Norme di Progetto
Progetto ShopChain

yakuzaishi.swe@gmail.com

Informazioni sul documento

Responsabile	Dario Furlan
Redattori	Luca Busacca
	Francesco Mattarello
Verificatori	Francesco Bugno
	Luca Carturan
Uso	Interno
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Ing. Fabio Pallaro - Sync Lab S.r.l.
Versione	3.0.0

Sommario

Questo documento raccoglie le regole, gli strumenti e le convenzioni adottate dal gruppo *Yakuzaishi* definendone il *way of working*.

Registro delle Modifiche

Versione	Data	Autore	Ruolo	Descrizione
3.0.0	2022/06/08	Dario Furlan	Responsabile	Approvato per il rilascio
2.1.0	2022/06/05	Francesco Bugno	Verificatore	Verifica generale del documento
2.0.1	2022/06/02	Francesco Mattarello Luca Carturan	Amministratore Verificatore	Aggiornata struttura documento post PB e verifica
2.0.0	2022/05/06	Dario Furlan	Responsabile	Approvato per il rilascio
1.2.1	2022/04/21	Luca Carturan	Verificatore	Verifica generale del documento
1.1.1	2022/04/20	Francesco Bugno Luca Carturan	Amministratore Verificatore	Aggiornamento §C.2 e verifica
1.1.0	2022/03/29	Francesco Bugno	Verificatore	Verifica generale del documento
1.0.4	2022/03/28	Luca Busacca Luca Carturan	Amministratore Verificatore	Aggiornata struttura documento §4 e verifica
1.0.3	2022/03/22	Francesco Mattarello Francesco Bugno	Amministratore Verificatore	Aggiornata struttura documento §3 e verifica
1.0.2	2022/03/20	Luca Busacca Francesco Bugno	Amministratore Verificatore	Aggiornata struttura documento §2 e verifica
1.0.1	2022/03/17	Francesco Mattarello Luca Carturan	Amministratore Verificatore	Aggiornamento §3.1.4.3.2, §1.1 e verifica
1.0.0	2022/01/21	Dario Furlan	Responsabile	Approvato per il rilascio
0.4.0	2022/01/19	Luca Carturan	Verificatore	Verifica generale del documento
0.3.5	2022/01/15	Luca Busacca	Amministratore	Aggiunta §C.2
0.3.4	2022/01/10	Francesco Mattarello	Amministratore	Correzione errori ortografici, inizio stesura §C
0.3.3	2022/01/07	Francesco Mattarello	Amministratore	Continuo stesura §A, inserimento §A.2 e §A.3
0.3.2	2022/01/05	Luca Busacca	Amministratore	Stesura §B
0.3.1	2022/01/03	Francesco Mattarello	Amministratore	Aggiornamento §4, inizio stesura §A

Versione	Data	Autore	Ruolo	Descrizione
0.3.0	2021/12/30	Luca Carturan	Verificatore	Verifica generale del documento
0.2.2	2021/12/21	Luca Busacca	Amministratore	Stesura §4.2
0.2.1	2021/12/18	Luca Busacca	Amministratore	Sistemata §3.4
0.2.0	2021/12/17	Francesco Bugno	Verificatore	Verifica generale del documento
0.1.6	2021/12/15	Luca Busacca	Amministratore	Stesura §3.4
0.1.5	2021/12/14	Francesco Mattarello	Amministratore	Aggiustamento §3: modificate immagine e tabella, correzione errori ortografici
0.1.4	2021/12/12	Francesco Mattarello	Amministratore	Stesura §3.3 e §3.5
0.1.3	2021/12/11	Luca Busacca	Amministratore	Stesura §3.2
0.1.2	2021/12/08	Francesco Mattarello	Amministratore	Inizio stesura §4
0.1.1	2021/12/07	Luca Busacca	Amministratore	Modificata immagine caso d'uso, aggiornamento §2.2.4.1.1
0.1.0	2021/12/06	Luca Carturan	Verificatore	Verifica generale del documento
0.0.7	2021/12/05	Luca Busacca	Amministratore	Stesura da §2.2.4.1.1 a §2.2.4.1.2
0.0.6	2021/12/02	Francesco Mattarello	Amministratore	Stesura da §2.2.4.2 a §2.2.4.2.4
0.0.5	2021/11/28	Francesco Mattarello	Amministratore	Inizio stesura §3
0.0.4	2021/11/27	Luca Busacca	Amministratore	Stesura da §2.2 a §2.2.4.1.1
0.0.3	2021/11/25	Francesco Mattarello	Amministratore	Stesura da §2.1 a §2.1.6
0.0.2	2021/11/22	Francesco Mattarello	Amministratore	Stesura §1
0.0.1	2021/11/04	Luca Busacca	Amministratore	Creata struttura documento

Contenuti

Elenco delle tabelle

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Il fine del seguente documento è quello di normare le regole e le procedure fondamentali che ciascun membro del gruppo *Yakuzaishi* si impegna a visionare e rispettare. Allo stato attuale il documento risulta incompleto, dato che il team di sviluppo nella redazione dei vari documenti ha deciso di intraprendere un approccio di tipo incrementale, aggiornando il documento di volta in volta al seguito di ogni decisione presa dal gruppo. Le norme già presenti possono subire cambiamenti (aggiunte, rimozioni, modifiche), e dovranno essere comunicate dal *responsabile* di progetto a ciascun membro del gruppo. A seguito delle considerazioni fatte dal prof. Vardanega durante la revisione *Requirements and Technology Baseline*, il gruppo *Yakuzaishi*, dopo una riunione interna, ha deciso di passare da un modello incrementale ad un modello agile, avvalendosi di alcuni elementi caratteristici del framework Scrum$_{\mathbf{G}}$. I dettagli implementativi sono descritti nel *Piano di progetto v3.0.0*.

1.2 Scopo del capitolato

L'avvento delle tecnologie blockchain$_{\mathbf{G}}$ ha portato e porterà nei prossimi anni a grandi cambiamenti nella società. In particolare, ha aperto le porte a una nuova forma di finanza, la cosiddetta "DeFi" (Finanza Decentralizzata) che ha permesso a chiunque sia dotato di connessione internet di creare un wallet$_{\mathbf{G}}$ e possedere quindi criptovalute$_{\mathbf{G}}$. Questo ha delineato due profili critici strettamente legati; da un lato il controllo del proprio portafoglio è passato completamente nelle mani dell'utente, dall'altro ha comportato la mancanza di un ente terzo che si occupi di gestire transazioni e offrire garanzie.
Il capitolato in questione si pone quindi il seguente obiettivo: in un e-commerce$_{\mathbf{G}}$ basato su blockchain$_{\mathbf{G}}$, dove il pagamento avviene tramite cryptovalute$_{\mathbf{G}}$, si vuole tutelare le parti coinvolte nell'acquisto.
Il fine del progetto è la realizzazione di un prototipo di una piattaforma integrabile con un "crypto-ecommerce$_{\mathbf{G}}$", che si occupi di gestire gli ordini dalla fase di pagamento fino alla consegna del prodotto.

1.3 *Glossario*

I termini utilizzati in questo documento potrebbero generare dubbi riguardo al loro significato, richiedendo pertanto una definizione al fine di evitare ambiguità.
Tali termini vengono contrassegnati da una G maiuscola finale a pedice della parola.
La loro spiegazione è riportata nel *Glossario v2.0.0*.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Regolamento del progetto didattico:**

  https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/PD2.pdf

- **Capitolato d'appalto C2 - ShopChain:**

  https://www.math.unipd.it/~tullio/IS-1/2021/Progetto/C2.pdf

1.4.2 Riferimenti informativi

- **Corso sulla blockchain$_G$ tenuto alla laurea Magistrale in informatica dell' Università di Verona:**

  https://www.di.univr.it/?ent=avviso&dest=&id=157271

- **Software Engineering - Ian Sommerville - 10 th Edition:**
  Parte 4 - Software management:
  - Capitolo 25 - Configuration management:
    - Paragrafo 25.1 - Version management (da pag. 735 a 740);
    - Paragrafo 25.2 - System building (da pag. 740 a 745).

- **Guida LaTeX:**

  http://www.lorenzopantieri.net/LaTeX_files/LaTeXimpaziente.pdf

- **Standard ISO/IEC 12207:**

  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  - Capitolo 5 - Primary life cycle processes (da pag. 10 a 24);
  - Capitolo 6 - Supporting life cycle processes (da pag. 28 a 41);
  - Capitolo 7 - Organizational life cycle processes (da pag. 42 a 47).

- **Standard ISO/IEC 9126:**

  http://www.colonese.it/00-Manuali_Pubblicatii/07-ISO-IEC9126_v2.pdf
  - Capitolo 2 - Il modello ISO/IEC 9126 (da pag. 12 a 24);
  - Capitolo 4 - Utilizzo del modello:
    - Paragrafo 4.1 - Processo di sviluppo e qualità (da pag. 31 a 34);
    - Paragrafo 4.2 - Implementazione del modello in progetti reali (da pag. 34 a 37);
    - Paragrafo 4.3 - Esempi di metriche (pag. 37).

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

In questa sezione vengono elencate tutte le metriche, gli strumenti e i documenti utilizzati al fine di realizzare il processo di fornitura.

2.1.2 Aspettative

Le aspettative nell'applicazione del processo di fornitura sono:

- Avere una chiara struttura dei documenti;
- Definire i tempi di lavoro;
- Chiarire dubbi e stabilire vincoli con il proponente.

2.1.3 Descrizione

Il processo di fornitura determina ogni compito, attività e risorsa necessaria allo svolgimento del progetto. Tale processo verrà avviato solamente in seguito alla comprensione delle richieste del proponente, seguito da uno studio di fattibilià delle richieste e concluso dalla definizione di un accordo contrattuale con il proponente. Il processo di fornitura consiste nelle seguenti fasi:

- Avvio;
- Contrattazione;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Consegna e completamento.

2.1.4 Istanziazione del processo

2.1.4.1 Inizializzazione

Gli analisti devono svolgere uno studio sui vari capitolati proposti, indicando per ognuno di essi gli aspetti positivi e negativi, infine tramite un' apposita motivazione devono comunicare il capitolato scelto.

2.1.4.2 Preparazione alla risposta

Il gruppo *Yakuzaishi* deve redigere una lettera di presentazione dove dichiara l'impegno nella realizzazione del prodotto proposto dal capitolato di interesse.

2.1.4.3 Rilascio del prodotto

Durante questa fase il gruppo *Yakuzaishi* deve rilasciare un prodotto che soddisfi le aspettative del proponente e dei committenti, presentando a questi ultimi:

- Codice soregente del prodotto;
- Documentazione, alla quale in questa fase si aggiungono:
  - Manuale utente;
  - Manuale sviluppatore.

2.1.4.4 Pianificazione

Il gruppo *Yakuzaishi* deve redigere un piano per la gestione del progetto e della qualità. In §3.1.4.2 vengono descritti i documenti di interesse.

2.1.4.5 Esecuzione e controllo

Il gruppo deve seguire ed implementare il *piano di progetto* e controllare l'avanzamento, i costi e la qualità dei prodotti durante tutte le fasi del loro ciclo di vita.

2.1.4.6 Revisione e valutazione

Il gruppo *Yakuzaishi* deve organizzare le attività di revisione, verifica e validazione del prodotto svolto, garantendo che quest'ultimo soddisfi le aspettative del proponente.

2.1.4.7 Proponente

Il team di sviluppo si è accordato con il proponente per avere un continuo contatto sul buon proseguo del progetto, organizzando periodicamente incontri e mantenendo un costante contatto asincrono mediante un apposito canale sulla piattaforma Discord$_{\mathbf{G}}$. Il team di sviluppo vuole avere un costante contatto con il proponente riguardo alle seguenti tematiche:

- Chiarimento su eventuali dubbi;
- Riscontro sulla documentazione redatta;
- Riscontro sulle tecnologie utilizzate;
- Opinione su soluzioni innovative proposte dal team di sviluppo;
- Tempistiche di consegna del prodotto;
- Stima dei costi;
- Vincoli e requisiti obbligatori.

2.1.4.8 Documenti

Di seguito sono descritti i documenti redatti in questa fase.

Piano di qualifica

Il *piano di qualifica* viene redatto dal *verificatore* e contiene le misure necessarie a garantire la qualità del prodotto e dei processi. Esso è formato dalle seguenti parti:

- Qualità di processo;
- Qualità di prodotto;
- Specifica dei test;
- Resoconto attività di verifica.

Piano di progetto

Il *piano di progetto* viene redatto dagli amministratori e dal *responsabile* di progetto, esso verrà seguito per tutta la durata del progetto, ed è diviso nelle seguenti parti:

- Analisi dei rischi;
- Modello di sviluppo;
- Pianificazione;
- Preventivo;
- Consuntivo di periodo;
- Attualizzazione dei rischi.

2.1.5 Metriche

Il processo di fornitura non fa uso di metriche particolari.

2.1.6 Strumenti

Di seguito sono riportati gli strumenti utilizzati dal team di sviluppo per la realizzazione del processo di fornitura:

- **Microsoft Excel**: Utilizzato per creare grafici, eseguire calcoli e presentare tabelle organizzative;

  https://www.microsoft.com/it-it/microsoft-365/excel

- **Microsoft Project**: Utilizzato per creare i grafici di Gantt$_G$ relativi alla pianificazione.

  https://www.microsoft.com/it-it/microsoft-365/project/project-management-software

- **Projects di Github**: utilizzato per gestire le issues$_G$ che ciascun membro del gruppo deve svolgere. Permette di assegnare attivittà a specifici membri e suddividerle per categorie in modo da garantire un buon livello di organizzazione.

  https://github.com/

2.2 Sviluppo

2.2.1 Scopo

Scopo del processo di sviluppo è quello di descrivere i compiti e le attività da svolgere relative allo sviluppo del prodotto software. In questa sezione vengono dunque descritte le attività, le norme e le convenzioni adottate per la composizione di tale processo.

2.2.2 Aspettative

Le aspettative nell'applicazione del processo di sviluppo sono:

- Determinare vincoli tecnologici;
- Determinare gli obiettivi di sviluppo;
- Determinare vincoli di design;
- Realizzare un prodotto finale che superi i test e soddisfi requisiti e richieste del proponente.

2.2.3 Descrizione

Il processo di sviluppo contiene le attività e i compiti dello sviluppatore, tra cui le attività per l'analisi dei requisiti, la progettazione, la codifica, l'integrazione, il test, l'installazione e l'accettazione relative ai prodotti software.

2.2.4 Istanziazione del processo

2.2.4.1 Attività

Di seguito sono elencate e successivamente trattate le attività caratterizzanti tale processo:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.4.1.1 *Analisi dei requisiti*

Scopo

Lo scopo dell'*analisi dei requisiti* è quello di individuare, a partire da un approfondito studio del capitolato, i requisiti diretti e indiretti, impliciti ed espliciti che il proponente richiede per la realizzazione del prodotto e i vari casi d'uso del prodotto stesso. In questa attività è importante suddividere il problema iniziale in requisiti quanto più elementari possibile, così da diminuire la complessità del problema originale facilitando il lavoro durante la fase di sviluppo.

Aspettative

L'obiettivo dell'attività di *analisi dei requisiti* consiste nella creazione della documentazione formale contenente tutti i requisiti richiesti dal proponente, che siano essi stati definiti all'inizio o che siano stati concordati in fase di sviluppo.

Requisiti

I requisiti vengono raccolti da diverse fonti quali:

- Lettura investigativa del capitolato;
- Confronto interno tra i membri del gruppo;
- Confronto con il proponente;
- Analisi dei casi d'uso.

Classificazione dei requisiti

Ciascun requisito viene classificato mediante l'uso di un codice identificativo. Per tale codice è stato definito uno standard dal gruppo che viene spiegato nel seguito.

Codice Identificativo:

R[Importanza][Tipologia][Codice]

dove:

Importanza: indica l'importanza associata a un requisito e può assumere un valore numerico compreso tra 1 e 3 con il seguente significato:

Numero	Descrizione
1	Requisito obbligatorio
2	Requisito desiderabile ma non obbligatorio
3	Requisito opzionale

Tabella 2: Importanza dei requisiti

Tipologia: rappresenta una tipologia di requisito e può assumere uno fra i seguenti valori letterali:

Sigla	Descrizione
V	Vincolo
P	Prestazionale
Q	Qualitativo
F	Funzionale

Tabella 3: Tipologie di requisiti

Codice: è un identificativo univoco del requisito espresso in forma gerarchica padre/figlio.

Una volta associato a un requisito, l'identificativo è immutabile. Ogni requisito deve inoltre essere accompagnato da una serie di informazioni aggiuntive:

- **Descrizione**: Descrizione sintetica ed esplicativa del requisito in questione;
- **Classificazione**: Al fine di rendere la tabella più leggibile viene rimarcata ancora una volta l'importanza del requisito anche se già presente nel codice identificativo;
- **Fonte**: L'origine del requisito viene riportata in questa sezione.

Requisito	Descrizione	Classificazione	Fonte
R1V1	Creazione smart contract$_G$	Obbligatorio	Capitolato C2

Tabella 4: Classificazione di un requisito

Casi d'uso

I casi d'uso esprimono un comportamento o un modo di utilizzare il prodotto.
Vengono descritti graficamente mediante l'ausilio di diagrammi UML$_G$.
Ciascun caso d'uso è costituito da:

- Codice identificativo;
- Attore Primario;
- Precondizioni;
- Postcondizioni;
- Scenario principale;
- Generalizzazioni;
- Estensioni.

Classificazione dei casi d'uso

Ciascun caso d'uso viene classificato mediante l'uso di un codice identificativo. Per tale codice è stato definito uno standard dal gruppo che viene spiegato nel seguito.

Codice Identificativo:

UC[Numero caso d'uso](.[Caso d'uso figlio])* - [Titolo caso d'uso]

dove:

UC: Acronimo di "use case";

Numero caso d'uso: Numero associato al caso d'uso principale;

Caso d'uso figlio: Numero associato al sottocaso del caso d'uso principale;

Titolo caso d'uso: Titolo assegnato al caso d'uso.

Una volta associato a un caso d'uso, l'identificativo è immutabile.

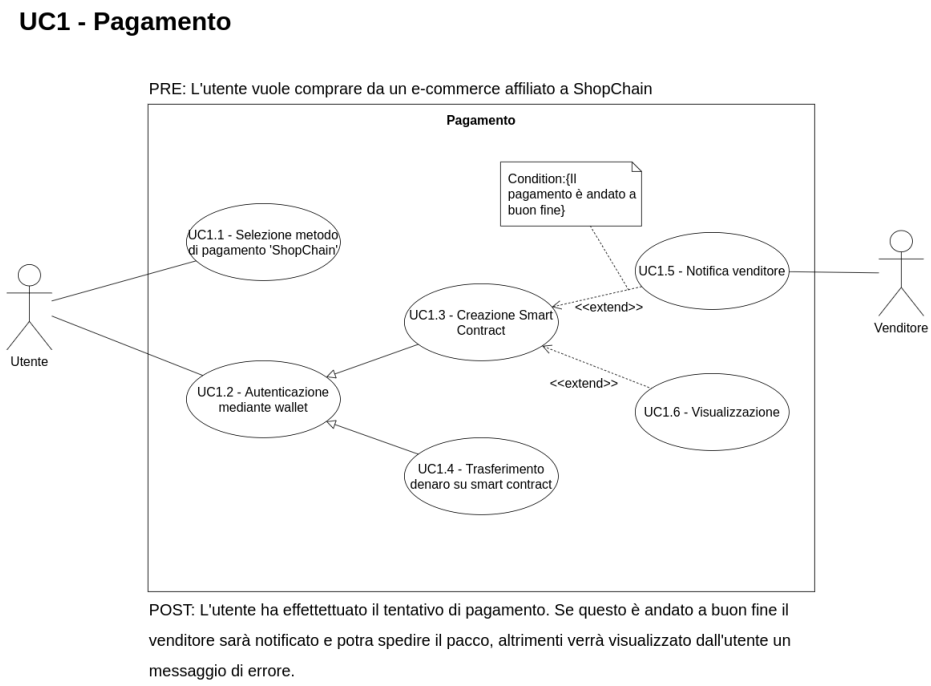

Figura 1: Esempio di caso d'uso

2.2.4.1.2 UML

I diagrammi UML$_{\mathbf{G}}$ verranno realizzati usando la versione 2.0 del linguaggio.

2.2.4.2 Progettazione

2.2.4.2.1 Scopo

Lo scopo dell'attività di progettazione è quello di incorporare le parti ottenute durante l'*analisi dei requisiti*, specificando le funzionalità dei sottosistemi e riconducendo a un'unica soluzione.

2.2.4.2.2 Aspettative

L'obiettivo dell'attività di progettazione consiste nel realizzare l'architettura del sistema, inizialmente realizzata da un Proof of Concept$_{\mathbf{G}}$ sviluppato come una demo prototipale del sistema alla Requirements & Technology Baseline, e inseguito approfondita e descritta nel documento tecnico allegato alla Product Baseline.

2.2.4.2.3 Requirements & Technology Baseline

Questa fase fissa i requisiti che il fornitore si impegna a soddisfare, in accordo con il proponente e vengono motivate le tecnologie, i framework$_{\mathbf{G}}$ e le librerie selezionate per la realizzazione del prodotto. Dimostra adeguatezza e fattibilità coerente con gli obbiettivi.
Documenti utili a tali fini sono:

- *Piano di progetto*;
- *Piano di qualifica*;
- *Norme di progetto*;

- *Verbali* interni ed esterni.

Salvo i *verbali*, i documenti sopracitati vanno mantenuti ed evoluti anche per le fasi successive.

2.2.4.2.4 Product Baseline

Questa fase illustra le linee guida architetturali del prodotto. Oltre all'evoluzione dei documenti sopracitati, in questa fase risulteranno necessari:

- *Manuale utente*;
- *Verbali di periodo.*

Come per la fase precedente, anche i documenti di questa fase vanno mantenuti ed evoluti fino alla fase di consegna e accettazione del prodotto finale.

2.2.4.3 Codifica

2.2.4.3.1 Scopo

L'attività di codifica ha il fine di concretizzare la progettazione con la programmazione del software vero e proprio.

2.2.4.3.2 Aspettative

Questa attività dovrà avere come risultato un prodotto software avente le caratteristiche e i requisiti concordati con il proponente. Il codice generato dovrà rispettare alcune norme per poter essere leggibile e per poter facilmente intervenire in seguito nelle attività di manutenzione, modifica, verifica e validazione.

2.2.4.3.3 Stile della codifica

- **Indentazione**: I blocchi di codice innestati dovranno avere un'indentazione di quattro spazi;
- **Parentesi**: La parentesi aperta dovrà essere inserita nella stessa riga di dichiarazione del costrutto, separata da uno spazio, mentre la parentesi chiusa dovrà essere inserita con la giusta indentazione alla riga immediatamente successiva all'ultima riga di codice del costrutto;
- **Metodi**: il nome dei metodi dovrà iniziare con lettera minuscola e, se composto da più parole, le successive dovranno iniziare con lettera maiuscola. È preferibile mantenere metodi brevi, con poche righe di codice;
- **Classi**: il nome delle classi dovrà sempre iniziare con la lettera maiuscola e, come per i metodi, se composto da più parole, le successive dovranno iniziare con la lettere maiuscola;
- **Variabili**: il nome delle variabili deve sempre essere scritto in minuscolo e in inglese. Se il nome è composto da più parole, la seconda dovrà iniziare con la lettera maiuscola;
- **Costanti**: il nome deve essere sempre scritto in maiuscolo e in inglese. Se il nome è composto da più parole, queste dovranno essere separate dal carattere '_';
- **Univocità dei nomi**: tutti i costrutti dovranno avere nomi univoci e significativi;
- **Commenti**: i commenti dovranno essere inseriti prima dell'inizio del costrutto e presentati in lingua italiana;
- **File**: dovranno avere un nome che inizia per lettera maiuscola che ne specifichi il contenuto.

2.2.5 Metriche

Di seguito viene riportato un riferimento alle metriche utilizzate durante questo processo:

2.2.5.1 Metriche per la qualità del prodotto

- Versioni browser supportate: §C.2.7.

2.2.5.2 Metriche per la funzionalità

- Copertura requisiti obbligatori §C.2.3;
- Copertura requisiti desiderabili §C.2.4;
- Copertura requisiti opzionali §C.2.5

2.2.5.3 Metriche per l' usabilità

- Facilità di utilizzo §C.2.6

2.2.6 Strumenti

Di seguito sono riportati gli strumenti utilizzati dal team di sviluppo durante la fase di codifica e quelli utilizzati a supporto di tale processo.

2.2.6.1 Strumenti per la codifica

- **Solidity**: È un linguaggio di programmazione orientato agli oggetti per la scrittura di smart contract$_G$. Viene utilizzato per implementare smart contract$_G$ su varie piattaforme blockchain$_G$, quali ad esempio Ethereum$_G$, Avalanche$_G$ o Fantom$_G$;

  https://docs.soliditylang.org/

- **Truffle Suite [Deprecato]**: Truffle è un ambiente di sviluppo che consente agli utenti di testare applicazioni blockchain$_G$;

  https://trufflesuite.com/

- **Node.js**: È un runtime system$_G$ open source$_G$ multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript$_G$;

  https://nodejs.org/it/

- **Express**: È un framework$_G$ open source$_G$ per applicazioni web per Node.js

  https://expressjs.com/it/

- **ESLint**: È uno strumento di analisi del codice statico per identificare i modelli problematici trovati nel codice JavaScript$_G$;

  https://eslint.org/

- **Solidity-Coverage**: È una libreria di Node.js per testare il coverage degli smart contract$_G$;

  https://www.npmjs.com/package/solidity-coverage

- **React**: È una libreria open source$_G$, frontend$_G$, JavaScript$_G$ per la creazione di interfacce utente;

  https://it.reactjs.org/

- **Mocha**: È un framework di test JavaScript$_G$ per i programmi Node.js;

  https://mochajs.org/

- **Chai**: È una libreria di testing per Node.js e browser.

  https://www.chaijs.com/

- **Jest**: È un framework di test JavaScript$_G$ per i programmi Typescript.js;

  https://jestjs.io/

- **Cypress**: È un framework di test e2e.

  https://www.cypress.io/

- **Hardhat**: È un framework di test di compilazione e di esecuzione di smart contract$_G$ per Ethereum$_G$.

  https://hardhat.org/

2.2.6.2 Strumenti per la documentazione

2.2.6.3 Strumenti di supporto alla codifica

- **Docker**: Semplifica la creazione, la gestione e la distribuzione di applicazioni tramite dei contenitori, ovvero ambienti compatti (ad esempio Macchine virtuali) che forniscono una piattaforma per la compilazione e l'esecuzione di app, ma senza le dimensioni complete e il sovraccarico del sistema operativo completo;

  https://www.docker.com/

- **Metamask**: È un'estensione browser che funge da portafoglio di criptovalute$_G$. MetaMask consente ai propri utenti di conservare, scambiare e ricevere token$_G$ ETH, e più in generale tutti i token$_G$ basati sulla blockchain$_G$ Ethereum$_G$;

  https://metamask.io/

- **Fantom testnet**: È una rete di test in cui è possibile effettuare operazioni di scambio di criptovalute$_G$ senza nessuna spesa;

  https://docs.fantom.foundation/tutorials/set-up-metamask-testnet

- **Figma**: È un editor di grafica vettoriale e uno strumento di prototipazione. È principalmente basato sul Web con particolare attenzione alla collaborazione in tempo reale.

  https://www.figma.com/

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

Scopo di questa sezione è quello di normare la documentazione dei vari processi e le attività di sviluppo. Ci occuperemo quindi di definire le norme per la definizione della struttura che i vari documenti redatti dal gruppo *Yakuzaishi* dovranno avere.

3.1.2 Aspettative

Le aspettative del gruppo *Yakuzaishi* in questo processo sono:

- Delineare una valida e chiara struttura dei documenti;
- Definire una convenzione che accomuni tutti i tipi di documentazione redatta.

3.1.3 Descrizione

Lo scopo della documentazione è quello di trascrivere i fatti accaduti e le decisioni prese dal gruppo durante l'intera durata del progetto.

3.1.4 Istanziazione del processo

3.1.4.1 Ciclo di vita del documento

Le fasi del ciclo di vita$_{\mathbf{G}}$ del documento sono le seguenti:

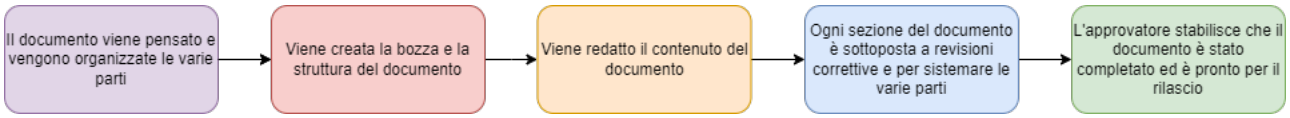

Figura 2: Descrizione del ciclo di vita del documento

3.1.4.2 Sviluppo e design

3.1.4.2.1 Template

Il team di sviluppo ha deciso di realizzare un template mediante l'utilizzo di LaTeX in modo da avere una struttura uniforme di tutte le pagine nei vari documenti. Un altro scopo del template è quello di velocizzare la stesura dei documenti stessi, dato che con la struttura già disponibile, i membri del gruppo che si occupano di redigere i documenti dovranno solamente occuparsi della stesura degli stessi, senza badare alla parte grafica. Il template in particolare definisce la prima pagina, il registro delle modifiche e la presentazione del documento.

3.1.4.2.2 Struttura del documento

In questa sezione viene descritta la struttura dei documenti. Questi ultimi sono stati suddivisi in due macrofamiglie la cui struttura differisce in alcune parti. Le due macrofamiglie vengono identificate come *verbali* e *documentazione*.

3.1.4.2.3 Verbali

I *verbali* vengono redatti successivamente alla riunione del gruppo oppure dopo un incontro con il proponente, essi contengono il rapporto dettagliato delle tematiche discusse durante il meeting. Prevedono una singola stesura, dato che contengono delle decisioni che non vengono modificate successivamente.

3.1.4.2.4 Prima pagina

La struttura di ogni prima pagine di un *verbale* è la seguente:

- **Logo**: il logo del gruppo *Yakuzaishi* è collocato in alto ed è centrato verticalmente;
- **Titolo**: collocato sotto al logo del gruppo, centrato verticalmente, composto dalla data e dal tipo di *verbale* interno/esterno;
- **Nome del capitolato**: collocato sotto il nome del documento, centrato verticalmente vi è il nome del capitolato in questione;
- **E-mail del gruppo**: collocata sotto il nome del capitolato, centrata verticalmente vi è la e-mail del gruppo *Yakuzaishi*;
- **Informazioni su documento**:
  - ***Responsabile***: indica il nome del *responsabile* in quel preciso momento;
  - ***Redattori***: indica chi si è occupato della redazione del documento;
  - ***Verificatori***: indica chi sono stati i verificatori in quel preciso documento;
  - **Uso**: indica se il documento è destinato a uso interno o esterno;
  - **Destinatari**: indica a chi è destinato il documento;
- **Sommario**: contenente una breve descrizione del documento.

Contenuti

Collocati nella seconda pagina del *verbale*. Si tratta di un indice che raccoglie i vari punti trattati nel *verbale*.

Generale

A seguire i contenuti, la sezione sopracitata contiene:

- **Informazioni sulla riunione**: contenenti data, ora di inizio, ora di fine, luogo e partecipanti dell'incontro;
- **Ordine del giorno**: contenente i temi trattati durante la riunione.

Resoconto

Ultima sezione dei *verbali*, contiene:

- **Verbale e azioni da intraprendere**: riporta in maniera dettagliata i punti discussi durante la riunione;
- **Tracciamento delle decisioni**: riassunto in forma tabellare delle decisioni prese.

3.1.4.2.5 Struttura delle pagine

Ogni pagina di contenuto è strutturata come segue:

- **Intestazione** in alto composta da:
  - **Nome del gruppo** in alto a sinistra;
  - **Logo del gruppo** in alto a destra.
- **Piè di pagina** in basso composto da:
  - **Nome del documento** in basso a sinistra;
  - **Numero di pagina** in basso a destra, contenente sia il numero di pagina corrente sia il numero di pagine totali.
- **Contenuto** compreso tra l'intestazione e il piè di pagina.

3.1.4.3 Documentazione

Fanno parte di questa famiglia tutti i documenti diversi dai *verbali*.

3.1.4.3.1 Prima pagina

La struttura di ogni prima pagina dei vari documenti è la seguente:

- **Logo**: il logo del gruppo *Yakuzaishi* è collocato in alto ed è centrato verticalmente;
- **Nome del documento**: collocato sotto il logo del gruppo, centrato verticalmente vi è il nome del documento in questione;
- **Nome del capitolato**: collocato sotto il nome del documento, centrato verticalmente vi è il nome del capitolato in questione;
- **E-mail del gruppo**: collocata sotto il nome del capitolato, centrata verticalmente vi è la e-mail del gruppo *Yakuzaishi*;
- **Informazioni sul documento**:
  - ***Responsabile***: indica il nome del *responsabile* in quel preciso momento;
  - ***Redattori***: indica chi si è occupato della redazione del documento;
  - ***Verificatori***: indica chi si è occupato della verifica in quel preciso documento;
  - **Uso**: indica se il documento è destinato a uso interno o esterno;
  - **Destinatari**: indica a chi è destinato il documento;
  - **Versione**: indica la versione del documento.
- **Sommario**: spiega brevemente le finalità del documento.

3.1.4.3.2 Registro delle modifiche

Il file *changelog.tex* contiene il registro delle modifiche, presente in ogni singolo documento, eccezion fatta per i *verbali*. Esso corrisponde a una tabella nella quale si tiene traccia di ogni attività svolta (stesura, modifica, verifica, approvazione) e viene aggiornato ogni volta che avviene una delle attività precedentemente citate. La tabella del registro delle modifiche contiene le seguenti voci:

- **Versione**: versione attuale del documento;

- **Data**: data in cui accade un determinato evento;
- **Autore**: indica chi ha apportato dei cambiamenti al documento;
- **Ruolo**: ruolo che l'autore svolgeva in quel preciso momento;
- **Descrizione**: breve descrizione della modifica effettuata.

Dalla fase di progettazione e codifica si è deciso di inserire nella riga dove viene indicato l'autore anche il nome del verificatore che verifica la parte del documento redatta dall' autore precedentemente citato, in modo tale da essere conformi con in principio di *continous integration*.

3.1.4.3.3 Indice

L'indice, posto in seguito al registro delle modifiche, deve contenere tutte le varie sezioni di cui si compone il documento.

3.1.4.3.4 Elenco delle tabelle

L'elenco delle tabelle, successivo all'indice, deve contenere tutte le tabelle che compaiono nel documento.
Questo non sarà presente qualora il documento in questione non prevede che vi siano tabelle al suo interno.

3.1.4.3.5 Elenco delle figure

L'elenco delle tabelle, posto a seguito dell'elenco delle tabelle, deve contenere tutte le figure che compaiono nel documento.
Questo non sarà presente qualora il documento in questione non prevede che vi siano immagini al suo interno.

3.1.4.3.6 Struttura delle pagine

Ogni pagina di contenuto è strutturata come segue:

- **Intestazione** in alto composta da:
  - **Nome del gruppo** in alto a sinistra;
  - **Logo del gruppo** in alto a destra.
- **Piè di pagina** in basso composto da:
  - **Nome del documento** in basso a sinistra;
  - **Numero di pagina** in basso a destra, contenente sia il numero di pagina corrente sia il numero di pagine totali.
- **Contenuto** compreso tra l'intestazione e il piè di pagina.

3.1.4.4 Convenzioni

3.1.4.4.1 Nomi dei file

Di seguito viene descritta la rappresentazione dei nomi dei file, i quali sono validi per tutti i documenti:

- I nomi dei file iniziano tutti con la lettera minuscola;
- Se il nome comprende più parole allora ognuna di esse è separata dal simbolo '_';

Esempi corretti:

- introduzione;
- norme_di_progetto.

Esempi non corretti:

- Norme_di_Progetto (sono presenti lettere maiuscole);
- NormeDiProgetto (sono presenti lettere maiuscole e sono assenti i caratteri separatori).

3.1.4.4.2 Stile di testo

Nella sezione sottostante vengono riportati i vari stili di testo utilizzati nei documenti e i contesti in cui essi sono utilizzati:

- **Grassetto**: lo stile grassetto viene utilizzato per indicare i termini negli elenchi puntati e per i titoli delle sezioni;
- **Corsivo**: lo stile corsivo viene utilizzato per indicare il nome del gruppo, per il nome del proponente e per le parole di particolare rilevanza all'interno dei documenti;
- **Link**: i link rappresentano dei collegamenti esterni al documento, essi verranno rappresentati di colore blu e sottolineati;
- **Nome dei documenti:** il nome dei documenti viene rappresentato in corsivo. Se si fa un riferimento specifico a un particolare documento bisogna indicarne anche la versione (sempre in corsivo);
- **Collegamenti interni:** le parole che si riferiscono a una parte del documento vanno sottolineate.

3.1.4.4.3 Glossario

Le norme relative al *glossario* sono:

- Ogni parola presente nel *glossario* viene contrassegnata con una 'G' a pedice;
- Se un termine compare nella sua stessa definizione all'interno del *glossario* esso viene contrassegnato.

3.1.4.4.4 Elenchi puntati e numerati

Di seguito viene la descrizione di come il team di sviluppo ha deciso di utilizzare gli elenchi puntati e numerati.

- Ogni punto dell'elenco inizia con la lettera maiuscola;
- Alla fine di ogni punto vi è un ';';
- Dopo l'ultima voce vi è un '.';
- Se vi è un concetto da spiegare esso viene scritto in grassetto seguito da ':' e segue la spiegazione di esso.

3.1.4.4.5 Sigle

In questa sezione vengono raccolte e spiegate le sigle presenti nei vari documenti:

- Sigle che rappresentano i ruoli che ogni membro del gruppo a rotazione deve svolgere:

Ruolo	Sigla
Responsabile	Re
Amministratore	Am
Analista	An
Progettista	Pt
Programmatore	Pr
Verificatore	Ve

Tabella 5: Sigle dei ruoli

3.1.4.4.6 Formato della data

Il team di sviluppo ha deciso di adottare la seguente convenzione per rappresentare le date che compaiono nei vari documenti:

YYYY-MM-DD

dove:

- **YYYY**: indica l'anno;
- **MM**: indica il mese;
- **DD**: indica il giorno.

3.1.4.5 Elementi grafici

3.1.4.5.1 Tabelle

Ad eccezione del registro delle modifiche, le tabelle di ogni documento seguono le seguenti convenzioni:

- Ogni tabella contiene al di sotto di essa, in posizione centrale, una didascalia descrittiva;
- Ogni tabella viene identificata con un numero progressivo a partire da 1, che la identifica univocamente all'interno del documento;
- Per chiarezza grafica e per aumentarne la leggibilità, verrà usato un trattino (-) nelle celle in cui è presente uno 0 (zero). Inoltre si precisa che laddove venga indicato un costo all'interno di una tabella, quest'ultimo verrà espresso in euro (€).

3.1.4.5.2 Immagini

Le immagini sono collocate al centro della pagina in questione e al di sotto di ciascuna di esse è presente, in posizione centrale, una descrizione esplicativa e un numero progressivo a partire da 1, che la identifica univocamente all'interno del documento.

3.1.4.5.3 Diagrammi

Sia i diagrammi UML$_G$ che i diagrammi di Gantt$_G$ vengono riportati come immagini, quindi sono soggetti alle regole sopra riportate.

3.1.4.6 Verifica ortografica

Per la verifica ortografica dei documenti scritti in LaTeX si utilizzerà lo strumento di correzione automatica **LanguageTool**:

https://languagetool.org

3.1.5 Metriche

Di seguito vengono riportati i riferimenti alle metriche utilizzate durante questa fase:

- Indice di Gulpease: §C.2.1;
- Errori ortografici: §C.2.2.

3.1.6 Strumenti

Di seguito sono riportati gli strumenti utilizzati dal team di sviluppo in fase di redazione dei documenti:

- **LaTeX**: linguaggio di markup per la preparazione di testi, basato sul programma di composizione tipografica TEX;

  https://www.latex-project.org

- **Visual Studio Code**: IDE$_G$ versatile e gratuito, scelto per la stesura del codice LaTeX$_G$ necessario a produrre i documenti in quanto, grazie all'estensione **LaTeX Workshop**, comprende un compilatore integrato, l'autocompletamento dei comandi LaTeX$_G$ e la segnalazione di errori di sintassi;

  https://code.visualstudio.com

- **Diagrams.net**: software di disegno grafico multipiattaforma gratuito e open source$_G$. Il gruppo ha scelto questo strumento per la progettazione di diagrammi UML$_G$ in quanto il software è gratuito e presenta tutte le funzionalità necessarie allo scopo.

  https://www.diagrams.net

3.2 Gestione della configurazione

3.2.1 Scopo

Scopo di questa sezione è definire come il team *Yakuzaishi* ha deciso di affrontare la tematica della gestione della configurazione, ovvero come il team di sviluppo ha deciso di mantenere tracciata la documentazione redatta ed il codice sviluppato.

3.2.2 Aspettative

Le aspettative del gruppo *Yakuzaishi* nell'utilizzo di questo processo sono:

- Possibilità di tracciare tutte le modifiche effettuate;
- Possibilità di ripristino, qualora fosse necessario, a una versione precedente;
- Possibilità di condivisione dei file configurati tra tutti i membri del gruppo;
- Possibilità di individuare e correggere eventuali errori e/o conflitti.

3.2.3 Descrizione

Il processo di gestione della configurazione ha lo scopo di mantenere organizzata e tracciabile la documentazione redatta e il codice sviluppato, creando una storia per ciascun file prodotto. In particolare si vuole gestire la struttura e la disposizione delle varie parti di ogni file all'interno di repository$_{\mathbf{G}}$ facilmente accessibili e navigabili.

3.2.4 Istanziazione del processo

3.2.4.1 Versionamento

Codice di versione

Ogni versione di documento è identificata tramite un codice numerico di tre cifre:

[X].[Y].[Z]

il cui significato viene spiegato nel seguito:

- **X**: versione stabile, sottoposta ad approvazione del *responsabile* del documento;
- **Y**: versione controllata, sottoposta a revisione da parte del *verificatore* del documento;
- **Z**: versione modificata dal redattore del documento, seguita da una verifica di quanto scritto.

3.2.4.1.1 Metriche del codice di versione

Le cifre del codice di versione precedentemente descritte seguono una particolare metrica di avanzamento:

1. Tutte le cifre iniziano dal valore 0;
2. Ciascuna cifra aumenta di un'unità ogni qual volta viene compiuta un'operazione sul documento e in particolare:
   - Se la cifra **X** viene modificata, le cifre **Y** e **Z** ritornano al valore 0;
   - Se la cifra **Y** viene modificata, la cifra **Z** ritorna al valore 0;

3.2.4.2 Sistemi software utilizzati

Per gestire i repository$_{\mathbf{G}}$ Git$_{\mathbf{G}}$ si è scelto di utilizzare il servizio offerto da GitHub$_{\mathbf{G}}$ in quanto:

- Gran parte dei membri del gruppo hanno già precedentemente familiarizzato con questo software;
- Offre un servizio multipiattaforma dall'utilizzo tramite linea di comando, alla web app$_{\mathbf{G}}$, fino ad arrivare alle applicazioni desktop e mobile;

- Possibilità di suddividere il tempo in milestone$_G$;
- Possibilità di organizzare e suddividere il lavoro tramite la creazione di issues$_G$ assegnabili alle milestone$_G$.

3.2.4.3 Struttura del repository

Al fine di organizzare meglio il lavoro si è deciso di creare quattro repository$_G$ distinti, tutti pubblici, con lo scopo di tenere separati i documenti e il software:

- **Yakuzaishi-SWE/docs**: per il versionamento dei documenti;
- **Yakuzaishi-SWE/shopchain**: per il versionamento del codice backend$_G$;
- **Yakuzaishi-SWE/shopchain-frontend**: per il versionamento del codice frontend$_G$;
- **Yakuzaishi-SWE/smart-contract**: per il versionamento del codice dello smart contract$_G$.

3.2.4.3.1 Yakuzaishi-SWE/docs

L'organizzazione del repo è così riassunta:

- **Branch$_G$ master**: è il branch$_G$ principale in cui è presente la sola documentazione, in formato tex, pronta alla revisione;
- **Branch$_G$ dev**: è un branch$_G$ intermedio tra il master e i branch$_G$ dei singoli documenti. Utile per mantenere pulito il master ed evitare confusione;
- **Branch$_G$ derivanti dal dev**: sono diversi in numero e ciascuno di questi ha un nome parlante riferito al singolo documento. Ognuno è dedicato alla stesura del documento da cui prende il nome e, quando il lavoro su uno di questi branch$_G$ sarà finito e sottoposto a revisione, verrà eseguito un merge$_G$ del branch$_G$ in questione con il dev;
- **Branch$_G$ gh-pages** è un branch$_G$ nel quale vengono caricati i file compilati in formato pdf della documentazione presente nel master. I file presenti in questo branch$_G$ vengono automaticamente aggiunti e resi consultabili in formato pdf alla seguente pagina web:

  https://yakuzaishi-swe.github.io/docs/.

Gestione dei cambiamenti

La separazione del flusso di lavoro tra i vari documenti da redarre permette una notevole diminuzione dei conflitti. Il punto focale è che il branch$_G$ master rimanga pulito da ogni tipo di errore, per cui non è utilizzabile da nessun membro del gruppo finché ciascun *responsabile* non abbia dato l'approvazione al corrispettivo documento. Solo in quel momento è permesso il merge$_G$ di uno dei branch$_G$ minori nel master. I cambiamenti da gestire sui documenti possono essere:

- **Modifiche minori**: riguardano errori grammaticali, lessicali o di sintassi, che possono essere corretti dai *redattori* o dai *verificatori* senza l'approvazione del *responsabile*;
- **Modifiche generali**: riguardano cambiamenti più generali come la struttura del documento o convenzioni da utilizzare e richiedono il consulto con il *responsabile*, il quale potrà accettare o declinare la proposta di modifica.

3.2.4.3.2 Yakuzaishi-SWE/shopchain, Yakuzaishi-SWE/shopchain-frontend e Yakuzaishi-SWE/smart-contract

Yakuzaishi-SWE/shopchain, Yakuzaishi-SWE/shopchain-frontend e Yakuzaishi-SWE/smart-contract sono organizzate allo stesso modo. In particolare:

- **Branch$_G$ master**: è il branch$_G$ principale in cui sono presenti i soli file pronti per le release di consegna;
- **Branch$_G$ dev**: è un branch$_G$ intermedio tra il master e i branch$_G$ delle singole features. Utile per mantenere pulito il master ed evitare confusione;
- **Branch$_G$ derivanti dal dev**: sono diversi in numero e ciascuno di questi ha un nome parlante riferito alla singola feature. Ognuno è dedicato allo sviluppo della feature da cui prende il nome e, quando il lavoro su uno di questi branch$_G$ sarà finito e sottoposto a revisione, verrà eseguito un merge$_G$ del branch$_G$ in questione con il dev.

Gestione dei cambiamenti

La separazione del flusso di lavoro tra le varie features da sviluppare permette una notevole diminuzione dei conflitti. Il punto focale è che il branch$_G$ master rimanga pulito da ogni tipo di errore, per cui non è utilizzabile da nessun membro del gruppo finché ciascun *responsabile* non abbia dato l'approvazione per il rilascio. Solo in quel momento è permesso il merge$_G$ di uno dei branch$_G$ minori nel master.

3.2.5 Metriche

Questo processo non fa uso di metriche particolari.

3.2.6 Strumenti

Questo processo non fa uso di strumenti particolari.

3.3 Gestione della qualità

3.3.1 Scopo

Lo scopo di questa sezione è la definizione del modo in cui il gruppo si impegna nella gestione della qualità.

3.3.2 Aspettative

Le aspettative del team di sviluppo durante questo processo sono le seguenti:

- Conseguimento della qualità del prodotto, in linea con le richieste del proponente;
- Buona qualità di organizzazione del gruppo;
- Prova oggettiva della qualità del prodotto.

3.3.3 Descrizione

La gestione di qualità è un processo che viene descritto nel *piano di qualifica*, esso infatti ha il compito di determinare le metriche e le modalità usate per valutare la qualità di prodotti e processi. Il team di sviluppo ritiene fondamentale raggiungere una buona qualità del prodotto, e ha ritenuto che il modo migliore per raggiungere questo obiettivo è mediante un approccio di tipo sistematico, ovvero assegnando a ciascun componente del gruppo un compito e un ruolo, svolgendo così diversi processi simultaneamente e ottenendo così un ottimizzazione delle risorse, effettuando infine dei test sul prodotto finito.

3.3.4 Istanziazione del processo

3.3.4.1 Attività

Affinché il prodotto software e la documentazione redatta raggiungano una buona qualità, ogni componente del gruppo *Yakuzaishi* deve:

- Attenersi al *piano di qualifica*;
- Porsi obiettivi incrementali, in modo da ridurre il margine di errore;
- Creare un avanzamento continuo della formazione personale, mediante conoscenze pregresse, conoscenze di altri membri del gruppo o mediante autoformazione.

3.3.4.2 Classificazione delle metriche

Le metriche identificate mantengono la seguente struttura:

M[Utilizzo][ID Numerico]

Dove utilizzo può essere:

- **PC:** processo;
- **PD:** prodotto.

Invece l' ID Numerico è un numero intero a partire da 1 che distingue le metriche dello stesso sottoinsieme (processo o prodotto).

3.3.5 Metriche

Questo processo non fa uso di metriche particolari.

3.3.6 Strumenti

Questo processo non fa uso di strumenti particolari.

3.4 Verifica

3.4.1 Scopo

Il processo di verifica ha come obiettivo la realizzazione di prodotti corretti, coesi e completi. Tale processo prende in input ciò che è già stato prodotto e lo restituisce in uno stato conforme alle aspettative. Per ottenere tale risultato ci si affida a processi di analisi e test così da accertare che non siano stati introdotti errori durante lo sviluppo del software e la stesura dei documenti. Scopo di questa sezione è definire come il gruppo ha deciso di attuare il processo di verifica.

3.4.2 Aspettative

Le aspettative del gruppo *Yakuzaishi* nell'utilizzo di questo processo sono:

- Verifica di ciascuna fase, rispettando criteri precisi, consistenti e modificabili qualora dovesse essere necessario;
- Svolgimento di una verifica attenta al fine di ottenere successo in fase di validazione;
- Le attività svolte durante il processo di verifica devono essere rese quanto più possibile automatizzate;
- Devono essere rispettati gli obiettivi di copertura indicati nel *piano di qualifica*.

3.4.3 Descrizione

Durante questo processo, il compito dei verificatori è quello di effettuare l'analisi dei prodotti del team. Tale analisi si differenzia in due diverse tipologie:

- **Analisi statica**: processo di valutazione di un sistema o di un suo componente basato sulla sua forma, struttura, contenuto, documentazione. Questo tipo di analisi viene generalmente svolto tramite ispezioni e revisioni e può essere svolta sui documenti così come sul software o su parti di esso;
- **Analisi dinamica**: processo di valutazione di un sistema software o di un suo componente basato sull'osservazione del suo comportamento in esecuzione. Questo tipo di analisi viene generalmente chiamato testing e per motivi logici non può essere svolta sui documenti.

3.4.4 Istanziazione del processo

3.4.4.1 Verifica della documentazione

Il processo di verifica della documentazione può essere svolto mediante l'uso di strumenti automatici, oppure può essere condotta a mano. In questo secondo caso si possono utilizzare due diverse tecniche di analisi descritte nel seguito:

- **Walktrough**: Il *verificatore* esegue un controllo completo del documento alla ricerca di eventuali errori;
- **Inspection**: Il *verificatore* esegue un controllo mirato ai soli punti del documento nei quali vi è una più ampia possibilità di errore. Il controllo mirato avviene grazie all'esperienza del *verificatore* e a una lista di controllo$_{\mathbf{G}}$.

La tecnica **Walkthrough** risulta molto onerosa a causa dell'adozione di una ricerca esaustiva degli errori. Questa metodologia sarà applicata come fase iniziale di verifica della documentazione. Con l'avanzare dell'attività di progetto e il ripetersi del processo di verifica sulla documentazione, si stilerà una lista di controllo$_{\mathbf{G}}$ grazie alla quale, successivamente, sarà possibile passare alla tipologia di verifica **Inspection** che risulta essere più rapida.

3.4.4.2 Verifica del codice

Il processo di verifica del codice farà uso di test automatici appositamente creati e si servirà inoltre di analisi statica e dinamica per quanto riguarda il codice scritto e in particolare:

- **Analisi statica**: verificatori e programmatoti durante le fasi di stesura e verifica del codice si accertano che siano stati rispettati i principi di buona programmazione preimpostati dal team;
- **Analisi dinamica**: verificatori e programmatori durante la fase di test vanno alla ricerca di bug_G ed errori tramite l'esecuzione del software.

3.4.4.2.1 Test

Come esposto in precedenza, questa attività ha lo scopo di evidenziare eventuali bug_G e/o errori riscontrabili a runtime. L'esecuzione dei test deve essere ripetibile ed è quindi buona norma renderla quanto più possibile automatizzata.

Classificazione dei test

I test sono identificati tramite un codice descritto dalla seguente espressione regolare:

T[TipologiaTest][ImportanzaRequisito]*[TipologiaRequisito]*[IdNumerico]

dove:

TipologiaTest può assumere uno tra i seguenti valori letterali:

Sigla	Significato	Descrizione
A	Test di accettazione	Vengono eseguiti immediatamente prima del rilascio del prodotto
I	Test di integrazione	Servono a testare che i moduli quando integrati funzionino come previsto, ovvero testare che i moduli che funzionano bene individualmente non abbiano problemi quando integrati
S	Test di sistema	Condotti su un sistema integrato completo, servono per valutare la conformità del sistema ai requisiti specificati
U	Test di unità	Condotti su una parte di codice, servono per verificare che tale parte funzioni correttamente

Tabella 6: Tipologie test

A loro volta i test di unità si identificano nella seguente maniera:

Sigla	Significato
UF	Test di unità frontend$_G$
UO	Test di unità Solidity OrderManager
UM	Test di unità Solidity MoneyBoxManager

Tabella 7: Classificazione test di unità

–

ImportanzaRequisito e **TipologiaRequisito** sono presenti solo nei codici per i test di sistema e accettazione. Questi campi identificano il requisito che si vuole testare, possono assumere i valori riportati e precedentemente descritti al paragrafo 2.2.4.1.1.

IdNumerico è un valore crescente che parte da 1 e serve per distinguere i diversi test riguardanti la stessa tipologia.

3.4.5 Metriche

Segue un riferimento alle varie metriche utilizzate durante questo processo:

- Earned Value (EV): §C.1.1;
- Actual Cost (AC): §C.1.2;
- Planned Value (PV): §C.1.3;
- Cost Variance (CV): §C.1.4;
- Schedule Variance (SV): §C.1.5;
- Estimated At Completion (EAC): §C.1.6;
- Estimate To Complete (ETC): §C.1.7;
- Requirements Stability Index (RSI): §C.1.8;
- Passed Tests: §C.1.9;
- Metrics Satisfied: §C.1.10;
- Risks Found: §C.1.11.

3.4.6 Strumenti

Questo processo non fa uso di strumenti particolari.

3.5 Validazione

3.5.1 Scopo

Scopo di questa sezione è definire come il team di sviluppo ha deciso di affrontare il processo di validazione, ovvero per assicurarsi che il prodotto sviluppato sia in linea con i requisiti concordati con il proponente e che soddisfi i bisogni del committente.

3.5.2 Aspettative

Le aspettative del team di sviluppo nell'utilizzo di questo processo sono le seguenti:

- Assicurarsi che il prodotto software sia conforme con i requisiti riportati nell'*analisi dei requisiti*;
- Dimostrare la correttezza delle attività svolte in fase di verifica.

3.5.3 Descrizione

Tale processo consiste nell'esaminare il prodotto nella fase di verifica e assicurarsi che esso sia in linea con i requisiti concordati con il proponente e che soddisfi i bisogni del committente. Sarà il *responsabile* di progetto che avrà la responsabilità di controllare i risultati decidendo se:

- Accettare il prodotto;
- Rifiutarlo richiedendo una nuova verifica.

3.5.4 Istanziazione del processo

3.5.4.1 Validazione della documentazione

Al termine della stesura di un documento il *responsabile* rilegge il documento, ne calcola l'*indice di Gulpease* e si assicura che gli obiettivi stabiliti siano stati raggiunti. Se il documento soddisfa le attese allora viene approvato per il rilascio, invece se non le soddisfa il *responsabile* assegna il compito di sistemare le parti carenti e la procedura si ripete.

3.5.4.2 Validazione del codice

Il *verificatore* esegue i test di unità, integrazione e sistema di una issue$_{\mathbf{G}}$ assegnatagli, oltre ai test di accettazione preventivamente concordati.

3.5.5 Metriche

Seguono le metriche utilizzate in questo processo:

- Solidity Statement Coverage: §C.2.8;
- Solidity Branch Coverage: §C.2.9;
- Solidity Function Coverage: §C.2.10;
- Solidity Line Coverage: §C.2.11.
- Frontend Statement Coverage: §C.2.12;
- Frontend Branch Coverage: §C.2.13;
- Frontend Function Coverage: §C.2.14;
- Frontend Line Coverage: §C.2.15.

3.5.6 Strumenti

Questo processo non fa uso di strumenti particolari.

4 Processi organizzativi

I processi organizzativi sono tutti quelli che riguardano l'organizzazione del gruppo *Yakuzaishi*.
Essi si suddividono in:

- **Processo di gestione organizzativa**;
- **Processo di formazione**.

4.1 Gestione Organizzativa

4.1.1 Scopo

Lo scopo di questa sezione è quello di normare le modalità di coordinamento tra i vari membri del gruppo:

- Quantificare le spese;
- Rispettare le scadenze;
- Quantificare i rischi.

4.1.2 Aspettative

Le aspettative in questa fase sono le seguenti:

- Ottenere un'organizzazione ragionevole ed efficace tra i membri del gruppo;
- Adottare un buon *way of working*;
- Riuscire a controllare le spese;
- Ottenere un'equa distribuzione dei ruoli, riuscendo ad avere una rotazione efficace.

4.1.3 Descrizione

Le attività di gestione sono:

- Assegnazione dei ruoli e dei compiti;
- Inizio e definizione dello scopo;
- Istanziazione dei processi;
- Pianificazione e stima di tempi, risorse e costi;
- Esecuzione e controllo;
- Revisione e valutazione periodica delle attività.

4.1.4 Istanziazione del processo

4.1.4.1 Ruoli di progetto

Responsabile di progetto

Il suo compito consiste nel garantire lo svolgimento delle attività pianificate entro i tempi e le modalità previste dal gruppo.
Rappresenta il gruppo nelle comunicazioni esterne con committenti e proponenti.
Le sue responsabilità sono:

- Gestire l'assegnazione dei compiti agli altri membri del gruppo;
- Organizzare il lavoro in modo da minimizzare la probabilità che si verifichino problemi;
- Approvare la documentazione nella fase finale del processo di verifica.

Amministratore

È responsabile degli strumenti necessari all'ambiente di lavoro.
Le sue responsabilità sono:

- Gestire il sistema di archiviazione e versionamento di documentazione e codice;
- Gestire il sistema di configurazione e versionamento del prodotto;
- Mantenere efficiente l'ambiente di sviluppo, fornendo strumenti adeguati ai membri del gruppo;
- Redigere le *Norme di progetto v3.0.0* che l'intero gruppo deve seguire;
- È responsabile della redazione e attuazione di piani e procedure di gestione per la qualità.

Analista

Il suo compito consiste nell'individuare, analizzare e documentare i servizi che il sistema deve fornire.
Le sue responsabilità sono:

- Analizzare la complessità del problema generale e delle funzionalità da implementare;
- Modellare concettualmente il sistema;
- Individuare i requisiti del progetto e suddividerli in categorie;
- Redigere la *Analisi dei requisiti v3.0.0*.

Progettista

Il suo compito consiste nel definire la struttura architetturale del sistema.
È responsabile delle attività di progettazione, le quali devono portare alla realizzazione di un prodotto in grado di soddisfare i requisiti individuati dagli analisti.
Le sue responsabilità sono:

- Studiare l'architettura più adatta al prodotto da realizzare;
- Garantire la qualità del prodotto;
- Identificare l'architettura ad alto livello del sistema e l'architettura a livello di componenti del sistema.

Programmatore

Il suo compito consiste nella scrittura del codice richiesto dallo svolgimento del progetto.
Le sue responsabilità sono:

- Scrivere codice documentato, manutenibile e versionato;
- Rendere l'attività di verifica semplice svolgendo il proprio compito con gli strumenti e le modalità previsti da questo documento.

Verificatore

Il suo compito consiste nella verifica del progetto.
Le sue responsabilità sono:

- Controllare che vengano rispettate le norme di progetto, inserite nell'omonimo documento *Norme di progetto v3.0.0*;
- Assicurarsi della conformità di ogni stadio del ciclo di vita$_{\mathbf{G}}$ del prodotto;
- Segnalare eventuali anomalie riscontrate nella verifica in modo che vengano corrette tempestivamente.

4.1.4.2 Procedure

Per il coordinamento e le comunicazioni durante l'intera durata del progetto il gruppo *Yakuzaishi* adotterà le seguenti procedure:

- **Comunicazione interna:** coinvolge tutti i membri del gruppo;
- **Comunicazione esterna:** coinvolge proponente e committenti.

4.1.4.3 Rotazione dei ruoli

È prevista una rotazione dei ruoli a cadenza periodica da parte del gruppo *Yakuzaishi*.
L'attribuzione dei ruoli viene svolta secondo i seguenti criteri:

- **Equità:** ogni membro del gruppo deve svolgere un numero di ore per ogni ruolo in maniera equa nei confronti degli altri membri;
- **Continuità:** i ruoli vengono fatti ruotare in concomitanza con l'inizio di un nuovo periodo, in modo da garantire che il lavoro svolto dai membri del gruppo abbia una continuità nel corso di ciascun periodo;
- **Assenza di conflitti:** poiché ogni membro può ricoprire più ruoli in un periodo, si cerca di assegnarli in modo che non ci sia conflitto di interesse tra essi. Vige in ogni caso la regola che un *responsabile* non può approvare il proprio lavoro di verifica se ha assunto anche il ruolo di *verificatore*, mentre un *verificatore* non può verificare i propri testi o codice qualora abbia assunto il ruolo di *analista* o *programmatore* nello stesso periodo.

4.1.4.4 Impiego delle infrastrutture interne

4.1.4.4.1 Gestione delle comunicazioni

Comunicazioni interne

Le comunicazioni interne del gruppo *Yakuzaishi* avvengono tramite i canali Discord$_{\mathbf{G}}$ e Telegram$_{\mathbf{G}}$.
Il canale Discord$_{\mathbf{G}}$ viene utilizzato principalmente per gli incontri settimanali tra i membri del gruppo, essendo uno strumento utilizzato anche in ambito aziendale, ciò favorisce uno spazio di lavoro condiviso tra i membri del gruppo. Per comunicazioni rapide invece viene preferito il canale Telegram$_{\mathbf{G}}$.

Comunicazioni esterne

Le comunicazioni con utenti esterni al gruppo sono gestite dal *responsabile* del progetto. Le modalità utilizzate sono le seguenti:

- Tramite e-mail (All'indirizzo *yakuzaishi.swe@gmail.com*);
- Tramite Google Meet$_{\mathbf{G}}$ (Tramite link di invito mandato di volta in volta dal proponente);
- Tramite Discord$_{\mathbf{G}}$ (Tramite il server dell'azienda SyncLab).

4.1.4.4.2 Gestione degli incontri

Incontri interni

Il *responsabile* ha il compito di organizzare gli incontri interni utilizzando l'apposito canale di comunicazione.
Per ogni incontro viene redatto un *verbale* che raccoglie tutti gli argomenti trattati durante la riunione.

4.1.4.4.3 *Verbali* di incontri interni

In occasione di ogni incontro interno viene redatto un *verbale* dal segretario scelto dal *responsabile*. Il contenuto della riunione deve essere riportato nel *verbale* corrispondente e approvato dal *responsabile*.

Incontri esterni

Il *responsabile* ha il compito di organizzare gli incontri esterni con il proponente o con il committente.
Un incontro esterno viene richiesto se vi è la necessità impellente di chiarimento di una problematica riscontrata da parte del gruppo.
In questo caso il *responsabile* provvede a formalizzare una richiesta all'entità esterna (proponente o committente) tramite le mail di contatto fornite.
In caso di richiesta accettata, verranno comunicate ai membri del gruppo, tramite canali di comunicazione interni, le informazioni riguardanti data, ora e luogo dell'incontro.
Come per gli incontri interni, deve essere redatto un *verbale* che descriva gli argomenti trattati.

4.1.4.4.4 *Verbali* di incontri esterni

In occasione di ogni incontro esterno viene redatto un *verbale* da un redattore scelto dal *responsabile*.
Il contenuto della riunione deve essere riportato nel *verbale* esterno corrispondente e approvato dal *responsabile*.

4.1.4.4.5 Gestione degli strumenti di coordinamento

Ticketing

Per la suddivisione delle issue$_{\mathbf{G}}$ viene utilizzato *GitHub Projects Board*$_{\mathbf{G}}$.
Il *responsabile* avrà il compito di creare e assegnare le varie issue$_{\mathbf{G}}$ ai membri del gruppo.
GitHub Projects Board$_{\mathbf{G}}$ permette una buona gestione delle issue$_{\mathbf{G}}$, infatti:

- Le issue$_{\mathbf{G}}$ possono trovarsi in quattro stati diversi:
  - **New:** sono presenti le issue$_{\mathbf{G}}$ appena create che devono ancora essere assegnate;
  - **To Do:** sono presenti le issue$_{\mathbf{G}}$ assegnate e da portare a termine;
  - **In progress:** sono presenti le issue$_{\mathbf{G}}$ attualmente in esecuzione;
  - **Done:** sono presenti le issue$_{\mathbf{G}}$ portate a termine.

- Ogni membro del gruppo ha la possibilità di spostare le proprie issue$_G$ nelle diverse colonne che rappresentano gli stati;
- Le issue$_G$ hanno una descrizione e possono ricevere commenti da parte degli altri componenti del gruppo;
- All'interno di una issue$_G$ possono essere presenti dei task sotto forma di checklist nella sezione commenti.

Ogni membro del gruppo ha il dovere di tenere aggiornate le sue attività e deve rispettare le scadenze delle varie issue$_G$. Se si completa un task bisogna spuntarlo come completato all'interno dell'issue$_G$. Inoltre è possibile creare delle milestone$_G$ a cui assegnare le issue$_G$, permettendo così di stabilire un punto di riferimento sulla linea temporale entro il quale è previsto il completamento di tali issue$_G$.

4.1.4.4.6 Gestione degli strumenti di versionamento

Repository

Per il versionamento dei file è stata scelta la piattaforma GitHub$_G$. L'*amministratore* ha creato il repository$_G$ e ha aggiunto come collaboratori tutti i membri del gruppo *Yakuzaishi*.

Tipi di file e *.gitignore*

Nelle cartelle che contengono i documenti, sono presenti solo i file *.tex* e *.png*.
Nel file *.gitignore* sono state aggiunte tutte le estensione che devono essere ignorate da un commit, al fine di evitare la presenza di file indesiderati o non rilevanti che devono quindi non essere versionati.

Gestione dei rischi

Il *responsabile* ha il compito di individuare i rischi e renderli noti. Tale attività verrà svolta nel *piano di progetto* e contiene i seguenti punti:

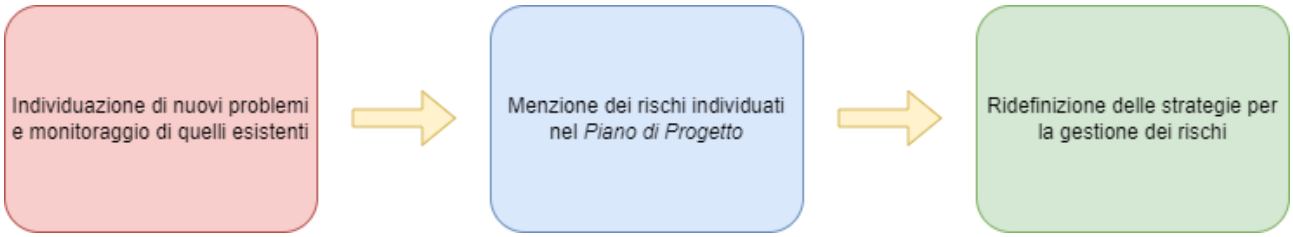

Figura 3: Attività di gestione dei rischi

Codifica dei rischi

I rischi sono codificati nella seguente maniera:

R[Tipo][ID Numerico]

dove:

R: Sta per rischio;

Tipo: Rappresenta la tipologia del rischio e può assumere uno dei seguenti valori letterali:

Sigla	Descrizione
T	Tecnologico
R	Requisito
O	Organizzativo
I	Interno

Tabella 8: Sigle utilizzate per la codifica dei rischi

ID numerico: È un id crescente che parte da 1 e viene incrementato di 1 per ciascun rischio.

La codifica di ciascun rischio è univoca. Non possono esserci due rischi dello stesso tipo con lo stesso ID numerico.

4.1.5 Metriche

Questo processo non fa uso di metriche particolari.

4.1.6 Strumenti

Gli strumenti utilizzati in questa fase sono i seguenti:

- **Discord:** utilizzato per le comunicazioni interne tra i membri del gruppo e per le riunioni con il proponente;

  https://discord.com/

- **Google Meet:** utilizzato per gli incontri con il proponente;

  https://meet.google.com/

- **Gmail:** utilizzato per le comunicazioni con i committenti ed il proponente;

  https://www.google.com/intl/it/gmail/about/

- **Telegram:** utilizzato per la comunicazione tra i membri del gruppo.

  https://telegram.org/

4.2 Formazione

4.2.1 Scopo

Lo scopo di questa sezione è quello di definire le norme riguardanti il processo di formazione dei membri del gruppo *Yakuzaishi* e quindi lo studio delle tecnologie utilizzate per produrre i documenti e costruire il prodotto richiesto.

4.2.2 Aspettative

Le aspettative per il processo di formazione sono:

- Ottenere una buona conoscenza di LaTeX;
- Avere una buona familiarità con l'ambiente in cui si sviluppa il capitolato di interesse;
- Avere una buona familiarità con i vari linguaggi di programmazione, delle librerie e dei vari strumenti necessari per la realizzazione del prodotto software assegnato dal proponente.

4.2.3 Descrizione

Il processo di formazione di ogni membro del gruppo *Yakuzaishi* è ritenuto fondamentale, e ha il fine di colmare le mancate conoscenze di uno o più componenti del gruppo nei vari ambiti che copre il capitolato di interesse, in modo da avere una formazione uniforme all'interno del gruppo.

4.2.4 Istanziazione del processo

La formazione di ogni singolo componente del gruppo avviene in maniera autonoma, tramite documentazione trovata in rete e tramite materiale fornito dal proponente e dai docenti.

4.2.5 Metriche

Il processo di formazione non fa uso di metriche particolari.

4.2.6 Strumenti

Alcuni strumenti, librerie e framework necessitano di una ampia formazione, questi sono:

- **React:** libreria JavaScript$_G$ per la creazione di interfacce utente;
- **LaTeX:** linguaggio di markup per la creazione di testi;
- **GitHub:** Servizio di controllo di versione e hosting per progetti software;
- **Solidity:** Linguaggio di programmazione ad alto livello orientato agli oggetti per scrivere smart contract$_G$. Viene utilizzato su varie piattaforme blockchain$_G$, in particolare Ethereum$_G$.

A Standard ISO/IEC 12207

ISO/IEC 12207 è uno standard internazionale per i processi del ciclo di vita$_G$ del software.
Lo standard stabilisce i processi presenti nel ciclo di vita$_G$ del software, e per ciascuno di essi, le attività da svolgere e i risultati da produrre.
I processi sono divisi in tre categorie:

- **Processi primari:** comprendono le attività direttamente legate allo sviluppo del software;
- **Processi di supporto:** includono la gestione dei documenti e dei processi di controllo della qualità;
- **Processi organizzativi:** coprono gli aspetti manageriali e di gestione delle risorse.

A.1 Processi primari

A.1.1 Acquisizione

Questo processo ha lo scopo di ottenere il prodotto/servizio che soddisfa le necessità del cliente.
Il processo inizia con l'identificazione dei requisiti e termina con l'accettazione della fornitura.
È suddiviso nelle seguenti attività:

- Acquisition preparation;
- Supplier selection;
- Supplier monitoring;
- Customer acceptance.

A.1.2 Fornitura

Questo processo ha lo scopo di fornire al cliente il prodotto/servizio che soddisfa i requisiti concordati.
È suddiviso nelle seguenti attività:

- Proposal preparation;
- Contract;
- Planning;
- Execution and control;
- Review and evaluation;
- Release and completion.

A.1.3 Sviluppo

Questo processo ha lo scopo di sviluppare un prodotto software che indirizzi le esigenze del cliente.
Le attività del processo sono suddivise rispetto al ruolo dello sviluppatore e a quello del cliente e sono le seguenti:

- Requirements elicitation;
- System requirements analysis;
- System architecture design;

- Software requirements analysis;
- Software architecture design;
- Software construction (code and unit test);
- Software integration;
- Software testing;
- System integration;
- System testing;
- Software installation.

A.1.4 Esercizio

È svolto simultaneamente alla fase di Manutenzione.
Il processo ha lo scopo di mantenere operativo il sistema e di fornire il supporto agli utenti.
È suddiviso nelle seguenti attività:

- Operational use;
- Customer support.

A.1.5 Manutenzione

È svolto simultaneamente alla precedente fase di Esercizio.
Questo processo ha lo scopo di modificare il prodotto software dopo il suo rilasci per correggere i difetti, migliorare le sue prestazioni o altri attributi o adattarlo a cambiamenti nell'ambiente operativo.
È suddiviso nelle seguenti attività:

- Defect or request for change analysis;
- Change implementation;
- Review/acceptance of changes;
- Migration;
- Software withdraw.

A.2 Processi di supporto

A.2.1 Gestione della documentazione

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

A.2.2 Gestione della configurazione

Questo processo ha lo scopo di definire e mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto.

A.2.3 Gestione della qualità

Questo processo ha lo scopo di assicurare che tutti i prodotti di fase siano conformi con i piani e gli standard definiti.

A.2.4 Verifica

Questo processo ha lo scopo di confermare che ciascun prodotto o servizio realizzato da un processo soddisfi i requisiti specificati.
Il processo di Verifica deve essere integrato nei processi di Sviluppo, Fornitura e Manutenzione.

A.2.5 Validazione

Questo processo ha lo scopo di confermare che i requisiti siano rispettati quando uno specifico prodotto sia utilizzato nell'ambiente destinatario.

A.2.6 Revisione congiunta

Questo processo ha lo scopo di rivedere con le parti interessate, i processi eseguiti rispetto agli obiettivi definiti negli accordi e le cose da fare per assicurare lo sviluppo di un prodotto che soddisfi i requisiti concordati.
Le revisioni sono svolte durante l'intero ciclo di vita$_{\mathbf{G}}$, sia a livello di progetto che a livello tecnico.
La revisione congiunta è svolta tra gli stessi componenti del team, quando si revisiona un componente del prodotto, oppure tra fornitore e committente, quando si revisiona l'intero prodotto.

A.2.7 Audit

Questo processo ha lo scopo di determinare in maniera indipendente la conformità di prodotti e processi selezionati ai requisiti, piani e accordi.
L'attività di auditing è svolta da personale che non ha partecipato direttamente allo sviluppo dei prodotti, dei servizi o dei sistemi oggetto delle revisioni.

A.2.8 Risoluzione dei problemi

Questo processo ha lo scopo di assicurare che tutti i problemi individuati siano analizzati e risolti secondo trend riconosciuti.

A.2.9 Usabilità

Questo processo ha lo scopo di assicurare che siano prese in considerazione, ed opportunamente indirizzate, le considerazioni espresse dalle parti interessate relativamente alla facilità d'uso del prodotto finale da parte degli utenti a cui è rivolto, al supporto che ne riceverà, alla formazione, all'incremento della produttività, alla qualità del lavoro, all'accettazione del prodotto stesso.

A.2.10 Valutazione del prodotto

Questo processo ha lo scopo principale di assicurare, tramite esami e misure, che il prodotto soddisfi le necessità esplicite ed implicite degli utilizzatori del prodotto stesso.

A.3 Processi organizzativi

A.3.1 Gestione organizzativa

Questo processo ha lo scopo di organizzare, monitorare e controllare l'avvio e le prestazioni di un processo per il raggiungimento dei loro obiettivi in accordo con quelli di business dell'organizzazione. Il processo è stabilito da una organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti.

A.3.2 Gestione delle infrastrutture

Questo processo ha lo scopo di mantenere un'infrastruttura stabile ed affidabile, necessaria a supportare le prestazioni di qualsiasi processo.
L'infrastruttura può includere hardware, software, metodi, tools, tecniche, standard ed utilità per lo sviluppo, operatività o manutenzione.

A.3.3 Miglioramento

Questo processo ha lo scopo di stabilire, valutare, controllare e migliorare il ciclo di vita$_{\mathbf{G}}$ del software.

A.3.4 Risorse umane

Questo processo ha lo scopo di fornire all'organizzazione risorse umane adeguate e di mantenere le loro competenze consistenti con le necessità del business.

B Standard ISO/IEC 9126

ISO/IEC 9126 è uno standard internazionale per la valutazione della qualità software. Il gruppo *Yakuzaishi* ha deciso di fare riferimento a questo standard poiché considerato un caposaldo in materia di qualità. Questo standard permette di diffondere una comprensione comune degli obiettivi di un progetto software.

B.1 Modello della qualità del software

Il modello di qualità stabilito dallo standard si articola in sei caratteristiche principali, ognuna delle quali a sua volta presenta delle sottocaratteristiche, misurabili tramite delle metriche di qualità. Di seguito sono esposte le sei caratteristiche sopra citate:

- **Funzionalità:** insieme di attributi riguardanti un insieme di funzioni e le loro proprietà. Tali funzioni mirano a soddisfare requisiti stabiliti o implicitamente dedotti. Le sottocaratteristiche della funzionalità sono:
  - **Adeguatezza:** capacità del prodotto di fornire un insieme di funzioni in grado di svolgere compiti e soddisfare obiettivi prefissati;
  - **Accuratezza:** capacità del prodotto di fornire i risultati desiderati con la precisione richiesta;
  - **Interoperabilità:** capacità del prodotto di interagire ed operare con uno o più sistemi;
  - **Sicurezza:** capacità del prodotto di proteggere informazioni e dati monitorando gli accessi ad essi;
  - **Aderenza alle funzionalità:** grado di adesione del prodotto agli standard scelti dal gruppo, alle convenzioni e ai regolamenti;
- **Affidabilità:** insieme di attributi riguardanti la capacità del prodotto di mantenere un dato livello di performance sotto condizioni di esecuzione prestabilite. Le sottocaratteristiche dell'affidabilità sono:
  - **Maturità:** capacità del prodotto di evitare errori e risultati non corretti durante l'esecuzione;
  - **Tolleranza ai guasti:** capacità del prodotto di conservare il livello di prestazioni anche in caso di malfunzionamenti o di uso inappropriato del prodotto;
  - **Recuperabilità:** capacità di un prodotto di ripristinare il livello di prestazioni e di recupero delle informazioni rilevanti, a seguito di un malfunzionamento. Il periodo di inaccessibilità del prodotto a seguito di un errore è valutato proprio dalla recuperabilità;
  - **Aderenza all'affidabilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'affidabilità.
- **Efficienza:** insieme di attributi riguardanti il rapporto tra il livello delle prestazioni e la quantità di risorse usate durante la loro esecuzione, sotto condizioni prestabilite. Le sottocaratteristiche dell'efficienza sono:
  - **Comportamento rispetto al tempo:** capacità di un prodotto di fornire appropriati tempi di risposta e di elaborazione e quantità di lavoro eseguendo le funzionalità richieste in date condizioni di lavoro;
  - **Utilizzo delle risorse:** capacità di un prodotto di usare appropriati numero e tipo di risorse durante la fase di esecuzione sotto condizioni di utilizzo date;

  - **Aderenza all'efficienza:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'efficienza.

- **Usabilità:** insieme di attributi riguardanti lo sforzo necessario all'utilizzo del prodotto e la valutazione individuale su tale uso da parte di un insieme di utenti. Le sottocaratteristiche dell'usabilità sono:
  - **Comprensibilità:** facilità di comprensione dei concetti base del prodotto, per permettere all'utente di capire se il prodotto è appropriato;
  - **Apprendibilità:** facilità con cui un utente medio può comprendere il funzionamento del prodotto ed imparare ad usarlo;
  - **Operabilità:** misura della possibilità degli utenti di usare il prodotto in vari contesti e di adattarlo ai propri scopi;
  - **Attrattività:** misura della gradevolezza e dell'essere "attraente" del prodotto durante l'uso;
  - **Aderenza all'usabilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'usabilità.

- **Manutenibilità:** insieme di attributi riguardanti lo sforzo richiesto per apportare modifiche specifiche al prodotto. Le sottocaratteristiche della manutenibilità sono:
  - **Analizzabilità:** misura della difficoltà incontrata nel diagnosticare un errore nel prodotto;
  - **Modificabilità:** facilità di apportare modifiche al prodotto originale o di introdurre nuove funzionalità; per modifiche si intendono cambiamenti al codice, alla progettazione o alla documentazione;
  - **Stabilità:** capacità del prodotto di evitare effetti indesiderati a causa di modifiche;
  - **Provabilità:** capacità del prodotto di essere verificato;
  - **Aderenza alla manutenibilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti alla manutenibilità.

- **Portabilità:** insieme di attributi riguardanti la capacità del software di essere trasferito da un ambiente di esecuzione ad un altro. Le sottocaratteristiche della portabilità sono:
  - **Adattabilità:** capacità del prodotto di essere adattato per differenti ambienti operativi senza richiedere azioni specifiche diverse da quelle previste dal prodotto per quell'attività; include la scalabilità delle capacità interne del prodotto;
  - **Installabilità:** capacità del prodotto di essere installato in un dato ambiente;
  - **Coesistenza:** capacità di un prodotto di coesistere con altre applicazioni in ambienti comuni e condividere le risorse;
  - **Sostituibilità:** capacità di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente di sviluppo;
  - **Aderenza alla portabilità:** capacità del prodotto di aderire a standard e convenzioni relative alla portabilità.

B.2 Qualità interna

B.2.1 Metriche per la qualità interna

Le metriche interne si applicano al prodotto non eseguibile durante la progettazione e la codifica. Sono anche dette misure statiche. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale, poiché gli attributi interni influiscono su quelli esterni e quelli in uso. Le metriche interne permettono di individuare eventuali problemi che potrebbero inficiare la qualità finale del prodotto prima che sia realizzato il prodotto eseguibile.

B.3 Qualità esterna

B.3.1 Metriche per la qualità esterna

Le metriche esterne misurano i comportamenti del prodotto sulla base dei test, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti. Le metriche esterne sono scelte in base alle caratteristiche che il prodotto finale dovrà dimostrare durante la sua esecuzione.

B.4 Qualità in uso

La qualità in uso rappresenta il punto di vista dell'utente sul prodotto. Il livello di qualità in uso è raggiunto quando sono state conseguite sia la qualità esterna che quella interna.

B.4.1 Metriche per la qualità in uso

La qualità in uso permette di abilitare specificati utenti ad ottenere dati obiettivi con efficacia, produttività, sicurezza e soddisfazione.

- **Efficacia:** capacità del prodotto di consentire agli utenti di raggiungere gli obiettivi specificati con accuratezza e completezza;
- **Produttività:** capacità di consentire agli utenti di adoperare un'appropriata quantità di risorse rispetto all'efficacia ottenuta in uno dato contesto d'uso;
- **Soddisfazione:** capacità del prodotto di corrispondere alle aspettative degli utenti;
- **Sicurezza:** capacità del prodotto di raggiungere accettabili livelli di rischio di danni a persone, software, apparecchiature tecniche e all'ambiente d'uso.

C Metriche di qualità

C.1 Metriche per la qualità di processo

La presente sezione espone le metriche selezionate dal gruppo *Yakuzaishi* per misurare il raggiungimento degli obiettivi di qualità del processo.

Parametri per comprendere le metriche scelte:

- Budget At Completion (BAC): È il budget stimato in preventivo per il completamento del progetto;
- Actual Calendar Days (ACD): È il numero effettivo di giorni impiegati in un determinato periodo;
- Planned Calendar Days (PCD): È la durata preventivata di un determinato periodo;
- Number of Requirements Added (NRA): Numero di requisiti aggiunti in un determinato periodo;
- Number of Requirements Changed (NRC): Numero di requisiti modificati in un determinato periodo;
- Number of Requirements Removed (NRR): Numero di requisiti rimossi in un determinato periodo;
- Total Number of Initial Requirements (TNIR): Numero totale dei requisiti all'inizio di un determinato periodo.

C.1.1 MPC01: Earned Value (EV)

È il valore del lavoro effettuato fino ad un determinato periodo.

C.1.2 MPC02: Actual Cost (AC)

È il costo effettivamente sostenuto fino ad un determinato periodo.

C.1.3 MPC03: Planned Value (PV)

È il costo stimato per il completamento di un determinato periodo.

C.1.4 MPC04: Cost Variance (CV)

Metrica che indica, in percentuale, se il gruppo è regolare rispetto al budget prestabilito.
La formula adottata è la seguente:

$$CV = (\frac{EV}{AC} - 1) * 100$$

Se il valore risulta negativo si è fuori budget.

C.1.5 MPC05: Schedule Variance (SV)

Metrica che indica, in percentuale, se il gruppo è regolare rispetto i limiti temporali imposti.
La formula adottata è la seguente:

$$SV = (1 - \frac{ACD}{PCD}) * 100$$

Se il valore risulta negativo si è indietro rispetto i tempi preventivati.

C.1.6 MPC06: Estimated At Completion (EAC)

Revisione del valore stimato per la realizzazione del progetto in un determinato periodo, ossia il BAC rivisto allo stato corrente del progetto.
La formula adottata è la seguente:

$$EAC = AC + ETC$$

C.1.7 MPC07: Estimate To Complete (ETC)

Valore stimato del costo del lavoro rimanente in un determinato periodo.
La formula adottata è la seguente:

$$ETC = BAC - EV$$

C.1.8 MPC08: Requirements Stability Index (RSI)

Indice che misura la variazione dei requisiti nel tempo.
La formula adottata è la seguente:

$$RSI = (1 - \frac{NRA + NRC + NRR}{TNIR}) * 100$$

C.1.9 MPC09: Passed Tests

Indica la percentuale dei test che sono stati superati fino ad un determinato periodo.
La formula adottata è la seguente:

$$Passed\ Tests = \frac{Number\ of\ Passed\ Tests}{Tot.\ Number\ of\ Tests} * 100$$

C.1.10 MPC10: Metrics Satisfied

Indice che rappresenta la percentuale di metriche di qualità soddisfatte in un determinato periodo. La formula adottata è la seguente:

$$Metrics\ Satisfied = \frac{Number\ of\ Metrics\ Satisfied}{Tot.\ Number\ of\ Metrics} * 100$$

C.1.11 MPC11: Risks Found

Indica il numero di rischi incontrati in un determinato periodo.

C.2 Metriche per la qualità di prodotto

La presente sezione espone le metriche selezionate dal gruppo *Yakuzaishi* per misurare il raggiungimento degli obiettivi di qualità del prodotto.

C.2.1 MPD01: Indice di Gulpease

Indice che riporta il grado di leggibilità di un testo redatto in lingua italiana. La formula adottata è la seguente:

$$GULP = 89 + \frac{300 * (totale\ frasi) - 10 * (totale\ lettere)}{(totale\ parole)}$$

L'indice così calcolato può assumere valori tra 0 e 100:

- **GULP $<$ 80:** indica una leggibilità difficile per un utente con licenza elementare;
- **GULP $<$ 60:** indica una leggibilità difficile per un utente con licenza media;
- **GULP $<$ 40:** indica una leggibilità difficile per un utente con licenza superiore.

Per facilitare il calcolo di questo indice, il gruppo si è appoggiato al sito , copiando l'intero contenuto dei documenti in formato pdf escludendo la prima pagina e gli indici.

C.2.2 MPD02: Errori ortografici

Il controllo ortografico viene eseguito secondo quanto descritto in 3.1.4.6 del presente documento.

C.2.3 MPD03: Copertura requisiti obbligatori

Indice che misura in ogni istante la percentuale di requisiti obbligatori soddisfatti. La formula adottata è la seguente:

$$CROB = \frac{ROBC}{ROB} * 100$$

dove:

- **ROBC:** indica il numero di requisiti obbligatori coperti dall'implementazione;
- **ROB:** indica il numero complessivo di requisiti obbligatori.

C.2.4 MPD04: Copertura requisiti desiderabili

Indice che misura in ogni istante la percentuale di requisiti desiderabili soddisfatti. La formula adottata è la seguente:

$$CRD = \frac{RDC}{RD} * 100$$

dove:

- **RDC:** indica il numero di requisiti desiderabili coperti dall'implementazione;
- **RD:** indica il numero complessivo di requisiti opzionali.

C.2.5 MPD05: Copertura requisiti opzionali

Indice che misura in ogni istante la percentuale di requisiti opzionali soddisfatti. La formula adottata è la seguente:

$$CROP = \frac{ROPC}{ROP} * 100$$

dove:

- **ROPC:** indica il numero di requisiti opzionali accettati coperti dall'implementazione;
- **ROP:** indica il numero complessivo di requisiti opzionali accettati.

C.2.6 MPD06: Facilità di utilizzo

Numero di click necessari con cui l'utente raggiunge la funzionalità cercata.

C.2.7 MPD07: Versioni browser supportate

Percentuale di versioni di browser supportate dal prodotto. Calcolabile con la seguente formula:

$$BS = \frac{Bsup}{Btot} * 100$$

dove:

- **Bsup** indica il numero di browser in cui il prodotto può essere eseguito;
- **Btot** indica il numero complessivo di browser presi in considerazione.

Dove in **Btot** i browser presi in considerazione sono i seguenti:

- Chrome;
- Firefox;
- Microsoft Edge;
- Brave.

L'integrazione con Metamask$_G$ in Safari non è supportata, di conseguenza l'applicativo non funziona in tale browser.

C.2.8 MPD08: Solidity Statement Coverage

Viene utilizzato per calcolare il numero di istruzioni nel codice sorgente che sono state eseguite.
Lo scopo dello Statement Coverage è quello di coprire tutti i possibili percorsi, righe e istruzioni nel codice sorgente.
Il valore è stato calcolato per mezzo di solidity-coverage.

C.2.9 MPD09: Solidity Branch Coverage

Lo scopo del Branch Coverage è garantire che ogni condizione decisionale venga eseguita almeno una volta.
Il valore è stato calcolato per mezzo di solidity-coverage.

C.2.10 MPD10: Solidity Function Coverage

Indica la misura in cui le funzioni presenti nel codice sorgente sono coperte durante il test.
Tutte le funzioni presenti nel codice sorgente vengono testate durante l'esecuzione del test.
Il valore è stato calcolato per mezzo di solidity-coverage.

C.2.11 MPD11: Solidity Line Coverage

Il Line Coverage di un programma è il numero di righe eseguite diviso per il numero totale di righe. Vengono considerate solo le righe che contengono istruzioni eseguibili. Il valore è stato calcolato per mezzo di solidity-coverage.

C.2.12 MPD12: Frontend Statement Coverage

Viene utilizzato per calcolare il numero di istruzioni nel codice sorgente che sono state eseguite.
Lo scopo dello Statement Coverage è quello di coprire tutti i possibili percorsi, righe e istruzioni nel codice sorgente.
Il valore è stato calcolato per mezzo di Jest.

C.2.13 MPD13: Frontend Branch Coverage

Lo scopo del Branch Coverage è garantire che ogni condizione decisionale venga eseguita almeno una volta.
Il valore è stato calcolato per mezzo di Jest.

C.2.14 MPD17: Frontend Function Coverage

Indica la misura in cui le funzioni presenti nel codice sorgente sono coperte durante il test.
Tutte le funzioni presenti nel codice sorgente vengono testate durante l'esecuzione del test.
Il valore è stato calcolato per mezzo di Jest.

C.2.15 MPD15: Frontend Line Coverage

Il Line Coverage di un programma è il numero di righe eseguite diviso per il numero totale di righe. Vengono considerate solo le righe che contengono istruzioni eseguibili. Il valore è stato calcolato per mezzo di Jest.