UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Ingegneria, Gestione ed Evoluzione del Software

Pre-Modification Unit Test Plan

TEAM MEMBER

Donia Daniele - 0522501575

La Marca Antonio - 0522501557

Somma Pasquale - 0522501543

Indice

CAPITOLO 1

Introduzione

1.1 Approccio

Dopo aver compreso la struttura e il funzionamento del sistema, si è proceduto con la verifica delle funzionalità del sistema. Il tool non è accompagnato da alcuna test suite. È stato quindi progettato ed eseguito un test di unità preliminare, per individuare le esatte responsabilità delle varie componenti a livello di metodi. L'approccio individuato per la progettazione del testing è il category partition. A tal proposito, oltre ai parametri delle funzioni, non sempre presenti, sono stati considerati i diversi oggetti dell'ambiente con cui i metodi interagiscono. Nello specifico, sono stati considerati:

- il dataset iniziale, contenente l'insieme di commit da analizzare;
- i dataset ridotti in cui è diviso quello iniziale;
- i file ottenuti dall'attività di mining relativa ai commit
- i file di log dell'attività di mining
- il file esportato da Sonarqube contenente i risultati dell'analisi statica
- i dizionari delle regole e delle vulnerabilità estratti dai dati di SonarQube
- i file .csv ottenuti dai vari processi intermedi (Understand, text-mining e SonarQube)
- i file .csv risultanti dall'unione dei singoli file .csv dei risultati

CAPITOLO 2

Categorie e scelte

Di seguito sono riportate le categorie e le scelte individuate per gli oggetti dell'ambiente citati nella sezione precedente.

2.1 Repo Mining

Riguardo il mining delle repository, si è deciso di testare i seguenti script: divide_dataset.py e repo_Mining. Di seguito sono riportati le funzioni, i parametri e gli oggetti dell'ambiente considerati e le relative categorie per la definizione dei test case.

2.1.1 divide_dataset.py

Lo script non presenta alcuna funzione. Per agevolare l'attività di testing e il mocking delle componenti esterne, è creata quindi una funzione contenente il codice dello script. Inoltre, si considerano i seguenti oggetti dell'ambiente:

- **Oggetto dell'ambiente 1:** 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]

- **Oggetto dell'ambiente 2:** Dir Dataset_Divided, ovvero la directory contenente i dataset suddivisi.
  - Categoria: esistenza (ED)
    - Scelta 1: esiste (ED1)
    - Scelta 2: non esiste (ED2) [single]

- **Oggetto dell'ambiente 3:** 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]

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

Tabella 2.1: Test case per lo script divide_dataset

2.1.2 repo_Mining.py

Lo script presenta due funzioni: initialize e startMiningRepo.

Per la funzione initialize si considerano i seguenti parametri e oggetti dell'ambiente:

- **Parametro 1:** 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]
- **Ogetto dell'ambiente 1:** 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]
- **Oggetto dell'ambiente 2:** 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]
- **Oggetto dell'ambiente 3:** 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]

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 e la repository è creata
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

Tabella 2.2: Test case per la funzione initialize

Per la funzione startingMiningRepo si considerano i seguenti parametri e oggetti dell'ambiente:

- **Parametro 1:** 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 sistema operativo (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]

- **Parametro 2:** cwd, ovvero il parametro contenente il path della directory corrente
  - Categoria: formato (FW)
    * Scelta 1: stringa con formato valido (FW1) [property cwdFormatOk]
    * Scelta 2: stringa con formato non valido (FW2) [error]
    * Scelta 3: non stringa (FW3) [error]
  - Categoria: esistenza percorso (EW)
    * Scelta 1: esiste (EW1) [if cwdFormatOk]
    * Scelta 2: non esiste (EW2) [if cwdFormatOk] [error]

- **Parametro 3:** 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]

- **Oggetto dell'ambiente 1:** repo Cvd_id
    - Categoria: esistenza (ERI)
        * Scelta 1: esiste (ERI1)
        * Scelta 2: non esiste (ERI2)

- **Oggetto dell'ambiente 2:** repo Commit
    - Categoria: esistenza (ERC)
        * Scelta 1: esiste (ERC1) [single]
        * Scelta 2: non esiste (ERC2)

- **Oggetto dell'ambiente 3:** 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)

- **Oggetto dell'ambiente 4:** 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)

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 CM1 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica status ok
TC_1.3_10	FD1 N3 CL1 CR1 CC1 CM2 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica status ok
TC_1.3_11	FD1 N3 CL1 CR1 CC1 CM3 FW1 EW1 FR1 ER1 ERI2 ERC2 EC1 SC1 EE2 SE1	CHECKS.txt indica status ok e dir. con id commit creata e vuota

Test Case	Combinazione	Oracolo
TC_1.3_12	FD1 N3 CL1 CR1 CC1 CM4 FW1 EW1 FR1 ER1 ERI1 ERC1 EC1 SC1 EE1 SE1	CHECKS.txt indica status ok e dir. con id commit ha file modificati
TC_1.3_13	FD1 N2 CL1 CR1 CC1 CM4 FW1 EW1 FR1 ER1 ERI1 ERC1 EC2 SC1 EE2 SE1	CHECKS.txt e ERRORS.txt creati CHECKS.txt indica status ok e dir. con id commit ha file modificati
TC_1.3_14	FW3	TypeError
TC_1.3_15	FW2	OSError
TC_1.3_16	FW1 EW2	FileNotFoundError
TC_1.3_17	FR2	OsError
TC_1.3_18	FR3	TypeError
TC_1.3_19	FR1 ER2	FileNotFoundError
TC_1.3_20	EC1 SC2	PermissionError
TC_1.3_21	EE1 SE2	PermissionError

Tabella 2.3: Test case per la funzione start_mining_repo

2.2 Text Mining

Riguardo il processo di estrazione di metriche relative all'utilizzo di parole chiave all'interno del codice, si è deciso di testare singolarmente i metodi dei seguenti script: text_mining.py, dict_generator.py, less_element_text_mining.py e creator_csv_for_TextMining.py. Di seguito sono riportati i parametri e gli oggetti dell'ambiente considerati, con le relative categorie e possibili scelte per la definizione dei test case.

2.2.1 text_mining.py

Lo script presenta 5 funzioni, 4 delle quali lavorano sul contenuto dei file Java estratti e hanno un parametro di input e l'ultima, denominata main, sprovvista di parametri e utiliz-

zata per avviare l'elaborazione. Analizziamo adesso i parametri e gli oggetti dell'ambiente considerati per il testing di tale script.

Per quanto riguarda la funzione **removeNotAlpha(tokens)**, abbiamo individuato le seguenti catergorie di parametri:

- **Parametro 1**: tokens
  - Categoria: contenuto della lista tokens (CN)
    - Scelta 1: Lista vuota (CN1)
    - Scelta 2: Lista con tutti caratteri alfabetici (CN2)
    - Scelta 3: Lista con caratteri non alfabetici (CN3)
    - Scelta 4: Lista mista (CN4)

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

Test Case	Combinazione	Oracolo
TC_2.1_1	CN1	Restituisce una lista vuota
TC_2.1_2	CN2	Restituisce una lista identica a quella di input
TC_2.1_3	CN3	Restituisce una lista vuota
TC_2.1_4	CN4	Restituisce una lista contenente le parole con soli caratteri alfabetici

Tabella 2.4: Test case per la funzione removeNotAlpha(tokens)

Per quanto riguarda la funzione **stringTokenizer(s)**, abbiamo individuato le seguenti catergorie di parametri:

- **Parametro 1**: s
  - Categoria: Contenuto stringa s (CN)
    - Scelta 1: Stringa vuota (CN1)
    - Scelta 2: Stringa senza costanti stringhe o commenti (CN2)
    - Scelta 3: Stringa con solo commenti (CN3)
    - Scelta 4: Stringa con solo costanti stringhe (CN4)
    - Scelta 5: Stringa con costanti stringhe, commenti e parole (CN5)

    * Scelta 6: Stringa con costanti stringhe contenenti "//", commenti e parole (CN6)

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

Test Case	Combinazione	Oracolo
TC_2.2_1	CN1	Restituisce una lista vuota
TC_2.2_2	CN2	Restituisce una lista con tutte e sole le parole all'interno della stringa di input
TC_2.2_3	CN3	Restituisce una lista con le parole all'interno dei commenti multi-line
TC_2.2_4	CN4	Restituisce una lista vuota
TC_2.2_5	CN5	Restituisce una lista con tutte e sole le parole all'interno della stringa di input, tranne nei commenti single-line
TC_2.2_6	CN6	Restituisce una lista con tutte e sole le parole all'interno della stringa di input, tranne nei commenti single-line

Tabella 2.5: Test case per la funzione stringTokenizer(s)

Per quanto riguarda la funzione **removeComments(java_file)**, abbiamo individuato le seguenti catergorie di parametri:

- **Parametro 1**: java_file
  - Categoria: Contenuto file (CN)
    * Scelta 1: File vuoto (CN1) [if accessFileOk]
    * Scelta 2: File senza commenti (CN2) [if accessFileOk]
    * Scelta 3: File con solo commenti (CN3) [if accessFileOk]
    * Scelta 4: File con codice e commenti (CN4) [if accessFileOk]
  - Categoria: Accessibilità del file (AC)
    * Scelta 1: File accessibile (AC1) [property accessFileOk]
    * Scelta 2: File non accessibile (AC2) [error]

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

Test Case	Combinazione	Oracolo
TC_2.3_1	AC2	PermissionError
TC_2.3_2	AC1 CN1	Restituisce una stringa vuota
TC_2.3_3	AC1 CN2	Restituisce una stringa uguale al contenuto del file
TC_2.3_4	AC1 CN3	Restituisce una stringa contenente i commenti single-line
TC_2.3_5	AC1 CN4	Restituisce una stringa con solo il codice e commenti single-line

Tabella 2.6: Test case per la funzione removeComments(java_file)

Per quanto riguarda la funzione **takeJavaClass(java_file_name)**, abbiamo individuato le seguenti catergorie di parametri:

- **Parametro 1**: java_file_name (nome del file)
  - Categoria: Stato del file (ST)
    - Scelta 1: File inesistente (ST1) [error]
    - Scelta 2: File esistente con permessi limitati (ST2) [error]
    - Scelta 3: File esistente e accessibile (ST3) [property fileOk]
    - Scelta 4: Si tratta di una directory (ST4) [error]
  - Categoria: Contenuto del file (CN)
    - Scelta 1: File vuoto (CN1) [if fileOk]
    - Scelta 2: File non vuoto con parole diverse (CN2) [if fileOk]
    - Scelta 3: File non vuoto con parole che si ripetono (CN3) [if fileOk]

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

Test Case	Combinazione	Oracolo
TC_2.4_1	ST1	FileNotFoundError
TC_2.4_2	ST2	PermissionError
TC_2.4_3	ST3 CN1	Dizionario vuoto
TC_2.4_4	ST3 CN2	Restituisce un dizionario con tutte le parole e frequenza 1
TC_2.4_5	ST3 CN3	Restituisce un dizionario con tutte le parole e relativa frequenza
TC_2.4_6	ST4	IsADirectoryError

Tabella 2.7: Test case per la funzione takeJavaClass(java_file_name)

Per quanto riguarda la funzione **main()**, abbiamo individuato le seguenti catergorie di parametri:

- **Oggetto dell'ambiente 1**: File system (cartella "/mining_results")
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1) [error]
    - Scelta 2: Directory esistente (DF2) [property miningOk]
- **Parametro 1**: repo
  - Categoria: Esistenza directory (ES)
    - Scelta 1: Directory non presente (ES1) [error]
    - Scelta 2: Directory presente (ES2) [if miningOk property repoOk]
  - Categoria: Contenuto directory (CN)
    - Scelta 1: Directory vuota (CN1) [error]
    - Scelta 2: Directory non vuota (CN2) [if repoOk property repoNotEmpty]
- **Parametro 2**: cvd_id
  - Categoria: Stato parametro (STC)
    - Scelta 1: Il parametro è un file (STC1) [error]

    - Scelta 2: Il parametro è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [error]
    - Scelta 3: Il parametro è una cartella (STC3) [if repoNotEmpty property isADir]
  - Categoria: Contenuto directory (CNC)
    - Scelta 1: La directory è vuota (CNC1) [error]
    - Scelta 2: La directory non è vuota (CNC2) [if isADir property cvdOk]

- **Parametro 3**: folder
  - Categoria: Stato Parametro (STF)
    - Scelta 1: Il parametro è un file (STF1) [error]
    - Scelta 2: Il parametro è un file non ammesso (STF2) [error]
    - Scelta 3: Il parametro è una cartella (STF3) [if cvdOk property folderOk]
  - Categoria: Contenuto directory (CNF)
    - Scelta 1: La directory è vuota (CNF1) [error]
    - Scelta 2: La directory non è vuota (CNF2) [if folderOk property foldNotEmpty]

- **Parametro 4**: file
  - Categoria: Contenuto directory (CNFI)
    - Scelta 1: Il parametro è un file Java (o non) (CNFI1) [if foldNotEmpty property isJavaFile]
    - Scelta 2: Il parametro è un file non ammesso (CNFI2) [error]
    - Scelta 3: Il parametro è una directory (CNFI3) [error]
    - Scelta 4: Il parametro è un file non Java (CNFI4) [if foldNotEmpty property isJavaFile] [single]
  - Categoria: Accessibilità file (AC)
    - Scelta 1: File accessibile (AC1) [if isJavaFile property fileAcc]
    - Scelta 2: File non accessibile (AC2) [error]
  - Categoria: Esistenza file (ES)
    - Scelta 1: File di text_mining.txt già esistente (ES1) [if fileAcc]
    - Scelta 2: File di text_mining.txt non esistente (ES2) [if fileAcc]
  - Categoria: Salvataggio text mining (SV)

* Scelta 1: File di text mining salvato (SV1) [if fileAcc]
* Scelta 2: File di text mining non salvato per i permessi (SV2) [error]
* Scelta 3: File di text_mining non salvato per valore errato (SV3) [error]

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

Test Case	**Combinazione**	**Oracolo**
TC_2.5_1	DF1	FileNotFoundError
TC_2.5_2	DF2 ES1	FileNotFoundError
TC_2.5_3	DF2 ES2 CN1	Il file text_mining.txt non viene creato
TC_2.5_4	DF2 ES2 CN2 STC1	NotADirectoryError
TC_2.5_5	DF2 ES2 CN2 STC2	Il file text_mining.txt non viene creato
TC_2.5_6	DF2 ES2 CN2 STC3 CNC1	Il file text_mining.txt non viene creato
TC_2.5_7	DF2 ES2 CN2 STC3 CNC2 STF1	NotADirectoryError
TC_2.5_8	DF2 ES2 CN2 STC3 CNC2 STF2	Il file text_mining.txt non viene creato
TC_2.5_9	DF2 ES2 CN2 STC3 CNC2 STF3 CNF1	Il file text_mining.txt non viene creato
TC_2.5_10	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV1	Il file text_mining.txt viene creato
TC_2.5_11	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES1 SV1	Il file text_mining.txt viene sovrascritto
TC_2.5_12	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV2	PermissionError

Test Case	Combinazione	Oracolo
TC_2.5_13	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 ES2 SV3	TypeError
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
TC_2.5_16	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI3	IsADirectoryError
TC_2.5_17	DF2 ES2 CN2 STC3 CNC2 STF3 CNF2 CNFI4	Il file text_mining.txt viene creato

Tabella 2.8: Test case per la funzione main()

2.2.2 dict_generator.py

Lo script presenta una sola funzione, denominata **main()**, la quale è sprovvista di parametri. Gli oggetti dell'ambiente considerati per il testing di tale script sono:

- **Oggetto dell'ambiente 1**: File system (cartella "/mining_results")
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1) [error]
    - Scelta 2: Directory esistente (DF2) [property miningOk]
- **Parametro 1**: repo
  - Categoria: Esistenza Directory (ES)
    - Scelta 1: Directory "RepositoryMining"+count esistente (ES1) [if miningOk property repoOk]
    - Scelta 2: Directory "RepositoryMining"+count non esistente (ES2) [error]

  - Categoria: Contenuto delle directory (CN)
    * Scelta 1: Directory RepositoryMining+count vuota (CN1) [error]
    * Scelta 2: Directory RepositoryMining+count non vuota (CN2) [if repoOk property repoNotEmpty]

- **Parametro 2**: cvd_id
  - Categoria: Stato Parametro (STC)
    * Scelta 1: Il parametro è un file (STC1) [error]
    * Scelta 2: Il parametro è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [error]
    * Scelta 3: Il parametro è una cartella (STC3) [if repoNotEmpty property cvdOk]
  - Categoria: Contenuto directory (CNC)
    * Scelta 1: La directory è vuota (CNC1) [error]
    * Scelta 2: La directory non è vuota (CNC2) [if cvdOk property cdvNotEmpty]

- **Parametro 3**: folder
  - Categoria: Stato Parametro (STF)
    * Scelta 1: Il parametro è un file (STF1) [error]
    * Scelta 2: Il parametro è un file non ammesso (STF2) [error]
    * Scelta 3: Il parametro è una cartella (STF3) [if cvdNotEmpty property folderOk]
  - Categoria: Contenuto directory (CNF)
    * Scelta 1: La directory è vuota (CNF1) [error]
    * Scelta 2: La directory non è vuota (CNF2) [if folderOk property folderNotEmpty]

- **Parametro 4**: file
  - Categoria: Contenuto directory (CNFI)
    * Scelta 1: Il parametro è text_mining.txt (CNFI1) [if folderNotEmpty property fileOk]
    * Scelta 2: Il parametro è un file non ammesso (CNFI2) [error]
    * Scelta 3: Il parametro è una directory che non contiene "text_mining.txt" nel nome (CNFI3) [error]

    - Scelta 3: Il parametro è una directory che contiene "text_mining.txt" nel nome(CNFI4) [error]
  - Categoria: Accessibilità file (AC)
    - Scelta 1: File accessibile (AC1) [if fileOk property fileAcc]
    - Scelta 2: File non accessibile (AC2) [error]
  - Categoria: Contenuto del file (CF)
    - Scelta 1: File di text mining vuoto (CF1) [error]
    - Scelta 2: File di text mining malformato (non contiene un dizionario valido) (CF2) [error]
    - Scelta 3: File di text mining (text_mining.txt) ben formato (contiene un dizionario valido) (CF3) [if fileAcc property contOk]
    - Scelta 4: File di text mining non contiene un dizionario (CF4) [error]

- **Parametro 5**: dict_file
  - Categoria: Salvataggio file (SV)
    - Scelta 1: File salvato correttamente (SV1) [if contOk]
    - Scelta 2: File non salvato per permessi insufficienti (SV2) [error]

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

Test Case	**Combinazione**	**Oracolo**
Test Case	**Combinazione**	**Oracolo**
TC_2.6_1	DF1	FileNotFoundError
TC_2.6_2	DF2 ES2	FileNotFoundError
TC_2.6_3	DF2 ES1 CN1	Il file viene creato con un dizionario vuoto
TC_2.6_4	DF2 ES1 CN2 STC1	NotADirectoryError
TC_2.6_5	DF2 ES1 CN2 STC2	Il file viene creato con un dizionario vuoto

Test Case	Combinazione	Oracolo
TC_2.6_6	DF2 ES1 CN2 STC3 CNC1	Il file viene creato con un dizionario vuoto
TC_2.6_7	DF2 ES1 CN2 STC3 CNC2 STF1	NotADirectoryError
TC_2.6_8	DF2 ES1 CN2 STC3 CNC2 STF2	Il file viene creato con un dizionario vuoto
TC_2.6_9	DF2 ES1 CN2 STC3 CNC2 STF3 CNF1	Il file viene creato con un dizionario vuoto
TC_2.6_10	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI2	Il file viene creato con un dizionario vuoto
TC_2.6_11	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI3	Il file viene creato con un dizionario vuoto
TC_2.6_12	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI4	IsADirectoryError
TC_2.6_13	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 CF1	SyntaxError
TC_2.6_14	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 CF2	SyntaxError
TC_2.6_15	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 CF3 SV1	Il file viene creato con un dizionario non vuoto, basato sui file di text mining
TC_2.6_16	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 CF3 SV2	PermissionError
TC_2.6_17	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC1 CF4	SyntaxError
TC_2.6_18	DF2 ES1 CN2 STC3 CNC2 STF3 CNF2 CNFI1 AC2	PermissionError

Tabella 2.9: Test case per la funzione main()

2.2.3 less_element_text_mining.py

Lo script ha 4 funzioni principali, due delle quali hanno un parametro in input su cui poter effettuare dei test, mentre per le altre verranno utilizzate solo variabili di ambiente. Inoltre, la funzione main non verrà testata, perchè viene unicamente utilizzata per richiamare le altre funzioni, senza alcuna sua logica interna.

Per quanto riguarda la funzione **splitCamelCase(fake_dic)**, abbiamo considerato le seguenti categorie di parametri:

- **Parametro 1**: fake_dic
  - Categoria: Contenuto Parametro (CN)
    - Scelta 1: Dizionario (CN1) [property dictOk]
    - Scelta 2: Dizionario con valori non validi (CN2) [property dictNotOk]
    - Scelta 3: Dizionario con chiavi non valide (CN3) [error]
    - Scelta 4: Altra tipologia (es. lista) (CN4) [error]
  - Categoria: Caratteristiche dizionario (CD)
    - Scelta 1: Dizionario vuoto (CD1) [if dictOk]
    - Scelta 2: Dizionario con chiavi in CamelCase (CD2) [if dictOk | dictNotOk]
    - Scelta 3: Dizionario con chiavi non in CamelCase (CD3) [if dictOk | dictNotOk]
    - Scelta 4: Dizionario con chiavi con parole ricorrenti (CD4) [if dictOk]

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

Test Case	Combinazione	Oracolo
TC_2.7_1	CN1 CD1	Restituisce un dizionario vuoto
TC_2.7_2	CN1 CD2	Restituisce un dizionario con chiavi le parole divise in minuscolo e la frequenza come valore
TC_2.7_3	CN1 CD3	Restituisce un dizionario uguale a quello in input
TC_2.7_4	CN1 CD4	Restituisce un dizionario con chiavi le parole divise in minuscolo e la frequenza come valore
TC_2.7_5	CN2 CD2	Restituisce un dizionario con chiavi le parole divise in minuscolo e la frequenza come valore
TC_2.7_6	CN2 CD3	Restituisce un dizionario uguale a quello in input
TC_2.7_7	CN2 CD4	TypeError
TC_2.7_8	CN3	TypeError
TC_2.7_9	CN4	TypeError

Tabella 2.10: Test case per la funzione splitCamelCase(fake_dic)

Per quanto riguarda la funzione **initialize()**, abbiamo considerato le seguenti categorie di parametri:

- **Oggetto dell'ambiente 1**: File system (cartella "/mining_results")
  - Categoria: Directory e file (DF)
    - Scelta 1: Directory non esistente (DF1) [error]
    - Scelta 2: Directory esistente (DF2) [property miningOk]
  - Categoria: Contenuto directory (CN)
    - Scelta 1: Directory vuota (CN1) [error]
    - Scelta 2: Directory non vuota (CN2) [if miningOk property minNotEmpty]
- **Parametro 1**: dict_file_name

  - Categoria: Stato file (ST)
    - Scelta 1: File presente e accessibile (ST1) [if minNotEmpty property fileOk]
    - Scelta 2: File presente ma non accessibile (ST2) [error]
    - Scelta 3: File non presente (ST3) [error]
  - Categoria: Contenuto del file (CND)
    - Scelta 1: File con dizionario valido (CND1) [if fileOk]
    - Scelta 3: File con dizionario con chiavi non valide (CND2) [error]
    - Scelta 2: File vuoto (CND3) [error]
    - Scelta 4: Altra tipologia (es. lista) (CND4) [error]

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

Test Case	**Combinazione**	**Oracolo**
TC_2.8_1	DF1	FileNotFoundError
TC_2.8_2	DF2 CN1	None
TC_2.8_3	DF2 CN2 ST1 CND1	Restituisce un dizionario con chiavi le parole divise in minuscolo e la frequenza come valore
TC_2.8_4	DF2 CN2 ST1 CND2	TypeError
TC_2.8_5	DF2 CN2 ST1 CND3	SyntaxError
TC_2.8_6	DF2 CN2 ST1 CND4	TypeError
TC_2.8_7	DF2 CN2 ST2	PermissionError
TC_2.8_8	DF2 CN2 ST3	None

Tabella 2.11: Test case per la funzione initialize()

Per quanto riguarda la funzione **writeToFile(dizionario_finale)**, abbiamo considerato le seguenti categorie di parametri:

- **Parametro 1**: FilteredTextMining.txt
  - Categoria: Accesso al file (AC)

    - Scelta 1: Permessi non presenti (AC1) [error]
    - Scelta 2: Permessi presenti (AC2) [property accOk]

- **Parametro 2**: dizionario_finale
  - Categoria: Contenuto del file (CN)
    - Scelta 1: Dizionario valido (CN1) [if accOk property dictOk]
    - Scelta 2: Dizionario vuoto (CN2) [if accOk property dictEmpty]
    - Scelta 3: Dizionario con chiavi non valide (CN3) [if accOk property dictNotOk]
    - Scelta 4: Altra tipologia (es. lista) (CN4) [if accOk property noyDict]
  - Categoria: Salvataggio file CSV (SC)
    - Scelta 1: File salvato correttamente (SC1) [if dictOk | dictEmpty | dictNotOk]
    - Scelta 2: File non salvato correttamente (SC2) [error]

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

Test Case	Combinazione	Oracolo
TC_2.9_1	AC1	PermissionError
TC_2.9_2	AC2 CN1 SC1	Crea un file che contiene il dizionario passato in input
TC_2.9_3	AC2 CN1 SC2	PermissionError
TC_2.9_4	AC2 CN2 SC1	Crea un file che contiene un dizionario vuoto
TC_2.9_5	AC2 CN2 SC2	PermissionError
TC_2.9_6	AC2 CN3 SC1	Crea un file che contiene il dizionario non valido passato in input
TC_2.9_7	AC2 CN3 SC2	PermissionError
TC_2.9_8	AC2 CN4 SC1	Crea un file che contiene l'input dato
TC_2.9_9	AC2 CN4 SC2	PermissionError

Tabella 2.12: Test case per la funzione writeToFile(dizionario_finale)

2.2.4 creator_csv_for_TextMining.py

Lo script presenta una sola funzione, denominata **main**, la quale è sprovvista di parametri. Gli oggetti dell'ambiente considerati per il testing di tale script sono:

- **Oggetto dell'ambiente 1**: File system (cartella "/mining_results")
  - Categoria: Esistenza directory (DF)
    - Scelta 1: Directory esistente (DF1) [error]
    - Scelta 2: Directory non esistente (DF2) [property miningOk]
- **Parametro 1**: dict_file_name
  - Categoria: Esistenza del file (ES)
    - Scelta 1: File FilteredTextMining.txt presente (ES1) [if miningOk property filtOk]
    - Scelta 2: File FilteredTextMining.txt non presente (ES2) [error]
  - Categoria: Contenuto del file (CN)
    - Scelta 1: File FilteredTextMining.txt vuoto (CN1) [error]
    - Scelta 2: File FilteredTextMining.txt contiene un dizionario valido (CN2) [if filtOk property filtValid]
    - Scelta 3: File FilteredTextMining.txt con contenuto non valido (es. dati non in formato dizionario) (CN3) [error]
- **Parametro 2**: repo
  - Categoria: Esistenza directory (ESDR)
    - Scelta 1: RepositoryMining+count esiste (ESDR1) [if miningOk property repoOk]
    - Scelta 2: RepositoryMining+count non esiste (ESDR2) [error]
  - Categoria: Contenuto directory (CND)
    - Scelta 1: RepositoryMining vuota (CND1) [error]
    - Scelta 2: RepositoryMining non vuota (CND2) [if repoOk property repoNotEmpty]
- **Parametro 3**: cvd_id

  - Categoria: Stato Parametro (STC)
    * Scelta 1: Il parametro è un file (STC1) [error]
    * Scelta 2: Il parametro è un file non ammesso (".DS_Store", "CHECK.txt", "ERRORS.txt") (STC2) [error]
    * Scelta 3: Il parametro è una cartella (STC3) [if repoNotEmpty property cvdOk]
  - Categoria: Contenuto directory (CNC)
    * Scelta 1: La directory è vuota (CNC1) [error]
    * Scelta 2: La directory non è vuota (CNC2) [if cvdOk property cvdNotempty]

- **Parametro 4**: folder
  - Categoria: Stato Parametro (STF)
    * Scelta 1: Il parametro è un file (STF1) [error]
    * Scelta 2: Il parametro è un file non ammesso (STF2) [error]
    * Scelta 3: Il parametro è una cartella (STF3) [if cvdNotEmpty property folderOk]
  - Categoria: Contenuto directory (CNF)
    * Scelta 1: La directory è vuota (CNF1) [error]
    * Scelta 2: La directory non è vuota (CNF2) [if folderOk property folderNotEmpty]

- **Parametro 5**: file
  - Categoria: Contenuto directory (CNFI)
    * Scelta 1: Il parametro è text_mining.txt (CNFI1) [if folderNotEmpty propert txtOk]
    * Scelta 2: Il parametro è un file non ammesso (CNFI2) [error]
    * Scelta 3: Il parametro è una directory (CNFI3) [error]
    * Scelta 4: Il parametro è una directory con text_mining.txt nel nome (CNFI4) [error]
  - Categoria: Accessibilità file (AC)
    * Scelta 1: File accessibile (AC1) [if txtok property txtAcc]
    * Scelta 2: File non accessibile (AC2) [error]
  - Categoria: Contenuto file (CNTM)

    * Scelta 1: File text_mining.txt contiene un dizionario valido (CNTM1) [if txtAcc property txtDict]
    * Scelta 2: File text_mining.txt vuoto (CNTM2) [if txtAcc property txtEmpty]
    * Scelta 3: File text_mining.txt con contenuto non valido (non in formato dizionario) (CNTM3) [error]
  - Categoria: Salvataggio file CSV (SC)
    * Scelta 1: File salvato correttamente (SC1) [if filtValid & (txtDict | txtEmpty)]
    * Scelta 2: File non salvato correttamente (SC2) [error]

Test Case	**Combinazione**	**Oracolo**
TC_2.10_1	DF1	FileNotFoundError
TC_2.10_2	DF2 ES2	FileNotFoundError
TC_2.10_3	DF2 ES1 CN1	SyntaxError
TC_2.10_4	DF2 ES1 CN3	AttributeError
TC_2.10_5	DF2 ES1 CN2 ESDR2	FileNotFoundError
TC_2.10_6	DF2 ES1 CN2 ESDR1 CND1	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_7	DF2 ES1 CN2 ESDR1 CND2 STC1	NotADirectoryError
TC_2.10_8	DF2 ES1 CN2 ESDR1 CND2 STC2	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_9	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC1	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_10	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1	NotADirectoryError

Test Case	Combinazione	Oracolo
TC_2.10_11	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF2	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_12	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF3 CNF1	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_13	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI1 AC1 CNTM1 SC1	Viene creato un file csv con le parole chiavi come colonne e una riga per ogni file con le frequenze
TC_2.10_14	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI1 AC1 CNTM1 SC2	PermissionError
TC_2.10_15	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI1 AC1 CNTM2 SC1	Viene creato un file csv con le parole chiavi come colonne e una riga per ogni file con le frequenze a 0
TC_2.10_16	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI1 AC1 CNTM3	TypeError
TC_2.10_17	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI1 AC2	PermissionError
TC_2.10_18	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI2	Viene creato un file csv con solo le parole chiavi come colonne
TC_2.10_19	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI3	Viene creato un file csv con solo le parole chiavi come colonne

Test Case	Combinazione	Oracolo
TC_2.10_20	DF2 ES1 CN2 ESDR1 CND2 STC3 CNC2 STF1 CNF2 CNFI4	IsADirectoryError

Tabella 2.13: Test case per la funzione main()

2.3 Rifinitura dati analisi statica

Riguardo il processing dei dati ricavati dall'analisi statica automatica, si è deciso di testare i seguenti script: ASA_vulnerability_dict_generator.py, rules_dict_generator_ASA.py e creator_csv_for_ASA.py. Di seguito sono riportati gli oggetti dell'ambiente considerati e le relative categorie per la definizione dei test case.

2.3.1 ASA_vulnerability_dict_generator.py

Lo script presenta una sola funzione, denominata main, la quale è sprovvista di parametri. Gli oggetti dell'ambiente considerati per il testing di tale script sono:

- **Oggetto dell'ambiente 1:** 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]

- **Oggetto dell'ambiente 2:** 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)

- **Oggetto dell'ambiente 3:** 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]

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 2.14: test case per ASA_vulnerability_dict_generator.py

2.3.2 rules_dict_generator_ASA.py

Lo script presenta una sola funzione, denominata main, la quale è sprovvista di parametri. Gli oggetti dell'ambiente considerati per il testing di tale script comprendono RepositoryMining_ASAResults_NEG e RepositoryMining_ASAResults_POS, con le rispettive categorie e scelte individuate sopra. Inoltre, si considera anche:

- **Oggetto dell'ambiente 3:** ASA_rules_dict.csv, ovvero il file contenente le regole relative alle vulnerabilità estratte tramite l'analisi statica.
  - Categoria: possibilità di scrittura (S)
    - Scelta 1: è possibile scrivere nel file (S1)
    - Scelta 2: non è possibile scrivere nel file (S2) [error]

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 2.15: test case per rules_dict_generator_ASA.py

2.3.3 creator_csv_for_ASA.py

Lo script presenta una sola funzione, denominata main, la quale è sprovvista di parametri. Gli oggetti dell'ambiente considerati per il testing di tale script sono:

- **Oggetto dell'ambiente 1:** Asa_dict_csv, ovvero il file contenente le componenti a cui è associata una vulnerabilità, con la relativa regola e classificazione.
  - Categoria: esistenza (EA)
    - Scelta 1: esiste (EA1) [property vulnDictExistOk]
    - Scelta 2: non esiste (EA2) [error]
  - Categoria: numero record (NA)
    - Scelta 1: zero (NA1) [if vulnDictExistOk]

        * Scelta 2: uno (NA2) [if vulnDictExistOk] [property vulnDictNotEmpty]
        * Scelta 3: maggiore di uno (NA3) [if vulnDictExistOk] [property vulnDictNotEmpty] [single]
    - Categoria: formato (FA)
        * Scelta 1: formato valido (FA1) [if vulnDictExistOk] [property vulnDictFormatOk]
        * Scelta 2: formato non valido (FA2) [if vulnDictExistOk] [error]
    - Categoria: contenuto (CA)
        * Scelta 1: un componente occorre al più una volta con una regola (CA1) [if vulnDictFormatOk and vulnDictNotEmpty]
        * Scelta 2: un componente occorre più volte con la stessa regola (CA2) [if vulnDictFormatOk and vulnDictNotEmpty] [single]

- **Oggetto dell'ambiente 2:** rules_dict_csv, ovvero il file contenente le regole relative alle vulnerabilità estratte tramite l'analisi statica.
    - Categoria: esistenza (ER)
        * Scelta 1: esiste (ER1) [property rulesDictExistOk]
        * Scelta 2: non esiste (ER2) [error]
        * Scelta 1: zero (NR1) [if rulesDictExistOk]
        * Scelta 2: uno (NR2) [if rulesDictExistOk] [property rulesDictNotEmpty]
        * Scelta 3: maggiore di uno (NR3) [if rulesDictExistOk] [property rulesDictNotEmpty]
    - Categoria: formato (FR)
        * Scelta 1: formato valido (FR1) [if rulesDictExistOk] [property rulesDictFormatOk]
        * Scelta 2: formato non valido (FR2) [if rulesDictExistOk] [error]
    - Categoria: contenuto (CR)
        * Scelta 1: regola con corrispondenza (CR1) [if rulesDictFormatOk and rulesDictNotEmpty and vulnDictNotEmpty] [single]
        * Scelta 2: regola senza corrispondenza (CR2) [if rulesDictFormatOk and rulesDictNotEmpty and vulnDictNotEmpty]

  - Categoria: numero record (NR)

- **Oggetto dell'ambiente 3:** final_csv, ovvero il file risultante dal processing in cui ogni componente è associata la frequenza di ciascuna regola di vulnerabilità.
  - Categoria: possibilità di scrittura (S)
    * Scelta 1: sì (S1)
    * Scelta 2: no (S2) [error]

Test case	Combinazione	Oracolo
TC_3.3_1	EA2	FileNotFoundError
TC_3.3_2	ER2	FileNotFoundError
TC_3.3_3	S2	PermissionError
TC_3.3_4	EA1 NA2 FA2	JSONDecodeError
TC_3.3_5	ER1 NR2 FR2	JSONDecodeError
TC_3.3_6	EA1 NA3 FA1 CA1 ER1 NR3 FR1 CR1 S1	final_csv contenente la classe java con un 1 in corrispondenza della regola
TC_3.3_7	EA1 NA2 FA1 CA2 ER1 NR2 FR1 CR1 S1	final_csv contenente la classe java con un contatore > 1 per la regola ripetuta
TC_3.3_8	EA1 NA2 FA1 CA1 ER1 NR2 FR1 CR1 S1	final_csv contenente la classe java con un 1 in corrispondenza della regola
TC_3.3_9	EA1 NA2 FA1 CA1 ER1 NR2 FR1 CR2 S1	final_csv contenente la classe java con 0 per la regola senza corrispondenza
TC_3.3_10	EA1 NA1 FA1 ER1 NR2 FR1 CR2 S1	final_csv con solo l'header contenente le regole
TC_3.3_11	EA1 NA2 FA1 CA1 ER1 NR1 FR1 S1	final_csv contenente l'elenco di classi analizzate e le relative classificazioni
TC_3.3_12	EA1 NA1 FA1 ER1 NR1 FR1 S1	final_csv vuoto

Tabella 2.16: test case per creator_csv_for_ASA.py

2.4 Union

Per il package Union si è deciso di testare i seguenti script presenti nel package: Union.py, Union_TMwithASA.py, Union_SMwithASA.py, TotalCombination.py. Di seguito sono riportati le funzioni, i parametri e gli oggetti dell'ambiente considerati e le relative categorie per la definizione dei test case. Per ugnuno di questi tre script saranno testate le tre funzioni getClass, another_option ed initialize.

2.4.1 Union.py

La funzione getClass, è provvista di un solo parametro line che rappresenta una stringa.

- Categoria: lunghezza (L)
  - Scelta 1: La stringa è vuota [property emptyString]
  - Scelta 2: La stringa è formata da 1 o più elementi [property notEmptyString]
- Categoria: contenuto (C):
  - Scelta 1: La stringa termina con la sottostringa 'pos' o 'neg' [if notEmptyString]
  - Scelta 2: La stringa non termina con la sottostringa 'pos' o 'neg' [if notEmptyString]

Test Case	Combinazione	Oracolo
TC_4.1_1	L1	La funzione restituisce una stringa vuota
TC_4.1_2	L2 C1	La funzione restituisce l'ultimo elemento della stringa line
TC_4.1_3	L2 C2	La funzione restituisce l'ultimo elemento della stringa line

Tabella 2.17: Test cases per getClass

La funzione another_option è provvista di 3 parametri: line_sm che rappresenta una riga del dataset contenente le metriche del software; line_tm che rappresenta una riga del dataset contenente le informazioni derivanti dal text mining; class_element inerente alla riga analizzata che può essere pos o neg.

- **Parametro 1:** line_sm (string)
  - Categoria: esistenza(SM)

    * Scelta 1: la stringa non esiste [property lineSMNone]
    * Scelta 2: la stringa esiste ed è composta da meno di 3 elementi [property lineSMEmpty]
    * Scelta 3: la stringa esiste ed è composta da almeno 3 elementi [property lineSM]

- **Parametro 2:** line_tm (string)
  - Categoria: esistenza (TM)
    * Scelta 1: la stringa non esiste [if lineSM or lineSMEmpty property lineTMNone]
    * Scelta 2: la stringa esiste ed è vuota [if lineSMNone property lineTMEmpty]
    * Scelta 3: la stringa esiste e non è vuota [if lineSMNone property lineTM]

- **Parametro 3:** class_element (string)
  - Categoria: esistenza (CE)
    * Scelta 1: la stringa non esiste [if lineTM or lineTMEmpty error]
    * Scelta 2: la stringa esiste ed è vuota [if lineTM error]
    * Scelta 3: la stringa esiste e non è vuota [if lineTM property classElement]
  - Categoria: contenuto (TMC)
    * Scelta 1: la stringa è contenuta in line_tm [if classElement error]
    * Scelta 2: la stringa non è contenuta in line_tm [if classElement]

Test Case	Combinazione	Oracolo
TC_4.2_1	SM1 TM2 CE1	Value Error
TC_4.2_2	SM1 TM3 CE1	ValueError
TC_4.2_3	SM2 TM1	La funzione restituisce stringa vuota
TC_4.2_4	SM3 TM1	La funzione restituisce `line_sm` con la rimozione dei primi due elementi e delle virgole che li seguono, aggiungendo una virgola finale
TC_4.2_5	SM1 TM3 CE2	ValueError
TC_4.2_6	SM1 TM3 CE3 TMC1	ValueError
TC_4.2_7	SM1 TM3 CE3 TMC2	La funzione restituisce `line_tm` con la rimozione della sottostringa `class_element`

Tabella 2.18: Test cases per anoter_option

La funzione initialize è provvista di 3 parametri: name_csv_mining che rappresenta il nome del csv mining; name_csv_soft_m che rappresenta il nome del csv contenente le metriche del software; new_Union che rappresenta il file che conterrà il risultato dell'unione dei due csv. Inolte, come oggetti dell'ambiente abbiamo csv_mining e csv_softare_metric che rappresentano i due file menzionati prima.

- **Parametro 1:** name_csv_mining(string)
  - Categoria: contenuto(CM)
    - Scelta 1: la stringa è vuota [error]
    - Scelta 2: la stringa è diversa da "csv_mining_final.csv" [error]
    - Scelta 3: la stringa è uguale a "csv_mining_final.csv"
- **Parametro 2:** name_csv_soft_m (string)
  - Categoria: contenuto (CSM)
    - Scelta 1: la stringa è vuota o non esiste [error]
    - Scelta 2: la stringa è diversa da "mining_results_sm_final.csv" [error]

        * Scelta 3: la stringa è uguale a "mining_results_sm_final.csv"

- **Parametro 3:** new_Union (file)
    - Categoria: accessibilità (AU)
        * Scelta 1: non ci sono permessi in scrittura [error]
        * Scelta 2: ci sono permessi in scrittura
- **Oggetto dell'ambiente 1:** csv_mining (file)
    - Categoria: esistenza (EMF)
        * Scelta 1: il file non esiste [error]
        * Scelta 2: il file esiste [property exCSV1]
    - Categoria: contenuto (CMF)
        * Scelta 1: il file è vuoto [if exCSV1][single]
        * Scelta 2: il file contiene solo l'intestazione [if exCSV1][single]
        * Scelta 3: il file contiene intestazione e dati [if exCSV1]
- **Oggetto dell'ambiente 2:** csv_software_metric (file)
    - Esistenza (ESMF)
        * Scelta 1: il file non esiste [error]
        * Scelta 2: il file esiste [property exCSV2]
    - Categoria: contenuto (CSMF)
        * Scelta 1: il file è vuoto [if property exCSV2]
        * Scelta 2: il file contiene solo l'intestazione [if property exCSV2]
        * Scelta 3: il file contiene intestazione e dati [if property exCSV2]

Test Case	Combinazione	Oracolo
TC_4.3_1	CM1	FileNotFoundError
TC_4.3_2	CM2	FileNotFoundError
TC_4.3_3	CM3 CSM1	FileNotFoundError
TC_4.3_4	CM3 CSM2	FileNotFoundError
TC_4.3_5	CM3 CSM3 AU1	PermissionError
TC_4.3_6	CM3 CSM3 AU2 EMF1	FileNotFoundError
TC_4.3_7	CM3 CSM3 AU2 EMF2 CMF1 ESMF2 CSMF1	Il file di Unione risulterà vuoto
TC_4.3_8	CM3 CSM3 AU2 EMF2 CMF2 ESMF2 CSMF2	La funzione combina le intestazioni dei due file nel file di Unione
TC_4.3_9	CM3 CSM3 AU2 EMF2 CMF3 ESMF1	FileNotFoundError
TC_4.3_10	CM3 CSM3 AU2 EMF2 CMF3 ESMF2 CSMF1	La funzione combina le intestazioni dei due file nel file di Unione
TC_4.3_11	CM3 CSM3 AU2 EMF2 CMF3 ESMF2 CSMF2	La funzione combina le intestazioni dei due file nel file di Unione
TC_4.3_12	CM3 CSM3 AU2 EMF2 CMF3 ESMF2 CSMF3	La funzione combina le intestazioni e i dati dei due file nel file di Unione, combinando i dati che hanno lo stesso nome nella prima colonna

Tabella 2.19: Test cases per anoter_option - Union.py

2.4.2 TotalCombination.py

La funzione getClass presente in TotalCombination.py è analoga a quella contenuta nello script Union.py, così come la funzione another_option e la funzione initialize.

2.4.3 Union_TMwithASA.py

La funzione getClass presente in Union_TMwithASA.py è analoga a quella contenuta nel modulo Union.py.

La funzione another_option è provvista di 3 parametri: line_asa che rappresenta una riga del primo dataset; line_tm che rappresenta una riga del secondo dataset; class_element inerente alla riga analizzata che può essere pos o neg.

- **Parametro 1:** class_element (string)
  - Categoria: esistenza (CE)
    - Scelta 1: la stringa non esiste [error]
    - Scelta 2: la stringa esiste ed è vuota [single]
    - Scelta 3: la stringa esiste e non è vuota
- **Parametro 2:** line_asa (string)
  - Categoria: esistenza(ASA)
    - Scelta 1: la stringa non esiste
    - Scelta 2: la stringa esiste ed è vuota [single]
    - Scelta 3: la stringa esiste e non è vuota [property lineASA]
  - Categoria: contenuto (ASAC)
    - Scelta 1: la stringa non contiene class element [if lineASA][error]
    - Scelta 2: la stringa contiene class element [if lineASA]
- **Parametro 3:** line_tm (string)
  - Categoria: esistenza (TM)
    - Scelta 1: la stringa non esiste
    - Scelta 2: la stringa esiste ed è vuota [single]
    - Scelta 3: la stringa esiste e non è vuota [property lineTM]
  - Categoria: contenuto (TMC)

    * Scelta 1: la stringa non contiene class element [if lineTM][error]
    * Scelta 2: la stringa contiene class element [if lineTM]

Test Case	Combinazione	Oracolo
TC_5.1_1	CE1	AttributeError
TC_5.1_2	CE2 ASA2 TM2	La funzione restituisce None
TC_5.1_3	CE3 ASA1 TM1	AttributeError
TC_5.1_4	CE3 ASA1 TM3 TMC1	ValueError
TC_5.1_5	CE3 ASA1 TM3 TMC2	La funzione restituisce line_tm rimuovendo il class element
TC_5.1_6	CE3 ASA3 TM1 ASAC1	ValueError
TC_5.1_7	CE3 ASA3 TM1 ASAC2	La funzione restituisce line_asa rimuovendo il primo elemento ed class element
TC_5.1_8	CE3 ASA3 ASAC2 TM3 TMC2	La funzione restituisce None

Tabella 2.20: Test cases per another_option

La funzione initialize è provvista di 3 parametri: name_csv_mining che rappresenta il nome del csv mining; name_csv_asa che rappresenta il nome del csv contenente le metriche del software; new_Union che rappresenta il file che conterrà il risultato dell'unione dei due csv. Inolte, come oggetti dell'ambiente abbiamo csv_mining e csv_softare_metric che rappresentano i due file menzionati prima.

- **Parametro 1:** name_csv_mining (string)
    - Categoria: contenuto (CM)
        * Scelta 1: la stringa è vuota [error]
        * Scelta 2: la stringa è diversa da "csv_mining_final.csv" [error]
        * Scelta 3: la stringa è uguale a "csv_mining_final.csv"

- **Parametro 2:** name_csv_asa (string)
  - Categoria: contenuto (CASA)
    - Scelta 1: la stringa non esiste [error]
    - Scelta 2: la stringa è diversa da "csv_ASA_final.csv" [error]
    - Scelta 3: la stringa è uguale a "csv_ASA_final.csv"
- **Parametro 3:** new_Union (file)
  - Categoria: esistenza (AU)
    - Scelta 1: il file non esiste [error]
    - Scelta 2: il file esiste
- **Oggetto dell'ambiente 1:** csv_mining (file)
  - Categoria: esistenza (EMF)
    - Scelta 1: il file non esiste [error]
    - Scelta 2: il file esiste [property exCSV1]
  - Categoria: contenuto (CMF)
    - Scelta 1: il file è vuoto [if exCSV1][single]
    - Scelta 2: il file contiene solo l'intestazione [if exCSV1][single]
    - Scelta 3: il file contiene intestazione e dati [if exCSV1]
- **Oggetto dell'ambiente 2:** csv_asa (file)
  - Categoria: esistenza (EASAF)
    - Scelta 1: il file non esiste [error]
    - Scelta 2: il file esiste [property exCSV2]
  - Categoria: contenuto (CASAF)
    - Scelta 1: il file è vuoto [if exCSV2]
    - Scelta 2: il file contiene solo l'intestazione [if exCSV2]
    - Scelta 3: il file contiene intestazione e dati [if exCSV2]

Test Case	Combinazione	Oracolo
TC_5.2_1	CM1	FileNotFoundError
TC_5.2_2	CM2	FileNotFoundError
TC_5.2_3	CM3 CASA1	FileNotFoundError
TC_5.2_4	CM3 CASA2	FileNotFoundError
TC_5.2_5	CM3 CASA3 AU1	PermissionError
TC_5.2_6	CM3 CASA3 AU2 EMF1	FileNotFoundError
TC_5.2_7	CM3 CASA3 AU2 EMF2 CMF1 EASAF2 CASAF1	La funzione non scriverà nulla nel file di Unione, che risulterà vuoto
TC_5.2_8	CM3 CASA3 AU2 EMF2 CMF2 EASAF2 CASAF2	La funzione combina le intestazioni dei due file nel file di Unione
TC_5.2_9	CM3 CASA3 AU2 EMF2 CMF3 EASAF1	FileNotFoundError
TC_5.2_10	CM3 CASA3 AU2 EMF2 CMF3 EASAF2 CASAF1	La funzione scrive nel file di Unione l'intestazione ed i dati del file inerente alle software metrics, ed aggiunge "0" per rimpiazzare i dati asa mancanti
TC_5.2_11	CM3 CASA3 AU2 EMF2 CMF3 EASAF2 CASAF2	La funzione combina le intestazioni dei due file nel file di Unione e rimpiazza i dati mancanti con '0'
TC_5.2_12	CM3 CASA3 AU2 EMF2 CMF3 EASAF2 CASAF3	La funzione combina i due CSV. Laddove ci siano dati mancanti per quanto riguarda il csv asa, aggiunge dei valori '0'

Tabella 2.21: Test cases per initialize

2.4.4 Union_SMwithASA.py

La funzione getClass presente in Union_SMwithASA.py è analoga a quella contenuta nel modulo Union.py, mentre le funzioni another_options ed initialize sono analoghe a quelle contenuta nello script Union_TMwithASA.py