CURS 02B.
TESTARE BLACK-BOX

Verificarea şi validarea sistemelor soft
[06 Martie 2018]

Lector dr. Camelia Chisăliţă-Creţu
Universitatea Babeş-Bolyai

Conţinut

- Criterii de testare
- Testare Black-Box
  - Definiţie. Caracteristici.
  - Clasificare. Tehnici de testare black-box
  - Partiţionarea în clase de echivalenţă. Exemple
  - Analiza valorilor limită. Exemple
  - Partiţionarea în clase de echivalenţă vs Analiza valorilor limită
  - Avantaje şi dezavantaje
- Pentru examen...
- Bibliografie

CRITERII DE TESTARE

Criteriu de testare. Clasificare

Tehnici de testare asociate

Criteriu de testare. Clasificare

- criteriu de testare
  - ansamblu de condiţii prin care se stabilesc cazurile de testare;
- clasificare
  - criteriul cutiei negre (testare Black-Box);
  - criteriul cutiei transparente (testare White-Box);
  - criteriul statistic.

Tehnici de testare asociate

- criteriul cutiei negre (testare Black-Box) – testare funcţională:
  - Partiţionarea în clase de echivalenţă;
  - Analiza valorilor limită;
  - Tabele de decizie, Grafe de tranziţie a stărilor, Scenarii de execuţie ale cazurilor de utilizare, etc.;
- criteriul cutiei transparente (testare White-Box) – testare structurală:
  - Acoperirea fluxului de control (e.g., instrucţiuni, ramificaţii, decizii, condiţii, bucle, drumuri);
  - Acoperirea fluxului de date;
- criteriul statistic:
  - generarea aleatoare de date de test pe baza unor modele;
  - experienţa testerului.

TESTARE BLACK-BOX

Definiţie. Caracteristici. Tehnici de testare black-box
Partiţionarea în clase de echivalenţă. Exemple
Analiza valorilor limită. Exemple
Partiţionarea în clase de echivalenţă vs Analiza valorilor limită
Avantaje şi dezavantaje

Definiţie. Caracteristici

- **criteriul cutiei negre** (*engl.* **black-box testing**, **data driven testing**, **input/output driven testing**):
  - testare funcţională;
  - datele de intrare se aleg pe baza specificaţiei problemei, programul fiind văzut ca o cutie neagră;
  - nu avem acces la structura internă a programului, i.e., codul sursă;
  - permite identificarea situaţiilor în care programul nu funcţionează conform specificaţiilor.

Tehnici de testare black-box

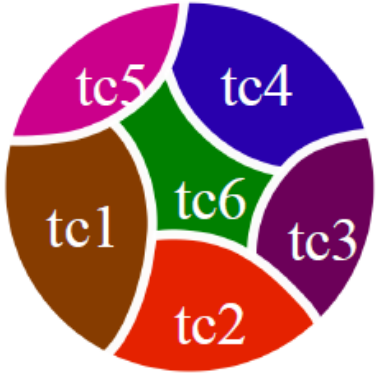
- tehnici de proiectare a cazurilor de testare bazate pe criteriul black-box:
  1. **Partiţionarea în clase de echivalenţă;**
  2. **Analiza valorilor limită;**
  3. Tabele de decizie;
  4. Grafe de tranziţie a stărilor;
  5. Scenarii de execuţie bazate pe cazurile de utilizare;
  6. *alte tehnici*.

Partiţionarea în clase de echivalenţă. Motivaţie

- în general, testarea exhausivă nu este posibil de realizat, e.g.:
  - există un set consistent de date de intrare sau domeniul de valori testat este infinit;
  - restricţii (e.g., timp, buget).

- partiţionarea în clase de echivalenţă (*engl.* Equivalence Class Partitioning, ECP) este eficientă pentru reducerea numărului de cazuri de testare care trebuie proiectate;

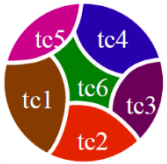
- clase de echivalenţă disjuncte:
  - se evită redundanţa cazurilor de testare;
- cazuri de testare:
  - se alege un singur element din fiecare clasă de echivalenţă;



Partiţionarea în clase de echivalenţă. Definiţie

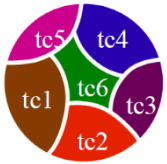
- **clasă de echivalenţă** (*engl.* **equivalence class, EC**):
  - mulţimea datelor de intrare/ieșire pentru care programul are comportament similar [Myers2004];

- **partiţionarea în clase de echivalenţă** (*engl.* **equivalence class partitioning, ECP**):
  - împărţirea (divizarea) domeniului datelor de intrare/ieșire în EC, astfel încât, dacă programul va rula corect pentru o valoare dintr-o EC, atunci va rula corect pentru orice valoare din acea EC.



ECP. Identificarea ECs

- **se identifică** clasele de echivalenţă pe baza condiţiilor de intrare/ieşire;
- **se clasifică** clasele de echivalență în:
  - valide – formate din datele de intrare/ieșire valide pentru program;
  - non-valide – formate din datele de intrare/ieșire eronate, corespunzătoare tuturor celorlalte stări ale condiției de intrare/ieșire.



ECP. Exemple (1)

- Se consideră un formular de înscriere la un concurs. Pentru data naşterii se introduce ziua, luna şi anul.
- Identificaţi clasele de echivalenţă corespunzătoare câmpului lună calendaristică (pentru data naşterii). Domeniul de valori este [1, 12].

Abordare primară: *un număr >=1 şi <=12;* 1 EC validă: **EC1:** D1 = [1, 12]; 2 EC non-valide: **EC2:** D2= {luna<1} = (-∞, 1); **EC3:** D3= {luna>12} = (12, +∞); **EC4:** D4= simboluri alfanumerice.	**Abordare secundară:** *numărul de ordine al unei luni dintr-un an;* 1 EC validă: D1 = {1, 2, …, 12}; 1 EC non-validă: D2 = mulţimea formată din orice combinaţie de simboluri alfanumerice diferită de valorile 1, 2, ..., 12.

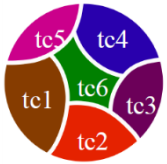
ECP. Exemple (2)

- Pentru un cont bancar se consideră următoarea ofertă de dobânzi:
  - 0,50% până la 1000 Euro depunere în cont;
  - 1,00% până la 2000 Euro depunere în cont, dar mai mult de 1000 Euro;
  - 1,50% pentru depuneri mai mari decât 2000 Euro;

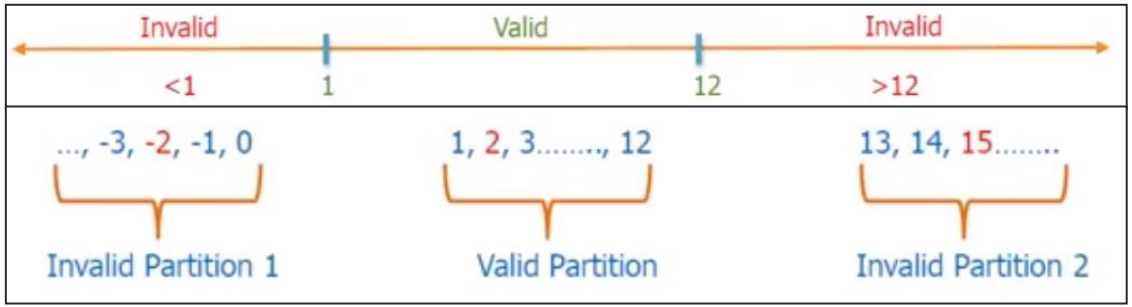
- **Care sunt clasele de echivalenţă valide pentru un cont? Dar clasele de echivalenţă non-valide?**
  - Clase de echivalenţă valide:
    - **EC1:** 0,00 Euro – 1000,00 Euro;
    - **EC2:** 1000,01 Euro – 2000,00 Euro;
    - **EC3:** >= 2000,01 Euro.
  - Clase de echivalenţă non-valide:
    - **EC4:** < 0,00 Euro;
    - **EC5:** > valoarea maximă admisă pentru un cont;
    - **EC6:** caractere din alfabet.

ECP. Proiectarea cazurilor de testare. Algoritm

- Algoritm de proiectare a cazurilor de testare:
  1. se asociază un identificator unic fiecărei clase de echivalență (e.g., EC1, EC2, etc.);
  2. `câttimp` *(nu au fost descrise cazuri de testare pentru toate clasele de echivalență valide/non-valide):*
     - `scrie` *(un nou caz de testare care corespunde la cât* ***mai multe clase de echivalență valide*** *încă neacoperite)*;
     - `scrie` *(un nou caz de testare care corespunde* ***doar uneia dintre clasele de echivalență de non-valide*** *încă neacoperite).*



ECP. Selectarea datelor de test. Exemple (1)



- **ECs identificate:**
  - 1 EC validă, **EC1: D1 =[1, 12];**
  - 3 EC non-valide, **EC2: D2= {luna| luna<1} = (-∞, 1), EC3: D3= {luna| luna>12} = (12, +∞), EC4: D4 = simboluri alfanumerice;**
- **Cazuri de testare proiectate:**
  - 1 EC validă ==> 1 caz de testare valid, e.g., **TC01: luna = 2;**
  - 3 EC non-valide ==> 3 cazuri de testare non-valide, e.g., **TC02: luna =-2, TC03: luna = 15, TC04: luna = ”%L10”;**

- **Din fiecare EC de intrare identificată se alege o singură valoare. ECP consideră că fiecare EC tratează în manieră similară toate valorile din EC.**

ECP. Selectarea datelor de test. Exemple (2)

- **ECs identificate:**
- **3 ECs valide:**
  - **EC1:** 0,00 Euro – 1000,00 Euro;
  - **EC2:** 1000,01 Euro – 2000,00 Euro;
  - **EC3:** >= 2000,01 Euro.
- **3 ECs non-valide:**
  - **EC4:** < 0,00 Euro;
  - **EC5:** > valoarea maximă admisă pentru un cont;
  - **EC6:** caractere din alfabet.

- **Cazuri de testare proiectate:**
  - **3 ECs valide ==>** 3 cazuri de testare valide,e.g.:
    - **TC01:** amount= 678,99;
    - **TC02:** amount = 1742,81;
    - **TC03:** amount = 5213,00;
  - **3 ECs non-valide ==>** 3 cazuri de testare non-valide, i.e., câte un TC care corespunde fiecărei EC non-valide identificate, e.g.:
    - **TC04:** amount = -0,79;
    - **TC05:** amount = 1234567890,123456;
    - **TC06:** amount = #12a.

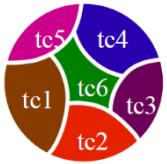
ECP. Proiectarea cazurilor de testare. Reguli (1)

1. **dacă o condiție de intrare precizează apartenența la un interval de valori [a,b]:**

- ==> 1 EC validă, 2 EC non-valide;
  - E.g.: luna, o valoare intervalul [1, 12];

2. **dacă o condiție de intrare precizează o mulțime de valori de intrare:**

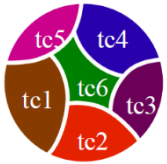
- ==> 1 EC validă pentru fiecare valoare, 1 EC non-validă;
  - E.g.: `tip curs ∈ CourseType = {opțional, obligatoriu, facultativ};`
  - 1 EC validă pentru fiecare element din `CourseType`:
    - **EC1:** {`opțional`},
    - **EC2:** {`obligatoriu`},
    - **EC3:** {`facultativ`} ===> 3 ECs valide;
  - 1 EC non-validă:
    - **EC4:** M= {e | $e \notin$ `CourseType`};



ECP. Proiectarea cazurilor de testare. Reguli (2)

3. **dacă o condiție de intrare precizează numărul de valori:**

- ==> 1 EC validă, 2 EC non-valide;
  - E.g.: “de la 1 și 5 studenți”;
  - 1 EC validă:
    - **EC1:** D=[1,5];
  - 2 EC non-valide:
    - **EC2:** nici un student;
    - **EC3:** mai mult de 5 studenți;

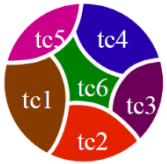


ECP. Proiectarea cazurilor de testare. Reguli (3)

4. **dacă o condiție de intrare precizează o situație de tipul “must be”:**

- ==> 1 EC validă, 1 EC non-validă.
  - E.g.,: “primul caracter din parolă trebuie să fie un simbol numeric”;
  - 1 EC validă:
    - **EC1:** primul caracter este un simbol numeric;
  - 1 EC non-validă:
    - **EC2:** primul caracter nu este un simbol numeric.

Dacă există argumente că programul nu tratează similar toate elementele dintr-o EC, atunci EC se împarte în EC mai mici.



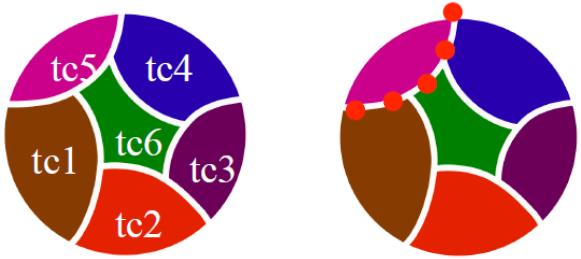
Analiza valorilor limită. Motivaţie

- ECP presupune că programul are un comportament similar pentru toate valorile dintr-o EC;
- ECP nu garantează că programul este testat şi la limitele ECs identificate;
- există greşeli de programare tipice care apar la limita ECs identificate;
  - e.g., pentru x≥3

```
if (x>3) y++;   //bug
if (x>=3) y++;
```

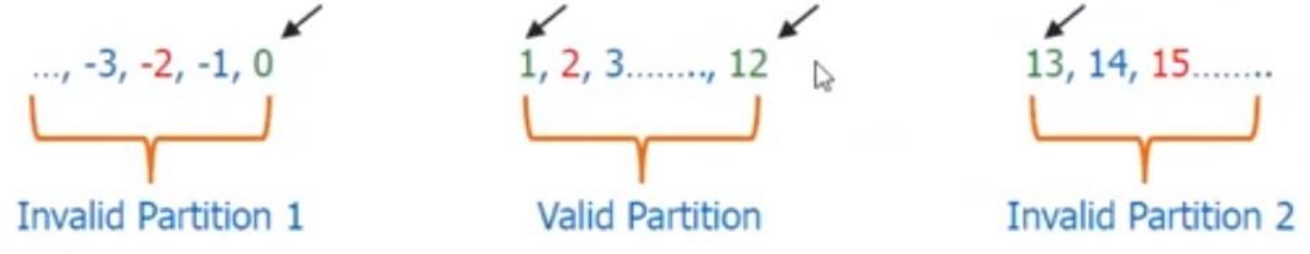
  - [ECP]: pentru EC1: [3, MaxInt] se alege TC01: x=4, dar TC01 nu surprinde bug-ul de implementare;

- analiza valorilor limită investighează posibilele bug-uri existente la limita dintre ECs identificate;
  - [BVA]: pentru EC1: [3, MaxInt] se alege TC02: x=3;



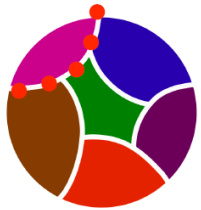
Analiza valorilor limită. Definiţie

- **analiza valorilor limită** (*engl.* **boundary value analysis, BVA**) [Myers2004]:
  - testarea realizată prin alegerea datelor de test pe baza limitelor EC de intrare/ieşire;



Condiţii BVA. Identificare

1. se identifică limitele tuturor ECs valide de intrare/ieşire;
2. se scriu condiţii BVA pentru fiecare limită a fiecărei EC identificate, astfel încât:
   - valoarea să fie sub (mai mică), e.g., x < 2;
   - valoarea să fie pe (egală), e.g., x = 2;
   - valoarea să fie deasupra (mai mare), e.g., x > 2;
3. se clasifică condiţiile BVA în:
   - valide – corespund unor date de intrare/ieşire valide pentru program;
   - non-valide – corespund unor date de intrare/ieşire non-valide pentru program.



Condiţii BVA. Exemple (1)

- **Limitele unei EC valide reprezintă punctul în care comportamentul programului se schimbă!**

- ECs identificate:
  - **1 EC validă: EC1: D1 =[1, 12];**
  - 3 EC non-valide: D2= {luna| luna<1} = (-∞, 1), D3= {luna| luna>12} = (12, +∞), D4 = simboluri alfanumerice;
- Condiţii BVA, construite pentru limitele ECs valide:

• Limita inferioară a EC1: • 1. luna = 0; (non-validă) • 2. luna = 1; • 3. luna = 2;	• Limita superioară a EC1: • 4. luna = 11; • 5. luna = 12; • 6. luna = 13; (non-validă)

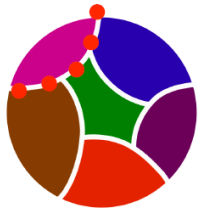
Condiţii BVA. Exemple (2)

- ECs valide identificate:
  - **EC1:** 0,00 Euro – 1000,00 Euro;
  - **EC2:** 1000,01 Euro – 2000,00 Euro;
  - **EC3:** >= 2000,01 Euro.

- Condiţii BVA identificate:
  - Limita inferioară a EC1:
    - 1. amount = -0,01; (non-validă)
    - 2. amount = 0,00;
    - 3. amount = 0,01;
  - Limita superioară a EC1:
    - 4. amount = 999,99;
    - 5. amount = 1000,00;
    - 6. amount = 1000,01; (non-validă)
  - Limita inferioară a EC2:
    - 1. amount = 1000,00; (non-validă)
    - 2. amount = 1000,01;
    - 3. amount = 1000,02;
  - Limita superioară a EC2:
    - 4. amount = 1999,99;
    - 5. amount = 2000,00;
    - 6. amount = 2000,01; (non-validă)
  - Limita inferioară a EC3:
    - 1. amount = 2000,00; (non-validă)
    - 2. amount = 2000,01;
    - 3. amount = 2000,02;
  - Limita superioară a EC3, **MAX_VALUE** (float):
    - 4. amount = MAX_VALUE-0,01;
    - 5. amount = MAX_VALUE;
    - 6. amount = MAX_VALUE+0,01; (non-validă)

BVA. Proiectarea cazurilor de testare . Algoritm

- Algoritm de proiectare a cazurilor de testare:
  1. se asociază un identificator unic fiecărei condiţii BVA (e.g., c1, c2, etc.);
  2. `câttimp` *(nu au fost descrise cazuri de testare pentru toate condiţiile BVA valide/non-valide):*
     - `scrie` *(un caz de testare nou, care corespunde la* ***cât mai multe condiţii BVA valide*** *încă neacoperite);*
     - `scrie` *(un caz de testare nou, care corespunde* ***doar uneia dintre condiţiile BVA non-valide*** *încă neacoperite).*



BVA. Proiectarea cazurilor de testare . Exemple (1)

- ECs valide identificate:
  - 1 EC validă:
    - **EC1: D1 = [1, 12];**

- Cazuri de testare proiectate pe baza condiţiilor BVA identificate:
  - Limita inferioară a EC1:
    - 1. luna = 0 ==> **TC01:** luna = 0; (non-valid)
    - 2. luna = 1 ==> **TC02:** luna = 1; (valid)
    - 3. luna = 2 ==> **TC03:** luna = 2; (valid)
  - Limita superioară a EC1:
    - 4. luna = 11 ==> **TC04:** luna = 11; (valid)
    - 5. luna = 12 ==> **TC05:** luna = 12; (valid)
    - 6. luna = 13 ==> **TC06:** luna = 13; (non-valid)

BVA. Proiectarea cazurilor de testare. Exemple (2)

- ECs valide identificate:
  - **EC1:** 0,00 Euro – 1000,00 Euro;
  - **EC2:** 1000,01 Euro – 2000,00 Euro;
  - **EC3:** >= 2000,01 Euro.

- similar, se proiectează cazuri de testare valide şi non-valide pentru limitele inferioare şi superioare ale EC2 şi EC3;

- Cazuri de testare proiectate pe baza condiţiilor BVA identificate:
  - Limita inferioară a EC1:
    - 1. amount = -0,01; **TC1:** amount = -0,01; (non-valid)
    - 2. amount = 0,00; **TC2:** amount = 0,00 (valid)
    - 3. amount = 0,01; **TC3:** amount = 0,01; (valid)
  - Limita superioară a EC1:
    - 4. amount = 999,99; **TC4:** amount = 999,99; (valid)
    - 5. amount = 1000,00; **TC5:** amount = 1000,00; (valid)
    - 6. amount = 1000,0; **TC6:** amount = 1000,01; (non valid)

BVA. Proiectarea cazurilor de testare. Reguli.

1. **dacă o condiţie de intrare/ieșire precizează apartenența la un interval de valori [a,b]:**

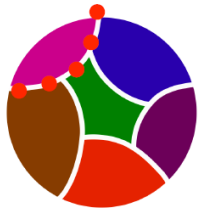
- ==> cazuri de testare pentru:
  - (1) condiții BVA valide - limitele intervalului (e.g., a, a+1; b-1, b);
  - (2) condiții BVA non-valide - valori aflate în afara intervalului (e.g., a-1, b+1);

2. **dacă o condiție de intrare/ieșire precizează o mulțime ordonată de valori:**

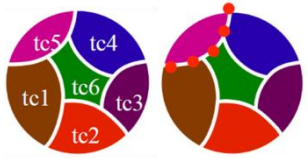
- ==> cazuri de testare pentru:
  - (1) condiții BVA valide - primul și ultimul element din mulțime;
  - (2) condiții BVA non-valide – valoarea imediat mai mică decât cea mai mică valoare din mulțime și valoarea imediat mai mare decât cea mai mare valoare in mulțime;

3. **dacă o condiție de intrare/ieșire precizează numărul de valori (e.g., “între 1 și 5 studenți”):**

- ==> cazuri de testare pentru:
  - (1) condiții BVA valide – numărul minim și maxim de valori, i.e., 1 și 5;
  - (2) condiții BVA non-valide – valoarea imediat mai mică și imediat mai mare, i.e. 0 și 6;



ECP vs BVA



ECP

- presupune că programul tratează similar toate valorile din aceeaşi EC;
- se poate selecta orice valoare din EC;
- se alege o singură valoare din EC, considerată reprezentativă pentru a acoperi testarea acelei EC;
- ECs se construiesc pentru condiţii de intrare/ieșire valide şi non-valide;

BVA

- valorile identificate de condiţiile BVA sunt prelucrate individual, nu în grup;
- valorile se găsesc la limitele dintre ECs, acolo unde programul îşi schimbă comportamentul;
- se iau în considerare valori egale cu limita, valori imediat inferioare şi valori imediat superioare limitei;
- sunt luate în considerare atât datele de intrare cât şi cele de ieşire, corespunzătoare fiecărei EC valide;

ECP + BVA

- **ECP şi BVA dau cele mai bune rezultate atunci când sunt aplicate împreună!**
- **ECP şi BVA sunt tehnici de testare black-box complementare.**

Testarea Black-box

Avantaje

- nu se există informaţii despre implementare;
- activitatea testerului este independentă de cea a programatorului;
- reflecta punctul de vedere al utilizatorului;
- suprinde ambiguităţile sau inconsistenţele din specificaţii;
- începe imediat după finalizarea specificaţiilor.

Dezavantaje

- daca specificaţia nu este clară ==> dificultate de construire a cazurilor de testare;
- la execuţia programului, multe drumurile din graful de execuţie asociat codului rămân netestate ==> secvenţele de cod sursă corespunzătoare pot conţine bug-uri care nu sunt identificate;
- doar un număr foarte mic de date de intrare va fi efectiv testat.

PENTRU EXAMEN…

Pentru examen...

- **testare:**
  - false definiţii ale testării (3); definiţii ale testării (4);
  - terminologie: program, caz de testare;
  - tipuri de testare: exhaustivă, selectivă;
  - reguli de raportare a unui bug;
  - ciclul de viaţă al unui bug (diagrame, descriere);
- **testare black-box:**
  - definiţie, caracteristici;
  - ECP, BVA, ECP vs. BVA, ECP + BVA;
  - aplicarea ECP şi BVA pentru probleme concrete;
  - avantaje şi dezavanataje.

Cursul următor…

- **Testare White-Box**
  - Tehnici de testare white-box
  - Testare bazată pe fluxul de control. Componente
    - Graful fluxului de control. Drumuri în CFG. Complexitatea ciclomatică
    - Testare bazată pe acoperirea drumurilor
  - Testare bazată pe acoperirea codului sursă
    - Acoperirea instrucţiunilor, deciziilor, condiţiilor, deciziilor şi condiţiilor, condiţiilor multiple, drumurilor, buclelor
- **Testare White-box vs Testare Black-box**
- **Testare bazată pe experienţă**
  - Error guessing. Exploratory testing

Referințe bibliografice

- **[Pal2013]** Kaushik Pal, *Software Testing: Verification and Validation,* http://mrbool.com/software-testing-verification-and-validation/29609
- **[Myers2004]** Glenford J. Myers, *The Art of Software Testing,* John Wiley & Sons, Inc., 2004
- **[Frentiu2010]** M. Frentiu, *Verificarea si validarea sistemelor soft*, Presa Universitara Clujeana, 2010.
- **[Patton2005]** R. Patton, *Software Testing,* Sams Publishing, 2005.
- **[NT2005]** K. Naik and P. Tripathy. *Software Testing and Quality Assurance*, Wiley Publishing, 2005.
- **[BBST2010]** Black-Box Software Testing (BBST), Foundations, http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf.