Laurea Magistrale in informatica
Università di Salerno

Impact Analisys

Team members
Saverio Napolitano - 0522501400
Gerardo Festa - 0522501452
Alessandra Parziale - 0522501422

https://github.com/GerardoFesta/infozilla

Sommario

1. Descrizione modifiche Change Request 1(CR_1)

La presente richiesta di modifica è finalizzata a risolvere alcune problematiche riscontrate nell'ambito dell'applicazione delle patch. La criticità è stata riscontrata nel processo di testing e documentata nell'incident report con id UTI1, UTI2, UTI4, di seguito raggruppate.

- **Descrizione della Problematica:**
  La prima problematica riscontrata riguarda la corretta identificazione del punto di inizio delle patch che comporta la non eliminazione del componente e la corrispettiva visualizzazione all'interno del file "Cleaned".

  Comportamento Attuale:
  Attualmente, il processo di identificazione si basa sulla ricerca dell'header presente nella patch all'interno del testo dato in input. Tuttavia, questa metodologia di identificazione si è dimostrata non corretta a causa della differenza di formattazione tra l'header della patch e il testo corrispondente.

  Proposta di Modifica:
  Si propone di adottare una nuova strategia per l'identificazione del punto di inizio delle patch all'interno del file dato in input. Questa strategia dovrebbe essere basata su un criterio di identificazione più robusto e accurato, che tenga conto delle differenze di formattazione tra l'header della patch e il testo.

  Benefici Previsti:
  L'implementazione di questa modifica dovrebbe portare ai seguenti benefici:

  1. Miglioramento della precisione nell'identificazione delle patch all'interno del testo dato in input con conseguente individuazione della patch ed eliminazione dal file "Cleaned".

- **Descrizione della Problematica:**
  La seconda problematica affrontata nella presente richiesta di modifica riguarda l'identificazione del punto di fine delle patch e della gestione di patch multiple all'interno del nostro sistema. Queste problematiche influenzano la corretta identificazione delle patch.

Comportamento Attuale:
Abbiamo notato che tale malfunzionamento è principalmente causato dalla discrepanza tra i caratteri presenti nella patch e quelli nel file di destinazione. In particolare, dalla presenza di caratteri come "\n" e "\r".

Inoltre, la gestione di patch multiple è risultata problematica. Attualmente, il sistema esegue il match considerando la prima patch, ignorando le patch successive con lo stesso header. Questo comporta problemi di riconoscimento e applicazione delle patch multiple.

Proposta di Modifica:
Si propone di adottare le seguenti modifiche per affrontare le problematiche descritte:

1. Correzione del riconoscimento della fine della patch: Rivedere il processo di riconoscimento in modo da gestire correttamente i caratteri speciali come "\n" e "\r", garantendo un confronto accurato tra la patch e il testo di input.

2. Gestione di Patch Multiple: Implementare un meccanismo che consenta al sistema di riconoscere e applicare correttamente patch multiple. Dopo il riconoscimento della prima patch, incrementare il punto di partenza ("start") per la successiva ricerca, impostandolo ad un valore pari alla fine della patch precedente.

Benefici Previsti:
L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Corretto riconoscimento della fine della patch, consentendo un confronto accurato tra la patch e il testo dato in input.
2. Gestione efficiente di patch multiple, garantendo che tutte le patch vengano riconosciute e applicate correttamente.

- **Descrizione della Problematica:**
  La terza problematica affrontata nella presente richiesta di modifica è finalizzata a risolvere l'errata individuazione degli hunk.
  Se all'interno dell'hunk text sono situati degli "\n\n" oppure un testo che non inizia con lo spazio, allora, l'applicazione elimina tutti gli elementi dell'hunk text che si trovano al di sotto.
  Un ulteriore problema riguarda l'ultima linea dell'hunk text che non viene identificata.

Comportamento Attuale:
Il problema principale risiede nella formattazione del testo degli hunk, in cui la presenza di "\n\n" viene erroneamente interpretata come la fine di un hunk valido. Tuttavia, questa interpretazione può portare al non riconoscimento di elementi validi all'interno dell'hunk.

Proposta di Modifica:
Si propone di apportare le seguenti modifiche al processo di identificazione degli hunk:

1. Rilevamento delle Linee Vuote: Durante la scansione, se si incontrano linee vuote (\n), queste devono essere aggiunte all'hunk text. Ciò garantisce che le linee vuote tra le parti dell'hunk non vengano eliminate erroneamente e quindi che i successivi elementi vengano riconosciuti correttamente.

2. Terminazione dell'Hunk: Dopo aver identificato la prima linea di hunk, procedere alla verifica delle linee successive. La linea successiva in base alla formattazione dell'hunk text sarà una linea vuota, se successivamente ci sarà una linea di hunk non valida, interrompere la scansione dell'hunk text. Se l'ultima linea è una linea di hunk valida, aggiungerla all'hunk text completando la scansione dell'hunk.

Benefici Previsti:
L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Riconoscimento accurato degli hunk, evitando l'eliminazione errata di elementi validi.
2. Gestione delle linee vuote tra le parti dell'hunk senza perdita di dati.
3. Miglioramento complessivo della precisione del processo di identificazione degli hunk.

1.1. Analisi d'impatto

Starting Impact Set - 5 Componenti

Sono state individuate le seguenti classi e metodi:

- *FilterPatches*: *getPatches*
- *PatchParser*: *isHunkLine*, *FindAllHunks*, *ParseForPatches*, *FindNextIndex.*

	FilterPatches - getPatches	PatchParser - isHunkLine	PatchParser - findAllHunks	PatchParser - parseForPatches	PatchParser - findNextIndex
FilterPatches - getPatches	\	3	2	1	3
PatchParser - isHunkLine		\			
PatchParser - findAllHunks		1	\		
PatchParser - parseForPatches		2	1	\	2
PatchParser - findNextIndex					\
FilterChainEclipse	1		3	2	
partitionbyIndex					1

Figura 1: ***Matrice di raggiungibilità con distanza***

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set **(CIS)** è stata generata la matrice di raggiungibilità considerando l'approccio distance based(fino al livello 3 di propagazione) per ridurre le occorrenze di false positive **[Figura 1]** e sono state analizzate le componenti influenzate dalle modifiche di *FilterPatches* e *PatchParser* ad una distanza <= di 3.

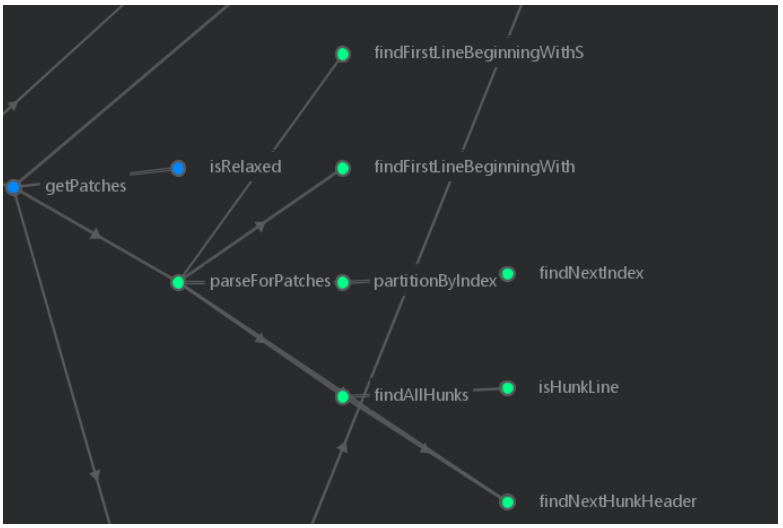

Figura 2: ***Call graph***

Inoltre, come evidenziato nel documento dell'Incident Report, le componenti coinvolte sono *FilterPatches* e *PatchParser*. *PatchParser* ha come componente principale il metodo *parseForPatches*; analizzando il Call Graph **[Figura 2]** della classe,

aggiungiamo, come oggetti che potrebbero essere impattati dalla modifica, tutti i metodi che si trovano alla distanza di una chiamata da *parseForPatches*.
Inoltre, risalendo nel grafo delle chiamate a partire da *parseForPatches*, troviamo *getPatches*, ovvero il metodo che viene lanciato a partire dal *runFilter* di *FilterPatches* e anche questo è aggiunto al set.

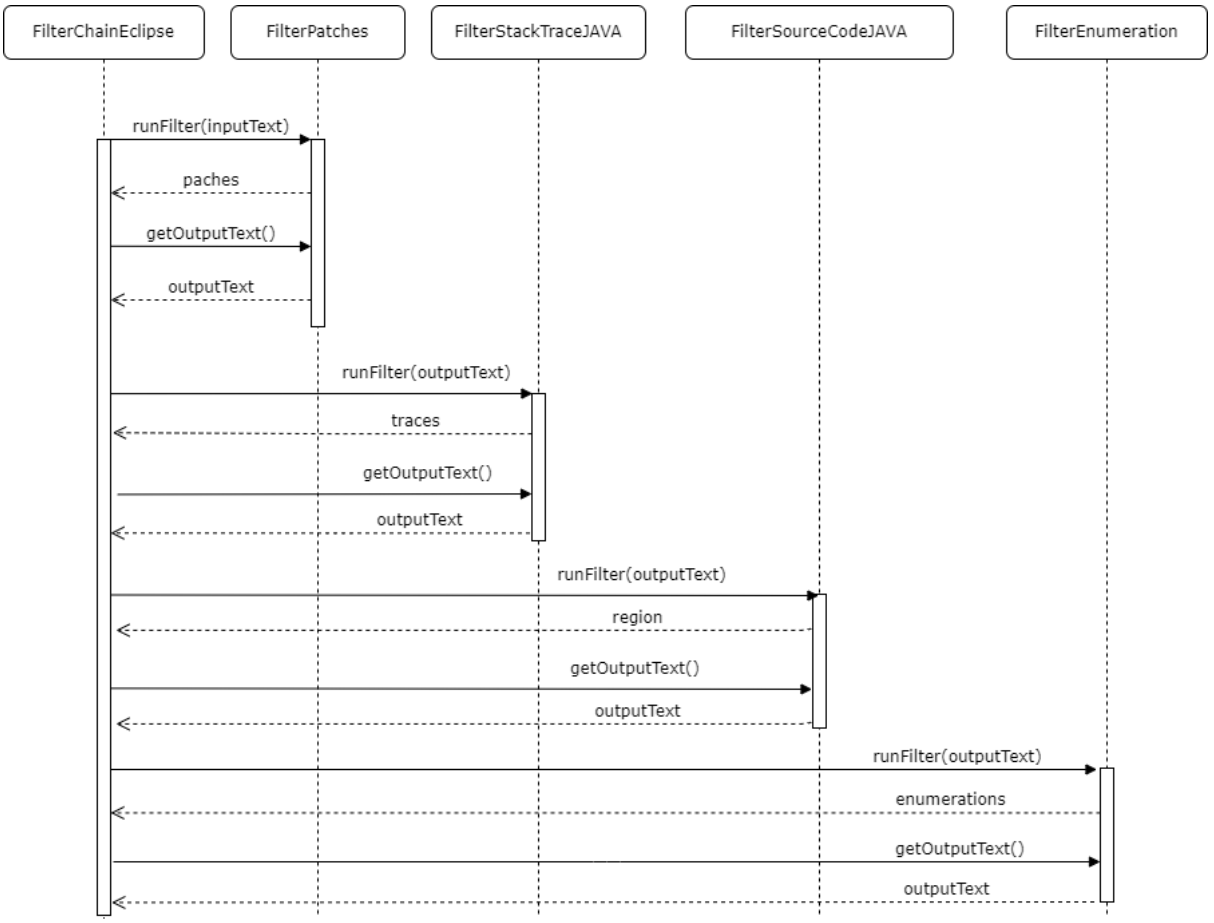

Figura 3: ***Sequence Diagram FilterChainEclipse***

Uso Variabili	outputD44	outputI46	patchesD46	outputA47	outputI49	tracesD49	outputA50	outputI52	regionsD52	outpuA53	outputI55	enumsD55	outputA57
outputD44	1	1											
outputI46		1	1	1									
patchesD46			1										
outputA47				1	1								
outputI49					1	1	1						
tracesD49						1							
outputA50							1	1					
outputI52								1	1	1			
regionsD52									1				
outpuA53										1	1		
outputI55											1	1	1
enumsD55												1	
outputA57													1

Legenda:
D -> definition
I -> input to an external module
A -> assignment

Figura 4: ***Matrice di Propagazione delle Variabili***

Considerando invece *FilterPatches*, questa viene utilizzata all'interno di *FilterChainEclipse*. Osservando il Sequence Diagram **[Figura 3]** della classe e ricostruendo dal codice la Matrice di Propagazione della Variabili **[Figura 4]**, utile a computare i Ripple Effects, capiamo come l'output text di un filtro sia l'input text del filtro successivo, e quindi c'è necessità di tenere in conto un possibile impatto su questi filtri. Ciò sarà verificabile mediante test di regressione che comprende i test di integrazione. Aggiungiamo dunque anche gli altri filtri al candidate impact set.
Supponiamo infatti che la modifica comporti un cambiamento dell'output non strettamente legato alla funzionalità aggiunta, ad esempio l'aggiunta/rimozione di caratteri di escape. A quel punto potrebbe essere necessario adattare gli altri filtri a riconoscere queste istanze, in modo da preservare il corretto comportamento.

Per le ragioni descritte all'interno del Candidate Impact Set (CIS) abbiamo inserito:

Candidate Impact Set - 10 Componenti

- *FilterPatches*: *getPatches*
- *FilterStackTracesJAVA*: *getStackTraces*
- *FilterSourceCodeJAVA*: *getCodeRegions*
- *FilterEnumeration*: *getCharEnums*, *getNumEnums*, *getItemizations*
- *PatchParser*: *isHunkLine*, *FindAllHunks*, *ParseForPatches*, *FindNextIndex*

1.2. Implementazione modifiche

Attraverso l'assegnazione *"int patchStart = text.indexOf(p.getHeader())"*, si cercava di individuare all'interno del testo fornito in input la parte che corrispondeva all'Header.
Tuttavia, a causa del modo in cui l'header viene generato, questa corrispondenza non è mai verificata, poiché nel header vengono aggiunti caratteri di ritorno a capo (\r) e nuova riga (\n), il che comporta una struttura diversa rispetto al testo di input **[Figura 5]**.

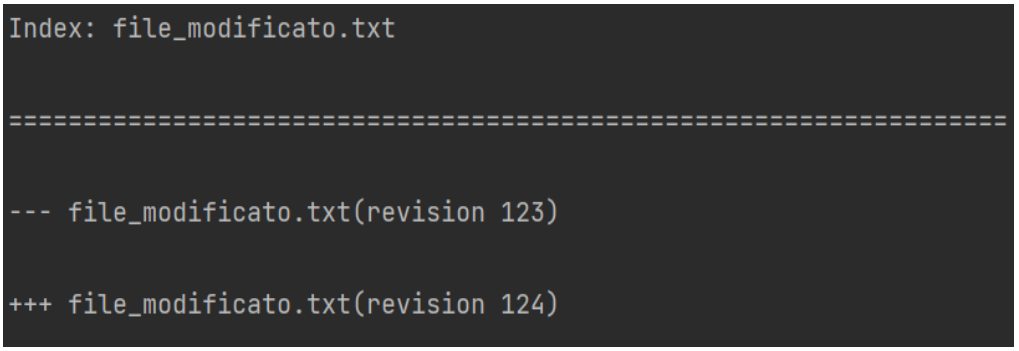

Figura 5: ***Esempio Header***

Per quanto riguarda l'identificazione dell'inizio della patch (patch start), abbiamo cercato una corrispondenza tra "index" e il nome corrispondente della patch. In caso di patch con lo stesso nome, abbiamo progressivamente aggiornato l'indicatore di inizio dopo ogni riconoscimento.

Per quanto riguarda la fine della patch (patch end), dato che i principali problemi riguardavano la formattazione, abbiamo apportato modifiche al riconoscimento della fine della patch. Inizialmente, abbiamo modificato la formattazione, sostituendo i doppi ritorni a capo (\n\n) con un singolo ritorno a capo (\n) e gli (\r) con spazi.

Successivamente, abbiamo modificato "lastIndexOf" poiché non riusciva a riconoscere adeguatamente la fine della patch.
Abbiamo calcolato l'inizio del "texHunk" utilizzando il metodo "indexOf" e abbiamo aggiunto la lunghezza dell'intero "textHunk" per ottenere il valore della fine della patch.

Per riconoscere correttamente gli "Hunk", è stato apportato un cambiamento al metodo "findAllHunks", poiché inizialmente non era stato previsto il caso in cui vi fossero due ritorni a capo (\n\n). Quindi la

modifica ha previsto l’esaminazione della linea corrente. Se è vuota, essa viene aggiunta “all'hunktext". Altrimenti, se la linea corrente non è una "HunkLine" e la linea successiva è vuota, e la linea dopo quella è una "HunkLine", continuiamo a esaminare le linee successive. In caso contrario, usciamo dal processo di verifica.

Per identificare la linea finale “dell'HunkText", abbiamo effettuato una verifica per determinare se l'ultima linea fosse un "HunkLine", in modo da poterla aggiungere all'interno dell'"HunkText".

Actual Impact Set - 3 Componenti

- *PatchParser*: modifica dei metodi *ParseForPatches*, *FindAllHunks* e *FindNextIndex*.

False Positive Impact Set - 7 Componenti

- *FilterPatches*: getPatches
- *PatchParser*: *isHunkLine*
- *FilterStackTracesJAVA*: *getStackTraces*
- *FilterSourceCodeJAVA*: *getCodeRegions*
- *FilterEnumeration*: *getCharEnums*, *getNumEnums*, *getItemizations*

Discovered Impact Set - 0 Componenti

1.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 3/3 = 1.
Precision = (CIS ∩ AIS) ÷ CIS = 3/10 = 0.3.

2. Descrizione modifiche Change Request 2(CR_2)

Descrizione della Problematica:

La presente richiesta di modifica è finalizzata ad ampliare la fase di riconoscimento delle patch, introducendo anche il formato Unified Diff. L'applicazione non è in grado di riconoscere correttamente tali patch, portando a problemi nella gestione delle patch e nel conteggio degli hunk qualora il seguente formato si trovasse immediatamente al di sotto del formato riconosciuto.

Comportamento Attuale:

Attualmente, le patch in formato Unified Diff non vengono riconosciute.

Proposta di Modifica:

Si propone di apportare le seguenti modifiche per implementare il riconoscimento delle patch in formato Unified Diff:

1. Aggiunta di una Variabile Booleana: Nella classe `Patch`, aggiungeremo una variabile booleana per indicare se la patch è in formato Unified Diff oppure no. Questa variabile verrà impostata a `true` quando la prima linea della patch inizia con "diff --git".

2. Verifica nel Metodo `findNextIndex`: Nel metodo `findNextIndex`, verificheremo se il formato della patch è Unified Diff. In tal caso, controlleremo se la linea iniziale inizia con "diff --git" e se la successiva inizia con "index <spazio>".

3. Controllo aggiuntivo in PatchParser: Quando si localizzano le patch nel `sourceCode`, utilizzeremo la variabile booleana per determinare se la patch è di tipo Unified Diff. In tal caso, il punto di partenza della patch deve iniziare con "diff --git".

Benefici Previsti:

L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Riconoscimento accurato delle patch in formato Unified Diff.
2. Evitare errori nel conteggio degli hunk causati da patch non riconosciute.
3. Miglioramento complessivo della precisione e dell'affidabilità nell'identificazione delle patch.

2.1. Analisi d'impatto

Starting Impact Set - 1 Componenti

Sono state individuate le seguenti classi e metodi:

- PatchParser: ParseForPatches

	PatchParser: ParseForPatches
FilterPatches - getPatches	1
FilterChainEclipse	2
FilterStackTracesJAVA	3
FilterEnumeration	3
FilterSourceCoseJAVA	3

Figura 6: ***Matrice di raggiungibilità con distanza***

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set **(CIS)** è stata generata la matrice di raggiungibilità considerando l'approccio distance based(fino al livello 3 di propagazione) per ridurre le occorrenze di false positive **(Figura 6)**

Inoltre, per evidenziare gli usi e le dichiarazioni è stata utilizzata la seguente Matrice di Propagazione delle Variabili **(Figura 7)** per le variabili della classe FilterChainEclipse:

Uso Variabili	outputD44	outputI46	patchesD46	outputA47	outputI49	tracesD49	outputA50	outputI52	regionsD52	outpuA53	outputI55	enumsD55	outputA57
outputD44	1	1											
outputI46		1	1	1									
patchesD46			1										
outputA47				1	1								
outputI49					1	1	1						
tracesD49						1							
outputA50							1	1					
outputI52								1	1	1			
regionsD52									1				
outpuA53										1	1		
outputI55											1	1	1
enumsD55												1	
outputA57													1

Legenda:
D -> definition
I -> input to an external module
A -> assignment

Figura 7: ***Matrice di Propagazione delle Variabili***

Quello che è stato riscontrato è che l'output di un filtro è l'input del successivo quindi la modifica di FilterPaches potrebbe comportare una modifica ai filtri StackTraces, SourceCode e Enumeration.

Per le ragioni descritte all'interno del Candidate Impact Set (CIS) abbiamo inserito:

Candidate Impact Set - 7 Componenti

- *FilterPatches*: *getPatches*
- *FilterStackTracesJAVA*: *getStackTraces*
- *FilterSourceCodeJAVA*: *getCodeRegions*
- *FilterEnumeration*: *getCharEnums*, *getNumEnums* e *getItemizations*
- *PatchParser*: *ParseForPatches*

2.2. Implementazione modifiche

Per consentire il riconoscimento delle patch nel formato Unified Diff, è stato aggiunto un booleano nella classe Patch e sono state apportate alcune modifiche al metodo "*parseForPatches*".
In particolare, abbiamo aggiunto il riconoscimento della prima linea della nuova patch, che deve essere "*diff --git*" e nel metodo "*findNextIndex*", abbiamo effettuato delle operazioni di riconoscimento per individuare l'inizio della patch.

Actual Impact Set - 4 Componenti

- *PatchParser*: modifica dei metodi *ParseForPatches* e *FindNextIndex*
- Patch: inserimento della variabile UnifiedDiff e dei relativi metodi *setUnifiedDiff* e *isUnifiedDiff*

False Positive Impact Set - 6 Componenti

- *FilterPatches*: *getPatches*
- *FilterStackTracesJAVA*: *getStackTraces*
- *FilterSourceCodeJAVA*: *getCodeRegions*
- *FilterEnumeration*: *getCharEnums*, *getNumEnums*, *getItemizations*

Discovered Impact Set - 3 Componenti

- *Patch*: *setUnifiedDiff* e *isUnifiedDiff*
- *PatchParser*: *FindNextIndex*

2.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 1/4 = 0.25.
Precision = (CIS ∩ AIS) ÷ CIS = 1/7 = 0.14.

3. Descrizione modifiche Change Request 3(CR_3)

Descrizione della Problematica:
La presente richiesta di modifica è finalizzata a risolvere alcune problematiche legate al riconoscimento e alla gestione delle enumerazioni all'interno dell'applicazione. Vi sono errori nel conteggio dei simboli e nella formattazione delle enumerazioni, che richiedono modifiche per una gestione corretta.

Comportamento Attuale:

1. Conteggio Errato delle Enumerazioni della Stessa Tipologia: L'applicazione non riconosce correttamente le enumerazioni della stessa tipologia, in quanto il contatore viene erroneamente azzerato, ma il primo elemento dell'enumerazione successiva è già stato letto. Ciò comporta che se dopo la prima enumerazione è presente un’altra enumerazione di soli due elementi allora quest’ultima non verrà riconosciuta.

2. Enumerazione Contiene Tutte le Enumerazioni Successive: L'applicazione riconosce le diverse enumerazioni, ma la prima enumerazione contiene tutte le successive, la seconda contiene le successive a quella, e così via.

3. Eliminazione dell'Ultimo Elemento dell'Enumerazione: L'applicazione non elimina l'ultimo elemento dell'enumerazione.

Proposta di Modifica:

1. Conteggio delle Enumerazioni della Stessa Tipologia: Modificheremo il comportamento dei metodi che ricercano le enumerazioni, andando a rivedere i vari conteggi che vengono effettuati.

2. Gestione delle Enumerazioni Successive: la modalità di inserimento delle enumerazioni verrà revisionata.

3. Eliminazione dell'Ultimo Elemento: Verificheremo il caso limite dell'eliminazione dell'ultimo elemento.

Benefici Previsti:
L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Conteggio accurato delle enumerazioni della stessa tipologia.
2. Corretta formattazione delle enumerazioni senza l'inclusione di tutte le successive.
3. Eliminazione corretta dell'ultimo elemento dell'enumerazione.

3.1. Analisi d'impatto

Starting Impact Set - 4 Componenti

Sono state individuate le seguenti classi e metodi:

- FilterEnumeration: getCharEnums, getNumEnums, getItemizations e getEnumerationAndItemizations.

	FilterEnumeration - getCharEnums	FilterEnumeration - getNumEnums	FilterEnumeration - getItemization	FilterEnumeration - getEnumerationAndItemizations
FilterEnumeration - getCharEnums	\			
FilterEnumeration - getNumEnums		\		
FilterEnumeration - getItemization			\	
FilterEnumeration - getEnumerationAndItemizations	1	1	1	\
FilterChainEclipse	2	2	2	1

Figura 8: ***Matrice di raggiungibilità con distanza***

Per individuare le componenti coinvolte e stabilire il Candidate Impact Set **(CIS)** è stata generata la matrice di raggiungibilità considerando l'approccio distance based(tenendo conto le componenti influenzate dalle modifiche di FilterEnumeration ad una distanza <= di 3) per ridurre le occorrenze di false positive **[Figura 8]**.

Inoltre, per evidenziare gli usi e le dichiarazioni è stata utilizzata la seguente Matrice di Propagazione delle Variabili **[Figura 9]** per le variabili della classe FilterChainEclipse:

Uso Variabili	enumsD55	outputA57
enumsD55	1	
outputA57		1

Legenda:
D -> definition
A -> assignment

Figura 9: ***Matrice di Propagazione delle Variabili***

Com'è visibile, essendo le enumerazioni l'ultimo filtro che viene richiamato, l'impatto della modifica non viene propagato ad altri filtri attraverso i dati. Questo ci suggerisce che, al netto della scoperta di ulteriori componenti da modificare, nel regression testing ci basterà valutare il comportamento di unità del filtro.

Per le ragioni descritte all'interno del Candidate Impact Set (CIS) abbiamo inserito:

Candidate Impact Set - 4 Componenti

- *FilterEnumeration*: *getCharEnums*, *getNumEnums*, *getItemizations* e *getEnumerationAndItemizations*.

3.2. Implementazione modifiche

Per consentire il conteggio delle Enumerazioni della stessa tipologia, il valore di "*symbolCount*" è stato modificato da 0 a 1. In precedenza, il conteggio veniva azzerato dopo aver individuato il primo elemento.

Per risolvere il problema legato alle Enumerazioni che contengono tutte le Enumerazioni successive, abbiamo eliminato un ciclo "for" che inseriva gli elementi in modo scorretto.

Infine, per l'eliminazione dell'ultimo elemento dell'Enumerazione, abbiamo aggiunto "+1" al parametro passato a "*markForDeletion*".

Actual Impact Set - 4 Componenti

- *FilterEnumeration*: modifica dei metodi *FilterLine*, *CreateEnumeration*, *GetCharEnums*, *GetNumEnums*

False Positive Impact Set - 0 Componente

Discovered Impact Set - 2 Componenti

- FilterEnumeration: FilterLine e CreateEnumeration

3.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 2/4 = 0.5.
Precision = (CIS ∩ AIS) ÷ CIS = 2/4 = 0.5.

4. Descrizione modifiche Change Request 4(CR_4)

Attualmente Infozilla è in grado di esportare solo in formato xml. La modifica comporta l'inserimento di quattro casi nella classe main dove vengono create quattro tipologia di file in cui verranno salvati gli output: json, xls, csv, xml (xml anche come default). La tipologia di output desiderata deve essere specificata dell'utente nelle configurazioni altrimenti di default è xml.
La modifica avviene anche in DataExportUtility con l'aggiunta di un metodo per ogni filtro e per ogni tipologia di file al fine di formattare il formato desiderato dei vari filtri.

Descrizione della Problematica:
La presente richiesta di modifica è finalizzata a migliorare la funzionalità di Infozilla consentendo l'esportazione dei dati in diversi formati di file, tra cui JSON, XLS, CSV, oltre al formato XML attualmente supportato.

Modifiche Proposte:

1. Aggiunta di Supporto per Formati di File Diversi: Nella classe `main`, saranno aggiunti quattro casi per gestire la creazione di quattro tipologie di file di esportazione: JSON, XLS, CSV e XML (come formato predefinito). Gli utenti potranno specificare la tipologia di file desiderata nelle configurazioni o, in caso contrario, verrà utilizzato il formato XML come predefinito.

2. Modifiche a DataExportUtility: Sarà necessario apportare modifiche alla classe `DataExportUtility` per supportare l'esportazione dei dati nei formati desiderati. Saranno aggiunti metodi specifici per ciascun filtro e ciascuna tipologia di file al fine di formattare correttamente i dati per l'esportazione.

Benefici Previsti:
L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Maggiore flessibilità per gli utenti nell'esportazione dei dati.
2. Maggiore compatibilità con diverse applicazioni e strumenti di gestione dati.
3. Maggiore adozione dell'applicazione grazie alla possibilità di esportare dati in formati più comuni.

4.1. Analisi d’impatto

La modifica comporta l’introduzione di nuovi metodi all’interno della classe DataExportUtility che permettono di eseguire l’esportazione in diversi formati. Oltre a questo, si prevede una modifica della classe *Main*, introducendo una modifica del metodo *run*.
In questo caso, dunque, non viene realizzata la matrice delle dipendenze, in quanto i metodi vengono richiamati solo dal metodo *run* della classe *Main*, ovvero il metodo principale di questa classe e, in generale, il primo richiamato quando viene lanciata un’esecuzione del tool.

Vengono dunque formulati Starting e Candidate Impact Set come segue:

Starting Impact Set - 5 Componenti

Sono state individuate le seguenti classi e metodi:

- Main: process
- DataExportUtility: getXMLExportOfStackTraces, getXMLExportOfPatches, getXMLExportOfSourceCode, getXMLExportOfEnumerations

Candidate Impact Set - 5 Componenti

- *Main*: process
- *DataExportUtility*: *getXMLExportOfStackTraces*, *getXMLExportOfPatches*, *getXMLExportOfSourceCode*, *getXMLExportOfEnumerations*

4.2. Implementazione modifiche

Per l'aggiunta di supporto per diversi formati di file (come JSON, XLS, CSV e XML), sono stati inseriti metodi specifici per ciascun formato e per ciascun filtro nella classe *DataExportUtility*.

Actual Impact Set - 13 Componenti

- *Main*: modifica del metodo Process e inserimento di in @option
- *DataExportUtility*: Inserimento dei metodi
  per JSON: *getJsonExportEnumeration*, *getJsonExportOfSourceCode*, *getJsonExportOfPatches*, *getJsonExportStackTraces*
  per CSV: *ExportPatchesToCSV*, *getCSVExportOfSourceCode*, *getCSVExportOfEnumeration*, *getCSVExportOfStackTraces*
  per XLS : *getXLSExportOfStackTraces*, *getXLSExportOfPatches*, *getXLSExportOfSourceCode*, *getXLSExportOfEnumeration*

False Positive Impact Set - 4 Componenti

- *DataExportUtility*: *getXMLExportOfStackTraces*, *getXMLExportOfPatches*, *getXMLExportOfSourceCode*, *getXMLExportOfEnumerations*

Discovered Impact Set - 12 Componenti

- *DataExportUtility: getJsonExportEnumeration, getJsonExportOfSourceCode, getJsonExportOfPatches, getJsonExportStackTraces, ExportPatchesToCSV, getCSVExportOfSourceCode, getCSVExportOfEnumeration, getCSVExportOfStackTraces, getXLSExportOfStackTraces, getXLSExportOfPatches, getXLSExportOfSourceCode, getXLSExportOfEnumeration*

4.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 1/13 = 0.07.
Precision = (CIS ∩ AIS) ÷ CIS = 1/12 = 0.08.

5. Descrizione modifiche Change Request 5(CR_5)

Attualmente Infozilla è in grado di eseguire le estrazioni delle informazioni solo a partire da file locali. Ciò comporta la necessità di scaricare il contenuto di una pagina web in locale, procedura che può risultare complessa o fastidiosa per gli utenti. La modifica comporta l'inserimento di un parametro della classe Main (e di riflesso nella command line) in modo da poter inserire un url web di una issue GitHub, permettendo che il sistema scarichi autonomamente la issue ed esegua l'elaborazione. La modifica comporta l'aggiunta di un nuovo modulo atto alla formattazione della issue.

Descrizione della Problematica:
La presente richiesta di modifica è finalizzata a migliorare la funzionalità di Infozilla consentendo l'elaborazione su issue di repository Github remote, non presenti quindi sul proprio computer.

Modifiche Proposte:

1. Aggiunta di Supporto per l'analisi da remoto: Nella classe `main`, sarà aggiunto un parametro che permetta agli utenti di inserire uno o più url di issue Github. Naturalmente, verrà garantito il funzionamento con file locali.
2. Aggiunta modulo di download: Sarà necessario aggiungere un nuovo modulo che permetta, tramite l'url fornito, di scaricare e formattare in maniera corretta il testo della issue e i commenti presenti.

Benefici Previsti:
L'implementazione di queste modifiche dovrebbe portare ai seguenti benefici:

1. Maggiore comodità per gli utenti nell'utilizzo del tool.

5.1. Analisi d’impatto

Starting Impact Set - 3 Componenti

Sono state individuate le seguenti classi e metodi:

- Main: run
- *IssueScraper*: *runScraper*, *scrapeIssue*

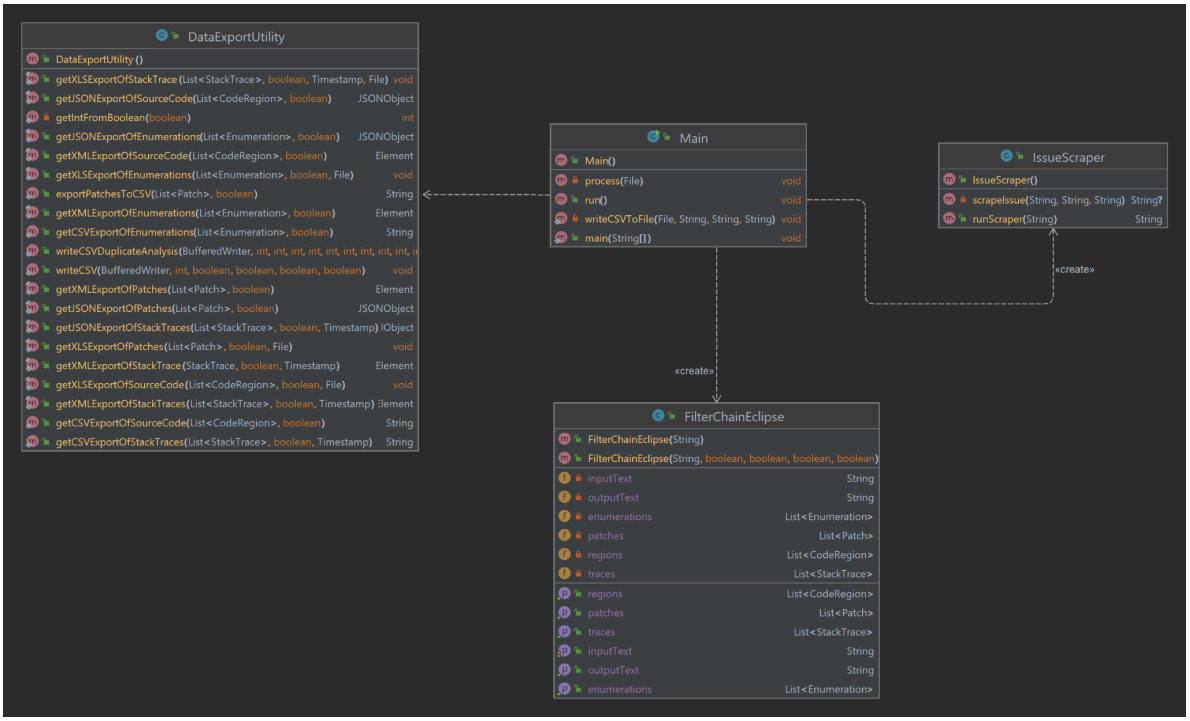

Figura 10: *Modifica attesa del Class Diagram (ridotto)*

Analizzando il Class Diagram **[Figura 10]**, possiamo immaginare che l’impatto della modifica non sia molto importante, e che dunque questa necessiti solo di ulteriori e specifici test per assicurarsi del corretto funzionamento della nuova componente. L’aggiunta, inoltre, avviene in un nuovo package, modificando dunque il package diagram **[Figura 11]**.

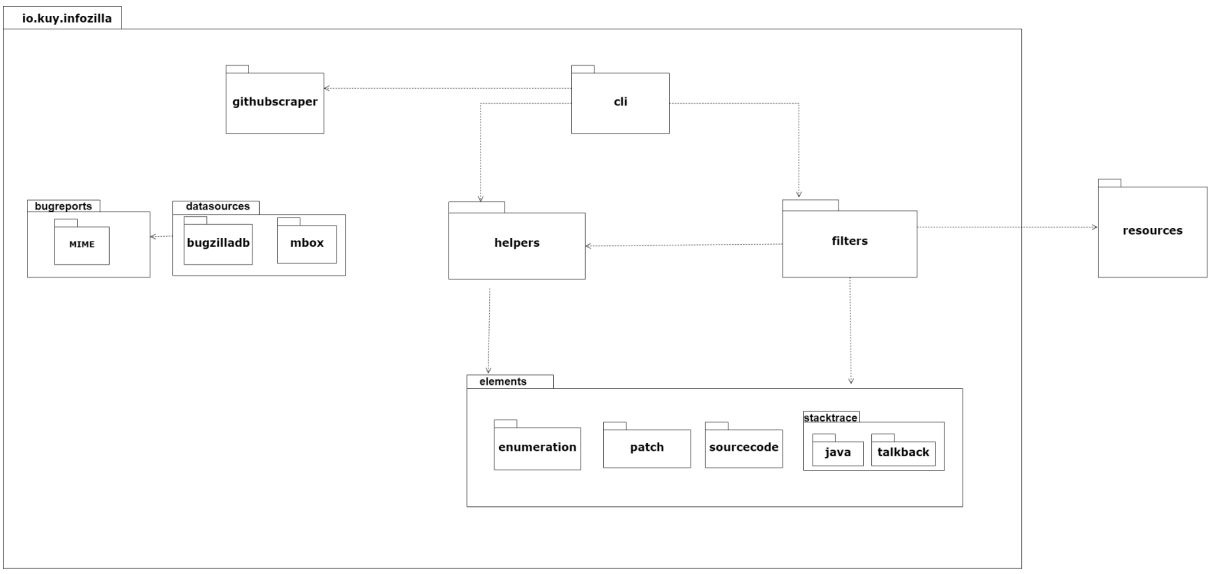

Figura 11: ***Modifica attesa del package diagram***

Candidate Impact Set - 3 Componenti

- Main: run
- *IssueScraper*: *runScraper*, *scrapeIssue*

5.2. Implementazione della modifica

La modifica non ha riscontrato particolari problematiche, se non per il parsing della stringa URL di input. Per far sì che quanti più casi possibili fossero coperti (stringa con http/https, stringa con www, stringa con github.com ecc.), vengono creati due metodi appositi per l'estrazione del nome della repository e del numero della issue. Viene utilizzata la libreria eGit per fare lo scraping della issue. Nel Main, e quindi nella command line, è stato deciso di rendere opzionale il file di input (locale) precedentemente obbligatorio per il sistema. Nonostante questa modifica abbia comunque un costo, ovvero la modifica della sintassi command line del sistema, con un parametro "-f" adesso divenuto necessario per dare in input un file locale, si ritiene che questa sia sensata. Non modificando questo parametro, infatti, sarebbe necessario inserire comunque un file locale, anche se si volesse lanciare solo un'analisi da remoto.
La classe creata comporta viene testata in unità, e nuovi test di sistema vengono introdotti.
Dopo l'implementazione, viene formulato l'Actual Impact Set, vengono calcolati il FPIS e il DIS e le metriche di precision e recall.

Actual Impact Set - 5 Componenti

- Main: run
- *IssueScraper*: *runScraper*, *scrapeIssue*, *getRepoIdFromUrl*, *getIssueIdFromUrl*

False Positive Impact Set - 0 Componenti

Discovered Impact Set - 2 Componenti

- IssueScraper: *getRepoIdFromUrl*, *getIssueIdFromUrl*

5.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 3/5 = 0.7.
Precision = (CIS ∩ AIS) ÷ CIS = 3/3 = 1.

6. Descrizione modifiche Change Request 6(CR_6)

Come descritto per la CR5, infozilla è in grado di eseguire scraping solo in locale. Pur realizzando l'implementazione della CR5, il procedimento di scraping da remoto resterebbe prettamente manuale. Seppure, infatti, l'utente non debba necessariamente scaricare in locale il file manualmente, deve comunque individuare tutte le issue che gli interessano ed inserire ogni link manualmente.
La modifica proposta punta ad automatizzare questo processo, fornendo all'utente la possibilità di inserire un URL di una repository github e una serie di parametri di filtraggio, che permettano di scaricare in maniera automatica tutte le issue di interesse.
In particolare, si vuole filtrare per:

- data di apertura, con parametri "opened before" e "opened after",
- data di chiusura, con parametri "closed before" e "closed after",
- assegnatario della issue,
- stato della issue, "open", "closed" oppure "all",
- Labels della issue, "bug", "enhancement" ecc.

Descrizione della Problematica:
Infozilla, anche a seguito dell'eventuale implementazione di CR5, non permette di analizzare tutte le issue di una repository remota, né di filtrare le issue sulle quali fare scraping. Ciò può rendere l'esperienza prettamente manuale e frustrante per l'utente

Modifiche Proposte:

- Aggiunta dello scraping delle issue di un'intera repository: aggiunta di un parametro della command line per permettere all'utente di inserire l'url di una repository sulla quale automatizzare lo scraping delle issue
- Aggiunta del filtering di issue remote: aggiunta di più parametri per permettere di filtrare le issue sulle quali fare scraping per data, assegnatario, stato e labels.

Benefici Previsti:

- Maggiore comfort di utilizzo per l'utente
- Automazione dei task di filtering e scraping di intere repository

6.1. Analisi d'impatto

La modifica comporta l'introduzione di un metodo all'interno della classe *IssueScraper* (creata in CR5) che permetta di fare lo scraping delle issue dell'intera repository passata in input, eventualmente filtrando per data, assignee, stato e labels.
Oltre a questo, si prevede una modifica della classe *Main*, introducendo i relativi parametri per l'url della repository e per le opzioni di filtraggio, oltre a una modifica del metodo *run*.
In questo caso, dunque, non viene realizzata la matrice di raggiungibilità, in quanto il metodo aggiunto viene richiamato solo dal metodo *run* della classe *Main*, ovvero il metodo principale di questa classe e, in generale, il primo richiamato quando viene lanciata un'esecuzione del tool.
Vengono dunque formulati Starting e Candidate Impact Set come segue:

Starting Impact Set - 2 Componenti

Sono state individuate le seguenti classi e metodi:

- Main: run
- IssueScraper: runRepoScraper

Candidate Impact Set - 2 Componenti

- Main: run
- IssueScraper: runRepoScrape

6.2. Implementazione della modifica

L'implementazione della modifica non ha comportato particolari problematiche, il metodo è stato aggiunto correttamente e, anche grazie al testing di unità implementato, si considera la modifica corretta. L'unica modifica non prevista è stata al metodo *getRepoIdFromUrl* della classe *IssueScraper*, che è stata adattata per gli URL del tipo <prefisso>github.com/nomeUtente/nomeRepository, mentre prima funzionava solo per gli url nel formato <prefisso>github.com/nomeUtente/nomeRepository/issue/<numero>.
Inoltre, tramite lo stato della classe, viene implementata l'opzione di utilizzare il proprio Github Access Token - tramite OAuth - per avere la possibilità di effettuare più richieste con l'api eGit; senza l'autenticazione, l'api permette di effettuare al più 60 richieste in un'ora, mentre con l'autenticazione si passa a 5000 richieste/ora.
Di seguito, l'Actual Impact Set e i FPIS, e DIS:

Actual Impact Set - 3 Componenti

- Main: *run*
- IssueScraper: *runRepoScraper*, *getRepoIdFromUrl*

False Positive Impact Set - 0 Componenti

Discovered Impact Set - 1 Componente

- IssueScraper: *getRepoIdFromUrl*

6.3. Metriche

Dopo aver eseguito le modifiche, sono stati calcolati i parametri di ReCall, Precision al fine di valutare la qualità dell'analisi d'impatto condotta.

ReCall = (CIS ∩ AIS) ÷ AIS = 2/3 = 0.67.
Precision = (CIS ∩ AIS) ÷ CIS = 2/2 = 1.