Università degli Studi di Salerno
Corso di Ingegneria del Software: Tecniche Avanzate

Code Smile
Pre-Modification System Testing Document
Versione 1.0

Team:
Antonio Caiazzo
Emanuele Iovane
Salvatore Di Martino

Partecipanti:

Nome	**Matricola**
Antonio Caiazzo	NF22500205
Salvatore Di Martino	NF22500114
Emanuele Iovane	NF22500162

Scritto da:	Antonio Caiazzo, Salvatore Di Martino, Emanuele Iovane

Indice

1. Introduzione

Il presente documento descrive l'attività di testing svolta sulla versione iniziale del sistema **CodeSmile**, precedente all'applicazione delle Change Request.

L'obiettivo è verificare il **corretto funzionamento** dei moduli principali, dall'analisi del codice al rilevamento dei code smell fino alla generazione della reportistica, e stabilire una **baseline utile** per i futuri test di regressione che verranno condotti dopo l'introduzione di nuove funzionalità o ottimizzazioni.
La parte di analisi basata su modelli AI è **fuori dallo scope** del presente system test: ci concentriamo unicamente sull'analisi statica e sulla generazione dei report.

Oltre ai test di unità e di integrazione già effettuati sui componenti core, l'attenzione è qui posta sui **test di sistema** e sulle verifiche funzionali end-to-end. Tutte le esecuzioni funzionali sono state condotte tramite **interfaccia CLI**, scelta motivata dal fatto che: la business logic interna è la stessa anche per GUI e Web App, e che la CLI risulta più efficace per attività di scripting e automazione.

1.1. Ambiente di Testing

- **Sistema Operativo**: MacOS Tahoe 26.1
- **Python**: 3.11
- **Node.js**: 18+
- **Dipendenze**: installate tramite il file *requirements.txt*
- **Repository**: https://github.com/Antonio-Caiazzo/smell_ai

1.2. Tipologia di Testing

L'attività di testing è stata condotta adottando un approccio di tipo **black-box**, concentrandosi esclusivamente sul comportamento osservabile del sistema e lasciando da parte gli aspetti interni del codice, già coperti dai test unitari e di integrazione.
Per la progettazione dei test è stato utilizzato il **Category Partition Method**, che ha permesso di individuare in maniera sistematica le combinazioni più significative relative ai file di input, ai parametri di configurazione del tool e alla struttura dei progetti analizzati.
L'insieme dei test così costruito rappresenta inoltre una base solida per le successive attività di **regression testing**, con l'obiettivo di assicurare che le funzionalità già esistenti non subiscano regressioni dopo l'introduzione delle modifiche pianificate.

2. Test di Sistema

In questo capitolo descriviamo la suite di test di sistema (cioè i test funzionali "end-to-end") predisposta per il tool CodeSmile.

Osservazione su unit test e integration test: Prima dell'esecuzione dei test di sistema, sono già state condotte attività di verifica tramite unit test e integration test su tutti i principali componenti del tool (motore di analisi, rule checker, moduli di estrazione, ecc.). In aggiunta, è stato effettuato un assessment della suite di test esistente al fine di valutarne la completezza e l'affidabilità.
Nel corso di questa analisi è emerso il fallimento di uno dei test già presenti: ciò ha reso necessario un approfondimento mirato per individuare la causa dell'errore e procedere con la relativa correzione. L'intero processo di investigazione e risoluzione del problema è documentato nel
Pre-Modification Test Incident Report, disponibile al seguente link: Pre-Modification Test Incident Report.pdf

La suite di test di sistema qui presentata ha quindi l'obiettivo di completare la validazione complessiva del comportamento di CodeSmile, verificandone il corretto funzionamento sia dal punto di vista funzionale sia a livello sistemico.

2.1. Individuazione dei Parametri e delle Categorie (Category Partition)

Il criterio adottato è il **Category Partitioning**: individuiamo i principali parametri che caratterizzano gli scenari di utilizzo di CodeSmile e, per ciascuno, definiamo le categorie logiche che useremo in seguito per derivare le scelte e i Test Frame.

Di seguito i parametri sono raggruppati per "tipo di riferimento".
In questa sezione **descriviamo solo i parametri e i loro casi concettuali**, senza ancora elencare formalmente le singole scelte.

2.1.1. Parametri di Input (File)

Questi parametri descrivono le caratteristiche dei file forniti in input al tool.

- **Numero di File Input (NF):** Determina la cardinalità degli input analizzabili, specificando se l'operazione coinvolge una singola unità oppure un set multiplo di file.
- **Estensione del File (EF):** Specifica il formato del documento oggetto di analisi, distinguendo tra codice sorgente Python (.py) e altre tipologie.

2.1.2. Parametro Struttura del Progetto

Questi parametri descrivono come sono organizzati i progetti su cui viene eseguita l'analisi.

- **Numero dei Progetti (NP):** Definisce l'ambito dell'analisi, permettendo la scelta tra la valutazione di un singolo progetto o l'elaborazione di più progetti simultanei.
- **Struttura del Progetto (SP):** Analizza la complessità dell'alberatura delle cartelle, indicando se la directory è organizzata in modo lineare (singola) o gerarchico (multi-livello).

2.1.3. Parametro Smell

Riguarda il contenuto di debito tecnico nei file Python analizzati:

- **Numero di Code Smells (NCS):** Quantifica il totale delle anomalie (smell) individuate all'interno del file esaminato.
- **Tipo di Code Smell (TCS):** Classifica la natura dell'anomalia riscontrata, differenziando ad esempio tra pattern generici e quelli specifici relativi alle API.

2.1.4. Parametri di Configurazione del Tool

Questi parametri descrivono come viene configurato ed eseguito CodeSmile, indipendentemente dal contenuto dei file.

- **Modalità di Esecuzione (ME):** Indica attraverso quale interfaccia l'utente utilizza il tool. Nel documento distinguiamo esplicitamente fra: **modalità Web** (WebApp / gateway FastAPI), **modalità CLI** (riga di comando). Questo permette di verificare la coerenza del comportamento del sistema nelle due modalità di interazione principali (Web e CLI), che condividono la logica di analisi ma hanno canali di invocazione differenti.
- **Esecuzione Parallela (EP):** Flag di controllo che abilita l'elaborazione concorrente, attivando più walkers per l'analisi.
- **Numero di Walkers (NW):** Imposta la quantità numerica di agenti (walkers) impiegati attivamente nel processo di analisi.
- **Resume (RES):** Funzionalità di ripristino che consente di riavviare un'analisi multi-progetto interrotta, riprendendo il processo esattamente dall'ultimo progetto registrato nel log di esecuzione.
- **Percorso di Output (OUT):** Definisce il percorso di output dell'analisi condotta dal sistema, e se esso sia accessibile o meno dal sistema.

2.1.5. Parametri Backend WebApp

Questi parametri descrivono lo stato del backend quando CodeSmile viene utilizzato tramite la WebApp/gateway FastAPI. Sono rilevanti solo per la modalità di esecuzione Web.

- **Disponibilità backend (DB):** Rappresenta lo stato di disponibilità dei servizi backend necessari all'analisi. Serve a distinguere fra situazioni in cui tutti i servizi richiesti sono raggiungibili e situazioni in cui uno o più servizi risultano indisponibili (es. down, non raggiungibili, non configurati).
- **Esito Gateway (EG):** Rappresenta l'esito delle chiamate effettuate dalla WebApp verso il backend (gateway HTTP/REST).

Questi parametri saranno usati per modellare e testare il comportamento della WebApp in presenza di problemi lato backend.

2.2.Scelte e Combinazioni

La funzionalità core di CodeSmile risiede nell'analisi della codebase finalizzata all'identificazione dei code smell. Pertanto, l'implementazione efficace di tale meccanismo richiede l'integrazione e la valutazione congiunta dei parametri definiti nella fase preliminare.

2.2.1. Parametri di Input

CATEGORIA	SCELTE
Numero di File Input (NF)	1. 0 [ERRORE] 2. 1 3. >1
Estensione del File (EF)	1. Almeno un file Python (.py) 2. Nessun File .py [ERRORE]

2.2.2. Parametro Struttura del progetto

CATEGORIA	SCELTE
Numero dei Progetti (NP)	1. 1 2. >1
Struttura del Progetto (SP)	1. Struttura singola (directory lineare) 2. Struttura annidata (più livelli) 3. Struttura con sottocartelle inaccessibili [ERRORE]

2.2.3. Parametro Smell

CATEGORIA	SCELTE
Numero di Code Smells (NCS)	1. 0 2. 1 [proprietà SMELL] 3. >1 [proprietà SMELL]
Tipo di Code Smell (TCS)	1. Generico [se SMELL] 2. Specifico per API [se SMELL] 3. Altro/misto [se SMELL]

2.2.4. Parametri di Configurazione del Tool

CATEGORIA	SCELTE
Modalità di Esecuzione (ME)	1. Modalità WebApp [proprietà WEBAPP] 2. Modalità CLI [proprietà CLI]
Esecuzione Parallela (EP)	1. True – esecuzione con parallelismo attivo (più walkers) [proprietà PLL] [se CLI] 2. False – esecuzione sequenziale
Numero di Walkers (NW)	1. ≤ 0 [ERRORE] 2. < 5 [se PLL] 3. 5 (default) [se PLL] 4. > 5 [se PLL]
Resume (RES)	1. True – resume abilitato (ripresa di analisi multiprogetto da log esistente) [se CLI] 2. False – resume disabilitato (analisi sempre da zero) [se CLI]

Percorso di Output (OUT)	1. Percorso di Output funzionante/corretto/accessibile 2. Percorso non accessibile [ERRORE]

2.2.5. Parametri Backend WebApp

CATEGORIA	SCELTE
Disponibilità Backend (DB)	1. tutti i servizi backend necessari sono online [se WEBAPP] 2. almeno un servizio backend necessario (es. analisi statica) è offline o non raggiungibile [ERRORE]
Esito Gateway (EG)	1. successo (HTTP 2xx, richiesta completata correttamente) [se WEBAPP] 2. errore di input/validazione (HTTP 4xx) [ERRORE] 3. errore infrastrutturale o timeout (HTTP 5xx / timeout) [ERRORE]

2.3. Test Frame

Il criterio di copertura è applicato in modo da generare test funzionalmente coerenti e il più possibile rappresentativi. I casi di test, completi dei relativi oracoli, sono stati derivati a partire dalle categorie individuate, selezionando valori ammissibili per ciascuna di esse ed esplicitando, tramite la notazione *Alias0*, le scelte non applicabili (ad esempio, l'omissione del conteggio degli smell NCS quando l'estensione del file non corrisponde a un file Python, EF2).
Le combinazioni sono state costruite scegliendo, per ogni parametro, almeno un valore rappresentativo per ciascuna scelta rilevante, imponendo il vincolo di una sola sorgente di errore per test frame, così da evitare scenari con errori multipli difficili da interpretare.
Sono state inoltre rispettate le dipendenze tra parametri (ad esempio, disattivando la configurazione dei walkers quando il parallelismo è off, o valorizzando i parametri di backend solo in modalità Web), scartando le combinazioni che violano tali vincoli.
Nel category partition non sono state combinate tutte le scelte possibili quando una scelta copre funzionalmente un'altra e la differenza riguarda unicamente struttura o numerosità dell'input, mentre è stata mantenuta la combinazione completa per i parametri di configurazione del tool, al fine di esplorare sistematicamente lo spazio delle configurazioni.
Per la web app sono stati inoltre definiti specifici casi di test end-to-end, mirati a verificare i principali scenari di errore e un caso positivo rappresentativo del flusso nominale, così che ogni test case risultante rappresenti una combinazione consistente dal punto di vista funzionale.

Test Case ID	Test Frame	Oracolo atteso
TC_01	NF1, EF0, NP1, SP1, NCS0, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool segnala che non ci sono file in input da analizzare e non esegue alcuna analisi.
TC_02	NF2, EF1, NP1, SP3, NCS0, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool segnala un errore sulla struttura del progetto e non analizza le sottocartelle inaccessibili.
TC_03	NF2, EF1, NP1, SP1, NCS0, TCS0, ME2, EP1, NW1, RES2, OUT1, DB0, EG0	Il tool segnala che il numero di walkers è non valido (<= 0) e non avvia l'analisi.
TC_04	NF2, EF1, NP1, SP1, NCS0, TCS0, ME2, EP2, NW0, RES2, OUT2, DB0, EG0	Il tool segnala che il percorso di output è mancante o non accessibile
TC_05	NF2, EF2, NP1, SP1, NCS0, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool segnala che non ci sono file python da analizzare e non esegue alcuna analisi.
TC_06	NF3, EF1, NP1, SP1, NCS1, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool completa l'analisi del singolo progetto, e nessun code smell è individuato.
TC_07	NF3, EF1, NP2, SP1, NCS1, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool completa l'analisi multi progetto e nessun code smell è individuato
TC_08	NF3, EF1, NP1, SP2, NCS1, TCS0, ME2, EP2, NW0, RES2, OUT1, DB0, EG0	Il tool completa l'analisi singolo progetto, con una struttura annidata delle directory e nessun code smell è individuato

TC_09	NF2, EF1, NP1, SP1, NCS2, TCS1, ME2, EP2, NW0, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva un code smell di tipo generico.
TC_10	NF2, EF1, NP1, SP1, NCS2, TCS2, ME2, EP2, NW0, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva un code smell di tipo API-Specific.
TC_11	NF2, EF1, NP1, SP1, NCS3, TCS1, ME2, EP2, NW0, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva più code smell di tipo generico.
TC_12	NF2, EF1, NP1, SP1, NCS3, TCS2, ME2, EP2, NW0, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva più code smell di tipo API-Specific.
TC_13	NF2, EF1, NP1, SP1, NCS3, TCS3, ME2, EP2, NW0, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva più code smell di tipo misto.
TC_14	NF2, EF1, NP1, SP1, NCS2, TCS1, ME2, EP1, NW2, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva un code smell di tipo generico, tramite analisi parallela, e con numero di walkers < 5.
TC_15	NF2, EF1, NP1, SP1, NCS2, TCS1, ME2, EP1, NW3, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva un code smell di tipo generico, tramite analisi parallela, e con numero di walkers = 5.
TC_16	NF2, EF1, NP1, SP1, NCS2, TCS1, ME2, EP1, NW4, RES1, OUT1, DB0, EG0	Il tool completa l'analisi e rileva un code smell di tipo generico, tramite analisi parallela, e con numero di walkers > 5.
TC_17	NF2, EF1, NP1, SP1, NCS1, TCS0, ME1, EP2, NW0, RES0, OUT1, DB1, EG1	I servizi del backend della web app sono raggiungibili, e l'analisi è completata correttamente.
TC_18	NF2, EF1, NP1, SP1, NCS0, TCS0, ME1, EP2, NW0, RES0, OUT1, DB2, EG0	Non tutti i servizi del backend della web app sono raggiungibili.
TC_19	NF2, EF1, NP1, SP1, NCS0, TCS0, ME1, EP2, NW0, RES0, OUT1, DB1, EG2	I servizi del backend della web app sono raggiungibili, ma ci sono errori di validazione della richiesta.
TC_20	NF2, EF1, NP1, SP1, NCS0, TCS0, ME1, EP2, NW0, RES0, OUT1, DB1, EG3	I servizi del backend della web app sono raggiungibili, ma ci sono errori infrastrutturali o timeout.