Test Case Specification for Mark

a.prisco50@studenti.unisa.it 0522501976
c.ranieri7@studenti.unisa.it 0522501977

June 30, 2025

1 Version History

Versione	Data	Descrizione
0.1	28/06/2025	Stesura della struttura del documento.
1.1	28/06/2025	Stesura della sezione *Test Design per exec_analysis.py*.
1.2	28/06/2025	Stesura della sezione *Test Design per cloner.py*.
1.3	28/06/2025	Stesura della sezione *Test Design per cloning_check.py*.
1.4	28/06/2025	Stesura della sezione *Test Design per notebook_converter.py*.
1.5	28/06/2025	Stesura della sezione *Test Design per Results_Analysis.py*.
1.6	28/06/2025	Stesura della sezione *Test Design per merge.py*.

2 Riferimenti

All'interno del documento vengono citate le seguenti informazioni.

Riferimento	Descrizione
IGES_Mark_TPD	Documento relativo alla pianificazione della fase di testing.

3 Introduzione

Di seguito sono riportate tutte le informazioni relative alle specifiche necessarie all'esecuzione della fase di test.

Nota. La scelta di testare a livello funzionale solo un sottoinsieme delle funzionalità del sistema è nata dall'aver descritto i possibili test frame per ogni modulo. Motivo per cui sono descritti i test frame per ogni modulo ma sono definiti i test case solo per i moduli ritenuti fondamentali per testare le funzionalità descritte nel documento **IGES_Mark_TPD**.

Obiettivo del Documento All'interno di questo documento si tiene traccia dell'applicazione della metodologia per l'individuazione dei casi di test indicata nel documento **IGES_Mark_TPD**, dei test frame e delle specifiche dei casi di test.

Contesto del Progetto Mark è un tool scritto in Python che dato un progetto, tramite l'ausilio di una Knowledge Base e delle detection rules, può classificare progetti ML in tre diverse categorie: ML-Consumer, ML-Producer e ML-Producer e Consumer.

4 Test Design per *exec_analysis.py*

Il metodo *exec_analysis.py* accetta due parametri: una directory di output dei risultati e una directory contenente un insieme di repository (ognuno classificabile come *Producer*, *Consumer*, *Producer e Consumer*, *No Producer e no Consumer*). Per progettare i casi di test si è fatto uso della tecnica **category partition**, affiancata dalla definizione di **test frame**. Le categorie individuate, con le rispettive scelte e condizioni, sono riportate nella seguente tabella.

Parametro: Output Directory	
Categoria: Esistenza Directory (ED)	ED-1: Non esiste [*error*] ED-2: Esiste [*property ED-OK*]
Parametro: Directory contenente i repository	
Categoria: Lunghezza (LT)	LT-1: Lunghezza = 0 [*if ED-OK*] LT-2: Lunghezza = 1 [*property LT-OK*, *if ED-OK*] LT-3: Lunghezza ≥ 2 [*property LT-OK*, *if ED-OK*]
Categoria: Composizione Interna (CI)	CI-1: Presenza di Producer [*if LT-OK*] CI-2: Presenza di Consumer [*if LT-OK*] CI-3: Presenza di Producer e Consumer [*if LT-OK*] CI-4: Assenza di Producer e Consumer [*if LT-OK*]

Sulla base delle combinazioni valide tra le scelte, sono stati definiti i seguenti test frame:

ID	**Categorie (ED, LT, CI)**
EA_0	ED-1, LT-1, CI-1
EA_1	ED-2, LT-1, CI-1
EA_2	ED-2, LT-2, CI-1
EA_3	ED-2, LT-2, CI-2
EA_4	ED-2, LT-2, CI-3
EA_5	ED-2, LT-2, CI-4
EA_6	ED-2, LT-3, CI-1
EA_7	ED-2, LT-3, CI-2
EA_8	ED-2, LT-3, CI-3
EA_9	ED-2, LT-3, CI-4

Nota. È stato aggiunto un caso di test dedicato per la situazione in cui la directory di output non esiste, al fine di verificare che venga correttamente creata.

Di seguito si riportano i dettagli dei test case progettati:

ID	Input specifico	Comportamento Atteso
EA_0	Directory con path errato	Nessuna analisi; avviso che il path non esiste.
EA_1	Directory vuota	Nessuna classificazione; vengono creati solo *results_first_step.csv* e *results_consumer.csv*.
EA_2	*aaronlam88/cmpe295*	Corretto riconoscimento come *producer*.
EA_3	*keums/melodyExtraction_JDC*	Corretto riconoscimento come *consumer*.
EA_4	*appinho/SAComputerVisionMachineLearning*	Corretto riconoscimento come *producer* e *consumer*.
EA_5	*kelaberetiv/TagUI*	Corretto riconoscimento come *no-producer-consumer*.
EA_6	*aaronlam88/cmpe295*, *keums/melodyExtraction_JDC*	Producer e consumer correttamente classificati.
EA_7	*keums/melodyExtraction_JDC*, *kelaberetiv/TagUI*	Consumer e no-pro-and-con correttamente classificati.
EA_8	*appinho/...*, *aaronlam88/...*	Producer-consumer e producer correttamente classificati.
EA_9	*keums/...*, *kelaberetiv/...*, *appinho/...*, *aaronlam88/...*	Corrette classificazioni per tutti e quattro i tipi: no-pro-and-con, producer, consumer, producer-consumer.

5 Test Design per *cloner.py*

Il metodo *cloner.py* accetta come parametro un file *.csv* contenente la lista dei nomi dei repository da clonare (*selected_projects.csv*). Per progettare i casi di test si è fatto uso della tecnica **category partition**, affiancata dalla definizione di **test frame**. Le categorie individuate, con le rispettive scelte e condizioni, sono riportate nella seguente tabella.

Parametro: File *.csv* contenente la lista dei repository da clonare	
Categoria: Esistenza File (EF)	EF-1: Non esiste [*error*] EF-2: Esiste [*property EF-OK*]
Categoria: Lunghezza (LT)	LT-1: Lunghezza = 0 [*if EF-OK*] LT-2: Lunghezza = 1 [*if EF-OK*] LT-3: Lunghezza ≥ 2 [*if EF-OK*]

Sulla base delle combinazioni valide tra le scelte, sono stati definiti i seguenti test frame:

ID	**Categorie (EF, LT)**
CL_0	EF-1, LT-1
CL_1	EF-2, LT-1
CL_2	EF-2, LT-2
CL_3	EF-2, LT-3

Nota. È stato aggiunto un caso di test dedicato per la situazione in cui il file contenente i nomi dei repository da scaricare non esiste, al fine di verificare che venga gestito correttamente e venga fornito un messaggio d'errore.

Di seguito si riportano i dettagli dei test case progettati:

ID	**Input specifico**	**Comportamento Atteso**
CL_0	Path del file *selected_projects.csv* non esistente.	Nessuna clonazione. Il sistema deve segnalare che il path indicato non esiste.
CL_1	File *selected_projects.csv* esistente ma vuoto.	Nessuna clonazione. Il sistema riconosce correttamente l'assenza di repository.
CL_2	Contenuto del file: ProjectName *921kiyo/3d-dl*	Nessuna clonazione. Il sistema ignora il repository oppure segnala l'impossibilità di eseguire la clonazione.
CL_3	Contenuto del file: ProjectName *aaronlam88/cmpe295* *abdullahselek/koolsla* *abreheret/PixelAnnotationTool*	Nessuna clonazione. Il sistema interpreta correttamente il file ma non esegue alcuna clonazione automatica.

6 Test Design per *cloning_check.py*

Il metodo *cloning_check.py* accetta due parametri: una directory dove sono presenti le copie delle repository clonate e un file .csv contenente la lista dei nomi delle repository da controllare che siano state effettivamente clonate (*applied_projects.csv*). Per progettare i casi di test si è fatto uso della tecnica **category partition**, affiancata dalla definizione di **test frame**. Le categorie individuate, con le rispettive scelte e condizioni, sono riportate nella seguente tabella.

Parametro: Directory contenente le copie delle repository	
Categoria: Esistenza Directory (ED)	ED-1: Non esiste [*error*] ED-2: Esiste [*property ED-OK*]
Categoria: Lunghezza (LD)	LD-1: Lunghezza = 0 [*if ED-OK*] LD-2: Lunghezza = 1 [*if ED-OK*] LD-3: Lunghezza ≥ 2 [*if ED-OK*]
Parametro: File .csv contenente la lista dei repository da controllare	
Categoria: Esistenza File (EF)	EF-1: Non esiste [*error*, *if ED-OK*] EF-2: Esiste [*property EF-OK*, *if ED-OK*]
Categoria: Lunghezza (LT)	LT-1: Lunghezza = 0 [*if EF-OK*] LT-2: Lunghezza = 1 [*if EF-OK*] LT-3: Lunghezza ≥ 2 [*if EF-OK*]

Sulla base delle combinazioni valide tra le scelte, sono stati definiti i seguenti test frame:

ID	Categorie (ED, LD, EF, LT)
CLCHK_0	ED-1, LD-1, EF-2, LT-1
CLCHK_1	ED-2, LD-1, EF-1, LT-1
CLCHK_2	ED-2, LD-1, EF-2, LT-1
CLCHK_3	ED-2, LD-1, EF-2, LT-2
CLCHK_4	ED-2, LD-1, EF-2, LT-3
CLCHK_5	ED-2, LD-2, EF-2, LT-1

ID	Categorie (ED, LD, EF, LT)
CLCHK_6	ED-2, LD-2, EF-2, LT-2
CLCHK_7	ED-2, LD-2, EF-2, LT-3
CLCHK_8	ED-2, LD-3, EF-2, LT-1
CLCHK_9	ED-2, LD-3, EF-2, LT-2
CLCHK_10	ED-2, LD-3, EF-2, LT-3

7 Test Design per *notebook_converter.py*

Il metodo *notebook_converter.py* accetta come parametro una directory contenente le repository in cui convertire i file *.ipynb* in *.py*. Per progettare i casi di test si è fatto uso della tecnica **category partition**, affiancata dalla definizione di **test frame**. Le categorie individuate, con le rispettive scelte e condizioni, sono riportate nella seguente tabella.

Parametro: Directory contenente le repository	
Categoria: Esistenza Directory (EF)	EF-1: Non esiste [*error*] EF-2: Esiste [*property EF-OK*]
Categoria: Lunghezza (LT)	LT-1: Lunghezza = 0 [*if EF-OK*] LT-2: Lunghezza = 1 [*if EF-OK*] LT-3: Lunghezza ≥ 2 [*if EF-OK*]

Sulla base delle combinazioni valide tra le scelte, sono stati definiti i seguenti test frame:

ID	**Categorie (EF, LT)**
NC_0	EF-1, LT-1
NC_1	EF-2, LT-1
NC_2	EF-2, LT-2
NC_3	EF-2, LT-3

Nota. Nel caso EF-1.LT-1, il sistema termina l'esecuzione con un messaggio di errore che indica la mancanza della directory.

8 Test Design per *Results_Analysis.py*

Per la progettazione dei casi di test si è adottata la tecnica **category partition**, in combinazione con la costruzione di **test frame**, tenendo conto dell'assenza di controlli sui file e directory in input. Le categorie individuate sono riportate nella seguente tabella.

Parametro: File oracolo per producer/consumer	
Categoria: Esistenza File (EF)	EF-1: Non esiste [*error*] EF-2: Esiste [*property EF-OK*]
Categoria: Lunghezza Oracolo (LO)	LO-1: Lunghezza = 0 [*if EF-OK*] LO-2: Lunghezza = 1 [*if EF-OK*] LO-3: Lunghezza ≥ 2 [*if EF-OK*]
Parametro: Directory contenente i file di analisi producer/consumer	
Categoria: Esistenza Directory (ED)	ED-1: Non esiste [*error*, *if EF-OK*] ED-2: Esiste [*property ED-OK*, *if EF-OK*]
Categoria: Lunghezza (LT)	LT-1: Lunghezza = 0 [*if ED-OK*] LT-2: Lunghezza = 1 [*if ED-OK*] LT-3: Lunghezza ≥ 2 [*if ED-OK*]

Sulla base delle combinazioni valide, sono stati definiti i seguenti test frame:

ID	Categorie (EF, LO, ED, LT)
RA_0	EF-1, LO-1, ED-2, LT-1
RA_1	EF-2, LO-1, ED-1, LT-1
RA_2	EF-2, LO-1, ED-2, LT-1
RA_3	EF-2, LO-1, ED-2, LT-2
RA_4	EF-2, LO-1, ED-2, LT-3
RA_5	EF-2, LO-2, ED-2, LT-1

ID	Categorie (EF, LO, ED, LT)
RA_6	EF-2, LO-2, ED-2, LT-2
RA_7	EF-2, LO-2, ED-2, LT-3
RA_8	EF-2, LO-3, ED-2, LT-1
RA_9	EF-2, LO-3, ED-2, LT-2
RA_10	EF-2, LO-3, ED-2, LT-3

Nota. Il sistema non implementa controlli sull'esistenza dei file o directory di input: pertanto, i casi **RA_0** (file oracolo mancante) e **RA_1** (directory mancante) causano il crash del sistema.

Nota. Lo script *Results_Analysis.py* è composto dalla ripetizione di blocchi di codice per la generazione dei file *result_consumer_X.csv* (5 blocchi) e *result_producer_X.csv* (3 blocchi).
Nota. Per la generazione dei file *result_consumer_X.csv* vengono utilizzati i dati contenuti nel file *oracle_consumer.csv* e nei file della directory *Consumer/Consumer_X*. Analogamente, i file *result_producer_X.csv* vengono generati utilizzando *oracle_producer.csv* e i file della directory *Producers/Producers_X*.

9 Test Design per *merge.py*

Per la progettazione dei casi di test si è adottata la tecnica **category partition**, in combinazione con la costruzione di **test frame**, tenendo conto dell'assenza di controlli sui file e directory in input. Le categorie individuate sono riportate nella seguente tabella.

Parametro: File oracolo per producer/consumer	
Categoria: Esistenza File (EO)	EO-1: Non esiste [*error*] EO-2: Esiste [*property EO-OK*]
Categoria: Lunghezza Oracolo (LO)	LO-1: Lunghezza = 0 [*if EO-OK*] LO-2: Lunghezza = 1 [*if EO-OK*] LO-3: Lunghezza ≥ 2 [*if EO-OK*]
Parametro: File dei risultati dell'analisi per producer/consumer	
Categoria: Esistenza File (EA)	EA-1: Non esiste [*error*, *if EO-OK*] EA-2: Esiste [*if EO-OK*]
Categoria: Lunghezza Risultati (LT)	LT-1: Lunghezza = 0 [*if EA-2*] LT-2: Lunghezza = 1 [*if EA-2*] LT-3: Lunghezza ≥ 2 [*if EA-2*]

Sulla base delle combinazioni valide, sono stati definiti i seguenti test frame:

ID	Categorie (EO, LO, EA, LT)
MG_0	EO-1, LO-1, EA-2, LT-1
MG_1	EO-2, LO-1, EA-1, LT-1
MG_2	EO-2, LO-1, EA-2, LT-1
MG_3	EO-2, LO-1, EA-2, LT-2
MG_4	EO-2, LO-1, EA-2, LT-3
MG_5	EO-2, LO-2, EA-2, LT-1

ID	Categorie (EO, LO, EA, LT)
MG_6	EO-2, LO-2, EA-2, LT-2
MG_7	EO-2, LO-2, EA-2, LT-3
MG_8	EO-2, LO-3, EA-2, LT-1
MG_9	EO-2, LO-3, EA-2, LT-2
MG_10	EO-2, LO-3, EA-2, LT-3

Nota. I casi di test MG_0 e MG_1 causano il crash del sistema a causa della mancanza rispettivamente del file oracolo o del file dei risultati dell'analisi.

Nota. Il file *merge.py* è composto dalla replicazione di 2 blocchi di codice in sequenza, uno per i producer e uno per i consumer. In entrambi i casi vengono utilizzati come file di input: per il producer *results_first_step.csv* e *oracle_producer.csv*, mentre per il consumer *results_consumer.csv* e *oracle_consumer.csv*. Poiché i blocchi utilizzano lo stesso codice, si è deciso di riutilizzare la medesima struttura dei casi di test per entrambi.