RELAZIONE SULL'UTILIZZO DELL'INTELLIGENZA ARTIFICIALE

Progetto: Multi-Agent Test Generator
Corso: Ingegneria dei Linguaggi di Programmazione
Autore: Gianmarco Riviello

1. TIPOLOGIE DI UTILIZZO DELL'INTELLIGENZA ARTIFICIALE

Durante lo sviluppo del progetto, l'IA è stata impiegata trasversalmente in diverse fasi critiche:

a) Sviluppo Codice

- Generazione automatica di test unitari tramite il framework pytest.
- Creazione di test mirati per branch specifici non coperti inizialmente.
- Ottimizzazione iterativa della test suite per aumentare la coverage complessiva.
- Suggerimenti per refactoring e miglioramento della manutenibilità del codice esistente.

b) Generazione delle Specifiche

- Supporto nella definizione della grammatica Lark per il subset del linguaggio Python.
- Progettazione dell'architettura del sistema basata su un approccio multi-agente.
- Ottimizzazione dei prompt (Prompt Engineering) per guidare l'output del modello.

c) Generazione di Test

- Creazione automatica di test per la copertura integrale dei branch identificati.
- Identificazione e generazione di casi edge (valori limite, zero, negativi, divisioni per zero).
- Implementazione di test parametrizzati tramite pytest.mark.parametrize.
- Sviluppo di test per la gestione di condizioni logiche annidate complesse.

2. STRUMENTI AI UTILIZZATI

Strumento	Descrizione / Utilizzo
Google Gemini 2.5 Flash	LLM principale per la generazione dinamica dei test (integrato via LangChain).
GitHub Copilot	Supporto alla scrittura di codice boilerplate e assistenza in tempo reale.
ChatGPT/Claude	Approfondimento di concetti teorici su parsing, AST e grammatiche formali.

3. CONTESTO DI INTEGRAZIONE

L'IA è stata integrata come motore generativo all'interno di un'architettura multi-agente. A differenza dei sistemi tradizionali, l'IA agisce in un ciclo di feedback chiuso:

> ***Codice → Analisi Lark → LLM genera test → Coverage misurata → [Se insufficiente] → LLM ottimizza → Ripeti***

4. VANTAGGI RISCONTRATI

- VELOCITÀ: Riduzione drastica dei tempi di scrittura dei test (da ore a secondi).
- COMPLETEZZA: Identificazione di branch complessi e annidati (es. demo con 80+ branch).
- CREATIVITÀ: Suggerimento di input limite (stringhe vuote, numeri enormi) non ovvi.
- APPRENDIMENTO: Acquisizione di pattern avanzati di pytest e best practices.

5. SVANTAGGI E SFIDE AFFRONTATE

- **Allucinazioni:** Generazione di import inesistenti. Soluzione: validazione formale con AST.
- **Non Determinismo:** Output variabili per lo stesso prompt. Soluzione: implementazione di MockLLMClient.
- **Latenza:** Tempi di risposta API (~2-3s per chiamata).
- **Context Window:** Limiti di token su file estesi. Soluzione: chunking del codice.

6. LEZIONI APPRESE

- L'IA è un assistente, non un sostituto: la validazione formale è indispensabile.
- Il prompt è cruciale: la precisione delle istruzioni determina la qualità dell'output.
- L'approccio ibrido vince: l'unione di IA e strumenti deterministici è la chiave del successo.
- L'IA non compensa lacune teoriche: la comprensione di branch coverage e AST rimane essenziale.

CONCLUSIONE

L'IA si è dimostrata un alleato potente ma 'indisciplinato'. Il successo del progetto risiede nell'architettura che la circonda (Lark + AST + Coverage), garantendo output utili, corretti e verificabili.