UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Esame di Ingegneria, Gestione ed Evoluzione del Software

PROGETTO CODESMILE

Impact Analysis

TEAM MEMBERS

Dario Mazza - 0522501553

Nicolò Delogu - 0522501556

REPOSITORY

`https://github.com/xDaryamo/smell_ai`

VERSIONE 0.1

-

Anno Accademico 2024-2025

1 Descrizione CR1

La presente proposta di modifica si presta alla trasformazione del codice sorgente, adottando un design orientato agli oggetti (OO).

Comportamento attuale:
Allo stato attuale, il sistema è implementato con un approccio procedurale, il che comporta una scarsa modularità, una difficile estensione per nuovi code smells e un'elevata probabilità di impatti imprevisti in caso di modifiche al codice.

Proposta di modifica:
Rifattorizzare il sistema con un design orientato agli oggetti, introducendo classi che incapsulino le funzionalità attuali. L'obiettivo è suddividere le funzionalità in unità distinte e riutilizzabili, garantendo una migliore organizzazione e separazione delle responsabilità.

Benefici previsti:
L'implementazione della modifica dovrebbe portare ai seguenti benefici:

- Miglioramento della modularità e della manutenibilità del codice.
- Semplificazione della struttura del progetto.
- Semplificazione nell'aggiunta di nuovi code smells al sistema.
- Incremento dell'affidabilità del testing grazie a funzionalità ben incapsulate.

1.1 Analisi di Impatto

1.1.1 Starting Impact Set (SIS)

Nel caso di un refactoring completo del sistema verso un'architettura orientata agli oggetti, tutte le componenti che attualmente gestiscono le funzionalità principali in modo procedurale verranno direttamente impattate. Le aree identificate includono:

- *components*: contiene le componenti del tool finalizzate all'ispezione e al recupero di progetti;
- *controller*: si occupa di gestire sia l'interfaccia utente sia il flusso di controllo del tool;
- *cs_detector*: il punto focale del sistema di rilevamento dei code smells, diviso in due sotto-moduli principali:
  - **`code_extractor`**: gestisce l'estrazione delle strutture dal codice;
  - **`detection_rules`**: contiene le regole di rilevamento utili all'analisi;
- *general_output*: contiene un semplice sistema di reporting e di conteggio degli smells individuati;

1.1.2 Candidate Impact Set (CIS)

Non sono state individuate componenti che verranno impattate indirettamente dalle modifiche, poiché il refactoring è focalizzato sull'intero sistema esistente e tutte le funzionalità principali saranno interessate in modo diretto. Pertanto:

$$CIS = SIS$$

Le modifiche non comporteranno impatti su parti del sistema che non siano già direttamente incluse nel Starting Impact Set.

1.2 Implementazione delle Modifiche

1.2.1 Actual Impact Set (AIS)

Sono state identificate le seguenti componenti:

- *components*
- *controller*
- *cs_detector*
- *general_output*

1.2.2 False positive Impact Set

Non sono state individuate ulteriori funzionalità.

1.2.3 Discovered Impact Set

Non sono state individuate ulteriori funzionalità.

1.3 Metriche

Per valutare qualitativamente l'analisi d'impatto, sono state adoperate le metriche di *Precision* e di *Recall*, dopo aver applicato le modifiche:

- **Precision**:

$$((CIS \cap AIS) \div AIS) = 4 \div 4 = 1$$

- **Recall**:

$$((CIS \cap AIS) \div CIS) = 4 \div 4 = 1$$

2 Descrizione CR2

La presente proposta di modifica riguarda lo sviluppo ex novo di una web application per l'interazione con i servizi di analisi dei code smells. Questa funzionalità è stata realizzata senza impattare il codice preesistente, ma come un'estensione esterna completamente nuova.

Comportamento attuale:
Non esiste alcuna interfaccia utente per interagire con i servizi di analisi dei code smells. Gli utenti devono fare uso del tool da riga di comando o tramite una GUI per poter analizzare il codice sorgente. La funzionalità di generazione dei report è da richiamare separatamente sempre da riga di comando, facendo a meno di un'esperienza visiva e intuitiva.

Proposta di modifica:
Sviluppare una web application basata su React e Next.js per consentire agli utenti di:

- Caricare il codice (o i progetti) da analizzare direttamente dall'interfaccia.
- Avviare l'analisi tramite servizi dedicati (AI Analysis e Static Analysis).
- Generare e scaricare report dettagliati dei risultati dell'analisi.

La web application è stata progettata come un sistema indipendente e modulare, sfruttando un'architettura basata su microservizi.

Benefici previsti:
Lo sviluppo della web application apporterà i seguenti benefici:

- Miglioramento dell'esperienza utente attraverso un'interfaccia grafica intuitiva.
- Riduzione della complessità per l'utilizzo delle funzionalità esistenti.

2.1 Analisi di Impatto

2.1.1 Starting Impact Set (SIS)

Poiché la web application è stata sviluppata ex novo e non modifica alcun elemento del sistema preesistente, il *Starting Impact Set (SIS)* risulta vuoto:

$$SIS = \emptyset$$

2.1.2 Candidate Impact Set (CIS)

Essendo il sistema completamente nuovo, non esistono componenti preesistenti che possano essere indirettamente impattate dalle modifiche. Pertanto, anche il *Candidate Impact Set (CIS)* risulta vuoto:

$$CIS = \emptyset$$

2.2 Implementazione delle Modifiche

2.2.1 Actual Impact Set (AIS)

Analogamente, il *Actual Impact Set (AIS)* risulta vuoto, in quanto le modifiche non hanno impattato componenti preesistenti:

$$AIS = \emptyset$$

2.2.2 False Positive Impact Set

Non sono state individuate ulteriori funzionalità impattate erroneamente.

2.2.3 Discovered Impact Set

Non sono state individuate ulteriori funzionalità impattate che non fossero incluse nel *Candidate Impact Set*.

2.3 Metriche

Essendo il sistema sviluppato ex novo, sia il *Actual Impact Set (AIS)* che il *Candidate Impact Set (CIS)* risultano vuoti. Le metriche di *Precision* e *Recall* sono calcolate come segue:

- **Precision**:
$$((CIS \cap AIS) \div AIS) = \frac{0}{0}$$
- **Recall**:
$$((CIS \cap AIS) \div CIS) = \frac{0}{0}$$

Questi valori riflettono l'assenza di componenti errate o mancate, coerentemente con la natura della modifica descritta.

3 Descrizione CR3

La presente proposta di modifica riguarda l'utilizzo di modelli di linguaggio di grandi dimensioni (LLM) per l'iniezione automatica di code smells nei codebase, con l'obiettivo di generare dataset di addestramento in assenza di dati reali etichettati. Questa funzionalità rappresenta un'integrazione autonoma al sistema esistente.

Comportamento attuale:
Attualmente, non esiste un metodo automatizzato per generare dataset contenenti code smells. La generazione manuale è onerosa e soggetta a limiti in termini di scalabilità e variabilità dei dati prodotti.

Proposta di modifica:
Implementare un sistema basato su LLM per:

- Generare automaticamente varianti di codice con specifici code smells.

- Configurare i parametri per selezionare i tipi di smells desiderati.
- Consentire l'iniezione mirata di code smells su porzioni di codice specifiche all'interno di un code snippet.

Questo sistema sarà sviluppato come modulo separato e autonomo, facilmente integrabile nei flussi di lavoro esistenti.

Benefici previsti:
L'implementazione di questa modifica apporterà i seguenti benefici:

- Possibilità di creare dataset di addestramento per modelli di analisi dei code smells, colmando la mancanza di dati reali etichettati.
- Generazione scalabile di dataset contenenti code smells, mantenendone la diversità per l'addestramento.
- Riduzione dell'onere manuale nella creazione di dataset.

3.1 Analisi di Impatto

3.1.1 Starting Impact Set (SIS)

Poiché il sistema di iniezione basato su LLM è progettato come modulo separato e non modifica il codice esistente, il *Starting Impact Set (SIS)* risulta vuoto:

$$SIS = \emptyset$$

3.1.2 Candidate Impact Set (CIS)

Essendo il modulo completamente nuovo, non esistono componenti preesistenti che possano essere indirettamente impattate dalle modifiche. Pertanto, anche il *Candidate Impact Set (CIS)* risulta vuoto:

$$CIS = \emptyset$$

3.2 Implementazione delle Modifiche

3.2.1 Actual Impact Set (AIS)

Analogamente, il *Actual Impact Set (AIS)* risulta vuoto, in quanto le modifiche non hanno impattato componenti preesistenti:

$$AIS = \emptyset$$

3.2.2 False Positive Impact Set

Non sono state individuate ulteriori funzionalità impattate erroneamente.

3.2.3 Discovered Impact Set

Non sono state individuate ulteriori funzionalità impattate che non fossero incluse nel *Candidate Impact Set*.

3.3 Metriche

Essendo il modulo sviluppato ex novo, sia l' *Actual Impact Set (AIS)* che il *Candidate Impact Set (CIS)* risultano vuoti. Le metriche di *Precision* e *Recall* sono calcolate come segue:

- **Precision**:
$$((CIS \cap AIS) \div AIS) = \frac{0}{0}$$
- **Recall**:
$$((CIS \cap AIS) \div CIS) = \frac{0}{0}$$

4 Descrizione CR4

La presente proposta di modifica riguarda l'addestramento di un modello di intelligenza artificiale per il rilevamento automatico dei code smells. Tale modello utilizza come base i dataset generati dagli LLM (CR3) e consente l'identificazione efficiente e scalabile dei difetti di codice, migliorando l'analisi della qualità del software. Questo modulo rappresenta un'estensione autonoma e integrabile al sistema esistente.

Comportamento attuale:
Attualmente, non esiste un modulo basato su IA per il rilevamento automatico dei code smells.

Proposta di modifica:
Implementare un sistema basato su modelli di machine learning per:

- Addestrare un modello utilizzando i dataset generati dagli LLM con smells iniettati.
- Valutare le prestazioni del modello attraverso metriche standard (accuratezza, precisione, richiamo e F1 score).
- Creare un modulo in grado di analizzare nuovo codice e identificare automaticamente i difetti rilevati.

Benefici previsti:
L'implementazione di questa modifica apporterà i seguenti benefici:

- Miglioramento della capacità di identificare smells *unfeasible* che non possono essere rilevati in modo deterministico tramite regole statiche.
- Fornire un modo alternativo e valido per il rilevamento di *ML-specific smells* tramite un modello addestrato su dataset deticati.

4.1 Analisi di Impatto

4.1.1 Starting Impact Set (SIS)

Poiché il sistema di rilevamento basato su IA è progettato come modulo separato e non modifica il codice esistente, il *Starting Impact Set (SIS)* risulta vuoto:

$$SIS = \emptyset$$

4.1.2 Candidate Impact Set (CIS)

Essendo il modulo completamente nuovo, non esistono componenti preesistenti che possano essere indirettamente impattate dalle modifiche. Pertanto, anche il *Candidate Impact Set (CIS)* risulta vuoto:

$$CIS = \emptyset$$

4.2 Implementazione delle Modifiche

4.2.1 Actual Impact Set (AIS)

Analogamente, il *Actual Impact Set (AIS)* risulta vuoto, in quanto le modifiche non hanno impattato componenti preesistenti:

$$AIS = \emptyset$$

4.2.2 False Positive Impact Set

Non sono state individuate ulteriori funzionalità impattate erroneamente.

4.2.3 Discovered Impact Set

Non sono state individuate ulteriori funzionalità impattate che non fossero incluse nel *Candidate Impact Set*.

4.3 Metriche

Essendo il modulo sviluppato ex novo, sia l' *Actual Impact Set (AIS)* che il *Candidate Impact Set (CIS)* risultano vuoti. Le metriche di *Precision* e *Recall* sono calcolate come segue:

- **Precision**:
$$((CIS \cap AIS) \div AIS) = \frac{0}{0}$$
- **Recall**:
$$((CIS \cap AIS) \div CIS) = \frac{0}{0}$$