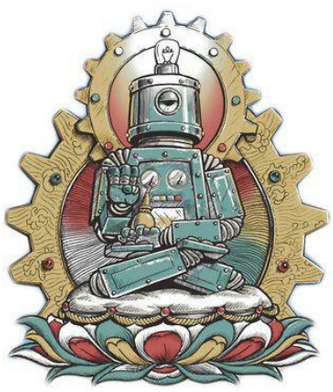
UNIVERSITÀ DEGLI STUDI DI SALERNO
CORSO DI INGEGNERIA DEL SOFTWARE

ASCETIC

Automated Code Smell Identification and Correction

IMPACT ANALYSIS REPORT

VERSION 1.0



8 novembre 2018

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**	**C.R. Interessata**	**Descrizione**	**Autore**
24/10/2018	1.0	CR3	Prima Stesura	Manuel De Stefano

Indice

1 Introduzione

In questo documento verrà analizzato l'effetto che la **Change Request n°3** avrà sul sistema.

La Change Request n°3 ha come scopo la modifica, o meglio, il *refactoring* di alcuni sottosistemi con l'obiettivo di favorirne il riuso, la manutenzione e l'evoluzione. Nella change request viene suggerito l'applicazione di ***Design Patterns*** specifici e l'uso di particolari astrazioni, proprio per poter raggiungere lo scopo.

In questo documento verrà riportata l'analisi fatta sul codice e le modifiche ad esso riportate, verificandone la fattibilità e l'impatto su altre componenti codice. I risultati di queste modifiche verranno riportati, oltre che in questo documento, anche nel documento di **Object Design**.

2 Identificazione degli Impact Set

2.1 Starting Impact Set

L'identificazione dello **starting impact set** è stata particolarmente semplice, in quanto il testo della change request riportava esplicitamente i moduli su cui effettuare le modifiche: il package ***Parser***, il package ***CodeSmellDetection*** e il package ***Refactoring***, compresi delle classi in essi contenuti. Lo *starting impact set*, dunque, comprende i seguenti componenti:

- Package **Parser**, contenente le seguenti classi:
  - BeanConverter
  - PackageParser
  - ClassPaerser
  - MethodParser
  - InstanceVariableParser
- Package **CodeSmellDetection**, contenente le seguenti classi:
  - BlobRule
  - FeatureEnvyRule
  - MisplacedClassRule
  - MisplacedComponentsUtilities
  - PromiscuousPackageRule
  - ComponentMutation
  - StructuralFeatureEnvyRule
  - StructuralMisplacedClassRule
- Package **Refactoring**, contenente le seguenti classi:
  - RefactorManager
  - MoveMethodRefactoring
  - SearchUtil

2.2 Candidate Impact Set

L'identificazione del **Candidate Impact Set** delle componenti codice è stata effettuata tramite l'utilizzo di (due) tools di analisi statica messi a disposizione dall'IDE Intellij IDEA. Il primo è la DSM, acronimo per **Dependency Structure Matrix**, ovvero, un metodo per esplorare dipendenze tra parti di programma (moduli, classi, ecc.) e fornire una rappresentazione matriciale compatta del progetto [1]. Consente di visualizzare le dipendenze tra le parti di un progetto e di evidenziare il flusso di informazioni in un progetto.

Quando è pronta, la DSM View è aperta in una finestra, e consente di esaminare le dipendenze. Gli **indici di riga** rappresentano la struttura del programma. Gli **indici di colonna** sono gli stessi dei corrispondenti indici di riga. La riga selezionata e la colonna corrispondente sono evidenziate per visualizzare le dipendenze. La colonna mostra le dipendenze **della** componente che è indice di riga (cioè quelle **da cui dipende** la componente selezionata), mentre la riga mostra le dipendenze **verso** la riga selezionata (ovvero le componenti che **dipendono da quella selezionata**) [1].

Queste ultime sono di particolare importanza per la nostra analisi, poiché, indicano componenti che possono essere impattate da una modifica fatta ad una componente da cui dipendono. Dunque, nel **candidate impact set** sono state inserite tutte le componenti che la DSM ha indicato come **direttamente dipendenti** dalle componenti presenti nello **starting impact set**. Nel nostro caso, andremo, dunque, a considerare le dipendenze verso i package **parser**, **CodeSmellDetection** e **Refactoring**.

2.2.1 Componenti dipendenti dal package Parser

Dalla Figura 1 si evincono le componenti che dipendono dal package **Parser**. Analizzando la **DSM** si può notare che le componenti che dipendono dal package in analisi sono due: il package **Actions** (con 3 dipendenze), ed il package **ProjectAnalysis** (con 1 dipendenza).

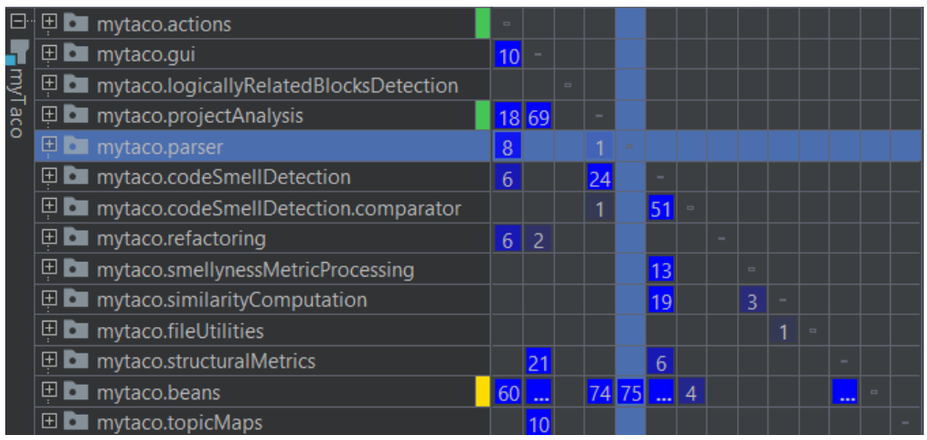


Figura 1: DSM del package Parser

Approfondendo l'analisi delle dipendenze per il package Actions, notiamo esse siano tutte legate ad un'unica classe del package Parser, ovvero la classe ***BeanConverter***. Ciò comporta che solo una modifica a tale classe potrebbe impattare sulle classi presenti nel package Actions.

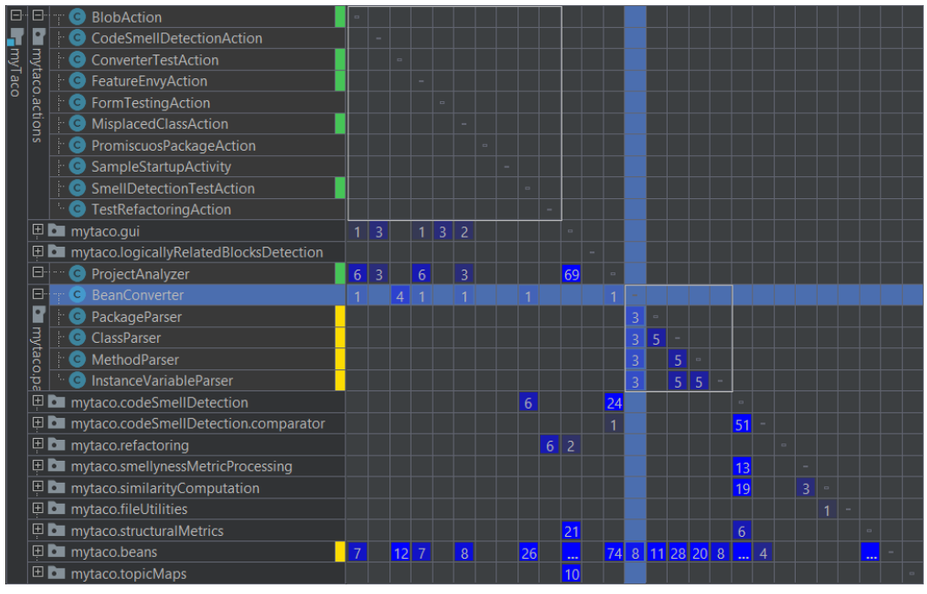


Figura 2: DSM del package Parser

La Figura 2 mostra nel dettaglio quali classi dipendono dalla classe *BeanConerter*. Nella fattispecie:

- BlobAction (1 dipendenza)
- FeatureEnvyAction (1 dipendenza)
- MisplacedClassAction (1 dipendenza)

- ProjectAnalyzer (1 dipendenza)

Dunque queste classi saranno inserite nel **Candidate Impact Set**.

2.2.2 Componenti dipendenti dal package CodeSmellDetection

2.2.3 Componenti dipendenti dal package Refactoring

Il secondo tool utilizzato è lo strumento di **Data Flow Analysis**, che permette di analizzare il flusso dati **da** o **verso** il simbolo selezionato, che può essere un'espressione, una variabile o un metodo. L'output presenta una traccia di nodi che corrispondono a catene di assegnamento o invocazioni di metodi, entranti o uscenti al simbolo selezionato, a seconda dal tipo di analisi richiesta. Nel nostro caso, è stato utile analizzare il flusso dati uscente dai vari metodi delle classi presenti nello **starting impact set**, in modo da raffinare l'insieme calcolato precedentemente tramite la DSM.

3 Analisi Post-Modifica e Calcolo di Metriche

Riferimenti bibliografici

[1] Maria Khalusova. "IntelliJ IDEA: Dependency Analysis with DSM". *Intellij IDEA Blog*. Jet Brains s.r.o, 2008. Web. Consultato il 7 Novembre 2018.

[2] CDR. "Analyzing Dataflow with IntelliJ IDEA". *Intellij IDEA Blog*. Jet Brains s.r.o, 2009. Web. Consultato il 7 Novembre 2018.