Aggiunta CodeSmell cASpER

* **Change Request**

|  |  |
| --- | --- |
| **Change\_Request\_id** | 2 |
| **Priority** | Normal |
| **Description** | Implementazione del modulo di detection per il code smell “Spaghetti Code” |
| **Maintenance Type** | Perfective |
| **Component** | GUI, Parser, Analysis |
| **Product** | cASpER |
| **Severity** | Normal |

|  |  |
| --- | --- |
| **Change\_Request\_id** | 3 |
| **Priority** | Normal |
| **Description** | Implementazione del modulo di detection per il code smell “Swiss Army Knife” |
| **Maintenance Type** | Perfective |
| **Component** | GUI, Parser, Analysis |
| **Product** | cASpER |
| **Severity** | Normal |

* **Fase 1: Problem Identification**

Le change request vengono accettate, in quanto il sistema cASpER supporta l’implementazione di nuovi detector per l’individuazione di code smell.

* **Fase 2: Analysis**

Essendo che per il progetto cASpER, non esiste documentazione ma soltanto il codice sorgente, verrà eseguito un reverse engineering di quest’ultimo con lo scopo di individuare il design del sistema e i suoi requisiti funzionali.

Come primo step tramite l’ide Intellij è stato estratto in modo totalmente automatico l’object model del sistema. Esso è disponibile nella cartella Object\_Model\_cASpER.

Tramite l’analisi del object model e un’ispezione del codice sorgente è stato identificato il design del sistema. Esso può essere diviso in 6 sottosistemi:

1. Action, modulo che esegue l’operazione di avvio del sistema e avvio dell’analisi del project
2. Analysis, modulo che esegue la parte di detection dei code smell. Può essere suddiviso in due sottosistemi interni:

* Code Smell Detection, modulo in cui vengono dichiarate le varie strategie di detection
* Code Smell, modulo in cui viene implementato il code smell

1. GUI, modulo di presentazione che implementa le interfacce da mostrare all’utente
2. Parser, modulo che converte il project da analizzare
3. Refactor, modulo in cui sono implementate le strategie di refactoring per risolvere il code smell

L’ultimo step del reverse engineering e l’identificazione dei requisiti funzionali. Sono stati identificati 14 requisti funzionali:

**Requisiti per il rilevamento dei code smell**

* **RF\_RCS\_1:** Rilevamento code smell Blob

Il sistema dovrà supportare il rilevamento del code smell Blob

* **RF\_RCS\_2:** Rilevamento code smell Feature Envy

Il sistema dovrà supportare il rilevamento del code smell Feature Envy

* **RF\_RCS\_3:** Rilevamento code smell Misplaced Class

Il sistema dovrà supportare il rilevamento del code smell Misplaced Class

* **RF\_RCS\_4:** Rilevamento code smell Parallel Inheritance

Il sistema dovrà supportare il rilevamento del code smell Parallel Inheritance

* **RF\_RCS\_5:** Rilevamento code smell Promiscuous Package

Il sistema dovrà supportare il rilevamento del code smell Promiscuous Package

* **RF\_RCS\_6:** Rilevamento code smell Shotgun Surgery

Il sistema dovrà supportare il rilevamento del code smell shotgun surgery

* **RF\_RCS\_7:** Rilevamento code smell Divergent Change

Il sistema dovrà supportare il rilevamento del code smell divergent change

**Requisiti per il refactoring dei code smell**

* **RF\_RFCS\_1:** Refactoring del code smell Blob

Il Sistema dovrà eseguire il refactoring della classe in cui e presente il code smell Blob se richiesto dall’utente

* **RF\_RFCS\_2:** Refactoring del code smell Feature Envy

Il Sistema dovrà eseguire il refactoring del metodo in cui e presente il code smell Feature Envy se richiesto dall’utente

* **RF\_RFCS\_3:** Refactoring del code smell Misplaced Class

Il Sistema dovrà eseguire il refactoring della classe in cui e presente il code smell Misplaced Class se richiesto dall’utente

* **RF\_RFCS\_4:** Refactoring del code smell Parallel inheritance

Il Sistema dovrà eseguire il refactoring della classe in cui e presente il code smell Parallel inheritance se richiesto dall’utente

* **RF\_RFCS\_5:** Refactoring del code smell Shotgun Surgery

Il Sistema dovrà eseguire il refactoring della classe in cui e presente il code smell Shotgun Surgery se richiesto dall’utente

* **RF\_RFCS\_6:** Refactoring del code smell Divergent Change

Il Sistema dovrà eseguire il refactoring della classe in cui e presente il code smell Divergent Change se richiesto dall’utente

* **RF\_RFCS\_7:** Refactoring del code smell Promiscuous Package

Il Sistema dovrà eseguire il refactoring del package in cui e presente il code smell Promiscuous Package se richiesto dall’utente

**Requisiti per la visulizzazione dei code smell rilevati e del loro eventuale refactoring**

* **RF\_VLCSR\_1:** Visualizzazione lista code smell rilevalti

L’utente deve avere la possibilità di visualizzare i code smell presenti nel project analizzato

* **RF\_VCSR\_1:** Visualizzazione dettagliata code smell rilevato

L’utente deve avere la possibilità di visualizzare in modo dettagliato il code smell rilevato

* **RF\_VCSRF\_1:** Visualizzazione refactoring code smell

L’utente deve avere la possibilità di poter scegliere se eseguire il refactoring di un code smell e visualizzare l’anteprima del refactoring

Con l’intervento di manutenzione che andremo ad eseguire verranno introdotti due nuovi requisiti funzionali per il rilevamento dei code smell:

* **RF\_RCS\_8:** Rilevamento code smell Spaghetti Code

Il sistema dovrà supportare il rilevamento del code smell spaghetti code

* **RF\_RCS\_9:** Rilevamento code smell Swiss Army Knife

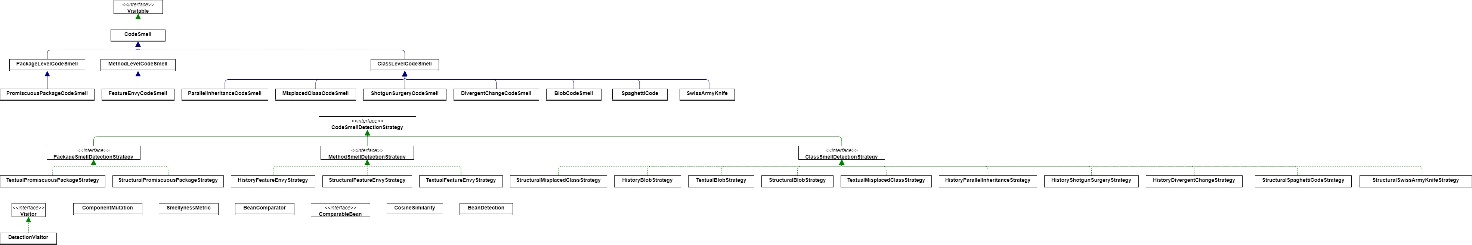
Il sistema dovrà supportare il rilevamento del code smell Swiss Army Knife.

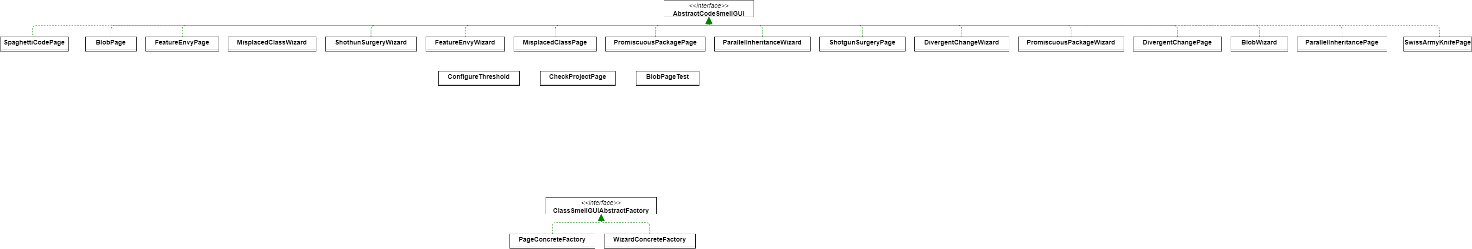
Dopo aver eseguito il reverse engineering è stata eseguita l’attività dell’impact analysis per capire l’impatto del change request sul sistema. Essa è disponibile nel file Impact Analysis cASpER CR2 CR3.

In seguito all’impact analysis per concludere la fase di analisi è stata stabilita la test Strategy e sono stati generati Master Test Plan, Regression Test Plan e System Test Plan. Essi sono disponibili nella cartella /Documentazione Testing

* **Fase 3: Design**

In tale fase si procede con l’implementazione della modifica a livello di design.





Tutte le immagini sono riportate nella cartella \Object Model cASpER

Successivamente all’implementazione della modifica a livello di design è stato creato il Test Plan ed e stata identificata una Test Case Suite.

* **Fase 4: Implementation**

In tale fase sono stati eseguiti i test di regressione, i test di unita, i test d’integrazione ed è stata implementata la modifica. I risultati dei test sono presenti negli appositi test plan,

Per l’implementazione delle strategy dei code smell abbiamo seguito i passi seguenti:

1. Abbiamo identificato una descrizione dello smell
2. Abbiamo analizzato la descrizione dello smell creando una rule card
3. Su tale rule card e stato implementato l’algoritmo di identificazione
4. Sono stati eseguiti i test di unita sulle strategy
5. Sono stati eseguiti i test d’integrazione tra parser e strategy
6. Sono state create le GUI per i code smell.

* **Fase 5: System test**

In tale fase è stato eseguito il test di sistema ed i risultati sono presenti nel documento System test plan, nella sezione System Test Report.

Inoltre è stata valutata la precision e la recall dell’impact analysis condotta nella fase 2.

La recall risulta essere 0,88 mentre la precision 1.