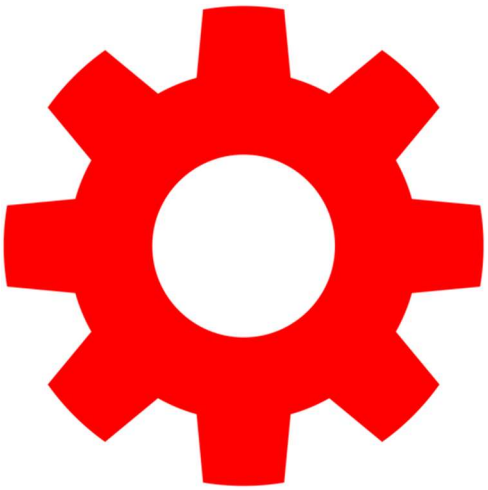
Università degli Studi di Salerno

Corso di Ingegneria del Software

ePADs

Requirements Analysis Document - RAD
Versione 3.0



INDICE

1. INTRODUCTION

1.1. Purpose of the system

Si vuole realizzare un tool, denominato ePADs, che sia capace di identificare Design Pattern, che consiste nel riconoscimento di istanze di Design Pattern all'interno di sistemi software o applicazioni esistenti.

ePADs combina l'analisi statica, basata su visual language parsing, con l'analisi dinamica, basata sulla strumentazione del codice sorgente.

1.2. Scope of the system

Le funzionalità del tool che si vuole realizzare sono, principalmente, le seguenti:

- Effettuare delle operazioni che consentono di indentificare istanze di design pattern.
- Visualizzazione delle istanze di design pattern trovate nella console di output di eclipse.
- Salvataggio delle istanze di design pattern trovate in file di testo.

2. CURRENT SYSTEM

Il tool che si vuole realizzare è ancora in fase di Analysis. Il tool non sostituisce nessun altro tool esistente. Inoltre, essendo ancora in fase di Analysis non è stata sviluppata nessuna funzionalità del sistema.

3. PROPOSED SYSTEM

3.1. Overview

Requirement Analysis Document (RAD) dedicato ai:

- Requisiti Funzionali
- Requisiti Non Funzionali
- Scenari
- Use Case Model
- Sequence Diagrams

3.2. Functional requirements

RF1: Il tool dovrà permettere al **developer** di poter salvare tutte le istanze riscontrate di design pattern strutturali in file di testo.

RF2: Il tool dovrà permettere al **developer** di poter salvare tutte le istanze riscontrate di design pattern comportamentali in file di testo.

RF3: Il **developer** dovrà poter inserire la path del Sistema da controllare.

RF4: Il **developer** dovrà poter inserire dei comandi che permettono di eseguire le fasi che svolgono l'estrazione dei design pattern.

3.3. Nonfunctional requirements

3.3.1. Performance

- Il tool dovrà rispondere velocemente ; La latenza massima di attesa, per una risposta, non dovrà superare i 60 secondi.

3.3.2. Supportability

Il tool dovrà essere suddiviso in vari package per permettere una facile modifica e

aggiornabilità in futuro.

3.3.3. Gestione

Il tool sarà gestito dal developer stesso.

3.3.4. Implementation

il developer inserirà da linea di comando la path, la jarPath dove si trova il Sistema da controllare e una serie di comandi che serviranno per effettuare tutte le funzioni del tool.

3.4. System models

3.4.1. Scenarios

Scenario developer inserisce le checkoptions e la path

Attori: developer

Il **developer**, apre eclipse.

Il **developer** prima di lanciare il tool, inserisce i parametri come la path che identifica il percorso dove si trova l'applicazione da controllare e la relativa jarPath, e una serie di comandi (es –A) che identificano le checkoptions.

Scenario Class Diagram

Attori: developer

Se nei parametri il **developer** ha inserito il comando -PHASEBROWSING, allora si

svolge la fase di browsing.

Nella fase di browsing i file del codice sorgente vengono analizzati,e vengono così recuperate informazioni quali nomi delle classi, tipo delle classi, nomi dei metodi e delle variabili, tipo dei metodi e delle variabili, parametri dei metodi, associazione tra le classi. Vengono poi salvate in un file di testo.

Scenario Model Analysis

Attori: developer

Se nei parametri il **developer** ha inserito il comando che svolge la fase di model Analysis. Il primo passo che viene compiuto è quello di estrarre le informazioni strutturali dal codice OO in input per il recovery delle istanze di Design Pattern, come le relazioni tra le classi, le dichiarazioni dei metodi e le invocazioni dei metodi. Tali informazioni vengono poi memorizzate in un file di testo.

Scenario Source Code Analysis

Attori: developer

Se nei parametri il **developer** ha inserito il comando che svolge la fase di validating. Dal file di testo prodotto nella fase precedente, vengono creati due nuovi file di testo, uno per le istanze di design pattern strutturali e uno per le istanze di design pattern comportamentali.

Scenario EvoSuite Test Case

Attori: developer

Se nei parametri il **developer** ha inserito il comando che svolge l'Evosuite test allora questa funzione verrà eseguita.

Con questa funzione si generano automaticamente test suite che raggiungono un'alta

copertura del codice.

Scenario Profiling

Attori: developer

Se nei parametri il **developer** ha inserito il comando che svolge il profiling allora questa funzione verrà eseguita.

Grazie al plug-in probekit, l'approccio proposto in ePADs utilizza dei probe che permettono di monitorare solo le istanze candidate dell'analisi statica. Per tracciare l'invocazione dei metodi per ciascun candidato si generano dei file probe in grado di iniettare codice Java all'ingresso e all'uscita dei metodi.

Scenario Sequence Analysis

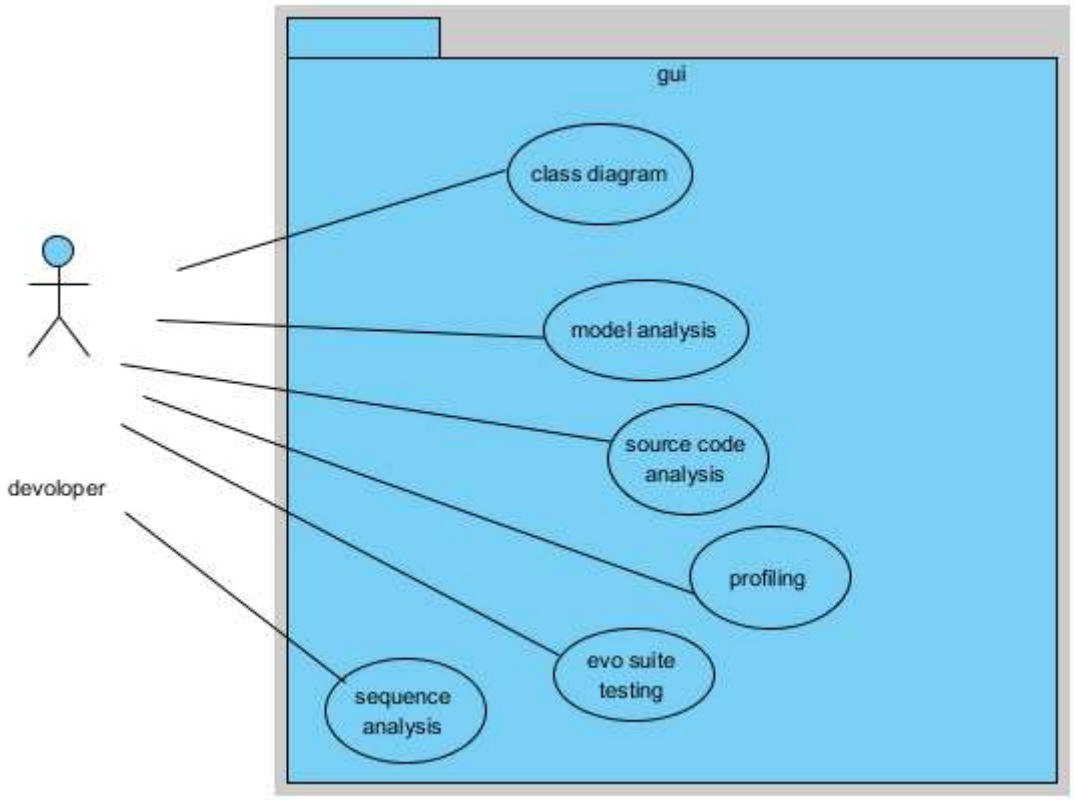
Attori: developer

Se nei parametri il **developer** ha inserito il comando che svolge il sequence analysis allora questa funzione verrà eseguita.

In particolare, la sequenza di metodi viene poi confrontata con il comportamento dei Design Pattern specificato nel sequence diagram presente nella Design Pattern Library. Per eseguire questo match viene utilizzato un compilatore che prende in input una rappresentazione XML del sequence diagram che descrive il comportamento del pattern e costruisce una grammatica di monitoraggio con azioni semantiche associate. Tale grammatica codifica le sequenze di scambi definiti dal sequence diagram. In questo modo si costruisce un validatore del comportamento dei pattern in grado di tracciare le chiamate a metodi che si verificano durante l'esecuzione.

3.4.2. Use case model

Caso d'uso: class diagram,model analysis, source Code Analysis, evo suite test, profiling, sequence analysis



Caso d'uso: class diagram

Nome del caso d'uso: **class diagram**

- Attori partecipanti: **developer**

- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:
    - Se il developer ha inserito il comando per il class diagram allora viene invocata la funzione classDiagram.
- Exit Condition:
    - Una volta terminata la funzione viene stampato a video il numero delle classi trovate.

Caso d'uso: model analysis

Nome del caso d'uso: **model analysis**

- Attori partecipanti: **developer**
- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:
    - Se il developer ha inserito il comando per il model analysis allora viene invocata la funzione modelAnalysis.

- Exit Condition:
    - Una volta termitata la funzione viene stampato tutto a video.

Caso d'uso: source code analysis

Nome del caso d'uso: **source code analysis**

- Attori partecipanti: **developer**
- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:
    - Se il developer ha inserito il comando per il source code analysis allora viene invocata la funzione sourceCodeAnalysis.
- Exit Condition:
    - Una volta termitata la funzione viene stampato tutto a video.

Caso d’uso: evosuite testing

Nome del caso d’uso: **evosuite testing**

- Attori partecipanti: **developer**
- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:
    - Se il developer ha inserito il comando per lì’evosuite testing allora viene invocata la funzione evoSuiting.
- Exit Condition:
    - Una volta termitata la funzione viene stampato a video il risultato del testing.

Caso d’uso: profiling

Nome del caso d’uso: **profiling**

- Attori partecipanti: **developer**
- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:
    - Se il developer ha inserito il comando per il profiling allora viene invocata la funzione profiling.
- Exit Condition:
    - Una volta termitata la funzione viene stampato tutto a video.

Caso d’uso: sequence analysis

Nome del caso d’uso: **sequence analysis**

- Attori partecipanti: **developer**
- Entry Condition:
    - Il developer si posiziona sulla console di output.
- Flusso Eventi:

- Se il developer ha inserito il comando per il sequence analysis allora viene invocata la funzione sequenceAnalysis.
- Vengono stampati a video tutte le istanze di design pattern.

- Exit Condition:
  - Una volta termitata la funzione il programma termina.

3.4.3. Object model

3.4.3.1 Data dictionary

GuiRecovery: Questa è classe principale che contiene l'implementazione per effettuare il class diagram,model analysis,suorce code analysis,evosuite testing,profiling e sequence analysis.	
Indica la mappa che contiene tutte le preferenze.	**toView**
Istanza della classe Extractions.	**_extractions**
Stampa un messaggio di errore.	**showErrorWarning(msg)**
Fa il log e stampa un messaggio di errore.	**PrintAndLog(msg)**

E’ la funzione che effettua il class Diagram.	**Browsing()**
E’ la funzione che effettua il model analysis.	**Extracting()**
E’ la funzione che effettua il Source CodeAnalysis.	**Validating()**
E’ la funzione che effettua il profiling.	**Profiling()**
E’ la funzione che effettua il sequence Analysis.	**Matching()**
E’ la funzione che effettua evostuite testing.	**Evosuiting()**

ProjectExtencion: Questa è classe che viene istanziata per effettuare il class diagram.	
E’ un arrayList di BaseExtension	**_base**
Questo metodo crea le cartelle dove verranno salvati i file txt	**Dirs(runtime, source, dest**)
Il metodo che svolge il class diagram.	**Run()**

ProjectExtraction: Questa è classe che viene istanziata per effettuare il model analysis. In questa classe vengono cercati tutti i design pattern che si vogliono cercare.	
E' la varibile che contiene il nome del file di testo "descriptionExtract.txt"	**_desc**
E' la varibile che contiene il nome del file di testo "descriptionValidate.txt"	**_dest**
E' la varibile che contiene il nome del file di testo "descriptionS.txt" dei pattern strutturali	**_splitS**
E' la varibile che contiene il nome del file di testo "descriptionB.txt" dei pattern comportamentali	**_splitB**
Il metodo che svolge il model analysis. I pattern che trova vengono inseriti nel file descriptionExtract.txt	**Run()**

ProjectValidation: Questa è classe esegue lo split dei design pattern estratti precedentemente.	
E' la varibile che contiene il nome del file di testo "descriptionExtract.txt"	**_source**
E' la varibile che contiene il nome del file di testo "descriptionValidate.txt"	**_dest**
E' la varibile che contiene il nome del file HTML	**_result**

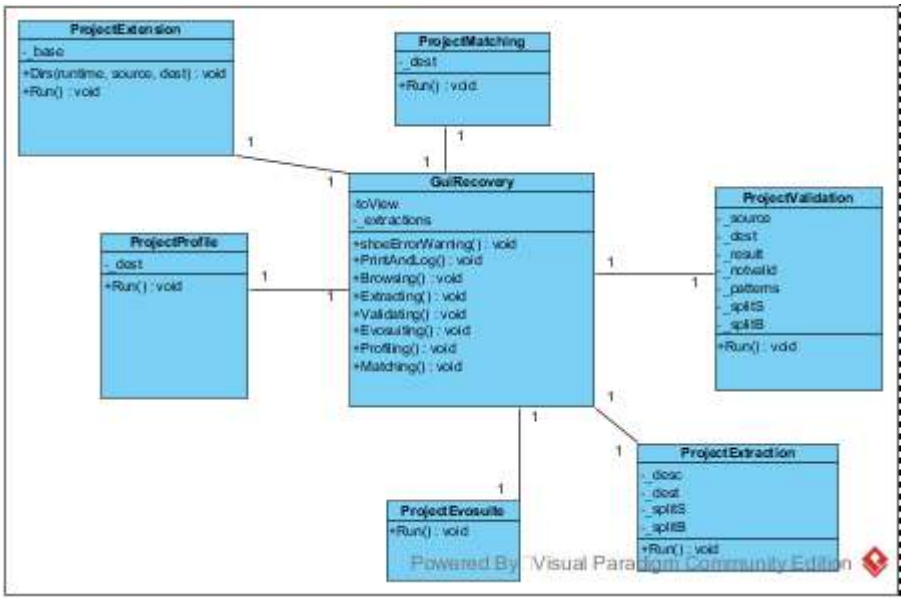
"resut.html" dei pattern estratti.	
E' la varibile che contiene il nome del file HTML dei design pattern non validi.	**_notvalid**
E' la varibile che contiene il nome del file di testo "patterns.txt"che contiene tutti i pattern.	**_patterns**
E' la varibile che contiene il nome del file di testo "descriptionS.txt" dei pattern strutturali	**_splitS**
E' la varibile che contiene il nome del file di testo "descriptionB.txt" dei pattern comportamentali	**_splitB**
Il metodo che svolge il source code analysis. Esegue lo split dei design pattern nei rispettivi file di testo "descriptionS.txt" e "descriptionB.txt"	**Run()**

ProjectEvosuite: Questa è classe che effettua l' Evosuite test sui candidati trovati.	
E' il metodo che svolge l'EvoSuite.	**Run()**

ProjectProfile: Questa è classe che effettua la fase di profiling .	
E' la varibile che contiene il nome del file di testo "descriptionB.txt" dei pattern comportamentali	**_dest**
Questo è il metodo che crea i file probe dei design pattern.	**Run()**

ProjectMatching: Questa è classe che effettua la fase di matching .	
E' la varibile che contiene il nome del file di testo "descriptionBP.txt" dei pattern comportamentali	**_dest**
Questo è il metodo che svolge la funzione di matching e salva i pattern nel file "descriptionBP.txt".	**Run()**

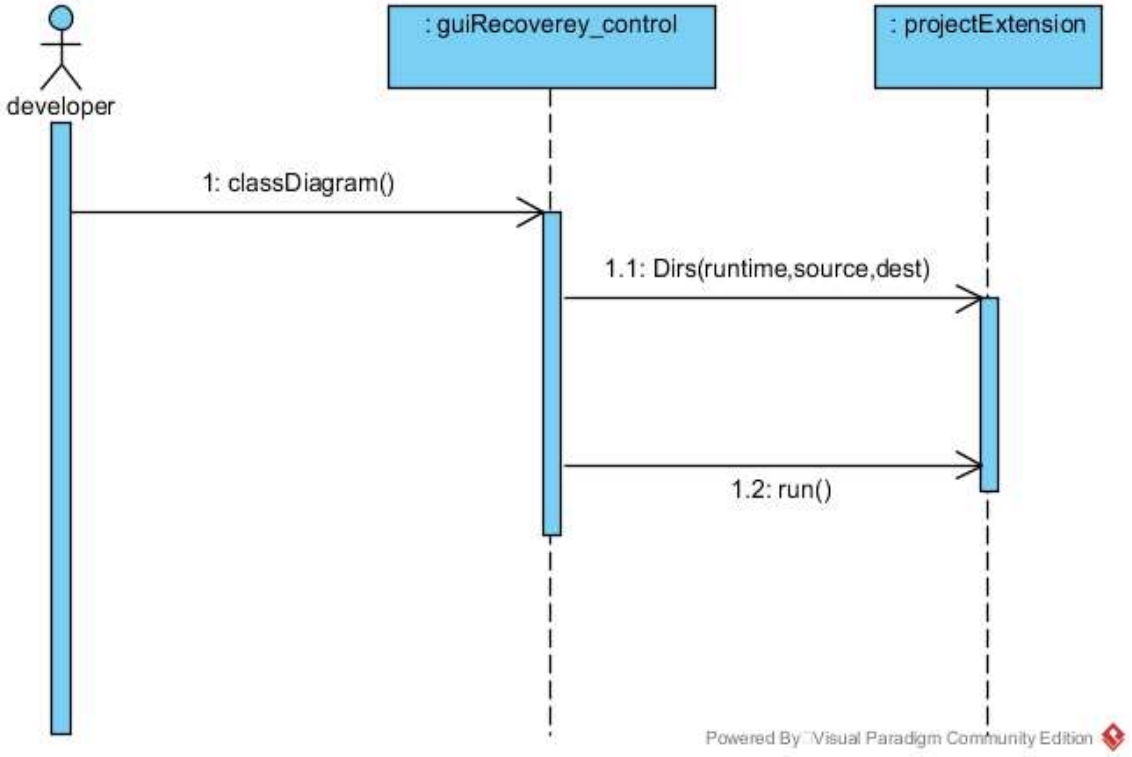
3.4.3.2 Class diagram



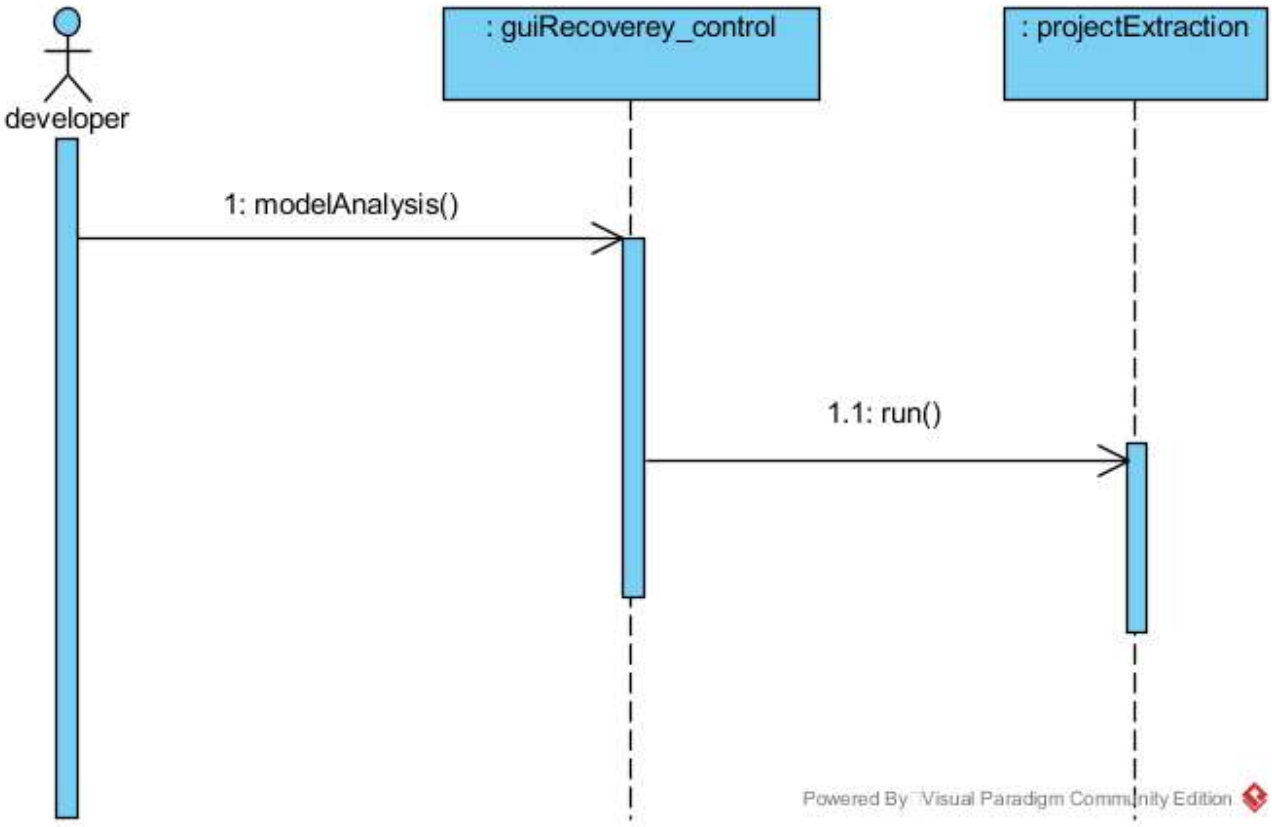
3.4.4. Dynamic model

SEQUENCE DIAGRAMS

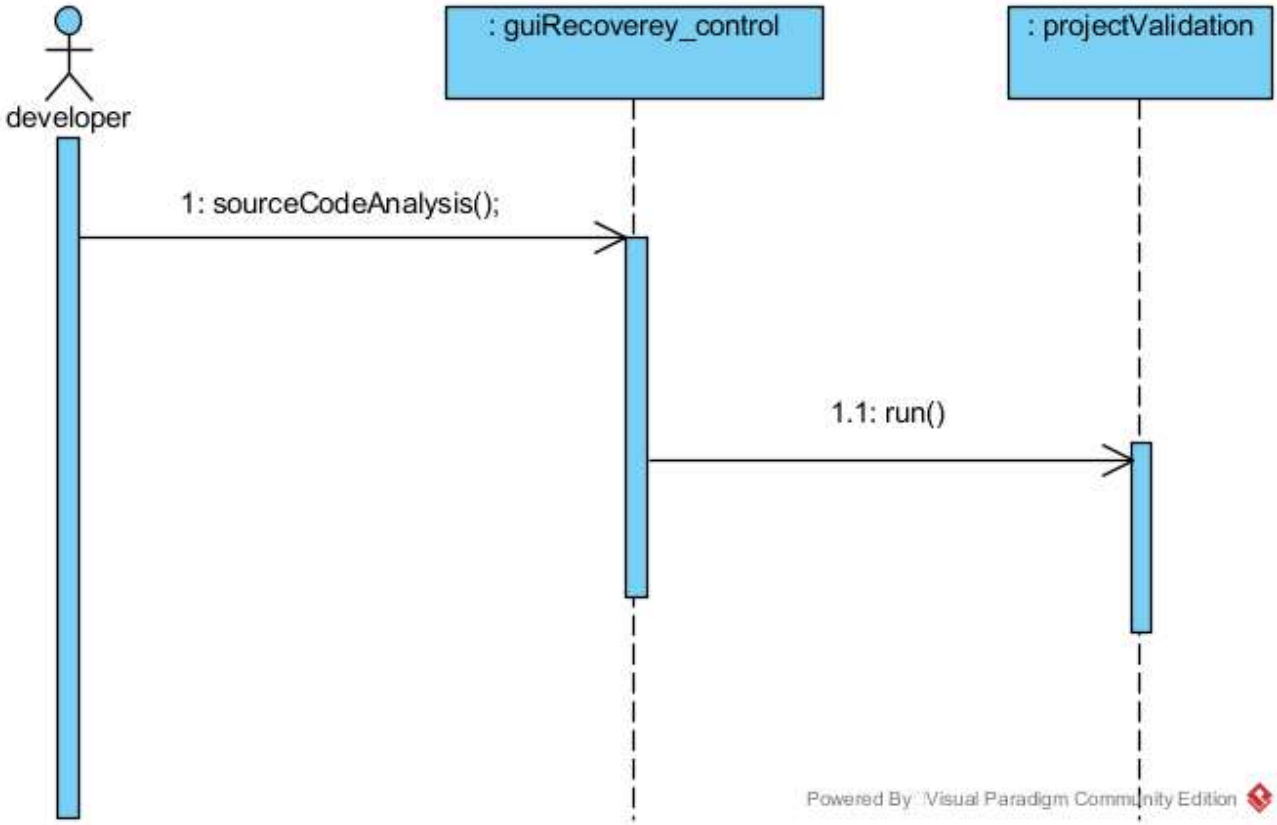
Class Diagram



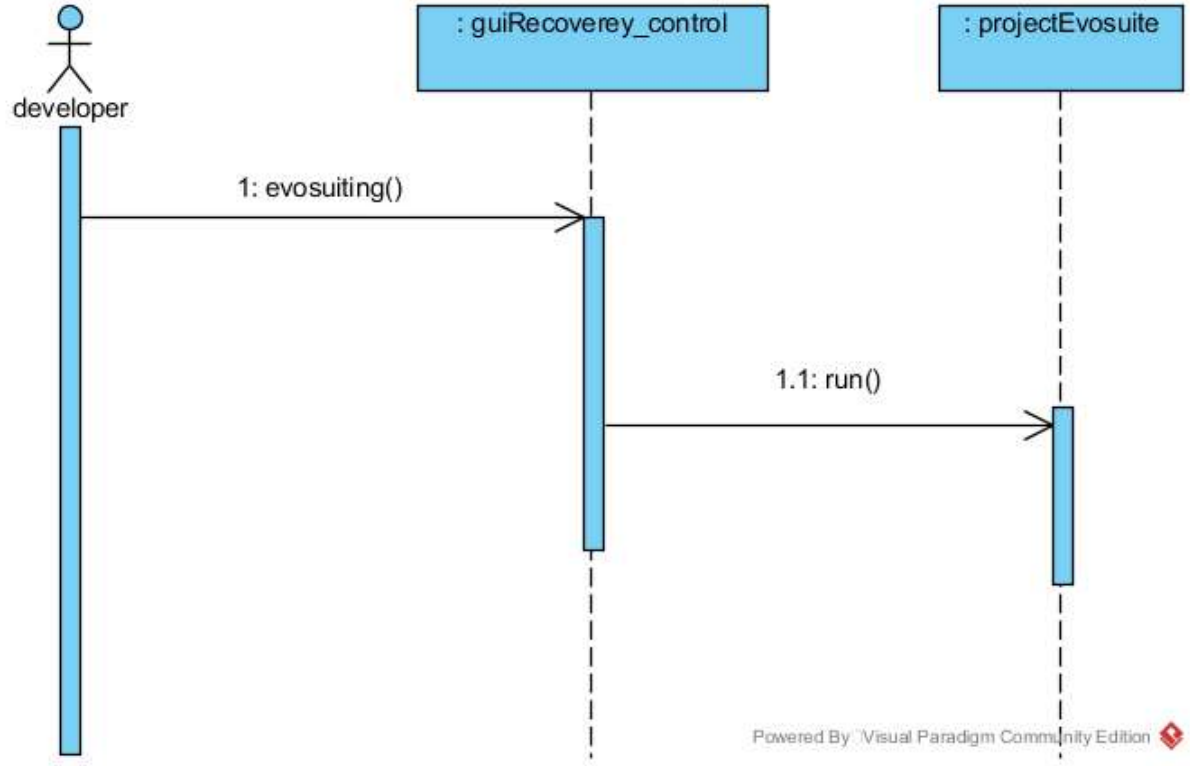
Model analysis



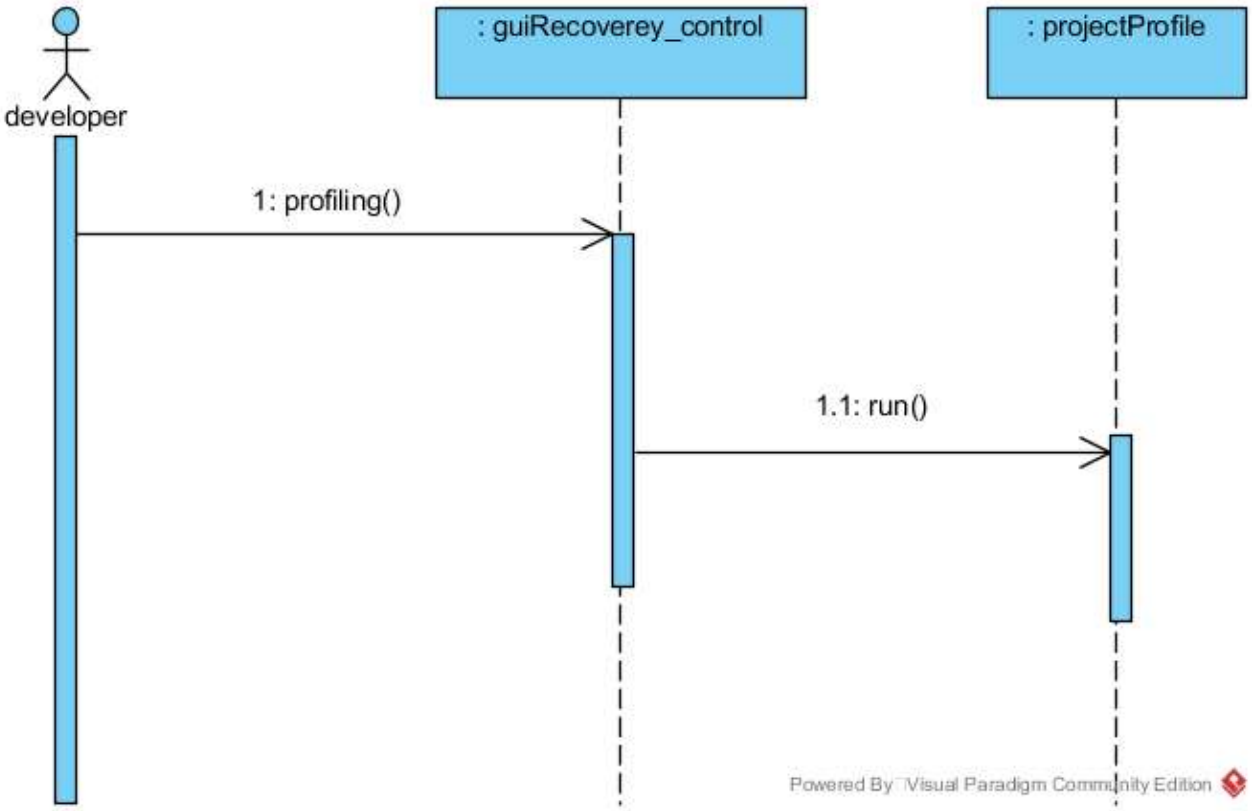
sourceCodeAnalysis


developer
: guiRecoverey_control
: projectValidation
1: sourceCodeAnalysis();
1.1: run()
Powered By Visual Paradigm Community Edition

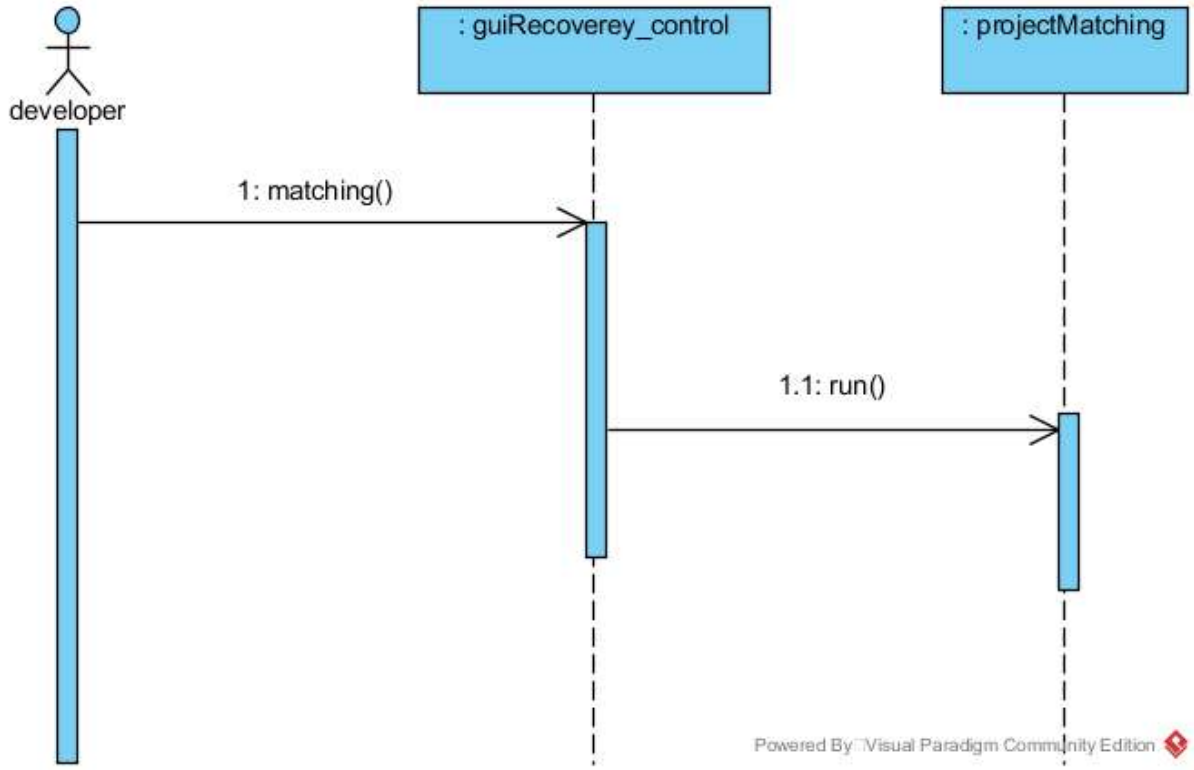
Evosuiting


developer
: guiRecoverey_control
: projectEvosuite
1: evosuiting()
1.1: run()
Powered By Visual Paradigm Community Edition

Profiling


developer
: guiRecoverey_control
: projectProfile
1: profiling()
1.1: run()
Powered By Visual Paradigm Community Edition

Matching



4. GLOSSARY

SINONIMI	DESCRIZIONE	TERMINE
Sviluppatore utente	Persona che utilizza il tool ePADs.	**developer**