Contents

0. Enunţ 2
1. Specificarea problemei 2
1.1. Analiza cerințelor 2
1.2. Specificații 2
2. Aplicarea tehnicilor de testare Black-Box 3
2.1. Equivalence Class Partitioning (ECP) 3
2.1.1. Detalii ECP 3
2.1.2. Aplicarea ECP 3
0. Identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații 3
1. Identificarea claselor de echivalenţă (equivalence classes, ECs) 4
2. Proiectarea cazurilor de testare (test cases, TCs) corespunzătoare ECs identificate 5
2.1.3. Acoperirea testării folosind ECP 5
2.1.4. Avantaje și dezavantaje ale utilizării ECP 6
2.2. Boundary Value Analysis (BVA) 6
2.2.1. Detalii BVA 6
2.2.2. Aplicarea BVA 7
0. Identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații 7
1. Identificarea claselor de echivalenţă (equivalence classes, ECs) 7
2. Identificarea condițiilor BVA valide și non-valide 7
3. Proiectarea TCs pentru condițiile BVA identificate 9
2.2.3. Acoperirea testării folosind BVA 10
2.2.4. Avantaje și dezavantaje ale utilizării BVA 10

0. Enunţ

Verifică dacă un număr natural n este prim.

1. Specificarea problemei

1.1. Analiza cerințelor.

Număr prim = număr care are ca divizori doar pe 1 şi pe el însuşi.

Exemple de numere prime:

- 0 și 1 nu sunt numere prime;
- 2 și 3 sunt numere prime;
- 2 este sigurul număr prim care este număr par;
- 4 nu este număr prim, deoarece are divizorii 1, 2 și 4.
- 7 este număr prim, deoarece are divizorii 1 și 7.

Idee de lucru: putem căuta posibili divizori începând cu 2 până la:

- n
- $n-1$
- $\frac{n}{2}$
- $\frac{n}{2}-1$
- $\sqrt{n}$

1.2. Specificații

Date de intrare și precondiții:

$X = (n)$

$\varphi(X)$: $(n \in \mathbb{N})$

Date de ieșire și postcondiții:

$Z = (r)$

$\psi(X, Z)$: $(r \in \{0,1\})$,

$$[(r = 1), (\nexists\, i \in \mathbb{N}, i = \overline{2, \sqrt{n}},\ n\ mod\ i\ \neq 0)] \vee$$
$$[(r = 0), (\exists\, i \in \mathbb{N}, i \in \{2, \dots, \sqrt{n}\},\ n\ mod\ i = 0)]$$

Alte variante de modelare a rezultatelor:

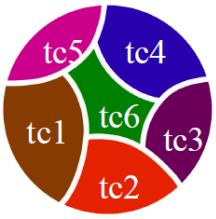
- $r \in \{0,1\}$, dar cu semnificație diferită, 0 – prim, 1 – neprim;
- $r \in \{true, false\}$, cu semnificații corespunzătoare;
- $r \in \{"prim", "neprim"\}$, cu semnificații corespunzătoare;
- etc.

2. Aplicarea tehnicilor de testare Black-Box

Tehnici de testare Black-Box:

- ECP (Equivalence Class Partitioning) sau ECA (Equivalence Class Analysis);
- BVA (Boundary value Analysis) sau BT (Boundary Testing);
- Best Representative (BR);
- Domain Testing (DT).

2.1. Equivalence Class Partitioning (ECP)



2.1.1. Detalii ECP

- tehnică de testare adecvată pentru verificarea respectării conformității cu specificațiile a ECs;
- aplicabilitate: unit testing, system testing, etc.

EC (Equivalence Class) = mulțime de valori pentru care programul are un comportament similar

Clasificare ECs:

– *valide* – asociate domeniilor de valori valide;

– *non-valide*: asociate domeniilor de valori non-valide.

Obiectivul testării:

- evaluarea comportamentului programului pentru *valori uzuale*, i.e., **building confidence in software**.

Etape de aplicare:

0. identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații;
1. identificarea ECs valide/non-valide pentru datele de intrare/ieşire;
2. proiectarea TCs pentru ECs identificate;

2.1.2. Aplicarea ECP

0. Identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații

Ca regulă generală, analizăm precondițiile și postcondițiile din specificație și extragem toate condițiile relevante. Aceste condiții trebuie să includă constrângeri referitoare la domeniile de valori asociate datelor de intrare/ieșire.

Pentru problema exemplificată putem identifica 2 condiții, (n∈N) și (r∈{0,1}).

Date de intrare și precondiții:

$X = (n)$

$\varphi(X)$: $(n \in \mathbb{N})$

Date de ieșire și postcondiții:

$Z = (r)$

$\psi(X, Z)$: $(r \in \{0,1\})$,

$$[(r = 1), (\nexists\, i \in \mathbb{N}, i = \overline{2, \sqrt{n}},\ n\ mod\ i\ \neq 0)] \vee$$
$$[(r = 0), (\exists\, i \in \mathbb{N}, i \in \{2, \dots, \sqrt{n}\},\ n\ mod\ i = 0)]$$

1. Identificarea claselor de echivalenţă (equivalence classes, ECs)

Folosind condițiile obținute anterior, putem identifica clase de echivalență (ECs). Există reguli pentru identificarea ECs, în funcție de modul în care putem reprezenta/descrie condițiile sau domeniile de valori ale variabilelor investigate.

ECP. Identificarea ECs. 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 finită 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. Identificarea ECs. 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. Identificarea ECs. 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.

Exemple: Care dintre regulile de mai sus se aplică pentru următoarele domenii de valori:

- Domeniu = {Dl, Dna, Dra} ==> 3 ECs valide și 1 EC non-validă;
- Domeniu = {1, …, 12 } ==> 12 ECs valide și 1 EC non-validă;
- Domeniu = {true, false} ==> 2 ECs valide și 1 EC non-validă;

Pentru problema exemplificată putem construi următorul tabel:

Nr. EC	Condiție/ Domeniu de valori	EC validă (mulțime de valori)	EC non-validă (mulțime de valori)
1.	$n \in \mathbb{N}$	$n \in \mathbb{N}$	---
2.		---	$n \notin \mathbb{N}$
3.	$r \in \{0,1\}$	$r \in \{0\}$	---
4.		$r \in \{1\}$	---
5.		---	$r \notin \{0,1\}$

2. Proiectarea cazurilor de testare (test cases, TCs) corespunzătoare ECs identificate

Corespunzător ECs identificate anterior se proiectează TCs astfel încât următoarele reguli să fie îndeplinite:

- un nou caz de testare corespunde la cât mai multe clase de echivalență valide încă neacoperite (pentru care nu s-au proiectat anterior TCs);
- un nou caz de testare corespunde doar uneia dintre clasele de echivalență de non-valide de intrare/ieșire încă neacoperite.

Pentru problema exemplificată avem tabelul următor:

Nr. TC	EC input	EC output	Input n	Expected Output r	Actual Output	Observații
TC01	1	4	11	1=prim		executabil
TC02	1	3	9	0=neprim		executabil
TC03	2	5	-1	mesaj de eroare		executabil *)
TC04	2	5	“abc”	mesaj de eroare		neexecutabil **)

*) Acest TC poate fi proiectat, dar *datele de intrare din n nu sunt valide. Deși testul a fost implementat și compilat, la execuție ar trebui să se afișeze un mesaj de eroare (TC03).*

***)* Acest TC poate fi proiectat, dar d*atele de intrare din n nu sunt valide. Testul nu a fi compilat și nu va putea fi executat (TC04).*

2.1.3. Acoperirea testării folosind ECP

Pentru calculul acoperii testării se folosește formula de mai jos.

Pentru problema exemplificată, să se calculeze următoarele:

- Acoperirea realizată prin proiectarea de TCs pentru ECs valide:

$$Coverage\ ECs\ valide = \frac{3}{5} \times 100 = 60\%$$

Avem ECs valide 1, 3 și 4.

- Acoperirea realizată prin proiectarea de TCs pentru ECs non-valide:

$$Coverage\ ECs\ non\_valide = \frac{2}{5} \times 100 = 40\%$$

Avem ECs non-valide 2 și 5.

2.1.4. Avantaje și dezavantaje ale utilizării ECP

Avantaje ECP:

- reduce mult numărul de TCs proiectate deoarece asociază fiecărei EC un singur caz de testare; se elimină redundanțele de TCs.
- se alege **o singură valoare** din fiecare EC, considerată **reprezentativă** pentru a acoperi testarea acelei EC.

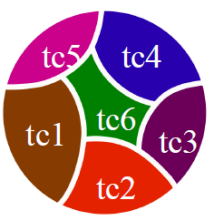
Dezvantaje ECP:

- Nu este eficientă la limita dintre ECs.

2.2. Boundary Value Analysis (BVA)

2.2.1. Detalii BVA

- tehnica de testare adecvată pentru analiza *limitelor* ECs;
- aplicabilitate: unit testing, system testing, etc.



limita = "locul" în care programul își schimbă comportamentul

analiza limitei = testarea valorii limită propriu-zise și a valorilor aflate în jurul ei

condiții BVA:

- *valide* – asociate limitelor valide;
- *non-valide*: asociate limitelor non-valide.

valorile identificate de condiţiile BVA sunt prelucrate individual, nu în grup;

Obiectivul testării:

- identificarea bug-urilor uzuale, i.e., **bug hunting**.

Etape:

0. identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații;
1. identificarea claselor de echivalenţă (equivalence classes, ECs);
2. identificarea condițiilor BVA valide și non-valide;
3. proiectarea TCs pentru condițiile BVA identificate;

2.2.2. Aplicarea BVA

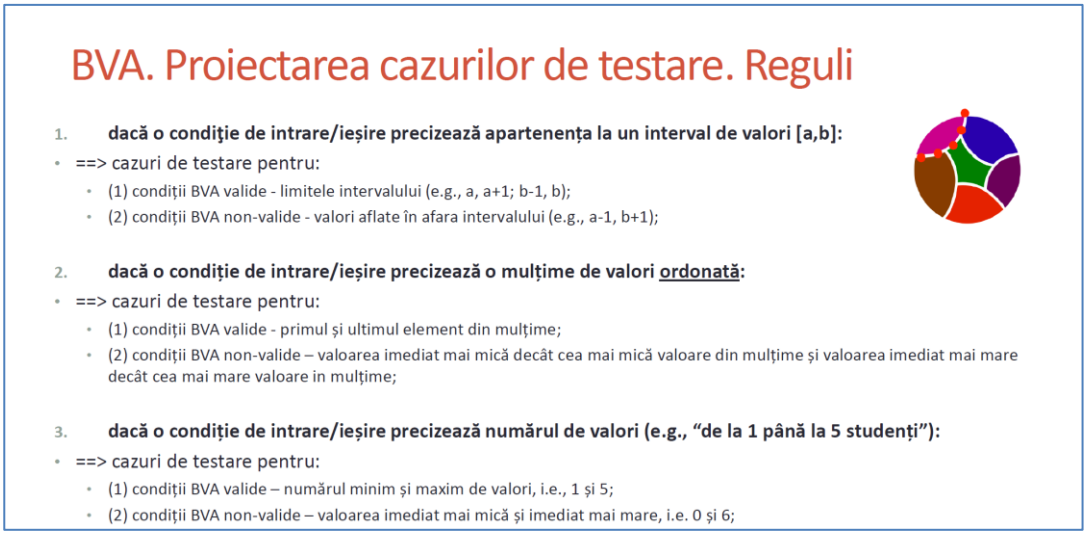
0. Identificarea condițiilor impuse asupra datelor de intrare/ieșire din specificații

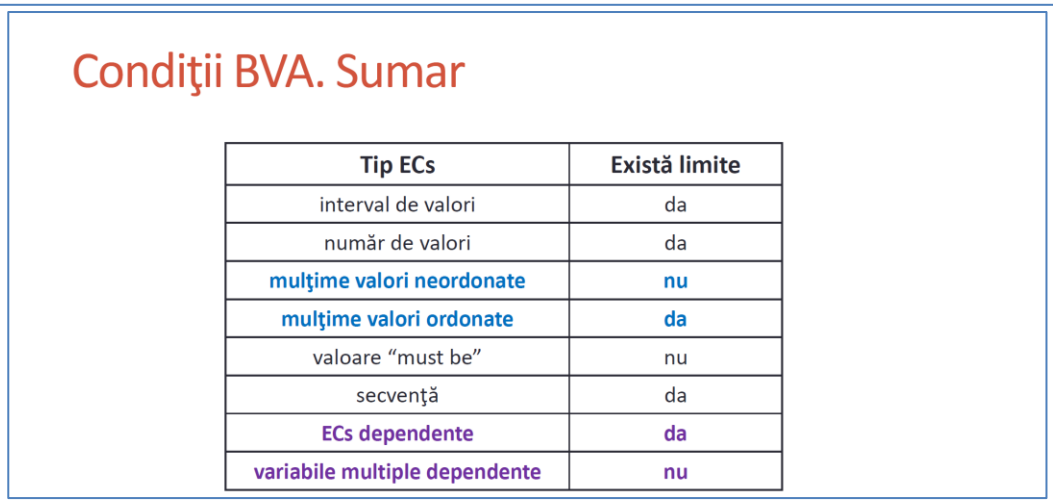
1. Identificarea claselor de echivalenţă (equivalence classes, ECs)

Cele două etape se realizează în manieră similară cu aplicarea tehnicii ECP. În cazul în care ECP și BVA se aplică secvențial, se trece la realizarea următoarei etape.

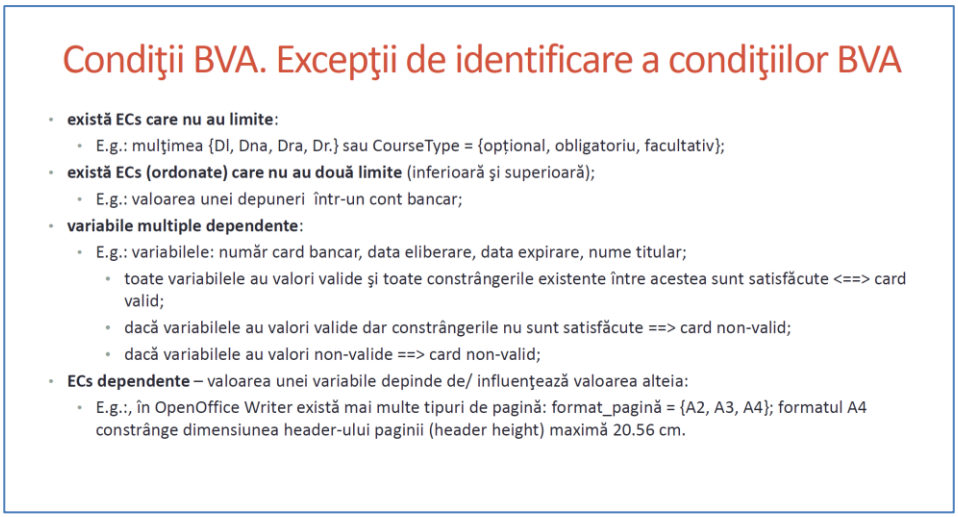
2. Identificarea condițiilor BVA valide și non-valide

Folosind ECs obținute anterior, se identifică condiţiile BVA pentru fiecare limită a fiecărei EC identificate. Există reguli pentru identificarea condițiilor BVA, în funcție de modul în care putem reprezenta/descrie o EC corespunzătoare variabilelor testate.





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



Exemple: Care dintre regulile de mai sus se aplică pentru următoarele domenii de valori:

- Domeniu = {1, …, 12 } ==> avem 2 limite finite==> condițiile BVA: 0, 1, 12, 13;
- Domeniu = {Dl, Dna, Dra} ==> 3 ECs valide care nu au limite ==> nu avem condiții BVA;
- Condiție aferentă unei EC valide = size < capacitate ==>
  - $s \in (-\infty;\ c]$ ==> avem doar 1 limită finită ==> condițiile BVA: c-1, c, c+1;
- Domeniu CourseType = {opțional, obligatoriu, facultativ}
  - ==> 3 ECs valide care nu au limite ==> nu avem condiții BVA;
- Card bancar cu informațiile (id card, name, expiry date, code)
  - variabile multiple dependente ==> nu avem condiții BVA;
- Format pagină A4 ==> height=29.7, width=21
  - ECs dependente ==> avem condiții BVA diferite, în funcție de ECs;
- Domeniu = (a, b)
  - ==> 1 EC validă ==> avem 2 limite finite==> condițiile BVA: a, a+1, b-1, b;
- Domeniu = [a, b]
  - ==> 1 EC validă ==> avem 2 limite finite==> condițiile BVA: a-1, a, a+1, b-1, b, b+1;

Pentru problema exemplificată (Număr prim) putem construi următorul tabel:

Nr. Condiție BVA	EC	Conditie BVA	Nr. TC
1.	$n \in \mathbb{N}$ $n \in [0, +\infty) \cap \mathbb{N}$ $n \in [0, MAXINT)$	1. n=-1	TC2
		2. n=0	TC4
		3. n=+1	TC7
		4. n=MAXINT-1	inclus in TC1
		5. n=MAXINT	…
		6. n=MAXINT+1	…
2.	$r \in \{0,1\}$ *putem stabili relație de ordine între valori ==> avem condiții BVA*	7. r=-1	
		8. r=0	
		9. r=1	
		10. r=2	

Observație:

Pentru modelarea $r \in \{true, false\}$ sau $r \in \{"prim", "neprim"\}$ avem mulțimi finite și nerodonate, nu avem relație de ordine între elemente în cadrul domeniilor de valori ==> nu există limite ==> tehnica BVA nu se poate aplica. Pentru aceste cazuri se aplică alte tehnici de testare Black-Box. De exemplu, tehnica **Best Representative**, care folosește ca și criteriu de selecție a valorilor de testare posibilitatea ca una sau mai multe valori din domeniu să aibă relevanță pentru identificarea de bug-uri sau cazuri de utilizare.

3. Proiectarea TCs pentru condițiile BVA identificate

Corespunzător condițților BVA identificate anterior se proiectează TCs astfel încât următoarele reguli să fie îndeplinite:

- un caz de testare nou, care corespunde la cât mai multe condiţii BVA valide încă neacoperite (pentru care nu s-au proiectat anterior TCs);
- un caz de testare nou, care corespunde doar uneia dintre condiţiile BVA non-valide încă neacoperite.

Pentru problema exemplificată avem tabelul următor:

Nr. TC	Condiție BVA	Input n	Expected Output r	Actual Output	Observații
TC03	1. n = -1				
TC05	2. n = 0	0	0=neprim		executabil
TC06	3. n = +1	1	0=neprim		executabil
TC07	4. n = MAXINT-1	MAXINT-1	0 sau 1		executabil *)
TC08	5. n = MAXINT	MAXINT	0 sau 1		executabil *)
TC09	6. n = MAXINT+1	MAXINT+1			neexecutabil **)
TC10	7. r = -1	?	-1		neexecutabil ***)
TC01, TC05	8. r = 0				
TC02	9. r = 1				
TC11	10. r = 2	?	2		neexecutabil ***)

) Acest TC poate fi proiectat, iar valoarea lui n este validă. Programatorul va verifica dacă valorile MAXINT-1 și MAXINT sunt prime sau nu. Corespunzător va completa în tabel pentru coloana **Expected Output cu 0 sau 1. Testele se pot implementa, compila și executa. La execuție ar trebui să se afișeze rezultatul 0 sau 1 (TC07, TC08).*

***) Acest TC poate fi proiectat, dar valoarea lui n nu este validă, depășește domeniul de valori asociat tipului de date folosit pentru variabila n. Testul nu va fi compilat și nu va putea fi executat (TC09).*

****) Acest TC poate fi proiectat, dar valoarea lui n nu este validă, conform modelării. Nu avem date de intrare pentru variabila n pentru care să putem obține rezultate non-valide, ca -1 sau 2. Testul nu se poate implementa și nici executa (TC10, TC11).*

2.2.3. Acoperirea testării folosind BVA

Pentru calculul acoperii testării se folosește formula de mai jos.

BVA. Acoperirea testării condiţiilor BVA

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

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

Pentru problema exemplificată, să se calculeze următoarele:

- Acoperirea realizată prin proiectarea de TCs pentru condiții BVA valide:

$$Coverage\ cond\ BVA\ valide = \frac{6}{10} \times 100 = 60\%$$

  Avem condițiile BVA valide 2, 3, 4, 5, 8 și 9.

- Acoperirea realizată prin proiectarea de TCs pentru condiții BVA non-valide:

$$Coverage\ Cond\ BVA\ non\_valide = \frac{4}{10} \times 100 = 40\%$$

  Avem condițiile BVA non-valide 1, 6, 7 și 10.

2.2.4. Avantaje și dezavantaje ale utilizării BVA

Avantaje BVA:

- Valorile testate se găsesc la limita dintre ECs.

Dezvantaje BVA:

- Se testează mai multe valori, nu doar limita.
- Volumul de teste este mai mare,d epinde de numărul de limite finite ale Ecs identificate.