Università degli Studi di Salerno

Corso di Ingegneria, Gestione ed Evoluzione Software

TEMEVI

Test Metrics Visualisation

IMPACT ANALYSIS

Versione 1.0

TOP MANAGER:

Prof. Andrea De Lucia

PARTECIPANTI:

Giosuè Sulipano - 0522500845

Gianluca di Lillo - 0522500843

Revision History

Data	Versione	Descrizione	Autore
08/01/2020	1.0	Stesura del documento	Giosuè Sulipano Gianluca di Lillo
31/01/2020	1.0	Revisione dei capitoli 5 e 6	Giosuè Sulipano Gianluca di Lillo

Indice

1. Scopo del documento

In questo documento saranno descritti gli obiettivi del processo di aggiunta di funzionalità, in riferimento al documento di “Change Request”.

Verrà analizzato l’impatto di queste modifiche sul resto del sistema, analizzando i possibili rischi collegati ad esse.

Infine verrà fatto uno studio di fattibilità e verranno pianificate le fasi successive di progettazione, implementazione e testing.

2. Panoramica del sistema attuale

Il sistema attuale, sviluppato in Java 8 utilizzando l'IDE IntelliJ IDEA, permette di estrarre e visualizzare allo sviluppatore i fattori legati ai test trattati. L'interfaccia comprende la visualizzazione dei fattori relativi sia al progetto sia a ogni classe di test; la lista di queste ultime può, inoltre, essere filtrata per visualizzare solo le classi affette da un determinato Test Smell. Colori diversi applicati al nome delle classi nella lista indicano il grado di criticità della classe relativo alla presenza di Test Smell. Nel momento in cui si clicca su una classe per visualizzare i valori dei fattori ad essa relativi, nell'interfaccia viene mostrato anche il grafico dell'andamento delle metriche strutturali di ogni Test Smell, con la possibilità di filtrarlo per periodo.

Le soglie di rilevamento e di guardia dei Test Smell, inoltre, possono essere modificate dallo sviluppatore attraverso la schermata di configurazione del plugin.

2.1 Requisiti funzionali

RF 1 - Identificazione delle classi affette da Test Smell

Il sistema permette di rilevare ed etichettare le classi di test affette da Test Smell.

RF1.1: Il sistema permette di identificare le classi affette dal Test Smell "Assertion Roulette".

RF 1.2: Il sistema permette di identificare le classi affette dal Test Smell "Eager Test".

RF 1.3: Il sistema permette di identificare le classi affette dal Test Smell "Sensitive Equality".

RF 1.4: Il sistema permette di identificare le classi affette dal Test Smell "General Fixture".

RF 1.5: Il sistema permette di identificare le classi affette dal Test Smell "Mystery Guest".

RF 1.6: Il sistema permette di identificare le classi affette dal Test Smell "Resource Optimism".

RF 1.7: Il sistema permette di identificare le classi affette dal Test Smell "Indirect Testing".

RF 2 - Visualizzazione dei risultati dell'analisi

Il sistema permette di visualizzare i risultati dell'analisi effettuata.

RF2.1: Il sistema permette di visualizzare i test-related factors statici del progetto, tra cui diverse metriche strutturali e numero di classi affette da Test Smell.

RF 2.2: Il sistema permette di visualizzare i test-related factors dinamici del progetto, tra cui Line Coverage e Branch Coverage.

RF 2.3: Il sistema permette di visualizzare i test-related factors statici relativi a ogni singola classe di test, tra cui diverse metriche strutturali e Test Smell rilevati.

RF 2.4: Il sistema permette di visualizzare i test-related factors dinamici relativi a ogni singola classe di test, tra cui Line Coverage e Branch Coverage

RF2.5: Il sistema permette di visualizzare l'andamento nel tempo del valore delle metriche strutturali relative a ciascun Test Smell.

RF 2.6: Il sistema permette di distinguere le classi di test per criticità relativa alla presenza di Test Smell.

RF 3 - Filtraggio dei risultati

Il sistema permette di filtrare i risultati dell'analisi per fornirne una vista più specifica.

RF 3.1: Il sistema permette di filtrare i risultati dell'analisi in base al Test Smell che si vuole analizzare.

RF 3.2: Il sistema permette di filtrare l'andamento delle metriche strutturali relative a ogni Test Smell a partire da un preciso momento nel tempo.

RF 4 - Modifica soglie di rilevamento e di guardia dei Test Smell

Il sistema permette di modificare la configurazione delle soglie di rilevamento e di guardia dei Test Smell.

RF 4.1: Il sistema permette di modificare le soglie di rilevamento e di guardia per ogni metrica relativa a ciascun Test Smell.

RF 4.2: Il sistema permette di ripristinare le impostazioni predefinite riguardanti le soglie di rilevamento e di guardia dei Test Smell.

2.2 Design e implementazione

Il sistema è stato suddiviso fondamentalmente in due *layer*:

1. **GUI Layer**: Rappresenta il livello più alto dell'applicazione, si occupa principalmente dell'interazione con l'utente e della visualizzazione dei contenuti. Si compone dei seguenti sottosistemi:

- *analysisResults* si occupa di una delle funzionanlità più importanti del plugin, ovvero la visualizzazione semplice ed intuitiva dei risultati dell'analisi tramite interfacce che utilizzano la libreria Java Swing. Inoltre questo sottosistema sfrutta la libreria esterna XChart per creare i grafici relativi all'evoluzione delle metriche strutturali di ogni Test Smell.

- *configUI* ha lo scopo di gestire l'interfaccia utente per quanto riguarda la pagina di configurazione del plugin, in cui l'utente può modificare le soglie di rilevamento e di guardia di ogni metrica strutturale.

2. **Application Layer**: Rappresenta la logica del plugin. Esso contiene i sottosistemi che si occupano di ricavare le informazioni dell'analisi sui test-related factors trattati:

- *actions* rappresenta il cuore dell'integrazione in IntelliJ del plugin, poiché fornisce il modo di eseguire delle azioni nel momento in cui l'utente richiede l'avvio del plugin cliccando sulla relativa voce nei menù a tendina dell'IDE. Esso contiene due classi fondamentali: la prima, PluginInit, ha il compito di inizializzare il plugin a seguito dell'avvio da parte dell'utente e di chiamare le componenti dello static e dynamic Processor per ottenere i dati dell'analisi, per poi passarli al sottosistema analysisResults. La seconda, PluginConfig, si occupa di inizializzare la pagina di configurazione del plugin, ottenendo la configurazione corrente dal configurationManager e passandola al sottosistema configUI.

- *staticProcessor* ha l'obiettivo di estrapolare gli static factors relativi ai test supportati dal plugin, ovvero le metriche strutturali trattate e la presenza di Test Smell.

- *dynamicProcessor* ha lo scopo di calcolare i fattori dinamici relativi ai test, eseguendoli ed estraendo delle metriche tra cui Line Coverage e Branch Coverage. L'esecuzione dei test e il calcolo di Line Coverage e Branch Coverage per ogni classe di test viene effettuata utilizzando il tool esterno **COBERTURA**: in particolare,

se il progetto è Maven viene eseguito il plugin Cobertura integrato in Maven stesso; in caso contrario, vengono eseguite varie operazioni manualmente per permettere a Cobertura di effettuare la sua analisi, tra cui "Classes Instrumentation", "Tests Execution" e "Report Generation".
Fondamentale è la situazione in cui il progetto non sia Maven: in questo caso, l'analisi dei fattori dinamici può essere compiuta solo su classi di test compatibili con JUnit4.

- *configurationManager* si occupa di ottenere la configurazione corrente relativa alle soglie di rilevamento e di guardia dei Test Smell leggendo il file di configurazione.

- *reportManager* si occupa della lettura dei file di report per ottenere i valori storici di Line Coverage, Branch Coverage e delle metriche strutturali di ogni Test Smell relativi ad esecuzioni precedenti del plugin, a partire da un mese ed un anno specificati dall'utente.

3. Analisi della modifica richiesta

La change request effettuata prevede che vengano conservati tutti i requisiti funzionali attuali. La modifica sostanziale riguarda l'aggiunta di nuove funzionalità e metriche, ovvero il supporto alla Mutation Coverage, alla rilevazione dei Flaky Tests e l'aggiunta di informazioni nell'interfaccia grafica. Inoltre prevede la correzione del problema che riguarda il calcolo di Line Coverage e Branch Coverage.

4. Individuazione della soluzione progettuale

4.1 Problematiche affrontate

Issue 1: Come si intende gestire la problematica relativa al lungo tempo di esecuzione del calcolo della Mutation Coverage?

Proposal 1.1
Introdurre un valore di timeout, superato il quale viene terminata l'analisi della classe considerata e si procede all'analisi delle successive.
Proposal 1.2
Introdurre un valore di timeout, superato il quale viene proposto all'utente se continuare o meno ad eseguire l'analisi.

Criterion 1.1
Evitare l'interruzione dell'analisi, diminuire costi di manutenzione.
Criterion 1.2
Maggiore libertà all'utente.

Argument 1.1
L'analisi continua in modo che l'utente possa visualizzare i relativi risultati. Potrebbero, però, esserci molte classi per cui la Mutation Coverage non è stata calcolata, rendendo necessario il riavvio dell'analisi. Il vantaggio è che l'utente può visualizzare i risultati relativi ad altre metriche, ma se il numero di fallimenti nel calcolo della Mutation Coverage è elevato, quasi sicuramente dovrà effettuarne un'altra subito dopo.
Argument 1.2
La possibilità di interrompere l'analisi comporta un maggior costo di manutenzione, ma potrebbe essere vantaggioso per l'utente, poiché gli viene data più libertà nel decidere se voler vedere lo stesso i risultati dell'analisi, o riavviarla inserendo un valore di timeout più largo.

Resolution 1.1
Analizzando le argomentazioni Argument 1.1 e Argument 1.2, si è deciso di risolvere l'Issue 1 utilizzando la Proposal 1.1.

Issue 2: Come si vuole gestire l'esecuzione dell'analisi delle nuove metriche?

Proposal 2.1
Viene effettuata l'estrazione di tutte le metriche automaticamente all'avvio del plugin.
Proposal 2.2
Viene offerta allo sviluppatore la possibilità di scegliere all'avvio del plugin quali informazioni analizzare.

Criterion 2.1
Rendere disponibili subito tutte le informazioni dell'analisi.
Criterion 2.2
Diminuire i tempi di caricamento all'avvio.

Argument 2.1
Lo sviluppatore, in questo modo, ha a disposizione tutte le informazioni relative all'analisi delle classi, a discapito di un maggior tempo di attesa all'avvio.
Argument 2.2
Il calcolo di alcune metriche, soprattutto la Mutation Coverage, potrebbe impiegare molto tempo per essere calcolate e non interessare allo sviluppatore. Renderle opzionali diminuirebbe di molto i tempi di caricamento iniziali.

Resolution 2.1
Analizzando le argomentazioni Argument 2.1 e Argument 2.2, si è deciso di risolvere l'Issue 2 utilizzando la Proposal 2.2.

4.2 Soluzione individuata

La soluzione individuata consiste nell'unione di tutte le "Resolution" ottenute nel capitolo 4.1. In dettaglio, essa dovrà avere le seguenti caratteristiche:

1. Introdurre una schermata all'avvio che consente di selezionare le tipologie di metriche da calcolare.
2. Nell'implementazione del calcolo della Mutation Coverage, introdurre un valore di timeout e dare la possibilità all'utente di interrompere l'analisi in caso di superamento di esso.

Inoltre, per quanto riguarda la correzione del problema relativo al calcolo di Line e Branch Coverage a causa di un bug del tool COBERTURA, si è deciso di sostituirlo col tool JACOCO, il quale risulta essere tuttora supportato.

4.3 Soluzioni alternative

Le soluzioni alternative sono rappresentate da tutte le proposte e argomentazioni effettuate nel capitolo 4.1, ma che non sono state selezionate per la soluzione adottata.

5. Individuazione dell'Impact Set

La soluzione individuata ha coinvolto diversi artefatti del sistema attuale. La tabella che segue descrive il Candidate Impact Set individuato, ovvero l'insieme degli artefatti che verranno coinvolti nelle modifiche in fase di manutenzione. Inoltre, ad ogni artefatto verrà associato un livello di impatto che indica il modo in cui la modifica va ad impattare sul sistema attuale.

Per individuare le componenti del sistema impattate si utilizzerà un approccio top-down, ovvero a partire dai documenti di alto livello si andrà ad individuare gli artefatti di più basso livello che andranno ad essere modificati durante la fase di manutenzione.

La change request prevede, tra le altre cose, l'aggiunta di nuovi requisiti funzionali, pertanto sicuramente andrà ad impattare il documento di analisi e specifica dei requisiti.

Per quanto riguarda il design di sistema non ci saranno cambiamenti dal momento che l'architettura del sistema attuale rispecchia quella che sarà l'architettura del sistema futuro.

Le funzionalità da introdurre, descritte nel documento di Change Request, vanno ad impattare gli stessi artefatti e prevedono:

1. l'introduzione di un'interfaccia grafica all'avvio per la selezione delle metriche da analizzare, tra cui le nuove;
2. la modifica dell'interfaccia grafica relativa ai risultati, con l'aggiunta dei risultati delle metriche introdotte.

Inoltre, l'introduzione di queste nuove metriche prevede la creazione delle seguenti classi:

- MutationCoverageProcessor
- FlakyTestsProcessor
- PluginInitGUI
- FlakyTestsInfo
- MutationCoverageInfo

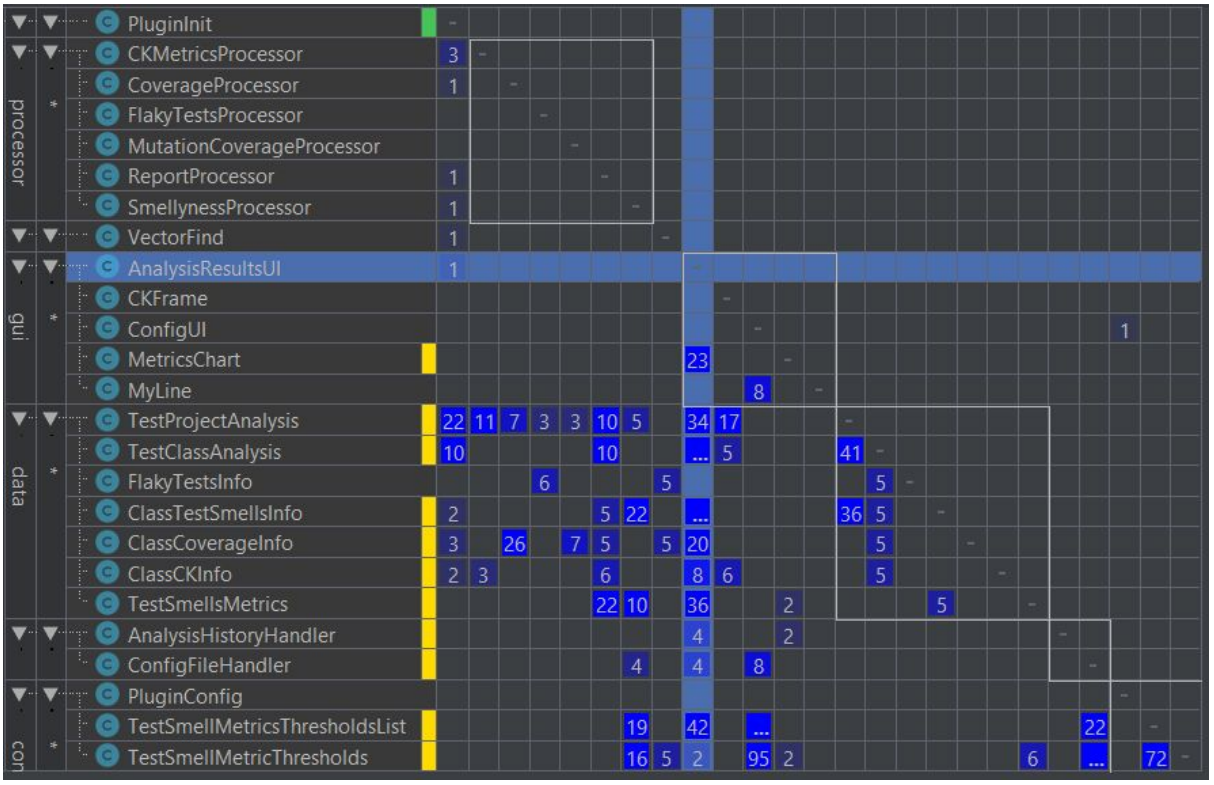
Starting Impact Set per le CR #1 e #2

Per una prima stima delle componenti impattate da queste due modifiche, il team ha stabilito uno Starting Impact Set tramite concept analysis. In seguito ha fatto utilizzo del tool DSM presente in Intellij IDEA, il quale permettere di analizzare la matrice delle dipendenze, per trovare eventuali ripple effects a partire dalle classi presenti nello Starting Impact Set.

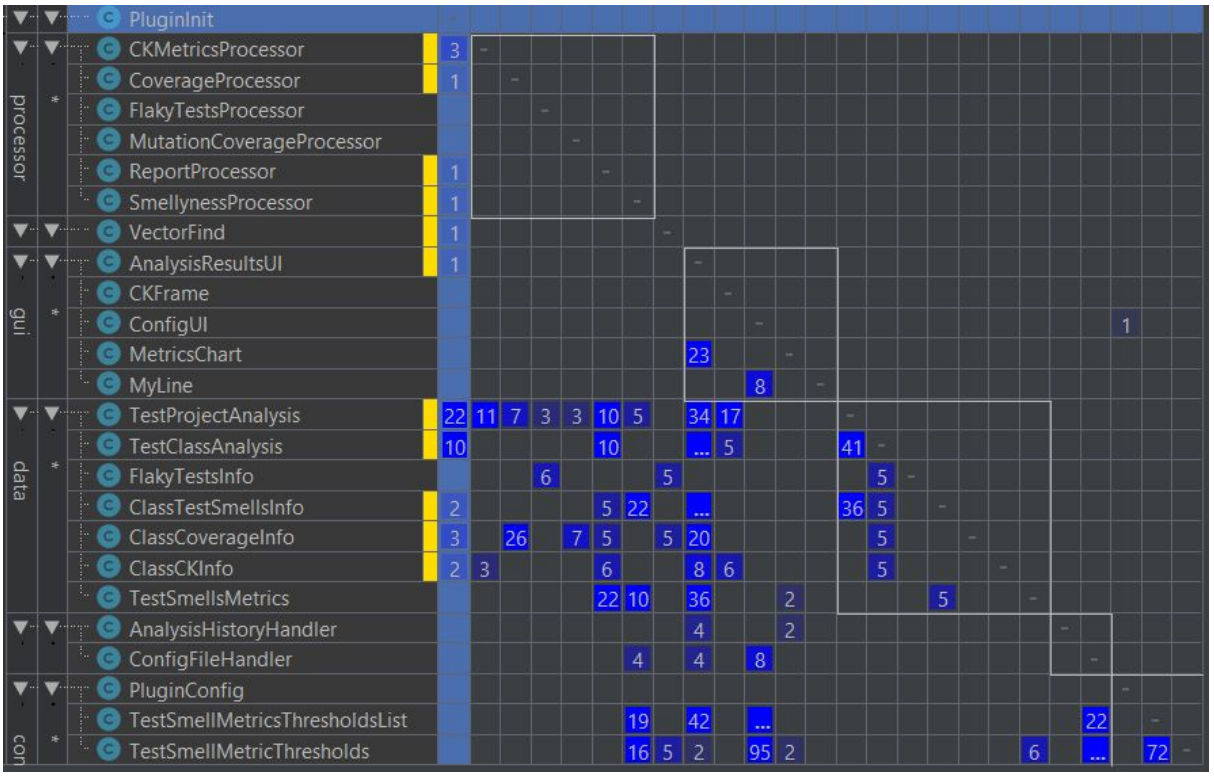
Il SIS trovato è caratterizzato dalle seguenti classi:

- AnalysisResultsUI (package GUI)
- ConfigUI (package GUI)
- PluginInit (package init)
- TestClassAnalysis (package data)

Analisi di Ripple Effects e Candidate Impact Set per le CR #1 e #2



Matrice delle dipendenze 1.1



Matrice delle dipendenze 1.2

1. La classe che si occupa della realizzazione dell'interfaccia grafica, com'è possibile notare nella Matrice 1.1, ha particolari dipendenze con le classi in giallo, dalle quali recupera le metriche da visualizzare nei risultati. Essa viene utilizzata dalla classe PluginInit (in verde), la quale è stata già identificata nell'Impact Set poichè sarà modificata come segue nel punto 2.

2. Attualmente, come mostrato anche dalla Matrice delle dipendenze 1.2, il plugin non permette la selezione delle metriche da analizzare: infatti, la classe PluginInit ha una dipendenza diretta con tutte le classi processor (in giallo). L'introduzione di un'interfaccia per la selezione delle metriche da analizzare all'avvio, andrà a porsi tra la classe PluginInit e quelle in giallo utilizzate da quest'ultima.

3. Nel momento in cui si va a modificare la classe TestClassAnalysis, poiché essa viene utilizzata dalla classe ReportManager, verrà modificata anche quest'ultima per riflettere l'aggiunta delle nuove metriche nella scrittura del report.

Dunque, è stato trovato un singolo ripple effect a partire dalle classi nello Starting Impact Set. Il **CIS**, dunque, sarà costituito dagli artefatti individuati nel **SIS** più l'artefatto ReportManager.java.

Di seguito è riportata la tabella che mostra gli artefatti impattati individuati nel **CIS**; per ogni artefatto è mostrato un livello d'impatto rappresentabile in tre diverse categorie:

- **FORTE:** se le modifiche richieste sono fortemente impattanti o se l'artefatto deve essere completamente sostituito;
- **MEDIO:** se le modifiche richieste saranno sostanziali, senza però andare a modificare la struttura dell'artefatto in maniera eccessiva;
- **DEBOLE:** se saranno necessarie solo modifiche marginali.

Artefatto	Impatto	Descrizione
Requisiti Funzionali	MEDIO	Poiché sono state aggiunte delle funzionalità, di conseguenza devono essere introdotti nuovi requisiti funzionali
Casi d'uso	MEDIO	Si dovranno aggiungere gli UC relativi ai RF introdotti e modificare gli UC esistenti relativi all'avvio del sistema.

Scendendo più basso livello troviamo le classi che compongono il sistema e quelle impattate dalle modifiche sono elencate nella tabella in basso.

Package "data": Contiene le classi che memorizzano i dati relativi ai risultati delle analisi effettuate.

Artefatto	Impatto	Descrizione
TestClassAnalysis.java	DEBOLE	La classe addetta a memorizzare i dati relativi alle metriche sarà modificata per integrare le nuove.

Package “gui”: Contiene i moduli che compongono l’interfaccia utente.

Artefatto	Impatto	Descrizione
AnalysisResultsUI.java	MEDIO	La GUI verrà adattata per contenere le nuove informazioni relative alle nuove metriche.
ConfigUI.java	MEDIO	Verranno aggiunte alla GUI della configurazione del plugin i valori relativi al timeout della Mutation Coverage e il numero di esecuzioni dei test per il rilevamento di Flaky Tests.

Package “init”: Contiene la classe che si occupa dell’inizializzazione del sistema.

Artefatto	Impatto	Descrizione
PluginInit.java	MEDIO	Si aggiungerà all’avvio del sistema la visualizzazione di una schermata per la selezione delle metriche da analizzare che si interpone tra PluginInit e le classi Processor.

Package “manager”: Contiene le classi che si occupano della creazione dei file di report e di recupero dei valori calcolati in passato.

Artefatto	Impatto	Descrizione
ReportManager.java	DEBOLE	Si dovranno aggiungere le nuove metriche nel salvataggio del report

Candidate Impact Set per la CR #3

Per quanto riguarda la correzione del bug da cui è afflitto il calcolo della Line Coverage e Branch Coverage, verranno modificati i seguenti artefatti:

Package “processor”: Contiene le classi che si occupano del calcolo delle metriche supportate.

Artefatto	Impatto	Descrizione
CoverageProcessor.java	FORTE	Si dovrà modificare il comportamento della classe intera per integrare il supporto al nuovo tool utilizzato, ovvero JACOCO

6. Studio di fattibilità

6.1 Identificazione, descrizione e valutazione dei costi

Identificazione	Valutazione	Motivazioni
Aggiunta dei nuovi Requisiti Funzionali	MEDIA	La modifica è caratterizzata prevalentemente dall'aggiunta di diverse nuove funzionalità.
Modifica dei Requisiti Funzionali	DEBOLE	Vi sono poche modifiche da effettuare alle funzionalità già esistenti.
Correzione del calcolo di Line Coverage e Branch Coverage	FORTE	E' stato già individuato il nuovo tool per il calcolo di Line Coverage e Branch Coverage, ma bisogna integrarlo nel sistema da zero.
Introduzione del calcolo e visualizzazione della Mutation Coverage	FORTE	Il calcolo della Mutation Coverage può essere effettuato tramite tool esterni, ma esistono diversi problemi che bisogna risolvere (e.g. tempi di esecuzione del tool, possibili timeouts).
Introduzione della rilevazione di Flaky Tests	MEDIA	L'implementazione non richiede costi elevati, ma bisogna gestire diverse problematiche, tra cui la scelta del numero di esecuzioni dei test per garantire un buon compromesso tra la rilevazione e i tempi di esecuzione.
Introduzione della schermata di avvio	DEBOLE	L'aggiunta della schermata di avvio non richiede una grande concentrazione di lavoro.

Modifica dell'interfaccia attuale	MEDIA	L'interfaccia attuale dovrà essere modificata per contenere le informazioni sulle nuove metriche.
Organizzazione del lavoro nel team di sviluppo	FORTE	I componenti del team non hanno le stesse conoscenze relative al dominio applicativo relativo al sistema. È necessaria, quindi, una fase di apprendimento.
Testing Funzionale	FORTE	Bisogna effettuare il testing completo del sistema, includendo le nuove funzionalità.

6.2 Identificazione, descrizione e classificazione dei benefici

Identificazione	Valutazione	Motivazioni
Correzione del calcolo di Line Coverage e Branch Coverage	FORTE	E' di vitale importanza correggere il calcolo di Line Coverage e Branch Coverage, poiché è una funzionalità essenziale del plugin.
Aggiunta del calcolo e visualizzazione della Mutation Coverage	MEDIA	Viene migliorata la capacità del plugin di misurare la qualità delle Test Classes, introducendo come nuovo fattore dinamico la Mutation Coverage, la quale si dimostra essere un buon indicatore per quanto riguarda la bassa qualità di un test.
Aggiunta della rilevazione di Flaky Tests	FORTE	Viene migliorata la capacità del plugin di misurare la qualità delle Test Classes, introducendo come nuovo fattore statico la presenza di Flaky Tests. Si è dimostrato che essi possono essere indotti dalla presenza di

		Test Smell.
Introduzione della schermata di avvio	FORTE	Migliora l'esperienza dell'utente. Permette di scegliere le metriche da includere nell'analisi, in modo tale da ridurre il tempo di attesa.

6.3 Identificazione, descrizione e classificazione dei rischi

Identificazione	Probabilità	Effetti	Motivazioni
Introduzione di nuovi fault	MODERATA	SERI	L'implementazione delle modifiche e delle aggiunte potrebbero introdurre nuovi fault. Inoltre verrà effettuato un testing più accurato, che potrebbe portare alla luce nuovi errori. Il tempo speso a correggere eventuali nuovi costituisce un problema serio per lo scheduling e il budget.
Mancanza training per apprendimento del dominio applicativo	MOLTO ALTA	TOLLERABILI	Parte del team non ha buone conoscenze del dominio applicativo, quindi potrebbero esserci dei rallentamenti nella fase di apprendimento. Il training per l'apprendimento potrebbe non essere disponibile.

7. Report post-modifica

L'Actual Impact Set relativo alle modifiche implementate, rappresenta gli artefatti effettivamente modificati ed è indicato nella tabella che segue. Di seguito, inoltre, verranno indicati i valori di *Precision* e *Recall*, utilizzate per verificare l'accuratezza dell'intero processo di impact analysis.

Artefatto	Impatto
TestClassAnalysis.java	DEBOLE
AnalysisResultsUI.java	MEDIO
PluginInit.java	MEDIO
ReportManager.java	DEBOLE
CoverageProcessor.java	FORTE

Precision: $\frac{|CIS \cap AIS|}{|CIS|} = \frac{5}{6} = 0.83$
Recall: $\frac{|CIS \cap AIS|}{|AIS|} = \frac{5}{5} = 1$

La Precision risulta essere minore di uno, poiché è stato individuato nel Candidate Impact Set una classe che non è stata poi modificata, ovvero ConfigUI.java. Essa fa dunque parte del **False Positive Impact Set (FPIS)**. La motivazione sta nel fatto che i valori relativi alle impostazioni della Mutation Coverage e dei Flaky Tests non dovevano essere inserite nella pagina di configurazione delle metriche con relativo salvataggio su file, ma nella nuova classe PluginInitGUI, poiché non c'è necessità del loro salvataggio sul file di configurazione.

La Recall risulta essere uguale a uno, poiché non sono state trovate nuove classi da modificare oltre a quelle individuate nel **CIS**. Il **Discovered Impact Set (DIS)** è dunque vuoto.

Sono state introdotte le nuove classi per soddisfare le modifiche richieste dalle tre Change Requests, ovvero:

- MutationCoverageProcessor,
- FlakyTestsProcessor,
- PluginInitGUI,
- FlakyTestsInfo,
- MutationCoverageInfo.

Inoltre è stato effettuato il testing dell'intero plugin, sia per verificare il comportamento delle nuove funzionalità sia per controllare il suo funzionamento globale.