soddisfino i bisogni emersi nell'Analisi dei Requisiti.
   In particolare il software deve avere le seguenti caratteristiche:
   - Appropriatezza;
   - Accuratezza;
   - Interoperabilità;
   - Sicurezza.
2. **Affidabilità:** il software deve mantenere un certo livello di prestazioni quando utilizzato in condizione specificate.
   In particolare il software deve avere le seguenti caratteristiche:
   - Maturità;
   - Robustezza;
   - Recuperabilità.
3. **Efficienza:** il software deve eseguire le proprie funzioni con minimo tempo e consumo di risorse possibile.
   In particolare efficienza$_G$ nel tempo, con veloci tempi di risposta e nello spazio, con una appropriata quantità di risorse.
4. **Usabilità:** il software deve essere comprensibile e poter essere studiato senza troppe difficoltà.
   In particolare il software deve avere le seguenti caratteristiche:
   - Comprensibilità;
   - Apprendibilità;

- Operabilità;
- Attrattiva.

5. **Manutenibilità:** il software deve potersi evolvere con modifiche, correzioni e adattamenti.
   In particolare il software deve avere le seguenti caratteristiche:
   - Analizzabilità;
   - Modificabilità;
   - Stabilità;
   - Testabilità.
6. **Portabilità:** il software deve poter essere trasferito da un ambiente hardware/software ad un altro seguendo le evoluzioni tecnologiche.
   In particolare il software deve avere le seguenti caratteristiche:
   - Adattabilità;
   - Installabilità;
   - Conformità;
   - Sostituibilità.