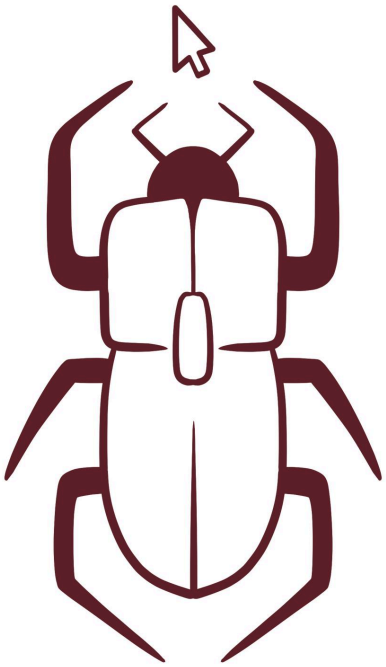

Norme di Progetto

v0.2.1

Registro delle Modifiche

Data	Versione	Descrizione	Redattore	Verificatore
09/01/2026	0.2.1	Correzioni minori e miglioramenti alla documentazione	Basso Kevin	Martinello Riccardo
02/01/2026	0.2.0	Arricchimento con standard industriali, norme di codifica dettagliate e integrazioni progetto-specifiche	Martinello Riccardo	Suar Alberto, Basso Kevin
28/12/2025	0.1.0	Rilascio iniziale con norme, processi e strumenti aggiornati	Martinello Riccardo	Suar Alberto, Basso Kevin
20/12/2025	0.0.1	Prima bozza iniziale	Martinello Riccardo	Suar Alberto, Basso Kevin

Indice

Indice tabelle

Introduzione

Scopo del documento

Il presente documento definisce le norme, le convenzioni, gli strumenti e i processi che i membri del gruppo Skarab Group sono tenuti a rispettare durante l'intero ciclo di vita del progetto Code Guardian. L'obiettivo è garantire uniformità, qualità e manutenibilità dei prodotti realizzati, facilitando la collaborazione interna e la comunicazione con l'esterno.

Scopo del prodotto

Code Guardian è una piattaforma software progettata per l'analisi automatizzata della qualità del codice e della documentazione di repository GitHub. Il sistema mira a supportare sviluppatori e manager fornendo metriche dettagliate e feedback semantico tramite agenti intelligenti.

Glossario e Riferimenti

I termini tecnici o ambigui sono definiti nel glossario di progetto (`RTB/glossario.yml`). Nel testo, i termini importanti possono essere evidenziati utilizzando la macro `#def`, questo permette di rendere la parola un link al glossario presente nel sito del progetto.

I riferimenti normativi includono:
- Regolamento del progetto didattico.
- Standard ISO/IEC 12207 per i processi del ciclo di vita del software. -Ultimo accesso: 2026/01/02.

Standard Industriali e Riferimenti

Il progetto adotta standard industriali riconosciuti per garantire qualità, conformità e best practices nel ciclo di vita del software. Di seguito i principali riferimenti:

- **IEEE 830 - Standard for Software Requirements Specifications**: Guida alla documentazione dei requisiti funzionali e non funzionali, inclusa la classificazione (obbligatori, desiderabili, opzionali) e l'uso di casi d'uso.

  -Ultimo accesso: 2026/01/02.

- **IEEE 1016 - Recommended Practice for Software Design Descriptions**: Fornisce linee guida per descrivere l'architettura software, inclusi diagrammi UML e design pattern (es. MVC, Singleton).

  -Ultimo accesso: 2026/01/02.

- **IEEE 829 - Standard for Software and System Test Documentation**: Definisce la struttura per piani di test, casi di test, procedure e report per unit, integration e system testing.

  -Ultimo accesso: 2026/01/02.

- **ISO/IEC 12207 - Software Life Cycle Processes**: Framework internazionale per processi di acquisizione, fornitura, sviluppo, operazione, manutenzione e disposal del software, supportando approcci iterativi e gestione del rischio.

  -Ultimo accesso: 2026/01/02.

Questi standard sono integrati nei processi di analisi, progettazione, verifica e documentazione per assicurare tracciabilità e qualità.

Processi Primari

Processo di Fornitura

Questo processo copre le attività necessarie per fornire il prodotto al committente.

- **Studio di fattibilità**: Analisi preliminare dei capitolati per valutare rischi, tecnologie e interesse del gruppo.
- **Pianificazione**: Definizione di scadenze, milestone e ripartizione dei compiti tramite strumenti di project management.

Processo di Sviluppo

Analisi dei Requisiti

Attività volta a comprendere e documentare le necessità degli stakeholder.

- **Casi d'uso**: Descrizione funzionale delle interazioni attore-sistema (formato tabellare con pre/post condizioni).
- **Classificazione Requisiti**: Requisiti funzionali, Requisiti non funzionali, Obbligatori, Desiderabili, Opzionali.

Progettazione

Definizione dell'architettura logica e tecnica del sistema.

- **UML**: Utilizzo di diagrammi di classi, sequenza e attività per modellare il sistema.
- **Design Pattern**: Adozione di pattern noti (es. MVC, Singleton, Factory) dove applicabile per risolvere problemi ricorrenti.

Codifica

Implementazione del software seguendo le norme di codifica definite (vedi Sezione "Norme di Sviluppo").

Processi di Supporto

Documentazione

Il ciclo di vita dei documenti prevede:

1. **Creazione/Modifica**: Redazione o aggiornamento del contenuto in formato source (Typst).
2. **Verifica**: Controllo di conformità (ortografia, stile, contenuti) da parte di un membro diverso dal redattore.
3. **Approvazione**: Validazione finale per il rilascio.

La documentazione è redatta in Typst per garantire consistenza tipografica e facilità di versione.

Gestione della Configurazione

Utilizzo di **Git** come sistema di controllo versione.

Branching Strategy

- `main`: Ramo stabile, contenente le versioni rilasciate o pronte al rilascio.
- `develop`: Ramo di integrazione principale.
- `feature/nome-feature`: Rami per lo sviluppo di nuove funzionalità.
- `fix/nome-fix`: Rami per la correzione di bug.

Inoltre si utilizzano branch diversi per la redazione dei diversi documenti, in modo che un gruppo assegnato ad un documento possa lavorarci senza interferire con il lavoro degli altri.

Conventional Commits

I messaggi di commit devono seguire il formato: `tipo(ambito): descrizione breve` Tipi ammessi: `feat`, `fix`, `docs`, `style`, `refactor`, `test`, `chore`.

Verifica e Qualità

La verifica ha l'obiettivo di accertare che i prodotti soddisfino i requisiti.

Metriche

Il gruppo adotta un sistema di metriche per monitorare processi e prodotti.

ID	Descrizione	Soglia / Obiettivo
MPC1 (SV)	Schedule Variance: differenza tra lavoro pianificato ed eseguito	>= 0
MPC2 (BV)	Budget Variance: differenza tra costo pianificato ed effettivo	>= 0
MPD1 (Gulpease)	Indice di Gulpease per documenti in italiano	> 40 (accettabile) > 60 (buono)
MPD2 (Ortografia)	Numero di errori ortografici rilevati	0
MPS1 (Coverage)	Code Coverage (Unit Test)	>= 80%
MPS2 (Req. Obb.)	Percentuale soddisfacimento requisiti obbligatori	100% al rilascio
MPS3 (Comprens.)	Complessità Ciclomatica media delle funzioni	<= 15

Table 1: Tabella delle metriche di progetto

Strategie di Verifica

- **Analisi Statica**: Review manuale del codice e uso di linter (ESLint per JS/TS, Pylint/Black per Python).
- **Analisi Dinamica**: Esecuzione della suite di test.
- **Test**:
  - **Unit Test**: Verifica di singole unità di codice.
  - **Integration Test**: Verifica delle interazioni tra moduli.
  - **System Test**: Verifica del sistema completo rispetto ai requisiti.

Software Quality Assurance SQA

La SQA monitora tutti i processi per garantire conformità agli standard (es. ISO 9001, ISO 25010). Include:

- **Politiche**: Definizione di procedure per ogni fase (requisiti, design, testing).
- **Audit**: Revisioni periodiche per identificare non conformità.
- **Attività**: Review di documenti, controllo qualità codice, Gestione Rischi.

Validazione

La validazione conferma che il prodotto soddisfi le esigenze degli utenti.

- **Test di Accettazione**: Verifica finale con stakeholder per requisiti non funzionali (es. usabilità, prestazioni).
- **Feedback Utente**: Raccolta di input durante demo o beta testing.

- **Allineamento Obiettivi**: Verifica rispetto agli obiettivi di progetto (es. automazione analisi qualità repository).

Processi Organizzativi

Gestione dei Processi

Il Responsabile di Progetto monitora l'avanzamento dei lavori e assegna i task. Strumenti di coordinamento: Jira, Slack, Telegram, Discord.

Ruoli di Progetto

- **Responsabile di Progetto**: Coordinamento generale e gestione stakeholder.
- **Amministratore**: Gestione configurazione, documentazione e repository.
- **Analista**: Analisi requisiti e modellazione sistema (casi d'uso, UML).
- **Progettista**: Architettura e design tecnico.
- **Programmatore**: Implementazione e sviluppo codice.
- **Verificatore**: Controllo qualità, testing e verifica conformità.

Approcci di Gestione Progetto (PMI)

- **Predictive**: Pianificazione dettagliata per requisiti stabili (sequenziale, con milestone fisse).
- **Adaptive**: Iterativo per requisiti incerti (es. Agile/Scrum, adatto a sviluppo incrementale).
- **Hybrid**: Combinazione di entrambi, bilanciando flessibilità e struttura.

Gestione Rischi

Identificazione, valutazione e mitigazione rischi (es. tecnologici, di schedule). Aggiornamenti periodici durante riunioni.

Gestione delle Riunioni

Per ogni riunione (interna o con esterni) viene redatto un **Verbale** che riporta:

- Data, ora, luogo e partecipanti.
- Ordine del giorno.
- Riassunto delle discussioni.
- Decisioni prese e task assegnati.

Norme di Sviluppo

Convenzioni Generali

- Utilizzare l'inglese per commenti, nomi di variabili e funzioni nel codice sorgente.
- Utilizzare l'italiano per la documentazione destinata agli utenti o capitolato (se richiesto).

TypeScript / JavaScript (Frontend & Script)

- Stile: `camelCase` per variabili/funzioni, `PascalCase` per Classi/Componenti.
- Indentazione: 2 spazi.
- Utilizzare `const` e `let`, evitare `var`.
- Componenti React: Funzionali con Hooks.

Python (Backend/AI Agents)

- Stile: Standard PEP 8.
- `snake_case` per variabili e funzioni.
- `PascalCase` per le Classi.
- Docstrings per documentare moduli, classi e funzioni.

Typst (Documentazione)

- Utilizzare la sintassi standard per headings e liste.
- Mantenere i file sorgente organizzati per capitoli o sezioni se il documento è esteso.

Node.js / JavaScript (Scripting/Backend)

- **Naming**: `camelCase` per variabili e funzioni, `PascalCase` per classi e componenti.
- **Indentazione**: 2 spazi.
- **Variabili**: Utilizzare `const` e `let`, evitare `var`.
- **Linting**: ESLint per controllo qualità codice; seguire regole standard (es. no-unused-vars, eqeqeq).
- **Documentazione**: JSDoc per moduli e funzioni (`@param`, `@returns`).
- **Esempio**:

```
/**
 * Analizza un repository GitHub.
 * @param {string} repoUrl - URL del repository.
 * @returns {Promise<Object>} Risultati dell'analisi.
 */
async function analyzeRepository(repoUrl) {
    // Implementazione
}
```

Strumenti Utilizzati

- **Redazione**: Visual Studio Code, Typst.
- **Gestione Versioni**: Git, GitHub.
- **Comunicazione**: Slack, Telegram, Discord.
- **Diagrammi**: Draw.io.
- **Project Management**: Jira.