UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Ingegneria, Gestione ed Evoluzione del Software

Modification Testing and Regression Testing

TEAM MEMBER

Donia Daniele - 0522501575

La Marca Antonio - 0522501557

Somma Pasquale - 0522501543

Indice

CAPITOLO 1

Introduzione

In seguito all'implementazione delle Change Request 1 e 2, sono stati eseguiti test di regressione con l'obiettivo di verificare la risoluzione dei problemi precedentemente individuati e di controllare eventuali nuove problematiche.

La Change Request 4 ha comportato una revisione significativa del sistema, trasformando gli script esistenti in classi. Questa modifica ha reso obsoleti i test precedentemente sviluppati, rendendo necessaria una loro revisione per adattarli alla nuova architettura. Di conseguenza, sono state apportate piccole modifiche ai test esistenti e sono stati sviluppati nuovi casi di test per garantire la correttezza delle funzionalità aggiornate.

Le Change Request 5, 6 e 7 hanno introdotto tre nuovi moduli, per i quali sono stati progettati e implementati nuovi casi di test specifici, volti a garantire la copertura funzionale di ciascun modulo.

Con l'introduzione dell'interfaccia grafica tramite la Change Request 8, è stato possibile avviare il testing di sistema, precedentemente posticipato, migliorando l'accessibilità e la personalizzazione dell'esperienza utente. Questa Change Request ha inoltre richiesto lievi adattamenti dei test case esistenti, includendo la generalizzazione della gestione dei path all'interno del sistema.

Infine, per la Change Request 3 è stato eseguito un ulteriore ciclo di test di regressione per verificare la stabilità del sistema e identificare eventuali nuove criticità derivanti dalle modifiche apportate.

CAPITOLO 2

Test introdotti

2.1 Change Request 4

2.1.1 Unit testing

Repo_Mining

In seguito alla modifica, la logica dello script main_repo_mining è gestita dalla funzione **run_repo_mining** nella classe Main. Per il testing di unità di tale funzione, sono stati considerate le seguenti variabili e relative categorie:

- Variabile 1: dataset
  - Categoria: Esistenza (E)
    - Esiste (E1) [property dataExistOk]
    - Non esiste (E2) [error]
  - Categoria: Formato (F)
    - Formato valido (F1) [if dataExistOk]
    - Formato non valido (F2) [if dataExistOk] [error]
  - Categoria: contenuto (C)
    - 0 righe (C1)
    - 1 riga (C2)

    * Più di una riga (C3)

Test Case	Combinazione	Oracolo
TC_1	E2	FileNotFoundError
TC_2	E1 F2	ValueError
TC_3	E1 F1 C1	Nessuna repo inizializzata
TC_4	E1 F1 C2	Una repo inizializzata
TC_5	E1 F1 C3	Numero di repo inizializzate pari alle righe del dataset

Tabella 2.1: Test case per la funzione run_repo_mining

Per il testing della funzione **analyze_commit**, introdotta con la trasformazione ad oggetti, sono stati considerati i seguenti parametri e oggetti dell'ambiente:

- commit, ovvero il parametro contenente le informazioni relative al commit.
  - Categoria: esistenza (CE)
    * Scelta 1: Commit esistente [property commitExistOk]
    * Scelta 2: Commit pari a None [error]
  - Categoria: contenuto modifica (CM)
    * Scelta 1: non esiste una modifica per un commit (CM1) [if commitExistOk] [single]
    * Scelta 2: esiste una modifica per commit e non comprende classi java (CM2) [single] [if commitExistOk]
    * Scelta 3: esiste una modifica per commit che riguarda classi .java introdotte con esso (CM3) [if commitExistOk]
    * Scelta 4: esiste una modifica per commit che riguarda classi .java già presenti prima di esso (CM4) [if commitExistOk]
- cve_path, ovvero il parametro contenente il percorso del file CVE.
  - Categoria: formato (FCV)
    * Scelta 1: Formato valido [property cvePathFormatOk]
    * Scelta 2: Formato non valido [error]

  - Categoria: esistenza (ECV)
    * Scelta 1: Si [if cvePathFormatOk]
    * Scelta 2: No

- commit_path, ovvero il parametro contenente il percorso del commit.
  - Categoria: formato (FCM)
    * Scelta 1: Formato valido [property commitPathFormatOk]
    * Scelta 2: Formato non valido [error]
  - Categoria: esistenza (ECM)
    * Scelta 1: Si [if commitPathFormatOk]
    * Scelta 2: No [if commitPathFormatOk] [error]

- directory cve
  - Categoria: esistenza (EDCV)
    * Scelta 1: Si [if cvePathFormatOk]
    * Scelta 2: No [if cvePathFormatOk]

- directory commit
  - Categoria: esistenza (EDCM)
    * Scelta 1: Si [if commitPathFormatOk]
    * Scelta 2: No [if commitPathFormatOk]

Sono stati aggiunti, quindi, i seguenti test case:

Test Case	Combinazione	Oracolo
TC_1.4_1	CE2	AttributeError
TC_1.4_2	FCV2	OSError
TC_1.4_3	FCV1 ECV2	FileNotFoundError
TC_1.4_4	FCM2	OSError
TC_1.4_5	FCM1 ECM2	FileNotFoundError
TC_1.4_6	CE1 CM1 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM2	Dir cve e commit non create
TC_1.4_7	CE1 CM2 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM2	Dir cve e commit non create
TC_1.4_8	CE1 CM3 FCV1 ECV1 FCM1 ECM1 EDCV1 EDCM1	Dir commit vuota
TC_1.4_9	CE1 CM3 FCV1 ECV1 FCM1 ECM1 EDCV1 EDCM2	Dir commit è creata e vuota
TC_1.4_10	CE1 CM3 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM1	Dir cve creata e commit è vuota
TC_1.4_11	CE1 CM3 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM2	Dir cve e commit create e commit è vuota
TC_1.4_12	CE1 CM4 FCV1 ECV1 FCM1 ECM1 EDCV1 EDCM1	Dir commit contiene java file modificati
TC_1.4_13	CE1 CM4 FCV1 ECV1 FCM1 ECM1 EDCV1 EDCM2	Dir commit creata e commit contiene java file modificati
TC_1.4_14	CE1 CM4 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM1	Dir cve creata e commit contiene file java modificati
TC_1.4_15	CE1 CM4 FCV1 ECV1 FCM1 ECM1 EDCV2 EDCM2	Dir cve e commit create e commit contiene file java modificati

Tabella 2.2: Test case per la funzione initialize_repo_mining

2.2 Change Request 5

L'implementazione della change request 5 ha portato all'introduzione delle classi SoftwareMetrics e MetricsWriter, per le quali non esistono test realizzati nelle fasi precedenti. Quindi sono stati definiti i test case per le due nuove classi, utilizzando lo stesso approccio utilizzato precedentemente.

2.2.1 Unit testing

Software Metrics

Riguardo la funzione **__remove_comments**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]
    - Scelta 2: Il percorso esiste (ED2)
- **Variabile 2**: text
  - Categoria: Contenuto del codice (CN)
    - Scelta 1: La stringa è vuota (CN1)
    - Scelta 2: Solo commenti multilinea (CN2)
    - Scelta 3: Solo commenti single-line (CN3)
    - Scelta 4: Commenti all'interno di costanti stringhe (CN4)
    - Scelta 5: Codice misto con vari tipi di commenti (CN5)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.1_1	ED1	FileNotFoundError
TC_5.1_2	ED2 CN1	Restituisce una stringa vuota
TC_5.1_3	ED2 CN2	Restitusice una stringa vuota senza i commenti multi-linea
TC_5.1_4	ED2 CN3	Restitusice una stringa vuota senza i commenti single-line
TC_5.1_5	ED2 CN4	Restituisce la stringa data in input
TC_5.1_6	ED2 CN6	Restituisce la stringa senza alcun tipo di commento

Tabella 2.3: Test case per la funzione __remove_comments

Per quando riguarda la funzione **calculate_max_nesting**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]
    - Scelta 2: Il percorso esiste (ED2)
- **Variabile 2**: file_content
  - Categoria: Formato codice (FC)
    - Scelta 1: Stringa vuota (FC1)
    - Scelta 2: Stringa con codice sintatticamente sbagliato (FC2) [single]
    - Scelta 3: Stringa con codice semanticamente sbagliato (FC3) [single]
    - Scelta 4: Stringa con codice corretto (FC4)
  - Categoria: Contenuto file (CN)
    - Scelta 1: Non vi è alcun commento (CN1)
    - Scelta 2: Il codice presenta commenti con {} (CN2) [single]
  - Categoria: Strutture di controllo (SC)
    - Scelta 1: Nessuna struttura di controllo (SC1)
    - Scelta 2: If come struttura di controllo presente (SC2) [property ifControl]
    - Scelta 3: For come struttura di controllo presente (SC3)
    - Scelta 4: While come strttura di controllo presente (SC4)
    - Scelta 5: Do come strttura di controllo presente (SC5)

    - Scelta 6: Switch come struttura di controllo presente (SC6)
    - Scelta 7: Try come struttura di controllo presente (SC7)
    - Scelta 8: Tutte le strutture di controllo presenti [property allControl] (SC8)
  - Categoria: Caratteristiche else (CE)
    - Scelta 1: Else non presente [if ifControl] (CE1)
    - Scelta 2: Else presente con { [if ifControl property elseOk] (CE2)
    - Scelta 3: Else presente senza { [if ifControl property elseOk] (CE3)
  - Categoria: Annidamento (AN)
    - Scelta 1: Struttura di controllo sequeziali
    - Scelta 2: Strutture di controllo annidata [if allControl or elseOk]
    - Scelta 3: Strttura di controllo sequenziali e annidati [if allControl]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	**Combinazione**	**Oracolo**
TC_5.2_1	ED1	FileNotFoundError
TC_5.2_2	ED2 FC1	Restituisce 0
TC_5.2_3	ED2 FC2 CN1 SC2	Restituisce il livello di annidamento massimo dei controlli che rileva
TC_5.2_4	ED2 FC3 CN1 SC1	Restituisce il livello di annidamento massimo dei controlli che rileva
TC_5.2_5	ED2 FC4 CN1 SC1	Restitusice 0
TC_5.2_6	ED2 FC4 CN1 SC2 CE1 AN1	Restituisce il livello di annidamento massimo dell'if che rileva
TC_5.2_7	ED2 FC4 CN1 SC2 CE2 AN1	Restituisce il livello di annidamento massimo dell'if o else che rileva
TC_5.2_8	ED2 FC4 CN1 SC2 CE2 AN2	Restituisce il livello di annidamento massimo dell'if o else che rileva
TC_5.2_9	ED2 FC4 CN1 SC2 CE3 AN1	Restituisce il livello di annidamento massimo individuato, non considerando l'else come livello

Test Case	Combinazione	Oracolo
TC_5.2_10	ED2 FC4 CN1 SC2 CE3 AN2	Restituisce il livello di annidamento massimo individuato, non considerando l'else come livello
TC_5.2_11	ED2 FC4 CN1 SC3 AN1	Restituisce il livello di annidamento massimo del for che rileva
TC_5.2_12	ED2 FC4 CN1 SC4 AN1	Restituisce il livello di annidamento massimo del while che rileva
TC_5.2_13	ED2 FC4 CN1 SC5 AN1	Restituisce il livello di annidamento massimo del do che rileva
TC_5.2_14	ED2 FC4 CN1 SC6 AN1	Restituisce il livello di annidamento massimo dello switch che rileva
TC_5.2_15	ED2 FC4 CN1 SC7 AN1	Restituisce il livello di annidamento massimo del try che rileva
TC_5.2_16	ED2 FC4 CN1 SC8 AN1	Restituisce il livello di annidamento massimo dei controlli che rileva
TC_5.2_17	ED2 FC4 CN1 SC8 AN2	Restituisce il livello di annidamento massimo dei controlli che rileva
TC_5.2_18	ED2 FC4 CN1 SC8 AN3	Restituisce il livello di annidamento massimo dei controlli che rileva
TC_5.2_19	ED2 FC4 CN2 SC2 CE2 AN1	Restituisce il livello di annidamento massimo dei controlli che rileva

Tabella 2.4: Test case per la funzione calculate_max_nesting

Per quanto riguarda la funzione **count_declarative_lines**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]
    - Scelta 2: Il percorso esiste (ED2)
- **Variabile 2**: file_content
  - Categoria: Formato codice (FC)

    - Scelta 1: Stringa vuota (FC1)
    - Scelta 2: Stringa con codice sintatticamente sbagliato (FC2) [single]
    - Scelta 3: Stringa con codice semanticamente sbagliato (FC3) [single]
    - Scelta 4: Stringa con codice sintatticamente corretto (FC4)
  - Categoria: Tipi di dichiarazioni (TD)
    - Scelta 1: Nessuna dichiarazione (TD1)
    - Scelta 2: Dichiarazioni di classi (TD2)
    - Scelta 3: Dichiarazioni di interfacce (TD3)
    - Scelta 4: Dichiarazioni di metodi (TD4)
    - Scelta 5: Dichiarazioni di campi o variabili (TD5)
    - Scelta 6: Dichiarazioni di annotazioni (TD6)
    - Scelta 7: Dichiarazioni con parole chiave (TD7)
    - Scelta 8: Dichiarazioni di import (TD8)
    - Scelta 9: Tutte i tipi di dichiarazioni presenti (TD9)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	**Combinazione**	**Oracolo**
TC_5.3_1	ED1	FileNotFoundError
TC_5.3_2	ED2 FC1	Restitusice 0
TC_5.3_3	ED2 FC2 TD4	Restituisce il numero di linee di dichiarazioni rilevate correttamente
TC_5.3_4	ED2 FC3 TD4	Restituisce il numero di linee di dichiarazioni rilevate correttamente
TC_5.3_5	ED2 FC4 TD1	Restituisce 0
TC_5.3_6	ED2 FC4 TD2	Restituisce il numero di linee di dichiarazioni di classi
TC_5.3_6	ED2 FC4 TD3	Restituisce il numero di linee di dichiarazioni di interfacce
TC_5.3_8	ED2 FC4 TD4	Restituisce il numero di linee di dichiarazioni di classi e metodi
TC_5.3_9	ED2 FC4 TD5	Restituisce il numero di linee di dichiarazioni di variabili
TC_5.3_10	ED2 FC4 TD6	Restituisce il numero di linee di

Test Case	Combinazione	Oracolo
		dichiarazioni di annotazioni
TC_5.3_11	ED2 FC4 TD7	Restituisce il numero di linee di dichiarazioni di parole chiave
TC_5.3_12	ED2 FC4 TD8	Restituisce il numero di linee di dichiarazioni di import
TC_5.3_13	ED2 FC4 TD9	Restituisce il numero di linee di dichiarazione

Tabella 2.5: Test case per la funzione count_declarative_lines

Per quanto riguarda la funzione **count_lines_of_code**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]
    - Scelta 2: Il percorso esiste (ED2)
- **Variabile 2**: file_content
  - Categoria: Grandezza file (GF)
    - Scelta 1: Stringa vuota (GF1) [property emptyString]
    - Scelta 2: Stringa con una riga (GF2) [single] [property oneLine]
    - Scelta 3: Stringa con più righe (GF3) [property multiLine]
  - Categoria: Formato codice (FS)
    - Scelta 1: Stringa con codice sintatticamente sbagliato (FS1) [single]
    - Scelta 2: Stringa con codice semanticamente sbagliato (FS2) [single]
    - Scelta 3: Stringa con codice sintatticamente corretto (FS3)
  - Categoria: Presenza commenti (CF)
    - Scelta 1: Nessun commento (CF1) [property noComment]
    - Scelta 2: Commenti su righe singole (CF2) [property singleComment]
    - Scelta 3: Blocchi di commento multilinea (CF3) [if not singleLine]
    - Scelta 4: Righe vuote (CF4) [if multiLine property emptyLines]
  - Categoria: Presenza codice (PC)

    * Scelta 1: Contiene codice (PC1) [if (oneLine and not singleComment and not emptyLines) or multiLine]
    * Scelta 2: Nessuna linea di codice (PC2) [if (not noComment and not emptyLines)]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.4_1	ED1	FileNotFoundError
TC_5.4_2	ED2 GF1	Restituisce 0
TC_5.4_3	ED2 GF2 FS1 CF1 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_4	ED2 GF2 FS2 CF1 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_5	ED2 GF2 FS3 CF1 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_6	ED2 GF2 FS3 CF2 PC2	Restituisce 0
TC_5.4_7	ED2 GF3 FS3 CF1 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_8	ED2 GF3 FS3 CF2 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_9	ED2 GF3 FS3 CF2 PC2	Restituisce 0
TC_5.4_10	ED2 GF3 FS3 CF3 PC1	Restituisce il numero di righe con codice della stringa data in input
TC_5.4_11	ED2 GF3 FS3 CF3 PC2	Restituisce 0
TC_5.4_12	ED2 GF3 FS3 CF4 PC1	Restituisce il numero di righe con codice della stringa data in input

Tabella 2.6: Test case per la funzione count_lines_of_code

Per quanto riguarda la funzione **count_method_declarations**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)

        * Scelta 1: Il percorso non esiste (ED1) [error]
        * Scelta 2: Il percorso esiste (ED2)

- **Variabile 2**: file_content
    - Categoria: Formato codice (FC)
        * Scelta 1: Stringa vuota (FC1)
        * Scelta 2: Stringa con codice sintatticamente sbagliato (FC2) [single]
        * Scelta 3: Stringa con codice semanticamente sbagliato (FC3) [single]
        * Scelta 4: Stringa con codice corretto (FC4)
    - Categoria: Tipo di dichiarazioni (TD)
        * Scelta 1: Nessuna dichiarazione (TD1)
        * Scelta 2: Dichiarazioni di metodi semplici (TD2)
        * Scelta 3: Dichiarazioni di costruttori (TD3)
        * Scelta 4: Dichiarazioni di espressioni lambda (TD4)
        * Scelta 5: Dichiarazioni di metodi in classi anonime (TD5)
        * Scelta 6: Combinazione di metodi, costruttori e lambda (TD6)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.5_1	ED1	FileNotFoundError
TC_5.5_2	ED2 FC1	Restituisce 0
TC_5.5_3	ED2 FC2 TD2	Restitusice il numero di metodi individuati correttamente
TC_5.5_4	ED2 FC3 TD2	Restitusice il numero di metodi individuati correttamente
TC_5.5_5	ED2 FC4 TD1	Restituisce 0
TC_5.5_6	ED2 FC4 TD2	Restitusice il numero di metodi individuati correttamente
TC_5.5_7	ED2 FC4 TD3	Restitusice il numero di metodi individuati correttamente
TC_5.5_8	ED2 FC4 TD4	Restitusice il numero di metodi individuati correttamente
TC_5.5_9	ED2 FC4 TD5	Restitusice il numero di metodi individuati correttamente
TC_5.5_10	ED2 FC4 TD6	Restitusice il numero di metodi individuati correttamente

Tabella 2.7: Test case per la funzione count_method_declarations

Per quanto riguarda la funzione **count_class_declarations**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]
    - Scelta 2: Il percorso esiste (ED2)
- **Variabile 2**: file_content
  - Categoria: Formato codice (FC)
    - Scelta 1: Stringa vuota (FC1)
    - Scelta 2: Stringa con codice sintatticamente sbagliato (FC2) [single]
    - Scelta 3: Stringa con codice semanticamente sbagliato (FC3) [single]
    - Scelta 4: Stringa con codice corretto (FC4)
  - Categoria: Tipi di dichiarazioni (TD)
    - Scelta 1: Nessuna dichiarazione di classe (TD1)
    - Scelta 2: Presenza di dichiarazioni di classe (TD2)
    - Scelta 3: Presenza di dichiarazioni di interfaccia (TD3)
    - Scelta 4: Dichiarazioni di classi anonime (TD4)
    - Scelta 5: Dichiarazioni nidificate di classe (TD5)
    - Scelta 6: Dichiarazioni di tutti i tipi precedenti (TD6)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.6_1	ED1	FileNotFoundError
TC_5.6_2	ED2 FC1	Restituisce 0
TC_5.6_3	ED2 FC2 TD2	Restituisce il numero di classi individuate correttamente
TC_5.6_4	ED2 FC3 TD1	Restituisce il numero di classi individuate correttamente
TC_5.6_5	ED2 FS4 TD1	Restituisce 0
TC_5.6_6	ED2 FS4 TD2	Restitusice il numero di classi dichiarate
TC_5.6_7	ED2 FS4 TD3	Restitusice il numero di interfacce dichiarate
TC_5.6_8	ED2 FS4 TD4	Restitusice il numero di classi anonime dichiarate
TC_5.6_9	ED2 FS4 TD5	Restitusice il numero di classi dichiarate
TC_5.6_10	ED2 FS4 TD6	Restitusice il numero di classi di ogni tipo dichiarate

Tabella 2.8: Test case per la funzione count_class_declarations

Per quanto riguarda la funzione **compute_essential_complexity_metrics**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: file_content
  - **Categoria: lessico (L)**
    - Scelta 1: lessico ammissibile [property lexiconValidOk]
    - Scelta 2: lessico inammissibile [error]
  - **Categoria: sintassi (S)**
    - Scelta 1: sintassi corretta [if lexiconValidOk] [syntaxValidOk]
    - Scelta 2: sintassi errata [if lexiconValidOk] [error]
  - **Categoria: numero elementi classe (N)**
    - Scelta 1: zero [if syntaxValidOk]
    - Scelta 2: uno [if syntaxValidOk] [property ClassNotEmpty]
    - Scelta 3: >uno [if syntaxValidOk] [property ClassNotEmpty] [single]
  - **Categoria: complessità classe (CC)**
    - Scelta 1: classe con solo metodi [if syntaxValidOk and ClassNotEmpty]
    - Scelta 2: classe con solo costruttori [if syntaxValidOk and ClassNotEmpty] [single]
    - Scelta 3: classe con metodi e costruttori [if syntaxValidOk and ClassNotEmpty] [single]
  - **Categoria: presenza control flow statement (PC)**
    - Scelta 1: presenza if [if ClassNotEmpty] [property ifPresent]
    - Scelta 2: presenza while [if ClassNotEmpty] [property whilePresent]
    - Scelta 3: presenza for [if ClassNotEmpty] [property forPresent]
    - Scelta 4: presenza do [if ClassNotEmpty] [property doPresent]
    - Scelta 5: presenza switch [if ClassNotEmpty] [property switchPresent]
    - Scelta 6: presenza try-catch [if ClassNotEmpty] [property tryCatchPresent]
    - Scelta 7: tutti cfs presenti [if ClassNotEmpty] [property allPresent] [single]
    - Scelta 8: nessun cfs presente [if ClassNotEmpty] [property notPresent] [single]
  - **Categoria: presenza block statement (PB)**
    - Scelta 1: sì [if ClassNotEmpty]

    * Scelta 2: no [if ClassNotEmpty] [single]

- **Categoria: presenza synchronized statement (PS)**
    * Scelta 1: sì [if ClassNotEmpty] [single]
    * Scelta 2: no [if ClassNotEmpty]
- **Categoria: presenza then statement non riducibile (PTH)**
    * Scelta 1: sì [if ifPresent or allPresent] [property thenNotRedPresent]
    * Scelta 2: no [if ifPresent or allPresent] [single]
- **Categoria: presenza else statement non riducibile (PE)**
    * Scelta 1: sì [if ifPresent or allPresent] [property elseNotRedPresent]
    * Scelta 2: no [if ifPresent or allPresent] [single]
- **Categoria: numero if presenti (NI)**
    * Scelta 1: uno [if ifPresent and (thenNotRedPresent or elseNotRedPresent)]
    * Scelta 2: più di uno [if ifPresent and (thenNotRedPresent or elseNotRedPresent)]
- **Categoria: presenza corpo loop non riducibile (PL)**
    * Scelta 1: sì [if whilePresent or forPresent or doPresent or allPresent] [single]
    * Scelta 2: no [if whilePresent or forPresent or doPresent or allPresent]
- **Categoria: presenza corpo try non riducibile (PT)**
    * Scelta 1: sì [if tryPresent or allPresent]
    * Scelta 2: no [if tryPresent or allPresent] [single]
- **Categoria: presenza finally (PF)**
    * Scelta 1: finally assente [if tryPresent or allPresent]
    * Scelta 2: finally presente e riducibile [if tryPresent or allPresent] [single]
    * Scelta 3: finally presente e non riducibile [if tryPresent or allPresent] [single]
- **Categoria: numero catch (NCH)**
    * Scelta 1: zero [if tryPresent or allPresent] [single]
    * Scelta 2: uno [if tryPresent or allPresent] [property catchPresent]
    * Scelta 3: più di uno [if tryPresent or allPresent] [property catchPresent]
- **Categoria: presenza statement non riducibili in catch (PCH)**
    * Scelta 1: nessuno statement non riducibile [if catchPresent] [single]

    - Scelta 2: statement non riducibile in al più un catch [if catchPresent] [property statNotRedPresent] [single]
    - Scelta 3: statement non riducibile in ogni catch [if catchPresent] [property statNotRedPresent]
  - **Categoria: tipologia statement non riducibili in catch (TS)**
    - Scelta 1: statement di tipo control flow [if statNotRedPresent] [single]
    - Scelta 2: statement non control flow [if statNotRedPresent] [property normStatNotRed]
  - **Categoria: numero switch (NS)**
    - Scelta 1: uno [if switchPresent or allPresent]
    - Scelta 2: più di uno [if switchPresent or allPresent] [single]
  - **Categoria: numero case (NC)**
    - Scelta 1: uno [if switchPresent or allPresent] [single]
    - Scelta 2: più di uno [if switchPresent or allPresent]
  - **Categoria: presenza statement non riducibile diverso da break in almeno un case (PCS)**
    - Scelta 1: sì
    - Scelta 2: no
  - **Categoria: presenza break (PBK)**
    - Scelta 1: nessun break in nessun case [single]
    - Scelta 2: break in solo un case
    - Scelta 3: break in ogni case
  - **Categoria: organizzazione control flow statement (O)**
    - Scelta 1: sequenziali [if not notPresent]
    - Scelta 2: annidati [if not notPresent] [single]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.7_1	L2	LexerError
TC_5.7_2	L1 S2	SyntaxError

Test Case	Combinazione	Oracolo
TC_5.7_3	L1 S1 N1	Max Complexity = 0 Sum Complexity = 0
TC_5.7_4	L1 S1 N3 CC3 PC7 PB1 PS1 PTH1 PE1 PL1 PT1 PF2 NCH3 PCH2 TS1 NS1 NC2 PCS1 PBK2 O2	M.E.C. = num. if + num. while + num. for + num. do + num. catch + num. case di switch + 1 , S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_5	L1 S1 N2 CC1 PC8 PB1 PS2	M.E.C. = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_6	L1 S1 N2 CC2 PC1 PB2 PS2 PTH2 PE2 O1	M.E.C = 1 , S.E.C. = S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_7	L1 S1 N2 CC1 PC1 PB1 PS2 PTH1 PE1 NI1 O1	M.E.C = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_8	L1 S1 N2 CC1 PC1 PB1 PS2 PTH1 PE1 NI2 O1	M.E.C = num. di if + 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_9	L1 S1 N2 CC1 PC2 PB1 PS2 PL2 O1	M.E.C = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_10	L1 S1 N2 CC1 PC3 PB1 PS2 PL2 O1	M.E.C. = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_11	L1 S1 N2 CC1 PC4 PB1 PS2 PL2 O1	M.E.C = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_12	L1 S1 N2 CC1 PC5 PB1 PS2 NS2 NC1 PCS2 PBK1 O1	M.E.C. = num. switch + 1, S.E.C = somma delle C.E. dei metodi presenti

Test Case	Combinazione	Oracolo
TC_5.7_13	L1 S1 N2 CC1 PC5 PB1 PS2 NS1 NC2 PCS1 PBK2 O1	M.E.C. = num. case dello switch + 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.8_14	L1 S1 N2 CC1 PC5 PB1 PS2 NS1 NC2 PCS1 PBK3 O1	M.E.C = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_15	L1 S1 N2 CC1 PC5 PB1 PS2 NS1 NC2 PCS2 PBK2 O1	M.E.C. = num. case dello switch + 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_16	L1 S1 N2 CC1 PC5 PB1 PS2 NS1 NC2 PCS2 PBK3 O1	M.E.C. = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_17	L1 S1 N2 CC1 PC6 PB1 PS2 PT2 PF3 NCH1 O1	M.E.C. = C.E. del corpo del finally, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_18	L1 S1 N2 CC1 PC6 PB1 PS2 PT1 PF1 NCH3 PCH1 O1	M.E.C. = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_19	L1 S1 N2 CC1 PC6 PB1 PS2 PT1 PF1 NCH2 PCH3 TS2	M.E.C. = 1, S.E.C = somma delle C.E. dei metodi presenti
TC_5.7_20	L1 S1 N2 CC1 PC6 PB1 PS2 PT1 PF1 NCH3 PCH3 TS2	M.E.C = num. catch + 1, S.E.C = somma delle C.E. dei metodi presenti

Tabella 2.9: Test case per la funzione calculate_max_nesting

Per quanto riguarda la funzione **analyze**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: base_dir
  - Categoria: Esistenza directory (ED)
    - Scelta 1: Il percorso non esiste (ED1) [error]

        * Scelta 2: Il percorso esiste (ED2)

- **Variabile 2**: file_path
  - Categoria: Esistenza percorso (EP)
    * Scelta 1: Stringa vuota (EP1) [single]
    * Scelta 2: Stringa non vuota indicante un file .java (EP2)
    * Scelta 3: Stringa non vuota indicante un file non .java (EP3)

- **Variabile 3**: file_content
  - Categoria: Formato codice (FC)
    * Scelta 1: Stringa vuota (FC1)
    * Scelta 2: Stringa con codice sintatticamente sbagliato (FC2) [error]
    * Scelta 3: Stringa con codice semanticamente sbagliato (FC3) [error]
    * Scelta 4: Stringa con codice corretto (FC4)
  - Categoria: Numero di funzioni (NF)
    * Scelta 1: Nessuna funzione (NF1) [property noFunction]
    * Scelta 2: Una sola funzione (NF2)
    * Scelta 3: Più funzioni (NF3)
  - Categoria: Complessità ciclomatica (CC)
    * Scelta 1: Nessuna struttura di controllo (CC1) [If not noFunction]
    * Scelta 2: Un livello di controllo nella funzione (CC2) [if not noFunction]
    * Scelta 3: Più livelli di controllo nella funzione (CC3) [if not noFunction]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.8_1	ED1	FileNotFoundError
TC_5.8_2	ED2 EP2 FC1	Restituisce 0 come massimo e somma della cyclomatic complexity
TC_5.8_3	ED2 EP2 FC2	Il log presenta un messaggio di errore
TC_5.8_4	ED2 EP2 FC3	Il log presenta un messaggio di errore
TC_5.8_5	ED2 EP2 FC4 NF1	Restituisce 0 come massimo e somma della cyclomatic complexity
TC_5.8_6	ED2 EP2 FC4 NF2 CC1	Restituisce 1 come massimo e somma della cyclomatic complexity
TC_5.8_7	ED2 EP2 FC4 NF2 CC2	Restituisce 2 come massimo e somma della cyclomatic complexity
TC_5.8_8	ED2 EP2 FC4 NF2 CC3	Restituisce il numero di livelli di controllo
TC_5.8_9	ED2 EP2 FC4 NF3 CC1	Restituisce 1 come massimo e come somma il numero di funzioni
TC_5.8_10	ED2 EP2 FC4 NF3 CC2	Restituisce la massimo cyclomati complexity tra le funzioni e la loro somma
TC_5.8_11	ED2 EP2 FC4 NF3 CC3	Restituisce la massimo cyclomati complexity tra le funzioni e la loro somma
TC_5.8_12	ED2 EP1 FC4 NF2 CC3	Restituisce la massimo cyclomati complexity tra le funzioni e la loro somma
TC_5.8_13	ED2 EP1 FC4 NF2 CC3	Restituisce la massimo cyclomati complexity tra le funzioni e la loro somma

Tabella 2.10: Test case per la funzione analyze

Passiamo adesso a testare la classe MetricsWriter, composta da due funzioni, e che si occupa si salvare le metriche ottenute in un file csv. Vediamo, innanzitutto, le categorie e le possibili scelte individuate per la funzione write_header:

- **Variabile 1**: csv_file
  - Categoria: Contenuto variabile(CV)
    - Scelta 1: None (CV1) [error]
    - Scelta 2: Stringa vuota (CV2) [error]

    - Scelta 3: Il nome di un file csv (CV3) [property fileOk]
    - Scelta 4: Il nome di un file non csv (CV4) [single] [property fileOk]
  - Categoria: Permesso di scrittura (EP)
    - Scelta 1: Sì (EP1) [if fileOk]
    - Scelta 2: No (EP2) [if fileOk]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.9_1	CV1	TypeError
TC_5.9_2	CV2	FileNotFoundError
TC_5.9_3	CV3 EP2	PermissionError
TC_5.9_4	CV3 EP1	Il file viene creato correttamente con gli header impostati
TC_5.9_5	CV4 EP1	Il file viene creato correttamente con gli header impostati

Tabella 2.11: Test case per la funzione write_header

Per la funzione cardine della classe, write_metrics, abbiamo individuato le seguenti categorie e possibili scelte:

- **Variabile 1**: csv_file
  - Categoria: Contenuto variabile(CV)
    - Scelta 1: None (CV1) [error]
    - Scelta 2: Stringa vuota (CV2) [error]
    - Scelta 3: Il nome di un file csv (CV3) [property fileOk]
  - Categoria: Permesso di scrittura (EP)
    - Scelta 1: Sì (EP1) [if fileOk property saveOk]
    - Scelta 2: No (EP2) [if fileOk]
- **Variabile 2**: kind
  - Categoria: Contenuto variabile (CK)
    - Scelta 1: None (CK1)
    - Scelta 2: Non una stringa (CK2)
    - Scelta 4: Una stringa non vuota (CK3)

- **Variabile 3**: name
  - Categoria: Contenuto variabile (CN)
    - Scelta 1: None (CN1)
    - Scelta 2: Non una stringa (CN2)
    - Scelta 3: Una stringa vuota (CN3)
    - Scelta 4: Una stringa non vuota (CN4)
- **Variabile 4**: metrics
  - Categoria: Contenuto variabile (CM)
    - Scelta 1: None (CM1) [error]
    - Scelta 2: Non un dizionario (CM2) [error]
    - Scelta 3: Un dizionario (CM3)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_5.10_1	CV1 CK3 CN4 CM3	TypeError
TC_5.10_2	CV2 CK3 CN4 CM3	FileNotFoundError
TC_5.10_3	CV3 EP2 CK3 CN4 CM3	PermissionError
TC_5.10_4	CV3 EP1 CK1 CN1 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_5	CV3 EP1 CK1 CN2 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_6	CV3 EP1 CK1 CN3 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_7	CV3 EP1 CK1 CN4 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_8	CV3 EP1 CK2 CN1 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_9	CV3 EP1 CK2 CN2 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_10	CV3 EP1 CK2 CN3 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_11	CV3 EP1 CK2 CN4 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_12	CV3 EP1 CK3 CN1 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_13	CV3 EP1 CK3 CN2 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_14	CV3 EP1 CK3 CN3 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_15	CV3 EP1 CK3 CN4 CM3	Il file viene creato con Kind, Name e le chiavi del dizionario come colonne, e i rispettivi valori
TC_5.10_16	CV3 EP1 CK3 CN4 CM1	TypeError
TC_5.10_17	CV3 EP1 CK3 CN4 CM2	TypeError

Tabella 2.12: Test case per la funzione write_metrics

Ai test sono stati aggiunti anche quelli della funzione run_software_metrics della classe

Main per avviare il calcolo delle metriche sui file interessati. Abbiamo definito per questa funzione le seguenti categorie e scelte per testarne il funzionamento:

- **Oggetto dell'ambiente 1**: dataset_divided_path
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1) [error]
    - Scelta 2: Directory esistente (DF2)
- **Variabile 1**: repo_path
  - Categoria: Esistenza directory (ES)
    - Scelta 1: Directory non presente (ES1)
    - Scelta 2: Directory presente (ES2) [property repoOk]
  - Categoria: Contenuto directory (CN)
    - Scelta 1: Directory vuota (CN1) [if repoOk]
    - Scelta 2: Directory non vuota (CN2) [if repoOk property repoNotEmpty]
- **Variabile 2**: cvd_id (file ID in repository)
  - Categoria: Stato variabile (STC)
    - Scelta 1: La variabile è un file (STC1) [if repoNotEmpty]
    - Scelta 2: La variabile è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [if repoNotEmpty]
    - Scelta 3: La variabile è una cartella (STC3) [if repoNotEmpty property cvdOk]
  - Categoria: Contenuto directory (CNC)
    - Scelta 1: La directory è vuota (CNC1) [if cvdOk]
    - Scelta 2: La directory non è vuota (CNC2) [if cvdOk property cvdNotEmpty]
- **Variabile 3**: folder (sotto-cartelle in cvd_id)
  - Categoria: Stato variabile (STF)
    - Scelta 1: La variabile è un file (STF1) [if cvdNotEmpty]
    - Scelta 2: La variabile è un file non ammesso (STF2) [if cvdNotEmpty]
    - Scelta 3: La variabile è una cartella (STF3) [if cvdNotEmpty property folderOk]
  - Categoria: Contenuto directory (CNF)

        * Scelta 1: La directory è vuota (CNF1) [if folderOk]
        * Scelta 2: La directory non è vuota (CNF2) [if folderOk property folderNotEmpty]

- **Variabile 4**: file (file nelle sotto-cartelle)
  - Categoria: Contenuto directory (CNFI)
    * Scelta 1: La variabile è file non ammesso (CNFI1) [if folderNotEmpty]
    * Scelta 2: La variabile è una directory (CNFI2) [if folderNotEmpty]
    * Scelta 3: La variabile è un file Java (CNFI3) [if folderNotEmpty property fileOk]
    * Scelta 4: La variabile è un file non Java (CNFI4) [if folderNotEmpty]
    * Categoria: Permessi sui file (PFJ)
      · Scelta 1: Accessibile (PFJ1) [if fileOk]
      · Scelta 2: Non accessibile (PFJ2) [error]

- **Variabile 5**: metrics
  - Categoria: Contenuto metriche (CNM)
    * Scelta 1: Un dizionario contente tutte le metriche previste (CNM1) [if fileOk]
    * Scelta 2: Un dizionario vuoto (CNM2) [if fileOk]
    * Scelta 3: Un dizionario con una metrica mancante (CNM3) [if fileOk]
    * Scelta 4: Un dizionario con una metrica in più (CNM4) [if fileOk]
    * Scelta 5: Non è un dizionario (CNM5) [if fileOk]

- **Oggetto dell'ambiente 2**: csv_metrics
  - Categoria: Permessi sui file (PF)
    * Scelta 1: Accessibile e ci si può scrivere dentro (PF1)
    * Scelta 2: Non accessibile (PF2) [error]

Test Case	Combinazione	Oracolo
TC_5.11_1	DF1	FileNotFoundError
TC_5.11_2	DF2 ES1	Scrive solo l'header nel csv finale
TC_5.11_3	DF2 ES2 CN1	Scrive solo l'header nel csv finale

Test Case	Combinazione	Oracolo
TC_5.11_4	DF2 ES2 CN2 STC1	Scrive solo l'header nel csv finale
TC_5.11_5	DF2 ES2 CN2 STC2	Scrive solo l'header nel csv finale
TC_5.11_6	DF2 ES2 CN2 STC3 CNC1	Scrive solo l'header nel csv finale
TC_5.11_7	DF2 ES2 CN2 STC3 CNC2 STF1	Scrive solo l'header nel csv finale
TC_5.11_8	DF2 ES2 CN2 STC3 CNC2 STF2	Scrive solo l'header nel csv finale
TC_5.11_9	DF2 ES2 CN2 STC3 CNC2 STF3 CNF1	Scrive solo l'header nel csv finale
TC_5.11_10	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM1 PF1	Scrive l'header nel csv finale e il valore delle metriche per ogni file java
TC_5.11_11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM2 PF1	Scrive l'header nel csv finale e lascia uno spazio vuoto per ogni metrica
TC_5.11_12	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM3 PF1	Scrive solo l'header nel csv finale. scrive il valore delle metriche presenti e uno spazio vuoto per quella mancante
TC_5.11_13	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM4 PF1	ValueError
TC_5.11_14	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM5 PF1	TypeError
TC_5.11_15	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ1 CNM1 PF2	PermissionError
TC_5.11_16	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 PFJ2	PermissionError
TC_5.11_17	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1	Scrive solo l'header nel csv finale
TC_5.11_18	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI2	Scrive solo l'header nel csv finale
TC_5.11_19	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI4	Scrive solo l'header nel csv finale

Tabella 2.13: Test case per la funzione run_software_metrics

2.2.2 Integration testing

Software Metrics

Il metodo che si occupa di avviare l'analisi delle metriche software dei vari file java è run_software_metrics della classe Main. Per tale metodo abbiamo individuato le seguenti categorie e possibili scelte su cui basare i nostri test case:

- **Variabile 1**: csv_file
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1)
    - Scelta 2: Directory esistente (DF2) [property miningOk]
- **Variabile 2**: repo
  - Categoria: Esistenza directory (ES)
    - Scelta 1: Directory non presente (ES1) [if miningOk]
    - Scelta 2: Directory presente (ES2) [if miningOk property repoOk]
  - Categoria: Contenuto directory (CN)
    - Scelta 1: Directory vuota (CN1) [if repoOk]
    - Scelta 2: Directory non vuota (CN2) [if repoOk property repoNotEmpty]
- **Variabile 3**: cvd_id
  - Categoria: Stato variabile (STC)
    - Scelta 1: La variabile è un file (STC1) [if repoNotEmpty]
    - Scelta 2: La variabile è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [if repoNotEmpty]
    - Scelta 3: La variabile è una cartella (STC3) [if repoNotEmpty property cvdOk]
  - Categoria: Contenuto directory (CNC)
    - Scelta 1: La directory è vuota (CNC1) [if cvdOk]
    - Scelta 2: La directory non è vuota (CNC2) [if cvdOk property cvdNotEmpty]
- **Variabile 4**: folder
  - Categoria: Stato variabile (STF)
    - Scelta 1: La variabile è un file (STF1) [if cvdNotEmpty]

        * Scelta 2: La variabile è un file non ammesso (STF2) [if cvdNotEmpty]
        * Scelta 3: La variabile è una cartella (STF3) [if cvdNotEmpty property folderOk]
    - Categoria: Contenuto directory (CNF)
        * Scelta 1: La directory è vuota (CNF1) [if folderOk]
        * Scelta 2: La directory non è vuota (CNF2) [if folderOk property folderNotEmpty]

- **Variabile 5**: file
    - Categoria: Contenuto directory (CNFI)
        * Scelta 1: La variabile è file non ammesso (CNFI1) [if folderNotEmpty]
        * Scelta 2: La variabile è una directory (CNFI2) [if folderNotEmpty]
        * Scelta 3: La variabile è un file Java (CNFI3) [if folderNotEmpty property fileOk]
        * Scelta 4: La variabile è un file non Java (CNFI4) [if folderNotEmpty]
    - Categoria: Contenuto file java (CFJ)
        * Scelta 1: Il file è vuoto (CFJ1) [if fileOk]
        * Scelta 2: Il file non contiene codice java corretto (CFJ2) [if fileOk property logError]
        * Scelta 3: Il file contiene codice java corretto (CFJ3) [if fileOk]
    - Categoria: Esistenza file (ESTM)
        * Scelta 1: File di mining_results_sm_final.csv già esistente (ESTM1) [if fileOk] [single]
        * Scelta 2: File di di mining_results_sm_final.csv non esistente (ESTM2) [if fileOk]

- **Oggetto dell'ambiente**: file di log
    - Categoria: Esistenza log (EL)
        * Scelta 1: File di log già esistente (EL1) [single] [if logError]
        * Scelta 2: File di log non esistente (EL2)

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_7.1	DF1	FileNotFoundError
TCI_7.2	DF2 ES1	Il file csv viene creato con solo l'header
TCI_7.3	DF2 ES2 CN1	Il file csv viene creato con solo l'header
TCI_7.4	DF2 ES2 CN2 STC1	Il file csv viene creato con solo l'header
TCI_7.5	DF2 ES2 CN2 STC2	Il file csv viene creato con solo l'header
TCI_7.6	DF2 ES2 CN2 STC3 CNC1	Il file csv viene creato con solo l'header
TCI_7.7	DF2 ES2 CN2 STC3 CNC2 STF1	Il file csv viene creato con solo l'header
TCI_7.8	DF2 ES2 CN2 STC3 CNC2 STF2	Il file csv viene creato con solo l'header
TCI_7.9	DF2 ES2 CN2 STC3 CNC2 STF3 CNF1	Il file csv viene creato con solo l'header
TCI_7.10	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1	Il file csv viene creato con solo l'header
TCI_7.11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI2	Il file csv viene creato con solo l'header
TCI_7.12	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ1	Il file di log viene creato vuoto e i file vengono inserit nel csv con tutti 0 per le metriche
TCI_7.13	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM2 EL2	Il log viene creato con un errore per ogni file errato e i file vengono aggiunti al csv con i valori individuati prima dell'errore
TCI_7.14	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ2 ESTM2 EL1	Al log viene aggiunto un errore per ogni file errato e i file vengono aggiunti al csv con i valori individuati prima dell'errore
TCI_7.15	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ3 ESTM2 EL2	Il log viene creato vuoto i file vengono aggiunti al csv con i valori individuati per le metriche
TCI_7.16	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3 CFJ3 ESTM1 EL2	Il log viene creato vuoto i file vengono aggiunti al csv con i valori individuati per le metriche
TCI_7.17	DF2 ES2 CN2 STC3 CNC2	Il csv viene creato con solo l'header

Test Case	Combinazione	Oracolo
	STF3 CNF2 CNFI4	

Tabella 2.14: Test case per il modulo Software_Metrics

2.3 Change Request 6

La change request 6 ha comportato l'aggiunta di una classe SonarAnalyzer per l'automatizzazione del processo di analisi statica. É stato necessario introdurre metodi di test per la nuova classe, per verificarne il corretto funzionamento.

2.3.1 Unit testing

ASA

Per quanto riguarda la funzione **create_sonar_properties**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: project_key:
  - Categoria: Esistenza della stringa (E)
    - Scelta 1: La stringa è nulla [error]
    - Scelta 2: La stringa non è nulla
- **Variabile 2**:commit_dir
  - Categoria: Validità del path (V)
    - Scelta 1: La directory non contiene file java [error]
    - Scelta 2: La directory contiene file java

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.1_1	E1	FileNotFoundError
TC_8.1_2	E2 V1	FileNotFoundError
TC_8.1_3	E2 V2	La funzione crea il file sonar-project.properties nella directory commit_dir

Tabella 2.15: Test case per la funzione create_sonar_properties

Per quanto riguarda la funzione **run_sonar_scanner**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: commit_dir:

  - Categoria: Esistenza directory (DIR)
    - Scelta 1: Directory non esistente [error]
    - Scelta 2: Directory esistente

- **Variabile 2**: sonar_project_key:
  - Categoria: Esistenza directory (KEY)
    - Scelta 1: Key del progetto non valida [error]
    - Scelta 2: Key del progetto valida

- **Oggetto dell'ambiente 1**: logger:
  - Categoria: Esistenza (LOG)
    - Scelta 1: logger non esistente [error]
    - Scelta 2: logger esistente

- **Oggetto dell'ambiente 2**: SonarQube:
  - Categoria: Stato (SQ)
    - Scelta 1: SonarQube non è attivo [error]
    - Scelta 2: SonarQube è attivo

- **Oggetto dell'ambiente 3**: sonar_host:
  - Categoria: Validità host (H)
    - Scelta 1: Host non valido [error]
    - Scelta 2: Host valido

- **Oggetto dell'ambiente 4**: sonar_token:
  - Categoria: Correttezza token (T)
    - Scelta 1: Token non valido [error]
    - Scelta 2: Token valido

- **Oggetto dell'ambiente 5**: sonar_path:
  - Categoria: Correttezza path di SonarScanner (SS)
    - Scelta 1: Percorso ad un altro file [error]
    - Scelta 2: Percorso del file bath di SonarScanner

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.2_1	DIR1	FileNotFoundError
TC_8.2_2	DIR2 KEY1	Exception, con errore riportato nel logger
TC_8.2_3	DIR2 KEY1 LOG1	AttributeError
TC_8.2_4	DIR2 KEY2 LOG2 SQ1	Exception, con errore riportato nel logger
TC_8.2_5	DIR2 KEY2 LOG2 SQ2	Exception, con errore riportato nel logger
TC_8.2_6	DIR2 KEY2 LOG2 SQ2 H1	Exception, con errore riportato nel logger
TC_8.2_7	DIR2 KEY2 LOG2 SQ2 H2 T2 SS1	Exception, con errore riportato nel logger
TC_8.2_8	DIR2 KEY2 LOG2 SQ2 H2 T2 SS2	Il comando SonarScanner viene eseguito con successo

Tabella 2.16: Test case per la funzione run_sonar_scanner

Per quanto riguarda la funzione **get_analysis_id**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: project_key:
  - Categoria: Esistenza directory (KEY)
    - Scelta 1: Key del progetto non valida [error]
    - Scelta 2: Key del progetto valida
- **Oggetto dell'ambiente 1**: SonarQube:
  - Categoria: Stato (SQ)
    - Scelta 1: SonarQube non è attivo [error]
    - Scelta 2: SonarQube è attivo
- **Oggetto dell'ambiente 3**: sonar_host:
  - Categoria: Validità host (H)
    - Scelta 1: Host non valido [error]
    - Scelta 2: Host valido
- **Oggetto dell'ambiente 4**: sonar_token:
  - Categoria: Correttezza token (T)

        * Scelta 1: Token non valido [error]
        * Scelta 2: Token valido

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.3_1	KEY1	Exception
TC_8.3_2	KEY2 SQ1	Exception
TC_8.3_3	KEY2 SQ2 H1	Exception
TC_8.3_4	KEY2 SQ2 H2 T1	None
TC_8.3_5	KEY2 SQ2 H2 T2	La funzione restituisce l'ID del progetto del server SonarQube

Tabella 2.17: Test case per la funzione get_analysis_id

Per quanto riguarda la funzione **check_analysis_status**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: analysis_id:
  - Categoria: Esistenza progetto (ID)
    * Scelta 1: Id non valido o di un progetto non analizzato [error]
    * Scelta 2: Id valido (progetto analizzato)
- **Oggetto dell'ambiente 1**:sonar_host:
  - Categoria: Validità host (H)
    * Scelta 1: Host non valido [error]
    * Scelta 2: Host valido
- **Oggetto dell'ambiente 2**:sonar_token:
  - Categoria: Correttezza token (T)
    * Scelta 1: Token non valido [error]
    * Scelta 2: Token valido

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.4_1	ID1	None
TC_8.4_2	ID2 H1	Exception
TC_8.4_3	ID2 H2 T1	Exception
TC_8.4_4	ID2 H2 T2	La funzione restituisce lo stato del progetto ('SUCCESS' o 'FAILED')

Tabella 2.18: Test case per la funzione check_analysis_status

Per quanto riguarda la funzione **get_project_issues**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: project_key:
  - Categoria: Esistenza progetto (ID)
    - Scelta 1: Id non valido o di un progetto non analizzato [error]
    - Scelta 2: Id valido (progetto analizzato)
- **Oggetto dell'ambiente 1**: logger:
  - Categoria: Esistenza (LOG)
    - Scelta 1: logger non esistente [error]
    - Scelta 2: logger esistente
- **Oggetto dell'ambiente 2**:sonar_host:
  - Categoria: Validità host (H)
    - Scelta 1: Host non valido [error]
    - Scelta 2: Host valido
- **Oggetto dell'ambiente 3**:sonar_token:
  - Categoria: Correttezza token (T)
    - Scelta 1: Token non valido [error]
    - Scelta 2: Token valido

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.5_1	ID1	La lista delle issues è vuota
TC_8.5_2	ID2 LOG1	AttributeError
TC_8.5_3	ID2 LOG2 H1	La lista delle issues è vuota
TC_8.5_4	ID2 LOG2 H2 T1	La lista delle issues è vuota
TC_8.5_5	ID2 LOG2 H2 T2	La funzione restituisce lo issue del progetto

Tabella 2.19: Test case per la funzione get_project_issues

Per quanto riguarda la funzione **save_issues_to_csv**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: issue:
  - Categoria: Lunghezza (LI)
    - Scelta 1: La lista di issue è vuota [property EmptyIssue]
    - Scelta 2: La lista di issue non è vuota [property notEmptyIssue]
- **Variabile 2**: project:
  - Categoria: Lunghezza (LP)
    - Scelta 1: La directory esiste [if EmptyIssue]
    - Scelta 2: La directory non esiste [if EmptyIssue][error]
  - Categoria: Contenuto (CP)
    - Scelta 1: La directory contiene file java [if EmptyIssue]
    - Scelta 2: La directory non contiene file java [if EmptyIssue][single]
- **Oggetto dell'ambiente 1**: file_exists:
  - Categoria: Esistenza (ECSV)
    - Scelta 1: Il file non esiste
    - Scelta 2: Il file esiste

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.6_1	L1 LP2	FileNotFoundError
TC_8.6_2	L1 LP1 CP2 ECSV2	Viene creato un nuovo file, ed aggiunta l'intestazione
TC_8.6_3	L1 LP1 CP1 ECSV1	Viene creato un nuovo file, aggiunta l'intestazione, e una issue in al progetto contenuto in *project*
TC_8.6_4	L1 LP1 CP1 ECSV2	Viene aggiunta la issue in append
TC_8.6_5	L2 LP1 ECSV1	Viene creato un nuovo file, aggiunta l'intestazione, e le issue contenu
TC_8.6_6	L2 LP1 ECSV2	Viene aggiunta la issue in append

Tabella 2.20: Test case per la funzione save_issues_to_csv

Per quanto riguarda la funzione **process_repositories**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: mining_results:
  - Categoria: Esistenza (MR)
    - Scelta 1: La directory mining_results esiste
    - Scelta 2: La directory mining_results non esiste [error]
  - Categoria: Sottodirectory (RM)
    - Scelta 1: Esiste una directory RepositoryMining
    - Scelta 2: Non esiste nessuna directory RepositoryMining [property emptyRM]
- **Variabile 2**: cve_id:
  - Categoria: Contenuto (CVE)
    - Scelta 1: La directory RepositoryMining contiene sottodirectory [if not emptyRM]
    - Scelta 2: La directory RepositoryMining è vuota [if not emptyRM][property emptyCVE]
- **Variabile 3**: commit_id:
  - Categoria: Validità (CI)
    - Scelta 1: La directory commit_id esiste e contiene file .java [if not emptyCVE]
    - Scelta 2: La directory commit_id non contiene file .java [error][if not emptyCVE] [property noFiles]

- **Oggetto dell'ambiente 1**: SonarQube:
  - Categoria: Validità (SQ)
    - Scelta 1: La connessione al server SonarQube è attiva e accessibile. [if not noFiles]
    - Scelta 2: La connessione al server SonarQube è interrotta [error]
- **Oggetto dell'ambiente 2**: issue:
  - Categoria: Validità (I)
    - Scelta 1: La lista delle issue è vuota
    - Scelta 2: La lista delle issue non è vuota

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_8.7_1	MR2	FileNotFoundError
TC_8.7_2	MR1 RM2	Non viene generato alcun file
TC_8.7_3	MR1 RM1 CVE2	Non viene generato alcun file
TC_8.7_4	MR1 RM1 CVE1 CI2	Exception
TC_8.7_5	MR1 RM1 CVE1 CI1 SQ2	Exception
TC_8.7_6	MR1 RM1 CVE1 CI1 SQ2 I1	Viene creato il file di output ed inserito il progetto java
TC_8.7_7	MR1 RM1 CVE1 CI1 SQ2 I2	Viene creato il file di output ed inserite le vulnerabilità prese da SonarQube

Tabella 2.21: Test case per la funzione process_repositories

Ai test sono stati aggiunti anche quelli della funzione run_ASA della classe Main per avviare il calcolo delle metriche sui file interessati. Abbiamo definito per questa funzione le seguenti categorie e scelte per testarne il funzionamento:

- **Variabile 1**: sonar_host
  - Categoria: Accessibilità (H)
    - Scelta 1: L'host non è accessibile [error]
    - Scelta 2: L'host è accessibile

- **Variabile 2**: sonar_token
  - Categoria: Validità del token (T)
    - Scelta 1: Il token di SonarQube non è valido [error]
    - Scelta 2: Il token di SonarQube è valido
- **Variabile 3**: sonar_path
  - Categoria: Correttezza del path di SonarScanner (SS)
    - Scelta 1: Il path di SonarScanner non è corretto [error]
    - Scelta 2: Il path di SonarScanner è corretto
- **Oggetto dell'ambiente 1**: base_dir
  - Categoria: Esistenza directory (BD)
    - Scelta 1: Il percorso non esiste [error]
    - Scelta 2: Il percorso esiste
  - Categoria: Esistenza sottodirectory "RepositoryMining"(ER)
    - Scelta 1: la directory esiste
    - Scelta 2: la directory non esiste
  - Categoria: Contenuto (CR)
    - Scelta 1: La directory contiene progetti senza issue
    - Scelta 2: La directory contiene progetti con issue

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_8.8_1	H1	Exception, Host unreachable
TCI_8.8_2	H2 T1	Exception, Host unreachable
TCI_8.8_3	H2 T2 SS1	Exception, Host unreachable
TCI_8.8_4	H2 T2 SS2 BD1	Exception
TCI_8.8_5	H2 T2 SS2 BD2 ER1 CR1	File di output senza vulnerabilità
TCI_8.8_6	H2 T2 SS2 BD2 ER1 CR2	File di output con vulnerabilità
TCI_8.8_7	H2 T2 SS2 BD2 ER2	File di output senza vulnerabilità

Tabella 2.22: Test case run_ASA

2.3.2 Integration testing

ASA

Il metodo che si occupa di avviare l'analisi statica dei vari file java utilizzano SonarQube e SonarScanner è run_ASA della classe Main. Per tale metodo abbiamo individuato le seguenti categorie e possibili scelte su cui basare i nostri test case:

- **Variabile 1**: sonar_host
  - Categoria: Accessibilità (H)
    - Scelta 1: L'host non è accessibile [error]
    - Scelta 2: L'host è accessibile
- **Variabile 2**: sonar_token
  - Categoria: Validità del token (T)
    - Scelta 1: Il token di SonarQube non è valido [error]
    - Scelta 2: Il token di SonarQube è valido
- **Variabile 3**: sonar_path
  - Categoria: Correttezza del path di SonarScanner (SS)
    - Scelta 1: Il path di SonarScanner non è corretto [error]
    - Scelta 2: Il path di SonarScanner è corretto
- **Oggetto dell'ambiente 1**: base_dir
  - Categoria: Esistenza directory (BD)
    - Scelta 1: Il percorso non esiste [error]
    - Scelta 2: Il percorso esiste
  - Categoria: Esistenza sottodirectory "RepositoryMining"(ER)
    - Scelta 1: la directory esiste
    - Scelta 2: la directory non esiste
  - Categoria: Contenuto (CR)
    - Scelta 1: La directory contiene progetti senza issue
    - Scelta 2: La directory contiene progetti con issue

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_3.1	H1	Exception, Host unreachable
TCI_3.2	H2 T1	Exception, Host unreachable
TCI_3.3	H2 T2 SS1	Exception, Host unreachable
TCI_3.4	H2 T2 SS2 BD1	Exception
TCI_3.5	H2 T2 SS2 BD2 ER1 CR1	File di output senza vulnerabilità
TCI_3.6	H2 T2 SS2 BD2 ER1 CR2	File di output con vulnerabilità
TCI_3.7	H2 T2 SS2 BD2 ER2	File di output senza vulnerabilità

Tabella 2.23: Test case per il modulo mining_results_asa

2.4 Change Request 7

L'introduzione dei modelli di intelligenza artificiale ha portato l'aggiunta di un nuovo metodo run_prediction nella classe Main per la predizione delle vulnerabilità. Quindi necessita la realizzazione di test per verificarne il correto funzionamento.

2.4.1 Unit testing

Modulo IA

Per il metodo **run_prediction** abbiamo individuato le seguenti categorie e possibili scelte su cui basare i nostri test case:

- **Variabile 1**: vocab_path,
  - Categoria: Esistenza del vocabolario (EV)
    - Scelta 1: Directory non esistente (EV1) [error]
    - Scelta 2: Directory esistente (EV2) [property vocabOk]
  - Categoria: Estensione file del vocabolario (EFV)
    - Scelta 1: Estensione diversa da .pkl (EFV1) [error]
    - Scelta 2: Estensione .pkl (EFV2) [if vocabOk property vpklOk]
- **Variabile 2**: model_path,
  - Categoria: Esistenza del modello (EM)
    - Scelta 1: Directory non esistente (EM1) [error]
    - Scelta 2: Directory esistente (EM2) [if vpklOk property modelOk]
  - Categoria: Estensione file del modello (EFM)
    - Scelta 1: Estensione diversa da .pkl (EFM1) [error]
    - Scelta 2: Estensione .pkl (EFM2) [if modelOk property mpklOk]
- **Variabile 3**: label_encoder_path,
  - Categoria: Esistenza dell'encoder(EE)
    - Scelta 1: Directory non esistente (EE1) [error]
    - Scelta 2: Directory esistente (EE2) [if mpklOk property labelOk]
  - Categoria: Estensione file dell'encoder (EFE)

        * Scelta 1: Estensione diversa da .pkl (EFE1)
        * Scelta 2: Estensione .pkl (EFE2) [if labelOk property lpklOk]

- **Variabile 4**: input_csv_path
  - Categoria: Esistenza file di input(EI)
    * Scelta 1: Directory non esistente (EI1) [error]
    * Scelta 2: Directory esistente (EI2) [if lpklOk property inputOk]
  - Categoria: Estensione file di input (EFI)
    * Scelta 1: Estensione diversa da .csv (EFI1) [error]
    * Scelta 2: Estensione .csv (EFI2) [if inputOk property pathOk]
  - Categoria: Contenuto del file di input (CI)
    * Scelta 1: Il file è vuoto (CI1) [error]
    * Scelta 2: Non contiene la colonna Name (CI2) [error]
    * Scelta 2: Contiene la colonna Name (CI3) [if pathOk property contOk]

- **Variabile 5**: X_new
  - Categoria: Contenuto input del modello (CIM)
    * Scelta 1: L'input dato non corrisponde ai dati su cui è stato addestrato il modello (CIM1) [error]
    * Scelta 2: L'input dato corrisponde ai dati su cui è stato adddestrato il modello (CIM2) [if contOk property predOk]

- **Variabile 5**: predictions
  - Categoria: Etichette delle prediction (EP)
    * Scelta 1: Etichette non previste (EP1) [error]
    * Scelta 2: Etichette previste (EP2) [if predtOk property inputOk]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_7.1	EV1	FileNotFoundError
TCI_7.2	EV2 EFV1	ValueError
TCI_7.3	EV2 EFV2 EM1	FileNotFoundError

Test Case	Combinazione	Oracolo
TCI_7.4	EV2 EFV2 EM2 EFM1	ValueError
TCI_7.5	EV2 EFV2 EM2 EFM2 EE1	FileNotFoundError
TCI_7.6	EV2 EFV2 EM2 EFM2 EE2 EFE1	ValueError
TCI_7.7	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI1	FileNotFoundError
TCI_7.8	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI1	ValueError
TCI_7.9	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI2 CI1	KeyError
TCI_7.10	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI2 CI2	KeyError
TCI_7.11	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI2 CI3 CIM1	ValueError
TCI_7.12	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI2 CI3 CIM2 EP1	ValueError
TCI_7.13	EV2 EFV2 EM2 EFM2 EE2 EFE2 EI2 EFI2 CI3 EP2	Il file csv viene creato con il nome dei file e le relative classi predette

Tabella 2.24: Test case per la funzione run_prediction

2.4.2 Integration testing

Modulo IA

Di seguito è illustrato come è stata verificata l'integrazione del metodo run_prediction con la navigazione tra le directory del sistema, l'utilizzo dei modelli di intelligenza artificiale e l'encoding delle loro predizioni. Per tale metodo abbiamo individuato le seguenti categorie e possibili scelte su cui basare i nostri test case:

- **Variabile 1**: vocab_path,
  - Categoria: Esistenza del vocabolario (EV)
    - Scelta 1: Directory non esistente (EV1) [error]

        * Scelta 2: Directory esistente (EV2) [property vocabOk]
    - Categoria: Contenuto vocabolario (CV)
        * Scelta 1: Contiene il vocabolario nel formato atteso (CV1) [error]
        * Scelta 2: Non contiene un vocabolario (CV2) [if vocabOk property contvOk]

- **Variabile 2**: model_path,
    - Categoria: Esistenza del modello (EM)
        * Scelta 1: Directory non esistente (EM1) [error]
        * Scelta 2: Directory esistente (EM2) [if contvOk property modelOk]
    - Categoria: Contenuto modello (CM)
        * Scelta 1: Contiene il modello nel formato atteso (Cm1) [error]
        * Scelta 2: Non contiene un modello (Cm2) [if modelOk property contmOk]

- **Oggetto d'ambiente 1**: Coerenza file
    - Categoria: Compatibilità vocabolario e modello (COM)
        * Scelta 1: Il modello è quello addestrato sul vocabolario scelto (COM1) [if contmOk]
        * Scelta 2: Il modello non è quello addestrato sul vocabolario scelto (COM2) [error]

- **Variabile 3**: label_encoder_path,
    - Categoria: Esistenza dell'encoder(EE)
        * Scelta 1: Directory non esistente (EE1) [error]
        * Scelta 2: Directory esistente (EE2) [if mpklOk property labelOk]

- **Variabile 4**: input_csv_path
    - Categoria: Esistenza file di input(EI)
        * Scelta 1: Directory non esistente (EI1) [error]
        * Scelta 2: Directory esistente (EI2) [if labelOk property inputOk]
    - Categoria: Contenuto header del file di input (CI)
        * Scelta 1: Il file è vuoto (CI1) [error]
        * Scelta 2: Non contiene la colonna Name (CI2) [error]

    * Scelta 2: Contiene la colonna Name con colonne non presenti nel vocabolario (CI3) [if inputOk property contOk]
    * Scelta 2: Contiene la colonna Name con colonne presenti nel vocabolario (CI4) [if inputOk property contOk]
  - Categoria: Contenuto del file di input (CI)
    * Scelta 1: Contiene dati di tipo stringa (CFI1) [if contOk]
    * Scelta 2: Contiene dati di tipo intero (CFI2) [if contOk]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TCI_7.1	EV1	FileNotFoundError
TCI_7.2	EV2 CV1	KeyError
TCI_7.3	EV2 CV2 EM1	FileNotFoundError
TCI_7.4	EV2 CV2 EM2 CM1	KeyError
TCI_7.5	EV2 CV2 EM2 CM2 COM1 EE1	FileNotFoundError
TCI_7.6	EV2 CV2 EM2 CM2 COM1 EE2 EI1	FileNotFoundError
TCI_7.7	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI1	EmptyDataError
TCI_7.8	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI2	KeyError
TCI_7.9	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI3 CFI1	Il csv viene creato col nome dei file e le relative predizioni
TCI_7.10	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI3 CFI2	Il csv viene creato col nome dei file e le relative predizioni
TCI_7.11	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI4 CFI1	Il csv viene creato col nome dei file e le relative predizioni
TCI_7.12	EV2 CV2 EM2 CM2 COM1 EE2 EI2 CI4 CFI2	KeyError
TCI_7.13	EV2 CV2 EM2 CM2 COM2 EE2 EI2 CI4 CFI2	KeyError

Tabella 2.25: Test case per la funzione run_prediction

2.5 Change Request 8

La generalizzazione della gestione dei percorsi ha comportato l'aggiunta dei test per testare la variabile base_dir presente nelle varie classi.

2.6 Unit testing

Repo Mining

Riguardo il testing della funzione **divide_dataset**, appartenente alla classe DatasetDivider, è stata introdotta una nuova variabile, il percorso base_dir. Sono state considerate quindi le seguenti variabili con le rispettive categorie e scelte:

- Initial_Dataset.csv, ovvero il dataset iniziale.
  - Categoria: esistenza (EI)
    - Scelta 1: esiste (EI1) [property dataExistOk]
    - Scelta 2: non esiste (EI2) [error]
  - Categoria: numero record (N)
    - Scelta 1: 0 (N1) [if dataExistOk]
    - Scelta 2: 50 (N2) [if dataExistOk]
    - Scelta 3: maggiore di 50 (N3) [if dataExistOk]
- Dir Dataset_Divided, ovvero la directory contenente i dataset suddivisi.
  - Categoria: esistenza (ED)
    - Scelta 1: esiste (ED1)
    - Scelta 2: non esiste (ED2) [single]
- n.csv, ovvero il file CSV contenente una parte di commit .
  - Categoria: possibilità di scrittura (S)
    - Scelta 1: sì (S1)
    - Scelta 2: no (S2) [error]
- **Variabile 1**: base_dir
  - Categoria: Formato (F)
    - Scelta 1: Formato valido
    - Scelta 2: Formato non valido [error]
    - Scelta 3: Non stringa [error]
  - Categoria: Esistenza (EB)

    * Scelta 1: Percorso esistente
    * Scelta 2: Percorso non esistente [error]

Considerando le categorie e le scelte già definite, è stata ottenuta la seguente test suite:

Test Case	Combinazione	Oracolo
TC_1.1_1	EI2	FileNotFoundError
TC_1.1_2	EI1 N1 ED1 S1 F1 E1	Dataset_Divided con un singolo CSV contenente solo l'header
TC_1.1_3	S2	PermissionError
TC_1.1_4	EI1 N2 ED1 S1 F1 E1	Dataset_Divided con un singolo file CSV contenente i 50 record del dataset
TC_1.1_5	EI1 N3 ED1 S1 F1 E1	Dataset_Divided contenente file CSV con i record del dataset suddivisi in gruppi di 50
TC_1.1_6	EI1 N3 ED2 S1 F1 E1	Dataset_Divided creata e contenente file CSV con i record del dataset suddivisi in gruppi di 50
TC_1.1_7	F3	TypeError
TC_1.1_8	F2	OSError
TC_1.1_9	F1 EB2	FileNotFoundError

Tabella 2.26: Test case per la funzione divide_dataset

Anche per la funzione **initialize_repo_mining**, è stata introdotta una nuova variabile, il percorso base_dir. Sono state considerate, quindi, le seguenti variabili e le relative categorie e scelte:

- miniDatasetName, ovvero il parametro che indica il nome di uno dei sottodataset da minare.
    - Categoria: formato (FM)
        * Scelta 1: formato valido (FM1)
        * Scelta 2: formato non valido (FM2) [error]
- Il dataset da minare

  - Categoria: esistenza (ED)
    - Scelta 1: esiste (ED1) [property dataExistOk]
    - Scelta 2: non esiste (ED2) [error]
  - Categoria: formato (FD)
    - Scelta 1: formato valido (FD1) [if dataExistOk] [property dataFormatOk]
    - Scelta 2: formato non valido (FD2) [error]
  - Categoria: numero record (ND)
    - Scelta 1: zero (ND1) [if dataExistOk]
    - Scelta 2: uno (ND2) [if dataExistOk]
    - Scelta 3: maggiore di uno (ND3) [if dataExistOk]

- La directory in cui salvare i risultati del mining (mining_results)
  - Categoria: esistenza (EM)
    - Scelta 1: esiste (EM1) [property mrExistOk]
    - Scelta 2: non esiste (EM2) [error]

- La directory in cui salvare i risultati del mining per ciascuna repo.
  - Categoria: esistenza (ER)
    - Scelta 1: esiste (ER1) [if mrExistOk]
    - Scelta 2: non esiste (ER2) [if mrExistOk] [single]

- base_dir, il percorso alla directory Dataset2
  - Categoria: Formato (FB)
    - Scelta 1: Formato valido
    - Scelta 2: Formato non valido [error]
  - Categoria: Esistenza (EB)
    - Scelta 1: Percorso esistente
    - Scelta 2: Percorso non esistente [error]

Ciò ha prodotto la seguente test suite:

Test Case	Combinazione	Oracolo
TC_1.2_1	FM2	OSError
TC_1.2_2	FM1 ED2	FileNotFoundError
TC_1.2_3	EM2	FileNotFoundError
TC_1.2_4	FM1 ED1 EM1 FD2	Value Error
TC_1.2_5	FM1 ED1 FD1 ND3 EM1 ER1	Il dizionario data contiene i record contenuti nel dataset
TC_1.2_6	FM1 ED1 FD1 ND1 EM1 ER2	Il dizionario data è vuoto
TC_1.2_7	FM1 ED1 FD1 ND1 EM1 ER1	Il dizionario data è vuoto
TC_1.2_8	FM1 ED1 FD1 ND2 EM1 ER1	Il dizionario data contiene il record contenuto nel dataset
TC_1.2_9	FB2	OSError
TC_1.2_10	FB1 EB2	FileNotFoundError

Tabella 2.27: Test case per la funzione initialize_repo_mining

Lo stesso è avvenuto per la funzione **start_mining_repo**, con l'introduzione della variabile base_dir.

Sono state quindi considerate le seguenti variabili:

- data, ovvero il parametro contenente i riferimenti ai commit da analizzare.
  - Categoria: formato (FD)
    - Scelta 1: formato valido (FD1) [property dataFormatOk]
    - Scelta 2: formato non valido (FD2) [error]
  - Categoria: numero record (N)
    - Scelta 1: zero (N1)
    - Scelta 2: uno (N2) [property dataNotEmpty]
    - Scelta 3: maggiore di uno (N3) [property dataNotEmpty]
  - Categoria: contenuto link (CL)
    - Scelta 1: link esistente e valido (CL1) [if dataFormatOk and dataNotEmpty] [property linkValidOk]

        * Scelta 2: link valido e non esistente (CL2) [if dataFormatOk and dataNotEmpty] [error]
        * Scelta 3: link non valido (CL3) [if dataFormatOk and dataNotEmpty] [error]
    - Categoria: contenuto repo (CR)
        * Scelta 1: presente repo valida (CR1) [if linkValidOk] [property repoValidOk]
        * Scelta 2: presente repo non valida (CR2) [if linkValidOk]
    - Categoria: Contenuto commit (CC)
        * Scelta 1: Presente commit valido (CC1) [if repoValidOk] [property commitValidOk]
        * Scelta 2: Presente commit valido con path che eccede la lunghezza massima del S.O. (CC2) [if repoValidOk]
        * Scelta 3: Presente commit non valido con risposta non riuscita (CC3) [if repoValidOk]
        * Scelta 4: Presente commit non valido con risposta riuscita (CC4) [if repoValidOk]

- reponame, ovvero il parametro contenente il nome del repository.
    - Categoria: formato (FR)
        * Scelta 1: stringa con formato valido (FR1) [property reponameFormatOk]
        * Scelta 2: stringa con formato non valido (FR2) [error]
        * Scelta 3: non stringa (FR3) [error]
    - Categoria: esistenza repository (ER)
        * Scelta 1: esiste (ER1) [if reponameFormatOk]
        * Scelta 2: non esiste (ER2) [if reponameFormatOk] [error]

- Repo Cvd_id
    - Categoria: esistenza (ERI)
        * Scelta 1: esiste (ERI1)
        * Scelta 2: non esiste (ERI2)

- Repo Commit
    - Categoria: esistenza (ERC)

        * Scelta 1: esiste (ERC1) [single]
        * Scelta 2: non esiste (ERC2)

- CHECK.txt, ovvero il file che contiene i log dell'attività di mining.
    - Categoria: esistenza (EC)
        * Scelta 1: esiste (EC1)
        * Scelta 2: non esiste (EC2) []
    - Categoria: possibilità di scrittura (SC)
        * Scelta 1: sì (SC1)
        * Scelta 2: no (SC2)

- ERROR.txt, ovvero il file che contiene le informazioni sugli errori.
    - Categoria: esistenza (EE)
        * Scelta 1: esiste (EE1)
        * Scelta 2: non esiste (EE2)
    - Categoria: possibilità di scrittura (SE)
        * Scelta 1: sì (SE1)
        * Scelta 2: no (SE2)

- base_dir, ovvero il percorso alla directory Dataset2
    - Categoria: Formato (FB)
        * Scelta 1: Formato valido
        * Scelta 2: Formato non valido [error]
    - Categoria: Esistenza (EB)
        * Scelta 1: Percorso esistente
        * Scelta 2: Percorso non esistente [error]

È stata considerata, quindi, la seguente test-suite:

Test Case	**Combinazione**	**Oracolo**

TC_1.3_1	FD1 N1 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECK.txt vuoto
TC_1.3_2	FD2	KeyError
TC_1.3_3	FD1 N3 CL3	MissingSchema
TC_1.3_4	FD1 N3 CL2	ConnectionError
TC_1.3_5	FD1 N3 CL1 CR2 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECKS.txt indica che la repo non è disponibile
TC_1.3_6	FD1 N3 CL1 CR1 CC2 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECKS.txt indica che il commit non è esistente
TC_1.3_7	FD1 N3 CL1 CR1 CC3 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECKS.txt indica che il commit non è esistente
TC_1.3_8	FD1 N3 CL1 CR1 CC4 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECKS.txt e ERRORS.txt indicano che il commit non è definito
TC_1.3_9	FD1 N3 CL1 CR1 CC1 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1 FB1 EB1	CHECKS.txt indica status ok
TC_1.3_10	FR3	TypeError
TC_1.3_11	FR2	OSError
TC_1.3_12	FR1 ER2	FileNotFoundError
TC_1.3_13	FB2	OSError
TC_1.3_14	FB1 EB2	FileNotFoundError
TC_1.3_15	EC1 SC2	PermissionError
TC_1.3_16	EE1 SE2	PermissionError

Tabella 2.28: Test case per la funzione start_mining_repo

2.6.1 Integration testing

Repo Mining

Riguardo il testing di integrazione del modulo repo mining, la generalizzazione dei path ha comportato l'utilizzo di una nuova variabile, il percorso base_dir. Sono state considerate quindi le seguenti variabili con le rispettive categorie e scelte:

- **Oggetto dell'ambiente 1:** initial_dataset.csv
  - Categoria: Esistenza (EI)
    - Scelta 1: Esiste (EI1) [property dataExistOk]
    - Scelta 2: Non esiste (EI2) [error]
  - Categoria: Formato (FI)
    - Scelta 1: Formato valido (FD1) [if dataExistOk] [property dataFormatOk]
    - Scelta 2: Formato non valido (FD2) [error]
  - Categoria: Numero record (N)
    - Scelta 1: 0 (N1) [if dataExistOk]
    - Scelta 2: $\leq$50 (N2) [if dataExistOk] [property dataNotEmpty]
    - Scelta 3: >50 (N3) [if dataExistOk] [property dataNotEmpty] [single]
  - Categoria: Contenuto link (CL)
    - Scelta 1: Link esistente e valido (CL1) [if dataFormatOk and dataNotEmpty] [propery linkValidOk]
    - Scelta 2: Link valido e non esistente (CL2) [if dataFormatOk and dataNotEmpty]
    - Scelta 3: Link non valido (CL3) [if dataFormatOk and dataNotEmpty]
  - Categoria: Contenuto repo (CR)
    - Scelta 1: Presente repo valida (CR1) [if linkValidOk] [property repoValidOk]
    - Scelta 2: Presente repo non valida (CR2) [if linkValidOk]
  - Categoria: Contenuto commit (CC)
    - Scelta 1: Presente commit valido (CC1) [if repoValidOk] [property commitValidOk]
    - Scelta 2: Presente commit valido con path che eccede la lunghezza massima del S.O. (CC2) [if repoValidOk]

        * Scelta 3: Presente commit non valido con risposta non riuscita (CC3) [if repoValidOk]
        * Scelta 4: Presente commit non valido con risposta riuscita (CC4) [if repoValidOk]
    - Categoria: Contenuto modifica (CM)
        * Scelta 1: Non esiste una modifica per un commit (CM1) [if commitValidOk]
        * Scelta 2: Esiste una modifica per commit e non comprende classi java (CM2) [if commitValidOk]
        * Scelta 3: Esiste una modifica per commit che riguarda classi .java introdotte con esso (CM3) [if commitValidOk]
        * Scelta 4: Esiste una modifica per commit che riguarda classi .java già presenti prima di esso (CM4) [if commitValidOk]

- **Oggetto dell'ambiente 2:** directory Dataset_Divided
    - Categoria: Esistenza (ED)
        * Scelta 1: Esiste (ED1)
        * Scelta 2: Non esiste (ED2) [single]

- **Oggetto dell'ambiente 3:** directory mining_results
    - Categoria: Esistenza (EM)
        * Scelta 1: Esiste (EM1)
        * Scelta 2: Non esiste (EM2) [error]

- **Oggetto dell'ambiente 4:** file CHECK.txt
    - Categoria: Esistenza (EC)
        * Scelta 1: Esiste (EC1)
        * Scelta 2: Non esiste (EC2) [single]

- **Oggetto dell'ambiente 5:** file ERRORS.txt
    - Categoria: Esistenza (EE)
        * Scelta 1: Esiste (EE1)
        * Scelta 2: Non esiste (EE2) [single]

- **Variabile 1**: base_dir
  - Categoria: Formato (FB)
    - Scelta 1: Formato valido (FB1)
    - Scelta 2: Formato non valido (FB2) [error]
    - Scelta 3: Non stringa (FB3) [error]
  - Categoria: Esistenza (EB)
    - Scelta 1: Percorso esistente (EB1)
    - Scelta 2: Percorso non esistente (EB2) [error]

Test Case	Combinazione	Oracolo
TCI_1.1	EI2	FileNotFoundError
TCI_1.2	EM2	FileNotFoundError
TCI_1.3	EI1 FD2	KeyError
TCI_1.4	EI1 FI1 N1 ED2 EM1 EC2 EE2 EB1 FB1	Dataset_Divided creata e contiene 1.csv e mining_results vuota
TCI_1.5	EI1 FI1 N2 CL3	Missing Schema Error
TCI_1.6	EI1 FI1 N2 CL2	Connection Error
TCI_1.7	EI1 FI1 N3 CL1 CR2 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica che la repo non è disponibile
TCI_1.8	EI1 FI1 N2 CL1 CR1 CC3 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica che il commit non è esistente
TCI_1.9	EI1 FI1 N2 CL1 CR1 CC4 ED1 EM1 EC1 EE1 EB1 FB1	CHECKS.txt e ERRORS.txt indicano che il commit non è definito
TCI_1.10	EI1 FI1 N2 CL1 CR1 CC2 ED1 EM1 EC2 EE2 EB1 FB1	GitCommandError
TCI_1.11	EI1 FI1 N2 CL1 CR1 CC1 CM1 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica status ok
TCI_1.12	EI1 FI1 N2 CL1 CR1 CC1 CM2 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica status ok
TCI_1.13	EI1 FI1 N2 CL1 CR1 CC1 CM3 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica status ok e dir. con id commit creata e vuota

Test Case	Combinazione	Oracolo
TCI_1.14	EI1 FI1 N2 CL1 CR1 CC1 CM4 ED1 EM1 EC2 EE2 EB1 FB1	CHECKS.txt indica status ok e dir. con id commit contiene java file modificati e già esistenti
TCI_1.15	FB3	TypeError
TCI_1.16	FB2	OSError
TCI_1.17	FB1 EB2	FileNotFoundError

Tabella 2.29: Test case per il modulo Repo Mining

CAPITOLO 3

Test di regressione

3.1 Approccio

Per il test di regressione, abbiamo eseguito nuovamente tutti i test a seguito di ogni Change Request (CR). Questo approccio ci ha permesso di correggere eventuali errori introdotti dalle modifiche e di adeguare i test in base al lavoro svolto. Tale procedura è stata applicata a tutte le change request che impattavano l'intero sistema. Mentre per la CR3 abbiamo rieseguito esclusivamente i test relativi al modulo RepoMining, per la CR6 solo i test relativi al modulo ASA, e per le CR5 e CR7 che hanno introdotto nuovi moduli, non è stato rieseguito alcun test.

3.2 Change Request 1-2

L'esecuzione del test di regressione in seguito alla realizzazione delle change request 1 e 2 non ha evidenziato alcun fallimento, anzi i test che prima fallivano ora vengono eseguiti con successo.

3.3 Change Request 3

In seguito alla gestione degli errori e all'introduzione del sistema di log, sono stati registrati degli insuccessi per i test i cui oracoli prevedevano il lancio di eccezioni. È stato necessario, inoltre, introdurre e/o adattare alcuni test per la presenza dei file di log.

3.3.1 Unit testing

Repo_Mining

Per la funzione **initialize_repo_mining**, la gestione del fallimento dovuto all'assenza della directory mining_results ha reso necessario modificare l'oracolo del test case corrispondente a tale scelta.

Test Case	Combinazione	Oracolo
TC_1.2_3	FM1 ED1 FD1 ND3 EM2 ER1	Il dizionario data contiene i record del dataset e la directory è creata

Tabella 3.1: Modifica oracolo per la funzione initialize_repo_mining.

La gestione delle eccezioni MissingSchema, ConnectionError e GitCommandError durante il processo di mining ha portato alla modifica dell'oracolo dei relativi test case per la funzione **start_mining_repo**:

Test Case	Combinazione	Oracolo
TC_1.3_3	FD1 N3 CL3 FW1 EW1 FR1 ER1 ERI1 ERC1 EL1 SL1 FB1 EB1	CHECK.txt indica: Invalid Url
TC_1.3_4	FD1 N3 CL2 FW1 EW1 FR1 ER1 ERI1 ERC1 EL1 SL1 FB1 EB1	CHECK.txt indica: ConnectionError
TC_1.3_6	FD1 N3 CL1 CR1 CC4 FW1 EW1 FR1 ER1 ERI1 ERC1 EL1 SL1 FB1 EB1	CHECK.txt indica: GitCommandError

Tabella 3.2: Test case modificati per gestione errori

Inoltre, l'introduzione del file di log ha portato all'unificazione dei file CHECK.txt e ERRORS.txt. Ciò ha comportato una modifica degli oggetti dell'ambiente considerati e le relative categorie, sostituendo i due file con il file di log. Si considera, nello specifico, la seguente categoria:

- **Oggetto dell'ambiente:** file di log
  - Categoria: possibilità di scrittura (SL)
    - Scelta 1: Si (SL1)
    - Scelta 2: No (SL2) [error]

Sono stati adattati, di conseguenza, anche gli oracoli in cui tali file erano coinvolti, andando anche ad unire i test che differivano per specifiche caratteristiche dei due file.

Test Case	Combinazione	Oracolo
TC_1.3_1	FD1 N1 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log è vuoto
TC_1.3_3	FD1 N3 CL3 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica error: Invalid Url
TC_1.3_4	FD1 N3 CL2 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica error: ConnectionError
TC_1.3_5	FD1 N3 CL1 CR2 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica errore: la repo non è disponibile
TC_1.3_6	FD1 N3 CL1 CR1 CC2 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica errore: GitCommandError
TC_1.3_7	FD1 N3 CL1 CR1 CC3 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica errore: il commit non è esistente
TC_1.3_8	FD1 N3 CL1 CR1 CC4 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log indica errore: il commit non è definito
TC_1.3_9	FD1 N3 CL1 CR1 CC1 FW1 EW1 FR1 ER1 ERI1 ERC1 SL1 FB1 EB1	log informa: status ok
TC_1.3_15	SL2	PermissionError

Tabella 3.3: Test case con modifiche per log in start_mining_repo

3.3.2 Integration testing

Repo Mining

Il controllo della formattazione del dataset durante l'inizializzazione e la gestione delle eccezioni MissingSchema, ConnectionError e GitCommandError durante il processo di mining ha portato alla modifica dell'oracolo dei relativi test case:

TCI_1.3	EI1 FD2	ValueError
Test Case	**Combinazione**	**Oracolo**
TCI_1.5	EI1 FI1 N2 CL3 ED1 EM1 EC2 EE2 FB1 EB1	CHECK.txt indica Invalid Url
TCI_1.6	EI1 FI1 N2 CL2 ED1 EM1 EC2 EE2 FB1 EB1	CHECK.txt indica Connection Error
TCI_1.10	EI1 FI1 N2 CL1 CR1 CC2 ED1 EM1 EC2 EE2 FB1 EB1	CHECKS.txt e ERRORS.txt indicano GitCommand Error

Tabella 3.4: Test case modificati per gestione errori

L'introduzione del file di log ha portato all'unificazione dei file CHECK.txt e ERRORS.txt. Ciò ha comportato una modifica degli oggetti dell'ambiente considerati e le relative categorie, sostituendo i due file con il file di log. Si considera, nello specifico, la seguente categoria:

- **Oggetto dell'ambiente:** file di log
  - Categoria: esistenza (EL)
    - Scelta 1: esiste (EL1)
    - Scelta 2: non esiste (EL2)

Sono stati adattati, di conseguenza, anche gli oracoli in cui tali file erano coinvolti, andando anche ad unire i test che differivano per specifiche caratteristiche dei due file.

Test Case	**Combinazione**	**Oracolo**
TCI_1.1	EI2	FileNotFoundError
TCI_1.2	EM2	FileNotFoundError
TCI_1.3	EI1 FD2	ValueError
TCI_1.4	EI1 FI1 N1 ED2 EM1 EL2 EB1 FB1	Dataset_Divided creata e contiene 1.csv e mining_results contentente dir repo vuota
TCI_1.5	EI1 FI1 N2 CL3 EB1 FB1	log indica errore: MissingSchema
TCI_1.6	EI1 FI1 N2 CL2 EB1 FB1	log indica errore: ConnectionError
TCI_1.7	EI1 FI1 N3 CL1 CR2 ED1 EM1 EL2 EB1 FB1	log indica errore: la repo non è disponibile

Test Case	Combinazione	Oracolo
TCI_1.8	EI1 FI1 N2 CL1 CR1 CC3 ED1 EM1 EL2 EB1 FB1	log indica errore: il commit non è esistente
TCI_1.9	EI1 FI1 N2 CL1 CR1 CC4 ED1 EM1 EL1 EB1 FB1	log indica errore: il commit non è definito
TCI_1.10	EI1 FI1 N2 CL1 CR1 CC2 ED1 EM1 EL2 EB1 FB1	log indica errore: GitCommandError
TCI_1.11	EI1 FI1 N2 CL1 CR1 CC1 CM1 ED1 EM1 EL2 EB1 FB1	log informa: status ok
TCI_1.12	EI1 FI1 N2 CL1 CR1 CC1 CM2 ED1 EM1 EL2 EB1 FB1	log informa: status ok
TCI_1.13	EI1 FI1 N2 CL1 CR1 CC1 CM3 ED1 EM1 EL2 EB1 FB1	log informa: status ok e dir. con id commit creata e vuota
TCI_1.14	EI1 FI1 N2 CL1 CR1 CC1 CM4 ED1 EM1 EL2 EB1 FB1	log informa: status ok e dir. con id commit contiene java file modificati e già esistenti
TCI_1.15	FB3	TypeError
TCI_1.16	FB2	OSError
TCI_1.17	FB1 EB2	FileNotFoundError

Tabella 3.5: Test case per il modulo Repo Mining

La gestione del fallimento dovuto all'assenza della directory mining_results ha reso necessario modificare l'oracolo del test case corrispondente a tale scelta.

Test Case	Combinazione	Oracolo
TCI_1.2	EI1 FI1 N2 CL1 CC3 ED1 EM2 EL1 FB1 EB1	log indica errore: repo non esistente

Tabella 3.6: Modifica oracolo test integrazione

3.4 Change Request 4

L'esecuzione del test di regressione ha visto il fallimento di tutti i test, sia di unità che ti integrazione, a causa della trasformazione del sistema basato su script in un sistema object-oriented. La trasformazione del sistema ha comportato, in primis, una modifica generale alla struttura dei test: è stato necessario creare un'istanza della classe interessata e poi invocare i metodi da testare, invece di chiamare i metodi main dei vari script. Detto ciò, le modifiche maggiori hanno riguardato il testing di unità.

3.4.1 Unit testing

Repo_Mining

I fallimenti relativi allo script divide_dataset.py sono stati risolti sostituendo l'esecuzione di tale script con l'istanziazione della classe DatasetDivider e l'invocazione del metodo **divide_dataset**.

In merito allo script repo_mining.py, per la funzione initialize, è stato necessario istanziare un oggetto della classe RepoMiner, per poi invocare la funzione **initialize_repo_mining**.
Lo stesso è avvenuto per la funzione **start_mining_repo**. È stato inoltre necessario adattare i test la modifica della firma della funzione, con la rimozione del parametro cwd (ovvero del path corrente). Ciò ha comportato anche l'eliminazione dei test corrispondenti alle scelte relative a questo. L'incapsulamento, poi, della logica di estrazione nella funzione **analyze** commit ha reso superflua la categoria "contenuto modifica" relativa alla variabile *data*, che è stata invece presa in considerazione nel test di unità della funzione start_mining_repo.
Sono state quindi considerate le seguenti variabili:

- data, ovvero il parametro contenente i riferimenti ai commit da analizzare.
  - Categoria: formato (FD)
    - Scelta 1: formato valido (FD1) [property dataFormatOk]
    - Scelta 2: formato non valido (FD2) [error]
  - Categoria: numero record (N)
    - Scelta 1: zero (N1)
    - Scelta 2: uno (N2) [property dataNotEmpty]
    - Scelta 3: maggiore di uno (N3) [property dataNotEmpty]
  - Categoria: contenuto link (CL)

        * Scelta 1: link esistente e valido (CL1) [if dataFormatOk and dataNotEmpty] [property linkValidOk]
        * Scelta 2: link valido e non esistente (CL2) [if dataFormatOk and dataNotEmpty] [error]
        * Scelta 3: link non valido (CL3) [if dataFormatOk and dataNotEmpty] [error]
    - Categoria: contenuto repo (CR)
        * Scelta 1: presente repo valida (CR1) [if linkValidOk] [property repoValidOk]
        * Scelta 2: presente repo non valida (CR2) [if linkValidOk]
    - Categoria: Contenuto commit (CC)
        * Scelta 1: Presente commit valido (CC1) [if repoValidOk] [property commitValidOk]
        * Scelta 2: Presente commit valido con path che eccede la lunghezza massima del S.O. (CC2) [if repoValidOk]
        * Scelta 3: Presente commit non valido con risposta non riuscita (CC3) [if repoValidOk]
        * Scelta 4: Presente commit non valido con risposta riuscita (CC4) [if repoValidOk]

- reponame, ovvero il parametro contenente il nome del repository.
    - Categoria: formato (FR)
        * Scelta 1: stringa con formato valido (FR1) [property reponameFormatOk]
        * Scelta 2: stringa con formato non valido (FR2) [error]
        * Scelta 3: non stringa (FR3) [error]
    - Categoria: esistenza repository (ER)
        * Scelta 1: esiste (ER1) [if reponameFormatOk]
        * Scelta 2: non esiste (ER2) [if reponameFormatOk] [error]

- Repo Cvd_id
    - Categoria: esistenza (ERI)
        * Scelta 1: esiste (ERI1)
        * Scelta 2: non esiste (ERI2)

- Repo Commit

  - Categoria: esistenza (ERC)
    - Scelta 1: esiste (ERC1) [single]
    - Scelta 2: non esiste (ERC2)

- CHECK.txt, ovvero il file che contiene i log dell'attività di mining.
  - Categoria: esistenza (EC)
    - Scelta 1: esiste (EC1)
    - Scelta 2: non esiste (EC2) []
  - Categoria: possibilità di scrittura (SC)
    - Scelta 1: sì (SC1)
    - Scelta 2: no (SC2)

- ERROR.txt, ovvero il file che contiene le informazioni sugli errori.
  - Categoria: esistenza (EE)
    - Scelta 1: esiste (EE1)
    - Scelta 2: non esiste (EE2)
  - Categoria: possibilità di scrittura (SE)
    - Scelta 1: sì (SE1)
    - Scelta 2: no (SE2)

È stata considerata, quindi, la seguente test-suite:

Test Case	**Combinazione**	**Oracolo**
TC_1.3_1	FD1 N1 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECK.txt vuoto
TC_1.3_2	FD2	KeyError
TC_1.3_3	FD1 N3 CL3	MissingSchema
TC_1.3_4	FD1 N3 CL2	ConnectionError
TC_1.3_5	FD1 N3 CL1 CR2 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica che la repo non è disponibile

TC_1.3_6	FD1 N3 CL1 CR1 CC2 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	GitCommandError
TC_1.3_7	FD1 N3 CL1 CR1 CC3 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica che il commit non è esistente
TC_1.3_8	FD1 N3 CL1 CR1 CC4 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt e ERRORS.txt indicano che il commit non è definito
TC_1.3_9	FD1 N3 CL1 CR1 CC1 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica status ok
TC_1.3_10	FR3	TypeError
TC_1.3_11	FR2	OSError
TC_1.3_12	FR1 ER2	FileNotFoundError
TC_1.3_13	EC1 SC2	PermissionError
TC_1.3_14	EE1 SE2	PermissionError

Tabella 3.7: Test case per la funzione start_mining_repo

Text_Mining

Per quanto riguarda i test di unità del modulo Text_Mining, la modifica ha comportato una riorganizzazione significativa che ha comportato il fallimento della totalità dei test. Inizialmente, il testing del modulo aveva portato allo sviluppo di quattro file di test separati, ciascuno dedicato a uno specifico script e ai relativi metodi. Tuttavia, con l'introduzione delle classi, tre di questi script sono stati unificati in un'unica classe, ovvero JavaTextMining. Di conseguenza, anche i test sono stati accorpati in un singolo file, TestJavaTextMining, che va a testare tutti i metodi della nuova classe. In particolare lo script dict_generator.py, diventato un metodo della nuova classe, non si occupa più di navigare tra le directory, ma solo di eseguire il suo compito principale, utilizzando adesso dei parametri dati in input al metodo. I suoi test sono, quindi, cambiati profondamente rispetto a quelli effettuati in test_dict_generator.py. Per quanto riguarda la funzione **mergeDict**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: dict1
  - Categoria: Contenuto variabile (CV)
    - Scelta 1: Non è un dizionario (CV1) [error]
    - Scelta 2: è un dizionario non vuoto (CV2) [property dict1Ok]
    - Scelta 3: è un dizionario vuoto (CV3)
- **Variabile 2**: dict2
  - Categoria: Contenuto variabile (CVA)
    - Scelta 1: Non è un dizionario (CVA1)
    - Scelta 2: è un dizionario non vuoto (CVA2) [property dict2Ok]
    - Scelta 3: è un dizionario vuoto (CVA3)
  - Categoria: Valori chiavi (VC)
    - Scelta 1: Non ha chiavi in comune con dict1 (VC1) [if dict1Ok and dict2Ok]
    - Scelta 2: Ha chiavi in comune con dict1 (VC2) [if dict1Ok and dict2Ok]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_2.6_1	CV1 CVA1	TypeError
TC_2.6_2	CV1 CVA2	TypeError
TC_2.6_3	CV1 CVA3	Restitusice il valore del primo parametro
TC_2.6_4	CV3 CVA1	TypeError
TC_2.6_5	CV3 CVA2	Restituise il valore del secondo parametro
TC_2.6_6	CV3 CVA3	Restituisce un dizionario vuoto
TC_2.6_7	CV2 CVA1	TypeError
TC_2.6_8	CV2 CVA2 VC1	Restituise l'unione dei due dizionari
TC_2.6_9	CV2 CVA2 VC2	Restituise l'unione dei due dizionari sommando le frequenze delle parole in comune
TC_2.6_10	CV2 CVA3	Restituisce il valore del primo dizionario

Tabella 3.8: Test case per la funzione mergeDict

In precedenza ogni script disponeva del proprio metodo main, per navigare tra le directory ed eseguire i vari calcoli, adesso esiste un unico main. Di conseguenza, i test relativi ai singoli

main sono stati unificati in una sola classe di test TestRunTextMining all'interno del file TestMain, che è stata implementata seguendo l'impostazione del testing del metodo main() di text_mining.py in test_text_mining.py, con l'aggiunta di un solo test per verificare gli effetti dell'assenza della directory Dataset_Divided, utilizzata ora per recuperare il numero di directory in cui sono salvati i file java.

Lo script che creava il csv con i risultati dell'eleborazione del modulo, è diventata una classe a sè stante, CSVWriter, con due nuovi metodi, che si occupano unicamente di creare il csv dai dizionari dati in input. Ciò ha portato ad una modifica dei test, che non si dovranno occupare più di testare lo spostamento all'interno della directory per il ritrovamento dei file necessari per creare il csv. I nuovi test quindi estraggono solo quelli più rilevanti dai precedenti e li ampliano vista la presenza di nuove variabili in input ai metodi della classe. Andiamo a vederli nel dettaglio:

Per quanto riguarda la funzione **write_header**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: filtered_dict
  - Categoria: Contenuto variabile (CV)
    - Scelta 1: Non è un dizionario (CV1) [error]
    - Scelta 2: è un dizionario non vuoto (CV2)
    - Scelta 3: è un dizionario vuoto (CV3)
- **Variabile 2**: mining_dict
  - Categoria: Contenuto variabile (CVA)
    - Scelta 1: Non è un dizionario (CVA1)
    - Scelta 2: è un dizionario non vuoto (CVA2)
    - Scelta 3: è un dizionario vuoto (CVA3)
- **Variabile 3**: csv_mining_final.csv
  - Categoria: Contenuto del codice (CN)
    - Scelta 1: Accessibile (CN1)
    - Scelta 2: Non accessibile (CN2) [single]
    - Scelta 3: Accessibile ma non si può scrivere all'interno (CN3) [single]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_2.10_1	CV1	AttributeError
TC_2.10_2	CV2 CVA2 CN1	L'header viene scritto correttamente (con Name e le chiavi)
TC_2.10_3	CV2 CVA3 CN1	L'header viene scritto correttamente (con Name e le chiavi)
TC_2.10_4	CV2 CVA1 CN1	L'header viene scritto correttamente (con Name e le chiavi)
TC_2.10_5	CV3 CVA2 CN1	L'header viene scritto correttamente (solo colonna Name)
TC_2.10_6	CV3 CVA3 CN1	L'header viene scritto correttamente (solo colonna Name)
TC_2.10_7	CV3 CVA1 CN1	L'header viene scritto correttamente (solo colonna Name)
TC_2.10_8	CV3 CVA3 CN2	PermissionError
TC_2.10_9	CV3 CVA3 CN3	IOError

Tabella 3.9: Test case per la funzione write_header

Per quanto riguarda la funzione **write_rows**, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: filtered_dict
  - Categoria: Contenuto variabile (CV)
    - Scelta 1: è un dizionario non vuoto (CV1)
    - Scelta 2: è un dizionario non vuoto con chiavi in camelcase (CV2) [single]
    - Scelta 3: è un dizionario vuoto (CV3)
- **Variabile 2**: mining_dict
  - Categoria: Contenuto variabile (CVA)
    - Scelta 1: Non è un dizionario (CVA1) [error]
    - Scelta 2: è un dizionario non vuoto (CVA2) [property dictOk]
    - Scelta 3: è un dizionario non formattato correttamente (CVA3) [error]
    - Scelta 4: è un dizionario vuoto (CVA4)
  - Categoria: Contenuto java (CJ)
    - Scelta 1: Non è un dizionario (CJ1) [if dictOk]
    - Scelta 2: è un dizionario non vuoto (CJ2) [if dictOk]
    - Scelta 3: è un dizionario vuoto (CJ3) [if dictOk]

- **Variabile 3**: csv_mining_final.csv
  - Categoria: Contenuto del codice (CN)
    - Scelta 1: Accessibile (CN1)
    - Scelta 1: Non accessibile (CN2) [error]
    - Scelta 2: Accessibile ma non si può scrivere all'interno (CN3) [error]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_2.11_1	CV2 CVA2 CJ2 CN1	La riga viene scritto col nome del file e tutti 0
TC_2.11_2	CV1 CVA2 CJ2 CN1	La riga viene scritta col nome del file e le relative frequenze delle chiavi
TC_2.11_3	CV1 CVA2 CJ3 CN1	TypeError
TC_2.11_4	CV1 CVA2 CJ1 CN1	La riga viene scritta col nome del file e tutti 0
TC_2.11_5	CV1 CVA3 CN1	TypeError
TC_2.11_6	CV1 CVA4 CN1	La riga non viene scritta
TC_2.11_7	CV1 CVA1 CN1	AttributeError
TC_2.11_8	CV3 CVA2 CJ2 CN1	La riga non viene scritta
TC_2.11_9	CV3 CVA2 CJ3 CN1	TypeError
TC_2.11_10	CV3 CVA2 CJ1 CN1	La riga viene scritta con solo il nome del file
TC_2.11_11	CV3 CVA3 CN1	TypeError
TC_2.11_12	CV3 CVA4 CN1	La riga non viene scritta
TC_2.11_13	CV3 CVA1 CN1	AttributeError
TC_2.11_14	CV3 CVA2 CJ2 CN2	PermissionError
TC_2.11_15	CV3 CVA2 CJ2 CN3	IOError

Tabella 3.10: Test case per la funzione write_rows

Per questo modulo, per verificare di non aver introdotto errori o alterato il comportamento del sistema, come test di regressione abbiamo effettuato anche un confronto tra il file CSV finale generato dopo la modifica e quello generato eseguendo il sistema nella sua versione originale. I due file CSV risultano identici, confermando così che la modifica non ha avuto impatti indesiderati sul funzionamento del sistema. Il confronto si trova nella classe di testing TestRegression nella directory tests/Test_Mining.

Union

Il fallimento dei test è stato causato dalla trasformazione degli script Union_TM_with_ASA.py, Union_SMwithASA.py, Union.py e TotalCombination.py in classi.
In particolare, con l'utilizzo della libreria *pandas*, è stato possibile attribuire i compiti dei quattro script, ad una singola classe: DatasetCombiner, che mette a disposizione il metodo merge. Per quest'ultima, abbiamo considerato le seguenti categorie di variabili:

- **Variabile 1**: csv_files:
  - Categoria: Numero di file (E)
    - Scelta 1: Meno di due file [error]
    - Scelta 2: Due o più file
  - Categoria: Formato dei file (F)
    - Scelta 1: File non formattato correttamente [error]
    - Scelta 2: File formattato correttamente
  - Categoria: Formato delle colonne (H)
    - Scelta 1: Presenza della colonna 'Name'
    - Scelta 2: Assenza della colonna 'Name'[error]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_4_1	E1	ValueError
TC_4_2	E2 F1	KeyError
TC_4_3	E2 F2 H1	Combinazione dei file in corrispondenza della colonna *Name*
TC_4_4	E2 F2 H2	KeyError

Tabella 3.11: Test case per la funzione merge

ASA

La modifica ha comportato la trasformazione degli script rules_dict_generator_ASA.py e ASA_vulnerability_dict_generator.py in classi. I compiti dei due script sono stati rimpiazzati rispettivamente dalle funzioni **generate_rules_dict** e **generate_vulnerability_dict**

della classe DictGenerator. Questo cambio di approccio, ha reso necessario modificare i test precedentemente implementati.

Per quanto riguarda la funzione **generate_rules_dict**, abbiamo considerato le seguenti categorie di variabili:

- RepositoryMining_ASAResults_NEG, ovvero il file contenente i risultati dell'analisi statica, relativi alle repository considerate di classe negativa.
  - Categoria: esistenza del file (EN)
    - Scelta 1: il file esiste (EN1) [property negExistOk]
    - Scelta 2: il file non esiste (EN2) [single]
  - Categoria: numero record (NN)
    - Scelta 1: zero (NN1) [if negExistOk]
    - Scelta 2: uno (NN2) [if negExistOk] [property negNotEmpty]
    - Scelta 3: maggiore di uno (NN3) [if negExistOk] [property negNotEmpty] [single]
  - Categoria: formato contenuto (FCN)
    - Scelta 1: formato valido (FCN1) [if negExistOk] [property negFormatOk]
    - Scelta 2: formato non valido (FCN2) [if negExistOk] [error]
  - Categoria: tipologia contenuto (TN)
    - Scelta 1: vulnerabilità presente (TCN1) [if negFormatOk and negNotEmpty]
    - Scelta 2: vulnerabilità assente (TCN2) [if negFormatOk and negNotEmpty]
- RepositoryMining_ASAResults_POS, ovvero il file contenente i risultati dell'analisi statica, relativi alle repository considerate di classe positiva.
  - Categoria: esistenza del file (EP)
    - Scelta 1: il file esiste (EP1) [property posExistOk]
    - Scelta 2: il file non esiste (EP2)
    - Scelta 1: zero (NP1) [if posExistOk]
    - Scelta 2: uno (NP2) [if posExistOk]
    - Scelta 3: maggiore di uno (NP3) [if posExistOk] [single]
  - Categoria: formato contenuto (FCP)

        * Scelta 1: formato valido (FCP1) [if posExistOk] [property posFormatOk]
        * Scelta 2: formato non valido (FCP2) [if posExistOk] [error]
    - Categoria: tipologia contenuto (TP)
        * Scelta 1: vulnerabilità presente (TCP1) [if posNotEmpty and posFormatOk]
        * Scelta 2: vulnerabilità assente (TCP2) [if posNotEmpty and posFormatOk]
    - Categoria: numero record (NP)

- ASA_dict.csv, ovvero il file contenente le componenti a cui è associata una vulnerabilità, con la relativa regola e classificazione.
    - Categoria: possibilità di scrittura (S)
        * Scelta 1: è possibile scrivere nel file (S1)
        * Scelta 2: non è possibile scrivere nel file (S2) [error]
    - Categoria: possibilità di scrittura (S)
        * Scelta 1: è possibile scrivere nel file (S1)
        * Scelta 2: non è possibile scrivere nel file (S2) [error]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.2_1	EN2 EP1 NP3 FCP1 TCP1 S1	ASA_rules_dict con solo regole relative a vulnerabilità di classe 'pos'
TC_3.2_2	EN1 NN3 FCN1 TCN1 EP2 S1	ASA_rules_dict con solo regole relative a vulnerabilità di classe 'neg'
TC_3.2_3	S2	Permission Error
TC_3.2_4	EN1 FCN2	Index Error
TC_3.2_5	EP1 FCP2	Index Error
TC_3.2_6	EN1 NN2 FCN1 TCN2 EP1 NP2 FCP1 TCP2 S1	ASA_rules_dict vuoto
TC_3.2_7	EN1 NN2 FCN1 TCN1 EP1 NP2 FCP1 TCP2 S1	ASA_rules_dict con regole relative a vulnerabilità di classe 'pos' e 'neg'
TC_3.2_8	EN1 NN1 FCN1 EP1 NP1 FCP1 S1	ASA_rules_dict vuoto

Tabella 3.12: Test case per la funzione generate_rules_dict

Per quanto riguarda la funzione **generate_vulnerability_dict**, abbiamo considerato le seguenti categorie di variabili:

- RepositoryMining_ASAResults_NEG, ovvero il file contenente i risultati dell'analisi statica, relativi alle repository considerate di classe negativa.
  - Categoria: esistenza del file (EN)
    - Scelta 1: il file esiste (EN1) [property negExistOk]
    - Scelta 2: il file non esiste (EN2) [single]
  - Categoria: numero record (NN)
    - Scelta 1: zero (NN1) [if negExistOk]
    - Scelta 2: uno (NN2) [if negExistOk] [property negNotEmpty]
    - Scelta 3: maggiore di uno (NN3) [if negExistOk] [property negNotEmpty] [single]

  - Categoria: formato contenuto (FCN)
    * Scelta 1: formato valido (FCN1) [if negExistOk] [property negFormatOk]
    * Scelta 2: formato non valido (FCN2) [if negExistOk] [error]
  - Categoria: tipologia contenuto (TN)
    * Scelta 1: vulnerabilità presente (TCN1) [if negFormatOk and negNotEmpty]
    * Scelta 2: vulnerabilità assente (TCN2) [if negFormatOk and negNotEmpty]

- RepositoryMining_ASAResults_POS, ovvero il file contenente i risultati dell'analisi statica, relativi alle repository considerate di classe positiva.
  - Categoria: esistenza del file (EP)
    * Scelta 1: il file esiste (EP1) [property posExistOk]
    * Scelta 2: il file non esiste (EP2)
    * Scelta 1: zero (NP1) [if posExistOk]
    * Scelta 2: uno (NP2) [if posExistOk]
    * Scelta 3: maggiore di uno (NP3) [if posExistOk] [single]
  - Categoria: formato contenuto (FCP)
    * Scelta 1: formato valido (FCP1) [if posExistOk] [property posFormatOk]
    * Scelta 2: formato non valido (FCP2) [if posExistOk] [error]
  - Categoria: tipologia contenuto (TP)
    * Scelta 1: vulnerabilità presente (TCP1) [if posNotEmpty and posFormatOk]
    * Scelta 2: vulnerabilità assente (TCP2) [if posNotEmpty and posFormatOk]
  - Categoria: numero record (NP)

- ASA_dict.csv, ovvero il file contenente le componenti a cui è associata una vulnerabilità, con la relativa regola e classificazione.
  - Categoria: possibilità di scrittura (S)
    * Scelta 1: è possibile scrivere nel file (S1)
    * Scelta 2: non è possibile scrivere nel file (S2) [error]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.1_1	EN2 EP1 NP3 FCP1 TCP1 S1	ASA_dict.csv con solo elementi con classe 'pos' e tipo 'vulnerability'
TC_3.1_2	EN1 NN3 FCN1 TCN1 EP2 S1	ASA_dict.csv con solo elementi con classe 'neg' e tipo 'vulnerability'
TC_3.1_3	S2	Permission Error
TC_3.1_4	EN1 FCN2	Index Error
TC_3.1_5	EP1 FCP2	Index Error
TC_3.1_6	EN1 NN2 FCN1 TCN2 EP1 FCP1 NP2 TCP2 S1	ASA_dict.csv vuoto
TC_3.1_7	EN1 NN2 FCN1 TCN1 EP1 FCP1 NP2 TCP1 S1	ASA_dict.csv con elementi di classe 'pos' ed elementi di classe 'neg' e tipo 'vulnerability'
TC_3.1_8	EN1 NN1 FCN1 EP1 NP1 FNP FCP1 S1	ASA_dict.csv vuoto

Tabella 3.13: Test case per la funzione generate_vulnerability_dict

È stata attuata anche la trasformazione dello script creator_csv_for_ASA.py nella classe CreatorCsvForASA. Questo cambio di approccio, ha reso necessario modificare i test precedentemente implementati, in quanto hanno registrato fallimenti in seguito alla modifica. Per quanto riguarda la funzione **process_vulnerabilities**, abbiamo considerato le seguenti categorie di variabili:

- **Oggetto dell'ambiente 1**: vulnerabilities:
  - Categoria: Esistenza (E)
    - Scelta 1: La lista non esiste [error]
    - Scelta 2: La lista esiste
  - Categoria: Formato(F)
    - Scelta 1: formato corretto

        * Scelta 2: formato non corretto [error]
    - Categoria: Lunghezza(L)
        * Scelta 1: La lista è vuota
        * Scelta 2: La lista non è vuota

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.3_1	E1	TypeError
TC_3.3_2	E2 F2	TypeError
TC_3.3_3	E2 F1 L1	Dizionario vuoto
TC_3.3_4	E2 F1 L2	Dizionari con il numero di occorrenze di ogni rule java, per ogni progetto

Tabella 3.14: Test case per la funzione process_vulnerabilities

Per quanto riguarda la funzione **write_final_csv**, abbiamo considerato le seguenti categorie di variabili:

- **Oggetto dell'ambiente 1**: final_csv:
    - Categoria: Esistenza (E)
        * Scelta 1: Il file non esiste [error]
        * Scelta 2: Il file esiste
- **Oggetto dell'ambiente 2**: big_dict:
    - Categoria: Contenuto (CBD)
        * Scelta 1: Il dizionario è vuoto
        * Scelta 2: Il dizionario non è vuoto
- **Oggetto dell'ambiente 3**: rules_dict:
    - Categoria: Contenuto (CRD)
        * Scelta 1: Il dizionario è vuoto
        * Scelta 2: Il dizionario non è vuoto

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.4_1	E1	FileNotFoundError
TC_3.4_2	E2 CBD1 CRD1	Il file di output non contiene nessuna vulnerabilità
TC_3.4_3	E2 CBD1 CRD2	Il file di output contiene le regole come intestazione
TC_3.4_4	E2 CBD2 CRD1	Il file di output non contiene nessuna vulnerabilità
TC_3.4_5	E2 CBD2 CRD2	Il file di output contiene le vulnerabilità e i progetti di big_dict in corrispondenza delle java rules presenti in rules_dict

Tabella 3.15: Test case per la funzione write_final_csv

3.5 Change Request 6

L'esecuzione del test di regressione ha evidenziato il fallimento dei test di unità relativi alla classe `DictGenerator`, a causa dell'introduzione della classe `SonarAnalyzer`. Quest'ultima, dopo aver completato il processo di analisi statica, genera un singolo file CSV come output, il quale viene utilizzato come input per la classe `DictGenerator`. Tuttavia, la versione di `DictGenerator` implementata con la CR4 prevedeva la generazione di due file CSV separati come output. Con l'introduzione della CR6, la classe `DictGenerator` è stata modificata per accettare in input un singolo file CSV. Di conseguenza, i test di unità associati sono stati aggiornati per riflettere il cambiamento nella gestione degli input e garantire la compatibilità con la nuova struttura di output della classe `SonarAnalyzer`.

DictGenerator

Per quanto riguarda la funzione **generate_rules_dict**, abbiamo considerato le seguenti categorie di variabili:

- **Oggetto dell'ambiente 1**: file:
  - Categoria: Esistenza (E)
    - Scelta 1: Il file non esiste [error]
    - Scelta 2: Il file esiste
  - Categoria: Numero di record(NR)

    - Scelta 1: Il file è vuoto
    - Scelta 2: Il file contiene 1 record [property headerCSV]
    - Scelta 3: Il file contiene più di un record [property dataCSV]
  - Categoria: Lunghezza(L)
    - Scelta 1: Il file contiene meno di 9 colonne [if headerCSV or dataCSV]
    - Scelta 2: Il file contiene esattamente 9 colonne [if headerCSV or dataCSV][single]
    - Scelta 3: Il file contiene più 9 colonne [if headerCSV or dataCSV]
  - Categoria: Contenuto(C)
    - Scelta 1: il file contiene vulnerabilità [if dataCSV]
    - Scelta 2: il file non contiene vulnerabilità [if dataCSV]

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.2_1	E1	FileNotFoundError
TC_3.2_2	E1 NR1	Il dizionario delle rules risulterà vuoto
TC_3.2_3	E1 NR2 L1	Il dizionario delle rules risulterà vuoto
TC_3.2_4	E1 NR2 L2	Il dizionario delle rules risulterà vuoto
TC_3.2_5	E1 NR2 L3	Il dizionario delle rules risulterà vuoto
TC_3.2_6	E1 NR3 L1 C1	IndexError
TC_3.2_7	E1 NR3 L1 C2	IndexError
TC_3.2_8	E1 NR3 L2 C1	Dizionario delle rules contenute in file
TC_3.2_9	E1 NR3 L2 C2	Il dizionario delle rules risulterà vuoto
TC_3.2_10	E1 NR3 L3 C1	Dizionario delle rules contenute in file
TC_3.2_11	E1 NR3 L3 C2	Il dizionario delle rules risulterà vuoto

Tabella 3.16: Test case per la funzione generate_rules_dict

Per quanto riguarda la funzione **generate_vulnerability_dict**, abbiamo considerato le seguenti categorie di variabili:

- **Oggetto dell'ambiente 1**: file:
  - Categoria: Esistenza (E)
    - Scelta 1: Il file non esiste [error]

    - Scelta 2: Il file esiste
  - Categoria: Numero di record(NR)
    - Scelta 1: Il file è vuoto
    - Scelta 2: Il file contiene 1 record [property headerCSV][single]
    - Scelta 3: Il file contiene più di un record [property dataCSV]
  - Categoria: Lunghezza(L)
    - Scelta 1: Il file contiene meno di 10 colonne
    - Scelta 2: Il file contiene esattamente 10 colonne
    - Scelta 3: Il file contiene più 10 colonne [single]
  - Categoria: Contenuto issue "VULNERABILITY" (CV)
    - Scelta 1: Il file contiene vulnerabilità di tipo "VULNERABILITY"
    - Scelta 2: Il file non contiene vulnerabilità di tipo "VULNERABILITY" [property noVuln]
  - Categoria: Contenuto issue "NO_ISSUES_FOUND" (CN)
    - Scelta 1: Il file contiene vulnerabilità di tipo "NO_ISSUES_FOUND" [if noVuln]
    - Scelta 2: Il file non contiene vulnerabilità di tipo "NO_ISSUES_FOUND"

Dalle categorie e possibili scelte abbiamo ricavato la seguente test suite:

Test Case	Combinazione	Oracolo
TC_3.1_1	E1	FileNotFoundError
TC_3.1_2	E1 NR1	La lista di dizionari delle vulnerabilità risulterà vuota
TC_3.1_3	E1 NR2 L3	La lista di dizionari delle vulnerabilità risulterà vuota
TC_3.1_4	E1 NR3 L1 CV1 CN1	IndexError
TC_3.1_5	E1 NR3 L1 CV1 CN2	IndexError
TC_3.1_6	E1 NR3 L1 CV2 CN1	IndexError
TC_3.1_7	E1 NR3 L2 CV1 CN1	La lista conterrà un dizionario per ogni issue di tipo 'VULNERABIILTY' o 'NO_ISSUE_FOUND'
TC_3.1_8	E1 NR3 L2 CV1 CN2	La lista conterrà un dizionario per ogni issue di tipo 'VULNERABIILTY'
TC_3.1_9	E1 NR3 L2 CV2 CN1	La lista conterrà un dizionario per ogni issue di tipo 'NO_ISSUE_FOUND'

Tabella 3.17: Test case per la funzione generate_vulnerability_dict

3.6 Change Request 8

La Change Request 8 ha portato a delle modifiche ai test già presenti, soprattutto degli oracoli, in quanto la gestione differente dei path ha causato il fallimento di alcuni test.

3.6.1 Unit testing

Text Mining

Per quanto riguarda la funzione run_text_mining della classe Main un'unica categoria è cambiata, ovvero la prima individuata, che verifica l'esistenza o meno della directory /mining_results (DF), che, invece, adesso riguarda la directory /Dataset_Divided. Andiamo a vedere come sono cambiati i test case:

Test Case	Combinazione	Oracolo
TC_2.5_1	DF1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_2	DF2 ES1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_3	DF2 ES2 CN1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_4	DF2 ES2 CN2 STC1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_5	DF2 ES2 CN2 STC2	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_6	DF2 ES2 CN2 STC3 CNC1	Il file text_mining.txt non viene creato,

Test Case	Combinazione	Oracolo
		text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_7	DF2 ES2 CN2 STC3 CNC2 STF1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_8	DF2 ES2 CN2 STC3 CNC2 STF2	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_9	DF2 ES2 CN2 STC3 CNC2 STF3 CNF1	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_10	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV1	Vengono creati i file text_mining.txt, text_mining_dict.txt e FilteredTextMining.txt con i relativi dizionari e il csv con intestazione e una riga per ogni file java
TC_2.5_11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES1 SV1	Vengono creati i file text_mining.txt, text_mining_dict.txt e FilteredTextMining.txt con i relativi dizionari e il csv con intestazione e una riga per ogni file java
TC_2.5_12	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV2	PermissionError
TC_2.5_13	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV3	ValueError
TC_2.5_14	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC2	PermissionError
TC_2.5_15	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI2	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto,

Test Case	Combinazione	Oracolo
		il csv ha solo Name nell'intestazione
TC_2.5_16	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione
TC_2.5_17	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI4	Il file text_mining.txt non viene creato, text_mining_dict.txt e FilteredTextMining.txt contengono un dizionario vuoto, il csv ha solo Name nell'intestazione

Tabella 3.18: Test case modificati per la funzione run_text_mining()

Possiamo notare come alcuni errori precedentemente dovuti ai path, adesso non sono più presenti, ma semplicemente i file vengono creati vuoti, senza interrompere l'esecuzione del tool. Inoltre, essendo adesso un unico main unificato tramite la CR4, si occupa della creazione di tutti i file di interesse, dai text_mining fino al csv finale.

3.6.2 Integration testing

Text Mining

Anche in questo caso per la funzione run_text_mining alcuni oracoli sono differenti rispetto ai test precedenti. La categoria DF non fa più riferimento all'esistenza o meno della directory /mining_results ma /Dataset_Divided. Vediamo quali test case sono cambiati:

Test Case	Combinazione	Oracolo
TCI_2.2	DF2 ES1	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.4	DF2 ES2 CN2 STC1	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.7	DF2 ES2 CN2 STC3 CNC2 STF1	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI2	Il file di text mining non viene creato, gli altri .txt vengono creati con dizionari vuoti e il csv solo l'intestazione
TCI_2.19	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI4 CFJ2 ESTM1 ESD2 ESF2 ESC2	Il file text_mining.txt non viene creato, gli altri file vengono creati con dizionari vuoti e il csv solo l'intestazione

Tabella 3.19: Test case per il modulo Text_Mining

Anche in questo caso non vi sono più errori dovuti a path non esistenti e adesso in caso di file non .java i file di text_mining.txt non vengono più creati.

CAPITOLO 4

Risultati

Come metrica aggiuntiva, è stata calcolata la statement coverage relativa ai test effettuati, ottenendo una copertura generale pari all'94%. I risultati dettagliati sono riportati di seguito:

Coverage report: 94%

Files | Functions | Classes

coverage.py v7.6.4, created at 2024-11-15 17:00 +0100

File ▾	statements	missing	excluded	coverage
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Union\DatasetCombiner.py	21	2	0	90%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Text_Mining\JavaTextMining.py	64	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Text_Mining\CSVWriter.py	23	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Software_Metrics\SoftwareMetrics.py	362	25	0	93%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Software_Metrics\MetricsWriter.py	20	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\RepoMining\RepoMiner.py	111	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\RepoMining\DatasetDivider.py	27	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\mining_results_asa\SonarAnalyzer.py	127	11	0	91%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\mining_results_asa\DictGenerator.py	25	0	0	100%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\mining_results_asa\CsvCreatorForAsa.py	30	1	0	97%
C:\Users\Utente\PycharmProjects\Predicting-Vulnerable-Code\Dataset2\Main.py	245	113	0	54%
Total	**5997**	**361**	**0**	**94%**

Figura 4.1: Coverage report

Tali risultati confermano l'efficacia dell'approccio di testing black-box adottato. In particolare, la corretta selezione delle categorie e delle diverse opzioni ha contribuito significativamente all'esito positivo delle verifiche. Inoltre, è opportuno sottolineare l'importanza della fase preliminare di individuazione accurata degli oggetti dell'ambiente e dei relativi parametri.