IronWorks

Utility per la Costruzione di Software Robusto

swearoncode@gmail.com

Norme di Progetto

Versione	4.0.0
Redattori	Anna Poletti, Sharon Della Libera Francesco Sacchetto, Mirko Gibin Stefano Nordio, Antonio Moz
Verificatori	Anna Poletti
Responsabili	Mirko Gibin
Uso	Interno
Distribuzione	Gruppo Swear on Code Prof. Tullio Vardanega Prof. Riccardo Cardin

Descrizione

Questo documento contiene strumenti, convenzioni e norme adottate dal gruppo Swear on Code per lo svolgimento del progetto **IronWorks**.

Registro delle modifiche

Versione	Data	Autori	Ruolo	Descrizione
4.0.0	2018/08/22	Mirko Gibin	Responsabile	Approvazione
3.1.0	2018/07/29	Anna Poletti	Verificatore	Verifica
3.0.2	2018/07/27	Mirko Gibin	Amministratore	Stesura §2.2.3.3.3
3.0.1	2018/07/27	Mirko Gibin	Amministratore	Modifica §2.2.3.2.3, §2.2.3.2.4, §2.2.3.3.9
3.0.0	2018/07/12	Sharon Della Libera	Responsabile	Approvazione
2.1.0	2018/06/21	Anna Poletti	Verificatore	Verifica
2.0.5	2018/06/20	Stefano Nordio	Amministratore	Modifica §3.3.3.1.7
2.0.4	2018/06/20	Stefano Nordio	Amministratore	Stesura §2.2.3.3.5
2.0.3	2018/06/20	Antonio Moz	Amministratore	Stesura §2.2.3.3.4
2.0.2	2018/06/20	Antonio Moz	Amministratore	Modifica §2.2.3.2.5
2.0.1	2018/06/20	Stefano Nordio	Amministratore	Modifica §2.1
2.0.0	2018/06/07	Antonio Moz	Responsabile	Approvazione
1.2.0	2018/05/13	Stefano Nordio	Verificatore	Verifica
1.1.4	2018/05/12	Francesco Sacchetto	Amministratore	Stesura §3.5.3.3
1.1.3	2018/05/10	Mirko Gibin	Amministratore	Modifica §3.3, §3.4, §3.5
1.1.2	2018/05/09	Mirko Gibin	Amministratore	Stesura §2.2.3.3
1.1.1	2018/05/07	Francesco Sacchetto	Amministratore	Stesura §2.2.3.2
1.1.0	2018/05/05	Stefano Nordio	Verificatore	Verifica
1.0.4	2018/05/02	Francesco Sacchetto	Amministratore	Stesura §2.1.3.1
1.0.3	2018/04/27	Mirko Gibin	Amministratore	Stesura Procedure per i Processi
1.0.2	2018/04/25	Mirko Gibin	Amministratore	Stesura §3.3.3.1
1.0.1	2018/04/25	Francesco Sacchetto	Amministratore	Stesura §3.2
1.0.0	2018/04/07	Francesco Sacchetto	Responsabile	Approvazione
0.1.0	2018/03/16	Stefano Nordio	Verificatore	Verifica

0.0.10	2018/03/15	Sharon Della Libera	Amministratore	Modifica §4
0.0.9	2018/03/15	Anna Poletti	Amministratore	Modifica §4
0.0.8	2018/03/14	Francesco Sacchetto	Amministratore	Stesura §4
0.0.7	2018/03/13	Francesco Sacchetto	Amministratore	Modifiche §3
0.0.6	2018/03/13	Anna Poletti	Amministratore	Modifiche §3
0.0.5	2018/03/13	Sharon Della Libera	Amministratore	Stesura §3
0.0.4	2018/03/12	Sharon Della Libera	Amministratore	Modifiche §2
0.0.3	2018/03/12	Anna Poletti	Amministratore	Stesura §2
0.0.2	2018/03/12	Francesco Sacchetto	Amministratore	Stesura §1
0.0.1	2018/03/12	Francesco Sacchetto	Amministratore	Creazione del documento

Tabella 1: Storico versioni del documento

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del Documento

Definizione di una struttura per garantire un linguaggio comune basato su processi, attività, compiti e risultati prodotti. A tal fine viene adottato lo *Standard*$_G$ *ISO*$_G$/*IEC*$_G$ 12207:2008.

1.2 Ambiguità

Al fine di dipanare qualsiasi dubbio o ambiguità relativa al linguaggio impiegato nel documento viene fornito il *Glossario v3.0.0*, documento contenente la definizione di tutti i termini scritti in corsivo e marcati con una 'G' pedice.
Le guide citate vengono intese come estensione delle norme di progetto.

1.3 Natura Incrementale del Documento

Per la stesura dei documenti viene adottato il modello incrementale: dopo ogni revisione si effettuano correzioni e aggiornamenti.
Per questo motivo alcuni processi, attività e/o compiti non verranno trattati nella versione attuale del documento, in quanto inerenti a periodi di sviluppo successivi.
La correzione e l'incremento dei documenti ricoprono la primaria attività svolta all'inizio di ogni revisione come indicato dal *Piano di Progetto v4.0.0*. In tal modo il gruppo mira ad ottenere una completa normatura dei singoli compiti ed attività antecedentemente alla loro esecuzione.

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- *ISO*$_G$/*IEC*$_G$ *12207*:
  https://en.wikipedia.org/wiki/ISO/IEC_12207
- *ISO*$_G$/*IEC*$_G$ *9001*:
  http://www.colonese.it/00-Manuali_Pubblicatii/06-Qualit%E0Software_v2.pdf, *pagine 9-12, 17-23*;
- *Capitolato*$_G$ *C5 - IronWorks: utilità per la costruzione di software robusto*:
  http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/C5.pdf
- *VE_2018-05-02*

1.4.2 Riferimenti Informativi

- *Glossario v3.0.0*;
- *Guida Git*$_G$ *v1.0.0*;
- *Guida LaTeX*$_G$ *v1.0.0*;

- *Slides del Corso*:
  - Analisi dei Requisiti:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L08.pdf;
  - Progettazione:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L10.pdf;
  - Verifica e Validazione:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L17.pdf.
- *Slides del Corso - UML*$_G$:
  - *Diagrammi delle classi*$_G$:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/E03.pdf;
  - *Diagramma dei package*$_G$:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/E04.pdf;
  - *Diagramma di sequenza*$_G$:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/E05.pdf;
  - Diagramma di attività:
    http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/E06.pdf.

2 Processi primari

2.1 Processo di Fornitura

2.1.1 Scopo e Aspettative

Il processo ha lo scopo di fornire al *proponente*$_G$ il prodotto che soddisfa le sue necessità e i requisiti concordati.
Il gruppo si propone di mantenere un dialogo costante col *proponente*$_G$ in modo da aggiornarlo sull'avanzamento e avere un riscontro sull'andamento del progetto.

2.1.2 Descrizione

In questo processo si definisce il contratto col *committente*$_G$ Prof. Tullio Vardanega e Prof. Riccardo Cardin per diventare *fornitori*$_G$ del *proponente*$_G$ Gregorio Piccoli (a nome dell'azienda Zucchetti S.p.A.).
In questa fase il processo di fornitura prevede la consegna della documentazione finale per la Revisione di Accettazione, composta di:

- Documenti interni:
  - *Norme di Progetto v4.0.0*;
  - Verbali interni.
- Documenti esterni:
  - *Analisi dei Requisiti v4.0.0*;
  - *Glossario v3.0.0*;
  - *Manuale Sviluppatore*$_G$ *v2.0.0*;
  - *Manuale Utente*$_G$ *v2.0.0*;
  - *Piano di Progetto v4.0.0*;
  - *Piano di Qualifica v4.0.0*;
  - Verbali esterni.

Oltre ai documenti si consegna su supporto fisico il prodotto sviluppato.

2.1.3 Attività

2.1.3.1 Studio di Fattibilità

Gli *Analisti*, dopo aver analizzato i *capitolati*$_G$ in ingresso per valutarne vantaggi e svantaggi, redigono il documento *Studio di Fattibilità*.
La decisione del *capitolato*$_G$ da sviluppare viene presa da tutti i componenti del gruppo tramite discussioni collettive e scambi di opinioni.
Per ogni progetto proposto il gruppo valuta:

- **Dominio tecnologico e applicativo**: si prendono in considerazione le conoscenze tecnologiche dei membri del gruppo tenendo conto di eventuali esperienze positive o negative a riguardo e l'interesse del gruppo rispetto alle tematiche del *capitolato*$_G$;
- **Requisiti richiesti**: si analizzano quantità e qualità dei requisiti obbligatori tenendo conto del tempo a disposizione e dei costi da affrontare;
- ***Rischi***$_G$: si prendono in considerazione eventuali problemi o difficoltà che potrebbero verificarsi durante lo sviluppo del progetto.
- **Valutazione finale**: dopo un'attenta analisi dei punti precedenti viene data una valutazione generale del *capitolato*$_G$, considerando *rischi*$_G$ e benefici che si possono riscontrare nella realizzazione del progetto.

2.1.3.2 Piano di Progetto

Il *Responsabile di Progetto*, coadiuvato dagli *Amministratori*, istanzia i processi e le attività, pianificando i tempi e i costi per lo svolgimento di queste.
La pianificazione stabilita viene riportata nel documento *Piano di Progetto* con opportuni grafici in modo da risultare di facile comprensione.
In particolare i *Responsabili*:

- Individuano e analizzano i *rischi*$_G$, definendo delle procedure per ridurre eventuali effetti negativi;
- Collocano temporalmente le attività, definendo quindi le date di inizio e fine di queste, tenendo conto di alcuno *giorni di slack*$_G$;
- Suddividono il lavoro per ruoli e li assegnano alle persone incaricate;
- Stilano un prospetto economico, indicando le ore da impiegare per i singoli ruoli;

Alla fine di ogni revisione, il *Responsabile* riporta il consuntivo di periodo. Questo permette di confrontare ciò che si è pianificato è ciò che si è fatto in termini di costi e tempi.

2.1.3.3 Piano di Qualifica

Gli *Amministratori* definiscono le strategie per perseguire la qualità nelle *Norme di Progetto* e identificano le metriche per valutare i processi e i prodotti.
È compito dei *Verificatori* invece definire gli obiettivi metrici e riportare i risultati ottenuti dalle misurazioni effettuate e dai test nel *Piano di Qualifica*.
Questo serve per:

- Assicurare la qualità dei processi;
- Assicurare la qualità di prodotto;

2.1.3.4 Collaudo e Consegna

La consegna del prodotto segue al collaudo di questo in presenza del *proponente*$_G$ e del *committente*$_G$.
Il gruppo deve assicurarsi che il software da consegnare sia corretto, completo e affidabile e per questo deve:

- Eseguire tutti i test di validazione presentati nel *Piano di Qualifica*;
- Soddisfare i requisiti obbligatori e desiderabili secondo gli obiettivi metrici riportati nel *Piano di Qualifica*.

2.2 Processo di Sviluppo

2.2.1 Scopo e Aspettative

Lo scopo principale del processo è sviluppare un prodotto software finale conforme alle aspettative di *proponente*$_G$ e *committente*$_G$.
A tal fine si definiscono le norme da seguire nello svolgimento delle varie attività e dei relativi compiti.

2.2.2 Descrizione

Questo processo segue lo *standard*$_G$ *ISO*$_G$/*IEC*$_G$ 12207:2008.
Vengono trattate le attività di *Analisi dei Requisiti*, *Progettazione* e *Codifica*.

2.2.3 Attività

2.2.3.1 Analisi dei Requisiti

2.2.3.1.1 Scopo e Aspettative

Questa attività serve ad individuare tutti i requisiti del progetto per identificare e tradurre le esigenze del *proponente*$_G$.

2.2.3.1.2 Descrizione

Durante l'analisi dei requisiti si analizzano:

- *Capitolato*$_G$ d'appalto;
- Verbali interni;
- Verbali esterni, che riassumono gli incontri con il *proponente*$_G$ e gli eventuali incontri con il *committente*$_G$;
- Eventuali discussioni interne.

Viene definita una nomenclatura per identificare univocamente ogni requisito e facilitarne il tracciamento.
Il risultato di questa attività è il documento *Analisi dei Requisiti v4.0.0*.
Tale documento si propone come accordo tra Gregorio Piccoli, Zucchetti S.p.A.ed il gruppo riguardo ai requisiti e alle funzionalità del prodotto.

2.2.3.1.3 Procedura

Dal *capitolato*$_G$ e dai vari verbali interni ed esterni si ricavano i requisiti e i relativi *casi d'uso*$_G$, i quali sono tracciati in una tabella per assicurare una corretta e completa futura implementazione.
In particolare, dopo un'attenta lettura del testo del *capitolato*$_G$, vengono individuati e classificati i requisiti richiesti dal *proponente*$_G$.
Questi requisiti, attraverso discussioni creative e collaborative, vengono analizzati e suddivisi in sotto-requisiti al fine di comprendere nel dettaglio le funzionalità implicite.
Dopo aver compreso a fondo le richieste è utile richiedere un colloquio con il *proponente*$_G$ per esprimere eventuali dubbi e cogliere eventuali suggerimenti e preferenze.
Una volta eseguita la suddivisione dei requisiti, descritta nel paragrafo successivo, si procede con l'identificazione dei *casi d'uso*$_G$.
Come *best practice*$_G$ interna si procede nel seguente modo:

- Identificare *attori*$_G$ e relative responsabilità;
- Identificare obiettivi per ogni *attore*$_G$;
- Valutare *attore*$_G$ e *casi d'uso*$_G$, partendo dalle funzionalità principali (kite-level), evidenziandone poi le sotto-funzionalità (sea-level) ed infine scendendo nel dettaglio (fish-level);
- Trovare inclusioni, estensioni e poi generalizzazioni.

2.2.3.1.4 Suddivisione dei Requisiti

I requisiti vengono suddivisi per:

- Tipologia:
  - Funzionali (F);
  - Prestazionali (P);
  - Di Qualità (Q);
  - Di Vincolo (V).
- Priorità:
  - Obbligatorio (O);
  - Desiderabile (D);
  - Facoltativo (F).

Ogni requisito viene identificato rispettando il seguente formalismo:

R{A}{B}{Gerarchia}

- **A**: tipologia (F, P, Q, V);
- **B**: priorità (O, D, F);

- **Gerarchia**: identificazione numerica a livelli:
  - Livello generale;
  - Livello specifico;
  - Eventuali sottolivelli.

La tabella del tracciamento da un requisito ai relativi *casi d'uso*$_G$ contiene le seguenti informazioni:

- Identificativo del requisito;
- Priorità;
- Breve descrizione;
- Fonti (origini del requisito e *casi d'uso*$_G$ correlati).

2.2.3.1.5 Qualità dei Requisiti

I requisiti fissati dovranno essere:

- **Non ambigui**: devono avere una sola interpretazione formale;
- **Corretti**: devono riguardare funzionalità effettivamente richieste dal *proponente*$_G$ o necessarie all'*utente*$_G$ finale;
- **Completi**: devono descrivere tutte le caratteristiche richieste;
- **Verificabili**: si deve verificare che il sistema soddisfi ogni requisito;
- **Consistenti**: nessun requisito deve essere in conflitto con altri requisiti;
- **Modificabili**: eventuali modifiche devono preservare la consistenza e la completezza;
- **Tracciabili**: l'origine dei requisiti deve essere chiara;
- **Ordinati per rilevanza**: per definire delle priorità dei requisiti.

2.2.3.1.6 Casi d'Uso

Ogni *caso d'uso*$_G$ è redatto seguendo la seguente struttura:

- Nomenclatura;
- Nome;
- Eventuale diagramma *UML*$_G$ (versione 2.0);
- *Attori*$_G$ primari e secondari;
- Pre e Post condizioni;

- Scenario principale;
- Eventuali generalizzazioni;
- Eventuali scenari alternativi.

La nomenclatura adotta la seguente codifica:

UC{A}.{B}.{C}.{D}

- **A**: Numero del livello generale;
- **B**: Numero del livello specifico;
- **C**: Numero del livello del dettaglio;
- **D**: Numero del livello di altri sotto-dettagli.

2.2.3.1.7 Strumenti

Come strumento per la creazione dei diagrammi *UML*$_G$ viene usato *PragmaDB*$_G$.
La scelta di questo strumento deriva da tre fattori principali:

- Include un sistema di tracciamento dei requisiti;
- Genera automaticamente il codice *LaTeX*$_G$;
- Genera automaticamente i diagrammi dei *casi d'uso*$_G$.

Il gruppo ha adattato tale software alle proprie esigenze ed ha aggiunto un *modulo*$_G$ per la generazione automatica dei diagrammi dei *casi d'uso*$_G$, il quale si appoggia al *tool*$_G$ *open source*$_G$ *PlantUML*$_G$ per ottenere tale risultato.

2.2.3.2 Progettazione

2.2.3.2.1 Scopo e Aspettative

L'attività di progettazione si pone l'obiettivo di definire tutti gli aspetti necessari per la realizzazione del software.
Collocandosi ad un livello più astratto della codifica permette di costruire una solida base per dominare la complessità del sistema da implementare.

2.2.3.2.2 Descrizione

L'attività di progettazione viene svolta dai *Progettisti*, che hanno quindi il compito di progettare la struttura ed il comportamento del software a partire dai requisiti presenti nel documento *Analisi dei Requisiti v4.0.0*. Nello specifico, i *Progettisti* definiscono l'*architettura*$_G$ del software suddividendo il sistema in componenti, ovvero sottosistemi di minor complessità.
Le componenti sono a loro volta divise in unità. Questa ulteriore suddivisione delinea elementi che corrispondono alla più piccola quantità di software verificabile, prodotta da un

singolo *Programmatore*.
A loro volta ciascuna *componente*$_G$ è suddivisa in *moduli*$_G$, la più piccola *entità*$_G$ strutturale utilmente rappresentabile.

2.2.3.2.3 Procedura per la Progettazione Architetturale

I *Progettisti* si occupano di definire le componenti del sistema.
Ciò permette di:

- Individuare le componenti da realizzare;
- Definire ruoli e responsabilità per ogni *componente*$_G$;
- Definire le interazioni tra le componenti;
- Assicurarsi che ogni requisito sia soddisfatto da almeno una *componente*$_G$ e viceversa;
- Realizzare e documentare i test di integrazione, al fine di verificare il corretto funzionamento di più componenti integrati assieme.

Questa suddivisione permette di procedere con la progettazione dell'*architettura*$_G$ del software.
I *Progettisti* devono riunirsi per discutere l'*architettura*$_G$ più adatta. Devono comprendere a fondo i requisiti identificati nell'*Analisi dei Requisiti v4.0.0* e rappresentare attraverso i diagrammi dei package le macro componenti in cui si intende suddividere il sistema.
Sono inoltre fondamentali le dipendenze tra i package, che sanciscono le relazioni tra questi. Bisogna assolutamente evitare di creare dipendenze circolari poichè riducono la riusabilità.

2.2.3.2.4 Procedura per la Progettazione di Dettaglio

Per ogni *componente*$_G$ individuata durante la progettazione architetturale si deve procedere in questo modo:

- Suddividere la *componente*$_G$ in unità;
- Definire i ruoli di ogni unità;
- Definire le interazioni tra le varie unità;
- Assicurarsi che ogni requisito sia soddisfatto da almeno una delle unità e viceversa;
- Definire le interfacce che ogni *componente*$_G$ mette a disposizione dell'ambiente esterno.

Per ogni unità individuata si deve procedere in questo modo:

- Individuare i *moduli*$_G$ che implementano ciascuna unità;
- Definire i ruoli di ogni *modulo*$_G$;
- Definire ogni *modulo*$_G$ nel dettaglio;

- Assicurarsi che ogni requisito sia soddisfatto da almeno uno dei *moduli*$_G$ e viceversa.

Affinché la programmazione possa procedere in modo certo e disciplinato è necessario implementare e documentare i test di unità.
I *Progettisti* devono riunirsi per realizzare i *diagrammi delle classi*$_G$ da sottoporre ai *Programmatori* per l'implementazione.
Non è consigliato utilizzare un numero eccessivo di *design pattern*$_G$ per non vincolare troppo l'implementazione del codice. Ogni scelta va pesata e misurata e discussa con tutti i *Progettisti*.
Le classi devono essere pensate per essere facilmente manutenibili e quindi aperte all'estensione ma chiuse alle modifiche. Ciò implica in particolare l'adozione di gerarchie di classi.
Inoltre ognuna di queste deve avere una propria funzione e responsabilità ben definita.

2.2.3.2.5 Diagrammi

Per descrivere l'*architettura*$_G$ e il comportamento del sistema, i *Progettisti* utilizzano i seguenti diagrammi *UML*$_G$ (versione 2, con riferimento alle slides del corso):

- ***Diagrammi delle classi***$_G$: descrivono le interfacce degli elementi del sistema e le relazioni tra essi. Vengono costruiti secondo lo *standard*$_G$ *UML*$_G$, quindi con un rettangolo suddiviso in tre aree: nome, attributi e operazioni. Per uniformità, gli attributi si devono identificare con la visibilità, il nome e il tipo, mentre le operazioni con la visibilità, il nome, gli eventuali parametri passati e il tipo di ritorno. Le linee per collegare le classi devono essere utilizzate con attenzione. In particolare:
  - **Linea di dipendenza da 'A' a 'B' (freccia tratteggiata)**: si utilizza quando una classe lavora con un'altra. Solitamente è utilizzata quando la classe 'A' chiama e/o utilizza un'operazione di 'B' o crea istanze della classe 'B';
  - **Linea di associazione da 'A' a 'B' (freccia semplice)**: si utilizza quando una classe lavora strettamente con un'altra, in particolare quando 'A' ha fra i propri campi dati una o più istanze della classe 'B';
  - **Linea di aggregazione da 'A' a 'B' (freccia a diamante vuota)**: si utilizza quando 'A' è composta da elementi di 'B', i quali possono esistere anche in modo indipendente da 'A'.
  - **Linea di composizione da 'A' a 'B' (freccia a diamante piena)**: si utilizza quando 'A' è composta da elementi di 'B', i quali possono essere creati o distrutti solo da 'A'.
  - **Linea di eredità da 'A' a 'B' (freccia vuota)**: si utilizza quando un oggetto di 'A' è anche un oggetto di 'B'.
- ***Diagrammi dei package***$_G$: raggruppano i diagrammi *UML*$_G$ delle classi in una sola unità di livello più alto. Vengono rappresentati da rettangoli e da un'etichetta che ne identifica il nome. Secondo lo *standard*$_G$, una classe può appartenere ad un singolo pacchetto e ogni pacchetto può essere contenuto in altri pacchetti e le dipendenze tra package sono identificate da una linea tratteggiata;

- ***Diagrammi delle attività***$_G$: rappresentano il flusso di operazioni relativo ad un'unità, in particolare vengono utilizzati per descrivere la logica di un algoritmo. Pertanto gli elementi rappresentati devono essere letti in sequenza, a meno che non venga introdotto un "join" che individua l'esecuzione di due attività parallele. La notazione da utilizzare segue lo *standard*$_G$ *UML*$_G$;
- ***Diagrammi di sequenza***$_G$: descrivono uno scenario, dove le azioni sono disposte in sequenza e le varie scelte sono già state prese. Gli oggetti coinvolti verranno rappresentati tramite un rettangolo con un nome per identificarli. Sotto ad ogni istanza si utilizza una "linea della vita" tratteggiata, sormontata in alcuni tratti da una barra di attivazione che indica i momenti in cui l'oggetto è attivo. Anche la notazione di questi diagrammi deve seguire lo *standard*$_G$. Particolare attenzione va data all'utilizzo delle frecce: le frecce semplici indicano un evento asincrono, mentre quelle piene un evento sincrono.

2.2.3.2.6 Qualità dell'Architettura

L'*architettura*$_G$ definita deve perseguire le seguenti caratteristiche:

- **Sufficienza**: soddisfare tutti i requisiti;
- **Modularità**: essere suddivisa in parti chiare e ben distinte;
- **Robustezza**: reagire ad ogni input in modo corretto;
- **Affidabilità**: garantire una buona usabilità del prodotto;
- **Flessibilità**: permettere la modifica dell'*architettura*$_G$ a costo contenuto.

La scelta dei *design pattern*$_G$ è una *best practice*$_G$ consigliata rispetto alla creazione ex novo di schemi architetturali.

2.2.3.2.7 Strumenti

Per la produzione di diagrammi *UML*$_G$ viene utilizzato Lucidchart, servizio online che permette la realizzazione di diagrammi in modo semplice e veloce.

2.2.3.3 Codifica

2.2.3.3.1 Scopo e Aspettative

Lo scopo dell'attività di codifica è quello di realizzare il prodotto software richiesto concretizzando l'attività di progettazione attraverso la programmazione.
Al fine di migliorare la leggibilità del codice sorgente e renderlo più facilmente manutenibile vengono definite delle convenzioni.

2.2.3.3.2 Descrizione

L'attività di codifica consiste nel tradurre in un linguaggio di programmazione le parti dell'*architettura*$_G$ definite nella progettazione.
La parte *client*$_G$ è scritta in *HTML5*$_G$, *CSS*$_G$ e *JavaScript*$_G$.
La parte *server*$_G$ è scritta in linguaggio *JavaScript*$_G$.
Per un corretto utilizzo dei costrutti utilizzati dai linguaggi si fa riferimento alla documentazione riportata nel sito W3Schools, ed in caso di ulteriori necessità e chiarimenti si fa riferimento al *W3C*$_G$ o alla documentazione ufficiale della specifica tecnologia.

2.2.3.3.3 Procedura

I *Programmatori* ricevono l'*architettura*$_G$ di dettaglio dai *Progettisti* e implementano le classi utilizzando le convenzioni sotto riportate.
Nel caso si presentassero dei problemi nella codifica dovuti ad una *Progettazione* errata, i *Programmatori* devono informare i *Progettisti* e discutere con loro di una diversa implementazione.
Per quanto riguarda i nomi delle classi, degli attributi e dei metodi devono attenersi a quanto deciso dai *Progettisti* per mantenere la coerenza in caso di dipendenze.
Per l'algoritmica che riguarda l'implementazione dei metodi ogni *Programmatore* può scegliere quali costrutti utilizzare, cercando di sviluppare un codice corretto che rispetti gli obiettivi metrici descritti nel *Piano di Qualifica v4.0.0*.

2.2.3.3.4 Convenzioni

Le convenzioni definite per una stesura uniforme del codice sono le seguenti:

- **Intestazione e commenti**: per ogni *file*$_G$ prodotto si inserisce la seguente intestazione sotto-forma di commento:

  ```
  File_G: nome file_G
  Versione: versione attuale file_G
  Autore: autore file_G
  Registro Modifiche: autore, data, descrizione modifica Descrizione
  della funzione del file_G
  ```

  Per ogni metodo invece è necessario fornire la seguente descrizione:

  ```
  Descrizione del metodo
  Elenco parametri: nome, tipo, breve descrizione
  ```

  Quando parti di codice non sono di immediata comprensibilità, il *Programmatore* è tenuto a documentarli con un commento in linea utilizzando le seguenti convenzioni:

  - Utilizzare frasi chiare e concise;
  - Porre il commento sopra al codice da documentare per garantire maggiore leggibilità;

  - Aggiornare il commento in caso di modifiche al codice.

- **Indentazione**: è richiesto di utilizzare esattamente 4 spazi, piuttosto che utilizzare il tab, che potrebbe essere interpretato in modo diverso da differenti *editor*$_G$;
- **Operatori**: gli operatori vanno preceduti e seguiti da uno spazio. Nel caso si utilizzi una virgola, lo spazio va inserito al seguito di essa;
- **Parentesi graffe**: per costrutti complessi è utile:
  - Porre la parentesi aperta alla fine della prima riga;
  - Usare uno spazio prima della parentesi aperta;
  - Mettere la parentesi chiusa in una nuova riga, senza lasciare spazi.
- **Nomenclatura**: gli elementi devono rispettare le seguenti convenzioni:
  - I nomi degli elementi sono scritti in lingua inglese;
  - I nomi di variabili e funzioni sono scritte con lo stile camelCase;
  - I nomi di variabili globali e costanti sono scritte con lo stile *UPPERCASE*$_G$;
  - Gli elementi devono avere un nome univoco che esprima al meglio il loro scopo;
  - Ogni metodo che realizza operazioni su uno specifico oggetto deve utilizzare la struttura *verboNome*;
  - Sono da evitare abbreviazioni o acronimi poco chiari.

2.2.3.3.5 Convenzioni Front-End

Per produrre l'interfaccia grafica del prodotto si adottano alcune best-practices del *Material Design*$_G$. Le icone devono essere semplici, intuitive, geometriche, pulite e piatte.
Ogni bottone:

- Deve essere facilmente individuabile tra gli elementi presenti nella schermata, compresi gli altri bottoni;
- L'azione corrispondente deve essere intuibile e non ambigua;
- A seconda dell'importanza dell'azione che comporta può essere essere testuale, contornato o riempito;
- Eventuale contenuto testuale deve essere sintetico e in-line;
- È consentito affiancare al testo massimo un'icona.

I colori indicano gli elementi interattivi della pagina e il loro utilizzo è mirato a risaltarne le funzionalità più importanti, senza comprometterne la leggibilità.
È presente una distinzione tra il colore primario, ovvero quello che appare più frequentemente, e il colore secondario, utilizzato per accentuare le caratteristiche del prodotto e renderlo distinguibile.
Di entrambe le tipologie possono essere utilizzate le loro varianti chiare e scure.

2.2.3.3.6 Convenzioni Back-End

Il codice scritto in *NodeJS*$_G$ deve essere strutturato in classi e permettere l'estensione di questo grazie all'ereditarietà.
JavaScript$_G$ versione *ECMAScript 2017*$_G$ supporta l'utilizzo del comando `class` e del comando `extends`.
Per costruire una classe è necessario:

- Creare il *file*$_G$ (per convenzione il suo nome coincide con quello della classe al suo interno);
- Creare la classe nel seguente modo:

```
class A {
     constructor(parameter) {
          this.attribute=parameter;
     }
     method() {
          return;
     }
}
```

- Esportare la classe con:
  `module.exports=A;`

Per creare un oggetto 'A' si utilizzano le seguenti righe di codice:

- Per includere la classe è necessario:
  `var A=require("path-to/A.js");`
- Per creare il nuovo oggetto:
  `var object=new A(paramether);`

Le classi che ereditano dalla classe 'A' vengono costruite allo stesso modo, aggiungendo nel loro *file*$_G$:

- L'inclusione della classe base, attraverso il `require` ;
- Il comando `extends A` dopo la dichiarazione della classe derivata;
- La chiamata al costruttore della classe base nel costruttore della derivata.

2.2.3.3.7 Nomenclatura File

I *file*$_G$ contenenti il codice sorgente devono essere inseriti in cartelle rispettando l'*architettura*$_G$ del sistema.
I nomi dei *file*$_G$ devono essere univoci, chiari e coincisi.
Inoltre devono essere scritti con lo stile *lowercase*$_G$.

2.2.3.3.8 Versione

Il formalismo da inserire nell'intestazione di ogni *file*$_G$ per identificarne la versione è il seguente:

v{X}.{Y}

Dove:

- **X**: è l'indice principale della versione e rappresenta modifiche stabili e significative. Un incremento di tale indice azzera il valore dell'indice 'Y';
- **Y**: è l'indice che rappresenta le modifiche di minor *entità*$_G$.

2.2.3.3.9 Qualità Codifica

La codifica deve rispettare gli obiettivi di qualità definiti nel *Piano di Qualifica v4.0.0*.
Il *Programmatore* deve inoltre evitare la ricorsione, facendo il possibile per cercare di sostituirla con un metodo iterativo equivalente. Nel caso in cui la ricorsione sia necessaria, il *Programmatore* è tenuto a fornire una prova di terminazione dell'algoritmo.
Per una maggiore manutenibilità è opportuno evitare la duplicazione del codice affidandosi a funzioni esterne che possono essere invocate in base alle necessità.
Evitare anche di creare variabili o costrutti non necessari che inquinano il codice.

2.2.3.3.10 Strumenti

L'*IDE*$_G$ utilizzato in fase di codifica è *IntelliJ IDEA*$_G$.
Questo *IDE*$_G$ è compatibile con Linux e Windows e viene utilizzato per la codifica in *JavaScript*$_G$ e *TypeScript*$_G$.

3 Processi di Supporto

3.1 Processo di Documentazione

3.1.1 Scopo e Aspettative

Questo processo definisce lo sviluppo e la manutenzione delle informazioni prodotte e registrate relativamente al software.

3.1.2 Descrizione

Il gruppo stabilisce regole e strutture per redigere in modo uniforme i documenti facilitandone la comprensione.

3.1.3 Struttura

3.1.3.1 Nome del Documento

- Prima lettera maiuscola di ogni parola;
- Nome del documento con spaziature;
- Versione inserita dopo il nome, divisa da una spaziatura.

3.1.3.2 Prima Pagina

Impostata secondo il template LaTeX$_G$ a disposizione del gruppo, fornisce le seguenti informazioni centrate orizzontalmente:

- Nome del *capitolato*$_G$, seguito da una breve descrizione visibile come primo elemento;
- Logo del gruppo, visibile dopo il nome del progetto, centrato orizzontalmente;
- E-mail del gruppo, visibile appena al di sotto del logo, centrata orizzontalmente;
- Titolo del documento;
- Tabella informativa contenente le seguenti informazioni:
  - Versione;
  - Redattori;
  - Verificatori;
  - Responsabili;
  - Uso;
  - Distribuzione.
- Descrizione sintetica del documento.

3.1.3.3 Registro delle Modifiche

Consiste in una tabella contenente le seguenti informazioni nell'ordine riportato:

- Versione;
- Data;
- Autori;
- Ruolo;
- Descrizione dell'attività svolta.

3.1.3.4 Indice

Agevola la consultazione e permette una lettura ipertestuale.
L'indice viene suddiviso in sezioni e sottosezioni numerate secondo il seguente formalismo:

{W}.{X}.{Y}.{Z}

- **W**: sezione generale;
- **X**: sottosezione generale;
- **Y**: sottosezione particolare;
- **Z**: paragrafo.

3.1.3.5 Introduzione

- Scopo del documento;
- Ambiguità riferite al *Glossario v3.0.0* e ad altre note del gruppo;
- Riferimenti informativi e normativi.

3.1.3.6 Contenuto

Ogni pagina deve rispettare i margini orizzontali e verticali previsti dal template.
Ad eccezione della prima, tutte le pagine devono avere un'intestazione ed un piè di pagina.
L'intestazione segue la seguente struttura:

- Logo del gruppo, posto a sinistra;
- Titolo della sezione corrente, posto a destra.

Il piè di pagina segue la seguente struttura:

- Nome e versione del documento, posti a sinistra;
- Numerazione progressiva della pagina rispetto al totale, posta a destra.

3.1.3.7 Note a Piè Pagina

Eventuali note a piè pagina interne al documento vanno inserite nella pagina corrente, in basso a sinistra.
Ogni nota deve riportare un numero e una descrizione.

3.1.4 Versionamento

Il versionamento dei documenti segue le norme definite nella sezione dedicata al processo di versionamento.
Durante la stesura del documento il cambio di versione avviene alla conclusione di una sezione da parte del redattore designato.
L'intervento di un diverso redattore sulla stessa sezione viene intesa come modifica e sancisce l'incremento del numero di versione.

3.1.5 Norme Tipografiche

3.1.5.1 Impaginazione

Ogni nuova sezione richiede una nuova pagina.

3.1.5.2 Stile del Testo

- Il grassetto viene utilizzato per marcare i titoli e gli elementi di un elenco puntato che riassumono il contenuto di tale voce;
- Il corsivo è utilizzato per marcare:
  - Parole contenute nel glossario;
  - Citazioni;
  - Riferimenti a specifici documenti;
  - Ruoli di progetto;
  - Processi, attività e compiti di progetto.
- Il maiuscolo è consentito per scrivere sigle e acronimi;
- I comandi, intesi come codice, devono essere con font "monospace";
- Ogni parola presente nel glossario, ad ogni occorrenza in un documento, deve essere marcata in corsivo e con una 'G' maiuscola a pedice.

3.1.5.3 Elenchi

Tutti gli elenchi presenti nei documenti sono puntati e non numerati, poiché i numeri potrebbero far intendere al lettore un presunto ordine all'interno delle voci.
Tutti gli elenchi sono rappresentati da un pallino pieno nel primo livello, da un trattino nel secondo e da un asterisco nel terzo.

Gli elenchi puntati servono ad esprimere concetti in modo chiaro e sintetico, evitando periodi lunghi e discorsivi.
Ogni voce dell'elenco inizia con la lettera maiuscola e termina con un punto e virgola, ad eccezione dell'ultima voce che terminerà con un punto.

3.1.5.4 Sigle e Abbreviazioni

Si prevede l'utilizzo delle seguenti sigle e abbreviazioni:

- **RR**: *Revisione dei Requisiti*;
- **RP**: *Revisione di Progettazione*;
- **RQ**: *Revisione di Qualifica*;
- **RA**: *Revisione di Accettazione*;
- **Re**: *Responsabile*;
- **Am**: *Amministratore*;
- **An**: *Analista*;
- **Pt**: *Progettista*;
- **Pr**: *Programmatore*;
- **Ve**: *Verificatore*;
- ***PDCA***$_G$: *Plan-Do-Check-Act*.

3.1.5.5 Formati Comuni

Vengono utilizzati i seguenti formalismi:

- Date:

  {AAAA/MM/GG}

  - **AAAA**: rappresenta l'anno utilizzando quattro cifre;
  - **MM**: rappresenta il mese utilizzando due cifre;
  - **GG**: rappresenta il giorno utilizzando due cifre.

- Orari:

  {HH:MM}

  - **HH**: rappresenta l'ora utilizzando due cifre da 00 a 23;
  - **MM**: rappresenta i minuti utilizzando due cifre da 00 a 59.

3.1.5.6 Apici

Vengono utilizzati i singoli apici per marcare singole lettere, doppi invece per parole o frasi.

3.1.5.7 Nomi Ricorrenti

- **Nomi propri**: ogni nome proprio di persona deve essere scritto con il Nome seguito dal Cognome;
- **Nomi dei documenti**: ogni nome di documento seguito dalla versione viene scritto con l'iniziale maiuscola e marcato in corsivo;
- **Ruoli di progetto**: ogni nome di ruolo di progetto viene scritto con l'iniziale maiuscola e marcato in corsivo.

3.1.5.8 Paragrafi e Spaziature

Paragrafi e spaziature seguono le impostazioni definite nel template *LaTeX*$_G$.

3.1.6 Elementi Grafici

3.1.6.1 Tabelle

Ogni tabella deve essere centrata orizzontalmente e deve essere seguita da una breve descrizione del contenuto di essa.
Nella descrizione deve essere presente il numero della tabella incrementale in tutto il documento al fine di agevolarne il tracciamento.

3.1.6.2 Immagini

Ogni immagine deve essere centrata orizzontalmente ed inoltre, per migliorare la leggibilità del documento, deve essere separata in modo netto dai paragrafi precedenti e successivi.
Analogamente a quanto stabilito per le tabelle, le immagini devono essere seguite da una didascalia che ne contiene descrizione e rispettivo numero identificativo.
Tutti i diagrammi *UML*$_G$ vengono inseriti nel documento sotto forma di immagine.

3.1.7 Classificazione Documenti

3.1.7.1 Documenti Formali

Una versione di un documento viene definita formale quando viene approvata dal *Responsabile* di progetto, dopo aver superato la verifica e la valutazione.
Soltanto i documenti formali possono essere distribuiti all'esterno.

3.1.7.2 Documenti Informali

Fino all'approvazione del *Responsabile*, tutti i documenti sono da ritenersi informali e ad uso esclusivamente interno.

3.1.7.3 Verbali

Essendo scritti di volta in volta e riportando le decisioni di una riunione non possono adottare la nomenclatura a versionamento.
Viene semplicemente numerato in ordine cronologico e distinto in verbale interno ed esterno seguendo la sintassi:

V{X}{D}

- **X**: 'I' se interno, 'E' se esterno;
- **D**: data dell'incontro.

Per quanto detto precedentemente, non è necessario introdurre il registro delle modifiche e viene omesso anche l'indice.
Un verbale è così strutturato:

- Informazioni generali;
- Informazioni incontro;
- Data;
- Luogo;
- Ora;
- Partecipanti;
- Riferimenti;
- Ordine del giorno;
- Sviluppo dei vari punti indicati (se consiste in un verbale esterno, stilare le domande e le risposte ricevute);
- Riepilogo tracciamento decisioni.

3.1.8 Approvazione Documenti

Ogni documento formale viene sottoposto al *Responsabile* che incarica i *Verificatori* di controllare la forma e il contenuto appoggiandosi alle norme del processo di documentazione.
Se vengono rilevati errori i *Verificatori* rigettano il documento, notificando ai redattori le correzioni da apportare.
Questa operazione viene fatta iterativamente fino al raggiungimento della corretta stesura del documento.
L'approvazione finale spetta al *Responsabile*.

3.1.9 Strumenti

3.1.9.1 LaTeX

I documenti vengono scritti utilizzando il linguaggio *LaTeX*$_G$, che offre funzionalità utili e pratiche al fine di ottenere una struttura più formale e pulita.
Inoltre offre la possibilità di ottenere uno *standard*$_G$ interno, grazie ai template, per una stesura più veloce.
Per facilitarne l'uso sono definiti dei comandi personalizzati forniti e spiegati nel documento interno *Guida LaTeX*$_G$ *v1.0.0*.
Per quanto riguarda la scelta dell'*editor*$_G$, si lascia liberi i componenti di utilizzare quello a loro più comodo.

3.1.9.2 Google Docs

Per una rapida stesura di bozze e documenti informali si sceglie di utilizzare *Google Docs*$_G$ per la possibilità di una visione e consultazione condivisa.

3.1.9.3 Script Glossario

Per automatizzare la marcatura dei termini a glossario è stato creato uno shell script che viene eseguito in prossimità delle revisioni.

3.2 Processo di Gestione della Configurazione

3.2.1 Scopo e Aspettative

Lo scopo di questo processo è di identificare, controllare, manutenere e verificare le versioni dei prodotti offerti.

3.2.2 Descrizione

Tale processo è regolato dallo *standard*$_G$ *ISO*$_G$ 10007:2017. Tra le attività dello *standard*$_G$ il gruppo prende in esame le seguenti:

- Identificazione della configurazione;
- Gestione dei cambiamenti.

3.2.3 Attività

3.2.3.1 Identificazione della Configurazione

3.2.3.1.1 Scopo e Aspettative

Lo scopo dell'attività è quello di identificare e attribuire dei numeri di versione ai prodotti da configurare, quindi etichettarli e inserirli in un sistema di archiviazione.

3.2.3.1.2 Descrizione

Il *controllo versione*$_G$ viene utilizzato in modo tale da rispettare la storia e l'evoluzione di ogni elemento durante tutto il progetto.
Questo permette di visionare ed eventualmente ripristinare *file*$_G$ generati in passato in caso di necessità.

3.2.3.1.3 Numero di Versione

Per ogni prodotto sottoposto a configurazione viene adottato il seguente formalismo:

v{X}.{Y}.{Z}

- **X**: identifica una *baseline*$_G$ di configurazione, ovvero l'attuale versione del prodotto approvata;
- **Y**: identifica il numero di controlli eseguiti dal verificatore prima dell'approvazione finale;
- **Z**: identifica i cambiamenti apportati al prodotto ancora da verificare.

3.2.3.2 Gestione dei Cambiamenti

3.2.3.2.1 Scopo e Aspettative

Lo scopo di questa attività è quello di regolare i cambiamenti apportati ai documenti o al codice per evitare modifiche incongruenti o errate.

3.2.3.2.2 Descrizione

La modifica di qualsiasi prodotto del progetto deve essere valutata, verificata e approvata dai membri incaricati.
Ciò garantisce una corretta gestione della configurazione e una riduzione dei *rischi*$_G$ dovuti a modifiche errate.

3.2.3.2.3 Procedura

Per garantire una buona gestione dei cambiamenti si attua la seguente procedura:

- La persona che apporta modifiche consistenti al prodotto (documento o codice) incrementa di un'unità la *componente*$_G$ 'Z' del formalismo sopra citato;
- Il *Verificatore* incaricato di controllare lo stato di avanzamento del prodotto analizza le modifiche apportate e in caso di valutazione positiva incrementa di un'unità la *componente*$_G$ 'Y' del formalismo sopra citato azzerando quindi la *componente*$_G$ 'Z'. Nel caso in cui la valutazione abbia esito negativo, il *Verificatore* si incarica di notificare all'autore della modifica i problemi rilevati;

- Il *Responsabile* ha il compito di approvare i prodotti con esito positivo nelle valutazioni del *Verificatore*, incrementando di un'unità la *componente*$_G$ 'X' del formalismo sopra citato.

Ognuna di queste azioni deve essere documentata nel registro delle modifiche presente in ogni documento.

3.2.4 Strumenti

Per quanto riguarda tale processo si è optato per l'uso di *Git*$_G$ appoggiandosi come servizio di *hosting*$_G$ a *GitHub*$_G$.
Lo strumento *Git*$_G$ archivia i cambiamenti apportati ai prodotti con una *commit*$_G$, identificandoli univocamente con una notazione propria e allegando un messaggio di archiviazione. Nei documenti tali *commit*$_G$ vengono riportate nel registro delle modifiche con il formalismo descritto sopra.

3.3 Processo di Verifica

3.3.1 Scopo e Aspettative

Il processo di verifica accerta che l'esecuzione delle attività attuate nel periodo in esame non abbia introdotto errori.
È costituito da attività e compiti atti a determinare se i requisiti del prodotto sono completi e corretti e se essi sono soddisfatti.
Il gruppo si aspetta di rispettare ogni norma definita in questo processo, per ottenere prodotti corretti e in linea con il *way of working*$_G$ prestabilito.
Il *Verificatore* ha il compito di seguire questo processo.

3.3.2 Descrizione

Il processo si suddivide in due attività:

- **Analisi**: si occupa dell'analisi del codice sorgente e della documentazione. Si suddivide in analisi statica e dinamica;
- **Test**: si occupa di testare il prodotto software.

3.3.3 Attività

3.3.3.1 Metriche

3.3.3.1.1 Scopo e Aspettative

Lo scopo di quest'attività è di definire le metriche per valutare il lavoro del gruppo. Tali metriche vengono applicate attraverso delle misurazioni, i cui risultati vengono riportati nel *Piano di Qualifica v4.0.0* per essere analizzati.

3.3.3.1.2 Descrizione

Le metriche vengono definite per monitorare:

- Processi;
- Prodotti;
- Progetto.

Tali metriche devono rispettare la seguente notazione:

MXY

In particolare:

- **X**: rappresenta il tipo di metrica, a scelta fra:
  - **PC**: metrica per i processi;
  - **SW**: metrica per il software;
  - **DC**: metrica per la documentazione;
  - **PG**: metrica per il progetto.
- **Y**: rappresenta il numero che identifica la metrica univocamente.

3.3.3.1.3 Procedura

Le misure vengono effettuate più volte durante la fase di revisione da parte del *Verificatore* utilizzando gli strumenti indicati in seguito.
I risultati intermedi vengono riportati nell'apposito grafico all'interno del *Piano di Qualifica v4.0.0*.
I risultati finali, ovvero quelli misurati a fine di ogni revisione, vengono riportati nell'apposita tabella all'interno del *Piano di Qualifica v4.0.0*.
Nel caso in cui i risultati finali risultino non conformi con gli obiettivi (ovvero inferiori al range accettabile) il *Verificatore* ha il compito di contattare il *Responsabile*.
Il *Responsabile* prenderà provvedimenti per migliorare la situazione nella successiva fase di revisione.

3.3.3.1.4 Metriche Qualità di Processo

Metrica	Codice	Descrizione
CMMI$_G$	MPC1	Indica la maturità di un processo seguendo lo *standard*$_G$ *ISO*$_G$/*IEC*$_G$ 15504.
Requirement Stability Index	MPC2	Indica la percentuale dei requisiti rimasti invariati nel tempo. La modifica o rimozione di alcuni requisiti durante lo sviluppo del progetto è un'attività lecita, in quanto possono sorgere nuove esigenze in corso d'opera. Un valore elevato di tale metrica indica un'attività di analisi attenta e corretta. $RSI = 1 - \frac{reqtolti+reqaggiunti+reqmodificati}{reqiniziali}$
Instability	MPC3	Indica la percentuale di instabilità di un package, ovvero la possibilità di effettuare modifiche ad una *componente*$_G$ del package senza influenzarne altri all'interno dell'applicazione. Si calcola quindi in seguito all'attività di *Progettazione* riportando il valore medio. $I = \frac{Ce}{Ca+Ce}$ dove: • **Ce**: il numero di riferimenti uscenti dal package; • **Ca**: il numero di riferimenti entranti dall'esterno del package. Più la percentuale di instabilità è alta, più il package è instabile.
Test di unità eseguiti	MPC4	Indica la percentuale di test di unità eseguiti. Più il valore è alto, più il processo di verifica è soddisfacente. $TUE = \frac{testuniteseguiti}{testunittotali}$ Viene calcolata da *PragmaDB*$_G$.
Test di integrazione eseguiti	MPC5	Indica la percentuale di test di integrazione eseguiti. Più il valore è alto, più il processo di verifica è soddisfacente. $TIE = \frac{testintegrazioneeseguiti}{testintegrazionetotali}$ Viene calcolata da *PragmaDB*$_G$.

Test di sistema eseguiti	MPC6	Indica la percentuale di test di sistema eseguiti. Più il valore è alto, più il processo di validazione è soddisfacente. $TSE = \frac{testsistemaeseguiti}{testsistematotali}$ Viene calcolata da *PragmaDB*$_G$.
Test di validazione eseguiti	MPC7	Indica la percentuale di test di validazione eseguiti. Più il valore è alto, più il processo di validazione è soddisfacente. $TVE = \frac{testvalideseguiti}{testvalidittotali}$ Viene calcolata da *PragmaDB*$_G$.

Tabella 2: Metriche per la qualità di processo

3.3.3.1.5 Metriche Qualità della Documentazione

Metrica	Codice	Descrizione
Indice di *Gulpease*$_G$	MDC1	Misura la leggibilità di un testo, individuandone la complessità dello stile. L'indice è tarato sulla lingua italiana ed ha il vantaggio di utilizzare la lunghezza delle parole in lettere invece che in sillabe, semplificando il procedimento di calcolo automatico. L'indice di *Gulpease*$_G$ considera due variabili: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere. La formula di calcolo è la seguente: $\mathbf{89} + \frac{\mathbf{300} \times (\mathbf{numerodellefrasi}) - \mathbf{10} \times (\mathbf{numerodellelettere})}{(\mathbf{numerodelleparole})}$ I risultati sono compresi tra 0 (leggibilità più bassa) e 100 (leggibilità più alta).

Tabella 3: Metriche per la qualità della documentazione

3.3.3.1.6 Metriche Qualità di Progetto

Metrica	Codice	Descrizione
Budget Variance$_G$	MPG1	Indica se la spesa sostenuta in una determinata data supera o meno quella preventivata (in percentuale, rispetto al preventivo finale). Una *Budget Variance*$_G$ positiva indica che si è speso meno di quanto inizialmente previsto nel Piano di Progetto. Una *Budget Variance*$_G$ negativa indica che si è speso più di quanto inizialmente previsto. $BV = \frac{BCWS-ACWP}{BCWS}$ con: • **BCWS**: *Budget Variance*$_G$ of Work Scheduled, costo pianificato per la data corrente valutato in Euro; • **ACWP**: Actual Cost of Work Performed, costo effettivamente sostenuto alla data corrente valutato in Euro.
Schedule Variance$_G$	MPG2	Indica l'efficacia, in quanto mostra se si è o meno coerenti con la pianificazione temporale rispetto alle attività nella *baseline*$_G$ (in giorni, rispetto al periodo di valutazione). Una Scheduled Variance positiva indica che il gruppo è in anticipo rispetto a quanto inizialmente previsto. Una Scheduled Variance negativa indica che il gruppo è in ritardo rispetto a quanto inizialmente previsto. $SV = BCWP - BCWS$ con: • **BCWP**: Budget Cost of Work Performed, valore delle attività realizzate alla data corrente valutato in giorni; • **BCWS**: Budget Cost of Work Scheduled, valore delle attività pianificate alla data corrente valutato in giorni.

Tabella 4: Metriche per la qualità di progetto

3.3.3.1.7 Metriche Qualità del Software

Metrica	Codice	Descrizione
Copertura Requisiti Obbligatori	MSW1	Indica la percentuale dei requisiti obbligatori coperti dal software. $CRO = \frac{reqobbligatorisoddisfatti}{reqobbligatoritotali}$ Più la percentuale è alta, più il software rispetta i requisiti obbligatori del software. Viene calcolata da *PragmaDB*$_G$.
Copertura Requisiti Desiderabili	MSW2	Indica la percentuale dei requisiti desiderabili coperti dal software. $CRD = \frac{reqdesiderabilisoddisfatti}{reqdesiderabilitotali}$ Più la percentuale è alta, più il software rispetta i requisiti desiderabili del software. Viene calcolata da *PragmaDB*$_G$.
Complessità Ciclomatica Media	MSW3	Indica la complessità del codice valutando il numero di cammini linearmente dipendenti attraverso il grafo di controllo del flusso. Viene calcolata per ogni funzione in modo da individuare eventuali costrutti troppo complessi. Più il valore è alto, meno il codice è manutenibile. Più il valore è basso, meno il codice è efficiente. Viene calcolata da *IntelliJ IDEA*$_G$.
Accoppiament Classi	MSW4	Calcola il numero di classi con cui ogni classe è accoppiata. Più il valore è alto, meno modulare è il codice. Viene calcolata da *IntelliJ IDEA*$_G$.
Attributi della Classe	MSW5	Indica il numero di campi dati per ogni classe. Più il valore è alto, più la classe ha responsabilità ed è meno manutenibile. Viene calcolata da *IntelliJ IDEA*$_G$.
Parametri	MSW6	Indica il numero di parametri passati ad un metodo. Più il valore è alto, meno il codice è manutenibile. Viene calcolata da *IntelliJ IDEA*$_G$.
Righe di Codice	MSW7	Indica il numero di righe di codice per ogni metodo, escludendo commenti o linee vuote. Più il valore è alto, meno il codice è manutenibile. Viene calcolata da *IntelliJ IDEA*$_G$.

Commenti	MSW8	Indica il rapporto tra le linee di commento e le linee di codice. Più il valore è alto, più il codice è manutenibile. Viene calcolata da *IntelliJ IDEA*$_G$.
Blocchi innestati	MSW9	Indica il massimo livello di annidamento di costrutti di controllo nei metodi (cicli for e while). Il valore viene calcolato contando i blocchi innestati presenti in tutto il codice. Più il valore è alto, più il codice è complesso. Viene calcolata da *IntelliJ IDEA*$_G$.
Copertura del Codice	MSW10	Indica la percentuale di codice testato con esito positivo. $CC = \frac{testsuperati}{testpianificati}$ Più è alto il valore, più il software è affidabile. Viene calcolata da *PragmaDB*$_G$.

Tabella 5: Metriche per la qualità del software

3.3.3.1.8 Strumenti

Gli strumenti utilizzati per automatizzare le attività di misurazione sono:

- **Script indice di *Gulpease*$_G$**: script Python *open source*$_G$ da eseguire sui *file*$_G$ in formato PDF;
- **Foglio di calcolo GoogleDocs**: *tool*$_G$ per produrre grafici con la possibilità di inserire nuovi dati senza dover rigenerare il grafico;
- ***PragmaDB*$_G$**: strumento utilizzato per il tracciamento dei requisiti e dei test;
- ***IntelliJ IDEA*$_G$**: *editor*$_G$ che offre funzionalità di calcolo di metriche del software.

3.3.3.2 Analisi

3.3.3.2.1 Scopo e Aspettative

L'attività di analisi si pone l'obiettivo di garantire la correttezza dei prodotti proposti.

3.3.3.2.2 Descrizione

L'analisi si suddivide in due tecniche:

- Analisi statica;
- Analisi dinamica.

3.3.3.2.3 Analisi Statica

L'analisi statica è una tecnica che non richiede l'esecuzione del software, ma richiede uno studio del codice e della documentazione.
Viene utilizzata durante tutto lo svolgimento del progetto tramite una delle seguenti strategie:

- **Walkthrough**: strategia per una lettura preliminare dei prodotti non efficiente. Consiste in una lettura a largo spettro senza presupposti e solitamente effettuata da più di una persona nella fase iniziale del progetto, in cui non tutti i membri del gruppo hanno padronanza delle Norme di Progetto e del Piano di Qualifica. La lettura è accompagnata da una discussione collettiva a cui seguono le opportune correzioni;
- **Inspection**: strategia efficiente per una lettura mirata dei prodotti. Consiste in una lettura dettagliata basata su errori presupposti appoggiandosi ad una lista stilata precedentemente. Normalmente viene effettuata da una sola persona.

3.3.3.2.4 Analisi Dinamica

L'analisi dinamica è una tecnica che richiede l'esecuzione del software che sfrutta dei test prestabiliti.
Grazie a questi test vengono riscontrati errori che possono essere identificati e risolti durante il processo di assicurazione della qualità.
I test devono essere ripetibili, ovvero, dato lo stesso input, deve essere possibile risalire allo stesso output.
Pertanto i test sono così definiti:

- **Ambiente**: il sistema sul quale viene effettuato il test;
- **Stato iniziale**: lo stato iniziale del sistema prima di eseguire il test;
- **Input**: l'input inserito;
- **Output**: l'output atteso;
- **Eventuali istruzioni**: se necessario, per definire come eseguire il test e come interpretare i risultati.

3.3.3.2.5 Strumenti Analisi Statica

La documentazione viene stilata utilizzando il linguaggio *LaTeX*$_G$.
Per la verifica ortografica vengono utilizzati gli strumenti integrati all'interno degli *IDE*$_G$ utilizzati dai membri del gruppo come Texmaker, TeXstudio e TeXnicCenter.
Per quanto riguarda l'analisi statica del codice, si utilizza il *tool*$_G$ online *JSHint*$_G$ che aiuta a trovare errori e potenziali problemi nel codice *JavaScript*$_G$.

3.3.3.2.6 Strumenti Analisi Dinamica

Per automatizzare l'esecuzione di test il gruppo utilizza gli strumenti seguenti:

- ***Jasmine***$_G$: è un *framework*$_G$ per scrivere test su codice *JavaScript*$_G$. Si basa sulle asserzioni, in particolare confronta l'output in uscita con l'output atteso verificando che siano equivalenti;
- ***Karma***$_G$: è un *tool*$_G$ che permette di eseguire i test *Jasmine*$_G$.

3.4 Processo di Validazione

3.4.1 Scopo e Aspettative

Questo processo ha lo scopo di confermare che i requisiti vengano rispettati quando uno specifico prodotto è utilizzato nell'ambiente destinatario e soddisfi l'utilizzo per cui è stato creato.
Il gruppo si aspetta che i prodotti verificati siano effettivamente conformi a quanto pattuito.

3.4.2 Descrizione

L'attività di validazione consiste nell'eseguire i test di validazione per poi analizzare i risultati e assicurarsi che il prodotto sia conforme alle caratteristiche per cui è stato sviluppato.

3.4.3 Attività

3.4.3.1 Analisi Dinamica

Il *Verificatore* ha il compito di rieseguire di nuovo tutti i test, in particolare quelli di validazione, e fornire i dati al *Responsabile*, il quale decide se rieseguire i test.

3.4.4 Strumenti

Per la validazione delle pagine *HTML*$_G$ del prodotto finale si utilizza lo strumento messo a disposizione dal *W3C*$_G$, raggiungibile al sito https://validator.w3.org/.
Per i fogli di stile *CSS*$_G$ invece si utilizza lo strumento disponibile al sito https://jigsaw.w3.org/css-validator/, sempre messo a disposizione dal *W3C*$_G$.

3.5 Processo di Assicurazione Qualità

3.5.1 Scopo e Aspettative

Il processo ha lo scopo di assicurare che tutti i prodotti siano conformi con i piani e gli *standard*$_G$ definiti, delineando le linee guida per pianificare, realizzare e controllare la qualità.

3.5.2 Descrizione

Tale processo identifica le procedure di verifica e validazione da applicare per garantire la qualità.
A tal fine si segue lo *standard*$_G$ *ISO*$_G$/*IEC*$_G$ 9001, che suggerisce di attuare una gestione della qualità a livello di processi e non solo sul prodotto finale.
Un processo ben definito ed eseguito correttamente infatti permette di limitare gli errori nelle fasi successive e garantire quindi un buon risultato.

3.5.3 Procedura

3.5.3.1 Controllo Qualità di Processo

La qualità di processo viene valutata in termini quantitativi utilizzando lo *standard*$_G$ *ISO*$_G$/*IEC*$_G$ 15504, comunemente denominato *SPICE*$_G$.
Per una corretta applicazione di questo *standard*$_G$, il gruppo adotta il modello iterativo *PDCA*$_G$, detto anche "*Ciclo di Deming*$_G$". Il *PDCA*$_G$ è uno schema di gestione iterativo che mira al miglioramento continuo dei processi e dei prodotti in un'ottica a lungo raggio.
Tale modello permette di pianificare e verificare ogni attività per consentire il passaggio ad un livello superiore nella scala definita dallo *SPICE*$_G$, il *CMMI*$_G$.

3.5.3.2 Controllo Qualità di Prodotto

La qualità di prodotto viene garantita in seguito al rispetto delle *Norme di Progetto v4.0.0* e delle attività di verifica e validazione.
In particolare:

- **Verifica**: grazie alle attività del processo di verifica e agli obiettivi definiti nel *Piano di Qualifica v4.0.0* si può monitorare la qualità dei prodotti sviluppati;
- **Validazione**: con il processo di validazione si può confermare il livello di qualità raggiunto;
- **Norme**: il rispetto delle norme e degli strumenti in esso definiti facilita la produzione di prodotti di qualità.

Il prodotto software è valutato in base alle caratteristiche definite nello *standard*$_G$ *ISO*$_G$/*IEC*$_G$ 9126.

3.5.3.3 Test

3.5.3.3.1 Nomenclatura

Per identificare univocamente i test da eseguire e registrare nel *Piano di Qualifica v4.0.0* il gruppo utilizza le seguenti notazioni:

T{X}{Y}

In particolare:

- **X**: indica la tipologia di test, utilizzando le lettere:
  - **U**: Test di unità;
  - **I**: Test di integrazione;
- **Y**: indica univocamente il test attraverso numeri in ordine progressivo.

Mentre:

T{W}{Z}

In particolare:

- **W**: indica la tipologia di test, utilizzando le lettere:
  - **S**: Test di sistema;
  - **V**: Test di validazione;
- **Z**: indica il codice identificativo del requisito a cui fa riferimento.

In base allo stato di avanzamento dei processi di verifica e validazione, i test possono trovarsi in uno dei seguenti stati:

- **Implementato**;
- **Non implementato**;
- **Eseguito**;
- **Non eseguito**;
- **Superato**;
- **Non superato**.

3.5.3.3.2 Test di Unità

L'obiettivo dei test di unità è verificare che il codice implementato sia corretto; pertanto la responsabilità di questi test è dello stesso *Programmatore*.
Tale attività viene effettuata prima dei test di integrazione per garantire il funzionamento delle singole unità.
Per automatizzare il più possibile, il *Programmatore* implementa dei *driver*$_G$ per gestire l'attività di test e fornire i dati di input e degli *stub*$_G$ per realizzare il test e verificarne l'output, che deve essere corretto rispetto all'input e all'output stabilito.

3.5.3.3.3 Test di Integrazione

Una volta svolti i test sulle unità, si può procedere con l'esecuzione dei test di integrazione, che combinano tali unità in componenti testandone la corretta integrazione.
Grazie a questi test è possibile inoltre rilevare errori residui nella realizzazione delle unità.
Implementando *stub*$_G$ e *driver*$_G$ si procede aggregando le parti in strutture più grandi per determinare se esse hanno il comportamento atteso anche quando si trovano ad operare insieme come un unica *componente*$_G$.

3.5.3.3.4 Test di Sistema

Con i test di sistema il gruppo si accerta la copertura dei requisiti software per prepararsi al collaudo e valutare il lavoro svolto.
In particolare questi test sanciscono una versione definitiva del software sviluppato.

3.5.3.3.5 Test di Validazione

I test di validazione servono per dimostrare la conformità del prodotto come richiesto dal *proponente*$_G$.
Consiste nell'attività di collaudo supervisionata dal *committente*$_G$ a cui segue il rilascio ufficiale del prodotto sviluppato.
Per l'esecuzione dei test di validazione è necessario seguire le istruzioni nella descrizione indicata in ognuno nel *Piano di Qualifica v4.0.0*.

3.5.3.3.6 Strumenti

Per mantenere il tracciamento test-requisiti e requisiti-test, il gruppo utilizza *PragmaDB*$_G$.
Tale strumento, utilizzato anche per l'analisi dei requisiti, contiene già tutti i requisiti del sistema che possono quindi essere facilmente collegati ai test.

4 Processi di Organizzazione

4.1 Scopo e Aspettative

Lo scopo di questo processo è quello di definire regole e strumenti per rendere efficiente l'organizzazione del gruppo.

4.2 Descrizione

In questo processo vengono definiti i ruoli di progetto e le procedure per gestire i processi e le infrastrutture necessarie.

4.3 Ruoli di Progetto

Innanzitutto occorre delineare i vari ruoli di progetto che compaiono durante lo sviluppo del prodotto. Di ognuno vengono descritte le principali mansioni e responsabilità.

4.3.1 Responsabile

- Gestione del progetto;
- Relazioni con *committente*$_G$ e *proponente*$_G$;
- Istanziare processi nel progetto;
- Stimare i costi e le risorse necessarie;
- Pianificare le attività e assegnarle alle risorse umane;
- Controllare le attività e verificare i risultati;
- Valutazione dei *rischi*$_G$.

4.3.2 Amministratore

- Si occupa dell'infrastruttura di lavoro;
- Scelta di strumenti e tecnologie utili a perseguire qualità;
- Responsabile della capacità operativa e dell'efficienza;
- Gestione della documentazione di progetto;
- Controllo di versioni e configurazioni;
- Evita conflitti tra ruoli e responsabilità.

4.3.3 Analista

- Si occupa dell'analisi dei problemi e del dominio applicativo;
- Rappresenta un ruolo fondamentale solo nella fase di analisi iniziale e la sua presenza non è sempre necessaria;
- Traduce le esigenze del *proponente*$_G$.

4.3.4 Progettista

- Si occupa degli aspetti tecnologici e tecnici del progetto;
- Fa in modo che il progetto sia manutenibile;
- Effettua scelte efficienti su aspetti tecnici del progetto.

4.3.5 Verificatore

- Conosce le norme di progetto e ne garantisce la verifica;
- È necessario durante tutto il progetto;
- Controlla le attività di progetto seguendo le norme;
- Verifica che le attività non introducano errori.

4.3.6 Programmatore

- Si occupa dell'attività di codifica del progetto;
- Implementa le soluzioni dei progettisti;
- Scrive codice pulito e manutenibile;
- Ha il compito di gestire il *controllo versione*$_G$ del codice prodotto;
- Redige il *manuale utente*$_G$;
- Realizza strumenti per il test.

4.4 Suddivisione Documentale per i Ruoli

Ogni documento formale viene redatto da uno o più ruoli specifici.

- **Analisi dei requisiti**: *Analisti*;
- **Studio di fattibilità**: *Analisti*;
- **Norme di progetto**: *Amministratori* per conto del *Responsabile*;
- **Piano di Progetto**: *Responsabile* in collaborazione con gli *Amministratori*;

- **Piano di Qualifica**: *Verificatori* e *Amministratori*;
- **Glossario**: Collaborazione collettiva;
- **Verbali**: *Amministratori*.

4.5 Procedure e Strumenti

4.5.1 Gestione dei Processi

Lo scopo di questo processo è di ottenere una collaborazione produttiva, gestendo a livello organizzativo il team durante i vari processi.
Per questo motivo, il documento formale *Piano di Progetto v4.0.0*, in allegato, si occupa di:

- Identificare, analizzare e pianificare i *rischi*$_G$;
- Pianificare processi, attività e compiti nel corso di ogni periodo;
- Redigere un preventivo di ore e costi, per ogni ruolo e membro del gruppo;
- Redigere, di volta in volta, un consuntivo finale, da confrontare con il rispettivo preventivo.

Per quanto riguarda l'attività di *Analisi dei rischi*$_G$, derivanti da risorse umane e tecnologiche, è sempre di competenza del *Responsabile* che, all'interno del documento *Piano di Progetto v4.0.0*, ne approfondisce i vari aspetti.
La procedura per la gestione dei *rischi*$_G$ è la seguente:

- Identificazione del *rischio*$_G$;
- Analisi e comprensione del problema;
- Pianificazione su possibile risoluzione;
- Applicazione della soluzione rispettando la pianificazione;
- Registrazione di ogni riscontro;
- In caso di esito negativo, cambiamento delle strategie di progetto.

4.5.2 Miglioramento dei Processi

Questo processo vigila tutti gli altri processi e si occupa di correggere eventuali errori e apportare migliorie costanti.

4.5.3 Processo di Formazione

Questo processo si occupa della formazione del gruppo. Inizialmente è previsto un periodo di formazione agli strumenti di supporto, per permettere ad ogni membro l'utilizzo dell'infrastruttura.
Successivamente sono previsti periodi di formazione alle tecnologie richieste per lo sviluppo del prodotto.

4.5.4 Gestione delle Infrastrutture

Lo scopo di questo processo è di permettere l'utilizzo e la gestione di un'infrastruttura efficiente ed efficace.

4.5.4.1 Gestione delle Comunicazioni

4.5.4.1.1 Interne

Il gruppo comunica principalmente attraverso due strumenti: *Telegram*$_G$ e *Slack*$_G$.
Nel gruppo *Telegram*$_G$ appositamente creato si comunica in modo colloquiale, per confronto e discussioni.
Il *workspace*$_G$ di *Slack*$_G$ è suddiviso in canali più formali, che vengono utilizzati dai membri per questioni ad hoc e condivisione di risorse utili.
Nello specifico:

- **#calendario**: qui vengono inserite le principale scadenze (riunioni, consegne ed incontri lavorativi) ed i vari impegni personali ed universitari dei membri del gruppo;
- **#contatti**: elenco dei contatti interni ed esterni utili alle comunicazioni;
- **#email**: canale collegato direttamente alla casella di posta elettronica del gruppo;
- **#general**: discussioni di carattere generale;
- **#github**: canale collegato direttamente alla *repository*$_G$ del gruppo che notifica le operazioni di push;
- **#glossario**: canale che contiene le parole da inserire nel glossario;
- **#latex**: canale dedicato all'uso del supporto LaTeX$_G$ e dei suoi comandi;
- **#problemi**: canale in cui vengono segnalate problematiche da risolvere;
- **#rendicontazione**: canale in cui viene tenuta traccia di tutte le ore che il gruppo dedica al progetto;
- **#tecnologie**: elenco delle risorse utili al fine dello sviluppo del progetto;
- **#todolist**: canale collegato a *Trello*$_G$, che notifica task e scadenze.

4.5.4.1.2 Esterne

Il mezzo principale per comunicazioni con il *committente*$_G$ ed il *proponente*$_G$ è la casella di posta elettronica Gmail.
È stata creata appositamente per il gruppo con lo scopo di accordare le parti per un incontro o per semplici chiarimenti.

4.5.4.2 Gestione degli Incontri

4.5.4.2.1 Incontri Attivi

Grazie alla relativa vicinanza dei membri del gruppo è possibile incontrarsi ogni giorno.
A seconda degli impegni, sia universitari che personali, è di volta in volta confermato l'incontro successivo dal *Responsabile*.
Se sono presenti almeno quattro membri del gruppo, qualsiasi decisione viene automaticamente sottoscritta dagli assenti.
Generalmente si lavora dalle 09:30 alle 12:30 e dalle 13:30 alle 16:30 in modo collettivo, mentre il tempo rimanente viene utilizzato dai singoli per formazione ed approfondimenti.

4.5.4.2.2 Incontri Decisionali

Le riunioni del gruppo, da cui deriverà il corrispondente verbale interno, vengono fissate dal *Responsabile*, il quale ha il compito di:

- Accogliere l'eventuale richiesta di incontro da uno o più membri;
- Conciliare eventuali impegni dei singoli;
- Scegliere la prima data utile per l'incontro;
- Notificare la data dell'incontro nel canale #calendar di *Slack*$_G$.

4.5.4.2.3 Incontri Esterni

Il *Responsabile* ha il compito di chiedere al *proponente*$_G$ la prima data utile per un incontro attraverso e-mail.
All'effettivo incontro rappresenta il gruppo ed è affiancato da almeno un *Amministratore* che si occupa della *minuta*$_G$.

4.5.4.3 Gestione Strumenti di Coordinamento

Per la gestione dei task dei vari membri e le rispettive scadenze, il gruppo ha optato per l'utilizzo della piattaforma *Trello*$_G$.
La bacheca creata appositamente per il gruppo viene gestita dal *Responsabile* che ha il compito di inserire ed assegnare i nuovi task ai singoli membri, i quali dovranno eventualmente aggiornarli e/o modificarli fino al loro completamento.
Nello specifico un task segue la seguente procedura:

- Inserimento, da parte del *Responsabile*, nella lista "Da fare" con relativo commento, scadenza e assegnatari;
- Spostamento nella lista "In corso" dai rispettivi assegnatari;
- Alla conclusione del task, l'assegnatario provvederà a spostare il task nella lista "Da verificare";
- I *Verificatori* controlleranno l'esito del task:

  - In caso negativo lo sposteranno in "Da rivedere" con commento contenente gli errori e le modifiche da apportare;
  - In caso di esito positivo il task verrà spostato nella lista "Completati".

- Il *Responsabile* procede allo spostamento nella lista "Approvati".

4.5.4.4 Gestione Strumenti di Versionamento

4.5.4.4.1 Versionamento Formale

Il versionamento viene gestito attraverso il servizio di *hosting*$_G$ *GitHub*$_G$ con tre *repository*$_G$.
La *repository*$_G$ IronWorksDocuments contiene tutta la documentazione formale ed è accessibile al seguente link: http://www.github.com/SwearOnCode/IronWorksDocuments.
La *repository*$_G$ IronWorksCode contiene il codice dell'applicativo richiesto ed è accessibile al seguente link: http://www.github.com/SwearOnCode/IronWorksCode.
La *repository*$_G$ ProofOfConcept contiene il codice relativo al *Proof of Concept*$_G$ ed è accessibile al seguente link: http://www.github.com/SwearOnCode/ProofOfConcept.
I comandi principali per relazionarsi con la *repository*$_G$ vengono delineati nel documento interno *Guida Git*$_G$ *v1.0.0*.
In particolare la struttura della *repository*$_G$ IronWorksDocuments segue queste norme:

- Le *directory*$_G$ principali si basano sulle diverse revisioni (RR, RP, RQ, RA);
- Le sotto-*directory*$_G$ dividono i documenti interni da quelli esterni.

4.5.4.4.2 Versionamento Informale

Tutta la documentazione informale viene depositata nel servizio *Google Drive*$_G$ del gruppo.
Questa *repository*$_G$ viene gestita senza versionamento ufficiale, utilizzata da qualsiasi membro del gruppo, supportando anche modifiche contemporanee.