# Università Degli Studi Di Salerno

## Dipartimento di Informatica

**Corso di Laurea Magistrale in Softaware Engeneering and IT Mangament**


INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

ANNO ACCADEMICO 2021/22

# DARTS_2.0

## Analisi, Progettazione e Realizzazione

| *Versione* | *Modifiche* |
|---|---|
| 0.1 | Stesura del primo capitolo |
| 0.2 | Descrizione del sistema attuale |
| 0.3 | Aggiunta delle Change Request |
| 0.4 | Descrizione Change Request di Gradle |
| 1.0 | Stesura del terzo capitolo |
| 1.1 | Impact analysis |
| 1.2 | Calcolo metriche |
| 1.3 | Test di regressione |
| 1.4 | Terminazione testing |

| *Autori* | *Matricola* |
|---|---|
| Gerardo Iuliano | 0522501329 |
| Alessandro Bacco | 0522501104 |
| Tiziano La Monica | 0522501258 |

# Sommario

# 1 Scopo del documento

In questo documento saranno descritti gli obiettivi del processo di estensione del plugin DARTS: Detection And Refactoring of Test Smell.
Si studierà come le modifiche identificate impattano sugli artefatti del sistema esistente, per ognuna di esse è stato condotto uno studio di fattibilità apposito.
Infine, tale documento include la pianificazione del testing delle componenti che derivano dalle CR, con la descrizione di come viene effettuato Regression Testing.

# 2 Panoramica del sistema attuale

DARTS (Detection And Refactoring of Test Smells) è un plugin IntelliJ che permette la rilevazione, attraverso le tecniche presenti nello stato dell'arte, di tre tipi particolari di test smell: General Fixture, Eager Test e Lack of Cohesion of Methods e permette di utilizzare le operazioni di refactoring provviste dalle API integrate di IntelliJ.

## 2.1 Requisiti Funzionali

Qui di seguito viene descritta la scala nominale scelta per etichettare ogni requisito funzionale con la relativa priorità:

- **Alta**: Per priorità alta si intende che la modifica da effettuare deve essere rilasciata nel minor tempo possibile poiché è una modifica che influenza il sistema in maniera critica
- **Media**: Per priorità media si intende che la modifica da effettuare deve essere implementata solo dopo che la modifica con priorità massima è stata implementata
- **Bassa**: Per priorità bassa si intende che la modifica da effettuare ha impatto minimo in termini di influenza sul nuovo sistema modificato, la si può vedere come quasi opzionale

| ***ID*** | ***Descrizione*** | ***Priorità*** |
|---|---|---|
| RF_1 | Il sistema deve consentire all'utente di individuare lo smell EagerTest nei casi di test. | Alta |
| RF_2 | Il sistema deve consentire all'utente di individuare lo smell GeneralFixture nei casi di test. | Alta |
| RF_3 | Il sistema deve consentire all'utente di individuare lo smell LackOfCohesion nei casi di test. | Alta |
| RF_4 | Il sistema deve consentire all'utente di effettuare il refactoring del codice affetto da smell. | Alta |
| RF_5 | Il sistema deve avviare la detection in fase di commit | Alta |

## 2.2 Attori

Il sistema, sino al momento dello sviluppo, prevede un unico attore, lo sviluppatore, il quale mediante l'interfaccia messa a disposizione può rilevare i vari test smell ed effettuare le operazioni di refactoring.

## 2.3 Design

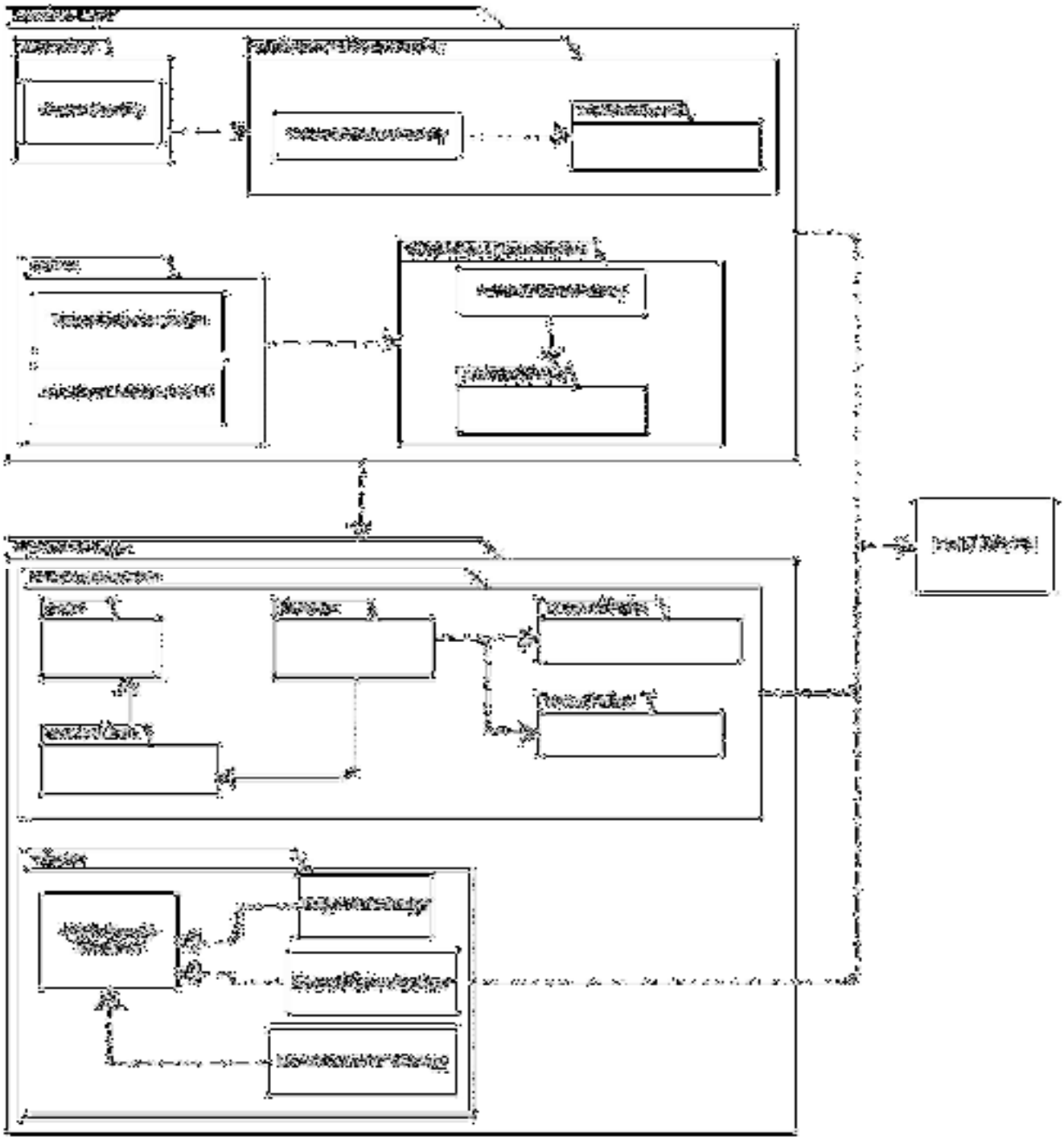
Il sistema è stato progettato in maniera tale da essere modulare; infatti, esso si compone di due layer differenti:

- Interface Layer: questo livello contiene tutta la logica necessaria per mostrare le informazioni all'utente e dare la possibilità di interagire col sistema;
- Application Layer: questo livello contiene tutta la logica del plugin, ovvero la possibilità di identificare gli smell ed effettuare il refactoring.

Il sistema software è stato sviluppato utilizzando il linguaggio di programmazione Object Oriented Java, coniugato alle librerie JetBrains per l'implementazione della logica di refactoring.

## 2.4 Implementazione

L'implementazione del software è stata effettuata in riferimento al modello descritto in precedenza, in particolare associando ad ogni layer un insieme di classi che sviluppano l'insieme delle funzionalità descritte in fase di analisi dei requisiti. In seguito, è osservabile l'architettura del software proposto:



*Architettura di DARTS*

Come si può notare dal diagramma, le componenti destinate alla logica di presentazione, e quindi all' Interface Layer, e le componenti destinati alla logica core, e quindi all' Application Layer, sono due componenti distinte e separate. I due layer quindi si compongono delle seguenti componenti:

### 2.4.1 Interface Layer

- Extensions: Contiene la classe CommitFactory che interfaccia il plugin all'handler di IntelliJ per effettuare il commit.
- Actions: Contiene le Classi e i metodi che saranno richiamati all'avvio del plugin, nel momento in cui vengono premuti i pulsanti all'interno del menu contestuale di IntelliJ.
- WindowCommitConstruction: contiene le classi e i metodi relativi all'interfaccia necessaria per la visualizzazione dei risultati della detection

### 2.4.2 Application Layer

- Test Smell Detection: Contiene la logica relativa alle regole per identificare i test smell. Ovviamente le rule fanno riferimento ai tipi di smell che il plug-in va a considerare.
- Refactor: Questo modulo fa riferimento alle operazioni di refactoring implementate dal tool. Ovviamente, le operazioni di refactoring possono essere "chiamate" solo se la fase di Detection è terminata. Quindi si pone l'obiettivo di effettuare una manipolazione (in termini di struttura) del codice scritto in modo da eliminare lo smell rilevato in precedenza
- Contextual Analysis: Questo modulo contiene la logica relativa all'estrazione dei dati e che fa riferimento ad una componente esterna: RepoDriller in modo tale da clonare la repository e raccogliere dati su di essa.

Il plugin inoltre fa utilizzo di diversi tool esterni, i quali sono:

- ViTRuM: Plugin per la visualizzazione e l'analisi di metriche relative ai test.
- Intellij API, PSI: Tool per la modellazione semantica e sintattica di codice.
- Repodriller API: Java Framework per il Mining Software Repositories (MSR)

A valle di quanto appena menzionato nella sezione relativa al Design, si può apprezzare come le componenti destinati alla logica di presentazione e le componenti destinati alla logica core siano separate.

# 3 Analisi delle modifiche richieste

Change request richieste:

CR_00_Gradle

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_00_Gradle | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede l'aggiunta del sistema di build gradle. | | |
| ***Scopo*** | Lo scopo della modifica è quello di rendere il progetto facilmente buildabile su ogni tipo di ambiente in modo da facilitare le varie attività di sviluppo e implementazione nelle sue versioni future. | | |
| ***Priorità*** | Alta[X] Media[ ] Bassa[ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, ma si perderebbe l'occasione di facilitare i vari contributors open-source nell'applicare modifiche e aggiunte che migliorerebbero il sistema. | | |

CR_00_Infrastruttura Plugin e Logica Core

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_00_IPeLC | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede la separazione dell'infrastruttura Plugin dalla logica Core | | |
| ***Scopo*** | Lo scopo della modifica è quello di rendere il progetto facilmente riutilizzabile all'interno di altri plugin o altri progetti. È bene separare le due componenti per poter facilitare l'estrazione e il riuso delle componenti che implementano l'analisi e l'individuazione degli smell. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso ma, una netta separazione della logica core dall'infrastruttura plugin permette una migliore manutenibilità, portabilità e riuso del codice implementato in altri progetti simili. | | |

CR_01_MagicNumberTest_Detector

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_01_MNT_Detector | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede la modifica del codice per implementare un nuovo Detector relativo al test smell. Il tipo di Detection sarà basato su informazioni strutturali e testuali. | | |
| ***Scopo*** | Modificando l'attuale implementazione si aggiungerà una nuova funzionalità che l'utente potrà sfruttare per il rilevamento di nuovi smell. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, ma l'utente potrebbe preferire altri plug-in che offrono più funzionalità di DARTS. | | |

CR_01_MagicNumberTest_GUI

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_01_MNT_GUI | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede l'aggiunta della relativa interfaccia utente per mantenere la coerenza del sistema | | |
| ***Scopo*** | Aggiungendo l'interfaccia grafica si renderanno i dati più leggibili e le informazioni facilmente individuabili. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, tuttavia, non vi sarà feedback in relazione all'aggiunta della nuova funzionalità. | | |

CR_02_ConditionalTestLogic_Detector

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_02_CTL_Detector | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede la modifica del codice per implementare un nuovo Detector relativo al test smell. Il tipo di Detection sarà basato su informazioni strutturali e testuali. | | |
| ***Scopo*** | Modificando l'attuale implementazione si aggiungerà una nuova funzionalità che l'utente potrà sfruttare per il rilevamento di nuovi smell. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, ma l'utente potrebbe preferire altri plug-in che offrono più funzionalità di DARTS. | | |

CR_02_ConditionalTestLogic_GUI

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_02_CTL_GUI | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede l'aggiunta della relativa interfaccia utente per mantenere la coerenza del sistema | | |
| ***Scopo*** | Aggiungendo l'interfaccia grafica si renderanno i dati più leggibili e le informazioni facilmente individuabili. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, tuttavia, non vi sarà feedback in relazione all'aggiunta della nuova funzionalità. | | |

CR_03_ConstructorInitialization_Detector

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_03_CI_Detector | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede la modifica del codice per implementare un nuovo Detector relativo al test smell. Il tipo di Detection sarà basato su informazioni strutturali e testuali. | | |
| ***Scopo*** | Modificando l'attuale implementazione si aggiungerà una nuova funzionalità che l'utente potrà sfruttare per il rilevamento di nuovi smell. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se*** | Il normale comportamento del sistema non verrebbe compromesso, ma l'utente | | |

| | |
|---|---|
| ***non accettata*** | potrebbe preferire altri plug-in che offrono più funzionalità di DARTS. |

CR_03_ConstructorInitialization_GUI

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_03_CI_GUI | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede l'aggiunta della relativa interfaccia utente per mantenere la coerenza del sistema | | |
| ***Scopo*** | Aggiungendo l'interfaccia grafica si renderanno i dati più leggibili e le informazioni facilmente individuabili. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, tuttavia, non vi sarà feedback in relazione all'aggiunta della nuova funzionalità. | | |

CR_04_ExceptionHandling_Detector

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_04_EH_Detector | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede la modifica del codice per implementare un nuovo Detector relativo al test smell. Il tipo di Detection sarà basato su informazioni strutturali e testuali. | | |
| ***Scopo*** | Modificando l'attuale implementazione si aggiungerà una nuova funzionalità che l'utente potrà sfruttare per il rilevamento di nuovi smell. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, ma l'utente potrebbe preferire altri plug-in che offrono più funzionalità di DARTS. | | |

CR_04_ExceptionHandling_GUI

| | ***Id*** | ***Data Richiesta*** | ***Richiesta da*** |
|---|---|---|---|
| | CR_04_EH_GUI | 02/04/2022 | Andrea De Lucia |
| ***Descrizione*** | Si richiede l'aggiunta della relativa interfaccia utente per mantenere la coerenza del sistema | | |
| ***Scopo*** | Aggiungendo l'interfaccia grafica si renderanno i dati più leggibili e le informazioni facilmente individuabili. | | |
| ***Priorità*** | Alta [X] Media [ ] Bassa [ ] | | |
| ***Conseguenze, se non accettata*** | Il normale comportamento del sistema non verrebbe compromesso, tuttavia, non vi sarà feedback in relazione all'aggiunta della nuova funzionalità. | | |

Le CR relative all'implementazione dei detector prevedono un'impact analysis praticamente identica, in quanto ogni detector, seppur identifica uno smell diverso, richiede la stessa analisi degli altri detector.
Analogamente le CR relative all'implementazione della GUI richiedo la stessa impact analysis.
Ovviamente deve essere preservato il comportamento attuale del plugin.

Di seguito vengono studiate nel dettaglio le singole Change Request.

## 3.1 Conversione a Progetto Gradle

Change Request di riferimento:

- *CR_00_Gradle*

### 3.1.1 Descrizione

Quasi tutti i tool seguono la struttura di progetto IntelliJ Platform Plugin, una struttura e sistema di build proprio di IntelliJ adibito ai soli plugin. Questa struttura è adesso sconsigliata, e JetBrains consiglia di migrare ad una struttura Gradle, noto sistema di build per Java simile a Maven e già usato per le app native Android. Bisogna anzitutto convertire il progetto secondo Gradle.

### 3.1.2 Problematiche affrontate

| ***Issue 1*** | |
|---|---|
| ***Issue*** | • Alcune librerie utilizzate non sono integrabili tramite Gradle. |
| ***Proposal*** | • Aggiungere queste librerie localmente. |
| ***Criterion*** | • Facilitare il recupero delle librerie necessarie al funzionamento del plugin.<br>• Agevolare il mantenimento del plugin. |
| ***Argument*** | • Gradle prevede l'aggiunta di libreria locali, supponiamo quindi che sia la scelta più veloce e semplice. |
| ***Solution*** | • Le librerie esterne non integrabili tramite Gradle verranno aggiunte localmente all'interno del progetto e messe a disposizione all'interno del repository GitHub. |

| ***Issue 2*** | |
|---|---|
| ***Issue*** | • Non esiste una chiara documentazione per quanto riguarda la conversione di un progetto non Gradle ad uno Gradle. |
| ***Proposal*** | • Utilizzare esempi reperibili sul web<br>• Utilizzare tool di scaffolding messi a disposizione da IntelliJ |
| ***Criterion*** | • Rendere la migrazione il più semplice possibile |
| ***Argument*** | • Dagli esempi è semplice individuare la struttura ma non sempre questi sono chiari o sono alterati dagli sviluppatori.<br>• IntelliJ mette a disposizione il tool per la generazione di un nuovo progetto conforme ai requisiti Gradle. |
| ***Solution*** | • Genereremo la struttura del progetto tramite IntelliJ così da essere sicuri di seguire la struttura conforme ai requisiti da Gradle, nulla esclude la consultazione di esempi reperibili sul web. |

### 3.1.3 Impact Analysis

Al momento della proposta, DARTS, vi era una struttura delle directory simile ma non conforme a quella consigliata dalla documentazione di JetBrains. Quindi, cambiando la struttura delle directory abbiamo dovuto aggiornare gli statement import e gli statement package all'interno dei file java che compongono il progetto.
L'impatto della modifica verrà valutato utilizzando questa scala:

- Debole: se saranno necessarie solo modifiche marginali;
- Medio: se saranno necessarie modifiche all'artefatto, non facendo cambiare però la sua struttura in maniera eccessiva;

- Forte: se saranno necessarie sostanziali modifiche dell'artefatto o se l'artefatto dovrà essere completamente sostituito

| ***Artefatto*** | ***Impatto*** | ***Descrizione*** |
|---|---|---|
| Tutte le classi Java contenute all'interno del package *main*. | Debole | Occorrerà modificare le import e i package di ogni classe in modo tale da rispecchiare la nuova struttura delle directory. |

| | ***SIS*** | ***CIS*** | ***AIS*** | ***DIS*** | ***FPIS*** | ***Impatto*** |
|---|---|---|---|---|---|---|
| ***Tutte le classi*** | X | X | X | | | Debole |

### 3.1.4 Studio di fattibilità

Identificazione, descrizione e valutazione dei costi

| ***Identificazione*** | ***Valutazione*** | ***Motivazione*** |
|---|---|---|
| Modifica della struttura del progetto secondo standard Gradle. | Bassa | La struttura originale è molto vicina a quella consigliata dalla documentazione. |

Identificazione, descrizione e valutazione dei benefici

| ***Identificazione*** | ***Valutazione*** | ***Motivazione*** |
|---|---|---|
| Gestione semplificata delle dipendenze. | Media | Usando il file gradle.build è possibile dichiarare le dipendenze del progetto che saranno automaticamente risolte da Gradle stesso. |

### 3.1.5 Analisi Post-Implementazione

Le modifiche apportate per il completamento della Change Request sono uguali a quelle previste nell'Impact Analysis. La gestione delle dipendenze è stata gestita in modo che il maggior numero possibile di componenti esterne fosse gestito da Gradle. Tuttavia, per quanto riguarda le componenti TestFactorsPlugin.jar e TestSmellDiffusion.jar, le quali non sono disponibili sul repository web, è stato deciso di inserirle manualmente includendo la libreria all'interno del progetto.

## 3.2 Separazione Infrastruttura Plugin dalla logica Core

Change Request di riferimento:

- *CR_00_IPeLC*

### 3.2.1 Descrizione

Il progetto Gradle deve avere (almeno) due moduli ben distinti: uno per la sola logica legata all'infrastruttura plugin IntelliJ e l'altro contenente il cuore del tool, che è indipendente dal fatto di essere un plugin per un IDE.

### 3.2.2 Problematiche affrontate

| ***Issue 1*** | |
|---|---|
| ***Issue*** | • Quali componenti si occupano della logica del plugin e quali si occupano della logica core? |
| ***Proposal*** | • Ispezione del codice.<br>• Studio della documentazione presente. |
| ***Criterion*** | • È bene separare le due componenti per poter facilitare l'estrazione e il riuso delle componenti che implementano l'analisi e l'individuazione degli smell. |
| ***Argument*** | • La documentazione, dando una visione generale di tutto il codice sorgente, può facilitare il lavoro di comprensione. |
| ***Solution*** | • Verranno utilizzati entrambi gli approcci, quindi ispezione del codice e studio della documentazione presente. |

### 3.2.3 Impact Analysis

La versione attuale del plugin presenta già una netta separazione tra quelle che sono le componenti che rappresentano l'infrastruttura del plugin e le componenti che implementano la logica core del plugin: analisi e individuazione degli smell.

### 3.2.4 Studio di fattibilità

Identificazione, descrizione e valutazione dei costi

| ***Identificazione*** | ***Valutazione*** | ***Motivazione*** |
|---|---|---|
| Separazione Infrastruttura Plugin e Logica Core. | Bassa | La struttura originale presenta già questa netta separazione. |

Identificazione, descrizione e valutazione dei benefici

| ***Identificazione*** | ***Valutazione*** | ***Motivazione*** |
|---|---|---|
| Portabilità e riuso della logica core. | Media | Così facendo il tool è facilmente riutilizzabile all'interno di altri plugin o altri progetti. |

### 3.2.5 Analisi Post-Implementazione

Nulla da specificare

## 3.3 Implementazione test smell detector - Business Logic

In questa sezione viene presentata la logica implementativa dei detector protagonisti della change request.

La relativa impact analysis è mostrata nella sezione 4.1.

Change Request di riferimento:

- *CR_01_MagicNumberTest_Detector*
- *CR_02_ConditionalTestLogic_Detector*
- *CR_03_ConstructorInitialization_Detector*
- *CR_04_ExceptionHandling_Detector*

### 3.3.1 Descrizione

Per ogni detector, sia esso testuale e/o strutturale, sono state identificate tutte le classi coinvolte nell'implementazione della feature. Fondamentalmente, ogni detector segue lo stesso tipo di implementazione, indice per cui, le classi che sono state affette dal ripple-effect sono state sempre le medesime per ogni *CR_xx_smellName_Detector*. Tuttavia, per ogni nuovo detector vi è stata la necessità di creare le due classi apposite per la gestione dell'identificazione dello smell stesso e delle classi esterne affette dal tipo di smell.

### 3.3.2 Problematiche Affrontate

Individuato ciò che consta essere lo starting impact set e la struttura generale della logica core degli altri smell già implementati in DARTS, non sono state affrontate problematiche tanto gravi da essere considerate bloccanti per il prosieguo del progetto. Tuttavia, ecco alcune tematiche di interesse per il paragrafo.

| ***Issue 1*** | |
|---|---|
| ***Issue*** | • Comprendere il flusso d'esecuzione di DARTS |
| ***Proposal*** | • Seguire il flusso di esecuzione partendo dal metodo anAction, metodo dal quale parte l'esecuzione del plugin in seguito all'attivazione dell'ultimo.<br>• Seguire il flusso di chiamate relative a un detector per comprendere quali classi vengono impattate. |
| ***Criterion*** | • Individuare l'insieme di classi che partecipano al flusso di esecuzione della detection di uno smell. |
| ***Argument*** | • |
| ***Solution*** | • Seguiremo il flusso di esecuzione del detector relativo allo smell Eager. |

| ***Issue 2*** | |
|---|---|
| ***Issue*** | • Comprendere il funzionamento delle classi del pacchetto "*com.intellij.psi.**" |
| ***Proposal*** | • Utilizzare la documentazione ufficiale sul web.<br>• Consultare il codice già implementato |
| ***Criterion*** | • Capire come maneggiare queste classi, quali metodi mettono a disposizione e quali tipi di classi fanno al caso nostro.<br>• Capire come utilizzare queste classi nell'implementazione della logica di detection data la natura dello smell. |
| ***Argument*** | • Dalla documentazione si possono conoscere ulteriori classi di questo pacchetto molto utili alla causa.<br>• Analizzare l'utilizzo di queste classi nei detector già implementati può aiutare a capire meglio la loro applicazione. |
| ***Solution*** | • È stata consultata sia la documentazione che il codice. |

### 3.3.3 Analisi Post-Implementazione

L'analisi post implementazione susseguitasi al completamento di ogni nuovo detector consiste nel confronto tra l'appena identificato AIS con i precedenti SIS e CIS. Tale analisi è servita per il calcolo delle metriche di precision e recall per valutare la bontà delle stime.

Per l'analisi dettagliata degli impact set si consideri il paragrafo apposito, descritto nel capitolo 4.

## 3.4 Implementazione test smell detector - GUI Logic

In questa sezione viene presentata la logica implementativa della GUI.

La relativa impact analysis è mostrata nella sezione 4.2.

Change Request di riferimento:

- *CR_01_MagicNumberTest_GUI*
- *CR_02_ConditionalTestLogic_ GUI*
- *CR_03_ConstructorInitialization_ GUI*
- *CR_04_ExceptionHandling_ GUI*

### 3.4.1 Descrizione

La parte di implementazione relativa alla GUI (Graphical User Interface) è consistita nell'individuazione delle relative classi apposite allo svolgimento della funzione, in particolare, il processo che si è seguito è stato analogo a quello seguito per la logica di business dei detector implementati, informazioni più dettagliate sono descritte nella sezione dell'impact set, capitolo 5.

### 3.4.2 Problematiche Affrontate

Le problematiche relative all'implementazione della GUI sono le stesse di quelle individuate nella sezione 4.1.2, tuttavia, è bene considerare che per ogni pannello di necessaria implementazione si è dovuto sacrificare del tempo in maniera da essere sicuri del corretto funzionamento dei componenti, sul progetto fantoccio TestProjectForDARTS. Qui di seguito, viene riportata una panoramica dell'issue.

| ***Issue 1*** | |
|---|---|
| ***Issue*** | • Time-Consuming feedback, il tempo necessario a visualizzare le modifiche apportate alla GUI è considerevole in quanto bisogna istallare ogni volta la nuova versione del plugin sull'IDE e tastare il plugin su un progetto. |
| ***Proposal*** | • Implementare le componenti grafiche necessarie alla visualizzazione dei dati estratti da un singolo detector e testare la relativa GUI. Ottenuto un riscontro positivo, implementare le componenti grafiche per i detector restanti prendendo come riferimento la prima implementazione. |
| ***Criterion*** | • Individuare l'insieme di classi da implementare per le restanti componenti grafiche. |
| ***Argument*** | • Implementata una GUI, le restanti seguono lo stesso identico criterio |
| ***Solution*** | • La soluzione scelta consiste nel seguire il processo descritto nella proposal. |

### 3.4.3 Analisi Post-Implementazione

L'analisi post implementazione susseguitasi al completamento di ogni nuovo detector consiste nel confronto tra l'appena identificato AIS con i precedenti SIS e CIS. Tale analisi è servita per il calcolo delle metriche di precision e recall per valutare la bontà delle stime relative alle modifiche della GUI.

Per l'analisi dettagliata degli impact set si consideri il paragrafo apposito, descritto nel capitolo 4.

# 4. Impact Analysis

## *4.1 Business Logic*

### 4.1.1 Starting Impact Set – SIS

Per individuare lo starting impact set, si è preso come riferimento il flusso di esecuzione del detector dello smell Eager Test, in quanto la modifica richiesta dalla CR consiste nell'aggiungere un nuovo detector. Partendo dalla specifica della CR, dalla documentazione e dal codice sorgente, abbiamo identificato il SIS (Starting Impact Set), ovvero l'insieme di oggetti o componenti iniziali che molto probabilmente verranno impattati dalla CR.

### 4.1.2 Candidate Impact Set – CIS

Analizzata la natura della CR e individuata la strategia di detection, si è ritenuto opportuno implementare dei detector di tipo strutturale. Tale scelta non ha impattato sul SIS in quanto, nonostate i detector verranno implementati secondo regole strutturali, le classi relative ai detector testuali restano impattate in modo indiretto in quanto verranno cambiate alcune interfacce comuni ai due tipi di detector.

Le classi relative alle metriche invece, secondo la natura della change request, in realtà non richiedono modifiche, in quanto non è necessario creare delle metriche di supporto alla detection di tali smell.

Valutando la change request, abbiamo deciso di concentrare lo sforzo sulla principale funzione di detection del plugin. Ulteriori funzionalità del plugin, come ad esempio la detection in fase di commit, saranno sviluppate in future release. Quindi la classe CommitFactory.java è stata rimossa.

### 4.1.3 Actual Impact Set – AIS

Insieme di classi o componenti effettivamente impattate dalla change request al termine dell'implementazione.

### 4.1.4 False Positive Impact Set – FPIS

Insieme di classi o componenti stimate nel CIS ma non realmente impattate. FPIS = CIS / AIS

### 4.1.5 Discovered Impact Set – DIS

Insieme di classi o componenti non presenti nel CIS ma scoperte durante l'implementazione della change request.

| | ***SIS*** | ***CIS*** | ***AIS*** | ***DIS*** | ***FPIS*** | ***Impatto*** |
|---|---|---|---|---|---|---|
| ***StructuralDetectionAction.java*** | X | X | X | | | |
| ***TextualDetectionAction.java*** | X | X | X | | | |
| ***IDetector.java*** | X | X | X | | | |
| ***TestSmellStructuralDetector.java*** | X | X | X | | | |
| ***TestSmellTextualDetector.java*** | X | X | X | | | |
| ***DeveloperVisitor.java*** | | | | | | |
| ***CommitFactory.java*** | X | | | | | |
| ***ClassWithEagerTestPanel.java*** | | | | | | |
| ***ClassWithGeneralFixturePanel.java*** | | | | | | |
| ***ClassWithLackOfCohesionPanel.java*** | | | | | | |
| ***EagerTestPanel.java*** | | | | | | |
| ***GeneralFixturePanel.java*** | | | | | | |
| ***LackOfCohesionPanel.java*** | | | | | | |
| ***PrincipalFrame.java*** | | | | | | |
| ***TestSmellWindowFactory.java*** | | | | | | |
| ***EagerTestStrategy.java*** | | | | | | |
| ***GeneralFixtureStrategy.java*** | | | | | | |
| ***LackOfCohesionStrategy.java*** | | | | | | |
| ***IRefactor.java*** | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| ***PSIClassBean.java*** | | | | | | |
| ***PSIMethodBean.java*** | | | | | | |
| ***ProductionClassesSingleton.java*** | | | | | | |
| ***StringChecker.java*** | | | | | | |
| ***DataMiner.java*** | | | | | | |
| ***EagerTestStructural.java*** | | | | | | |
| ***GeneralFixtureStructural.java*** | | | | | | |
| ***LackOfCohesionOfTestSmellStructural.java*** | | | | | | |
| ***EagerTestInfo.java*** | | | | | | |
| ***MethodWhitEagerTest.java*** | | | | | | |
| ***GeneralFixtureInfo.java*** | | | | | | |
| ***MethodWhitGeneralFixture.java*** | | | | | | |
| ***LackOfCohesionInfo.java*** | | | | | | |
| ***MethodWithLackOfCohesion.java*** | | | | | | |
| ***TestSmellInfo.java*** | | | | | | |
| ***CosineSimilarity.java*** | | | | | | |
| ***EagerTestStructural.java*** | | | | | | |
| ***EagerTestTextual.java*** | | | | | | |
| ***GeneralFixtureStructural.java*** | | | | | | |
| ***GeneralFixtureTextual.java*** | | | | | | |
| ***LackOfCohesionStructural.java*** | | | | | | |
| ***LackOfCohesionTextual.java*** | | | | | | |
| ***ConverterUtilities.java*** | | | | | | |
| ***FileUtility.java*** | | | | | | |
| ***PSITestSmellUtilities.java*** | | | | | | |
| ***TestSmellUtilities.java*** | | | | | | |
| ***ContextualAnalysisFrame.java*** | | | | | | |
| ***ContextualAnalysisResultFrame.java*** | | | | | | |
| ***CustomLable.java*** | | | | | | |
| ***CustomLable2.java*** | | | | | | |
| ***CustomListRendere.java*** | | | | | | |
| ***CustomListRendere2.java*** | | | | | | |
| ***RefactorWindow.java*** | | | | | | |
| ***WarningWindow.java*** | | | | | | |
| ***ETSmellPanel.java*** | | | | | | |
| ***GFSmellPanel.java*** | | | | | | |
| ***LOCSmellPanel.java*** | | | | | | |
| ***CommitPrincipalFrame.java*** | | | | | | |
| ***CommitWindowFactory.java*** | | | | | | |
| ***EagerTestCP.java*** | | | | | | |
| ***GeneralFixtureCP.java*** | | | | | | |
| ***LackOfCohesionCP.java*** | | | | | | |
| ***LoadingFrame.java*** | | | | | | |

*Tabella che raffigura tutti i set relativi all'impact analysis della business logic*

## *4.2 GUI Logic*

### 4.2.1 Starting Impact Set – SIS

Come effettuato nel SIS della business logic, abbiamo individuato il SIS analizzando il codice e la documentazione. In particolare, abbiamo seguito il flusso di chiamate relativo alla creazione dell'interfaccia grafica. Sono state inserite nel SIS le classi che sono impattate durante la creazione della window principale.

### 4.2.2 Candidate Impact Set – CIS

Dopo ulteriori controlli non sono state aggiunte o rimosse ulteriori classi.

### 4.2.3 Actual Impact Set – AIS

Durante la realizzazione delle CR relative alla GUI, ci siamo resi conto che durante l'implementazione di alcune nuove classi (ex. CTLSmellPanel), la classe RefactorWindow.java veniva istanziata. Tale classe offre all'utente la possibilità di effettuare il refactoring e mostra all'utente un suggerimento su come risolvere lo smell. Di conseguenza nel DIS è stata descritta la motivazione che ci ha portato ad effettuare le modifiche sulla classe RefactorWindow.java.

### 4.2.4 False Positive Impact Set – FPIS

Nessuna classe risulta come falso positivo.

### 4.2.5 Discovered Impact Set – DIS

In fase di implementazione dei panel, ci siamo resi conto che ogni componente panel era composta da un ulteriore panel relativo al refactoring. Di conseguenza è stata modificata anche la classe RefactorWindow.java, la quale mostra all'utente un suggerimento di come risolvere lo smell individuato. Nonostante la fase di refactoring risulta non funzionante (vedesi test di regressione), per coerenza grafica e per offrire comunque all'utente almeno un consiglio su come risolvere lo smell, abbiamo deciso di modificare anche questa classe per offrire il miglior supporto allo sviluppatore.

| | ***SIS*** | ***CIS*** | ***AIS*** | ***DIS*** | ***FPIS*** | ***Impatto*** |
|---|---|---|---|---|---|---|
| ***RefactorWindow.java*** | | | X | X | | |
| ***CommitWindowFactory.java*** | X | X | X | | | |
| ***ProductionClassesSingleton.java*** | | | | | | |
| ***StringChecker.java*** | | | | | | |
| ***DataMiner.java*** | | | | | | |
| ***DeveloperVisitor.java*** | | | | | | |
| ***CommitFactory.java*** | | | | | | |
| ***ClassWithEagerTestPanel.java*** | | | | | | |
| ***ClassWithGeneralFixturePanel.java*** | | | | | | |
| ***ClassWithLackOfCohesionPanel.java*** | | | | | | |
| ***EagerTestPanel.java*** | | | | | | |
| ***GeneralFixturePanel.java*** | | | | | | |
| ***LackOfCohesionPanel.java*** | | | | | | |
| ***PrincipalFrame.java*** | | | | | | |
| ***TestSmellWindowFactory.java*** | | | | | | |
| ***EagerTestStrategy.java*** | | | | | | |
| ***GeneralFixtureStrategy.java*** | | | | | | |
| ***LackOfCohesionStrategy.java*** | | | | | | |
| ***IRefactor.java*** | | | | | | |
| ***PSIClassBean.java*** | | | | | | |
| ***PSIMethodBean.java*** | | | | | | |
| ***IDetector.java*** | | | | | | |
| ***TestSmellStructuralDetector.java*** | | | | | | |
| ***TestSmellTextualDetector.java*** | | | | | | |
| ***EagerTestStructural.java*** | | | | | | |
| ***GeneralFixtureStructural.java*** | | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| ***LackOfCohesionOfTestSmellStructural.java*** | | | | | | |
| ***EagerTestInfo.java*** | | | | | | |
| ***MethodWhitEagerTest.java*** | | | | | | |
| ***GeneralFixtureInfo.java*** | | | | | | |
| ***MethodWhitGeneralFixture.java*** | | | | | | |
| ***LackOfCohesionInfo.java*** | | | | | | |
| ***MethodWithLackOfCohesion.java*** | | | | | | |
| ***TestSmellInfo.java*** | | | | | | |
| ***CosineSimilarity.java*** | | | | | | |
| ***EagerTestStructural.java*** | | | | | | |
| ***EagerTestTextual.java*** | | | | | | |
| ***GeneralFixtureStructural.java*** | | | | | | |
| ***GeneralFixtureTextual.java*** | | | | | | |
| ***LackOfCohesionStructural.java*** | | | | | | |
| ***LackOfCohesionTextual.java*** | | | | | | |
| ***ConverterUtilities.java*** | | | | | | |
| ***FileUtility.java*** | | | | | | |
| ***PSITestSmellUtilities.java*** | | | | | | |
| ***TestSmellUtilities.java*** | | | | | | |
| ***ContextualAnalysisFrame.java*** | | | | | | |
| ***ContextualAnalysisResultFrame.java*** | | | | | | |
| ***CustomLable.java*** | | | | | | |
| ***CustomLable2.java*** | | | | | | |
| ***CustomListRendere.java*** | | | | | | |
| ***CustomListRendere2.java*** | | | | | | |
| ***StructuralDetectionAction.java*** | | | | | | |
| ***WarningWindow.java*** | | | | | | |
| ***ETSmellPanel.java*** | | | | | | |
| ***GFSmellPanel.java*** | | | | | | |
| ***LOCSmellPanel.java*** | | | | | | |
| ***CommitPrincipalFrame.java*** | | | | | | |
| ***TextualDetectionAction.java*** | | | | | | |
| ***EagerTestCP.java*** | | | | | | |
| ***GeneralFixtureCP.java*** | | | | | | |
| ***LackOfCohesionCP.java*** | | | | | | |
| ***LoadingFrame.java*** | | | | | | |

*Tabella che raffigura tutti i set relativi all'impact analysis della GUI logic*

# 5 Metriche

Si presentano le metriche relative ai dati del lavoro svolto.

5.1 Recall

$$Recall = \frac{|CIS \cap AIS|}{|AIS|}$$

5.2 Precision

$$Precision = \frac{|CIS \cap AIS|}{|CIS|}$$

5.3 F-Measure

$$F - Measure = \frac{(2 \times Precision \times Recall)}{Precision + Recall}$$

| ***Results*** | | | | | |
|---|---|---|---|---|---|
| | ***\|CIS\|*** | ***\|AIS\|*** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| • **CR_01_Gradle** | 62 | 62 | 1 | 1 | 1 |
| • **CR_01_MagicNumberTest_Detector**<br>• **CR_02_ConditionalTestLogic_Detector**<br>• **CR_03_ConstructorInitialization_Detector**<br>• **CR_04_ExceptionHandling_Detector** | 5 | 5 | 1 | 1 | 1 |
| • **CR_01_MagicNumberTest_GUI**<br>• **CR_02_ConditionalTestLogic_GUI**<br>• **CR_03_ConstructorInitialization_GUI**<br>• **CR_04_ExceptionHandling_GUI** | 1 | 2 | 1 | 0.5 | 0.66 |

# 6 Studio di Fattibilità

Scala ordinale di valutazione:

- Alta: si intende che la maggior parte dello sforzo verrà impiegato nel focus dell'attività.
- Media: si intende che buona parte dello sforzo verrà impiegato nel focus dell'attività.
- Bassa: si intende lo sforzo minimo necessario allo svolgimento dell'attività.

## 6.1 Identificazione, Descrizione e Valutazione dei Costi

| Identificazione | Valutazione | Descrizione |
|---|---|---|
| Sviluppo di nuovi test smell | Alta | La modifica richiesta prevede la creazione della logica di identificazione dei nuovi test smell. |
| Organizzazione del lavoro nel team di sviluppo | Medio | I componenti del team di sviluppo condividono poche ore settimanali per le modifiche da apportare al sistema. Pertanto, verranno effettuare riunioni anche nel fine settimana. |
| Progettazione testing | Alta | Il testing dei nuovi detector è di fondamentale importanza per capire se si è effettivamente in grado di rilevare lo smell in classi di progetti d'uso reale. |

## 6.2 Identificazione, Descrizione e Valutazione dei Benefici

| Identificazione | Valutazione | Descrizione |
|---|---|---|
| Standardizzazione del sistema di build con Gradle | Alta | Vi è la necessità di rendere il progetto compatibile con un sistema di build in modo da velocizzare il processo di sharing dello stesso tra tutti i collaboratori presenti e futuri |

6.3 Identificazione, Descrizione e Valutazione dei Rischi

| Identificazione | Valutazione | Descrizione | Strategia |
|---|---|---|---|
| Errori di identificazione dei test smell | Alta | Le operazioni di identificazione dei nuovi test smell potrebbero segnalare falsi positivi. | Contingenza: Spendere più effort da parte dell'intero team nel testing per capire il malfunzionamento della detection e relativo bug fixing. |
| Complessità del sistema da realizzare | Medio | Il numero di funzionalità del software potrebbe aumentare. | Minimization: eventuali funzionalità emerse durante la fase di sviluppo, se non di vitale importanza, saranno implementate nelle release future. |
| Moduli del sistema non funzionanti | Media | Alcuni moduli di supporto al progetto potrebbero non funzionare correttamente | Minimization: se il modulo non è oggetto delle change requests in questione sarà ignorato. |

# 7 Sviluppi futuri

Poiché l'effort totale del team è stato speso per la maggior parte nella conversione del progetto a plugin di gradle, nell'implementazione dei nuovi detector per i test smell identificati nella change request, nel processo di testing dei suddetti e del sistema finale in modo da constatare che sia stato preservato il comportamento originale di DARTS, per gli sviluppi futuri si ha intenzione di effettuare:

- Una nuova modifica riguardo la detection dei relativi nuovi smell anche in fase di commit.
- Il fixing di alcune parti relative alla logica di refactoring del plugin
- Implementazione della logica di refactor per i nuovi smell identificati nella change request.