CR Proposal + Test Plan

Progetto ISTA – CodeSmile #Idea 2

Autori:

Nome e Cognome	Matricola
Daniele Pio Scaparra	NF22500101
Grazia Bonagura	NF22500105
Antonio Nappi	In attesa di matricola

Outline

1. Obiettivi del Progetto

Il progetto mira a migliorare CodeSmile, un tool di analisi statica che rileva code smell relativi alle pipeline di Machine Learning.
Dopo un'analisi iniziale del codice e della documentazione, sono state individuate tre aree di miglioramento che consentono un'estensione significativa delle capacità del tool:

1. **migliorare un detector esistente**, attualmente limitato e fonte di falsi negativi/falsi positivi;
2. **introdurre una rappresentazione strutturale del progetto tramite call graph;**
3. **abilitare l'utente allo studio visuale e analitico delle relazioni tra funzioni.**

Ciò costituisce la base per le change request ideate e illustrate di seguito.

---

2. Change Requests

CR1 — Miglioramento del Detector "In-Place APIs Misused"

Tipo di Manutenzione:

- Correttiva (risolve falsi positivi non segnalati)
- Perfettiva (migliora recall e precision sulla detection di smell IPA)

Motivazione

Il detector originale, pensato per identificare usi scorretti di API non-in-place, soffre di alcune limitazioni:

- copertura parziale delle API Pandas
- mancanza di supporto per NumPy, ciò suggerisce un disallineamento con quello che è presente in letteratura in merito
- mancato riconoscimento dei risultati non utilizzati
- presenza di falsi positivi in situazioni corrette

Obiettivo della modifica

- Ampliare il detector a un insieme completo di API Pandas e NumPy.
- Identificare correttamente i casi in cui il risultato di un'operazione viene ignorato.
- Migliorare la capacità del detector di distinguere casi corretti vs. casi scorretti.

Benefici attesi

- Aumento significativo dell'accuratezza del detector.

- Riduzione dei falsi positivi e falsi negativi.
- Maggiore coerenza con le definizioni dei ML-specific smells.
- Qualità dei report più elevata, utile per sviluppatori e analisti.

CR2 — Creazione del Call Graph del Progetto

Tipologia di manutenzione: Manutenzione evolutiva (enhancement)

Motivazione

Attualmente CodeSmile analizza funzioni isolate, senza considerare l'architettura logica delle chiamate tra componenti.
L'assenza di un call graph impedisce di:

- interpretare la struttura complessiva del progetto,
- collegare gli smell a relazioni funzionali,
- comprendere dipendenze significative tra funzioni.

Obiettivo della modifica

Realizzare un sottosistema dedicato a:

- estrazione automatica delle chiamate tra funzioni e metodi,
- costruzione del call graph del progetto,
- annotazione dei nodi con gli smell rilevati da CodeSmile,
- esportazione della struttura in formato JSON e DOT,
- estensione della CLI con un comando per generare il grafo.

Benefici attesi

- Introduce una dimensione architetturale nell'analisi degli smell.
- Permette di osservare i legami tra le funzioni del progetto.
- Costituisce la base per ulteriori analisi avanzate (CR3).

CR3 — Analisi e Visualizzazione del Call Graph

Tipologia di manutenzione: Manutenzione evolutiva (enhancement)

Motivazione

Il call graph generato dalla CR2 rappresenta la struttura funzionale del progetto, ma necessita di strumenti che ne facilitano l'interpretazione.
Una rappresentazione puramente testuale o JSON/DOT non è sufficiente per evidenziare:

- nodi critici,
- relazioni complesse,
- cicli,
- funzioni più rilevanti.

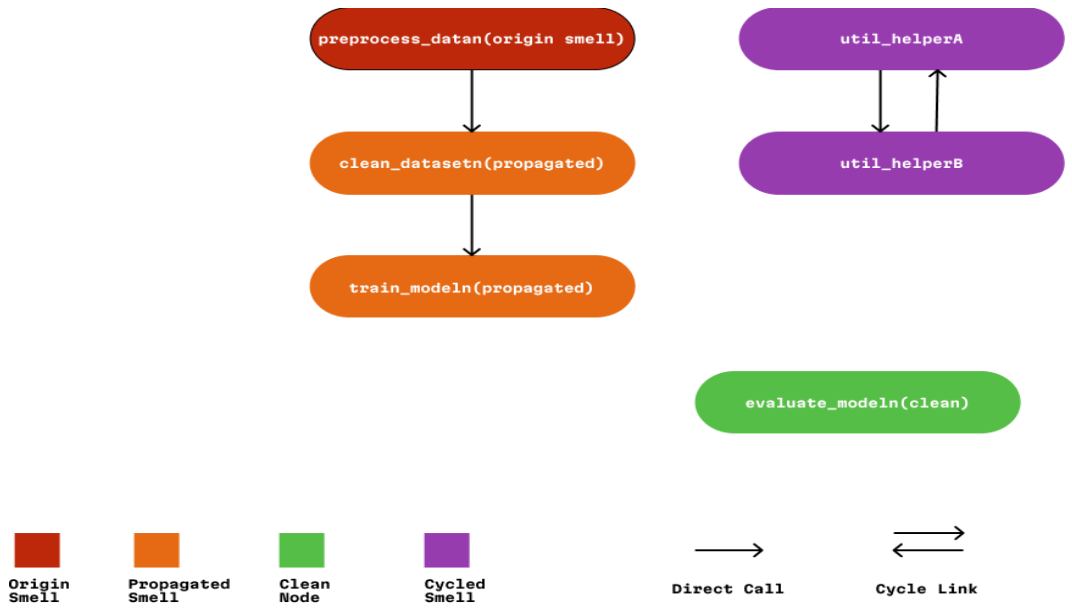
Obiettivo della modifica

La CR3 introduce un sottosistema per:

- visualizzazione grafica del call graph (PNG/SVG),
- calcolo di metriche sulle dipendenze (es. archi entranti/uscente),
- rilevazione automatica dei cicli nel grafo,
- supporto all'utente nella comprensione dell'architettura logica del progetto.

Benefici attesi

- Visione chiara e immediata della struttura del codice.
- Identificazione rapida di componenti critiche (funzioni centrali, cicli).
- Miglior supporto alla manutenzione e al refactoring.



Mockup della CR3:
L'immagine qui riportata rappresenta un mockup realizzato in Figma, pensato per mostrare visivamente come potrebbe apparire il grafo finale generato dal sistema di analisi.
Si tratta di un'anteprima concettuale, utile a illustrare:

- *la distinzione tra funzioni con smell originari (in rosso),*
- *funzioni che ricevono smell per propagazione (in arancione),*
- *funzioni pulite (in verde),*
- *e funzioni inserite in cicli di dipendenze (in viola).*

Il mockup include anche una legenda, che chiarisce il significato dei colori e dei collegamenti:

- *frecce semplici per rappresentare le chiamate dirette tra funzioni*
- *doppia freccia per indicare la presenza di un ciclo*

L'obiettivo non è mostrare l'output definitivo, ma fornire un'idea immediata della struttura e dell'aspetto generale del grafo che verrà prodotto.
Nella versione finale, il grafo sarà generato automaticamente da strumenti Python che produrranno codice DOT eseguibile; di conseguenza, l'aspetto potrà variare leggermente, ma la logica visiva rimarrà coerente con il mockup.

Questo prototipo serve quindi come riferimento visivo preliminare per comprendere il tipo di rappresentazione che l'utente avrà a disposizione nell'analisi degli smell e delle loro propagazioni.

---

3. Valutazione del Rischio e dell'Impatto

CR	Rischio	Impatto
CR1 – Miglioramento del Detector "In-Place APIs Misused"	**Medio** – modifica il comportamento di un detector core, richiede attenzione per evitare regressioni	**Medio** – aumenta significativamente la precisione del rilevamento senza modificare altri componenti
CR2 – Creazione del Call Graph	**Medio** – introduce un nuovo sottosistema, ma isolato dal resto del tool	**Alto** – abilita una visione architetturale del progetto e costituisce base per analisi avanzate

CR3 – Analisi e Visualizzazione del Call Graph	**Medio-Basso** – utilizza un output già strutturato; il rischio deriva soprattutto dalla complessità della visualizzazione	**Medio-Alto** – migliora l'usabilità e la capacità di interpretare il progetto, aggiungendo valore analitico significativo

---

4. Test Plan — Executive Summary

Il Test Plan ha l'obiettivo di assicurare che:

- le nuove funzionalità introdotte da CR1, CR2 e CR3 funzionino correttamente,
- non si verifichino regressioni rispetto al comportamento esistente,
- gli output generati siano coerenti e interpretabili.

4.1 Componenti già presenti nel tool

L'analisi preliminare ha evidenziato che il tool possiede:

- una buona copertura di test sulle componenti core (CLI, extractors, detectors, report)
- test non eseguibili su parti non rilevanti per il progetto (GUI e WebApp)
- un insieme già adeguato di casi di riferimento per la regressione

Questi elementi costituiscono una buona base di partenza.

4.2 Strategia di Testing

Approccio multilivello

1. Unit Test

- detector migliorato (CR1)
- estrazione del call graph (CR2)
- funzioni di analisi e visualizzazione (CR3)

2. Integration Test

- interazione tra analyzer, call graph generator, CLI
- verifica del pipeline completo JSON/DOT → analisi → visualizzazione

3. System Test

- esecuzione end-to-end con progetto di test multi-file
- coerenza del grafo e della visualizzazione

4. Regression Test selettivi

- esecuzione dei test core già presenti
- esclusione delle componenti non rilevanti (GUI/WebApp)

4.3 Test da aggiungere

Per CR1 – Detector migliorato

Saranno aggiunti test che coprano i casi principali:

- utilizzo corretto delle API non-in-place (da non segnalare)
- utilizzo scorretto (risultati ignorati)
- test per ridurre falsi positivi (assegnazioni, return, argomenti di funzione)
- casi con NumPy e Pandas

L'obiettivo è verificare la precisione e la robustezza della nuova logica.

Per CR2 - Creazione del Call Graph

Saranno aggiunti test che coprano:

- corretta estrazione delle chiamate tra funzioni
- validità dei file JSON e DOT generati
- annotazione corretta degli smell nei nodi
- verifica CLI dedicata

L'obiettivo è verificare la correttezza dell'implementazione della nuova funzionalità

Per CR3 - Visualizzazione del Call Graph

Saranno aggiunti test per verificare:

- generazione dei file PNG/SVG
- calcolo metriche del grafo

- detection dei cicli
- verifica coerenza visualizzazione/grafo

L’obiettivo è verificare la correttezza dell’implementazione della nuova funzionalità

4.4 Criteri di Successo

Le implementazioni previste dalle Change Request saranno considerate complete e corrette se:

- **tutti i nuovi test** introdotti per CR1, CR2 e CR3 vengono superati con successo;
- **non si verificano regressioni** nel comportamento dei detector e delle funzionalità già presenti nel tool;
- **gli output generati** (come file JSON, DOT e relative visualizzazioni grafiche) risultano coerenti, interpretabili e corrispondenti alle specifiche definite nelle CR;
- il flusso complessivo delle nuove funzionalità risulta stabile, ripetibile e integrato senza introdurre effetti collaterali.

---

5. Benefici per il Progetto CodeSmile

L’introduzione delle tre Change Request proposte e del Test Plan previsto apporta benefici concreti e complementari, migliorando sia la qualità dell’analisi sia le capacità complessive dello strumento.

Sul piano funzionale

- miglioramento dell’accuratezza nel rilevamento degli smell attraverso un detector più completo e affidabile (CR1);
- introduzione della possibilità di analizzare il progetto non solo a livello di singole funzioni, ma tramite la sua struttura di chiamate (CR2);
- disponibilità di visualizzazioni e metriche che rendono il call graph facilmente interpretabile e utile per l’analisi architetturale (CR3).

Sul piano manutentivo

- design più modulare, grazie alla separazione tra costruzione del call graph e sua analisi (CR2 e CR3);
- maggiore chiarezza interna del codice, favorita dal miglioramento del detector e dall'organizzazione dei nuovi sottosistemi;
- una test suite più robusta, arricchita da nuovi test unitari, di integrazione e di sistema che migliorano la stabilità del progetto nel tempo.
