CURS 04.
TESTARE BLACK-BOX

Verificare, validare şi testare automată
[12 Octombrie 2019]

Lector dr. Camelia Chisăliţă-Creţu
Universitatea Babeş-Bolyai, NTT Data
Programul Postuniversitar de Pregătire şi Formare Profesională în Informatică

Conţinut

- Abordări ale testării
- 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. Exemplu
  - Testarea domeniului de valori
  - Avantaje şi dezavanataje
- Întrebări pentru examen
- Bibliografie

ABORDĂRI ALE TESTĂRII

Abordări ale testării. Clasificare

Tehnici de testare asociate

Abordări ale testării. Clasificare

- abordare de testare
  - modalitate de aplicare a unei tehnici de testare;
- clasificare
  - testare Black-box (**criteriul cutiei negre**, *engl.* **Black-box testing**);
  - testare White-box (**criteriul cutiei transparente**, *engl.* **White-box testing**);
  - testare Grey-box (**criteriul cutiei gri**, *engl.* **Grey-box testing**);
  - testare bazată pe experienţă (*engl.* **Experienced-based testing**).

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, Cazuri de utilizare, Scenarii 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 cutiei gri (testare Grey-box) – testare mixtă:
  - folosirea simultană a avantajelor abordărilor black-box şi white-box pentru proiectarea cazurilor de testare;
- 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 se utilizează informaţii referitoare la structura internă a programului, i.e., codul sursă;
  - permite identificarea situaţiilor în care programul nu funcţionează conform specificaţiilor.

Tehnici de testare black-box

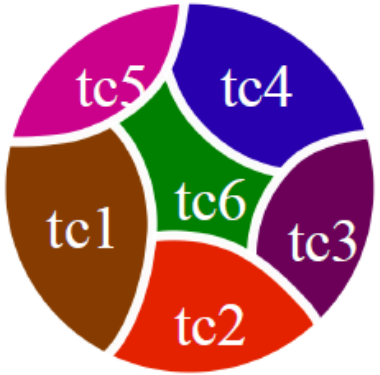
- tehnici de proiectare a cazurilor de testare bazate pe criteriul black-box:
  1. **Partiţionarea în clase de echivalenţă;**
  2. **Analiza valorilor limită;**
  3. Testarea domeniului de valori;
  4. Tabele de decizie;
  5. Grafe de tranziţie a stărilor;
  6. Cazuri de utilizare;
  7. Scenarii de utilizare;
  8. *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;
  - există restricţii, e.g., timp, buget, resursa umană.

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

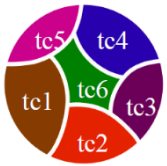
- *clase de echivalenţă disjuncte*:
  - se evită redundanţa cazurilor de testare;
- cazuri de testare:
  - se alege un singur element din fiecare clasă de echivalenţă;



Partiţionarea în clase de echivalenţă. Definiţie

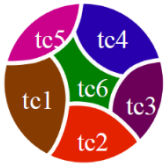
- **clasă de echivalenţă** (*engl.* **equivalence class, EC**):
  - mulţimea datelor de intrare/ieșire pentru care programul are comportament similar [Myers2004];

- procesul de **partiţionare 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.



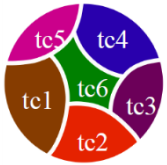
ECP. Identificarea ECs

- **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. Proiectarea cazurilor de testare. Algoritm

- Algoritm de proiectare a cazurilor de testare:
  1. se asociază un identificator unic fiecărei clase de echivalență (e.g., EC_1, EC_2, 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).*

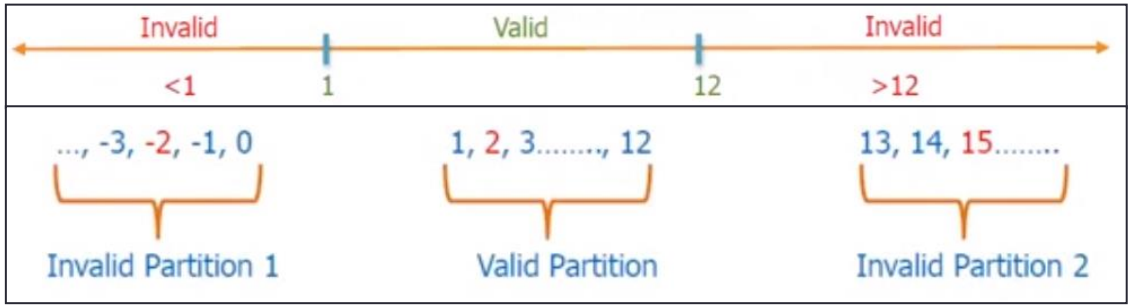


ECP. Identificarea ECs. Exemplu 1

- 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:	**Abordare secundară:**
un număr >=1 şi <=12; 1 EC validă: **EC_1:** D_1 = [1, 12]; 2 EC non-valide: **EC_2:** D_2= {luna<1} = (-∞, 1); **EC_3:** D_3= {luna>12} = (12, +∞); **EC_4:** D_4 = simboluri/caractere din alfabet.	• *Numărul de ordine al lunii în cadrul unui an: prima, a doua, a treia, etc.* • *Numărul de cifre: 0 cifre, 1-2 cifre (1 to 12), 3 cifre non-valid);* • *Numărul de spaţii înainte de/după cifră/e: 0 (cazul general), >0 (caz excepţional);* • *Numărul de spaţii între cifre: 0 (caz general), >0 (unele programe (OOWriter) ignoră caracterele "non-valide" din interiorul numărului dat ca string);* • *Codurile ASCII: cifre (48-57), non-cifre (58 to 127), etc.*

ECP. Selectarea datelor de test. Exemplu 1



- **ECs identificate:**
  - 1 EC validă, **EC_1: D_1 =[1, 12];**
  - 3 EC non-valide, **EC_2: D_2= {luna| luna<1} = (-∞, 1), EC_3: D_3= {luna| luna>12} = (12, +∞), EC_4: D_4 = simboluri alfanumerice;**
- **Cazuri de testare proiectate:**
  - 1 EC validă ==> 1 caz de testare valid, e.g., **TC_{01}: luna = 2;**
  - 3 EC non-valide ==> 3 cazuri de testare non-valide, e.g., **TC_{02}: luna =-2, TC_{03}: luna = 15, TC_{04}: 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 acea EC.**

ECP. Identificarea ECs. Exemplu 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:
    - **EC_1:** 0,00 Euro – 1000,00 Euro;
    - **EC_2:** 1000,01 Euro – 2000,00 Euro;
    - **EC_3:** >= 2000,01 Euro.
  - Clase de echivalenţă non-valide:
    - **EC_4:** < 0,00 Euro;
    - **EC_5:** > valoarea maximă admisă pentru un cont;
    - **EC_6:** caractere din alfabet.

ECP. Selectarea datelor de test. Exemplu 2

- **ECs identificate:**
- **3 ECs valide:**
  - **EC_1:** 0,00 Euro – 1000,00 Euro;
  - **EC_2:** 1000,01 Euro – 2000,00 Euro;
  - **EC_3:** >= 2000,01 Euro.
- **3 ECs non-valide:**
  - **EC_4:** < 0,00 Euro;
  - **EC_5:** > valoarea maximă admisă pentru un cont;
  - **EC_6:** caractere din alfabet.

- **Cazuri de testare proiectate:**
  - **3 ECs valide ==>** 3 cazuri de testare valide,e.g.:
    - **TC_{01}:** amount= 678,99;
    - **TC_{02}:** amount = 1742,81;
    - **TC_{03}:** 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.:
    - **TC_{04}:** amount = -0,79;
    - **TC_{05}:** amount = 9876543210,123;
    - **TC_{06}:** amount = #12a.

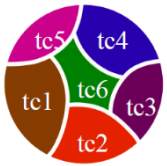
ECP. Proiectarea cazurilor de testare. Reguli (1)

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:**

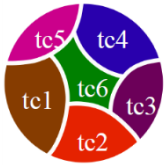
- ==> 1 EC validă pentru fiecare valoare, 1 EC non-validă;
  - E.g.: `tip curs ∈ CourseType = {opțional, obligatoriu, facultativ};`
  - 1 EC validă pentru fiecare element din `CourseType`:
    - **EC_1:** {`opțional`},
    - **EC_2:** {`obligatoriu`},
    - **EC_3:** {`facultativ`} ===> 3 ECs valide;
  - 1 EC non-validă:
    - **EC_4:** M= $\{e \mid e \notin$ `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:
    - **EC_1:** D=[1,5];
  - 2 EC non-valide:
    - **EC_2:** nici un student;
    - **EC_3:** mai mult de 5 studenți;

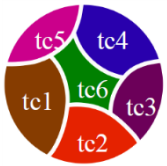


ECP. Proiectarea cazurilor de testare. Reguli (3)

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:
    - **EC_1:** primul caracter este un simbol numeric;
  - 1 EC non-validă:
    - **EC_2:** primul caracter nu este un simbol numeric.

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



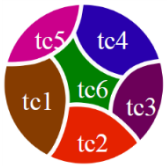
ECP. Acoperirea testării ECs

- **calculul acoperirii (*engl.* coverage) testării ECs pentru tehnica de testare ECP:**

$$\text{Acoperirea ECs} = \frac{\text{numărul de ECs testate}}{\text{numărul de ECs identificate}} \times 100$$

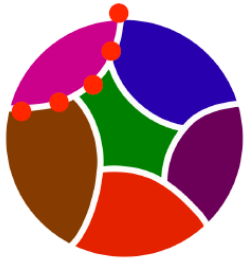
- E.g.:
  - pe baza specificaţiilor au fost identificate 18 ECs (pentru datele de intrare şi ieşire);
  - 15 ECs au fost testate prin cazurile de testare proiectate;
  - **Acoperirea ECs= (15/18)*100 = 83,33%.**

- **Acoperirea ECs poate fi folosită ca şi criteriu de terminare a testării, i.e., exit criteria.**



Este ECP eficientă la limita dintre ECs ?

- 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++;
```

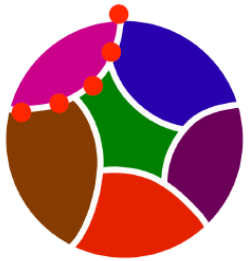
  - [ECP]: pentru EC_1: [3, MaxInt] se alege TC_{01}: x=4, dar TC_{01} nu surprinde bug-ul de implementare;

Analiza valorilor limită. Motivaţie

- analiza valorilor limită investighează posibilele bug-uri existente la limita dintre ECs identificate;
  - E.g.: pentru x≥3

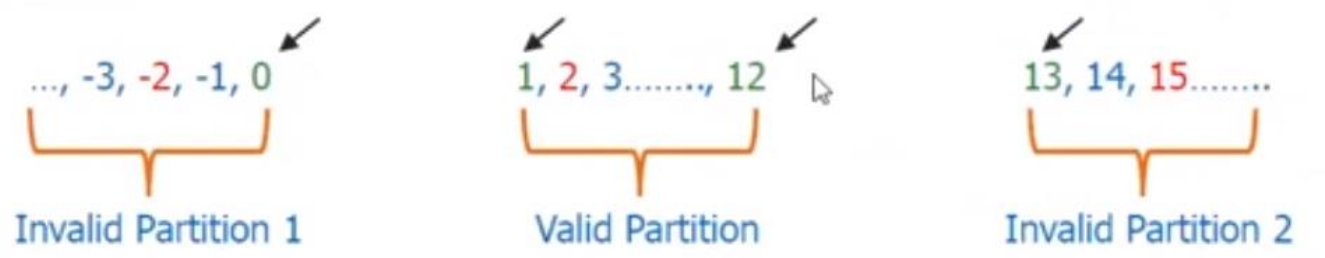
```
if (x>3) y++;   //bug
if (x>=3) y++;
```

  - [ECP]: pentru EC_1: [3, MaxInt] se alege TC_{01}: x=4, dar TC_{01} nu surprinde bug-ul de implementare;
  - [BVA]: pentru EC_1: **[3**, MaxInt] se alege TC_{02}: 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 ECs de intrare/ieşire;



- **valoare limită** (*engl.* **boundary value, BV):**
  - valoare a domeniului în care comportamentul programului se modifică.

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. Excepţii de identificare a condiţiilor BVA

- există ECs care **nu au limite**:
  - E.g.: mulţimea {Dl, Dna, Dra, Dr.};
- există ECs (ordonate) care **nu au două limite** - inferioară şi superioară;
  - E.g.: soldul unui cont bancar;
- **variabile multiple dependente**:
  - E.g.: variabilele: număr card bancar, data eliberare, data expirare, nume titular;
    - toate variabilele au valori valide <==> card valid;
    - se verifică fiecare variabilă individual, cu valori non-valide ==> card non-valid;
- **ECs dependente** - valoarea unei variabile a depinde de alta:
  - E.g.:, format pagină A4 are width=29.7cm;

- **Este necesară identificarea limitelor ECs?**
  - factori: **analiza specificaţiilor, experienţa testerului**;
    - E.g.: numărarea, ordonarea elementelor din ECs.

Condiţii BVA. Sumar

Tip ECs	Există limite
interval de valori	da
număr de valori	da
mulţime valori neordonate	**nu**
mulţime valori ordonate	**da**
valoare “must be”	nu
secvenţă	da
ECs dependente	**da**
variabile multiple dependente	**nu**

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).*

Condiţii BVA. Exemplu 1

- **Limitele unei EC valide indică situaţiile în care comportamentul programului se schimbă!**

- ECs identificate:
  - **1 EC validă: EC_1: D_1 =[1, 12];**
  - 3 EC non-valide: D_2= {luna| luna<1} = (-∞, 1), D_3= {luna| luna>12} = (12, +∞), D_4 = simboluri alfanumerice;
- Condiţii BVA, construite pentru limitele ECs valide:

• Limita inferioară a EC_1: • 1. luna = 0; (non-validă) • 2. luna = 1; • 3. luna = 2;	• Limita superioară a EC_1: • 4. luna = 11; • 5. luna = 12; • 6. luna = 13; (non-validă)

BVA. Proiectarea cazurilor de testare. Exemplu 1

- ECs valide identificate:
  - 1 EC validă:
    - **EC_1: D_1 = [1, 12];**

- Cazuri de testare proiectate pe baza condiţiilor BVA identificate:
  - Limita inferioară a EC_1:
    - 1. luna = 0 ==> **TC_{01}:** luna = 0; (non-valid)
    - 2. luna = 1 ==> **TC_{02}:** luna = 1; (valid)
    - 3. luna = 2 ==> **TC_{03}:** luna = 2; (valid)
  - Limita superioară a EC_1:
    - 4. luna = 11 ==> **TC_{04}:** luna = 11; (valid)
    - 5. luna = 12 ==> **TC_{05}:** luna = 12; (valid)
    - 6. luna = 13 ==> **TC_{06}:** luna = 13; (non-valid)

Condiţii BVA. Exemplu 2

- ECs valide identificate:
  - **EC_1:** 0,00 Euro – 1000,00 Euro;
  - **EC_2:** 1000,01 Euro – 2000,00 Euro;
  - **EC_3:** >= 2000,01 Euro.

- Condiţii BVA identificate:
  - Limita inferioară a EC_1:
    - 1. amount = -0,01; (non-validă)
    - 2. amount = 0,00;
    - 3. amount = 0,01;
  - Limita superioară a EC_1:
    - 4. amount = 999,99;
    - 5. amount = 1000,00;
    - 6. amount = 1000,01; (non-validă)
  - Limita inferioară a EC_2:
    - 1. amount = 1000,00; (non-validă)
    - 2. amount = 1000,01;
    - 3. amount = 1000,02;
  - Limita superioară a EC_2:
    - 4. amount = 1999,99;
    - 5. amount = 2000,00;
    - 6. amount = 2000,01; (non-validă)
  - Limita inferioară a EC_3:
    - 1. amount = 2000,00; (non-validă)
    - 2. amount = 2000,01;
    - 3. amount = 2000,02;
  - Limita superioară a EC_3, **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. Exemplu 2

- ECs valide identificate:
  - **EC_1:** 0,00 Euro – 1000,00 Euro;
  - **EC_2:** 1000,01 Euro – 2000,00 Euro;
  - **EC_3:** >= 2000,01 Euro.

- similar, se proiectează cazuri de testare valide şi non-valide pentru limitele inferioare şi superioare ale EC_2 şi EC_3;

- Cazuri de testare proiectate pe baza condiţiilor BVA identificate:
  - Limita inferioară a EC_1:
    - 1. amount = -0,01; **TC_{01}:** amount = -0,01; (non-valid)
    - 2. amount = 0,00; **TC_{02}:** amount = 0,00 (valid)
    - 3. amount = 0,01; **TC_{03}:** amount = 0,01; (valid)
  - Limita superioară a EC_1:
    - 4. amount = 999,99; **TC_{04}:** amount = 999,99; (valid)
    - 5. amount = 1000,00; **TC_{05}:** amount = 1000,00; (valid)
    - 6. amount = 1000,0; **TC_{06}:** 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;

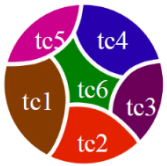
BVA. Acoperirea testării condiţiilor BVA

- **calculul acoperirii (*engl.* coverage) testării condiţiilor BVA:**

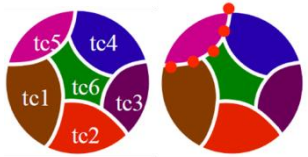
$$\text{Acoperirea BVAs} = \frac{\text{numărul de condiţii BVA testate}}{\text{numărul de condiţii BVA identificate}} \times 100$$

- E.g.:
  - pe baza specificaţiilor au fost identificate 64 BVAs (pentru datele de intrare şi ieşire, corespunzător ECs valide);
  - 48 BVAs au fost testate prin cazurile de testare proiectate;
  - **Acoperirea BVAs= (48/64)*100 = 75%.**

- **Acoperirea BVAs poate fi folosită ca şi criteriu de terminare a testării, i.e., exit criteria.**



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;

- *obiectiv al testării* = verificarea respectării specificaţiilor pentru valori uzuale, i.e., **building confidence in software**;

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;
- *obiectiv al testării* = căutarea bug-urilor uzuale, i.e., **bug hunting**;

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, care definesc testarea domeniului de valori, i.e., domain testing.**

ECP + BVA. Exemplu

- **Se consideră metoda**

  `computeTotalAmount (int code, int quantity): int`

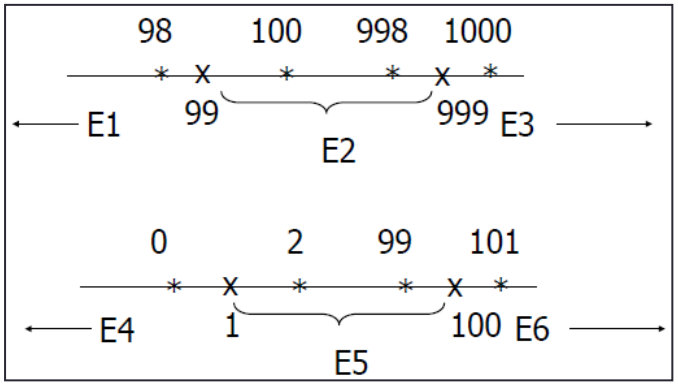
- **care calculează valoarea totală a unei comenzi pentru produsul cu id-ul dat (`code`) şi cantitatea precizată (`quantity`), dacă valorile date sunt valide;**
- Constrângeri:
  - `code: [99, 999];`
  - `quantity: [1, 100];`
- **Etape de realizare ECP + BVA:**
  1. identificarea ECs valide şi non-valide;
  2. identificarea condiţiilor BVA pentru ECs valide existente;
  3. proiectarea TCs pentru condiţiile BVA identificate;
  4. acoperirea individuală a condiţiilor BVA non-valide;
  5. minimizarea mulţimii de TCs pentru testarea ECP + BVA.

ECP + BVA. ECs valide şi non-valide

- **1. Identificarea ECs valide şi non-valide:**
  - Constrângeri:
    - `code: [99, 999];`
    - `quantity: [1, 100];`
- ECs valide și non-valide identificate:
  - **ECs pentru code:**
    - **E_1:** code < 99; (non-validă)
    - **E_2:** code: [99, 999];
    - **E_3:** code > 999. (non-validă)
  - **ECs pentru quantity:**
    - **E_4:** quantity < 1; (non-validă)
    - **E_5:** quantity: [1, 100];
    - **E_6:** quantity > 100. (non-validă)

ECP + BVA. Condiţii BVA

- **2. identificarea condiţiilor BVA pentru ECs valide existente;**
  - EC valide:
    - **E_2: code: [99, 999];**
    - **E_5: quantity: [1, 100];**
- Condiţii BVA pentru E_2 şi E_5;
  - Condiţii pentru limita inferioară a lui E_2:
    - code = 98;
    - code = 99;
    - code = 100;
  - Condiţii pentru limita superioară a lui E_2:
    - code = 998;
    - code = 999;
    - code = 1000;

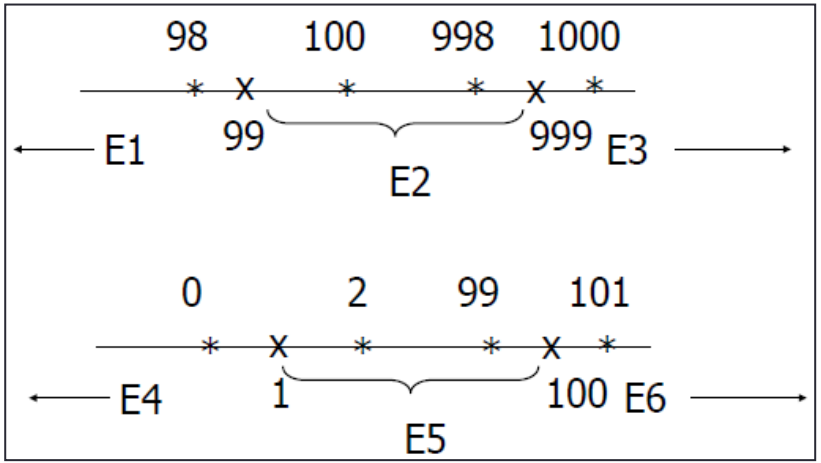


- x – limită a unei EC; * - valoare invecinată limitei;

- Condiţii pentru limita inferioară lui E_5:
  - quantity = 0;
  - quantity = 1;
  - quantity = 2;
- Condiţii pentru limita superioară a lui E_5:
  - quantity = 99;
  - quantity = 100;
  - quantity = 101;

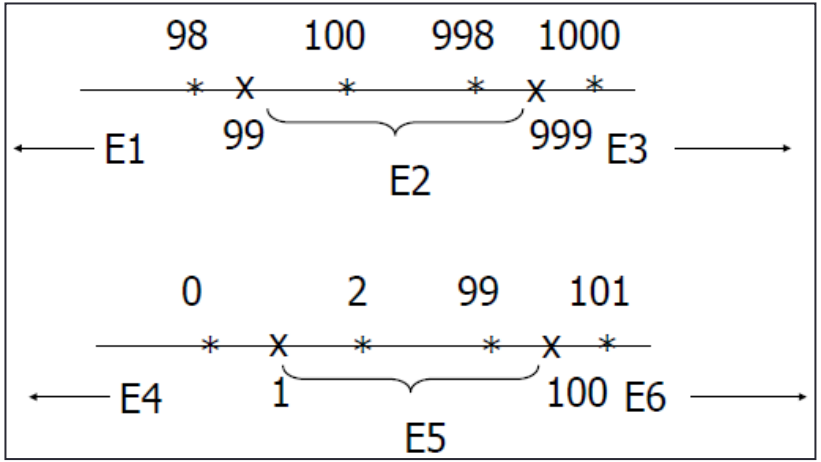
ECP + BVA. TCs pentru condiţiile BVA

- **3. proiectarea TCs pentru condiţiile BVA identificate;**
  - **Cazuri de testare identificate, astfel încât toate condiţiile BVA să fie testate cel puţin o dată:**
    - **TC_1: (code = 98, quantity = 0);**
    - **TC_2: (code = 99, quantity = 1);**
    - **TC_3: (code = 100, quantity = 2);**
    - **TC_4: (code = 998, quantity = 99);**
    - **TC_5: (code = 999, quantity = 100);**
    - **TC_6: (code = 1000, quantity = 101).**



ECP + BVA. Acoperirea condiţiilor BVA non-valide

- **4. acoperirea individuală a condiţiilor BVA non-valide;**
  - **Cazurile de testare TC_1 şi TC_6 nu acoperă *individual* condiţiile BVA non-valide;**
    - **TC_1: (code = 98, quantity = 0);**
    - **TC_2: (code = 99, quantity = 1);**
    - **TC_3: (code = 100, quantity = 2);**
    - **TC_4: (code = 998, quantity = 99);**
    - **TC_5: (code = 999, quantity = 100);**
    - **TC_6: (code = 1000, quantity = 101).**



- **TC_1 şi TC_6 se pot înlocui cu:**
  - **TC_7: (code = 98, quantity = 10);**
  - **TC_8: (code = 1000, quantity = 10);**
  - **TC_9: (code = 40, quantity = 0);**
  - **TC_{10}: (code = 40, quantity = 101).**

ECP + BVA. Minimizarea mulţimii de TCs

- **5. minimizarea mulţimii de TCs pentru testarea ECP + BVA;**
  - ~~TC_1: (code = 98, quantity = 0);~~
  - TC_2: (code = 99, quantity = 1);
  - TC_3: (code = 100, quantity = 2);
  - TC_4: (code = 998, quantity = 99);
  - TC_5: (code = 999, quantity = 100);
  - ~~TC_6: (code = 1000, quantity = 101);~~

  - TC_7: (code = 98, quantity = 10);
  - TC_8: (code = 1000, quantity = 10);
  - TC_9: (code = 40, quantity = 0);
  - TC_{10}: (code = 40, quantity = 101).

ECP + BVA. Lista finală de TCs

- Pentru metoda

```
computeTotalAmount (int code,
      int quantity): int
```

- care calculează valoarea totală a unei comenzi pentru produsul specificat şi cantitatea dată;
- Constrângeri:
  - code : [99, 999];
  - quantity: [1, 100];

- **Lista de TCs finală bazată pe ECP şi BVA este:**
  - **TC_2: (code = 99, quantity = 1);**
  - **TC_3: (code = 100, quantity = 2);**
  - **TC_4: (code = 998, quantity = 99);**
  - **TC_5: (code = 999, quantity = 100);**
  - **TC_7: (code = 98, quantity = 10);**
  - **TC_8: (code = 1000, quantity = 10);**
  - **TC_9: (code = 40, quantity = 0);**
  - **TC_{10}: (code = 40, quantity = 101).**

  - **Observaţie:**
    - **Lista de TCs nu este unică, dar poate fi considerată minimală pe baza ECP şi BVA.**

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.

ÎNTREBĂRI PENTRU EXAMEN

Întrebări cu răspuns scurt

Întrebări cu răspuns lung

Întrebări cu răspuns scurt

- **Întrebări cu răspuns scurt:**
  1. Definiţi noţiunea: criteriu de testare. Enumeraţi criteriile de testare studiate şi două tehnici de testare pentru fiecare criteriu prezentat.
  2. Definiţi noţiunea: testare black-box. Enumeraţi tehnicile de testare bazate pe testare black-box studiate. Exemplificaţi.
  3. Definiţi noţiunea: partiţionarea în clase de echivalenţă. Exemplificaţi.
  4. Definiţi noţiunea: clasă de echivalenţă. Clasificaţi şi exemplificaţi.
  5. Descrieţi o regulă de proiectare a cazurilor de testare bazată pe ECP. Exemplificaţi.
  6. Definiţi noţiunea: analiza valorilor limită. Exemplificaţi.
  7. Definiţi noţiunea: condiţie BVA. Clasificaţi şi exemplificaţi.
  8. Descrieţi o regulă de proiectare bazată pe BVA. Exemplificaţi.
  9. Descrieţi pe scurt trei avantaje ale testării black-box.
  10. Descrieţi pe scurt trei dezavantaje ale testării black-box.

Întrebări cu răspuns lung

- **Întrebări cu răspuns lung:**
  1. Comparaţi noţiunile: testare black-box şi testare white-box. Exemplificaţi.
  2. Comparaţi noţiunile: testare black-box şi criteriul statistic. Exemplificaţi.
  3. Comparaţi noţiunile: partiţionare în clase de echivalenţă şi analiza valorilor limită. Exemplificaţi.
  4. Descrieţi tehnica partiţionării în clase de echivalenţă. Motivaţi importanţa acestei tehnici. Exemplificaţi.
  5. Descrieţi şi comparaţi două reguli de proiectare a cazurilor de testare bazate pe ECP. Exemplificaţi.
  6. Descrieţi tehnica analizei valorilor limită. Motivaţi utilitatea acestei tehnici de testare. Exemplificaţi.
  7. Descrieţi şi comparaţi două reguli de proiectare a cazurilor de testare bazate pe BVA. Exemplificaţi.
  8. Motivaţi de ce ECP şi BVA dau rezultate mai bune atunci când sunt utilizate împreună.
  9. Descrieţi avantajele tehnicii bazate pe ECP faţă de avantajele tehnicii bazate pe BVA. Exemplificaţi.
  10. Descrieţi avantajele şi dezavantajele testării black-box. Exemplificaţi.

Referințe bibliografice

- **[Pal2013]** Kaushik Pal, *Software Testing: Verification and Validation,* http://mrbool.com/software-testing-verification-and-validation/29609l
- **[Dijkstra1969]** E.W. Dijkstra*, Software engineering techniques*, Report on a conference sponsored by the NATO Science Committee, Rome, Italy, 2731 October 1969.
- **[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.
- **[BBST2010]** Black-Box Software Testing (BBST), Foundations, http://www.testingeducation.org/BBST/foundations/BBSTFoundationsNov2010.pdf.
- **[Patton2005]** R. Patton, *Software Testing,* Sams Publishing, 2005.
- **[NT2005]** K. Naik and P. Tripathy. *Software Testing and Quality Assurance*, Wiley Publishing, 2005.