Pre-Modifications System Testing Document

Versione 1.0

CodeSmile

Team Members:
Matteo Ercolino — 0522501462
Simone Silvestri — 0522501419

Repository: GitHub link

Anno Accademico 2024/2025

Indice

1 Introduzione

Il presente documento descrive e documenta l'attività di testing eseguita sul sistema **CodeSmile** nella sua versione iniziale, prima dell'introduzione delle modifiche richieste (Change Requests). L'obiettivo è:

- Validare il corretto funzionamento dei moduli principali (analisi del codice, rilevamento code smell, reportistica).
- Fornire una baseline per successivi test di regressione, successivi all'introduzione di nuove feature o migliorie.

Oltre ai test di **unit** e di **integrazione** già effettuati sui componenti core del sistema, qui ci concentriamo sui **test di sistema** (*system testing*) e **funzionali end-to-end**. Tutte le esecuzioni sono state effettuate tramite **interfaccia CLI**, poiché:

1. La *business logic* interna è la medesima anche per `GUI` e `Web App`.
2. L'esecuzione via CLI risulta più pratica per scripting e automazione.
3. Gli aspetti di *UI/UX* o di caricamento da browser verranno eventualmente trattati in *UI Testing* o *Acceptance Testing*.

1.1 Ambiente di Test

- **Sistema Operativo**: MacOS Sequoia 15.3.2
- **Python**: 3.11
- **Dipendenze**: installate tramite `requirements.txt` del progetto
- **Repository**: github.com/smell_ai

1.2 Tipologia di Testing

L'attività di testing è stata condotta con un approccio **black-box**, focalizzandosi sul comportamento osservabile del sistema senza considerare la struttura interna del codice (già coperta da test unitari e di integrazione). Come metodologia di progettazione, si è adottato il **Category Partition Method**, per identificare in modo sistematico le combinazioni più rilevanti di:

- *Tipo e numero di file* di input.
- *Parametri di configurazione del tool*.
- *Struttura* del progetto (directory, numero di progetti).

I test così definiti costituiscono inoltre una base solida per il **regression testing** dopo le modifiche pianificate, in modo da garantire che il comportamento pregresso non subisca regressioni.

2 Test di Sistema

In questo capitolo presentiamo la **suite di test di sistema** (o test funzionale "end-to-end") per il tool `CodeSmile`. Tali test sono stati progettati in base al *Category Partition Method*, concentrandosi sui principali **parametri** e **categorie** che caratterizzano l'esecuzione dell'analisi.

Nota su unit test e integration test. Prima di questa fase, sono stati già realizzati test di tipo **unitario** e di **integrazione** su tutti i componenti principali del sistema (motore di analisi, rule checker, moduli di estrazione, *etc.*). Di conseguenza, la suite attuale si pone come obiettivo la *validazione completa* del comportamento del tool a livello **funzionale** e di **sistema**.

Interfaccia di esecuzione. Sebbene `CodeSmile` offra molteplici interfacce (CLI, GUI e Web App), per il **System Testing** si è scelto di eseguire tutti i test **tramite CLI**, in quanto:

- **La logica interna è già coperta da test unitari e di integrazione** (sui componenti core).
- **La modalità CLI è più diretta e automatizzabile**.
- **GUI e Web App condividono la stessa logica di analisi (quindi la stessa logica funzionale)**.

2.1 Identificazione di Parametri e Categorie (Category Partition)

L'analisi del sistema `CodeSmile` ha permesso di individuare i seguenti parametri che influenzano l'esecuzione:

- **Parametro File (F)**
  - *Numero di File* (NF) = {0, 1, >1}
  - *Estensione File* (EF) = {`.py`, altro}
- **Parametro Smell (S)**
  - *Numero di Code Smells* (NCS) = {0, 1, >1}
  - *Tipo di Code Smell* (TCS) = {generico, API-specific, altro}
- **Parametro Struttura (ST)**
  - *Numero di Progetti* (NP) = {1, >1}
  - *Struttura Directory* (SD) = {semplice, annidata}
- **Parametro Configurazione Tool (CT)**
  - *Modalità di Esecuzione* (ME) = {`CLI`}
  - *Parallel* (EP) = {`true`, `false`}
  - *Walkers* (NW) = {<5, 5, >5}
  - *Resume* (RES) = {`true`, `false`}
  - *Errori* (ERR) = {file non leggibile, interruzione, nessun errore}

2.2 Scelte e Combinazioni (Weak Equivalence Class)

Sulla base delle categorie precedenti, *non* generiamo la combinazione per *GUI* o *Web App*, ma **solo** per `CLI`. Di seguito, presentiamo una selezione di test case (Weak Equivalence Class Testing) ritenuti **più rilevanti** in termini di copertura funzionale.

2.3 Test Frame: Analisi

Le tabelle seguenti riportano i **Test Case** con i relativi parametri e l'oracolo atteso. Tutti sono eseguiti in `CLI` (*ME=CLI*).

Tabella 2.1: Test Frame (Analisi) in modalità CLI

Test Case ID	Configurazione e Oracolo
TC_1	**Parametri:** NF = 0, EF = (n/a), NP = (n/a), SD = (n/a), NCS = 0, TCS = (n/a), ME = CLI, EP = false, NW = n/a, ERR = file mancante, RES = false **Oracolo**: Il tool restituisce un errore poiché il percorso di input/output è mancante.
TC_2	**Parametri:** NF >1, EF = .py, NP = 1, SD = annidata, NCS >1, TCS = API-specific, ME = CLI, EP = true, NW = >5, ERR = nessun errore, RES = false **Oracolo**: Il tool rileva correttamente code smells API-specific nei file Python in un progetto con struttura annidata.
TC_3	**Parametri:** NF = 1, EF = (n/a), NP = (n/a), SD = (n/a), NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = 5, ERR = file non leggibile, RES = false **Oracolo**: il tool restituisce un errore (file non leggibile) e non rileva code smells.
TC_4	**Parametri:** NF >1, EF = .py, NP >1, SD = annidata, NCS >1, TCS = generico, ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: il tool analizza più file Python con code smells generici su più progetti annidati, senza errori.
TC_5	**Parametri:** NF = 1, EF = (n/a), NP = 1, SD = semplice, NCS = 0, TCS = (n/a), ME = CLI, EP = false, NW = n/a, ERR = nessun errore, RES = false **Oracolo**: il tool non rileva code smells, perché il progetto (singolo file o directory) è vuoto.
TC_6	**Parametri:** NF >1, EF = .py, NP = 1, SD = annidata, NCS = 0, TCS = (n/a), ME = CLI, EP = false, NW = n/a, ERR = nessun errore, RES = false **Oracolo**: Il tool analizza correttamente più file Python in un progetto annidato **senza** rilevare code smells.
TC_7	**Parametri:** NF = 2, EF = .py, NP = 1, SD = annidata, NCS >1, TCS = API-specific, ME = CLI, EP = false, NW = n/a, ERR = nessun errore, RES = false **Oracolo**: Il tool rileva correttamente più code smells API-specific in un file Python in un progetto annidato.

(continua alla pagina successiva)

(continua dalla pagina precedente)

Test Case ID	Configurazione e Oracolo
TC_8	**Parametri:** NF >1, EF = .py, NP >1, SD = semplice, NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: Il tool analizza più file Python in più progetti a directory singola, **senza** rilevare code smells.
TC_9	**Parametri:** NF = 2, EF = .py, NP = 1, SD = semplice, NCS = 1, TCS = generico, ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: il tool rileva esattamente **un** code smell generico (singolo) in un file Python di un progetto semplice.
TC_10	**Parametri:** NF >1, EF = .py, NP = 1, SD = annidata, NCS >1, TCS = API-specific, ME = CLI, EP = true, NW = <5, ERR = nessun errore, RES = false **Oracolo**: il tool rileva code smells API-specific in un progetto annidato con più file; esecuzione parallela ridotta (NW <5).
TC_11	**Parametri:** NF >1, EF = .py, NP >1, SD = semplice, NCS = 2, TCS = API-specific, ME = CLI, EP = true, NW = >5, ERR = interruzione, RES = false **Oracolo**: durante l'analisi di più file Python (con code smells API-specific) su più progetti, il tool viene interrotto.
TC_12	**Parametri:** NF >1, EF = .py, NP = 1, SD = annidata, NCS >1, TCS = altro, ME = CLI, EP = true, NW = <5, ERR = nessun errore, RES = false **Oracolo**: Il tool analizza correttamente più file e rileva code smells sia generici sia API-specific in un progetto annidato.
TC_13	**Parametri:** NF >1, EF = .py, NP >1, SD = annidata, NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = >5, ERR = file non leggibile, RES = false **Oracolo**: Più file Python non sono leggibili nei progetti annidati. Il tool genera errore, nessun code smell rilevato.
TC_14	**Parametri:** NF = 2, EF = .py, NP = 1, SD = annidata, NCS = 1, TCS = API-specific, ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: il tool rileva **uno** smell API-specific in un progetto annidato, esecuzione parallela con NW=5.
TC_15	**Parametri:** NF = 2, EF = altro, NP = 1, SD = semplice, NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = n/a, ERR = nessun errore, RES = false **Oracolo**: il file con estensione non supportata (`.py` mancante) viene ignorato; nessun code smell rilevato.
TC_16	**Parametri:** NF >1, EF = .py, NP >1, SD = annidata, NCS >1, TCS = altro, ME = CLI, EP = true, NW = 5, ERR = interruzione, RES = false **Oracolo**: L'esecuzione si interrompe durante l'analisi di più file in più progetti annidati. Segnalato l'errore di interruzione.
TC_17	**Parametri:** NF >1, EF = .py, NP >1, SD = annidata, NCS = 0, TCS = (n/a), ME = CLI, EP = false, NW = n/a, ERR = file non leggibile, RES = false **Oracolo**: Alcuni file Python non sono leggibili nei progetti annidati; il tool segnala errore e non rileva smell.

(continua alla pagina successiva)

(continua dalla pagina precedente)

Test Case ID	Configurazione e Oracolo
TC_18	**Parametri:** NF >1, EF = .py, NP >1, SD = annidata, NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: il tool analizza più file Python in più progetti annidati, non rilevando alcun code smell.
TC_19	**Parametri:** NF >1, EF = .py, NP = 1, SD = semplice, NCS >1, TCS = altro, ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: Il tool rileva code smells generici e/o API-specific in un progetto a struttura semplice.
TC_20	**Parametri:** NF = 2, EF = .py, NP = 1, SD = annidata, NCS = 1, TCS = API-specific, ME = CLI, EP = true, NW = 5, ERR = nessun errore, RES = false **Oracolo**: il tool rileva esattamente **un** code smell API-specific in un file di un progetto annidato.
TC_21	**Parametri:** NF >1, EF = altro, NP >1, SD = annidata, NCS = 0, TCS = (n/a), ME = CLI, EP = true, NW = >5, ERR = nessun errore, RES = false **Oracolo**: I file con estensione non supportata, in più progetti annidati, vengono ignorati; nessun code smell rilevato.