UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Corso di
Ingegneria, Gestione ed Evoluzione del Software

IDEAL - Modification Design Document

LINK REPOSITORY
`https://github.com/xrenegade100/ProjectSunshine`

LINK REPOSITORY DOCUMENTAZIONE DEL CODICE
`https://github.com/xrenegade100/ideal-api-docs`

AUTORI

Antonio Scognamiglio
Matricola: 0522501496
Antonio Domenico Gioia
Matricola: 0522501518
Giovanni Scorziello
Matricola: 522501701

Anno Accademico 2022-2023

Indice

1 Introduzione

Questo documento specifica la progettazione delle modifiche relative alle change request. Saranno presentate le change request correlate della loro priorità, impatto stimato e difficoltà di implementazione. Nei successivi capitoli, poi, saranno effettuate delle *impact analysis* per stimare l'impatto delle modifiche sul sistema; per ognuna di essere saranno forniti gli insiemi dell'impact analysis, una spiegazione dell'approccio utilizzato per stimare tali insiemi, le metriche di processo finali, e un paragrafo che riassume la fase di implementazione.

2 Panoramica del sistema

Il sistema è progettato per assistere gli sviluppatori nello sviluppo di programmi Java e C#. Può analizzare i file di codice sorgente per individuare i linguistic antipattern definiti da Arnaoudova et al. 2016. L'utente può analizzare cartelle intere o specifici file. Vengono controllati elementi come classi, metodi e variabili e le violazioni alle regole vengono registrate in un file CSV.
Le change request proposte, includono di permettere al software di analizzare file di codice sorgente Python, oltre che semplificare le fasi di installazione, configurazione, ed esecuzione del tool. Le suddette change request sono progettate tramite questo documento.

3 Change Request

Durante l'utilizzo del software e la creazione dei casi di test, è emerso che il codice sorgente presentava una serie di problemi legati alla manutenibilità. Di conseguenza, è stata utilizzata la piattaforma web SonarCloud per condurre un'analisi relativa alla ricerca di problemi riguardo la manutenibilità, la sicurezza e l'utilizzo di "best-practice" nel codice del software. Dall'analisi è emersa la presenza di 3 bugs, 75 code smells, 61 security hotspots. **Di conseguenza, è stato necessario introdurre una nuova change request con l'obiettivo di avviare un processo di refactoring del software, mirato a migliorare la sua manutenibilità.** Nella Tabella 1, sono state elencate le change request ordinate in base alle loro priorità.
Il codice sarà ispezionato tramite il processo di *code review* di GitHub prima di essere integrato nella repository. Verrà anche inserita una pipeline di Continuous Integration/Continuous Testing in modo da permettere frequenti integrazioni rilevando eventuali problemi tempestivamente.

Tabella 1: Change requests

CR_ID	Descrizione	Tipo	Priorità	Impatto	Difficoltà
CR_0	Effettuare un refactoring del software per migliorarne la manutenibilità.	Perfettiva	10	Alto	Medio
CR_1	Permettere di analizzare codice Python per identificare linguistic antipattern.	Adattiva	8.5	Alto	Alta
CR_2	Permettere di effettuare un'istallazione semplificata del tool.	Perfettiva	7	Alto	Media
CR_3	Semplificare l'inserimento dei file che si richiedono di analizzare.	Perfettiva	6	Medio	Media
CR_4	Migliorare la visualizzazione dei risultati ottenuti dalle analisi.	Perfettiva	6	Medio	Bassa
CR_5	Riduzione dei file di configurazione.	Perfettiva	5	Alto	Medio

4 Impact Analysis

4.1 CR_0 - Refactoring

4.1.1 Obiettivo della change request

L'obiettivo della change request è il potenziamento della manutenibilità, della sicurezza del software e la rimozione di componenti non utilizzate. La necessità di questa change request sorge dalla fondamentale esigenza di migliorare il codice esistente prima di procedere con qualsiasi intervento di modifica sostanziale. Le sezioni di codice che necessitano di miglioramenti sono state individuate attraverso un'analisi condotta con l'ausilio dello strumento SonarCloud.

4.1.2 Starting impact set

Le componenti identificate che richiedono modifiche sono elencate nella tabella 2, la quale presenta il file soggetto alle modifiche e la tipologia di intervento necessario. Le principali attività eseguite durante questo processo di refactoring comprendono principalmente la correzione di problemi di code smell e bug, il miglioramento complessivo della qualità del codice, nonché l'eliminazione di file e metodi non utilizzati.

Tabella 2: Starting Impact Set - CR_0

File	Operazione
src.service.config_reader	Rimozione file
src.service.factory	Fix code smell
src.service.parser	Fix bug
src.nlp.pos_tagger_stanford	Fix bug
src.nlp.term_property	Fix bug
src.nlp.related_terms	Fix code smell
src.nlp.splitter	Rimozione metodo inutilizzato
src.nlp.term_list	Fix code smell
src.nlp.pos_tag	Fix code smel
src.model.project	Fix code smell
src.model.issue	Miglioramento qualità del codice
src.model.identifier	Fix code smell
src.model.entity	Fix code smell

Continued on next page

Tabella 2: Starting Impact Set - CR_0 (Continued)

src.common.testing_list	Fix code smell
src.common.types_list	Fix code smell
src.common.util_parsing	Fix code smell
src.common.util	Fix code smell
src.apps.IDEAL.analyzer	Rimozione codice non utilizzato
src.apps.IDEAL.main	Rimozione codice non utilizzato
src.apps.IDEAL.result_writer	Fix code smell
src.rule.linguistic_antipattern. attribute_name_type_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. attribute_signature_comment_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. contains_only_special_characters	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. expecting_not_getting_collection	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. expecting_not_getting_single	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. get_more_than_accessor	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. get_no_return	Miglioramento qualità del codice
src.rule.linguistic_antipattern. is_no_return_bool	Miglioramento qualità del codice

Continued on next page

Tabella 2: Starting Impact Set - CR_0 (Continued)

src.rule.linguistic_antipattern. method_name_return_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. method_signature_comment_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. name_suggest_boolean_type_not	Miglioramento qualità del codice
src.rule.linguistic_antipattern. not_answered_question	Miglioramento qualità del codice
src.rule.linguistic_antipattern. not_implemented_condition	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. says_many_contains_one	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. says_one_contains_many	Miglioramento qualità del codice
src.rule.linguistic_antipattern. set_returns	Miglioramento qualità del codice
src.rule.linguistic_antipattern. starts_with_special_character	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. transform_not_return	Fix code smell
src.rule.linguistic_antipattern. validate_not_confirm	Fix code smell

4.1.3 Candidate Impact Set

Dopo aver identificato le componenti dello *Starting Impact Set*, sono state individuate le dipendenze dirette di queste componenti. È stata presa la decisione di analizzare e includere nel *Candidate Impact Set* solo le dipendenze dirette. Questa scelta è stata fatta perché nel set di partenza erano inclusi circa l'80% delle classi del progetto, e ulteriori livelli di analisi delle dipendenze avrebbero portato all'individuazione di classi già presenti nell'insieme iniziale. In effetti, attraverso l'analisi delle sole dipendenze dirette, si è già ottenuta una copertura del 96% delle classi del progetto. La ricerca delle dipendenze dirette è stata effettuata tramite l'osservazione

delle importazioni dei moduli utilizzati nei singoli file coinvolti nell'analisi. Pertanto, il *Candidate Impact Set* include tutti gli elementi dello *Starting Impact Set* e tutti i file individuati tramite dipendenze dirette. La tabella 3 mostra gli elementi presenti nel *Candidate Impact Set*.

Tabella 3: Candidate Impact Set - CR_0

File	Operazione
src.service.config_reader	Rimozione file
src.service.factory	Fix code smell
src.service.parser	Fix bug
src.nlp.pos_tagger_stanford	Fix bug
src.nlp.term_property	Fix bug
src.nlp.related_terms	Fix code smell
src.nlp.splitter	Rimozione metodo inutilizzato
src.nlp.term_list	Fix code smell
src.nlp.pos_tag	Fix code smel
src.model.project	Fix code smell
src.model.input	Dipendenza diretta di src.common.util e src.apps.IDEAL.main
src.model.issue	Miglioramento qualità del codice
src.model.identifier	Fix code smell
src.model.entity	Fix code smell

Continued on next page

Tabella 3: Candidate Impact Set - CR_0 (Continued)

src.common.testing_list	Fix code smell
src.common.types_list	Fix code smell
src.common.util_parsing	Fix code smell
src.common.util	Fix code smell
src.common.singleton	Dipendenza diretta di src.common.testing_list, src.nlp.pos_tagger_standford e src.nlp.splitter
src.common.enum	Dipendenza diretta di src.common.testing_list, src.common.types_list, src.common.util_parsing, src.model.entity e src.rule.linguistic_antipattern.*
src.common.error_handler	Dipendenza diretta di src.service.factory, src.common.util, src.model.entity, src.model.project, src.apps.IDEAL.main e src.rule.linguistic_antipattern.*
src.common.logger	Dipendenza diretta di src.apps.IDEAL.main e src.common.error_handler
src.apps.IDEAL.analyzer	Rimozione codice non utilizzato
src.apps.IDEAL.main	Rimozione codice non utilizzato
src.apps.IDEAL.result_writer	Fix code smell

Continued on next page

Tabella 3: Candidate Impact Set - CR_0 (Continued)

src.rule.linguistic_antipattern. attribute_name_type_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. attribute_signature_comment_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. contains_only_special_characters	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. expecting_not_getting_collection	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. expecting_not_getting_single	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. get_more_than_accessor	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. get_no_return	Miglioramento qualità del codice
src.rule.linguistic_antipattern. is_no_return_bool	Miglioramento qualità del codice
src.rule.linguistic_antipattern. method_name_return_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. method_signature_comment_opposite	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. name_suggest_boolean_type_not	Miglioramento qualità del codice
src.rule.linguistic_antipattern. not_answered_question	Miglioramento qualità del codice
src.rule.linguistic_antipattern. not_implemented_condition	Fix code smell Miglioramento qualità del codice

Continued on next page

Tabella 3: Candidate Impact Set - CR_0 (Continued)

src.rule.linguistic_antipattern. says_many_contains_one	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. says_one_contains_many	Miglioramento qualità del codice
src.rule.linguistic_antipattern. set_returns	Miglioramento qualità del codice
src.rule.linguistic_antipattern. starts_with_special_character	Fix code smell Miglioramento qualità del codice
src.rule.linguistic_antipattern. transform_not_return	Fix code smell
src.rule.linguistic_antipattern. validate_not_confirm	Fix code smell
src.rule.linguistic_antipattern. test_annotation_setup	Dipendenza diretta di src.apps.IDEAL.analyzer
src.rule.linguistic_antipattern. test_annotation_teardown	Dipendenza diretta di src.apps.IDEAL.analyzer
src.rule.linguistic_antipattern. test_annotation_test	Dipendenza diretta di src.apps.IDEAL.analyzer
src.rule.linguistic_antipattern. test_missing_null_check	Dipendenza diretta di src.apps.IDEAL.analyzer
src.rule.linguistic_antipattern. test_nonverb_starting	Dipendenza diretta di src.apps.IDEAL.analyzer

4.1.4 Discoverd Impact Set

Dopo l'operazione di refactoring, il *Discovered Impact Set* è risultato essere vuoto, poiché nessuna componente non inclusa nel *Candidate Impact Set* è stata coinvolta in questo processo.

4.1.5 Actual Impact Set

La tabella relativa all'*Actual Impact Set* è stata omessa, poiché nella sezione successiva è possibile rilevare quali file sono stati inclusi nel *Candida-*

te Impact Set ma che non hanno effettivamente subito alcun processo di refactoring.

4.1.6 False Positive Impact Set

Gli elementi enumerati nella Tabella 4 rappresentano gli elementi che, sebbene inclusi nel Candidate Impact Set, non hanno subito alcuna modifica.

Tabella 4: False Positive Impact Set

File
src.mode.input
src.common.singleton
src.common.enum
src.common.error_handler

4.1.7 Metriche di processo

Per questa change request sono state elaborate diverse metriche per valutare quanto l'approccio utilizzato sia stato d'aiuto al processo. Di seguito sono elencati i risultati ottenuti:

- **Recall** = $\frac{|CIS \cap AIS|}{|AIS|} = 1$
  La metrica di Recall rappresenta la proporzione degli elementi correttamente identificati nel *Candidate Impact Set* che sono anche presenti nell'*Actual Impact Set* ed è pari a 1, in quanto il *Discovered Impact Set* è vuoto.
- **Precision** = $\frac{|CIS \cap AIS|}{|CIS|} = \frac{45}{49} \approx 0.918$
  La metrica di Precision rappresenta la proporzione degli elementi effettivamente modificati del *Candidate Impact Set* rispetto a quelli dell'*Candidate Impact Set*. Questo significa che circa il 91.8% delle componenti del *Candidate Impact Set* è stato soggetto a cambiamenti.
- **Inclusiveness** = 1, perché $AIS \subseteq CIS$
  Questa metrica è pari a 1, in quanto l'*Actual Impact Set* è incluso nel *Candidate Impact Set*. Questo valore indica l'efficienza dell'approccio utilizzato per stimare il *CIS* di individuare tutti gli elementi interessati da modifiche.

4.1.8 Implementazione

L'implementazione di questa change request consiste nella rimozione di file e metodi non utilizzati, nella riduzione della complessità cognitiva di alcu-

ne funzioni, ottenuta riducendo il numero di cicli annidati, la sostituzione di costrutti if-else dove possibile e la rimozione di importazioni di moduli non utilizzati. In particolare, come si può constatare dalla prima riga della Tabella 2, è stata eseguita la rimozione del file *src.service.config_reader*. Il motivo di questa scelta è dato dal fatto che i metodi all'interno del file risultano inutilizzati, oltre ad essere sostituiti da alcuni metodi in *src.common.util* che svolgono le stesse funzioni. Analogamente, decisioni simili sono state adottate per altri metodi presenti in diversi file, come il metodo *split_ronin* presente nel modulo *src.nlp.splitter*. Oltre ad operazioni stabilite ispezionando manualmente il codice, sono stati risolti tutti i 75 code smell e 4 bug identificati da SonarCloud; per quanto riguarda i 61 security hotspots, sono stati segnalati come falsi positivi sulla stessa piattaforma, perché riguardavano l'utilizzo di URL non in formato *https* ma *http*, che però servono solo per formare i percorsi xpath utilizzati dal parser per analizzare l'output di srcML. Dopo le operazioni di refactoring, sono stati eseguiti tutti i test per condurre un test di regressione, che ha dato esito positivo in quanto sono passati tutti.
Inoltre, è stata arricchita la documentazione associata al codice esistente, con l'obiettivo di fornire una comprensione più chiara e dettagliata del funzionamento del software, aggiungendo commenti in formato *docstring*, seguendo la ***Google Python Style Guide*** [1], e successivamente, generando dal codice, automaticamente, in formato *Markdown* con *lazydocs*[2], un insieme di file contenenti la documentazione del codice, caricata, infine, nella repository GitHub **ideal-api-docs**. Sono stati, inoltre, aggiunti i *type hints* necessari per migliorare la comprensione delle variabili e degli oggetti utilizzati, riducendo la possibilità di introdurre errori nel codice relativi ai tipi.

4.2 CR_1 Analisi Codice Python

4.2.1 Obiettivo della change request

L'obiettivo di questa change request è estendere la capacità di analisi del tool IDEAL, permettendo l'analisi di file di codice sorgente Python. Questa modifica nasce dalla necessità di avere un tool per effettuare l'analisi dei linguistic antipattern nei sistemi implementati con linguaggi Python, utilizzato soprattutto in sistemi ad alta densità di IA. Per raggiungere questo obiettivo, sarà impiegato un modello di intelligenza artificiale in grado di classificare il codice Python.

4.2.2 Starting Impact Set

La Tabella 5 mostra lo *Starting Impact Set* della change request in esame.

[1] https://google.github.io/styleguide/pyguide.html
[2] https://pypi.org/project/lazydocs/

Tabella 5: Starting Impact Set - CR_1

File	Operazione
src.apps.IDEAL.main	Modifica run_analysis()
src.apps.IDEAL.analyzer	Modifica analyze()
src.apps.IDEAL.result_writer	Modifica save_issues()
src.common.util	Modifica get_supported_file_extensions() e read_input()
src.common.enum	Modifica LanguageType
src.model.input	Modifica costruttore
src.model.issue	Modifica costruttore
src.service.parser	Aggiunta parser Python

4.2.3 Candidate Impact Set

L'approccio utilizzato per la stima del *Candidate Impact Set* è quello dell'analisi delle dipendenze dirette relative ai file inclusi nello *Starting Impact Set*. La motivazione di questa scelta è da ricercare soprattutto nel numero di file presenti nel progetto, che renderebbe troppo complesso utilizzare approcci come matrici delle dipendenze o call graph. Nel *Candidate Impact Set* non saranno inseriti i file del modulo src.nlp.linguistic_antipattern – 24 in totale – nonostante siano dipendenze dirette di *analyzer*, incluso nello *Starting Impact Set*, poiché per questa specifica change request è possibile dire a priori che non verranno modificati.

Tabella 6: Candidate Impact Set - CR_1

File	Operazione
src.apps.IDEAL.main	Modifica run_analysis()
src.apps.IDEAL.analyzer	Modifica analyze()

Continued on next page

Tabella 6: Candidate Impact Set - CR_1 (Continued)

src.apps.IDEAL.result_writer	Modifica save_issues()
src.common.util	Modifica get_supported_file_extensions() e read_input()
src.common.enum	Modifica LanguageType
src.common.testing_list	Dipendenza diretta di src.apps.IDEAL.main
src.common.logger	Dipendenza diretta di src.apps.IDEAL.main
src.common.error_handler	Dipendenza diretta di src.apps.IDEAL.main e src.common.util
src.model.input	Modifica costruttore
src.model.issue	Modifica costruttore
src.model.project	Dipendenza diretta di src.apps.IDEAL.main
src.service.parser	Aggiunta parser Python
src.nlp.splitter	Dipendenza diretta di src.apps.IDEAL.main
src.nlp.pos_tagger_stanford	Dipendenza diretta di src.apps.IDEAL.main
src.service.factory	Dipendenza diretta di src.apps.IDEAL.analyzer

4.2.4 Discoverd Impact Set

. Dopo l'esecuzione delle modifiche, il *Discovered Impact Set* è risultato essere vuoto, poiché nessuna componente non inclusa nel *Candidate Impact Set* è stata coinvolta in questo processo.

4.2.5 Actual Impact Set

Tabella 7: Actual Impact Set - CR_1

File	Operazione
src.apps.IDEAL.main	Modifica run_analysis()
src.apps.IDEAL.analyzer	Modifica analyze()
src.apps.IDEAL.result_writer	Modifica save_issues()
src.common.util	Modifica get_supported_file_extensions() e read_input()
src.common.enum	Modifica LanguageType
src.model.input	Modifica costruttore
src.model.issue	Modifica costruttore
src.service.parser	Aggiunta parser Python

4.2.6 False Positive Impact Set

Dalla tabella 8 è possibile notare che tutti i file inclusi nel *Candidate Impact Set* come dipendenze dirette di file dello *Starting Impact Set*, non sono stati modificati.

Tabella 8: False Positive Impact Set - CR_1

File
src.common.testing_list

Continued on next page

Tabella 8: False Positive Impact Set - CR_1 (Continued)

src.common.logger
src.common.error_handler
src.model.project
src.nlp.splitter
src.nlp.pos_tagger_stanford
src.service.factory

4.2.7 Metriche di processo

Anche per questa change request sono state elaborate diverse metriche per valutare quanto l'approccio utilizzato sia stato d'aiuto al processo di elaborazione delle change request. Di seguito sono elencati i risultati ottenuti:

- **Recall** = $\frac{|CIS \cap AIS|}{|AIS|} = \frac{8}{8} = 1$
  La metrica di Recall è pari ad 1 in quanto il *Discovered Impact Set* è risultato vuoto dopo le modifiche.

- **Precision** = $\frac{|CIS \cap AIS|}{|CIS|} = \frac{8}{15} \approx 0.533$
  La metrica di Precision ci indica che circa il 53% delle componenti del *Candidate Impact Set* è stato soggetto a cambiamenti.

- **Inclusiveness** = 1, perché $AIS \subseteq CIS$
  La metrica di Inclusiveness è pari a 1, in quanto l'*Actual Impact Set* è incluso nel *Candidate Impact Set*.

4.2.8 Implementazione

La maggior parte delle modifiche incluse con questa change request si basano su un prototipo realizzato per l'esame di Software Engineering For Artificial Intelligence, visualizzabile a questo **Link Repository GitHub**.
Il supporto per l'analisi di codice sorgente Python è stato realizzato con un modello di deep learning di NLP, che non verrà sottoposto a test in quanto già testato durante la sua realizzazione. Data la drastica differenza tra le modalità in cui viene effettuata l'analisi con l'approccio IA rispetto al codice già presente in IDEAL è stato necessario aggiungere dei moduli; sono state aggiunte le classi di tipo *model* che rappresentano le varie entità del codice sorgente che possono essere analizzate (funzioni, attributi), le classi per il parsing dei file Python e i moduli per il caricamento del modello di IA e la generazione delle predizioni:

- **src.classifier.classifier**: ha la responsabilità di scaricare automaticamente il modello da internet se non viene trovato nel sistema e caricarlo con la libreria *pytorch*. In particolare il task di download del modello è progettato per essere "lazy": questo significa che il modello viene scaricato solo se non già presente nel sistema e solo nel momento in cui è necessario analizzare un file Python. La scelta di rendere il download automatico è stata presa anche in vista della change request CR_2.

- **src.classifier.predict**: è responsabile della generazione delle predizioni tramite il modello.

- **src.model.greet.***: classi di tipo model responsabili della rappresentazione di attributi e funzioni che possono trovarsi in codice sorgente Python.

Le modifiche al codice esistente, invece, riguardano principalmente l'adattamento di alcune funzioni utilizzate per ricerca e lettura dei file, l'esecuzione dell'analisi generando predizioni sulle entità Python, e il salvataggio dei risultati; in particolare la lista delle modifiche maggiori comprende:

- **src.apps.IDEAL.result_writer**: aggiunta funzione per elaborare il risultato delle predizioni e adattarlo per la scrittura nel file dei risultati.

- **src.apps.IDEAL.analyzer**: aggiunta la logica necessaria a distinguere i file Python per eseguire l'analisi tramite il modello di classificazione.

- **src.apps.IDEAL.main**: aggiornata la chiamata al costruttore di *Analyzer*, sostituendo i parametri relativi alle informazioni sul file da analizzare con l'istanza di *Input* che le contiene.

- **src.common.enum**: aggiunta la chiave "Python" all'enumeratore *LanguageType*.

- **src.common.util**: modificato il metodo *get_supported_file_extensions* per supportare l'estensione "*.py*" e il metodo *read_input* per impostare l'attributo *language* nell'istanza di *Input*.

In seguito all'implementazione delle modifiche, è stata condotta una fase di testing di regressione al fine di assicurare che le modifiche apportate al codice esistente e il nuovo codice introdotto non avessero un impatto negativo sul funzionamento del software preesistente. Il risultato di questa fase di testing è stato positivo, poiché tutti i test precedentemente implementati hanno superato con successo le verifiche. Successivamente, sono stati creati nuovi test per valutare il corretto funzionamento della nuova funzionalità e dei moduli aggiunti. In seguito, è stata effettuata un'ulteriore fase di testing in cui sono stati eseguiti sia i test preesistenti che i nuovi test. Anche

in questa occasione, il risultato del testing è stato positivo, poiché tutti i test hanno dato esito favorevole. I nuovi casi di test sono stati aggiunti nel documento di Test Plan(Post CR1) e di Test Case Specification (Post CR1) i cui link sono presenti nella sezione 6.

La copertura del progetto, in seguito alle modifiche implementate, è stata registrata all'80%, rispetto al precedente 84%. Si è quindi verificata una riduzione del 4% nella copertura. Tuttavia, è importante notare che questa cifra rimane superiore ai requisiti di accettazione specificati nel documento di Test Plan.

4.3 CR_2 Installazione

4.3.1 Obiettivo della change request

Lo scopo di questa change request è semplificare la fase di installazione del tool. L'idea per la realizzazione di questa modifica è nata analizzando le istruzioni di installazione originali fornite dagli autori, visualizzabili a **questo link**: il processo risulta infatti richiede l'installazione di 5 eseguibili esterni al progetto IDEAL, oltre che alla necessità di dover manualmente modificare numerosi file di configurazione relativi alle path di questi eseguibili.
Questi motivi hanno dato luogo alla necessità di semplificare e snellire questo processo per permettere a più utenti di usare IDEAL e quindi, potenzialmente, scrivere codice più manutenibile.

4.3.2 Starting Impact Set

Tabella 9: Starting Impact Set - CR_2

File	Operazione
src.nlp.pos_tagger_stanford	Modifica costruttore e metodo get_pos()
src.service.parser	Modifica __run_srcml()
src.classifier.classifier	Modifica costruttore

4.3.3 Candidate Impact Set

L'approccio utilizzato per stimare il *Candidate Impact Set* si basa sull'analisi del codice sorgente; in particolare sono state identificate le dipendenze dirette dei file dello *Starting Impact Set* e i file che a loro volta li utilizzano.
Il file *src.nlp.pos_tagger_stanford* ha come dipendenze dirette solo *src.common.util*,

che però non viene incluso nell'insieme dal momento che le modifiche previste lasciano invariate interfaccia dei metodi e costruttore.
Tra i file che invece utilizzano *src.nlp.pos_tagger_stanford* ci sono *src.nlp.pos_tag* e *src.apps.IDEAL.main* che analogamente non sono inclusi nel candidate impact set per i motivi precedenti.
Il file *src.service.parser* comprende tra le dipendenze *src.common.util*, *src.model.greet.greet_attribute* e *src.model.greet_function*, ma nuovamente non rientrano tra i file che subiscono un impatto dato che non sono utilizzati nella classe oggetto della modifica. Un solo file ha come dipendenza diretta *src.service.parser*, ed è *src.service.factory*, ed anche in questo caso non è inserito nel set per le motivazioni degli altri file.
L'ultimo file dello *Starting Impact Set* è *src.classifier.classifier* le cui modifiche riguardano solo il cambiamento di una path, per cui non verranno analizzate ne dipendenze dirette né riferimenti.
A valle di queste scelte il *Candidate Impact Set* risulta essere uguale allo *Starting Impact Set*.

4.3.4 Discoverd Impact Set

Il *Discovered Impact Set* dopo le modifiche è risultato vuoto in quanto nessuna componente al di fuori di esso ha preso parte delle modifiche.

4.3.5 Actual Impact Set

Dal momento che il *Discovered Impact Set* è risultato vuoto, allora l'*Actual Impact Set* è risultato essere identico allo s*Starting Impact Set*.

4.3.6 False Positive Impact Set

Il *False Positive Impact Set* è risultato vuoto in quanto tutti gli elementi del *Candidate Impact Set* hanno preso parte delle modifiche.

4.3.7 Metriche di processo

I risultati ottenuti dalle metriche di processo sono:

- **Recall** = $\frac{|CIS \cap AIS|}{|AIS|} = 1$
  Dato che il il *Candidate Impact Set* e l'*Actual Impact Set* sono uguali la cardinalità della loro intersezione è pari alla cardinalità di uno dei due, percui la formula ha valore 1.

- **Precision** = $\frac{|CIS \cap AIS|}{|CIS|} = \frac{3}{3} = 1$
  Dato che il il *Candidate Impact Set* e l'*Actual Impact Set* sono uguali la cardinalità della loro intersezione è pari alla cardinalità di uno dei due, percui la formula ha valore 1.

- **Inclusiveness** = 1, perché $AIS \subseteq CIS$
  Questa metrica è pari a 1, in quanto l'*Actual Impact Set* e il *Candidate Impact Set* sono identici.

Da queste metriche si dimostra che l'approccio utilizzato per identificare il *Candidate Impact Set* si è rivelato efficace nello stimare l'impatto delle modifiche, considerato anche il basso numero di file coinvolti.

4.3.8 Implementazione

L'implementazione principalmente riguarda la modifica del Dockerfile presente nella repository, il quale, presentava problemi nell'esecuzione causati dal non rispetto dei prerequisiti di esecuzione. Per questo motivo sono stati aggiornati i comandi nel modo seguente:

- La base image è stata cambiata in *ubuntu:22.04*
- Sono stati importati i file dalla base image *openjdk:8-jre-slim* per ottenere un'installazione di Java 8
- Segue questi comandi un blocco di installazione delle dipendenze; nello specifico tutte le librerie necessarie ad eseguire srcML, la distribuzione binaria di quest'ultimo, Python 3.10, Stanford POS Tagger e tutte le librerie listate nel file *requirements.txt* tramite il modulo *pip*
- Infine, viene scaricato il modello da utilizzare con Stanford POS Tagger e i dataset e moduli per la parte relativa al natural language processing della libreria *nltk*.

L'aggiornamento del Dockerfile semplifica estremamente l'installazione e la configurazione del tool, visto che per eseguire l'analisi di un insieme di file è sufficiente, per l'utente, buildare il Dockerfile, ed eseguirlo montando eventualmente dei volumi per condividere i file tra l'host ed il container. Inoltre, l'immagine è stata rilasciata anche su DockerHub, a questo link. In ogni caso, IDEAL, può comunque essere eseguito effettuando l'installazione manualmente, e delle guide step-by-step sono state inserite nel file *README* della repo, visualizzabile a **questo link**.
Riguardo le modifiche al codice invece, i file aggiornati sono stati:

- ***src.nlp.pos_tagger_stanford***: in questo file è stato aggiunto un costrutto if che permette di verificare su che piattaforma è in esecuzione il programma in modo da utilizzare le librerie di Python adatte per lanciare il processo Java di Stanford POS Tagger. Sono state, inoltre, aggiornate le espressioni regolari utilizzate con il modulo *pexpect* per controllare l'output del processo lanciato.

- ***src.service.parser***: in modo analogo al file precedente anche in questo caso nel costruttore è stata aggiunta una logica che permette di lanciare il processo in modalità diverse per essere compatibile con il sistema su cui si trova il tool.
- ***src.classifier.classifier***: nel costruttore della classe contenuta in questo file è stata modificata la path di download e caricamento del modello per essere localizzata all'interno del progetto in modo da essere presente nel container Docker quando viene eseguito montando il volume *./:/app/*

Dopo l'attuazione delle modifiche, si è proceduto a modificare la GitHub Action al fine di consentire l'esecuzione dei test all'interno del container Docker.

5 Report finale

Dopo l'implementazione delle richieste di modifica, il software ha subito un processo di evoluzione, presentando nuovi aspetti da valutare. Le operazioni di refactoring hanno notevolmente migliorato la leggibilità del codice, eliminando grossi errori che erano presenti in precedenza. Inoltre, con la corretta Dockerizzazione, è ora possibile effettuare la ricerca di anti-pattern linguistici nel codice Python in modo molto più agevole. L'installazione e l'utilizzo del software sono stati notevolmente semplificati grazie a questa modifica.
Inoltre è stato riscontrato che sfruttare una metodologia agile corredata da una pipeline di CI/CT ha permesso di velocizzare i tempi di progettazione e implementazione delle modifiche oltre a garantire la possibilità di integrare piccoli, frequenti, cambiamenti; oltretutto, è stato possibile scomporre i vari task in più sottoproblemi di minore difficoltà in maniera tale da dividere meglio il lavoro tra il team e completarli più velocemente.
Bisogna notare che le change request con id CR_3, CR_4, CR_5 non hanno preso parte alle modifiche illustrate in questo documento, perché l'effort per la realizzazione dell'intera test suite, e delle change request CR_0 e CR_2, ha azzerato il tempo disponibile.

6 Correlazione con altri documenti

1. Allegato 1: Test Plan v2 post CR1
2. Allegato 2: Test Plan v2 post CR1 (Test Case Specification)
3. Test Plan

Bibliografia

Arnaoudova, Venera et al. (2016). "Linguistic antipatterns: What they are and how developers perceive them". In: *Empirical Software Engineering* 21, pp. 104–158.