Norme di Progetto

2024-01-08 — v0.2.1

overture.unipd@gmail.com

Destinatari	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo *Overture*
Responsabile	Alex Vedovato
Redattori	Eleonora Amadori Michele Bettin Riccardo Bonavigo Francesco Costantino Bulychov Riccardo Fabbian Francesco Furno Alex Vedovato
Verificatori	Eleonora Amadori Francesco Costantino Bulychov Riccardo Fabbian Francesco Furno Alex Vedovato

Registro delle modifiche

Versione	Data	Autori	Verificatori	Dettaglio
0.2.1	2024-01-08	Riccardo Bonavigo	Francesco Costantino Bulychov	Descrizione della repo `jmap` e chiarimenti su branch e PR
0.2.0	2023-12-19	Michele Bettin	Francesco Furno	Aggiornate le sezioni 'Metriche di qualitá del processo' e 'Metriche di qualitá del prodotto'
0.1.2	2023-12-19	Francesco Costantino Bulychov	Francesco Furno	Migliorate le sezioni 'Verifica' e 'Validazione'
0.1.1	2023-12-17	Francesco Costantino Bulychov	Riccardo Fabbian	Terminata la stesura della sezione 'Sviluppo'
0.1.0	2023-11-30	Francesco Costantino Bulychov	Alex Vedovato	Aggiunta delle sezioni 'Metriche di qualitá del processo' e 'Metriche di qualitá del prodotto'
0.0.11	2023-11-26	Francesco Costantino Bulychov	Alex Vedovato	Aggiunta della sezione 'Standard ISO/IEC 9126'
0.0.10	2023-11-22	Eleonora Amadori	Alex Vedovato	Aggiunta della sezione 'Gestione della qualità'
0.0.9	2023-11-21	Francesco Furno	Alex Vedovato	Aggiunta della sezione 'Gestione della configurazione'
0.0.8	2023-11-20	Riccardo Fabbian	Alex Vedovato	Aggiunta prima parte della sezione 'Sviluppo'
0.0.7	2023-11-20	Francesco Furno	Alex Vedovato	Aggiunta della sezione 'Fornitura'
0.0.6	2023-11-18	Alex Vedovato	Eleonora Amadori	Aggiunta della sezione 'Formazione'
0.0.5	2023-11-17	Alex Vedovato	Eleonora Amadori	Aggiunta della sezione 'Miglioramento'
0.0.4	2023-11-16	Riccardo Fabbian, Francesco Furno	Eleonora Amadori	Aggiunta della sezione 'Gestione dei processi'
0.0.3	2023-11-13	Francesco Furno	Eleonora Amadori	Aggiunta della sezione 'Documentazione'
0.0.2	2023-11-11	Riccardo Fabbian	Eleonora Amadori	Aggiunta delle sezioni 'Verifica' e 'Validazione'
0.0.1	2023-11-11	Alex Vedovato	Eleonora Amadori	Struttura iniziale del documento ed introduzione

Indice

1) Introduzione 6
1.1) Scopo del documento 6
1.2) Scopo del progetto 6
1.3) Glossario 6
1.4) Riferimenti 6
1.4.1) Riferimenti normativi 6
1.4.2) Riferimenti informativi 6
2) Processi primari 7
2.1) Fornitura 7
2.1.1) Scopo 7
2.1.2) Descrizione 7
2.1.3) Aspettative 7
2.1.4) Comunicazioni con il proponente 7
2.1.5) Documentazione fornita 7
2.1.5.1) `Valutazione dei Capitolati` 8
2.1.5.2) `Analisi dei Requisiti` 8
2.1.5.3) `Piano di Progetto` 8
2.1.5.4) `Piano di Qualifica` 9
2.1.5.5) `Glossario` 9
2.1.5.6) `Lettera di Presentazione` 9
2.1.6) Strumenti 9
2.1.6.1) Google Calendar 9
2.1.6.2) Typst 9
2.1.6.3) Carbonio - Zextras Chats 9
2.1.6.4) Discord 9
2.2) Sviluppo 9
2.2.1) Scopo 9
2.2.2) Descrizione 10
2.2.2.1) Implementazione del processo 10
2.2.2.2) Analisi dei requisiti di sistema 10
2.2.2.3) Progettazione architetturale del sistema 10
2.2.2.4) Analisi dei requisiti software 11
2.2.2.5) Progettazione architetturale del software 11
2.2.2.6) Progettazione dettagliata del software 12
2.2.2.7) Codifica e testing del software 12
2.2.2.8) Integrazione del software 12
2.2.2.9) Test di qualifica del software 13
2.2.2.10) Integrazione di sistema 13
2.2.2.11) Test di qualifica del sistema 14
2.2.2.12) Installazione del software 14
2.2.2.13) Supporto all'accettazione del software 14
3) Processi di supporto 14
3.1) Documentazione 14
3.1.1) Scopo 14
3.1.2) Descrizione 15
3.1.3) Aspettative 15
3.1.4) Ciclo di vita 15

3.1.5) Sistema di composizione tipografica 15
3.1.6) Struttura dei documenti 15
3.1.6.1) Intestazione 15
3.1.6.2) Registro delle modifiche 16
3.1.6.3) Indice 16
3.1.6.4) Corpo del documento 16
3.1.6.5) Corpo del verbale 16
3.1.6.6) Documenti del progetto 16
3.1.7) Regole stilistiche 17
3.1.7.1) Nomi assegnati ai file 17
3.1.7.2) Stile del testo 17
3.1.8) Elenchi puntati 18
3.1.8.1) Formato delle date 18
3.1.9) Strumenti 18
3.2) Verifica 18
3.2.1) Scopo ed aspettative 18
3.2.2) Descrizione 18
3.2.3) Analisi statica 18
3.2.4) Analisi dinamica 19
3.2.4.1) Test di unità 19
3.2.4.2) Test di integrazione 19
3.2.4.3) Test di sistema 20
3.2.4.4) Testi di regressione 20
3.2.4.5) Test di accettazione 20
3.3) Validazione 20
3.3.1) Scopo ed aspettative 20
3.3.2) Descrizione 20
3.4) Gestione della configurazione 20
3.4.1) Descrizione 20
3.4.2) Scopo 20
3.4.3) Versionamento 21
3.4.4) Tecnologie utilizzate 21
3.4.5) Repository 21
3.4.5.1) Lista Repository 21
3.4.5.2) Pull Requests e Branch 21
3.4.5.3) Struttura della repository `docs` 22
3.4.5.4) Struttura della repository `jmap` 22
3.5) Gestione della qualità 23
3.5.1) Scopo 23
3.5.2) Descrizione 23
3.5.3) Aspettative 23
3.5.4) Metriche e strumenti 23
4) Processi organizzativi 23
4.1) Gestione dei processi 23
4.1.1) Scopo 23
4.1.2) Descrizione 24
4.1.3) Pianificazione 24
4.1.3.1) Scopo 24
4.1.3.2) Descrizione 24

4.1.3.3) Aspettative ... 24
4.1.3.4) Ruoli ... 24
4.1.3.5) Ticketing ... 25
4.1.4) Coordinamento ... 26
4.1.4.1) Comunicazioni ... 26
4.1.4.2) Riunioni ... 26
4.1.4.3) Verbali ... 27
4.2) Miglioramento ... 27
4.2.1) Scopo ... 27
4.2.2) Descrizione ... 27
4.2.2.1) Stabilimento dei processi ... 28
4.2.2.2) Valutazione dei processi ... 28
4.2.2.3) Miglioramento dei processi ... 28
4.2.3) Metriche ... 28
4.2.4) Strumenti ... 28
4.3) Formazione ... 28
4.3.1) Scopo ... 28
4.3.2) Descrizione ... 28
4.3.3) Aspettative ... 29
4.3.4) Strumenti ... 29
5) Standard ISO/IEC 9126 per la qualità ... 29
5.1) Funzionalitá ... 29
5.2) Affidabilitá ... 29
5.3) Usabilitá ... 30
5.4) Efficienza ... 30
5.5) Manutenibilità ... 30
5.6) Portabilità ... 30
6) Metriche di qualitá del processo ... 31
6.1) Processi primari ... 31
6.1.1) Fornitura ... 31
6.1.2) Sviluppo ... 31
6.2) Processi di supporto ... 32
6.2.1) Documentazione ... 32
6.2.2) Verifica ... 32
6.2.3) Gestione della qualitá ... 32
6.3) Processi organizzativi ... 32
6.3.1) Gestione dei processi ... 32
7) Metriche di qualitá del prodotto ... 32
7.1) Funzionalitá ... 32
7.2) Affidabilitá ... 33
7.3) Usabilitá ... 33
7.4) Efficienza ... 33
7.5) Manutenibilitá ... 33

1) Introduzione

1.1) Scopo del documento

Questo documento ha lo scopo di descrivere le regole relative al Way of Working adottato da parte del gruppo per lo svolgimento del progetto didattico. In esso, dunque, appaiono tutte le *best practices* da seguire per ciascun *processo* e correlate attività che lo compongono, seguendo nel fare ciò la struttura definita dallo standard ISO/IEC 12207:1995.
Per la stesura è stato intrapreso un approccio di tipo incrementale, ovvero che prevede una realizzazione in più passi con aggiunte successive ad un impianto base. Di conseguenza ogni aggiornamento avverrà in funzione delle decisioni prese dal gruppo durante lo svolgimento del progetto stesso.
I membri del gruppo si impegnano a visionare regolarmente questo documento e a rispettare con disciplina le regole definite in esso, per fare in modo di essere professionali, coerenti, sistematici ed uniformi nello svolgere il lavoro necessario.

1.2) Scopo del progetto

L'obiettivo primario del progetto è eseguire una valutazione approfondita sulle performance di un server implementato con il *protocollo JMAP* e sviluppato utilizzando il linguaggio di programmazione Java. Tale valutazione non si limiterà alla semplice analisi delle prestazioni, ma comprenderà un'analisi dettagliata dei limiti di tale implementazione.
Questo processo di valutazione è stato concepito con l'intento di fornire all'azienda *proponente* un quadro completo, facilitandola nel confronto con gli standard attualmente implementati. L'analisi sarà arricchita dai risultati ottenuti attraverso *stress test* elaborati dal nostro gruppo. Questi stress *test* mirano a mettere alla prova l'implementazione del protocollo JMAP sotto diverse condizioni, valutando aspetti cruciali come le performance in situazioni di carico elevato, la manutenibilità del sistema e la completezza dell'implementazione del protocollo.

1.3) Glossario

Per evitare ambiguità o incomprensioni riguardanti la terminologia usata nel documento, é stato deciso di adottare un glossario in cui vengono riportate le varie definizioni. In questa maniera in esso verranno posti tutti i termini specifici del dominio d'uso con relativi significati.
La presenza di un termine all'interno del glossario viene indicata applicando *questo stile*.

1.4) Riferimenti

1.4.1) Riferimenti normativi

- Capitolato d'appalto C8: JMAP, il nuovo protocollo standard per la comunicazione email https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C8.pdf
- Standard ISO/IEC 12207:1995: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf

1.4.2) Riferimenti informativi

- I processi di ciclo di vita del software https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf

2) Processi primari

2.1) Fornitura

2.1.1) Scopo

Come stabilito dallo standard *ISO*/*IEC* 1995, il processo di fornitura contiene le attività e i compiti del fornitore necessari per lo svolgimento del progetto. Il fornitore dovrà accordarsi con il proponente per stabilire, all'interno di un contratto, i requisiti, i vincoli e la data di consegna del prodotto finale.
Lo scopo di questo processo è di tracciare e descrivere le attività svolte da ogni componente del gruppo *Overture*, per determinare quanto lavoro è ancora da completare oppure è stato ultimato rispetto alle richieste del proponente.
Una volta trovato l'accordo con il proponente sarà possibile passare alla fase esecutiva, redigendo il `Piano di Progetto`.

2.1.2) Descrizione

Questo processo è diviso nelle seguenti fasi:
- Inizializzazione;
- Preparazione delle risposte alle richieste;
- Contrattazione;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Consegna e completamento.

2.1.3) Aspettative

Il gruppo *Overture* intende instaurare e mantenere uno stretto rapporto di collaborazione con l'azienda proponente *Zextras* e, in particolare, con le figure dei referenti: Alessio Crestani e Federico Rispo. Grazie ad un dialogo continuo, il gruppo intende:
- Ricevere feedback sul lavoro svolto;
- Verificare che i vincoli e i requisiti individuati corrispondano a quanto richiesto dal *capitolato* e dall'azienda proponente.

2.1.4) Comunicazioni con il proponente

L'azienda proponente *Zextras* fornisce l'indirizzo di posta elettronica, i profili *discord* dei referenti e la piattaforma *Carbonio* come canali di comunicazione attraverso i quali chiarire dubbi e stabilire futuri incontri telematici. Le riunioni esterne non hanno una cadenza regolare, ma vengono richieste in base alle necessità dal gruppo o dall'azienda.
Un incontro può essere richiesto, per esempio, per alcune delle seguenti necessità:
- Chiarimenti relativi a requisiti o vincoli del capitolato;
- Chiarimenti relativi alle tecnologie utilizzate;
- Richieste di feedback su quanto prodotto.

Per ogni colloquio con l'azienda proponente verrà redatto un resoconto sotto forma di `Verbale Esterno`, che riporterà nel nome e all'interno del documento la data del relativo incontro.
I verbali redatti potranno essere consultati all'interno della relativa cartella presente sul *repository* https://github.com/overture-unipd/docs/tree/master/documents, disponibile per ogni *baseline* del progetto.

2.1.5) Documentazione fornita

Vengono elencati di seguito i documenti che il gruppo *Overture* consegnerà all'azienda proponente *Zextras* e ai committenti *Prof. Tullio Vardanega* e *Prof. Riccardo Cardin*.

2.1.5.1) `Valutazione dei Capitolati`

La `Valutazione dei capitolati` è un documento che fornisce una panoramica dettagliata sui capitolati d'appalto, presentati il giorno 2023-10-17. Per ogni progetto si individuano le richieste del proponente, le possibili soluzioni ed eventuali criticità.
È suddiviso nelle seguenti sezioni:

- **Descrizione**: nome del progetto, azienda proponente, informazioni generali relative al prodotto da sviluppare secondo quanto descritto nella presentazione del capitolato;
- **Dominio applicativo**: contesto del progetto;
- **Dominio tecnologico**: tecnologie da utilizzare per lo sviluppo;
- **Aspetti positivi**;
- **Fattori critici**;
- **Conclusione**: motivazioni sulla scelta/non scelta del capitolato.

2.1.5.2) `Analisi dei Requisiti`

L'`Analisi dei Requisiti` è un documento che definisce le *funzionalità* che il prodotto è in grado di offrire ed i requisiti da soddisfare affinchè il software sviluppato sia conforme alle richieste fatte dal proponente. Contiene le seguenti informazioni:

- **Descrizione del prodotto**: obiettivo finale del prodotto e le sue funzionalità principali;
- **Lista dei casi d'uso**: identificazione di tutti gli scenari di utilizzo del sistema da parte degli utenti. Per ogni caso d'uso sono analizzati:
  - Scenario;
  - Attori coinvolti;
  - Azioni eseguibili.
- **Lista dei requisiti**: tutte le richieste o vincoli definiti dal proponente o dedotti dal team per la realizzazione del prodotto finale. I requisiti possono essere obbligatori, desiderabili e opzionali e verranno classificati dal gruppo a seconda della loro importanza.

2.1.5.3) `Piano di Progetto`

Il `Piano di Progetto` è un documento soggetto a *versionamento* e approvazione, redatto e aggiornato dal Responsabile con il supporto degli Amministratori durante tutta la durata del progetto. Ha l'obiettivo di delineare la pianificazione e la gestione delle attività necessarie per la realizzazione del progetto. Contiene le seguenti informazioni:

- **Analisi dei Rischi**: identificazione di eventuali problematiche riscontrate durante lo sviluppo che potrebbero rallentare o ostacolare le attività di progetto. Al fine di fare prevenzione sui problemi, il gruppo metterà a disposizione delle soluzioni per il team il prima possibile. Sono classificati in:
  - Rischi organizzativi;
  - Rischi tecnologici.
- **Modello di sviluppo**: descrizione dell'approccio metodologico e strutturato utilizzato dal gruppo per lo sviluppo del prodotto;
- **Pianificazione**: scansione dei periodi, con i relativi eventi e attività da svolgere, all'interno di un calendario. Per ogni periodo verranno inserite le suddivisioni dei ruoli e una stima dell'impegno richiesto da ogni componente del gruppo per svolgere le proprie attività.
- **Preventivo**: stima della durata di ciascun periodo, indicando il tempo necessario per completare tutte le attività previste;
- ***Consuntivo***: analisi del lavoro effettivamente svolto rispetto al preventivo, finalizzata a ottenere uno stato di avanzamento del progetto alla fine di ciascun periodo.

2.1.5.4) `Piano di Qualifica`

Il `Piano di Qualifica` è un documento formale che descrive i compiti e le attività svolte dal Verificatore all'interno del progetto, necessarie a garantire *qualità* al prodotto software che si intende sviluppare. È uno strumento fondamentale per la gestione del processo di sviluppo software: assicura che il prodotto finale sia conforme alle specifiche richieste e alle aspettative del committente, monitorando il suo stato di avanzamento rispetto agli obiettivi prefissati. Ogni membro del gruppo che parteciperà allo sviluppo farà riferimento a questo documento al fine di garantire la qualità richiesta.
È suddiviso nelle seguenti sezioni:

- **Qualità di processo**: definizione dei parametri e delle metriche da rispettare per garantire processi di qualità elevata;
- **Qualità di prodotto**: definizione dei parametri e delle metriche da rispettare per garantire un prodotto finale di qualità elevata;
- **Test**: descrizione dei test necessari per assicurare che i requisiti stabiliti vengano soddisfatti nel prodotto;
- **Valutazioni per il miglioramento**: resoconto dell'attività di verifica svolta e delle criticità riscontrate durante il processo di sviluppo del software.

2.1.5.5) `Glossario`

Il `Glossario` è un documento di supporto concepito sia per i membri del gruppo, ma anche per i committenti e l'azienda proponente. La sua stesura permette di evitare ambiguità o incomprensioni riguardanti la terminologia utilizzata in tutta la documentazione del progetto.

2.1.5.6) `Lettera di Presentazione`

La `Lettera di Presentazione` è il documento con cui il gruppo *Overture* esprime la volontà di partecipare alla fase di revisione del prodotto software. All'interno del documento viene elencata la documentazione messa a disposizione dei committenti e del proponente e i termini stabiliti per la consegna del prodotto ultimato.

2.1.6) Strumenti

Gli strumenti software utilizzati per il processo di fornitura sono di seguito descritti.

2.1.6.1) Google Calendar

È un sistema di calendari creato da Google, offre la possibilità di pianificare le riunioni interne e condividerle a tutti i membri del gruppo.

2.1.6.2) Typst

Linguaggio utilizzato per la creazione dei diari di bordo, attraverso l'omonimo sito typst.app.

2.1.6.3) Carbonio - Zextras Chats

Il prodotto principale dell'azienda *Zextras*. Verrà utilizzato dal gruppo *Overture* per effettuare videoconferenze con il proponente.

2.1.6.4) Discord

Piattaforma di messaggistica e videochat utilizzata dal gruppo per le riunioni interne e come metodo informale per contattare l'azienda proponente.

2.2) Sviluppo

2.2.1) Scopo

Facendo riferimento allo standard ISO/IEC 1995: lo scopo del processo di sviluppo è definire le attività di analisi, progettazione, codifica, integrazione, testing, installazione ed accettazione dei requisiti richiesti nel contratto.

2.2.2) Descrizione

Le attività che compongono il processo di sviluppo sono:

1. Implementazione del processo;
2. *Analisi dei Requisiti* di sistema;
3. Progettazione architetturale del sistema;
4. Analisi dei requisiti software;
5. Progettazione architetturale del software;
6. Progettazione dettagliata del software;
7. Codifica e testing del software;
8. Integrazione del software;
9. Test di qualifica del software;
10. Integrazione di sistema;
11. Test di qualifica di sistema;
12. Installazione del software;
13. Supporto all'accettazione del software.

2.2.2.1) Implementazione del processo

Nel processo di implementazione per lo sviluppo del software, il primo passo consiste nella scelta di un modello di ciclo di vita adatto al progetto. Dopo aver definito le responsabilità dello sviluppatore, che includono aspetti come la documentazione, la gestione delle configurazioni, la risoluzione dei problemi e l'esecuzione dei processi di supporto, si procede ad affinare ulteriormente il processo mediante la specifica dell'utilizzo di standard, metodi, strumenti e linguaggi di programmazione.
Segue la richiesta di formulare piani dettagliati, inclusivi di standard, metodi, strumenti e responsabilità, allo scopo di guidare le attività di sviluppo. Infine, si sottolinea l'importanza di garantire che, nonostante elementi non consegnabili possano essere impiegati durante lo sviluppo o la *manutenzione* del software, l'operatività e la manutenzione del prodotto software, dopo la consegna all'acquirente, siano indipendenti da tali elementi.

2.2.2.2) Analisi dei requisiti di sistema

Nel contesto dell'Analisi dei Requisiti di Sistema, il processo coinvolge il developer nell'esecuzione o supporto di attività contrattualmente previste. La prima fase richiede l'analisi dell'uso specifico del sistema in sviluppo per definire accuratamente i requisiti di sistema. Questi requisiti devono dettagliare funzioni, capacità, requisiti aziendali, organizzativi e utente, nonché aspetti come sicurezza, ingegneria dei fattori umani, interfaccia, operazioni e manutenzione, insieme a vincoli di progettazione e requisiti di qualifica, tutti documentati in modo approfondito.
La fase successiva richiede la valutazione dei requisiti di sistema secondo criteri come rintracciabilità ai bisogni di acquisizione, coerenza con tali bisogni, testabilità e fattibilità della progettazione architettonica del sistema, oltre alla fattibilità delle operazioni e della manutenzione. Anche i risultati di queste valutazioni devono essere accuratamente documentati.

2.2.2.3) Progettazione architetturale del sistema

Nel contesto della progettazione architetturale del sistema, la prima fase prevede la definizione di un'architettura di alto livello del sistema, identificando gli elementi hardware, software e le operazioni manuali. Si assicura l'allocazione di tutti i requisiti di sistema tra gli elementi e successivamente si individuano gli elementi di configurazione hardware, gli elementi di configurazione software e le

operazioni manuali. L'architettura del sistema e i requisiti di sistema allocati agli elementi vengono documentati.
La fase successiva comporta la valutazione dell'architettura di sistema e dei requisiti degli elementi, con attenzione ai criteri di rintracciabilità, coerenza, adeguatezza degli standard e metodi di progettazione, fattibilità degli elementi software nel soddisfare i requisiti loro allocati, e la fattibilità di operazione e manutenzione. I risultati di tali valutazioni vengono accuratamente documentati.

2.2.2.4) Analisi dei requisiti software

L'Analisi dei Requisiti Software richiede che, per ciascun elemento software o elemento di configurazione software identificato, lo sviluppatore esegua i seguenti compiti:

- Definizione e documentazione dei requisiti software, inclusi dettagliati orientamenti per le caratteristiche qualitative consultando la norma ISO/IEC 9126. Tale definizione comprende:
  1. Specifiche funzionali e di capacità, con considerazioni su prestazioni, caratteristiche fisiche e condizioni ambientali in cui l'elemento software opererà;
  2. Identificazione delle interfacce esterne all'elemento software;
  3. Stipula dei requisiti di qualifica;
  4. Specifiche di sicurezza, riguardanti i metodi di operazione e manutenzione, le influenze ambientali e il *rischio* di lesioni al personale;
  5. Specifiche di sicurezza, focalizzate sui compromessi di informazioni sensibili;
  6. Specifiche di ingegneria dei fattori umani (ergonomia), comprendenti operazioni manuali, interazioni umano-apparecchio, vincoli sul personale e ambiti che richiedono concentrazione umana, sensibili agli errori umani e formazione;
  7. Definizione dei dati e requisiti del *database*
  8. Requisiti di installazione e accettazione del prodotto software presso i siti di operazione e manutenzione;
  9. Documentazione per l'utente;
  10. Requisiti di operazione ed esecuzione dell'utente;
  11. Requisiti di manutenzione dell'utente.
- Valutazione dei requisiti software secondo i seguenti criteri, con documentazione dei risultati:
  1. Rintracciabilità ai requisiti di sistema e di progettazione del sistema;
  2. Coerenza esterna con i requisiti di sistema;
  3. Coerenza interna;
  4. Testabilità;
  5. Fattibilità della progettazione software;
  6. Fattibilità di operazione e manutenzione.
- Conduzione di una o più revisioni congiunte al cui completamento con successo, sarà istituita una baseline per i requisiti dell'elemento software.

2.2.2.5) Progettazione architetturale del software

Per ogni elemento software (o elemento di configurazione software, se identificato), l'attività include i seguenti compiti:

- Lo sviluppatore converte i requisiti dell'elemento software in un'architettura, delineandone la struttura di alto livello e identificando i componenti software. L'assegnazione di tutti i requisiti dell'elemento software ai suoi componenti e la loro ulteriore definizione per agevolare la progettazione dettagliata sono garantite. L'architettura dell'elemento software dovrà essere documentata.
- Sviluppo e documentazione di un progetto di alto livello per le interfacce esterne all'elemento software e tra i componenti software dello stesso;
- Sviluppo e documentazione di un progetto di alto livello per il database;
- Sviluppo e documentazione di versioni preliminari della documentazione per l'utente;

- Definizione e documentazione dei requisiti preliminari di prova e della pianificazione per l'integrazione del software;
- Valutazione dell'architettura dell'elemento software e dei progetti delle interfacce e del database, considerando criteri come rintracciabilità, coerenza interna ed esterna, appropriata utilizzo di metodi e standard di progettazione, fattibilità della progettazione dettagliata, e fattibilità di operazione e manutenzione. I risultati di queste valutazioni sono documentati.
- Conduzione di una o più revisioni.

2.2.2.6) Progettazione dettagliata del software

Per ogni elemento software (o elemento di configurazione software, se identificato), l'attività include i seguenti compiti:
- Sviluppo della progettazione dettagliata dei componenti software, ovvero, lo sviluppatore crea una progettazione dettagliata per ciascun componente software, definendo livelli inferiori con unità software che possono essere codificate, compilate e testate;
- Sviluppo e documentazione di una progettazione dettagliata per le interfacce esterne all'elemento software, tra i componenti software e tra le unità software. La progettazione dettagliata delle interfacce deve consentire la codifica senza ulteriori informazioni.
- Sviluppo e documentazione di una progettazione dettagliata per il database;
- Aggiornamento della documentazione utente;
- Definizione e documentazione dei requisiti di prova e pianificazione per i test delle unità software;
- Aggiornamento dei requisiti di prova e pianificazione per l'integrazione del software;
- Valutazione della progettazione dettagliata del software e dei requisiti di prova in base a criteri come rintracciabilità, coerenza interna ed esterna, appropriato utilizzo di metodi e standard di progettazione, fattibilità dei test, e fattibilità di operazione e manutenzione. I risultati vengono documentati.
- Conduzione di una o più revisioni.

2.2.2.7) Codifica e testing del software

Innanzitutto, il programmatore è tenuto a sviluppare e documentare i seguenti:
- Ogni singola unità software e database;
- Procedure di test e dati per il testing di ciascuna unità software e database.

Successivamente, il programmatore deve condurre i test su ogni unità software e database, assicurandosi che soddisfino i requisiti stabiliti. I risultati di tali test devono essere accuratamente documentati. In parallelo, è necessario aggiornare la documentazione utente in base alle modifiche apportate. Inoltre, si deve procedere all'aggiornamento dei requisiti di test e della pianificazione per l'Integrazione del Software.
La valutazione del codice software e dei risultati dei test è una fase cruciale e deve essere eseguita secondo i seguenti criteri, con i relativi risultati documentati:
- Tracciabilità rispetto ai requisiti e al design dell'elemento software;
- Coerenza esterna con i requisiti e il design dell'elemento software;
- Coerenza interna tra i requisiti delle singole unità;
- Copertura dei test sulle singole unità;
- Appropriato utilizzo di metodi e standard di codifica;
- Fattibilità dell'integrazione e del testing del software;
- Fattibilità di operatività e manutenzione del software.

2.2.2.8) Integrazione del software

Per prima cosa, il programmatore è tenuto a sviluppare un piano di integrazione per unire le unità software e i componenti software nell'elemento software. Questo piano deve includere requisiti di test, procedure, dati, responsabilità e una pianificazione, il tutto accuratamente documentato.

Successivamente, il programmatore deve procedere con l'integrazione delle unità software e dei componenti software, effettuando i test conforme al piano di integrazione. Si deve garantire che ciascun aggregato soddisfi i requisiti dell'elemento software e che l'integrazione sia completata al termine dell'attività. Tutti i risultati di integrazione e test devono essere accuratamente documentati.
Parallelamente, è necessario aggiornare la documentazione utente in base alle modifiche apportate.
Successivamente, per ciascun *requisito* di qualificazione dell'elemento software, il programmatore deve sviluppare e documentare un insieme di test, casi di test (input, output, criteri di test) e procedure di test per condurre il Test di Qualificazione del Software. Si deve garantire che l'elemento software integrato sia pronto per il Test di Qualificazione del Software.
La valutazione del piano di integrazione, del design, del codice, dei test, dei risultati dei test e della documentazione utente deve essere eseguita considerando i seguenti criteri, con i relativi risultati documentati:
- Tracciabilità ai requisiti di sistema;
- Coerenza esterna con i requisiti di sistema;
- Coerenza interna;
- Copertura dei test rispetto ai requisiti dell'elemento software;
- Appropriato utilizzo di standard e metodi di test;
- Conformità ai risultati attesi;
- Fattibilità del test di qualificazione del software;
- Fattibilità di operatività e manutenzione.

2.2.2.9) Test di qualifica del software

Inizialmente, il programmatore deve condurre il testing di qualificazione conformemente ai requisiti di qualificazione dell'elemento software. Si deve garantire che l'implementazione di ciascun requisito software sia testata per la conformità. I risultati del testing di qualificazione devono essere accuratamente documentati.
Parallelamente, è necessario aggiornare la documentazione utente in base alle modifiche apportate.
Successivamente, il programmatore deve valutare il design, il codice, i test, i risultati dei test e la documentazione utente considerando i seguenti criteri, con i risultati delle valutazioni documentati:
- Copertura dei test rispetto ai requisiti dell'elemento software;
- Conformità ai risultati attesi;
- Fattibilità dell'integrazione e del testing di sistema, se condotti;
- Fattibilità di operatività e manutenzione.

I risultati delle verifiche devono essere accuratamente documentati. Alla conclusione con successo delle verifiche il programmatore deve:
- Aggiornare e preparare il prodotto software consegnabile per l'Integrazione di Sistema;
- Stabilire una baseline per il design e il codice dell'elemento software.

2.2.2.10) Integrazione di sistema

Inizialmente, gli elementi di configurazione del software devono essere integrati, insieme agli elementi di configurazione dell'hardware, alle operazioni manuali e ad altri sistemi, come necessario, all'interno del sistema. Gli aggregati devono essere testati man mano che vengono sviluppati, verificando la conformità ai loro requisiti. L'integrazione e i risultati dei test devono essere accuratamente documentati.
Successivamente, per ciascun requisito di qualificazione del sistema, è necessario sviluppare e documentare un insieme di test, casi di test (input, output, criteri di test) e procedure di test. La valutazione del sistema integrato deve essere effettuata considerando i seguenti criteri, con i risultati delle valutazioni documentati:
- Copertura dei test rispetto ai requisiti di sistema;

- Appropriatezza dei metodi di test e degli standard utilizzati;
- Conformità ai risultati attesi;
- Fattibilità del test di qualificazione di sistema;
- Fattibilità di operatività e manutenzione.

2.2.2.11) Test di qualifica del sistema

Inizialmente, il testing di qualificazione di sistema deve essere condotto conformemente ai requisiti di qualificazione specificati per il sistema. È necessario garantire che l'implementazione di ciascun requisito di sistema sia testata per la conformità e che il sistema sia pronto per la consegna. I risultati del testing di qualificazione devono essere accuratamente documentati.
Successivamente, il sistema deve essere valutato considerando i seguenti criteri, con i risultati delle valutazioni documentati:
- Copertura dei test rispetto ai requisiti di sistema;
- Conformità ai risultati attesi;
- Fattibilità di operatività e manutenzione.

I risultati delle verifiche devono essere accuratamente documentati. Si noti che questa sotto-clausola non è applicabile agli elementi di configurazione del software per i quali le verifiche sono state condotte in precedenza.
Al termine con successo delle verifiche, se condotte, il programmatore deve:
- Aggiornare e preparare il prodotto software consegnabile per l'installazione del software e il supporto all'accettazione del software;
- Stabilire una baseline per il design e il codice di ciascun elemento di configurazione del software.

2.2.2.12) Installazione del software

Inizialmente, il programmatore deve sviluppare un piano per installare il prodotto software nell'ambiente di destinazione designato nel contratto. Le risorse e le informazioni necessarie per l'installazione del prodotto software devono essere determinate e rese disponibili. Come specificato nel contratto, il programmatore deve assistere l'acquirente nelle attività di configurazione. Nel caso in cui il prodotto software installato sostituisca un sistema esistente, il programmatore deve supportare eventuali attività di esecuzione parallela richieste dal contratto. Il piano di installazione deve essere documentato.
Successivamente, il programmatore deve procedere con l'installazione del prodotto software conformemente al piano di installazione. Si deve garantire che il codice del software e i database si inizializzino, eseguano e terminino come specificato nel contratto. Gli eventi e i risultati dell'installazione devono essere accuratamente documentati.

2.2.2.13) Supporto all'accettazione del software

Innanzitutto, il programmatore è tenuto a supportare la revisione e il testing di accettazione del prodotto software da parte dell'acquirente. La revisione e il testing di accettazione devono tenere conto dei risultati delle Verifiche, del Testing di Qualificazione del Software e del Testing di Qualificazione di Sistema. I risultati della revisione e del testing di accettazione devono essere accuratamente documentati.
Successivamente, il programmatore deve completare e consegnare il prodotto software come specificato nel contratto.

3) Processi di supporto

3.1) Documentazione

3.1.1) Scopo

Il processo di documentazione serve a tenere traccia di tutti processi e attività relativi al ciclo di vita del software, riportando le decisioni adottate e le *norme* attuate dal gruppo durante lo svolgimento del progetto. Le norme stabilite all'interno di questo documento verranno rispettate da tutti i membri del gruppo *Overture*.

3.1.2) Descrizione

La documentazione software traccia il lavoro svolto e le decisioni prese. Questa sezione del documento si occupa di tutte le norme adottate dal gruppo relative alla documentazione.

3.1.3) Aspettative

- Definire delle regole chiare e concise per la stesura di tutti i documenti relativi al progetto;
- Creare template per tutti i tipi di documenti per garantire omogeneità.

3.1.4) Ciclo di vita

Il ciclo di vita di un documento è composto da sette fasi:

1. Creazione o adattamento del template: la prima fase prevede la creazione o l'adattamento di un template per il documento corrente. Il template contiene la struttura e la formattazione del documento, nonché le informazioni di base, come titolo, autore e data.
2. Pianificazione e assegnazione delle sezioni: nella seconda fase le sezioni del documento vengono pianificate e assegnate ai Redattori incaricati. I Redattori sono responsabili della stesura delle proprie sezioni in conformità con le `Norme di Progetto`.
3. Raccolta dei contenuti e creazione della prima bozza: nella terza fase i Redattori raccolgono i contenuti da discutere e creano una prima bozza del documento. La bozza viene utilizzata come punto di partenza per la discussione e la revisione.
4. Stesura effettiva del documento: la quarta fase vede i Redattori redirigere le proprie sezioni in conformità con il modello e le `Norme di Progetto`;
5. Controllo dei contenuti: nella quinta fase i Redattori verificano che il contenuto delle proprie sezioni sia conforme alle `Norme di Progetto` e non contenga errori di compilazione;
6. Revisione: la sesta fase prevede che un Verificatore incaricato revisioni il documento per assicurarsi che le modifiche apportate siano corrette;
7. Approvazione e rilascio: nell'ultima fase il documento viene approvato da un Responsabile e rilasciato in versione finale.

3.1.5) Sistema di composizione tipografica

Per la composizione tipografica dei documenti, abbiamo deciso di utilizzare *Typst*, al posto del noto *LaTeX*. Typst offre diversi vantaggi rispetto a LaTeX:

- semplicità di utilizzo (simile a Markdown);
- programmabilità reale invece di un sistema di macro;
- compilazione pressochè immediata.

L'utilizzo di Typst semplifica la creazione e la manutenzione dei documenti, liberando i redattori dalla responsabilità della visualizzazione grafica e garantendo una certa coerenza nella documentazione del progetto.
Il template sviluppato ed utilizzato è presente nella repository docs, nella cartella `templates`.

3.1.6) Struttura dei documenti

Ogni documento prodotto viene organizzato nelle seguenti sezioni:

3.1.6.1) Intestazione

La prima pagina funge da intestazione del documento e presenta gli elementi di seguito:

- **Nome del documento**;
- **Data**: la data in cui è stata approvata l'ultima versione del documento;
- **Versione**: la versione corrente del documento;
- **Logo del gruppo**: presente nel percorso `imgs/group_logo.png`;
- **Email**: overture.unipd@gmail.com
- **Destinatari**: a chi è il rivolto il documento;
- **Responsabile**: chi ha approvato il documento;
- **Redattori**: incaricati della stesura del documento;
- **Verificatori**: incaricati della verifica del documento.

3.1.6.2) Registro delle modifiche

La seconda pagina è dedicata al registro delle modifiche. Le informazioni sono organizzate in una tabella e permettono di tenere traccia dei cambiamenti subiti dal documento. La tabella riporta i seguenti dati:

- **Versione**: il numero di versione del documento;
- **Data**: data di approvazione del documento;
- **Autori**: chi ha effettuato le modifiche;
- **Verificatori**: chi ha approvato le modifiche;
- **Dettaglio**: una breve descrizione.

3.1.6.3) Indice

Nella terza pagina è presente l'indice che elenca le sezioni contenute nel documento.

3.1.6.4) Corpo del documento

Il contenuto del documento è suddiviso in capitoli, ognuno dei quali è formato da più sezioni ed eventuali sottosezioni.

3.1.6.5) Corpo del verbale

Il contenuto del verbale è suddiviso nelle seguenti sezioni:

- **Informazioni sulla riunione**:
  - **Luogo**: può essere il luogo fisico dove si è tenuto l'incontro oppure il nome della piattaforma online utilizzata;
  - **Ora di inizio**;
  - **Ora di fine**;
  - **Partecipanti**: i nomi dei componenti del gruppo che hanno partecipato alla riunione;
  - **Partecipanti esterni**: i nomi di eventuali partecipanti esterni.
- **Ordine del giorno**: un elenco di ciò che verrà discusso durante la riunione;
- **Sintesi dell'incontro**: contiene un breve riassunto delle discussioni e dei temi affrontati durante l'incontro;
- **Decisioni prese**: sezione che elenca in forma testuale le decisioni prese durante l'incontro. Alcune di queste potrebbero risultare in "attività individuate".
- **Attività individuate**: illustrazione dettagliata delle attività assegnate ai diversi membri del gruppo a conclusione dell'incontro. Queste informazioni, inserite in un'apposita tabella, riportano:
  - **ID**: collegamento alla relativa *issue* su *GitHub*
  - **Dettaglio**: breve spiegazione dell'attività;
  - **Assegnatari**: i nomi degli incaricati a svolgere l'attività.

3.1.6.6) Documenti del progetto

Verranno prodotti i seguenti documenti:

- **`Norme di Progetto`**;
- **`Piano di Progetto`**;
- **`Piano di Qualifica`**;
- **`Analisi dei Requisiti`**;
- **`Glossario`**;
- **`Verbali Interni`**;
- **`Verbali Esterni`**;

3.1.7) Regole stilistiche

3.1.7.1) Nomi assegnati ai file

I documenti PDF presenti nella repository docs, rispettano le seguenti regole per la nominazione dei file:

- Minuscolo per i nomi, tranne che per i verbali (marcati `VI` e `VE`, rispettivamente per interni ed esterni);
- Spaziatura fra le parole sostituita da un underscore;
- Per i verbali, la data dell'incontro è presente nel nome;
- Versione del documento alla fine del nome del file.

I nomi dei documenti presenti nel progetto saranno quindi del tipo:

- **`Norme di Progetto`**: `norme_di_progetto_vX.X.X`;
- **`Piano di Progetto`**: `piano_di_progetto_vX.X.X`;
- **`Piano di Qualifica`**: `piano_di_qualifica_vX.X.X`;
- **`Analisi dei Requisiti`**: `analisi_dei_requisiti_vX.X.X`;
- **`Glossario`**: `glossario_vX.X.X`;
- **`Verbali Interni`**: `VI_YYYY_MM_DD_vX.X`;
- **`Verbali Esterni`**: `VE_YYYY_MM_DD_vX.X`.

Si noti che i sorgenti `.typ` non includono la versione nel nome, ma è aggiunta ai PDF dopo la compilazione.
Usare lo stesso nome per i documenti consente di utilizzare Git in modo appropriato: tracciare i cambiamenti di file di testo, con relativa facilità di utilizzo della funzione di "diff".

3.1.7.2) Stile del testo

Nei documenti, esclusi i verbali, verranno utilizzati:

- Il *corsivo* per:
  - Il nome del gruppo (*Overture*);
  - Il nome dell'azienda proponente (*Zextras*).
- Il **grassetto** per:
  - Parole seguite da descrizione negli elenchi puntati;
  - Termini importanti.
- Un `font monospace` per:
  - I nomi dei documenti;
  - I nomi dei file;
  - I nomi delle repository;
  - I nomi delle cartelle;
  - I nomi dei *branch*
  - Esempi di codice.
- Il sottolineato per:
  - I link;

  - L'indirizzo email.
- Le lettere maiuscole per:
  - Le iniziali dei nomi;
  - Gli acronimi;
  - Le iniziali dei ruoli svolti dai componenti del gruppo.

3.1.8) Elenchi puntati

Ogni voce dell'elenco termina con un ";" tranne per l'ultima, che finisce con un ".". Fanno eccezione le voci composte da più di una frase, che possono terminare con ".", indipendentemente dalla posizione nell'elenco. Inoltre, il grassetto è usato dove necessario, come descritto nella sezione subito sopra.

3.1.8.1) Formato delle date

Viene adottato lo standard internazionale **ISO 8601**, nella forma `YYYY-MM-DD`, indicante rispettivamente:
- **`YYYY`**: l'anno con 4 cifre;
- **`MM`**: il mese con 2 cifre;
- **`DD`**: il giorno con 2 cifre.

3.1.9) Strumenti

I seguenti strumenti sono stati scelti dal gruppo per la realizzazione della documentazione:
- **Typst**: linguaggio per la stesura dei documenti, tramite typst.app
- **GitHub**: servizio di *hosting* di repository.

3.2) Verifica

3.2.1) Scopo ed aspettative

La verifica nel ciclo di vita del software è un processo continuo che inizia dalla fase iniziale di progettazione e si estende fino alla manutenzione successiva. Questo elemento cruciale mira a garantire la conformità tra gli output del software (codice sorgente, documentazione, test e così via) e le relative aspettative, fondandosi su criteri come coerenza, completezza e correttezza dei risultati.
L'obiettivo principale è implementare un processo di verifica per ogni prodotto, assicurando *efficienza* e accuratezza. Attraverso l'applicazione di tecniche di analisi e test, il processo verifica che i prodotti soddisfino i requisiti specificati. Seguire procedure definite, adottare criteri affidabili e validare il prodotto al termine della verifica sono elementi chiave per garantirne il corretto sviluppo.
La stabilità del prodotto, risultato del processo di verifica, è fondamentale per agevolare il passaggio verso la successiva fase di validazione, assicurando complessivamente la qualità del software.

3.2.2) Descrizione

La verifica rappresenta un processo affidato a un team di verificatori, che si estende a tutti i processi in corso al fine di garantire l'aderenza agli standard stabiliti. Questo procedimento non è un evento isolato, ma piuttosto un ciclo continuo che si ripete periodicamente, adattandosi alle mutevoli esigenze del progetto nel corso del tempo.
Il fulcro di questo processo è il `Piano di Qualifica`, un documento dettagliato che traccia il percorso della verifica. Esso delinea chiaramente gli obiettivi da raggiungere, i criteri di accettazione da rispettare e i metodi specifici che saranno impiegati per condurre la verifica in modo accurato ed efficiente.
La documentazione derivante dalla verifica non è semplicemente un compito burocratico, ma riveste un ruolo cruciale nell'assicurare la trasparenza, la ripetitibilità e la tracciabilità dell'intero processo. Nei seguenti punti, saranno elencate le possibili attività da adottare.

3.2.3) Analisi statica

L'analisi statica rappresenta una metodologia di verifica che prescinde dall'esecuzione del prodotto, fondata su una revisione critica del codice e della documentazione. L'obiettivo primario di questa analisi è verificare la conformità ai vincoli, l'assenza di difetti e la presenza delle proprietà desiderate. Applicabile a qualsiasi prodotto del progetto, l'analisi statica adotta comunemente due metodi di lettura: il *walkthrough* e l'*inspection*.
Il walkthrough, una tecnica collaborativa che coinvolge sia il verificatore che l'autore del prodotto, prevede una lettura a pettine della documentazione e del codice sorgente.
L'inspection, preferibile al walkthrough per la sua velocità ed efficienza, si basa su una lista di controllo e consente di identificare tempestivamente e sistematicamente potenziali problematiche.
L'analisi statica costituisce una fase cruciale nel processo di verifica, ed è vantaggiosa nelle prime fasi del progetto quando i documenti sono ancora relativamente semplici e tutte le attività accuratamente documentate.

3.2.4) Analisi dinamica

L'analisi dinamica costituisce una tecnica di verifica del software che si basa sull'esecuzione del codice al fine di individuare difetti o eventuali problemi di funzionamento ed assicurare la qualità del prodotto finale.
Le tecniche principali utilizzate in questa fase sono i test, rappresentati da esecuzioni del codice con insiemi specifici di dati di input, finalizzati a verificare il comportamento atteso del software.
L'*efficacia* di un test è legata alla sua natura decidibile e ripetibile. Il termine "decidibile" implica che, dati gli stessi input, il test deve produrre sempre lo stesso risultato, mentre il termine "ripetibile" suggerisce che il test può essere eseguito più volte senza che i risultati siano influenzati da fattori esterni. Fattori determinanti per l'efficacia dei test includono la qualità del codice, l'accurata identificazione dei requisiti e la definizione di un dominio di casi di prova adeguato. Quest'ultimo rappresenta l'insieme completo di tutti i possibili casi di prova che possono essere eseguiti sul software, ed è responsabilità del verificatore definirlo in base ai requisiti del software e alla sua complessità.
L'automazione dei test può essere realizzata attraverso l'utilizzo di strumenti specifici come driver, stub e logger. I driver sono componenti attive che guidano il test, mentre gli stub sono componenti passive che simulano parti del sistema non direttamente coinvolte nel test. I logger, invece, registrano i risultati dei test.
Diversi tipi di test sono disponibili, differenziandosi sulla base della dimensione dell'oggetto del test.

3.2.4.1) Test di unità

I test di unità sono definiti sulle componenti software autonome più piccole e sono implementati principalmente durante la progettazione. Assumono due connotazioni diverse, in base al tipo di controllo che si vuole effettuare:
- **Test funzionali**: verificano che l'output effettivo corrisponda con il valore atteso;
- **Test strutturali**: verificano tutti i possibili cammini del codice tramite una serie di test.

3.2.4.2) Test di integrazione

Sono concepiti durante la fase di progettazione architetturale, successivamente ai test di unità. Verificano, con un approccio incrementale, la corretta integrazione tra le diverse unità software già testate. In aggiunta, è possibile annullare tali modifiche in modo da ripristinare uno stato sicuro nel caso si verifichino errori durante l'esecuzione di questo genere di test.
E' possibile seguire due approcci di integrazione:
- **Top-down**: si comincia con le componenti di sistema che hanno più dipendenze e maggiore rilevanza esterna, garantendo così la disponibilità immediata delle funzionalità di alto livello. Questo approccio consente di testare in modo più approfondito le funzionalità centrali, rendendole disponibili prioritariamente.

- **Bottom-up**: si inizia dalle componenti di sistema con meno dipendenze e maggior valore interno, ovvero quelle meno visibili all'utente.

3.2.4.3) Test di sistema

Sono concepiti successivamente ai test di integrazione, durante la stesura dell'`Analisi dei Requisiti`. Servono a garantire il corretto funzionamento del sistema. Verificano, in particolare, che tutti i requisiti software specificati nel capitolato siano presenti e funzionanti.

3.2.4.4) Testi di regressione

I controlli di regressione si assicurano che le correzioni o le espansioni apportate a specifiche componenti architetturali non arrechino danni al resto del sistema. Questi controlli prevedono la ripetizione mirata di test di unità, integrazione e sistema essenziali per garantire che le modifiche non compromettano le funzionalità precedentemente verificate, evitando così il verificarsi di regressioni.

3.2.4.5) Test di accettazione

I test di accettazione servono a verificare che il prodotto finale soddisfi tutti i requisiti richiesti dal committente: per questo motivo devono essere svolti obbligatoriamente in sua presenza.

3.3) Validazione

3.3.1) Scopo ed aspettative

La validazione costituisce l'essenziale verifica che il prodotto software sia in linea con i requisiti e le aspettative del cliente, rappresentando una fase cruciale nello sviluppo del software.
Questo processo si concentra attentamente su diversi aspetti:

- **Conformità ai requisiti**: il prodotto deve soddisfare integralmente tutti i requisiti specificati dal cliente;
- **Funzionamento corretto**: il prodotto deve operare correttamente, in conformità con la logica di progettazione;
- **Usabilità**: il prodotto deve essere intuitivo e di facile utilizzo;
- **Efficacia**: il prodotto deve dimostrarsi efficace nel soddisfare le necessità dei clienti.

L'aspettativa finale è di giungere ad un prodotto che risponda pienamente ai requisiti identificati ed operi correttamente.

3.3.2) Descrizione

Durante la fase di validazione, concentreremo l'attenzione sull'utilizzo dei test precedentemente eseguiti durante l'attività di verifica, dettagliatamente normati nelle sezioni pertinenti delle `Norme di Progetto`. Con l'esecuzione del test di accettazione concluderemo la validazione del prodotto.

3.4) Gestione della configurazione

3.4.1) Descrizione

Il processo di gestione della configurazione viene attuato durante tutta la vita del software e identifica le norme adottate dal gruppo al fine di garantire la tracciabilità della documentazione e del codice prodotto.

3.4.2) Scopo

Lo scopo di questo processo è di gestire e organizzare la procedura di modifica sulla documentazione e sul codice prodotto relative al progetto. Le modifiche passate saranno accessibili in qualsiasi momento,

in questo modo sarà possibile controllare rapidamente le motivazioni alla base dei cambiamenti effettuati ed i relativi autori.

3.4.3) Versionamento

Il versionamento consente di tracciare le modifiche avvenute all'interno di un documento. In questo modo è possibile visualizzare i cambiamenti avvenuti nel tempo e, nel caso ce ne fosse bisogno, riportare il documento ad uno stadio precedente.
Il gruppo *Overture* ha deciso di adottare il seguente formato per il versionamento dei documenti:

`X.Y.Z`

dove:
- **`X`**: rappresenta il completamento in vista di una delle fasi del progetto. Viene creata dal Responsabile al momento della validazione.
- **`Y`**: rappresenta una versione intermedia, che integra più di una modifica incrementale (`Z`) in maniera coerente. Il documento ad una versione del tipo `X.Y.0` è considerato stabile e coerente, seppur potenzialmente incompleto.
- **`Z`**: rappresenta una qualsiasi modifica incrementale. Un esempio è l'aggiunta e/o modifica di una sezione.

Le versioni dei verbali variano da questa convenzione: la versione è formata solamente da due cifre, dovuta alla brevità del testo e relativo tempo di scrittura. Ogni cambiamento, seguito dalla relativa approvazione, produce un incremento di versione che ha un peso diverso a seconda dell'entità della modifica effettuata. Inoltre, l'incremento di una qualsiasi cifra produrrà l'azzerameto di tutte le cifre alla sua destra.

3.4.4) Tecnologie utilizzate

- **Git**: software utilizzato per il controllo di versione dei documenti e del codice;
- **Github**: servizio di hosting per progetti software utilizzato dal gruppo per coordinare le operazioni di versionamento. Viene utilizzato anche come Issue Tracking System.

3.4.5) Repository

3.4.5.1) Lista Repository

Il gruppo utilizza 2 repository all'interno della propria "organizzazione *GitHub*":
- https://github.com/overture-unipd/docs: contenente tutta la documentazione del progetto;
- https://github.com/overture-unipd/jmap: contenente tutto il codice del progetto.

È poi presente la repository contenente il sito web del gruppo. Questa non è rilevante al fine di questa discussione, poichè non riceve aggiornamenti.

3.4.5.2) Pull Requests e Branch

Ogni attività da svolgere è tracciata come "issue" su GitHub, nella repository corrispondente: `docs` per quelle sulla documentazione, `jmap` per quelle sul codice.

Ogni attività viene sviluppata su un branch dedicato, separato da quello principale. Attività collegate possono essere sviluppate sullo stesso branch, per agevolare l'integrazione tra le parti.

Quando lo sviluppo di una attività (o serie di attività) è terminato su un branch, viene aperta una Pull Request, dal branch interessato a quello principale. Il verificatore designato verificherà quindi il lavoro svolto. Se conforme e senza errori, la Pull Request viene approvata ed il branch eliminato.

Utilizzare i branch come descritto, permette a tutti gli effetti di avere ambienti di lavoro distinti. Questo riduce il tempo di risoluzione di eventuali conflitti al solo processo di Pull Request.

3.4.5.3) Struttura della repository `docs`

Il contenuto della repository è separato da due branch che svolgono funzioni differenti: `master` contiene i PDF compilati, `sources` contiene invece i sorgenti Typst.

Per entrambe le repository, abbiamo deciso di separare in modo netto i documenti dal resto dei file di supporto (`LICENSE`, `README`, `actions`, `templates`, ...). Nella cartella `documents` si trova quindi la documentazione relativa alle varie fasi del progetto: `candidatura` (per la gara di appalto dei capitolati), `rtb` (Requirements and Technology Baseline), e `pb` (Product Baseline). Le cartelle relative alle versioni dei documenti sono organizzate nel modo seguente:

- **`candidatura`** contenente in:
  - `dichiarazione_impegni_v1.1.pdf`;
  - `lettera_di_presentazione.pdf`;
  - `valutazione_capitolati_v1.1.pdf`;
  - Una cartella **`verbali`** con i verbali divisi nelle cartelle:
    - **`interni`**;
    - **`esterni`**.
- **`rtb`**, che contiene le cartelle:
  - **`esterni`**, in cui si trovano:
    - `piano_di_qualifica_v1.0.0.pdf`;
    - `piano_di_progetto_v1.0.0.pdf`;
    - `analisi_dei_requisiti_v1.0.0.pdf`;
    - `glossario_v1.0.0.pdf`;
    - **`verbali`**: cartella contenente i verbali esterni.
  - **`interni`**, con all'interno:
    - `norme_di_progetto_v1.0.0.pdf`;
    - **`verbali`**: cartella contenente i verbali interni.
- **`pb`** suddivisa in:
  - **`esterni`**, in cui si troveranno:
    - `piano_di_qualifica_v2.0.0.pdf`;
    - `piano_di_progetto_v2.0.0.pdf`;
    - `analisi_dei_requisiti_v2.0.0.pdf`;
    - `manuale_utente_v1.0.0.pdf`;
    - `glossario_v2.0.0.pdf`;
    - **`verbali`**: cartella contenente i verbali esterni.
  - **`interni`**, con all'interno:
    - `norme_di_progetto_v2.0.0.pdf`;
    - **`verbali`**: cartella contenente i verbali interni.

3.4.5.4) Struttura della repository `jmap`

I file più importanti presenti nel repository sono quelli relativi all'implementazione del server. Abbiamo

- `src/`: cartella contentente il codice del server, compreso di test di unità ed integrazione
- `.env`: variabili d'ambiente utilizzate per la configurazione del server
- `compose.yml`: configurazione di *Docker Compose*
- `.justfile`: configurazione per https://github.com/casey/just. Serve a facilitare l'esecuzione di vari comandi all'interno della repository.

- `build.gradle.kts`: file di configurazione di *Gradle* (il sistema di build).

Ci sono poi file alcuni file di supporto: `README`, `LICENSE` e `.github/` (contentente la configurazione delle actions GitHub).

In questa repository, utilizziamo due branch principali: `master` contenente l'ultima versione stabile del codice e `develop` gli ultimi sviluppi. Quest'ultima funziona quindi da "buffer" di test delle varie attività svolte.
Salvo occasioni speciali, quindi, i nuovi branch di sviluppo partono direttamente da `develop` per riportare poi le modifiche direttamente in `develop`.
In ogni caso, le issue relative al ruolo svolto, vengono chiuse dopo l'unione in master.

Si noti che durante le prime fasi di sviluppo, tutte le attività sono state svolte direttamente in `develop`, per essere poi unite in `master`.
La motivazione è semplice: poichè la struttura non era ancora precisamente delineata e le attività da svolgere molto interconnesse, si rischiava di sviluppare più volte le stesse cose oltre che generare conflitti per modifica parallela delle stesse linee di codice.
L'approccio utilizzato è il seguente: all'inzio di ogni sessione di sviluppo e prima di ogni push, i componenti integravano lo stato remoto nella repository locale, risolvendo eventuali conflitti ed errori di compilazione.
Questa scelta ci ha fatto risparmiare parecchio tempo e fatica.

3.5) Gestione della qualità

3.5.1) Scopo

Il processo di gestione della qualità mira a garantire in modo adeguato che il software, gli artefatti ed i processi nel ciclo di vita del progetto siano allineati ai piani stabiliti e conformi ai canoni di qualità rispetto ai requisiti specificati.

3.5.2) Descrizione

Per poter garantire un determinato livello di qualità, abbiamo scelto di fare affidamento sui processi di verifica e validazione. Difatti, una volta definiti gli standard qualitativi da rispettare nel `Piano di Qualifica`, ciò che resta da fare è assicurarsi che questi vengano effettivamente applicati. Dunque inizialmente sono state definite le best practices per guidare l'esecuzione dei vari processi, oltre ai diversi canoni di qualità per i prodotti del progetto, ed in seguito per certificare l'adesione a quest'ultimi abbiamo delegato tale responsabilità ai Verificatori.

3.5.3) Aspettative

Ci si attende che il gruppo rispetti gli standard qualitativi definiti di comune accordo, realizzando documentazione adeguata e prodotti software all'altezza delle aspettative.

3.5.4) Metriche e strumenti

Per poter garantire la gestione della qualità utilizziamo delle metriche che sono differenti a seconda del processo coinvolto. In questo modo abbiamo uno strumento oggettivo per valutare il lavoro svolto.

4) Processi organizzativi

4.1) Gestione dei processi

4.1.1) Scopo

Il processo di gestione, come stabilito dallo standard ISO/IEC 12207:1997, identifica le attività generali e compiti che ogni membro del gruppo dovrà attuare per la gestione dei processi di progetto.

4.1.2) Descrizione

Questo processo è suddiviso nelle seguenti attività:
- Inizio e definizione dello scopo;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Chiusura.

4.1.3) Pianificazione

4.1.3.1) Scopo

Come stabilito dallo standard ISO/IEC 12207:1997, il Responsabile ha il compito di predisporre i piani per l'esecuzione di tutte le attività relative alla pianificazione. I piani dovranno contenere la descrizione delle attività e dei compiti associati.
Il Responsabile redigerà questa pianificazione all'interno del documento `Piano di Progetto`, che riporterà una descrizione delle attività e dei compiti necessari a raggiungere l'obiettivo prefissato in un determinato periodo.

4.1.3.2) Descrizione

L'attività di pianificazione verrà articolata nelle seguenti sezioni:
- Ruoli;
- Ticketing.

4.1.3.3) Aspettative

L'attività di pianificazione serve a stabilire delle regole comuni che il gruppo *Ouverture* attuerà per la sua organizzazione lavorativa.

4.1.3.4) Ruoli

I ruoli svolti dai membri del gruppo per il progetto sono decisi dal Responsabile di Progetto. Al termine del progetto ogni componente dovrà aver ricoperto tutti i ruoli, che sono di seguito descritti.

Responsabile di Progetto
Il Responsabile ha il compito fondamentale di coordinare i membri del gruppo e rappresentarlo presso il proponente e i committenti.
I suoi principali compiti sono:
- Approvare la documentazione;
- Gestire la pianificazione del progetto: determina le attività da svolgere e la loro priorità;
- Coordinare i membri: assegna e verifica l'avanzamento dei compiti che devono essere protati a termine;
- Studiare e gestire l'analisi dei rischi;
- Curare i rapporti tra i membri del gruppo e soggetti esterni.

Amministratore di Progetto
L'Amministratore definisce, controlla e gestisce l'ambiente e gli strumenti di lavoro del gruppo, con piena responsabilità sull'efficacia ed efficienza del Way of Working.
I suoi principali compiti sono:
- Migliorare l'ambiente di lavoro: ricercare gli strumenti necessari ad automatizzare i processi;
- Gestione dei processi: atta a risolverne i problemi legati ai processi;

- Redigere e mantenere la documentazione: gestisce il versionamento;
- Gestire la configurazione di prodotto: controllo sul prodotto software.

Analista
L'*Analista* approfondisce le richieste del Capitolato ed è presente principalmente nelle fasi iniziali del progetto. E' fondamentale che l'`Analisi dei Requisiti` sia adeguata: l'identificazione errata dei requisiti può compromettere in modo significato la fase di Progettazione e l'esito del progetto. Conosce meglio degli altri componenti il dominio del problema.
Ha il compito di:
- Studiare il problema e il relativo contesto applicativo;
- Raccogliere e studiare i bisogni dei committenti;
- Scrivere il documento `Analisi dei Requisiti`;
- Studiare i requisiti definendo la loro complessità.

Progettista
Il Progettista determina le scelte realizzative del progetto, trasformando i requisiti individuati dagli Analisti in un'architettura che modelli il problema. Il Progettista seguirà lo sviluppo particolarmente, ma non la manutenzione.
Ha il compito di:
- Sviluppare un prodotto economico, facilmente manutenibile a partire dal lavoro dell'analista;
- Favorire efficienza ed efficacia grazie alle scelte tecniche effettuate;
- Garantire un basso grado di accoppiamento grazie ad un sistema ben strutturato.

Verificatore
Il Verificatore controlla il lavoro svolto dagli altri componenti del gruppo, assicurandosi che le norme vengano attuate correttamente.
Ha il compito di:
- Verificare la correttezza delle attività tramite gli strumenti e tecniche definiti nelle `Norme di Progetto`;

Programmatore
Il Programmatore è incaricato di svolgere l'attività di codifica del progetto e delle componenti di supporto con lo scopo di realizzare l'architettura proposta dal progettista.
Ha il compito di:
- Implementare la Specifica Tecnica scritta dal Progettista;
- Scrivere codice mantenibile, che rispetti le `Norme di Progetto`;
- Creare test per la verifica e validazione del codice;
- Scrivere il *manuale utente*.

4.1.3.5) Ticketing

Il gruppo *Ouverture* adotta l'**Issue Tracking System** (ITS) interno di GitHub. GitHub permette una gestione semplice e chiara dei compiti da svolgere: le Issue vengono create molto velocemente e possono essere chiuse con altrettanta rapidità.
È compito del Responsabile creare i task ed assegnarli ai vari membri del gruppo, il cui stato di avanzamento è consultabile all'interno della *Board*.
Le Issue sono create dal Responsabile e sono composte da:
- **Titolo**: identifica in modo univoco il compito da svolgere;
- **Descrizione**: una lista dei nomi dei file coinvolti nel task;
- **Assegnatario**: il componente incaricato a svolgere il task;
- **Verificatore**: il componente incaricato ad accertare il corretto completamento del task;
- ***Milestone***: il traguardo da raggiungere;

- **Etichetta**: il tipo di task;
- **Stato**: avanzamento del task.

Ogni qualvolta ci sia la necessità di portare a termine un compito è necessario seguire la seguente procedura:
1. Il Responsabile crea una nuova Issue con stato “to do” su GitHub e la assegna;
2. All’inizio del lavoro di produzione la Issue cambia stato, passando da “to do” ad “in progress”, inoltre viene creato un nuovo branch per ogni Issue;
3. Finito il lavoro di produzione, viene aperta una pull request su GitHub, viene inserito nella descrizione il comando `closes #X`, dove X identifica univocamente la Issue e viene assegnato il Verificatore;
4. Il Verificatore si controlla il lavoro svolto e:
   - Se la verifica ha esito **positivo**:
     1. Il Verificatore conferma su GitHub la pull requet ed effettua il merge al branch principale;
     2. La Issue viene marcata “Done” su GitHub automaticamente.
   - Se la verifica ha esito **negativo**:
     1. Il Verificatore rilascia una lista di cambiamenti suggeriti nella relativa Issue su GitHub;
     2. L’incaricato apporta le modifiche suggerite e si torna al punto 3.

4.1.4) Coordinamento

Il Coordinamento è l’attività responsabile della gestione delle comunicazioni e degli incontri tra le diverse parti coinvolte nel progetto, ovvero membri del team, proponente e committenti. Il coordinamento assume un ruolo di rilievo nell’assicurare l’efficienza del progetto e il coinvolgimento di tutte le parti interessate.
Le attività di coordinamento comprendono la gestione della comunicazione interna ed esterna, la conduzione delle riunioni e la definizione di comportamenti comuni per i membri del team.
La comunicazione, cruciale per garantire chiarezza e concisione nel dialogo tra le parti coinvolte, si configura, seppur complessa, come un elemento essenziale per il successo del progetto.

4.1.4.1) Comunicazioni

Comunicazioni interne Le comunicazioni saranno gestite attraverso due canali principali: ***Telegram*** e **Discord**. Telegram, un servizio di messaggistica istantanea, sarà impiegato per facilitare conversazioni rapide, informali e accessibili tramite dispositivi mobili. Le riunioni a distanza e le discussioni più strutturate saranno invece condotte su Discord.
Al fine di coordinare in modo efficace le attività di gruppo, le discussioni di routine avverranno su Telegram, mentre eventuali questioni critiche saranno affrontate durante incontri straordinari attraverso videochiamate su Discord. In caso di inconvenienti tecnici con Telegram, il gruppo si trasferirà temporaneamente su Discord, comprese le conversazioni informali.
Il canale Discord includerà una sezione testuale informale dedicata alle comunicazioni rapide con il proponente e canali specifici saranno creati per tracciare le diverse conversazioni in base agli argomenti trattati.

Comunicazioni esterne Il Responsabile del progetto sarà incaricato di gestire il dialogo esterno attraverso l’indirizzo email: overture.unipd@gmail.com. Si assicurerà che ogni membro del gruppo sia informato sulle corrispondenze con committenti e proponente, seguendo le norme precedentemente stabilite per le comunicazioni interne.

4.1.4.2) Riunioni

Al fine di garantire l'efficienza delle riunioni il responsabile corrente avrà il compito di introdurre l'agenda e trattare in modo chiaro gli argomenti di discussione, inoltre sarà responsabile di riepilogare i punti principali e l'esito delle votazioni nel verbale successivo alla riunione.

Riunioni Interne:
Le riunioni interne sono programmate settimanalmente alle ore 16:00 di ogni mercoledì, concordate di comune accordo tra i membri del gruppo. In caso di necessità, è possibile richiedere riunioni straordinarie durante la settimana tramite il canale dedicato su Telegram, con data e orario stabiliti attraverso un sondaggio. Tutte le riunioni online si svolgeranno nel canale Discord appositamente designato.

Compiti del responsabile
- Esporre i punti all'ordine del giorno relativamente alla loro priorità;
- Aggiornare il resto del gruppo in caso di variazioni orarie;
- Pianificare le attività da svolgere;
- Assegnare i task ai membri del gruppo;
- Approvare il verbale.

Doveri partecipanti
- Essere presenti e puntuali alle riunioni settimanali;
- Mantenere un comportamento consono durante lo svolgimento della riunione.

Riunioni Esterne:
Le riunioni esterne coinvolgono i membri del gruppo *Overture*, il proponente e i committenti.
Per le riunioni con il proponente, viene utilizzata la piattaforma Carbonio, e l'indirizzo viene comunicato al team di volta in volta. I membri del gruppo si impegnano a partecipare costantemente, cercando di adattare i propri impegni per garantire la presenza a tali incontri. Nel caso in cui gli impegni irrinunciabili dei membri rendano impossibile la partecipazione, il responsabile si assicurerà di informare tempestivamente il proponente o i committenti, proponendo di posticipare la riunione a una data successiva.

4.1.4.3) Verbali

Verbali Interni:
L'obiettivo di una sessione di incontri è affrontare e risolvere gli argomenti specificati nell'ordine del giorno. Al termine di ogni incontro, viene aperta una Issue su Github per la preparazione, la verifica e l'approvazione del verbale. Il compito di redigere il verbale, seguendo il formato indicato nella sezione 3.1 di questo documento, è affidato al Responsabile, il quale deve assicurarsi di includere tutte le informazioni rilevanti discusse.

Verbali Esterni
Come per il caso delle riunioni interne verrà redatto un Verbale con le stesse modalità descritte in precedenza.

4.2) Miglioramento

4.2.1) Scopo

Il miglioramento rappresenta un procedimento volto a istituire, valutare, misurare, controllare e ottimizzare il ciclo di vita del software. In questo processo si va ad adottare un approccio ciclico, ovvero in cui le fasi vengono continuamente rivisitate e migliorate per garantire che il prodotto software risponda alle aspettative e mantenga elevati *standard di qualità* e efficienza.

4.2.2) Descrizione

Questo processo organizzativo è costituito da tre attività:

- Stabilimento dei processi;
- Valutazione dei processi;
- Miglioramento dei processi.

4.2.2.1) Stabilimento dei processi

Innanzitutto occorre stabilire una serie di processi organizzativi per l'intero ciclo di vita del software applicabili alle varie attività di progetto. Quest'ultimi devono essere documentati, come viene fatto proprio qui, e va implementato un meccanismo di controllo per sviluppare, monitorare e migliorare i processi stessi.

4.2.2.2) Valutazione dei processi

Come anticipato prima occorre sviluppare, documentare e applicare una procedura di valutazione del processo. Questa viene eseguita basandosi sugli obiettivi del processo, sulle metriche adottate e sull'analisi dei dati raccolti per poter poi proporre delle migliorie da applicare. Pianificare ed effettuare revisioni dei processi a intervalli appropriati garantisce la loro continua idoneità ed efficacia.

4.2.2.3) Miglioramento dei processi

Una volta identificati i potenziali miglioramenti questi vanno effettivamente implementati. La documentazione del processo di conseguenza deve essere aggiornata a riflettere il miglioramento dei processi organizzativi ed inoltre i dati storici, tecnici e di valutazione devono essere raccolti e analizzati per avere un avanzamento continuo e non regredire.

4.2.3) Metriche

La misurazione del miglioramento nello sviluppo del software può coinvolgere diverse metriche che riflettono aspetti chiave del processo stesso. Generalmente andiamo a valutare le seguenti:
- Velocità di sviluppo;
- Tasso di errori;
- Conformità agli standard.

4.2.4) Strumenti

I seguenti strumenti sono stati scelti dal gruppo per il miglioramento:
- **GitHub**: per valutare tramite issues o project board fattori come la velocità di sviluppo, ed inoltre per automatizzare molte attività ripetitive nel processo di sviluppo, come la build degli artefatti, migliorando l'efficienza complessiva.

4.3) Formazione

4.3.1) Scopo

Dato che alla base di un progetto di successo troviamo un personale esperto e qualificato, per fare in modo che tutti i componenti del gruppo siano preparati ed aggiornati nel ricoprire i ruoli di progetto è fondamentale il processo di formazione.
In esso definiamo come approfondire i temi necessari per il lavoro che andiamo a svolgere, al fine di essere tempestivi ed efficaci nell'apprendimento.

4.3.2) Descrizione

Per poter formare i membri del gruppo è necessario innanzitutto comprendere a pieno il dominio del problema. Occorre quindi capire quali sono i temi da approfondire ed identificare nei vari processi da svolgere quali nozioni è necessario apprendere.
Fatto ciò si deve passare all'individuazione del materiale di formazione, il quale crescerà nel tempo,

dato che man mano che il progetto avanza anche il nostro grado di conoscenza e comprensione del problema dovrà aumentare il più possibile.
Infine, una volta capito cosa e da dove studiare, è necessario che ognuno dei componenti del gruppo vada ad aggiornarsi individualmente oppure, quando possibile, tramite supporto di altri membri più esperti.

4.3.3) Aspettative

È previsto che ciascun membro del gruppo acquisisca le competenze fondamentali per lo svolgimento del progetto attraverso un adeguato percorso formativo. Questo non riguarda soltanto le tecnologie necessarie, ma si estende anche a ogni altro aspetto coinvolto nell'ingegneria del software.

4.3.4) Strumenti

Al fine di agevolare il processo di formazione, abbiamo optato per l'adozione di un repository privato, accessibile esclusivamente ai membri del gruppo, nel quale tutte le risorse utili sono organizzate in categorie per favorire un accesso efficiente e ordinato.

5) Standard ISO/IEC 9126 per la qualità

La norma ISO/IEC 9126 è uno standard internazionale che ha contribuito a definire i parametri essenziali per valutare la qualità del software. Questa norma rappresenta un insieme di linee guida dettagliate e criteri di valutazione per gli attributi chiave della qualità del software. In particolare, identifica sei macro-categorie di attributi di qualità del software, ognuna delle quali è ulteriormente scomposta in sottoattributi specifici:
- **Funzionalità**;
- **Affidabilità**;
- **Usabilità**;
- **Efficienza**;
- **Manutenibilità**;
- **Portabilità**.

5.1) Funzionalitá

La funzionalità si concentra sulla valutazione della capacità del software di fornire funzioni che soddisfano i requisiti specificati e impliciti.
I suoi sottoattributi sono:
- **Adeguatezza**: é la capacità del software di fornire un insieme di funzionalità che soddisfano i requisiti specifici dell'utente e del contesto d'uso;
- **Accuratezza**: é la capacità del software di fornire risultati corretti con il grado di precisione richiesto;
- **Interoperabilità**: é la capacità del software di cooperare ed interagire efficacemente con altri sistemi;
- **Sicurezza**: é la capacitá del software di proteggere i dati dagli accessi non autorizzati;
- **Conformità**: é la conformità del software agli standard, alle norme e alle specifiche funzionali pertinenti.

5.2) Affidabilitá

L'affidabilità si concentra sulla valutazione della capacità del software di mantenere un adeguato livello di prestazioni anche in presenza di errori o malfunzionamenti. I suoi sottoattributi sono:
- **Maturitá**: é la capacitá del software di mantenere una stabilitá durante l'esecuzione evitando errori e risultati non corretti;
- **Tolleranza agli Errori**: é la capacità del software di gestire e tollerare errori;

- **Capacità di Recupero**: é la capacità del software di ripristinare le prestazioni desiderate dopo che si è verificato un errore;
- **Affidabilità delle Informazioni**: é la precisione e l'affidabilità delle informazioni fornite dal software.

5.3) Usabilitá

L'usabilità è un aspetto critico nella valutazione della qualità del software, poiché influenza direttamente l'esperienza dell'utente durante l'interazione con il sistema.
I suoi sottoattributi sono:

- **Comprensibilità**: é la facilità con cui l'utente può comprendere l'interfaccia utente, le informazioni presentate e il modo in cui eseguire le operazioni;
- **Apprendibilità**: é la facilità con cui nuovi utenti possono imparare ad utilizzare il software, attraverso una curva di apprendimento rapida e un accesso chiaro alle funzionalità;
- **Operabilità**: é la facilità con cui gli utenti possono eseguire le operazioni desiderate in modo efficiente, senza errori e con un utilizzo ottimale delle risorse;
- **Attrattività**: é la piacevolezza estetica dell'interfaccia utente e del design complessivo del software;
- **Aderenza all'usabilità**: si riferisce alla misura in cui il software rispetta le linee guida e le convenzioni dell'usabilità.

5.4) Efficienza

L'efficienza si concentra sulla capacità del software di utilizzare in modo ottimale le risorse a sua disposizione, garantendo prestazioni elevate e tempi di risposta rapidi.
I suoi sottoattributi sono:

- **Utilizzo delle Risorse**: misura quanto efficientemente il software sfrutti la capacità delle risorse di sistema durante l'esecuzione;
- **Aderenza all'Efficienza**: indica quanto il software rispetti le linee guida e le migliori pratiche per l'ottimizzazione del codice e delle risorse;
- **Comportamento rispetto al Tempo**: Rappresenta la capacità del software di rispondere rapidamente alle richieste degli utenti e di adattarsi dinamicamente alle variazioni di carico.

5.5) Manutenibilità

La manutenibilità si concentra sulla facilità con cui il software può essere modificato, corretto, adattato e migliorato nel tempo.
I suoi sottoattributi sono:

- **Analizzabilità**: é la facilità con cui è possibile analizzare il codice sorgente per identificare errori, problemi di progettazione o aree di miglioramento;
- **Modificabilità**: é la facilità con cui il software può essere modificato senza causare effetti indesiderati o introduzione di nuovi errori;
- **Stabilità**: é la capacità del software di evitare effetti collaterali indesiderati durante le modifiche o l'introduzione di nuove funzionalità;
- **Testabilità**: é la facilità con cui è possibile testare il software per garantire che le modifiche apportate non abbiano effetti indesiderati su funzionalità esistenti;
- **Aderenza alla manutenibilità**: si riferisce alla misura in cui il software rispetta le linee guida e le convenzioni della manutenibilità.

5.6) Portabilità

La portabilità si riferisce alla facilità con cui il software può essere trasferito da un ambiente a un altro senza dover apportare modifiche sostanziali.
I suoi sottoattributi sono:
- **Adattabilità**: é la capacità del software di adattarsi a diversi ambienti senza richiedere modifiche sostanziali al codice sorgente o all'architettura;
- **Installabilità**: é la facilità con cui il software può essere installato in un nuovo ambiente;
- **Sostituibilità**: é la facilità con cui il software può sostituire o essere sostituito da altre applicazioni nello stesso ambiente;
- **Coesistenza**: é la capacità del software di operare in modo efficace e senza conflitti all'interno di un ambiente condiviso con altre applicazioni;
- **Aderenza alla portabilità**: si riferisce alla misura in cui il software rispetta le linee guida e le convenzioni della portabilità.

6) Metriche di qualitá del processo

L'acronimo principale utilizzato in questa sezione è MPC da Metriche di qualità del ProCesso.

6.1) Processi primari

6.1.1) Fornitura

- **MPC01 - Earned value (EV)**: misura il valore del lavoro effettivamente completato rispetto al valore pianificato;
- **MPC02 - Planned value(PV)**: rappresenta il valore pianificato del lavoro da completare in un determinato momento;
- **MPC03 - Actual cost (AC)**: indica il costo effettivamente sostenuto per completare il lavoro fino a un certo momento;
- **MPC04 - Cost variance (CV)**: misura la differenza tra il valore guadagnato e il costo effettivamente sostenuto. Se ha valore negativo ciò significa che si è fuori budget.

$$\mathrm{CV} = \mathrm{EV} - \mathrm{AC}$$

- **MPC05 - Schedule variance (SV)**: indica la differenza tra il valore guadagnato e il valore pianificato. Se ha valore negativo ciò significa che si è in ritardo rispetto alle previsioni.

$$\mathrm{SV} = \mathrm{EV} - \mathrm{PV}$$

- **MPC06 - Estimated at completion (EAC)**: fornisce una stima del costo totale del progetto basato sul rendimento attuale;

$$\mathrm{EAC} = \mathrm{AC} + \mathrm{ETC}$$

- **MPC07 - Estimate to complete (ETC)**: indica la stima dei costi rimanenti per completare il progetto;

$$\mathrm{ETC} = \mathrm{BAC} - \mathrm{EV}$$

  - **BAC**: rappresenta il *Budget At Completion*

6.1.2) Sviluppo

- **MPC08 - Requirements stability index (RSI)**: valuta la stabilità dei requisiti nel corso del tempo:

$$\mathrm{RSI} = 100 - \left(\frac{\mathrm{RA}+\mathrm{RC}+\mathrm{RR}}{\mathrm{TR}}\right) * 100$$

  - **RA**: rappresenta il numero di requisiti aggiunti nel periodo considerato;
  - **RC**: rappresenta il numero di requisiti cambiati nel periodo considerato;
  - **RR**: rappresenta il numero di requisiti rimossi nel periodo considerato;
  - **TR**: rappresenta il numero totale di requisiti al momento dell'analisi.

- **MPC09 - Structural Fan-In (SFIN)**: indice di utilità che esprime la quantità di componenti che sfruttano un modulo specifico;
- **MPC10 - Structural Fan-Out (SFOUT)**: indice di dipendenza che rappresenta il numero di componenti utilizzate dal modulo preso in considerazione;

6.2) Processi di supporto

6.2.1) Documentazione

- **MPC11 - Indice Gulpease**: valuta la leggibilità di un documento in lingua italiana:

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

  - $\mathbf{N_f}$: numero di frasi;
  - $\mathbf{N_l}$: numero di lettere;
  - $\mathbf{N_p}$: numero di parole.
- **MPC12 - Correttezza ortografica**: misura il numero di errori ortografici in un documento.

6.2.2) Verifica

- **MPC13 - Code coverage**: indica la percentuale di codice sorgente testato rispetto al totale;
- **MPC14 - Passed test cases percentage**: valuta la percentuale di casi di test superati con successo.

6.2.3) Gestione della qualitá

- **MPC15 - Quality metrics satisfied**: monitora il grado di soddisfacimento delle metriche di qualità stabilite.

6.3) Processi organizzativi

6.3.1) Gestione dei processi

- **MPC16 - Non-calculated risk**: monitora i rischi non inclusi nelle stime e nelle previsioni;
- **MPC17 - Efficienza temporale**: valuta quanto tempo viene effettivamente impiegato in attività produttive, ovvero che portano al raggiungimento di obiettivi, rispetto al tempo totale trascorso, misurato in ore di orologio:

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

  - $\mathbf{O_o}$: ore di orologio;
  - $\mathbf{O_p}$: ore produttive.

7) Metriche di qualitá del prodotto

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

7.1) Funzionalitá

- **MPD01 - Copertura dei requisiti obbligatori**: misura la percentuale di requisiti obbligatori coperti:

$$\mathrm{CRO} = \frac{\mathrm{ROC}}{\mathrm{TRO}} * 100$$

  - **ROC**: numero di requisiti obbligatori coperti dall'implementazione;
  - **TRO**: numero totale di requisiti obbligatori.
- **MPD02 - Copertura dei requisiti desiderabili**: valuta la percentuale di requisiti desiderabili coperti:

$$\mathrm{CRD} = \frac{\mathrm{RDC}}{\mathrm{TRD}} * 100$$

  - **RDC**: numero di requisiti desiderabili coperti dall'implementazione;

  - **TRD**: numero totale di requisiti desiderabili.
- **MPD03 - Copertura dei requisiti opzionali**: quantifica la percentuale di requisiti opzionali coperti:

$$\text{CRP} = \frac{\text{RPC}}{\text{TRP}} * 100$$

  - **RDC**: numero di requisiti opzionali coperti dall'implementazione;
  - **TRD**: numero totale di requisiti opzionali.

7.2) Affidabilitá

- **MPD04 - Code coverage**: rappresenta la percentuale di codice sorgente eseguito durante i test rispetto al totale;
- **MPD05 - Branch coverage**: misura la percentuale di rami decisionali del codice coperti durante i test;
- **MPD06 - Statement coverage**: quantifica la percentuale di istruzioni del codice sorgente eseguite durante i test;
- **MPD07 - Failure density**: calcola il numero di fallimenti correttamente riscontrati durante i test per unità di dimensione del codice.

7.3) Usabilitá

- **MPD08 - Facilitá di utilizzo**: valuta la facilità con cui gli utenti possono interagire con il sistema;
- **MPD09 - Tempo di apprendimento**: valuta quanto tempo gli utenti impiegano per imparare ad utilizzare efficacemente il server mail e tutte le sue funzionalità.

7.4) Efficienza

- **MPD10 - Utilizzo risorse**: misura l'efficienza del sistema in termini di utilizzo delle risorse.

7.5) Manutenibilitá

- **MPD11 - Complessità ciclomatica**: valuta la complessità del codice sorgente attraverso la misurazione del numero di cammini indipendenti attraverso il grafo di controllo di flusso;
- **MPD12 - Code smell**: rileva potenziali problemi di progettazione o codice che potrebbe richiedere manutenzione;
- **MPD13 - Coefficient of Coupling (COC)**: fornisce una valutazione quantitativa del grado di dipendenza tra i moduli o le componenti di un sistema:

$$\text{COC} = \frac{\text{NDIP}}{\text{NMOD}} * 100$$

  - **NDIP**: numero totale di dipendenze tra moduli;
  - **NMOD**: numero totale di moduli nel sistema.