UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

MASTER TEST PLAN

Ingegneria del Software: Tecniche Avanzate - Master Test Plan

Luigi Auriemma

Matricola: NF22500161

Ivan Chiello

Matricola: NF22500167

Repository: `https://github.com/LuigiAuriemma/smell_ai.git`

Anno Accademico 2025-2026


sesa
lab
SOFTWARE ENGINEERING
SALERNO

Indice

CAPITOLO 1

Introduzione

Questo documento definisce le strategie pianificate per il di testing di CodeSmile. Il piano di testing definisce i diversi livelli di test: test di unità, di integrazione, di sistema ed E2E, con l'obiettivo di garantire la qualità del software considerando anche le nuove funzionalità descritte nel documento Change Requests.
Verranno escluse dal perimetro di testing le funzionalità strettamente legate al modello di intelligenza artificiale e alla creazione dei dataset.

CAPITOLO 2

Strategie di Testing

2.1 Testing di unità

La fase di testing di unità è finalizzata alla verifica dei singoli componenti di CodeSmile, con l'intento di isolare e validare il comportamento di classi e moduli critici.La metodologia adottata per il testing di unità è di tipo white-box, focalizzata principalmente sui criteri di line coverage e branch coverage. L'organizzazione della suite di test ricalca fedelmente l'architettura del progetto: la directory dei test di unità è strutturata in sottocartelle che rispecchiano la gerarchia dei package, garantendo così una tracciabilità immediata tra il codice sorgente e i relativi casi di test.

2.2 Testing di integrazione

La fase di testing di integrazione ha lo scopo di verificare la comunicazione tra i sottosistemi di CodeSmile. L'attività si concentra sulla validazione della pipeline di elaborazione completa: dall'input utente (via CLI o Web) fino alla generazione dei risultati, attraversando i moduli cruciali come l'Inspector e il RuleChecker. Nello specifico, verrà esaminata la corretta trasmissione dei dati dai moduli di analisi statica fino ai moduli preposti al contatto con l'utilizzatore.

2.3 Testing di sistema

Il testing di sistema verrà eseguito secondo un approccio black-box. La metodologia di riferimento per la progettazione dei casi di test è il Category Partition Method, che permette di individuare e definire sistematicamente le combinazioni di input. Tale strategia dovrà ovviamente coprire i cambiamenti, definiti nelle change requests, apportati al sistema.

2.4 Testing E2E per la WebApp

Il testing End-to-End (E2E) ha il compito di validare la WebApp nel suo complesso, simulando percorsi di utilizzo reali per garantire la coerenza tra tutti i sottosistemi. I test si definiscono sui percorsi principali quali ad esempio: la gestione della navigazione e la generazione dei report. I criteri di successo richiedono che l'intero ciclo si svolga senza interruzioni, garantendo la completa correttezza dello strumento. L'intera procedura è automatizzata tramite l'ausilio Cypress.

CAPITOLO 3

Criteri di Accettazione

3.1 Criteri di Copertura dei Test

3.1.1 Funzionalità incluse nel Testing

L'attività di testing coprirà le componenti architetturali impattate dalle modifiche evolutive (CR01 e CR02) e le funzionalità core necessarie al loro funzionamento. Nello specifico:

- Moduli preposti all'esecuzione dell'Analisi Statica (Core/CLI): Verranno testati i moduli all'interno del package components e code_extractor , oggetto di modifica per l'implementazione della generazione del Call Graph (CR01).
- Web Application: Saranno oggetto di test i componenti riguardanti la versione WebApp di CodeSmile, con particolare attenzione al nuovo modulo di visualizzazione dei grafi (CR02).

3.1.2 Funzionalità escluse dal Testing

Sono escluse dal perimetro di questo piano:

- Modulo AI Sperimentale: Tutte le componenti legate al rilevamento degli smell tramite LLM, situate nelle directory finetuning, data_preparation e il microservizio webapp/services/aiservice.
- Generazione Dataset: Gli script e le logiche per la creazione, il bilanciamento e l'iniezione di smell nei dataset sono esclusi dalle attività di validazione funzionale del sistema.

3.2 Criteri e Valori di Accettazione

Il successo della versione aggiornata di CodeSmile deve garantire i seguenti criteri:

- Copertura del Codice: Raggiungimento del 70% di line coverage e branch coverage nei package core modificati o creati.
- Validazione Funzionale: Il sistema deve rispondere correttamente alle modifiche apportate e definite nelle change requests.
- Stabilità: Non devono essere presenti bug bloccanti nei processi di testing.
- Tutti i casi di test precedenti devono continuare a passare con successo, garantendo che l'introduzione del Call Graph non abbia introdotto regressioni nelle funzionalità di analisi esistenti.