Reengineering Casper

* **Change Request**

|  |  |
| --- | --- |
| **Change\_Request\_id** | 1 |
| **Priority** | High |
| **Description** | Reengineering della struttura di cASpER per semplificare l’inserimento di un nuovo “Code Smell” applicando dei design pattern |
| **Maintenance Type** | Perfective |
| **Component** | GUI, Analysis |
| **Product** | cASpER |
| **Severity** | Normal |

* **Fase 1: Problem Identification**

La change request viene accettata con lo scopo di applicare il design pattern “Abstract Factory” per la creazione delle classi Wizard e Page che rientrano nel package GUI. Inoltre si applicherà il design pattern “Visitor” alla detection dei 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 suoi requisiti funzionali.

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

Tramite l’analisi del object model e del codice sorgente e stato identificato il design del sistema. Esso può essere diviso in 6 sotto sistemi:

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 2 sotto sistemi 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

Dopo aver eseguito il reverse engineering è stata eseguita l’attività dell’impact analisi per capire l’impatto del change request sul sistema. Essa e disponibile nel file Impact Analysis cASpER. In seguito ad essa per concludere la fase di analisi e stata stabilita la test Strategy e generato un Master Test Plan. Entrami sono disponibili nella cartella /Documentazione Testing

* **Fase 3: Design**

In tale fase si procede con l’implementazione della modifica a livello di design, pertanto saranno applicati:

* Il design pattern “Abstract Factory” al modulo per la gestione delle GUI
* Il design pattern “Visitor” al modulo per la detection dei code smell

Tutti i dettagli della ristrutturazione del object model sono riprtate nel documento /Ristrutturazione cASpER