Test Plan Document 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	27/06/2025	Stesura della struttura del documento.
1.0	27/06/2025	Prima stesura del documento.

2 Riferimenti

All'interno del documento vengono citate le seguenti informazioni.

Riferimento	Descrizione
IGES_Mark_CCR	Documento relativo alla comprensione del codice in cui sono stati identificati una serie di problemi relativi ai dettagli tecnici del sistema.
IGES_Mark_RER	Documento relativo alla fase di reverse engineering.

3 Introduzione

Di seguito sono riportate tutte le informazioni relative alla pianificazione del piano di testing.

Obiettivo del Documento All'interno di questo documento saranno trattate tutte le scelte relative alla pianificazione della fase di testing.

La prima decisione che è stata presa è quella di suddividere la fase di pianificazione del testing in tre fasi. Una prima fase di pianificazione inerente al sistema prima di applicare il processo di reengineering, una seconda fase di pianificazione inerente al sistema dopo aver applicato il processo di reengineering, entrambe necessarie alla risoluzione della CR_1, ed infine un'ulteriore fase di pianificazione inerente al sistema dopo aver applicato il processo di additive reengineering necessaria per la risoluzione della CR_2. Questa scelta è data dall'avere obiettivi differenti durante le tre fasi di testing. Di seguito sono riportate più nel dettaglio ognuna di esse.

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 Pianificazione della Fase di Testing Prima della Fase di Reengineering

Prima di applicare il processo di reengineering del sistema, come riportato nella CR_1, è necessario comprendere l'attuale stato di quest'ultimo sia dal punto di vista funzionale che strutturale. Di seguito sono quindi riportate due sezioni atte a dettagliare i processi di pianificazione per comprendere l'attuale stato del sistema. Non è necessario applicare una fase di testing di regressione in quanto non sono previste modifiche al sistema durante questa fase.

Testing funzionale L'obiettivo della seguente fase di testing funzionale pianificata è quello di accertarsi che le funzionalità del sistema soddisfino i requisiti specificati. Questo processo ha richiesto quindi un'ulteriore fase di reverse engineering riportata nel documento **IGES_Mark_RER** necessaria al recupero delle informazioni relative ai requisiti funzionali. Tra tutte le funzionalità identificate nel documento **IGES_Mark_RER** ci si è concentrati sul testing funzionale unicamente del requisito RF_4, e quindi del modulo *exec_analysis.py*, questo per diversi motivi definiti di seguito. Il primo motivo è dato dalla volontà di dedicare una maggiore attenzione alle fasi di testing successive, inoltre il sistema è stato ampiamente utilizzato durante le prime fasi per comprenderne obiettivi e funzionalità, per cui anche se tale processo non può essere considerato sistematico ha permesso di accertarsi del normale funzionamento dei vari moduli. Le considerazioni sono state fatte sulla base degli oracoli descritti nel documento **IGES_Mark_CCR** e quindi per i 341 progetti definiti nel file selected_projects.csv. Infine è stato preso in considerazione il fatto che tale sistema è già stato utilizzato in ambito di ricerca, motivo per cui dal punto di vista funzionale ci si aspetta che sia stata già effettuata una fase di testing. La tecnica scelta per questa fase di testing funzionale è il **category partition**, mentre a livello di implementazione dei casi di test si è scelto di utilizzare **unittest** per simulare l'interazione con il sistema da parte di un utente.

Testing strutturale L'obiettivo della seguente fase di testing strutturale è quello di ottenere maggiori informazioni sullo stato del sistema da poter sfruttare durante la fase di reengineering. In particolare l'obiettivo che ci si pone è quello di identificare codice morto ed identificare le parti del sistema da attenzionare. Gli elementi da testare sono tutti i moduli del sistema. Si considera tale fase di testing conclusa solo quando la branch coverage media, e quindi dell'intero sistema, è pari almeno all'80%. Tale metrica è descritta da una scala rateo. È definita quindi la metrica **branch_coverage_media** come strumento per tenere traccia della branch coverage media del sistema e quindi dello stato di avanzamento della fase di testing descritta. La tecnica scelta è, come detto in precedenza, quella della branch coverage e a livello implementativo è stato scelto di utilizzare un tool per Python chiamato **Coverage.py**.

5 Pianificazione della Fase di Testing Dopo la Fase di Reengineering

Dopo aver applicato il processo di reengineering del sistema, come riportato nella CR_1, è necessario comprendere l'impatto che tale modifica può aver avuto sul sistema sia dal punto di vista funzionale che strutturale. Di seguito sono quindi riportate tre sezioni atte a dettagliare i processi di pianificazione per comprendere l'attuale stato del sistema. L'ultima sezione è inerente alla fase di testing di regressione in quanto a differenza della fase precedente sono previste modifiche al sistema.

Testing funzionale L'obiettivo della seguente fase di testing funzionale pianificata è quello di accertarsi che le funzionalità del sistema soddisfino i requisiti specificati anche dopo la modifica al sistema. Tra tutte le funzionalità identificate nel documento **IGES_Mark_RER** ci si è concentrati sul testing funzionale unicamente dei requisiti RF_4 e RF_1, e quindi del modulo *exec_analysis.py* e *cloner.py*. La tecnica scelta per questa fase di testing funzionale è il **category partition**, mentre a livello di implementazione dei casi di test si è scelto di utilizzare **unittest** per simulare l'interazione con il sistema da parte di un utente.

Nota *È possibile che non tutti i casi di test già definiti per la fase di testing funzionale prima della fase di reengineering del sistema possano essere riutilizzati direttamente. È possibile quindi che un certo numero di casi di test debba essere ristrutturato per potersi adattare al cambio di paradigma utilizzato per la fase di reengineering.*

Testing strutturale L'obiettivo della seguente fase di testing strutturale è quello di ottenere maggiori informazioni sullo stato del sistema. In particolare l'obiettivo che ci si pone è quello di identificare codice morto ed identificare le parti del sistema da attenzionare. Gli elementi da testare sono tutti i moduli del sistema. Si considera tale fase di testing conclusa solo quando la branch coverage media, e quindi dell'intero sistema, è pari almeno all'80%. Tale metrica è descritta da una scala rateo. È definita quindi la metrica **branch_coverage_media** come strumento per tenere traccia della branch coverage media del sistema e quindi dello stato di avanzamento della fase di testing descritta. La tecnica scelta è, come detto in precedenza, quella della branch coverage e a livello implementativo è stato scelto di utilizzare un tool per Python chiamato **Coverage.py**.

Testing di regressione L'obiettivo della seguente fase di testing di regressione è quello di assicurarsi che il sistema continui a funzionare correttamente dopo ogni modifica applicata durante la fase di reengineering. In questo caso il processo è visto come un processo iterativo da ripetere dopo che un modulo è stato reingegnerizzato. Ad ogni iterazione verranno quindi testati, a livello strutturale, tutti i moduli che ancora non hanno subito una modifica, mentre a livello funzionale verranno testati i moduli citati nel paragrafo sul testing funzionali di questa sezione. L'ordine con cui verranno affrontate le modifiche, e di conseguenza i test, coincide con l'ordine riportato dal pipeline diagram descritto nel documento **IGES_Mark_RER**. Si considera tale fase di testing conclusa solo quando sono state portate a termine tutte le iterazioni necessarie ad assicurarsi che ogni modifica non abbia introdotto nuovi errori nel sistema. Il numero di iterazioni da eseguire è pari al numero di moduli di cui il sistema, prima del reengineering si compone. È definita quindi la metrica **#numero_iterazioni** come strumento per tenere traccia dello stato di avanzamento di questa fase di testing. Tale metrica è descritta da una scala assoluta. Il testing di regressione è eseguito sia a livello funzionale che strutturale attraverso i processi descritti nei paragrafi precedenti. La tecnica scelta è quindi quella del **test all** a livello strutturale dato che le dimensioni del sistema permettono questo approccio.

6 Pianificazione della Fase di Testing dopo il Processo di Additive Reengineering

Dopo aver applicato il processo di additive reengineering necessario alla risoluzione dalla CR_2 è necessario assicurarsi che il sistema soddisfi i requisiti funzionali descritti proprio da quest'ultima. Di seguito è quindi definita una fase di testing funzionale necessaria a verificare ciò.

Testing funzionale L'obiettivo della seguente fase di testing funzionale è quello di assicurarsi che il nuovo requisito funzionale introdotto dalla CR_2 sia soddisfatto dal sistema e che la GUI permetta di eseguire tutte le funzionalità che il sistema già metteva a disposizione. Tra tutte le funzionalità identificate nel documento **IGES_Mark_RER** ci si è concentrati sul testing funzionale unicamente dei requisiti RF_4 e RF_1, e quindi del modulo *exec_analysis.py* e *cloner.py*. La tecnica scelta per questa fase di testing funzionale è il **category partition**, mentre a livello di implementazione dei casi di test si è scelto di utilizzare **unittest** per simulare l'interazione con il sistema da parte di un utente.