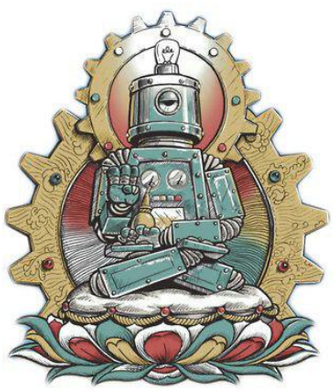
UNIVERSITÀ DEGLI STUDI DI SALERNO
CORSO DI INGEGNERIA DEL SOFTWARE

ASCETIC

Automated Code Smell Identification and Correction

REQUIREMENT ANALYSIS DOCUMENT

VERSION 1.1



8 gennaio 2019

Coordinatore Progetto:

Nome	**Matricola**
Manuel De Stefano	0522500633

Partecipanti:

Nome	**Matricola**
Amoriello Nicola	0512104742
Di Dario Dario	0512104758
Gambardella Michele Simone	0512104502
Iovane Francesco	0512104550
Pascucci Domenico	0512102950
Patierno Sara	0512103460

Revision History:

Data	**Versione**	**Descrizione**	**Autore**
26/10/2018	1.0	Prima stesura	Tutto il Team
08/01/2019	1.1	Correzione e aggiunta di "CodeSmell" al class Diagram	Tutto il Team

Indice

1 Introduction

1.1 Scope of the system

Durante il ciclo di vita di un software i cambiamenti sono inevitabili.
La manutenzione, che sia per correggere bug o per aggiungere nuove funzionalità, porta a un graduale deperimento del codice, il quale non inficia la correttezza del programma ma porta a debolezze di progettazione: i cosiddetti code smell.
Anche uno sviluppatore navigato e ben attento a queste problematiche può cadere in errore, a causa di tempistiche strette o codice ripreso da altri developer.
ASCETIC nasce per permettere a utenti, esperti e non, di refactorizzare il codice in maniera comoda e intelligente.
Il plug-in, sviluppato per l'IDE intelliJ IDEA, consente di analizzare il progetto, rilevando 4 possibili tipologie di smell(Blob, Promiscuous Package, Feature Envy, Missplaced Class) e di effettuare un eventuale correzione automatica.

1.2 Overview

ASCETIC (Inizialmente TACOR) è un'opera di reengineering, la quale ha l'obiettivo di migliorare le funzionalità già offerte dal precedente plug-in, offrirne delle nuove ed effettuare un restyling dell'interfaccia grafica per rendere migliore l'esperienza d'uso. ASCETIC si rivolge ad un pubblico composto principalmente da sviluppatori Java che utilizzano l'IDE IntelliJ IDEA e si pone l'obiettivo di facilitare l'aggiustamento del codice sorgente. Lo sviluppatore ha la possibilità di:

- Applicare la correzione automatica proposta dal sistema dei Code Smell rilevati durante la fase d'analisi. E' possibile applicare tale soluzione a tutti gli elementi rilevati oppure ad uno specifico insieme selezionato dall'utente.
- Porre determinati Smell, scelti ad hoc dallo sviluppatore, come "falsi positivi", cosicché il sistema ignori quella determinata sezione di codice fino a nuova modifica.
- Utilizzare la funzione "Reminder", la quale fornisce un comodo promemoria così da poter affrontare manualmente la correzione dello smell selezionato.

1.3 Definitions, acronyms, and abbreviations

Code Smell : Sezioni di codice scritte in maniera non ottimale, presentanti debolezze di progettazione che riducono la qualità del codice, non compromettendone il funzionamento.

ASCETIC : acronimo per Automated Code Smell Identification and Correction.

2 Current system

2.1 Requisiti Funzionali

RF 1 - Ricerca code smell

Il sistema ricerca i code smell all'interno del codice java.

RF 1.1: Il sistema permette l'analisi di code smell di tipo Blob,
RF 1.2: Il sistema permette l'analisi di code smell di tipo Misplaced Class
RF 1.3: Il sistema permette l'analisi di code smell di tipo Feature Envy
RF 1.4: Il sistema permette l'analisi di code smell di tipo Promiscuos Package

RF 2 - Correzione

Il sistema mostra una possibile soluzione al code smell.

RF 2.1: Il sistema permette di corregere code smell di tipo Feature Envy.
RF 2.2: Il sistema permette di corregere code smell di tipo Misplaced Class.

RF 3 - Refactoring

Il sistema permette allo sviluppatore di risolvere il code smell con degli automatismi del software.

RF 3.1: Il sistema permette di risolvere code smell di tipo Feature Envy.
RF 3.2: Il sistema permette di risolvere code smell di tipo Misplaced Class.

RF 4 - Metriche di Qualità

Il sistema permette di calcolare metriche di qualità.

RF 4.1: Il sistema permette di calcolare metriche di qualità per i metodi.

RF 4.2: Il sistema permette di calcolare metriche di qualità per le classi.

RF 4.3: Il sistema permette di calcolare metriche di qualità per i package.

RF 5 - Estrazione dei Topic

Il sistema consente di estrarre i topic implementati.

RF 6 - Verifica Correzione

Il sistema consente di verificare la correttezza del codice.

2.2 Requisiti Non Funzionali

- **RNF 1 - Performance**
  Il sistema è concepito come utilizzabile da singolo utente, con un'interazione diretta, il tempo di risposta varia in base all'analisi proposta.

- **RNF 2 - Robustezza**
  Il sistema allo stato attuale non soddisfa il requisito di robustezza.

- **RNF 3 - Sicurezza**
  Il sistema si presenta sufficientemente sicuro in quanto il suo ambiente d'azione è locale, quindi non vi è la necessità di protezione da minacce esterne.

- **RNF 4 - Usabilità**
  Il sistema presenta bassi criteri di usabilità, data una scarna e poco intuitiva GUI.

- **RNF 5 - Manutenibilità**
  Il sistema non presenta alcun tipo di documentazione pregressa, oltre ad adottare scelte implementative poco adatte a tale scopo.

- **RNF 6 - Implementazione**
  Il sistema è realizzato interamente in linguaggio Java, sia parte back-end che front-end.

- **RNF 7 - Affidabilità**
  Il sistema allo stato attuale non rispetta il requisito di affidabilità.

2.3 Use Case Model

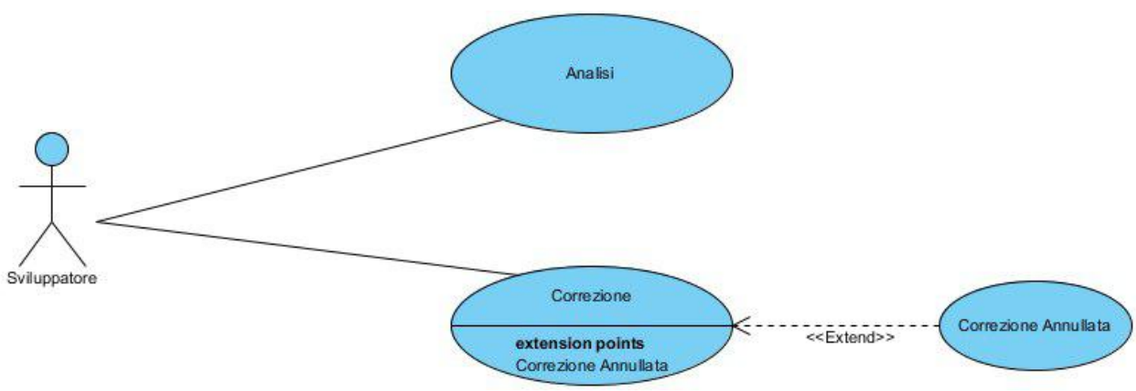


Figura 1: UCD1-Analisi e Correzione

Use case name	**UC 1 - Ricerca di code smell**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente avvia con successo Intellij.	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente su IntelliJ, tramite menù a tendina del plug-in, richiede l'analisi locale sulla classe o sul package designato che vuole correggere, evidenziando che tipologia di code smell analizzare.	
		2. Il sistema analizza il codice ed effettua il calcolo delle metriche strutturali. Sulla base dei dati estrapolati vengono poi mostrati i valori riguardanti le metriche di qualità.
	3. L'utente prende visione del risultato dell'analisi tramite una tabella e varie RadarMap riportanti i 5 termini più frequenti . La tabella riporta la posizione del code smell identificato e le metriche di qualità calcolate(LOC, WNC, RFC, CBO, LCOM, NOA, NOM, NOPA, NOP). L'utente può inoltre, tramite apposito pulsante "Perform Refactor", richiedere il refactoring sullo smell.	
Exit condition	Lo sviluppatore visiona correttamente i dati e il codice affetto da smell.	
Exception condition	Nel passo 1, in caso di codice java non corretto dal punto di vista sintattico, semantico o lessicale si verifica lo UC 1.1.	
Priority	Alta	
Quality requirements	• Tempo di elaborazione massimo: 3 minuti. • Il sistema è in grado di rilevare se il codice è java.	

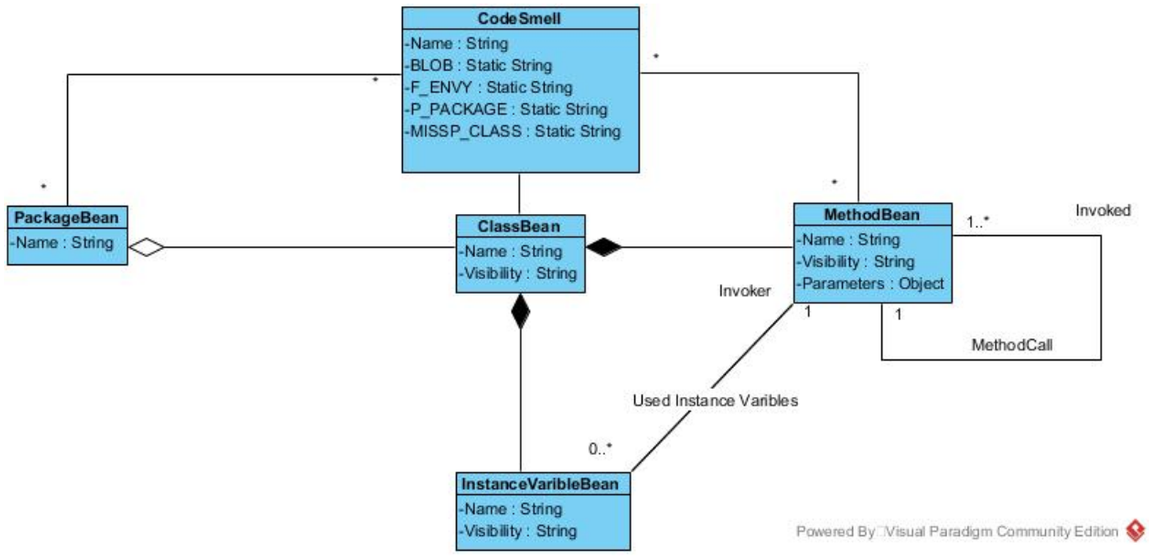
Use case name	**UC 1.1 - Analisi codice non compilabile**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente avvia l'analisi del codice.	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente attende l'elaborazioni richieste dal sistema.	
		2. Il sistema tenta di analizzare il codice ma, trovando testo non compilabile, annulla l'operazione e mostra un messaggio di errore.
	3. L'utente prende visione del messaggio che indica il fallimento dell'operazione.	
Exit condition	Il sistema termina l'esecuzione senza effettuare alcuna modifica	
Exception condition		
Priority	Alta	
Quality requirements	• Tempo di elaborazione massimo: 3 minuti.	

Use case name	**UC 2.1 - Correzione Code smell - Feature Envy**
Participating actors	**Sviluppatore**
Entry condition	L'utente ha visualizzato il codice affetto da smell
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri su cui intervenire per lo smells di tipo "Feature Envy" e avvia gli automatismi per la correzione automatica premendo il tasto "Refactor". **2.** Il sistema, mostra una schermata di riepilogo con delle sezioni mostranti una proposta di correzione confrontata con la situazione precedente all'analisi, e un tasto per apportare tali modifiche. **3.** L'utente, può scegliere di accettare tali modifiche consigliate dal sistema cliccando il tasto "Applica modifiche". **4.** Il sistema, applica la soluzione che precedentemente ha mostrato all'utente. **5.** L'utente prende visione di tutte le modifiche effettuate dal sistema.
Exit condition	Il codice è stato correttamente modificato.
Exception condition	Se nel passo 1 o 3 l'utente chiude la finestra di dialogo e non procede alla correzione si verifica lo UC 2.3
Priority	Alta
Quality requirements	

Use case name	**UC 2.2 - Correzione Code smell - Misplaced Class**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente ha visualizzato il codice affetto da smell	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri su cui intervenire per lo smells di tipo "Misplaced Class" e avvia gli automatismi per la correzione automatica premendo il tasto "Refactor".	
		2. Il sistema, mostra una schermata di riepilogo con delle sezioni mostranti una proposta di correzione confrontata con la situazione precedente all'analisi, e un tasto per apportare tali modifiche.
	3. L'utente, può scegliere di accettare tali modifiche consigliate dal sistema cliccando il tasto "Applica modifiche".	
		4. Il sistema, applica la soluzione che precedentemente ha mostrato all'utente.
	5. L'utente prende visione di tutte le modifiche effettuate dal sistema.	
Exit condition	Il codice è stato correttamente modificato.	
Exception condition	Se nel passo 1 o 3 l'utente chiude la finestra di dialogo e non procede alla correzione si verifica lo UC 2.3	
Priority	Alta	
Quality requirements		

Use case name	**UC 2.3 - Correzione annullata**	
Participating actors	**Sviluppatore**	
Entry Condition	L'utente chiude le finestre di dialogo e non procede alla correzione.	
Flow of events	**UTENTE**	**SISTEMA**
	1. Lo Sviluppatore decide di non correggere il code smell.	
		2. Il Sistema non effettua alcuna operazione e vengono chiuse le finestre di dialogo.
Exit condition	Il Sistema termina l'operazione senza effettuare modifiche al codice.	
Exception condition		
Priority	Alta	
Quality requirements		

2.4 Static Analysis Model



TIPO	NOME	DESCRIZIONE
Boundary	Search	Boundary che si occupa della riproduzione a schermo dei risultati dell'analisi.
	Smell List	Boundary che si occupa della visualizzazione della lista degli smell trovati in fase di analisi.
	Solution	Boundary che rende possibile visualizzare la lista delle soluzioni elaborate dal sistema.
	Result	Boundary che rende possibile visualizzare i risultati dopo l'esecuzione del Refactor.
Control	Search	Control che invia messaggi al Search Entity per ricercare smell ed istanzia lo Smell List Boundary, al quale invia un messaggio per richiedere la visualizzazione della lista degli smell.
	Proposed Solution	Control che crea il Solution Boundary, al quale invia un messaggio per visualizzare le proposte di correzione.
	Refactoring	Control che crea il Result Boundary, al quale invia un messaggio per notificare il successo dell'operazione di Refactoring.
Entity	Search	Entity che conserva i dati persistenti relativi ai Code Smell trovati in fase di analisi.
	CodeSmell	Entità che tiene traccia degli smells, ovvero quali sono e di che tipo.

Gli oggetti presenti nella tabella soprastante sono stati individuati a seguito dell'analisi dinamica sotto riportata.

2.5 Dynamic Analysis Model

2.5.1 Sequence Diagram

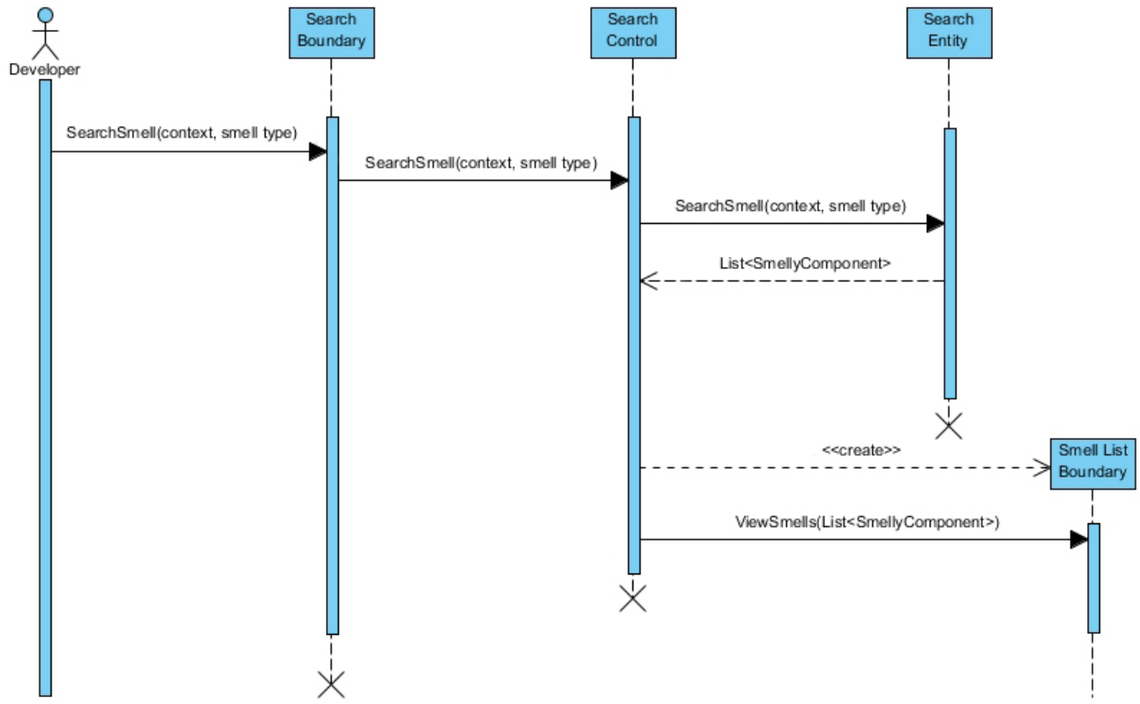


Figura 2: SD1-Analisi

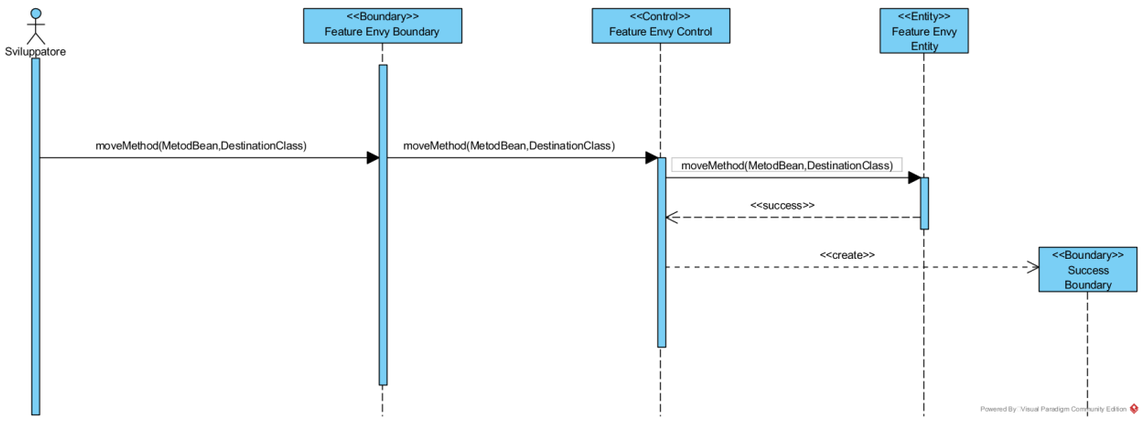


Figura 3: SD2.1-Correzione Feature Envy

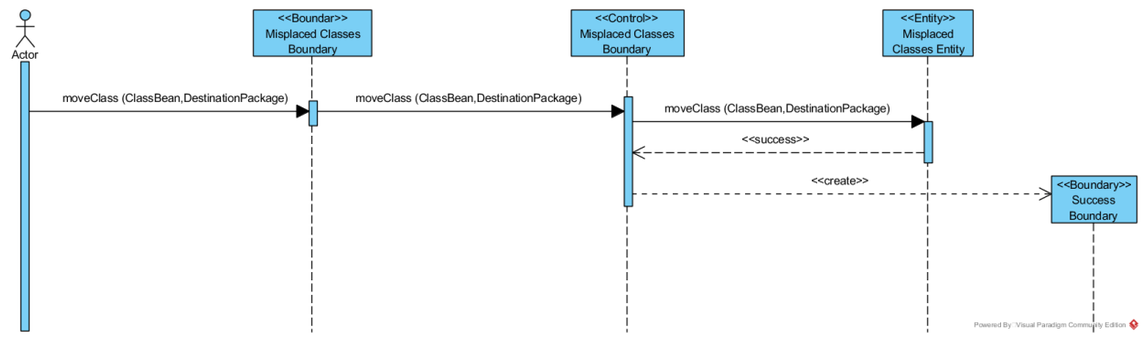


Figura 4: SD2.2-Correzione Misplaced Class

2.5.2 Activity Diagram

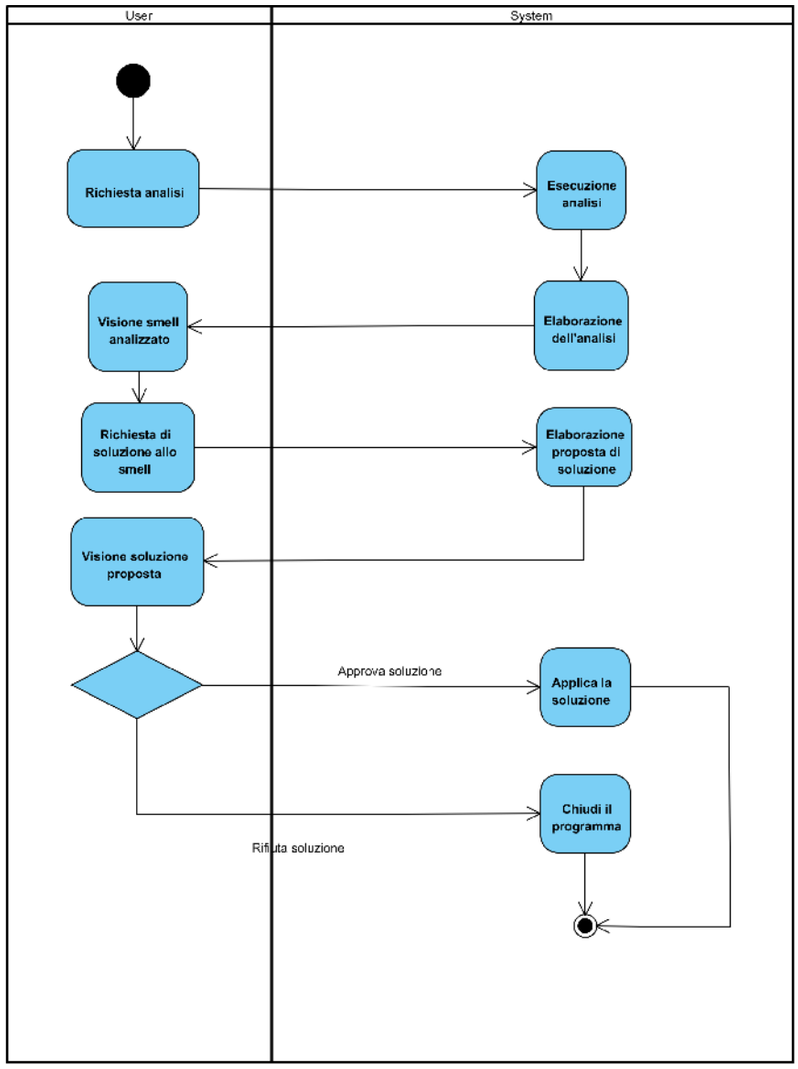


Figura 5: AD1-Analisi e Correzione

3 Proposed system

- I seguenti requisiti funzionali, appartenenti al Current Enviroment, non sono stati modificati:

  RF 1:
  RF 1.1
  RF 2:
  RF 2.1
  RF 2.2
  RF 3:
  RF 3.1
  RF 3.2
  RF 4:
  RF 4.1
  RF 4.2
  RF 5
  RF 6

- **RF 2 - Correzione**

  Il sistema consente la creazione di una possibile soluzione ai quattro principalei tipi di code smell.

  RF 2.3: Il sistema fornisce una possibile soluzione al code smell di tipo Blob.
  RF 2.3.1: Il sistema permette di modificare la possibile soluzione al Blob attraverso l'operazione che consente lo spostamento dei metodi applicata alle due classi proposte nell'operazione di "Extract class".
  RF 2.4: Il sistema fornisce una possibile soluzione al code smell di tipo Promiscuous Package.
  RF 2.4.1: Il sistema permette di modificare la possibile soluzione al Promiscuous Package attraverso l'operazione di spostamento delle classi applicata ai package proposti nell'operazione di "Extract package".
  RF 2.5: Il sistema permette di segnalare come falso positivo una componente indicata come difettosa dal correttore automatico.

- **RF 3 - Refactoring:**

  Il sistema consente la creazione di una possibile soluzione ai quattro principali tipi di code smell.

  RF 3.3: Il sistema permette allo sviluppatore di risolvere il code smell di tipo Blob.
  RF 3.3.1: Il sistema permette di risolvere il code smell di tipo Blob attraverso l'uso dell'operazione di "Extract class".

  RF 3.4: Il sistema permette allo sviluppatore di risolvere il code smell di tipo Promiscuous Package.
  RF 3.4.1: Il sistema permette di risolvere il code smell di tipo Promiscuous Package attraverso l'uso dell'operazione di "Extract package".

- **RF 7 - To Do:**

  RF 7.1: Il sistema consente, all'avvio del programma, di mostrare all'utente un promemoria per un'eventuale correzione manuale.
  RF 7.2: Il sistema consente di aggiungere un nuovo promemoria.

- **RF 8 - Statistiche di accoppiamento:**
  Il sistema consente di visualizzare le statistiche riguardanti classi e package precedentemente analizzati.

- **RF 9 - Numero modifiche e Revisioni:**
  Il sistema permette di tener traccia del numero di volte che è stato modificato un componente.

- **RF 10 - Analisi rischi:**
  Il sistema consente di visualizzare una stima di pericolosità che potrebbe occorrere nel caso in cui si decide di fare un Refactor.

3.1 Requisiti Non Funzionali

- **RNF 1 - Performance**
  Il sistema deve garantire brevi tempi di risposta, in particolare nelle operazioni di analisi del codice.

- **RNF 2 - Robustezza**
  Il sistema è in grado di funzionare correttamente anche in situazioni anomale o in caso di uso scorretto, notificando l'utente della situazione erronea rilevata, ma senza terminare la propria esecuzione.

- **RNF 3 - Sicurezza**
  Il sistema si presenta sufficientemente sicuro in quanto il suo ambiente d'azione è locale, quindi non vi è la necessità di protezione da minacce esterne.

- **RNF 4 - Usabilità**
  Il sistema risulta essere di facile utilizzo. L'interfaccia grafica risulta essere intuitiva, agevolando il lavoro dello sviluppatore.

- **RNF 5 - Manutenibilità**
  Il sistema presenta un elevato grado di manutenibilità, in particolare, favorisce l'aggiunta di nuove funzionalità.

- **RNF 6 - Implementazione**
  Il sistema è realizzato interamente in linguaggio Java, sia parte back-end che front-end.

- **RNF 7 - Affidabilità**
  Il sistema assicura un'alta affidabilità, riducendo al minimo i casi di arresto anomalo del sistema.

3.2 Use Case Model

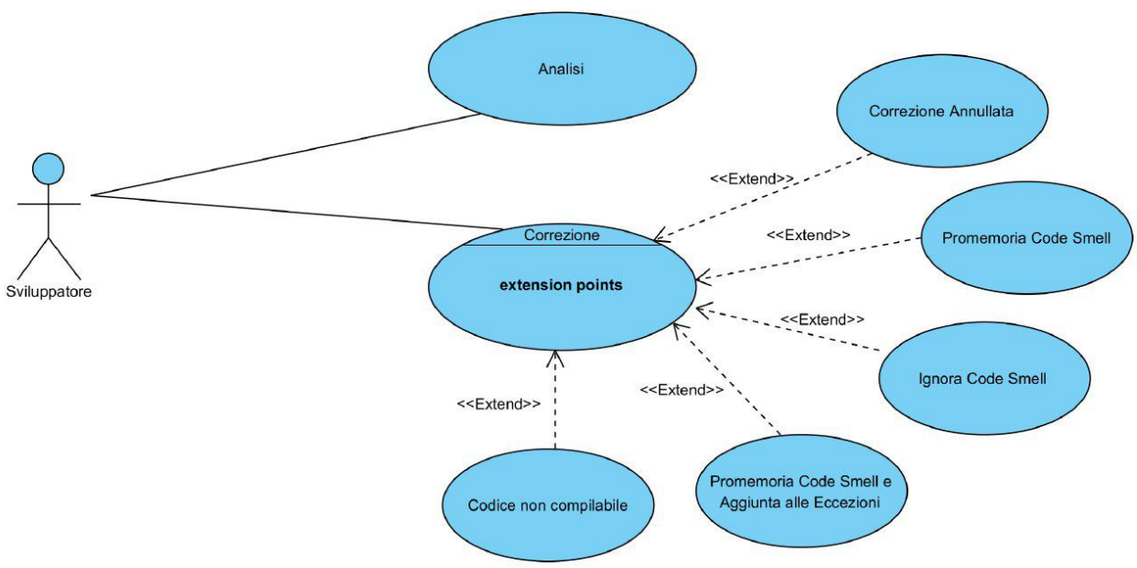


Figura 6: UCD1-Analisi e Correzione

Use case name	**UC 1- Analisi e ricerca smell**
Participating actors	**Sviluppatore**
Entry condition	L'utente avvia con successo Intellij.
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente nella sezione plugin di intelliJ, sceglie di effettuare una ricerca di "code smells" indicando il luogo e il tipo di "code smell", nel codice presente nel progetto aperto attualmente su intelliJ. La mancanza di parametri specificati, implica una ricerca globale su tutto il progetto. **2.** Il sistema, in base ai parametri specificati dall'utente, effettua una ricerca sul codice. Dopo una breve elaborazione, viene mostrata una finestra di riepilogo all'utente. In tale schermata sono presenti metriche, topic e una tabella che permette di filtrare i risultati. **3.** L'utente prende visione del risultato dell'analisi tramite una tabella e varie RadarMap riportanti i 5 termini più frequenti. La tabella riporta per ogni smell identificato: una parte dedicata per effettuare refactoring o ignorare smell, la locazione, le metriche di qualità calcolate(LOC, WNC, RFC, CBO, LCOM, NOA, NOM, NOPA, NOP) e il tipo di smell. Si può quindi selezionare l'elemento che si vuole esaminare e prendere visione del codice del progetto.
Exit condition	L'utente visualizza il codice affetto da smell
Exception condition	Nel passo 1, in caso di codice java non corretto dal punto di vista sintattico, semantico o lessicale si verifica lo UC 1.1.
Quality requirements	• Tempo di elaborazione massimo: 3 minuti. • Il sistema è in grado di rilevare se il codice è java.

Use case name	**UC 1.1 - Analisi codice non compilabile**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente avvia l'analisi del codice.	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente attende l'elaborazioni richieste dal sistema.	
		2. Il sistema tenta di analizzare il codice ma, trovando testo non compilabile, annulla l'operazione e mostra un messaggio di errore.
	3. L'utente prende visione del messaggio che indica il fallimento dell'operazione.	
Exit condition	Il sistema termina l'esecuzione senza effettuare alcuna modifica	
Exception condition		
Priority	Alta	
Quality requirements	• Tempo di elaborazione massimo: 3 minuti.	

Use case name	**UC 2.1 - Correzione Code smell - Feature Envy**
Participating actors	**Sviluppatore**
Entry condition	L'utente ha visualizzato il codice affetto da smell
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri su cui intervenire per lo smells di tipo "Feature Envy" e avvia gli automatismi per la correzione automatica premendo il tasto "Refactor". **2.** Il sistema, mostra una schermata di riepilogo con delle sezioni mostranti una proposta di correzione confrontata con la situazione precedente all'analisi, e un tasto per apportare tali modifiche. **3.** L'utente, può scegliere di accettare tali modifiche consigliate dal sistema cliccando il tasto "Applica modifiche". Se lo sviluppatore non volesse correggere automaticamente il codice può selezionare fra le apposite voci di ignorare lo smell selezionato (Caso eccezione UC-2.5) aggiungere un promemoria (caso eccezione UC-2.6) oppure entrambe (Caso eccezione UC-2.7). L'utente sceglie di applicare la soluzione proposta dal sistema. **4.** Il sistema, applica la soluzione che precedentemente ha mostrato all'utente. **5.** L'utente prende visione di tutte le modifiche effettuate dal sistema.
Exit condition	La finestra del plug-in si chiude.
Exception condition	Se nel passo 3 l'utente non volesse correggere automaticamente il codice, può: • Ignorare il code smell (UC- 2.5) • Aggiungere un promemoria (UC-2.6) • Promemoria code smell e aggiunta eccezioni (UC- 2.7)
Quality requirements	Tempo di elaborazione massimo: 3 minuti.

Use case name	**UC 2.2 - Correzione Code smell - Misplaced Class**
Participating actors	**Sviluppatore**
Entry condition	L'utente ha visualizzato il codice affetto da smell
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri su cui intervenire per lo smells di tipo "Misplaced Class" e avvia gli automatismi per la correzione automatica premendo il tasto "Refactor". **2.** Il sistema, mostra una schermata di riepilogo con delle sezioni mostranti una proposta di correzione confrontata con la situazione precedente all'analisi, e un tasto per apportare tali modifiche. **3.** L'utente, può scegliere di accettare tali modifiche consigliate dal sistema cliccando il tasto "Applica modifiche". Se lo sviluppatore non volesse correggere automaticamente il codice può selezionare fra le apposite voci di ignorare lo smell selezionato (Caso eccezione UC-2.5) aggiungere un promemoria (caso eccezione UC-2.6) oppure entrambe (Caso eccezione UC-2.7). L'utente sceglie di applicare la soluzione proposta dal sistema. **4.** Il sistema, applica la soluzione che precedentemente ha mostrato all'utente. **5.** L'utente prende visione di tutte le modifiche effettuate dal sistema.
Exit condition	La finestra del plug-in si chiude.
Exception condition	Se nel passo 3 l'utente non volesse correggere automaticamente il codice, può: • Ignorare il code smell (UC- 2.5) • Aggiungere un promemoria (UC-2.6) • Promemoria code smell e aggiunta eccezioni (UC- 2.7)
Quality requirements	Tempo di elaborazione massimo: 3 minuti.

Use case name	**UC 2.3 - Correzione Code smell - Blob**
Participating actors	**Sviluppatore**
Entry condition	L'utente ha visualizzato il codice affetto da smell
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 seleziona "Blob" come parametro su cui intervenire. **2.** Il sistema, mostra una nuova schermata di riepilogo con delle sezioni mostranti delle frecce per scegliere come "estrarre" gli smells e dei grafici riguardanti un confronto tra la situazione precedente all'analisi e un tasto per apportare tali modifiche. **3.** L'utente puo' selezionare come e dove estrarre gli smells tramite un interfaccia guidata, al termine di tale selezione, clicca sul tasto "applica" per apportare le modifiche oppure, se lo sviluppatore non volesse correggere il codice può selezionare fra le apposite voci di ignorare lo smell selezionato (Caso eccezione UC-2.5) aggiungere un promemoria (caso eccezione UC-2.6) oppure entrambe (Caso eccezione UC-2.7). L'utente sceglie di applicare la soluzione proposta dal sistema. **4.** Il sistema, applica la soluzione che precedentemente ha mostrato all'utente. **5.** L'utente prende visione di tutte le modifiche effettuate dal sistema.
Exit condition	La finestra del plug-in si chiude.
Exception condition	Se nel passo 3 l'utente non volesse correggere automaticamente il codice, può: • Ignorare il code smell (UC- 2.5) • Aggiungere un promemoria (UC-2.6) • Promemoria code smell e aggiunta eccezioni (UC- 2.7)
Quality requirements	Tempo di elaborazione massimo: 3 minuti.

Use case name	**UC 2.4 - Correzione Code smell - Promiscuous Package**
Participating actors	**Sviluppatore**
Entry condition	L'utente ha visualizzato il codice affetto da smell
Flow of events	**UTENTE** / **SISTEMA** **1.** L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 seleziona "Promiscuous Package" come parametro su cui intervenire. **2.** Il sistema, mostra una nuova schermata di riepilogo con delle sezioni mostranti delle frecce per scegliere come "estrarre" gli smells e dei grafici riguardanti un confronto tra la situazione precedente all'analisi e un tasto per apportare tali modifiche. **3.** L'utente puo' selezionare come e dove estrarre gli smells tramite un interfaccia guidata, al termine di tale selezione, clicca sul tasto "applica" per apportare le modifiche oppure, se lo sviluppatore non volesse correggere il codice può selezionare fra le apposite voci di ignorare lo smell selezionato (Caso eccezione UC-2.5) aggiungere un promemoria (caso eccezione UC-2.6) oppure entrambe (Caso eccezione UC-2.7). L'utente sceglie di applicare la soluzione proposta dal sistema. **4.** Il sistema, applica la soluzione che precedentemente ha mostrato all'utente. **5.** L'utente prende visione di tutte le modifiche effettuate dal sistema.
Exit condition	La finestra del plug-in si chiude.
Exception condition	Se nel passo 3 l'utente non volesse correggere automaticamente il codice, può: • Ignorare il code smell (UC- 2.5) • Aggiungere un promemoria (UC-2.6) • Promemoria code smell e aggiunta eccezioni (UC- 2.7)
Quality requirements	Tempo di elaborazione massimo: 3 minuti.

Use case name	**UC 2.5- Ignora code smell**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente ha deciso di ignorare il code smell	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri ed individua gli smells che vuole aggiungere ad un elenco di risultati da ignorare (che siano essi falsi positivi o rigettati per scelta del programmatore)	
		2. Il sistema dopo l'aggiunta alla "lista da ignorare" non mostrerà più i casi da ignorare tra i risultati di analisi a meno di modifiche al codice manuali, nel qual caso verrano eliminati da quest'elenco nella analisi successiva. Il sistema, applica le modifiche scelte dall'utente.
	3. L'utente prende visione della "lista da ignorare" contente tutti gli smell scelti.	
Exit condition	La finestra del plug-in si chiude	
Exception condition		
Quality requirements	Tempo di elaborazione massimo: 1 minuto.	

Use case name	**UC 2.6- Promemoria code smell**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente ha deciso di avere un promemoria dello smell	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente dopo l'analisi del caso UC 1.0, seleziona i propri parametri e seleziona la voce "aggiungi promemoria"	
		2. Il sistema aggiunge la parte di codice al "To do list" di intelliJ, ovvero aggiunge dei piccoli promemoria che compariranno in una finestra dedicata con la funzione di ricordare allo sviluppatore in quale parte del code deve effettuare delle correzioni.
	3. L'utente prende visione delle modifiche effettuate dal sistema.	
Exit condition	La finestra del plug-in si chiude	
Exception condition		
Quality requirements	Tempo di elaborazione massimo: 1 minuto.	

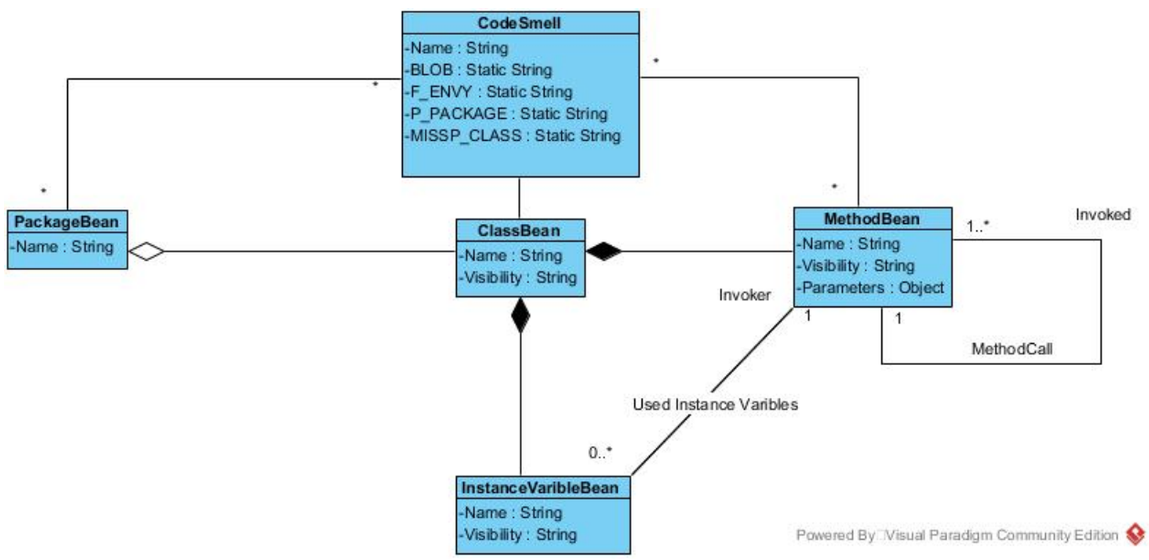
Use case name	**UC 2.7- Promemoria code smell e aggiunta alle eccezioni**	
Participating actors	**Sviluppatore**	
Entry condition	L'utente sceglie il promemoria e aggiunta eccezioni	
Flow of events	**UTENTE**	**SISTEMA**
	1. L'utente, dopo aver preso visione dei risultati dati del caso UC1.0 sceglie i parametri ed individua gli smells che vuole aggiungere ad un elenco di risultati da ignorare (che siano essi falsi positivi o rigettati per scelta del programmatore) e seleziona, inoltre, la voce "aggiungi promemoria"	
		2. Il sistema dopo l'aggiunta alla "lista da ignorare" non mostrerà più i casi da ignorare ma aggiunge le parti di codice al "To do list" di intelliJ
	3. L'utente prende visione delle modifiche effettuate dal sistema.	
Exit condition	La finestra del plug-in si chiude	
Exception condition		
Quality requirements	Tempo di elaborazione massimo: 1 minuto	

ID	**UC 2.8**
Nome caso d'uso	Statistiche classe/package
Attori partecipanti	Sviluppatore
Precondizione	UC 1.0
Flusso di eventi	Utente Dopo l'analisi (UC 1) lo sviluppatore si trova davanti ad una schermata di riepilogo nella quale una volta selezionata una voce tra i risultati proposti, è presente il tasto "Statistiche e Correlazioni" che rimanda ad una nuova finestra. Lo sviluppatore prende visione dei risultati generati dal sistema ed effettua le proprie valutazioni, al termine delle quali chiude semplicemente la finestra e torna alla schermata precedente. Sistema il sistema genera una nuova finestra contenente un elenco di **Associazioni**, **Aggregazioni**, **Composizioni** e **Specializzazioni** che riguardano la classe/package che abbiamo selezionato precedentemente
Condizione d'uscita	L'utente visualizza correttamente le statistiche
Eccezioni	
Priorità	Bassa
Requisiti di qualità	

ID	**UC 2.9**
Nome caso d'uso	Numero di modifiche e revisioni
Attori partecipanti	Sviluppatore
Precondizione	UC 1.0
Flusso di eventi	**Utente** / **Sistema** Dopo l'analisi (UC 1) lo sviluppatore si trova davanti ad una schermata di riepilogo nella quale una volta selezionata una voce tra i risultati proposti, è presente il tasto "Numero di modifiche e revisioni" che rimanda ad una nuova finestra. il sistema genera una nuova finestra contenente il numero di modifiche effettuate alla classe/package in esame. Lo sviluppatore prende visione dei risultati generati dal sistema ed effettua le proprie valutazioni, al termine delle quali chiude semplicemente la finestra e torna alla schermata precedente.
Condizione d'uscita	L'utente visualizza correttamente le statistiche
Eccezioni	
Priorità	Bassa
Requisiti di qualità	

ID	**UC 2.10**	
Nome caso d'uso	Analisi Rischi	
Attori partecipanti	Sviluppatore	
Precondizione	UC 1.0	
Flusso di eventi	Utente Dopo l'analisi (UC 1) lo sviluppatore si trova davanti ad una schermata di riepilogo nella quale una volta selezionata una voce tra i risultati proposti, è presente il tasto "Stima rischi modifica" che rimanda ad una nuova finestra. Lo sviluppatore prende visione dei risultati generati dal sistema ed effettua le proprie valutazioni, al termine delle quali chiude semplicemente la finestra e torna alla schermata precedente.	Sistema il sistema genera una nuova finestra contenente una stima di "pericolosità di modifica" basata sulle correlazioni tra classi e il numero di modifiche già effettuate alla classe/package in esame
Condizione d'uscita	L'utente visualizza correttamente le statistiche	
Eccezioni		
Priorità	Bassa	
Requisiti di qualità		

3.3 Static Analysis Model

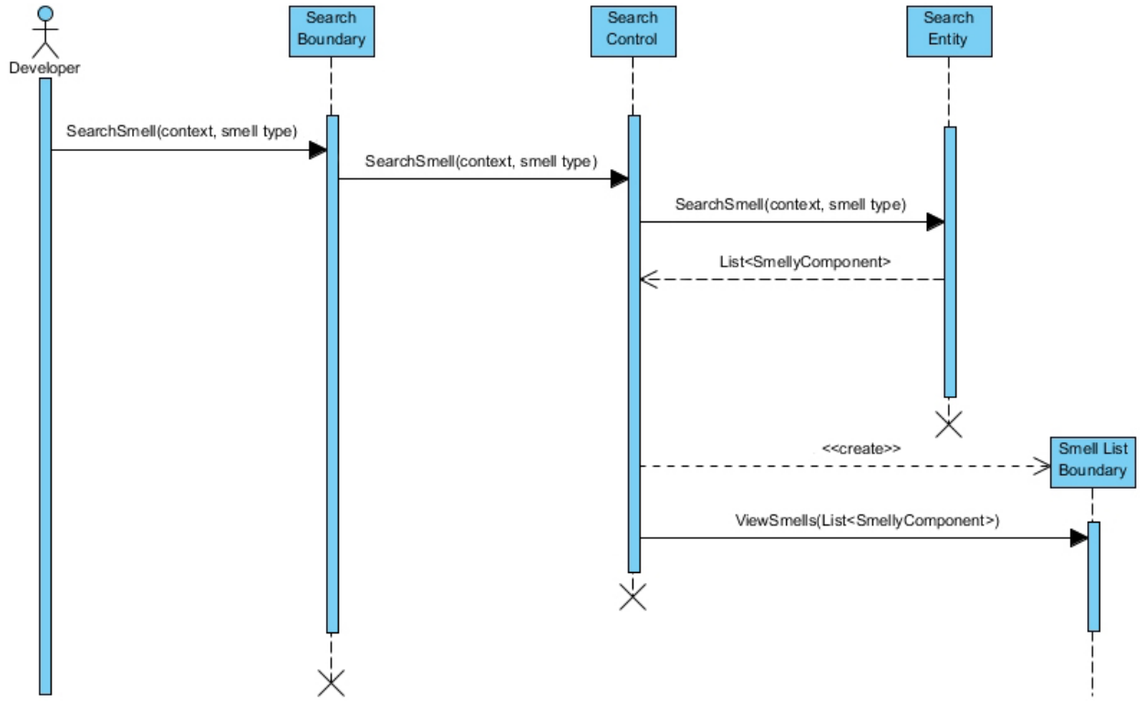


TIPO	NOME	DESCRIZIONE
Boundary	Search	Boundary che si occupa della riproduzione a schermo dei risultati dell'analisi.
	Smell List	Boundary che si occupa della visualizzazione della lista degli smell trovati in fase di analisi.
	Proposed Solution	Boundary che rende possibile visualizzare la lista delle soluzioni elaborate dal sistema.
	Result	Boundary che rende possibile visualizzare i risultati dopo l'esecuzione del Refactor.
	Smell Manager	Boundary che consente la visualizzazione dei Code Smell trovati in fase d'analisi, dal quale è possibile scegliere le operazioni da effettuare su questi ultimi.
	Success	Boundary preposto alla visualizzazione dei messaggi di successo delle avvenute operazioni.
	Statistics	Boundary che si occupa della visualizzazione dei dati relativi alle statistiche calcolate sui Code Smell.
	Statistics Summary	Boundary che si occupa della visualizzazione a schermo delle statistiche aggiornate calcolate sui Code Smell.
	Revision	Boundary che consente di visualizzare i dati relativi al numero di modifiche e revisioni effettuate su determinati Code Smell.
	Revision Summary	Boundary che rende possibile visualizzare le statistiche aggiornate relative a revisioni e modifiche effettuate su determinati Smell.
	Risk Analysis	Boundary che consente di visualizzare i fattori di rischio legati alla modifica di una classe.
	Risk Result	Boundary preposto alla visualizzazione dei dati relativi ai fattori di rischio aggiornati.
Control	Search	Control che invia messaggi al Search Entity per ricercare smell ed istanzia lo Smell List Boundary, al quale invia un messaggio per richiedere la visualizzazione della lista degli smell.
	Proposed Solution	Control che crea il Solution Boundary, al quale invia un messaggio per visualizzare le proposte di correzione.
	Refactoring	Control che crea il Result Boundary, al quale invia un messaggio per notificare il successo dell'operazione di Refactoring.
	Corrector	Control che comunica con lo Smell Mangaer entity, al quale invia un messaggio per procedere con la correzione dello smell, dopodiché istanzia il Success Boundary.
	Ignore	Control che si occupa di comunicare allo Smell Manager Entity un messaggio per far sì che un determinato Code Smell, selezionato precedentemente, venga ignorato.
	Reminder	Control che comunica con lo Smell Manager Entity e invia un messaggio per porre un Code Smell in "to do".
	Statistics	Control che crea lo Statistics Summary Boundary e comunica con lo Statistics Entity, al quale invia un messaggio per ottenere i dati sulle statistiche calcolate sui Code Smell.
	Revision	Control che comunica con il Revision Entity, al quale invia un messaggio per ottenere i dati relativi alle modifiche effettuate, dopodiché istanzia il Revision Summary Boundary.
	Risk Analysis	Control che comunica con il Risk Analysis Entity, al quale invia un messaggio per ottenere i dati sui fattori di rischio calcolati per la modifica di una classe, successivamente istanzia il Risk Result Boundary.
Entity	Search	Entity che conserva i dati persistenti relativi ai Code Smell trovati in fase di analisi.
	Smell Manager	Entity che conserva i dati relativi alle correzioni effettuate sui Code Smell.
	Statistics	Entity che conserva i dati relativi alle statistiche calcolate sui Code Smell.
	Revision	Entity che conserva i dati relativi alle modifiche e revisioni effettuate sui Code Smell.

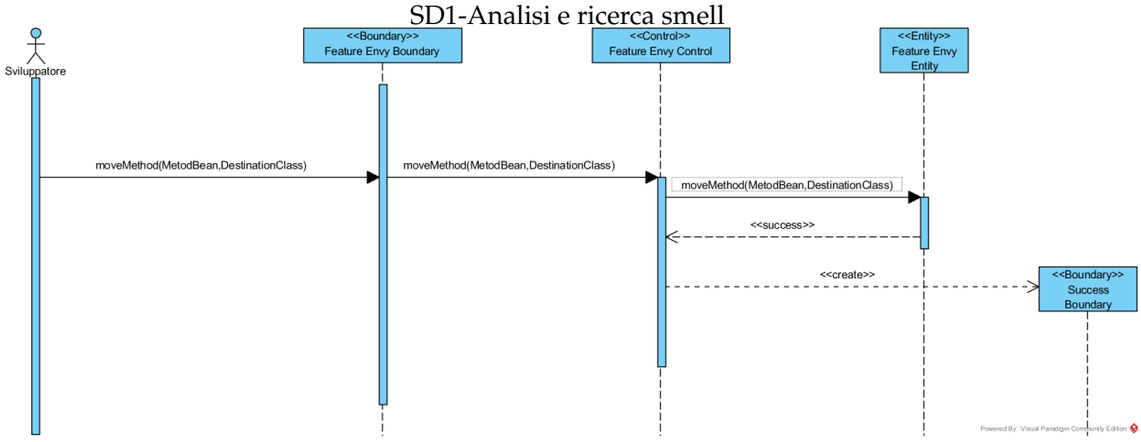
Gli oggetti presenti nella tabella soprastante sono stati individuati a seguito dell'analisi dinamica sotto riportata.

3.4 Dynamic Analysis Model

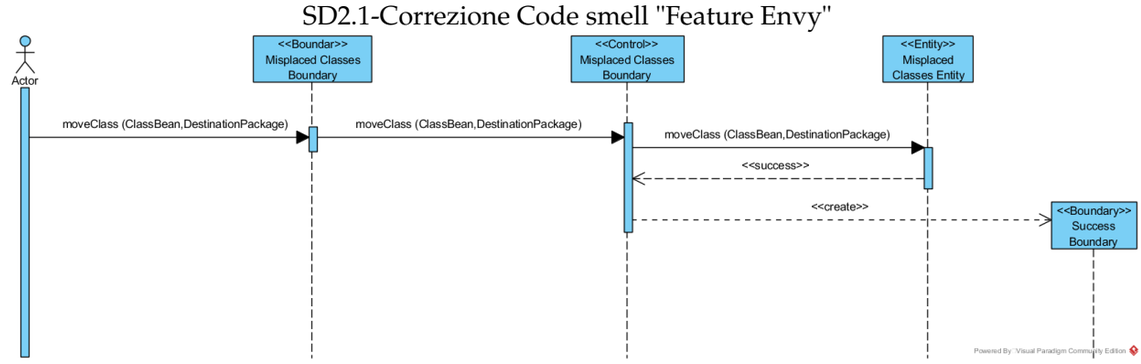
3.4.1 Sequence Diagram



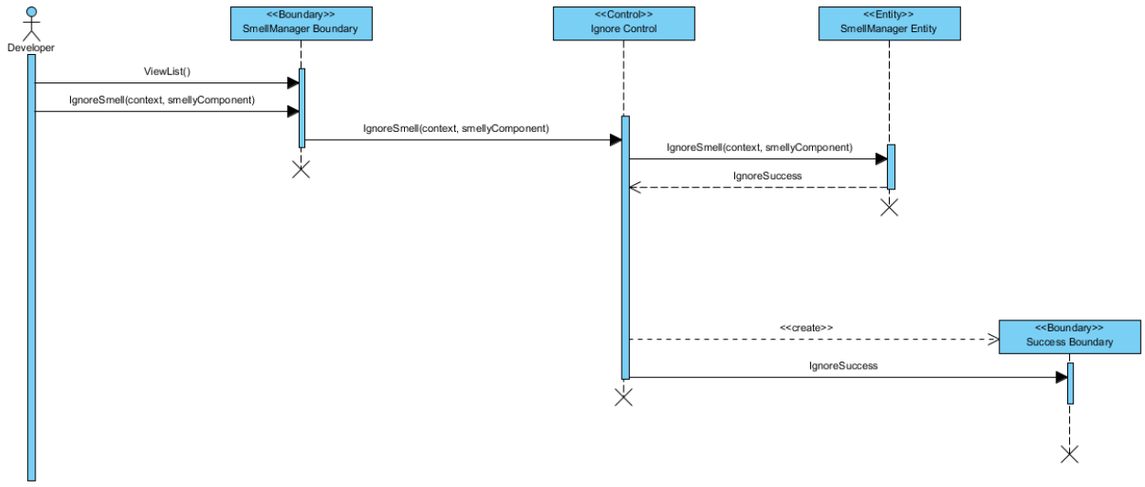
SD1-Analisi e ricerca smell



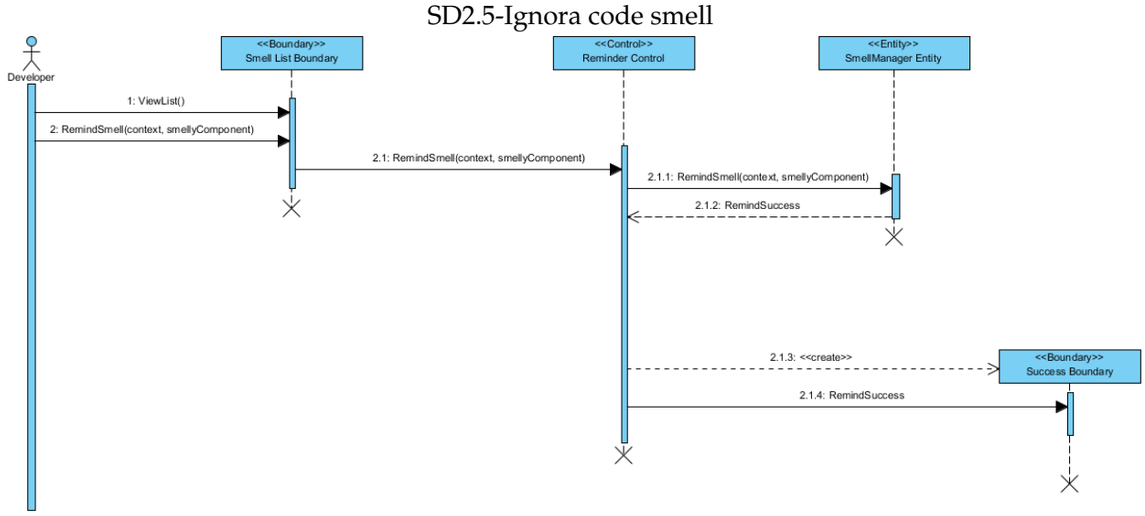
SD2.1-Correzione Code smell "Feature Envy"



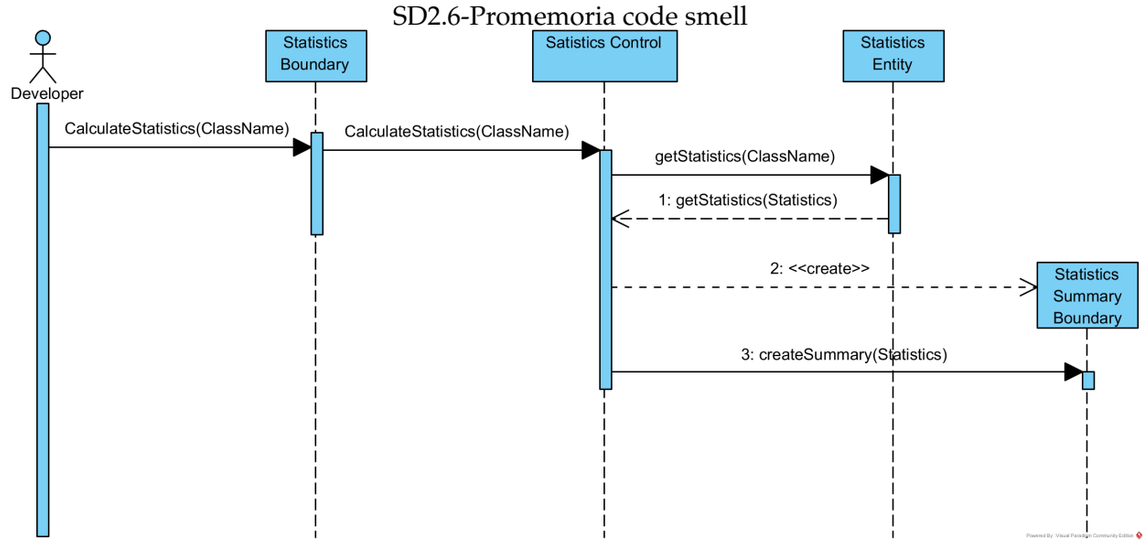
SD2.2-Correzione Code smell "Misplaced Class"



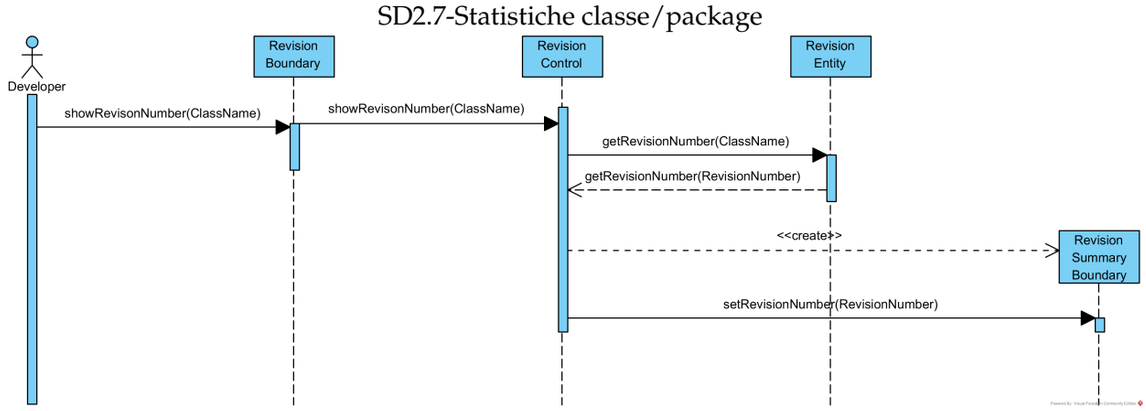
SD2.5-Ignora code smell



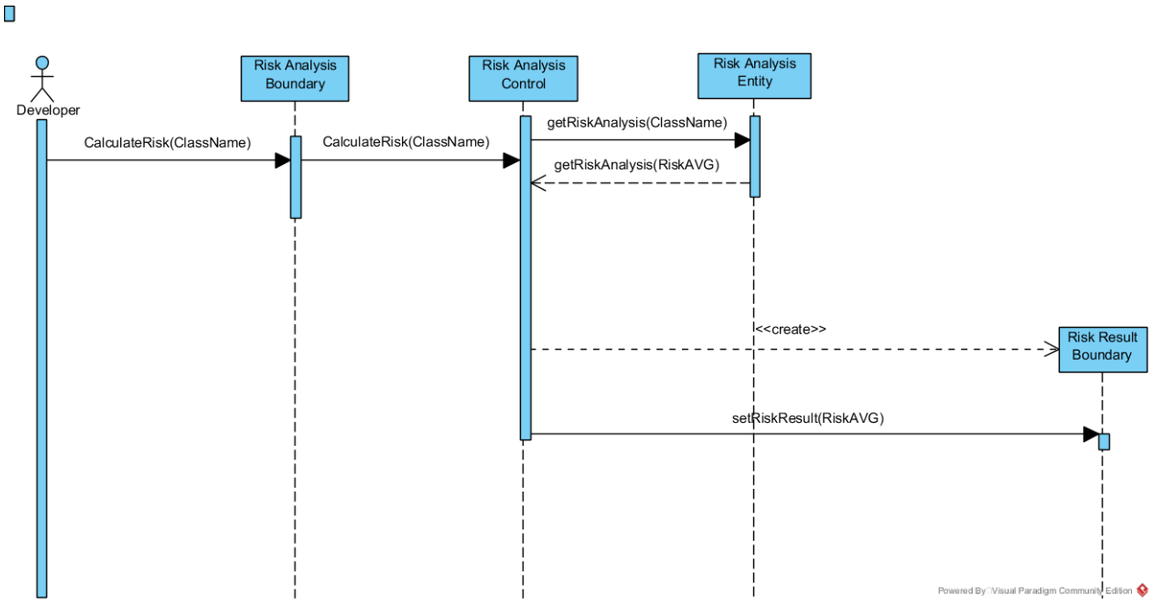
SD2.6-Promemoria code smell



SD2.7-Statistiche classe/package

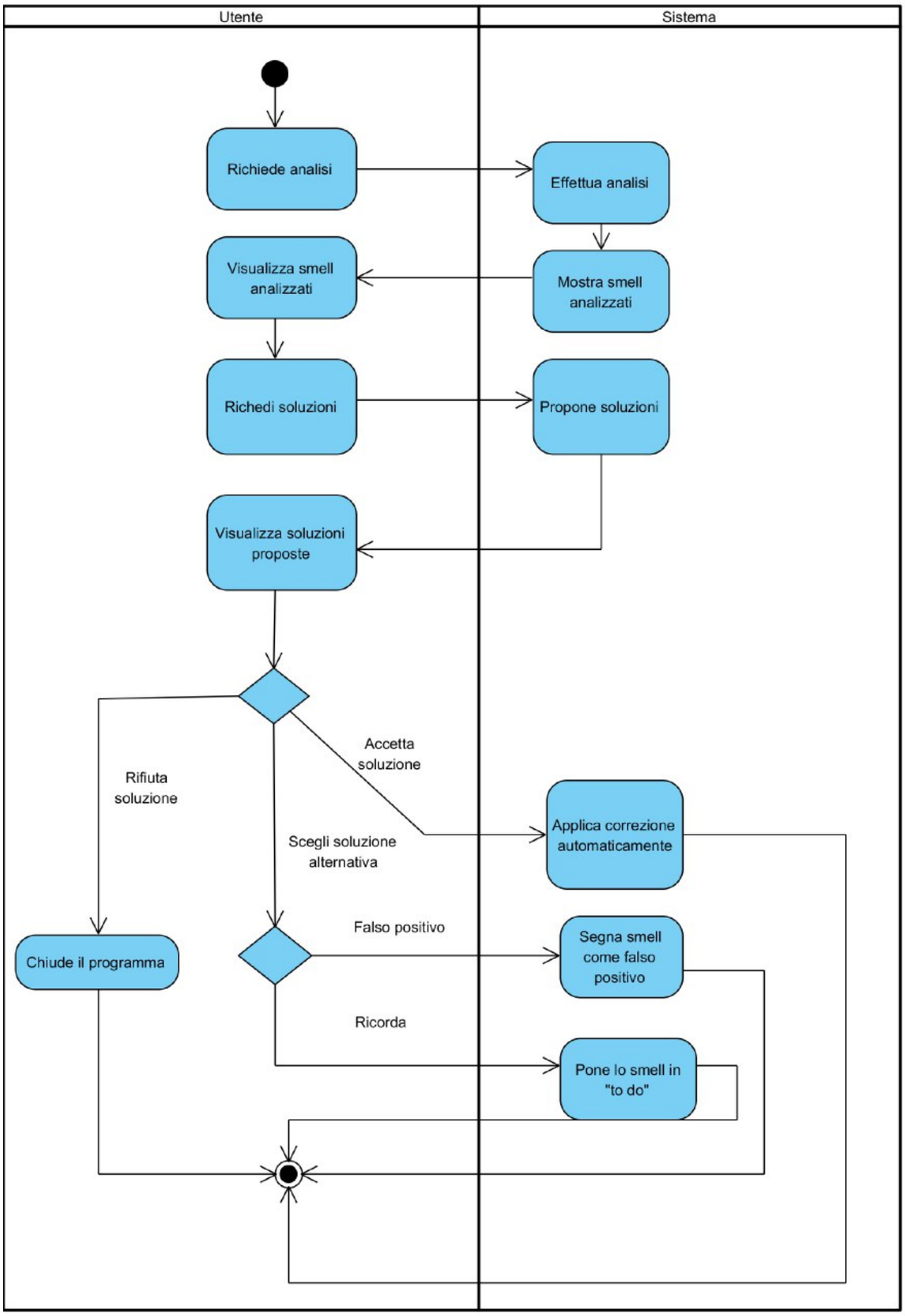


SD2.9-Numero di modifiche e revisioni

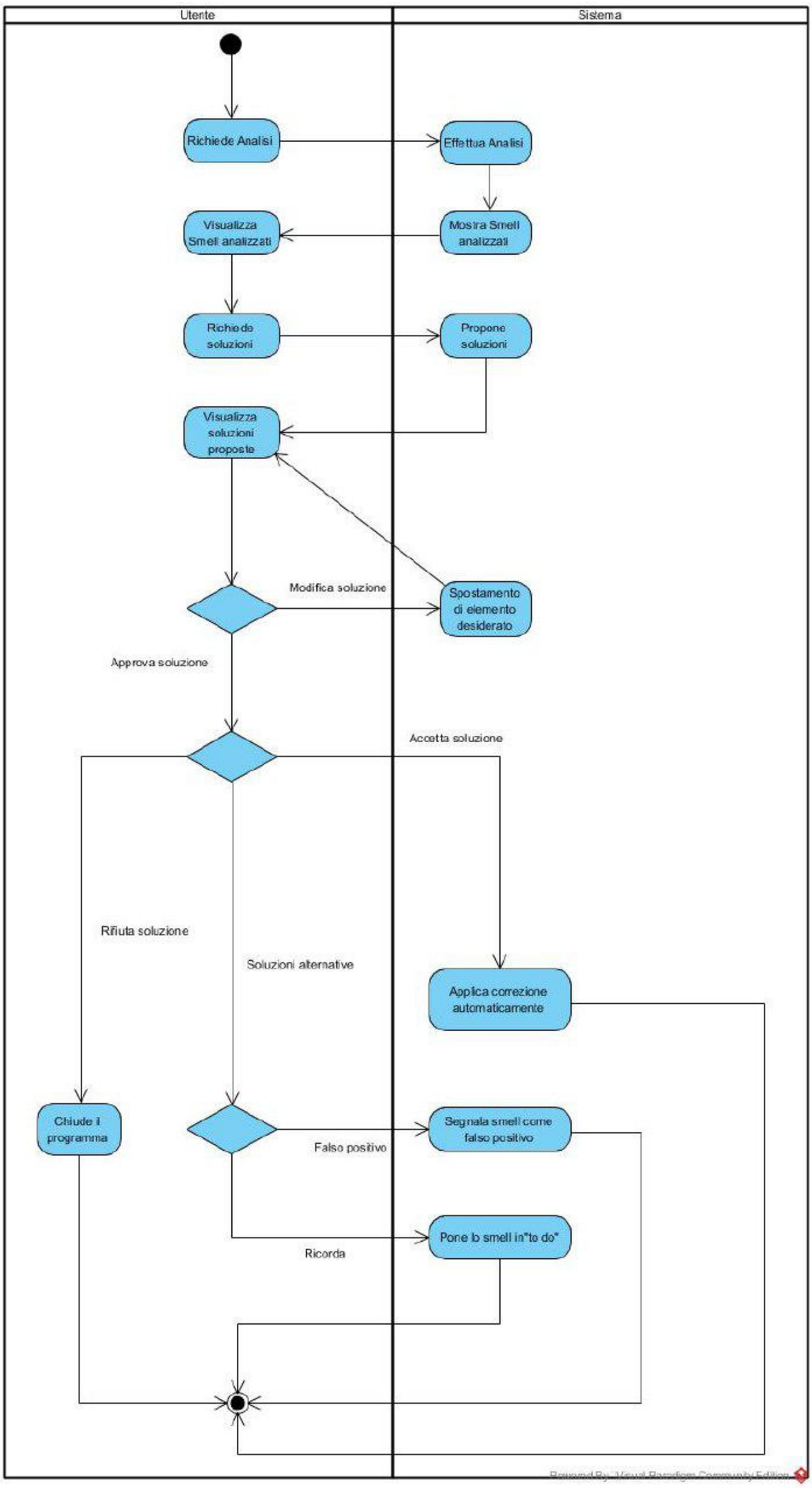


SD2.10-Analisi rischi

3.4.2 Activity Diagram

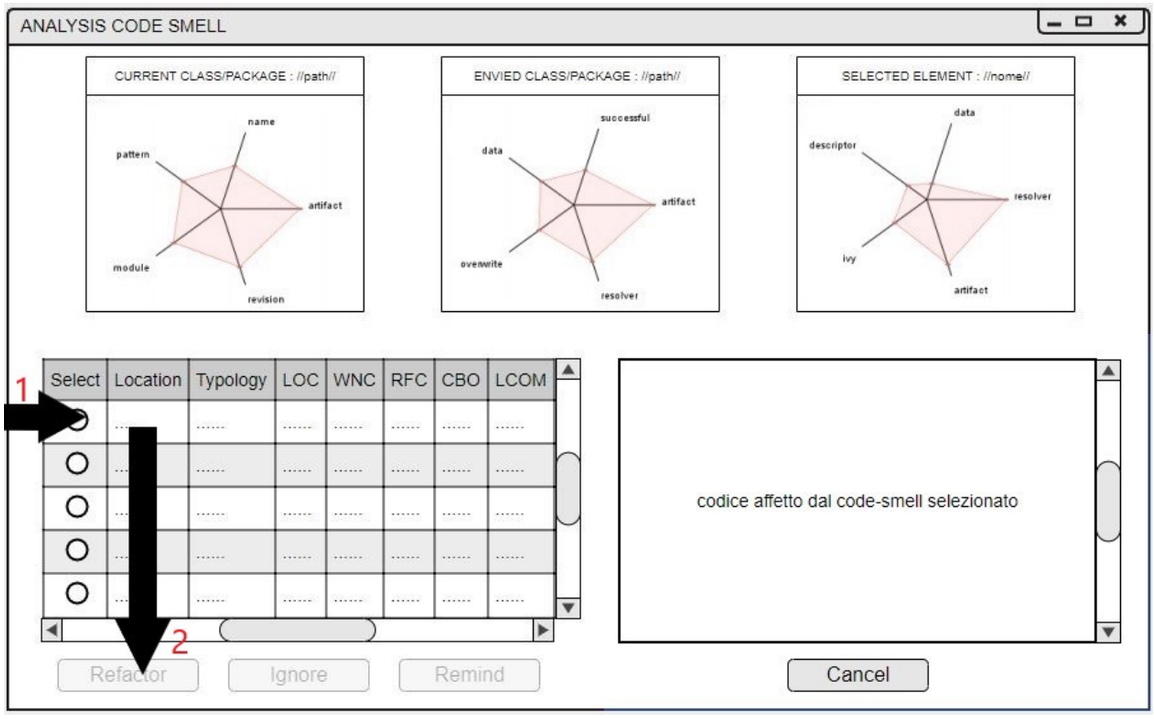


AD1-Analisi, Correzione e Casi Opzionali di "Misplaced Class" e "Feature Envy"


Utente
Sistema
Richiede Analisi
Effettua Analisi
Visualizza Smell analizzati
Mostra Smell analizzati
Richiede soluzioni
Propone soluzioni
Visualizza soluzioni proposte
Modifica soluzione
Spostamento di elemento desiderato
Approva soluzione
Accetta soluzione
Rifiuta soluzione
Soluzioni alternative
Applica correzione automaticamente
Chiude il programma
Segnala smell come falso positivo
Falso positivo
Pone lo smell in"to do"
Ricorda

AD2-Analisi, Correzione e Casi Opzionali di "Blob" e "Promiscuous Package"

3.4.3 New interface mockups

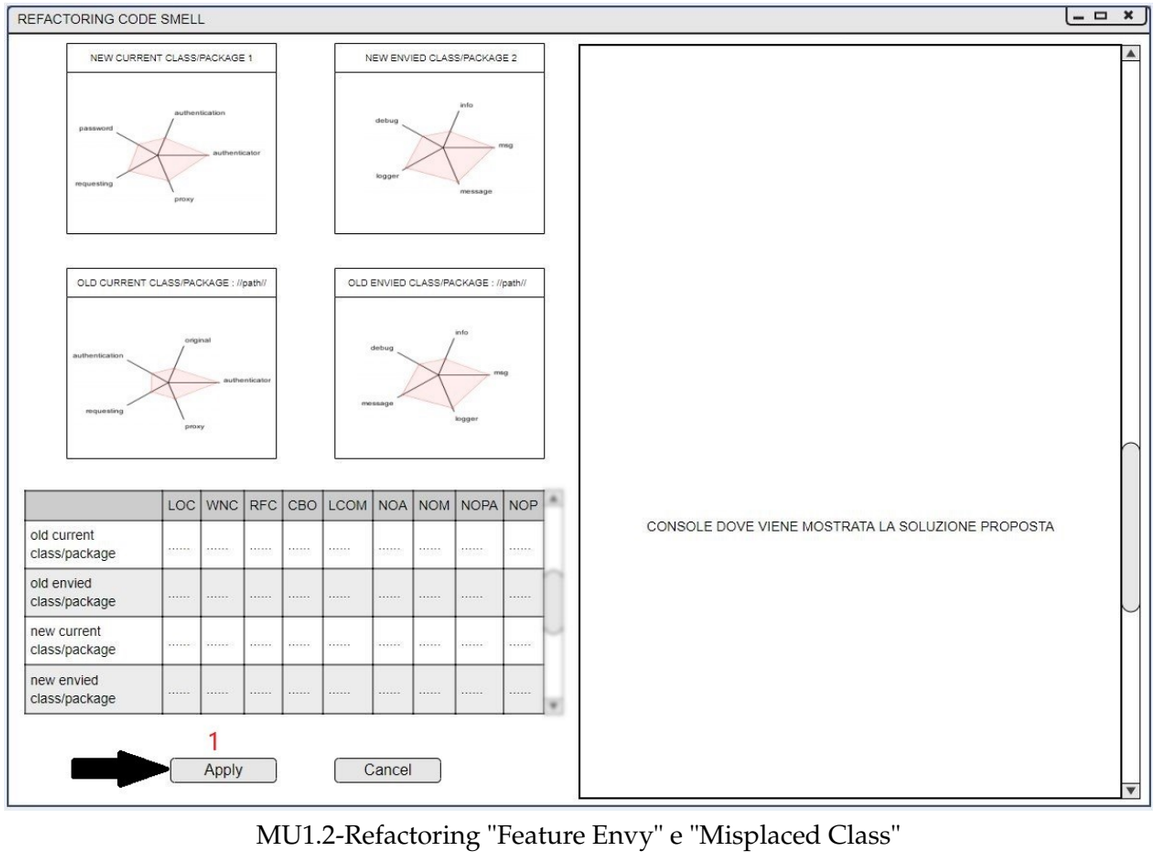


MU1.1-Analisi

L'utente, nel passo 1, seleziona nella tabella lo smell desiderato e visualizza automaticamente le 3 RadarMaps e la porzione di codice collegate. Prima della selezione i pulsanti risultano disabilitati.

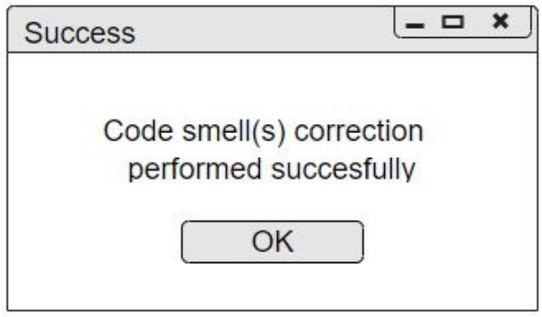
Tale schermata è collegata alla selezione nella tabella di code smell di tipo "Feature Envy" e "Misplaced Class". Nel caso di "Blob" e "Promiscuous Package" i pulsanti "Ignore" e "Remind" rimangono disabilitati e si viene rimandati a MU4.1 e successivi.

L'utente, nel passo 2, selezionando il pulsante "Refactoring" richiede la proposta di soluzione allo smell selezionato.

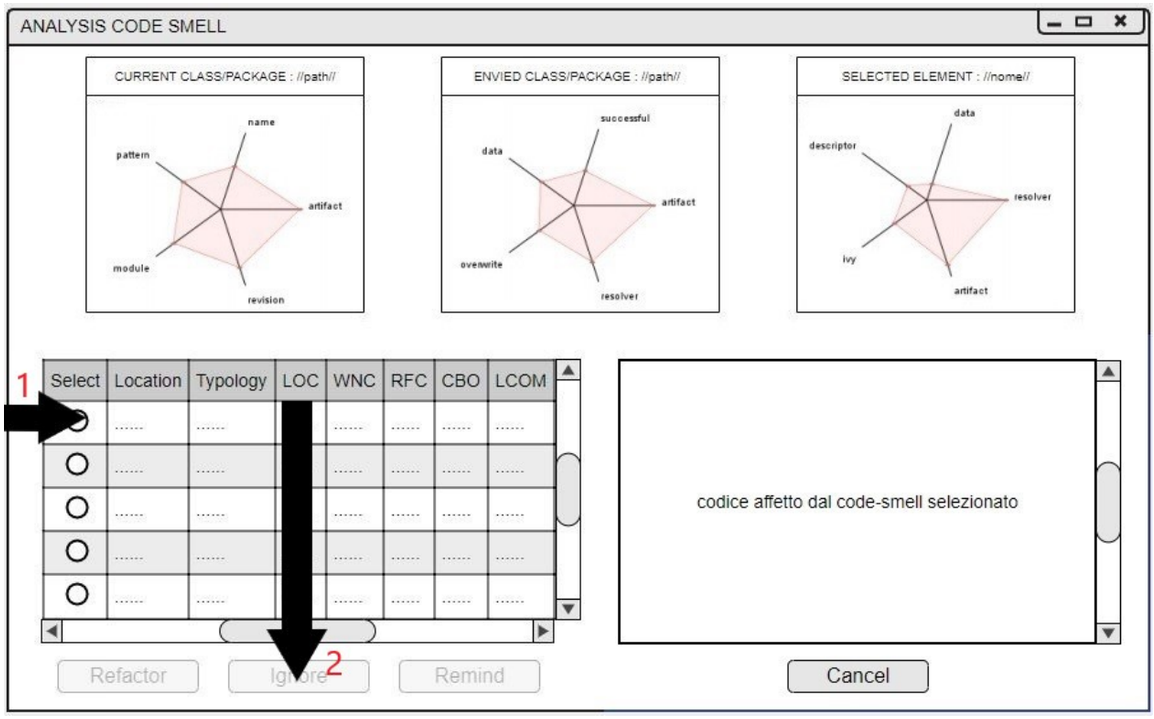


MU1.2-Refactoring "Feature Envy" e "Misplaced Class"

L'utente, nel passo 1, selezionando il pulsante "Apply" richiede l'applicazione della proposta di soluzione.



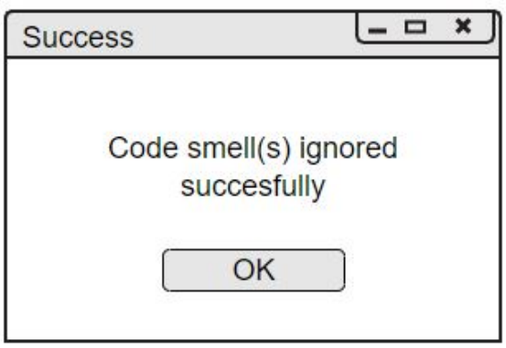
MU1.3-Pop-up di successo



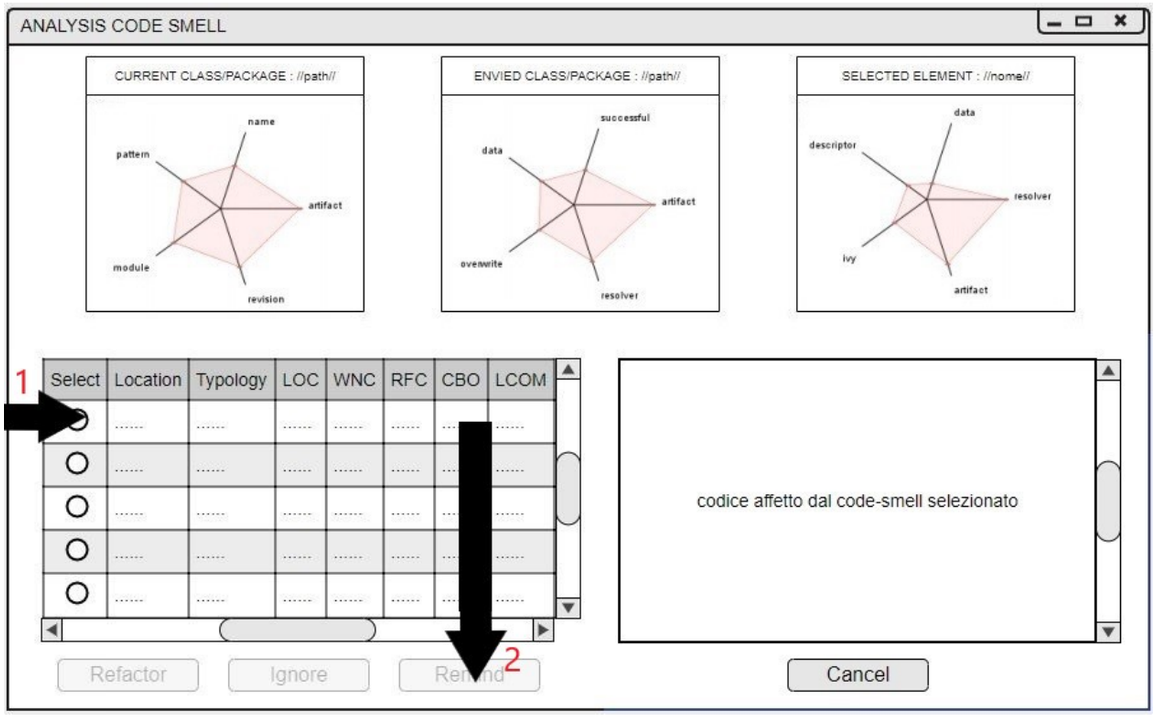
MU2.1-Analisi

L'utente, nel passo 1, seleziona nella tabella lo smell desiderato e visualizza automaticamente le 3 RadarMaps e la porzione di codice collegate.

L'utente, nel passo 2, selezionando il pulsante "Ignore" richiede di ignorare lo smell selezionato.



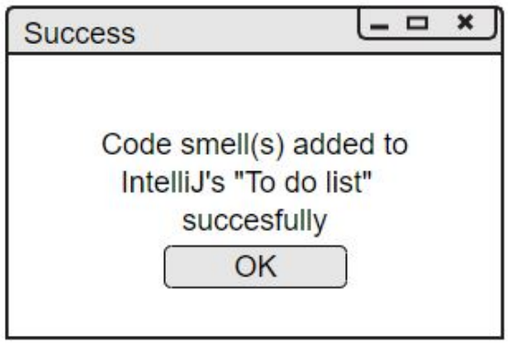
MU2.2-Pop-up ignora



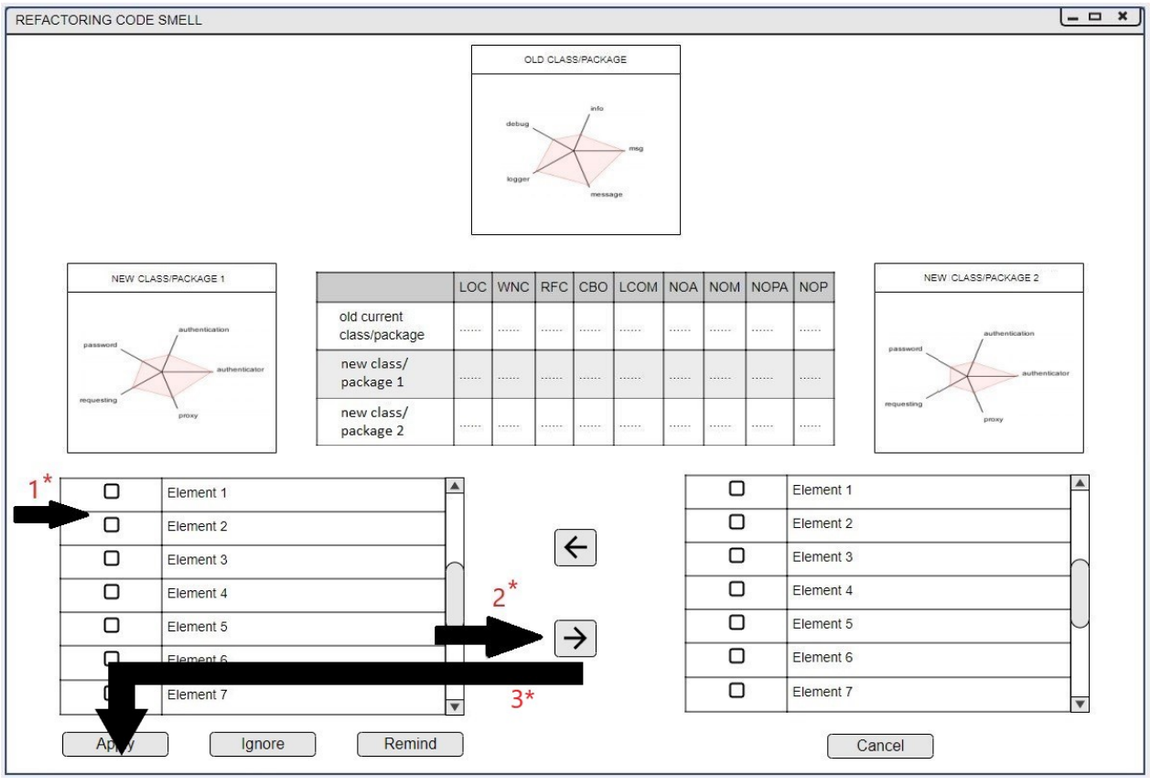
MU3.1-Analisi

L'utente, nel passo 1, seleziona nella tabella lo smell desiderato e visualizza automaticamente le 3 RadarMaps e la porzione di codice collegate.

L'utente, nel passo 2, selezionando il pulsante "Remind" richiede di ricordare lo smell selezionato ed aggiungerlo alla lista "to do".



MU3.2-Pop-up remind

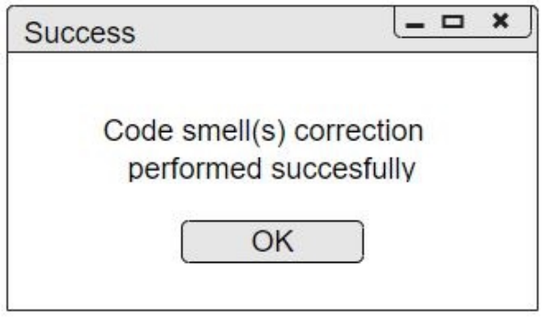


MU4.1-Refactoring "Blob" e "Promiscuous Package"

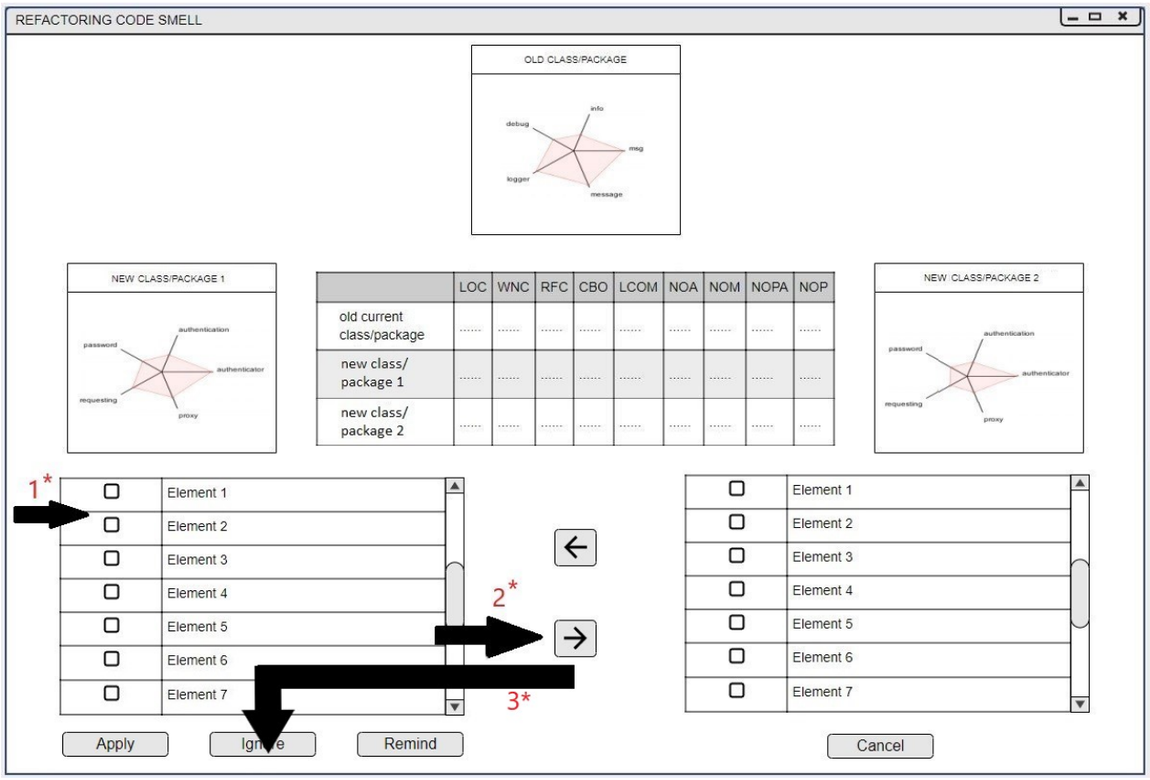
L'utente, nel passo 1, selezionando una o più riga nella tabella e successivamente (tramite passo 2) premendo su apposito pulsante si effettua lo spostamento degli elementi selezionati nell'altra tabella. Tale operazione può essere effettuata anche sulla tabella di destra con la stessa logica.

L'utente, nel passo 3, selezionando il pulsante "Apply" richiede l'applicazione della proposta di soluzione.

*=identifica una operazione opzionale che può quindi essere anche non effettuata.



MU4.2-Pop-up di successo

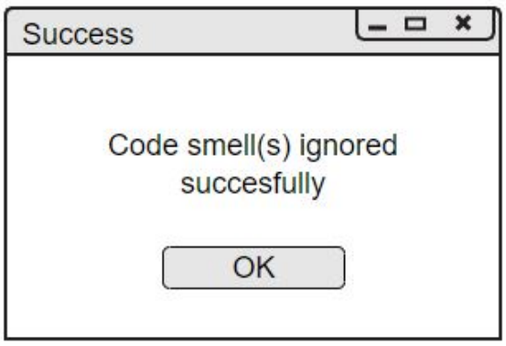


MU5.1-Refactoring seconda schermata

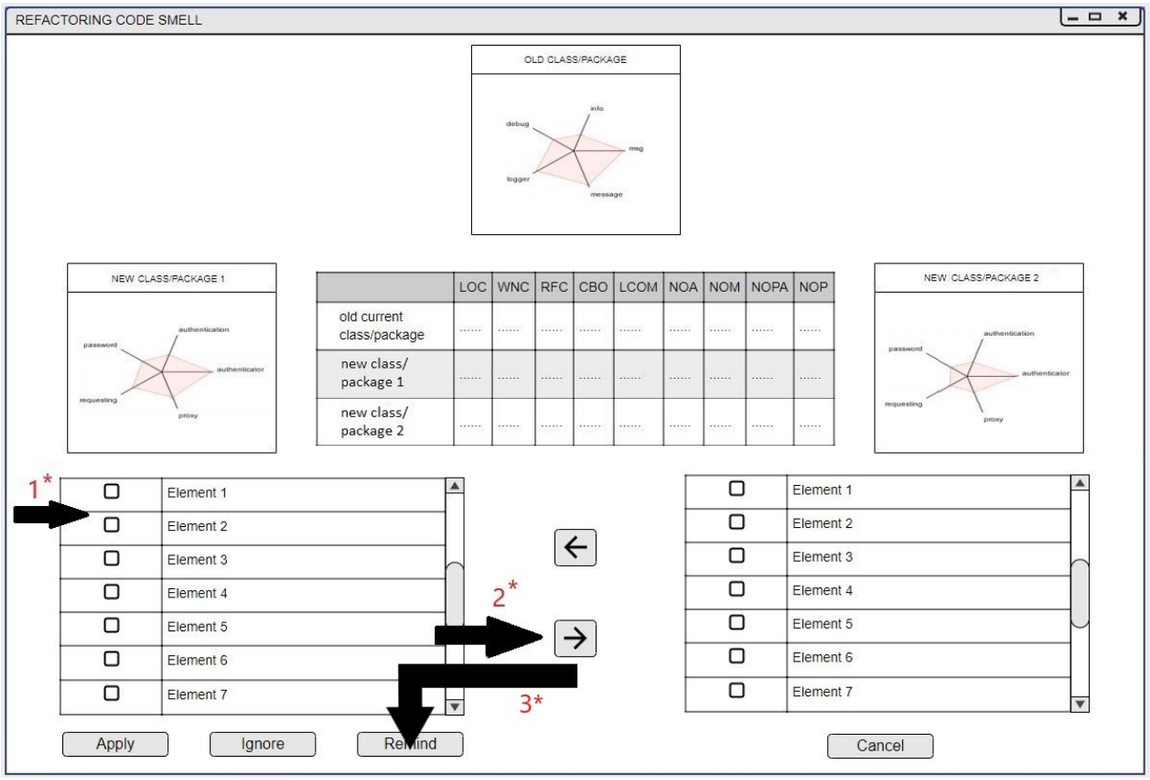
L'utente, nel passo 1, selezionando una o più riga nella tabella e successivamente (tramite passo 2) premendo su apposito pulsante si effettua lo spostamento degli elementi selezionati nell'altra tabella. Tale operazione può essere effettuata anche sulla tabella di destra con la stessa logica.

L'utente, nel passo 3, selezionando il pulsante "Ignore" richiede di ignorare lo smell selezionato.

*=identifica una operazione opzionale che può quindi essere anche non effettuata.



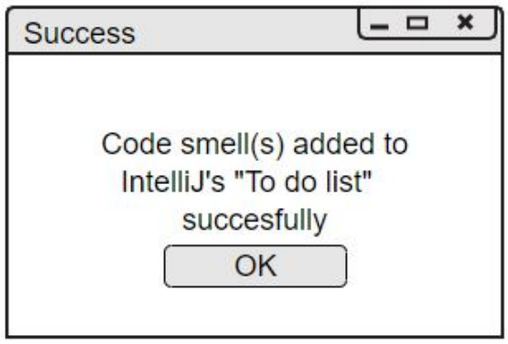
MU5.2-Pop-up ignora



MU6.1-Refactoring seconda schermata

L'utente, nel passo 1, selezionando una o più riga nella tabella e successivamente (tramite passo 2) premendo su apposito pulsante si effettua lo spostamento degli elementi selezionati nell'altra tabella. Tale operazione può essere effettuata anche sulla tabella di destra con la stessa logica. L'utente, nel passo 3, selezionando il pulsante "Remind" richiede di ricordare lo smell selezionato ed aggiungerlo alla lista "to do".

*=identifica una operazione opzionale che può quindi essere anche non effettuata.



MU6.2-Pop-up remind

4 Glossary

- **Blob:** Tipoligia di code smell. E' una Classe che implementa responsabilità molto diverse tra di loro.
- **Feature Envy:** Tipoligia di code smell. E' un Metodo che è maggiormente interessato a variabili e metodi di una classe differente dalla sua.
- **IntelliJ IDEA:** IntelliJ IDEA è un ambiente di sviluppo integrato per il linguaggio di programmazione Java. E' stato sviluppato da JetBrains.
- **Misplaced Class:** Tipoligia di code smell. E' una Classe che non ha alcuna attinenza con le classi dello stesso package.
- **Promiscuous Package:** Tipoligia di code smell. E' un Package contenente classi che hanno responsabilità diverse.
- **TACOR:** Acronimo per Textual Analysis for Code smell detectiOn and Refactoring. E' un plug-in, sviluppato per l'IDE IntelliJ IDEA, creato per risolvere il problema dei "code smell" mediante l'analisi testuale. Esso è il progetto su cui è basato ASCETIC.