Testiranje softvera

Vežbe - CodeCover
Alat za jedinično testiranje

Dražen Drašković, asistent
Elektrotehnički fakultet
Univerziteta u Beogradu

Namena

- Alat CodeCover namenjen je testiranju java programa metodama bele kutije:
- Pokrivenost iskaza - da li je izvršen određeni iskaz
- Pokrivenost odluka (grana) - da li je izvršavanje programa prošlo određenom granom
- Pokrivenost prostih uslova (u smislu da li je tokom izvršavanja programa u datoj tački ispunjen uslov koji je napisan u programu - ne i njegova negacija)
- Pokrivenost petlji - posmatrana petlja se smatra pokrivenom ako je izvršeno nula, jedna i više od jedne iteracije.

Primer izveštaja

Name	Amount
default package	1
package	1
class	2
method	11

Date	Name	Comment
December 14, 2009 2:33:40 AM CET	UNNAMED TESTCASE	This is a test case containing all coverage results cause no startTestCase methods where captured.

	Statement Coverage		Branch Coverage		Loop Coverage		Strict Condition Coverage	
default package	15 / 27	55 %	0 / 2	0 %	1 / 6	16 %	0 / 3	0 %
calc	15 / 27	55 %	0 / 2	0 %	1 / 6	16 %	0 / 3	0 %
Calculator	3 / 8	37 %	0 / 0	---	0 / 3	0 %	0 / 0	---
public void add(int n)	1 / 1	100 %	0 / 0	---	0 / 0	---	0 / 0	---
public void substract(int n)	0 / 1	0 %	0 / 0	---	0 / 0	---	0 / 0	---
public void multiply(int n)	0 / 0	---	0 / 0	---	0 / 0	---	0 / 0	---
public void divide(int n)	0 / 1	0 %	0 / 0	---	0 / 0	---	0 / 0	---
public void square(int n)	0 / 1	0 %	0 / 0	---	0 / 0	---	0 / 0	---
public void squareRoot(int n)	0 / 1	0 %	0 / 0	---	0 / 3	0 %	0 / 0	---
public void clear()	1 / 1	100 %	0 / 0	---	0 / 0	---	0 / 0	---
public void switchOn()	1 / 1	100 %	0 / 0	---	0 / 0	---	0 / 0	---
public void switchOff()	0 / 0	---	0 / 0	---	0 / 0	---	0 / 0	---
public int getResult()	0 / 1	0 %	0 / 0	---	0 / 0	---	0 / 0	---
Main	12 / 19	63 %	0 / 2	0 %	1 / 3	33 %	0 / 3	0 %
public static void main(String[] args) throws java.io.IOException	12 / 19	63 %	0 / 2	0 %	1 / 3	33 %	0 / 3	0 %

Primer izveštaja

Main.java

```
4  public class Main {
5
6      public static void main(String[] args) throws java.io.IOException {
7          Calculator calculator = new Calculator();
8          Scanner in = new Scanner(System.in);
22         while ( o == '+' || o == '-' ) {
23             System.out.println("Unesite novi broj");
24             n = in.nextInt();
25
26             if ( o == '+' )
27                calculator.add(n);
28              else
29                calculator.substract(n);
30             System.out.println("Rezultat = " + calculator.getResult());
31             System.out.println("Unesite operaciju");
32             o = in.next().charAt(0);
33         }
34
35         calculator.switchOff();
36         System.out.println("...Switch Off...");
37     }
38 }
```

- Može se kliknuti na ime metoda, da se dobije izveštaj na nivou pojedinih iskaza:
- Crveno je nepokriveno, zeleno potpuno pokriveno, a žuto delimično

Rad sa alatom

- Alat prvo mora da instrumentuje program.
  Reč je o izmeni izvornog koda da se ubace brojači koji će u toku izvršavanja programa pratiti pokrivenost pojedinih kriterijuma.
- codecover instrument --root-directory *JavaApp*/src --destination *JavaApp*/instrSrc --container *JavaApp*/test-session-container.xml --language java --charset UTF-8
- Za objašenje pojedinih opcija videti codecover_batch.html u doc folderu primera sa sajta **ccjunit.zip**.

Rad sa alatom

- Zatim se tako izmenjeni program prevede (javac)
- Na uobičajeni način (java) pokrene se izvršavanje programa.
- Po završetku izvršavanja kreiran je .clf (coverage log file) sa podacima o pokrivenim jedinicama koda tokom izvršavanja.
- Napomena: .clf dobija default ime gde je uključen datum i vreme pokretanja. Ako se želi drugačiji naziv fajla, program se pokreće sa:

  java -Dorg.codecover.coverage-log-file=fajl.clf
- Ovako smo zadali naziv fajl.clf
- *U primeru sa sajta, **cc.bat** sadrži komande za normalno pokretanje Calculator primera sa CodeCoverom, a **ccjunit.bat** pokreće testiranje Calculatora JUnit alatom sa CodeCoverom. U prvom slučaju CC izveštaj je u **instr1**, a u drugom u **instr2** folderu arhive.*

Rad sa alatom

- Analiziranje loga: podatke iz .clf fajla treba ubaciti u tzv. Test session container fajl (tcs.xml u primeru). On služi kao trajno skladište ovih podataka.
- codecover analyze --container *JavaApp*/test-session-container.xml --coverage-log *JavaApp*/instrSrc/log.clf --name TestSession1 --comment "The first test session"
- Za objašenje pojedinih opcija videti codecover_batch.html. U isti kontejner moguće je ubaciti podatke iz više .clf fajlova (više izvršavanja programa) i komandom merge dobiti objedinjenu sesiju (kumulativno sabiranje podataka o pokrivenosti)

Rad sa alatom

- Na kraju se iz podataka iz kontejnera može generisati izveštaj o određenoj test sesiji komandom:

codecover report --container *JavaApp*/test-session-container.xml --destination *JavaApp*/report/Report.html --session "TestSession1" --template CODECOVER_HOME/report-templates/HTML_Report_SingleFile.xml

- Za objašenje pojedinih opcija videti codecover_batch.html. Izveštaj će biti u fajlu Report.html.

CodeCover i JUnit

- Sve što je rečeno važi i u ovom slučaju. Instrumentovati treba aplikaciju koja se testira (a radi jednostavnosti mogu i test klase). Dakle pri prevođenju treba zadati junit-4.7.jar u classpath-u.
- Izvršavanje programa ide na uobičajen način za JUnit:
- java -cp .;..\junit-4.7.jar -Dorg.codecover.coverage-log-file=log2.clf org.junit.runner.JUnitCore calc.CalculatorTest
- U nastavku se zadaju CC analyze i report komande kako je objašeno.