IGES - IDEAL

Domenico Antonio Gioia
d.gioia7@studenti.unisa.it
Matricola: 0522501518

Giovanni Scorziello
g.scorziello5@studenti.unisa.it
Matricola: 0512108926

Antonio Scognamiglio
a.scognamiglio32@studenti.unisa.it
Matricola: 0522501496

1 CONTESTO DEL PROGETTO

I Linguistic Antipattern sono problemi legati alla denominazione degli elementi nel codice sorgente di un progetto software. Causano incoerenze, ambiguità e confusione nel codice, rendendolo meno facile da capire e da manutenere.
Per questo motivo, sono considerati una cattiva pratica di scrittura del codice. La correzione di questi tipi di problemi potrebbe aiutare a migliorare la qualità del codice, semplificandone la manutenzione e l'evoluzione. Sfortunatamente, al momento non è disponibile alcuno strumento in grado di rilevare questi antipattern, nel codice Python, e notificarli agli sviluppatori. Inoltre, Python è un linguaggio critico per i sistemi basati sull'intelligenza artificiale, essendo il linguaggio più utilizzato per questo tipo di software [2]. L'unico strumento che supporta questa analisi è *IDEAL*[1], che però supporta solo *C#* e *Java*; è scritto in Python ed è possibile utilizzarlo anche come plugin per *Intellij Idea*. Questo strumento analizza il codice per cercare gli antipattern definiti da [1].

2 CHANGE REQUEST

E' stata effettuata un'analisi del tool IDEAL, per comprendere come esso svolgesse le sue funzioni. Dall'analisi effettuata sul sistema, sono emersi diversi aspetti che potrebbero essere migliorati per facilitare l'utilizzo del tool anche come sistema stand alone.
Sono state attribuite priorità ai vari aspetti in base a criteri specifici, tra cui quanto la modifica potrebbe migliorare l'usabilità del tool, la richiesta del mercato per la nuova funzionalità e l'aumento dell'impatto del sistema nel mondo reale che la modifica potrebbe generare. Le varie modifiche sono state valutate sulla base di una scala da 1 a 10 per determinare la loro priorità. Il valore 1 indica che la modifica ha un basso impatto su tutti i criteri elencati in precedenza, mentre il valore 10 indica che la modifica ha un impatto molto elevato. Tali aspetti riguardano:

- **Supporto all'analisi di codice Python** *[Priorità 8.5/10]*: Con l'aumento delle applicazioni che integrano sistemi di Intelligenza Artificiale, il linguaggio in questione è diventato molto popolare. Pertanto, è importante integrare un supporto per questo linguaggio all'interno del sistema.
- **Semplificare l'installazione** *[Priorità: 7/10]*: Il processo di installazione del tool come sistema a se stante richiede, da parte dell'utente, il download di diverse librerie ed eseguibili. In questa fase, non viene fornito alcun supporto allo sviluppatore, il quale dovrà individuare autonomamente le versioni appropriate e configurarle. Per semplificare questo processo, è necessario fornire assistenza o automatizzare l'installazione per l'utente finale.
- **Migliorare la fase di input** *[Priorità: 6/10]*: Affinché il sistema possa analizzare un progetto in input, è necessario inserire la sua path all'interno di un file chiamato input.csv, situato nella cartella del tool. Successivamente, il path di questo file viene inserito in un altro file chiamato run.cmd, che viene eseguito per effettuare l'analisi del progetto. Poiché le path devono essere assolute, lo sviluppatore che utilizza il sistema dovrà ricostruire l'intero percorso per accedere al proprio progetto e al file input.csv. Poiché il processo di inserimento della path del progetto all'interno di file specifici richiede molto tempo e risulta macchinoso, si prevede l'implementazione di una modalità di input più semplice ed intuitiva, che consenta all'utente di inserire il progetto da analizzare tramite interfaccia a riga di comando (CLI).
- **Migliorare le modalità di configurazione del tool** *[Priorità: 5/10]*: Dopo aver effettuato le operazioni di installazione, le librerie e gli eseguibili installati dovranno poi essere referenziati all'interno dei file di configurazione del tool. Si vuole snellire questo meccanismo, tramite la riduzione del numero di file di configurazione.
- **Migliorare l'output** *[Priorità: 6/10]*: Dopo aver eseguito l'analisi sul progetto preso in input, il risultato viene mostrato attraverso un file csv, all'interno del quale vi sono riportati: la path del file in cui è stato individuato un linguistic antipattern, il tipo di file, il tipo di identificatore (se si tratta quindi di un metodo, una variabile, un parametro o una classe), il numero della riga in cui è stato trovato il problema, la tipologia del linguistic antipattern individuato rappresentato tramite un ID corrispondente ad una riga della tabella riportata nella repository GitHub [2], la sua categoria, la data dell'analisi e altri dettagli.
  Questo tipo di formato è difficile da leggere, soprattutto se dall'analisi risultano diversi problemi, e quindi diverse righe all'interno del file. Si vuole cambiare tipo di output, utilizzando un formato migliore e più facile da interpretare.

REFERENCES

[1] Venera Arnaoudova, Massimiliano Di Penta, and Giuliano Antoniol. "Linguistic antipatterns: What they are and how developers perceive them". In: *Empirical Software Engineering* 21 (2016), pp. 104–158.
[2] Teik Toe Teoh and Zheng Rong. *Artificial intelligence with python*. Springer, 2022.

[1] https://github.com/SCANL/ProjectSunshine

[2] https://github.com/SCANL/ProjectSunshine/blob/master/documentaion/IDEAL/AntiPatternRules_Arnaoudova.md