UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Ingegneria, Gestione ed Evoluzione del Software

Impact Analysis

TEAM MEMBER

Donia Daniele - 0522501575

La Marca Antonio - 0522501557

Somma Pasquale - 0522501543

Indice

CAPITOLO 1

Approccio

L'Impact Analysis è un processo fondamentale per valutare le conseguenze delle modifiche apportate a un sistema software. Questo documento analizza le change request emerse durante il testing e dalla comprensione del codice, esaminando gli aspetti critici e i componenti coinvolti, sia potenzialmente che effettivamente dopo la loro implementazione.

Verrà utilizzata una matrice di raggiungibilità con approccio distance based (fino al livello 3 di propagazione) per visualizzare le dipendenze e le relazioni tra variabili e funzioni, evidenziando come le modifiche a una componente possano influenzare altre parti del sistema. Questi strumenti forniranno una comprensione chiara della portata delle modifiche e delle aree che richiedono attenzione. Nella matrice di raggiungibilità verranno considerati come componenti gli script (e in seguito le rispettive classi), visto che ogni script andrà ad eseguire sempre ognuno dei suoi metodi una volta avviato e che gli script necessitano sempre dell'esecuzione dell'intero script precedente per poter proseguire. Infine, la valutazione del lavoro sarà effettuata utilizzando le metriche di Recall e Precision per garantire che le modifiche migliorino la qualità del processo di elaborazione, ottimizzando l'efficacia degli script coinvolti.

Ovviamente tutte le modifiche impatteranno i test introdotti o necessiteranno di nuovi test, per questo al termine delle modifiche tutti i test verranno rieseguiti per correggere eventuali test falliti e verificare la nuova coverage. Tutto ciò verrà esaminato nel documento "8_Modification_Testing_and_Regression_Testing".

CAPITOLO 2

Descrizione modifiche Change request 1 (CR1)

Descrizione della Problematica:

Come riportato anche negli incidenti ITI5, ITI8 e ITI9, alcuni moduli del progetto "Perseverance" presentano un errore critico durante l'esecuzione degli script. Il problema si verifica quando il sistema tenta di accedere ai file Java, salvati all'interno di directory strutturate per rappresentare i commit da cui sono stati estratti. L'errore segnalato è un UnicodeDecodeError, causato dalla mancanza di una specificazione dell'encoding nei file di script, rendendo impossibile la lettura o scrittura dei file con caratteri speciali.

Comportamento Attuale:

Attualmente, tutti gli script del modulo Text_Mining tentano di leggere o scrivere file senza specificare l'encoding. Questa mancanza di specificazione porta a errori quando i file Java contengono caratteri speciali o non-ASCII, causando l'interruzione dell'esecuzione con un errore UnicodeDecodeError. Questo impatta l'esecuzione degli script text_mining.py, dict_generator.py, less_element_text_mining.py e creator_csv_for_TextMining.py.

Proposta di Modifica:

Si propone di modificare tutti gli script coinvolti nel modulo TextMining per gestire correttamente la lettura e scrittura dei file, specificando esplicitamente l'encoding 'utf-8' in

tutte le operazioni di I/O. Questo garantirà la corretta gestione dei caratteri speciali nei file Java, prevenendo errori di decodifica. Le modifiche previste sono:

1. Aggiunta del parametro di encoding: in tutte le chiamate alla funzione open() nei quattro script coinvolti, sarà necessario specificare l'encoding in questo modo: open(file_path, 'r', encoding='utf-8') per la lettura e open(file_path, 'w', encoding='utf-8') per la scrittura.

2. Test estensivi: Dopo la modifica, sarà necessario eseguire nuovamente tutti i test di unità e di integrazione del modulo TextMining per verificare che l'aggiunta dell'encoding risolva completamente il problema senza introdurre nuovi errori.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

1. Risoluzione del UnicodeDecodeError: Con l'encoding esplicitamente specificato come 'utf-8', gli script potranno leggere e scrivere correttamente i file che contengono caratteri speciali, evitando errori durante l'esecuzione.

2. Compatibilità con file contenenti caratteri speciali: I file Java e i file di testo all'interno delle directory dei commit verranno gestiti correttamente, anche se contengono caratteri speciali o non-ASCII.

3. Miglioramento della robustezza del modulo TextMining: La gestione corretta dell'encoding aumenterà l'affidabilità del modulo, riducendo la possibilità di future interruzioni dovute a problemi di codifica dei file.

2.1 Analisi d'impatto

Starting Impact Set - 1 Componente

- Script text_mining.py

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set (CIS) è stata generata la matrice di raggiungibilità per ridurre le occorrenze di false positive.

Per le ragioni descritte all'interno del Candidate Impact Set (CIS) abbiamo inserito:

	text_mining
text_mining	-
dict_generator	1
less_elem_tm	2
creator_csv_for_tm	1
Union_TM_with_ASA	2
Union	2
TotalCombination	3

Tabella 2.1: Matrice di raggiungibilità con distanza

Candidate Impact Set - 7 Componenti

- Script text_mining.py
- Script dict_generator.py
- Script less_elem_tm.py
- Script creator_csv_for_tm.py
- Script Union_TM_with_ASA.py
- Script Union.py
- Script TotalCombination.py

2.2 Implementazione modifiche

La modifica ha richiesto unicamente l'aggiunta del parametro 'encoding' nelle chiamate alla funzione open, col valore impostato a 'utf-8'. (Inoltre, le chiamate alla funzione open avvengono ora col costrutto with, così da assicurare la chiusura del file).

Actual Impact Set - 8 Componenti

- Script text_mining.py
- Script dict_generator.py
- Script less_elem_tm.py

- Script creator_csv_for_tm.py
- Script Union_TM_with_ASA.py
- Script Union.py
- Script TotalCombination.py
- Script repo_mining.py

False Positive Impact Set - 0 Componente

Discovered Impact Set - 1 Componente

- Script repo_mining.py

2.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = |(CIS ∩ AIS)| / |AIS| = 7 / 8 = 0.875
- Precision = |(CIS ∩ AIS)| / |CIS| = 7 / 7 = 1.0

CAPITOLO 3

Descrizione modifiche Change request 2 (CR2)

Descrizione della Problematica 1:

Come osservato per l'incidente ITI3, il modulo responsabile del mining delle repository è caratterizzato da una navigazione non corretta del file system. Ciò causa il fallimento del sistema se si estraggono classi con estensione .java; il sistema tenta di spostarsi nella directory in cui salvare le classi, utilizzando il parametro `cwd` della funzione startMiningRepo, ma il percorso specificato da cwd non è esistente, in quanto non include la directory `mining_results`.

Comportamento Attuale:

Attualmente, l'esecuzione di main_repo_Mining.py con un dataset contenente un commit valido con modifiche a classi .java, comporta la generazione di un `FileNotFoundError`.

Proposta di Modifica:

Per risolvere i problemi mostrati, nella funzione `initialize` si propone:

- estrazione dalla variabile `cwd`, ovvero il percorso corrente, del percorso padre
- modica parametro per os.chdir con concatenazione del percorso estratto con `mining_results`

Dopo la modifica, inoltre, sarà necessario eseguire nuovamente tutti i test di unità e di integrazione del modulo RepoMining per verificare che le modifiche introdotte risolvano completamente il problema senza introdurre nuovi errori.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

- Risoluzione errori: la corretta definizione e gestione dei percorsi delle directory risolverà i problemi legati alla navigazione del file system, eliminando l'errore FileNotFoundError che attualmente causa il fallimento dell'intero processo di mining quando vengono estratti file con estensione .java.

Descrizione della Problematica 2:

Il sistema di unione dei dati, che utilizza quelli ottenuti dal text mining, si basa sul file CSV che si trova nella directory `Text_Mining`. Tuttavia, il CSV aggiornato, prodotto dall'elaborazione, viene salvato nella directory `mining_results`. C'è quindi una incosistenza tra i percorsi utilizzati e quelli da utilizzare.

Comportamento Attuale:

Questa situazione causa un errore durante le operazioni di unione, poiché il file CSV non è presente all'interno della cartella di `Text_Mining`.

Proposta di Modifica:

Per risolvere il problema descritto, si propone di modificare gli script di Union affinché vadano a recuperare i file CSV più aggiornati direttamente dalla directory `mining_results`. Le modifiche includeranno:

- Modifica del percorso di origine dei file CSV in Union, puntando alla directory `mining_results` anziché `Text_Mining`.

L'implementazione di questa modifica porterà i seguenti benefici:

- Correttezza dei dati: Union utilizzerà sempre i file CSV più aggiornati dalla directory `mining_results`, eliminando il rischio di mancata disponibilità dei file necessari.
- Prevenzione di errori: la modifica eviterà errori causati dall'assenza dei file CSV in `Text_Mining`, garantendo che Union non fallisca a causa di file mancanti.

3.1 Analisi d'impatto

Starting Impact Set - 3 Componenti

- repo_mining
- Union.py
- Union_TMwithASA.py (Un_TM_ASA)

	repo_mining	**Union.py**	**Un_TM_ASA**
repo_mining	-	-	-
Union	3	-	-
Un_TM_ASA	3	-	-
Un_SM_ASA	-	-	-
text_mining	1	-	-
dict_generator	2	-	-
less_elem_tm	3	-	-
creator_csv_for_tm	2	-	-
TotalCombination	4	-	1

Tabella 3.1: Matrice di raggiungibilità con distanza

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set (CIS) è stata generata la matrice di raggiungibilità per ridurre le occorrenze di false positive. In particolare, sono state considerate le componenti del modulo responsabile per l'estrazione, in quanto le modifiche alla navigazione del file system riguardano tale modulo, e le componenti dei moduli di text-mining e union, in cui vi è una discrepanza tra il path di salvataggio e di utilizzo del file csv_mining_final.csv.

Per le ragioni descritte, all'interno del Candidate Impact Set (CIS) abbiamo inserito:

Candidate Impact Set - 7 Componenti

- repo_mining.py
- Union.py
- Union_TM_withASA.py

- text_mining.py
- dict_generator
- creator_csv_for_tm.py
- TotalCombination.py

3.2 Implementazione modifiche

L'implementazione della modifica richiede:

- l'estrazione dal parametro `cwd` del percorso padre nella funzione initialize di repo_mining.py
- la specifica della directory corrente come la concatenazione del percorso estratto con `mining_results` nella funzione start_mining_repo di repo_mining.py
- modifica del percorso di csv_mining_final negli script Union.py e Union_TMwithASA.py

Actual Impact Set - 3 Componenti

- repo_mining.py
- Union.py
- Union_TMwithASA.py

False Positive Impact Set - 2 Componenti

- text_mining.py
- dict_generator.py
- creator_csv_for_tm.py
- TotalCombination.py

Discovered Impact Set - 0 Componenti

3.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = |(CIS ∩ AIS)| / |AIS| = 3 / 3 = 1.0
- Precision = |(CIS ∩ AIS)| / |CIS| = 3 / 7 = 0.42

CAPITOLO 4

Descrizione modifiche Change request 4 (CR4)

Descrizione della Problematica:

La problematica principale affrontata in questo contesto riguarda la frammentazione della gestione del codice, dovuta all'esecuzione di script separati e indipendenti nel sistema. Questa architettura aumenta la complessità nella manutenzione del software e complica l'uso del sistema, poiché ogni script richiede esecuzioni individuali e non esiste un flusso unificato per l'elaborazione dei dati. Ciò comporta anche una maggiore difficoltà nel debugging e nell'estensione delle funzionalità.

Comportamento Attuale:

Attualmente, ogni script deve essere eseguito singolarmente e non esiste un'architettura modulare che consenta l'interazione tra gli script in modo efficiente. Questo approccio frammentario rende l'uso del sistema più complesso e limita la riusabilità del codice, richiedendo maggiore tempo per gestire l'esecuzione e la manutenzione delle diverse parti del software.

Proposta di Modifica:

La proposta di modifica prevede la conversione del sistema in un tool orientato agli oggetti (OOP) in Python, attraverso la ristrutturazione del codice esistente. Questo comporta:

1. Riprogettare gli script separati come classi Python, eliminando l'uso dei metodi main specifici per ciascun file.

2. Introdurre una classe principale (Main) che gestisca l'esecuzione centralizzata del sistema, unificando il processo di esecuzione.

3. Migliorare la modularità del sistema, permettendo il riutilizzo del codice e semplificando la manutenzione e l'estensione del software.

Benefici previsti:

L'implementazione di questa modifica apporterà i seguenti benefici:

1. Modularità e riusabilità: la riorganizzazione orientata agli oggetti permetterà di riutilizzare le classi in altri contesti, migliorando la scalabilità e riducendo il tempo necessario per sviluppare nuove funzionalità.

2. Manutenzione semplificata: la separazione del codice in classi e l'introduzione di un flusso unificato miglioreranno la leggibilità e la gestione del codice, riducendo la complessità e facilitando il debugging.

3. Esecuzione centralizzata: grazie alla classe Main, sarà possibile eseguire il sistema in modo unificato, riducendo la frammentazione e l'errore umano nell'esecuzione manuale di script multipli.

4. Affidabilità e robustezza: un sistema meglio strutturato e meno frammentato ridurrà la probabilità di errori e incoerenze durante l'esecuzione.

4.1 Analisi d'impatto

L'impatto della modifica riguarda l'intero sistema, in quanto tutti gli script dovranno essere modificati per rispettare gli standard dell'OOP. Per questo motivo non riteniamo sia utile realizzare la matrice di raggiungibilità.

Starting Impact Set - 15 Componenti

- Script repo_Mining.py
- Script main_repo_Mining.py
- Script divide_Dataset.py
- Script text_mining.py

- Script dict_generator.py
- Script less_elem_tm.py
- Script creator_csv_for_tm.py
- Script Union_TM_with_ASA.py
- Script Union_SMwithASA.py
- Script Union.py
- Script TotalCombination.py
- Script ASA_vulnerability_dict_generator.py
- Script rules_dict_generator_ASA.py
- Script creator_csv_for_ASA.py
- Script add_classes_type.py

Candidate Impact Set - 15 Componenti

- Script repo_Mining.py
- Script main_repo_Mining.py
- Script divide_Dataset.py
- Script text_mining.py
- Script dict_generator.py
- Script less_elem_tm.py
- Script creator_csv_for_tm.py
- Script Union_TM_with_ASA.py
- Script Union_SMwithASA.py
- Script Union.py
- Script TotalCombination.py
- Script ASA_vulnerability_dict_generator.py

- Script rules_dict_generator_ASA.py
- Script creator_csv_for_ASA.py
- Script add_classes_type.py

4.2 Implementazione modifiche

Il sistema originariamente basato su script è stato riorganizzato, convertendo la struttura in un'architettura a classi. A seguito di questa trasformazione, è stato introdotto un unico file principale (main) per gestire l'esecuzione complessiva del sistema, che presenta più metodi, ognuno per avviare le diverse elaborazioni del sistema. Di conseguenza, sono stati rimossi i vari metodi main presenti all'interno dei singoli script (se avevano come unica responsabilità l'avvio dell'esecuzione, altrimenti sono diventati metodi della classe), nonché lo script denominato main_repo_Mining.py, che si occupava esclusivamente di avviare l'esecuzione dello script repo_Mining.py. Inoltre, alcuni script che presentavano funzioni simili o che operavano in sequenza (ad esempio, uno script elaborava l'output dell'altro) e contenevano ciascuno un singolo metodo, sono stati accorpati in un'unica classe per migliorare l'organizzazione del codice.
In particolare, gli script sono stati trasformati come segue:

- Lo script divide_Dataset.py è stato sostituito dalla classe DatasetDivider;
- Lo script repo_Mining.py è stato sostituito dalla classe RepoMiner;
- Gli script text_mining.py, less_element_tm.py e dict_generator.py sono stati sostituiti dalla classe JavaTextMining, mentre lo script creator_csv_for_tm.py è stato sostituito dalla classe CSVWriter;
- Gli script ASA_vulnerability_dict_generator.py e rules_dict_generator_ASA.py sono stati sostituiti dalla classe DictGenerator, mentre lo script creator_csv_for_ASA.py è stato sostituito dalla classe CSVCreatorForAsa;
- Gli script Union_TM_with_ASA.py, Union_SMwithASA.py, Union.py e TotalCombination.py sono stati sostituiti dalla classe DatasetCombiner.

Vediamo, adesso, attraverso un Class Diagram, come si presenta il nostro sistema:

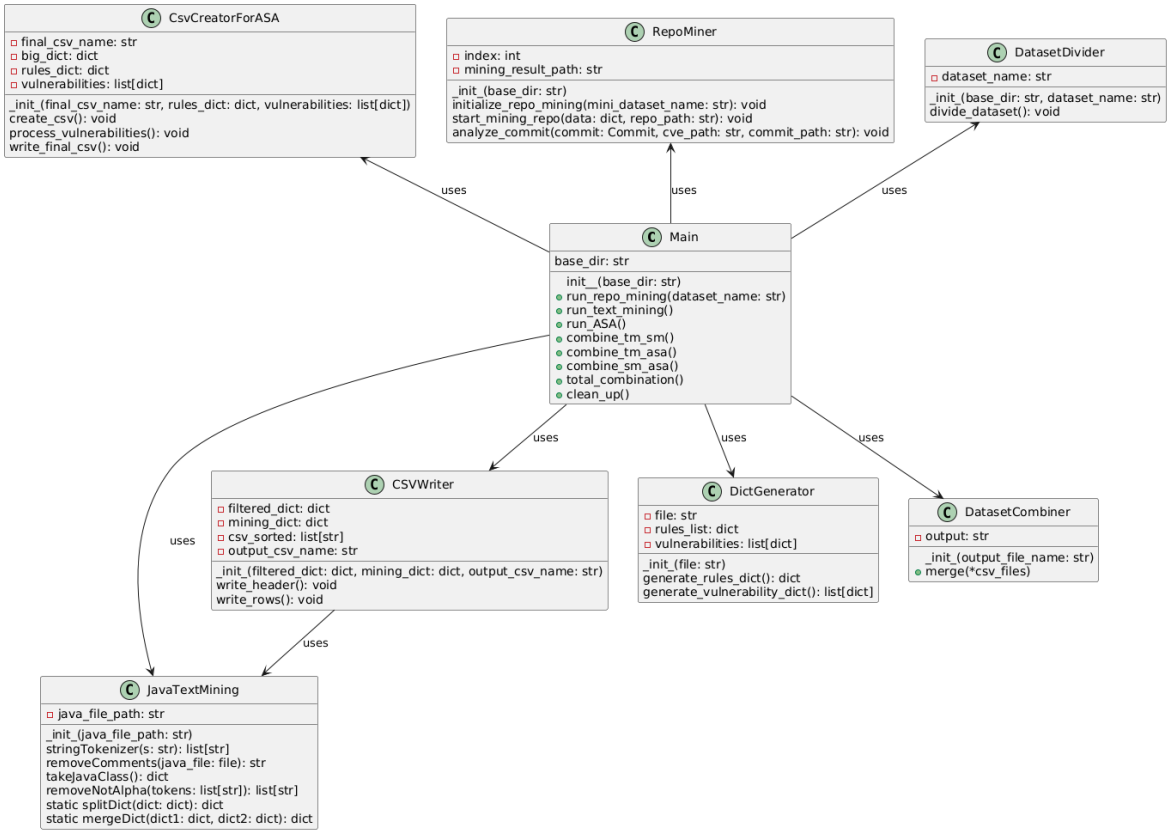

Figura 4.1: Class Diagram del sistema dopo la CR4

Actual Impact Set - 15 Componenti

- Script repo_Mining.py
- Script main_repo_Mining.py
- Script divide_Dataset.py
- Script text_mining.py
- Script dict_generator.py
- Script less_elem_tm.py
- Script creator_csv_for_tm.py
- Script Union_TM_with_ASA.py
- Script Union_SMwithASA.py
- Script Union.py

- Script TotalCombination.py
- Script ASA_vulnerability_dict_generator.py
- Script rules_dict_generator_ASA.py
- Script creator_csv_for_ASA.py
- Script add_classes_type.py

False Positive Impact Set - 0 Componenti

Discovered Impact Set - 0 Componenti

4.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = $|(CIS \cap AIS)| \ / \ |AIS| = 15 \ / \ 15 = 1.0$
- Precision = $|(CIS \cap AIS)| \ / \ |CIS| = 15 \ / \ 15 = 1.0$

CAPITOLO 5

Descrizione modifiche Change request 5 (CR5)

Descrizione della Problematica:

Attualmente, l'utente si affida a un tool a pagamento, Understand, per estrarre le metriche software dal codice sorgente. Questo strumento genera un progetto che contiene i file sorgente di interesse, successivamente utilizzati per creare file in formato CSV con i risultati delle metriche calcolate. La dipendenza da un software commerciale introduce costi e vincoli che si vorrebbe eliminare, al fine di rendere il processo più flessibile ed economico.

Comportamento Attuale:

Attualmente, il tool Understand viene utilizzato come soluzione esterna per raccogliere le metriche del codice sorgente e creare i file CSV. Il processo è ben funzionante ma dipende interamente da un software a pagamento, e ogni modifica alle metriche o al progetto richiede l'uso di questo strumento specifico, limitando l'adattabilità del sistema.

Proposta di Modifica:

La proposta di modifica mira a rimuovere la dipendenza dal tool Understand, sviluppando un modulo home-made per l'estrazione delle metriche software direttamente dal codice sorgente. Il nuovo modulo sarà integrato nel sistema esistente e dovrà:

1. Calcolare le seguenti metriche software in modo indipendente e automatico:
    - il numero di dichiarazioni di classi presenti;

- il numero di dichiarazioni di funzioni presenti;
- il numero di righe di codice che rappresentano dichiarazioni;
- la somma delle complessità essenziali delle funzioni presenti;
- il valore massimo della complessità essenziale tra le funzioni presenti;
- la somma delle complessità ciclomatiche delle funzioni presenti;
- il valore massimo della complessità ciclomatica delle funzioni presenti;
- la massima profondità di annidamento.

2. Creare i file CSV contenenti i risultati, mantenendo la compatibilità con i flussi di lavoro esistenti.

3. Essere modulare e facilmente adattabile per future estensioni delle funzionalità.

Sarà inoltre necessario aggiornare il file Main.py per includere il nuovo modulo e permetterne l'esecuzione come parte del sistema unificato.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

1. Eliminazione dei costi del tool commerciale: Sviluppando un modulo interno, si elimina la dipendenza da Understand e i costi associati alla licenza.

2. Maggiore flessibilità: il nuovo modulo home-made sarà interamente controllabile e adattabile alle esigenze specifiche del progetto, consentendo modifiche e aggiornamenti senza vincoli esterni.

3. Integrazione nel sistema esistente: l'aggiunta del modulo all'interno del sistema permetterà una gestione unificata e integrata delle metriche, riducendo la frammentazione dei processi.

4. Migliore mantenibilità: avere il codice per l'estrazione delle metriche come parte del sistema ne faciliterà il mantenimento e l'aggiornamento, migliorando la manutenibilità a lungo termine.

5.1 Analisi d'impatto

La modifica porta all'introduzione di una nuova classe, con i metodi necessari per calcolare alcune metriche del software del codice sorgente in modo automatico e gratuito, una volta

calcolate tramite il tool Understand agendo esternamente al sistema. Proprio perchè non vi sono componenti atte a effettuare queste elaborazioni, ma ci si basa su metriche calcolate a priori, non vi sono componenti impattate da questa modifica. L'unica è il metodo Main, introdotto con la CR6, che può lanciare l'esecuzione dei metodi per il calcolo delle metriche. Verranno introdotti inoltre dei metodi di test per verificare il corretto funzionamente di questa nuova classe.

Starting Impact Set - 1 Componente

- Main: run

Candidate Impact Set - 1 Componente

- Main: run

5.2 Implementazione modifiche

L'implementazione per il calcolo delle metriche del software, ha richiesto l'utilizzo principalmente della libreria lizard, per il calcolo della complessità ciclomatica, e di librerie (come javalang e tree_sitter) per la creazione dell'AST a partire dal codice dei vari file java per poter navigare all'interno dello stesso ed estrarre le altre metriche, calcolcate seguendo la teoria e in modo quanto più simile al tool Understand. Ciò ha portato alla realizzazione di una nuova classe SoftwareMetrics, con diversi metodi, ognuno dedito al calcolo di una metrica.

Esaminiamo il funzionamento interno della nuova classe osservando il suo sequence diagram:

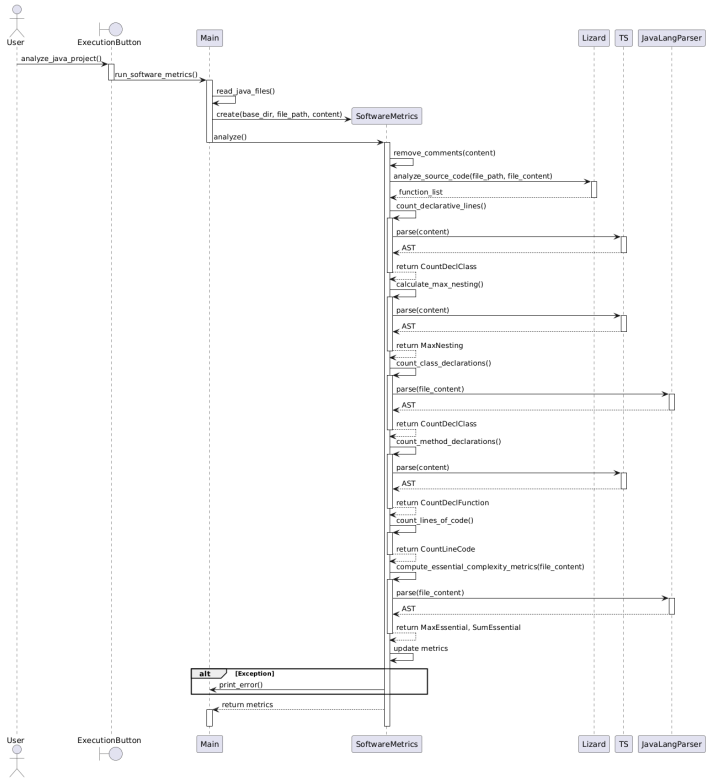

Figura 5.1: Sequence diagram SoftwareMetrics

Vediamo com'è cambiato il Class Diagram del sistema dopo l'introduzione di questo nuovo modulo:

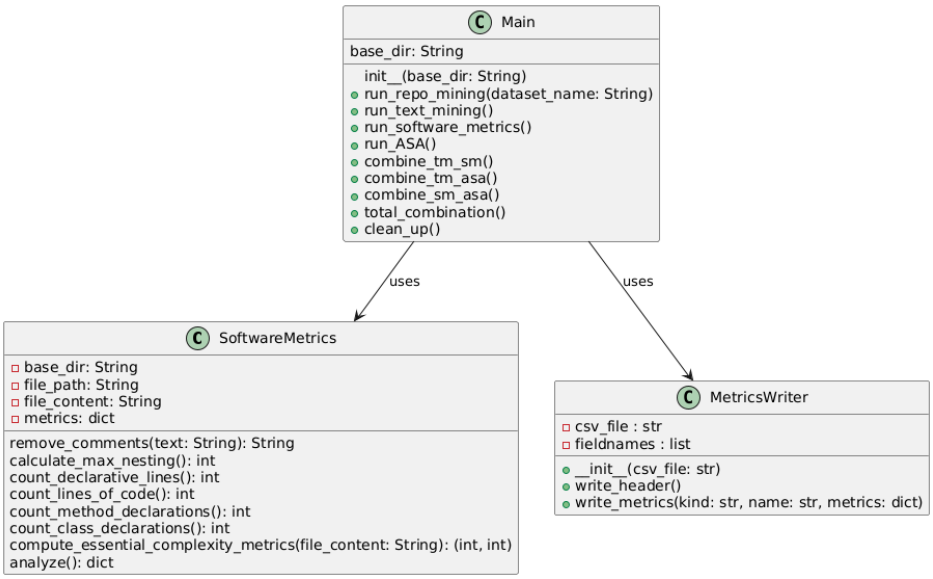

Figura 5.2: Class Diagram delle parti del sistema modificate dopo la CR5

Actual Impact Set - 1 Componente

- Main: run

False Positive Impact Set - 0 Componenti

Discovered Impact Set - 0 Componenti

5.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = |(CIS ∩ AIS)| / |AIS| = 1 / 1 = 1.0
- Precision = |(CIS ∩ AIS)| / |CIS| = 1 / 1 = 1.0

CAPITOLO 6

Descrizione modifiche Change request 6 (CR6)

Descrizione della Problematica:

Attualmente, l'utente utilizza il tool Sonarqube per eseguire l'analisi statica automatica delle repository, esportando i risultati tramite il plugin CNESREPORT in formato CSV. Questo processo avviene esternamente al sistema e richiede interventi manuali per l'esportazione e l'integrazione dei risultati, riducendo l'efficienza del flusso di lavoro e aumentando la frammentazione tra i vari strumenti utilizzati.

Comportamento Attuale:

Il tool Sonarqube viene attualmente impiegato come una soluzione esterna per eseguire l'analisi statica del codice sorgente. I risultati vengono poi esportati manualmente tramite il plugin CNESREPORT e riportati in file CSV, i quali vengono successivamente elaborati per ulteriori analisi. Tuttavia, non vi è un'automazione nel processo di esecuzione o nell'integrazione dei dati risultanti all'interno del sistema esistente.

Proposta di Modifica:

La proposta prevede l'integrazione dell'analisi statica automatica all'interno del sistema esistente, utilizzando SonarScanner per eseguire l'analisi del codice sorgente e le API di SonarQube per raccogliere i risultati in modo automatico. Questo comporterà:

1. La creazione di un modulo che consenta di avviare automaticamente l'analisi delle repository tramite SonarScanner, che eseguirà la scansione del codice e invierà i dati a SonarQube.

2. L'estrazione e l'integrazione automatica dei risultati in formato CSV utilizzando le API di SonarQube, riducendo la necessità di intervento manuale.

3. La modifica del file Main.py per abilitare l'esecuzione del nuovo modulo e integrarlo nel flusso di lavoro generale, consentendo l'analisi automatizzata e la gestione dei risultati.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

1. Automazione del processo di analisi statica: L'integrazione delle API di Sonarqube permetterà l'automazione dell'analisi delle repository, riducendo i tempi di esecuzione e aumentando l'efficienza.

2. Riduzione degli interventi manuali: Con l'automatizzazione dell'esportazione e dell'integrazione dei risultati, si eliminerà la necessità di interventi manuali, riducendo i rischi di errore e migliorando la coerenza dei dati.

3. Maggiore efficienza e coerenza: Automatizzando il processo e centralizzando i risultati all'interno del sistema, si migliorerà l'efficienza complessiva del flusso di lavoro, con una maggiore coerenza tra i dati raccolti e quelli processati.

4. Integrazione diretta nel sistema esistente: L'analisi statica diventerà parte integrante del sistema, semplificando il processo di mantenimento e aggiornamento e migliorando la modularità e manutenibilità del codice.

6.1 Analisi d'impatto

La modifica porta all'introduzione di una nuova classe, con i metodi necessari per verificare quali vulnerabilità sono presenti nel codice sorgente in modo automatico tramite l'API di Sonarqube, precedentemente estratte tramite il sito stesso di Sonarqube, agendo però esternamente al sistema. Proprio perchè non vi sono componenti atte a effettuare queste elaborazioni, ma ci si basa su vulnerabilità estratte a priori, non vi sono componenti impattate

da questa modifica. L'unica è il metodo Main, introdotto con la CR6, che può lanciare l'esecuzione dei metodi per il calcolo delle vulnerabilità. Inoltre, il metodo del main per eseguire l'elaborazione dei risultati ottenuti da SonarQube (run_ASA), è stato modificato affinchè utilizzi un solo csv come input e non due, come in precedenza. Verranno introdotti inoltre dei metodi di test per verificare il corretto funzionamente di questa nuova classe.

Starting Impact Set - 1 Componente

- Main: run

Candidate Impact Set - 1 Componente

- Main: run

6.2 Implementazione modifiche

L'implementazione ha portato alla creazione della classe SonarAnalyzer, che automatizza il processo di analisi statica dei progetti Java, integrando SonarQube e SonarScanner. La classe gestisce l'intero flusso: genera automaticamente i file di configurazione necessari per ogni progetto, avvia l'analisi del codice tramite SonarScanner e raccoglie i risultati dall'API di SonarQube. In caso di errori durante l'analisi, come problemi di parsing o connessione, la classe li gestisce e li registra in un log. Una volta conclusa l'analisi, i risultati vengono salvati in un file CSV, dove ogni issue rilevata è documentata. Se non vengono trovate issue, la classe registra comunque l'assenza di issue, assicurando una traccia completa dell'analisi. La classe necessita di tre input principali, ovvero: l'URL del server SonarQube, il token di autenticazione per accedere al server SonarQube e il percorso al file eseguibile di SonarScanner.

Esaminiamo il funzionamento interno della nuova classe osservando il suo sequence diagram:

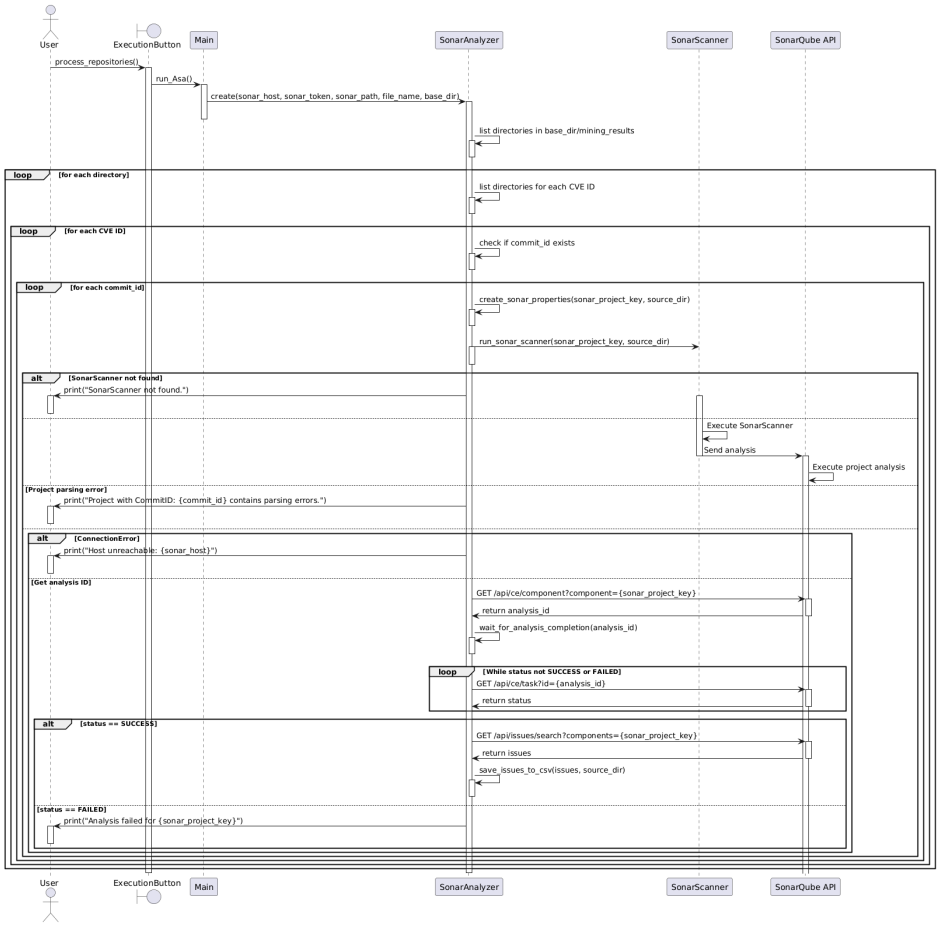

Figura 6.1: Sequence diagram SonarAnalyzer

Vediamo com'è cambiato il Class Diagram del sistema dopo l'introduzione di questo nuovo modulo:

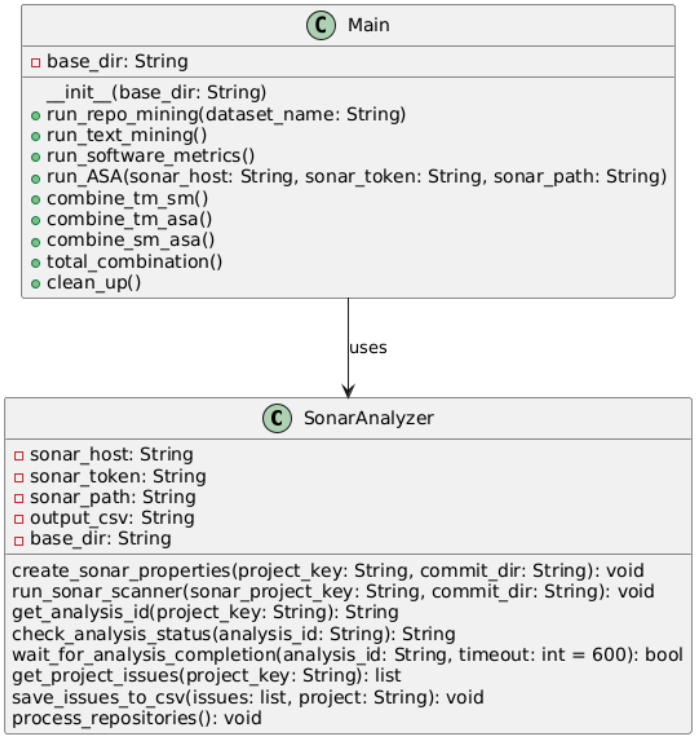

Figura 6.2: Class Diagram delle parti del sistema modificate dopo la CR6

Actual Impact Set - 1 Componente

- Main: run

False Positive Impact Set - 0 Componenti

Discovered Impact Set - 0 Componenti

6.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = $|(CIS \cap AIS)| \ / \ |AIS| = 1 \ / \ 1 = 1.0$
- Precision = $|(CIS \cap AIS)| \ / \ |CIS| = 1 \ / \ 1 = 1.0$

CAPITOLO 7

Descrizione modifiche Change request 7 (CR7)

Descrizione della Problematica:

Attualmente, sono stati condotti studi approfonditi su quattro diverse tecniche di intelligenza artificiale, ognuna addestrata su metriche e analisi del codice differenti, e sulle loro combinazioni. Tuttavia, i modelli risultanti non sono attualmente integrati nel sistema, il che impedisce agli utenti di usufruire della funzionalità di predizione delle vulnerabilità.

Comportamento Attuale:

Sebbene siano stati sviluppati modelli promettenti tramite l'analisi delle tecniche di intelligenza artificiale, il sistema non è in grado di fornire previsioni sulle vulnerabilità a causa della mancanza di integrazione. Ciò limita la capacità degli utenti di sfruttare questi avanzamenti nella predizione delle vulnerabilità e influisce negativamente sull'efficacia del sistema nella gestione della sicurezza.

Proposta di Modifica:

La modifica proposta prevede la creazione di sette versioni diverse della tecnica di intelligenza artificiale che ha mostrato le migliori prestazioni nei test condotti sui vari dataset. Queste versioni saranno integrate nel sistema per fornire funzionalità di predizione delle vulnerabilità su nuovi input, consentendo agli utenti di identificare proattivamente i potenziali rischi.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

1. Abilitazione della predizione delle vulnerabilità: Gli utenti potranno ricevere informazioni riguardo alle vulnerabilità nel codice, migliorando la sicurezza complessiva.

2. Integrazione fluida con il sistema esistente: Non si prevede un impatto significativo sulla struttura attuale del sistema, poiché l'integrazione delle nuove tecniche non interferirà con le analisi già presenti.

3. Miglioramento delle performance: Le nuove versioni dei modelli di intelligenza artificiale possono aumentare l'accuratezza delle previsioni, fornendo agli utenti risultati più affidabili.

4. Semplificazione dell'uso del sistema: Sarà necessario modificare il file Main per consentire l'esecuzione del nuovo modulo, gestire l'inserimento di nuove istanze e ottenere previsioni dal modello, rendendo il sistema più intuitivo e user-friendly.

7.1 Analisi d'impatto

L'implementazione del modulo non avrà impatto su nessuno degli script e delle classi esistenti, in quanto esterno alle varie analisi condotte sul dataset al momento.

7.2 Implementazione modifiche

L'implementazione ha richiesto lo sviluppo di uno script per l'addestramento e la generazione di predizioni utilizzando vari modelli di Random Forest, ciascuno applicato ai diversi dataset ottenuti dall'analisi del dataset di repository, per i quali è già noto se contengono vulnerabilità. Il modulo prevede anche il salvataggio dei vocabolari, ovvero dizionari contenenti i nomi delle colonne su cui i modelli sono stati addestrati. Questo passaggio è essenziale poiché gli input per le predizioni potrebbero variare (in particolare quelli ottenuti tramite text mining e l'analisi con SonarQube) e devono essere coerenti con le caratteristiche utilizzate per l'addestramento del modello. Infine, è stato aggiunto alla classe Main, il metodo run_prediction, che prende in input il csv contenente i dati su cui effettuare la predizione, il percorso del modello, del vocabolario e del label encoder da utilizzare in conformità ai dati del csv e il percorso dove salvare i risultati.

CAPITOLO 8

Descrizione modifiche Change request 8 (CR8)

Descrizione della Problematica:

Attualmente, l'utente interagisce con il sistema Perseverance tramite l'esecuzione di diversi script da linea di comando, senza alcun supporto grafico che faciliti l'uso del software. Questo approccio rende l'interazione meno intuitiva e richiede una conoscenza tecnica dettagliata da parte dell'utente, riducendo l'accessibilità del sistema.

Comportamento Attuale:

Attualmente, l'utente esegue manualmente più script per avviare i vari processi di mining, analisi e generazione dei risultati. Non è presente alcuna interfaccia grafica, quindi tutte le interazioni devono essere effettuate tramite terminale, senza supporto visivo o un'interfaccia che guidi l'utente nelle operazioni, come il caricamento di dataset o la configurazione dei parametri di esecuzione.

Proposta di Modifica:

La proposta prevede l'integrazione di una Graphical User Interface (GUI) per migliorare l'esperienza utente. I principali cambiamenti includono:

1. La creazione di un eseguibile contenente interfacce grafiche che permettano all'utente di caricare dataset, configurare i parametri, e scaricare i risultati finali.

2. La modifica del file Main.py per gestire l'avvio del sistema tramite un'interfaccia grafica invece dell'esecuzione manuale.

3. L'introduzione di schermate guidate per ogni fase, dal caricamento dei dati alla scelta delle opzioni di elaborazione, fino al download dei risultati.

Benefici previsti:

L'implementazione di questa modifica porterà i seguenti benefici:

1. Facilitazione dell'interazione: La GUI semplificherà l'uso del sistema, consentendo all'utente di interagire con il software senza la necessità di digitare comandi manualmente, rendendo Perseverance accessibile anche a utenti non esperti.

2. Maggiore usabilità: La struttura guidata dell'interfaccia migliorerà l'esperienza utente, offrendo un flusso logico per ogni operazione, dal caricamento dei dataset alla configurazione dei parametri.

3. Riduzione degli errori manuali: Una GUI ben progettata ridurrà gli errori causati dall'input manuale errato, poiché l'utente sarà guidato passo dopo passo attraverso le varie fasi di utilizzo.

4. Aumento dell'efficienza: L'integrazione della GUI migliorerà la velocità con cui l'utente può avviare i processi, riducendo la necessità di ripetere manualmente operazioni complesse.

8.1 Analisi d'impatto

Le classi interessate dalle modifiche riguardo principalmente il cambio dei path, che nel sistema sono fissi e posso portare facilmente ad errori. Per questo abbiamo:

Starting Impact Set - 1 Componente

- Class Main

- Class DatasetDivider

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set (CIS) è stata generata la matrice di raggiungibilità per cercare di ridurre le occorrenze di false positive. Per le ragioni descritte, all'interno del Candidate Impact Set (CIS) abbiamo inserito:

	Main	DatasetDivider
Main	-	-
DatasetDivider	1	-
RepoMiner	1	1
CSVWriter	1	3
JavaTextMining	1	2
MetricsWriter	1	3
SoftwareMetrics	1	2
SonarAnalyzer	1	2
DictGenerator	1	3
CsvCreatorForAsa	1	4
DatasetCombiner	1	4

Tabella 8.1: Matrice di raggiungibilità con distanza

Candidate Impact Set - 9 Componenti

- Class Main
- Class DatasetDivider
- Class RepoMiner
- Class CSVWriter
- Class JavaTextMining
- Class MetricsWriter
- Class SoftwareMetrics
- Class SonarAnalyzer
- Class DictGenerator

8.2 Implementazione modifiche

L'implementazione ha portato alla realizzazione di una classe GUI, che gestisce l'interfaccia grafica in cui poter scegliere il tipo di eleborazione richiesta tra le tre disponibili, il download dei rispetti risultati (introducendo il metodo download_results nella classe Main)

e le varie predizioni ottenute dal modello di intelligenza artificiale. Si è cercato inoltre di rendere il tool più generico, modificando la navigazione all'interno del sistema tramite i path. Per far ciò, abbiamo sfruttato il parametro base_dir della classe Main, all'interno dei suoi stessi metodi e che è stato aggiunto come parametro alle classi RepoMiner e DatasetDivider. È stata eliminata la colonna 'class' all'interno dei vari CSV (che non sarà più diponibile già dall'inizio), calcolato a run-time il numero di cartelle di RepositoryMining create (prima fisso a 35) ed eliminato la limitazione sulla RepositoryMining18, mai creata nel precedente sistema. Infine, è stata introdotta una classe ExeBuilder per costruire un eseguibile con cui poter avviare il tool. La realizzazione della classe GUI ha apportato le modifiche che possiamo osservare attraverso il seguente Sequence Diagram:

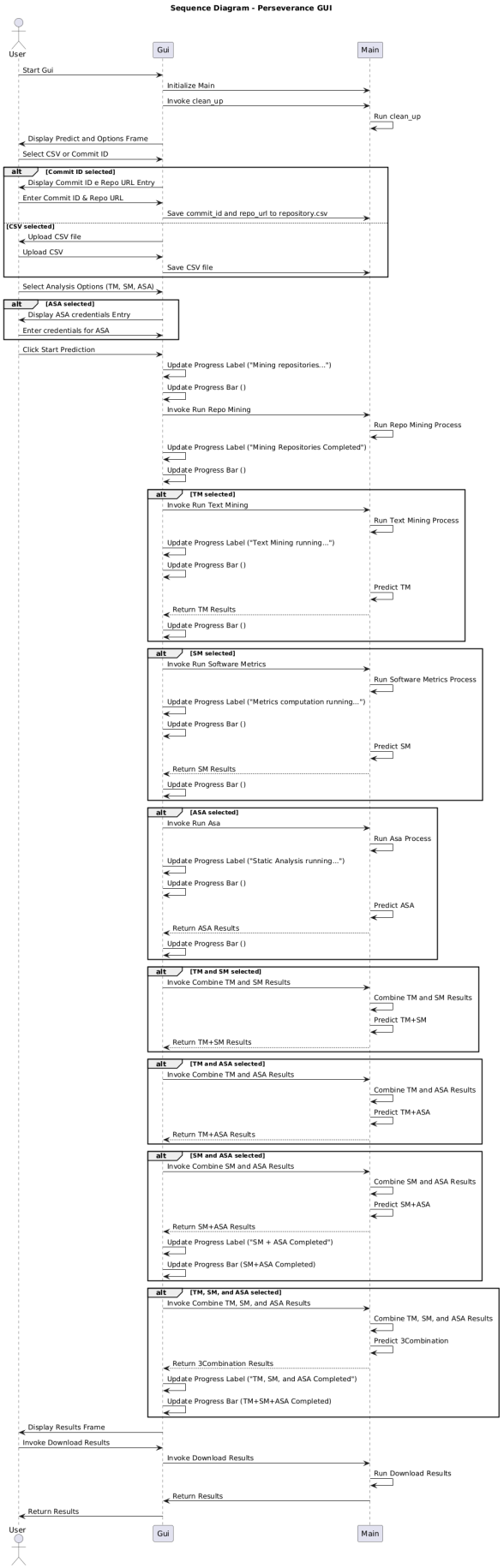

Figura 8.1: Sequence Diagram delle parti del sistema modificate dopo la CR8

Vediamo, con l'implementazione di questa CR, come si collocano le principali aggiunte all'interno del Class Diagram del sistema:

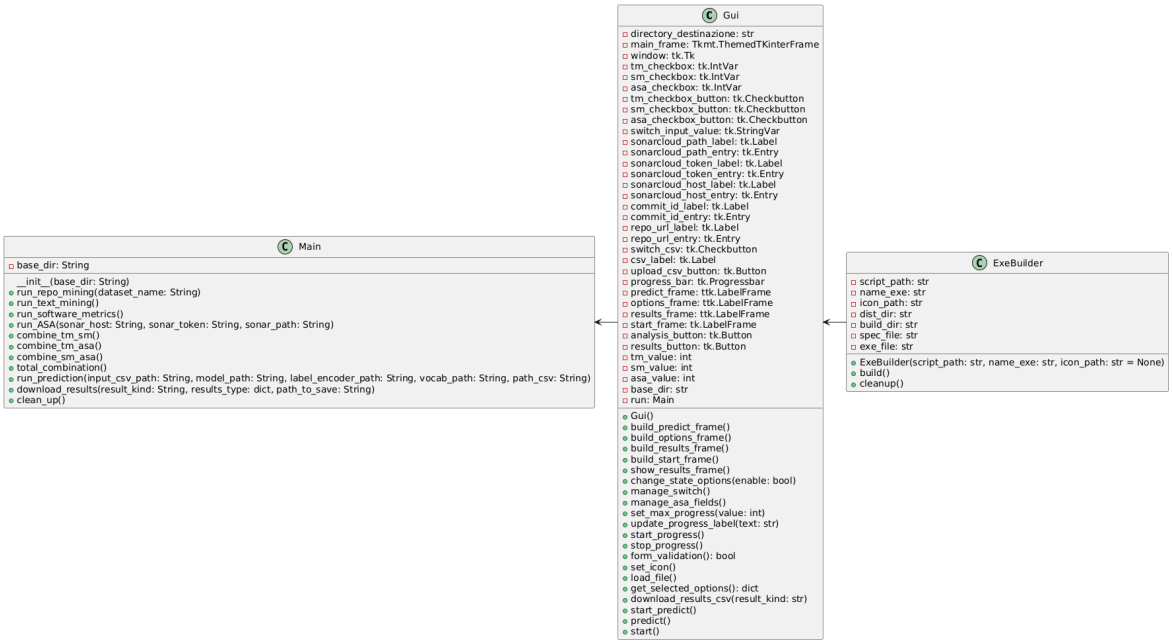

Figura 8.2: Class Diagram delle parti del sistema modificate dopo la CR8

In definitiva, il sistema finale sarà suddiviso nei seguenti package:

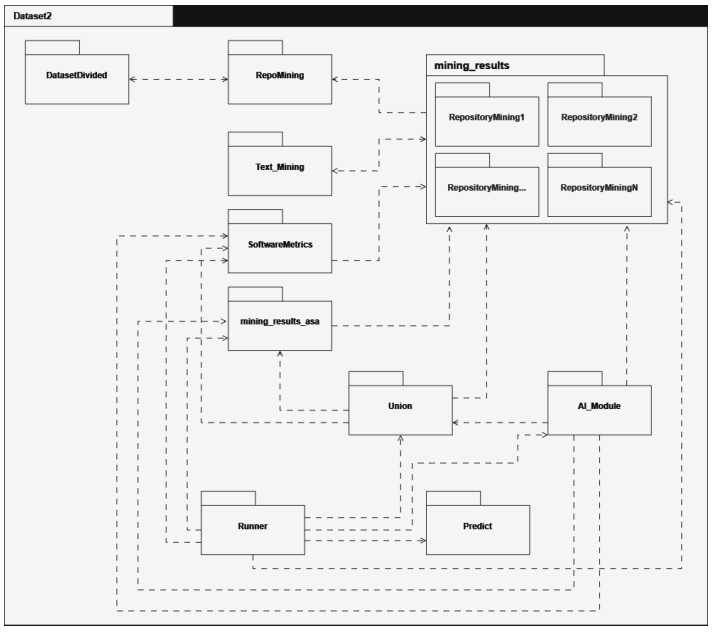

Figura 8.3: Package Diagram del nuovo sistema

Actual Impact Set - 6 Componenti

- Class Main

- Class SonarAnalyzer
- Class CSVWriter
- Class MetricsWriter
- Class DatasetDivider
- Class RepoMiner

False Positive Impact Set - 3 Componenti

- Class JavaTextMining
- Class DictGenerator
- Class SoftwareMetrics

Discovered Impact Set - 0 Componenti

8.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = $|(CIS \cap AIS)| \ / \ |AIS| = 6 \ / \ 6 = 1.0$
- Precision = $|(CIS \cap AIS)| \ / \ |CIS| = 6 \ / \ 9 = 0.67$

CAPITOLO 9

Descrizione modifiche Change request 3 (CR3)

Comportamento Attuale:

Attualmente, il sistema gestisce correttamente la maggior parte degli errori, ma alcune eccezioni durante il mining dei repository non vengono catturate, portando a possibili crash o risultati incompleti. Inoltre, non viene neanche controllato se l'header del csv iniziale dato al sistema è corretto.

Proposta di Modifica:

La proposta di modifica prevede l'introduzione di un sistema di gestione delle eccezioni più robusto, in particolare nei seguenti punti:

1. Inserire blocchi try-except nella classe RepoMiner per catturare eventuali errori durante il mining dei repository, assicurando che il processo non termini in modo anomalo.
2. Inserire controlli nelle classi RepoMiner per verificare la correttezza del formato dei csv in input.
3. Creazione di un file di log per riportare le anomalie registrate durante l'esecuzione del sistema

Benefici previsti:

L'implementazione di questa modifica apporterà i seguenti benefici:

1. Maggiore robustezza del sistema: la gestione più completa delle eccezioni ridurrà il rischio di crash e permetterà al sistema di gestire meglio gli errori imprevisti.

2. Affidabilità dei risultati: con l'introduzione di controlli nelle operazioni di unione delle liste, il sistema garantirà risultati più consistenti e accurati.

3. Manutenibilità migliorata: la gestione centralizzata delle eccezioni semplificherà il debugging e la futura manutenzione del codice.

9.1 Analisi d'impatto

In questa analisi dell'impatto utilizzeremo i nomi delle classi che hanno sostituito gli script originali, in quanto, questa CR3 è postuma alla CR4, vista la sua priorità bassa.

Starting Impact Set - 1 Componente

- RepoMiner.py

	RepoMiner
RepoMiner	-
Main	1
Gui	2
CSVWriter	2
JavaTextMining	1
DictGenerator	2
CsvCreatorForAsa	3
DatasetCombiner	3

Tabella 9.1: Matrice di raggiungibilità con distanza

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set (CIS) è stata generata la matrice di raggiungibilità per ridurre le occorrenze di false positive. Per le ragioni descritte, all'interno del Candidate Impact Set (CIS) abbiamo inserito:

Candidate Impact Set - 8 Componenti

- Class RepoMiner

- Class Main
- Class Gui
- Class CSVWriter
- Class JavaTextMining
- Class DictGenerator
- Class CsvCreatorForAsa
- Class DatasetCombiner

9.2 Implementazione modifiche

L'implementazione ci ha permesso di gestire vari errori principali, migliorando la robustezza del sistema. Ci siamo serviti di blocchi try-except per catturare errori e riportarli nel log del relativo modulo. In particolare, in RepoMining, abbiamo migliorato il sistema di log già presente utilizzando un unico file di log anzichè due, mantenendo la distinzione tra errori e messaggi informativi. Sono stati gestiti (con annessa registrazione nel log), poi, i seguenti errori:

- MissingSchema, nel caso di un url non formattato correttamente;
- ConnectionError, dovuto a problemi di connessione;
- GitCommandError, sollevato durante il clone della repo da analizzare.

Abbiamo aggiunto controlli per verificare il formato dell'header del CSV in input, e nella parte di analisi con SonarQube, abbiamo creato il sistema di log per segnalare all'utente gli errori legati all'avvio di SonarQube stesso, del token, dell'indirizzo host e del percorso di sonarscanner sbagliati.

In sintesi, ogni errore critico viene gestito e tracciato senza interrompere l'esecuzione complessiva del tool e i log sono stati aggiunti al download dei file csv che l'utente può effettuare attraverso la GUI.

Actual Impact Set - 3 Componenti

- Class RepoMiner

- Class Main
- Class Gui

False Positive Impact Set - 5 Componenti

- Class CSVWriter
- Class JavaTextMining
- Class DictGenerator
- Class CsvCreatorForAsa
- Class DatasetCombiner

Discovered Impact Set - 0 Componenti

9.3 Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

- ReCall = |(CIS ∩ AIS)| / |AIS| = 3 / 3 = 1.0
- Precision = |(CIS ∩ AIS)| / |CIS| = 3 / 8 = 0.38