Post-Modification Testing

MARK 2.0 Plus

Data	23/01/2026
Presentato da	Cerchia Giovanni (NF22500202) Medica Vincenzo (NF22500203)
Repository	https://github.com/vmedica/MARK-2.0-Plus

Sommario

1 Introduzione

Il presente documento descrive le attività di testing svolte su **MARK 2.0 Plus** a seguito dell'introduzione delle Change Requests **CR1–CR3**.
Gli obiettivi principali sono:

- **Unit e Integration Testing**: verificare, in modalità **white-box**, la correttezza dei metodi ritenuti critici nelle componenti modificate/aggiunte, con obiettivo di **≥ 80% branch coverage** sulle classi target.
- **Post-Modification System Testing**: validare a livello di sistema le nuove funzionalità introdotte dalle CR (Change Requests), tramite **testing black-box** progettato con **Category Partition**.
- **Regression Testing:** assicurare che il sistema preesistente non abbia subito regressioni funzionali tramite la riesecuzione della suite di test di sistema definita nel **Pre-Modification System Document**, composta da **16 test case** (12 per UC-1 Analysis, 4 per UC-2 Cloning) al fine di verificare l'assenza di regressioni dopo l'integrazione delle CR.

2 Unit e Integration Testing (white-box)

Per le componenti selezionate è stato svolto:

- **Basic Unit Testing (white-box)**: focalizzato sui metodi, con mocking dove necessario per isolare dipendenze (file system/servizi).
- **Integration Testing (white-box)**: sugli stessi metodi ma senza mocking, verificando la corretta interazione con il sistema.

La selezione degli input è stata guidata dalla costruzione del CFG dei metodi target (script Python basato su AST), scegliendo percorsi che coprono il maggior numero di branch.

2.1 Basic Unit Testing — tabella test

Test ID	Classe/Metodo	Path target
UT-CR1-01	`MLAnalyzer.analyze_single_file`	Il file non esiste.
UT-CR1-02	`MLAnalyzer.analyze_single_file`	Errore nella lettura del file.

UT-CR1-03	`MLAnalyzer.analyze_single_file`	Il file viene letto con successo, CC e MI sollevano eccezioni, keywords trovate.
UT-CR1-04	`MLAnalyzer.analyze_single_file`	CC e MI hanno successo, ma non vengono trovate keywords.
UT-CR1-05	`MLAnalyzer.analyze_project`	Role != METRICS, include file non valido, file valido senza keywords, file valido con keywords.
UT-CR1-06	`MLAnalyzer.analyze_project`	Role == METRICS, include file con SLOC > 0 e file con SLOC == 0.
UT-CR1-07	`MLAnalyzer.analyze_projects_set`	Role != METRICS con progetto non-directory, percorso non-directory, e directory valida che ritorna df non vuoto.
UT-CR1-08	`MLAnalyzer.analyze_projects_set`	Role == METRICS con progetto A (cc/sloc vuoti) e progetto B (con cc/sloc), tutti i df vuoti.
UT-CR2-01	`PipelineService.run_pipeline`	Cloning + cloner check abilitati, analisi disabilitate.
UT-CR2-02	`PipelineService.run_pipeline`	Tutte le analisi abilitate (producer, consumer, metrics), nessun cloning.
UT-CR2-03	`PipelineService.run_pipeline`	Percorso CSV non valido - dovrebbe gestire l'errore correttamente.
UT-CR2-04	`OutputReader.scan_output_tree`	Directory di output vuota → restituisce albero vuoto.
UT-CR2-05	`OutputReader.scan_output_tree`	Tutte le categorie (producer, consumer, metrics) con file CSV.
UT-CR2-06	`OutputReader.load_csv`	File non esistente → solleva FileNotFoundError.

UT-CR2-07	`OutputReader.load_csv`	File CSV valido → restituisce CSVData con headers e righe.
UT-CR2-08	`OutputReader.load_csv`	File CSV vuoto → restituisceCSVData vuoto.
UT-CR3-01	`OutputReader.find_complete_analyses`	Nessuna directory di analisi → restituisce lista vuota.
UT-CR3-02	`OutputReader.find_complete_analyses`	Tutte le categorie con stesso ID analisi → restituisce quell'ID.

2.2 Integration Testing — tabella test

Gli stessi path coperti dai test case da UT-CR1-01 a UT-CR3-02 sono stati riutilizzati per generare un numero equivalente di test case di integrazione. Poiché tali path risultano identici e ripetitivi, se ne omette la descrizione dettagliata.
In aggiunta, sono stati testati i metodi della classe `AppController` appartenente alla GUI.

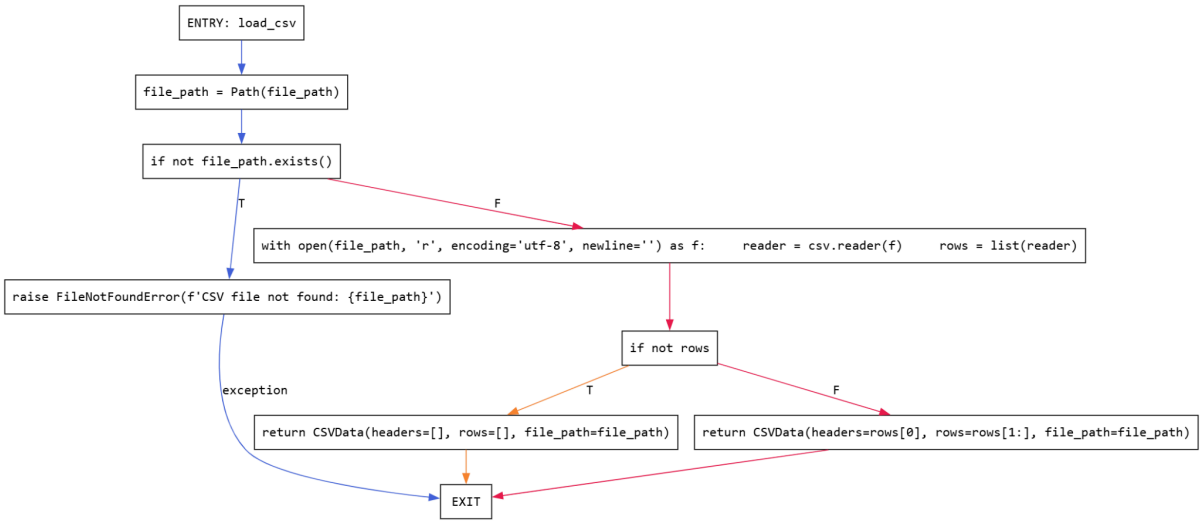
Test ID	Classe/Metodo	Path target
IT-CR2-09	`AppController._on_start_pipeline`	Repo Path non esiste → messaggio d'errore.
IT-CR2-10	`AppController._on_start_pipeline`	Repo Path esiste → pipeline avviata con thread.
IT-CR2-11	`AppController._on_pipeline_complete`	Pipeline completata con successo → mostra info e aggiorna output.
IT-CR2-12	`AppController._on_pipeline_complete`	Pipeline fallisce → mostra errore.
IT-CR2-13	`AppController._on_pipeline_complete`	Pipeline completa senza result → errore sconosciuto.
IT-CR3-03	`AppController._refresh_output_tree`	Aggiorna tree con successo quando esistono analisi.
IT-CR3-04	`AppController._on_file_select`	Selezione file CSV valido → mostra dati.

IT-CR3-05	`AppController._on_file_select`	File non esiste → mostra errore.
IT-CR3-06	`AppController._on_analysis_select`	Producer/Consumer/Metrics CSV esistono → calcola metriche complete.
IT-CR3-07	`AppController._on_analysis_select`	Producer/Consumer CSV non trovati → metriche a zero.

2.3 Esempio CFG e tracciamento dei path

Metodo selezionato per il basic unit testing
`OutputReader.load_csv`

CFG generato



Path coperti

P1 (UT-CR2-06): File non esistente → solleva FileNotFoundError.
P2 (UT-CR2-07): File CSV valido → ritorna CSVData con headers e righe.
P3 (UT-CR2-08): File CSV vuoto → ritorna CSVData vuoto.

2.4 Coverage (pytest-cov, branch coverage)

La coverage è stata calcolata con **pytest-cov** limitatamente alle **classi di interesse** (quelle effettivamente testate in white-box), misurando in particolare la **branch coverage** con soglia obiettivo **≥ 80%**.

2.4.1 Risultati per lo Unit Testing

Package/Classe	Branch coverage	Note
`modules/analyzer/MLAnalyzer`	98%	CR1
`gui/services/pipeline_service.py`	100%	CR2
`gui/services/output_reader.py`	93%	CR2/CR3
Totale	**97%**	

2.4.2 Risultati per l'Integration Testing

Package/Classe	Branch coverage	Note
`modules/analyzer/MLAnalyzer`	100%	CR1
`gui/services/pipeline_service.py`	100%	CR2
`gui/services/output_reader.py`	93%	CR2/CR3
`gui/controller.py`	85%	CR2/CR3
Totale	**95%**	

3 Post-Modification System Testing

Il system testing è condotto in modalità **black-box**, progettando i casi di test tramite **Category Partition** che prevede: l'identificazione di parametri/oggetti dell'ambiente, definizione di categorie e scelte, costruzione di test frame validi e derivazione dei test case con oracoli.

3.1 UC-CR1 — Analisi del calcolo delle metriche

3.1.1 Use Case

Descrizione	L'utente esegue l'analisi su una directory locale contenente repository già disponibili; il tool calcola le metriche Maintainability Index (MI) e Cyclomatic Complexity (CC) per i progetti contenuti, salvando i risultati in `io/output/`.
Attori	Utente (ricercatore)
Entry condition	`main_args.py` disponibile; dipendenze installate (inclusa Radon); directory `--repository-path` accessibile (per scenari "success").
Exit condition	**Successo:** metriche (MI, CC) calcolate e risultati salvati in `io/output/`. **Errore:** messaggio d'errore e assenza di output significativo.
Flusso di eventi principale	1. Invocazione con `--repository-path` e `--metrics`. 2. Validazione del path. 3. Scansione dei repository nella directory. 4. Calcolo MI e CC sui progetti analizzati. 5. Salvataggio output metriche in `io/output/`.

3.1.2 Category Partition — Parametri, oggetti, categorie e scelte

Parametro: Path della directory contenente i repository (`repository_path`)

Oggetti dell'ambiente: Filesystem, contenuto dei progetti appartenenti alla directory analizzata

Categorie e scelte:

Categoria	ID	Descrizione	Proprietà / Vincoli
Esistenza input directory	ED1	Directory non esiste	[Error]
	ED2	Directory esiste	[property DirOk]
Contenuto directory	CD0	Directory vuota (0 progetti)	[if DirOk] [property Empty]
	CD1	Directory con 1+ progetti	[if DirOk] [property MultiProject]

Composizione Progetti	CP0	Tutti i progetti senza file Python	[if DirOk and if MultiProject]
	CP1	Tutti i progetti con file Python ma vuoti	[if DirOk and if MultiProject]
	CP2	Tutti i progetti con file Python validi e con codice	[if DirOk and if MultiProject]
	CP3	Mix di progetti validi e non validi	[if DirOk and if MultiProject]

3.1.3 Test Frame

ID	**Combinazioni (categorie/scelte)**	**Oracolo (risultato atteso)**
TF1	ED1	Visualizza messaggio di errore “Input folder not found”
TF2	ED2, CD0	Nessuna metrica calcolata
TF3	ED2, CD1, CP0	Per tutti i progetti i risultati dell’MI e della CC sono pari a 0
TF4	ED2, CD1, CP1	Per tutti i progetti i risultati dell’MI e della CC sono pari a 0
TF5	ED2, CD1, CP2	Per tutti i progetti i risultati dell’MI e della CC sono dei valori esatti
TF6	ED2, CD1, CP3	Per tutti i progetti i risultati dell’MI e della CC sono dei valori esatti

3.1.4 Test Case

TC	Input (repository_path)	Descrizione ambiente	Oracolo
TC1	*"repo_does_not_exist"*	Directory inesistente	Messaggio "Input folder not found"
TC2	*"empty_repo"*	Directory vuota	Nessun metrica calcolata
TC3	*"test_repos/TC3"*	Più progetti senza file Python	Per tutti i progetti i risultati dell'MI e della CC sono pari a 0
TC4	*"test_repos/TC4"*	Più progetti con file Python vuoti	Per tutti i progetti i risultati dell'MI e della CC sono valori pari a 0
TC5	*"test_repos/TC5"*	Più progetti con file Python contenente codice valido	Valori esatti calcolati manualmente: **project1:** CC_avg = 1.67, MI_avg = 77.5 **project2:** CC_avg = 1.33, MI_avg = 88.75
TC6	*"test_repos/TC6"*	Più progetti senza file Python, con file Python vuoti e contenenti file Python con codice valido	Valori esatti per ogni progetto calcolati manualmente: **project_empty_python_1:** CC_avg = 0, MI_avg = 0 **project_empty_python_2:** CC_avg = 0, MI_avg = 0 **project_no_python_1:** CC_avg = 0, MI_avg = 0 **project_no_python_1:** CC_avg = 0, MI_avg = 0 **project1:** CC_avg = 1.67, MI_avg = 77.51 **project2:** CC_avg = 1.33, MI_avg = 88.75

3.2 UC-CR2-1 — Configurazione e avvio analisi pipeline tramite GUI

3.2.1 Use Case

Descrizione	L'utente configura i parametri della pipeline nella GUI e avvia l'analisi; al termine, il sistema salva i risultati in `io/output/` e mostra l'esito (successo/errore).
Attori	Utente (ricercatore)
Entry condition	GUI aperta; utente nella tab Configuration; interfaccia in stato Idle (pulsante Start Analysis abilitato).
Exit condition	**Successo**: pipeline completata, risultati salvati in `io/output/`, dialogo di successo mostrato, interfaccia in stato Idle sulla tab Output. **Errore**: pipeline fallita o non avviata, dialogo errore/validazione mostrato, interfaccia in stato Idle sulla tab Configuration.
Flusso di eventi principale	1. L'utente imposta i path (IO Path, Repository Path, Project List Path), il numero di repo da clonare e seleziona i pipeline steps e l'opzione Enable Rules 3. 2. L'utente avvia l'analisi con Start Analysis. 3. Il sistema valida i parametri, disabilita Start Analysis e avvia la pipeline in un thread separato. 4. Il sistema completa l'analisi, salva i risultati in `io/output/` e mostra un dialogo di successo.

3.2.2 Category Partition — Parametri, oggetti, categorie e scelte

Parametri:

- `io_path`: percorso alla directory di IO (String)
- `repos_path`: percorso alla directory contenente i progetti (String)
- `project_list_path`: percorso al file CSV contenente la lista di progetti (String)
- `num_repos`: numero di repository da clonare (Integer 1-1000)
- `clone_enabled`: flag per abilitare lo step di cloning (Boolean)
- `verify_enabled`: flag per abilitare la verifica del cloning effettuato (Boolean)

- `producer_enabled`: flag per abilitare l'analisi dei progetti producer (Boolean)
- `consumer_enabled`: flag per abilitare l'analisi dei progetti consumer (Boolean)
- `metrics_enabled`: flag per abilitare l'analisi delle metriche di qualità (Boolean)
- `rules3_enabled`: flag per abilitare la Rule 3 per l'analisi dei consumer (Boolean)

Oggetti dell'ambiente:

- Filesystem: stato del filesystem
- Project CSV Content: contenuto e validità del file CSV contenente i progetti
- Repository State: stato della cartella contenente le repository

Categorie e scelte:

Categoria	**ID**	**Descrizione**	**Vincoli**
Step (Producer, Consumer, Metrics, Cloning e Verify) selezionati	ST0	Nessuno Step selezionato	[Single]
	ST1	Selezionato almeno uno Step	[property PARTIAL (non tutti gli step)]
	ST2	Selezionati tutti gli Step	[property ALL] [Single]
Cloning + Verify	CV1	Cloning e Verify selezionati	[if PARTIAL] [property CV_ON]
	CV2	Cloning e Verify non selezionati	[if PARTIAL] [property CV_OFF]
Esistenza IO directory	IO0	IO directory non esiste	[Error]
	IO1	IO directory esiste	[property IO_OK]
Esistenza directory repo	RP0	Directory repo non esiste	[Single]
	RP1	Directory repo esiste	[property REPO_OK]

Esistenza file CSV	CSV0	File CSV non esiste	[if CV_ON and IO_OK and REPO_OK] [Error]
	CSV1	File CSV esiste	[if CV_ON and IO_OK and REPO_OK] [property CSV_OK]
Stato CSV	CS0	File CSV vuoto	[if CSV_OK] [Single] [property CSV_EMPTY]
	CS1	File CSV non vuoto con almeno una riga	[if CSV_OK] [property CSV_NONEMPTY]
Rule 3	RU3_0	Regola 3 non selezionata	[property RU3_OFF][Single]
	RU3_1	Regola 3 selezionata	[property RU3_ON][Single]
Valore N-repos	N1	N-repos < 0	[if CSV_NONEMPTY and CV_ON] [Error]
	N2	N-repos = 0	[if CSV_NONEMPTY and CV_ON] [Single]
	N3	0 < N-repos < #righeProgettoCSV	[if CSV_NONEMPTY and CV_ON] [property N_OK]
	N4	N-repos > #righeProgettoCSV	[if CSV_NONEMPTY and CV_ON] [Error] [property N_GT]

3.2.3 Test Frame

ID	**Combinazioni (categorie/scelte)**	**Oracolo (risultato atteso)**

TF1	ST0	[Single] Nessuno step
TF2	ST1, CV1, IO0	[Error] IO dir non esiste
TF3	ST1, CV1, IO1, RP0	[Single] repo dir non esiste e viene creata
TF4	ST1, CV1, IO1, RP1, CSV0	[Error] CSV mancante
TF5	ST1, CV1, IO1, RP1, CSV1, CS0, RU3_0	[Single] Regola 3 non eseguita e analisi completata con successo
TF6	ST1, CV1, IO1, RP1, CSV1, CS0, RU3_1	[Single] Regola 3 eseguita e analisi completata con successo
TF7	ST1, CV1, IO1, RP1, CSV1, CS1, N1	[Error] N-repos < 0
TF8	ST1, CV1, IO1, RP1, CSV1, CS1, N2	[Single] N-repos = 0 e analisi completata con successo
TF9	ST1, CV1, IO1, RP1, CSV1, CS1, N3	Analisi completata con successo
TF10	ST1, CV1, IO1, RP1, CSV1, CS1, N4	[Error] N-repos > #righeProgettoCSV
TF11	ST1, CV2, IO1, RP1	Analisi completata con successo (Solo lo step Producer)
TF12	ST2, CV1, IO1, RP1, CSV1, CS1, N3	[Single] Tutti gli Step eseguiti ed analisi completata con successo

3.2.4 Test Case

TC	**Input**	**Descrizione ambiente**	**Oracolo**

TC1	Tutti gli Step non selezionati.		Nessuno Step eseguito. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.
TC2	Solo Cloner e Verify selezionati. Directory IO: *"nonexistent_io_directory".*	Directory IO inesistente.	Messaggio mostrato: Titolo: 'Invalid Path', Messaggio: 'IO path does not exist: ...\nonexistent_io_directory...'
TC3	Solo Cloner e Verify selezionati. Directory IO: "temp_io_structure". Directory repo: "test_repos". File CSV: "test_projects.csv".	Directory IO esiste. Directory repo inesistente. File CSV con solo header.	Creata la Directory 'test_repos'.
TC4	Solo Cloner e Verify selezionati. File CSV: *"nonexistent_projects.csv".* Directory IO: *"temp_io_structure".* Directory repo: *"repos".* File CSV*: "nonexistent_projects.csv".*	Directory IO esiste. Directory repo esiste. File CSV inesistente.	Messaggio mostrato: Titolo: Pipeline Failed Messaggio: 'Error: [Errno 2] No such file or directory:'
TC5	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"empty_projects.csv".* Regola 3 non selezionata.	Directory IO esiste. Directory repo esiste. File CSV con solo header.	Regola 3 non attiva. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.

TC6	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"empty_projects.csv".* Regola 3 selezionata.	Directory IO esiste. Directory repo esiste. File CSV con solo header.	Regola 3 attiva. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.
TC7	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"projects_TF7.csv".* Numero repos: *"-1"* .	Directory IO esiste. Directory repo esiste. File CSV contiene: "owner1/project1", "owner2/project2".	Messaggio mostrato: Titolo errore: 'Invalid Value'; Messaggio: 'N-repos cannot be negative: -1'.
TC8	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"projects_TF8.csv".* Numero repos: *"0"* .	Directory IO esiste. Directory repo esiste. File CSV contiene: "owner1/project1".	Numero repos è 0. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.
TC9	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"projects_TF9.csv".* Numero repos: *"3".*	Directory IO esiste. Directory repo esiste. File CSV contiene: "owner1/project1", "owner2/project2", "owner3/project3", "owner4/project4", "owner5/project5".	Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.
TC10	Solo Cloner e Verify selezionati. Directory IO: *"temp_io_structure".* Directory r*epo: "repos"* . File CSV: *"projects_TF10.csv".* Numero repos: *"100".*	Directory IO esiste. Directory repo esiste. File CSV contiene: "owner1/project1", "owner2/project2", "owner3/project3".	Messaggio mostrato: Titolo: 'Invalid Value Messaggio errore: N-repos (100) exceeds CSV rows (3)'.

TC11	Solo *Producer* selezionato. *Directory IO: "temp_io_structure".* *Directory repo: "repos" .*	Directory IO esiste. Directory repo esiste.	Solo lo step *Producer* abilitato. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.
TC12	Tutti gli Step selezionati. Directory IO: "temp_io_structure". Directory repo: "repos" . File CSV: "projects_TF12.csv". Numero repos: 3	Directory IO esiste. Directory repo esiste. File CSV contiene: "owner1/project1", "owner2/project2", "owner3/project3", "owner4/project4", "owner5/project5".	Tutti gli step eseguiti. Messaggio mostrato: Titolo: 'Success'; Messaggio: 'Pipeline completed successfully!'.

Nota: Nell'implementazione, al fine di semplificare la definizione e l'esecuzione dei Test Case, sono state sviluppate 6 funzioni di utilità. Tali funzioni sono state a loro volta testate, ma, per non appesantire il documento, i relativi dettagli sono stati omessi.

3.3 UC-CR3 — Consultazione delle analisi tramite dashboard

3.3.1 Use Case

Descrizione	L'utente accede alla tab Dashboard della GUI, seleziona una delle analisi disponibili e ne visualizza i risultati in forma grafica e aggregata (overview, metriche e top librerie).
Attori	Utente (ricercatore)
Entry condition	GUI aperta; utente nella tab Dashboard; interfaccia in stato Idle.
Exit condition	**Successo**: analisi selezionata visualizzata correttamente; interfaccia in stato Idle sulla tab Dashboard.

	Errore: errore imprevisto/visualizzazione fallita; interfaccia in stato Idle sulla tab Dashboard (messaggio di errore mostrato).
Flusso di eventi principale	1. Il sistema mostra l'elenco delle analisi disponibili e attende una selezione. 2. L'utente seleziona un'analisi (es. Analisi_1). 3. Il sistema visualizza: Analysis Overview (conteggi/percentuali Producer/Consumer/Producer&Consumer/non-ML), Code Quality Metrics (CC e MI medi), Top 10 ML Libraries Detected (top librerie più usate) per l'analisi selezionata.

3.3.2 Category Partition — Parametri, oggetti, categorie e scelte

Parametri:

- "`io\output\consumer`" e "`io\output\producer`": percorso contenente i rispettivamente i risultati delle analisi per producer e consumer
- "`io\output\metrics`": percorso contenente i risultati delle analisi per le metriche CC (Cyclomatic Complexity) ed MI (Maintenance Index)

Oggetti dell'ambiente:

- Filesystem: stato del filesystem

Categorie e scelte:

Categoria	ID	Descrizione	Proprietà / Vincoli
Directory producer e consumer	DPC0	Directory vuota (0 risultati per progetti producer e consumer)	[property DirProConEmp]
	DPC1	Directory con 1+ risultati per progetti producer e consumer	[property DirProMulPro]
Directory metrics	DM0	Directory vuota (0 progetti analizzati)	[property DirMetEmp]
	DM1	Directory con 1+ risultati per progetti analizzati	[property DirMetMulPro]

3.3.3 Test Frame

ID	**Combinazioni (categorie/scelte)**	**Oracolo (risultato atteso)**
TF1	DPC0, DM0	Non compare nessuna Analysis
TF2	DPC0, DM1	Compare Analysis e selezionandola mostra la media delle metriche dei progetti in input, mentre i restanti valori sono pari a 0.
TF3	DPC1, DM0	Compare Analysis e selezionandola mostra il numero di consumer e producer corrispondi, e le keywords più usate per tali progetti, mentre i restanti valori sono pari a 0.
TF4	DPC1, DM1	Compare Analysis e selezionandola mostra il numero di consumer e producer corrispondi, e le keywords più usate per tali progetti, e la media delle metriche dei progetti in input.

3.3.4 Test Case

TC	**Input**	**Descrizione ambiente**	**Oracolo**
TC1	Directory "*consumer*". Directory *"producer".* Directory "*metrics".*	Directory vuote	Lista analisi: [] . Messaggio default visibile: "No analysis selected".
TC2	Directory "*consumer*". Directory *"producer".*	Le directory "consumer" e "producer" sono vuote.	Lista analisi: ['1'].

	File CSV *"\metrics\metrics_1\metrics.csv".*	Il file CSV contiene*:* *"2knal, 4.74, 38.84",* *"5hirish, 2.8, 51.22",* *"921kiyo, 2.3, 50.41",* *"aaronlam88, 2.3, 64.56".*	Summary labels: Producer: 0, Consumer: 0, Producer & Consumer: 0 . Metrics labels: Media CC: 3.04 Media MI: 51.26 . Keywords table: [] .
TC3	*File CSV "\producer\producer_1\results.csv".* *File CSV "\consumer\consumer_1\results.csv".* Directory *"metrics".*	File CSV *"producer_1\results.csv" contiene:* *"5hirish/adam_qas, Yes, sklearn, .../classifier.py, .fit(, 68",* *"5hirish/adam_qas, Yes, sklearn, .../trainer.py, .fit(, 44",* *"921kiyo/3d-dl, Yes, keras, .../train.py, .fit_generator(, 336",* *"921kiyo/3d-dl, Yes, tensorflow, .../pipeline.py, .train., 37",* *"aaronlam88/cmpe295, Yes, sklearn, .../classifier.py, .fit(, 40".* File CSV *"\consumer\consumer_1\results.csv" contiene:* *"921kiyo/3d-dl, Yes, keras, .../flask_app.py, .predict(, 82".* La Directory "metrics" è vuota. Il progetto "921kiyo/3d-dl" è	Lista analisi: ['1'] . Summary labels: Producer = 3 Consumer = 1 Producer & Consumer = 1 . Metrics labels: Media CC = 0 Media MI = 0 . Keywords table: "sklearn, .fit(, 3 keras, .fit_generator(, 1 keras, .predict(, 1 tensorflow, .train., 1".

		presente sia come producer sia come consumer.	
TC4	*File CSV "\producer\producer_1\results.csv".* *File CSV "\consumer\consumer_1\results.csv".* *File CSV "\metrics\metrics_1\metrics.csv".*	File CSV *"producer_1\results.csv" contiene:* *"5hirish/adam_qas, Yes, sklearn, ..., .fit(, 68",* *"921kiyo/3d-dl, Yes, keras, ..., .fit_generator(, 336",* *"921kiyo/3d-dl, Yes, tensorflow, ..., .train., 37",* *"921kiyo/3d-dl, Yes, tensorflow, ..., .train., 876",* *"abojchevski/graph2gauss, Yes, tensorflow, ..., .train., 45".* File CSV *"\consumer\consumer_1\results.csv" contiene:* *"921kiyo/3d-dl, Yes, keras, ..., .predict(, 82",* *"921kiyo/3d-dl, Yes, keras, ..., .predict(, 53",* *"abojchevski/graph2gauss, Yes, tensorflow, ..., .predict_proba(, 120".* File CSV *"\metrics\metrics_1\metrics.csv" contiene:* *"5hirish, 2.8, 51.22",* *"921kiyo, 2.3, 50.41",* *"abojchevski, 4.38, 41.65".* I progetti *"921kiyo/3d-dl"* e *"abojchevski/graph2gaus*	Lista analisi: ['1'] . Summary labels: Producer = 3 Consumer = 2 Producer & Consumer = 2 . Metrics labels: Media Complexity Cyclomatic = 3.16 Media Maintainability Index = 47.76 . Keywords table contiene: "tensorflow, .train., 3 keras, .predict(, 2 keras, .fit_generator(, 1 sklearn, .fit(, 1 tensorflow, .predict_proba(, 1".

		s" sono sia producer che consumer.	

4 Regression Testing

Il regression testing è stato condotto tramite la riesecuzione completa della suite di test preesistenti, in particolare i test di sistema pre-modification.
Nel nostro caso, la suite era composta da **16 test case**, definiti sui due casi d'uso principali (Analysis 12 test case e Cloning 4 test case) e completati senza fallimenti in fase pre-modification.

Riepilogo nuova esecuzione

Totale	**Superati**	**Falliti**
16	16	0

5 Conclusioni

Le attività di Post-Modification Testing e Regression Testing hanno permesso di:

- verificare con test white-box (unit e integration) i metodi selezionati, guidati dai CFG, raggiungendo una branch coverage pari a **95%** sulle classi target;
- validare a livello di sistema le funzionalità introdotte con **CR1–CR3** tramite Category Partition (UC-CR1… UC-CR3);
- confermare l'assenza di regressioni mediante la riesecuzione della suite di pre-modifica (16 test case).

5.1 Dati finali

Al termine dell'esecuzione dei vari test, tutti i 74 casi di test previsti sono stati eseguiti con successo, senza riscontrare alcun fallimento.

Risultati finali

- Test eseguiti: 74
- Fallimenti: 0
- Incident / anomalie rilevate: nessuna

Fase	**Tipo**	**Riferimento**	**Numero**
Test eseguiti post-modifica	**Unit Test**	CR1	8
		CR2	8
		CR3	2
	Integration Test	CR1	8
		CR2	13
		CR3	7
	System Test (CR1, CR2, CR3)	CR1	6
		CR2	12
		CR3	4
Regression Testing	**System Test Pre Modification**	UC-1 Analysis	12
		UC-2 Cloning	4
Totale			**84**