DOCUMENTO INTERNO

Carbon7team

carbon7team@gmail.com

16 Maggio 2022

Organizzazione github: Carbon7team

Norme Di Progetto

v3.0.0

Redattori	**Revisori**
Matteo Noro	Filippo Brugnolaro
Andrea Polato	Adnan Latif Gazi
Filippo Brugnolaro	Marco Odinotte

Sommario

Documento Interno relativo alle $NormediProgetto_G$
del Carbon7team

Storico modifiche al documento

Legenda:

- +: Prima redazione di contenuto
- #: Estensione di contenuto
- [n]: Sezione n del documento

Versione	Operazione	Autore	Verificatore	Data
3.0.0	Approvazione documento	Filippo Brugnolaro	Filippo Brugnolaro	2022/05/16
2.1.0	Revisione documento	Adnan Latif Gazi	Adnan Latif Gazi	2022/05/11
2.0.6	+ ManualeUtente [2.1.5] + ManualeDelloSviluppatore [2.2.6]	Filippo Brugnolaro	Adnan Latif Gazi	2022/04/20
2.0.5	# Processi Organizzativi [4.2]	Matteo Noro	Adnan Latif Gazi	2022/04/01
2.0.4	+ Processi Organizzativi [4.2]	Matteo Noro	Adnan Latif Gazi	2022/03/23
2.0.3	# Processi di Supporto [3.1],[3.2],[3.3]	Matteo Noro	Adnan Latif Gazi	2022/03/18
2.0.2	+ Processi di Supporto [3.1],[3.2],[3.3]	Matteo Noro	Adnan Latif Gazi	2022/03/12
2.0.1	# Processi Primari [2.2]	Matteo Noro	Adnan Latif Gazi	2022/03/06
2.0.0	Approvazione Documento	Filippo Brugnolaro	Filippo Brugnolaro	2022/02/23
1.1.0	Revisione Documento	Marco Odinotte	Marco Odinotte	2022/02/16
1.0.1	# Denominazione documenti [3.2.1]	Matteo Noro	Adnan Latif Gazi	2022/02/11
1.0.0	Appprovazione Documento	Filippo Brugnolaro	Filippo Brugnolaro	2022/01/30

Versione	Operazione	Autore	Verificatore	Data
0.3.0	Revisione Documento	Marco Odinotte	Adnan Latif Gazi	2022/01/29
0.2.6	# Processi Primari [2.2]	Andrea Polato	Matteo Noro	2022/01/11
0.2.5	# Processi Primari [2.2] + Sistema di Qualità [5]	Andrea Polato	Matteo Noro	2022/01/06
0.2.4	# Processi di Supporto [3.1] + Introduzione [1]	Matteo Noro	Marco Odinotte	2021/12/22
0.2.3	#Processi Primari [2.1],[2.2] # Processi di Supporto [3.2]	Matteo Noro	Filippo Brugnolaro	2021/12/13
0.2.2	+ Processi Primari [2] # Processi Organizzativi [4.3]	Matteo Noro	Filippo Brugnolaro	2021/11/29
0.2.1	# Processi di Supporto [3.1] # Processi Organizzativi [4.2]	Matteo Noro	Adnan Latif Gazi	2021/11/28
0.2.0	Revisione Documento	Adnan Latif Gazi	Adnan Latif Gazi	2021/11/27
0.1.2	# Processi di Supporto [3.1]	Andrea Polato	Filippo Brugnolaro	2021/11/26
0.1.1	# Processi Organizzativi [4.1],[4.2] # Processi di Supporto [3.2]	Andrea Polato	Filippo Brugnolaro	2021/11/25
0.1.0	Revisione Documento	Filippo Brugnolaro	Filippo Brugnolaro	2021/11/24
0.0.4	# Processi di Supporto [3.1]	Matteo Noro	Adnan Latif Gazi	2021/11/24
0.0.3	+ Processi Organizzativi [4]	Matteo Noro	Adnan Latif Gazi	2021/11/23

Versione	Operazione	Autore	Verificatore	Data
0.0.2	+ Processi di Supporto [3]	Matteo Noro	Filippo Brugnolaro	2021/11/22
0.0.1	Generazione Documento	Matteo Noro	-	2021/11/22

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo di tale documento gestionale Interno è quello di definire procedure, strumenti e metriche di qualità con la funzione di normare il Team.
Tale documento si pone quindi come un "manuale di comportamento", da seguire in modo preciso e rigoroso.
La sua redazione avviene in maniera *Just-In-Time*, per assicurare che il prossimo futuro sia normato.
Bisogna quindi ricercare un miglioramento continuo al fine di garantire la qualità desiderata.
Ogni norma sarà descritta e inserita nel documento previa approvazione dei membri del Team, i quali sono tenuti a restare aggiornati con le future modifiche che saranno apportate al documento, in modo da svolgere i compiti loro assegnati in modo coerente come previsto.
Si vuole precisare come, nonostante nella stesura del documento si facciano riferimento a standard ISO/IEC/IEEE, il Team non pretende in alcun modo di adattarsi a tali standard, in quanto non garante di tale professionalità data l'insperienza. I riferimenti si riducono a pura citazione e riferimento di scopo.
Questo definisce il $WayOfWorking_G$ del Team.

1.2 Glossario

Per assicurare la massima trasparenza e fruibilità del documento, il *Carbon7team* ha deciso di stilare il $Glossario\_2.0.0_G$. Qui verranno inserti tutti i termini ambigui o relativi all'attività del progetto, che il gruppo individua come degni di nota. I termini qui presenti saranno identificati attraverso una 'G' a pedice.

1.3 Riferimenti

1.3.1 Riferimenti normativi

- **Capitolato d'appalto**
- **Documenti Carbon7team**
  - Verbale_interno_carbon7team_13NOV21
  - Verbale_interno_carbon7team_24NOV21
  - Verbale_interno_carbon7team_17DIC21

1.3.2 Riferimenti informativi

- **Altri documenti del capitolato d'appalto**
  - Video capitolato d'appalto;
  - Presentazione capitolato d'appalto.
- **Materiale didattico del corso di Ingegneria del Software**
  - Slide lezione su Analisi dei $Requisiti_G$;
  - Diagrammi $Use\ Case_G$;
  - Libro di testo: Iam Sommerville, Software Engineering
    Part 1: Introduction to Software Engineering; Chapter 4: Requirements.

- **Documentazione esterna**
  - WebRTC;
  - Stati, allarmi e misurazioni dell'UPS.

2 Processi Primari

2.1 Processi di fornitura

2.1.1 Scopo

Secondo lo standard ISO/IEC/IEEE 12207:1995 lo scopo del processo di fornitura è quello di consegnare all'acquirente un prodotto o servizio che soddisfi i requisiti richiesti. Il fornitore determina l'esistenza di un acquirente che necessita di un prodotto o servizio e definisce una strategia di fornitura di quanto richiesto, dopo averne analizzato i rischi e le criticità mediante la redazione di uno Studio di Fattibilità. Il fornitore deve inoltre definire un accordo contrattuale che sancisca i rapporti con il committente$_G$ ed in particolare l'accettazione da parte di quest'ultimo dei requisiti e delle tempistiche di consegna. Si potrà quindi dare avvio alla parte esecutiva stabilendo le procedure e risorse che andranno definite nel *PianoDiProgetto*_3.0.0$_G$. Il processo di fornitura pertanto si compone delle seguenti attività:

- avvio;
- approntamento di risposte alle richieste;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

2.1.2 Descrizione

Tale sezione include le norme che i membri del team dovranno rispettare in tutte le attività di progettazione, sviluppo e consegna del prodotto software al fine di diventare fornitori nei confronti del proponente$_G$ Socomec e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.3 *PianoDiProgetto*$_G$

Deve essere redatto un *PianoDiProgetto*$_G$ da seguire e produrre *Just in Time*$_G$ durante lo svolgimento del progetto stesso.
Tale documento deve contenere:

- **Analisi dei Rischi:** dove vengono analizzati i rischi che potrebbero presentarsi nel corso del progetto e vengono preventivate delle contromisure per mitigare gli stessi. Tali rischi vengono inoltre codificati con un identificativo univoco e un livello di pericolo in una scala da 1 a 5;
- **Pianificazione di Progetto:** dove si descrivono nel dettaglio i periodi in cui è stato preventivamente suddiviso il lavoro. La descrizione avverrà dunque per fasi di sviluppo;

- **Preventivo e consuntivo:** dove viene fornita, in due sezioni separate, una stima del lavoro necessario per ciascuna fase, ottenendo prima un preventivo per il costo totale e fornendo poi un consuntivo di periodo relativo all'andamento della fase secondo quanto preventivato.

2.1.4 $PianoDiQualifica_G$

Il documento ha lo scopo di illustrare in che modo si intende perseguire la qualità all'interno del progetto. Il documento deve essere composto da quattro parti fondamentali:

- Qualità di Processo;
- Qualità di Prodotto;
- Suite di Test$_G$;
- Resoconto e Valutazione della Verifica.

Le due sezioni di Qualità di Processo e Qualità di Prodotto devono a loro volta essere suddivise in più processi, per quanto riguarda la prima sezione, e in più aspetti, per la seconda. Ognuna di queste divisioni è a sua volta così caratterizzata:

- Nome del processo/aspetto;
- Descrizione;
- Elenco delle metriche$_G$ utili alla misurazione;
- Tabella con valori obiettivo.

La sezione di *Resoconto e Valutazione della Verifica* deve essere composta prevalentemente da grafici che dimostrino immediatamente e chiaramente i valori ottenuti.

2.1.5 $ManualeUtente$

Il documento ha lo scopo di aiutare il cliente nell'utilizzo del prodotto.
Compito di redigere tale documento è degli amministratori, i quali sia per Virtual Display che per Remote Support devono:

- definire le specifiche minime;
- definire l'eventuale installazione;
- fornire le istruzioni all'uso;
- fornire le informazioni di contatto per supporto agli applicativi.

2.2 Processi di sviluppo

2.2.1 Scopo

In accordo con quanto scritto nello standard ISO/IEC/IEEE 12207:1995, l'obiettivo del processo di sviluppo è quello di trasformare i requisiti, l'architettura e il design in azioni che permettono la creazione di un prodotto che rispetti i requisiti prestabiliti.

2.2.2 Descrizione

Questo processo definisce le seguenti attività:

- Analisi dei Requisiti;
- Progettazione dell'architettura;
- Codifica.

2.2.3 *AnalisiDeiRequisiti*$_G$

2.2.3.1 Scopo

Il processo di *AnalisiDeiRequisiti*$_G$ è quello di individuare le necessità del proponente$_G$, convertendole in requisiti. Tali requisiti possono essere impliciti o espliciti. Il documento di *AnalisiDeiRequisiti*$_G$ è redatto dagli analisti, i quali devono:

- definire lo scopo del prodotto che si andrà a realizzare;
- definire le funzionalità e i requisiti concordati con il proponente$_G$;
- fornire ai progettisti una base solida da dove partire per applicare le tecnologie scelte;
- fornire ai verificatori i riferimenti per il processo di verifica;
- fornire una stima oraria del lavoro per definire una stima dei costi.

2.2.3.2 Descrizione

Le informazioni sopra vengono tratte dallo studio del capitolato d'appalto, da incontri interni al Team e con il proponente$_G$.

Il documento di *AnalisiDeiRequisiti*$_G$ dovrà essere quindi sviluppato secondo tali sezioni:

- Scopo del Documento;
- Scopo prodotto;
- Attori;
- Casi d'uso$_G$;
- Requisiti:
  - Requisiti funzionali;
  - Requisiti di qualità;
  - Requisiti di vincolo;
  - Requisiti prestazionali;

2.2.3.3 Descrizione dei casi d'uso$_G$

I casi d'uso$_G$ dovranno essere descritti secondo i seguenti punti:

- Immagine * ;
- Attore primario;
- Attore secondario * ;
- Precondizione;
- Postcondizione;
- Scenario principale;
- Inclusioni * ;
- Estensioni * .

Nota: * indica un punto che non è sempre richiesto e pertanto non presente per ogni caso d'uso$_G$.

2.2.3.4 Classificazione dei casi d'uso$_G$

Ogni caso d'uso$_G$ è identificato obbligatoriamente da un codice identificativo univoco.
Il codice identificativo è descritto come segue:

UC[NumeroBase].[NumeroSottoCaso]

dove:

- **NumeroBase**: numero che identifica il caso d'uso$_G$ generico;
- **NumeroSottoCaso**: numero progressivo che identifica i sotto casi a partire dal caso base. Può a sua volta diramarsi in ulteriori sottocasi.

2.2.3.5 Descrizione dei requisiti

Ogni requisito dovrà avere un codice identificativo univoco, corredato da una breve descrizione. Ogni requisito dovrà inoltre palesare i casi d'uso$_G$ di riferimento.
I requisiti dovranno essere descritti in modo tabellare, nella seguente forma:

ID Requisito | Descrizione | Fonti

Pertanto ogni requisito sarà corredato di tali informazioni aggiuntive:

- **Descrizione**: descrizione breve e concisa del requisito;
- **Fonte**: specifica la fonte da cui deriva il requisito:
  - capitolato;
  - verbale interno;
  - verbale esterno;
  - caso d'uso$_G$.

2.2.3.6 Classificazione dei requisiti

Ogni requisito è identificato obbligatoriamente da un codice identificativo univoco.
Il codice identificativo è descritto come segue:

R[Importanza][Tipologia][Codice]

- **Importanza** : indica l'importanza di tale requisito attraverso i seguenti valori:
  - 1: requisito obbligatorio;
  - 2: requisito desiderabile;
  - 3: requisito opzionale.
- **Tipologia**
  - V: requisito di vincolo;
  - F: requisito funzionale;
  - P: requisito prestazionale;
  - Q: requisito di qualità.
- **Codice**: identificatore univoco in forma gerarchica: Requisito Base - Sotto Requisito.

  [NumeroReqBase][NumeroSottoReq]

  dove:
  - **NumeroReqBase**: numero che identifica il requisito base
  - **NumeroSottoReq**: numero progressivo che identifica i sotto requisiti a partire dal caso base. Può a sua volta diramarsi in ulteriori sotto requisiti.

2.2.4 Classificazione dei rischi

Di seguito viene presentata la modalità di identificazione dei rischi.
Il codice identificativo dei rischi è descritto come segue:

R[Tipologia][NumeroBase]

dove:

- **Tipologia**: che descrive se un rischio è di tipo:
  - Tecnologico [T];
  - Organizzativo [O];
- **NumeroBase**: numero progressivo che identifica il rischio in base alla tipologia.

2.2.5 Progettazione dell'architettura

2.2.5.1 Scopo

L'attività di progettazione si basa su quanto svolto nel processo di analisi dei requisiti per definire le caratteristiche del prodotto software in grado di soddisfare i requisiti imposti dal proponente. Tale fase deve garantire la qualità del prodotto sviluppato ed organizzare, ottimizzando l'uso delle risorse, la fase implementativa sezionando il problema in unità di complessità ridotta al fine di facilitare il lavoro di programmazione.

2.2.5.2 Descrizione

Le parti principali sono le seguenti:

- **Technology baseline**: contiene le specifiche della progettazione ad alto livello del prodotto e delle sue componenti, l'elenco dei diagrammi UML che saranno utilizzati per la realizzazione dell'architettura e i test di verifica;
- **Product baseline**: dettaglia ulteriormente l'attività di progettazione, integrando ciò che è riportato nella Technology baseline. Definisce i test necessari alla verifica;
- **Diagrammi UML**: diagrammi che permettono una facile comprensione della soluzione progettuale redatta dai progettisti. Saranno disponibili:
  - diagrammi delle classi;
  - diagrammi dei casi d'uso;
  - diagrammi di sequenza.
- **Tecnologie utilizzate**: devono essere descritte le tecnologie che si prevede di utilizzare nel progetto specificandone pregi e difetti;
- **Design pattern**: devono essere descritti i design pattern utilizzati per realizzare l'architettura corredati da una descrizione, anche grafica, che ne descriva il significato e la struttura;
- **Tracciamento delle componenti**: ogni requisito deve riferirsi al componente che lo soddisfa;
- **Test di integrazione**: l'unione delle parti permette di verificare che ogni componente del sistema funzioni nella maniera voluta;
- **Product baseline**: redatta dal progettista, dovrà includere:
  - **definizione delle classi**: ogni classe dovrà essere descritta in modo breve e coinciso;
  - **tracciamento delle classi**: ogni requisito deve essere tracciato in modo che per ognuno esista una classe che lo soddisfi;
  - **test di unità**: devono essere definiti al fine di verificare che le parti funzionino individualmente nel modo stabilito.

2.2.6 *ManualeDelloSviluppatore*

2.2.6.1 Scopo

Il processo *ManualeDelloSviluppatore* ha il compito di rappresentare l'architettura software del prodotto, in modo tale da poter essere consultata da eventuali manutentori che necessiteranno dunque di sapere come è stato costruito il prodotto e agire dunque di conseguenza sulle eventuali scelte di estensione del prodotto stesso. Il documento *ManualeDelloSviluppatore* è redatto dai progettisti, i quali dovranno:

- definire le tecnologie utilizzate con le corrispondenti versioni ed eventuale breve descrizione;
- definire la configurazione minima per lo sviluppo;
- definire i diagrammi delle classi e i principali diagrammi di sequenza.

2.2.6.2 Descrizione

Il documento sarà composto da 4 parti principali:

- Tecnologie adottate;
- Virtual Display;
- Remote Support.

Le ultime 2 sezioni devono contenere:

- Diagrammi delle classi con descrizione degli eventuali design pattern utilizzati;
- Diagrammi di sequenza principali.

N.B.: nei diagrammi delle classi, tutte le classi appartenenti a **LIBRERIE ESTERNE** devono essere colorate in maniera differente in modo tale da facilitare il riconoscimento di quelle non create dal *Carbon7team*.

2.2.7 Codifica

2.2.7.1 Scopo

Questa sezione ha lo scopo di normare l'effettiva realizzazione del prodotto software. In questo processo si concretizza attraverso la programmazione quanto è stato progettato.

2.2.7.2 Descrizione

Obiettivo dell'attività è la creazione del prodotto software conforme alle aspettative e richieste del proponente. L'adozione di regole specifiche in questa fase è fondamentale per perseguire gli obiettivi di qualità e poter agevolare le successive fasi di manutenzione.

2.2.7.3 Stile di Codifica

Ciascun programmatore è tenuto a rispettare le seguenti norme:

- **indentazione**: i blocchi innestati devono essere correttamente indentati.
- **parentesizzazione**: le parentisi di delimitazione dei costrutti vanno inserite in linea, non al di sotto di essi;
- **scrittura dei metodi**: come definito nello standard Kotlin, se possibile i metodi devono rispettare le seguenti norme:
  - prima lettera del nome minuscolo utilizzando una notazione "*camel case*" per le successive parole;
  - tra il nome del metodo e l'eventuale parentesi di apertura deve essere inserita una singola spaziatura;
- **classi**: i nomi delle classi devono iniziare sempre con una lettera maiuscola;
- **costanti**: i nome delle costanti devono essere tutti in maiuscolo;
- **lingua**: i nomi di variabili, costruttori, metodi, classi e commenti vanno scritti in lingua inglese per una maggiore formalità.

3 Processi di Supporto

3.1 Documentazione

3.1.1 Scopo

In questo capitolo verranno illustrate le regole e gli standard$_G$ che normano la documentazione. Tali regole permetteranno di mantenere uno stile grafico e organizzativo coerente.

3.1.2 Descrizione

Le prossime sezioni illustreranno nei dettagli le norme per la redazione di ogni documento, parte dei prodotti di progetto. Il giusto adeguamento alle seguenti regole permetterà una superiore qualità della documentazione.

3.1.3 Versionamento$_G$

Ogni documento è una successione di date fasi di vita, quali:

1. **Pianificazione**
   Si identificano le necessità del documento, per delinearne il contenuto fondamentale;
2. **Redazione**
   Viene costruito lo scheletro del documento e messo per esteso da un redattore$_G$/autore. La prima stesura cerca di coprire tutte le necessità richieste, fornendo un contenuto quanto più completo;
3. **Revisione**
   Il documento previa redazione, viene revisionato sul lato grammaticale e contenutistico. In caso di errori, questi verranno corretti in una nuova versione del documento;
4. **Approvazione**
   Il documento, valutato dal Responsabile, soddisfa quanto richiesto nella fase corrente: assunto come revisionato, esso è corretto in forma e contenuto ed è pronto per esser valutato dal committente$_G$.

Ogni documento, esclusi i verbali, è soggetto a versionamento$_G$, rappresentato nella forma:

[X].[Y].[Z]

Il versionamento$_G$ è definito come segue:

- **[X]** Tale numero verrà aggiornato ogni qual volta verrà aggiunto, o eventualmente rimosso, contenuto al documento;
- **[Y]** Tale numero verrà aggiornato ogni qual volta verrà effettuata una revisione del documento;
- **[Z]** Tale numero verrà aggiornato ogni qual volta verrà effettuata un'approvazione del documento: in questo modo il documento sarà pronto per essere oggetto di valutazione.

La versione di ogni documento partirà, pertanto, dalla 0.0.1.

Dopo la revisione z torna a 0 e dopo l'approvazione y e z tornano a 0.

3.1.4 Classificazione dei documenti

I documenti prodotti dal Team devono essere classificati come segue:

- **Formali**: Documenti che richiedo la verifica di un revisore e l'approvazione da parte del responsabile di progetto;
- **Informali**: Documenti che permettono ai membri del gruppo di condividere e tener nota delle informazioni circa le decisioni prese.

I documenti formali e informali sono a loro volta suddivisi in:

- **Interni**
- **Esterni**

3.1.4.1 Documenti formali:

I documenti formali interni sono tutti quelli che interessano i membri del Team e che li aiutano nelle scelte e nella programmazione della redazione dei documenti successivi. Tali documenti saranno:

- *NormeDiProgetto*_3.0.0$_G$: si tratta del seguente documento, esso contiene tutte le norme che i membri del Team devono seguire.
- *Glossario*_2.0.0 **dei termini**: contiene un elenco di tutti i termini che ricorrono nei documenti e che necessitano di una definizione esplicita. Nel *Glossario*_2.0.0 sono inseriti inoltre termini di carattere generale riguardanti il progetto, che potrebbero ricorrere sia nei documenti che nel parlato.

Quelli esterni invece sono di interesse per committente$_G$, proponente$_G$ o per manutentori futuri, infatti vengono consegnati loro nell'ultima versione, quindi previa approvazione. Tali documenti saranno:

- *PianoDiProgetto*_3.0.0$_G$: pianificazione delle attività che il Team dovrà svolgere, indicando come avverrà l'utilizzo delle risorse;
- *PianoDiQualifica*_3.0.0$_G$: contiene tutti gli standard$_G$ e le metriche$_G$ da utilizzare per la valutazione della qualità;
- *AnalisiDeiRequisiti*_4.0.0$_G$: vengono descritti tutti i requisiti che il prodotto finale dovrà soddisfare;
- *ManualeDelloSviluppatore*_1.0.0: vengono descritte le tecnologie e l'architettura finale del software;
- *ManualeUtente*_1.0.0: vengono descritti i requisiti minimi, l'installazione e le principali funzionalità del prodotto;
- **Candidatura***: contiene la proposta di candidatura al capitolato scelto dal Team, con motivazioni e impegni.

Nota : * indica che il documento rappresenta un'eccezione in quanto non soggetto a versionamento$_G$

3.1.4.2 Documenti informali:

I documenti che appartengono a questa tipologia sono i **verbali**. Essi non sono soggetti a versionamento$_G$ in quanto non vengono modificati nel tempo.
I verbali **interni** sono di interesse per i componenti del Team e aiutano a ricapitolare le decisioni prese durante i meeting interni.
Quelli **esterni** invece riguardano i meeting con committente$_G$ e proponente$_G$.

3.1.5 Gestione dei documenti nelle Repository

L'organizzazione GitHub$_G$ avrà a disposizione due repository$_G$ distinte per la gestione dei documenti: una pubblica, accessibile da utenti esterni, e una privata accessibile solo dal Team.

Di seguito vengono riportati nomi e contenuti delle repo$_G$ sopracitate:

- `Docs/...` Per documentazione pubblica, solo file `.pdf` e complementari;
- `Docs-Private/...` Per documentazione completa di tutti i file sorgenti LaTeX$_G$, accessibile solo dal Team.

Ogni documento stilato dovrà essere presente nella repository$_G$ `Docs-Private`.
Al contrario, soltanto il singolo PDF aggiornato all'ultima versione dovrà essere presente in `Docs`, sempre rispettando le regole di caricamento e nomenclatura che verranno riportate di seguito.
I redattori, al fine di mantenere la documentazione più aggiornata possibile sulla Repository$_G$, devono aver cura di caricare ogni nuova stesura, con annessa versione.
I revisori possono correggere gli errori minori o grammaticali direttamente nel file LaTeX$_G$, mentre gli errori più consistenti vengono segnalati nel file LaTeX$_G$ sotto forma di commento e vengono corretti dal redattore$_G$ a seguito di discussione con il Team.

3.1.6 Struttura generica cartelle per la Documentazione

La cartella contenente i file dei singoli documenti dovrà essere strutturata come segue:

- Nome cartella: `<nomeDocumento>`;
- Contenuto cartella:
  - file `<nomeDocumento>.tex`;
  - file `<nomeDocumento>.pdf`;
  - cartella `res`, cui contenuto è composto da:
    - cartella con il logo del Team ed eventuali immagini;
    - cartella con varie ed eventuali *
    - copertina nella forma: `copertina_<nomeDocumento>.tex`;
    - storico delle versioni nella forma: `storicoModifiche_<nomeDocumento>.tex`;
    - file di contenuto nella forma: `<sezione/contenuto>_<nomeDocumento>.tex`,

Dove la parte `<sezione/contenuto>` sta a indicare, appunto, a quale sezione o contenuto del documento fa riferimento.

Note:
- * indica che la cartella non è sempre presente in quanto non sempre strettamente necessaria.

3.1.7 Struttura generica documento

In prima pagina si trova sempre **la copertina**, che deve avere i seguenti elementi:

1. Tipologia documento e sua destinazione: verbale/documento, interno/esterno;
2. Logo del Carbon7Team;
3. Info di contatto del Team;
4. Data ultimo aggiornamento;
5. Organizzazione github$_G$ del Team;
6. Titolo del documento;
7. Versione attuale (se soggetto a versionamento$_G$);
8. Elenco degli autori e revisori;
9. Sommario.

Unica eccezione alla regola è il documento di Candidatura, dove a seguito del titolo del documento, non vi saranno né la versione attuale, in quanto non soggetta a versionamento$_G$, né l'Elenco degli autori e revisori. Vi saranno invece i componenti del gruppo e il committente$_G$.

3.1.7.1 Indice e Storico modifiche

In seconda pagina vi sarà l'indice dei contenuti e, se il documento è soggetto a versionamento$_G$, nella pagina seguente deve essere presente uno **Storico delle modifiche al documento** (scrittura, revisione, accettazione) strutturata come segue:
`autore` | `operazione svolta` | `versione` | `data`.

3.1.8 Verbali

I verbali, sia in forma interna che esterna, non sono soggetti al versionamento$_G$, in quanto non vengono modificati nel tempo. Non presentano quindi la versione attuale e i revisori. Non sarà nemmeno presente l'indice, per via delle dimensioni e della struttura del documento.
Quest'ultimi dovranno comunque rispettare sempre le regole strutturali e di nomenclatura già citate, con la convenzione che il sommario consiste nella lista dei membri presenti e assenti al meeting d'interesse.
Ogni verbale presenta il suo **contenuto** nella forma:

1. Ordine del giorno;
2. Contenuto, con struttura questioni - conclusioni;
3. Considerazioni finali.

3.1.9 Glossario

Il *Glossario_*2.0.0 dei termini, data la natura stessa del documento, avrà una struttura uguale alla documentazione prima descritta per quanto riguarda copertina e storico modifiche, ma differirà nel formato del contenuto.
Tale documento per la definizione di un termine ha all'interno un comando apposito invocabile con `\entry{<Termine>}{<Ricorrenza nei documenti>}{<Significato>}`.
Per semplicità, i documenti riportati nelle loro ricorrenza andranno trascritti tramite apposita sigla.

3.1.10 Norme Tipografiche

Le norme tipografiche hanno il fine di normare la stesura tipografica dei documenti, al fine di mantenere consistenza nella scrittura e coesione fra i documenti.

3.1.10.1 Convenzioni per la denominazione e la siglatura

I documenti dovranno sempre essere scritti in $\$mathmode\$$ e citati nella forma:

$$\textbf{[NomeDocumento]\_[Versione]}_G$$

Dove:

- NomeDocumento: nome proprio del documento, con primi caratteri sempre in maiuscolo;
- Versione: versione attuale del documento;
- $_G$: pedice per riferimento al *Glossario_2.0.0* (escluso per quest'ultimo).

Tale convenzione dovrà essere rispecchiata anche nella siglatura di questi, mettendo la iniziale della preposizione in minuscolo, come riportato:

- $AnalisiDeiRequisiti\_4.0.0_G$: $AdR\_4.0.0_G$;
- $PianoDiProgetto\_3.0.0_G$: $PdP\_3.0.0_G$;
- $PianoDiQualifica\_3.0.0_G$: $PdQ\_3.0.0_G$;
- $NormeDiProgetto\_3.0.0_G$: $NdP\_3.0.0_G$.

Tale convenzione di siglatura dovrà essere seguita anche in tutti quei prodotti che la richiedono (per esempio PoC_G per il $\text{Proof of Concept}_G$).

3.1.10.2 Formattazione del testo

Il testo dovrà essere formattato in modo differente in base a quello che vuole rappresentare.
Dove il testo non formattato è privo di particolare significato, le varie formattazioni per il testo dovranno essere:

- **Bold**: per tutti i titoli, sottotitoli, sezioni e varie sottosezioni. Viene utilizzato anche per tutte quelle componenti testuali che richiedono maggiore enfasi;
- `Monospace`: per i blocchi di codice e per tutte quelle parti di testo che vogliono rimandare a cartelle, nomi di file, estensioni e affini. Le parti di codice variabile dovranno essere inserite fra parentesi angolari, quali: `< >`;
- *Italic*: per nomi propri, tecnologie, citazioni e terminologie tecniche.

3.1.10.3 Elementi testuali

I documenti redatti per loro stessa forma sintattica presentano molti elenchi testuali che necessitano di essere normati per mantenere la coerenza tipografica citata nello scopo della sottosezione.
Gli elenchi, che essi siano numerati o meno, dovranno presentare un "**;**" alla fine di ogni item, a meno che non introduca un sotto-elenco con "**:**". Solo nell'ultimo item dovrà vi dovrà essere un "**.**" o una "**,**" in base al contesto.
Il titolo di ogni item, se presente, dovrà essere in **bold**. Per eventuali sotto-elenchi, le modalità rimangono uguali. Un esempio è il seguente:

- **Item 1** Contenuto 1;
  - **Item 1.1** Contenuto 1.1;
- **Item 2** Contenuto 2;
- **Item 3** Contenuto 3.

3.1.10.4 Tabelle

Le tabelle possono variare nello stile in base alla loro locazione. È fondamentale che queste siano fruibili in termini di colore, formattazione, grafica e dimensioni nelle colonne.
Ogni redattore$_G$ dovrà aver particolare cura nel scegliere la quantità di contenuto da inserire ad ogni record, prediligendo uno stile breve e conciso.

3.1.10.5 Diagrammi

I diagrammi UML, di qualsiasi natura, dovranno essere inseriti come immagine.

3.1.11 Strumenti

3.1.11.1 LaTeX

Per la redazione dei documenti si dovrà utilizzare LaTeX. Tale strumento permette il versionamento su Git, una documentazione più formale e strumenti utili alla creazione di questa.

3.1.11.2 Diagrammi

Per la creazione dei diagrammi si utilizzeranno, a discrezione propria, due strumenti:

- StarUML
- draw.io

3.1.11.3 Verifica ortografica

Per la verifica ortografica ci si affiderà al proprio editor di testo, impostato nella corretta lingua, e a strumenti online a propria discrezione se ritenuti affidabili.

3.2 Gestione della configurazione

3.2.1 Scopo

L'obiettivo della sezione è quello di regolare la produzione di documenti e codice sorgente.

3.2.2 Descrizione

Si descrivono gli strumenti utilizzati per la produzione di codice, documentazione e diagrammi. Tale sezione descrive anche le modalità di versionamento.

3.2.3 Versionamento

3.2.3.1 Codice di versione per documenti e software

Il numero di versione di ogni prodotto e di ogni documento è definito nel seguente formato:

[X].[Y].[Z]

dove:

- **[X]** indica il numero di versione approvato dal responsabile di progetto;
- **[Y]** indica il numero di un incremento consistente dopo la precedente approvazione; questo numero parte da 0 e incrementa di 1 ad ogni parte di documento aggiunta e modificata da un redattore e contestualmente verificata da un verificatore, viene azzerato ogni qualvolta incrementa **[X]**;
- **[Z]** indica il numero di un incremento minore.

3.2.3.2 Tecnologie coinvolte

Per il processo di versionamento ci si affiderà al software Git$_G$ offerto come servizio dalla piattaforma GitHub$_G$.

3.2.3.3 Branching

Il branching è una modalità di versionamento che consente di creare una copia di un documento o di un software in una sottosezione del repository.
Questo permette di apportare delle feature senza intaccare il branch master.

Ogni branch dovrà essere nominato come segue:

[Sigle creatore]-[feature]

dove:

- **[Sigle creatore]** è il sigla del creatore del branch, con nome e cognome. Questo serve per tener traccia di chi sta lavorando alla feature;
- **[feature]** è il nome della feature a cui si sta lavorando.

3.3 Gestione di qualità

3.3.1 Scopo

La finalità è quella di garantire la qualità prestabilita di prodotti e processi da sviluppare, rispettando le richieste del proponente.

3.3.2 Descrizione

Per esporre i valori di soglia delle metriche e gli standard da applicare al progetto, è stato introdotto il documento $PianoDiQualifica\_3.0.0_G$. In questa sezione ci proponiamo di illustrare come avviene il processo di gestione della qualità e l'istanziamento di un processo.

3.3.3 Processo gestione della qualità

Il processo di gestione della qualità si articola nelle seguenti attività:

- **studio**: il Team individua l'obiettivo che ogni lavoro deve perseguire, studiando la quantità di risorse che esso richiederebbe;
- **regolamentazione**: vengono stabilite le strategie da adottare per conseguire la qualità prestabilita, organizzando le risorse a disposizione;
- **attuazione**: viene eseguito quanto scelto in precedenza. Questa fase permette di ottenere dei risultati concreti che possiamo verificare nella fase successiva;
- **valutazione**: si verifica se i risultati ottenuti rispettano gli standard precedentemente richiesti.

3.3.4 Classificazione delle metriche

Di seguito viene presentata la modalità di identificazione delle metrice$_G$.
Il codice identificativo delle delle metrice$_G$ è descritto come segue:

M[Tipologia][NumeroBase]

dove:

- **Tipologia**: descrive se la metrica$_G$ si riferisce a un:
  - Processo [PS];
  - Aspetto del prodotto [PT];
- **NumeroBase**: numero progressivo che identifica la metrica$_G$ in base alla tipologia.

Inoltre, i valori obiettivo delle metriche$_G$ devono essere rappresentati in forma tabellare come segue:

Codice Metrica | Valore Accettabile | Valore Ottimale

3.3.5 Denominazione test

I test di unità, integrazione, sistema e collaudo sono identificati con il seguente pattern di codici:

T[X][Y]

dove:

- **[X]** assume i seguenti valori:
  - U per i test di unità;
  - I per i test di integrazione;
  - S per i test di sistema;
  - C per i test di collaudo.
- **[Y]** è un numero intero incrementale che parte da 1, ed è univoco per ogni **[X]**.

3.4 Verifica

3.4.1 Scopo

Il processo di verifica ha come obiettivo la realizzazione di prodotti corretti, completi e coesi secondo delle norme stabilite. Sia il software che la documentazione devono essere sottoposti al processo di verifica.

3.4.2 Descrizione

Con la verifica si cercano e risolvono i possibili difetti presenti all'interno della documentazione e del codice.
L'output è il processo stesso reso conforme alle aspettative. Questo risultato si ottiene seguendo determinati punti:

1. definizione di un criterio di accettazione;
2. definizione delle attività di verifica;
3. definizione e implementazione di test di verifica;
4. correzione di eventuali difetti individuati.

Dopo che la verifica è ultimata è possibile avviare il processo di validazione.

3.4.3 Verifica della documentazione

L'attività verifica per un documento è assegnata al *Verificatore*. I verificatori dovranno eseguire un'analisi accurata del documento assegnatogli con l'obiettivo di:

1. verificare la correttezza grammaticale e la semplicità sintattica;
2. assicurarsi che il documento rispetti tutte le norme tipografiche descritte in questo documento;
3. controllare la struttura del documento;
4. analizzare la pertinenza e la coesione delle sezioni e di tutto il contenuto del documento.

3.4.3.1 Analisi statica

Questo tipo di analisi viene effettuata sul prodotto senza eseguirlo e serve per verificare che non ci siano errori. I due tipi di analisi statica sono:

- **Walkthrough**: consiste nell'analizzare il documento nella sua interezza, ad ampio spettro e con forte senso critico, per trovare i difetti.
- **Inspection**: tecnica che prevede la focalizzazione sui punti in cui si sa che si concentrano gli errori. Questa modalità si deve appoggiare su una lista di errori comuni denominata *Lista di Controllo*.

3.4.4 Verifica del codice

Il codice viene verificato con test automatici e con misurazioni quantificabili specificate all'interno del $PianoDiQualifica\_3.0.0_G$. Il codice quindi deve obbedire a questi criteri:

- deve essere corretto rispetto ai test definiti e alle metriche adottate per quantificare la sua qualità;
- deve derivare dai requisiti richiesti in $AnalisiDeiRequisiti\_4.0.0_G$.

3.4.4.1 Analisi statica

Anche per il codice, come per la documentazione, la prima fase di verifica sarà caratterizzata dall'analisi statica.

3.4.4.2 Analisi dinamica

L'analisi dinamica prevede la verifica del codice mirata a rilevare gli errori mentre questo è in fase di esecuzione. Questa fase è caratterizzata dall'esecuzione di una batteria di test per verificare il corretto funzionamento del codice prodotto.
Ogni test eseguito deve essere:

- **decidibile**: sempre in grado di terminare restituendo un feedback binario (riuscito/ fallito);
- **ripetibile**: dato un certo input e delle precondizioni fissate, deve produrre sempre lo stesso output per ogni prova effettuata.

Ogni test presenta i seguenti parametri:

- **Ambiente**: caratteristiche hardware e software sulle quali viene eseguito il test;
- **Stato iniziale**: lo stato del software al momento dell'avvio del test;
- **Input**: dati in ingresso inseriti;
- **Output**: dati in uscita attesi;
- **Ulteriori istruzioni**: ulteriori informazioni sulla configurazione iniziale, sull'interpretazione dei risultati ottenuti e sugli eventuali effetti collaterali che l'esecuzione può produrre.

Per il formato di output di un test è da preferirsi un file di log che esponga in maniera chiara e immediata i risultati.

3.4.5 Verifica dei requisiti

Anche i requisiti sono sottoposti al processo di verifica. Perché la verifica dia esito positivo è necessario che essi:

- siano coerenti con la loro implementazione;
- abbiano un grado di complessità adeguato rispetto al tempo e alle risorse pianificate per la riuscita del progetto;
- si dimostrino verificabili e quantificabili;
- siano coerenti con gli accordi presi con il proponente, riportati all'interno del documento $AnalisiDeiRequisiti\_4.0.0_G$.

3.4.5.1 Analisi statica

I requisiti devono rispettare le proprietà di:

- **verificabilità**: un requisito deve essere misurabile oggettivamente;
- **codice univoco**: un requisito deve essere identificato da un codice, differente per ogni requisito, in accordo con quanto descritto precedentemente;
- **atomicità**: un requisito non deve essere divisibile.

3.4.6 Test

I test costituiscono l'attività fondamentale dell'analisi dinamica: il loro utilizzo permette di individuare bug e di dimostrare che l'applicativo sviluppato è conforme rispetto a tutti i requisiti concordati nell' *AnalisiDeiRequisiti*_4.0.0$_G$.
Verranno implementati tipi di test descritti di seguito, ognuno con uno scopo diverso e con un oggetto di verifica differente.

3.4.6.1 Test di unità

Consiste nel testare la partizione più piccola di software: l'unità. Il corretto funzionamento di ogni unità deve essere testato prima di procedere alla sua integrazione. Verranno normati e implementati durante il processo di codifica.

3.4.6.2 Test di integrazione

Necessari per testare che le diverse unità si interfaccino correttamente: questo tipo di test è eseguito in maniera ricorsiva: ogni volta che un gruppo di unità esegue in maniera corretta questo viene testato unendolo ad un altro gruppo di unità via via più grande fino ad arrivare al test di sistema. Verranno normati e considerati nel processo di progettazione.

3.4.6.3 Test di sistema

Dopo che tutte le unità sono state testate ed è stata testata la loro integrazione viene eseguito il test dell'intero sistema. In questa fase si verifica che tutte le componenti interagiscano nel modo atteso e che l'applicazione soddisfi ciò che è stato definito nell' *AnalisiDeiRequisiti*_4.0.0$_G$.

3.4.6.4 Test di regressione

Quando una o più unità vengono modificate si esegue questa tipologia di test che serve per accertarsi che le funzionalità precedentemente implementate e testate non siano state danneggiate dai cambiamenti apportati al software.

3.5 Validazione

3.5.1 Scopo

Questo processo assicura che il prodotto rispetti i requisiti e soddisfi le aspettative del proponente.

3.5.2 Descrizione

Per svolgere quanto sopra descritto si usa come input l'output del processo di verifica, restituendolo con la garanzia che soddisfi i requisiti.
Il principale attore è il *Responsabile di Progetto*, il quale ha l'onere di decidere se accettare e quindi approvare il prodotto oppure rigettarlo, indicando quali verifiche sono mancanti.

3.5.3 Procedimento

Per avviare il processo di validazione devono essere presenti questi elementi:

- identificazione degli elementi da validare (documenti e/o codice sorgente);
- adottare una strategia di validazione;

- valutazione dei risultati rispetto alle attese.

Il *Responsabile di Progetto* ha l'onere per questo processo.
Due sono i possibili esiti:

- superamento dei controlli: lo stato del prodotto diviene approvato e ne deriva l'aggiornamento di versione come normato;
- mancato superamento dei controlli: il prodotto torna in una fase di redazione/riscrittura e deve essere corretto, con conseguente verifica prima di una nuova richiesta di validazione.

4 Processi Organizzativi

4.1 Gestione di processo

4.1.1 Scopo

Secondo lo standard ISO-12207:1995 la gestione di processo contiene le attività e i compiti generici che vengono impiegati per gestire i vari processi. Il Responsabile di progetto è responsabile della gestione del prodotto, del progetto, e delle attività o processi ad esso applicabili.

4.1.2 Descrizione

In questo processo discuteremo delle seguenti attività:

- comunicazioni e riunioni tra i membri del gruppo;
- comunicazioni e riunioni con i proponenti e committenti;
- pianificazione di risorse, tempi e costi;
- assegnazione dei ruoli e dei compiti.

4.1.3 Comunicazione interna ed esterna

Vengono illustrati di seguito le modalità di comunicazione interne ed esterne al gruppo.

4.1.3.1 Strumenti di comunicazione interna

Si fa obbligo di utilizzare i seguenti strumenti di comunicazione in base alle necessità riportate:

- Comunicazione generica, discussioni: **Telegram**$_G$, **Discord**$_G$
- Annunci importanti: **WhatsApp**
  Struttura annuncio: `#<argomento>`, *a capo*, `<testo annuncio>`.
  L'importanza dell'argomento è quella di poter recuperare eventuali commenti relativi ad esso in altri canali di comunicazione.

4.1.3.2 Strumenti di comunicazione esterna

La comunicazione con l'azienda Socomec avverrà tramite **GMail** e **Microsoft Teams**

4.1.4 Organizzazione del lavoro

Si utilizza il metodo SCRUM$_G$ per organizzare i periodi lavorativi e impostare gli obiettivi.

- Sprint di 1/2 settimane (corrispondenti al cambio ruoli)
- Piccoli obiettivi semplici da raggiungere
- Daily SCRUM$_G$ della durata di 15 minuti circa.
- Il **Responsabile** assume il ruolo di **SCRUM Master**$_G$, il quale esprime le proprie opinioni in merito all'argomento trattato solo per ultimo, al fine di garantire il rispetto dei tempi stabiliti e concludere la seduta.
- Il piano di lavoro è configurato in modo da operare dal lunedì al venerdì, è **FONDAMENTALE** ritagliarsi il tempo per rispettare quanto pattuito. Solo in caso di necessità reali è possibile assentarsi dal daily SCRUM$_G$.
- Come strumento di supporto si utilizza la kanban$_G$ di GitHub$_G$, adibita a SCRUM$_G$ Board.

4.1.4.1 Incontri

Come specificato sopra, il Team si impegna ad incontrarsi giornalmente per un **Daily SCRUM**$_G$ della durata di 15 minuti, per dare un breve resoconto dei compiti assegnati al Responsabile.
Inoltre, viene concordato **un giorno a settimana** per poter sviluppare un incontro più approfondito e duraturo. Tale incontro servirà per definire gli obiettivi settimanali ed eventualmente per aggiornare i ruoli dei membri del Team.

4.1.5 Assegnazione Ruoli

I ruoli vengono assegnati ai singoli membri del team ruotando. Tali devono essere assegnati per garantire che non vi sia alcuna sovrapposizione in termini di richiesta di risorse.
I ruoli verranno assegnati cercando di attribuire lo stesso quantitativo di ore ad ogni membro.
Al momento di ogni nuova assegnazione si devono considerare le abilità nel ricoprire quel ruolo per ogni singolo componente, in modo da aumentare la produttività.
La parola finale sulle assegnazioni andrà sempre al Responsabile. I ruoli nel progetto vengono descritti di seguito.

4.1.6 Ruoli di Progetto

I ruoli saranno assegnati in modo tale che ogni componente del gruppo ruoti attorno a tali figure professionali. Nel fare ciò bisognerà evitare conflitti di interesse, cioè un componente non potrà verificare un prodotto da lui stesso realizzato. I ruoli che saranno ricoperti nel corso del progetto sono:

- Responsabile
- Amministratore
- Analista
- Progettista
- Programmatore
- Verificatore

Di seguito le funzioni che ogni ruolo ha:

4.1.6.1 Responsabile

Il Responsabile di progetto è un ruolo fondamentale che deve essere ricoperto per l'intera durata del progetto. Il suo compito principale è coordinare il team e rappresentarlo verso l'esterno (Livello Customer).
In particolare questo ruolo comporta:

- responsabilità di scelte e approvazioni;
- responsabilità sulla pianificazione delle attività rispettando le scadenze;
- coordinare i membri del gruppo e i compiti da svolgere;
- controllare, coordinare e relazionarsi verso soggetti esterni;
- avere conoscenze e capacità per saper valutare rischi, scelte, alternative.

4.1.6.2 Amministratore

L'Amministratore di progetto è la figura che ha il controllo sull'ambiente di lavoro, pertanto il suo ruolo comporta:

- amministrare le infrastrutture e servizi di supporto;
- risolvere problemi legati alla gestione dei processi;
- salvaguardare la documentazione di progetto, controllando che venga verificata e corretta;
- controllare il versionamento$_G$ e le configurazioni dei prodotti;
- individuare strumenti che portino ad una maggiore automazione dei processi.

4.1.6.3 Analista

L'Analista si occupa di studiare a pieno il problema comprendendone tutte le caratteristiche. Conosce il dominio del problema e ha esperienza professionale. Questo ruolo è fondamentale nella fase iniziale, in particolare nella stesura dell'*AnalisiDeiRequisiti*$_G$. Tale ruolo ha molta influenza sul successo del progetto
Si deve occupare di:

- studiare il dominio del problema;
- analizzare e definire le richieste, quindi i requisiti del proponente$_G$ (anche ciò che non è stato descritto esplicitamente);
- analizzare dove verrà applicato il prodotto finito, quindi i relativi casi d'uso$_G$;
- redigere l'*AnalisiDeiRequisiti*$_G$.

4.1.6.4 Progettista

Il Progettista ha competenze tecniche e tecnologiche aggiornate. Deve sviluppare una soluzione al problema precedentemente analizzato dagli analisti, soddisfacendone i requisiti individuati.
I suoi compiti sono:

- creare un'architettura che sia coerente e consistente nelle sue parti;

- scegliere una soluzione che sia realizzabile nei costi stabiliti;
- cercare di limitare le dipendenze tra le varie componenti;
- far si che il prodotto possa essere nelle sue parti riusabile.

4.1.6.5 Programmatore

Il Programmatore partecipa alla realizzazione e alla manutenzione del progetto. Ha competenze tecniche ma autonomia e responsabilità circoscritte. È colui che si occupa della codifica, cioè deve implementare l'architettura che gli viene data dal Progettista.
I suoi compiti sono:

- codificare ciò che viene passato dal Progettista, documentando e versionando il tutto per agevolare la manutenzione;
- scrivere il Manuale utente del codice del prodotto.

4.1.6.6 Verificatore

Il Verificatore è un ruolo presente per l'intera durata del progetto. Ha competenze tecniche, esperienza professionale, conoscenza del way of working$_G$ e si occupa, quindi, di controllare che le attività vengano svolte nel rispetto delle norme e della qualità aspettata.
I suoi compiti sono:

- controllare che le attività si siano concluse senza la presenza di errori;
- incaricare chi di dovere di correggere eventuali problemi riscontrati durante lo svolgimento di un'attività;
- redigere la parte di retrospettiva del piano di qualifica$_G$.

4.2 Gestione dell'Infrastruttura

4.2.1 Scopo

Il processo di gestione dell'infrastruttura permette di stabilire e mantenere un modello necessario per qualsiasi altro processo. Essa può includere hardware, software, strumenti, tecniche, standard e strutture per lo sviluppo.

4.2.2 Descrizione

Le prossime sezioni espongono gli strumenti impiegati dal Team per quanto riguarda le attività di coordinamento e pianificazione del progetto. La lista presenterà i software utilizzati per tutte le comunicazioni e le conferenze del gruppo. Per ciascuno strumento viene fornita una breve descrizione e l'utilità all'interno dell'organizzazione dei lavori nel gruppo.

4.2.3 Per il Coordinamento

Di seguito viene presentata una breve descrizione degli strumenti utilizzati.

4.2.3.1 Telegram

Strumento di messaggistica istantanea. Servizio primario e fondamentale per la comunicazione interna del Team.

4.2.3.2 Whatsapp

Strumento di messaggistica istantanea. Usato come canale di comunicazione ulteriore per annunci di rilevante importanza che potrebbero perdersi nel gruppo Telegram.

4.2.3.3 GitHub

GitHub$_G$ è un sito Web e un servizio basato su cloud che aiuta gli sviluppatori a memorizzare e gestire il proprio codice, nonché a tracciarne e controllarne le modifiche. GitHub$_G$ è essenzialmente sviluppato su due principi connessi tra di loro:

- controllo versioni;
- *Git*$_G$.

Il controllo della versione aiuta il Team a tenere traccia e gestire le modifiche al codice di un progetto software. Man mano che un progetto software cresce, il controllo della versione diventa essenziale per il completamento dei requisiti. Il controllo della versione consente agli sviluppatori di lavorare in sicurezza attraverso branch$_{GG}$ e merge$_{GG}$. Il branch$_G$ duplica parte del codice sorgente (chiamato repository$_G$). Ogni componente del gruppo può quindi apportare modifiche in modo sicuro a quella parte del codice senza influire sul resto del progetto. Una volta che lo sviluppatore ottiene che la sua parte di codice funzioni correttamente, può unire nuovamente quel codice nel codice sorgente principale per renderlo ufficiale. Il processo di unione è definito merge$_G$. Git$_G$ è invece un sistema di controllo della versione distribuito: l'intera base di codice e la cronologia sono disponibili sul computer di ogni sviluppatore, il che consente facili branch$_G$ e merge$_G$.

4.2.3.4 Issue Tracking System

L' Issue Tracking System$_G$ di GitHub$_G$ è un ottimo strumento per tenere traccia di attività, miglioramenti e bug per quanto riguarda il progetto. Il gruppo utilizzerà questo sistema per gestire l'avanzamento del software, i compiti assegnati ad ogni componente, le milestone e la verifica del codice.

4.2.4 Per la Pianificazione

Di seguito viene invece esposto lo strumento utilizzato per pianificare le attività da svolgere.

4.2.4.1 GitHub Projects

Per il supporto all'attività di pianificazione del progetto e per la realizzazione di diagrammi è stato deciso di utilizzare il sistema interno di gestione dei progetti offerto da GitHub$_G$. L'utilizzo delle automazioni delle Issue e della kanban risultano sufficienti al coordinamento delle attività da svolgere.

5 Sistema di Qualità

Nelle seguenti sezioni verranno trattati gli elementi che costituiscono il sistema garante della qualità del prodotto e dei processi che lo compongono.

5.1 Obiettivi

Gli obiettivi del sistema di qualità istituito dal gruppo sono:

- Garantire una documentazione chiara, senza ambiguità e di livello;
- Affrontare ogni processo nella maniera più efficiente ed efficace possibile;
- Realizzare un prodotto conforme alle aspettative e piacevole da usare.

5.2 Descrizione

La descrizione di ogni processo e ogni aspetto del prodotto è consultabile tramite il documento $PianoDiQualifica\_3.0.0_G$.

5.3 Spiegazione metriche$_G$ - Qualità di processo

Codice	Nome	Descrizione	Formula
MPS1	Schedule Variance [SchV]	Mostra di quanto ci si è discostati dalla previsione delle tempistiche	$(\frac{h_i}{h_p} - 1) \cdot 100$
MPS2	Budget Variance [ResV]	Mostra di quanto ci si è discostati dalla previsione dei costi	$(\frac{bud_i}{bud_p} - 1) \cdot 100$
MPS3	Requirements Variance [ReqV]	Mostra quanto siano evoluti i requisiti dall'inizio del progetto	$\frac{req_m}{req_t} \cdot 100$
MPS4	Testing Success Rate [TSR]	Mostra la percentuale di test$_G$ eseguiti con successo in rapporto al totale di questi	$\frac{t_p}{t_t} \cdot 100$
MPS5	Quality Level Obtained [QLO]	Mostra il livello di qualità complessivo ottenuto dai test$_G$ precedenti	$\frac{SchV+ResV+ReqV+TSR}{4}$
MPS5	Risks Actualization [QLO]	Mostra i rischi concretizzati complessivo ottenuto dai test$_G$ precedenti	n^o *rischi per sprint*

Continua nella pagina successiva...

Legenda:

- $h_{i/p}$: numero di ore impegnate/previste;
- $bud_{i/p}$: numero di risorse impegnate/previste;
- $req_{m/t}$: numero di requisiti mutati/totali;
- $t_{p/t}$: numero di test$_G$ positivi/totali.

5.4 Spiegazione metriche$_G$ - Qualità di prodotto

Codice	Nome	Descrizione	Formula
MPT1	Orthographic Correctness [OC]	Mostra la percentuale di parole grammaticamente corrette in rapporto al totale	$\frac{pr_c}{pr_t} \cdot 100$
MPT2	Gulpease Index [GuI]	Indica la difficoltà di lettura del testo (più alto = più leggibile)	$89 + \frac{300 \cdot nr_f - 10 \cdot nr_l}{nr_p}$
MPT3	Mandatory Requirements Coverage [MRC]	Mostra la percentuale di requisiti obbligatori soddisfatti	$\frac{mReq_s}{mReq_t} \cdot 100$
MPT4	Optional Requirements Coverage [ORC]	Mostra la percentuale di requisiti opzionali soddisfatti	$\frac{oReq_s}{oReq_t} \cdot 100$
MPT5	Code Failure Rate [CFR]	Indica la percentuale di esecuzioni che terminano prematuramente	$\frac{ex_f}{ex_a} \cdot 100$
MPT6	Newcomer Max Learning Time [NMLT]	Indica la quantità massima di tempo necessaria a un nuovo utente per imparare ad usare l'app	$n^o\, minuti$
MPT7	Functionality Max Depth [FMD]	Indica la quantità massima di click necessaria a raggiungere una funzione	$n^o\, click$

Continua nella pagina successiva...

Codice	Nome	Descrizione	Formula
MPT8	Application Response Time [ART]	Tempo massimo necessario al caricamento di una qualsiasi schermata dell'app	n^o *secondi dal click al completo caricamento*
MPT9	Server Response Time [SRT];	Tempo massimo necessario al server per rispondere ad una richiesta	n^o *secondi dall' invio della richiesta alla ricezione della risposta*
MPT10	UPS Refresh Time [UpsRT]	Tempo massimo necessario all'UPS per inviare i dati aggiornati all'app	n^o *secondi dall' invio della richiesta alla ricezione della risposta*
MPT11	Web Application Refresh Time [WebRT]	Indica il tempo di aggiornamento dei dati dell'UPS visibili dalla web app	$UpsRT + n^o$ *secondi dall'invio dei dati dall'app alla ricezione nella web app*
MPT12	Automated Test Coverage [ATC]	Indica quanto codice è coperto dai test_G automatici	$\frac{cod_{test}}{cod_{tot}} \cdot 100$
MPT13	Application Installability [AppI]	Indica la percentuale di dispositivi su cui è installabile l'app	*valore ricavato dalle statistiche online*
MPT14	Server Installability [SrvI]	Indica la percentuale di dispositivi su cui è installabile il nodo server	*valore ricavato dalle statistiche online*
MPT15	WebApp Installability [WebI]	Indica la percentuale di browser_G compatibili con l'applicazione web	*valore ricavato dalle statistiche online*

Legenda:

- $pr_{c/t}$: numero di parole corrette/totali;
- $nr_{f/l/p/c}$: numero di frasi/lettere/parole/parole complesse;
- $mReq_{s/t}$: numero di requisiti obbligatori soddisfatti/totali;
- $oReq_{s/t}$: numero di requisiti opzionali soddisfatti/totali;
- $ex_{f/a}$: numero di esecuzioni fallimentari/avviate;
- $cod_{test/tot}$: quantità di codice coperto da test$_G$/totale.