UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

PRE-MODIFICATION TEST INCIDENT REPORT

Ingegneria del Software: Tecniche Avanzate - Pre-Modification Test Incident Report

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

Il presente documento costituisce l'**Incident Report** relativo alla fase di validazione preliminare del progetto **CodeSmile**. In questo documento vengono analizzate le problematiche identificate durante l'esecuzione della suite di test esistente, effettuata preventivamente rispetto all'introduzione delle nuove modifiche definite nel documento Change Requests. Nell'ambito di questa analisi vengono pertanto presi in esame e coperti tutti i livelli di testing.

1.1 Obiettivi del Documento

L'obiettivo primario è fornire una mappatura chiara delle eventuali criticità individuate durante questa fase "Pre-Modifiche". Questa attività di revisione e correzione preliminare è propedeutica al miglioramento della qualità complessiva del sistema e all'aumento dell'affidabilità della suite di test, garantendo una base solida per gli sviluppi futuri.

CAPITOLO 2

Incident Report

In questa sezione vengono dettagliati gli incidenti specifici affrontati durante la fase di validazione. Per ogni anomalia viene riportata la descrizione dell'errore, l'analisi delle cause e l'intervento correttivo applicato.

2.1 Test Incident n.1

Componente: Test di Unità - `test_project_analyzer.py`
Severità: Minore
Data: 23 Gennaio 2026

2.1.1 Descrizione del Problema

Durante l'esecuzione della suite di unit test, il caso di test `test_analyze_project_empty_directory` ha riportato un fallimento, interrompendo l'esecuzione con un'eccezione di tipo `ValueError`.
Il log dell'errore riportato è il seguente:

```
FAILED test/unit_testing/components/test_project_analyzer.py::
   test_analyze_project_empty_directory
```

```
ValueError: The project '.../mock_project_path' contains no Python
    files.
```

Questa versione del test era stata progettata basandosi sull'assunzione che, in assenza di file da analizzare, il metodo 'analyze_project' avrebbe completato la sua esecuzione restituendo un valore intero pari a '0' (indicante che nessun "code smell" era stato trovato). Di conseguenza, il codice di test invocava direttamente il metodo e tentava di asserire che il valore di ritorno fosse zero ('assert total_smells == 0').
Tuttavia, l'analisi del codice in 'components/project_analyzer.py' ha rivelato che il metodo interroga 'FileUtils.get_python_files' e, qualora la lista dei file risulti vuota, interrompe immediatamente il flusso operativo sollevando esplicitamente un'eccezione 'ValueError' con il messaggio "contains no Python files".
Il test falliva non perché i valori non corrispondessero, ma perché l'esecuzione veniva interrotta da un'eccezione non gestita ('ValueError') prima ancora di raggiungere l'istruzione di asserzione. Il di test interpretava questo evento come un errore imprevisto nel codice, piuttosto che come un comportamento corretto da validare.

2.1.2 Risoluzione Tecnica

Per correggere il test e allinearlo alle specifiche reali del software, la logica di verifica è stata aggiornata. Invece di aspettarsi un valore di ritorno numerico, il test ora utilizza 'pytest.raises(ValueError)'. Questo permette al test di:

- Anticipare il sollevamento dell'eccezione.
- Catturarla correttamente senza far fallire il test.
- Verificare che il tipo di errore e il messaggio associato siano quelli previsti.

Di seguito il dettaglio della modifica al codice:

Codice Errato: Il test falliva nel mocking e si aspettava un valore numerico.

```
# Esecuzione del metodo
total_smells = project_analyzer.analyze_project(mock_project_path)
# Asserzione
assert total_smells == 0
```

Codice Corretto: Il test ora inietta correttamente il mock e gestisce l'eccezione attesa.

```
# Esecuzione del metodo con gestione aspettativa eccezione
with pytest.raises(ValueError, match="contains no Python files"):
    project_analyzer.analyze_project(mock_project_path)
```

2.1.3 Esito

Questa modifica conferma che il comportamento del sistema, che tratta un progetto vuoto come una condizione di errore bloccante e non come un risultato valido "a zero", è intenzionale e correttamente implementato.
A seguito della modifica, il test `test_analyze_project_empty_directory` viene eseguito con successo.

2.2 Test Incident n.2

Componente: Integration Test - `webapp/integration_tests/test_gateway_to_model.`
Severità: Minore (Flaky Test)
Data: 26 Gennaio 2026

2.2.1 Descrizione del Problema

Durante l'esecuzione (Regression Testing per CR-02) dei test di integrazione della Webapp, il caso di test `test_gateway_to_ai_analysis_no_smell` ha fallito. Il test invia uno snippet di codice vuoto/dummy al servizio AI e si aspetta uno smell di default.

```
E  AssertionError: assert ...
E  Differing items:
E  {'smells': [{'smell_name': 'PyTorch Call Method Misused'}]} !=
   {'smells': [{'smell_name': 'Hyperparameter Not Explicitly Set
      '}]}
```

Il fallimento è dovuto alla natura non deterministica del modello LLM sottostante. Sebbene l'infrastruttura di gateway e comunicazione servizi abbia funzionato correttamente, il contenuto semantico della risposta è variato rispetto all'aspettativa hardcoded nel test ("PyTorch Call Method Misused" vs "Hyperparameter Not Explicitly Set").

2.2.2 Risoluzione Tecnica

È stata modificata l'asserzione del test per renderla resiliente alle variazioni del modello. Invece di verificare l'esatta stringa dello "smell name" (che è soggetta alla creatività probabilistica dell'LLM), il test è stato rifattorizzato per validare:

- Lo status code HTTP (200 OK).
- Il campo 'success' (True).
- La struttura dei dati ritornati (presenza della lista 'smells'), garantendo il rispetto del contratto dell'interfaccia API.

Questa modifica disaccoppia il test di integrazione dell'infrastruttura (Gateway) dalla varianza del componente di intelligenza artificiale.

2.2.3 Esito

L'incidente è stato classificato come "Flaky Test" dovuto alla variabilità del modello AI. Dopo la modifica, il test è stabile e deterministico, garantendo che future esecuzioni della suite di regressione non falliscano per falsi positivi legati al contenuto generativo.