Laurea Magistrale in informatica
Università di Salerno

Modifications Testing and Regression Testing

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

https://github.com/GerardoFesta/infozilla

Sommario

1. Introduzione

L'introduzione delle modifiche CR5 e CR6 ha portato anche all'introduzione di nuovi test, sia di unità che di sistema, mentre si è scelto di testare le modifiche di CR4 solo a livello di sistema. Questo perché CR4 incide su una classe, *DataExportUtility*, che già in precedenza si era scelto di non testare, in quanto non è parte delle funzionalità core del progetto e si occupa solo dell'output dei dati. Oltre a questa classe, CR4 incide sul Main, classe che fa le veci della command line del tool, testata dunque tramite System Test.
Per quanto riguarda CR1, CR2 e CR3, invece, viene eseguito solo testing di regressione, in modo da verificare sia se la modifica ha portato una soluzione ai problemi individuati, sia se questa ha introdotto nuovi difetti.

2. Test di unità IssueScraper (CR5 e CR6)

La test suite di questa classe è suddivisa sulla base delle CR.

2.1. Test per CR5

Classi Testate: *IssueScraper*
Nomi classi di test: *IssueScraperTest*

Test:

testValidRepoAndIssueURL()

Input	**Oracolo**
url = https://github.com/gerardofesta/infozilla/issues/1	File "gerardofesta-infozilla-1Issue.txt" esiste

testInvalidURLFormats()

Input	**Oracolo**
lista di url non validi, ad es: www.github.com/gerardofesta	Ogni esecuzione della classe con ciascuno degli url lancia IOException

testAcceptedURLFormats()

Input	**Oracolo**
lista di url formattati validamente, ad es: www.github.com/gerardofesta/infozilla/issue/3	Ogni esecuzione della classe con ciascuno degli url crea correttamente il relativo file.

testGetClosedIssue()

Input	**Oracolo**
url di una issue chiusa (https://github.com/gerardofesta/infozilla/issues/1)	Il file "gerardofesta-infozilla-1Issue.txt" esiste

Questo insieme di test permette di raggiungere la code/branch coverage necessaria, quindi viene ritenuto sufficiente.

2.2. Test per CR6

Classi Testate: *IssueScraper*
Nomi classi di test: *IssueScraperTest*

Test:

testInvalidDateRange1()

Input	**Oracolo**
url repository corretto, opened_before= 2022-01-01 opened_after = 2023-01-01	L'esecuzione lancia IOException

testInvalidDateRange2()

Input	**Oracolo**
url repository corretto, closed_before= 2022-01-01 closed_after = 2023-01-01	L'esecuzione lancia IOException

testInvalidDateRange3()

Input	**Oracolo**
url repository corretto, closed_before = 2022-01-01 opened_after = 2023-01-01	L'esecuzione lancia IOException

testValidDateRange1()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) opened_before= 2023-01-01 opened_after = 2022-01-01	Non vengono creati file, ma non viene lanciata eccezione

testValidDateRange2()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) opened_before= 2023-09-19 opened_after = 2023-08-26	Vengono creati i file relativi a 4 issue

testValidDateRange3()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) closed_before= 2023-09-24 closed_after = 2023-08-25	Vengono creati i file relativi a 6 issue

testInvalidDateFormat()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) closed_before= 01/01/2023	L'esecuzione lancia IOException

testInvalidAssignee()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) assignee = abcdef	L'esecuzione non lancia eccezione, ma non crea file

testValidAssignee()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) assignee = gerardofesta	Viene scaricato almeno un file

testGetClosedIssues()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofestaa/infozilla) state = open	Viene creato almeno un file

testGetOpenIssues()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) state = closed	I path non sono nulli

testGetIssuesBySingleLabel()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) labels = {"bug"}	Vengono creati almeno i file delle issue 1, 10, 12, ma non della issue 2

testGetIssuesByMultipleLabels()

Input	**Oracolo**
url repository corretto (https://github.com/gerardofesta/infozilla) labels = {"bug", “enhancement”}	Vengono creati almeno i file delle issue 3, 4, 5, 9, 11, 13, 14, 10, 12, 1, ma non della issue 2

3. Test di sistema aggiornato

I parametri della classe Main, a seguito delle modifiche, sono stati aggiornati e sono state aggiunte nuove funzionalità. Dunque, oltre a effettuare regression testing di quei TC interessati dalle modifiche, vengono aggiunti ulteriori Casi di test. La tecnica utilizzata è sempre quella del Category Partition sui parametri di input, con Weak Equivalence Class Test come criterio di selezione dei casi di test.
Di seguito vengono riportati i parametri disponibili a riga di comando, le relative categorie e scelte.

In particolare, viene aggiunta la categoria “Present” a file-path, in quanto adesso non è un parametro obbligatorio e vengono adeguati i selettori che utilizzavano la property “fileOk” adeguatamente.

3.1. Parametri, categorie e scelte

A) *Parametri relativi alla command line*

With-source-code	
Categoria	Scelte
Value (WSCV)	1. True [if fileOk or repoUrlOk or iUrlOk] 2. False [if fileOk or repoUrlOk or iUrlOk] 3. Malformed [error]

with-lists	
Categoria	Scelte
Value (WLV)	1. True [if fileOk or repoUrlOk or iUrlOk] 2. False [if fileOk or repoUrlOk or iUrlOk] 3. Malformed [error]

with-patches	
Categoria	Scelte
Value (WPV)	1. True [if fileOk or repoUrlOk or iUrlOk] 2. False [if fileOk or repoUrlOk or iUrlOk] 3. Malformed [error]

with-stacktraces	
Categoria	Scelte
Value (WSTV)	1. True [if fileOk or repoUrlOk or iUrlOk] 2. False [if fileOk or repoUrlOk or iUrlOk] 3. Malformed [error]

file-path	
Categoria	Scelte
Present (FPP)	1. True [property filePresent] 2. False
Valid (FPV)	1. Valid [if filePresent] [property fileOk] 2. Invalid [if filePresent] [error]

Il parametro del formato di output è una nuova aggiunta, a seguito della CR4.

outputFormat	
Categoria	Scelte
FileType(OFFT)	1. None (Default: XML) [if fileOk or repoUrlOk or iUrlOk] 2. XML [if fileOk or repoUrlOk or iUrlOk] 3. CSV [if fileOk or repoUrlOk or iUrlOk] 4. XLS [if fileOk or repoUrlOk or iUrlOk] 5. JSON [if fileOk or repoUrlOk or iUrlOk] 6. Invalid [error]

IssueUrl	
Categoria	Scelte
Present(IUP)	1. Present [property iUrlPresent] 2. Not Present
Validity(IURLV)	1. Valid [if iUrlPresent][property iUrlOk] 2. Invalid [if iUrlPresent][error]

repoUrl (-a)	
Categoria	Scelte
Present(RUP)	1. Present [property repoUrlPresent] 2. Not Present
Validity(RUV)	1. Valid [if repoUrlPresent][property repoUrlOk] 2. Invalid [if repoUrlPresent][error]

Nel caso delle date, non teniamo in considerazione la possibilità che gli intervalli non siano compatible (opened-after > closed-before, opened-before < opened-after, closed-before < closed-after) in quanto questa è già stata testata in maniera esaustiva nel test di unità.

opened-before	
Categoria	Scelte
Present(OBP)	1. True [if repoUrlOk][property obpresent] 2. False
Validity(OBV)	1. Valid [if obpresent] 2. Invalid [if obpresent] [error]

opened-after	
Categoria	Scelte
Present(OAP)	1. True [if repoUrlOk][property oapresent] 2. False
Validity(OAV)	1. Valid [if oapresent] 2. Invalid [if oapresent] [error]

closed-before	
Categoria	Scelte
Present(CBP)	1. True [if repoUrlOk] [property cbpresent] 2. False
Validity(CBV)	1. Valid [if cbpresent] 2. Invalid [if cbpresent] [error]

closed-after	
Categoria	Scelte
Present(CAP)	1. True [if repoUrlOk] [property capresent] 2. False
Validity(CAV)	1. Valid [if capresent] 2. Invalid [if capresent] [error]

state	
Categoria	Scelte
Present(SP)	1. True [if repoUrlOk] [property statepresent] 2. False
Value(SV)	1. open [if statepresent] 2. closed [if statepresent] 3. all [if statepresent] 4. Invalid [if statepresent] [error]

assignee	
Categoria	Scelte
Present(AP)	1. present [if repoUrlOk] [property apPresent] 2. not present
Exists(AE)	1. exists [if apPresent] 2. does not exist [if apPresent]

labels	
Categoria	Scelte
Number(LN)	1. 0 2. 1 [property labelSet] 3. n [property labelSet]
Exists(LE)	1. exists [if repoUrlOk and labelSet] 2. does not exist [if repoUrlOk and labelSet]

B) ***Relativi al file/issue in input***

Qui si opera su un oggetto dell'ambiente. Ci sarebbero più categorie da porter selezionare, come la "validità" di ogni elemento rispetto alle regular-expression sulle quali questo subisce un match. Tuttavia, la casistica è piuttosto vasta e riteniamo di aver testato a sufficienza nei test di unità e di integrazione, dunque, selezioniamo come categoria solo la quantità.

Java Source Code Regions	
Categoria	Scelte
Quantità (JSCQ)	1. 0 [if fileOk] 2. 1 [if fileOk] 3. Many [if fileOk]

Patches	
Categoria	Scelte
Quantità (PQ)	1. 0 [if fileOk] 2. 1 [if fileOk] 3. Many [if fileOk]

Enumerations	
Categoria	Scelte
Quantità (EQ)	1. 0 [if fileOk] 2. 1 [if fileOk] 3. Many [if fileOk]

Java Stacktraces	
Categoria	Scelte
Quantità (JSTQ)	1. 0 [if fileOk] 2. 1 [if fileOk] 3. Many [if fileOk]

C) ***Relativi alla repository in input***

Qui si opera su un oggetto dell'ambiente, ovvero la repository che viene passata come oggetto del parametro -all. L'unica categoria che viene gestita è quella relativa al numero delle issue presenti nella repository. Seppur potrebbe avere senso controllare questa complessità anche in relazione alle date, in realtà la verifica è già stata effettuata a livello di unità. Inoltre, si tratterebbe di un caso specifico delle quantità di issue presenti; la selezione per data può dar luogo, infatti, a 0-1-n issues.

Repository	
Categoria	Scelte
NumIssue (RNI)	1. 0 [if repoUrlOk] 2. 1 [if repoUrlOk] 3. Many [if repoUrlOk]

3.2. WECT Test Cases

Il criterio di copertura scelto è il Weak Equivalence Class Testing, con l'aggiunta di alcuni casi che potrebbero rivelarsi particolari (l'assenza di tutto contemporaneamente con i flag true, ad esempio). Formuliamo dunque i seguenti casi di test:

Test Case ID	Combinazione	Oracolo
TC1	FPP1, RUP2, IUP2, FPV2	FAILURE
TC2	JSCQ1, PQ1, EQ1, JSTQ1, WSCV1, WLV1, WPV1, WSTV3, FPP1, FPV1, OFFT1, RUP2, IUP2	FAILURE
TC3	JSCQ1, PQ1, EQ1, JSTQ1, WSCV1, WLV1, WPV3, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	FAILURE
TC4	JSCQ1, PQ1, EQ1, JSTQ1, WSCV1, WLV3, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	FAILURE
TC5	JSCQ1, PQ1, EQ1, JSTQ1, WSCV3, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	FAILURE
TC6	JSCQ1, PQ1, EQ1, JSTQ1, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml di output vuoto
TC7	JSCQ3, PQ3, EQ3, JSTQ3, WSCV1, WLV1, WPV1, WSTV2, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml con molte (2) patch, molte (3) source code area, molte (3) enumerations, 0 stackTrace
TC8	JSCQ3, PQ3, EQ3, JSTQ3, WSCV1, WLV1, WPV2, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml con 0 patch, molte source code area, molte enumerations, molte (2) stacktrace
TC9	JSCQ3, PQ3, EQ3, JSTQ3, WSCV1, WLV2, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml con molte patch, molte source code area, 0 enumerations, molte (2) stacktrace
TC10	JSCQ3, PQ3, EQ3, JSTQ3, WSCV2, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml con molte patch, 0 source code area, molte enumerations, molte (2) stacktrace
TC11	JSCQ3, PQ3, EQ3, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	File xml con molte patch, mote source code area, molte enumerations, molte (2) stacktrace
TC12	JSCQ2, PQ2, EQ2, JSTQ2, WSCV1, WLV1, WPV1,	File xml con una patch,

	WSTV1, FPP1, FPV1, OFFT1, RUP2, IUP2	una source code area, una enumeration, una stacktrace
TC13	JSCQ3, PQ2, EQ2, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT3, RUP2, IUP2	File csv con una patch, più source code area, una enumeration, più stacktrace
TC14	JSCQ3, PQ2, EQ2, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT4, RUP2, IUP2	File xls con una patch, più source code area, una enumeration, più stacktrace
TC15	JSCQ3, PQ2, EQ2, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT5, RUP2, IUP2	File json con una patch, più source code area, una enumeration, più stacktrace
TC16	JSCQ3, PQ2, EQ2, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT2, RUP2, IUP2	File csv con una patch, più source code area, una enumeration, più stacktrace
TC17	JSCQ3, PQ2, EQ2, JSTQ3, WSCV1, WLV1, WPV1, WSTV1, FPP1, FPV1, OFFT6, RUP2, IUP2	FAILURE
TC18	FPP2, RUP2, IUP1, IURLV1, JSCQ1, PQ1, EQ2, JSTQ1, WSCV1, WLV1, WPV1, WSTV1, OFFT1	File xml con solo una enumerazione
TC19	FPP2, RUP2, IUP1, IURLV2	FAILURE
TC20	FPP2, IUP2, RUP1, RUV2	FAILURE
TC21	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP2, AP2, LN1	Vengono processati tutte le issue della repository (almeno 1-14)
TC22	FPP2, IUP2, RUP1, RUV1, RNI2, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP2, AP2, LN1	Viene processata la singola issue presente sulla repository
TC23	FPP2, IUP2, RUP1, RUV1, RNI1, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP2, AP2, LN1	Non vengono processate issue
TC24	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP1, OBV2, OAP2, CBP2, CAP2, SP2, AP2, LN1	FAILURE
TC25	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP1, OAV2, CBP2, CAP2, SP2, AP2, LN1	FAILURE
TC26	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP1, CBV2, CAP2, SP2, AP2,	FAILURE

	LN1	
TC27	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP1, CAV2, SP2, AP2, LN1	FAILURE
TC28	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP1, OBV1, OAP1, OAV1, CBP1, CBV1, CAP1, CAV1, SP2, AP2, LN1	Vengono processate le sole issue presenti nel time range
TC29	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP1, SV4, AP2, LN1	FAILURE
TC30	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP1, SV3, AP1, AE1, LN1	Vengono processate tutte le issue (status=All) assegnate all'assignee
TC31	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP1, SV2, AP2, LN2, LE1	Vengono processate solo le issue chiuse con la label fissata in input
TC32	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP1, SV1, AP2, LN3, LE1	Vengono processate solo le issue aperte contrassegnate con una delle labels fissate in input
TC33	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP1, SV1, AP2, LN2, LE2	Non vengono processate issue a causa della label inesistente
TC34	FPP2, IUP2, RUP1, RUV1, RNI3, WSCV1, WLV1, WPV1, WSTV1, OBP2, OAP2, CBP2, CAP2, SP2, AP1, AE2, LN1	Non vengono processate issue a causa dell'assegnatario inesistente

4. Test di regressione

4.1. Approccio

Il test di regressione viene eseguito prima in maniera separata per ogni CR, poi sul merge di tutte le modifiche.
Per ogni modifica vengono rieseguiti i test di unità delle classi modificate (ad es. CR1 e CR2 modificano FIlterPatches e PatchParser, dunque verranno eseguiti i test di unità delle due classi), i test di integrazione e i test di sistema, questi ultimi automatizzati con github action sulla repository.
Per le modifiche di CR4, verranno rieseguiti solo i test di sistema - e verranno eseguiti i test da TC13 a TC17, ovvero quelli introdotti specificamente per testare la modifica - per verificare che il funzionamento del Main non sia stato alterato.
Per le modifiche di CR5, che prevede l'introduzione di un nuovo modulo che ha interazione solo con la classe Main, oltre ai test di unità specificati nella sezione 1.1 del documento (che

però non fanno parte della regressione), verranno eseguiti i test case di sistema introdotti appositamente.
Per quanto riguarda CR6, invece, essendo una modifica fatta sulla base di CR5, verranno rieseguiti i test di CR5, e verranno eseguiti tutti i test di sistema, siccome vengono introdotti molti parametri nella classe Main.
Per ogni singola change request, non è necessario che tutti i test di sistema passino, perché alcuni dei test, in particolare da TC7 a TC12, falliscono a causa di una combinazione di fattori (le enumerazioni e le patch principalmente) che vengono risolti in CR diverse.
Tuttavia, ci aspettiamo che, nella fase di integrazione delle modifiche, questi test passino, al netto di qualche possibile modifica che potrebbe rendersi necessaria, ma che non incide sulla correttezza del test in sé.

4.2. Esito Test di Regressione su modifiche isolate

Di seguito viene dettagliato l'esito del test di regressione effettuato in presenza di una sola delle modifiche richieste dalle cinque CR. Questo è stato possibile perché la repository è stata utilizzata con un branch per ogni CR.
La maggior parte delle modifiche non ha creato né fatto emergere problemi significativi, ad eccezione della CR4.

Dopo avere eseguito il test di regressione sono state rilevate ulteriori problematiche per il filtro Enumeration, in particolare, abbiamo notato che nel caso di enumerazioni differenti mescolate, queste ultime venivano riconosciute all'interno della stessa enumerazione.

Esempio: A. Item 1 - Item 2 B. Item 3 C. Item 4

Per risolvere questa problematica abbiamo effettuato degli ulteriori controlli: nel caso in cui già siano stati stati trovati due simboli di uguale natura e successivamente ci si trovi in presenza di un simbolo diverso, allora viene impostata la fine dell'enumerazione; nel caso in cui, invece, il symbol count è inferiore a 2 e viene trovato subito un simbolo di differente natura, settiamo il symbol count a 0 ed lo start dell'enumerazione viene spostato al suddetto simbolo.

4.3. Esito Test di Regressione su integrazione branch

L'unione dei vari branch ha comportato la necessità di modificare alcuni oracoli. In particolare in:

- TC7, TC9 e TC12. Viene cambiata la sezione sugli snippet di codice per adattarli al riconoscimento della patch, modificando lo start e l'end, ovvero gli indici di inizio e fine della sezione di codice, in termini di numero di caratteri. La patch viene ora rimossa, quindi era presente una discrepanza sugli indici.
- TC8. Era presente un errore nell'oracolo, difatti erano presenti le patch che non dovevano essere riconosciute in quanto il parametro "with-patches" è impostato a false.
- TC11. Viene modificato il diff (patch) in quanto la formattazione era erronea, in particolare, index e parte di hunk nell'oracolo erano stati inseriti in modo erroneo (principalmente "*n*" e caratteri speciali).

È bene notare come queste non fossero le cause per le quali i test non passavano precedentemente. Infatti, guardando il documento "3_Test Incident", e in particolare gli incident STI2, STI3 e STI4, i problemi riguardavano principalmente l'individuazione delle patch, con hunks che non venivano riconosciuti correttamente e che portava ad errori anche sul fronte del source code.