

Laborator 2.2

1. Code coverage

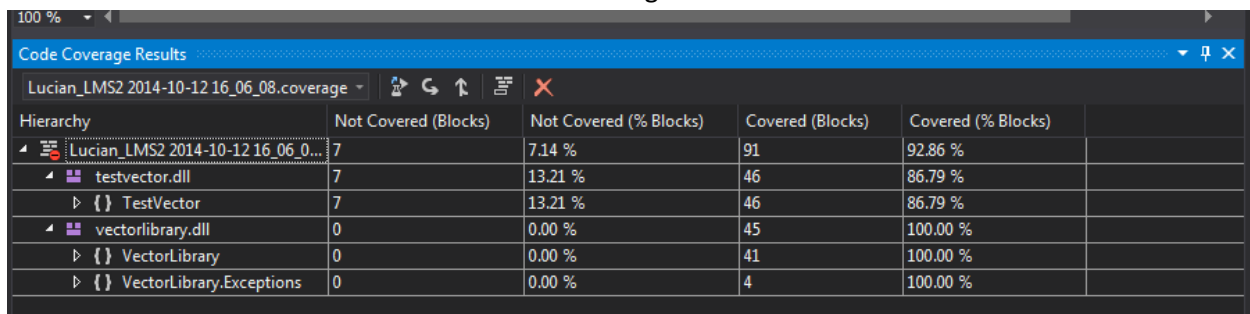
La o rulare de teste se poate pune intrebarea: cat din codul scris a fost acoperit de testele rulate = code coverage?

Definitie Code coverage: http://en.wikipedia.org/wiki/Code_coverage

Lucru cu code coverage: demo laborator

Secventa de lucru pentru Visual Studio 2017 Enterprise Edition (resursa: <http://msdn.microsoft.com/en-us/library/dd537628.aspx>): Test -> Analyze code coverage -> "All tests" (sau "Selected tests").

Rezultatele se vor vedea in fereastra de "Code coverage results":



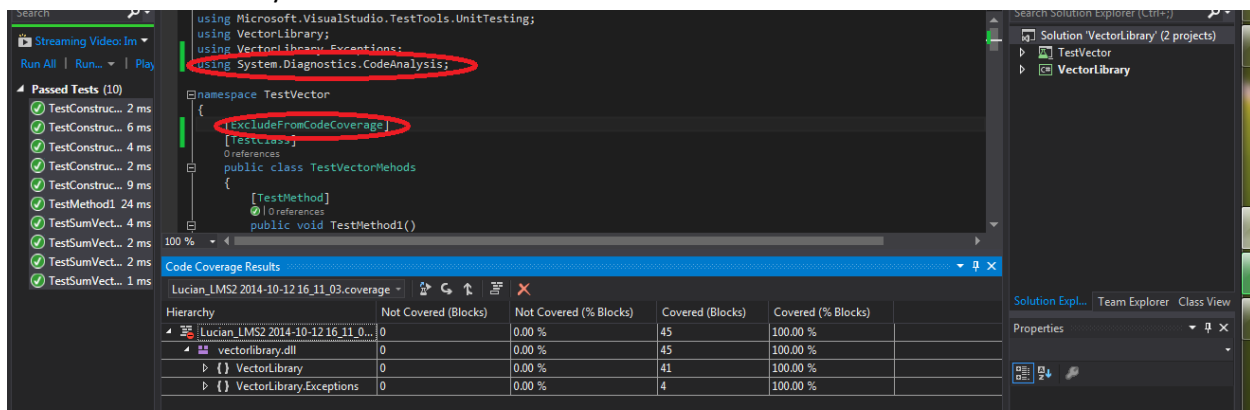
| Hierarchy | Not Covered (Blocks) | Not Covered (% Blocks) | Covered (Blocks) | Covered (% Blocks) |
|--|----------------------|------------------------|------------------|--------------------|
| Lucian_LMS2 2014-10-12 16_06_08.coverage | 7 | 7.14 % | 91 | 92.86 % |
| testvector.dll | 7 | 13.21 % | 46 | 86.79 % |
| TestVector | 7 | 13.21 % | 46 | 86.79 % |
| vectorlibrary.dll | 0 | 0.00 % | 45 | 100.00 % |
| VectorLibrary | 0 | 0.00 % | 41 | 100.00 % |
| VectorLibrary.Exceptions | 0 | 0.00 % | 4 | 100.00 % |

Figura 1. Code coverage, incluzand proiectul de testare

Se observa ca si proiectul care contine testele unitare este masurat din punct de vedere al acoperirii cu teste, ceea ce nu e de dorit (testarea vizeaza functionalitatile din biblioteca vectorlibrary, nu testarea in sine) .

Pentru eliminarea proiectului de testare de la code coverage se adauga atributul

ExcludeFromCodeCoverage la clasele de testare. Se observa ca acum code coverage se refera strict la biblioteca vectorlibrary:



| Hierarchy | Not Covered (Blocks) | Not Covered (% Blocks) | Covered (Blocks) | Covered (% Blocks) |
|--|----------------------|------------------------|------------------|--------------------|
| Lucian_LMS2 2014-10-12 16_11_03.coverage | 0 | 0.00 % | 45 | 100.00 % |
| vectorlibrary.dll | 0 | 0.00 % | 45 | 100.00 % |
| VectorLibrary | 0 | 0.00 % | 41 | 100.00 % |
| VectorLibrary.Exceptions | 0 | 0.00 % | 4 | 100.00 % |

Figura 2. Code coverage fara proiectul de testare

Alternativa: utilizarea si particularizarea unui fisier de configurare, conform

<http://msdn.microsoft.com/en-us/library/jj159530.aspx>. Fisierul poate fi descarcat de la

<http://msdn.microsoft.com/en-us/library/jj159530.aspx#sample>.

2. Testarea pentru verificarea codului

Variante:

- TDD = scrierea de teste in paralel cu codul propriu-zis
- Unit testing: cazul in care un programator scrie cod pentru testarea de unitati individuale (metoda)
Rezultat: se poate detecta prezenta erorilor, ***dar nu se demonstreaza astfel si absenta lor totala!***

Cum testezi?

Principiul **Right -BICEP**:

Right: Sunt solutiile date de program **corecte**? (corect = right)

B —sunt toate conditiile de **frontiera** corecte? (frontiera = boundary)

I — poti verifica relatiile **inverse**?

C — poti face **verificare incrucisata** folosind alte metode? (cross check)

E — poti simula aparitia conditiilor de **eroare**?

P — sunt performantele in limite rezonabile?

Sunt solutiile date de program corecte?

Rezultatele se verifica pe baza specificatiilor

Daca specificatiile sunt vagi/incomplete, atunci se poate merge pe niste simulari rezonabile; acestea sunt definite impreuna cu clientul.

Datele de testare pot fi puse in fisiere simple; datele continute pot referi partea de BICEP anterioara

Exemplu pentru testarea unei metode ce face calculul maximului unui sir de numere:

#

Simple tests:

#

9 7 8 9

9 9 8 7

9 9 8 9

#

Negative number tests:

#

-7 -7 -8 -9

-7 -8 -7 -8

-7 -9 -7 -8

#

Mixture:

#

7 -9 -7 -8 7 6 4

9 -1 0 9 -7 4

#

```
# Boundary conditions:
#
1 1
0 0
2147483647 2147483647
-2147483648 -2147483648
```

Sunt toate conditiile de frontiera corecte?

Exemple:

- Pentru vectorul {8,9,7} se verifica daca maximul este mutat pe diferite pozitii; de regula, la “capete” apar erori.
- Numele de fisier sunt cu intrari inconsistente: !*W:X\&Gi/w_>g/h#WQ@
- Date cu format gresit: adrese de email incorecte: a@yahoo.
- Valori nule sau care lipsesc
- Valori cu depasiri: varsta de persoana ≤ 0 , varsta > 300 , intrari cu lungimi prea mari (nume de utilizator, parola)
- Elemente care apar in liste si sunt duplicate, dar nu ar trebui
- Elemente care nu sunt dispuse in ordine, dar procesul depinde de o ordonare buna (pachete, documente care se tiparesc)

Verificarea relatiilor inverse

Exemplu: daca se efectueaza un calcul matematic, uneori se poate face uneori si verificarea solutiei: pentru calculul radicalului dintr-un numar, verificarea consta in ridicarea numarului la patrat si compararea cu argumentul initial. Atentie la metodele folosite pentru calcularea inverse: daca metodele sunt dezvoltate de acelasi programator, erorile se pot masca una pe cealalta.

In cazul calculului cu numere in virgule mobile, se va lua in considerare eroare de reprezentare datorata stocarii numerelor (ex. $1.0-0.9-0.1$ nu este 0.0)

Poti face verificare incrucisata folosind alte metode?

Este o generalizare a metodei precedente:

```
double root1 = MyMath.SquareRoot(number);
```

```
double root2 = Math.Sqrt(number);
```

```
Assert.AreEqual(root1, root2);
```

Pentru simularea de imprumut de carti: numarul de carti imprumutate + numarul de carti de pe raft = numarul total de carti (se poate face grupare pe titluri).

De preferat: metodele folosite ar trebui sa fie exterioare sistemului programat.

Poti simula aparitia conditiilor de eroare?

Pot aparea erori care sa fie externe sistemului. Intrebarea este cum se comporta codul propriu in asemenea situatii?

- Epuizarea cantitatii de memorie
- Umplerea harddisk-ului
- Erori pe retea
- Permisuni insuficiente
- Incarcarea sistemului
- Ecrane cu rezolutie prea mare/mica

Caracteristici de performanta

Se masoara tendintele: comportamentul in functie de dimensiunea intrarii, complexitatea problemei etc.

Este baza pentru masuratorile de performanta care trebuie facute la modificari majore in aplicatie.

Activitate de laborator:

1. De instalat pe masina virtuala NUnit si TestDriven.NET, Personal version (folositi pentru inregistrare, la numele institutiei: Transilvania University of Brasov)
2. De folosit test driven development cu NUnit si TestDriven.NET pentru problema 1 din laboratorul precedent.
3. Cum se pot rula teste in paralel in NUnit?
4. De citit cap 4 si 5 din pragmatic-unit-testing-in-c-with-nunit-2nd-edition