Testarea statică și dinamică

Testarea statică și dinamică nu este doar un act de cercetare științifică, ci și un material util de lucru care permite acumularea experienței în domeniul testării software, începând cu etapa de analiză a cerințelor. În plus, având în vedere numărul impunător de produse software diferite, specialiștii din domeniul testării contribuie la dezvoltarea și utilizarea unor mini-tehnici utile pentru identificarea erorilor critice specifice produsului elaborat, doar după obținerea experienței necesare.

Testarea statică(verificare și procesare documentație)

Testarea statică nu implică execuția produsului software care urmează să fie testat. Este o metodă la fel de importantă ca testarea dinamică pentru descoperirea defectelor. Cu cât un defect este identificat mai devreme în ciclul de viață al produsului, cu atât costul de remediere este mai mic. Testarea statică include revizuirea documentelor și poate identifica următoarele tipuri de defecte:

- Deviere de la standarde
- Lipsa unor cerințe
- Defecte de design
- Cod care nu poate fi întreținut
- Specificații incomplete

Tehnici de testare statică:

1) Recenzii;
2) Ghid;
3) Analiză tehnică;
4) Inspecție.

Recenzii – De exemplu, verificarea unei lucrări. Se referă la procesul de evaluare a documentației sau a produselor pentru a asigura calitatea și corectitudinea acestora.

Ghid – Modul de elaborare a documentelor, acoperind întregul proces de la început până la sfârșit. Oferă instrucțiuni și bune practici pentru crearea și gestionarea documentației

Analiză tehnică – Analiza detaliată a produsului software, de obicei dintr-o perspectivă tehnică, pentru a evalua arhitectura, designul și implementarea acestuia.

Inspecție – Analiza realizată de cei mai importanți membri ai echipei, cum ar fi cei care au contribuit la concepția proiectului sau programatorii. Este un proces de evaluare a produsului sau documentelor pentru a identifica probleme și a asigura conformitatea cu cerințele.

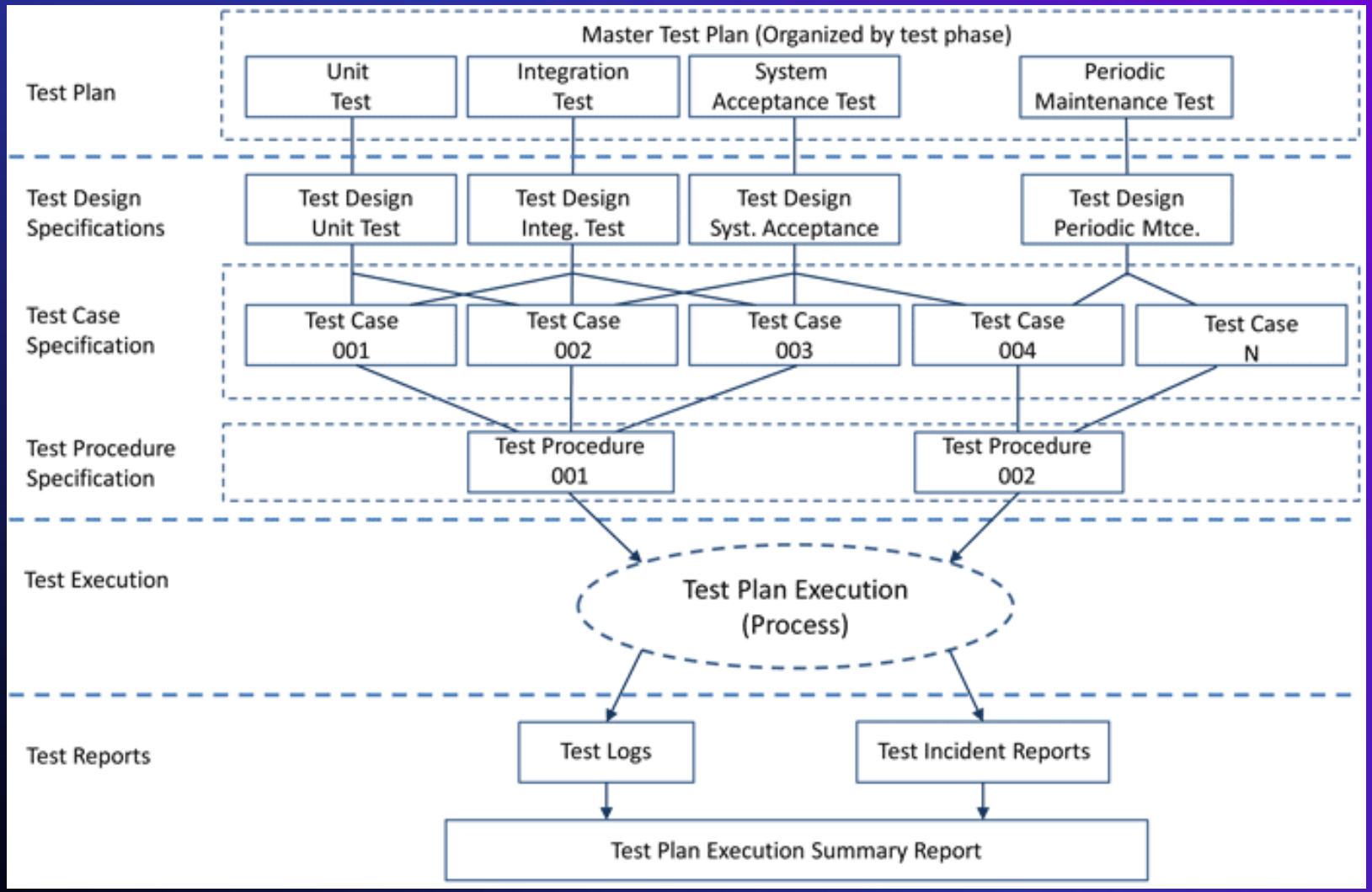
Tehnici de proiectare a testării dinamice

1. Tehnica bazata pe specificații/funcționalități (black-box)
2. Tehnica bazata pe structura (white-box)
3. Testarea exploratorie și experimentală

- Proiectarea testării presupune 3 etape:

1. Identificarea condițiilor de testare este un aspect esențial al procesului de testare, asigurând că toate aspectele relevante ale produsului sunt verificate.
2. Specificarea cazurilor de testare este un pas esențial în procesul de testare software, care implică detalierea a ceea ce trebuie să fie verificat pentru a asigura că produsul funcționează conform cerințelor.
3. Specificarea procedurilor de testare este un aspect crucial al planificării și execuției testării software, asigurându-se că testele sunt realizate într-o manieră consistentă și ordonată.

Reprezentare schematica:



Alcătuirea unui caz de testare:

- Valori permise : 20 litere, “-”. Datele sunt case-sensitive validate când se tastează Enter. Dacă datele sunt corecte se schimbă fereastra pentru introducerea unei cereri, dacă nu sunt valide datele, apare un mesaj de eroare pe pagina curenta.
- Cazuri valide:

Mr Ion Craciun

Ms Ana-Maria Gutu

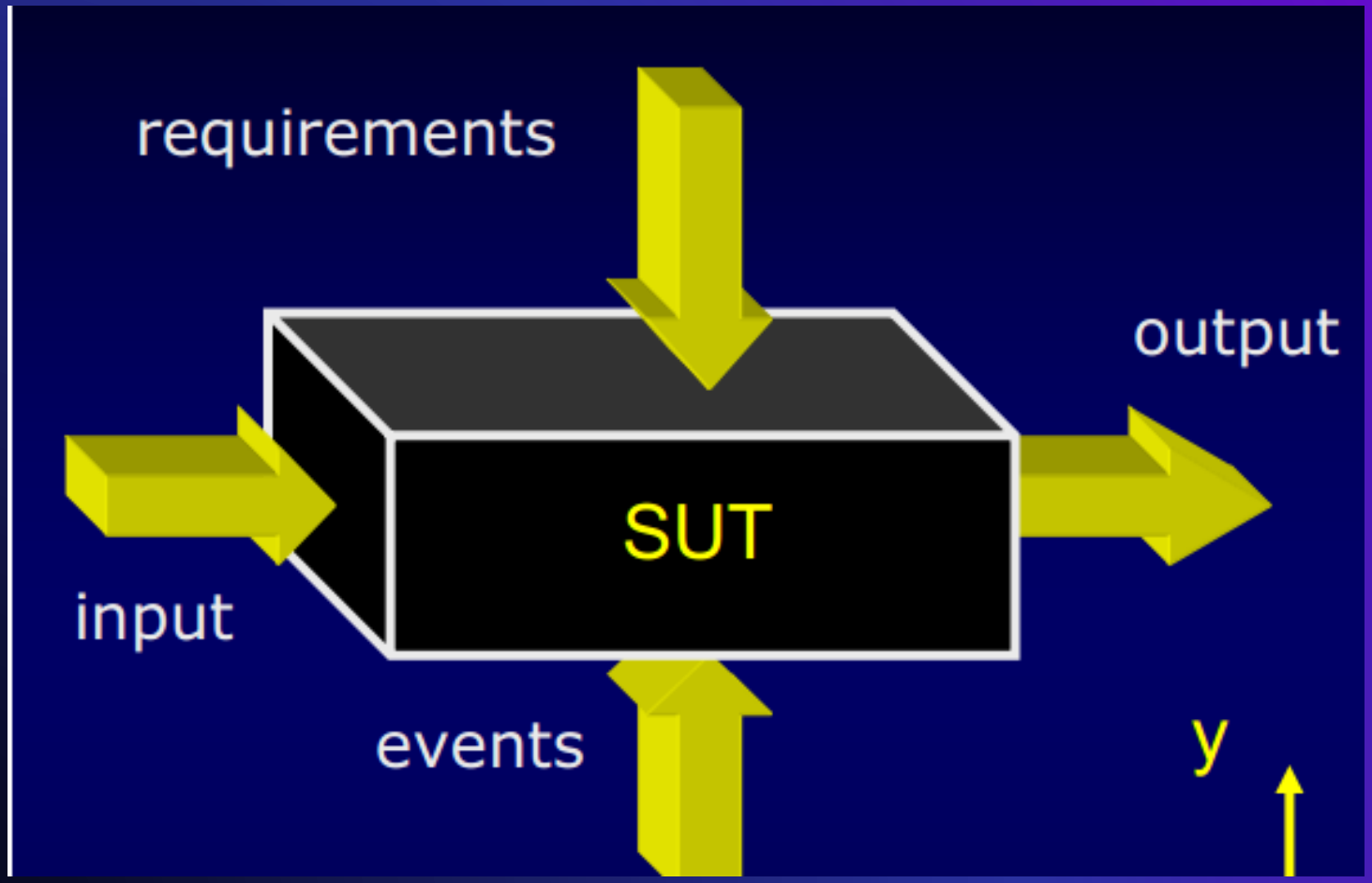
- Cazuri invalide:

Mr George1 Graf

Ms “Cort” Alina

Ms Popescu ‘Ana’

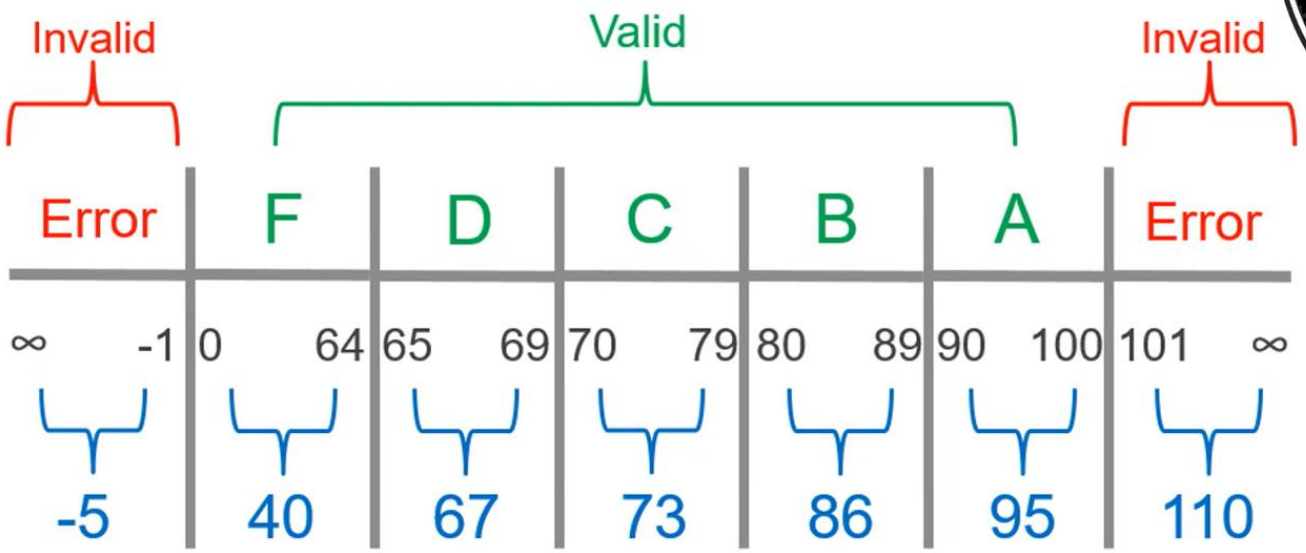
Tehnica bazata pe specificatii/functionalitati (black-box)



‘Mini- tehnici’ bazate pe specificații”

- Stabilirea claselor de valori – equivalence partitioning;
- Stabilirea limitelor claselor – boundary value analysis;
- Alcătuirea tabelelor de decizii – decision table testing;
- Alcătuirea stărilor de tranziție – state transition testing;
- Stabilirea cazurilor de testare – use case testing.

Stabilirea claselor de valori



Stabilirea limitelor claselor

A	B	C
1-4	5-7	8-10
2	6	9

0,1, 4,5, 7, 8,10, 11

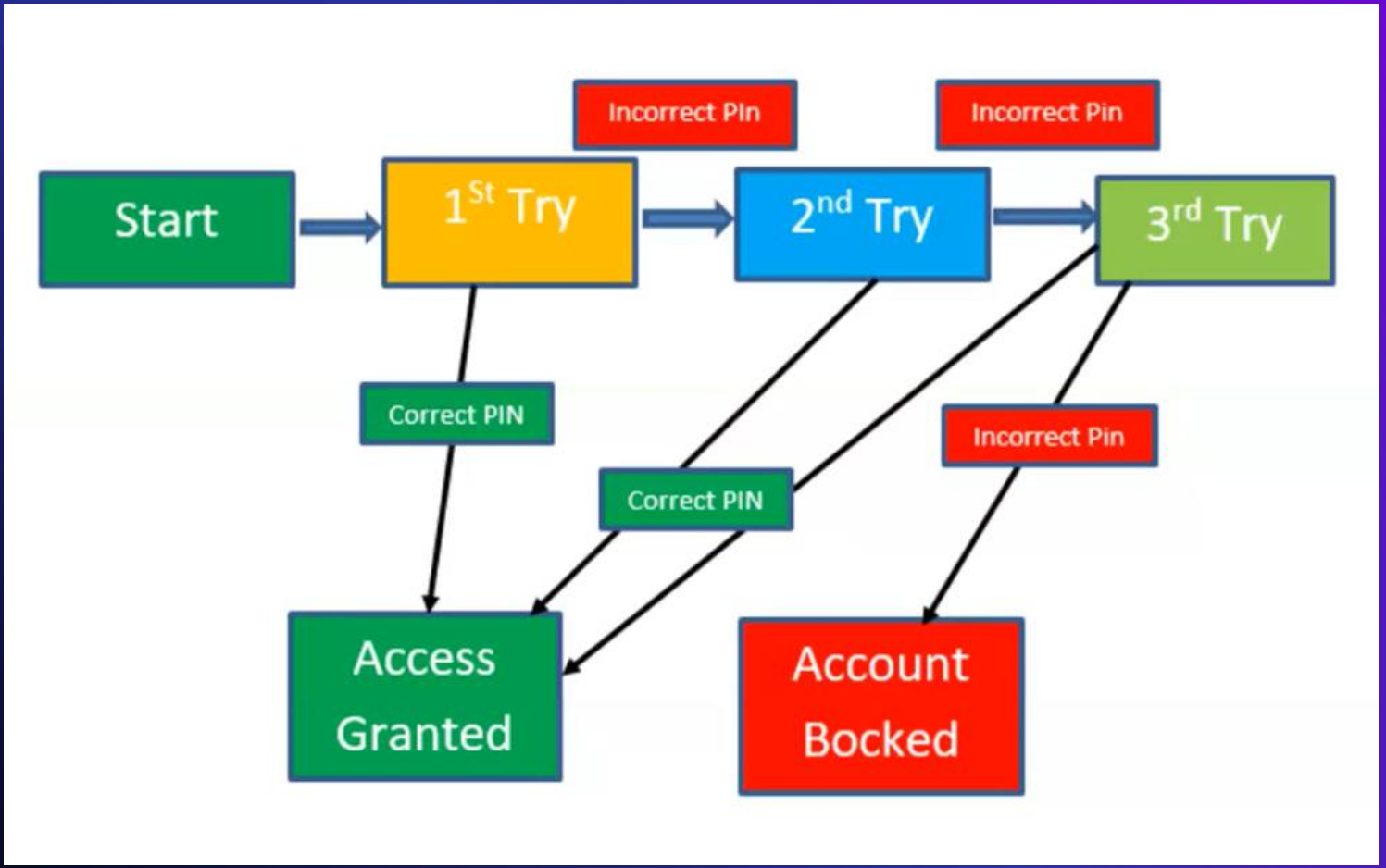
Alcătuirea tabelului de decizie

Condiții	Regula 1	Regula 2	Regula 3	Regula 4
Nume utilizator (T/F)	F	T	F	T
Parola (T/F)	F	F	T	T
Ieșire (E/H)	E	E	E	H

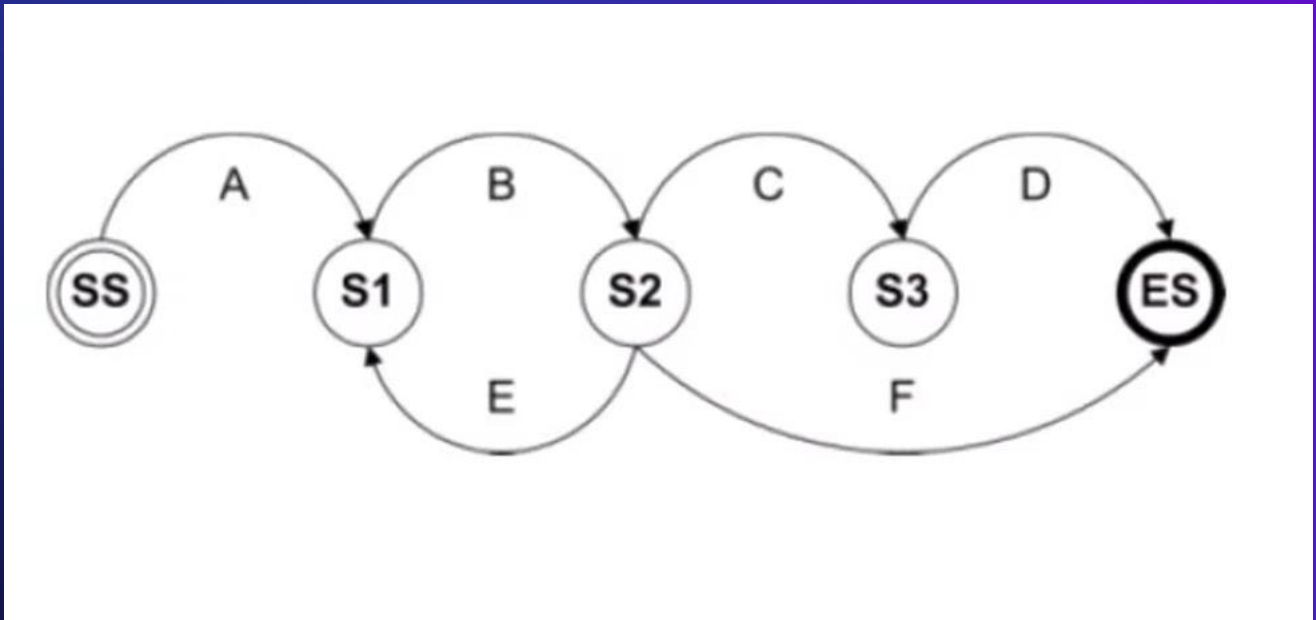
Legendă:

- T – Nume de utilizator/parolă corecte
- F – Nume de utilizator/parolă greșit
- E – Este afișat un mesaj de eroare
- H – Este afișat ecranul de pornire

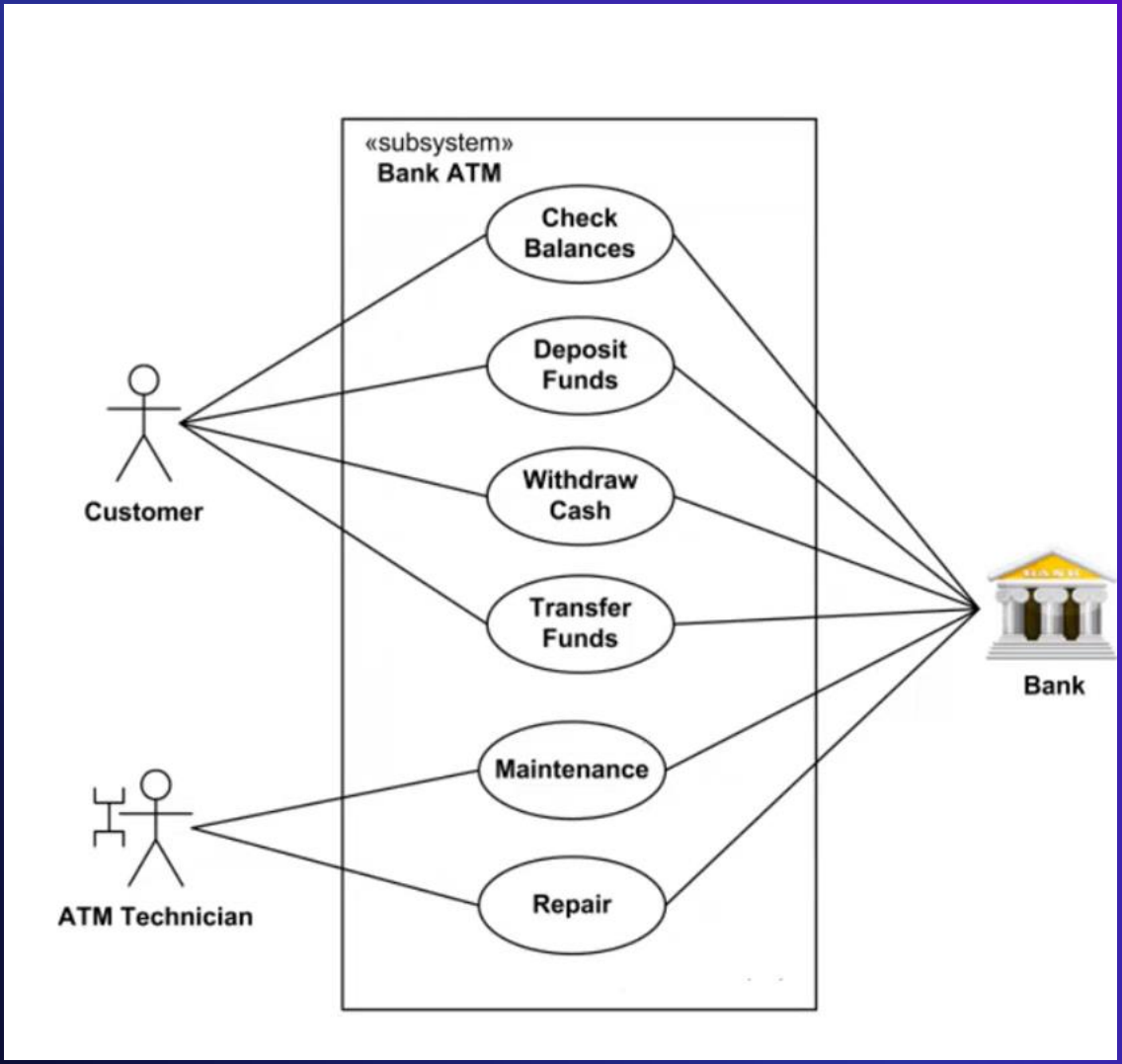
Alcătuirea stărilor de tranziție



Alcătuirea stărilor de tranziție



Stabilirea cazurilor de testare:



Tehnica bazată pe structura (white-box):

- Diagrame de flux - Flow charts
  - Secvențe
  - Selecție
  - Iterații
- Stabilirea căilor programului

Diagramele de flux:

- **Componentele:**
  - **Secvența -** o asfel de structură de control se compune din pași care se înlanțuie unul după altul, în ordinea în din algoritm.

```
Read(a);
Writeln(b);
C=a+b;
```

Diagramele de flux:

- **Componentele:**
  - **Selecția –** permite alegerea unui căi din mai multe posibile in funcție de condiția stabilită.

```
If  a>b then writeln(a)
            else writeln(b)
```

Diagramele de flux:

- **Componentele:**
  - **Iterația –** permite repetarea unei secvențe de mai multe ori.

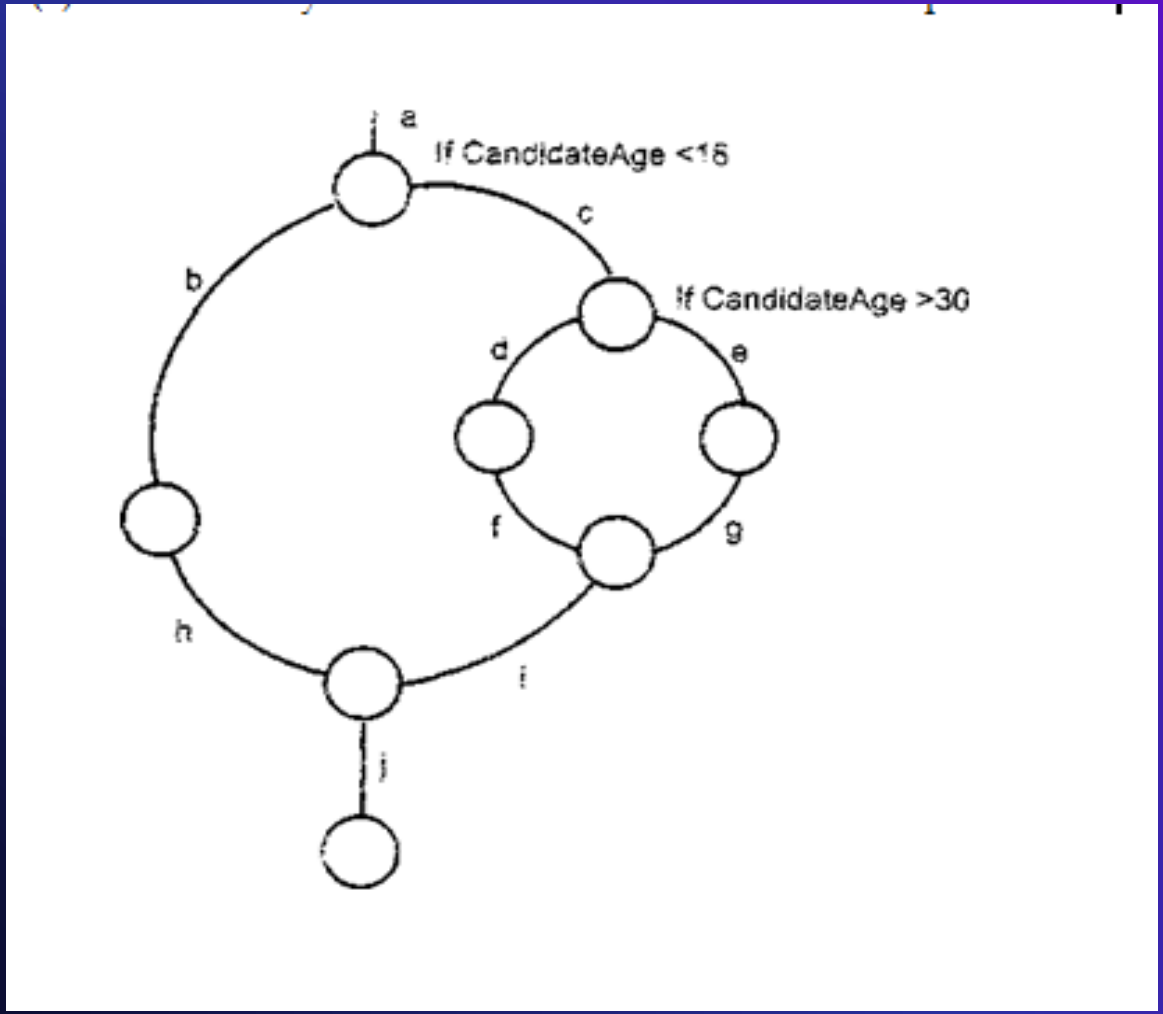
```
X=15
Count=0
While x<20 do
    x:=x+1
    count=count+1
End do
```

Exemplu:

Let us try an exampie without a ioop now.

```
1    Program Age Check
2
3    CandidateAge: Integer;
4
5    Begin
6
7    Read(CandidateAge!
8
9    If CandidateAge < 18
10   Then
11       Print ("Candidate is too    young")
12   Else
13       If CandidateAge > 30
14       Then
15            Print ("Candidate is  too old
16       Else
17            Print("Candidate may  join CI
13       Endif
19   Endif
20
21   End
```

Diagrama:



Testarea exploratorie și experimentală

Se remarcă două abordări de testare utilizate. Prima dintre ele se numește testare experimentală și presupune definirea unui caz de testare, efectuarea testului propriu zis și verificarea rezultatului. În cazul în care rezultatele nu sunt conforme cu ceea ce este specificat în cazurile de testare, la rubrica rezultatelor așteptate, tester-ul completează un raport prin care se indică defectul echipei de dezvoltare. Metoda este foarte utilă pentru a asigura funcționalitățile de bază negociate cu clientul.

Cea de a doua metoda de testare este testarea exploratorie. Aceasta reprezintă de cele mai multe ori rularea unor teste pe care le execută un tester fără a avea un plan stabilit. Nu este necesară elaborarea unui caz de testare, ci se urmărește tocmai testarea aplicației în situații speciale. Scopul acestei metode este acela de a îmbunătăți calitatea generală a produsului și de a învăța mai multe despre acesta. De asemenea, poate fi îmbinată și cu rezolvarea defectelor descoperite pe parcursul testelor, ceea că mărește eficiența etapei de lucru.

De cele mai multe ori, utilitatea testării exploratorii depinde de calitățile tester-ului care o efectuează. Un angajat bine pregătit va fi capabil să inventeze situații mai diverse și să observe erori mai greu de depistat.

Un dezavantaj al acestei metode este dat de faptul că, de cele mai multe ori, procedurile se execută manual și nu pot fi automatizate, așa cum se poate face în cadrul testării experimentale.