



Norme di Progetto

sevenbits.swe.unipd@gmail.com



Registro modifiche

Versione	Data	Autore	Verificatore	Descrizione
1.0.5	2025-04-03	Federico Pivetta	Manuel Gusella	Correzioni alle metriche e alla gestione della configurazione
1.0.4	2025-03-26	Manuel Gusella	Federico Pivetta	Aggiornamento processo di verifica della documentazione e aggiornato i ruoli
1.0.3	2025-03-26	Leonardo Trolese	Federico Pivetta	Correzione sezione metriche a seguito di inizio stesura codice
1.0.2	2025-03-20	Giovanni Cristellon	Federico Pivetta	Correzioni minori e aggiornamento della stesura dei verbali
1.0.1	2025-03-12	Federico Pivetta	Alfredo Rubino	Aggiunti paragrafi Specifica Tecnica, Manuale Utente, Qualità dell'architettura e Design Pattern
1.0.0	2025-02-21	Giovanni Cristellon	Manuel Gusella	Approvazione documento per RTB
0.5.18	2025-02-17	Federico Pivetta	Uncas Peruzzi	Correzioni ai nomi dei documenti
0.5.17	2025-02-17	Leonardo Trolese	Federico Pivetta	Correzione minore di alcune metriche
0.5.16	2025-01-17	Manuel Gusella	Uncas Peruzzi	Stesura metriche RSI_G e CPI_G nella sezione Metriche per la qualità
0.5.15	2025-01-16	Manuel Gusella	Alfredo Rubino	Rimozione metriche non utilizzate e cambiamento dei codici delle metriche nella sezione Metriche per la qualità
0.5.14	2025-01-15	Leonardo Trolese	Alfredo Rubino	Aggiunta delle metriche Fan IN e Fan OUT nella sezione delle metriche
0.5.13	2025-01-11	Manuel Gusella	Leonardo Trolese	Aggiunte tabelle delle metriche per le sezioni Codifica, Progettazione e Analisi dei Requisiti
0.5.12	2025-01-02	Leonardo Trolese	Uncas Peruzzi	Correzioni minori
0.5.11	2025-01-02	Leonardo Trolese	Uncas Peruzzi	Scrittura della sezione "Standard ISO/IEC 12207"
0.5.10	2024-12-27	Leonardo Trolese	Uncas Peruzzi	Aggiunta immagine descrittiva nella sezione "Standard ISO/IEC 9126"
0.5.9	2024-12-27	Federico Pivetta	Riccardo Piva	Completate sottosezioni "Miglioramento" e "Formazione"

Versione	Data	Autore	Verificatore	Descrizione
0.5.8	2024-12-27	Leonardo Trolese	Uncas Peruzzi	Aggiornamento della sezione "Standard ISO/IEC 9126"
0.5.7	2024-12-27	Federico Pivetta	Uncas Peruzzi	Completata sottosezione "Gestione della configurazione"
0.5.6	2024-12-26	Alfredo Rubino	Riccardo Piva	Stesura della sottosezione "Gestione della qualità"
0.5.5	2024-12-24	Riccardo Piva	Uncas Peruzzi	Ulteriore continuazione stesura metriche
0.5.4	2024-12-24	Leonardo Trolese	Riccardo Piva	Continuazione stesura sezione metriche
0.5.3	2024-12-19	Manuel Gusella	Alfredo Rubino	Fine stesura sottosezione Verifica
0.5.2	2024-12-18	Manuel Gusella	Alfredo Rubino	Continuazione sottosezione Verifica
0.5.1	2024-12-16	Manuel Gusella	Alfredo Rubino	Stesura iniziale della sottosezione Verifica
0.5.0	2024-12-16	Alfredo Rubino	Manuel Gusella	Aggiunta sezione "Metriche per la qualità" e inizio stesura
0.4.3	2024-12-11	Federico Pivetta	Manuel Gusella	Completata sottosezione "Gestione dei processi"
0.4.2	2024-12-11	Leonardo Trolese	Manuel Gusella	Inizio scrittura sezione 5 relativa agli standard e correzione errori di battitura
0.4.1	2024-12-06	Alfredo Rubino	Giovanni Cristellon	Aggiunte generali e correzioni
0.4.0	2024-12-06	Federico Pivetta	Alfredo Rubino	Inizio stesura sottosezione "Gestione dei processi" e abbandono impostazione in sottodocumenti
0.3.9	2024-12-05	Alfredo Rubino	Leonardo Trolese	Completata sezione "Processi Primari"
0.3.8	2024-12-05	Federico Pivetta	Leonardo Trolese	Completato paragrafo "Diagrammi delle classi"
0.3.7	2024-12-04	Alfredo Rubino	Leonardo Trolese	Aggiornamento sottosezione "Sviluppo" (terza parte) e completata modifica ai termini del Glossario
0.3.6	2024-12-04	Federico Pivetta	Leonardo Trolese	Aggiornata sottosezione "Sviluppo" (seconda parte)
0.3.5	2024-12-02	Alfredo Rubino	Giovanni Cristellon	Aggiornata sottosezione "Sviluppo"
0.3.4	2024-11-30	Leonardo Trolese	Giovanni Cristellon	Correzioni grammaticali e di contenuti non più aderenti al Way-of-working _G stabilito

Versione	Data	Autore	Verificatore	Descrizione
0.3.3	2024-11-30	Alfredo Rubino	Giovanni Cristellon	Aggiornata sottosezione "Sviluppo" e aggiunti livelli di profondità nell'indice
0.3.2	2024-11-28	Alfredo Rubino	Giovanni Cristellon	Modifica ai termini del Glossario
0.3.1	2024-11-25	Federico Pivetta	Riccardo Piva	Completata sottosezione "Fornitura" e aggiunta sottosezione "Sviluppo"
0.3.0	2024-11-24	Federico Pivetta	Riccardo Piva	Aggiunta sezione "Processi primari"
0.2.1	2024-11-21	Federico Pivetta	Riccardo Piva	Completata sezione "Introduzione" e sottosezione "Documentazione", inclusa modifica alla tabella Registro modifiche
0.2.0	2024-11-20	Leonardo Trolese	Federico Pivetta	Aggiunta sezione "Processi di supporto" e impostazione della divisione in sottodocumenti
0.1.0	2024-11-11	Leonardo Trolese	Federico Pivetta	Creazione del documento secondo la struttura definita dal gruppo

Indice

1	Introduzione	9
1.1	Scopo del documento	9
1.2	Scopo del prodotto	9
1.3	Glossario	9
1.4	Riferimenti	9
1.4.1	Riferimenti Normativi	9
1.4.2	Riferimenti Tecnologici	9
1.4.3	Riferimenti Informativi	10
2	Processi primari	11
2.1	Fornitura	11
2.1.1	Descrizione e Scopo	11
2.1.2	Attività	11
2.1.3	Rapporti con l'azienda proponente	11
2.1.4	Documentazione fornita	11
2.1.4.1	Analisi dei Requisiti	12
2.1.4.2	Piano di Progetto	12
2.1.4.3	Piano di Qualifica	12
2.1.4.4	Specifica Tecnica	12
2.1.4.5	Manuale Utente	12
2.1.4.6	Glossario	12
2.1.4.7	Lettera di Presentazione	12
2.1.5	Strumenti	12
2.2	Sviluppo	13
2.2.1	Descrizione e Scopo	13
2.2.2	Analisi-dei-Requisiti	13
2.2.2.1	Descrizione e Scopo	13
2.2.2.2	Casi d'uso	13
2.2.2.3	Diagrammi dei casi d'uso	14
2.2.2.4	Requisiti	17
2.2.2.5	Metriche	18
2.2.3	Progettazione	18
2.2.3.1	Descrizione e Scopo	18
2.2.3.2	Qualità dell'architettura	19
2.2.3.3	Diagrammi delle classi	19
2.2.3.4	Design Pattern	21
2.2.3.5	Metriche	22
2.2.4	Codifica	22
2.2.4.1	Descrizione e Scopo	22
2.2.4.2	Norme di codifica	22
2.2.4.3	Integrazione	23
2.2.4.4	Verifica	23
2.2.4.5	Metriche	23
2.2.5	Strumenti	23
3	Processi di supporto	24
3.1	Documentazione	24
3.1.1	Descrizione e Scopo	24
3.1.2	Lista documenti	24
3.1.3	Template in L ^A T _E X	24
3.1.4	Nomenclatura	24
3.1.5	Struttura documenti	25
3.1.5.1	Prima Pagina	25
3.1.5.2	Intestazione	25
3.1.5.3	Registro modifiche	25
3.1.5.4	Indice	25

3.1.5.5	Verbali	25
3.1.6	Convenzioni stilistiche	26
3.1.6.1	Stile del testo	26
3.1.6.2	Formato delle date	26
3.1.7	Strumenti	26
3.2	Gestione della configurazione	26
3.2.1	Descrizione e Scopo	26
3.2.2	Project Board	26
3.2.3	Ciclo di vita e Versionamento dei documenti	27
3.2.3.1	Versionamento dei documenti	27
3.2.3.2	Workflow verbali	27
3.2.3.3	Workflow Documenti Ufficiali	27
3.2.4	Ciclo di vita del software	28
3.2.4.1	Workflow Software	28
3.2.5	Repository	28
3.2.5.1	Struttura della repository 7BitsDocs	29
3.2.5.2	Struttura della repository PoC	29
3.2.5.3	Struttura della repository MVP	29
3.2.6	Pagina web	29
3.2.7	Controllo dei termini del glossario	29
3.3	Verifica	29
3.3.1	Descrizione e Scopo	29
3.3.2	Strumenti	30
3.3.3	Analisi statica	30
3.3.3.1	Inspection	30
3.3.3.2	Walkthrough	30
3.3.4	Analisi dinamica	30
3.3.5	Classificazione dei test	30
3.3.5.1	Test di unità	30
3.3.5.2	Test di integrazione	31
3.3.5.3	Test di sistema	31
3.3.5.4	Test di regressione	31
3.3.5.5	Test di accettazione	31
3.3.5.6	Classificazione dei test	31
3.3.5.7	Stato dei test	31
3.4	Gestione della Qualità	32
3.4.1	Descrizione e Scopo	32
3.4.2	Aspettative	32
3.4.3	Metriche e Strumenti	32
3.4.4	Struttura e identificazione delle metriche	32
3.4.5	Documentazione della Qualità	32
4	Processi organizzativi	33
4.1	Gestione dei processi	33
4.1.1	Descrizione e Scopo	33
4.1.2	Attività	33
4.1.3	Pianificazione degli sprint	33
4.1.4	Ruoli di progetto	33
4.1.4.1	Responsabile	33
4.1.4.2	Amministratore	34
4.1.4.3	Analista	34
4.1.4.4	Progettista	34
4.1.4.5	Programmatore	34
4.1.4.6	Verificatore	35
4.1.5	Cambio dei ruoli	35
4.1.6	Tracciamento delle ore	35
4.1.6.1	Preventivo	35
4.1.6.2	Consuntivo	35

4.1.7	Ticketing	35
4.1.8	Rischi	36
4.1.9	Coordinamento	36
4.1.10	Comunicazioni	36
4.1.10.1	Comunicazioni interne	36
4.1.10.2	Comunicazioni esterne	37
4.1.11	Riunioni	37
4.1.11.1	Riunioni interne	37
4.1.11.2	Riunioni esterne	37
4.2	Miglioramento	37
4.2.1	Descrizione e Scopo	37
4.2.2	Ciclo di miglioramento continuo	37
4.3	Formazione	38
4.3.1	Descrizione e Scopo	38
4.3.2	Formazione individuale	38
5	Standard di progetto	39
5.1	Standard ISO/IEC 12207 - 1995	39
5.1.1	Processi primari	39
5.1.2	Processi di supporto	39
5.1.3	Processi organizzativi	39
5.2	Standard ISO/IEC 9126 - 1991	39
5.2.1	Qualità del prodotto software	39
5.2.1.1	Funzionalità	40
5.2.1.2	Affidabilità	40
5.2.1.3	Usabilità	40
5.2.1.4	Efficienza	41
5.2.1.5	Manutenibilità	41
5.2.1.6	Portabilità	41
6	Metriche per la qualità	42
6.1	Metriche di Qualità del Prodotto	42
6.1.1	Funzionalità	42
6.1.2	Affidabilità	42
6.1.3	Usabilità	43
6.1.4	Manutenibilità	43
6.1.5	Portabilità	43
6.2	Metriche di qualità di Processo	43
6.2.1	Processi Primari	43
6.2.1.1	Fornitura	43
6.2.1.2	Sviluppo	44
6.2.2	Processi di Supporto	45
6.2.2.1	Documentazione	45
6.2.2.2	Verifica	45
6.2.2.3	Gestione della Qualità	45
6.2.3	Processi Organizzativi	45
6.2.3.1	Gestione dei Processi	45

Elenco delle figure

1	Rappresentazione di un attore _G	14
2	Rappresentazione di un caso-d'uso _G	14
3	Rappresentazione di un sottocaso d'uso.	15
4	Rappresentazione di un sistema _G	15
5	Rappresentazione di una associazione tra attore _G e UC.	15
6	Rappresentazione di una generalizzazione tra attori.	16
7	Rappresentazione di una inclusione tra UC.	16
8	Rappresentazione di una estensione tra UC.	17
9	Rappresentazione di una generalizzazione tra UC.	17
10	Rappresentazione di una classe.	20
11	Rappresentazione della relazione di dipendenza.	20
12	Rappresentazione della relazione di aggregazione.	20
13	Rappresentazione della relazione di composizione.	21
14	Rappresentazione della relazione di associazione.	21
15	Rappresentazione della relazione di generalizzazione.	21
16	Schema riassuntivo dello Standard ISO/IEC:9126	40

Elenco delle tabelle

2	Metriche relative all'attività di analisi-dei-requisiti	18
3	Metriche relative all'attività di progettazione	22
4	Metriche relative all'attività di codifica	23

1 Introduzione

1.1 Scopo del documento

Il presente documento è stato realizzato al fine di definire e raccogliere le best-practices_G e il way-of-working_G a cui ogni componente del gruppo Seven Bits dovrà aderire per l'intera realizzazione del progetto_G, garantendo così l'adozione di un metodo di lavoro completamente omogeneo.

La formulazione delle norme_G di progetto_G avviene in maniera progressiva, permettendo al gruppo di apportare continui aggiornamenti ad esse in risposta alle esigenze che il team dovrà affrontare durante lo svolgimento del progetto_G stesso.

1.2 Scopo del prodotto

Ogni giorno, le persone vengono sommerse da una miriade di annunci generici che spesso non rispecchiano i loro reali interessi o il contesto in cui si trovano. Questa separazione tra il messaggio e il destinatario porta ad una bassa interazione con gli utenti e una riduzione delle conversioni per i brand.

Il progetto_G "Near You" si concentra sulla creazione di una dashboard_G composta principalmente da una mappa, sulla quale verranno visualizzate in tempo reale le posizioni degli utenti. Mediante un pop-up o una finestra a parte, verranno visualizzati messaggi personalizzati solo in prossimità dei punti di interesse. L'obiettivo finale è generare annunci pubblicitari in base agli interessi del cliente e alla sua posizione in quel momento.

1.3 Glossario

Ai fini di garantire l'adesione dei membri del gruppo ad un vocabolario comune e condiviso, che non lasci spazio ad ambiguità, dubbi o imprecisioni; il gruppo ha definito un documento denominato *Glossario-v1.0.0.pdf*, nel quale sono presenti tutti i termini tecnici adottati dal gruppo per l'intera durata della realizzazione del progetto_G. Tali termini saranno contrassegnati con una _G a pedice. Nel caso di termini composti, essi saranno uniti da un trattino (es. pull-request_G)

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- Capitolato_G di progetto_G C4 - Near You - Smart custom advertising platform
<https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C4p.pdf>
(Consultato: 2025-02-19).
- Standard ISO/IEC 12207:1995
https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
(Consultato: 2025-02-19).

1.4.2 Riferimenti Tecnologici

- Documentazione_G Git_G: <https://git-scm.com/docs>
(Consultato: 2025-02-19).
- Documentazione_G Github_G: <https://docs.github.com/en>
(Consultato: 2025-02-19).
- Documentazione_G L^AT_EX: <https://www.latex-project.org/help/documentation/>
(Consultato: 2025-02-19).
- Documentazione_G Python_G: <https://www.python.org/doc/>
(Consultato: 2025-02-19).

1.4.3 Riferimenti Informativi

- Riferimenti per le metriche:
 - https://lia.disi.unibo.it/Courses/IngSW/SE_C_2_Metriche_.pdf
(Consultato: 2025-02-19).
 - <https://youtu.be/CXgDazMODWA?feature=shared>
(Consultato: 2025-02-19).

2 Processi primari

2.1 Fornitura

2.1.1 Descrizione e Scopo

Come stabilito dallo standard ISO/IEC 12207:1995, il processo_G di fornitura definisce un insieme di linee guida necessario per una buona comunicazione tra fornitore e proponente_G. Il processo_G di fornitura si occupa di controllare e coordinare tutte le attività svolte dal gruppo, dalla comprensione dei requisiti fino alla consegna, per garantire che il prodotto finale soddisfi le esigenze concordate con la proponente_G.

2.1.2 Attività

Il processo_G di fornitura, come stabilito dallo standard ISO/IEC 12207:1995, si compone delle seguenti attività:

1. **Avvio:** Composto dall'identificazione e comprensione delle richieste della proponente_G, con successiva verifica della fattibilità tecnologica di queste ultime;
2. **Preparazione dell'offerta:** Composta dall'elaborazione della proposta in grado di soddisfare le richieste della proponente_G, che dettagli i requisiti, i tempi, i costi e le condizioni contrattuali;
3. **Contrattazione:** Composta dalla collaborazione tra fornitore e proponente_G per finalizzare i punti cardine della proposta;
4. **Pianificazione:** Composta dalla pianificazione delle attività necessarie per soddisfare i requisiti della proponente_G, seguita da una suddivisione delle ore produttive disponibili ed una stima dei costi;
5. **Esecuzione:** Composta dalla pianificazione e dallo sviluppo del prodotto in conformità ai requisiti concordati, insieme ad un monitoraggio continuo delle attività;
6. **Revisione:** Composta dalla verifica periodica del progresso rispetto ai criteri definiti dal contratto;
7. **Consegna:** Composta dalla consegna del prodotto software alla proponente_G, accompagnato dalla documentazione_G finale.

2.1.3 Rapporti con l'azienda proponente

L'azienda proponente_G SyncLab_G si è resa disponibile mediante diversi canali tra cui: e-mail, Discord e Google Meet ad una comunicazione frequente con il gruppo Seven Bits, così da risolvere tempestivamente eventuali domande o dubbi che possono emergere durante lo svolgimento del progetto_G.

Durante la prima riunione organizzativa con l'azienda, è stata definita l'organizzazione dei periodi di sprint_G, inizialmente stabilendo una durata di due settimane per ciascun ciclo. Tuttavia, a seguito della revisione RTB, questa è stata ridotta ad una settimana per accelerare il ritmo di lavoro e garantire il rispetto della data di consegna.

Al termine di ogni sprint_G avviene un incontro SAL_G (Stato di Avanzamento Lavori), durante il quale vengono analizzati i risultati del lavoro svolto e si procede con una sprint-review_G. Inoltre, tra un SAL_G e l'altro, secondo le necessità di proponente_G e team, è possibile concordare un incontro intermedio per monitorare i progressi raggiunti e rispondere ad eventuali quesiti emersi.

Ogni incontro con l'azienda viene formalizzato attraverso un verbale esterno. Tale verbale è successivamente sottoposto alla proponente_G per la validazione mediante firma, in modo da ottenere un'approvazione formale del resoconto delle discussioni svolte durante la riunione.

2.1.4 Documentazione fornita

Di seguito sono elencati i documenti che il gruppo si impegna a consegnare ai Committenti, Prof. Tullio Vardanega e Prof. Riccardo Cardin, nonché all'azienda proponente_G:

2.1.4.1 Analisi dei Requisiti

È un documento essenziale per lo sviluppo del prodotto software, che include la descrizione degli attori coinvolti, dei casi d'uso e l'elenco dei requisiti, suddivisi in requisiti funzionali, di qualità, di vincolo e prestazionali.

2.1.4.2 Piano di Progetto

È un documento che ha lo scopo di definire in modo chiaro le modalità con cui ogni membro del gruppo svolgerà le attività per la realizzazione del progetto_G. Include l'analisi dei rischi, la pianificazione delle attività, la suddivisione dei ruoli e la stima di costi e risorse.

2.1.4.3 Piano di Qualifica

È un documento che ha l'obiettivo di garantire la qualità del prodotto e dei processi durante l'intero ciclo-di-vita_G del progetto_G, per questo motivo sarà aggiornato nel tempo per riflettere eventuali modifiche e i risultati delle verifiche effettuate. Include le sezioni sulla qualità di processo_G, sulla qualità di prodotto, le modalità di testing e il cruscotto_G di valutazione delle qualità.

2.1.4.4 Specifica Tecnica

È un documento che descrive gli aspetti tecnici e progettuali del prodotto software, offrendo una visione dell'architettura implementativa e di deployment attraverso l'analisi di moduli, design pattern e scelte tecnologiche. Ha l'obiettivo di monitorare le decisioni progettuali, guidare lo sviluppo e garantire la tracciabilità dei requisiti, costituendo così un punto di riferimento fondamentale per tutti gli stakeholder.

2.1.4.5 Manuale Utente

È un documento che fornisce una guida sull'utilizzo della piattaforma software "NearYou - Smart custom advertising platform", sviluppata per l'azienda proponente SyncLab_G. Descrive le principali funzionalità, le modalità di interazione e le opzioni di personalizzazione, oltre a fornire informazioni sui requisiti minimi, le procedure di configurazione e come sfruttare al meglio le potenzialità del sistema.

2.1.4.6 Glossario

È un documento che raccoglie dei termini specifici e le loro definizioni chiare e concise. Il suo scopo è quello di facilitare la comprensione dei concetti chiave presenti nei vari documenti redatti.

2.1.4.7 Lettera di Presentazione

La Lettera di Presentazione è un documento che accompagna la consegna del prodotto software e della relativa documentazione_G durante le fasi di revisione di progetto_G. Il contenuto di questo documento comprende un link alla pagina web che contiene tutta la documentazione_G finora prodotta, un preventivo aggiornato rispetto a quello presentato alla revisione precedente, e dei link alle varie repository del gruppo.

2.1.5 Strumenti

Gli strumenti utilizzati per la gestione del processo_G di fornitura sono i seguenti:

- **Canva:** Piattaforma che permette la creazione di presentazioni multimediali, utilizzata per la realizzazione dei diari di bordo;
- **Google Meet e Discord:** Servizi che permettono di effettuare videochiamate, utilizzati dal team per le discussioni sincrone e asincrone con la proponente_G;
- **Google Sheets:** Servizio che permette la creazione di fogli di calcolo, utilizzato dal gruppo per la rendicontazione delle ore produttive impiegate durante ogni sprint_G.

2.2 Sviluppo

2.2.1 Descrizione e Scopo

Il processo_G di sviluppo rappresenta l'insieme delle attività necessarie per realizzare il prodotto software, garantendo il rispetto dei requisiti e delle scadenze concordate con la proponente_G. Questo processo_G si articola in diverse fasi fondamentali, quali l'analisi-dei-requisiti, la progettazione, la codifica, l'integrazione, e la verifica, assicurando che ogni fase contribuisca al raggiungimento degli obiettivi prefissati.

Le linee guida descritte in questa sezione sono volte a strutturare il lavoro in modo chiaro e uniforme, promuovendo la qualità del prodotto finale. Seguendo standard solidi e definiti, nel caso specifico l'ISO/IEC 12207:1995, si crea un ambiente di lavoro orientato a garantire la coerenza nei metodi utilizzati e il rispetto delle aspettative.

L'obiettivo è consegnare un prodotto software di alta qualità, che soddisfi le esigenze richieste, rispettando le tempistiche e garantendo il successo del progetto_G.

2.2.2 Analisi-dei-Requisiti

2.2.2.1 Descrizione e Scopo

L'Analisi-dei-Requisiti_G è la prima fase cruciale del processo_G di sviluppo software. Lo scopo principale di questa fase è definire con chiarezza le funzionalità e le caratteristiche che il sistema_G dovrà offrire, in base alle necessità degli utenti. Per raggiungere tale obiettivo, è essenziale stabilire una comunicazione efficace con la proponente_G, assicurandosi che tutte le esigenze siano documentate e validate. Questo processo_G permette di chiarire gli obiettivi del prodotto, identificare i vincoli operativi e fornire ai Progettisti le informazioni necessarie per sviluppare un'architettura coerente e un design adeguato. Il risultato di questa attività è formalizzato nel documento *Analisi-dei-Requisiti*, redatto dagli Analisti, che contiene una descrizione dettagliata degli obiettivi del prodotto, delle funzionalità previste e delle caratteristiche degli utenti. Include inoltre una sezione dedicata ai casi d'uso, che descrivono le interazioni tra gli attori e il sistema_G.

Infine, l'Analisi-dei-Requisiti_G contribuisce a migliorare la comunicazione tra tutti gli stakeholder_G, agevola la pianificazione del progetto_G in termini di tempistiche e costi e fornisce un riferimento chiaro per le attività di verifica e test_G.

2.2.2.2 Casi d'uso

I casi d'uso rappresentano scenari concreti che descrivono come gli utenti e altri attori interagiranno con il sistema_G per raggiungere obiettivi specifici. Questi scenari aiutano a identificare requisiti chiave e prevenire malintesi.

È importante sottolineare che i diagrammi dei casi d'uso non si concentrano sui dettagli implementativi. Il loro scopo è quello di rappresentare le funzionalità del sistema_G esclusivamente dal punto di vista esterno, evidenziando come esso interagisce con gli attori e soddisfa le loro esigenze.

Ogni caso-d'uso_G ha le seguenti caratteristiche:

- **Identificativo:**

UC [Numero caso-d'uso_G] . [Numero sottocaso d'uso] - [Titolo]

Dove:

- **Numero caso-d'uso_G**: ID relativo al caso-d'uso_G principale;
- **Numero sottocaso d'uso**: ID relativo allo specifico sottocaso d'uso (se applicabile);
- **Titolo**: Descrizione breve ed esplicativa;

- **Attore principale**: Entità esterna che interagisce attivamente con il sistema_G per raggiungere uno scopo, ad esempio un utente specifico o un altro sistema_G esterno;
- **Attore secondario**: Eventuale entità esterna invocata dal sistema_G per fornire un supporto, in modo da soddisfare il bisogno dell'attore principale;
- **Scenario principale**: Sequenza di eventi che si susseguono da quando un attore_G inizia ad interagire con il sistema_G;

- **Precondizioni:** Descrizione dello stato del sistema_G necessario per attivare il caso-d'uso_G;
- **Postcondizioni:** Descrizione dello stato del sistema_G al completamento del caso-d'uso_G;
- **Estensioni:** Eventuali scenari alternativi, descritti in questa sezione se presenti;
- **User story associata:** Funzionalità descritta dal punto di vista dell'utente, scritta in linguaggio naturale perché risulti semplice e comprensibile. Definita nella forma seguente:

- Come [utente], desidero poter [funzionalità] per [valore aggiunto]. -

2.2.2.3 Diagrammi dei casi d'uso

I diagrammi dei casi d'uso sono strumenti visivi che rappresentano le funzionalità del sistema_G dal punto di vista dell'utente. Mostrano come gli attori esterni (utenti, dispositivi, altri sistemi) interagiscono con il sistema_G, senza entrare nei dettagli tecnici. Il loro obiettivo principale è descrivere le interazioni tra attori e sistema_G, facilitando la comprensione dei requisiti funzionali e la comunicazione tra gli stakeholder_G. Ogni caso-d'uso_G descrive una sequenza di azioni che un attore_G compie per raggiungere un obiettivo specifico. Questi casi sono interconnessi e rappresentano i flussi di lavoro principali, evidenziando come l'attore interagisce con il sistema_G. I diagrammi non trattano i dettagli implementativi, ma si concentrano sulle funzionalità, trattando il sistema_G come un "black box" esterno.

Di seguito sono elencati i principali componenti di un diagramma dei casi d'uso:

• Attori

Gli attori sono entità esterne al sistema_G che interagiscono con esso per utilizzare le sue funzionalità. Possono essere utenti umani, altri sistemi software, dispositivi, macchine o organizzazioni. Un caso-d'uso_G definisce una specifica funzionalità che il sistema_G fornisce agli attori, senza entrare nei dettagli implementativi. Nel diagramma dei casi d'uso, gli attori sono rappresentati da figure stilizzate posizionate all'esterno del rettangolo che delinea il sistema_G, e ciascuno è identificato da un'etichetta con il suo nome.

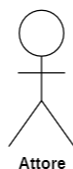


Figure 1: Rappresentazione di un attore_G.

• Casi d'uso

Ogni caso-d'uso_G identifica una specifica azione o funzionalità offerta dal sistema_G, con cui l'attore può interagire. I casi d'uso sono collegati tramite una linea continua agli attori che hanno accesso a quella funzionalità, creando una chiara relazione tra gli utenti e le azioni che possono compiere. Nel diagramma dei casi d'uso, ogni caso è rappresentato da una forma ovale contenente un ID univoco e un titolo esplicativo.

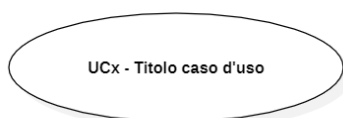


Figure 2: Rappresentazione di un caso-d'uso_G.

• Sottocasi d'uso

Un sottocaso d'uso rappresenta una versione più dettagliata di un caso-d'uso_G generico. Esso offre un livello di dettaglio più approfondito sulle funzionalità o sui particolari scenari di utilizzo rispetto al caso-d'uso_G principale. Non è obbligatorio per un caso-d'uso_G possedere uno o più sottocasi.

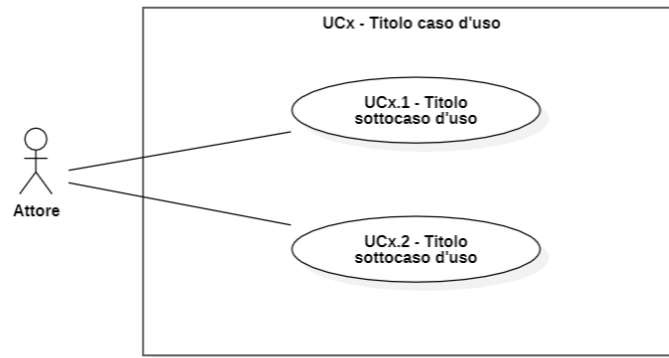


Figure 3: Rappresentazione di un sottocaso d'uso.

• Sistema

Il sistema_G viene rappresentato con un rettangolo all'interno del quale vengono collocati i casi d'uso. All'esterno del rettangolo sono invece posizionati i vari attori.

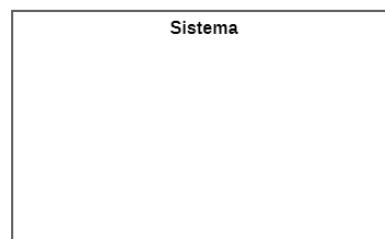


Figure 4: Rappresentazione di un sistema_G.

• Associazione

Una linea di associazione stabilisce una connessione tra un attore_G e un caso-d'uso_G quando l'attore è coinvolto nell'attività descritta dal caso-d'uso_G. Questo collegamento visivo rappresenta il ruolo dell'attore nell'attivare o nell'utilizzare una specifica funzionalità del sistema_G.



Figure 5: Rappresentazione di una associazione tra attore_G e UC.

• Generalizzazione (attori)

La generalizzazione tra attori rappresenta una relazione gerarchica in cui un attore_G specializzato

(figlio) eredita comportamenti e caratteristiche da un attore_G base (genitore). Questo meccanismo aiuta a organizzare gerarchicamente gli attori nei diagrammi dei casi d'uso, definendo i casi d'uso che un attore_G può utilizzare e che sono anche disponibili per gli attori più specializzati. La generalizzazione viene rappresentata con una linea solida e una freccia vuota che va dall'attore figlio all'attore genitore, indicando la direzione dell'eredità.

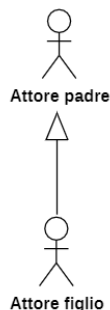


Figure 6: Rappresentazione di una generalizzazione tra attori.

• Inclusione

La relazione di inclusione indica quando un caso-d'uso_G (inclusente) comprende l'esecuzione di un altro caso-d'uso_G (incluso). Quando un attore_G interagisce con il caso-d'uso_G includente, il caso-d'uso_G incluso viene eseguito automaticamente come parte di quest'ultimo. Questa relazione è utile per il riutilizzo delle funzionalità e per evitare la duplicazione delle stesse logiche in più casi d'uso. La relazione di inclusione viene rappresentata da una freccia tratteggiata che collega il caso-d'uso_G incluso al caso-d'uso_G includente.

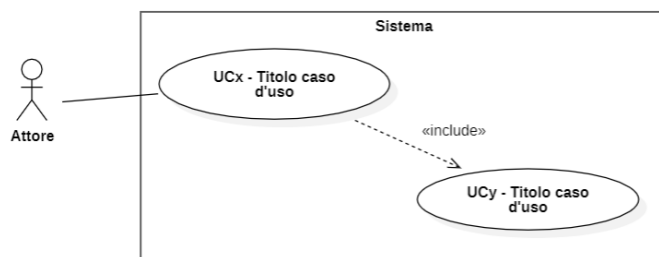


Figure 7: Rappresentazione di una inclusione tra UC.

• Estensione

La relazione di estensione indica quando un caso-d'uso_G (estendente) può modificare o arricchire il comportamento di un altro caso-d'uso_G (esteso) in determinate circostanze. Questa relazione viene utilizzata quando un evento o una condizione porta il flusso del caso-d'uso_G a deviare dallo scenario principale verso uno scenario alternativo, come nei casi di errore, ma mantenendo le stesse precondizioni del caso-d'uso_G principale. La relazione di estensione è rappresentata da una freccia tratteggiata che collega il caso-d'uso_G estendente al caso-d'uso_G esteso.

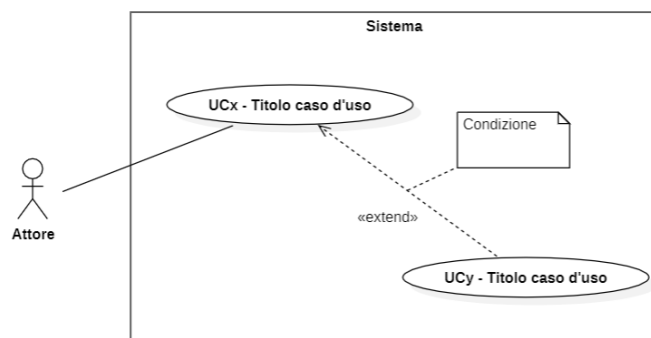


Figure 8: Rappresentazione di una estensione tra UC.

• Generalizzazione (casi d'uso)

La generalizzazione nei diagrammi dei casi d'uso rappresenta una relazione di ereditarietà tra casi d'uso, in cui un caso-d'uso_G più specifico eredita il comportamento da un caso-d'uso_G più generico. Questa relazione è utile per definire funzionalità più dettagliate, ed è simboleggiata da una linea con una freccia vuota che va dal caso-d'uso_G specifico al caso-d'uso_G generico.

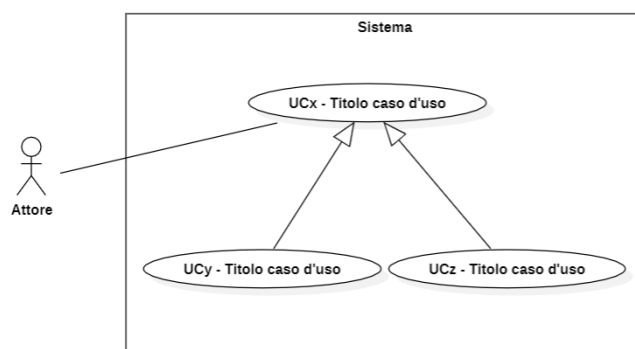


Figure 9: Rappresentazione di una generalizzazione tra UC.

2.2.2.4 Requisiti

I requisiti di un prodotto software sono specifiche documentate che delineano le funzionalità che il sistema_G deve soddisfare. Essi fungono da guida per lo sviluppo, il testing e la validazione del prodotto, garantendo che risponda alle esigenze degli utenti e agli obiettivi del progetto_G. Ogni requisito_G deve essere definito con precisione, e riflettere pienamente le attese del cliente o del proponente_G.

Ad ogni requisito_G è assegnato un grado di importanza:

- **Obbligatorio:** Essenziale per soddisfare le esigenze del proponente_G;
- **Desiderabile:** Utile ma non prioritario, può essere implementato in seguito;
- **Opzionale:** Aggiunge valore al prodotto, ma può essere tralasciato per motivi di costo o tempistica.

I requisiti si suddividono inoltre in tre diverse tipologie:

- **Requisiti funzionali (F):** Specificano le funzionalità che il sistema_G deve essere in grado di svolgere, descrivendo le azioni principali e le informazioni che fornisce;
- **Requisiti di qualità (Q):** Definiscono standard relativi alle prestazioni, affidabilità, sicurezza e altri criteri di qualità che il sistema_G deve rispettare;
- **Requisiti di vincolo (V):** Rappresentano restrizioni o condizioni imposte al progetto_G, come vincoli tecnologici o normativi;

- **Requisiti prestazionali (P):** Descrivono le capacità o le prestazioni minime richieste, come tempi di risposta, scalabilità o capacità di carico.

Ogni requisito_G è composto da:

- **Identificativo:** Un codice univoco nel formato:

R[Tipologia]X

Dove:

- **R:** Abbreviazione di "Requisito";
- **Tipologia:**
 - * **F:** Funzionale;
 - * **Q:** Qualità;
 - * **V:** Vincolo;
 - * **P:** Prestazionale.
- **X:** Numero progressivo per ogni requisito_G aggiunto.
- **Descrizione:** Una descrizione dettagliata che spiega la caratteristica richiesta al sistema_G;
- **Fonte:** L'origine del requisito_G;
- **Casi d'uso associati:** Un elenco di casi d'uso che forniscono il contesto operativo per il requisito_G;

Alcuni requisiti, pur non avendo un identificativo, vengono documentati per assicurare la loro tracciabilità. Questi comprendono:

- **Requisiti d'ambiente:** Specificano le condizioni e le risorse necessarie per sviluppare, testare e implementare il software;
- **Requisiti di sicurezza:** Delineano le misure e i comportamenti richiesti per proteggere il sistema_G da minacce;
- **Requisiti di performance:** Definiscono i livelli di prestazione richiesti, come velocità di elaborazione o capacità di gestione del carico.

2.2.2.5 Metriche

Metrica	Descrizione
MPD01	Requisiti Obbligatori Soddisfatti
MPD02	Requisiti Desiderabili Soddisfatti
MPD03	Requisiti Opzionali Soddisfatti

Table 2: Metriche relative all'attività di analisi-dei-requisiti

2.2.3 Progettazione

2.2.3.1 Descrizione e Scopo

La progettazione, o design, è un'altra fase cruciale del processo_G di sviluppo software che ha l'obiettivo di tradurre i requisiti individuati durante la fase di analisi in una struttura architeturale definita. Questa attività definisce in dettaglio i vincoli e le caratteristiche del prodotto semplificando la pianificazione e la suddivisione delle attività di codifica.

Prima di avviare la progettazione vera e propria, si intraprende una fase preliminare che prevede la creazione di un PoC_G (Proof of Concept), un prototipo preliminare "usa e getta" realizzato per dimostrare la fattibilità tecnologica del prodotto previsto. Questa fase serve a confermare le tecnologie da adottare e a definire insieme alla proponente_G le componenti principali che costituiranno l'MVP (Minimum Viable Product).

2.2.3.2 Qualità dell'architettura

Una buona architettura software è fondamentale per il successo di un progetto. Di seguito sono elencate le principali qualità che caratterizzano una buona architettura:

- **Sufficienza:** l'architettura soddisfa tutti i requisiti funzionali e non funzionali definiti per il sistema;
- **Comprensibilità:** l'architettura è facilmente comprensibile da tutti gli stakeholder, facilitando la manutenzione e l'evoluzione del sistema;
- **Modularità:** l'architettura è suddivisa in moduli separati e indipendenti, che possono essere sviluppati, testati e aggiornati autonomamente;
- **Robustezza:** l'architettura è progettata per essere resistente a diversi tipi di ingressi, gestendo correttamente anche condizioni di errore o situazioni impreviste;
- **Flessibilità:** l'architettura può adattarsi facilmente a nuovi requisiti, cambiamenti o evoluzione delle tecnologie senza richiedere modifiche radicali o strutturali;
- **Riusabilità:** le componenti e i moduli possono essere riutilizzati in contesti diversi, riducendo la duplicazione del lavoro e migliorando l'efficienza complessiva;
- **Efficienza:** l'architettura è progettata per ottimizzare l'utilizzo delle risorse, garantendo prestazioni elevate con il minor consumo di risorse possibile;
- **Affidabilità:** la capacità del software di svolgere correttamente le sue funzioni in modo coerente e prevedibile nel tempo;
- **Disponibilità:** il software è disponibile e operativo quando richiesto, riducendo al minimo i periodi di inattività;
- **Sicurezza rispetto a malfunzionamenti:** l'architettura è progettata per garantire che il sistema continui a funzionare correttamente anche in caso di guasti o malfunzionamenti;
- **Sicurezza rispetto a intrusioni:** il sistema è protetto da accessi non autorizzati e tentativi di intrusione, garantendo la riservatezza e l'integrità dei dati;
- **Semplicità:** l'architettura è il più semplice possibile, riducendo la complessità e facilitando la comprensione, la manutenzione e l'evoluzione;
- **Incapsulazione:** i dettagli implementativi interni delle componenti sono nascosti all'esterno, permettendo l'accesso solo attraverso un'interfaccia definita;
- **Coesione:** la misura in cui i componenti all'interno di un modulo o un'unità lavorano insieme per un obiettivo comune senza essere eccessivamente interdipendenti;
- **Basso accoppiamento:** il grado di dipendenza tra i diversi moduli o componenti del software deve essere minimizzato per migliorare la manutenibilità e flessibilità del sistema.

2.2.3.3 Diagrammi delle classi

Sono una tipologia di diagramma UML_G utile a rappresentare la struttura statica di un sistema_G software orientato agli oggetti. Questi diagrammi visualizzano le classi del sistema_G con i loro attributi e metodi insieme alle relazioni tra di esse.

Le classi sono rappresentate tramite rettangoli suddivisi in tre sezioni:

1. **Nome:** Contiene il nome della classe in grassetto, se la classe è astratta allora viene scritto anche in corsivo;
2. **Attributi:**

Visibilità nome : tipo [molteplicità] = default

- **Visibilità:** Se privata viene indicata con il - , se protetta viene indicata con il # e se pubblica viene indicata con il + ;
- **Nome:** Il nome dell'attributo, se statico viene sottolineato;

- **Tipo:** Rappresenta il tipo di dato dell'elemento;
- **Molteplicità:** Quante istanze dell'elemento possono esistere in relazione ad altri elementi;
- **Default:** Se configurato, indica il valore predefinito per l'elemento;

3. Metodi:

Visibilità nome (lista-parametri) : tipo-ritorno

- **Visibilità:** Segue quanto definito sopra;
- **Nome:** Nome del metodo, se statico viene sottolineato;
- **Lista-Parametri:** Se la funzione prevede più di un parametro, questi vengono separati tramite virgola;
- **Ritorno:** Il tipo restituito dal metodo;

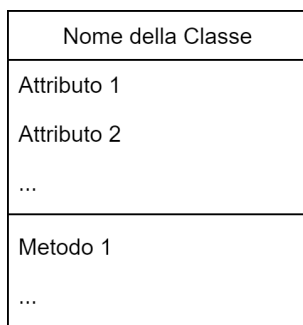


Figure 10: Rappresentazione di una classe.

Di seguito vengono elencate tutte le possibili relazioni:

- **Dipendenza:** La relazione di dipendenza tra due classi è rappresentata da una freccia tratteggiata con la punta, che parte dalla classe dipendente A e punta alla classe da cui si origina la dipendenza B. Questa freccia indica che eventuali modifiche nella classe B potrebbero influenzare o avere un impatto sulla classe A;

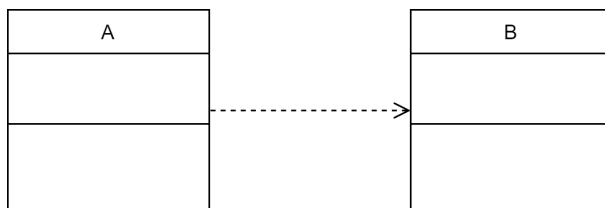


Figure 11: Rappresentazione della relazione di dipendenza.

- **Aggregazione:** La relazione di aggregazione rappresenta un legame di tipo "part of" tra due classi, in cui una classe è associata ad un'altra ma può esistere anche indipendentemente da essa. Questa relazione viene utilizzata quando una classe A (contenitore) contiene un riferimento ad un oggetto di tipo B (contenuto). L'aggregazione è rappresentata graficamente con una linea che collega le due classi, con un rombo vuoto posto vicino alla classe che rappresenta il contenitore;

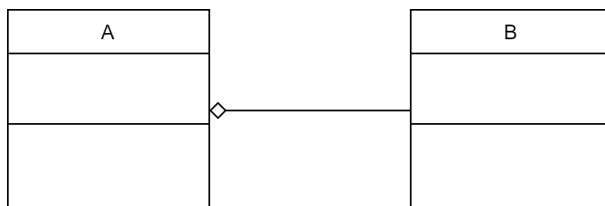


Figure 12: Rappresentazione della relazione di aggregazione.

- **Composizione:** La relazione di composizione si verifica quando una classe A contiene un oggetto di tipo B e ne ha la responsabilità di crearne e gestirne l'esistenza. La vita di B (contenuto) è vincolata a quella di A (contenitore): se A viene eliminata, anche B viene distrutta. La composizione è rappresentata graficamente con un rombo pieno vicino alla classe che rappresenta il contenitore e una linea che collega le due classi;

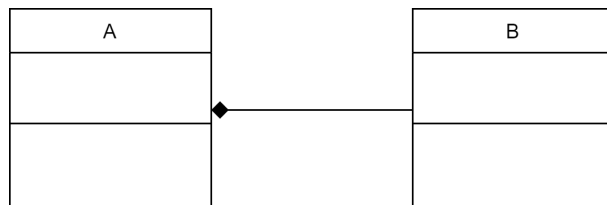


Figure 13: Rappresentazione della relazione di composizione.

- **Associazione:** La relazione di associazione si verifica quando una classe può contenere riferimenti o istanze di un'altra classe senza implicare una stretta dipendenza. L'associazione è rappresentata graficamente con una linea che collega le classi, spesso accompagnata da valori di molteplicità. Per indicare il senso della relazione, può essere utilizzata una freccia posizionata su uno dei due estremi;

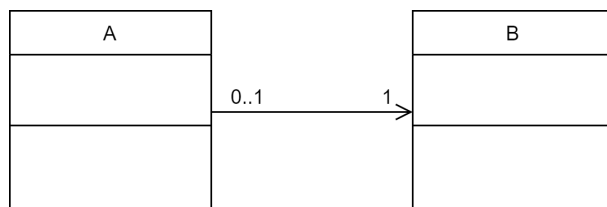


Figure 14: Rappresentazione della relazione di associazione.

- **Generalizzazione:** La relazione di generalizzazione si verifica quando una classe eredita attributi, comportamenti e relazioni dalla classe genitore. Nello specifico ogni istanza della classe figlia è anche un'istanza della classe genitore, ma con caratteristiche o dettagli aggiuntivi specifici. La generalizzazione è rappresentata graficamente con una freccia vuota che collega la classe figlia (B) alla classe genitore (A).

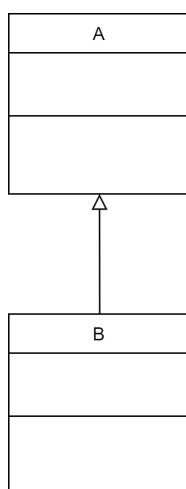


Figure 15: Rappresentazione della relazione di generalizzazione.

2.2.3.4 Design Pattern

I design pattern costituiscono delle soluzioni progettuali consolidate a problemi ricorrenti nello sviluppo software. Essi rappresentano delle best practices che permettono di migliorare la qualità, la manutenibilità

e la scalabilità del codice, facilitando la comprensione e la comunicazione tra sviluppatori. I design pattern non sono implementazioni specifiche, ma schemi generali che possono essere adattati a diversi contesti e linguaggi di programmazione. Essi si suddividono in tre categorie principali:

- **Creazionali:** gestiscono il processo di creazione degli oggetti in modo flessibile ed efficiente (es. Singleton, Builder o Abstract Factory);
- **Strutturali:** definiscono le modalità per semplificare e ottimizzare la struttura del codice (es. Adapter, Decorator, Facade o Proxy);
- **Comportamentali:** migliorano la comunicazione e cooperazione tra gli oggetti (es. Command, Iterator, Observer, Strategy o Template Method).

L'adozione dei design pattern può introdurre un certo grado di complessità nel codice, pertanto è fondamentale valutarne l'utilizzo con attenzione. È importante adottarli solo quando i benefici giustificano l'aumento della complessità, evitando di rendere il sistema più difficile da comprendere e gestire qualora non sia necessario.

2.2.3.5 Metriche

Metrica	Descrizione
MPD06	Tempo medio di risposta
MPD07	Facilità di utilizzo
MPD08	Tempo medio di apprendimento
MPD14	Versioni browser supportati

Table 3: Metriche relative all'attività di progettazione

2.2.4 Codifica

2.2.4.1 Descrizione e Scopo

L'attività di codifica rappresenta la fase cruciale in cui le funzionalità richieste dal proponente_G vengono tradotte in istruzioni eseguibili. I Programmatori, incaricati di questa attività, hanno il compito di implementare il design concettuale definito dai Progettisti, seguendo scrupolosamente le linee guida e le norme_G stabilite. Questo metodo di lavoro garantisce che il codice segua accuratamente le specifiche, promuovendo qualità, manutenibilità e coerenza nell'intero progetto_G.

L'obiettivo principale della codifica è quindi creare un prodotto software che soddisfi le esigenze del proponente_G e rispetti gli accordi contrattuali. Le aspettative del codice sviluppato includono:

- **Conformità alle specifiche:** Garantire che il codice traduca fedelmente i requisiti e le funzionalità richieste;
- **Chiarezza e leggibilità:** Scrivere codice autoesplicativo, riducendo la necessità di commenti superflui;
- **Ottimizzazione delle prestazioni:** Assicurare che il codice sia efficiente e scalabile;
- **Testabilità:** Integrare test_G di unità e di integrazione per verificare la correttezza e l'affidabilità.

2.2.4.2 Norme di codifica

Per garantire uno sviluppo del codice coerente e di alta qualità, i Programmatori adottano le seguenti regole:

- **Nomi significativi:** Utilizzare nomi univoci, oltre che chiari e descrittivi per variabili, metodi e classi, evitando abbreviazioni ambigue;
- **Indentazione:** Applicare uno schema di formattazione uniforme per migliorare la leggibilità del codice. Ogni livello di annidamento deve essere chiaramente separato con un tab o spazi coerenti;
- **Lunghezza dei metodi:** I metodi devono essere brevi e focalizzati su una singola responsabilità, per favorire test_G di unità efficaci.

2.2.4.3 Integrazione

Durante la fase di integrazione le diverse componenti, moduli o servizi del progetto_G vengono combinati e testati per creare un sistema_G coeso e funzionante.

L'obiettivo principale dell'integrazione è garantire che tutte le parti del sistema_G interagiscano correttamente e rispettino i requisiti funzionali e di prestazione definiti. Sin dalle prime fasi di sviluppo è necessario integrare le componenti del prodotto adottando un approccio solido ma flessibile, essenziale per garantire che le soluzioni esplorate durante la fase di Proof of Concept (PoC) siano utili una volta che il progetto_G entra nella fase di sviluppo del Minimum Viable Product (MVP).

Questa fase è supportata da test_G di integrazione che verificano che i moduli lavorino correttamente insieme. L'integrazione continua, supportata da strumenti automatizzati come CI_G/CD (Continuous Integration/Continuous Delivery), permette di garantire che ogni nuova modifica non comprometta l'affidabilità del sistema_G, mantenendo un flusso di lavoro agile_G e risolvendo eventuali problemi o incompatibilità prima che si presentino nel sistema_G finale.

2.2.4.4 Verifica

La verifica è una fase cruciale del ciclo-di-vita_G del software, durante la quale si controlla che il codice sviluppato rispetti le specifiche e soddisfi i requisiti stabiliti. Questo processo_G garantisce che il prodotto sia conforme agli standard di qualità, riducendo al minimo la presenza di errori e assicurando un'implementazione efficiente e mantenibile.

Il processo_G di verifica comprende sia un'analisi statica del codice, per rilevare eventuali errori o problemi di formattazione, sia l'esecuzione di test_G mirati per validare le funzionalità implementate.

2.2.4.5 Metriche

Metrica	Descrizione
MPD04	Line coverage
MPD05	Branch _G coverage
MPD09	Linee di codice per metodo
MPD10	Parametri per metodo
MPD11	Attributi per classe
MPD12	Structure Fan IN
MPD13	Structure Fan OUT

Table 4: Metriche relative all'attività di codifica

2.2.5 Strumenti

Gli strumenti utilizzati per la gestione del processo_G di sviluppo sono i seguenti:

- **Draw.io:** Software utilizzato per creare diagrammi e grafici di vario tipo, in particolare è stato impiegato per realizzare i diagrammi UML_G per i casi d'uso, nel documento di *Analisi dei Requisiti*;
- **StarUML:** Software per la modellazione UML utilizzato per creare diagrammi delle classi e altri diagrammi di progettazione software, in particolare è stato impiegato per definire l'architettura del nostro prodotto software;
- **Visual Studio Code:** L'IDE principale scelto per la codifica, grazie alla sua versatilità e facilità d'uso.

3 Processi di supporto

3.1 Documentazione

3.1.1 Descrizione e Scopo

La documentazione_G è l'insieme delle informazioni che accompagna lo sviluppo di un prodotto software, svolge un ruolo essenziale nella descrizione del prodotto per coloro che lo realizzano, lo distribuiscono e lo utilizzano.

Il suo scopo principale è quello di semplificare il lavoro dei membri del team durante l'intero ciclo-di-vita_G del progetto_G, monitorando tutti i processi e le attività coinvolte. Questo permette di migliorare il risultato finale e semplifica notevolmente la manutenzione.

3.1.2 Lista documenti

I documenti prodotti nel contesto della realizzazione del progetto_G sono:

- *Analisi_dei_Requisiti*
- *Glossario*
- *Norme_di_Progetto*
- *Piano_di_Progetto*
- *Piano_di_Qualifica*
- *Specifica_Tecnica*
- *Manuale_Utente*
- *Verbali_esterni*
- *Verbali_interni*

3.1.3 Template in L^AT_EX

Per la stesura dei documenti, viene utilizzato un template in formato L^AT_EX. Questo template ha lo scopo di semplificare la redazione dei documenti, garantire la coerenza e risparmiare del tempo, in modo da rendere la produzione dei documenti più efficiente e professionale. Sono stati sviluppati tre diversi modelli di template:

- Documenti ufficiali;
- Verbale Interno;
- Verbale Esterno.

3.1.4 Nomenclatura

La nomenclatura dei documenti prevede che il nome del file sia composto dal nome del documento e dalla versione, separati da underscore (_), come nell'esempio: "*Piano_di_Qualifica_v1.0.0.pdf*".

Nel caso dei verbali, la nomenclatura prevede l'uso del nome "VerbaleInterno" o "VerbaleEsterno", seguito dalla data nel formato "YYYY-MM-DD", con un trattino (-) che li unisce, come nell'esempio "*VerbaleEsterno-2024-11-10.pdf*".

3.1.5 Struttura documenti

3.1.5.1 Prima Pagina

- **Logo Team:** Situato in alto al centro;
- **Titolo:**
 - Nome del documento, qualora non sia un verbale;
 - Verbale Interno;
 - Verbale Esterno.
- **Sottotitolo:** Nome del capitolato_G;
- **Contatti:** Email del team;
- **Logo Università:** Situato in basso a destra.

3.1.5.2 Intestazione

Su ogni pagina del documento, eccetto la prima, si trova il logo del gruppo seguito dal titolo del documento e dalla sua versione.

3.1.5.3 Registro modifiche

Il registro delle modifiche è una tabella dettagliata che tiene traccia di ogni modifica avvenuta al documento nel corso del tempo. È utile per tenere traccia dell'evoluzione del documento e per consentire a chiunque stia lavorando sul progetto_G di comprendere quali modifiche sono state apportate e quando. L'intestazione comprende:

- **Versione:** Versione del documento;
- **Data:** Data della modifica apportata;
- **Autore:** Autore della modifica;
- **Verificatore:** Autore della verifica;
- **Descrizione:** Cosa è stato modificato o aggiunto al file.

3.1.5.4 Indice

Nella pagina successiva al registro delle modifiche è presente l'indice, che permette di facilitare la ricerca e la navigazione all'interno del documento.

3.1.5.5 Verbalì

I verbalì sono dei documenti di sintesi di un incontro che sia interno al team o esterno con l'azienda, per questo motivo la loro struttura è diversa rispetto agli altri documenti ufficiali.

La differenza più lampante è la mancanza del registro modifiche al loro interno, visto che sono documenti che non hanno bisogno di un ciclo di vita prolungato.

I verbalì hanno lo scopo di tenere traccia di chi ha partecipato agli incontri ed in particolare quali decisioni sono state prese.

Sono composti da 2 sezioni principali:

- **Durata e partecipanti:** Viene indicata la data di inizio e fine dell'incontro, insieme al luogo in cui si è svolto. Successivamente, vengono elencati i nomi dei partecipanti del gruppo. Se il verbale è esterno, vengono inclusi anche i partecipanti dell'azienda SyncLab_G;
- **Sintesi e Decisioni/Obiettivi:** La prima sezione contiene un riassunto degli argomenti trattati durante il meeting, anche detto: "ordine del giorno", mentre la seconda include un elenco delle decisioni prese, collegate alle issue_G corrispondenti tramite una tabella.

Inoltre, per i verbalì esterni è presente una sezione per la convalida del documento mediante una firma.

3.1.6 Convenzioni stilistiche

3.1.6.1 Stile del testo

- **Grassetto:** Viene utilizzato per i titoli di sezioni/sottosezioni/paragrafi di un documento e per le definizioni di termini negli elenchi puntati;
- **Corsivo:** Viene utilizzato per i nomi dei documenti;
- **Link:** Sono collegamenti ipertestuali, consentono di accedere a risorse esterne o interne, come altre pagine, sezioni, immagini o file, con un semplice click;
- **Glossario:** I termini che si possono ritrovare nel *Glossario* sono seguiti dall'apice G .

3.1.6.2 Formato delle date

É stato adottato il formato "YYYY-MM-DD", ovvero:

- **YYYY:** anno con 4 cifre;
- **MM:** mese con 2 cifre;
- **DD:** giorno con 2 cifre.

3.1.7 Strumenti

Il gruppo ha deciso di utilizzare i seguenti strumenti:

- **Git:** Strumento per il controllo di versione distribuito impiegato dal gruppo per gestire i documenti e le versioni successive di essi prodotte in maniera asincrona dai membri;
- **GitHub:** Piattaforma di versionamento $_G$ e repository $_G$ per la documentazione $_G$. Permette la gestione del codice sorgente, il controllo delle modifiche tramite funzionalità come le issue $_G$ e le pull-request (PR), facilitando la collaborazione all'interno di un team di sviluppo;
- **L^AT_EX:** Linguaggio scelto per la redazione dei documenti, spesso utilizzato in ambito accademico, scientifico e tecnico;
- **Overleaf:** Servizio utilizzato dal gruppo per la creazione e modifica sincrona da parte di due o più componenti di file .tex usati per produrre documentazione $_G$.

3.2 Gestione della configurazione

3.2.1 Descrizione e Scopo

Il processo $_G$ di gestione della configurazione identifica le norme $_G$ che sono state adottate durante lo svolgimento del progetto $_G$ al fine di garantire la tracciabilità, la coerenza e la comprensibilità del codice e della documentazione $_G$ prodotta dal gruppo.

Il suo scopo è gestire e organizzare le modifiche al codice o alla documentazione $_G$, garantendo che siano accompagnate dalle motivazioni alla base dei cambiamenti effettuati e dai relativi autori.

3.2.2 Project Board

Per la gestione delle attività viene utilizzata la board di Github $_G$, che offre una struttura organizzativa chiara ed efficiente. All'interno della board sono presenti diverse corsie di stato, ognuna delle quali descrive il progresso delle attività e sono suddivise come segue:

1. **ToDo:** attività identificate e pianificate ma ancora da avviare;
2. **In Progress:** attività in corso, assegnate ad uno o più membri del gruppo;
3. **Done:** attività completate, in attesa di verifica.

3.2.3 Ciclo di vita e Versionamento dei documenti

I documenti seguono due workflow distinti a seconda che si tratti di verbali (interni o esterni) oppure di documenti più corposi, e le versioni delle modifiche successive apportate ai documenti seguono il sistema_G di versionamento_G indicato di seguito.

3.2.3.1 Versionamento dei documenti

Il sistema_G adottato dal team per il versionamento_G dei documenti è il sistema_G di versionamento_G semantico: {x.y.z}, in cui ogni numero ha un significato specifico:

- "x" indica la versione maggiore, incrementata per cambiamenti incompatibili con versioni precedenti;
- "y" rappresenta la versione minore, usata per aggiungere informazioni compatibili;
- "z" è la versione di patch, aggiornata per correzioni di errori poco significativi e retrocompatibili.

Questo sistema_G aiuta il team a comprendere velocemente l'impatto di un aggiornamento fatto ad un documento prodotto.

3.2.3.2 Workflow verbali

I verbali seguono il seguente workflow:

1. Creazione del documento a partire da un template diverso a seconda che si tratti di un verbale interno o esterno;
2. Redazione del documento indicando la durata dell'incontro, i partecipanti (interni ed esterni) e la piattaforma utilizzata. Seguono l'ordine del giorno, la sintesi di quanto fatto e una descrizione di ciascuna delle considerazioni fatte e successive decisioni prese;
3. Nella sezione Decisioni prese si compila una tabella in cui ciascuna azione da intraprendere viene associata ad una issue_G corrispondente;
4. Creazione di una pull-request_G (PR) dal branch_G Verbali al branch_G Main;
5. Se ci sono correzioni o ulteriori modifiche da fare, queste devono essere indicate nella sezione Registro modifiche, e successivamente verificate;
6. Solo se la verifica dà esito positivo si può passare alla fase di approvazione, anch'essa da verificare; Da questo punto se il verbale è interno si passa all'ultimo passaggio mentre se il verbale è esterno va mandato all'azienda SyncLab_G perché venga firmato;
7. Quando il documento è completo l'ultimo Verificatore chiude la pull-request_G ed esegue il merge_G nel branch_G Main.

3.2.3.3 Workflow Documenti Ufficiali

I documenti ufficiali seguono il seguente workflow:

1. Creazione di un branch_G dedicato esclusivamente alla redazione di un documento specifico;
2. Creazione del documento a partire da un template comune a tutti i documenti (esclusi i verbali). La versione iniziale del documento corrisponde a {0.1.0};
3. Creazione di un branch di feature per la redazione di una sezione del documento, o per una modifica da apportare al documento partendo dal branch_G del documento;
4. Compilazione dei campi della sezione Registro modifiche;
5. Creazione di una pull-request dal branch_G di feature, precedentemente creato, al branch_G del documento desiderato;
6. Verifica della documentazione_G prodotta da parte del Verificatore indicato nel Registro modifiche e associato alle modifiche effettuate in quella seduta di lavoro;

7. Se ci sono correzioni o ulteriori modifiche da effettuare, queste devono essere prima indicate tramite suggestion, attraverso la sezione di Review della pull-request, e successivamente apportate alla feature in cui si sta lavorando;
8. Se la pull-request viene approvata, si esegue il merge_G nel branch del documento e si elimina il branch di feature creato;
9. Si ripetono le operazioni indicate dal punto {3} al punto {8} fino a quando il documento non è stato completato;
10. Quando il documento è completo si aprirà una pull-request verso il branch_G Main e, dopo l'approvazione del documento da parte dei due Verificatori presenti durante il periodo di lavoro, si esegue il merge_G nel branch_G Main. Successivamente, si procede ad eliminare il branch_G dedicato, solo se non sono previsti ulteriori sviluppi.

3.2.4 Ciclo di vita del software

Il ciclo di vita del software adottato si basa su un approccio ispirato al modello di Gitflow semplificato. In particolare, si fa uso di branch_G derivati da Main, con verifiche e merge che rispettano il workflow descritto di seguito.

3.2.4.1 Workflow Software

Il software segue il seguente workflow:

- Creazione di un nuovo branch_G funzionale derivato da un branch_G collettivo, il quale a sua volta deriva direttamente dal branch_G Main. Ogni branch_G collettivo è dedicato a un gruppo di funzionalità correlate o a una specifica fase dello sviluppo;
- Sviluppo della parte assegnata nel branch_G funzionale. Ogni modifica rilevante deve essere tracciata mediante issue_G associata;
- Test locale e/o automatizzato della funzionalità sviluppata, con l'obiettivo di garantirne il corretto funzionamento e l'assenza di regressioni. Al termine, viene aperta una pull-request_G (PR) per richiedere il merge_G nel branch_G collettivo;
- Merge_G del branch_G funzionale nel branch_G collettivo, effettuato solo dopo la validazione della funzionalità da parte di un Verificatore;
- Eliminazione del branch_G funzionale una volta completato il merge_G nel branch_G collettivo, al fine di mantenere ordinata la struttura della repository;
- Verifica del branch_G collettivo da parte del Verificatore, che controlla la coerenza tra le parti, l'integrabilità delle componenti e l'assenza di conflitti. Tale verifica include l'analisi del risultato delle GitHub Actions, che eseguono test automatizzati e restituiscono metriche tramite badge integrati nella PR;
- Creazione di una pull-request_G dal branch_G collettivo al branch_G Main, subordinata all'approvazione di tutte le modifiche da parte del Verificatore;
- Solo a seguito della verifica positiva e dell'approvazione formale della PR è consentito il merge_G definitivo nel branch_G Main.

3.2.5 Repository

Il gruppo utilizza tre repository_G all'interno della propria organizzazione Github_G:

- <https://github.com/SevenBitsSwe/7BitsDocs>: questa repository_G è utilizzata per revisionare e mantenere aggiornati i documenti redatti, permettendo ai membri del gruppo di gestire in modo efficiente la documentazione_G;
- <https://github.com/SevenBitsSwe/PoC>: questa repository_G è dedicata alla condivisione e verifica del codice sorgente relativo al prodotto software realizzato per la revisione RTB_G. Viene utilizzata dal team per lavorare sul codice del prodotto, consentendo una gestione centralizzata dello sviluppo e delle modifiche apportate al software;

- <https://github.com/SevenBitsSwe/MVP>: questa repository_G è usata per la condivisione e la verifica del codice sorgente relativo al prodotto software realizzato per la revisione PB_G. Viene utilizzata dal team per lavorare sul codice del MVP.

3.2.5.1 Struttura della repository 7BitsDocs

Per ogni documento è stato creato un branch_G dedicato, al fine di lavorare in modo efficace ed efficiente, evitando di generare conflitti e prevenendo il caricamento sulla pagina web di documenti non ancora approvati, ossia con una versione precedente alla 1.0.0. Per questo motivo, è stato adottato un modello git-flow semplificato, che prevede l'uso di più branch_G di lavoro separati. Di conseguenza, nessuno dei branch_G dedicati deve essere eliminato.

Ogni branch_G, derivante direttamente dal branch_G principale Main, contiene diverse cartelle relative alle fasi del progetto_G: una per la Candidatura e una per la RTB_G, ciascuna contenente i documenti redatti per quella specifica fase. Inoltre, sono presenti una cartella per le immagini utilizzate nei documenti, due cartelle per la gestione della pagina web e i file di supporto (README, templates, ecc.).

3.2.5.2 Struttura della repository PoC

Per quanto riguarda il prodotto software sviluppato per la revisione RTB_G, sono stati creati diversi branch_G, ognuno derivante direttamente dal branch_G Main, e ciascuno utilizzato per esplorare e approfondire una specifica tecnologia. Solo una volta presa la decisione di implementare una determinata tecnologia, viene effettuato un merge_G in un branch_G specifico che include l'intero prodotto. Solo quest'ultimo può successivamente confluire nel branch_G Main attraverso una pull-request, a condizione che ogni modifica sia stata verificata e approvata.

3.2.5.3 Struttura della repository MVP

Per il prodotto software sviluppato per la fase di PB_G, sono stati creati diversi branch_G, ognuno derivante direttamente dal branch_G Main, e ciascuno utilizzato per sviluppare una diversa parte del prodotto software. Solo dopo aver completato e testato una parte del prodotto finale, viene effettuato un merge_G in un branch_G specifico che include l'intero prodotto. Solo quest'ultimo può successivamente confluire nel branch_G Main attraverso una pull-request, a condizione che ogni modifica sia stata verificata e approvata.

3.2.6 Pagina web

La pagina web, accessibile al link <https://sevenbitsswe.github.io/7BitsDocs/>, è stata progettata per offrire un'esperienza chiara e intuitiva, organizzando la documentazione_G in base alle diverse fasi per cui è stata redatta. Grazie all'integrazione di una Github_G Action, la pagina si aggiorna automaticamente ad ogni modifica apportata nel branch_G Main del repository_G 7BitsDocs, garantendo che siano visibili soltanto i documenti approvati.

3.2.7 Controllo dei termini del glossario

È stato sviluppato uno script in Python_G per verificare la presenza dei termini definiti nel *Glossario* all'interno dei vari documenti. Lo script utilizza espressioni regolari per identificare i termini nei documenti e confrontarli con quelli inclusi nella lista predefinita, formattandoli secondo lo standard del *Glossario*. Quando rileva un termine della lista non correttamente formattato, lo script aggiunge automaticamente una _G a pedice per garantirne la conformità.

Tutti i termini inclusi nel *Glossario* devono essere formattati correttamente ogni volta che compaiono nel documento, e non solo alla loro prima occorrenza. Questa regola semplifica la consultazione e favorisce una migliore comprensione dei termini chiave all'interno della documentazione_G.

3.3 Verifica

3.3.1 Descrizione e Scopo

La verifica è un processo_G essenziale che deve essere applicato a tutti gli altri processi per poterli considerare completati. Il suo scopo principale è quello di assicurare che i prodotti siano corretti e che aderiscano ai vincoli individuati ed elencati nel documento *Piano_di_Qualifica*. Le attività di verifica sono assegnate ai verificatori, che si impegnano a far rispettare i vincoli di qualità presenti nel *Piano_di_Qualifica*, correggendo e/o segnalando qualora siano presenti contenuti non conformi agli standard.

3.3.2 Strumenti

Lo strumento utilizzato per agevolare il processo_G di verifica è il seguente:

- **GitHub:** Piattaforma di versionamento_G e repository_G che fornisce un servizio di code review integrato nel meccanismo delle pull-request, consentendo ai verificatori di visualizzare le ultime modifiche. Dopo aver completato la verifica del documento, il Verificatore può aggiungere commenti con correzioni o suggerimenti di miglioramento e richiedere delle modifiche. In alternativa, può approvarla e passare alla verifica di altri prodotti in attesa di revisione nel repository_G. Il sistema_G di review di Github_G permette all'intero team di tracciare le modifiche apportate nel repository_G. Inoltre, è possibile assegnare regole alle pull-request per garantire che la verifica venga effettuata da almeno due verificatori prima di eseguire il merge_G nel branch_G Main.

3.3.3 Analisi statica

L'analisi statica è un tipo di verifica che viene effettuata sul prodotto software o sulla documentazione_G senza dover eseguire il sistema_G software. Serve ad accertare la conformità con le regole adottate, verificare l'assenza di difetti e la presenza di proprietà desiderate, garantendo un livello ottimale di qualità. Le due principali tecniche sono Inspection_G e Walkthrough_G. Il gruppo preferisce l'approccio ad Inspection_G, così da ridurre il tempo di verifica affinché l'autore possa dedicarsi alla stesura di altri documenti o codice.

3.3.3.1 Inspection

Il Verificatore segue una sequenza di passaggi prestabiliti per verificare che si stiano rispettando le regole presenti nel *Piano-di-Qualifica*.

3.3.3.2 Walkthrough

Il Verificatore e l'autore del prodotto da verificare discutono sulle modifiche da apportare. Questo processo_G richiede la presenza sincrona e la collaborazione di entrambi i ruoli, dunque risulta più complessa dell'Inspection in termini organizzativi. Viene effettuato un controllo ad ampio spettro dell'oggetto, senza avere un'idea precisa di cosa andare a verificare.

3.3.4 Analisi dinamica

L'analisi dinamica viene condotta direttamente sul prodotto software attraverso la sua esecuzione effettiva. Questo tipo di analisi si suddivide in diverse tipologie di test_G, ciascuna finalizzata a verificare aspetti specifici del funzionamento software. Ogni test_G corrisponde all'esecuzione di un programma, e simula il comportamento di singole parti di codice su un insieme finito di casi di prova.

Questa procedura misura la qualità del prodotto e assicura che esso raggiunga correttamente il suo scopo. I test_G effettuati devono essere ripetibili nel tempo per assicurare un funzionamento persistente durante tutto lo svolgimento del prodotto software.

3.3.5 Classificazione dei test

Tutti i test_G che verranno eseguiti sono presenti nel documento *Piano-di-Qualifica* ed il Verificatore è tenuto a svolgerli, riportandone gli esiti all'interno dello stesso documento.

Ad ogni test_G devono essere attribuiti:

- **Stato iniziale:** parametri del software al momento dell'esecuzione del test_G;
- **Input:** dati forniti in entrata per l'esecuzione del test_G;
- **Output:** risultati attesi in uscita, dato uno specifico input.

Inoltre i test_G sono divisi in varie categorie:

3.3.5.1 Test di unità

Test_G eseguiti su singole unità del software, come funzioni o metodi, in modo indipendente dal resto del sistema_G. L'obiettivo è assicurare che ogni unità di codice funzioni correttamente.

Devono essere eseguiti per primi visto che le unità garantiscono il corretto funzionamento dei singoli metodi che poi verranno integrati tra di loro.

Sono presenti due tipi di test_G di unità:

- **Funzionale:** verificano che ogni unità esegua le funzioni specificate nel design, facendo riferimento solo alla specifica input-output dell'oggetto di verifica;
- **Strutturale:** verificano la logica interna dell'oggetto di verifica esaminando il codice sorgente dell'unità.

3.3.5.2 Test di integrazione

Test_G eseguiti per controllare la corretta interazione tra le varie unità del software. Questo tipo di test_G vengono eseguiti dopo la conferma che tutti i test_G di unità siano passati con successo.

Vi sono due tipi di integrazione:

- **Bottom-up:** l'integrazione parte dalle componenti di sistema_G con minori dipendenze e un maggiore valore interno. Richiede pochi mock ma ritarda la messa a disposizione di funzionalità visibili all'utente;
- **Top-down:** l'integrazione parte dalle componenti di sistema_G con maggiori dipendenze e un maggiore valore esterno. Richiede molti mock ma integra prima le funzionalità visibili all'utente.

3.3.5.3 Test di sistema

Test_G eseguiti per controllare che i requisiti specificati durante l'analisi-dei-requisiti siano rispettati, garantendo che tutte le componenti siano integrate correttamente.

L'obiettivo di questi test_G è di accertarsi che l'intero sistema_G lavori come entità unica, verificando che tutte le singole componenti funzionino insieme. Vengono eseguiti al completamento dei test_G di integrazione.

3.3.5.4 Test di regressione

Garantiscono che le modifiche apportate al codice non introducano nuovi difetti o danneggino il sistema_G precedentemente funzionante.

Consistono nella ripetizione selettiva delle tipologie dei test_G elencati precedentemente e devono essere eseguiti ogni volta che si effettuano modifiche al codice.

3.3.5.5 Test di accettazione

Sono gli ultimi test_G da passare prima del rilascio del software. Servono a garantire che il prodotto finale soddisfi le aspettative del proponente_G e degli utenti finali.

3.3.5.6 Classificazione dei test

I test_G vengono identificati in base alla loro tipologia e tramite un codice numerico, specifico all'interno della categoria. I test_G devono avere la seguente forma:

T[Tipologia Test][Codice]

Dove:

- **Tipologia:**

- * **U:** Unità;
- * **I:** Integrazione;
- * **S:** Sistema_G;
- * **A:** Accettazione.

3.3.5.7 Stato dei test

Nel *Piano di Qualifica* ogni test_G viene seguito dal suo stato:

- **NI:** Il test_G non è stato implementato;
- **NP:** il test_G non è stato passato, con esito negativo;
- **P:** il test_G è stato passato, con esito positivo.

3.4 Gestione della Qualità

3.4.1 Descrizione e Scopo

La qualità viene assicurata tramite processi di verifica e validazione, che permettono di monitorare l'aderenza agli standard definiti. Una volta stabiliti i criteri qualitativi nel *Piano_di_Qualifica*, il gruppo si impegna a seguire le best-practices per ogni fase del progetto_G, dai processi di sviluppo alla produzione degli artefatti. La responsabilità di garantire l'applicazione degli standard è delegata ai Verificatori, che certificano la conformità dei prodotti e dei processi agli obiettivi qualitativi.

Il processo_G di gestione della qualità ha come obiettivo primario garantire che il software e i processi del ciclo-di-vita_G del progetto_G rispettino i requisiti specificati e siano allineati ai piani stabiliti. Attraverso l'adozione di standard qualitativi ben definiti, si mira a creare un prodotto affidabile ed efficiente, conforme alle aspettative e agli obiettivi concordati con il proponente_G.

3.4.2 Aspettative

CI_G si attende che:

- Il gruppo rispetti gli standard qualitativi definiti di comune accordo;
- La documentazione_G prodotta sia chiara, coerente e conforme agli obiettivi del progetto_G;
- I prodotti software sviluppati soddisfino i requisiti funzionali e non funzionali concordati;
- Le pratiche di verifica e validazione siano eseguite regolarmente e con accuratezza.

3.4.3 Metriche e Strumenti

Per monitorare e valutare la qualità, vengono utilizzate specifiche metriche, adattate ai diversi processi coinvolti. Queste metriche forniscono uno strumento oggettivo per analizzare il lavoro svolto e identificare eventuali criticità. Le metriche sono descritte nel dettaglio nel *Piano_di_Qualifica* e includono:

- **Metriche di processo_G:** utilizzate per valutare l'efficienza e l'efficacia dei processi produttivi e organizzativi;
- **Metriche di prodotto:** impiegate per analizzare la qualità degli artefatti e dei prodotti finali.

3.4.4 Struttura e identificazione delle metriche

Le metriche adottate nel progetto_G sono identificate e descritte secondo una struttura standardizzata che ne facilita il riconoscimento e l'utilizzo. Ogni metrica è caratterizzata dai seguenti elementi:

- **Codice:** identificativo univoco della metrica, definito nel formato:

M[abbreviazione][numero]

Dove:

- [abbreviazione]: PC se si tratta di qualità di processo_G, PD se si tratta di qualità di prodotto.
- [numero]: numero progressivo univoco per ciascuna metrica;

- **Nome:** il nome completo della metrica;
- **Descrizione:** breve descrizione della funzionalità e dell'ambito di applicazione della metrica.

Eventualmente, le metriche possono includere:

- **Formula:** modalità di calcolo della metrica.

3.4.5 Documentazione della Qualità

La gestione della qualità è documentata nel *Piano_di_Qualifica*, che definisce le metriche e le soglie accettabili per i vari processi e prodotti. Eventuali modifiche agli standard o agli strumenti vengono tracciate e aggiornate in modo continuo per garantire un processo_G di miglioramento costante.

4 Processi organizzativi

4.1 Gestione dei processi

4.1.1 Descrizione e Scopo

Come stabilito dallo standard ISO/IEC 12207:1997, il processo_G di gestione comprende tutte le attività necessarie per completare un progetto_G software, in modo tale da garantire il raggiungimento degli obiettivi prefissati nel rispetto dei requisiti, dei tempi e dei costi. Per conseguire questo obiettivo, questo processo_G si compone di una pianificazione dettagliata, un monitoraggio continuo dei progressi e l'adozione tempestiva di azioni correttive, quando necessario.

4.1.2 Attività

Il processo_G di gestione, come stabilito dallo standard ISO/IEC 12207:1997, si compone delle seguenti attività:

1. **Pianificazione:** Composta dall'identificazione degli obiettivi da raggiungere, dalla stima delle risorse necessarie, dalla definizione dei ruoli coinvolti e dalle scadenze da rispettare;
2. **Monitoraggio e controllo:** Composto dalla verifica continua dello stato di avanzamento rispetto alla pianificazione, con un'analisi degli eventuali scostamenti su tempi, costi e risorse. Se necessario, vengono attuate azioni correttive per riallineare il progetto_G agli obiettivi;
3. **Gestione dei rischi:** Composta dall'identificazione dei potenziali rischi, dalla valutazione del loro impatto e della probabilità di occorrenza, e dallo sviluppo ed implementazione di piani di attenuazione per ridurre al minimo gli effetti negativi che potrebbero derivarne;
4. **Gestione delle comunicazioni:** Composta dalla gestione delle comunicazioni per garantire un flusso costante di informazioni tra tutte le parti interessate, sia interne al gruppo che esterne come l'azienda proponente_G, assicurando che le comunicazioni siano chiare e tempestive;
5. **Gestione delle modifiche:** Composta dal riconoscimento e dall'approvazione delle richieste di modifica, dalla valutazione del loro impatto sui tempi e sui costi, e dall'aggiornamento della pianificazione per integrare le modifiche approvate;
6. **Revisione e chiusura:** Composta dalla valutazione complessiva del progetto_G al termine delle attività, dall'identificazione delle nozioni apprese e dalla formalizzazione della chiusura del progetto_G con la consegna dei risultati finali.

4.1.3 Pianificazione degli sprint

Include la durata di ogni sprint_G e una lista degli obiettivi da raggiungere entro la sua conclusione. Se si verificano dei rischi, viene aggiunta una sezione che descrive come il gruppo li ha gestiti e l'impatto che hanno avuto sulle attività pianificate. Inoltre, la pianificazione contiene una tabella con i ruoli assegnati ai membri del gruppo e due sezioni separate: una per il preventivo, che indica le ore e i costi stimati per lo sprint_G, e una per il consuntivo, che riporta le ore e i costi effettivi per lo stesso periodo.

4.1.4 Ruoli di progetto

Durante l'intero sviluppo del progetto_G, ciascun membro del gruppo assume sei ruoli distinti, ciascuno con responsabilità ben definite che riguardano specifiche attività di specifici processi. Ogni ruolo contribuisce al raggiungimento degli obiettivi del progetto_G, risultando essenziale per completare tutte le attività necessarie alla creazione di un prodotto finale di qualità.

I ruoli assunti a rotazione da ciascun membro del gruppo sono i seguenti:

4.1.4.1 Responsabile

Figura incaricata di supervisionare e coordinare le attività del gruppo, assicurandosi il raggiungimento degli obiettivi del progetto_G nei tempi e nei modi previsti. Si occupa di garantire una comunicazione efficace tra il gruppo e la proponente_G, pianifica e gestisce le risorse e valuta eventuali scelte e rischi che ne derivano.

I suoi compiti sono:

- Comunicare con la proponente_G attraverso i canali concordati;
- Stesura e avanzamento del documento *Piano_di_Progetto*;
- Redazione dei verbali interni ed esterni;
- Creazione e assegnazione delle issue_G ai membri del team;
- Creazione dei diari di bordo.

4.1.4.2 Amministratore

Figura responsabile della creazione, manutenzione e ottimizzazione degli strumenti, delle risorse e dei processi necessari per il corretto svolgimento del progetto_G. Supervisiona la configurazione degli ambienti di lavoro e il monitoraggio delle scadenze amministrative, fornendo supporto al gruppo nello svolgimento delle attività.

I suoi compiti sono:

- Configurazione e gestione gli ambienti di lavoro;
- Controllare e mantenere gli strumenti necessari per la collaborazione e la comunicazione del gruppo;
- Gestione del versionamento_G dei documenti;
- Stesura e avanzamento del documento *Norme_di_Progetto*.

4.1.4.3 Analista

Figura responsabile di raccogliere, analizzare e documentare i requisiti del progetto_G, traducendoli in casi d'uso. Collabora con la proponente_G per comprendere tutte le specifiche e verifica che il prodotto finale risponda alle esigenze e alle aspettative richieste.

I suoi compiti sono:

- Studiare il dominio del problema e relativo contesto applicativo;
- Studiare i bisogni dei committenti;
- Stesura del documento *Analisi_dei_Requisiti*;
- Stesura e avanzamento del documento *Piano_di_Qualifica*;
- Studiare i requisiti definendo la loro complessità.

4.1.4.4 Progettista

Figura specializzata nella progettazione architeturale e strutturale di sistemi software. Si occupa di tradurre i requisiti identificati dagli Analisti in un'architettura dettagliata, inoltre determina le scelte realizzative del progetto_G e supervisiona lo sviluppo, senza occuparsi direttamente della manutenzione.

I suoi compiti sono:

- Determinare le tecnologie, i linguaggi e i framework_G da utilizzare;
- Progettare l'architettura del sistema_G, definendo componenti, moduli e interazioni;
- Gestire eventuali rischi che possono sorgere durante lo sviluppo;
- Supervisionare lo sviluppo per garantire la conformità all'architettura.
- Stesura e avanzamento del documento *Specifica_Tecnica*.

4.1.4.5 Programmatore

Figura incaricata di trasformare l'architettura proposta dai Progettisti in un sistema_G software funzionante. Si occupa dello sviluppo delle componenti principali e di supporto, seguendo specifiche tecniche e garantendo l'aderenza agli standard di qualità e alle linee guida del progetto_G.

I suoi compiti sono:

- Tradurre le specifiche tecniche in codice funzionante;
- Creazione di test_G per la verifica e validazione del software;
- Scrittura di codice chiaro, leggibile e manutenibile;
- Risoluzione di eventuali bug_G.
- Stesura e avanzamento del documento *Manuale_Utente*.

4.1.4.6 Verificatore

Figura che si occupa di controllare che il lavoro svolto dagli altri membri del gruppo sia corretto, assicurandosi che vengano rispettate tutte le norme_G stabilite per il progetto_G.

I suoi compiti sono:

- Revisionare e valutare la documentazione_G prodotta dai membri del gruppo;
- Analizzare il codice per individuare errori, discrepanze o possibili miglioramenti;
- Identificare e segnalare eventuali problemi.

4.1.5 Cambio dei ruoli

Per garantire che, al completamento del progetto_G, ogni membro abbia ricoperto tutti i ruoli indicati, il gruppo si impegna a riassegnare i ruoli ad ogni componente del gruppo all'inizio di ogni sprint_G.

4.1.6 Tracciamento delle ore

Per tracciare il tempo impiegato nei diversi ruoli durante il progetto_G, viene utilizzato uno spreadsheet dedicato, disponibile su Google Drive. Al termine di ogni sprint_G, viene utilizzato per generare il consuntivo dello sprint_G appena concluso e il preventivo di quello appena iniziato, infine il membro con il ruolo di Responsabile ha il compito di inserire questi elementi all'interno del documento *Piano di Progetto*.

4.1.6.1 Preventivo

Il preventivo delle ore è presentato sotto forma di tabella, che dettaglia le ore stimate per ogni membro del gruppo, suddivise per ruolo e riferite al singolo sprint_G. Inoltre, è incluso un diagramma circolare che mostra la ripartizione delle ore stimate per ciascun ruolo, offrendo una visione chiara e immediata delle risorse che si pensa di allocare per quello specifico sprint_G.

4.1.6.2 Consuntivo

Il consuntivo delle ore è presentato sotto forma di tabella, che dettaglia le ore effettive registrate per ogni membro del gruppo, suddivise per ruolo e riferite al singolo sprint_G. Inoltre, è incluso un diagramma circolare che mostra la ripartizione delle ore effettive per ciascun ruolo, fornendo una visione chiara e immediata delle risorse effettivamente allocate durante lo sprint_G.

4.1.7 Ticketing

Il gruppo utilizza il sistema_G di Issue_G Tracking (ITS) integrato di Github_G per la gestione delle attività. Questo strumento offre una gestione semplice e chiara dei compiti da svolgere, grazie alla possibilità di creare e chiudere le issue_G in modo rapido ed efficiente. Lo stato di avanzamento delle attività è visualizzabile attraverso la project board integrata di Github_G, che garantisce una panoramica chiara e aggiornata del progresso del lavoro.

Le issue_G sono composte da:

- **Titolo:** identifica in modo univoco il compito da svolgere;
- **Descrizione:** fornisce informazioni dettagliate e utili per il completamento del compito;
- **Assegnatario:** specifica il membro del gruppo incaricato di svolgere il compito;
- **Etichetta:** classifica la tipologia del compito, agevolando l'organizzazione delle issue_G;
- **Board:** indica a quale project board appartiene la issue_G e permette di specificare ulteriori campi, tra cui:
 - **Stato:** rappresenta l'avanzamento del compito (Todo, In Progress, Done);
 - **Priorità:** assegna un livello di urgenza o importanza alla issue_G (Urgente, Importante, Moderato, Secondario);
 - **Grandezza:** stima la complessità o il livello di sforzo richiesto per completare il compito (XS, S, M, L, XL);
 - **Iterazione:** specifica a quale sprint_G la issue_G è assegnata;

- **Data di inizio:** data in cui viene iniziato il lavoro descritto dalla issue;
- **Data di fine:** data in cui viene terminata la issue.

- **Milestone:** identifica il traguardo o obiettivo specifico a cui la issue_G è associata.

Ogni volta che è necessario completare un compito, si segue la procedura descritta di seguito:

1. Il Responsabile crea la issue_G su Github_G, definendone i dettagli principali;
2. La issue_G viene assegnata ad un membro del gruppo, che si occuperà del compito;
3. La issue_G viene associata all'iterazione dello sprint_G attuale;
4. Il membro incaricato sposta la issue_G nello stato "In Progress" quando inizia a lavorarci;
5. Al completamento del compito, il Verificatore designato viene informato per procedere al controllo;
6. Il Verificatore esamina il lavoro svolto e comunica l'esito della verifica tramite Github_G;
7. Se la verifica ha esito positivo, la issue_G viene chiusa e spostata nello stato "Done".

4.1.8 Rischi

La gestione dei rischi è un processo_G volto a identificare, valutare e affrontare le situazioni avverse che potrebbero influire sul successo del progetto_G. Questo approccio prevede l'individuazione dei rischi, la valutazione della loro gravità e occorrenza, la definizione di strategie per mitigarli ed, infine, un monitoraggio costante per rilevare la possibile comparsa di nuovi rischi.

I rischi vengono suddivisi in tre diverse categorie:

- Rischi Tecnologici;
- Rischi di Organizzazione;
- Rischi di Pianificazione.

Ogni rischio_G viene classificato seguendo il formato:

R[Tipo][Indice] - [Nome]

dove:

- **R:** Abbreviazione di "Rischio";
- **Tipo:** Categoria del rischio_G (T per Tecnologico, O per Organizzazione, P per Pianificazione);
- **Indice:** Identifica univocamente ciascun rischio_G;
- **Nome:** Nome del rischio_G.

4.1.9 Coordinamento

Questa attività comprende la gestione delle comunicazioni e degli incontri tra tutte le parti coinvolte nel progetto_G, ossia i membri del gruppo, la proponente_G e i committenti.

4.1.10 Comunicazioni

Le comunicazioni sono fondamentali per mantenere un flusso costante di informazioni tra le parti coinvolte, garantendo efficienza_G e coordinamento nel progetto_G, supportando il buon andamento delle attività e il raggiungimento degli obiettivi prefissati.

Le comunicazioni si suddividono in:

4.1.10.1 Comunicazioni interne

Le comunicazioni interne avverranno attraverso due canali principali: Telegram e Discord. Telegram, un servizio di messaggistica istantanea, sarà utilizzato per le conversazioni rapide e informali, mentre Discord, una piattaforma che supporta chat e videochiamate, sarà impiegato per le riunioni a distanza e le discussioni più complesse.

Le attività quotidiane e le comunicazioni generali si svolgeranno su Telegram, mentre le questioni critiche saranno affrontate durante gli incontri su Discord.

4.1.10.2 Comunicazioni esterne

Le comunicazioni esterne avverranno tramite scambio di email, utilizzando l'indirizzo `sevenbits.swe.unipd@gmail.com`. Questo canale sarà utilizzato per organizzare incontri con la proponente_G, inviare documenti da visionare o firmare. Inoltre, su Discord è presente una sezione testuale dedicata alle comunicazioni rapide o dubbi con la proponente_G.

4.1.11 Riunioni

Le riunioni sono fondamentali per il progresso del progetto_G, poiché offrono l'opportunità di confrontarsi, prendere decisioni e risolvere eventuali problematiche. Servono anche per monitorare lo stato di avanzamento delle attività e garantire il raggiungimento degli obiettivi prefissati. Al termine di ogni riunione, viene redatto un verbale per documentare gli argomenti trattati e le decisioni prese, garantendo così la tracciabilità e la condivisione delle informazioni. Dopo ogni incontro, il Responsabile redige il verbale, includendo tutte le informazioni rilevanti discusse, lo fa verificare e infine il Verificatore richiede l'approvazione tramite un'ulteriore verifica da parte di un altro membro. Nel caso dei verbali esterni, è prevista anche la firma della proponente_G.

Le riunioni si distinguono in:

4.1.11.1 Riunioni interne

Le riunioni interne sono pianificate settimanalmente ogni mercoledì alle ore 16:00 o, in alternativa, subito dopo le riunioni esterne. In caso di necessità, possono essere richieste riunioni straordinarie durante la settimana, organizzate tramite un sondaggio sul canale Telegram. Tutte le riunioni si terranno sul canale Discord del gruppo. I membri del gruppo si impegnano a partecipare attivamente, cercando di garantire la loro presenza. In caso di impossibilità a partecipare, ogni membro potrà recuperare la sintesi delle discussioni e le decisioni prese leggendo il verbale della riunione.

4.1.11.2 Riunioni esterne

Le riunioni esterne coinvolgono i membri del gruppo, la proponente_G e i committenti. Gli incontri con la proponente_G si terranno sulla piattaforma Google Meet, con l'indirizzo comunicato al gruppo dall'azienda. I membri del gruppo si impegnano a partecipare attivamente, cercando di garantire la loro presenza. In caso di impossibilità a partecipare, ogni membro potrà recuperare la sintesi delle discussioni e le decisioni prese leggendo il verbale della riunione.

4.2 Miglioramento

4.2.1 Descrizione e Scopo

Il miglioramento è un processo_G finalizzato a valutare, misurare, controllare e ottimizzare il ciclo-di-vita_G del software. Il suo scopo principale è garantire che il prodotto soddisfi tutte le aspettative, mantenendo al contempo elevati standard di qualità ed efficienza_G. Adottando un approccio ciclico, è previsto che ogni fase venga periodicamente rivista e perfezionata, permettendo così di identificare e implementare modifiche mirate per un miglioramento continuo del prodotto.

4.2.2 Ciclo di miglioramento continuo

Il miglioramento si compone di quattro fasi fondamentali. La prima fase riguarda l'identificazione dei processi organizzativi da applicare alle diverse attività di progetto_G durante l'intero ciclo-di-vita_G del software. Questi processi devono essere adeguatamente documentati e supportati da un sistema_G di controllo che ne consenta lo sviluppo, il monitoraggio e il miglioramento nel tempo.

La seconda fase consiste nell'implementazione dei miglioramenti identificati, assicurando che le modifiche vengano integrate nei processi esistenti. È essenziale aggiornare la documentazione_G per riflettere i cambiamenti apportati, mentre i dati storici, tecnici e di valutazione devono essere raccolti e analizzati per garantire un progresso continuo.

La terza fase prevede la valutazione dei processi, basata sugli obiettivi predefiniti, le metriche adottate e l'analisi dei dati raccolti. Inoltre, la pianificazione e l'effettuazione di revisioni periodiche garantiscono la loro continua idoneità ed efficacia_G.

Infine, la quarta fase si concentra sulla standardizzazione e il consolidamento dei miglioramenti, assicurando che i processi evolvano costantemente e prevenendo il rischio_G di regressioni. Questo approccio ciclico permette di mantenere un'elevata qualità ed efficienza_G nel tempo.

4.3 Formazione

4.3.1 Descrizione e Scopo

La formazione assicura che ogni membro del gruppo sviluppi le competenze necessarie per affrontare con consapevolezza tutte le sfide che potrebbero emergere durante lo svolgimento del progetto_G.

Il percorso_G di apprendimento si basa su un'analisi dettagliata delle esigenze espresse dalla proponente_G, seguita dall'identificazione delle risorse più adeguate per il raggiungimento degli obiettivi stabiliti.

Attraverso questo processo_G si realizza un progressivo arricchimento delle conoscenze del gruppo, supportato dal consolidamento della comprensione degli strumenti e delle tecnologie adottate.

La formazione risulta un pilastro fondamentale per il successo del progetto_G, offrendo al contempo un'opportunità di crescita professionale per tutti i partecipanti.

4.3.2 Formazione individuale

Durante lo svolgimento del progetto_G, ogni membro del gruppo è tenuto ad approfondire autonomamente le tecnologie necessarie per lo sviluppo del prodotto. Il processo_G di apprendimento è facilitato dalla condivisione di conoscenze tra i membri, grazie al contributo di coloro che possiedono maggiore esperienza e competenza.

Questo approccio consente al gruppo di raggiungere un livello uniforme di conoscenze, adeguato al conseguimento degli obiettivi, e favorisce lo sviluppo complessivo del progetto_G.

5 Standard di progetto

5.1 Standard ISO/IEC 12207 - 1995

Lo standard ISO/IEC 12207 intitolato "Information technology – Software life cycle processes" individua delle linee guida internazionali per la suddivisione dei processi del ciclo-di-vita_G di un prodotto software. Questo standard definisce un framework_G per tali processi per tutto il corso del ciclo-di-vita_G di un prodotto software, dall'ideazione fino al ritiro.

Lo standard suddivide i processi in tre categorie principali che seguono la suddivisione effettuata in questo stesso documento:

- Processi primari;
- Processi di supporto;
- Processi organizzativi.

5.1.1 Processi primari

Trattasi dei processi che coinvolgono direttamente creazione, uso e supporto del software, ovvero di quei processi che descrivono le attività principali del ciclo-di-vita_G del software, dalla concezione dello stesso fino alla sua realizzazione. Appartenenti a questa categoria sono le attività di analisi-dei-requisiti, progettazione, implementazione, realizzazione e esecuzione di test_G e infine manutenzione del software.

5.1.2 Processi di supporto

Questa categoria comprende tutti i processi finalizzati a garantire che i processi primari siano svolti in maniera corretta, coerente ed efficiente. Esempi concreti di questi processi possono essere documentazione_G, gestione della configurazione, gestione della qualità, verifica e validazione.

5.1.3 Processi organizzativi

Si tratta dell'insieme dei processi che definiscono la gestione e definizione delle politiche adottate dal gruppo nell'ambito dello svolgimento del progetto_G.

5.2 Standard ISO/IEC 9126 - 1991

Lo standard ISO/IEC 9126 individua una serie di linee guida e normative, sviluppate dall'ISO (Organizzazione internazionale per la normazione) in collaborazione con l'IEC (Commissione Elettrotecnica Internazionale), finalizzate a descrivere un modello univoco e globalmente riconosciuto di qualità del software. Il modello nasce dalla necessità di fornire alle società di software un approccio organizzato che faciliti e migliori la produzione di software e conseguentemente anche la qualità dello stesso.

5.2.1 Qualità del prodotto software

Lo standard ISO/IEC:9126 definisce 6 caratteristiche centrali che influenzano la qualità di un software:

- Funzionalità;
- Affidabilità;
- Usabilità;
- Efficienza_G;
- Manutenibilità;
- Portabilità.

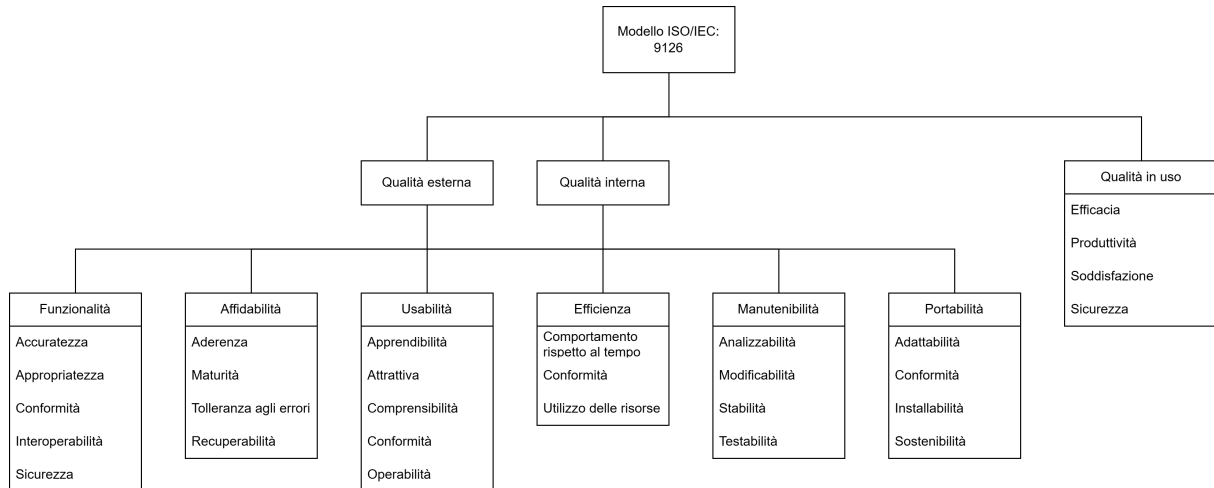


Figure 16: Schema riassuntivo dello Standard ISO/IEC:9126

5.2.1.1 Funzionalità

Valuta quanto il software soddisfi i requisiti funzionali espliciti e impliciti. Include cinque sotto-caratteristiche:

- **Accuratezza:** rappresenta la capacità del software di fornire i risultati concordati;
- **Appropriatezza:** il software deve offrire un appropriato insieme di funzionalità coerenti con il raggiungimento degli obiettivi prefissati all'utente;
- **Conformità:** rappresenta la capacità del prodotto di aderire agli standard e i regolamenti rilevanti nel settore in cui si opera;
- **Interoperabilità:** è la capacità del prodotto software di interagire e cooperare con uno o più sistemi esterni;
- **Sicurezza:** indica la capacità di un prodotto software di proteggere informazioni e dati, garantendo che tutti coloro che interagiscono con tale prodotto (sia persone che altri software) abbiano accesso alle sole informazioni che è loro consentito visualizzare secondo il livello di autorizzazione che possiedono.

5.2.1.2 Affidabilità

Misura la consistenza con cui il software svolge i suoi compiti mantenendo un certo livello di prestazioni e senza andare incontro a fallimento. Include quattro sotto-caratteristiche:

- **Aderenza:** misura la frequenza con la quale il software è "up-and-running";
- **Maturità:** un software maturo è stato testato a fondo, pertanto molti bug_G sono stati individuati e risolti;
- **Tolleranza agli errori:** rappresenta la capacità del software di funzionare mantenendo determinati livelli prestazionali anche in presenza di errori;
- **Recuperabilità:** rappresenta la capacità del software di ripristinare un livello adeguato di prestazioni e di recuperare dati rilevanti a seguito di un malfunzionamento.

5.2.1.3 Usabilità

Misura quanto intuitivo, soddisfacente e facile da usare è un software. Si compone di cinque sotto-caratteristiche:

- **Apprendibilità:** misura quanto facilmente un nuovo utente può imparare a fare uso di un software;
- **Attrattiva:** misura la capacità del software di essere piacevole da usare per l'utente;
- **Comprensibilità:** rappresenta la facilità con cui gli utenti possono capire le funzionalità offerte da un software;

- **Conformità:** indica la capacità del prodotto di aderire a standard e convenzioni relativi all'usabilità;
- **Operabilità:** indica quanto controllo l'utente ha sul software stesso.

5.2.1.4 Efficienza

Misura le performance del sistema_G relativamente al quantitativo di risorse usate in determinate condizioni. L'efficienza tiene conto sia della velocità del sistema_G software che del suo uso delle risorse fisiche disponibili. Questo aspetto si può ulteriormente dividere in tre sotto-caratteristiche:

- **Comportamento rispetto al tempo:** è indicatore di quanto velocemente un sistema_G può eseguire un compito o processare una richiesta;
- **Conformità:** rappresenta la capacità del software di aderire a standard e specifiche richieste in termini di efficienza_G;
- **Utilizzo delle risorse:** misura quanto efficientemente un software utilizza le risorse fisiche dell'hardware sottostante.

5.2.1.5 Manutenibilità

Rappresenta la predisposizione di un prodotto software a essere modificato ai fini di migliorarne le performance, renderlo adeguato a un ambiente in evoluzione o aggiungere o modificare funzionalità. Un software mantenibile consente di rispondere efficacemente alle necessità degli utenti, a bug_G o problemi rilevati o all'evoluzione del panorama tecnologico che circonda il software stesso. Si suddivide in:

- **Analizzabilità:** rappresenta la facilità con cui è possibile diagnosticare problemi del software;
- **Modificabilità:** indica la facilità con cui è possibile effettuare dei cambiamenti;
- **Stabilità:** rappresenta la robustezza del codice del software nell'evitare effetti inaspettati dovuti a modifiche errate;
- **Testabilità:** rappresenta la facilità con cui è possibile testare il software modificato o prodotto.

5.2.1.6 Portabilità

Misura la facilità con cui il software si può adattare a un ambiente hardware o software differente da quello originale. La portabilità di un sistema_G si traduce nella sua flessibilità o adattabilità. Accorpa le seguenti sotto-categorie:

- **Adattabilità:** rappresenta la capacità del sistema_G di adattarsi a diversi ambienti operativi senza dover apportare modifiche;
- **Conformità:** è la capacità del prodotto software di aderire a standard e convenzioni relative alla portabilità;
- **Installabilità:** indica la facilità di installazione del software in uno specifico ambiente;
- **Sostituibilità:** rappresenta la capacità del sistema_G di sostituire altri software che svolgono gli stessi compiti nel medesimo ambiente.

6 Metriche per la qualità

6.1 Metriche di Qualità del Prodotto

L'acronimo principale utilizzato in questa sezione è MPD, da Metriche di qualità del Prodotto.

6.1.1 Funzionalità

- **MPD01 - Requisiti Obbligatori Soddisfatti (ROS):**

- **Descrizione:** metrica che valuta quanti dei requisiti obbligatori definiti in fase di analisi-dei-requisiti sono stati soddisfatti.
- **Formula:**

$$ROS = \frac{ROC}{TRO} \cdot 100$$

Dove:

- * *ROC*: requisiti obbligatori coperti.
- * *TRO*: requisiti obbligatori totali.

- **MPD02 - Requisiti Desiderabili Soddisfatti (RDS):**

- **Descrizione:** metrica che valuta quanti dei requisiti desiderabili definiti in fase di analisi-dei-requisiti sono stati soddisfatti.
- **Formula:**

$$RDS = \frac{RDC}{TRD} \cdot 100$$

Dove:

- * *RDC*: requisiti desiderabili coperti.
- * *TRD*: requisiti desiderabili totali.

- **MPD03 - Requisiti Opzionali Soddisfatti (RPS):**

- **Descrizione:** metrica che valuta quanti dei requisiti opzionali definiti in fase di analisi-dei-requisiti sono stati soddisfatti.
- **Formula:**

$$RPS = \frac{RPC}{TRP} \cdot 100$$

Dove:

- * *RPC*: requisiti opzionali coperti.
- * *TRP*: requisiti opzionali totali.

6.1.2 Affidabilità

- **MPD04 - Line Coverage:**

- **Descrizione:** percentuale di linee di codice coperto da test_G.

- **MPD05 - Branch_G Coverage:**

- **Descrizione:** percentuale di rami decisionali testati.
- **Formula:**

$$BC = \frac{RDT}{NTR} \cdot 100$$

Dove:

- * *RDT*: rami decisionali testati.
- * *NTR*: numero totale rami decisionali.

- **MPD06 - Tempo medio di risposta:**

- **Descrizione:** misura l'efficienza del sistema_G in termini di reattività.

6.1.3 Usabilità

- **MPD07 - Facilità di utilizzo:**
 - **Descrizione:** misura la facilità con cui gli utenti possono utilizzare il sistema_G.
- **MPD08 - Tempo medio di apprendimento:**
 - **Descrizione:** tempo medio necessario agli utenti per apprendere l'uso del sistema_G.

6.1.4 Manutenibilità

- **MPD09 - Linee di codice per metodo:**
 - **Descrizione:** massimo numero di linee di codice consentito per singolo metodo.
- **MPD10 - Attributi per classe:**
 - **Descrizione:** massimo numero di attributi consentito per singola classe.
- **MPD11 - Structure Fan IN:**
 - **Descrizione:** rappresenta il numero di moduli, funzioni o classi che chiamano o utilizzano una specifica funzione o modulo, indicando quanto è riutilizzato.
- **MPD12 - Structure Fan OUT:**
 - **Descrizione:** misura il numero di moduli, funzioni o classi che una specifica funzione o modulo utilizza o chiama, evidenziando le sue dipendenze.

6.1.5 Portabilità

- **MPD13 - Versioni browser supportati:**
 - **Descrizione:** percentuale delle versioni di browser supportate rispetto al totale delle versioni disponibili.

6.2 Metriche di qualità di Processo

L'acronimo principale utilizzato in questa sezione è MPC, da Metriche di qualità di Processo_G.

6.2.1 Processi Primari

6.2.1.1 Fornitura

- **MPC01 - Estimated at Completion (EAC):**
 - **Descrizione:** stima il costo totale del progetto_G basandosi sul rendimento attuale.
 - **Formula:**

$$EAC = AC + ETC$$
- **MPC02 - Estimate to Complete (ETC):**
 - **Descrizione:** stima dei costi rimanenti per completare il progetto_G.
 - **Formula:**

$$ETC = BAC - EV$$
- **MPC03 - Actual Cost (AC):**
 - **Descrizione:** indica il costo effettivamente sostenuto per completare il lavoro fino a un certo momento.
- **MPC04 - Earned Value (EV):**

- **Descrizione:** misura il valore del lavoro effettivamente realizzato fino a un certo momento del progetto_G. Tale metrica è data dalla somma dei valori guadagnati fino a quel momento per ciascuna delle attività di cui si compone il progetto_G; e ognuno di questi è dato dal budget di tale attività moltiplicato per la percentuale di completamento della stessa attività.

- **MPC05 - Planned Value (PV):**

- **Descrizione:** rappresenta il valore pianificato del lavoro da completare in un determinato momento.

- **MPC06 - Schedule variance (SV):**

- **Descrizione:** indica se si è in ritardo o in anticipo rispetto al valore pianificato. Un valore negativo indica che si è in ritardo.
- **Formula:**

$$SV = EV - PV$$

- **MPC07 - Cost variance (CV):**

- **Descrizione:** indica se il costo attuale sta superando o meno il budget. Un valore negativo indica che si ha superato il budget.
- **Formula:**

$$CV = EV - AC$$

- **MPC08 - Cost Performance Index (CPI):**

- **Descrizione:** indica quanti obiettivi sono stati raggiunti rispetto al costo sostenuto in determinato periodo di tempo. Un valore negativo indica che il progetto_G è in ritardo rispetto al budget in quanto i costi sostenuti sono superiori al valore del lavoro completato.
- **Formula:**

$$CPI = \frac{EV}{AC}$$

6.2.1.2 Sviluppo

- **MPC09 - Requisiti Obbligatorî Soddisfatti (ROS):**

- **Descrizione:** metrica che valuta quanti dei requisiti obbligatori definiti in fase di analisi-dei requisiti sono stati soddisfatti.
- **Formula:**

$$ROS = \frac{ROC}{ROT} \cdot 100$$

Dove:

- * *ROC*: requisiti obbligatori coperti.
- * *ROT*: requisiti obbligatori totali.

- **MPC10 - Requirements Stability Index(RSI):**

- **Descrizione:** misura la stabilità dei requisiti nel ciclo-di-vita_G di un progetto_G. Un RSI_G basso indica che sono stati effettuati molti cambiamenti ai requisiti.
- **Formula:**

$$RSI = 100 - \frac{RA + RC + RR}{RT} \cdot 100$$

Dove:

- * *RA*: requisiti aggiunti dopo l'ultima misurazione.
- * *RC*: requisiti modificati dopo l'ultima misurazione.
- * *RR*: requisiti rimossi dopo l'ultima misurazione.
- * *RT*: requisiti iniziali totali dopo l'ultima misurazione.

6.2.2 Processi di Supporto

6.2.2.1 Documentazione

- **MPC11 - Indice Gulpease:**

- **Descrizione:** indice di leggibilità di un testo in lingua italiana. Il risultato è in percentuale e cresce al crescere della facilità di comprensione del testo:
 - * < 80: difficile da leggere per persone con la licenza elementare.
 - * < 60: difficile da leggere per persone con la licenza media.
 - * < 40: difficile da leggere per persone con la licenza superiore.

- **Formula:**

$$IG = 89 + \frac{300 \cdot N_f - 10 \cdot N_l}{N_p}$$

Dove:

- * N_f : numero totale di frasi nel testo.
- * N_l : numero totale di lettere nel testo.
- * N_p : numero totale di parole nel testo.

- **MPC12 - Correttezza ortografica:**

- **Descrizione:** metrica che misura il numero di errori ortografici presenti all'interno di un documento.

6.2.2.2 Verifica

- **MPC13 - Line Coverage:**

- **Descrizione:** percentuale di linee di codice coperto da test_G.

- **MPC14 - Passed test_G cases percentage:**

- **Descrizione:** percentuale di test_G superati.

6.2.2.3 Gestione della Qualità

- **MPC15 - Metriche di qualità soddisfatte:**

- **Descrizione:** percentuale di soddisfacimento delle metriche di qualità stabilite.

6.2.3 Processi Organizzativi

6.2.3.1 Gestione dei Processi

- **MPC16 - Rischi non previsti:**

- **Descrizione:** numero di rischi non inclusi nelle stime incontrati durante lo svolgimento del progetto_G.

- **MPC17 - Efficienza_G Temporale (ET):**

- **Descrizione:** rapporto tra ore totali (O_o) e ore produttive (O_p) svolte a partire dall'ultima misurazione.
- **Formula:**

$$ET = \frac{O_o}{O_p}$$