

CURS 02B.

TESTARE BLACK-BOX

Verificarea și validarea sistemelor soft
[03 Martie 2020]

Lector dr. Camelia Chisăliță-Crețu
Universitatea Babeș-Bolyai

Conținut

- Abordări ale testării
- Testare Black-Box
 - Definiție. Caracteristici.
 - Clasificare. Tehnici de testare black-box
 - Partiționarea în clase de echivalență. Exemple
 - Analiza valorilor limită. Exemple
 - Partiționarea în clase de echivalență vs Analiza valorilor limită
 - Testarea domeniului de valori
 - Avantaje și dezavantaje
- Pentru examen...
- Bibliografie

ABORDĂRI ALE TESTĂRII

Abordări ale testării. Clasificare

Tehnici de testare asociate

Abordări ale testării. Clasificare

- abordare a testării
 - modalitate de realizare a testării în care se aplică una sau mai multe tehnici de testare în cadrul unei strategii de testare stabilită anterior;
- 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 experiență (*engl. Experienced-based testing*);
 - testare bazată pe scripturi (*engl. Scripted-based testing*);

Abordări ale testării. Tehnici de testare asociate

- Testare Black-Box – testare funcțională:
 - Partiționarea în clase de echivalență;
 - Analiza valorilor limită;
 - Tabele de decizie, Cazuri de utilizare, Scenarii de utilizare, etc.;
- Testare White-box – testare structurală:
 - Acoperirea fluxului de control (e.g., instrucțiuni, ramificații, decizii, condiții, bucle, drumuri);
 - Acoperirea fluxului de date;
- Testare Grey-box – testare mixtă:
 - folosirea simultană a avantajelor abordărilor black-box și white-box pentru proiectarea cazurilor de testare.

TESTARE BLACK-BOX

Definiție. Caracteristici. Tehnici de testare black-box

Partiționarea în clase de echivalență. Exemple

Analiza valorilor limită. Exemple

Partiționarea în clase de echivalență vs Analiza valorilor limită

Avantaje și dezavantaje

Definiție. Caracteristici

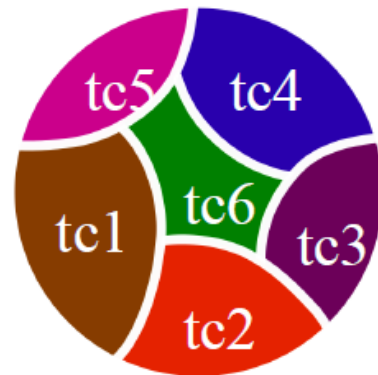
- **testare black-box** (*engl. black-box testing, data driven testing, input/output driven testing*):
 - testare funcțională;
 - datele de intrare se aleg pe baza **specificației problemei**, programul fiind văzut ca o cutie neagră;
 - nu se utilizează informații referitoare la structura internă a programului, i.e., codul sursă;
 - permite identificarea situațiilor în care programul nu funcționează conform specificațiilor.

Tehnici de testare black-box

- tehnici de proiectare a cazurilor de testare bazate pe criteriul black-box:
 1. **Partiționarea în clase de echivalență;**
 2. **Analiza valorilor limită;**
 3. Testarea domeniului de valori;
 4. Tabele de decizie;
 5. Cazuri de utilizare;
 6. Scenarii de utilizare;
 7. *alte tehnici.*

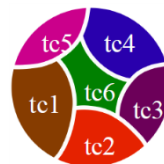
Partiționarea în clase de echivalență. Motivație

- în general, **testarea exhaustivă** nu este posibil de realizat, e.g.:
 - există un set consistent de date de intrare sau domeniul de valori testat este infinit;
 - există restricții, e.g., timp, buget, resursa umană.
- partiționarea în clase de echivalență (*engl. **Equivalence Class Partitioning, ECP***) este **eficientă** pentru reducerea numărului de cazuri de testare care trebuie proiectate;
- *clase de echivalență disjuncte*:
 - se evită redundanța cazurilor de testare;
- cazuri de testare:
 - se alege un singur element din fiecare clasă de echivalență;



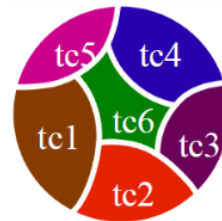
Partiționarea în clase de echivalență. Definiție

- **clasă de echivalență** (*engl. equivalence class, EC*):
 - mulțimea datelor de intrare/ieșire pentru care programul are comportament similar [[Myers2004](#)];
- procesul de **partiționare în clase de echivalență** (*engl. equivalence class partitioning, ECP*):
 - împărțirea (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.



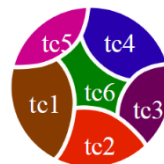
ECP. Algoritm

- Algoritm de aplicare a ECP (*identificarea ECs și proiectarea TCs*):
 1. se identifică clasele de echivalență pe baza condițiilor de intrare/ieșire;
 2. 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.
 3. se asociază un identificator unic fiecărei clase de echivalență (e.g., EC₁, EC₂, etc.);
 4. cât timp (*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. Proiectarea cazurilor de testare. Reguli (1)

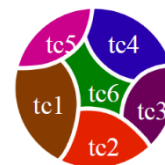
- 1. dacă o condiție de intrare precizează apartenența la un interval de valori $[a,b]$:**
 - \Rightarrow 1 EC validă, 2 EC non-valide;
 - E.g.: luna, o valoare intervalul $[1, 12]$;
- 2. dacă o condiție de intrare precizează o mulțime finită de valori de intrare:**
 - \Rightarrow 1 EC validă pentru fiecare valoare, 1 EC non-validă;
 - E.g.: `tip curs \in CourseType = {opțional, obligatoriu, facultativ}`;
 - 1 EC validă pentru fiecare element din `CourseType`:
 - $EC_1: \{\text{opțional}\}$,
 - $EC_2: \{\text{obligatoriu}\}$,
 - $EC_3: \{\text{facultativ}\} \Rightarrow$ 3 ECs valide;
 - 1 EC non-validă:
 - $EC_4: M = \{e \mid e \notin \text{CourseType}\}$;



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 până la 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;

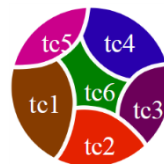


ECP. Proiectarea cazurilor de testare. Reguli (3)

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

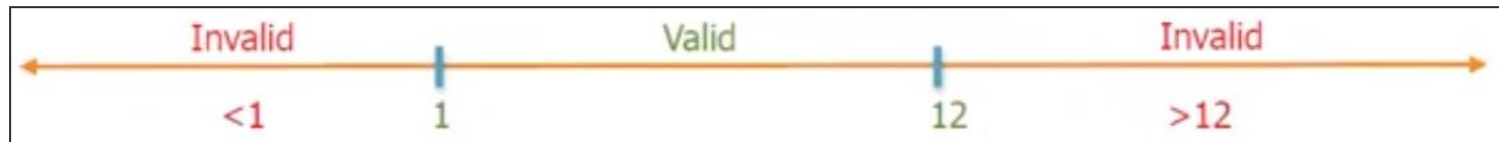
- ==> 1 EC validă, 1 EC non-validă.
 - E.g.,: “primul caracter din parolă trebuie să fie un simbol numeric”;
 - 1 EC validă:
 - EC_1 : primul caracter este un simbol numeric;
 - 1 EC non-validă:
 - EC_2 : primul caracter nu este un simbol numeric.

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



ECP. Exemplu 1. Identificarea ECs

- Se consideră un formular de înscriere la un concurs. Pentru data nașterii se introduce ziua, luna și anul.
- Identificați clasele de echivalență corespunzătoare câmpului lună calendaristică (pentru data nașterii). Domeniul de valori valide este $[1, 12]$.



Abordare primară:

un număr ≥ 1 și ≤ 12 ;

1 EC validă:

EC₁: $D_1 = [1, 12]$;

2 EC non-valide:

EC₂: $D_2 = \{\text{luna} < 1\} = (-\infty, 1)$;

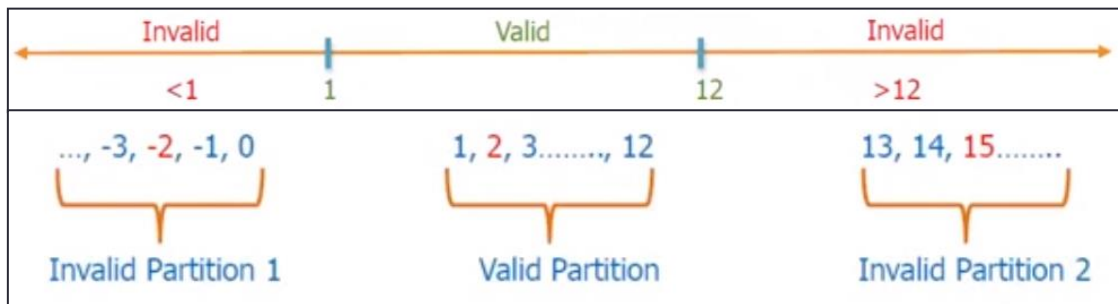
EC₃: $D_3 = \{\text{luna} > 12\} = (12, +\infty)$;

EC₄: $D_4 = \text{simboluri/litere din alfabet}$.

Abordare secundară:

- Numărul de ordine al lunii în cadrul unui an: prima, a doua, a treia, etc.
- Numărul de cifre: 0 cifre, 1-2 cifre (1 .. 12), 3 cifre (non-valid);
- Numărul de spații înainte de/după cifră/e: 0 (cazul general), >0 (caz excepțional);
- Numărul de spații 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. Exemplu 1. Proiectarea cazurilor de testare



- **ECs identificate:**
 - 1 EC validă, $EC_1: D_1 = [1, 12]$;
 - 3 EC non-valide, $EC_2: D_2 = \{luna \mid luna < 1\} = (-\infty, 1)$, $EC_3: D_3 = \{luna \mid luna > 12\} = (12, +\infty)$, $EC_4: D_4 = \text{simboluri alfanumerice}$;
- **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. Exemplu 2. Identificarea ECs

- Pentru constituirea unui depozit bancar se consideră următoarea ofertă de dobânzi:
 - 0,50% până la 1000,00 Euro;
 - 1,00% până la 2000,00 Euro, dar mai mult de 1000,00 Euro;
 - 1,50% peste 2000,00 Euro;
- **Care sunt clasele de echivalență valide și non-valide pentru valoarea depozitului constituit?**
 - Clase de echivalență valide:
 - **EC₁**: 0,00 Euro – 1000,00 Euro;
 - **EC₂**: 1000,01 Euro – 2000,00 Euro;
 - **EC₃**: $\geq 2000,01$ Euro.
 - Clase de echivalență non-valide:
 - **EC₄**: $< 0,00$ Euro;
 - **EC₅**: $>$ valoarea maximă admisă pentru un depozit.
 - **EC₆**: caractere din alfabet.

ECP. Exemplu 2. Proiectarea cazurilor de testare

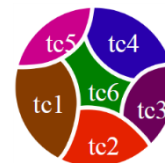
- **ECs identificate:**
- **3 ECs valide:**
 - EC_1 : 0,00 Euro – 1000,00 Euro;
 - EC_2 : 1000,01 Euro – 2000,00 Euro;
 - EC_3 : $\geq 2000,01$ Euro.
- **3 ECs non-valide:**
 - EC_4 : $< 0,00$ Euro;
 - EC_5 : $>$ valoarea maximă admisă pentru un depozit;
 - EC_6 : caractere din alfabet.
- **Cazuri de testare proiectate:**
 - **3 ECs valide \implies 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 \implies 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. Acoperirea testării ECs

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

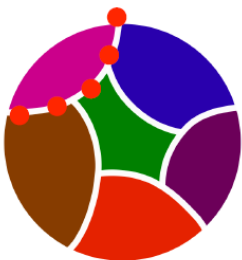
$$\text{Acoperirea ECs} = \frac{\text{numărul de ECs testate}}{\text{numărul de ECs identificate}} \times 100$$

- E.g.:
 - pe baza specificațiilor au fost identificate 18 ECs (pentru datele de intrare și ieșire);
 - pentru 15 ECs s-au proiectat, implementat și executat teste;
 - **Acoperirea ECs = $(15/18) \times 100 = 83,33\%$.**
- **Acoperirea ECs** poate fi folosită ca și 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 și la limitele ECs identificate;



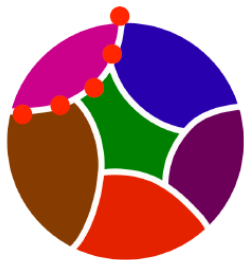
- există greșeli de programare tipice care apar la limita ECs identificate;
 - e.g., pentru $x \geq 3$

```
if (x>3) y++; //bug
if (x>=3) y++;
```
 - [ECP]: pentru $EC_1: [3, \text{MaxInt}]$ se alege $TC_{01}: x=4$, dar TC_{01} nu surprinde bug-ul de implementare.

Analiza valorilor limită. Motivație

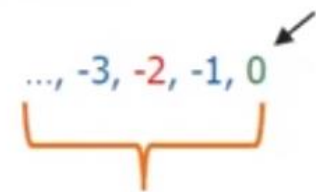
- analiza valorilor limită investighează posibilele bug-uri existente la limita dintre ECs identificate;
 - E.g.: pentru $x \geq 3$

```
if (x>3) y++; //bug
if (x>=3) y++;
```
 - [ECP]: pentru EC_1 : $[3, \text{MaxInt}]$ se alege TC_{01} : $x=4$, dar TC_{01} nu surprinde bug-ul de implementare;
 - [BVA]: pentru EC_1 : **3**, $\text{MaxInt}]$ se alege TC_{02} : $x=3$;

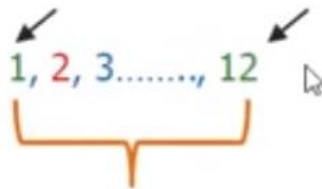


Analiza valorilor limită. Definiție

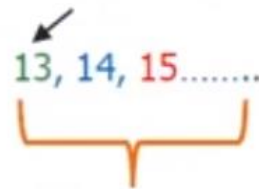
- **analiza valorilor limită** (*engl.* **boundary value analysis, BVA**) [[Myers2004](#)]:
 - testarea realizată prin alegerea datelor de test pe baza limitelor ECs de intrare/ieșire;



Invalid Partition 1



Valid Partition



Invalid Partition 2

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



Condiții BVA. Identificare

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



Condiții BVA. Excepții de identificare a condițiilor BVA

- **există ECs care nu au limite:**
 - E.g.: mulțimea {DI, Dna, Dra, Dr.} sau CourseType = {opțional, obligatoriu, facultativ};
- **există ECs (ordonate) care nu au două limite** (inferioară și 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 și toate constrângerile existente între acestea sunt satisfăcute \Leftrightarrow card valid;
 - dacă există o variabilă are valori non-valide \Rightarrow card non-valid;
- **ECs dependente** – valoarea unei variabile depinde de/ influențează valoarea alteia:
 - E.g.: în OpenOffice Writer există mai multe tipuri de pagină: format_pagină = {A2, A3, A4}; formatul A4 constrânge dimensiunea header-ului paginii (header height) maximă 20.56 cm.

Condiții BVA. Sumar

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

BVA. Algoritm

- Algoritm de aplicare a BVA (*identificarea condițiilor BVA și proiectarea TCs*):
 1. se identifică limitele tuturor ECs valide de intrare/ieșire;
 2. se scriu condiții BVA pentru fiecare limită a fiecărei EC identificate, astfel încât:
 - valoarea să fie sub limită (mai mică decât limita), e.g., $x < 2$;
 - valoarea să fie pe limită (egală cu limita) , e.g., $x = 2$;
 - valoarea să fie deasupra limitei (mai mare decât limita), e.g., $x > 2$;
 3. se clasifică condițiile BVA în
 - **valide** – corespund unor date de intrare/ieșire valide pentru program;
 - **non-valide** – corespund unor date de intrare/ieșire non-valide pentru program.
 4. se asociază un identificator unic fiecărei condiții BVA (e.g., c1, c2, etc.);
 5. cât timp (*nu au fost descrise cazuri de testare pentru toate condițiile BVA valide/non-valide*):
 - scrie (*un caz de testare nou, care corespunde la **cât mai multe condiții BVA valide** încă neacoperite*);
 - scrie (*un caz de testare nou, care corespunde **doar uneia dintre condițiile BVA non-valide** încă neacoperite*).



BVA. Proiectarea cazurilor de testare. Reguli

1. **dacă o condiție 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 de valori ordonată:**

- ==> 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., “de la 1 până la 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. Exemplu 1. Condiții BVA

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



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

<ul style="list-style-type: none"> • Limita inferioară a EC_1: <ul style="list-style-type: none"> • 1. luna = 0; (non-validă) • 2. luna = 1; • 3. luna = 2; 	<ul style="list-style-type: none"> • Limita superioară a EC_1: <ul style="list-style-type: none"> • 4. luna = 11; • 5. luna = 12; • 6. luna = 13; (non-validă)
---	--

BVA. Exemplu 1. Proiectarea cazurilor de testare

- ECs valide identificate:
 - 1 EC validă:
 - $EC_1: D_1 = [1, 12];$
- Cazuri de testare proiectate pe baza condițiilor 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)

BVA. Exemplu 2. Condiții BVA

- ECs valide identificate:
 - **EC₁**: 0,00 Euro – 1000,00 Euro;
 - **EC₂**: 1000,01 Euro – 2000,00 Euro;
 - **EC₃**: $\geq 2000,01$ Euro.
- Condiții 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. Exemplu 2. Proiectarea cazurilor de testare

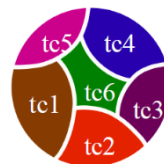
- ECs valide identificate:
 - **EC₁**: 0,00 Euro – 1000,00 Euro;
 - **EC₂**: 1000,01 Euro – 2000,00 Euro;
 - **EC₃**: $\geq 2000,01$ Euro.
- similar, se proiectează cazuri de testare valide și non-valide pentru limitele inferioare și superioare ale EC₂ și EC₃;
- Cazuri de testare proiectate pe baza condițiilor BVA identificate:
 - Limita inferioară a EC₁:
 - 1. amount = -0,01; **TC₀₁**: amount = -0,01; (non-valid)
 - 2. amount = 0,00; **TC₀₂**: amount = 0,00 (valid)
 - 3. amount = 0,01; **TC₀₃**: amount = 0,01; (valid)
 - Limita superioară a EC₁:
 - 4. amount = 999,99; **TC₀₄**: amount = 999,99; (valid)
 - 5. amount = 1000,00; **TC₀₅**: amount = 1000,00; (valid)
 - 6. amount = 1000,0; **TC₀₆**: amount = 1000,01; (non valid)

BVA. Acoperirea testării condițiilor BVA

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

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

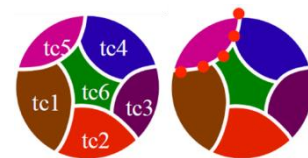
- E.g.:
 - pe baza specificațiilor au fost identificate 64 BVAs (pentru datele de intrare și ieșire, corespunzător ECs valide);
 - 48 BVAs au fost testate prin cazurile de testare proiectate;
 - **Acoperirea BVAs = $(48/64) \times 100 = 75\%$.**
- **Acoperirea BVAs** poate fi folosită ca și criteriu de terminare a testării, i.e., **exit criteria**.



ECP vs BVA

ECP

- **presupune că programul tratează similar toate valorile din aceeași 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 condiții de intrare/ieșire valide și non-valide;
- *obiectiv al testării* = verificarea respectării specificațiilor pentru valori uzuale, i.e., **building confidence in software;**



BVA

- **valorile identificate de condițiile BVA sunt prelucrate individual, nu în grup;**
- valorile se găsesc la limitele dintre ECs, acolo unde programul își schimbă comportamentul;
- se iau în considerare valori egale cu limita, valori imediat inferioare și valori imediat superioare limitei;
- sunt luate în considerare atât datele de intrare cât și cele de ieșire, corespunzătoare fiecărei EC valide;
- *obiectiv al testării* = căutarea bug-urilor uzuale, i.e., **bug hunting;**

ECP + BVA

- ECP și BVA dau cele mai bune rezultate atunci când sunt aplicate împreună!
- ECP și 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`) și cantitatea precizată (`quantity`), dacă valorile date sunt valide;
- Constrângeri:
 - `code`: [99, 999];
 - `quantity`: [1, 100];
- **Etape de realizare ECP + BVA:**
 1. identificarea ECs valide și non-valide;
 2. identificarea condițiilor BVA pentru ECs valide existente;
 3. proiectarea TCs pentru condițiile BVA identificate;
 4. acoperirea individuală a condițiilor BVA non-valide;
 5. minimizarea mulțimii de TCs pentru testarea ECP + BVA.

ECP + BVA. ECs valide și non-valide

- **1. Identificarea ECs valide și 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. Condiții BVA

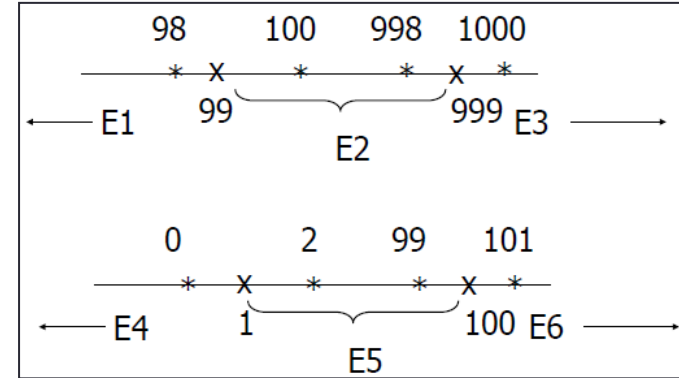
- 2. identificarea condițiilor BVA pentru ECs valide existente;

- EC valide:

- E_2 : code: [99, 999];
- E_5 : quantity: [1, 100];

- Condiții BVA pentru E_2 și E_5 ;

- Condiții pentru limita inferioară a lui E_2 :
 - code = 98;
 - code = 99;
 - code = 100;
- Condiții pentru limita superioară a lui E_2 :
 - code = 998;
 - code = 999;
 - code = 1000;



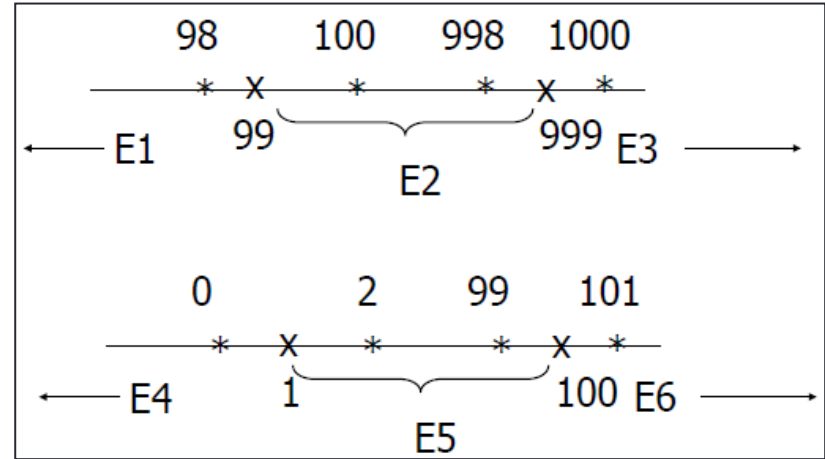
- x – limită a unei EC; * – valoare învecinată limitei;
- Condiții pentru limita inferioară a lui E_5 :
 - quantity = 0;
 - quantity = 1;
 - quantity = 2;
- Condiții pentru limita superioară a lui E_5 :
 - quantity = 99;
 - quantity = 100;
 - quantity = 101;

ECP + BVA. TCs pentru condițiile BVA

- 3. proiectarea TCs pentru condițiile BVA identificate;

- Cazuri de testare identificate, astfel încât toate condițiile BVA să fie testate cel puțin o dată:

- **TC₁: (code = 98, quantity = 0);**
- **TC₂: (code = 99, quantity = 1);**
- **TC₃: (code = 100, quantity = 2);**
- **TC₄: (code = 998, quantity = 99);**
- **TC₅: (code = 999, quantity = 100);**
- **TC₆: (code = 1000, quantity = 101).**

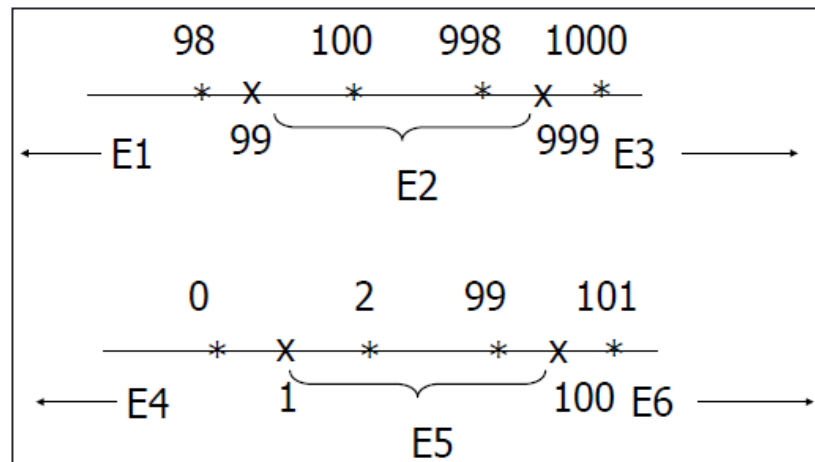


ECP + BVA. Acoperirea condițiilor BVA non-valide

- 4. acoperirea individuală a condițiilor BVA non-valide;

- Cazurile de testare **TC₁** și **TC₆** nu acoperă *individual* condițiile BVA non-valide;

- TC₁**: (code = 98, quantity = 0);
- TC₂**: (code = 99, quantity = 1);
- TC₃**: (code = 100, quantity = 2);
- TC₄**: (code = 998, quantity = 99);
- TC₅**: (code = 999, quantity = 100);
- TC₆**: (code = 1000, quantity = 101).



- TC₁ și TC₆ se pot înlocui cu:
 - TC₇**: (code = 98, quantity = 10);
 - TC₈**: (code = 1000, quantity = 10);
 - TC₉**: (code = 40, quantity = 0);
 - TC₁₀**: (code = 40, quantity = 101).

ECP + BVA. Minimizarea mulțimii de TCs

- 5. minimizarea mulțimii de TCs pentru testarea ECP + BVA;

- ~~TC₁: (code = 98, quantity = 0);~~
- TC₂: (code = 99, quantity = 1);
- TC₃: (code = 100, quantity = 2);
- TC₄: (code = 998, quantity = 99);
- TC₅: (code = 999, quantity = 100);
- ~~TC₆: (code = 1000, quantity = 101);~~
- TC₇: (code = 98, quantity = 10);
- TC₈: (code = 1000, quantity = 10);
- TC₉: (code = 40, quantity = 0);
- TC₁₀: (code = 40, quantity = 101).

ECP + BVA. Lista finală cu TCs

- Pentru metoda

```
computeTotalAmount (int code,  
                    int quantity): int
```
- care calculează valoarea totală a unei comenzi pentru produsul specificat și cantitatea dată;
- Constrângeri:
 - code : [99, 999];
 - quantity: [1, 100];
- **Lista de TCs finală bazată pe ECP și BVA este:**
 - **TC₂:** (code = 99, quantity = 1);
 - **TC₃:** (code = 100, quantity = 2);
 - **TC₄:** (code = 998, quantity = 99);
 - **TC₅:** (code = 999, quantity = 100);
 - **TC₇:** (code = 98, quantity = 10);
 - **TC₈:** (code = 1000, quantity = 10);
 - **TC₉:** (code = 40, quantity = 0);
 - **TC₁₀:** (code = 40, quantity = 101).
- **Observație:**
 - Lista de TCs nu este unică, dar poate fi considerată minimală pe baza ECP și BVA.

Testarea Black-box

Avantaje

- nu se există informații despre implementare;
- activitatea testerului este independentă de cea a programatorului;
- reflecta punctul de vedere al utilizatorului;
- surprinde ambiguitățile sau inconsistențele din specificații;
- începe imediat după finalizarea specificațiilor.

Dezavantaje

- dacă specificația *nu* este clară ==> dificultate de construire a cazurilor de testare;
- la execuția programului, multe drumurile din graful de execuție asociat codului rămân netestate ==> secvențele de cod sursă corespunzătoare pot conține 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 definiții ale testării (3); definiții ale testării (4);
 - terminologie: program, program testat, caz de testare;
 - tipuri de testare: exhaustivă, selectivă;
 - reguli de raportare a unui bug;
 - ciclul de viață al unui bug (diagrame, descriere);
- **testare black-box:**
 - definiție, caracteristici;
 - ECP, BVA, ECP vs. BVA, ECP + BVA;
 - aplicarea ECP și BVA pentru probleme concrete;
 - avantaje și dezavantaje.

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 instrucțiunilor, deciziilor, condițiilor, deciziilor și condițiilor, condițiilor multiple, drumurilor, buclilor
- **Testare White-box vs Testare Black-box**

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>.