**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**

Test Plan

Sommario

[***Versione*** 3](#_Toc105599856)

[***Modifiche*** 3](#_Toc105599857)

[***Autori*** 3](#_Toc105599858)

[**Obiettivi del Testing** 4](#_Toc105599860)

[**Approccio** 4](#_Toc105599861)

[**Tools utilizzati** 5](#_Toc105599862)

[***Unit Test Plan*** 6](#_Toc105599863)

[***Integration Test Plan*** 7](#_Toc105599864)

[***System Test Plan*** 8](#_Toc105599865)

[***Test di Regressione*** 13](#_Toc105599866)

|  |  |  |  |
| --- | --- | --- | --- |
| ***Versione*** | | ***Modifiche*** | |
| 0.1 | Obiettivi del testing | |
| 0.2 | Approccio al testing | |
| 0.3 | Modifica Unit Test Plan | |
| 0.4 | Generazione dei test frame | |
| 1.0 | Testing di sistema | |
| 1.1 | Revisione Finale | |

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

# **Obiettivi del Testing**

Come anticipato, lo scopo del testing è quello di individuare la presenza di faults all’interno del sistema e condurre attività di debug. Il testing ha due “parametri” fondamentali, che sono gli input e l’oracolo, ovvero il risultato che vogliamo ottenere. A valle di ciò ricordiamo che, il testing ha successo nel caso in cui, dato un input al sistema, il risultato ottenuto è diverso dal valore che ci aspettiamo, ovvero dall’oracolo. Il sistema DARTS è un plugin per l’ambiente di sviluppo IntelliJ IDEA e in quanto tale questo vuol dire che in fase di identificazione dei test smell, sia nel refactoring, vengono utilizzati strumenti forniti dalla stessa piattaforma che sfruttano l’ambiente di esecuzione; ambiente che non può essere facilmente emulato in uno scenario di testing automatizzato. Questo però non ci impedisce del tutto di effettuare test di regressione. Infatti, visto la mancata possibilità di scrivere casi di test per il plugin, è stato concepito in precedenza un “test pilota”, ovvero un test con delle istanze di test smell create a mano e che quindi potessero simulare l’oracolo. Lo scopo era far eseguire DARTS su questo progetto pilota e verificare che il plugin riuscisse a trovare ogni istanza di test smell inserita manualmente. Dopo il testing di regressione iniziale, che ci assicura che il plugin funzioni correttamente come descritto dalle documentazioni precedenti, si inizia a sviluppare le CR descritte. Dopo ogni task portato a termine, si passa al testing della CR e in caso di fault, si aggiusta la classe di produzione fino a quando il risultato del testing sia uguale all’oracolo. Dopo il testing della CR bisogna necessariamente eseguire di nuovo il testing di regressione, per avere la certezza che la nuova modifica effettuata non abbia intaccato il funzionamento di tutto il plugin.

# **Approccio**

Si è deciso di seguire un approccio black-box basato sulle funzionalità di DARTS. Avendo aggiunto nuove funzionalità al plugin, sono state testate le seguenti:

* Il sistema deve consentire all’utente di individuare lo smell Magic Number nei test case.
* Il sistema deve consentire all’utente di individuare lo smell Conditional Test Logic nei test case.
* Il sistema deve consentire all’utente di individuare lo smell Exception Handling nei test case.
* Il sistema deve consentire all’utente di individuare lo smell Constructor Initialization nei test case.

Tali funzionalità, per essere testate, richiedono in input oggetti della classe PSIClass che rappresentano le classi di test del progetto sul quale il plugin viene lanciato. Si è quindi deciso di applicare Category Partition e di usare come categoria il numero di istanze di smells presenti in un progetto per il testing di sistema mentre per il testing di unità e di integrazione la categoria sarà la presenza o meno del progetto.

Data la natura del testing su un plugin, per simulare come input un progetto sarà utilizzata la classe LightJavaCodeInsightFixtureTestCase. Il testing non prende in input un progetto vero e proprio ma attraverso l’uso di LightJavaCodeInsightFixtureTestCase si simula la presenza di un progetto nel quale le classi di produzione e le classi di testing sono impostate dal tester. Il progetto viene creato a partire da delle classi target scelte dal tester. Di conseguenza come input del parametro Project vengono date una o più classi partendo dalle quali viene creato un progetto fittizio da LightJavaCodeInsightFixtureTestCase.

***Esempio:***

*Progetto = “MagicNumberPresent.java”*

*Data la classe “MagicNumberPresent.java”, LightJavaCodeInsightFixtureTestCase crea un’istanza di project composta dalla classe “MagicNumberPresent.java”. Di conseguenza al parametro Progetto viene assegnata l’istanza di un project composto dalla classe indicata. Nei test case saranno quindi indicate le classi che compongono l’istanza di project.*

Infine è stato effettuato un ulteriore testing white-box per aumentare la coverage di ogni detector.

Si è scelto il criterio di copertura branch coverage basato sui Control Flow Graph di ogni detector. Per massimizzare la coverage saranno coperti i branch di ogni CFG non coperti dal testing black box. Per ulteriori dettagli consultare il paragrafo relativo al testing white box.

# **Tools utilizzati**

* JUnit: framework per il testing.
* Mockito: l’isolamento dei test di unità può essere ottenuto utilizzando oggetti che simulano il comportamento degli oggetti reali. Mockito fornisce questa possibilità.
* Jacoco: libreria che consente la di generare dei report sulla Code Coverage del testing.

# ***Unit Test Plan***

Nel test di unità si sono andate a testare principalmente le classi coinvolte nel processo di detection. Essendo l’obiettivo del testing verificare la presenza di faults all’interno del codice introdotto, è stato necessario testare tutte le singole classi che partecipano alla funzionalità sotto testing. Testare in modo isolato ogni singola componente permette di escludere eventuali malfunzionamenti derivanti da classi di supporto, concentrando così l’attenzione sulle componenti nuove. In particolare, le classi di supporto al detector sono:

*Test item*

|  |  |  |
| --- | --- | --- |
| ***ID*** | ***Class*** | ***Caratteristica da testare*** |
| CU\_Unit | ConverterUtilities.java | Dato in input un progetto, si occupa di estrarre tutte le classi dai packages? |
| TSU\_Unit | TestSmellUtilities.java | Dato in input le classi di un progetto, estrae tutte le classi di test? |

*Criteri pass/fail*

|  |  |
| --- | --- |
| ***Fail*** | Il test non ha trovato nessun fault. |
| ***Pass*** | Il test ha trovato un fault. |

|  |  |
| --- | --- |
| ***CU\_Unit*** | |
| ***Parametri*** | * Project |
| ***Categorie*** | * C\_1: progetto presente |
| ***Scelte*** | **C\_1**   * *SP\_1*: Progetto not null: * *SP\_2*: Progetto null |
| ***Vincoli*** | Nessun vincolo |

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| testConverterUtilitiesNotNull | *SP\_1* | Progetto individuato e classi estratte. |
| testConverterUtilitiesNull | *SP\_2* | Progetto non individuato e nessuna classe estratta. |

|  |  |
| --- | --- |
| ***TSU\_Unit*** | |
| ***Parametri*** | * PSIClass |
| ***Categorie*** | * C\_1: classi presenti |
| ***Scelte*** | **C\_1**   * *SC\_1*: Classi not null: * *SC\_2*: Classi null |
| ***Vincoli*** | Nessun vincolo |

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| testSmellUtilitiesNotNull | *SC\_1* | Le classi sono presenti e vengono estratte solo le classi di test |
| testSmellUtilitiesNull | *SC\_2* | Le classi non sono presenti |

# ***Integration Test Plan***

Nel test di integrazione sono state testate insieme le componenti di supporto alla detection.

*Test item*

|  |  |  |
| --- | --- | --- |
| ***ID*** | ***Classes*** | ***Caratteristica da testare*** |
| CU\_TSU\_Integration | ConverterUtilities.java  TestSmellUtilities.java | Dato in input un progetto, ConverterUtilities.java si occupa di estrarre tutte le classi dai packages. Le classi ottenute sono date in input a TestSmellUtilities.java che estrae esclusivamente tutte le classi di test. |

*Criteri pass/fail*

|  |  |
| --- | --- |
| ***Fail*** | Il test non ha trovato nessun fault. |
| ***Pass*** | Il test ha trovato un fault. |

|  |  |
| --- | --- |
| ***Parametri*** | * Project |
| ***Categorie*** | * C\_1: progetto presente |
| ***Scelte*** | **C\_1**   * *SP\_1*: Progetto not null * *SP\_2*: Progetto null |

|  |  |
| --- | --- |
| ***Parametri*** | * PSIClass |
| ***Categorie*** | * C\_1: classi presenti |
| ***Scelte*** | **C\_1**   * *SC\_1*: Classi not null * *SC\_2*: Classi null |

|  |  |
| --- | --- |
| **Vincoli** | * *SC\_1* [ *if* Progetto not null ] |

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| testIntegrationUtilitiesNotNull | *SP\_1, SC\_1* | Le classi di test vengono estratte |
| testIntegrationUtilities\_ClassNull | *SP\_1, SC\_2* | Nessuna classe di test estratta |
| testIntegrationUtilitiesNull | *SP\_2, SC\_2* | Nessuna classe di test estratta |

# ***System Test Plan***

Nel test di Sistema sono state unite tutte le componenti che partecipano alla detection di uno smell. Tali componenti rappresentano l’intero flusso di esecuzione del plugin per la detection degli smells.

*Test item*

|  |  |  |
| --- | --- | --- |
| ***ID*** | ***Class*** | ***Caratteristiche da testare*** |
| CTL\_System | ConverterUtilities.java  TestSmellUtilities.java  ConditionalTestLogicStructural.java | Capacità del sistema di individuare lo smell Conditionl Test Logic |
| EH\_System | ConverterUtilities.java  TestSmellUtilities.java  ExceptionHandlingStructural.java | Capacità del sistema di individuare lo smell Exception Handling |
| CI\_System | ConverterUtilities.java  TestSmellUtilities.java  ConstructorInitializationStructural.java | Capacità del sistema di individuare lo smell Constructor Initialization |
| MN\_System | ConverterUtilities.java  TestSmellUtilities.java  MagicNumberStructural.java | Capacità del sistema di individuare lo smell Magic Number |
| DA\_System | ConverterUtilities.java  TestSmellUtilities.java  DuplicateAssertStructural.java | Capacità del sistema di individuare lo smell Duplicate Assert |
| IT\_System | ConverterUtilities.java  TestSmellUtilities.java  IgnoredTestStructural.java | Capacità del sistema di individuare lo smell Ignored Test |

*Criteri pass/fail*

|  |  |
| --- | --- |
| ***Fail*** | Il test non ha trovato nessun fault. |
| ***Pass*** | Il test ha trovato un fault. |

Il seguente studio mostra i parametri in input necessari al testing di sistema del detector ConditionalTestLogic (CTL\_System) e ExceptionHandling (EH\_System). Per testare la funzionalità di detection è necessario prendere in input un *Project* e una *Threshold* inserita dall’utente. Essendo che tali detector prevedono gli stessi parametri di input, lo studio seguente è condiviso per i detector sopra citati.

Generati i Test Frame invece, la definizione dei test case sarà fatta per ogni detector.

|  |  |
| --- | --- |
| ***Parametri*** | * Project p |
| ***Categorie*** | * C\_1: istanze di smell nel project |
| ***Scelte*** | **C\_1**   * *PS\_1*: Progetto con 0 istanze di smell * *PS\_2*: Progetto con 1 istanza di smell * *PS\_3:* Progetto con 2 o più istanze di smell |

|  |  |
| --- | --- |
| ***Parametri*** | * Threshold x |
| ***Categorie*** | * C\_1: valore |
| ***Scelte*** | **C\_1**   * Valore in-range:   + *TS\_1*: 0   + *TS\_2*: 1-4   + *TS\_3*: 5 * Valore out-of-range:   + *TS\_4*: -1   + TS\_5: 6 |

|  |  |
| --- | --- |
| **Vincoli** | * *TS\_1, TS\_2, TS\_3, TS\_4, TS\_5* [ *if* progetto presenta classi di test] * *TS\_1, TS\_2, TS\_3, TS\_4, TS\_5* [ *if* progetto not null] |

**Test Frame CTL\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_01 | *PS\_1, TS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_02 | *PS\_1, TS\_2* | Il detector non individua nessuno smell |
| TC\_Sys\_03 | *PS\_1, TS\_3* | Il detector non individua nessuno smell |
| TC\_Sys\_04 | *PS\_1, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_05 | *PS\_1, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_06 | *PS\_2, TS\_1* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_07 | *PS\_2, TS\_2* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_08 | *PS\_2, TS\_3* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_09 | *PS\_2, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_10 | *PS\_2, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_11 | *PS\_3, TS\_1* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_12 | *PS\_3, TS\_2* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_13 | *PS\_3, TS\_3* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_14 | *PS\_3, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_15 | *PS\_3, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |

**Test Frame EH\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_16 | *PS\_1, TS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_17 | *PS\_1, TS\_2* | Il detector non individua nessuno smell |
| TC\_Sys\_18 | *PS\_1, TS\_3* | Il detector non individua nessuno smell |
| TC\_Sys\_19 | *PS\_1, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_20 | *PS\_1, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_21 | *PS\_2, TS\_1* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_22 | *PS\_2, TS\_2* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_23 | *PS\_2, TS\_3* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_24 | *PS\_2, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_25 | *PS\_2, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_26 | *PS\_3, TS\_1* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_27 | *PS\_3, TS\_2* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_28 | *PS\_3, TS\_3* | Il detector individua gli smell che rispettano la threshold |
| TC\_Sys\_29 | *PS\_3, TS\_4* | Il plugin segnala errore dell’input e non effettua la detection |
| TC\_Sys\_30 | *PS\_3, TS\_5* | Il plugin segnala errore dell’input e non effettua la detection |

Per quanto riguarda il test di sistema sui detector per gli smell Constructor Initialization, Magic Number, Duplicate Assert e Ignored Test, tali detector prendono in input solo l’istanza del progetto e non necessitano di una Threshold.

CI\_System, MN\_System, DA\_System e IT\_System condividono quindi la seguente tabella:

|  |  |
| --- | --- |
| ***Parametri*** | * Project p |
| ***Categorie*** | * C\_1: istanze di smell nel project |
| ***Scelte*** | **C\_1**   * *PS\_1*: Progetto con 0 istanze di smell * *PS\_2*: Progetto con 1 istanza di smell * *PS\_3:* Progetto con 2 o più istanze di smell |

|  |  |
| --- | --- |
| ***Vincoli*** | Nessun Vincolo |

**Test Frame CI\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_31 | *PS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_32 | *PS\_2* | Il detector individua 1 smell |
| TC\_Sys\_33 | *PS\_3* | Il detector individua 2 o più smell |

**Test Frame MN\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_34 | *PS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_35 | *PS\_2* | Il detector individua 1 smell |
| TC\_Sys\_36 | *PS\_3* | Il detector individua 2 o più smell |

**Test Frame DA\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_37 | *PS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_38 | *PS\_2* | Il detector individua 1 smell |
| TC\_Sys\_39 | *PS\_3* | Il detector individua 2 o più smell |

**Test Frame IT\_System**

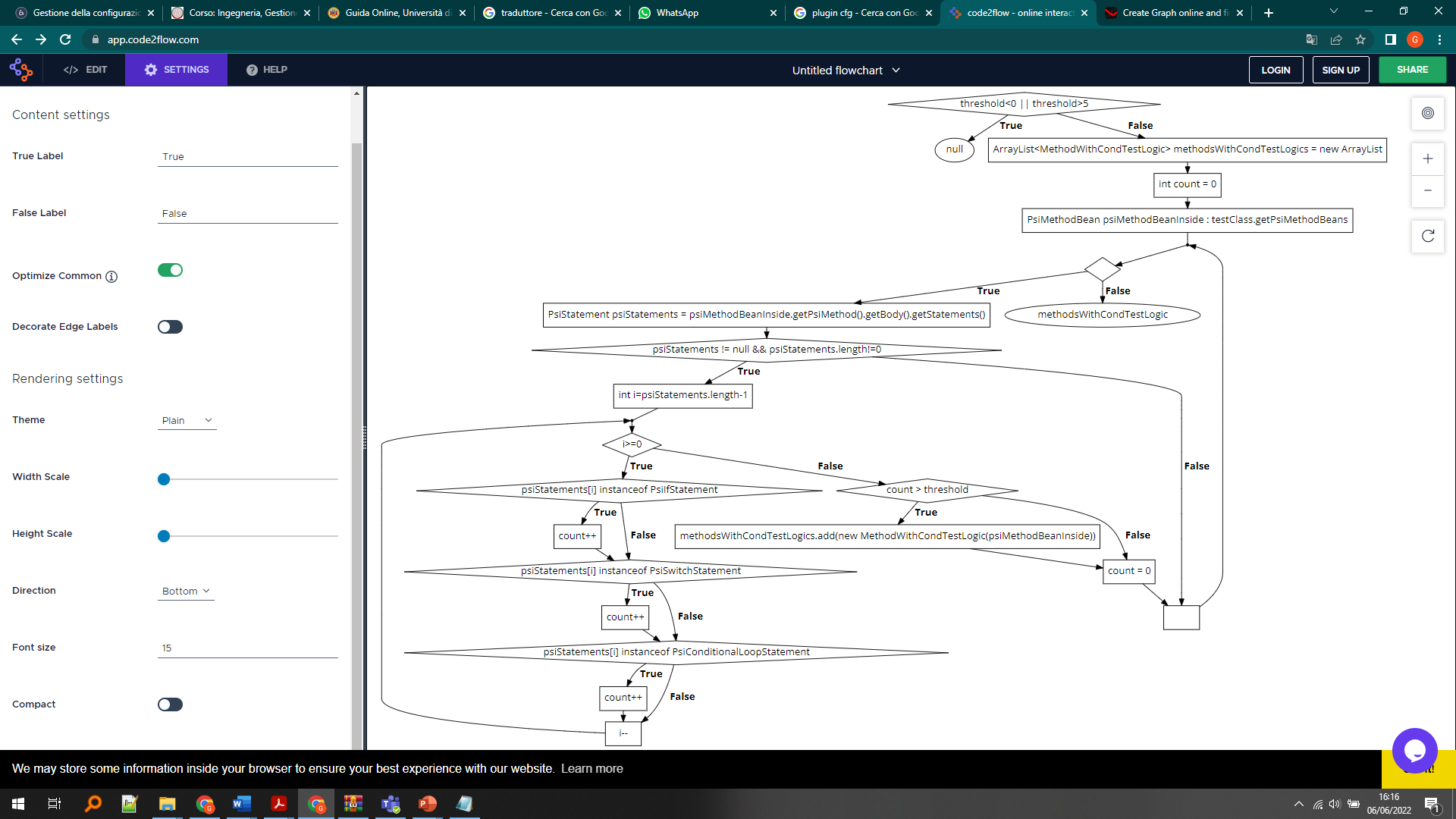
|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Sys\_40 | *PS\_1* | Il detector non individua nessuno smell |
| TC\_Sys\_41 | *PS\_2* | Il detector individua 1 smell |
| TC\_Sys\_42 | *PS\_3* | Il detector individua 2 o più smell |

# ***Control Flow Graph***

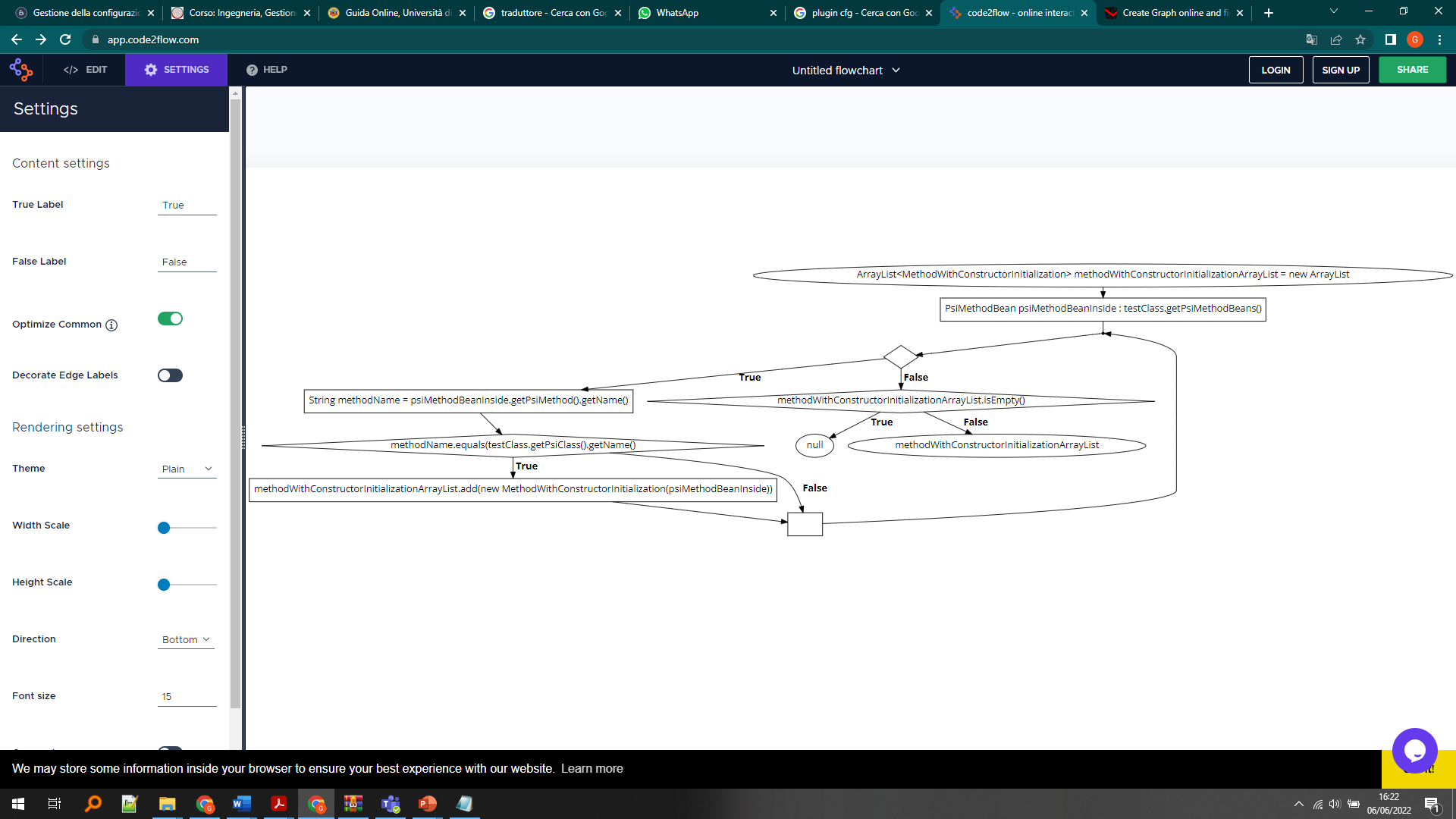
Di seguito sono riportati i CFG dei detector che saranno utilizzati per identificare quali condizioni soddisfare al fine di incrementare la coverage. I seguenti CFG saranno confrontati con il report di jacoco. Per ogni branch non evidenziato nel report di jacoco sarà creato un test case che copre quel branch, andando a soddisfare la condizione necessaria affinché quel branch venga coperto.

I seguenti CFG sono stati generati dal tool Code2Flow.

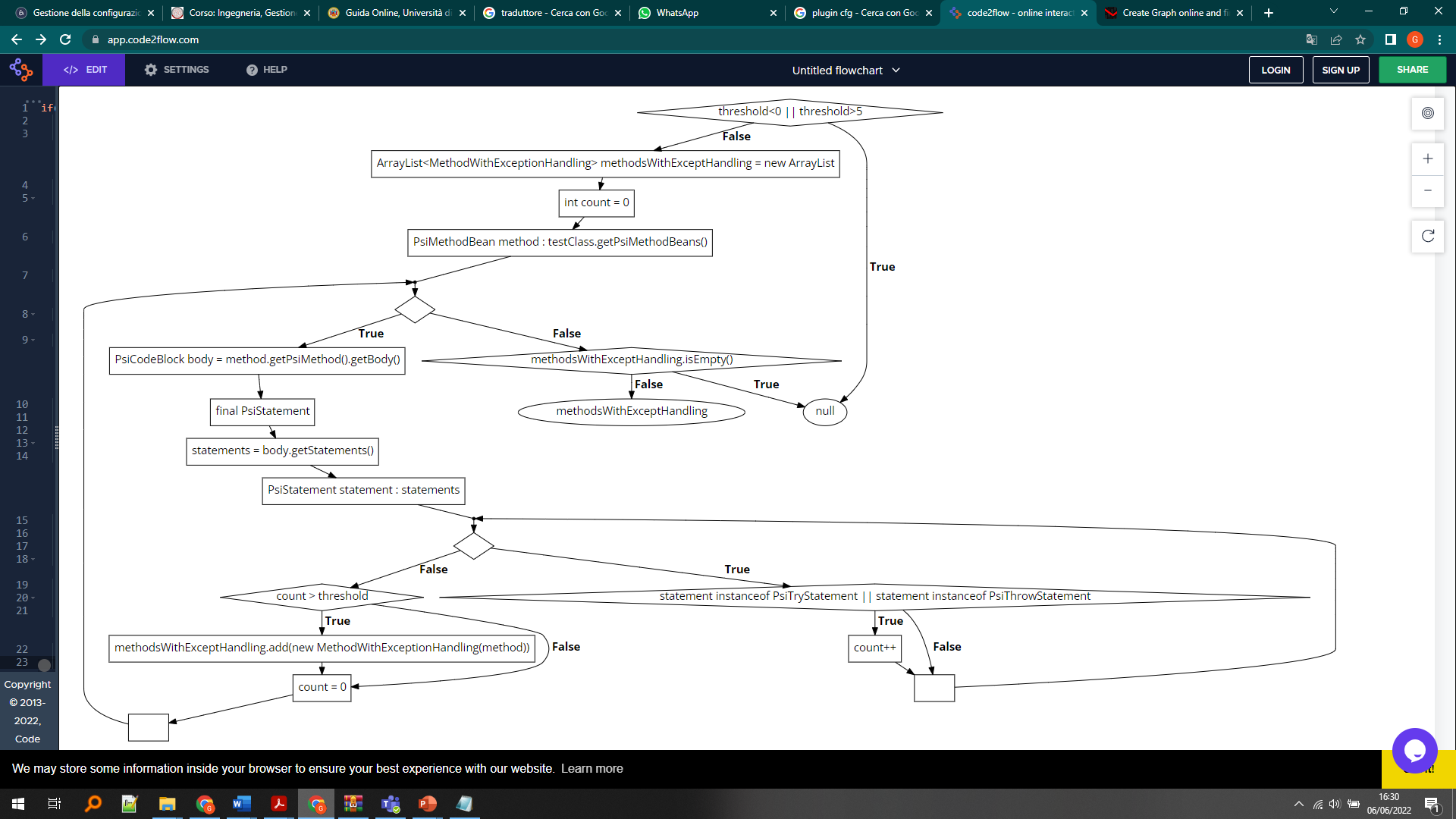
* *ConditionalTestLogic*



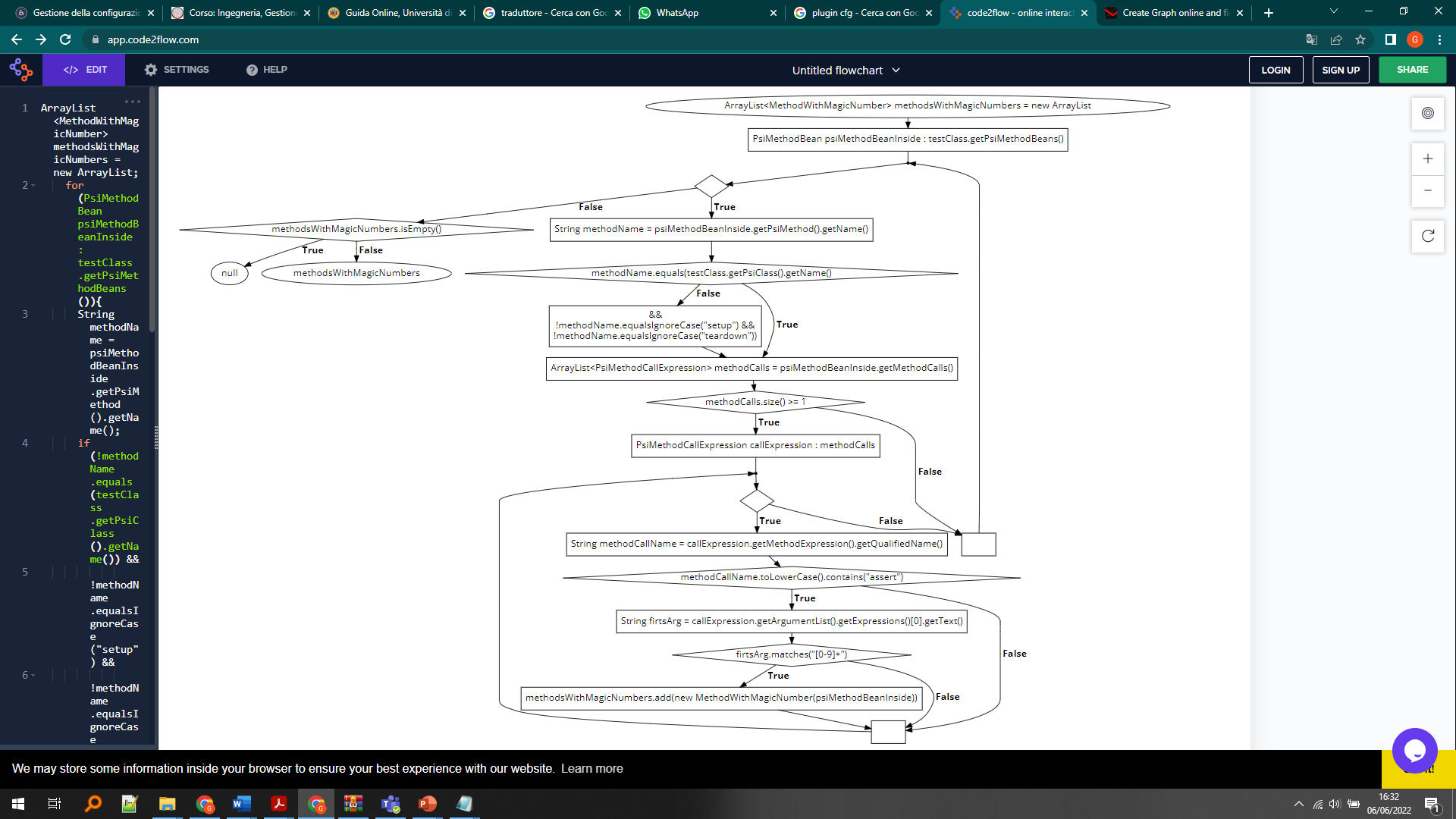
* *ConstructorInitialization*



* *ExceptionHandling*



* *MagicNumber*



# ***Test di Regressione***

In assenza di un’automazione dell’operazione di testing all’interno del tool, è stato necessario effettuare il test manualmente analogamente a quanto riportato nella documentazione della versione precedente di DARTS.

Quindi, è stato utilizzato il progetto fantoccio TestProjectForDarts, il quale presenta classi di test per ogni tipologia di test smell.

Il progetto TestProjectForDARTS è un progetto fantoccio utilizzato nella precedente versione del plugin per effettuare il testing di sistema.

Grazie all’esecuzione del plugin sul progetto TestProjectForDARTS è stato appurato che tra le funzionalità del sistema, la parte relativa al refactoring non funziona correttamente. Il plugin presenta la possibilità di effettuare il refactoring del codice affetto da smell tramite la pressione di un button presente sulla GUI. Alla pressione del button il plugin non effettua nessun refactoring rimanendo il codice affetto da smell inalterato. L’unica funzionalità relativa al refactoring è il suggerimento che il plugin propone per eliminare lo smell. Tale suggerimento viene mostrato nella GUI in corrispondenza del codice affetto da smell.

In seguito a questo risultato, abbiamo creato dei test case relativi alle funzionalità del plugin che non hanno mostrato problemi, in quanto il testing effettuato nella versione precedente non è sufficiente a darci una risposta affidabile. Sono stati quindi testati i detector del plugin. Tale testing ha mostrato l’effettivo funzionamento delle restanti funzionalità.

Il testing di regressione è stato lanciato prima di apportare le modifiche richieste dalle CR, al termine dell’implementazione di ogni CR e alla fine di tutte le modifiche apportate al sistema.

Nessuna funzionalità del sistema è stata intaccata dalle CR.

Di seguito è riportata la pianificazione dei test sulle funzionalità di detection del plugin originale. Il category partition applicato sul parametro project non è stato applicato in maniera esaustiva in quanto questo testing è stato fatto solo per confermare il funzionamento dei detector.

*Test item*

|  |  |  |
| --- | --- | --- |
| ***ID*** | ***Classes*** | ***Caratteristica da testare*** |
| ET\_System | ConverterUtilities.java  TestSmellUtilities.java  EagerTestStructural.java | Il detector è in grado di identificare lo smell? |
| GF\_System | ConverterUtilities.java  TestSmellUtilities.java  GeneralFixtureStructural.java | Il detector è in grado di identificare lo smell? |
| LOC\_System | ConverterUtilities.java  TestSmellUtilities.java  LackOfCohesionOfTextSmellTextual.java | Il detector è in grado di identificare lo smell? |

*Criteri pass/fail*

|  |  |
| --- | --- |
| ***Fail*** | Il test non ha trovato nessun fault. |
| ***Pass*** | Il test ha trovato un fault. |

|  |  |
| --- | --- |
| ***Parametri*** | * Project p |
| ***Categorie*** | * C\_1: istanze di smell nel project |
| ***Scelte*** | **C\_1**   * *PS\_1*: Progetto contiene 1 o più istanze di smell * *PS\_2*: Progetto non contiene istanze di smell |

|  |  |
| --- | --- |
| ***Vincoli*** | Nessun Vincolo |

**Test Frame ET\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Reg\_Sys\_1 | *PS\_1* | Il detector individua uno o più smell |
| TC\_Reg\_Sys\_2 | *PS\_2* | Il detector non individua nessuno smell |

**Test Frame GF\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Reg\_Sys\_3 | *PS\_1* | Il detector individua uno o più smell |
| TC\_Reg\_Sys\_4 | *PS\_2* | Il detector non individua nessuno smell |

**Test Frame LOC\_System**

|  |  |  |
| --- | --- | --- |
| ***Codice*** | ***Test Frame*** | ***Oracolo*** |
| TC\_Reg\_Sys\_5 | *PS\_1* | Il detector individua uno o più smell |
| TC\_Reg\_Sys\_6 | *PS\_2* | Il detector non individua nessuno smell |