UNIVERSITÀ DEGLI STUDI
DI SALERNO

Ingegneria, Gestione ed Evoluzione del Software

System Analysis

cASpER

Raimondo Rapacciuolo 0522501266

Contents

List of Figures

List of Tables

Chapter 1

Introduzione

cASpER è un plugin per IntelliJ IDEA in grado di ispezionare i file Java di un progetto in cerca di diverse tipologie di code smell e proporre i relativi metodi di refactoring per rimuoverli.

Il tool può essere installato in diversi modi, seguendo le istruzioni presenti nel file README.md presente nella seguente ***repository*** per versioni di IntelliJ IDEA comprese tra 2019.3 — 2019.3.5.

Selezionando il plugin dal menu Tools dell'IDE vengono mostrati due pannelli con interfaccai grafica, uno per l'esecuzione ed analisi dei risultati ed uno per la configurazione dei parametri.

Il pannello di configurazione permette di impostare le soglie per i parametri degli algoritmi che cASpER usa per trovare ognuno dei quattro code smell identificabili.

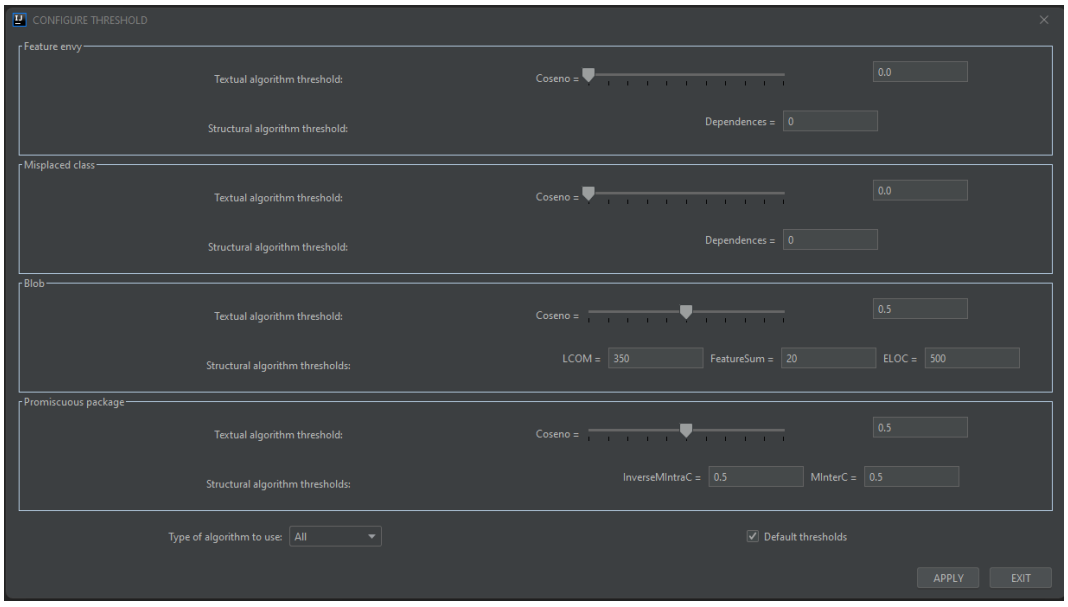

Figure 1.1: Pannello di configurazione.

Nel pannello di analisi dei risultati vengono mostrati in basso tutti i code smell trovati con i livelli di importanza, mentre sopra vengono visualizzate le soglie dei paramentri per l'analisi, da qui è possibile avviare tramite il pulsante INSPECT l'ispezione di uno smell ed è poi possibile richiedere il refactoring automatico.

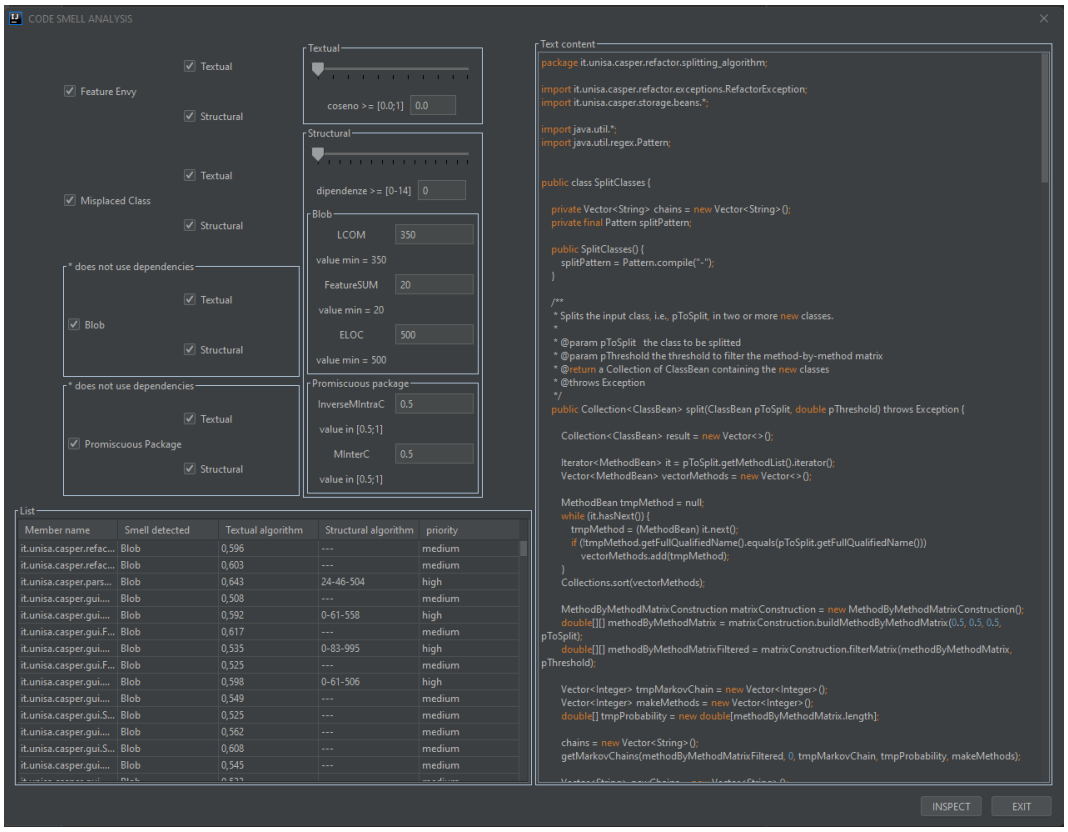

Figure 1.2: Pannello di analisi dei risultati.

1.1 Obiettivo del progetto

L'obiettivo principale del progetto è quello di aggiungere un nuovo detector di code smell basato su deep neural network. Il detector è stato sviluppato in python e sarà reso disponibile in cASpER come web services. Il nuovo detector sarà in grado di predire la presenza di 5 code smell: ComplexClass, BlobClass, LazyClass, RefusedBequest, SpaghettiCode, usando metriche strutturali. Il progetto sarà diviso nelle fasi di reverse engineering, impact analysis, implementazione e testing.

Chapter 2

Reverse Engineering

Per iniziare la fase di reverse engineering è stata effettuata un'analisi dei package presenti in *src*:

```
src/
├── main/
│   └── java/
│       └── it.unisa.casper/
│           ├── actions/
│           ├── analysis/
│           │   ├── code_smell/
│           │   ├── code_smell_detection/
│           │   │   ├── blob/
│           │   │   ├── comparator/
│           │   │   ├── divergent_change/
│           │   │   ├── feature_envy/
│           │   │   ├── misplaced_class/
│           │   │   ├── parallel_inheritance/
│           │   │   ├── promiscuous_package/
│           │   │   ├── shotgun_surgery/
│           │   │   ├── similarityComputation/
│           │   │   ├── smellynessMetricProcessing/
│           │   │   └── strategy/
│           │   └── history_analysis_utility/
│           ├── gui/
│           │   └── radarMap/
│           ├── parser/
│           ├── refactor/
│           │   ├── exceptions/
│           │   ├── manipulator/
│           │   ├── splitting_algorithm/
│           │   │   └── irEngine/
│           │   └── strategy/
│           ├── statistics/
│           ├── storage.beans/
│           ├── structuralMetrics/
│           └── topic/
└── test/
    └── it.unisa.casper/
        ├── analysis/
        │   └── code_smell_detection/
        │       ├── blob/
        │       ├── feature_envy/
        │       ├── Helper/
        │       ├── misplaced_class/
        │       └── promiscuos_package/
        └── refactor/
```

Figure 2.1: Packages del progetto

Nel package main troviamo it.unisa.casper che contiene i seguente package:

- **actions**: le classi di questo package si occupano delle azioni per il funzionamento del tool come l'avvio e la visualizzazione della gui.
- **analysis**: questo package contiene le classi per effettuare l'analisi per l'individuazione dei code smell ed è diviso in due subpackage principali:
- **subpackage code_smell**: questo package contiene le classi che modellano i code smell vi è una classe astratta CodeSmell principale che si specializza in altre tre classi astratte MethodLevelCodeSmell, ClassLevelCodeSmell, PackageLevelCodeSmell, questi vanno poi a specializzarsi nei code smell concreti identificabili dal tool.
- **subpackage code_smell_detection**: questo pacakge contiene dei package specifici per ogni code smell in cui sono presenti le strategie per identificarne la presenza.
- **gui**: questo package contiene le classi per la creazione e visualizzazione dell'interfaccia grafica del tool.
- **parser**: questo package contiene le classi che effettuano il parsing del progetto IntelliJ incapsulato in una variabile di tipo Project nelle classi bean.
- **refactor**: questo package contiene le classi le strategie da usare per il refactoring dei code smell individuati.
- **storage.bean**: questo package contiene le classi che servono per astrarre i concetti di variabile di istanza, metodo, classe e package di un progetto.
- **structuralMetrics**: questo package contiene la classe che permette di calcolare le diverse metriche che il tool usa per l'analisi.
- **topic**: questo package permette di estrarre informazioni utili all'analisi testuale del codice.

Nel package test troviamo it.unisa.casper che contiene i seguenti package:

- **analysis**: questo package contiene un package per ogni code smell di cui bisogna testare l'identificazione e per ogni code smell vengono testate le diverse strategie di identificazione.
- **refactor**: questo package contiene le classi di test per il refactoring.

Successivamente è stata una decomposizione in sottosistemi del tool, usando le informazioni acquisite dall'analisi dei package, ottento il seguente diagramma UML:

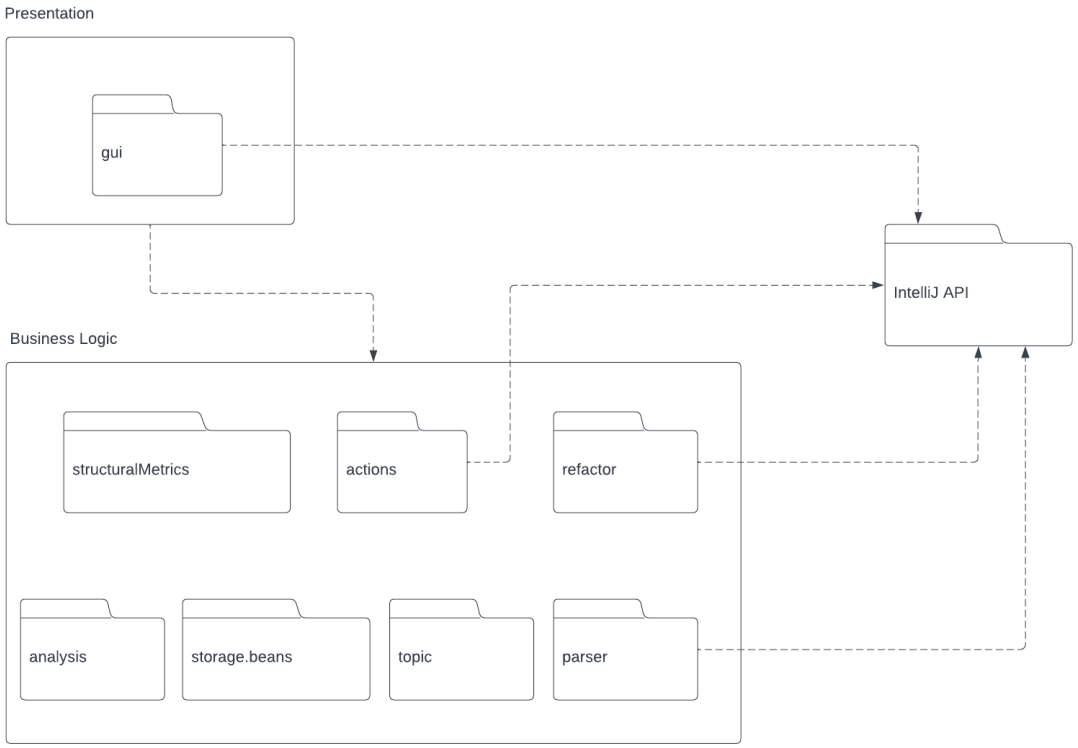

Figure 2.2: Decomposizione in sottosistemi

Chapter 3

Impact Analysis

Con le conoscenze, del sistema, ottenute dalla fase di reverse engineering è possibile eseguire una fase di impact analysis per le change request proposte. Per ognuna delle change request verrà seguito il seguente appoccio:

1. Viene descritta la change request;
2. Viene individuato uno Starting Impact Set e il traceability graph;
3. Viene individuato il Candidate Impact Set;
4. Dopo l'implementazione viene individuato l' Actual Impact Set, Il False Positive Impact Set e il Discovered Impact Set;
5. Infine vengono calcolate le metriche di Precisione, Recall e Adequacy.

3.1 Change Request

Il set di change request individuato per il progetto è il seguente:

ID	Descrizione
CR1	Aggiunta di nuove metriche strutturali
CR2	Aggiunta di nuovi code smell individuabili
CR3	Aggiunta del nuovo detector

Table 3.1: Set di Change Request

3.2 CR1

La change request CR1 prevede l'aggiunta di metodi per calcolare le metriche strutturali necessarie al nuovo detector per poter identificare i code smell, in particolare quelle che verranno aggiunte a quelle attualmente calcolate dal tool sono: LOCNAMM, NOMNAMM, PMMM, PRB, WLOCNAMM, WMCNAMM.

3.2.1 Starting Impact Set

Lo Starting Impact Set individuato con le informazioni ottenute dalla fase di reverse engineering è mostrato dal seguente traceability graph:

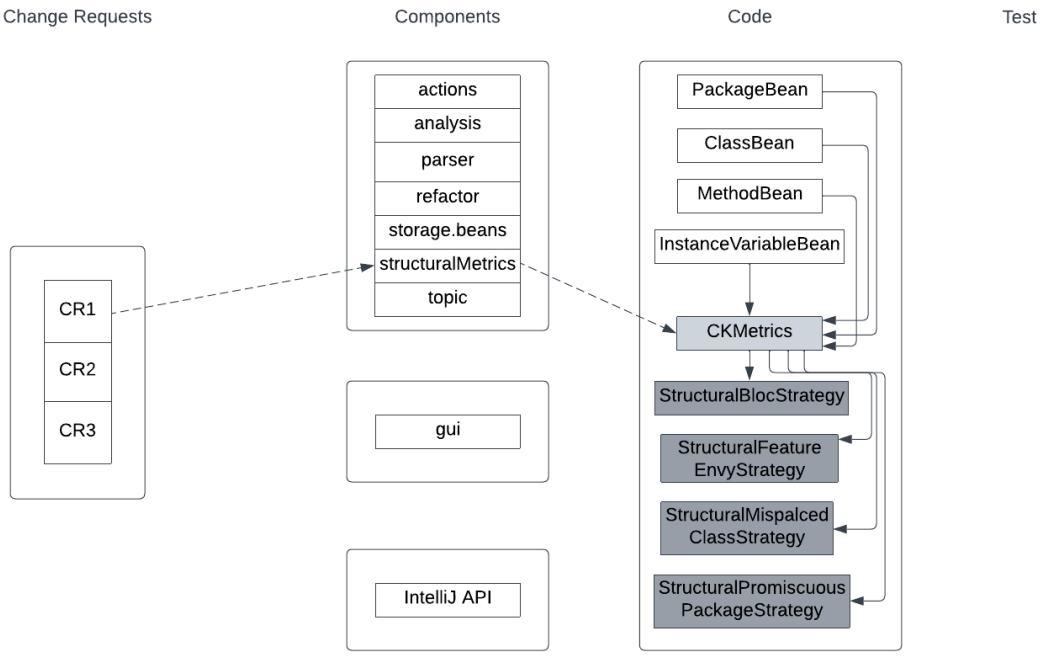

Figure 3.1: CR1 Traceability Graph

Lo starting Impact Set individuato sarà quindi formato da:

- CKMetrics
- StructuralBlobStrategy
- StructuralFeatureEnvyStrategy
- StructuralMisplacedClassStrategy
- StructuralPromiscuousPackageStrategy

3.2.2 Candidate Impact Set

I metodi già presenti nella classe CKMetrics non verranno modificati, ma ne verranno solo aggiunti di nuovi, questo fa sì che le classi che attualmente utilizzano i metodi di CKMetrics non verranno affette dalle modifiche e quindi possono essere eliminate dal CIS.

Il Candidate Impact Set individuato sarà quindi formato da:

- CKMetrics

3.2.3 Actual Impact Set

In seguito alle modifiche effettuate è stato individuato il seguente Actual Impact Set:

- CKMetrics

Quindi il False Positive Impact Set e il Discovered Impact Set sono vuoti.

3.2.4 Risultati

Le metriche calcolate per CR1 sono le seguenti:

$Recall = |CIS \cap AIS|/|AIS| = 1/1 = 1$

$Precision = |CIS \cap AIS|/|CIS| = 1/1 = 1$

$Inclusivness = AIS \subseteq CIS = 1$

3.3 CR2

La change request CR2 prevede l'aggiunta dei nuovi code smell individuabili dal detector e non attualmente presenti nel tool, in particolare verranno aggiunti: ComplexClass, LazyClass, RefusedBequest e SpaghettiCode.

3.3.1 Starting Impact Set

Lo Starting Impact Set individuato con le informazioni ottenute dalla fase di reverse engineering è mostrato dal seguente traceability graph:

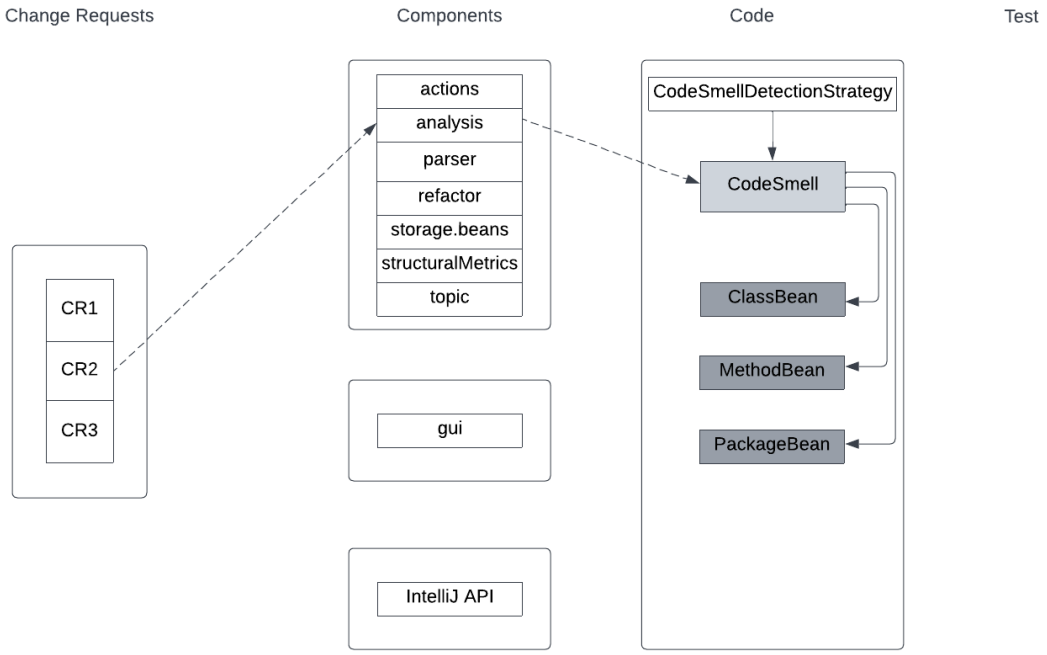

Figure 3.2: CR2 Traceability Graph

Lo starting Impact Set individuato sarà quindi formato da:

- CodeSmell
- ClassBean
- MethodBean
- PackageBean

3.3.2 Candidate Impact Set

I metodi già presenti nella classe CodeSmell non verranno modificati, ma ne verranno solo aggiunti di nuovi, questo fa sì che le classi che attualmente utilizzano i metodi di CodeSmell non verranno affette dalle modifiche e quindi possono essere eliminate dal CIS.

Il Candidate Impact Set individuato sarà quindi formato da:

- CodeSmell

3.3.3 Actual Impact Set

In seguito alle modifiche effettuate è stato individuato il seguente Actual Impact Set:

- CodeSmell

Quindi il False Positive Impact Set e il Discovered Impact Set sono vuoti.

3.3.4 Risultati

Le metriche calcolate per CR2 sono le seguenti:

$Recall = |CIS \cap AIS|/|AIS| = 1/1 = 1$

$Precision = |CIS \cap AIS|/|CIS| = 1/1 = 1$

$Inclusivness = AIS \subseteq CIS = 1$

3.4 CR3

La change request CR3 prevede l'implementazione all'interno del tool di un detecor basato su deep learning, il detector è stato sviluppato in python e sarà utilizzato da cASpER come web service. Il modulo prenderà in input, come richiesta GET, 19 metriche strutturali calcolate dal tool e restituirà una stringa contentente la predizione sulla presenza di uno dei 5 code smell identificabili: ComplexClass, BlobClass ,LazyClass, RefusedBequest e SpaghettiCode.

3.4.1 Starting Impact Set

Lo Starting Impact Set individuato con le informazioni ottenute dalla fase di reverse engineering è mostrato dal seguente traceability graph:

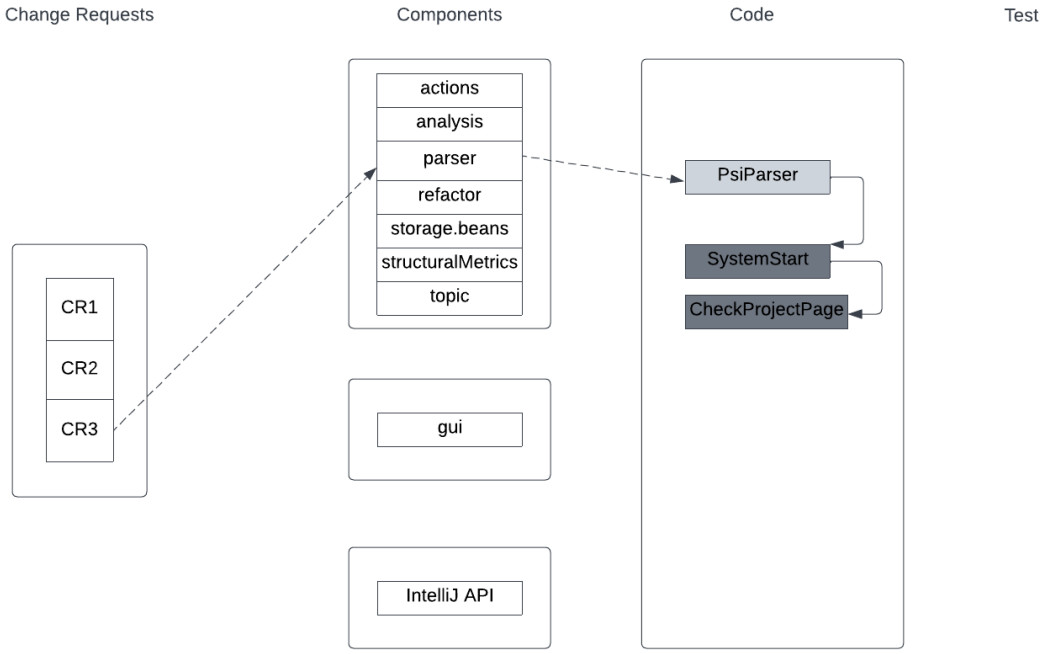

Figure 3.3: CR2 Traceability Graph

Lo starting Impact Set individuato sarà quindi formato da:

- PsiParser
- System Start
- CheckProjectPage

3.4.2 Candidate Impact Set

I metodi già presenti nella classe PsiParser non verranno modificati, ma ne verranno solo aggiunti di nuovi, questo fa sì che le classi che attualmente utilizzano i metodi di PsiParser non verranno affette dalle modifiche e quindi possono essere eliminate dal CIS, mentre le classi dell'attuale interfaccia grafica dovranno essere modificate per visualizzare i risultati della nuova funzionalità.

Il Candidate Impact Set individuato sarà quindi formato da:

- PsiParser
- CheckProjectPage
- ConfigureThreshold

3.4.3 Actual Impact Set

In seguito alle modifiche effettuate è stato individuato il seguente Actual Impact Set:

- PsiParser
- CheckProjectPage
- ConfigureThreshold

Quindi il False Positive Impact Set e il Discovered Impact Set sono vuoti.

3.4.4 Risultati

Le metriche calcolate per CR2 sono le seguenti:

$Recall = |CIS \cap AIS|/|AIS| = 3/3 = 1$

$Precision = |CIS \cap AIS|/|CIS| = 3/3 = 1$

$Inclusivness = AIS \subseteq CIS = 1$