Norme di Progetto

sevenbits.swe.unipd@gmail.com

Registro modifiche

Versione	Data	Autore	Verificatore	Descrizione
0.4.2	2024-12-11	Leonardo Trolese	Manuel Gusella	Scrittura sezione 6 relativa allo Standard ISO/IEC:25010 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 Organizzativa" 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 stabilito
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

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Il presente documento è stato realizzato ai fini di definire e raccogliere le best practices e il way of working a cui ogni componente del gruppo Seven Bits dovrà aderire per l'intera realizzazione del progetto, al fine di garantire l'adozione di un metodo di lavoro completamente omogeneo.
La formulazione delle norme di progetto 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 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 "Near You" si concentra sulla creazione di una dashboard composta principalmente da una mappa, sulla quale verranno visualizzate in tempo reale le posizioni degli utenti. Mediante un popup 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, nel quale sono presenti tutti i termini tecnici adottati dal gruppo per l'intera durata della realizzazione del progetto. Tali termini saranno contrassegnati con una $_G$ a pedice. Nel caso di termini composti, essi saranno uniti da un trattino (es. Analisi-dei-requisiti$_G$)

1.4 Riferimenti

1.4.1 Riferimenti normativi

- Capitolato di progetto C4 - Near You: https://www.math.unipd.it/ tullio/IS-1/2024/Progetto/C4.pdf
- Standard ISO/IEC 12207:1995: https://www.math.unipd.it/ tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf

1.4.2 Riferimenti tecnologici

- Documentazione Git: https://git-scm.com/docs
- Documentazione GitHub: https://docs.github.com/en
- Documentazione LaTeX: https://www.latex-project.org/help/documentation/
- Documentazione Python: https://www.python.org/doc/

2 Processi Primari

2.1 Fornitura

2.1.1 Descrizione e Scopo

Come stabilito dallo standard ISO/IEC 12207:1995, il processo di fornitura definisce un insieme di linee guida necessario per una buona comunicazione tra fornitore e proponente. Il processo 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.

2.1.2 Attività

Il processo 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, con successiva verifica della fattibilità tecnologica di quest'ultime;
2. **Preparazione dell'offerta**: Composta dall'elaborazione della proposta in grado di soddisfare le richieste della proponente, che dettagli i requisiti, i tempi, i costi e le condizioni contrattuali;
3. **Contrattazione**: Composta dalla collaborazione tra fornitore e proponente per finalizzare i punti cardine della proposta;
4. **Pianificazione**: Composta dalla pianificazione delle attività necessarie per soddisfare i requisiti della proponente, 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, accompagnato dalla documentazione finale.

2.1.3 Rapporti con l'azienda proponente

L'azienda proponente 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.
Durante la prima riunione organizzativa con l'azienda, è stata definita l'organizzazione dei periodi di sprint$_G$, stabilendo una durata di due settimane per ciascun ciclo. Al termine di ogni sprint$_G$ avviene un incontro SAL$_G$ (Stato di Avanzamento Lavori), dove verranno analizzati i risultati del lavoro svolto e si procederà con una sprint-review$_G$. Inoltre, tra un SAL$_G$ e l'altro, secondo le necessità di proponente 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 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:

2.1.4.1 Analisi dei Requisiti

É 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

Il 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. 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

Il 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 del progetto, per questo motivo sarà aggiornato nel tempo per riflettere eventuali modifiche e i risultati delle verifiche effettuate. Include le sezioni sulla qualità di processo, sulla qualità di prodotto, le modalità di testing e il cruscotto di valutazione delle qualità.

2.1.4.4 Glossario

Il 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.

2.1.4.5 Lettera di Presentazione

La Lettera di Presentazione è un documento che accompagna la consegna del prodotto software e della relativa documentazione durante le fasi di revisione di progetto. Il contenuto di questo documento comprende un link alla pagina web che contiene tutta la documentazione fin'ora prodotta ed un preventivo aggiornato rispetto a quello presentato alla revisione precedente.

2.1.5 Strumenti

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

- **Canva**: Piattaforma che permette la creazione di presentazioni multimediali, utilizzata per la realizzazione dei diari di bordo;
- **Draw.io**: Software utilizzato per creare diagrammi e grafici di vario tipo, in particolare è stato impiegato per realizzare diagrammi UML, come quelli dei casi d'uso;
- **Google Meet e Discord**: Servizi che permettono di effettuare videochiamate, utilizzati dal team per le discussioni sincrone e asincrone con la proponente;
- **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 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. Questo processo 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.

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 di sviluppo software. Lo scopo principale di questa fase è definire con chiarezza le funzionalità e le caratteristiche che il sistema dovrà offrire, in base alle necessità degli utenti. Per raggiungere tale obiettivo, è essenziale stabilire una comunicazione efficace con la proponente, assicurandosi che tutte le esigenze siano documentate e validate. Questo processo 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.pdf*, redatto dagli Analisti, che contiene una descrizione dettagliata degli obiettivi del prodotto, delle funzionalità previste, delle caratteristiche degli utenti e delle tecnologie coinvolte. Include inoltre una sezione dedicata ai casi d'uso, che descrivono le interazioni tra gli attori esterni e il sistema.
Infine, l'Analisi-dei-Requisiti contribuisce a migliorare la comunicazione tra tutti gli stakeholder, agevola la pianificazione del progetto in termini di tempistiche e costi e fornisce un riferimento chiaro per le attività di verifica e test.

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 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 esclusivamente dal punto di vista esterno, evidenziando come esso interagisce con gli attori e soddisfa le loro esigenze.
Ogni caso d'uso ha le seguenti caratteristiche:

- **Identificativo**:

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

  Dove:

  - **Numero caso d'uso**: ID relativo al caso d'uso 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 per raggiungere uno scopo, ad esempio un utente specifico o un altro sistema esterno;
- **Attore secondario**: Eventuale entità esterna invocata dal sistema per fornire un supporto, in modo da soddisfare il bisogno dell'attore principale;
- **Descrizione**: Breve descrizione della funzionalità del caso d'uso;
- **Scenario principale**: Sequenza di eventi che si susseguono da quando un attore inizia ad interagire con il sistema;
- **Precondizioni**: Descrizione dello stato del sistema necessario per attivare il caso d'uso;
- **Postcondizioni**: Descrizione dello stato del sistema al completamento del caso d'uso;
- **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 dal punto di vista dell'utente. Mostrano come gli attori esterni (utenti, dispositivi, altri sistemi) interagiscono con il sistema, senza entrare nei dettagli tecnici. Il loro obiettivo principale è descrivere le interazioni tra attori e sistema, facilitando la comprensione dei requisiti funzionali e la comunicazione tra gli stakeholder.
Ogni caso d'uso descrive una sequenza di azioni che un attore compie per raggiungere un obiettivo specifico. Questi casi sono interconnessi e rappresentano i flussi di lavoro principali, evidenziando come l'utente interagisce con il sistema. I diagrammi non trattano i dettagli implementativi, ma si concentrano sulle funzionalità, trattando il sistema 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 che interagiscono con esso per utilizzare le sue funzionalità. Possono essere utenti umani, altri sistemi software, dispositivi, macchine o organizzazioni. Un caso d'uso definisce una specifica funzionalità che il sistema 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, e ciascuno è identificato da un'etichetta con il suo nome.

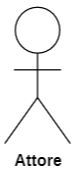

Figure 1: Rappresentazione di un attore.

- **Casi d'uso**
  Ogni caso d'uso identifica una specifica azione o funzionalità offerta dal sistema, 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.

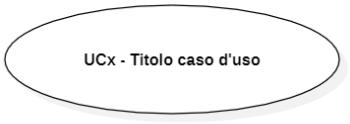

Figure 2: Rappresentazione di un caso d'uso.

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

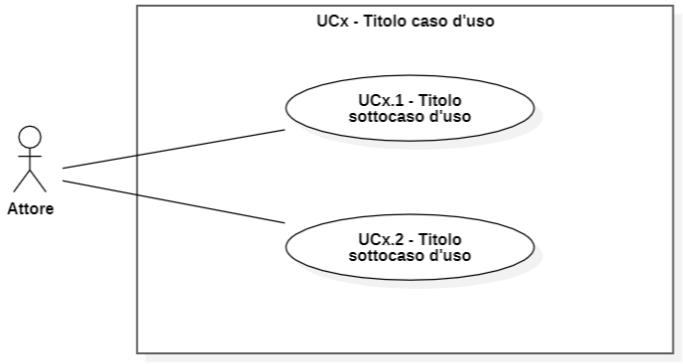

Figure 3: Rappresentazione di un sottocaso d'uso.

- **Sistema**
  Il sistema 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.

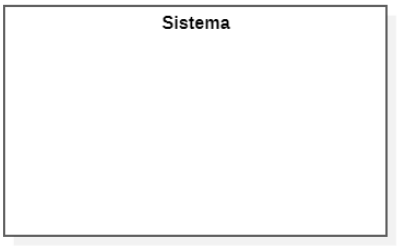

Figure 4: Rappresentazione di un sistema.

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

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

- **Generalizzazione (attori)**
  La generalizzazione tra attori rappresenta una relazione gerarchica in cui un attore specializzato (figlio) eredita comportamenti e caratteristiche da un attore base (genitore). Questo meccanismo aiuta a organizzare gerarchicamente gli attori nei diagrammi dei casi d'uso, definendo i casi d'uso che un attore 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 (includente) comprende l'esecuzione di un altro caso d'uso (incluso). Quando un attore interagisce con il caso d'uso includente, il caso d'uso 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 incluso al caso d'uso includente.

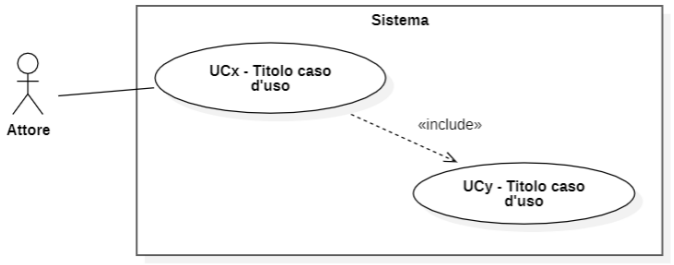

Figure 7: Rappresentazione di una inclusione tra UC.

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

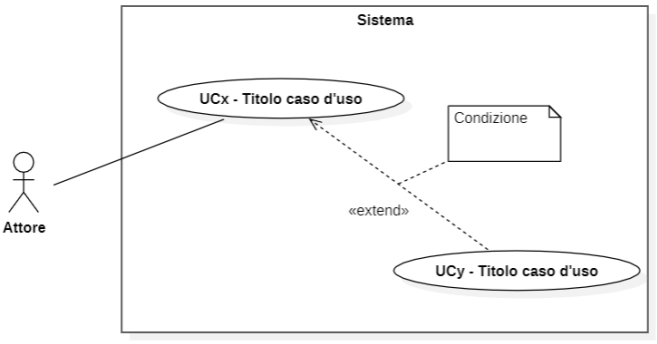

Figure 8: Rappresentazione di una estensione tra UC.

- **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 più specifico eredita il comportamento da un caso d'uso 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 specifico al caso d'uso generico.

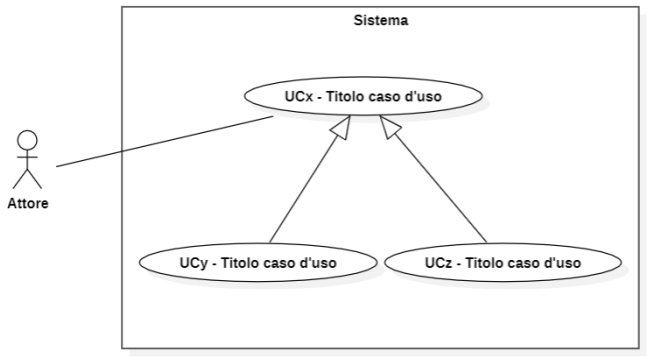

Figure 9: Rappresentazione di una generalizzazione tra UC.

2.2.2.4 Requisiti

I requisiti di un prodotto software sono specifiche documentate che delineano le funzionalità che il sistema 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. Ogni requisito deve essere definito con precisione, e riflettere pienamente le attese del cliente o del proponente.
Ad ogni requisito è assegnato un grado di importanza:

- **Obbligatorio**: Essenziale per soddisfare le esigenze del proponente.
- **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 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 deve rispettare;
- **Requisiti di vincolo (V)**: Rappresentano restrizioni o condizioni imposte al progetto, 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 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 aggiunto;
- **Descrizione**: Una descrizione dettagliata che spiega la caratteristica richiesta al sistema;
- **Fonte**: L'origine del requisito;
- **Casi d'uso associati**: Un elenco di casi d'uso che forniscono il contesto operativo per il requisito;

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 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

2.2.3 Progettazione

2.2.3.1 Descrizione e Scopo

La progettazione, o design, è un'altra fase cruciale del processo di sviluppo software che ha l'obiettivo di tradurre i requisiti individuati durante la fase di analisi in una struttura architetturale 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 (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 le componenti principali che costituiranno l'MVP (Minimum Viable Product).

2.2.3.2 Diagrammi delle classi

Sono una tipologia di diagramma UML utile a rappresentare la struttura statica di un sistema software orientato agli oggetti. Questi diagrammi visualizzano le classi del sistema 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;

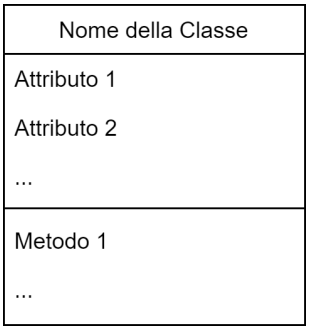

Figure 10: Rappresentazione di una classe.

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;

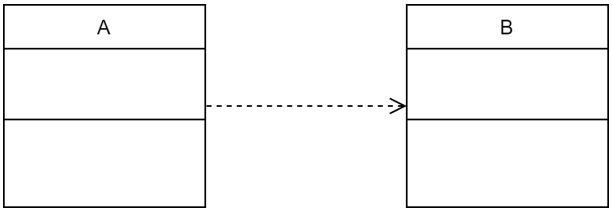

Figure 11: Rappresentazione della relazione di dipendenza.

- **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;

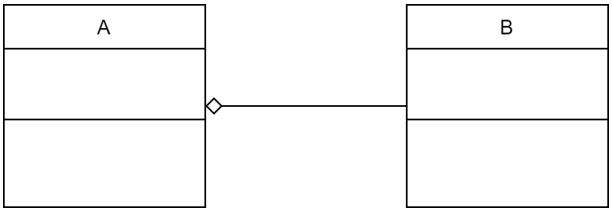

Figure 12: Rappresentazione della relazione di aggregazione.

- **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;

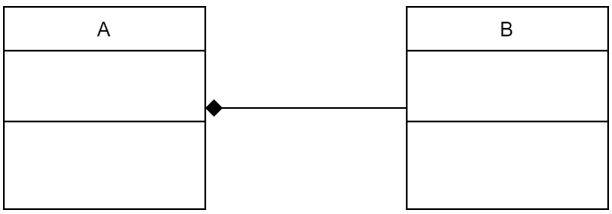

Figure 13: Rappresentazione della relazione di composizione.

- **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;

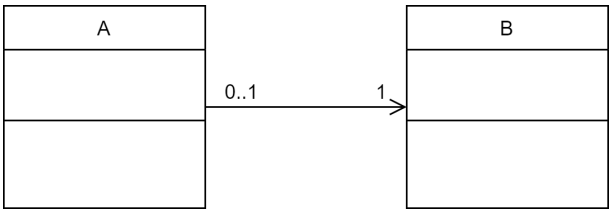

Figure 14: Rappresentazione della relazione di associazione.

- **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);

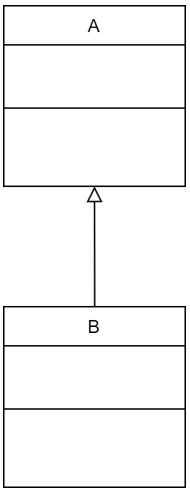

Figure 15: Rappresentazione della relazione di generalizzazione.

2.2.3.3 Metriche

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 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 stabilite. Questo metodo di lavoro garantisce che il codice segua accuratamente le specifiche, promuovendo qualità, manutenibilità e coerenza nell'intero progetto.

L'obiettivo principale della codifica è quindi creare un prodotto software che soddisfi le esigenze del proponente 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 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 di unità efficaci;

2.2.4.3 Strumenti utilizzati

Il team di sviluppo utilizza strumenti che supportano la scrittura di codice leggibile, mantenibile e conforme agli standard. Tra questi:

- **Visual Studio Code**: L'IDE principale scelto per la codifica, grazie alla sua versatilità e facilità d'uso;

2.2.4.4 Integrazione

Durante la fase di integrazione le diverse componenti, moduli o servizi del progetto vengono combinati e testati per creare un sistema coeso e funzionante.
L'obiettivo principale dell'integrazione è garantire che tutte le parti del sistema 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 entra nella fase di sviluppo del Minimum Viable Product (MVP).
Questa fase è supportata da test di integrazione che verificano che i moduli lavorino correttamente insieme. L'integrazione continua, supportata da strumenti automatizzati come CI/CD (Continuous Integration/Continuous Delivery), permette di garantire che ogni nuova modifica non comprometta l'affidabilità del sistema, mantenendo un flusso di lavoro agile e risolvendo eventuali problemi o incompatibilità prima che si presentino nel sistema finale.

2.2.4.5 Verifica

La verifica è una fase cruciale del ciclo di vita del software, durante la quale si controlla che il codice sviluppato rispetti le specifiche e soddisfi i requisiti stabiliti. Questo processo 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 di verifica comprende sia un'analisi statica del codice, per rilevare eventuali errori o problemi di formattazione, sia l'esecuzione di test mirati per validare le funzionalità implementate.

2.2.4.6 Metriche

3 Processi di supporto

3.1 Documentazione

3.1.1 Descrizione e Scopo

La documentazione 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 del progetto, 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 sono:

- *Analisi_dei_requisiti.pdf*
- *Glossario.pdf*
- *Norme_di_progetto.pdf*
- *Piano_di_progetto.pdf*
- *Piano_di_qualifica.pdf*
- *Verbali esterni*
- *Verbali interni*

3.1.3 Ciclo di vita documenti e Versionamento

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 di versionamento indicato di seguito.

3.1.3.1 Versionamento dei documenti

Il sistema adottato dal team per il versionamento dei documenti è il sistema di versionamento semantico: **x.y.z**, in cui ogni numero (x, y, z) 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 aiuta il team a comprendere velocemente l'impatto di un aggiornamento fatto ad un documento prodotto.

3.1.3.2 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 (la versione iniziale del documento corrisponde a *0.1.0*);
2. Compilazione dei campi della sezione *Registro modifiche*;
3. Redazione del documento indicando la durata dell'incontro, i partecipanti (interni ed esterni) e la piattaforma utilizzata. Seguono la sintesi di quanto fatto e una descrizione di ciascuna delle considerazioni fatte e successive decisioni prese;

4. Nella sezione *Decisioni prese* si compila una tabella in cui ciascuna azione da intraprendere viene associata ad una issue corrispondente;
5. Creazione di una pull-request (PR) dal branch Verbali al branch Main;
6. Verifica del verbale prodotto da parte del verificatore indicato nel *Registro modifiche* del documento stesso;
7. Se ci sono correzioni o ulteriori modifiche da fare, queste devono essere indicate a loro volta nella sezione *Registro modifiche* per poi essere verificate;
8. 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 perché venga firmato;
9. Quando il documento è completo l'ultimo verificatore chiude la pull-request ed esegue il merge nel branch Main;

3.1.3.3 Workflow altri documenti

Gli altri documenti seguono il seguente workflow:

1. Creazione di un branch 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 una draft pull request dal branch corrispondente a tale documento al branch Main;
4. Compilazione dei campi della sezione *Registro modifiche*;
5. Redazione del documento o di una sua sezione;
6. Verifica della documentazione 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 fare, queste devono essere indicate a loro volta nella sezione *Registro modifiche* per poi essere verificate a loro volta;
8. Si ripetono le operazioni indicate dal punto *3* al punto *6* fino a quando il documento non è stato completato;
9. Quando il documento è completo l'ultimo verificatore chiude la draft pull request ed esegue il merge nel branch Main, dopodiché si procede ad eliminare il branch$_G$ dedicato;

3.1.4 Template in LaTeX

Per la stesura dei documenti, viene utilizzato un template in formato LaTeX. 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

3.1.5 Nomenclatura

La nomenclatura dei documenti prevede l'unione del nome del file, utilizzando degli underscore (_), ad esempio Piano_di_Qualifica.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".

3.1.6 Struttura documenti

3.1.6.1 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;
- **Contatti**: Email del team;
- **Logo Università**: Situato in basso a destra;

3.1.6.2 Intestazione

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

3.1.6.3 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 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;

3.1.6.4 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.

3.1.6.5 Verbali

I verbali 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.
I verbali hanno lo scopo di tenere traccia di chi ha partecipato agli incontri ed in particolare quali decisioni sono state prese.
Sono composti da 2 macrosezioni:

- **Durata e Partecipanti**: Viene indicata la data di inizio e fine incontro e il luogo in cui si è svolto. A seguire, i nomi dei partecipanti del gruppo. Se il verbale è esterno si indicano anche i partecipanti dell'azienda SyncLab;
- **Sintesi e Decisioni Prese**: Riassunto degli argomenti trattati ed elenco delle decisioni prese durante il meeting, collegate alle issue corrispondenti tramite una tabella;

Per i verbali esterni è presente una sezione per la convalida del documento mediante una firma.

3.1.7 Convenzioni stilistiche

3.1.7.1 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 file e per i titoli delle sezioni.
- **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 $_G$.

3.1.7.2 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;

3.1.8 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 e repository per la documentazione. Permette la gestione del codice sorgente, il controllo delle modifiche tramite funzionalità come le issue e le pull request (PR), facilitando la collaborazione all'interno di un team di sviluppo;
- **LaTeX**: 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;

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Decisioni

4.1.2 Pianificazione

4.1.3 Ruoli di progetto

Durante l'intero sviluppo del progetto, 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, 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:

4.1.3.1 Responsabile

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

- Comunicare con la proponente attraverso i canali concordati;
- Stesura e avanzamento del documento "Piano di progetto";
- Redazione dei verbali interni ed esterni;
- Creazione e assegnazione delle issue ai membri del team;
- Creazione dei diari di bordo;

4.1.3.2 Amministratore

Figura responsabile della creazione, manutenzione e ottimizzazione degli strumenti, delle risorse e dei processi necessari per il corretto svolgimento del progetto. 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 dei documenti;
- Stesura e avanzamento del documento Norme di Progetto;
- Stesura e avanzamento del documento Piano di Qualifica;

4.1.3.3 Analista

Figura responsabile di raccogliere, analizzare e documentare i requisiti del progetto, traducendoli in casi d'uso. Collabora con la proponente 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;
- Studiare i requisiti definendo la loro complessità;

4.1.3.4 Progettista

Figura specializzata nella progettazione architetturale 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 e supervisiona lo sviluppo, senza occuparsi direttamente della manutenzione.
I suoi compiti sono:

- Determinare le tecnologie, i linguaggi e i framework da utilizzare;
- Progettare l'architettura del sistema, definendo componenti, moduli e interazioni;
- Gestire eventuali rischi che possono sorgere durante lo sviluppo;
- Supervisionare lo sviluppo per garantire la conformità all'architettura;

4.1.3.5 Programmatore

Figura incaricata di trasformare l'architettura proposta dai Progettisti in un sistema 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.
I suoi compiti sono:

- Tradurre le specifiche tecniche in codice funzionante;
- Creazione di test per la verifica e validazione del software;
- Scrittura di codice chiaro, leggibile e manutenibile;
- Risoluzione di eventuali bug;

4.1.3.6 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 stabilite per il progetto.
I suoi compiti sono:

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

4.1.4 Cambio dei ruoli

Per garantire che ogni membro ricopra tutti i ruoli essenziali al completamento del progetto, il gruppo si impegna a ruotarli all'inizio di ogni nuovo sprint, della durata di due settimane.

4.1.5 Tracciamento

4.1.6 Gestione strumenti coordinamento

4.1.7 Gestione dei rischi

4.1.8 Procedure comunicative

5 Standard ISO/IEC:25010 "System & software quality models"

Lo standard ISO/IEC:25010 appartiene alla famiglia di standard SQuaRE, ed è stato sviluppato e pubblicato nel 2011 congiuntamente dalla ISO e dalla IEC. Trattasi di un modello internazionale che definisce in maniera chiara un quadro di riferimento per la valutazione e la misurazione della qualità di un prodotto software. Lo scopo è di garantire che il prodotto realizzato soddisfi i requisiti di utenti, di proponenti e di committenti garantendo al contempo sicurezza e funzionalità. Lo standard non entra mai nel merito del "cosa" fa il software, ma definisce il "come" opera nel contesto della qualità.

5.1 Qualità del prodotto software

Lo standard ISO/IEC:25010 definisce 8 caratteristiche centrali che influenzano la qualità di un software:

- Functional suitability
- Reliability
- Usability
- Performance Efficiency
- Maintainability
- Portability
- Compatibility
- Security

5.1.1 Functional suitability

Misura quanto il software soddisfa i requisiti funzionali espliciti e impliciti. Include tre sotto-caratteristiche:

- Functional Completeness - il software deve mettere a disposizione tutte le funzionalità necessarie;
- Functional Correctness - il software deve eseguire le sue funzionalità correttamente e nel modo previsto;
- Functional Appropriateness - il software deve offrire funzionalità appropriate agli obiettivi;

5.1.2 Reliability

Misura la consistenza con cui il software svolge i suoi compiti senza andare incontro a fallimento. Include quattro sotto-caratteristiche:

- Maturity - un software maturo è stato testato a fondo, pertanto molti bug sono stati individuati e risolti;
- Availability - misura la frequenza con la quale il software è "up-and-running";
- Fault Tolerance - rappresenta la capacità del software di funzionare anche in presenza di errori;
- Recoverability - rappresenta la capacita del software di recuperare da un failure riprendendo le sue funzioni;

5.1.3 Usability

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

- Understandability - rappresenta la facilità con cui gli utenti possono capire le funzionalità offerte da un software;
- Learnability - misura quanto facilmente un nuovo utente può imparare a fare uso di un software;
- Operability - indica quanto controllo l'utente ha sul software stesso;
- Attractiveness - misura l'attrattiva estetica che l'interfaccia del software esercita sull'utente;
- Accessibility - indica la facilità di uso del sistema da parte di utenti con disabilità;

5.1.4 Performance Efficiency

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

- Time behavior - è indicatore di quanto velocemente un sistema può eseguire un compito o processare una richiesta;
- Resource utilization - misura quanto efficientemente un software utilizza le risorse fisiche dell'hardware sottostante;
- Capacity - misura la capacità del sistema di gestire determinati carichi di lavoro;

5.1.5 Maintainability

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 o problemi rilevati o all'evoluzione del panorama tecnologico che circonda il software stesso. Si suddivide in:

- Modularity - indica il grado di suddivisione in componenti fra loro indipendenti del sistema;
- Reusability - indica la capacità di riutilizzare componenti software;
- Analyzability - rappresenta la facilità con cui è possibile diagnosticare problemi del software;
- Modifiability - indica la facilità con cui è possibile effettuare dei cambiamenti;
- Testability - rappresenta la facilità con cui è possibile testare il software modificato o prodotto;

5.1.6 Portability

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 si traduce nella sua flessibilità o adattabilità. Accorpa le seguenti sotto-categorie:

- Adaptability - rappresenta la capacità del sistema di adattarsi a diversi ambienti;
- Installability - indica la facilità di installazione del software;
- Replaceability - rappresenta la capacità del sistema di sostituire altri software;

5.1.7 Compatibility

Misura la capacità del software di funzionare in un ambiente condiviso o in coordinamento con altri sistemi. Include due sotto-categorie:

- Co-Existence - è indicatore della capacità del software di coesistere senza conflitti con altri sistemi;
- Interoperability - misura la capacità del software di scambiare informazioni con altri sistemi;

5.1.8 Security

Indica la capacità di un prodotto software di proteggere informazioni o dati, così che tutti coloro che interagiscono con tale prodotto (sia persone che altri software) di avere accesso alle informazioni che è loro consentito visualizzare secondo il livello di autorizzazione che possiedono. Include:

- Confidentiality - rappresenta la protezione dei dati da accessi non autorizzati;
- Integrity - rappresenta la prevenzione della modifica non autorizzata di dati;
- Authenticity - indica la verifica dell'identità degli utenti;
- Accountability - rappresenta il tracciamento delle azioni degli utenti;
- Non-repudiation - rappresenta la capacità di garantire che un'azione non possa essere negata;

6 Metriche per la qualità