Università Degli Studi Di Salerno

Facoltà di Scienze MM.FF.NN.

Corso di Laurea Magistrale in Informatica

INGEGNERIA, GESTIONE ED EVOLUZIONE

DEL SOFTWARE

AA 2020/2021

DARTS

DOCUMENTAZIONE DI MANUTENZIONE:

ANALISI

Versione 1.0

3 Luglio 2021

Coordinatore del progetto:

Nome
Prof. Andrea De Lucia

Partecipanti:

Nome	Matricola
FS - Fasano Salvatore	0522500891
NA - Natale Alessia	0522501110
CG - Castaldo Giusy	0522501106

Revision History

Data	Versione	Descrizione	Autori
09/05/2021	0.1	Stesura documento iniziale.	Fasano Salvatore Natale Alessia Castaldo Giusy
28/06/2021	0.2	Aggiunta sezione metriche in Impact Analysis	Fasano Salvatore Natale Alessia Castaldo Giusy
03/07/2021	1.0	Aggiunta informazioni testing	Fasano Salvatore Natale Alessia Castaldo Giusy

Indice

1. Scopo del documento

In questo documento saranno presentati gli obiettivi del processo di manutenzione del progetto esistente DARTS: in particolare si provvederà ad effettuare le modifiche specificate nel documento di “Identificazione e classificazione delle modifiche richieste”. Sarà effettuato uno studio del sistema attuale, in modo da evidenziare le relazioni tra gli artefatti esistenti e le eventuali modifiche necessarie per l’introduzione dei cambiamenti elencati; si analizzerà quindi come tali modifiche impattino il sistema esistente e si valuterà se conviene introdurre tali cambiamenti in un’ottica costi/benefici e rischi aggiunti.
Nel caso in cui i cambiamenti vengano approvati, si provvederà con la pianificazione delle fasi di progettazione ed implementazione dei cambiamenti.

2. Panoramica del sistema attuale

Il sistema attuale consente di identificare ed effettuare il refactoring di diversi test smell, lo scopo è di consentire una migliore manutenibilità del codice di test e la possibilità di renderlo effettivo nelle pratiche di Test As Documentation, ovvero utilizzare i casi di test per comprendere velocemente le funzionalità ed il comportamento del software sviluppato.
In alcuni studi è stato dimostrato che molti sviluppatori, anche di seniority avanzata, non riescono a rilevare senza alcun supporto alcune tipologie di test smell, da qui la possibilità di creare un plugin capace di effettuare non solo la rilevazione ma anche le modifiche necessarie per risolvere il problema.
In questo caso specifico DARTS è stato pensato come plugin per gestire codice Java in software IntelliJ.
In particolare, allo stato attuale sono supportati i seguenti test smell:

- Eager Test
- General Fixture
- Lack Of Cohesion.

2.1 Attori e funzionalità

Il sistema 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.

2.2 Design, implementazione e testing del sistema attuale

2.2.1 Design

L’architettura del sistema DARTS si basa su due livelli principali:

- 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.

2.2.2 Implementazione

L'implementazione del software è stata effettuata in riferimento all'architettura descritta: i due livelli presentati sono, a loro volta, costituiti da sottosistema; tali sottosistemi vengono mappati all'interno della struttura del software come packages.
L'interface layer è costituito da due sottosistemi:

- Extension: contiene tutta la logica necessaria per consentire al sistema di attivarsi automaticamente quando si cerca di effettuare un commit;
- WindowConstruction: contiene tutta la logica necessaria per mostrare le informazioni mediante una GUI basata su Java Swing, dando anche la possibilità all'utente di interagire col sistema.

L'application layer è anch'esso costituito da due sottosistemi:

- TestSmellDetection: contiene tutta la logica necessaria per consentire l'identificazione degli smell, è stato utilizzato lo Strategy Design Pattern che, mediante la creazione di una interfaccia IDetector, permette di implementare poi delle strategie per gestire l'identificazione dei vari test smell con semplicità;
- Refactoring: contiene tutta la logica necessaria per consentire le attività di refactoring dei test smell dopo la fase di identificazione (per lo più vengono utilizzate funzionalità fornite dalla libreria Intellij PSI), anche in questo caso è stato utilizzato lo Strategy Design Pattern con un'interfaccia IRefactor, che ci consente poi di implementare in modo rapido strategie per gestire il refactoring dei vari test smell.

2.2.3 Testing

Il sistema allo stato attuale non presenta l'implementazione di casi di test, probabilmente dovuta alle complicazioni causate dal dover gestire i test in un IDE.

3. Analisi delle modifiche richieste

Ci sono due diverse tipologie di modifiche richieste: refactoring della struttura del progetto per essere conforme al nuovo standard definito da IntelliJ ed aggiunta del supporto ai tre nuovi test smell:

- Mystery Guest
- Hard Coded test Data
- Test Data Duplication.

Ovviamente deve essere mantenuto correttamente il supporto ai test smell precedentemente supportati.

4. Individuazione della soluzione progettuale

4.1 Problematiche affrontate

Issue 1: Quali strumenti si utilizzano per effettuare le operazioni di identificazione e refactoring dei nuovi test smell?

Proposal 1.1
Utilizzare strumenti forniti da IntelliJ.
Proposal 1.2
Sviluppare strumenti in autonomia.

Criterion 1.1
Effettuare le operazioni di identificazione e refactoring con costi ridotti.

Argument 1.1
IntelliJ mette a disposizione degli sviluppatori la libreria PSI. Tale libreria include tutti gli strumenti necessari per effettuare operazioni sull'AST del programma utente. Utilizzando questo strumento non ci sarebbero costi aggiuntivi ma ci si lega alla piattaforma IntelliJ.
Argument 1.2
Sviluppando gli strumenti necessari per creare un AST e per effettuarci delle operazioni è un lavoro costoso ma che consentirebbe il riutilizzo della soluzione proposta anche al di fuori della piattaforma IntelliJ.

Proposal 1.1
Analizzando le argomentazioni Argument 1.1 e Argument 1.2, l'Issue 1 è stata risolta applicando la Proposal 1.1.

Issue 2: Quali sorgenti dati esterne supportare per l'identificazione dello smell "Mystery Guest"?

Proposal 2.1
Tutte le possibili sorgenti.
Proposal 2.2
Solamente file di testo.

Criterion 2.1
Ottenere un supporto base del test smell.

Argument 2.1
Per sviluppare un supporto completo del test smell bisogna inserire logica in grado di gestire tutte le possibili sorgenti dati e crearne delle rappresentazioni significanti nel programma.
Argument 2.2
Si può inserire un supporto base al test smell in modo da gestire solamente file di testo.

Proposal 2.1
Analizzando le argomentazioni Argument 2.1 e Argument 2.2, l'Issue 2 è stata risolta applicando la Proposal 2.2.

4.2 Soluzione individuata

La soluzione individuata consiste nell'unione di tutte le "resolution" ottenute nel paragrafo 4.1. In dettaglio, la soluzione individuata dovrà avere le seguenti caratteristiche:

- Utilizzare strumenti forniti da IntelliJ per identificare ed effettuare il refactoring;
- Supportare solamente file di testo per il test smell "Mystery Guest".

4.3 Soluzioni alternative

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

5. Identificazione dell'Impact Set

Valutiamo come le modifiche che devono essere implementate impatteranno il sistema esistente, in particolare individuiamo gli artefatti che dovranno subire delle modifiche per consentire il corretto funzionamento del sistema.
Si precisa che, data la doppia natura delle modifiche da effettuare, la restante parte del capitolo riguarderà solamente le modifiche per l'aggiunta del supporto dei nuovi test smell; per portare l'attuale struttura del progetto ai nuovi standard IntelliJ bisognerà solamente effettuare delle modifiche all'organizzazione dei file (deve essere introdotta la cartella "main" che contiene tutti i diversi package) ed aggiungere i file per la configurazione di Gradle, nessuna modifica riguarderà gli artefatti presenti.
Per individuare gli artefatti del sistema impattati dalle modifiche abbiamo utilizzato un approccio top-down: grazie alla documentazione già presente del sistema DARTS abbiamo evidenziato i sottosistemi che necessitano di modifiche ed aggiunte; ovviamente ciò ci ha consentito progressivamente di avanzare ai livelli inferiori l'analisi del sistema (e l'individuazione di artefatti impattati).
Nelle seguenti tabelle viene quindi indicato:

- Artefatto: la componente impattata dalle modifiche;
- Impatto: livello di impatto sul sistema software attuale, può essere:
    - FORTE: se ci sono modifiche pesanti o se tali modifiche necessitano la sostituzione dell'intero artefatto;
    - MEDIO: se ci sono modifiche sostanziali che non coinvolgono un cambiamento eccessivo della struttura dell'artefatto;
    - DEBOLE: se ci sono modifiche marginali;
- Descrizione: informazioni circa la gestione dell'impatto.

5.1 Starting Impact Set

5.1.1 System Design Impact

Artefatto	Impatto	Descrizione
Decomposizione in sottosistemi	NESSUNO	
Mapping Hardware/Software	NESSUNO	
Flusso di controllo globale	NESSUNO	

5.1.2 Object Design Impact

Artefatto	Impatto	Descrizione
Packages	NESSUNO	
Comunicazione tra packages	NESSUNO	
Classi ed interfacce	DEBOLE	Nell'artefatto andranno aggiunte le classi necessarie per gestire la rappresentazione dei nuovi test smell.

5.1.3 Classes Impact

Artefatto	Impatto	Descrizione
IDetector.java	DEBOLE	Bisogna aggiungere all'interfaccia la firma del metodo per gestire l'identificazione del nuovo test smell.
CommitWindowFactory.java	MEDIO	Bisogna aggiungere alla GUI delle schede per gestire i nuovi test smell, ciò implicherà anche delle modifiche alle firme dei metodi definiti.
StructualDetectionAction.java	MEDIO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
TextualDetectionAction.java	MEDIO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
RefactorWindow.java	MEDIO	Bisogna aggiungere un nuovo costruttore che gestisce la creazione di pannelli per i nuovi test smell.

5.2 Candidate Impact Set

5.2.1 System Design Impact

Artefatto	Impatto	Descrizione

Decomposizione in sottosistemi	NESSUNO	
Mapping Hardware/Software	NESSUNO	
Flusso di controllo globale	NESSUNO	

5.2.2 Object Design Impact

Artefatto	Impatto	Descrizione
Packages	NESSUNO	
Comunicazione tra packages	NESSUNO	
Classi ed interfacce	DEBOLE	Nell'artefatto andranno aggiunte le classi necessarie per gestire la rappresentazione dei nuovi test smell.

5.2.3 Classes Impact

Artefatto	Impatto	Descrizione
IDetector.java	DEBOLE DIRETTO	Bisogna aggiungere all'interfaccia la firma del metodo per gestire l'identificazione del nuovo test smell.
TestSmellStructuralDetector.java	DEBOLE INDIRETTO	Dato che implementa l'interfaccia IDetector, deve aggiungere l'implementazione del metodo introdotto.
TestSmellTextualDetector.java	DEBOLE INDIRETTO	Dato che implementa l'interfaccia IDetector, deve aggiungere l'implementazione del metodo introdotto.
CommitWindowFactory.java	MEDIO DIRETTO	Bisogna aggiungere alla GUI delle schede per gestire i nuovi test smell, ciò implicherà anche delle modifiche alle firme dei metodi definiti.
StructualDetectionAction.java	MEDIO DIRETTO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare

		modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
TextualDetectionAction.java	MEDIO DIRETTO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
RefactorWindow.java	MEDIO DIRETTO	Bisogna aggiungere un nuovo costruttore che gestisce la creazione di pannelli per i nuovi test smell.

5.3 Actual Impact Set

5.3.1 System Design Impact

Artefatto	Impatto	Descrizione
Decomposizione in sottosistemi	NESSUNO	
Mapping Hardware/Software	NESSUNO	
Flusso di controllo globale	NESSUNO	

5.3.2 Object Design Impact

Artefatto	Impatto	Descrizione
Packages	NESSUNO	
Comunicazione tra packages	NESSUNO	
Classi ed interfacce	DEBOLE	Nell'artefatto andranno aggiunte le classi necessarie per gestire la rappresentazione dei nuovi test smell.

5.3.3 Classes Impact

Artefatto	Impatto	Descrizione
IDetector.java	DEBOLE	Bisogna aggiungere all'interfaccia la firma del metodo per gestire l'identificazione del nuovo test smell.

TestSmellStructuralDetector.java	DEBOLE	Dato che implementa l'interfaccia IDetector, deve aggiungere l'implementazione del metodo introdotto.
TestSmellTextualDetector.java	DEBOLE	Dato che implementa l'interfaccia IDetector, deve aggiungere l'implementazione del metodo introdotto.
CommitWindowFactory.java	MEDIO	Bisogna aggiungere alla GUI delle schede per gestire i nuovi test smell, ciò implicherà anche delle modifiche alle firme dei metodi definiti.
WarningWindow.java	DEBOLE	Dato che utilizza metodi forniti dalla classe precedente, dovrà essere adeguata.
StructualDetectionAction.java	MEDIO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
TextualDetectionAction.java	MEDIO	Bisogna aggiungere l'identificazione dei nuovi test smell ed effettuare modifiche dato che vengono invocati metodi di CommitWindowFactory.java.
RefactorWindow.java	MEDIO	Bisogna aggiungere un nuovo costruttore che gestisce la creazione di pannelli per i nuovi test smell.
CommitFactory.java	MEDIO	Bisogna aggiungere all'analisi automatica (effettuata al momento di commit) i nuovi test smell.

5.4 Discovered Impact Set

5.4.1 Classes Impact

Artefatto	Impatto	Descrizione
WarningWindow.java	DEBOLE	Dato che utilizza metodi forniti dalla classe precedente, dovrà essere adeguata.

CommitFactory.java	MEDIO	Bisogna aggiungere all'analisi automatica (effettuata al momento di commit) i nuovi test smell.

5.5 False Positive Impact Set

Non sono state pianificate modifiche non presenti nell'Actual Impact Set.

5.6 Metriche

Si presentano le metriche relative ai dati del lavoro svolto.

Gli insiemi impattati hanno le seguenti cardinalità:

- Starting Impact Set |SIS| = 5
- Candidate Impact Set |CIS| = 7
- Actual Impact Set |AIS| = 9
- Discovered Impact Set |CIS| = 2
- False Positive Impact Set |FPIS| = 0

5.6.1 Recall

Dopo aver effettuato le operazioni di modifica ed aver stabilito l'Actual Impact Set, è stato riscontrato che la recall ha un valore di 0.8, ovvero come presentato nelle tabelle precedenti, ci sono stati degli artefatti impattati non presenti all'interno del Candidate Impact Set.

$$Recall = \frac{|CIS \cap AIS|}{|AIS|} = \frac{7}{9} = 0.8$$

5.6.2 Precision

Dopo aver effettuato le operazioni di modifica ed aver stabilito l'Actual Impact Set, è stato riscontrato che la precision ha un valore di 1, ovvero come presentato nelle tabelle precedenti, non ci sono stati degli artefatti presenti all'interno del Candidate Impact Set che in realtà non sono stati impattati.

$$Precision = \frac{|CIS \cap AIS|}{|CIS|} = \frac{7}{7} = 1$$

5.6.3 Adeguacy

È stata effettuata una valutazione dell'adeguatezza dell'approccio di Impact Analysis utilizzato. Dato che l'Actual Impact Set è caratterizzato da elementi che non erano stati rilevati nel Candidate Impact Set, l'adeguatezza misurata mediante la metrica di Inclusiveness è 0.

5.6.4 Effectiveness

5.6.4.1 Ripple-Sensitivity

$$Amplification = \frac{|IISO|}{|DISO|} = \frac{2}{9} = 0{,}22$$

5.6.4.2 Sharpeness

$$ChangeRate = \frac{|CIS|}{|System|} = \frac{7}{48} = 0{,}14$$

6. Studio di fattibilità

6.1 Identificazione, descrizione e valutazione dei costi

Identificazione	Valutazione	Motivazioni
Supporto di nuovi test smell	MEDIO	La modifica richiesta prevede la creazione della rappresentazione, della logica di identificazione e di refactoring dei nuovi test smell.
Organizzazione del lavoro nel team di sviluppo	FORTE	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	FORTE	Implementare tecniche di testing in questo progetto risulta particolarmente complesso a causa delle interazioni con le librerie fornite da IntelliJ.

6.2 Identificazione, descrizione e classificazione dei benefici

Identificazione	Classificazione	Motivazioni
Standardizzazione con struttura definita da IntelliJ	FORTE	Rispettando lo standard di IntelliJ il plugin potrà essere installato anche sulle nuove versioni degli IDE IntelliJ e godere delle nuove funzionalità.

Miglioramento manutenibilità ed utilizzo di Test As Documentation	FORTE	Supportando un numero maggiore di test smell si fa in modo che l'utente finale possa produrre codice di test con un grado maggiore di manutenibilità e comprensibilità

6.3 Identificazione, descrizione e classificazione dei rischi

Identificazione	Probabilità	Motivazioni
Errori di identificazione dei test smell	MEDIA	Le operazioni di identificazione dei nuovi test smell potrebbero segnalare falsi positivi.
Strategia		
Nel caso in cui venissero identificati dei falsi positivi per un dato test smell, dato che le attività di refactoring vengono avviate solo con il permesso dell'utente, l'utente potrebbe decidere di non effettuare il refactoring di una istanza di un test smell grazie ai consigli visualizzati nell'interfaccia grafica (Contingency Plan).		
Errori di refactoring dei test smell	MEDIA	Le operazioni di refactoring automatico dei nuovi test smell potrebbero non essere svolte correttamente causando l'introduzione di errori nel codice di test dell'utente.
Strategia		
Effettuando un'analisi dei test smell specificati nella change request, risulta improbabile introdurre errori in caso di "Hard Coded Test Data" e "Test Code Duplication"; gestire la modifica in caso di "Mystery Guest" può invece causare errori: in questo caso si decide eliminare il rischio effettuando solo il refactoring della riga in cui viene dichiarato un oggetto contenente informazioni esterne al codice (e non tutte le posizioni in cui esso viene utilizzato). Si precisa che in ogni caso si implementerà la funzione di refactoring come cambiamento "univoco": eseguendo una operazione di undo si tornerà allo stato precedente alla modifica.		

6.4 Risultati previsti

Grazie all'utilizzo dello Strategy Design Pattern per l'identificazione ed il refactoring dei test smell, risulta semplice estenderne il sistema con dei nuovi; sicuramente lo possiamo notare anche dalla dimensione del Candidate Impact Set: ci sono poche modifiche da effettuare e sono modifiche che hanno un medio-basso impatto sul sistema attuale. Tenendo in considerazione anche costi, benefici e rischi risulta possibile effettuare le modifiche richieste in modo semplice.

7. Testing

Si vogliono progettare delle attività di testing per verificare la presenza di errori ed, in caso siano presenti, risolverli. In questo capitolo si illustreranno le strategie di testing che si intende adottare, le funzionalità da testare e gli strumenti per condurre il testing.

7.1 Obiettivi del testing

Come anticipato, lo scopo è di individuare la presenza di faults all'interno del sistema e condurre attività di debug. Diciamo che un test ha successo quando, dato un input al sistema, il risultato ottenuto è diverso dal valore previsto, chiamato oracolo. Dall'analisi effettuata è possibile intuire che ogni componente del sistema è legata agli strumenti forniti da IntelliJ, ciò rende notevolmente complesse e costose le attività di testing. Di fatto bisogna pensare che il sistema in esame è un plugin per l'ambiente di sviluppo integrato IntelliJ. Ciò significa che, sia in fase di identificazione degli 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. A causa di tali problemi, come precisato nell'analisi del sistema attuale, non sono presenti casi di test al momento, ciò significa che non ci sarà possibilità di effettuare test di regressione. Dato che si vuole incrementare il livello di fiducia circa il corretto funzionamento del sistema senza aumentare esageratamente i costi a causa della fase di testing, verrà effettuato test di unità relativo alle classi riguardanti l'identificazione testuale dei test smell supportati, in particolare il package di riferimento è main.testSmellDetection.textualRules.

7.2 Approccio

Inizialmente verrà testata l'identificazione dei test smell supportati correntemente (Eager Test, General Fixture e Lack Of Cohesion); si decide poi di testare l'identificazione di ogni nuovo test smell immediatamente dopo la sua implementazione, in modo da rilevare

subito eventuali errori. Per effettuare il testing di unità delle classi contenute in main.testSmellDetection.textualRules sarà necessario sfruttare utility di testing fornite da IntelliJ, in particolare le classi di test dovranno estendere la classe LightJavaCodeInsightFixtureTestCase: tale utility crea un ambiente fittizio in cui l'utilizzo degli strumenti forniti da IntelliJ potrà andare a buon fine, a basso livello tale utility utilizza JUnit 4. Le classi presenti all'interno del pacchetto citato sono:

- EagerTestTextual
- GeneralFixtureTextual
- LackOfCohesionOfTestSmellTextual.

Al termine delle modifiche, il package comprenderà ulteriori tre classi:

- HardCodedTestDataTextual
- MysteryGuestTextual
- TestCodeDuplicationTextual.

Ognuna delle classi citate conterrà un unico metodo statico che ha come input la classe di test su cui bisogna valutare la presenza del test smell specifico.
Si decide quindi si eseguire testing funzionale (Black-Box Testing), in particolare utilizzando il metodo Category Partition in quanto risulta immediata la rilevazione delle categorie per gli input: l'unico paramento è la classe di test e le categorie indicano la presenza o meno del test smell.

7.3 Test Case Plan

Si precisa che i titoli dei seguenti paragrafi hanno la seguente struttura:

- RF: Requisito funzionale
- IDT: Identificazione Testuale
- XXXX: Iniziali del test smell in esame.

7.3.1 RF_IDT_ET: Identificazione Eager Test

Parametro: Classe di test	
Presente [PE]	1. Eager Test non presente nella classe di test [PE_NP] 2. Eager Test presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito

TC1_1	PE1	Corretto
TC1_2	PE2	Corretto

7.3.2 RF_IDT_GF: Identificazione General Fixture

Parametro: Classe di test	
Presente [PE]	1. General Fixture non presente nella classe di test [PE_NP] 2. General Fixture presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito
TC2_1	PE1	Corretto
TC2_2	PE2	Corretto

7.3.3 RF_IDT_LOC: Identificazione Lack Of Cohesion

Parametro: Classe di test	
Presente [PE]	1. Lack Of Cohesion non presente nella classe di test [PE_NP] 2. Lack Of Cohesion presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito
TC3_1	PE1	Corretto
TC3_2	PE2	Corretto

7.3.4 RF_IDT_HCTD: Identificazione Hard Coded Test Data

Parametro: Classe di test	
Presente [PE]	1. Hard Coded Test Data non presente nella classe di test [PE_NP]

	2. Hard Coded Test Data presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito
TC4_1	PE1	Corretto
TC4_2	PE2	Corretto

7.3.5 RF_IDT_MG: Identificazione Mystery Guest

Parametro: Classe di test	
Presente [PE]	1. Mystery Guest non presente nella classe di test [PE_NP] 2. Mystery Guest presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito
TC5_1	PE1	Corretto
TC5_2	PE2	Corretto

7.3.6 RF_IDT_TCD: Identificazione Test Code Duplication

Parametro: Classe di test	
Presente [PE]	1. Test Code Duplication non presente nella classe di test [PE_NP] 2. Test Code Duplication presente nella classe di test [PE_P]

Codice	**Combinazione**	Esito
TC6_1	PE1	Corretto
TC6_2	PE2	Corretto

7.4 Test Case Specification

Nelle seguenti tabelle verrà indicato nella sezione “Specifica degli input”, sotto la voce “Valore”, il nome del file che contiene la classe creata per il particolare caso di test

piuttosto che l'intero codice della classe. È possibile vedere tali file all'interno del progetto nel percorso src/test/resources/testdata/test.

7.4.1 Identificazione Eager Test

7.4.1.1 Eager Test Non Presente

Nome Test Case	Identificazione Eager Test		
ID Test Case	TC1_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	EagerTestNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Eager Test nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.1.2 Eager Test Presente

Nome Test Case	Identificazione Eager Test		
ID Test Case	TC1_2		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE2	EagerTestPresent.java
Oracolo	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Eager Test nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.2 Identificazione General Fixture

7.4.2.1 General Fixture Non Presente

Nome Test Case	Identificazione General Fixture		
ID Test Case	TC2_1		
Specifica degli input	Input	Scelta	Valore

	Classe di test	PE1	GeneralFixtureNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di General Fixture nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.2.2 General Fixture Presente

Nome Test Case	Identificazione General Fixture		
ID Test Case	TC2_2		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE2	GeneralFixturePresent.java
Oracolo	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Genera lFixture nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.3 Identificazione Lack Of Cohesion

7.4.3.1 Lack Of Cohesion Non Presente

Nome Test Case	Identificazione Lack Of Cohesion		
ID Test Case	TC3_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	LackOfCohesionNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Lack Of Cohesion nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.3.2 Lack Of Cohesion Presente

Nome Test Case	Identificazione Lack Of Cohesion		
ID Test Case	TC3_2		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE2	LackOfCohesionPresent.java
Oracolo	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Lack Of Cohesion nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.4 Identificazione Hard Coded Test Data

7.4.4.1 Hard Coded Test Data Non Presente

Nome Test Case	Identificazione Hard Coded Test Data		
ID Test Case	TC4_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	HardCodedTestDataNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Hard Coded Test Data nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.4.2 Hard Coded Test Data Presente

Nome Test Case	Identificazione Hard Coded Test Data		
ID Test Case	TC4_2		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE2	HardCodedTestDataPresent.java
Oracolo	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate		

	istanze di Hard Coded Test Data nella classe di test in input
Dipendenze con altri test cases	N/A

7.4.5 Identificazione Mystery Guest

7.4.5.1 Mystery Guest Non Presente

Nome Test Case	Identificazione Mystery Guest		
ID Test Case	TC5_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	MysteryGuestNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Mystery Guest nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.5.2 Mystery Guest Presente

Nome Test Case	Identificazione Mystery Guest		
ID Test Case	TC5_2		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE2	MysteryGuestPresent.java
Oracolo	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Mystery Guest nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.6 Identificazione Test Code Duplication

7.4.6.1 Test Code Duplication Non Presente

Nome Test Case	Identificazione Test Code Duplication

ID Test Case	TC6_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	TestCodeDuplicationNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Test Code Duplication nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.4.6.2 Test Code Duplication Presente

Nome Test Case	Identificazione Test Code Duplication		
ID Test Case	TC6_1		
Specifica degli input	Input	Scelta	Valore
	Classe di test	PE1	TestCodeDuplicationNotPresent.java
Oracolo	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Test Code Duplication nella classe di test in input		
Dipendenze con altri test cases	N/A		

7.5 Test Execution Report

7.5.1 Identificazione Eager Test

7.5.1.1 Eager Test Non Presente

ID Test Case	TC1_1
Tester	Salvatore Fasano
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Eager Test nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A

Esito	Corretto

7.5.1.2 Eager Test Presente

ID Test Case	TC1_2
Tester	Salvatore Fasano
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Eager Test nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.2 Identificazione General Fixture

7.5.2.1 General Fixture Non Presente

ID Test Case	TC2_1
Tester	Alessia Natale
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di General Fixture nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.2.2 General Fixture Presente

ID Test Case	TC2_2
Tester	Alessia Natale
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di General Fixture nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A

Esito	Corretto

7.5.3 Identificazione Lack Of Cohesion

7.5.3.1 Lack Of Cohesion Non Presente

ID Test Case	TC3_1
Tester	Giusy Castaldo
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Lack Of Cohesion nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.3.2 Lack Of Cohesion Presente

ID Test Case	TC3_2
Tester	Giusy Castaldo
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Lack Of Cohesion nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.4 Identificazione Hard Coded Test Data

7.5.4.1 Hard Coded Test Data Non Presente

ID Test Case	TC4_1
Tester	Salvatore Fasano
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Hard Coded Test Data nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed

Anomalie	N/A
Esito	Corretto

7.5.4.2 Hard Coded Test Data Presente

ID Test Case	TC4_2
Tester	Salvatore Fasano
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Hard Coded Test Data nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.4.5 Identificazione Mystery Guest

7.5.5.1 Mystery Guest Non Presente

ID Test Case	TC5_1
Tester	Alessia Natale
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Mystery Guest nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.5.2 Mystery Guest Presente

ID Test Case	TC5_2
Tester	Alessia Natale
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Mystery Guest nella classe di test in input
Output Atteso	Executed

Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.6 Identificazione Test Code Duplication

7.5.6.1 Test Code Duplication Non Presente

ID Test Case	TC6_1
Tester	Giusy Castaldo
Risultati della procedura	Il sistema presenta un messaggio in cui avverte l'utente che non sono presenti istanze di Test Code Duplication nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.5.6.2 Test Code Duplication Presente

ID Test Case	TC6_2
Tester	Giusy Castaldo
Risultati della procedura	Il sistema presenta un messaggio in cui mostra all'utente le posizioni in cui sono state identificate istanze di Test Code Duplication nella classe di test in input
Output Atteso	Executed
Output Sistema	Executed
Anomalie	N/A
Esito	Corretto

7.6 Conclusioni

Gli obiettivi di testing prefissati sono stati raggiunti: le funzionalità di identificazione testuale dei test smell sono state correttamente testate grazie agli strumenti presentati nell'introduzione.