Testing Java Programs with JUnit

Lect.dr. Paul POCATILU
Catedra de Informatică Economică, A.S.E. Bucureşti

Software testing is an activity whose costs are very high. Manual testing involves a lot of effort, measured in person per month. Using automated testing, with specific tools, this effort can be dramatically reduced and the costs related with testing can decrease.
Keywords: *software testing, automated testing, JUnit, extreme programming.*

Introducere

Testarea software este un proces foarte importat în ciclul de dezvoltare software. Prin realizarea unei testări corespunzătoare este îmbunătățită considerabil calitatea produselor software finale. Testarea software este un proces costisitor în ceea ce priveşte consumul de resurse: bani, timp şi forță de muncă. Procesul de testare software constă din următoarele activități:

- Planificarea testelor;
- Proiectarea testelor;
- Implementarea testelor;
- Executarea testelor;
- Evaluarea testelor.

Fiecare activitate generează ieşiri specifice care sunt intrări pentru activitățile următoare. În final vor rezulta rapoarte privind erorile descoperite şi alte documente specifice. Toate aceste documente vor fi utilizate de echipa de dezvoltare pentru identificarea cauzelor erorilor şi corectarea acestora.

După elaborarea planurilor de test, pe baza unor intrări specifice (buget, resurse, timp disponibil), următorul pas constă în analiza cerințelor şi identificarea obiectivelor testării pentru echipa specializată în această activitate.

Etapa de proiectare are în vedere în principal definirea şi proiectarea procedurilor de test. În această etapă sunt luate decizii privind testele care se vor realiza manual precum şi cele care vor fi automatizate.

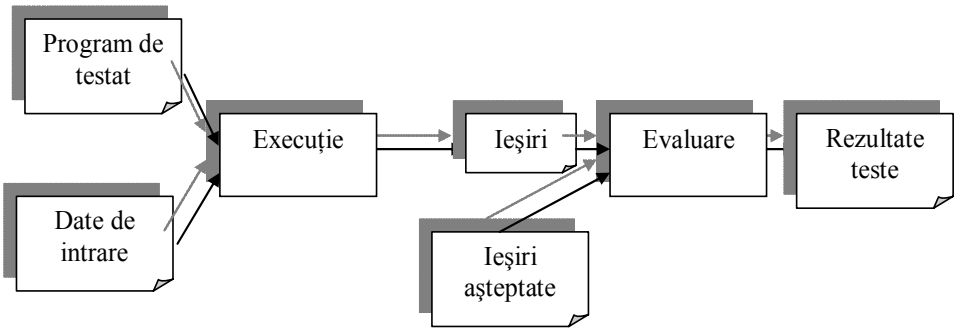

Fig. 1. Fazele de execuție şi evaluare pentru testarea de module

Cazurile de test şi procedurile de test sunt rezultatul fazei de implementare a testelor. Scripturile de test sunt scrise în diferite limbaje de programare precum Visual Basic, Java or C++. În această fază, scripturile de test pot fi reutilizate de la un test la altul.

Executarea testelor are ca intrări planul de test şi procedurile de test. După rularea testelor, rezultatele testării sunt evaluate de un oracol (figura 1).

Automatizarea testării software constă dintr-o serie de activități şi instrumente îmbinate pentru rularea programelor care sunt testate şi înregistrarea rezultatelor testelor.

Proiectul JUnit

În ultimii ani s-au dezvoltat o serie de metodologii de analiză şi proiectare pentru proiecte de dimensiuni reduse, precum: Adaptive Software Development (ASD), Extreme Programming (XP), SCRUM şi Crystal Clear. Majoritatea acestor metodologii sunt orientate obiect şi pun un accent important pe calitate şi pe testarea rezultatelor obținute.

Metodologia *Extreme Programming (XP)* are

la bază satisfacţia clientului şi lucrul în echipă. Metoda se utilizează pentru proiecte la care lucrează între 2 şi 10 oameni. În echipă sunt incluşi şi managerii şi clienţii. O cerinţă importantă este testabilitatea şi se are în vedere posibilitatea automatizării testelor de modul şi funcţionale. Obiectivul principal este livrarea de software de calitate la timp. În ceea ce priveşte procesul de testare, cazurile de test sunt scrise înaintea programelor care urmează a fi dezvoltate şi testate.

JUnit reprezintă un cadru de test pentru testarea regresivă a unit-urilor scrise în Java. Principalele obiecte utilizate sunt cele asociate cazurilor de test şi suitelor de teste.

Cazurile de test sunt asociate claselor care urmează a fi testate. Clasele de tip caz de test sunt derivate din clasa *TestCase*. Pentru fiecare metodă care urmează a fi testată este creată o metodă având numele metodei din clasa de test prefixat de numele *test*. În cadrul clasei asociată unui caz de test, există o instanţiere a unui obiect de tipul clasei de test.

Suita de teste include o serie de cazuri de test. Clasa de bază care va fi utilizată este *TestSuite*. Adăugarea de cazuri de test se realizează cu metoda `addTestSuite`, care are ca parametru o clasă de tip *TestCase*.

Pentru utilizarea JUnit după instalarea claselor şi bibliotecilor Java, în programele sursă sunt importate clasele din `junit.framework.*;`

Testarea clasei CosCumparaturi

Se considera listingurile din tabelul 1 asociate claselor *Produs* şi *CosCumparaturi*, utilizate în cadrul unei aplicaţii de comerţ electronic (e-DSI).

Tabelul 1 Listingul claselor care urmează a fi testate

Clasa Produs

```
package CC;
public class Produs
{
  public String Cod=null;
  public String Tip=null;
  public String Model=null;
  public String Producator=null;
  public double Pret=0.0;
  public String Descriere=null;
  public int Cant=0;

public Produs(String cod, String tip,
String producator, String model,
double pret, String descriere,int
cant)
  {
    Cod=cod;
    Tip=tip;
    Producator=producator;
    Model=model;
    Pret=pret;
    Cant=cant;
    Descriere=descriere;
  }

}
```

Clasa CosCumparaturi

```
package CC;
public class CosCumparaturi
{//numarul maxim de produse din cos
 public static int PMAX=20;
//vectorul de produse
 Produs produse[];//= new Produs();
 int n=0;//cite produse sint in cos
 //constructor
  public CosCumparaturi()
  {produse=new Produs[PMAX];
  }
  //obtine al i-lea produs din cos
  public Produs getProdus(int i)
  {return produse[i];
  }
  public boolean AdaugaProdus(Produs
p)
  {if(n<PMAX && p!=null)
   {
     produse[n] = p;
     n++;
     return true;
   }
   return false;
  }
  public int getNumarProduse()
  {return n;
  }
  public void setNumarProduseZero()
  {n=0;
  }
}
```

În tabelul 2 este prezentat listingul unui caz de test, în care sunt testate metodele clasei *CosCumparaturi*, pentru o serie de operaţii simple.

Tabelul 2 Listingul clasei asociată unui caz de test pentru clasa *CosCumparaturi*

```
package CC;
import junit.framework.*;
public class TestCosCumparaturi extends TestCase
{ private CosCumparaturi cosCumparaturiToTest = null;
  protected void setUp() throws Exception
  { super.setUp();
    cosCumparaturiToTest = new CosCumparaturi();
   }
  protected void tearDown() throws Exception
  { cosCumparaturiToTest = null;
    super.tearDown();
  }
  public void testAdaugaProdus()
  { Produs p = null;
    //produsele neinitializate nu sint adaugate
    assertEquals(cosCumparaturiToTest.AdaugaProdus(p),false);
    Produs p1=new Produs("ABB1", "FOTO", "PoL","AT200",30000.00, "Un pro-
dus",5);
    //s-a adaugat un produs cu success
    assertEquals(cosCumparaturiToTest.AdaugaProdus(p1),true);
  }
  public void testCosCumparaturi()
  {cosCumparaturiToTest = new CosCumparaturi();
   //produse nu este null si are lungimea PMAX
    assertNotNull(cosCumparaturiToTest.produse);
    assertEquals(cosCumparaturiToTest.produse.length,
          cosCumparaturiToTest.PMAX);
  }
  public void testGetNumarProduse()
  { //initial numarul de produse este zero
    assertEquals(cosCumparaturiToTest.getNumarProduse(),0);
    Produs p=new Produs("ABB1", "FOTO", "PoL","AT200",30000.00, "Un pro-
dus",5);
    cosCumparaturiToTest.AdaugaProdus(p);
    //dupa adaugarea unui produs, numarul acestora este 1
    assertEquals(cosCumparaturiToTest.getNumarProduse(),1);
  }

  public void testGetProdus()
  { //nu exista produs pe pozitia 0 (prima pozitie)
    Produs p = cosCumparaturiToTest.getProdus(0);
    assertEquals(p,null);
    Produs p1=new Produs("ABB1", "FOTO", "PoL","AT200",30000.00, "Un pro-
dus",5);
    cosCumparaturiToTest.AdaugaProdus(p1);
    Produs p2 = cosCumparaturiToTest.getProdus(0);
    //produsul returnat nu este null si are aceeasi referinta cu p1
    assertNotNull(p2);
    assertEquals(p2,p1);
  }
  public void testSetNumarProduseZero()
  {
    cosCumparaturiToTest.setNumarProduseZero();
    assertEquals(cosCumparaturiToTest.getNumarProduse(),0);
  }
}
```

În figura 2 este prezentată diagrama de clase generată de mediul de dezvoltare JBuilder X. asociată cazului de test *TestCosCumparaturi*,

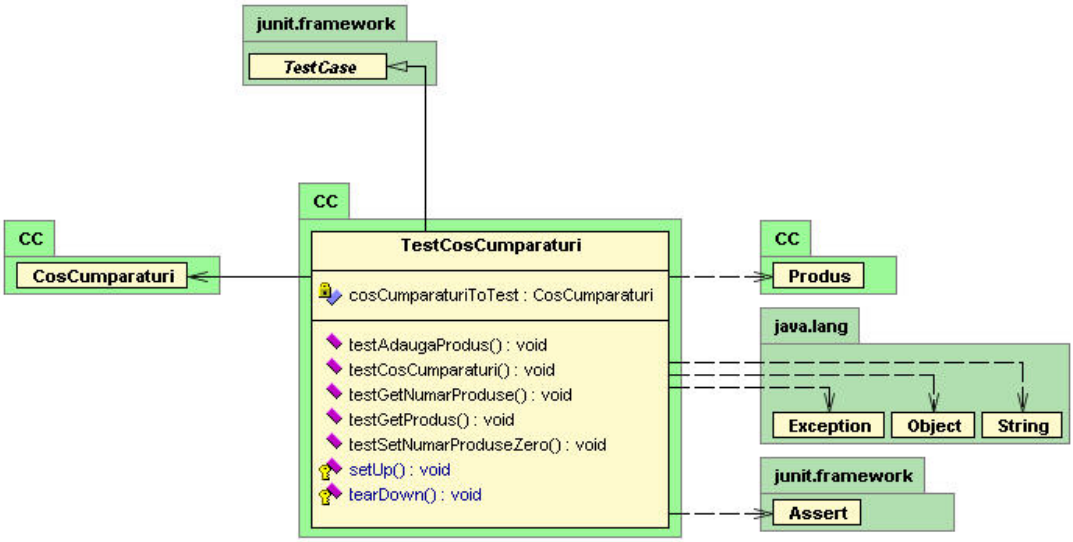

Fig. 2. Diagrama claselor de test asociate proiectului

Suita de teste generată utilizează cazul de test *TestCosCumparaturi*, rulând toate metodele acestuia, metode care sunt prefixate de *test** (tabelul 3).

Pentru execuția suitei de test s-a ales interfața grafică furnizată de biblioteca JUnit. Se observă în figura 3 execuția cu succes a testelor asociate clasei *CosCumparaturi*.

Tabelul 3. Listingul clasei *AllTests* asociată suitei de teste

```
package CC;
import junit.framework.*;
public class AllTests extends TestCase
{ public AllTests(String s)
  {super(s);
  }
  public static Test suite()
  { TestSuite suite = new TestSuite();
    suite.addTestSuite(CC.TestCosCumparaturi.class);
    return suite;
  }
}
```

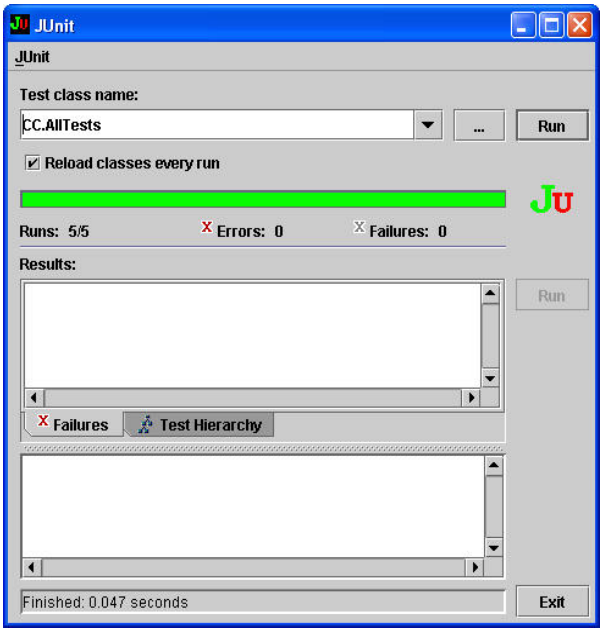

Fig. 3. Interfața grafică după rularea cu succes a suitei de teste

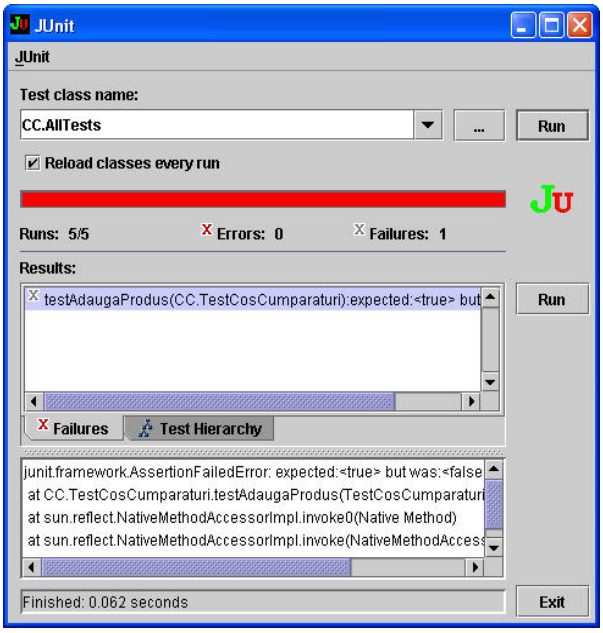

Fig. 4. Interfața grafică după rularea eşuată a suitei de test

În figura 4 este prezentată interfața grafică după rularea suitei de test, după ce în prealabil în metoda `testAdaugaProdus` linia de cod:

```
assertEquals(cosCumparaturiToTest.AdaugaProdus(p1),true);
```

a fost înlocuită cu:

```
assertEquals(cosCumparaturiToTest.AdaugaProdus(p1),false);
```

Există posibilitatea de a rula suitele de test şi în mod text, în cazul în care resursele sistemului sunt limitate.

Concluzii

Automatizarea testării programelor orientate obiect se realizează facil la nivelul funcțional utilizând bibliotecile dezvoltate pentru diferite limbaje de programare orientate obiect. Automatizarea testării conduce la creşterea eficienței acestui proces, costurile asociate reducându-se. JUnit reprezintă o modalitate de testare automată a programelor Java la nivel de unit, fără un efort prea mare. În aceeaşi manieră au fost dezvoltate medii de testare şi pentru aplicațiile Microsoft .NET.

Bibliografie

[DIAS00] Dias, Paulo, *Lightweight methodologies*, INESC, 2000

[DUST99] Dustin, Elfriede, Rashka, Jeff, Paul, John, *Automated Software Testing*, Addison Wesley, 1999

[EFTI02] Eftimescu, Despina, Ilioiu, Alexandru, *Testarea automată*, Net Report, Aprilie 2002, pp. 22-26

[KORE90] Korel, Bogdan, *Automated Software Test Data Generation*, IEEE Transactions on Software Engineering, vol. 16, No. 8, August 1990, pp. 870-879.

[IVAN99] Ivan, Ion, Pocatilu, Paul *Testarea software orientat obiect*, Editura, Bucureşti, 1999

[POCA02] Pocatilu, Paul, *Automated Software Testing Process*, în Economy Informatics, vol. II, nr. 1, 2002, pp. 97-99

[POCA04] Pocatilu, *Paul Costurile testării software*, Editura ASE, 2004

[TANA03] Tanasă, Ştefan, Olaru, Cristian, Andrei, Ştefan, *Java de la 0 la expert*, Editura Polirom, Iaşi, 2003

www.junit.org

www.extremeprogramming.org