Change Requests

MARK 2.0 Plus

Data	19/01/2026
Presentato da	Cerchia Giovanni (NF22500202) Medica Vincenzo (NF22500203)

Sommario

1 Introduzione

Il presente documento descrive le **Change Requests** previste per l'evoluzione del sistema nell'ambito dell'esame, partendo dallo stato iniziale descritto nel Report Iniziale e validato con System Testing pre-modifica.

1.1 Panoramica delle Change Requests

Le Change Requests definite sono:

- **CR01**: introduzione del calcolo di metriche di qualità del codice tramite la libreria Radon (CC e MI).
- **CR02**: introduzione di una GUI (Tkinter) per configurare ed eseguire analisi/cloning senza modifiche manuali a script e variabili.
- **CR03**: introduzione di una dashboard (matplotlib embedded in GUI) con grafici e statistiche aggregate per run.

Dipendenze tra CR

- **CR02** sfrutta/visualizza le informazioni introdotte da **CR01** (quando disponibili).
- **CR03** dipende logicamente da **CR01** (dati metriche) e da **CR02** (contenitore UI e gestione run).

1.2 Standard per la classificazione delle Change Requests

Secondo lo standard **ISO/IEC/IEEE 14764:2022**, una **Modification Request (MR)** è un item informativo che identifica e descrive cambiamenti proposti al prodotto/servizio.
Una MR può essere classificata secondo il seguente schema:

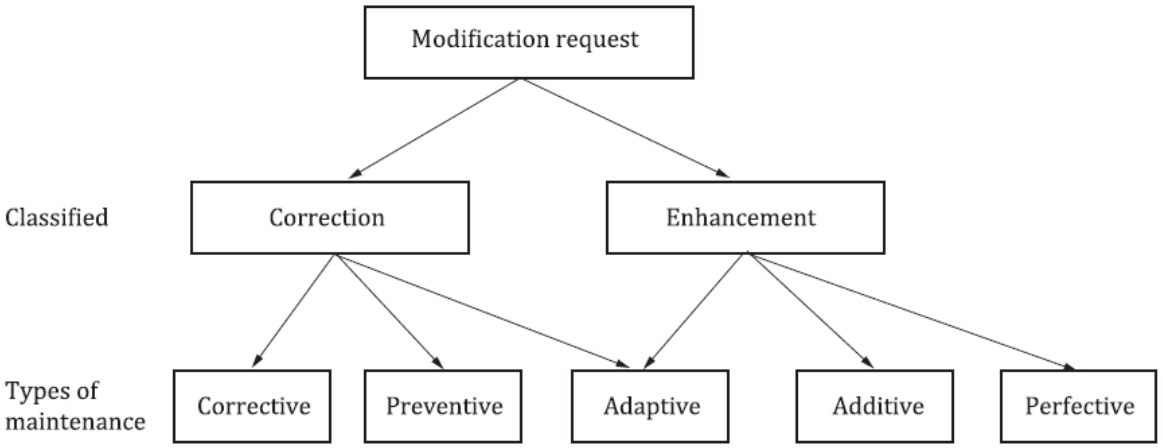

Figure 1 — Modification request

In particolare, una **Correction** è un cambiamento che risolve un problema e colma un gap per rendere il software operativo rispetto ai requisiti operativi definiti. Un **Enhancement** è un cambiamento che implementa un **nuovo requisito**; lo standard precisa che un enhancement **non** è una correction.
Dopo la classificazione principale, la MR può essere anche identificata con una *tipologia di manutenzione*, tra cui:

- **Adaptive maintenance** (collegabile a correction ed enhancements): modifica per mantenere il prodotto usabile in un ambiente cambiato/costante cambiamento;
- **Additive maintenance** (enhancement): modifica post-delivery per **aggiungere funzionalità o feature**;
- **Perfective maintenance** (enhancement): modifica per fornire miglioramenti agli utenti (understandability/informazione), e/o per migliorare performance/manutenibilità o altri attributi.

Nel presente documento, ogni CR riporta:

- **Classificazione** (Enhancement o Correction) e
- **tipologia di manutenzione** con una breve motivazione.

2 CR1 — Metriche di qualità del codice (Radon)

Motivazione

Nella versione attuale, MARK 2.0 produce risultati di classificazione ML (Producer/Consumer) ma non include indicatori quantitativi di qualità del codice del progetto. In contesti di analisi su dataset di repository, poter associare ai risultati anche misure come complessità e manutenibilità facilita:

- interpretazione dei risultati,
- confronto tra repository,
- selezione di progetti "più manutenibili" per analisi successive.

Situazione attuale (As-is)

- Il sistema effettua analisi e produce output (es. results.csv) relativi a librerie/keyword rilevate e classificazione.
- Non viene calcolata né esportata una metrica di complessità/manutenibilità a livello di progetto.

Situazione desiderata (To-Be)

Integrare nella pipeline il calcolo di metriche tramite **Radon**, in particolare:

- **Cyclomatic Complexity (CC)**
- **Maintainability Index (MI)**

Le metriche devono essere calcolate sui file sorgente analizzati e aggregate a livello di progetto.

Descrizione della modifica

- Durante l'analisi dei file, oltre all'estrazione di librerie/keyword, il sistema calcola CC e MI.
- I valori vengono raccolti e aggregati per progetto (ad es. medie; MI eventualmente pesato su SLOC se disponibile).
- Le metriche vengono salvate in output in modo consultabile e allineato alla run corrente.

Classificazione:

- **Enhancement**

Tipologia di manutenzione:

- **Additive**: aggiunge una nuova feature (calcolo e salvataggio di CC/MI) al tool.
- **Perfective**: arricchisce l'output con informazioni aggiuntive (CC e MI) che rendono l'analisi più completa, interpretabile e utile per l'utente.

3 CR2 — GUI per configurazione ed esecuzione analisi

Motivazione

L'uso del tool alla situazione attuale richiede interventi manuali su parametri hard-coded nello script `main.py`, rendendo meno pratico eseguire velocemente più run con configurazioni diverse e limitando l'accesso ad utenti non tecnici.

Situazione attuale (As-Is)

- L'esecuzione è orientata all'esecuzione tramite script, con configurazioni d'analisi impostate a livello di codice.
- La consultazione dei risultati è demandata ai file prodotti su disco.

Situazione desiderata (To-Be)

Introdurre una GUI (Tkinter) che consenta di:

- selezionare facilmente input (directory locale oppure CSV per cloning),
- configurare parametri di analisi (ruolo Producer/Consumer, opzioni regole, attivazione metriche),
- avviare l'analisi tramite un pulsante "Run",
- consultare i risultati direttamente dall'interfaccia.

Descrizione della modifica

La GUI organizza l'utilizzo del tool come un flusso guidato:

1. **Input selection**: scelta della sorgente (local folder / cloning da CSV).
2. **Configuration**: setting dei parametri principali (ruolo, regole, numero repo se cloning, abilita metriche).
3. **Execution**: avvio run senza bloccare l'interfaccia (gestione asincrona/thread).
4. **Results**: visualizzazione dei file prodotti e dei contenuti principali (tabella o viewer integrato).

Classificazione:

- **Enhancement**

Tipologia di manutenzione:

- **Additive**: aggiunge una nuova feature (GUI) per configurare ed eseguire il tool e per consultare i risultati, estendendo l'uso del sistema senza modificarne l'obiettivo funzionale principale.

4 CR3 — Dashboard di reportistica integrata nella GUI

Motivazione

Nella versione attuale i risultati sono principalmente consultabili tramite CSV. Per analisi su più repository, è utile una vista aggregata che fornisca insight rapidi e confrontabili, evitando interpretazione manuale di file.

Situazione attuale (As-Is)

- I risultati sono prodotti come file CSV per run/progetto, senza una vista aggregata visuale integrata.

Situazione desiderata (To-Be)

Aggiungere una sezione “Dashboard” nella GUI con grafici (matplotlib integrato) e statistiche aggregate per run.

Descrizione della modifica

Dopo l’esecuzione di una run, la GUI deve poter costruire una dashboard basata sugli output generati (CSV di risultati e metriche). La dashboard è pensata come una vista di sintesi per dataset di repository.

Dashboard previste

- **Analysis Overview**: numero e percentuale di progetti classificati come Producer, Consumer, Producer+Consumer e non classificati.
- **Code Quality Metrics Overview**: riepilogo delle metriche introdotte da CR01 (es. CC/MI medi e distribuzioni semplici).
- **Top ML keywords**: top 10 delle keyword ML (di training o inferenza) più rilevate nel dataset analizzato (conteggio occorrenze o progetti in cui compaiono).

Classificazione:

- **Enhancement**

Tipologia di manutenzione:

- **Additive**: aggiunge una nuova feature di reportistica.
- **Perfective**: fornisce miglioramenti informativi per gli utenti (viste aggregate, insight e visualizzazioni), migliorando la fruibilità dei risultati della run e supportando valutazioni complessive su dataset di repository..