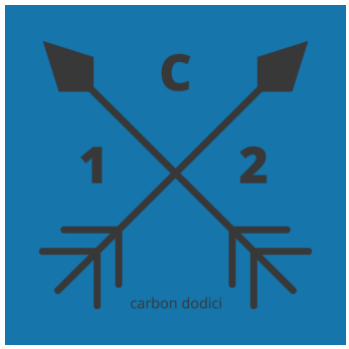

Carbon12 – Predire in Grafana

Norme di Progetto

Informazioni sul documento

Versione	0.2.0
Stato	Approvato
Data di creazione	2019/11/29
Data di approvazione	2020/03/08
Redazione	Nicolò Fassina Francesco Gobbo Alessandro Lovo
Verifica	Giacomo Callegari Manuel De Franceschi Veronica Pederiva
Approvazione	Nicolò Fassina
Uso	Interno
Destinatari	Carbon12 Prof. Tullio Vardanega Prof. Riccardo Cardin

Scopo del documento

Definizione delle norme, del way of working, degli strumenti e delle convenzioni alle quali il gruppo Carbon12 è tenuto ad essere conforme per la realizzazione del progetto Predire in Grafana.

Registro delle modifiche

Versione	Data	Descrizione	Nominativo	Ruolo
0.2.0-0	2020/03/08	Approvazione del documento	Nicolò Fassina	Responsabile
0.1.1-0	2020/03/06	Validazione del documento	Giacomo Callegari	Verificatore
0.1.0-3	2020/02/17	Verifica del documento	Manuel De Franceschi	Verificatore
0.1.0-3	2020/02/15	Aggiornamento sezione §3.2.2 e §2.2.6	Alessandro Lovo	Amministratore
0.1.0-2	2020/02/04	Aggiunta sezioni §2.1.4, §2.1.5.1, §2.1.5.2, §2.1.5.3, §2.1.5.4, §2.1.5.5, §2.1.5.6, §2.1.5.7, §2.1.6, §2.2.6, §3.3	Alessandro Lovo	Amministratore
0.1.0-1	2020/02/01	Riordino struttura documento e aggiunta Appendice A; integrazioni a seguito esito RR alle sezioni §1.1, §2.1.5.4.2, §2.1.5.4.2, §3.5.4.2, §4	Andrea Longo	Amministratore
0.0.1-0	2020/01/13	Approvazione del documento	Giacomo Callegari	Responsabile
0.0.0-16	2020/01/10	Verifica del documento	Veronica Pederiva	Verificatore
0.0.0-16	2020/01/08	Integrazione sezione §2.1.4 e §3.3.3	Nicolò Fassina	Amministratore
0.0.0-15	2020/01/08	Rettifica indicazione parole da glossario	Francesco Gobbo	Amministratore
0.0.0-14	2020/01/07	Integrazione alla sezione §4.5 e §3.1.6; correzioni di forma al documento	Francesco Gobbo	Amministratore
0.0.0-13	2019/12/28	Verifica del documento	Veronica Pederiva	Verificatore
0.0.0-13	2019/12/27	Redazione della sezione 3.5 – Processi di Supporto, Validazione	Alessandro Lovo	Amministratore
0.0.0-12	2019/12/24	Redazione della sezione 3.4 – Processi di Supporto, Verifica	Alessandro Lovo	Amministratore
0.0.0-11	2019/12/22	Redazione della sezione 3.3 – Processi di Supporto, Gestione della Qualità	Alessandro Lovo	Amministratore
0.0.0-10	2019/12/20	Correzioni ai contenuti dei documenti indicati nella sezione §2.1; redazione della sezione 3.2 – Processi di Supporto, Gestione della Configurazione	Alessandro Lovo	Amministratore
0.0.0-9	2019/12/19	Verifica del documento	Giacomo Callegari	Verificatore
0.0.0-9	2019/12/17	Redazione della sezione 2.2.3 – Processi Primari, Sviluppo, Codifica	Nicolò Fassina	Amministratore

Versione	Data	Descrizione	Nominativo	Ruolo
0.0.0-8	2019/12/15	Redazione della sezione 2.2.2 – Processi Primari, Sviluppo, Progettazione	Nicolò Fassina	Amministratore
0.0.0-7	2019/12/12	Redazione della sezione 2.2.1 – Processi Primari, Sviluppo, Analisi dei Requisiti	Nicolò Fassina	Amministratore
0.0.0-6	2019/12/11	Correzioni alla struttura e integrazioni degli strumenti indicati nelle sezioni §3.4.5 e §4.5	Nicolò Fassina	Amministratore
0.0.0-5	2019/12/10	Verifica del documento	Giacomo Callegari	Verificatore
0.0.0-5	2019/12/08	Redazione della sezione 2.1 – Processi Primari, Fornitura	Nicolò Fassina	Amministratore
0.0.0-4	2019/12/07	Redazione della sezione 3.1 – Processi di Supporto, Documentazione	Francesco Gobbo	Amministratore
0.0.0-3	2019/12/05	Redazione del capitolo 4 – Processi organizzativi	Francesco Gobbo	Amministratore
0.0.0-2	2019/12/04	Redazione struttura documento e capitolo 1 – Introduzione	Francesco Gobbo	Amministratore
0.0.0-1	2019/11/29	Creazione del documento	Andrea Longo	Responsabile

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento ha l'obiettivo di formalizzare il WAY OF WORKINGGI che il gruppo intende seguire durante l'intero svolgimento del progetto. Ogni membro del gruppo dovrà attenersi alle regole, alle norme e agli strumenti descritti al fine di perseguire una collaborazione EFFICIENTEGI ed EFFICACEGI che permetta di ottenere documenti conformi e facilmente verificabili.

1.2 Scopo del prodotto

Lo scopo del prodotto è realizzare un PLUG-INGI per lo strumento GRAFANAGI che monitori, tramite un'analisi predittiva, un flusso di dati. Il plug-in deve essere accompagnato da un programma per l'addestramento degli algoritmi SUPPORT-VECTOR MACHINEGI e REGRESSIONE LINEAREGI in grado di creare un file in formato JSONGI contenente i parametri per le previsioni. Il plug-in deve essere scritto in JavaScript e deve essere in grado di leggere dal file in formato JSON la definizione dell'algoritmo di previsione da applicare al flusso di dati ed eseguire i calcoli previsti, producendo dei valori che verranno aggiunti al flusso di monitoraggio, proprio come se fossero stati realmente misurati. I risultati ottenuti devono essere visualizzati in una DASHBOARDGI contenente i grafici prodotti dal sistema di creazione di grafici di Grafana.

1.3 Documenti complementari

Nel documento sono presenti sia termini tecnici che termini il cui significato è ambiguo e determinato dal contesto. Per evitare incomprensioni del contenuto esposto è presente un documento denominato *Glossario Interno v.0.1.0*, destinato ai soli membri del gruppo. I termini da glossario sono contrassegnati con il pedice GI e in maiuscoletto.

1.4 Riferimenti

1.4.1 Normativi

- Standard ISO/IEC 12207:1995:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
- Capitoloato d'appalto 4 - Predire in Grafana:
  https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C4.pdf
- Approfondimenti tecnologici sul capitolato d'appalto 4 - Predire in Grafana:
  https://www.math.unipd.it/~tullio/IS-1/2019/Dispense/C4a.pdf

1.4.2 Informativi

- Microsoft Teams:
  https://teams.microsoft.com/start
- Microsoft Word:
  https://products.office.com/it-it/word
- Astah:
  http://astah.net/
- UMLet
  https://www.umlet.com
- Gantt Project:

https://www.ganttproject.biz/
- GitHub:
  https://github.com/
- Gmail:
  https://mail.google.com
- Google Drive:
  https://www.google.com/drive/
- Telegram:
  https://telegram.org
- Slide del corso di Ingegneria del Software:
  https://www.math.unipd.it/~tullio/IS-1/2019/
- Product Baseline:
  https://www.dau.edu/acquipedia/pages/articledetails.aspx#!363
- JsDoc:
  https://jsdoc.app/index.html
- Airbnb JavaScript Style Guide:
  https://airbnb.io/javascript/
- Skype
  https://www.skype.com/it/

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura è di determinare le procedure e le risorse necessarie allo svolgimento del progetto.

2.1.2 Aspettative

Riteniamo essenziale mantenere una comunicazione costante con il proponente, per ottenere importanti riscontri sul lavoro svolto e scambiare idee ed opinioni volte al miglioramento del prodotto stesso.

2.1.3 Descrizione

Il processo di fornitura contiene le attività svolte dal fornitore. Lo standard *ISO/IEC 12207:1995* suddivide questo processo in sette attività che il fornitore necessita di attuare e sono riassunte nei seguenti punti:

1. Valutazione della proposta;
2. Presentazione dell'offerta;
3. Contrattazione;
4. Progettazione;
5. Realizzazione e verifica;
6. Revisione e validazione;
7. Rilascio e assistenza (come da contratto).

In questa sezione vengono riportate le norme che il gruppo Carbon12 intende perseguire per concorrere alla realizzazione del progetto *Predire in Grafana*, diventando così un fornitore per il proponente Zucchetti SPA.

2.1.4 Rapporti con il proponente

La comunicazione con il proponente Zucchetti SPA è fondamentale per qualunque tipo di aggiornamento rilevante, come ad esempio una baseline. In particolare, avere un feedback da parte del cliente ci è molto utile ai fini del proseguimento del progetto. Per ricevere questo feedback e procedere ad uno scambio di idee e opinioni, sarà possibile fissare un incontro programmato con il gruppo. In alternativa, nel caso di brevi domande volte al chiarimento di piccoli dubbi, sarà possibile comunicare attraverso uno scambio di e-mail tramite l'account ufficiale del gruppo.

2.1.5 Attività

2.1.5.1 Valutazione della proposta

Il fornitore valuta la richiesta del proponente tenendo conto delle politiche organizzative e delle altre normative. Il fornitore deciderà se accettare o meno quanto proposto.

2.1.5.1.1 Studio di Fattibilità

In questa attività il Responsabile di progetto ha il compito di riunire i membri del gruppo per discutere dei vari capitolati d'appalto proposti e i relativi approfondimenti tecnologici, per valutarne la fattibilità. Sulla base della discussione, gli Analisti procedono ad analizzare dettagliatamente i

capitolati andando a stendere un documento comune denominato *Studio di Fattibilità*. Questo documento contiene l'analisi prodotta dagli Analisti per ciascun capitolato d'appalto ed è suddiviso nei seguenti punti:

- **Informazioni generali:** identificazione dei proponenti e committenti;
- **Descrizione:** breve introduzione generale dell'oggetto d'appalto;
- **Finalità del progetto:** descrizione dello scopo del capitolato d'appalto;
- **Tecnologie interessate:** elencazione della strumentazione necessaria e delle tecnologie richieste per la realizzazione del progetto;
- **Aspetti positivi:** descrizione degli aspetti di interesse ritenuti utili sul piano della formazione personale;
- **Rischi:** descrizione degli aspetti ritenuti di ostacolo alla realizzazione del progetto;
- **Conclusioni:** valutazione derivante dalla considerazione dei punti precedenti.

2.1.5.2 Presentazione dell'offerta

Il fornitore definisce e prepara una proposta in risposta alle richieste del proponente.

2.1.5.3 Contrattazione

Il fornitore negozia e stipula un contratto con il committente per fornire il prodotto software richiesto.

2.1.5.4 Pianificazione

Il fornitore procede con la pianificazione della gestione del progetto in base ai requisiti e alle risorse necessarie.

2.1.5.4.1 Documentazione

Vengono di seguito riportati i documenti relativi alle attività che il gruppo Carbon12 intende fornire al proponente Zucchetti SPA e ai committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin:

- *Piano di Progetto;*
- *Analisi dei Requisiti;*
- *Piano di Qualifica.*

2.1.5.4.2 Piano di Progetto

Il documento denominato *Piano di Progetto* viene redatto dal Responsabile e contiene:

- **Analisi dei rischi:** vengono valutati dettagliatamente i rischi riscontrabili durante il progetto;
- **Modello di sviluppo:** viene definito il MODELLO DI SVILUPPOGI adottato;
- **Project schedule:** vengono pianificate le attività da svolgere durante il progetto;
- **Preventivo:** viene fatta la stima dei costi per ogni fase;
- **Consuntivo di periodo:** vengono rendicontati i costi di ogni fase terminata;
- **Preventivo a finire:** viene stimato il costo finale;
- **Attualizzazione dei rischi:** vengono descritti i metodi utilizzati per il contenimento dei vari rischi riscontrati**.**

2.1.5.4.3 Piano di Qualifica

Il documento denominato *Piano di Qualifica* viene redatto dall'Amministratore, per quanto riguarda la definizione di piani e procedure di gestione della qualità, dai Progettisti per quanto concerne la

parte programmatica del documento ed inoltre i Verificatori sono responsabili della parte retrospettiva del documento. Contiene:

- **Qualità del software:** vengono specificati obiettivi numerici e calcolo delle metriche descritte nella sezione 2.2.5.4 del presente documento, per misurare la qualità del software;
- **Qualità di processo:** vengono specificati obiettivi numerici e calcolo delle metriche descritte nei vari processi del presente documento, per misurare la qualità dei processi;
- **Resoconto attività di verifica:** vengono rendicontate le attività di verifica svolte per la Revisione dei Requisiti sulla base delle metriche utilizzate;
- **Valutazioni per il miglioramento:** vengono valutate le problematiche riscontrate durante le varie attività stilando una lista di possibili soluzioni per migliorare il WAY OF WORKINGGI.

2.1.5.5 Realizzazione e Verifica

Il fornitore deve attuare ed eseguire i piani definiti durante la fase di pianificazione e sviluppare il prodotto software perseguendo la qualità desiderata.

2.1.5.6 Revisione e Validazione

Il fornitore deve eseguire la validazione al fine di dimostrare al proponente che il prodotto software soddisfa i requisiti richiesti.

2.1.5.7 Rilascio e Assistenza

Il fornitore rilascia al cliente il prodotto software, come specificato da contratto.

2.1.6 Strumenti

2.1.6.1 Microsoft Excel

Software contenuto nella suite di Microsoft Office per la realizzazione di fogli elettronici. Utilizzato per fare calcoli e creare tabelle e grafici.

2.1.6.2 Gantt Project

Strumento open source per la realizzazione dei DIAGRAMMI DI GANTTGI.
Reperibile all'indirizzo: https://www.ganttproject.biz/

2.2 Sviluppo

2.2.1 Scopo

Lo scopo del processo di sviluppo è quello di definire le attività ed i compiti per produrre il software richiesto dal proponente.

2.2.2 Aspettative

Le aspettative sono di individuare con precisione gli obiettivi da perseguire e i vincoli da rispettare per realizzare un prodotto finale capace di soddisfare i requisiti e le richieste del proponente.

2.2.3 Descrizione

Le attività principali che il gruppo Carbon12 svolgerà, per produrre il software richiesto dal proponente, fanno parte dello standard *ISO/IEC 12207:1995* e sono:

- Analisi dei Requisiti;

- Progettazione;
- Codifica.

2.2.4 Attività

2.2.4.1 Analisi dei Requisiti

Il documento *Analisi dei Requisiti* viene redatto dagli Analisti e contiene:

- **Requisiti:** tutti i bisogni da soddisfare o i vincoli da rispettare imposti dai capitolati d'appalto, dagli approfondimenti tecnologici e dai casi d'uso;
- **Casi d'uso:** tutte le situazioni nella quale il sistema viene utilizzato per raggiungere gli obiettivi dell'utente, individuate dal capitolato d'appalto e dagli approfondimenti tecnologici.

Lo scopo del documento è di esporre i requisiti e i casi d'uso in modo semplice e comprensibile affinché i Progettisti e i Verificatori abbiano riferimenti precisi per le loro attività.

2.2.4.1.1 Tracciamento dei requisiti

È compito dei Verificatori controllare che ad ogni requisito corrispondano una o più fonti.

2.2.4.1.2 Classificazione dei requisiti

Ogni requisito individuato dagli Analisti è accompagnato dalla descrizione del suo scopo, dalle fonti che lo specificano e dalla sua utilità strategica, e viene codificato come segue:

R[categoria][utilità strategica][ID]

dove:

- **R:** indica che si sta definendo un requisito;
- **[categoria]:** indica la tipologia di requisito e può essere:
  - **F:** per indicare un requisito funzionale. Fanno parte di questa categoria i requisiti che descrivono i servizi che il sistema deve fornire. Vengono verificati tramite test, revisione o dimostrazione.
  - **P:** per indicare un requisito prestazionale. Fanno parte di questa categoria i requisiti che descrivono il grado di prestazione che deve raggiungere il sistema in certe situazioni. Vengono verificati tramite misurazione.
  - **Q:** per indicare un requisito qualitativo. Fanno parte di questa categoria i requisiti che descrivono le qualità del prodotto. Vengono verificati tramite tecniche apposite.
  - **D:** indica un requisito dichiarativo. Fanno parte di questa categoria i requisiti che descrivono i vincoli imposti dal cliente o dal problema per obblighi contrattuali oppure per necessità realizzative. Vengono verificati tramite revisione.
- **[utilità strategica]:** indica il grado di importanza del requisito e può essere:
  - **O:** per indicare un requisito Obbligatorio;
  - **D:** per indicare un requisito Desiderabile;
  - **F:** per indicare un requisito Facoltativo (Opzionale).
- **ID:** indica il codice identificativo del requisito e consiste di uno o più numeri separati da punti, secondo un ordine gerarchico.

2.2.4.1.3 Classificazione dei casi d'uso

Ogni caso d'uso individuato dagli Analisti viene rappresentato come segue:

- **Codifica:** codice univoco nella forma **UC[ID]-[Nome]**, dove:

    - **UC:** indica che si sta definendo uno Use Case (caso d'uso);
    - **[ID]:** indica il codice identificativo del caso d'uso e consiste di uno o più numeri separati da punti, secondo un ordine gerarchico;
    - **[Nome]:** indica l'argomento del caso d'uso.
- **Attori:** ruoli assunti dagli utenti o da altre entità. Si suddividono per importanza in:
    - **Attori primari**;
    - **Attori secondari (ove presenti)**.
- **Descrizione:** breve spiegazione del caso d'uso;
- **Precondizioni:** vincoli che devono essere veri prima che il caso d'uso possa essere eseguito;
- **Postcondizioni:** condizioni che devono essere vere al termine dell'esecuzione del caso d'uso;
- **Scenario principale:** sequenza di passi che descrivono l'interazione tra uno o più attori e il sistema;
- **Estensioni (ove presenti):** il comportamento di un caso d'uso viene esteso, aggiungendo obiettivi e passi, in un altro caso d'uso;
- **Inclusioni (ove presenti):** un caso d'uso può includere passi di un altro caso d'uso;
- **Eccezioni (ove presenti):** descrizione di uno o più scenari alternativi allo scenario principale;
- **Diagramma dei casi d'uso:** rappresentazione grafica dei punti precedenti. Verrà inserito a discrezione degli Analisti.

2.2.4.2 Sviluppo PoC - Proof of Concept

Viene fatta la codifica di un prototipo che è una baseline per il successivo sviluppo del prodotto da realizzare. Tramite l'esecuzione di questo prototipo, denominato PROOF OF CONCEPT (POC)GI, viene fatta la dimostrazione di come la tecnologia selezionata sia effettivamente adeguata rispetto agli obiettivi di progetto. Il PoC utilizzato verrà condiviso al committente tramite il REPOSITORYGI pubblico di GitHub: https://github.com/carbondodici/Technology_Baseline

2.2.4.3 Progettazione

Questa attività è svolta dai Progettisti e ha lo scopo di individuare una soluzione che permetta di soddisfare, nel modo più EFFICIENTEGI ed EFFICACEGI possibile, tutti i requisiti definiti dagli Analisti nell'*Analisi dei Requisiti*, adempiendo così alle richieste degli STAKEHOLDERGI imposte da contratto. L'attività di progettazione deve inoltre:

- Garantire qualità nel soddisfare bisogni e vincoli;
- Suddividere il problema in parti meno complesse;
- Definire l'architettura software del prodotto che sia:
    - **Sufficiente:** in grado di soddisfare tutti i requisiti;
    - **Comprensibile:** chiara a tutti gli stakeholder;
    - **Modulare:** suddivisa in parti distinte il più possibile indipendenti;
    - **Robusta:** in grado di comportarsi in modo ragionevole anche in situazioni impreviste;
    - **Flessibile:** sono permesse modifiche adattive ed evolutive ai requisiti;
    - **Efficiente:** nell'uso delle risorse;
    - **Affidabile:** alto grado di probabilità di successo quando utilizzata;

- **Disponibile:** almeno una parte del sistema deve essere disponibile durante la manutenzione;
- **Sicura:** rispetto a malfunzionamenti e intrusioni;
- **Semplice:** solo lo stretto necessario;
- **Incapsulabile:** la rappresentazione interna non è visibile all'esterno;
- **Coesa:** negli obiettivi da raggiungere.

La progettazione passa attraverso due fasi atte a definire l'architettura software del prodotto che sono: progettazione architetturale e progettazione di dettaglio.

2.2.4.3.1 Progettazione architetturale

In questa fase viene redatta la TECHNOLOGY BASELINEGI che descrive come il software viene suddiviso e organizzato in componenti evidenziando le tecnologie, i FRAMEWORKGI e le librerie utilizzate per lo sviluppo del prodotto. Costituisce il prototipo software del prodotto. La *Technology Baseline* verrà consegnata alla *Revisione di Progettazione* (RP).

2.2.4.3.2 Progettazione di dettaglio

In questa fase viene redatta la PRODUCT BASELINEGI che descrive dettagliatamente il comportamento delle componenti riportate nella fase di progettazione architetturale. Viene fornito un allegato tecnico contenente:

- Diagrammi UML 2.0;
- Contestualizzazione dei DESIGN PATTERNGI adottati.

La *Product Baseline* verrà consegnata alla *Revisione di Qualifica* (RQ).

2.2.4.3.2.1 Diagrammi UML

All'interno dell'allegato tecnico vengono forniti i seguenti diagrammi UML 2.0:

- **Diagrammi delle classi:** descrivono il tipo degli oggetti che compongono il sistema e le relazioni statiche esistenti tra loro;
- **Diagrammi di sequenza:** descrivono uno scenario in cui tutte le scelte sono state definite;
- **Diagrammi delle attività:** descrivono processi a cui prendono parte diversi oggetti.

2.2.4.3.2.2 Design Pattern

All'interno dell'allegato tecnico vengono descritti i DESIGN PATTERNGI utilizzati per realizzare l'architettura software del prodotto. Ogni design pattern deve essere accompagnato da:

- **Nome:** riferimento mnemonico di una o due parole che permette di identificare il problema e la soluzione;
- **Problema:** descrizione del contesto di applicazione del pattern;
- **Soluzione:** descrizione degli elementi fondamentali che costituiscono la soluzione e le relazioni che intercorrono tra questi;
- **Conseguenza:** descrizione dei risultati e dei vincoli che derivano dall'applicazione del pattern;
- **Diagramma:** rappresentazione della struttura del design pattern.

2.2.4.4 Codifica

Questa attività è svolta dai Programmatori e ha lo scopo di realizzare il prodotto software richiesto andando ad attuare la soluzione individuata. La codifica prodotta deve attenersi a quanto stabilito nella *Product Baseline* redatta dai Progettisti nella fase progettazione di dettaglio e deve rispettare le metriche relative alla qualità del software descritte nella sezione 2.2.5.4 del presente documento.

2.2.4.4.1 Versionamento

Per tracciare le versioni del codice viene utilizzata la stessa convenzione adottata per la documentazione, riportata nella sezione 3.2.2 del presente documento.

2.2.4.4.2 Nomi dei file

I nomi dei file sorgenti saranno brevi, concisi e scritti in minuscolo. Potranno contenere solo lettere separate al più da uno o più '_' non consecutivi. Il carattere '_' non potrà essere posto all'inizio o alla fine del nome.

2.2.4.4.3 Intestazione

Ogni file sorgente avrà la seguente intestazione:

```
/**
* File name: [nome completo del file]
* Date: [YYYY-MM-DD]
*
* @file   [descrizione del file]
* Authors:
*    @author [Nome Cognome <carbon.dodici@gmail.com>]
* @version [X.Y.Z]
*
* Changelog: [descrizione delle modifiche effettuate]
*/
```

2.2.4.4.4 Stile

Al fine di ottenere file sorgenti uniformi ogni membro del gruppo è tenuto a seguire le norme contenute nell'AIRBNB JAVASCRIPT STYLE GUIDEGI, che delineano come il codice dovrebbe essere scritto e organizzato. In aggiunta a quanto riportato in questo standard, la codifica dovrà rispettare le seguenti norme:

- **Pre e post condizioni:** per ogni funzione verranno indicate pre e post condizioni, rispettivamente prima e dopo la funzione stessa, tramite commento;
- **Ricorsione:** non verranno utilizzate funzioni ricorsive;
- **Parametri delle funzioni:** le funzioni potranno avere al più 4 parametri. Nel caso fosse necessaria una funzione con più di 4 parametri occorrerà suddividerla in due o più funzioni più piccole;
- **Complessità delle funzioni:** le funzioni devono essere brevi e devono svolgere un unico compito specificato dal loro nome;
- **Commenti:** possono essere aggiunti commenti per facilitare la comprensione del codice;
- **Tag:** in aggiunta ai tag utilizzati nell'intestazione, il programmatore potrà, ove ritenuto opportuno, usare:
    - `@example`: per fornire un esempio;

- `@see` *`elemento`*: per fornire un riferimento a qualcosa interno al file;
- `@see {@link` *`percorso_o_Url`*`}`: per fornire un riferimento a qualcosa esterno al file;
- `@todo`: per segnalare che una certa attività non è stata ancora iniziata;
- `@todo` *`descrizione`*: per segnalare cosa non è stato ancora completato di una certa attività.

L'adozione di uno standard per la codifica comporta l'aumento di:

- Coerenza;
- Leggibilità;
- Prevedibilità;
- EFFICIENZAGI del codice.

2.2.5 Metriche

2.2.5.1 Metriche di valutazione: Analisi dei Requisiti

- **Percentuale requisiti obbligatori soddisfatti (PROS):**
  - **Scopo**: garantire il soddisfacimento dei requisiti obbligatori concordati;
  - **Risultato**: percentuale di requisiti obbligatori soddisfatti.
- **Percentuale requisiti opzionali e desiderabili soddisfatti (PRODS):** I requisiti opzionali possono essere soddisfatti solo se prima del termine del progetto sono ancora disponibili risorse adeguate in termini di tempo per persona.

2.2.5.2 Metriche di valutazione: Progettazione architetturale

- **Structural Fan-in (SFIN):**
  - **Scopo**: è indice di utilità e indica quante componenti chiamano uno specifico modulo. Un alto valore indica un alto riuso della componente. È bene, per quanto possibile e ragionevole, cercare di massimizzare tale valore.
  - **Risultato**: valore intero dato dal conteggio delle componenti.
- **Structural Fan-out (SFOUT):**
  - **Scopo**: è indice di dipendenza e indica quante componenti vengono utilizzate da uno specifico modulo. Un alto valore indica un alto accoppiamento della componente, che non è autosufficiente, e una maggiore complessità della logica di controllo necessaria per coordinare i componenti chiamati. È bene cercare di minimizzare tale valore, per agevolare il riuso di componenti indipendenti.
  - **Risultato**: valore intero dato dal conteggio delle componenti.

2.2.5.3 Metriche di valutazione: Progettazione di dettaglio

- **Coupling between objects (CBO):**
  - **Scopo**: conteggiare il numero di collaborazioni di una classe. Le classi sono accoppiate quando i metodi in una classe usano metodi o variabili di istanza definiti in un'altra classe. Un valore elevato è indice di scarsa possibilità di riutilizzo del codice, minore modularità, maggiori difficoltà per quanto riguarda la manutenzione e i test e una maggiore complessità generale. L'obiettivo è quello di mantenere per questa metrica un valore basso.
  - **Risultato**: valore intero dato dall'accoppiamento delle classi.

2.2.5.4 Metriche di valutazione: Qualità del software

2.2.5.4.1 Funzionalità

- **Funzionalità sviluppate (FS):**
  - **Scopo**: misurare quanti requisiti sono stati implementati;
  - **Risultato**: percentuale di requisiti implementati sul totale di requisiti.
- **Correttezza funzionale (CF):**
  - **Scopo**: misurare la correttezza dei risultati, o effetti, prodotti dalle funzionalità implementate;
  - **Risultato**: percentuale dei risultati, o effetti, corretti ottenuti utilizzando le funzionalità del prodotto software.

2.2.5.4.2 Affidabilità

- **Densità degli errori (DE):**
  - **Scopo**: misurare l'attitudine del prodotto software a evitare errori o risultati anomali;
  - **Risultato**: percentuale dei test falliti sul totale di test eseguiti.
- **Capacità di recupero (CR):**
  - **Scopo**: misurare l'attitudine del prodotto software a mantenere i dati dopo errori o malfunzionamenti;
  - **Risultato**: percentuale di quante volte ha avuto successo il recupero totale delle informazioni dopo errori o malfunzionamenti.

2.2.5.4.3 Efficienza

- **Tempo medio di risposta (TMR):**
  - **Scopo**: misurare i tempi delle elaborazioni;
  - **Risultato**: tempo medio impiegato dal prodotto software per rispondere a una richiesta dell'utente o per svolgere un'attività di sistema.
- **Consumo medio di potenza elaborativa (CMPE):**
  - **Scopo**: misurare l'utilizzo di CPU per compiere una elaborazione;
  - **Risultato**: tempo medio di CPU utilizzato per compiere una elaborazione.
- **Consumo medio di memoria (CMM):**
  - **Scopo**: misurare l'utilizzo di memoria per compiere una elaborazione;
  - **Risultato**: quantità media di memoria utilizzata per compiere una elaborazione.

2.2.5.4.4 Usabilità

- **Ambiguità delle funzioni (AF):**
  - **Scopo**: misurare la qualità espositiva delle descrizioni delle funzioni presenti nella documentazione del prodotto software;
  - **Risultato**: numero di funzioni non comprese dall'utente dopo aver letto la documentazione.
- **Ambiguità degli errori (AE):**
  - **Scopo**: misurare la qualità espositiva delle descrizioni degli errori stampati a video;
  - **Risultato**: numero di errori stampati a video non compresi dall'utente.
- **Descrizione delle funzioni (DF):**

    - **Scopo**: misurare il livello di comprensibilità delle funzionalità offerte dal prodotto software;
    - **Risultato**: percentuale di funzioni descritte nella documentazione del prodotto software sul totale delle funzioni presenti.
- **Tempo medio di apprendimento (TMA):**
    - **Scopo**: misurare il tempo mediamente necessario all'utente per imparare ad usare il prodotto software;
    - **Risultato**: tempo impiegato dall'utente per prendere familiarità con le funzioni presenti nel prodotto software.
- **Estetica dell'interfaccia (EI):**
    - **Scopo**: misura del grado di attrattività dell'interfaccia del prodotto software;
    - **Risultato**: valutazione in decimi dell'estetica dell'interfaccia del prodotto software.

2.2.5.4.5 Manutenibilità

- **Presenza di commenti (PC):**
    - **Scopo**: misurare la facilità di comprensione del codice al fine di semplificarne l'analisi;
    - **Risultato**: percentuale di righe di commento sul totale di righe di codice.
- **Complessità ciclomatica (CC):**
    - **Scopo**: misurare la complessità strutturale del codice al fine di semplificare la modifica. Viene definita in riferimento ad un grafo contenente i blocchi base del programma, con un arco tra due blocchi se il controllo può passare dal primo al secondo;
    - **Risultato**: numero massimo di cammini linearmente indipendenti attraverso il codice sorgente.
- **Impatto negativo delle modifiche (INM):**
    - **Scopo**: misurare il comportamento del prodotto software dopo le modifiche effettuate;
    - **Risultato**: percentuale delle modifiche che hanno introdotto errori sul totale di modifiche effettuate.
- **Completamento dei test (CT):**
    - **Scopo**: misurare la completezza dei test implementati;
    - **Risultato**: percentuale dei test che sono stati implementati rispetto a quelli necessari per una copertura completa del codice. Questa metrica viene ulteriormente analizzata nella sezione §3.2.3.3 del *Piano di Qualifica*.

2.2.5.4.6 Portabilità

- **Browser supportati (BS):**
    - **Scopo**: misurare il livello di adattamento dell'applicativo web di ADDESTRAMENTOGE andando ad individuare i browser e le relative versioni che ne supportano l'esecuzione senza perdita di funzionalità;
    - **Risultato**: lista di coppie (nome, versione minima) dei browser supportati.

2.2.6 Strumenti

2.2.6.1 ESLint

ESLint è uno strumento open-source per garantire la coerenza e la qualità del codice JavaScript secondo regole di codifica predeterminate e personalizzabili.
Reperibile all'indirizzo: https://eslint.org/

2.2.6.2 Visual Studio Code

Visual Studio Code viene utilizzato per la codifica di codice JavaScript. Si è scelto di lavorare con questo IDE per la sua compatibilità con Linux, Windows, macOS, sistemi operativi utilizzati dai membri del team, ed in particolare per l'integrazione che esso offre con altri strumenti impiegati nel progetto, come Git ed ESlint.
Reperibile all'indirizzo: https://code.visualstudio.com/

2.2.6.3 UMLet

UMLet è uno strumento utilizzato per la creazione dei diagrammi UML_{GI.}
Reperibile all'indirizzo: https://www.umlet.com/

2.2.6.4 Node.js

Framework JavaScript progettato per creare applicazioni di rete scalabili.
Reperibile all'indirizzo: https://nodejs.org/

2.2.6.5 Angular

Framework open source per lo sviluppo di applicazioni e pagine web.
Reperibile all'indirizzo: https://angular.io/

2.2.6.6 NPM

NPM è un gestore di pacchetti in grado di scaricare e installare in completa autonomia i pacchetti necessari per lo sviluppo e l'esecuzione dell'applicativo.
Reperibile all'indirizzo: https://www.npmjs.com/

2.2.6.7 Webpack

Webpack è un pacchetto Node, installabile quindi tramite NPM, in grado di gestire le dipendenze di un'applicazione JavaScript andando a creare un pacchetto di assets utilizzabile direttamente nel browser a partire da un insieme di file sorgenti.
Reperibile all'indirizzo: https://webpack.js.org/

2.2.6.8 Travis

Servizio di integrazione continua utilizzato per testare progetti software ospitati su GitHub.
Reperibile all'indirizzo: https://travis-ci.org/

2.2.6.9 GitKraken

GitKraken è un client Git che offre un'intuitiva interfaccia grafica per rendere più semplice e veloce lavorare su repository condivise.
Reperibile all'indirizzo: https://www.gitkraken.com/

2.2.7.0 EJS

Template engine che permette di eseguire codice e accedere alle variabili del server dai documenti HTML.
Reperibile all'indirizzo: https://ejs.co/

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

La Documentazione ha l'obiettivo di convalidare i processi e le attività di sviluppo del progetto.
Questa sezione ha lo scopo di definire le norme da rispettare per lo sviluppo dei documenti prodotti durante il CICLO DI VITAGI del software, per delinearle il gruppo si è basato sullo standard *ISO/IEC 12207:1995* sezione 6.1. In questa sezione verranno definite le scelte fatte riguardo la struttura, la verifica e la validazione della documentazione.
I documenti sono consultabili accedendo al REPOSITORYGI:

https://github.com/carbondodici/Predire_in_Grafana

3.1.2 Aspettative

Durante lo sviluppo di questo processo ci si aspetta:

- La definizione di una struttura ben delineata su cui basarsi per la redazione dei documenti durante i processi di sviluppo;
- L'identificazione di norme che permettano una stesura corretta di documenti validi.

3.1.3 Implementazione

3.1.3.1 Ciclo di vita del documento

Ogni documento che viene stilato durante lo sviluppo del progetto si ritroverà in uno ed uno solo dei seguenti stati:

1. **Creazione documento**: ogni documento viene creato e strutturato seguendo le norme specificate in questo documento e adeguandosi a documenti precedenti della stessa tipologia. Alla creazione viene stilato un indice che definisce gli argomenti trattati nel documento in questione e all'approvazione del documento verrà inserita la tabella delle versioni.
2. **Redazione**: nel documento saranno sviluppato i punti indicati nell'indice. Saranno sviluppati dai Redattori assegnati dal Responsabile di Progetto. I Relatori avranno poi l'obbligo di correggere le eventuali segnalazioni riportate dai Verificatori e successivamente riconsegnare ai Verificatori la nuova versione del documento.
3. **Revisione**: il processo di revisione viene svolto dai Verificatori assegnati dal Responsabile, questi hanno il compito di riportare con opportune segnalazioni le eventuali correzioni da apportare al documento in merito alla qualità, conformità e correttezza.
   Se dovessero riscontrare sezioni da rielaborare, il documento con le segnalazioni viene riassegnato ai Redattori, altrimenti passa allo stato di Approvazione.
   I Verificatori e i Redattori per avere una valutazione più critica non possono essere le stesse persone.
4. **Approvazione**: il Responsabile di Progetto sancisce che il documento è giunto allo stato finale, perciò può procedere al rilascio, dopo aver inserito lo storico delle versioni come ultimo capitolo del documento.

3.1.3.2 Struttura dei documenti

Ogni documento è sviluppato seguendo le norme sulla struttura specificate in questa sezione. Per uno sviluppo dei documenti più agevole si sono creati 3 file template in *Word*, salvati come modelli con

estensione *.docx*, da cui partire per lo sviluppo dell'impaginazione dei documenti interni ed esterni e per i verbali.

3.1.3.2.1 Prima pagina

Il frontespizio, la prima pagina di un documento, è strutturato nel seguente modo:

- **Logo**: logo del team, posto al centro e ben visibile;
- **Gruppo e Progetto**: nome del gruppo seguito dal titolo del capitolato scelto (*Predire in Grafana*) posto al centro esattamente sotto al logo del gruppo;
- **Titolo del documento**: nella posizione centrale del documento troviamo il nome del documento;
- **Tabella dati generali**: seguente al titolo del documento e posta centralmente. Essa sviluppa le seguenti voci:
  - **Versione**: ultima versione del documento;
  - **Stato**: stato in cui si trova il documento;
  - **Data di creazione**: data in cui viene creato il documento;
  - **Data di approvazione**: data dell'ultima approvazione;
  - **Redazione**: nome e cognome dei membri del gruppo assegnati alla redazione del documento;
  - **Verifica**: nome e cognome dei membri del gruppo assegnati alla verifica del documento;
  - **Approvazione**: nome e cognome del membro del gruppo (Responsabile) che ha approvato il documento;
  - **Uso**: tipo di uso del documento. Esso può essere di due sole tipologie:
    - **Interno**: riservato ai membri del gruppo
    - **Esterno**: destinato ai membri del gruppo, al committente e proponente
  - **Destinatari**: nominativi dei destinatari del documento;
  - **E-mail di riferimento**: inserita solo nei documenti di uso esterno. È l'indirizzo di posta elettronica di riferimento, per contattare il gruppo.
- **Scopo del documento**: descrizione sintetica dell'obiettivo del documento, posto centrale sotto la tabella dei dati generali.

3.1.3.2.2 Indice

Ogni documento presenta un indice numerico che elenca i capitoli e le sezioni degli argomenti trattati in merito al contenuto del documento. Ciò deve permettere una consultazione del documento più agevole sugli argomenti che vengono trattati, visionando così la struttura macroscopica di cui è composto.

3.1.3.2.3 Contenuto

Le pagine del contenuto sono così strutturate:

- In alto a destra nell'intestazione [Nome del Documento]|[Nome del Gruppo];
- Il contenuto della pagina è sviluppato tra l'intestazione e il piè di pagina;
- In basso a destra è posto il numero della pagina corrente, tale numerazione parte da dove inizia effettivamente il documento.

3.1.3.2.4 Storico delle versioni

Sezione conclusiva del documento in cui viene disposto lo storico delle versioni, in rispetto delle norme del ISSUE TRACKING SYSTEMGI (specificato nella sezione 3.1.5.2 di questo documento).
Ogni documento sviluppato dispone di un CHANGELOGGI: tabella contenente lo storico delle modifiche apportate al documento. In esso vengono specificate:

- **Versione**: numero identificativo della versione del documento;
- **Data**: in formato YYYY/MM/GG;
- **Descrizione**: breve descrizione dei cambiamenti e modifiche al documento;
- **Nominativo**: nome e cognome del membro che ha apportato modifiche o notazioni di modifica al documento;
- **Ruolo**: a quale ruolo è assegnato il modificatore del documento.

Questo viene inserito appena il documento viene approvato e il Responsabile lo inserisce in testa al documento, prima dell'indice.

3.1.3.2.5 Note a piè di pagina

In caso si dovessero esplicitare parole o argomenti che non vengono trattati sui glossari, si inseriscono delle note a piè pagina, in basso a sinistra. Nel corpo del documento le parole sono esplicitate tramite un numero come apice a fianco della parola e testo scritto in formato *italic*.

3.1.4 Design

3.1.4.1 Norme tipografiche

3.1.4.1.1 Convenzioni sui nomi dei File

I nomi assegnati ai file (estensione esclusa) rispettano le seguenti regole, tratte dalla convenzione SNAKE CASEGI, e convenzioni usate nel mondo del lavoro:

1. I nomi dei file saranno scritti in minuscolo;
2. Le parole che compongono il nome saranno separate dal simbolo underscore;
3. Le preposizioni non vengono omesse;
4. I caratteri accentati sono sostituiti dalla parola senza l'accento;
5. Non si deve far uso di abbreviazioni o sigle;
6. Il carattere "v" viene usato per indicare la versione a cui si è arrivati, seguito da un "." e l'indice effettivo del versionamento.

3.1.4.1.2 Glossario

Si è stabilito di sviluppare due Glossari: *Glossario Interno*, specifico per i *Documenti Interni*, e il *Glossario Esterno*, in riferimento ai termini presenti nei *Documenti Esterni*.
Le regole per la definizione dei termini da inserire nei Glossari sono le seguenti:

1. Ogni termine da inserire nei glossari viene marcato con una sigla maiuscola come pedice alla fine della parola e può essere di due tipologie:
    - **GI**: per indicare il termine da inserire nel *Glossario Interno*;
    - **GE**: per indicare il termine da inserire nel *Glossario Esterno*.
2. I termini verranno scritti in formato maiuscoletto;
3. Se un termine dovesse essere presente più volte all'interno del documento, non è necessario marcarlo tutte le volte dalla sigla del glossario a cui si riferisce, essa viene inserita almeno la prima volta che viene usato il termine;

4. Ogni termine marcato dai pedici GI e GE viene inserito nel *Glossario* specifico, dando una spiegazione del significato che lo descriva nel contesto d'uso;
5. I termini di glossario non sono indicati né con il formato né con la marcatura a pedice se inseriti nelle didascalie di Immagini, Tabelle e Diagrammi UML$_{GI}$ e nemmeno negli indici.

3.1.4.1.3 Stile del testo

- **Carattere del testo**: verrà usato il font *Times New Roman;*
- **Carattere di parole/pezzi di codice**: verrà usato il font *Courier New;*
- **Bold**: usato per evidenziare termini di un elenco definizione-spiegazione o elenco di termini; inoltre, viene usato per i titoli dei capitoli e delle sezioni e per il titolo del documento;
- **Italic**: stile utilizzato per indicare nomi di documenti, termini che non necessitano di una spiegazione nel *Glossario* e che, se ritenuto necessario, vengono specificati da apposite note a piè di pagina;
- **Maiuscoletto**: usato per evidenziare termini da inserire nel Glossario;
- **Nomi dei documenti:**
    - Per la citazione di documenti in generale, i nomi vengono scritti con la lettera iniziale maiuscola, senza specificare la versione a cui si riferisce. Essi verranno scritti con formato *italic*;
    - Se invece si deve far riferimento ad una specifica versione di un documento, bisogna aggiungere l'indice della versione (Nome Documento v *.0.0);
    - Il riferimento a sezioni di documenti rispetterà la prima regola, ma non in formato *italic*.

3.1.4.1.4 Elenchi

Gli elenchi usati possono essere sia numerici che puntati, a seconda della decisione del Redattore:

- Il carattere iniziale della prima parola deve essere maiuscolo;
- Alla fine di ogni elemento dell'elenco si inserisce il ";";
- L'elenco termina con ".";
- I sottoelenchi rispettano le stesse regole, perché possiedono una funzione analoga;
- Se gli elenchi hanno lo scopo di dare una definizione o spiegazione di un termine, il termine, posto all'inizio, deve essere in bold.

3.1.4.1.5 Formati comuni

Le date sono scritte nel formato conforme allo standard *ISO 8601*:

YYYY-MM-DD

- **YYYY**: le quattro cifre dell'anno
- **MM**: le due cifre del mese
- **DD**: le due cifre del giorno

Per quanto riguarda il formato delle ore:

HH:MM

- **HH**: le due cifre dell'ora, da 0 a 23
- **MM**: le due cifre dei minuti, da 0 a 59

3.1.4.1.6 Sigle

Le sigle utilizzabili nella documentazione sono le seguenti:

- Documenti Esterni:
  - **AdR - Analisi dei Requisiti**: attività preliminare allo sviluppo di un prodotto software, che definisce i requisiti del capitolato e le caratteristiche del software;
  - **MU - Manuale Utente**: manuale creato per gli utilizzatori del software;
  - **MS - Manuale Sviluppatore**: manuale creato per gli sviluppatori del software;
  - **PdP - Piano di Progetto**: riguarda la gestione del progetto in merito all'EFFICIENZAGI della gestione del progetto e le criticità che presenta;
  - **PdQ - Piano di Qualifica**: documento destinato alla qualità del software e dei processi, specificando gli obiettivi che si vogliono raggiungere e mediante l'uso di quali strumenti e processi.
- Documenti Interni:
  - **GI - Glossario Interno**: raccolta dei termini che il team ritiene necessitino di un'opportuna descrizione per la chiarezza del significato;
  - **GE - Glossario Esterno**: raccolta di termini con relativa spiegazione, destinati a tutti i membri del team e ai destinatari esterni;
  - **NdP - Norme di Progetto**: riferimento per i membri del gruppo alle norme stabilite per lo svolgimento del progetto;
  - **SdF - Studio di Fattibilità**: descrizione dei capitolati proposti e le motivazioni per la scelta o l'esclusione.
- Verbali:
  - **VI - Verbale Interno**: riepiloga ciò di cui si è trattato durante le riunioni interne del team;
  - **VE - Verbale Esterno**: riepiloga ciò di cui si è trattato durante le riunioni del team, in presenza di partecipanti esterni.
- Milestone delle revisioni di avanzamento del progetto:
  - **RR - Revisione dei Requisiti**: studio dei requisiti del capitolato scelto e dell'analisi svolta dal team. Se si è svolto un buon lavoro ci si aggiudica l'appalto.
  - **RP - Revisione di Progetto**: riguardante la definizione della struttura dell'architettura del software e la presentazione di una *Technology Baseline*, che rappresenti un'adeguata conoscenza e motivazione d'uso delle tecnologie, dei FRAMEWORKGI e librerie, corredata da un PROOF OF CONCEPT(POC)GI coerente con gli obiettivi.
  - **RQ - Revisione di Qualifica**: riguarda la codifica del prodotto presentata con la *Product Baseline* e una definizione dettagliata della demo creata.
  - **RA - Revisione di Accettazione**: se il prodotto soddisfa i requisiti e le aspettative, viene accettato e rilasciato.
- Ruoli assegnati ai membri del team:
  - **Re - Responsabile di progetto**
  - **Am - Amministratore di progetto**
  - **An - Analista**
  - **Pt - Progettista**
  - **Pr - Programmatore**
  - **Ve – Verificatore**

3.1.4.2 Elementi grafici

3.1.4.2.1 Tabelle

Le tabelle sono sviluppate tramite l'uso dello strumento fornito da Word. Ogni tabella è accompagnata da un'opportuna didascalia descrittiva scritta in seguito alla tabella e preceduta dalla seguente notazione:
Tabella [numero tabella]

3.1.4.2.2 Immagini

Le immagini sono posizionate al centro, numerate con la seguente regola:
Figura [numero immagine]
seguita dalla didascalia descrittiva.

3.1.4.2.3 Diagrammi UML

I diagrammi UMLGI vengono inseriti nei documenti in formato immagine ed indicati con la seguente notazione:
Img UML [numero UML]
Se il Redattore lo ritiene necessario, a fianco della notazione può essere inserita una breve spiegazione del diagramma.

3.1.5 Mantenimento

3.1.5.1 Versionamento

Tutti i documenti sono identificati da un formato tale per cui l'identificazione sia univoca e relativa alla versione del documento a cui ci si riferisce, al fine di avere una consultazione dello storico della documentazione più diretta e versatile durante tutto il CICLO DI VITAGI del documento.
Lo schema adottato come modello di versionamento è il CHANGE SIGNIFICANGEGI e la notazione da usare è la seguente:
X.Y.Z-[Write]
dove:

- **X:** indica lo stato di avanzamento del prodotto. Può assumere i seguenti valori:
  - **0:** nessuna funzionalità del prodotto implementata;
  - **α:** lo sviluppo del prodotto è iniziato e sono state implementate alcune funzionalità. Intervallo funzionalità implementate (0%, 80%];
  - **β:** è stata implementata la maggior parte delle funzionalità del prodotto. Intervallo funzionalità implementate (80%, 100%);
  - **1:** il prodotto è completo di tutte le funzionalità e pronto per il rilascio.
- **Y:** indica la baseline di avanzamento rispetto agli obiettivi prefissati nel *Piano di Progetto*; parte da zero e viene incrementato dal Responsabile prima di ogni accesso alle revisioni di avanzamento fissate dal cliente. Può assumere i seguenti valori:
  - **0:** nessuna revisione di avanzamento sostenuta;
  - **1:** è stata sostenuta la *Revisione dei Requisiti* (RR);
  - **2:** è stata sostenuta la *Revisione di Progetto* (RP);
  - **3:** è stata sostenuta la *Revisione di Qualifica* (RQ);
  - **4:** è stata sostenuta la *Revisione di Accettazione* (RA);

- **Z:** indica il numero della versione del documento; parte da zero e viene incrementato dal Verificatore quando il documento è stato validato e approvato. Viene riportato a zero ad ogni modifica dell'indice Y;
- **[Write]:** indica gli incrementi effettuati dal team; parte da zero e viene incrementato dal redattore ad ogni modifica del documento. Viene riportato a zero quando cambia l'indice Y o l'indice Z.

3.1.5.2 Issue Tracking Sistem

Al fine di allineare gli ambienti di lavoro, per lo sviluppo dei documenti in cooperazione si è deciso di adottare il principio del ISSUE TRACKING SYSTEMGI.
Ciò permette di poter comunicare tramite appositi ticket tra membri del team evidenziando i problemi emersi, ma anche per ottenere una gestione migliore delle attività e del tempo e del personale ad esse collegati. Questo sistema permette anche l'analisi dell'allocazione delle risorse, della contabilità temporale, della gestione delle priorità e del flusso di lavoro di supervisione oltre all'implementazione di un registro centralizzato delle emissioni.
Tramite l'uso di strumenti quali Windows Teams e Google Drive per i documenti, mentre di Git hub per il software è possibile far uso di questo sistema per il tracciamento dei problemi e delle attività oltre a permettere il versionamento dei documenti e del codice.
Il team ha stabilito di separare l'implementazione del software, relegandolo al REPOSITORYGI di *Git hub*, dallo sviluppo dei documenti nel loro contenuto su *Google Drive* e la successiva impaginazione tramite l'uso del software *Microsoft Teams* su un file *Word*. I file verranno inseriti nel repository, appena ricevono l'approvazione.

3.1.5.3 Suddivisione tipologia documenti

3.1.5.3.1 Documenti

I documenti sviluppati durante il progetto sono suddivisi per macro-categoria tra *Documenti Interni*, destinati solamente al gruppo, e *Documenti Esterni*, destinati al gruppo, ma anche al committente e al proponente.
Inoltre, i documenti saranno suddivisi per stato, cioè se approvati sono raggruppati insieme a tutti i documenti che hanno superato lo stato di Approvazione, identificati come *Documenti Approvati*, in caso contrario saranno raggruppati tra i *Documenti in Rielaborazione*; questi ultimi sono rintracciabili o nel REPOSITORYGI di *Google Drive* o in *Microsoft Teams* a seconda dello stato di elaborazione del documento.
Infine, l'ultima tipologia, che non rientra tra le precedenti, è quella identificata come *Documenti Template*, questi documenti vengono usati per accelerare lo sviluppo della documentazione.

3.1.5.3.2 Verbali

Sono documenti riepilogativi dei vari incontri fisici svolti con i membri del gruppo. Questi sono identificati in due macro-categorie:

- **Verbali interni**: svolti solo tra i membri del gruppo presenti;
- **Verbali esterni**: svolti tra i membri presenti ed eventuali individui esterni come proponente e committente.

Questi documenti devono rispettare una struttura specificata, a partire da un file modello in *Word*, presente tra i *Documenti Template*.

La struttura sarà così sviluppata:

- **Frontespizio**
  - **Logo del gruppo**: posizionato centralmente;
  - **Nome del gruppo**: posizionato centralmente;
  - **Titolo verbale**: specificato nel formato:

Verbale [interno/esterno];

  - **Informazioni sul documento**
    - **Versione**: numero della versione;
    - **Data creazione**: data creazione verbale nel formato YYYY/MM/DD;
    - **Redattore**: nominativo (nome cognome) del Redattore del documento;
    - **Verificatore**: nominativo (nome cognome) del Verificatore del documento;
    - **Destinatario**: a chi è rivolto il documento;
    - **E-mail di riferimento**: indirizzo di posta elettronica utile in caso servissero chiarimenti (inserita solo nei verbali esterni);
- **Informazioni incontro**
  - **Data incontro**
  - **Luogo**
  - **Ora inizio**
  - **Ora fine**
  - **Partecipanti interni**
  - **Partecipanti esterni**
- **Ordine del giorno**: elenco degli argomenti trattati in quell'incontro;
- **Discussione argomenti**: corpo del documento.

3.1.6 Metriche

3.1.6.1 Metriche di valutazione: Pianificazione

- **Schedule Variance (SV):**
  - **Scopo**: indicare se si è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate;
  - **Risultato**: numero di attività svolte in confronto alle attività pianificate. Se il valore è positivo si sta procedendo con maggior EFFICIENZAGI e minor costo rispetto a quanto preventivato, viceversa se negativo.
- **Budget Variance (BV):**
  - **Scopo**: indicare se, al momento della misurazione, si è speso di più o di meno rispetto a quanto previsto alla data corrente. È un indicatore che ha valore unicamente contabile e finanziario;
  - **Risultato**: percentuale del budget usato in confronto al budget preventivato. Se il valore ottenuto è positivo significa che il budget di progetto viene speso con minor velocità di quanto pianificato, viceversa se negativo.

3.1.6.2 Metriche di valutazione: Documentazione

- **Indice di Gulpease (IG):**
  - **Scopo**: fornire un'indicazione sul grado di leggibilità del testo. È tarato sulla lingua italiana e considera la lunghezza delle parole, il numero delle frasi ed il numero delle parole totali;
  - **Risultato**: valore intero compreso tra 0 e 100 dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. In generale i testi con indice inferiore a 80 sono difficili da leggere per chi ha la licenza elementare, quelli con indice inferiore a 60 sono difficili da leggere per chi ha la licenza media e se l'indice è inferiore a 40 risultano difficili da leggere per chi ha un diploma superiore.
- **Correttezza ortografica (CO):**
  - **Scopo**: i testi prodotti non devono presentare errori ortografici;
  - **Risultato**: valore intero che indica il numero di errori grammaticali o ortografici presenti nel documento.

3.1.7 Strumenti

In questa sezione del capitolo si specificano gli strumenti e il loro uso per lo sviluppo della documentazione.

3.1.7.1 Documenti Google in Google Drive

Strumento usato nelle fasi iniziali della redazione del documento, al fine di inserire lo sviluppo di tutti gli argomenti che devono essere trattati. Essendo un file condiviso in una cartella condivisa a tutti i membri del gruppo, i Relatori possono inserire il loro paragrafo senza intaccare il lavoro svolto in parallelo da altri membri sullo stesso documento. È inoltre disponibile un sistema di versionamento che permette a tutti i membri del team di consultare le modifiche fatte e i loro autori.

3.1.7.2 Microsoft Office Word

Appena il documento risulta essere completo, in quanto i capitoli che costituiscono l'indice sono stati tutti sviluppati, si procede all'impaginazione. Un Redattore prende il file template di partenza presente su *Microsoft Teams* per l'impaginazione del documento, specificando le caratteristiche generali, definendo l'indice e inserendo il corpo del testo presente sul file di *Google Drive*.
Le attività di verifica e rielaborazione vengono svolte sul file *.docx*; i Verificatori possono utilizzare il meccanismo di commento messo a disposizione da *Microsoft Word* per evidenziare e notificare ai Redattori i punti dei documenti che necessitano di correzione o revisione. È possibile risalire allo storico delle modifiche grazie a un sistema di versionamento interno che permette di visualizzare la cronologia dei cambiamenti apportati al file e i rispettivi autori.

3.1.7.3 Microsoft Teams

Strumento usato per la coordinazione del Team in merito allo sviluppo della documentazione. In esso i membri del team inseriscono le versioni dei file sviluppati in *Word*, si contattano sullo sviluppo della documentazione tramite le chat dei canali creati per lo specifico documento in sviluppo e seguono lo sviluppo delle attività tramite il Planner.

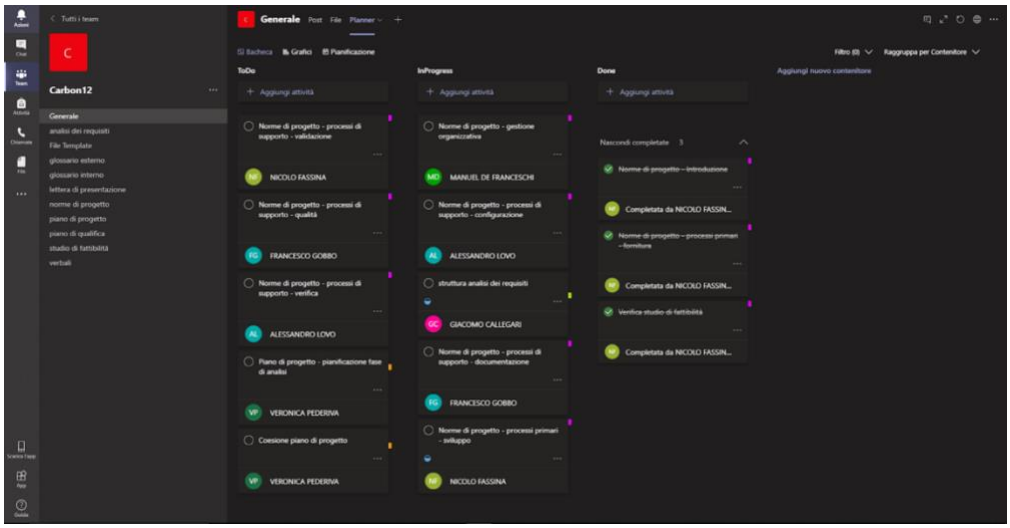

FIGURA 1 - MICROSOFT TEAMS

Reperibile all'indirizzo: https://teams.microsoft.com/start

3.1.7.4 Astah UML

Astah UML è lo strumento open source usato per la creazione dei diagrammi UMLGI.
Reperibile all'indirizzo: http://astah.net/

3.1.7.5 Gantt Project

Strumento open source per la realizzazione dei DIAGRAMMI DI GANTTGI.
Reperibile all'indirizzo: https://www.ganttproject.biz/

3.1.7.6 UMLet

UMLet è un altro strumento utilizzato per la creazione dei diagrammi UMLGI.
Reperibile all'indirizzo: https://www.umlet.com

3.2 Gestione della configurazione

3.2.1 Scopo

Definisce strategie di gestione di un progetto software in evoluzione, essenziali quando più sviluppatori vi lavorano contemporaneamente. Assicura infatti l'identificazione e la tracciabilità delle varie versioni delle componenti del progetto, ma anche il controllo sull'eventuale implementazione di modifiche garantendo che queste non interferiscano tra loro.

3.2.2 Versionamento

Sia il software che i documenti necessitano di essere versionati, in modo tale da poter ricostruire la storia delle loro versioni ed accedere a ciascuna di esse durante il loro intero CICLO DI VITAGI.
Il versionamento della documentazione è stato ampliamente trattato nella sezione 3.1.5.1 del presente documento. Per quanto riguarda il versionamento dei sorgenti software si è deciso di utilizzare la seguente notazione:

X.Y.Z-[Build]

dove:

- **X:** indica lo stato di avanzamento del prodotto. Può assumere i seguenti valori:
  - **0:** nessuna funzionalità del prodotto implementata;

    - **α:** lo sviluppo del prodotto è iniziato e sono state implementate alcune funzionalità. Intervallo funzionalità implementate (0%, 80%];
    - **β:** è stata implementata la maggior parte delle funzionalità del prodotto. Intervallo funzionalità implementate (80%, 100%);
    - **1:** il prodotto è completo di tutte le funzionalità e pronto per il rilascio.
- **Y:** indica la baseline di avanzamento rispetto agli obiettivi prefissati nel *Piano di Progetto*; parte da zero e viene incrementato dal Responsabile prima di ogni accesso alle revisioni di avanzamento fissate dal cliente. Può assumere i seguenti valori:
    - **0:** nessuna revisione di avanzamento sostenuta;
    - **1:** è stata sostenuta la *Revisione dei Requisiti* (RR);
    - **2:** è stata sostenuta la *Revisione di Progetto* (RP);
    - **3:** è stata sostenuta la *Revisione di Qualifica* (RQ);
    - **4:** è stata sostenuta la *Revisione di Accettazione* (RA);
- **Z:** indica il numero della versione del sorgente software; parte da zero e viene incrementato dal Verificatore quando il software è stato validato e approvato. Viene riportato a zero ad ogni modifica dell'indice Y;
- **[Build]:** indica gli incrementi effettuati dal team; parte da zero e viene incrementato dal redattore ad ogni modifica del sorgente software. Viene riportato a zero quando cambia l'indice Y o l'indice Z.

3.2.3 Gestione dei cambiamenti

La gestione dei cambiamenti adottata prevede la definizione di alcuni accorgimenti e regole a cui tutti i membri del team devono attenersi. Quando si vogliono apportare modifiche minori come delle correzioni grammaticali o dei miglioramenti sintattici ad un documento, è possibile procedere direttamente in autonomia. Per effettuare modifiche consistenti sui contenuti o la struttura dei documenti sarà necessario invece contattare un Responsabile del file: se le modifiche saranno accettate, potranno essere pubblicate. Per quanto riguarda il codice software prodotto, i membri del gruppo potranno liberamente modificare i file in ogni branch ad eccezione del ramo principale, sul quale le modifiche saranno introdotte solo dopo l'eventuale approvazione di una pull request da parte di un altro membro.

3.2.4 Repository

Si è scelto di utilizzare Git come sistema di versionamento distribuito. La scelta del servizio dove ospitare il REPOSITORYGI remota è invece ricaduta su GitHub. Tale repository è raggiungibile all'indirizzo:

https://github.com/carbondodici/Predire_in_Grafana

3.2.4.1 Struttura

Il REPOSITORYGI è organizzato in cartelle:

- **RR** contiene i documenti realizzati per la *Revisione dei Requisiti*, suddivisi in *Documenti Interni* e *Documenti Esterni*.

Le cartelle **RP**, **RQ** e **RA** saranno in seguito aggiunte per le successive consegne.

3.2.4.2 Utilizzo di Git

Per utilizzare il sistema di versionamento è possibile lavorare direttamente da linea di comando o sfruttare software con interfaccia grafica, come GitKraken, che consentono di velocizzare e semplificare le operazioni da svolgere.
Si sono stabilite alcune linee guida per una migliore collaborazione:

- Utilizzo di branch per sviluppi paralleli di compiti differenti;
- Commit frequenti e ben commentati per mantenere uno storico preciso dello sviluppo e per facilitare la comprensione del lavoro effettuato da tutti i membri del gruppo;
- Laddove fossero necessarie modifiche consistenti ad un file è richiesto di contattare il Responsabile di quel file per proporre i cambiamenti da effettuare, altrimenti procedere direttamente.

3.3 Gestione della qualità

3.3.1 Scopo

In questa sezione delle Norme si vuole definire il processo di GARANZIA DELLA QUALITÀGI. Ciò deve fornire la garanzia per le attività di produzione software e per i processi del CICLO DI VITAGI che conseguiranno la realizzazione del prodotto, in rapporto alla conformità dei requisiti specificati e al rispetto dei piani stabiliti.

3.3.2 Aspettative

La Gestione della Qualità ha le seguenti prerogative:

- Assicurare la qualità nell'organizzazione e nello sviluppo dei processi;
- Assicurare la qualità del prodotto;
- La qualità deve essere provata oggettivamente;
- Ottenere l'approvazione e la soddisfazione finale del committente e proponente.

3.3.3 Descrizione

La GARANZIA DELLA QUALITÀGI ha necessità di essere gestita in modo imparziale e organizzato, perciò il Responsabile di Progetto è il membro del gruppo che si deve assumere la responsabilità dello sviluppo del software e dell'esecuzione dei processi che conservino la qualità.
Inoltre, la gestione della Qualità viene trattata più ampiamente nel documento *Piano di Qualifica*, nel quale vengono descritte le modalità che garantiscono la qualità dei processi e del prodotto.

All'interno del *Piano di Qualifica* si sono spiegate le seguenti attività basilari per la garanzia della Qualità:

1. **Process implementation**: per stabilire gli standard di qualità, le metodologie, procedure e strumenti per l'esecuzione delle attività che garantiscano la qualità;
2. **Product assurance**: la garanzia che i prodotti software e la documentazione fornita siano conformi al contratto e ai piani stabiliti;
3. **Process assurance**: garanzia che il personale, gli strumenti e le attività del piano siano conformi al contratto e ai piani di sviluppo;
4. **Assurance of quality systems**: la garanzia che le ulteriori attività di gestione della qualità siano conformi alle clausole dell'*ISO 9001*, se specificate nel contratto.

Inoltre, nel *Piano di Qualifica* vengono analizzati i seguenti argomenti:

- Presentazione degli standard utilizzati;
- Individuazione dei processi d'interesse supportati da degli standard;
- Individuazione degli attributi software significativi per lo sviluppo del prodotto.

Ad ogni processo vengono stabiliti:

- Gli obiettivi da perseguire
- Le strategie da applicare
- Le metriche da utilizzare.

Gli obiettivi e le metriche sono i dati essenziali per l'analisi dei processi in ambito del perseguimento della qualità. Questi definiscono lo stato di miglioramento del gruppo, puntando alla realizzazione di un software e una documentazione che soddisfino le aspettative in merito alla qualità.

3.3.4 Attività

Le tre attività da applicare in questo ambito sono:

1. **Pianificazione**: con l'obiettivo di porsi obiettivi di qualità, la definizione delle strategie per il conseguimento della creazione del prodotto, gestendo al meglio tempo e risorse;
2. **Valutazione**: verificare tramite apposite metriche l'attuazione del piano, monitorando i dati della qualità di processo durante lo svolgimento e della qualità del prodotto ottenuto fino ad ora;
3. **Reazione**: sulla base dei risultati ottenuti, attivare le strategie in funzione del miglioramento continuo.

3.3.5 Metriche

- **Metriche soddisfatte (MS):**
  - **Scopo**: misurare la quantità di metriche soddisfatte, ovvero che rientrano nell'intervallo accettabile;
  - **Risultato**: percentuale di metriche soddisfatte sul totale di metriche proposte.
- **Metriche pienamente soddisfatte (MPS):**
  - **Scopo**: misurare la quantità di metriche pienamente soddisfatte, ovvero che rientrano nell'intervallo desiderabile;
  - **Risultato**: percentuale di metriche pienamente soddisfatte sul totale di metriche proposte.

3.3.6 Strumenti

Gli strumenti che supportano la qualità sono:

- Le norme dello standard *ISO 9126*;
- Le norme dello standard *ISO 12207*;
- Le norme dello standard *ISO 15504*;
- Le norme del metodo *Ciclo di Deming (o PDCA)*GI;
- Le metriche;
- Risultati emersi dalle seguenti attività danno supporto alla GARANZIA DELLA QUALITÀGI raggiunta:

- Verifica;
- Validazione;
- Revisioni;
- Risoluzione problemi.

3.4 Verifica

3.4.1 Scopo

La verifica è il processo attraverso il quale si controlla che la realizzazione del prodotto software stia procedendo correttamente. I prodotti devono infatti soddisfare requisiti e specifiche prefissati in precedenza e non devono presentare problematiche che ne impediscano la conformità all'uso a cui sono destinati. Anche la documentazione è soggetta a verifica.

3.4.2 Aspettative

Il processo di verifica segue alcuni punti principali:

- Vengono stabiliti criteri per la verifica;
- Viene effettuata una procedura di verifica;
- Vengono individuati eventuali errori;
- Dopo la verifica il prodotto è in uno stato stabile;
- Permette la validazione.

3.4.3 Descrizione

L'esecuzione e la buona riuscita del processo di verifica si basano su due attività: analisi e test.

3.4.4 Attività

3.4.4.1 Analisi

L'analisi consiste nell'analizzare il codice sorgente e nella sua esecuzione. In particolare, può essere statica o dinamica.

3.4.4.1.1 Analisi statica

L'analisi statica è solitamente il primo controllo effettuato sul codice poiché non richiede l'esecuzione del prodotto software, ma prevede di valutarne la correttezza e la conformità rispetto a regole e linee guida stabilite per il progetto. L'analisi statica viene effettuata sia manualmente dalle persone del gruppo, sia automaticamente da macchine con metodi formali. Nell'ultimo caso basti pensare all'analisi svolta in fase di compilazione dai compilatori per poter generare il codice oggetto (cercando anomalie come nomi di identificatori non dichiarati, incoerenza tra tipi, codice non raggiungibile dal flusso, ecc..). Due metodi manuali di analisi statica sono invece:

- **Walkthrough:** analisi informale svolta dai componenti del gruppo che pone l'attenzione sulla ricerca di possibili errori in tutto il prodotto piuttosto che sulla loro risoluzione;
- **Inspection:** prevede una ricerca degli errori attraverso una lettura mirata di parti specifiche.

Nel momento in cui i metodi di verifica saranno consolidati per tutto il gruppo, si definisce e si utilizza una lista di controllo, che andrà aggiornata man mano, contenente i tipici errori nei quali i verificatori possono imbattersi (formato della data, punteggiatura negli elenchi, tempi verbali, ecc..) per facilitare e velocizzare il processo di verifica.

3.4.4.1.2 Analisi dinamica

L'analisi dinamica valuta un prodotto software durante la sua esecuzione. Viene effettuata mediante l'esecuzione di test che devono essere ripetibili e dei quali è noto il comportamento atteso, in modo da poter stabilire la correttezza di quello osservato.

3.4.4.2 Test

L'obiettivo dei test è dimostrare che il prodotto software svolga i compiti per i quali è stato realizzato, portando alla luce eventuali errori presenti. I test durante il loro CICLO DI VITAGI possono trovarsi in uno di questi stati:

- **I**: test implementato;
- **NI**: test non implementato.

Inoltre:

- **S**: test soddisfa la richiesta;
- **NS**: test non soddisfa la richiesta.

In base allo scopo e all'oggetto di verifica si definiscono diversi tipi di test.

3.4.4.2.1 Test di unità

Si focalizzano sulla verifica delle funzionalità di singole unità di programma, come i metodi o le classi di oggetti. Prevedono, se necessario, l'utilizzo di DRIVERGI e STUBGI, rispettivamente un modulo guida che invoca l'unità oggetto di test inviandole opportuni valori e un modulo fittizio che viene chiamato dall'unità sotto test e che simula, per esempio, il funzionamento di una funzione esterna (per garantire maggiore velocità o per riprodurre eventi rari).

TU [id]

id = codice numerico identificativo del componente analizzato;

3.4.4.2.2 Test di integrazione

Vengono integrate più unità di programma testate in precedenza per creare componenti più complesse, che a loro volta devono essere verificate. Una volta fatto ciò, tali componenti possono a loro volta essere considerate come unità di aggregati maggiori e così via fino a raggiungere la dimensione totale del prodotto.

TI [id]

id = codice numerico identificativo del componente o componenti analizzato/i;

3.4.4.2.3 Test di sistema

Il test di sistema prevede l'esecuzione dei test sul sistema completo dopo aver effettuato l'integrazione dei vari componenti per controllare che questi siano compatibili e interagiscano correttamente. In caso di successive aggiunte di nuovi componenti, si procede nuovamente al test del sistema. L'importanza di questo tipo di test risiede nel fatto che i sistemi hanno un comportamento emergente, cioè che alcune funzionalità diventano evidenti solo quando tutti i componenti sono stati assemblati.

TS [id]

id = codice numerico identificativo gerarchico del componente o componenti analizzato/i;

3.4.4.2.4 Test di regressione

Una serie di test che è sempre possibile effettuare dopo ogni modifica al sistema per controllare che i cambiamenti apportati non abbiano introdotto nuovi bug.

TR [id]

id = codice numerico identificativo del componente o componenti analizzato/i;

3.4.4.2.5 Test di accettazione

Test eseguiti in presenza dei clienti affinché si possa decidere se il sistema è pronto per essere accettato e distribuito.

TA [Importanza] [Tipo] [Codice]

- **Importanza**: basandosi sempre sui requisiti, il test ne assume l'importanza, che può essere:
  - **F**→ requisito funzionale;
  - **P**→ requisito prestazionale;
  - **Q**→ requisito di qualità;
  - **V**→ requisito di vincolo.
- **Tipo**: in base al requisito che si vuole soddisfare, il test ne assume la stessa importanza. Può essere di tre tipologie:
  - **O**→ requisito obbligatorio;
  - **D**→ requisito desiderabile;
  - **F**→ requisito facoltativo.
- **Codice**: valore numerico per identificare il test e a quale requisito fa riferimento.

3.4.5 Metriche

- **Copertura dei test (CT):**
  - **Scopo**: indicare il numero di righe di codice interessate dai test durante la loro esecuzione;
  - **Risultato**: valore percentuale che esprime la copertura del codice testato. La copertura dei test si suddivide in:
    - **Function coverage (CT-FC):** verificare che ogni funzione sia stata chiamata;
    - **Statement coverage (CT-SC):** verificare che ogni statement sia stato eseguito;
    - **Branch coverage (CT-BC):** verificare che tutti i possibili branch derivanti da `if` e `case` statement, siano stati eseguiti;
    - **Condition coverage (CT-CC):** verificare che ogni condizione booleana sia stata valutata in entrambi i suoi possibili stati.
    - **Path coverage (CT-PC):** misura la capacità di coprire, mediante l'esecuzione di test, tutti i path di un modulo.
    - **Boolean coverage (CT-BC):** misura la capacità di coprire, mediante l'esecuzione di test, tutte le condizioni di un'espressione booleana di un modulo.

3.4.6 Strumenti

3.5.6.1 Verifica ortografica

Si utilizza lo strumento di controllo dell'ortografia e della grammatica di Microsoft Word durante la digitazione per individuare parole o frasi errate e ripetizioni.

3.4.6.2 Validazione W3C

Per la validazione delle pagine di markup HTML e dei fogli di stile CSS si utilizzano gli strumenti messi a disposizione dal W3C, raggiungibili agli indirizzi:

- HTML: https://validator.w3.org/
- CSS: https://jigsaw.w3.org/css-validator/

3.4.6.3 Verifica JavaScript

Si è deciso di utilizzare i seguenti strumenti per l'analisi del codice JavaScript:

- **ESlint:** strumento open source per l'analisi statica del codice, in grado di rilevare errori e in molti casi anche di correggerli in automatico. Risulta inoltre altamente personalizzabile in modo da assicurare che tutti i membri del gruppo seguano le stesse convenzioni di codifica prestabilite.
  Eslint è accessibile al seguente indirizzo: https://eslint.org/
- **Jest:** FRAMEWORKGI per l'analisi dinamica del codice JavaScript che permette di scrivere ed eseguire test al fine di assicurare la correttezza del codice sviluppato. Jest è raggiungibile al seguente indirizzo: https://jestjs.io/

3.5 Validazione

3.5.1 Scopo

Il processo di Validazione ha lo scopo di accertare che il prodotto realizzato sia pienamente conforme alle attese. In particolare, deve soddisfare i requisiti dell'utente, le esigenze di tutte le parti interessate, gli aspetti di sicurezza e quelli normativi applicabili. La validazione è possibile solo dopo avere fatto delle verifiche.

3.5.2 Attività

La validazione comprende i seguenti passi:

- **Esecuzione dei test:** i Verificatori eseguono i test e stilano i risultati. Ogni test eseguito viene tracciato con un preciso codice identificativo;
- **Analisi dei risultati prodotti:** il Responsabile di progetto valuta i risultati e può decidere se:
  - Concludere la validazione;
  - Far ripetere tutti, o solo alcuni, test con nuove indicazioni.
- **Consegna dei risultati al proponente:** una volta terminata la validazione si inviano i risultati al proponente.

4 Processi organizzativi

4.1 Gestione dei ruoli di progetto

4.1.1 Scopo

Lo scopo di questo processo è di definire quali sono i ruoli di progetto che dovranno essere svolti e di normare la loro rotazione tra i vari membri di Carbon12.

4.1.2 Aspettative

Da questo processo ci si aspetta che:

- vengano definiti i vari ruoli che potranno essere assunti dai vari membri di Carbon12 e le loro funzioni;
- sia descritto come avvengono i cambi di ruolo.

4.1.3 Ruoli di progetto

Durante il corso del progetto i vari membri di Carbon12 dovranno svolgere vari ruoli, che sono:

- Responsabile di Progetto;
- Amministratore;
- Analista;
- Progettista;
- Programmatore;
- Verificatore.

4.1.3.1 Responsabile di Progetto

Il Responsabile di Progetto rappresenta il progetto presso i Committenti e la Proponente. Ha responsabilità sulle scelte e la loro approvazione e ha il compito di coordinare l'intero progetto.
In particolare:

- Gestisce la pianificazione stimando le risorse necessarie, le scadenze e i costi da sostenere;
- Gestisce e controlla le attività delle risorse umane;
- Coordina il gruppo e le attività che i suoi membri devono svolgere;
- Controlla i progressi del progetto ed in base al suo andamento aggiorna le pianificazioni future;
- È responsabile dell'analisi dei rischi e della loro gestione;
- Gestisce la documentazione;
- Gestisce le comunicazioni esterne.

4.1.3.2 Amministratore

L'amministratore ha il compito di controllare l'ambiente di lavoro fornendo ai membri del gruppo gli strumenti necessari ad operare al meglio seguendo le regole. L'Amministratore quindi non deve effettuare scelte gestionali, ma piuttosto decidere come organizzare e gestire l'ambiente di lavoro scegliendo le regole, le procedure, gli strumenti ed i servizi. In particolare:

- Gestisce le risorse necessarie per i servizi di supporto;
- Risolve i problemi legati alla gestione dei processi;
- Gestisce e mantiene le regole e le procedure di lavoro;
- Individua strumenti utili al miglioramento dei processi;

- Si assicura che la documentazione sia corretta, verificata, approvata, versionata e di facile reperimento.

4.1.3.3 Analista

L'Analista ha il compito di analizzare il dominio del problema al fine di comprenderlo al meglio. Partecipa al progetto per un periodo di tempo limitato soprattutto nell'attività iniziale di analisi dei requisiti, la quale, se non svolta in modo ottimale, può portare a gravi errori di progettazione.
In particolare:

- Studia e definisce il problema da risolvere identificandone le criticità;
- Analizza il dominio delle richieste studiando i bisogni, espliciti ed impliciti, del proponente;
- Analizza il dominio applicativo, cioè gli utenti che utilizzeranno il prodotto e l'ambiente d'uso;
- Produce il documento di Analisi dei Requisiti e il documento di Studio di Fattibilità.

4.1.3.4 Progettista

Il progettista ha il compito di effettuare le scelte architetturali del progetto, influenzando gli aspetti tecnici e tecnologici. Partendo dai risultati dell'Analista, il Progettista ha il compito di trovare una possibile soluzione per i problemi ed i requisiti individuati. In particolare:

- Sviluppa l'architettura utilizzando un insieme di BEST PRACTICEGI per garantire la sua coerenza e consistenza;
- Effettua scelte che portino ad una soluzione attuabile, EFFICIENTEGI, sufficiente e comprensibile rispetto ai requisiti, rispettando i preventivi di costo e risorse;
- Definisce un'architettura logica del prodotto che sia facile da mantenere;
- Divide il sistema in componenti per favorire la modularità e il riutilizzo;
- Definisce una struttura che abbia un basso grado di accoppiamento;
- Sviluppa un'architettura che sia robusta, sicura e flessibile in modo che rimanga operativa in caso di malfunzionamento e sia modificabile a basso costo.

4.1.3.5 Programmatore

Il programmatore ha il compito di scrivere e mantenere il codice del prodotto seguendo l'architettura già definita dal Progettista. In particolare:

- Scrive codice documentato, versionato e mantenibile secondo quanto prestabilito dalle norme di codifica;
- Implementa i test necessari per la verifica e validazione del codice;

4.1.3.6 Verificatore

Il Verificatore ha il compito di controllare che le attività svolte siano conformi alle attese e alle norme prestabilite. In particolare:

- Controlla che le *Norme di Progetto* siano rispettate;
- Controlla che per ogni stadio del CICLO DI VITAGI del prodotto quest'ultimo sia conforme al *Piano di Qualifica*;
- Segnala al Responsabile eventuali situazioni di conflitto che violino la pianificazione stabilita nel *Piano di Progetto*.

4.1.4 Attività

4.1.4.1 Assegnazione ruoli

I ruoli assunti dai vari membri di Carbon12 potranno essere cambiati durante gli incontri interni. I cambiamenti di ruolo dovranno essere effettuati in modo da garantire la continuità delle attività. Inoltre bisognerà per quanto possibile evitare "conflitti di interesse", come per esempio cambiare ruolo ad un analista e metterlo come verificatore, per poi fargli verificare un documento scritto da lui.

4.2 Comunicazioni

4.2.1 Scopo

Lo scopo di questo processo è di definire le norme da seguire e gli strumenti da utilizzare per la comunicazione tra i vari stakeholder: proponente, committenti ed i membri del gruppo Carbon12.

4.2.2 Aspettative

Da questo processo ci si aspetta che descriva le tipologie di comunicazioni e i metodi con i quali vengono effettuate.

4.2.3 Attività

4.2.3.1 Comunicazioni interne

Per le comunicazioni interne vengono utilizzati due programmi: *Microsoft Teams* e *Telegram*.
Microsoft Teams sarà configurato con un canale dedicato per ogni attività. Inoltre, sarà presente un canale generale per le comunicazioni non strettamente legate ad alcuna attività specifica.
Telegram sarà configurato con un canale in cui saranno inclusi tutti i membri del gruppo e sarà usato per comunicazioni o che non appartengono a nessun argomento specifico dei canali esistenti su *Microsoft Teams*, oppure per messaggi urgenti, come per esempio la segnalazione di un ritardo a un incontro a causa di un imprevisto lungo la strada per arrivare al luogo prefissato.
Entrambi gli strumenti offrono la possibilità di menzionare nelle chat un membro del gruppo utilizzando la sintassi `@nome`; in questo modo è possibile evidenziare che un certo messaggio è rivolto principalmente a una o più persone, le quali riceveranno una notifica di menzione che attirerà maggiormente la loro attenzione.

4.2.3.2 Comunicazioni esterne

Questa sezione tratta le norme da seguire per la comunicazione con soggetti esterni al gruppo Carbon12.
I soggetti esterni individuati sono:

- I committenti, Prof. Tullio Vardanega e Prof. Riccardo Cardin, ai quali bisognerà fornire la documentazione richiesta ad ogni revisione di progetto e con i quali si intende stabilire un canale di comunicazione utile al miglioramento dei documenti e delle strategie del gruppo;
- La proponente Zucchetti SPA, con la quale si si intende mantenere un canale di comunicazione costante utile a stabilire bisogni e requisiti del prodotto richiesto.

Le comunicazioni esterne scritte dovranno essere effettuate esclusivamente mediante l'indirizzo e-mail del gruppo: carbon.dodici@gmail.com.
Le e-mail dovranno essere così strutturate:

- **E-mail verso i committenti**: nell'oggetto dovrà essere scritto nel modo più sintetico possibile il contenuto del messaggio. Ci si rivolgerà ai committenti dando loro del "Lei" o "Voi".

- **E-mail verso la proponente**: le e-mail saranno mandate al Sig. Gregorio Piccoli all'indirizzo gregorio.piccoli@zucchetti.it. L'oggetto dei messaggi dovrà essere "Grafana - Università di Padova" ed il corpo del messaggio dovrà iniziare con "Buongiorno," e terminare con "Cordiali saluti, Carbon12". Nei messaggi ci si rivolgerà al Sig. Gregorio Piccoli dandogli del "Lei".

4.2.4 Strumenti

Gli strumenti utilizzati saranno:

- Microsoft Teams e Telegram per le comunicazioni interne al gruppo;
- Gmail, Google Hangouts e Skype per le comunicazioni esterne.

4.3 Incontri

4.3.1 Scopo

Questo processo ha lo scopo di definire le modalità di svolgimento delle riunioni interne ed esterne a Carbon12.

4.3.2 Attività

Per ogni incontro verrà nominato dal gruppo Carbon12 un segretario che avrà il compito di far rispettare l'ordine del giorno, tenere nota degli argomenti trattati e delle decisioni prese e di scrivere il verbale dell'incontro, come specificato nel paragrafo successivo. Gli incontri potranno essere fisici, quindi svolti incontrandosi di persona, oppure virtuali, quindi svolti grazie a strumenti che permettano la comunicazione via Internet.
I membri di Carbon12 dovranno:

- Essere puntuali agli incontri;
- Comunicare eventuali ritardi o assenze al Responsabile di Progetto.

4.3.2.1 Incontri interni

Gli incontri interni verranno organizzati dal Responsabile di Progetto in accordo con i membri di Carbon12 attraverso i canali di comunicazione per le comunicazioni interne quando ritenuti necessari. Agli incontri interni potranno partecipare solamente i membri del gruppo.
Il Responsabile di Progetto dovrà:

- Decidere data e ora;
- Comunicare ai membri del gruppo eventuali cambiamenti di data o orario;
- Stabilire l'ordine del giorno dell'incontro.

Le decisioni che verranno discusse durante gli incontri saranno approvate o respinte mediante votazione dei membri del gruppo presenti all'incontro.

4.3.2.1.1 Incontri interni fisici

Gli incontri fisici saranno effettuati di persona dai vari membri di Carbon12. Il Responsabile di Progetto, oltre a quanto già detto, dovrà scegliere un luogo per l'incontro.

4.3.2.1.2 Incontri interni virtuali

Nel caso di incontri virtuali verrà utilizzato il software *Microsoft Teams* per effettuare chiamate vocali o video.

4.3.2.2 Incontri esterni
Gli incontri esterni includeranno tra i partecipanti anche soggetti esterni al gruppo. Gli incontri verranno organizzati dal Responsabile di Progetto insieme ai soggetti esterni interessati per quell'incontro.

4.3.2.2.1 Incontri esterni fisici
Gli incontri esterni fisici saranno effettuati di persona. Il luogo dell'incontro sarà scelto dal Responsabile di Progetto soggetti esterni interessati per quell'incontro.

4.3.2.2.1 Incontri esterni virtuali
Gli incontri esterni virtuali saranno effettuati mediante l'utilizzo di software che permetta di effettuare chiamate vocali e/o video. Il software da utilizzare sarà specificato dai soggetti esterni che parteciperanno a quell'incontro.

4.3.3 Verbale dell'incontro
Per ogni incontro fisico il Segretario avrà il compito di scrivere il verbale seguendo lo schema presente nel *Documento Template Verbali*, come specificato nella sezione 3.1.5.3.2 del presente documento.

4.3.4 Strumenti
Per l'organizzazione degli incontri saranno utilizzati gli strumenti per le comunicazioni interne. Per effettuare gli incontri interni virtuali sarà utilizzato Microsoft Teams, mentre per gli incontri esterni virtuali il software da utilizzare sarà scelto dai componenti esterni dell'incontro.

4.4 Formazione del personale
4.4.1 Scopo
Lo scopo della formazione del personale è di stabilire come ed attraverso quali fonti i componenti di Carbon12 dovranno studiare gli argomenti e gli strumenti necessari allo svolgimento del progetto.

4.4.2 Aspettative
I membri del gruppo dovranno conoscere i vari temi e programmi specificati in questo processo.

4.4.3 Attività
I membri del gruppo dovranno procedere in modo autonomo nello studio delle varie tecnologie richieste dal progetto. Si consiglia comunque di prendere in esame la seguente documentazione, indicata anche dal proponente:

- Per lo studio del funzionamento di GRAFANAGI:
  - https://grafana.com/docs/grafana/latest/
- Per lo studio delle SVMGI:
  - https://en.wikipedia.org/wiki/Support-vector_machine
  - https://lorenzogovoni.com/support-vector-machine/
  - https://en.wikipedia.org/wiki/Precision_and_recall
- Per lo studio della LRGI:
  - https://en.wikipedia.org/wiki/Linear_regression

- https://www.itl.nist.gov/div898/handbook//pmd/section4/pmd431.htm
- https://cran.r-project.org/doc/contrib/Ricci-regression-it.pdf
- https://it.wikipedia.org/wiki/Quartetto_di_Anscombe

Sarà inoltre necessario saper utilizzare i seguenti software per la realizzazione del progetto e per il coordinamento del gruppo. La lista qui indicata è da considerare non definitiva in quanto nelle fasi successive del progetto potrebbe sorgere la necessità di utilizzare altri strumenti al momento non individuati.

- **Microsoft Teams**: software che permette la gestione di un'area di lavoro condivisa tra tutti i membri del team nella quale è possibile definire dei canali tematici per una migliore organizzazione dei materiali e delle comunicazioni. Il programma permette infatti di inviare comunicazioni, lavorare sui file condivisi di documentazione, avviare discussioni su diverse tematiche legate al progetto, effettuare chiamate e videochiamate. La piattaforma è integrabile con diverse APP, quali il Planner che permette di definire e assegnare le attività da svolgere. Il software è disponibile anche come applicazione mobile, pertanto i membri del gruppo possono sempre avere una visione aggiornata dello stato del progetto anche accedendo con i propri dispositivi mobili.
- **Telegram**: programma di messaggistica utilizzato per le comunicazioni interne del gruppo;
- **Microsoft Word**: software utilizzato per la stesura dei documenti;
- **GanttProject**: software utilizzato per la pianificazione delle attività e delle risorse;
- **Git**: strumento utilizzato per il versionamento;
- **GitHub**: servizio di hosting che verrà utilizzato per il salvataggio in remoto del REPOSITORYGI di Git;
- **VS Code:** strumento utilizzato per la scrittura del codice JavaScript;
- **UMLet**: strumento utilizzato per la realizzazione dei diagrammi UMLGI;
- **Google Drive**: servizio di storage offerto da *Google* utilizzato per la condivisione di file tra i vari membri del gruppo. In *Google Drive* è possibile creare documenti collaborativi a cui possono contribuire tutti i membri del gruppo. I documenti di *Google* permettono la scrittura contemporanea di più persone e di mantenere uno storico delle modifiche apportate dagli utenti.

A Standard di Qualità

A.1 ISO/IEC 9126

ISO/IEC 9126 (1991): Software product evaluation - Quality characteristics and guidelines for their use - era lo standard internazionale sviluppato per definire le caratteristiche di qualità appropriate tenendo conto dell'uso finale del prodotto.
Questo standard ha definito sei caratteristiche di qualità che descrivono il modello del processo di valutazione del prodotto software.
Lo standard è suddiviso in 4 parti riportate di seguito.

A.1.1 Modello della qualità

Il modello di qualità del software definisce le sei caratteristiche principali, suddivise a loro volta in sotto-caratteristiche, misurabili attraverso specifiche metriche, al fine di fornire una scala e un metodo per la misurazione. Vengono riportate le caratteristiche alla base del modello.

A.1.1.1 Funzionalità

Caratteristica che descrive la capacità del software di soddisfare i requisiti, approfonditi *nell'Analisi dei Requisiti*, in un determinato contesto.
Affinché il software abbia questa caratteristica deve essere:

- **Appropriato**: deve fornire appropriate funzioni per specifiche attività, che permettano il raggiungimento degli obiettivi prefissati;
- **Accurato**: deve fornire i risultati concordati o la precisione ricercata;
- **Interoperabile**: l'interazione e l'operabilità tra uno o più sistemi;
- **Sicuro**: la capacità di fornire la protezione per le informazioni e i dati;
- **Conforme alla funzionalità**: l'adesione agli standard.

A.1.1.2 Affidabilità

Caratteristica del software di mantenere un determinato livello di prestazioni, in condizioni specifiche per un limitato periodo di tempo.

- **Maturità**: l'abilità di evitare il verificarsi di errori, malfunzionamenti o risultati non desiderati;
- **Robustezza**: il mantenimento del livello delle prestazioni stabilite anche al verificarsi di malfunzionamenti o uno scorretto uso del prodotto;
- **Recuperabilità**: la ricostituzione del livello appropriato delle prestazioni o il recupero di informazioni rilevanti anche successive ad un malfunzionamento;
- **Conforme alla affidabilità**: l'adesione agli standard in merito all'affidabilità.

A.1.1.3 Efficienza

Il prodotto ottiene questa capacità se esegue le proprie funzioni minimizzando il tempo d'esecuzione e sfruttando le risorse al meglio.

- **Tempo**: la capacità di reagire velocemente con opportuni tempi di risposta tendenzialmente brevi;
- **Spazio**: l'uso corretto e efficiente delle risorse disponibili;
- **Conforme all'efficienza**: l'adesione agli standard in merito all'EFFICIENZAGE.

A.1.1.4 Usabilità
Caratteristica per la quale l'uso del prodotto software permette una comprensione e una facilità nell'utilizzo da parte dell'utente, quando usato sotto determinate condizioni.
Per il raggiungimento di tale risultato, il prodotto deve essere:

- **Comprensibile**: l'essere chiaro in merito alle funzionalità e l'utilizzo;
- **Apprendibile**: l'essere facile da apprendere dagli utenti;
- **Operabile**: il permettere all'utente di poter eseguire le attività che vuole eseguire controllandone l'uso;
- **Attrattivo**: la capacità di attrarre l'utente per l'utilizzo;
- **Conforme all'usabilità**: l'adesione agli standard in merito all'usabilità.

A.1.1.5 Manutenibilità
Capacità del software nel poter essere modificato, ottenendo aggiornamenti per correzioni, miglioramenti o adattamenti.
Per il raggiungimento di tale caratteristica il prodotto deve poter essere:

- **Analizzabile**: facilità nell'analisi per la localizzazione di errori;
- **Modificabile**: avere il codice facilmente modificabile ed avere una progettazione e documentazione semplicemente rettificabili;
- **Stabile**: l'evitare effetti non desiderati a seguito di modifiche;
- **Testabile**: poter essere facilmente testato, per la validazione delle modifiche apportate;
- **Conforme alla manutenibilità**: l'adesione agli standard in merito alla manutenibilità.

A.1.1.6 Portabilità
Caratteristica che permette al prodotto di poter essere utilizzato anche dopo essere stato spostato in un ambiente di lavoro differente che sia di tipologia hardware o software.

- **Adattabile**: il potersi adattare a differenti ambienti operativi, senza la necessità di modifiche;
- **Installabile**: la possibilità di essere installato in un determinato ambiente;
- **Coesistente**: lo stare in un ambiente condiviso con altre applicazioni e risorse comuni;
- **Sostituibile**: la capacità di poter sostituire un altro software svolgendo gli stessi compiti nello stesso ambiente;
- **Conforme alla portabilità**: l'adesione agli standard in merito alla portabilità.

A.1.2 Metriche di qualità esterne
La qualità esterna è l'insieme di tutte le caratteristiche del prodotto software da un punto di vista esterno; viene generalmente misurata e valutata eseguendo il software ed effettuando i test in un ambiente simulato con dati simulati utilizzando metriche esterne. Durante i test, la maggior parte degli errori dovrebbero essere scoperti e corretti, tuttavia non sempre tali errori vengono evidenziati. Poiché, è difficile correggere l'architettura del software o altri aspetti fondamentali della progettazione del software, la progettazione fondamentale di solito rimane invariata durante i test.
Viene rilevata tramite analisi dinamica. Idealmente la qualità esterna determina la qualità in uso.

A.1.3 Metriche di qualità interne

La qualità interna è l'insieme delle caratteristiche del prodotto software da un punto di vista interno. La qualità interna viene misurata e valutata rispetto ai requisiti di qualità interna. I dettagli sulla qualità del prodotto software possono essere migliorati durante l'implementazione, la revisione del codice e i test, ma la natura fondamentale della qualità del prodotto software rappresentata dalla qualità interna rimane invariata se non ri-progettata.

A.1.4 Metriche di qualità in uso

Qualità in uso è la visione dell'utente riguardo la qualità del prodotto software quando viene utilizzato in un ambiente e in un contesto specifico di utilizzo. Misura la metrica con cui gli utenti possono raggiungere i propri obiettivi in un determinato ambiente, piuttosto che misurare le proprietà del software stesso.

Il livello di qualità nell'ambiente degli utenti può essere diverso da quello nell'ambiente degli sviluppatori, a causa delle differenze tra le esigenze e le capacità dei diversi utenti e le differenze tra i diversi ambienti hardware e di supporto. L'utente valuta solo quegli attributi del software, che vengono utilizzati per i suoi compiti. A volte, gli attributi software specificati da un utente finale durante la FASEge di analisi dei requisiti, non soddisfano più i requisiti dell'utente quando il prodotto è in uso, a causa della modifica dei requisiti dell'utente e della difficoltà di specificare le esigenze implicite.

Le metriche vengono applicate solo al prodotto finito ed in uso in condizioni reali. Tale qualità viene raggiunta solo se vengono raggiunti i livelli di qualità interna e qualità esterna.

A.2 ISO/IEC 15504

Il modello *ISO/IEC 15504* conosciuto come SPICE, acronimo di Software Process Improvement and Capability Determination è il modello standard per la valutazione oggettiva della qualità dei processi di sviluppo del software. Ogni valutazione permette di determinare il livello di maturità raggiunto e la capacità di ogni progetto tramite degli attributi, inoltre lo studio dei range risultanti permette di definire il margine di miglioramento.
I risultati delle singole valutazioni devono essere ripetibili, oggettivi e comparabili, perché possano contribuire al miglioramento dei processi.
Gli attributi delle capacità di processo sono:

- **Process performance**: misura del raggiungimento degli obiettivi fissati;
- **Performance management**: misura del grado di organizzazione con cui sono raggiunti gli obiettivi fissati;
- **Work product management**: misura della gestione corretta dei prodotti in merito alla documentazione, controllo e verifica;
- **Process definition**: misura dell'applicazione dello standard;
- **Process deployment**: misura del rilascio e distribuzione di un processo standard definito in modo tale da ottenere sempre gli stessi risultati;
- **Process measurement**: indica il grado in cui i risultati delle misure sono utilizzati per garantire che il processo raggiunga i suoi obiettivi;
- **Process control**: misura se il processo è stabile, capace e predicibile entro certi limiti;

- **Process innovation**: misura delle modifiche identificate da apportare al processo in FASEGE di analisi delle performance e in studio di approcci innovativi;
- **Process optimization**: misura dell'impatto dei cambiamenti dell'organizzazione, delle performance e della definizione del processo in funzione del raggiungimento degli obiettivi di miglioramento dei processi.

I livelli di misurazione degli attributi sono:

- **N - not implemented** (0-15%): il processo non possiede l'attributo o dimostra gravi carenze;
- **P - partially implemented** (15-50%): presenza di un approccio SISTEMATICOGE volto al possesso di un attributo già significativamente ottenuto, ma possiede aspetti non ancora prevedibili;
- **L - largely implemented** (50-85%): presenza di un approccio sistematico volto al possesso di un attributo già significativamente ottenuto, ma l'attuazione varia tra attività diverse;
- **F - fully implemented** (85-100%): attributo completamente ottenuto, seguendo un approccio sistematico e un'attuazione corretta per tutte le unità.

Basandosi sulla classificazione degli attributi, i livelli di capacità di un processo sono:

1. **Incomplete**: il processo è incompleto in quanto non è stato implementato o fallisce nel raggiungimento dell'obiettivo. Nessun attributo associato;
2. **Performed**: il processo è stato implementato e ha successo nel raggiungere l'obiettivo. L'attributo associato a questo livello è "process performance";
3. **Managed**: il processo che aveva già raggiunto il livello *performed*, è stato implementato ulteriormente con più organizzazione, tramite pianificazione, controllo e correzioni sui prodotti sicuri. Gli attributi associati a questo livello sono: "performance management" e "work product management";
4. **Established**: il processo che aveva già raggiunto il livello *managed*, è stato implementato ulteriormente come processo che è in grado di raggiungere sempre gli stessi risultati. Gli attributi associati a questo livello sono "process definition" e "process distribution";
5. **Predictable**: il processo che aveva già raggiunto il livello *established*, è stato implementato ulteriormente operando entro limiti definiti per raggiungere i risultati ambiti. Gli attributi associati a questo livello sono "process control" e "process measurement";
6. **Optimizing**: il processo che aveva già raggiunto il livello *predictable*, è stato implementato ulteriormente applicando il miglioramento continuo per raggiungere gli obiettivi di progetto/aziendali. Gli attributi associati a questo livello sono "process change" e "process improvement".

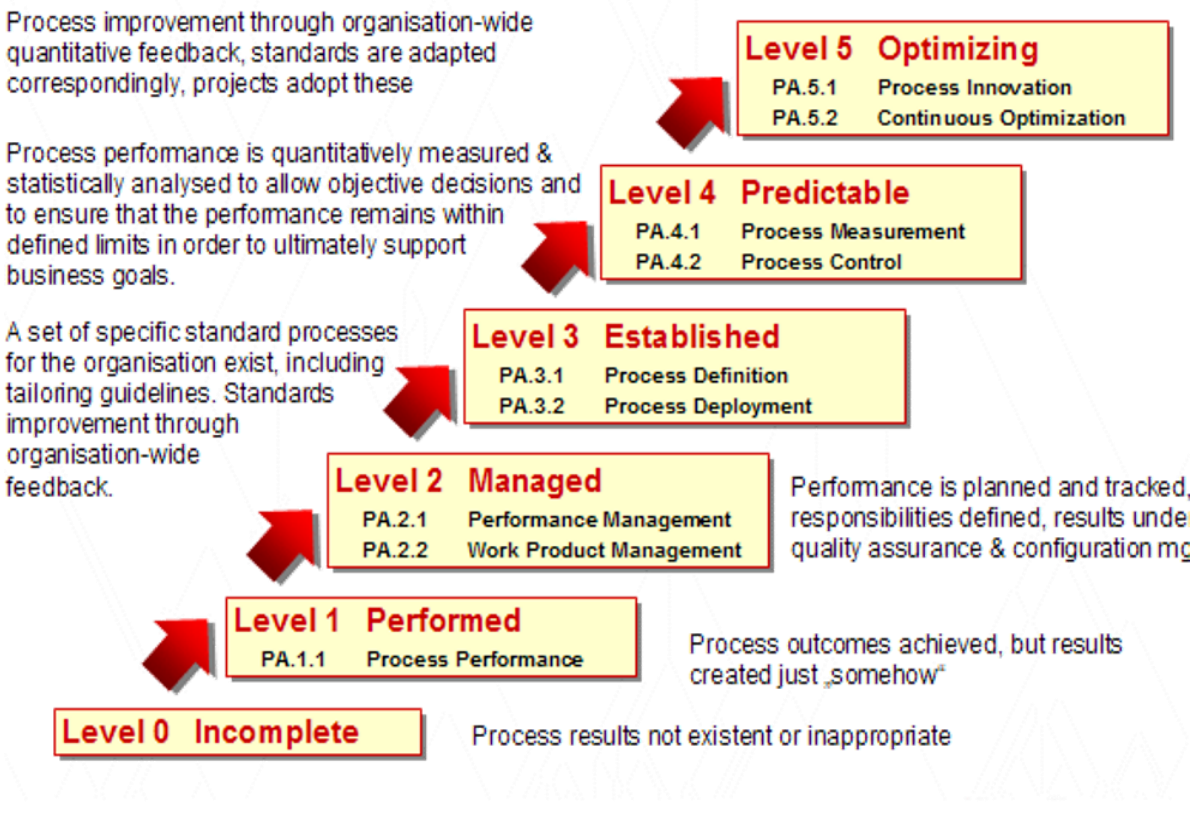

FIGURA 1 - SPICE CAPABILITY (FONTE: HM&S)

A.3 Ciclo di Deming

Il CICLO DI DEMINGGE (o PDCAGE, acronimo di Plan-Do-Check-Act) è un metodo di gestione iterativo, suddiviso in quattro fasi finalizzate al controllo dei processi e del miglioramento continuo della qualità dei processi e dei prodotti.

Come specificato dall'acronimo le quattro fasi di cui è costituito sono:

- **Plan**: FASEGE dedicata alla pianificazione degli obiettivi di miglioramento. In questa fase vengono definite le attività da svolgere, le risorse da assegnarvi e le scadenze;
- **Do**: fase di attuazione delle attività pianificate;
- **Check**: fase dedicata alla verifica per l'accertamento delle attività svolte nella fase *Do* rispetto alle attività pianificate nella fase *Plan* e all'esito ottenuto, se positivo o negativo;
- **Act**: fase di attuazione per standardizzare i processi che hanno subito modifiche, che si sono ritenute positive.

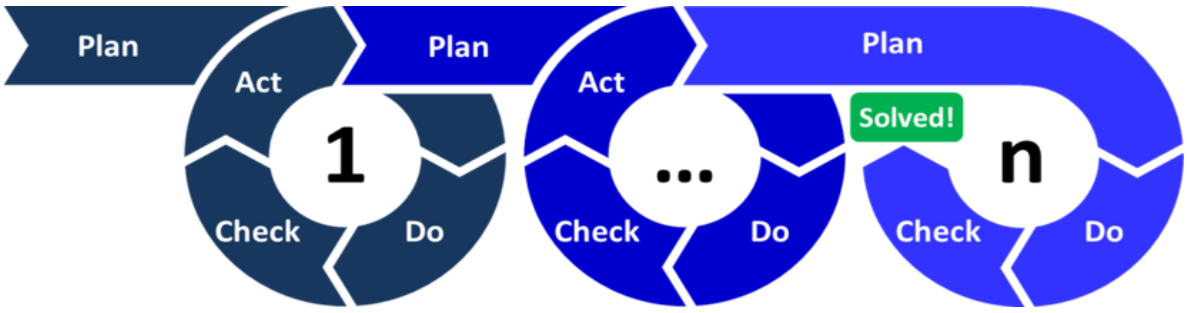

FIGURA 2 - RAPPRESENTAZIONE PDCA (FONTE: WIKIPEDIA)